9 情技応第 888 号

高度情報化支援ソフトウェア育成事業

自己反映計算に基づく Java 言語用の
開放型 Just-in-Time コンパイラ OpenJIT の研究開発

構造仕様書
(2 / 2)

平成 11 年 1 月

富士通株式会社

目次

第 1 章　全体概要 **1**

第 2 章　設計指針 **19**

第 3 章　プログラム構成 **30**

3.1 全体構成 (図) 30

3.2 サブプログラム概要 (目的 , 機能) 32

3.2.1 OpenJIT パッケージ 32

3.2.2 OpenJIT.frontend.asm パッケージ 36

3.2.3 OpenJIT.frontend.discompiler パッケージ 38

3.2.4 OpenJIT.frontend.flowgraph パッケージ 44

3.2.5 OpenJIT.frontend.tree パッケージ 47

3.2.6 OpenJIT.frontend.util パッケージ 62

3.2.7 OpenJIT Java Native Code API パッケージ 64

3.3 機能ブロックとの対応 66

3.3.1 OpenJIT フロントエンドシステム 70

3.3.2 OpenJIT バックエンドシステム 97

3.4 サブプログラム間の関連 (インタフェース) 106

3.4.1 OpenJIT フロントエンドシステム 106

3.4.2 OpenJIT バックエンドシステム 120

3.5 全体処理方式 130

第 4 章　プログラム仕様 **131**

4.1 OpenJIT パッケージ 131

4.1.1 OpenJIT.Constants インタフェース 132

4.1.2 OpenJIT.LinkedList クラス 162
4.1.3 OpenJIT.BCinfo クラス 163
4.1.4 OpenJIT.ILnode クラス 165
4.1.5 OpenJIT.BBinfo クラス 180
4.1.6 OpenJIT.Compile クラス 181
4.1.7 OpenJIT.Select クラス 197
4.1.8 OpenJIT.ParseBytecode クラス 199
4.1.9 OpenJIT.ConvertRTL クラス 207
4.1.10 OpenJIT.OptimizeRTL クラス 211
4.1.11 OpenJIT.Var クラス 219
4.1.12 OpenJIT.RegAlloc クラス 221
4.1.13 OpenJIT.RegAlloc$VirReg クラス 234
4.1.14 OpenJIT.RegAlloc$PhyRegs クラス 238
4.1.15 OpenJIT.ExceptionHandler クラス 245
4.1.16 OpenJIT.CompilerError クラス 247
4.1.17 OpenJIT.Sparc クラス 249
4.1.18 OpenJIT.Runtime クラス 250
4.2 OpenJIT.frontend.asm パッケージ 251
4.2.1 OpenJIT.frontend.asm.Constants インターフェース 251
4.2.2 OpenJIT.frontend.asm.RuntimeConstants インターフェース 272
4.2.3 OpenJIT.frontend.asm.ArrayData クラス 299
4.2.4 OpenJIT.frontend.asm.Assembler クラス 300
4.2.5 OpenJIT.frontend.asm.Instruction クラス 304
4.2.6 OpenJIT.frontend.asm.Label クラス 307
4.3 OpenJIT.frontend.discompiler パッケージ 310
4.3.1 OpenJIT.frontend.discompiler.ASTFactory インターフェース 311
4.3.2 OpenJIT.frontend.discompiler.Annotation クラス 354
4.3.3 OpenJIT.frontend.discompiler.AnnotationAnalyzer クラス 355
4.3.4 OpenJIT.frontend.discompiler.BBABasicBlock クラス . . . 358
4.3.5 OpenJIT.frontend.discompiler.BasicBlock クラス 362
4.3.6 OpenJIT.frontend.discompiler.BasicBlockAnalyzer クラス . 366

4.3.7 OpenJIT.frontend.discompiler.BranchTableEntry クラス . 376
4.3.8 OpenJIT.frontend.discompiler.BytecodeParser クラス . . . 378
4.3.9 OpenJIT.frontend.discompiler.CFAConstants インターフェース 381
4.3.10 OpenJIT.frontend.discompiler.CFAFlag クラス 383
4.3.11 OpenJIT.frontend.discompiler.CFGNode クラス 387
4.3.12 OpenJIT.frontend.discompiler.Case クラス 389
4.3.13 OpenJIT.frontend.discompiler.ClassSignature クラス . . . 391
4.3.14 OpenJIT.frontend.discompiler.ControlFlowAnalyzer クラス 394
4.3.15 OpenJIT.frontend.discompiler.ControlFlowGraph クラス . 397
4.3.16 OpenJIT.frontend.discompiler.Discompiler クラス 402
4.3.17 OpenJIT.frontend.discompiler.DiscompilerError クラス . . 405
4.3.18 OpenJIT.frontend.discompiler.DTNode クラス 407
4.3.19 OpenJIT.frontend.discompiler.DTVisitor 抽象クラス . . . 414
4.3.20 OpenJIT.frontend.discompiler.DefaultASTFactory クラス . 417
4.3.21 OpenJIT.frontend.discompiler.DominatorTree クラス . . . 459
4.3.22 OpenJIT.frontend.discompiler.ExpressionAnalyzer クラス . 462
4.3.23 OpenJIT.frontend.discompiler.FlowEdge クラス 467
4.3.24 OpenJIT.frontend.discompiler.LoopHead クラス 469
4.3.25 OpenJIT.frontend.discompiler.Metaclass 抽象クラス 471
4.3.26 OpenJIT.frontend.discompiler.MethodInformation インターフェース 472
4.3.27 OpenJIT.frontend.discompiler.NameAndType クラス . . . 480
4.3.28 OpenJIT.frontend.discompiler.RuntimeMethodInformation クラス 487
4.3.29 OpenJIT.frontend.discompiler.Switch クラス 495
4.3.30 OpenJIT.frontend.discompiler.SymbolicConstants インターフェース 497
4.3.31 OpenJIT.frontend.discompiler.SymbolicStack クラス 500
4.3.32 OpenJIT.frontend.discompiler.SymbolicValue クラス . . . 505
4.3.33 OpenJIT.frontend.discompiler.VMInstruction クラス . . . 510
4.4 OpenJIT.frontend.flowgraph パッケージ 531

4.4.1 OpenJIT.frontend.flowgraph.FlowGraph クラス 531
4.4.2 OepnJIT.frontend.flowgraph.CHInfo クラス 536
4.4.3 OpenJIT.frontend.flowgraph.ControlDependencyGraph クラス 537
4.4.4 OpenJIT.frontend.flowgraph.DataFlowGraph クラス 539
4.4.5 OpenJIT.frontend.flowgraph.ClassHierarchyGraph クラス . 543
4.4.6 OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラス . . 545
4.4.7 OpenJIT.frontend.flowgraph.DFFunctionRegister クラス . 548
4.4.8 OpenJIT.frontend.flowgraph.FlowGraphAnalyzer インターフェース 550
4.4.9 OpenJIT.frontend.flowgraph.ReachingAnalyzer クラス . . . 551
4.4.10 OpenJIT.frontend.flowgraph.AvailableAnalyzer クラス . . 553
4.4.11 OpenJIT.frontend.flowgraph.LivenessAnalyzer クラス . . . 555
4.4.12 OpenJIT.frontend.flowgraph.FixedPointDetector クラス . . 557
4.4.13 OpenJIT.frontend.flowgraph.ClassHierarchyAnalyzer クラス 559
4.4.14 OpenJIT.frontend.flowgraph.ASTTransformer クラス . . . 561
4.4.15 OpenJIT.frontend.flowgraph.Optimizer 抽象クラス 564
4.5 OpenJIT.frontend.tree パッケージ 566
4.5.1 OpenJIT.frontend.tree.BinaryBitExpression 抽象クラス . . 570
4.5.2 OpenJIT.frontend.tree.BinaryLogicalExpression 抽象クラス 574
4.5.3 OpenJIT.frontend.tree.AddExpression クラス 577
4.5.4 OpenJIT.frontend.tree.AndExpression クラス 583
4.5.5 OpenJIT.frontend.tree.ArrayAccessExpression クラス . . . 586
4.5.6 OpenJIT.frontend.tree.ArrayExpression クラス 594
4.5.7 OpenJIT.frontend.tree.AssignAddExpression クラス 598
4.5.8 OpenJIT.frontend.tree.AssignBitAndExpression クラス . . 601
4.5.9 OpenJIT.frontend.tree.AssignBitOrExpression クラス . . . 603
4.5.10 OpenJIT.frontend.tree.AssignBitXorExpression クラス . . 605
4.5.11 OpenJIT.frontend.tree.AssignDivideExpression クラス . . . 607
4.5.12 OpenJIT.frontend.tree.AssignExpression クラス 609
4.5.13 OpenJIT.frontend.tree.AssignMultiplyExpression クラス . 612
4.5.14 OpenJIT.frontend.tree.AssignOpExpression クラス 614

4.5.15 OpenJIT.frontend.tree.AssignRemainderExpression クラス 619
4.5.16 OpenJIT.frontend.tree.AssignShiftLeftExpression クラス . 621
4.5.17 OpenJIT.frontend.tree.AssignShiftRightExpression クラス 623
4.5.18 OpenJIT.frontend.tree.AssignSubtractExpression クラス . 625
4.5.19 OpenJIT. frontend. tree. AssignUnsignedShiftRightExpression クラス 627
4.5.20 OpenJIT.frontend.tree.BinaryArithmeticExpression クラス 629
4.5.21 OpenJIT.frontend.tree.BinaryAssignExpression クラス . . 631
4.5.22 OpenJIT.frontend.tree.BinaryCompareExpression クラス . 634
4.5.23 OpenJIT.frontend.tree.BinaryEqualityExpression クラス . 636
4.5.24 OpenJIT.frontend.tree.BinaryExpression クラス 638
4.5.25 OpenJIT.frontend.tree.BinaryShiftExpression クラス . . . 646
4.5.26 OpenJIT.frontend.tree.BitAndExpression クラス 648
4.5.27 OpenJIT.frontend.tree.BitNotExpression クラス 652
4.5.28 OpenJIT.frontend.tree.BitOrExpression クラス 656
4.5.29 OpenJIT.frontend.tree.BitXorExpression クラス 660
4.5.30 OpenJIT.frontend.tree.BooleanExpression クラス 664
4.5.31 OpenJIT.frontend.tree.BreakStatement クラス 668
4.5.32 OpenJIT.frontend.tree.ByteExpression クラス 671
4.5.33 OpenJIT.frontend.tree.CaseStatement クラス 673
4.5.34 OpenJIT.frontend.tree.CastExpression クラス 675
4.5.35 OpenJIT.frontend.tree.CatchStatement クラス 678
4.5.36 OpenJIT.frontend.tree.CharExpression クラス 682
4.5.37 OpenJIT.frontend.tree.CheckContext クラス 684
4.5.38 OpenJIT.frontend.tree.CodeContext クラス 685
4.5.39 OpenJIT.frontend.tree.CommaExpression クラス 687
4.5.40 OpenJIT.frontend.tree.CompoundStatement クラス 691
4.5.41 OpenJIT.frontend.tree.ConditionVars クラス 695
4.5.42 OpenJIT.frontend.tree.ConditionalExpression クラス . . . 696
4.5.43 OpenJIT.frontend.tree.ConstantExpression クラス 703
4.5.44 OpenJIT.frontend.tree.Context クラス 705

4.5.45 OpenJIT.frontend.tree.ContinueStatement クラス 707
4.5.46 OpenJIT.frontend.tree.ConvertExpression クラス 710
4.5.47 OpenJIT.frontend.tree.DeclarationStatement クラス 715
4.5.48 OpenJIT.frontend.tree.DivRemExpression クラス 718
4.5.49 OpenJIT.frontend.tree.DivideExpression クラス 720
4.5.50 OpenJIT.frontend.tree.DoStatement クラス 724
4.5.51 OpenJIT.frontend.tree.DoubleExpression クラス 728
4.5.52 OpenJIT.frontend.tree.EqualExpression クラス 731
4.5.53 OpenJIT.frontend.tree.ExprExpression クラス 736
4.5.54 OpenJIT.frontend.tree.Expression クラス 739
4.5.55 OpenJIT.frontend.tree.ExpressionStatement クラス 757
4.5.56 OpenJIT.frontend.tree.FieldExpression クラス 761
4.5.57 OpenJIT.frontend.tree.FinallyStatement クラス 763
4.5.58 OpenJIT.frontend.tree.FloatExpression クラス 765
4.5.59 OpenJIT.frontend.tree.ForStatement クラス 768
4.5.60 OpenJIT.frontend.tree.GreaterExpression クラス 773
4.5.61 OpenJIT.frontend.tree.GreaterOrEqualExpression クラス . 777
4.5.62 OpenJIT.frontend.tree.IdentifierExpression クラス 781
4.5.63 OpenJIT.frontend.tree.IfStatement クラス 783
4.5.64 OpenJIT.frontend.tree.IncDecExpression クラス 788
4.5.65 OpenJIT.frontend.tree.InlineMethodExpression クラス . . 792
4.5.66 OpenJIT.frontend.tree.InlineNewInstanceExpression クラス 796
4.5.67 OpenJIT.frontend.tree.InlineReturnStatement クラス . . . 800
4.5.68 OpenJIT.frontend.tree.InstanceOfExpression クラス 804
4.5.69 OpenJIT.frontend.tree.IntExpression クラス 809
4.5.70 OpenJIT.frontend.tree.IntegerExpression クラス 811
4.5.71 OpenJIT.frontend.tree.LengthExpression クラス 815
4.5.72 OpenJIT.frontend.tree.LessExpression クラス 817
4.5.73 OpenJIT.frontend.tree.LessOrEqualExpression クラス . . . 821
4.5.74 OpenJIT.frontend.tree.LocalField クラス 825
4.5.75 OpenJIT.frontend.tree.LongExpression クラス 828

4.5.76 OpenJIT.frontend.tree.MethodExpression クラス 831
4.5.77 OpenJIT.frontend.tree.MultiplyExpression クラス 833
4.5.78 OpenJIT.frontend.tree.NaryExpression クラス 837
4.5.79 OpenJIT.frontend.tree.NegativeExpression クラス 840
4.5.80 OpenJIT.frontend.tree.NewArrayExpression クラス 844
4.5.81 OpenJIT.frontend.tree.NewInstanceExpression クラス . . . 845
4.5.82 OpenJIT.frontend.tree.Node クラス 847
4.5.83 OpenJIT.frontend.tree.NotEqualExpression クラス 850
4.5.84 OpenJIT.frontend.tree.NotExpression クラス 855
4.5.85 OpenJIT.frontend.tree.NullExpression クラス 859
4.5.86 OpenJIT.frontend.tree.OrExpression クラス 862
4.5.87 OpenJIT.frontend.tree.PositiveExpression クラス 866
4.5.88 OpenJIT.frontend.tree.PostDecExpression クラス 868
4.5.89 OpenJIT.frontend.tree.PostIncExpression クラス 870
4.5.90 OpenJIT.frontend.tree.PreDecExpression クラス 872
4.5.91 OpenJIT.frontend.tree.PreIncExpression クラス 874
4.5.92 OpenJIT.frontend.tree.RemainderExpression クラス 876
4.5.93 OpenJIT.frontend.tree.ReturnStatement クラス 879
4.5.94 OpenJIT.frontend.tree.ShiftLeftExpression クラス 883
4.5.95 OpenJIT.frontend.tree.ShiftRightExpression クラス 886
4.5.96 OpenJIT.frontend.tree.ShortExpression クラス 889
4.5.97 OpenJIT.frontend.tree.Statement クラス 891
4.5.98 OpenJIT.frontend.tree.StringExpression クラス 907
4.5.99 OpenJIT.frontend.tree.SubtractExpression クラス 910
4.5.100 OpenJIT.frontend.tree.SuperExpression クラス 914
4.5.101 OpenJIT.frontend.tree.SwitchStatement クラス 917
4.5.102 OpenJIT.frontend.tree.SynchronizedStatement クラス . . . 918
4.5.103 OpenJIT.frontend.tree.ThisExpression クラス 922
4.5.104 OpenJIT.frontend.tree.ThrowStatement クラス 926
4.5.105 OpenJIT.frontend.tree.TryStatement クラス 927
4.5.106 OpenJIT.frontend.tree.TypeExpression クラス 928

4.5.107 OpenJIT.frontend.tree.UnaryExpression クラス 930
4.5.108 OpenJIT.frontend.tree.UnsignedShiftRightExpression クラス 938
4.5.109 OpenJIT.frontend.tree.VarDeclarationStatement クラス . . 941
4.5.110 OpenJIT.frontend.tree.WhileStatement クラス 943
4.6 OpenJIT.frontend.util パッケージ 947
4.6.1 OpenJIT.frontend.util.HashtableEntry クラス 947
4.6.2 OpenJIT.frontend.util.HashtableEnumerator クラス 949
4.6.3 OpenJIT.frontend.util.IntKeyHashtable クラス 952
4.6.4 OpenJIT.frontend.util.LookupHashtable 抽象クラス 959
4.6.5 OpenJIT.frontend.util.Queue クラス 961
4.6.6 OpenJIT.frontend.util.QueueEntry クラス 964
4.7 OpenJIT Java Native Code API パッケージ 965
4.7.1 OpenJIT Java Native Code API パッケージ 965

第 5 章 入出力仕様 975
5.1 OpenJIT フロントエンドシステム 975
5.1.1 概要 975
5.1.2 入出力データ仕様 976
5.2 OpenJIT バックエンドシステム 994
5.2.1 概要 994
5.2.2 入出力データ仕様 995

第 6 章 ファイル仕様 1007
6.1 概要 1007
6.2 物理ファイル仕様 1008
6.3 物理データベース仕様 1009

第 7 章 内部データ仕様 1010

第 8 章 システムの性能及び容量 1011

参考文献 1012

表目次

1.1 OpenJIT フロントエンドシステムを構成する機能一覧 9
1.2 OpenJIT バックエンドシステムを構成する機能一覧 9

図目次

1.1 OpenJIT システムの構成図 3
1.2 OpenJIT システムの概要 5

2.1 OpenJIT フロントエンドシステム 23
2.2 OpenJIT バックエンドシステム 24

3.1 OpenJIT システムを構成するパッケージ 31
3.2 OpenJIT コンパイラ基盤機能 107
3.3 OpenJIT バイトコードディスコンパイラ機能 109
3.4 OpenJIT クラスファイルアノテーション解析機能 111
3.5 OpenJIT 最適化機能 113
3.6 OpenJIT フローグラフ構築機能 115
3.7 OpenJIT フローグラフ解析機能 117
3.8 OpenJIT プログラム変換機能 119
3.9 OpenJIT ネイティブコード変換機能 121
3.10 OpenJIT 中間コード変換機能 123
3.11 OpenJIT RTL 変換機能 125
3.12 OpenJIT Peephole 最適化機能 127
3.13 OpenJIT レジスタ割付機能 129

5.1 OpenJIT コンパイラ基盤機能 977
5.2 OpenJIT バイトコードディスコンパイラ機能 983
5.3 OpenJIT クラスファイルアノテーション解析機能 985
5.4 OpenJIT 最適化機能 987
5.5 OpenJIT フローグラフ構築機能 989
5.6 OpenJIT フローグラフ解析機能 991

5.7 OpenJIT プログラム変換機能 993
5.8 OpenJIT ネイティブコード変換機能 996
5.9 OpenJIT 中間コード変換機能 998
5.10 OpenJIT RTL 変換機能 1001
5.11 OpenJIT Peephole 最適化機能 1004
5.12 OpenJIT レジスタ割付機能 1006

第 1 章

全体概要

(1)　目標及び目的

OpenJIT コンパイラシステムは従来型のコンパイラ技術とは異なり，自己反映計算 (リフレクション) の理論に基づいた “Open Compiler”(開放型コンパイラ) 技術をベースとする．具体的にはアプリケーションや計算環境に特化した言語の機能拡張と最適化が行う機能を有する JIT コンパイラである．開発のターゲットは実用性や広範な適用性を考慮して Java 言語とするが，技術的には他の同種のプログラム言語にも適用可能である．

(2) 概要説明

OpenJIT コンパイラシステムは，従来の JIT コンパイラと同様に動作し，かつ以下に述べる拡張機能を有する．最適化やクラスの拡張を行うクラスライブラリの開発者は，Open Compiler の API を用いて，プログラム解析，および最適化の為のプログラム変換，さらに実行時のプローブなどの記述を，クラスのベースレベルのプログラムとは別に Java を用いて記述する．これらは，クラスライブラリの作成者により，ベースレベルのバイトコード，言語拡張部分のバイトコード，および最適化を行うためのバイトコードと，クラスファイルの言語拡張アノテーションに分類される．次に，クラスファイルのユーザがこのクラスファイルをダウンロードすると，言語拡張および最適化機能を実現するライブラリ，さらに (必要な場合は) OpenJIT の各種サポートライブラリを同時にダウンロードする．次に，OpenJIT のディスコンパイラ機能が，言語拡張モジュールと最適化機能が必要な抽象レベル (AST) までバイトコードの逆変換を行う．変換されたプログラムに対し，最適化機能がクラスに特化したプログラム解析，アノテーションの認識，および最適化変換を行う．変換後，再びそれらはバイトコードに変換され，さらに，バイトコードからネィティブコードに変換される．その際にも，JIT に peephole 最適化を行う機能を含む各種最適化機能が起動される．

OpenJIT コンパイラシステムは OpenJIT フロントエンドシステムと OpenJIT バックエンドシステムに大きく分けられる．

さらにこれらは次のような機能によって構成されている (図 1.1)．

- OpenJIT フロントエンドシステム
  - OpenJIT コンパイラ基盤機能
  - OpenJIT バイトコードディスコンパイラ機能
  - OpenJIT クラスファイルアノテーション解析機能
  - OpenJIT 最適化機能
  - OpenJIT フローグラフ構築機能
  - OpenJIT フローグラフ解析機能
  - OpenJIT プログラム変換機能
- OpenJIT バックエンドシステム

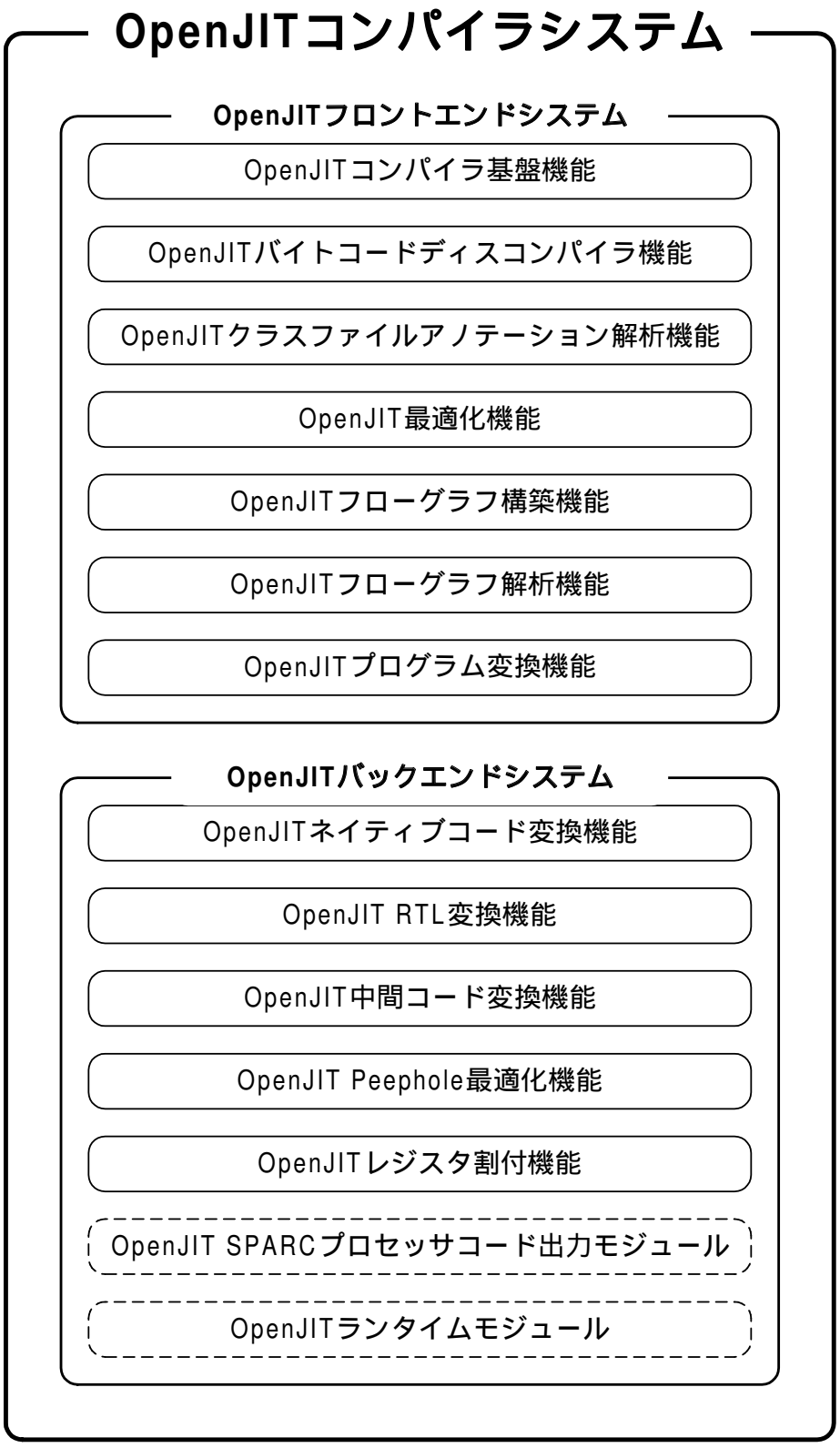

図 1.1: OpenJIT システムの構成図

- OpenJIT ネイティブコード変換機能
- OpenJIT 中間コード変換機能
- OpenJIT RTL 変換機能
- OpenJIT Peephole 最適化機能
- OpenJIT レジスタ割付機能
- OpenJIT SPARC プロセッサコード出力モジュール
- OpenJIT ランタイムモジュール

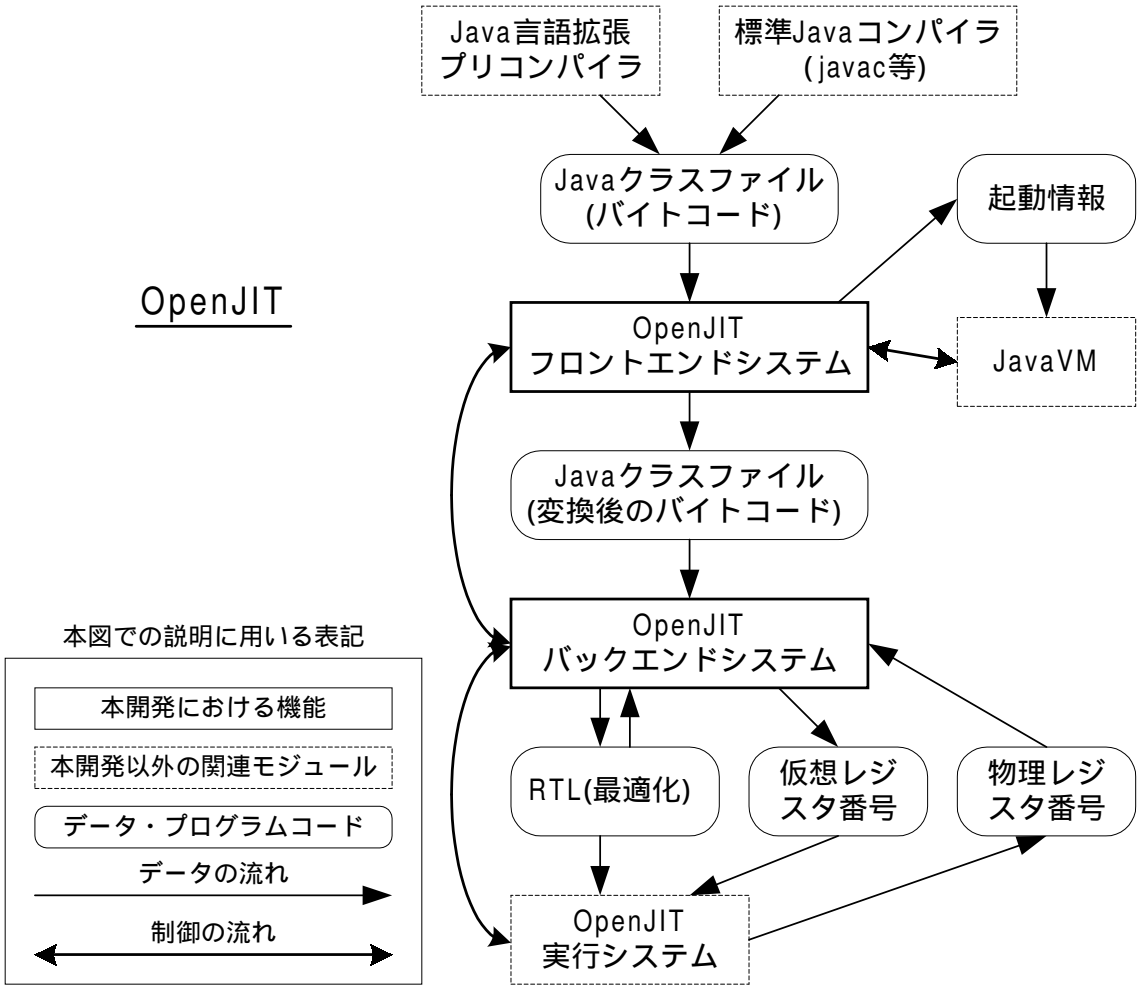

図 1.2: OpenJIT システムの概要

(3) 開発対象範囲

OpenJIT コンパイラシステム全体の構成は図 1.2の通りである．このうち，開発対象となるものは実線の矩形の OpenJIT フロントエンドシステムと OpenJIT バックエンドシステムである．

(4)　ハードウェア構成

本システムは，以下の構成のハードウェア環境で動作する:

- Sun Ultra 60 (UltraSPARC-II×2)
- メモリ量: 256MB
- ハードディスク容量: 4GB

(5)　ソフトウェア構成

本システムは，以下の構成のソフトウェア環境で動作する:

- OS: Sun Solaris 2.6
- Java 実行環境: JDK 1.1.6

(6) 機能仕様

本システムは大きくOpenJITフロントエンドシステムとOpenJITバックエンドシステムの二つに分けられる．OpenJITフロントエンドシステムでは，Javaのバイトコードを入力とし，高レベルな最適化を施して再びバイトコードを出力する．OpenJITバックエンドシステムでは，OpenJITフロントエンドシステムによって得られたバイトコードに対して，より細かいレベルでの最適化を行いネイティブコードを出力する．

表1.1，表1.2にOpenJITフロントエンドシステムとOpenJITバックエンドシステムを構成する機能の一覧を示す．ただし，OpenJITバックエンドシステム内のOpenJIT SPARCプロセッサコード出力モジュール，OpenJITランタイムモジュールは契約の対象外である．

表 1.1: OpenJIT フロントエンドシステムを構成する機能一覧

OpenJIT コンパイラ基盤機能
OpenJIT バイトコードディスコンパイラ機能
OpenJIT クラスファイルアノテーション解析機能
OpenJIT 最適化機能
OpenJIT フローグラフ構築機能
OpenJIT フローグラフ解析機能
OpenJIT プログラム変換機能

表 1.2: OpenJIT バックエンドシステムを構成する機能一覧

OpenJIT ネイティブコード変換機能
OpenJIT 中間コード変換機能
OpenJIT RTL 変換機能
OpenJIT Peephole 最適化機能
OpenJIT レジスタ割付機能
OpenJIT SPARC プロセッサコード出力モジュール
OpenJIT ランタイムモジュール

(a) OpenJIT フロントエンドシステム

OpenJIT コンパイラシステムが与えられたメソッドに対して起動されると，OpenJIT フロントエンドシステムは対象となるバイトコード列に対して以下の処理を行う．

まず，OpenJIT バイトコードディスコンパイラ機能は，バイトコードを逆変換して AST を出力する．この際には，与えられたバイトコード列から，元のソースプログラムから生成されるコントロールグラフのリカバリを行う．

同時に，OpenJIT クラスファイルアノテーション機能により，このクラスファイルのアトリビュート領域に何らかのアノテーションが付記されていたときに，その情報を得る．この情報は，AST 上の付加情報として用いられる．得られた AST に対し，OpenJIT 最適化機能によって，最適化が施される．最適化に必要な情報は，OpenJIT フローグラフ構築機能，OpenJIT フローグラフ解析機能により抽出される．最適化時のプログラム変換は，OpenJIT プログラム変換機能が司って実施され，変換後のバイトコードがバックエンドシステムに出力される．

OpenJIT フロントエンドシステムを構成する機能を以下に示す．

1. OpenJIT コンパイラ基盤機能

   本機能では OpenJIT 全体の基本動作を司る．

   Sun の JDK においては，Java Native Code API(Application Programmer's Interface) というコンパイラに対するインタフェースが用意されている．この API は JVM のインタプリタにネイティブコード生成を組み込むために用意されたものである．今回開発する OpenJIT コンパイラでは，この API に基づくことにより JDK に準拠の VM に OpenJIT コンパイラを組み込むことができる．この JIT コンパイラは JVM から必要なときに読み込まれ動作する．

   - OpenJIT 初期化部

     OpenJIT フロントエンドシステムの各機能，バックエンドシステムの各機能の初期化を指示し，OpenJIT システムの初期化を行う

   - OpenJIT コンパイラフロントエンド制御部

     OpenJIT フロントエンドシステムの各機能の起動および制御を行う．

- OpenJIT JNI API登録部

  バックエンド用のJNI (Java Native Interface) のAPIを登録する．

2. OpenJIT バイトコードディスコンパイラ機能

   OpenJIT バイトコードディスコンパイラ機能は，Javaのクラスファイルのそれぞれのバイトコードを，いわゆるdiscompiler技術により，バイトコードレベルからコントロールフローグラフ，AST(抽象構文木) を含む抽象化レベルのプログラム表現を復元し，以後のOpenJITの各モジュールの操作の対象とするような処理を行なう．

   - バイトコード解析部

     バイトコードを入力とし，その解析をコントロールグラフ出力部，AST出力部に指示する．

   - コントロールグラフ出力部

     バイトコード解析部に入力されたバイトコードに対し，そのコント ロールフローグラフを出力する．

   - AST出力部

     バイトコード解析部に入力されたバイトコードに対し，対応するJavaのASTを出力する．

3. OpenJIT クラスファイルアノテーション解析機能

   OpenJIT クラスファイルアノテーション解析機能は，アノテーション情報を解析し，OpenJITディスコンパイラ機能が生成したプログラムグラフ (AST) に対して，コンパイル時に適切な拡張されたOpenJITのメタクラスを起動できるようにする．

   - アノテーション解析部

     アノテーションを付記したクラスファイルおよびASTを入力とし，クラスファイルに付加されたアノテーションを解析して，必要に応じてメタクラス制御部，アノテーション登録部を制御する．

   - アノテーション登録部

解析されたアノテーションをASTに対する付加データとして登録して，追加情報を付加したASTを出力する．

- メタクラス制御部

  アノテーションにしたがって，適切なメタクラスが起動されるように制御を行う．

4. OpenJIT 最適化機能

   OpenJIT最適化機能は，OpenJITフローグラフ構築機能，OpenJITフローグラフ解析機能，およびOpenJITプログラム変換機能を用い，プログラム最適化を行なう．OpenJITコンパイラには，標準的なコンパイラの最適化を含む最適化ライブラリ構築のためのサポートが準備される．

   - 最適化制御部

     OpenJITフローグラフ構築機能，OpenJITフローグラフ解析機能，OpenJITプログラム変換機能，バイトコード出力部を制御して，入力のバイトコード，AST，コントロールフローグラフから，最適化されたバイトコードの生成を指示する．

   - バイトコード出力部

     OpenJITプログラム変換機能により最適化を行ったバイトコードを出力する．

5. OpenJIT フローグラフ構築機能

   OpenJITフローグラフ構築機能は，ASTおよびコントロールフローグラフを受け取り，対応するデータ依存グラフ，コントロール依存グラフ，を含むフローグラフを出力する．また，クラスファイル間のクラス階層情報情報を得られる場合は，クラスファイルの関係を読み込み，オブジェクト指向解析用のクラス階層グラフも出力する．

   - AST等入力部

     AST，コントロールフローグラフ，クラスファイル間のクラス階層情報を入力し，中間形式に変換して，その情報をもとにデータフローグラフ構築部，コントロール依存グラフ構築部，クラス階層解析部にそれぞれ処理を指示する．

- データフローグラフ構築部

  AST 等入力部に入力されたプログラム情報からデータフローグラフを構築して出力する．

- コントロール依存グラフ構築部

  AST 等入力部に入力されたプログラム情報からコントロール依存グラフを構築して出力する．

- クラス階層解析部

  AST 等入力部に入力されたプログラム情報からクラス階層解析を行い，その情報を付加したクラス階層グラフを構築して出力する．

6. OpenJIT フローグラフ解析機能

ここでは，OpenJIT フローグラフ構築機能で構築されたプログラム表現のグラフに対し，グラフ上の解析を行なう．基本的には，一般的なグラフのデータフロー問題として定式化され，トップダウンおよびボトムアップの解析のベースとなる汎用的なアルゴリズムをサポートする．具体的には，グラフ上のデータフロー問題，マージ，不動点検出，などの一連のアルゴリズムがメソッド群として用意される．

- データフロー関数登録部

  データフロー解析に用いるデータフロー関数を登録する．

  データフロー関数群はこのサブモジュールに含まれる．

- フローグラフ解析部

  コントロールフローグラフ，コントロール依存グラフ，データフローグラフ，クラス階層グラフを入力として，登録されたデータフロー関数を用いてデータフロー解析を行う．

- 不動点検出部

  フローグラフ解析部に入力されたデータフローグラフの不動点を検出する．

- クラス階層解析部

  フローグラフ解析部に入力されたクラス階層グラフと，データフローグラフの解析結果から，クラス階層の解析を行い，その結果を出力する．

7. OpenJIT プログラム変換機能

   OpenJIT プログラム変換機能では，OpenJIT フローグラフ解析機能の結果やユーザのコンパイラのカスタマイゼーションに従って，プログラム変換を行なう．プログラム変換のためには，AST の書き換え規則がユーザによって定義され，AST 上のパターンマッチが行われ，適用された規則に従ってプログラムの書き換えが行われる．書換え規則自身，全て Java のオブジェクトとして定義され，ユーザはあらかじめ書換え規則を定義して，OpenJIT プログラム変換機能に登録しておく．

   - AST 変換ルール登録部

     AST パターンマッチ部で用いるパターンマッチの変換ルールを登録する．パターンマッチの変換ルールはこのサブモジュールに含まれる．

   - AST パターンマッチ部

     AST とプログラム解析結果のデータを入力として，変換ルールのパターンマッチを行う．

   - AST 変換部

     パターンマッチした変換規則を用いて，AST の書き換えを行って変換された AST を出力する．

(b) OpenJIT バックエンドシステム

OpenJIT バックエンドシステムは，OpenJIT フロントエンドシステムによって最適化されたバイトコード列に対し，さらなる最適化処理を行い，ネイティブコードを出力する．

OpenJIT ネイティブコード変換機能はバックエンド系処理全体の抽象フレームワークであり，OpenJIT バックエンドシステムの各機能のインタフェースを定義する．このインタフェースに沿って具体的なプロセッサに応じたクラスでモジュールを記述することにより，様々なプロセッサに対応することが可能となる．

OpenJIT 中間コード変換機能によって，バイトコード列からスタックオペランドを使った中間言語へと変換を行なう．バイトコードの命令を解析して分類することにより，単純な命令列に展開を行う．

得られた命令列に対し，OpenJIT RTL 変換は，このスタックオペランドを使った中間言語からレジスタを使った中間言語 (RTL) へ変換する．バイトコードの制御の流れを解析し，命令列を基本ブロックに分割する．バイトコードの各命令の実行時のスタックの深さを計算することで，スタックオペランドから無限個数あると仮定した仮想的なレジスタオペランドに変換する．

次に，OpenJIT Peephole 最適化機能によって，RTL の命令列の中から冗長な命令を取り除く最適化を行ない，最適化された RTL が出力され，最後に OpenJIT SPARC プロセッサコード出力モジュールにより，SPARC プロセッサのネイティブコードが出力される．OpenJIT SPARC プロセッサモジュールは，ネイティブコード生成時のレジスタ割り付けのために OpenJIT レジスタ割付機能を利用する．出力されたネイティブコードは，JavaVM によって呼び出され実行されるが，その際に OpenJIT ランタイムモジュールを補助的に呼び出す．

1. OpneJIT ネイティブコード変換機能

   Java のバイトコードからネィティブコードを出力するための抽象フレームワークである．実際は，このクラスを具体的なプロセッサに応じたクラスで特化することによって，実際のコード出力機能を定義する．それぞれのバイトコードと，プログラムの各種グラフ，およびフロントエンドシステムのプログラム解析・変換の結果を用いて，ネィティブコードへの変換を行なう．

   - ネイティブコード変換

バイトコードを入力とし，SPARC ネイティブコードを出力する
OpenJIT 中間コード変換機能，OpenJIT RTL 変換機能，OpenJIT Peephole 最適化機能を制御する．

- メソッド情報

  メソッドに関する JDK の内部構造を Java のデータ構造に変換する．

- バイトコードアクセス

  JDK の内部構造であるバイトコードを読み取る．

- 生成コードメモリ管理

  生成するネイティブコードの領域の確保，その領域へのコードの書き込み，開放を司る．

2. OpenJIT 中間コード変換機能

   フロントエンドシステムの出力であるバイトコードを入力とする．バイトコードの各命令をグループに分別し，より単純な中間言語に変換を行なう．メソッド呼び出しのバイトコード命令について，メソッドの引数の数や型の解析を行い，中間言語列に展開する．この中間言語のオペランドはスタックで与えられる．また，Java 特有な命令列パターンを検出し，単純な中間言語に置き換える最適化を含めて行う．

   - 中間言語変換

     バイトコードを入力として，バイトコードの各命令を中間言語に変換して出力する．

   - メソッド引数展開

     メソッド呼び出しの引数を中間言語に展開する．

   - 命令パターンマッチング

     特定のバイトコードパターンに対し最適化した中間言語に変換する．

3. OpenJIT RTL 変換機能

   OpenJIT 中間コード変換機能の生成結果を入力とし，スタックオペランドを使った中間言語からレジスタを使った中間言語，RTL(Register Transfer Language) に変換を行なう．中間言語列を基本ブロックに分割し，実行の制御の流れを解

析することにより，スタックマシンコードからオペランドレジスタコードへの変換を行なう．OpenJIT では，無限資源のレジスタがあるとみなして RTL への変換を行なう．また，オペランドのうち型が未解決のものの型を決定する．

- 基本ブロック分割

  中間言語を入力とし，基本ブロックに分割して基本ブロック情報を出力するまた，中間言語列を入力とし，各中間言語を RTL に変換して出力する．

- コントロールフロー解析

  基本ブロック間の処理の流れを解析する．

4. OpenJIT Peephole 最適化機能

   OpenJIT RTL 変換機能の生成した RTL を入力として，RTL に対して Peephole 最適化を施す．Peephole 最適化としては，通常行われる redundant load/store elimination を行う．また，Java 固有の Peephole 最適化も行なわれる．Java に特有な配列のインデックスの境界チェックを取り除く最適化も行なう．このモジュールは冗長な命令を取り除いて最適化された RTL を出力する．

   - データフロー解析

     RTL 列のデータの流れを解析する．

   - 各種 Peephole 最適化

     基本ブロック情報と RTL を入力として，データフロー解析結果を元に最適化した RTL を出力する．

5. OpenJIT レジスタ割付機能

   ネイティブコード生成の際，実際のプロセッサレジスタへの割付を行なう．レジスタ割付アルゴリズムを適用し，実際のプロセッサレジスタに対して割付を行なう．物理レジスタの数が足りない場合は，一時レジスタを割り付け，スピル / フィルコードを生成する．

   - 仮想レジスタ管理

     中間言語で使用する仮想レジスタの管理をする．

   - 物理レジスタ管理

     生成するネイティブコードが使用する物理レジスタを管理する．

- レジスタ割付

  与えられた仮想レジスタ番号に対して，物理レジスタを割り付け，物理レジスタ番号を返す．

  割り付けできないときはスピル / フィルコードを生成する．

6. OpenJIT SPARC プロセッサコード出力モジュール

   (注: 本開発においては，本モジュールの開発は富士通・大学側の負担で行われ，契約の対象とはしない．)

7. OpenJIT ランタイムモジュール

   (注: 本開発においては，本モジュールの開発は富士通・大学側の負担で行われ，契約の対象とはしない．)

第 2 章

設計指針

(1)　実現方式

OpenJIT コンパイラシステムは，OpenJIT フロントエンドシステムと OpenJIT バックエンドシステムから構成される．

OpenJIT フロントエンドシステムは，Java のバイトコードを入力とし，高レベルな最適化を施して，再びバイトコードを出力する．OpenJIT バックエンドシステムは，得られたバイトコードに対し，より細かいレベルでの最適化を行い，ネイティブコードを出力する．

OpenJIT コンパイラシステムが与えられたメソッドに対して起動されると，OpenJIT フロントエンドシステムは対象となるバイトコード列に対して以下の処理を行う．

まず，OpenJIT バイトコードディスコンパイラ機能は，バイトコードを逆変換して AST を出力する．この際には，与えられたバイトコード列から，元のソースプログラムから生成されるコントロールグラフのリカバリを行う．

同時に，OpenJIT クラスファイルアノテーション機能により，このクラスファイルのアトリビュート領域に何らかのアノテーションが付記されていたときに，その情報を得る．この情報は，AST 上の付加情報として用いられる．得られた AST に対し，OpenJIT 最適化機能によって，最適化が施される．最適化に必要な情報は，OpenJIT フローグラフ構築機能，OpenJIT フローグラフ解析機能により抽出される．最適化時のプログラム変換は，OpenJIT プログラム変換機能が司って実施され，変換後のバイトコードがバックエンドシステムに出力される．

次に，OpenJIT フロントエンドシステムによって最適化されたバイトコード列に対

し，OpenJIT バックエンドシステムは，さらなる最適化処理を行い，ネイティブコードを出力する．

OpenJIT ネイティブコード変換機能はバックエンド系処理全体の抽象フレームワークであり，OpenJIT バックエンドシステムの各機能のインタフェースを定義する．このインタフェースに沿って具体的なプロセッサに応じたクラスでモジュールを記述することにより，様々なプロセッサに対応することが可能となる．

OpenJIT 中間コード変換機能によって，バイトコード列からスタックオペランドを使った中間言語へと変換を行なう．バイトコードの命令を解析して分類することにより，単純な命令列に展開を行う．

得られた命令列に対し，OpenJIT RTL 変換は，このスタックオペランドを使った中間言語からレジスタを使った中間言語 (RTL) へ変換する．バイトコードの制御の流れを解析し，命令列を基本ブロックに分割する．バイトコードの各命令の実行時のスタックの深さを計算することで，スタックオペランドから無限個数あると仮定した仮想的なレジスタオペランドに変換する．

次に，OpenJIT Peephole 最適化機能によって，RTL の命令列の中から冗長な命令を取り除く最適化を行ない，最適化された RTL が出力され，最後に OpenJIT SPARC プロセッサコード出力モジュールにより，SPARC プロセッサのネイティブコードが出力される．OpenJIT SPARC プロセッサモジュールは，ネイティブコード生成時のレジスタ割り付けのために OpenJIT レジスタ割付機能を利用する．出力されたネイティブコードは，JavaVM によって呼び出され実行されるが，その際に OpenJIT ランタイムモジュールを補助的に呼び出す．

OpenJIT コンパイラは，Java 言語自身でポータブルに記述され，Java Virtual Machine の JIT に対する標準 API を満たすように作成される．具体的には，OpenJIT は，以下の各機能で構成される (図 2.1,2.2参照．このうち実線の長方形は今回開発される各機能を，丸まった長方形はデータを，点線は既存のモジュール，あるいは本開発をベースとして，我々が将来研究開発するモジュールを表す)．

OpenJIT フロントエンドシステム

1. OpenJIT コンパイラ基盤機能

   OpenJIT 全体の基本動作を司る．

2. OpenJIT バイトコードディスコンパイラ機能
   Java のクラスファイルに含まれるバイトコードを，いわゆる discompiler 技術により，コンパイラ向けの中間表現に変換する．

3. OpenJIT クラスファイルアノテーション解析機能
   プログラムグラフ (AST) に対して，コンパイル時に適切な拡張された OpenJIT のメタクラスを起動できるようにする．

4. OpenJIT 最適化機能
   各種解析モジュールおよびプログラム変換モジュールを用い，プログラム最適化を行なう．

5. OpenJIT フローグラフ構築機能
   AST およびコントロールフローグラフを受け取り，対応するデータ依存グラフ，コントロール依存グラフ，などのフローグラフを出力する．

6. OpenJIT フローグラフ解析機能
   フローグラフ構築モジュールで構築されたプログラム表現のグラフに対し，グラフ上の解析を行なう．

7. OpenJIT プログラム変換機能
   OpenJIT フローグラフ解析機能の結果やユーザのコンパイラのカスタマイゼーションに従って，プログラム変換を行なう．

OpenJIT バックエンドシステム

8. OpenJIT ネイティブコード変換機能
   OpenJIT バックエンド全体の制御を行う．

9. OpenJIT 中間コード変換機能
   バイトコードから中間言語への変換を行う．

10. OpenJIT RTL 変換機能
    中間言語から RTL への変換を行う．

11. OpenJIT Peephole 最適化機能
    RTL を最適化し，無駄な命令を取り除いて最適化を行う．

12. OpenJIT レジスタ割付機能
    仮想レジスタから物理レジスタへの割付を行う．

13. OpenJIT SPARC プロセッサコード出力モジュール
    RTL からネイティブコードへの変換を行う．

14. OpenJIT ランタイムモジュール
    ネイティブコードが実行時に呼び出し，ネイティブコードの実行ををサポートする．

注: 13, 14は，今回開発は行われるが，契約の対象外とする．

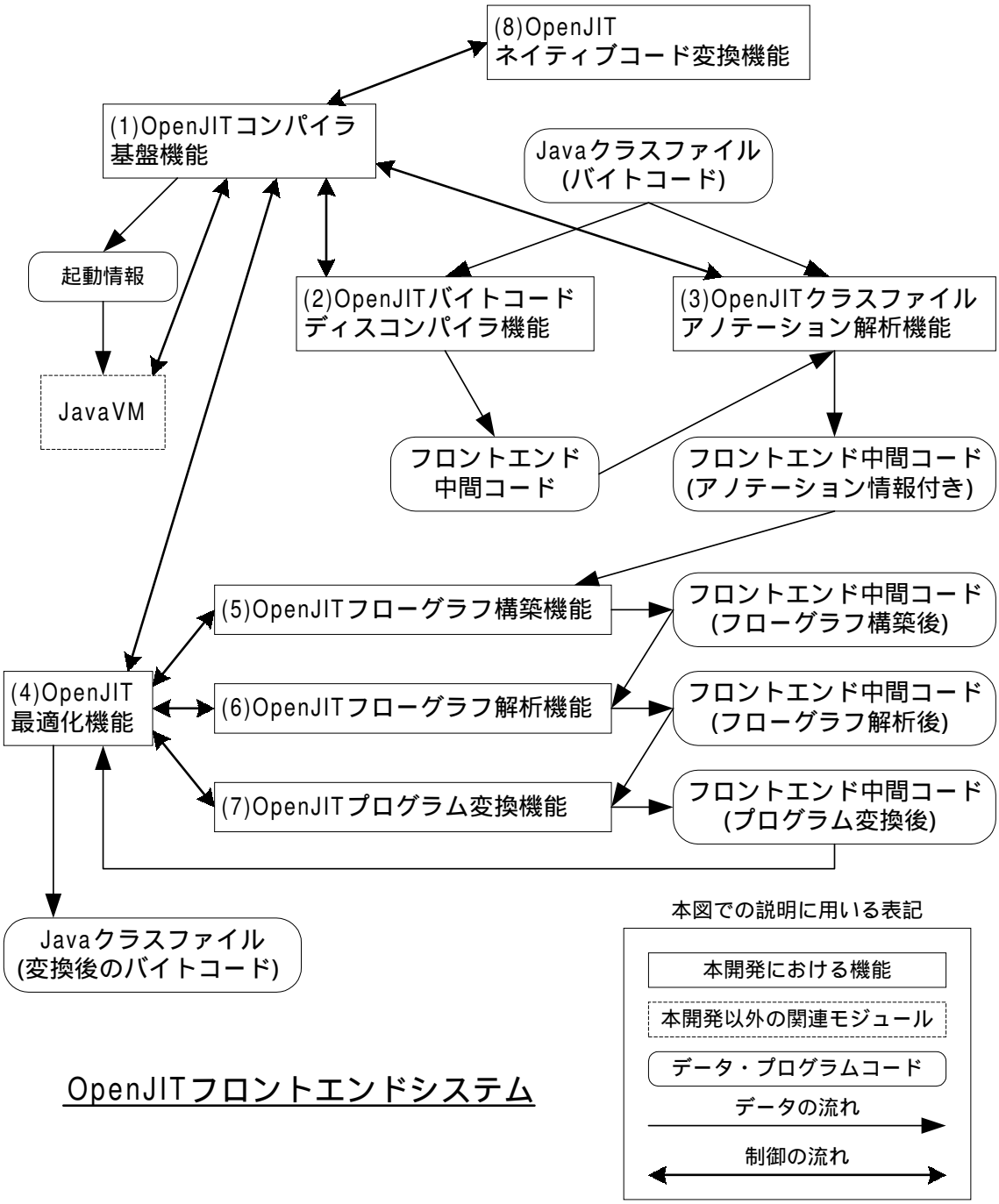

図 2.1: OpenJIT フロントエンドシステム

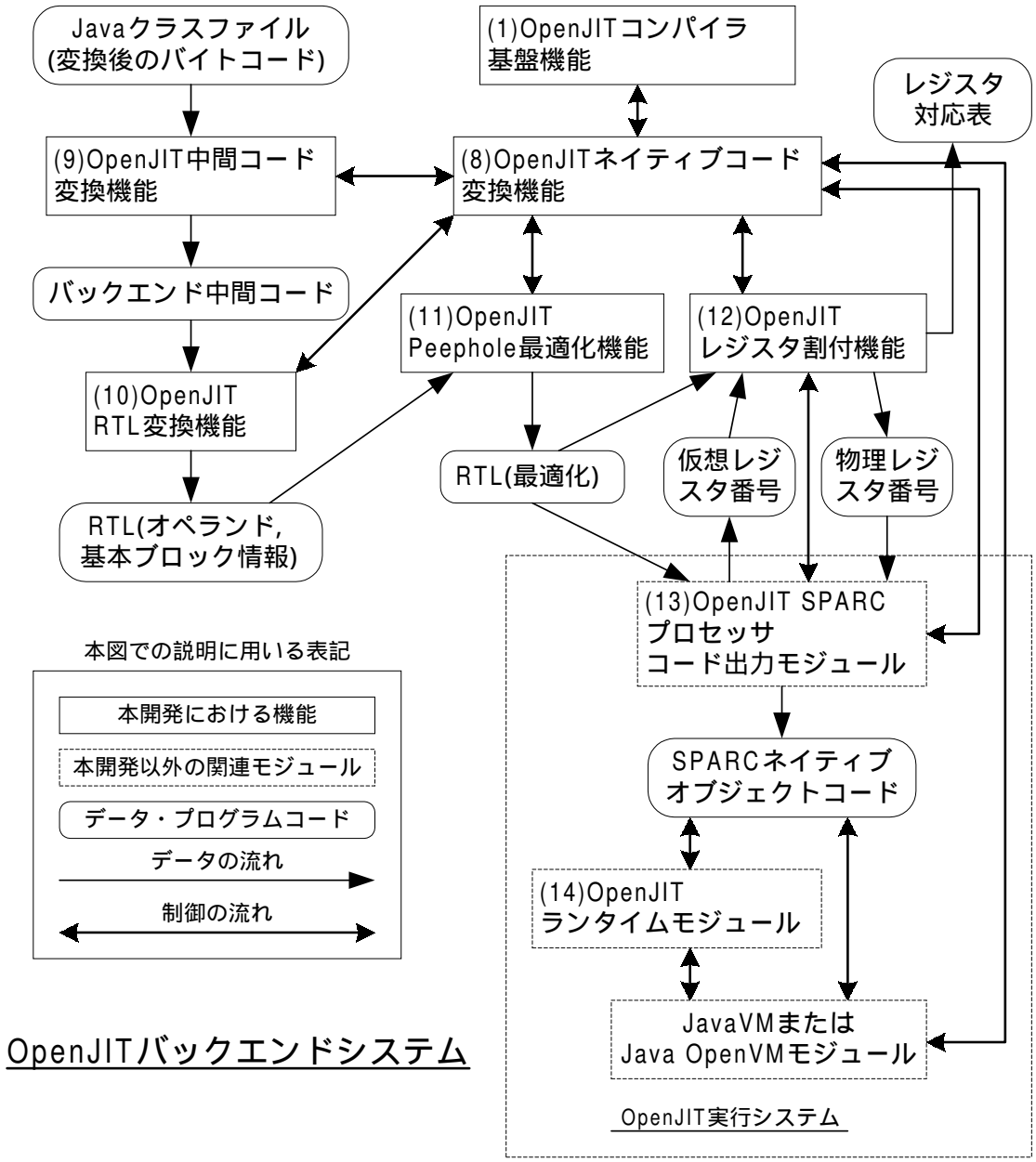

図 2.2: OpenJIT バックエンドシステム

(2)　他システムとの関連

Java 仮想マシン (JavaVM) には外部の JIT コンパイラを組み込むインターフェース・API が準備されている．具体的には JIT は各クラスに対するメソッドディスパッチ部位を書き換え，直接メソッド本体ではなく，JIT コンパイラが起動されるようにしておく．メソッド呼び出しによって起動された JIT コンパイラは，メソッドを構成するバイトコードをその場でコンパイルし，ネイティブコードを得て，ヒープ領域に格納する．メソッドディスパッチ部位をさらに書き換え，以後の起動では直接ネイティブコードが起動されるようにする．

このインターフェース・API は，Sun Microsystems により定められた Java JIT Interface の仕様に基づいている．

(3)　拡張性，保守性，信頼性

本システムの拡張性，保守性，信頼性における指針およびその利点を以下に示す．

- 本システムは Open Compiler 技術に基づき，コンパイラをオブジェクト指向設計でモジュール化することにより，ユーザによる変更を可能にしている．さらに，コンパイラメタコードを記述することにより，コンパイル時にメタ計算を行ない，言語の拡張やアプリケーション固有の最適化を行うなどの拡張性を持たせることが可能になっている．

- オブジェクト指向設計に基づき，各機能をモジュール化することによって，アーキテクチャ依存部分を限定することが可能となるため，移植性が高まる．

- オブジェクト指向設計に基づき，各機能をモジュール化することによって，機能の変更・拡張のような場合にはそのモジュールのみを更新すればよいため，システムの保守が容易である．

- Java 言語を用いることにより，Java 言語自身が持つセキュリティ機能により，不正なプログラムは実行前に検出される．また，ユーザによるコンパイラメタコードの記述ミスなど，なんらかの原因によりコンパイルが失敗した場合には，インタプリタに制御を戻すことが可能である．

(4)　設計方法，文書化，設計手順，変更方法

本システムの設計および文書化については次のように定める．

(a)　設計方針

本システムの設計にあたってはオブジェクト指向による設計を採用する．

本システムは実用性・広範囲な適用性を考慮して，基本的に Java 言語を用いて記述される．Java 言語自身，オブジェクト指向言語であるため，オブジェクト指向に基づいた設計を行うことは，上で述べた利点を確保する上でも不可欠である．

オブジェクト指向による設計においては，全てのプログラムモジュール・データ構造はオブジェクトとして表現する．つまり，各プログラムモジュールやデータ構造はそれぞれ密接に関連することになる．このため，前に述べた機能モジュールとプログラムモジュール・データ構造は一対一に必ずしも対応しない．この場合，機能モジュールとサブプログラムの関係がこれに相当する．そこで本システムの設計では，機能モジュールとサブプログラムを一対一に対応させるような制限は設けず，柔軟に対応できるようにする．これにより，上で述べた利点を十分に生かすことが可能となる．

(b)　文書化

本システムはオブジェクト指向による設計によって実現されている．このため，本システムを構成するモジュールおよびデータ構造は全てオブジェクトとして実現される．このため，これらモジュール・データ構造の文書化においては次のようなスタイルを採用する．

4 章で説明されるサブプログラムを構成する各クラス・インターフェースに対しては，

- 概要
- フィールド (変数)
- コンストラクタ
- メソッド

について列挙・概説する．ただし，フィールド・コンストラクタ・メソッド・例外に関してはクラスによっては存在しないものがあるため，この場合は記述されない．

このうちコンストラクタ・メソッドにおいては，

- 機能概要
- 機能説明
- 入力データ
- 出力データ
- 処理方式
- エラー処理

のようなスタイルで説明する．

C 言語で書かれる一部のモジュールでは次のようなスタイルとなる．

- 概要
- 変数

- 関数
- エラー処理

について列挙・概説する．ただし，変数・関数・エラー処理に関しては，存在しないものがあるため，この場合は記述されない．

　各機能モジュールに含まれるクラス・インターフェースは相当な数になるため，各クラス・インターフェース間の相関関係などについての記述は複雑で冗長なものとなるので，その場合は適時，表などに置き換えて概説する．

第 3 章

プログラム構成

本システムは Java 言語および，C 言語を用いて開発されている．Java 言語で開発された部分はクラスをパッケージとしてモジュール化している．このため，プログラム構成は，パッケージを単位とした構成で記述する．

3.1 全体構成 (図)

本システムは図 3.1に示すように以下の 7 つのパッケージから構成されている．

- OpenJIT パッケージ
- OpenJIT.frontend.asm パッケージ
- OpenJIT.frontend.disompiler パッケージ
- OpenJIT.frontend.flowgraph パッケージ
- OpenJIT.frontend.tree パッケージ
- OpenJIT.frontend.util パッケージ
- OpenJIT Java Native Code API パッケージ

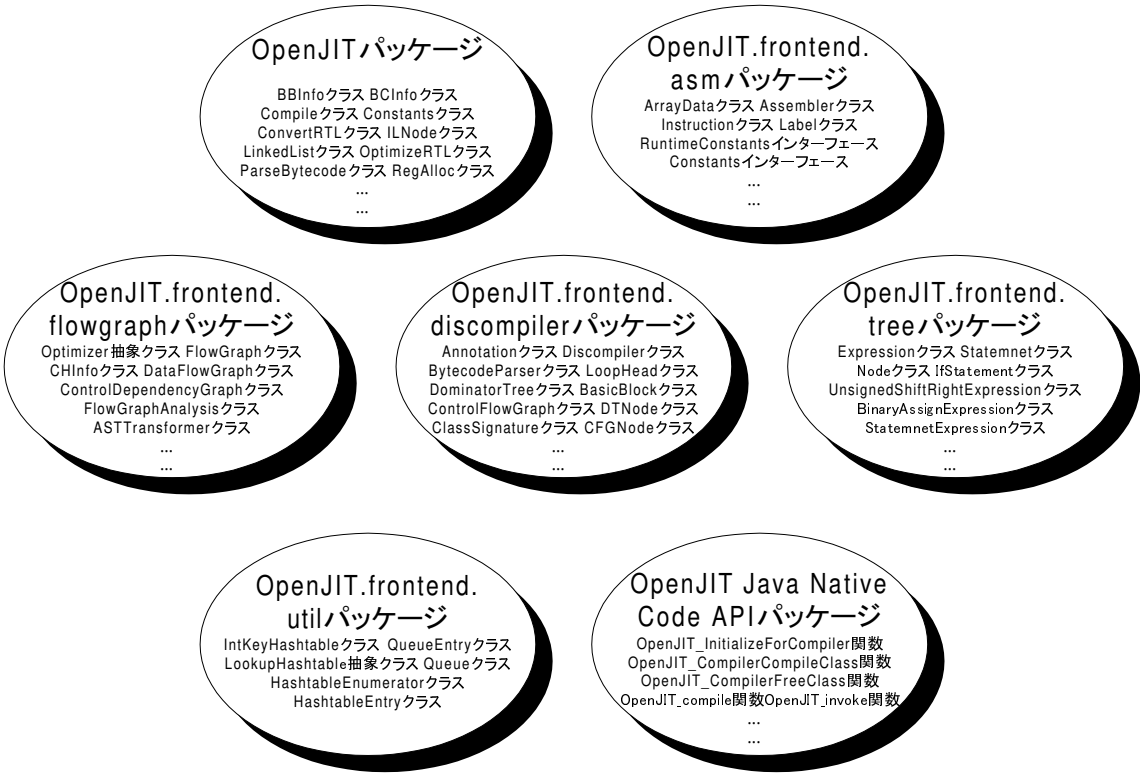

図 3.1: OpenJIT システムを構成するパッケージ

3.2 サブプログラム概要（目的，機能）

図 3.1に現れるパッケージの目的，および機能を記述する．

3.2.1 OpenJIT パッケージ

(1) パッケージの構成

OpenJIT パッケージには以下のクラスが含まれる．

- BBInfo クラス

  基本ブロック情報を表現する．

- BCinfo クラス

  バイトコードに関する情報を表現する．

- Compile クラス

  OpenJIT コンパイラ基盤機能と OpenJIT ネイティブコード変換機能の中核をなすクラス．

- CompileError クラス

  OpenJIT システムにおいて，コンパイル中にエラーが起きたときのエラーを表すクラス．

- Constants インターフェース

  OpenJIT の各クラスで利用する定数を定義したインタフェース．

- ConvertRTL クラス

  このクラスは OpenJIT RTL 変換機能を実現する．

- ExceptionHandler クラス

  バイトコードにおける例外処理に関する情報を表すクラスである．

- ILnode クラス

  バックエンド中間コードおよびＲＴＬのノードを表現するクラス．OpenJIT 中間コード変換機能により生成され，OpenJIT ＲＴＬ変換機能によりＲＴＬに変換される．

- LinkedList クラス

  連結リストを構成するための基本クラス．BCinfo, ILnode クラスのスーパークラスである．

- OptimizeRTL クラス

  このクラスは OpenJIT Peephole 変換機能を実現する．

- ParseBytecode クラス

  このクラスは OpenJIT 中間コード変換機能を実現する．

- RegAlloc クラス

  このクラスは OpenJIT レジスタ割付機能を実現する．

- RegAlloc$PhyRegs クラス

  OpenJIT.RegAlloc クラスのクラス内クラスであり，物理レジスタを管理する情報を扱う．このクラスの一つのオブジェクトで，32 個の物理レジスタを管理する．

- RegAlloc$VirReg クラス

  OpenJIT.RegAlloc クラスのクラス内クラスであり，仮想レジスタを管理する情報を扱う．一つの仮想レジスタに対して，このクラスの一つのオブジェクトが割り当てられる．

- Select クラス

  あるメソッドが与えらたとき，そのメソッドをコンパイルするかどうか選択するために利用するクラス．

- Var クラス

OpenJIT Peephole最適化機能の実現のために使われるクラス．OpenJIT.OptimizeRTL クラスからのみ参照される．ＲＴＬのオペランドのレジスタの定義，参照関係の情報を保持するために用いる．

- Runtime クラス

  (契約の対象外)

- Sparc クラス

  (契約の対象外)

(2) 実現機能

これらのクラスファイルで実現される機能は以下の通りである．

- OpenJIT コンパイラ基盤機能
  - OpenJIT 初期化部
  - OpenJIT コンパイラフロントエンド制御部
  - OpenJIT JNI API登録部
- OpneJIT ネイティブコード変換機能
  - ネイティブコード変換
  - メソッド情報
  - バイトコードアクセス
  - 生成コードメモリ管理
- OpenJIT 中間コード変換機能
  - 中間言語変換
  - メソッド引数展開
  - 命令パターンマッチング
- OpenJIT RTL 変換機能

  - 基本ブロック分割
  - コントロールフロー解析

- OpenJIT Peephole 最適化機能
  - データフロー解析
  - 各種 Peephole 最適化

- OpenJIT レジスタ割付機能
  - 仮想レジスタ管理
  - 物理レジスタ管理
  - レジスタ割付

- OpenJIT SPARC プロセッサコード出力モジュール (契約対象外)
- OpenJIT ランタイムモジュール (契約対象外)

3.2.2 OpenJIT.frontend.asm パッケージ

(1) パッケージの構成

OpenJIT.frontend.asm パッケージには以下のクラスが含まれる．

- ArrayData クラス

  行列データを格納するクラス．

- Assembler クラス

  バイトコードの生成を行う．

- OpenJIT.frontend.asm.Constants インターフェース

  OpenJIT.frontend.asm パッケージ，OpenJIT.frontend.tree パッケージで用いる定数を定義する．

- Instruction クラス

  バイトコード命令 1 個を表現するクラス．

- Label クラス

  バイトコード命令の内，ラベル命令を表現するクラス．

- OpenJIT.frontend.asm.RuntimeConstants インターフェース

  OpenJIT.frontend.asm パッケージ，OpenJIT.frontend.tree パッケージで用いる定数を定義する．

(2) 実現機能

これらのクラスファイルで実現される機能は以下の通りである．

- OpenJIT 最適化機能
  - 最適化制御部
  - バイトコード出力部

- OpenJIT プログラム変換機能
  - AST 変換ルール登録部
  - AST パターンマッチ部
  - AST 変換部

3.2.3 OpenJIT.frontend.discompiler パッケージ

(1) パッケージの構成

OpenJITfrontend.discompiler パッケージには以下のクラスが含まれる．

- ASTFactory インターフェース

  このインターフェースは，ディスコンパイルの結果として出力する AST の形式を定めるためのもので，AST の種類ごとにインターフェースメソッドとして用意されたファクトリメソッドを定義することにより，ユーザの求める AST オブジェクトの形式をディスコンパイラの出力として得ることができる．

- Annotaion クラス

  このクラスは，クラスファイルに付加されるアノテーション情報の構造を定義するものである．

- AnnotationAnalyzer クラス

  このクラスは，クラスファイルに付加されたアノテーション情報を読み込み，それを Annotation オブジェクトとして構築する機能を提供するものである．

- BBABasicBlock クラス

  このクラスは，コントロールフロー解析の対象となるコントロールフローグラフノードの形式を定義するものである．

- BasicBlock クラス

  このクラスは，ディスコンパイルの目的となったメソッドのバイトコードの解析により分割された，それぞれのベーシックブロックを表現するオブジェクトを定義するクラスであり，コントロールフローグラフのノードを表す CFGNode のサブクラスである．

- BasicBlockAnalyzer クラス

  このクラスはベーシックブロック単位のディスコンパイル処理 (シンボリック実行) を行い，コントロールフローグラフの各ノードがバイトコードで表現されて

いる ControlFlowGraph よりもより抽象度の高いコントロールフローグラフを構築し，また格納する．

- BranchTableEntry クラス

  VM 命令の tableswitch および lookupswitch に対応する VMInstruction オブジェクトのキーとジャンプ先の組を表すためのクラスである．

- BytecodeParser クラス

  このクラスは，ディスコンパイルの対象となるバイト列として表現されているメソッドを，より抽象度の高い VM の命令列に変換して，後のディスコンパイルの下準備を行うと共に，ベーシックブロックの先頭となるべき命令を見つけ出し，コントロールフローグラフの構築の下準備も行う，Java メソッドの命令レベルの解析器を実現するものである．

- CFAConstants インターフェース

  このインターフェースは，コントロールフロー解析で使用する定数を定義するためのものである．

- CFAFlag クラス

  CFAFlag はコントロールフロー解析の結果として，コントロールフローグラフのノードが Java 言語レベルの何らかの制御構造をつくるノードであるとわかった場合に，対応するドミネータツリーノードに付けられる情報であり，ディスコンパイルの途中結果を表現する．

- CFGNode クラス

  このクラスは，コントロールフローグラフノードの基本的な構造を定義するものである．

- Case クラス

  このクラスは，コントロールフロー解析の結果として判明した switch 構造の中の，case ラベルが付けられるブロックに対し，case ラベルの内容を格納するためのものである．

- ClassSignature クラス

  このクラスは，VM の命令 anewarray および multianewarray で用いられている配列の型記述子を解析し，ディスコンパイラ内で利用しやすい形式に変換するためのものである．

- ControlFlowAnalyzer クラス

  このクラスは，ディスコンパイルのために用いられるコントロールフロー解析の機能を提供すると共に，解析結果として得られる構造化された (structured な) コントロールフローグラフを保持するためのものである．

- ControlFlowGraph クラス

  このクラスは，ディスコンパイルで使用するコントロールフローグラフを生成する機能を提供すると共に，構造化されていない (unstructured な) コントロールフローグラフを保持するためのものである．

- Discompiler クラス

  このクラスは，OpenJIT のフロントエンドの提供する，Java バイトコードのディスコンパイル機能を実現するためのものである．

- DiscompilerError クラス

  このクラスは，ディスコンパイラモジュールにおいて，ディスコンパイル中に起きたエラーの内容を表すクラス．

- DTNode クラス

  このクラスは，ディスコンパイラがコントロールフロー解析で用いるドミネータツリーの各ノードの構造を定めるものである．

- DTVisitor 抽象クラス

  このクラスは，DTNode クラスの提供する，ドミネータツリーを渡り歩き，各ノードに対して何らかの処理を行うという機能について，各ノードで行うべき処理を定義するための構造を提供する．

- DefaultASTFactory クラス

このクラスは，OpenJIT ディスコンパイラのデフォルトの出力形式として OpenJIT.frontend.tree パッケージで定義されている AST の形式を採用するための，ASTFactory インターフェースを実装するものである．

- DominatorTree クラス

  このクラスは，デコンパイラがコントロールフロー解析の際に使用するドミネータツリーをコントロールフローグラフから構築し，また，構築したツリーを格納する機能を提供するものである．

- ExpressionAnalyzer クラス

  このクラスは，コントロールフローグラフを渡り歩くことにより，複数のベーシックブロックの組合わせが一つの式を表しているようなパターンを見つけ出し，Java 言語の式レベルの構造が，コントロールフローグラフに含まれる各ベーシックブロック内で閉じた状態のコントロールフローグラフに作り替える機能を提供するものである．

- FlowEdge クラス

  このクラスは，ドミネータツリーの構築時に使用されるコントロールフローグラフのエッジの内部表現形式を定義するものである．

- LoopHead クラス

  このクラスは，コントロールフロー解析の結果として，何らかのループ構造の先頭ノードであると判明したブロックに対して付けられ，ループ構造の先頭であることを示すためのものである．

- Metaclass 抽象クラス

  このクラスは，クラスファイルに付加されるアノテーション情報に対応するメタクラスのインターフェースを定めるものである．

- MethodInformation インターフェース

  このインターフェースは，ディスコンパイラが必要とするメソッドの情報を受け取るために使用するインターフェースを定義するものである．

- NameAndType クラス

  このクラスは，クラス名とフィールドもしくはメソッドの名前，そして型記述子を解析し，型情報を使用しやすい形に変換する機能を提供するものである．

- RuntimeMethodInformation クラス

  このクラスは，ディスコンパイラが必要とするメソッドの情報を OpenJIT のコンパイラ基盤モジュールから受け取るために必要なインターフェースを，実装するものである．

- Switch クラス

  このクラスは，コントロールフロー解析の結果として，switch 構造の先頭ノードであると判明したブロックに対して付けられるオブジェクトの構造を定義するものである．

- SymbolicConstants インターフェース

  このインターフェースは，SymbolicValue の種別を表す定数を定義するものである．

- SymbolicStack クラス

  このクラスは，ディスコンパイルの際に行われるシンボリック実行の途中経過および結果を格納するためのデータ構造を定義するものである．

- SymbolicValue クラス

  このクラスは，ディスコンパイルの際に行われるシンボリック実行の途中経過や結果を表すために使用される SymbolicStack に積まれるオブジェクトの構造を定義するものであり，それぞれのオブジェクトは，AST の部分式あるいは，コントロールフローに影響を与える命令などを表す．

- VMInstruction クラス

  このクラスは，ディスコンパイラで使用されるバイトコードの命令レベルの解析結果を表すために，VM の命令単位に作られるオブジェクトを定義するものである．

(2) 実現機能

これらのクラスファイルで実現される機能は以下の通りである．

- OpenJIT バイトコードディスコンパイラ機能
  - バイトコード解析部
  - コントロールグラフ出力部
  - AST 出力部
- OpenJIT クラスファイルアノテーション解析機能
  - アノテーション解析部
  - アノテーション登録部
  - メタクラス制御部

3.2.4 OpenJIT.frontend.flowgraph パッケージ

(1) 実現機能

OpenJIT.frontend.flowgraph パッケージには以下のクラスが含まれる．

- FlowGraphAnalyzer インターフェース

  フローグラフ解析を行うために必要なメソッド等を定義したインターフェース．フローグラフ解析に用いる関数はこのインターフェースを実装し，登録する必要がある．

- Optimizer 抽象クラス

  このクラスは，Java バイトコードの最適化機能を実現するための枠組みを定義するものである．

- FlowGraph クラス

  データフローグラフ・コントロール依存グラフ・クラス階層グラフを構築し，構築されたグラフの情報を保持する．

- CHInfo クラス

  クラス階層グラフの 1 ノードを表現する．

- ControlDependencyGraph クラス

  コントロール依存グラフの構築と，構築されたコントロール依存グラフを保持し，フローグラフ解析時に必要な情報を提供する．

- DataFlowGraph クラス

  データフローグラフの構築と，構築されたデータフローグラフを保持し，フローグラフ解析時に必要な情報を提供する．

- ClassHierarchyGraph クラス

  クラス階層解析を行うことによりクラス階層グラフを構築し，構築されたクラス階層グラフを保持する．さらに，フローグラフ解析時に必要な情報を提供する．

- FlowGraphAnalysis クラス

  フローグラフ解析を制御する．フローグラフ構築で得られたデータフローグラフ，コントロール依存グラフ，クラス階層グラフとコントロールフローグラフを用いて，必要な解析を行う関数を登録し，各種フローグラフ解析 (reaching analysis, available analysis, liveness analysis, fixed point detection) を行い，それらの解析で得られた結果を保持し，プログラム変換時に必要な情報を提供する．

- DFFunctionRegisiter クラス

  フローグラフ解析時に解析を行う関数を登録する．

- ReachingAnalyzer クラス

  データフロー解析の 1 つである到達定義の解析 (reaching analysis) を行い，その解析結果を保持する．

- AvailableAnalyzer クラス

  データフロー解析の 1 つである，利用可能な式の解析 (available analysis) を行い，その解析結果を保持する．

- LivenessAnalyzer クラス

  データフロー解析の 1 つである，生きている式の解析 (liveness analysis) を行い，そこ解析結果を保持する．

- FixedPointDetector クラス

  フローグラフ解析の 1 つである，不動点検出 (fixed point detection) を行い，その解析結果を保持する．

- ClassHierarchyAnalyzer クラス

  フローグラフ解析と共にクラス階層解析を行い，その解析結果を保持する．

- ASTTransformer クラス

  フローグラフ解析の結果を元に AST レベルでのプログラム変換を行う．

(2) 実現機能

これらのクラスファイルで実現される機能は以下の通りである．

- OpenJIT フローグラフ構築機能
  - AST 等入力部
  - データフローグラフ構築部
  - コントロール依存グラフ構築部
  - クラス階層解析部
- OpenJIT フローグラフ解析機能
  - データフロー関数登録部
  - フローグラフ解析部
  - 不動点検出部
  - クラス階層解析部
- OpenJIT プログラム変換機能
  - AST 変換ルール登録部
  - AST パターンマッチ部
  - AST 変換部

3.2.5 OpenJIT.frontend.tree パッケージ

(1) パッケージの構成

OpenJIT.frontend.tree パッケージには以下のクラスが含まれる．

- AddExpression クラス

  このクラスは，AST の + 式のノードを表現するとともに，コード生成の機能を実現する．

- AndExpression クラス

  このクラスは，AST の AND 演算子のノードを表現するとともに，コード生成の機能を実現する．

- ArrayAccessExpression クラス

  このクラスは，AST の配列アクセス式のノードを表現するとともに，コード生成の機能を実現する．

- ArrayExpression クラス

  このクラスは，AST の配列のノードを表現するとともに，コード生成の機能を実現する．

- AssignAddExpression クラス

  このクラスは，AST の += 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignBitAndExpression クラス

  このクラスは，AST の &= 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignBitOrExpression クラス

  このクラスは，AST の |= 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignBitXorExpression クラス

  このクラスは，AST の^= 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignDivideExpression クラス

  このクラスは，AST の /= 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignExpression クラス

  このクラスは，AST の = 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignMultiplyExpression クラス

  このクラスは，AST の *= 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignOpExpression クラス

  このクラスは，AST の +=, &=, |=, ^=, /=, *=, %=, <<=, >>=, -=, >>>= 式のノードのスーパークラスであり，これらに共通のコード生成の機能を実現する．

- AssignRemainderExpression クラス

  このクラスは，AST の %= 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignShiftLeftExpression クラス

  このクラスは，AST の <<= 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignShiftRightExpression クラス

  このクラスは，AST の >>= 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignSubtractExpression クラス

  このクラスは，AST の –= 式のノードを表現するとともに，コード生成の機能を実現する．

- AssignUnsignedShiftRightExpression クラス

  このクラスは，AST の >>>= 式のノードを表現するとともに，コード生成の機能を実現する．

- BinaryArithmeticExpression クラス

  このクラスは，AST の +, –, *, / 式のノードのスーパークラスであり，これらに共通のコード生成の機能を実現する．

- BinaryAssignExpression クラス

  このクラスは，AssignExpression, AssignOpExpression クラスのスーパークラスであり，これらに共通のコード生成の機能を実現する．

- BinaryBitExpression 抽象クラス

  このクラスは，BitAndExpression, BitOrExpression, BitXorExpression クラスのスーパークラスであり，ビット演算に共通するコード生成の機能を実現する．

- BinaryCompareExpression クラス

  このクラスは，GreaterExpression, GreaterOrEqualExpression, LessExpression, LessOrEqualExpression クラスのスーパークラスであり，比較演算に共通するコード生成の機能を実現する．

- BinaryEqualityExpression クラス

  このクラスは，EqualExpression, NotEqualExpression クラスのスーパークラスであり，等号・不等号演算に共通するコード生成の機能を実現する．

- BinaryExpression クラス

  このクラスは，BinaryArithmeticExpression, BinaryAssignExpression, BinaryCompareExpression, BinaryEqualityExpression, BinaryLogicalExpression, BinaryShiftExpression, CastExpression, CommaExpression, ConditionalExpres-

sion, InstanceOfExpression クラスのスーパークラスであり，2 項演算に共通するコード生成の機能を実現する．

- BinaryLogicalExpression 抽象クラス

  このクラスは，AndExpression, OrExpression クラスのスーパークラスであり，これらに共通するコード生成の機能を実現する．

- BinaryShiftExpression クラス

  このクラスは，ShiftLeftExpression, ShiftRightExpression, UnsignedShiftRightExpression クラスのスーパークラスであり，これらに共通のコード生成の機能を実現する．

- BitAndExpression クラス

  このクラスは，AST の & 演算子のノードを表現するとともに，コード生成の機能を実現する．

- BitNotExpression クラス

  このクラスは，AST の ! 演算子のノードを表現するとともに，コード生成の機能を実現する．

- BitOrExpression クラス

  このクラスは，AST の | 演算子のノードを表現するとともに，コード生成の機能を実現する．

- BitXorExpression クラス

  このクラスは，AST の ^ 演算子のノードを表現するとともに，コード生成の機能を実現する．

- BooleanExpression クラス

  このクラスは，AST の boolean 型ノードを表現するとともに，コード生成の機能を実現する．

- BreakStatement クラス

このクラスは，AST の Break 文のノードを表現するとともに，コード生成の機能を実現する．

- ByteExpression クラス

  このクラスは，AST の byte 型整数ノードを表現するとともに，コード生成の機能を実現する．

- CaseStatement クラス

  このクラスは，AST の case 文のノードを表現するとともに，コード生成の機能を実現する．

- CastExpression クラス

  このクラスは，AST のキャスト式のノードを表現するとともに，コード生成の機能を実現する．

- CatchStatement クラス

  このクラスは，AST の catch 文のノードを表現するとともに，コード生成の機能を実現する．

- CharExpression クラス

  このクラスは，AST の char 型整数ノードを表現するとともに，コード生成の機能を実現する．

- CheckContext クラス

  このクラスは，ブロック文のために新しくネストしたコンテクストを保持する機能を実現する．

- CodeContext クラス

  このクラスは，ブロック文のために新しくネストしたコンテクストを保持する機能を実現する．

- CommaExpression クラス

  このクラスは，AST の,(カンマ) 演算子のノードを表現するとともに，コード生成の機能を実現する．

- CompoundStatement クラス

  このクラスは，AST の複合文のノードを表現するとともに，コード生成の機能を実現する．

- ConditionVars クラス

  このクラスは，条件の評価値を保持するために用いられる．

- ConditionalExpression クラス

  このクラスは，AST の条件演算式 (?:) のノードを表現するとともに，コード生成の機能を実現する．

- ConstantExpression クラス

  このクラスは，BooleanExpression, DoubleExpression, FloatExpression, IntegerExpression, LongExpression, NullExpression, StringExpression クラスのスーパークラスであり，これらに共通のコード生成の機能を実現する．

- Context クラス

  このクラスは，メソッドのコンテクストを保持する機能を実現する．

- ContinueStatement クラス

  このクラスは，AST の continue 文のノードを表現するとともに，コード生成の機能を実現する．

- ConvertExpression クラス

  このクラスは，型変換を行う式のノードを表現するとともに，コード生成の機能を実現する．

- DeclarationStatement クラス

  このクラスは，AST の宣言文のノードを表現するとともに，コード生成の機能を実現する．

- DivRemExpression クラス

  このクラスは，DivideExpression, RemainderExpression クラスのスーパークラスであり，これらに共通するコード生成の機能を実現する．

- DivideExpression クラス

  このクラスは，AST の / 式のノードを表現するとともに，コード生成の機能を実現する．

- DoStatement クラス

  このクラスは，AST の Do 文のノードを表現するとともに，コード生成の機能を実現する．

- DoubleExpression クラス

  このクラスは，AST の double 型浮動少数点数ノードを表現するとともに，コード生成の機能を実現する．

- EqualExpression クラス

  このクラスは，AST の == 式のノードを表現するとともに，コード生成の機能を実現する．

- ExprExpression クラス

  このクラスは，AST の () で囲まれた式のノードを表現するとともに，コード生成の機能を実現する．

- Expression クラス

  このクラスは，AST のすべての式のノードのスーパークラスであり，式の評価に関係するコード生成の機能を実現する．

- ExpressionStatement クラス

  このクラスは，AST の式で文として機能するもののノードを表現するとともに，コード生成の機能を実現する．

- FieldExpression クラス

  このクラスは，“object.field” のように，フィールドを指定するオペレータのノードを表現するとともに，コード生成の機能を実現する．

- FinallyStatement クラス

このクラスは，AST の Finally 文のノードを表現するとともに，コード生成の機能を実現する．

- FloatExpression クラス

  このクラスは，AST の long 型浮動少数点数ノードを表現するとともに，コード生成の機能を実現する．

- ForStatement クラス

  このクラスは，AST の For 文のノードを表現するとともに，コード生成の機能を実現する．

- GreaterExpression クラス

  このクラスは，AST の > 式のノードを表現するとともに，コード生成の機能を実現する．

- GreaterOrEqualExpression クラス

  このクラスは，AST の >= 式のノードを表現するとともに，コード生成の機能を実現する．

- IdentifierExpression クラス

  このクラスは，AST の識別子のノードを表現するとともに，コード生成の機能を実現する．

- IfStatement クラス

  このクラスは，AST の If 文のノードを表現するとともに，コード生成の機能を実現する．

- IncDecExpression クラス

  このクラスは，PostDecExpression, PostIncExpression, PreDecExpression, PreIncExpression のスーパークラスであり，これらに共通するコード生成の機能を実現する．

- InlineMethodExpression クラス

このクラスは，AST のインライン展開されたメソッド呼び出しの式のノードを表現するとともに，コード生成の機能を実現する．

- InlineNewInstanceExpression クラス

  このクラスは，AST のインライン展開された newInstance の式のノードを表現するとともに，コード生成の機能を実現する．

- InlineReturnStatement クラス

  このクラスは，AST の inline return 文のノードを表現するとともに，コード生成の機能を実現する．

- InstanceOfExpression クラス

  このクラスは，AST の instanceof 演算子のノードを表現するとともに，コード生成の機能を実現する．

- IntExpression クラス

  このクラスは，AST の int 型整数ノードを表現するとともに，コード生成の機能を実現する．

- IntegerExpression クラス

  このクラスは，AST の整数ノードを表現するとともに，コード生成の機能を実現する．

- LengthExpression クラス

  このクラスは，AST の単項演算子 Length の式ノードを表現するとともに，コード生成の機能を実現する．

- LessExpression クラス

  このクラスは，AST の $<$ 式のノードを表現するとともに，コード生成の機能を実現する．

- LessOrEqualExpression クラス

  このクラスは，AST の $<=$ 式のノードを表現するとともに，コード生成の機能を実現する．

- LocalField クラス

  このクラスは，AST のローカル変数フィールドのノードを表現するとともに，コード生成の機能を実現する．

- LongExpression クラス

  このクラスは，AST の long 型整数ノードを表現するとともに，コード生成の機能を実現する．

- MethodExpression クラス

  このクラスは，AST のメソッド呼び出しのノードを表現するとともに，コード生成の機能を実現する．

- MultiplyExpression クラス

  このクラスは，AST の * 式のノードを表現するとともに，コード生成の機能を実現する．

- NaryExpression クラス

  このクラスは，AST の多項演算子の式のノードを表現するとともに，コード生成の機能を実現する．

- NegativeExpression クラス

  このクラスは，AST の単項演算子 – のノードを表現するとともに，コード生成の機能を実現する．

- NewArrayExpression クラス

  このクラスは，AST の newarray 式のノードを表現するとともに，コード生成の機能を実現する．

- NewInstanceExpression クラス

  このクラスは，AST の newInstance 式のノードを表現するとともに，コード生成の機能を実現する．

- Node クラス

このクラスは，AST の全てのノードのスーパークラスであり，全てに共通するコード生成の機能を実現する．

- NotEqualExpression クラス

  このクラスは，AST の != 式のノードを表現するとともに，コード生成の機能を実現する．

- NotExpression クラス

  このクラスは，AST の ! 式のノードを表現するとともに，コード生成の機能を実現する．

- NullExpression クラス

  このクラスは，AST の Null ノードを表現するとともに，コード生成の機能を実現する．

- OrExpression クラス

  このクラスは，AST の OR 演算子のノードを表現するとともに，コード生成の機能を実現する．

- PositiveExpression クラス

  このクラスは，AST の単項演算子 + のノードを表現するとともに，コード生成の機能を実現する．

- PostDecExpression クラス

  このクラスは，AST の後置演算子 – – のノードを表現するとともに，コード生成の機能を実現する．

- PostIncExpression クラス

  このクラスは，AST の後置演算子 ++ のノードを表現するとともに，コード生成の機能を実現する．

- PreDecExpression クラス

  このクラスは，AST の前置演算子 – – のノードを表現するとともに，コード生成の機能を実現する．

- PreIncExpression クラス

  このクラスは，AST の前置演算子 ++ のノードを表現するとともに，コード生成の機能を実現する．

- RemainderExpression クラス

  このクラスは，AST の % 演算式のノードを表現するとともに，コード生成の機能を実現する．

- ReturnStatement クラス

  このクラスは，AST の return 文のノードを表現するとともに，コード生成の機能を実現する．

- ShiftLeftExpression クラス

  このクラスは，AST の << 演算子のノードを表現するとともに，コード生成の機能を実現する．

- ShiftRightExpression クラス

  このクラスは，AST の >> 演算子のノードを表現するとともに，コード生成の機能を実現する．

- ShortExpression クラス

  このクラスは，AST の short 型整数ノードを表現するとともに，コード生成の機能を実現する．

- Statement クラス

  このクラスは，BreakStatement クラス，CaseStatement クラス，CatchStatement クラス，CompoundStatement クラス，ContinueStatement クラス，DeclarationStatement クラス，DoStatement クラス，ExpressionStatement クラス，FinallyStatement クラス，ForStatement クラス，IfStatement クラス，InlineReturnStatement クラス，ReturnStatement クラス，SwitchStatement クラス，SynchronizedStatement クラス，ThrowStatement クラス，TryStatement クラス，VarDeclarationStatement クラス，WhileStatement クラスのスーパークラスであり，これらに共通するコード生成の機能を実現する．

- StringExpression クラス

  このクラスは，AST の文字列ノードを表現するとともに，コード生成の機能を実現する．

- SubtractExpression クラス

  このクラスは，AST の – 式のノードを表現するとともに，コード生成の機能を実現する．

- SuperExpression クラス

  このクラスは，AST の super オブジェクトのノードを表現するとともに，コード生成の機能を実現する．

- SwitchStatement クラス

  このクラスは，AST の switch 文のノードを表現するとともに，コード生成の機能を実現する．

- SynchronizedStatement クラス

  このクラスは，AST の synchronized 文のノードを表現するとともに，コード生成の機能を実現する．

- ThisExpression クラス

  このクラスは，AST の this 式のノードを表現するとともに，コード生成の機能を実現する．

- ThrowStatement クラス

  このクラスは，AST の throw 文のノードを表現するとともに，コード生成の機能を実現する．

- TryStatement クラス

  このクラスは，AST の try 文のノードを表現するとともに，コード生成の機能を実現する．

- TypeExpression クラス

このクラスは，AST の型式のノードを表現するとともに，コード生成の機能を実現する．

- UnaryExpression クラス

  このクラスは，ArrayAccessExpression クラス, BinaryExpression クラス, BitNotExpression クラス, ConvertExpression クラス, ExprExpression クラス, FieldExpression クラス, IncDecExpression クラス, LengthExpression クラス, NaryExpression クラス, NegativeExpression クラス, NotExpression クラス, PositiveExpression クラスのスーパークラスであり，これらのクラスに共通するコード生成の機能を実現する．

- UnsignedShiftRightExpression クラス

  このクラスは，AST の >>> 演算子のノードを表現するとともに，コード生成の機能を実現する．

- VarDeclarationStatement クラス

  このクラスは，AST の変数宣言文のノードを表現するとともに，コード生成の機能を実現する．

- WhileStatement クラス

  このクラスは，AST の While 文のノードを表現するとともに，コード生成の機能を実現する．

(2)　実現機能

これらのクラスファイルで実現される機能は以下の通りである．

- OpenJIT 最適化機能
  - 最適化制御部
  - バイトコード出力部
- OpenJIT フローグラフ構築機能
  - AST 等入力部

  - データフローグラフ構築部
  - コントロール依存グラフ構築部
  - クラス階層解析部

- OpenJIT フローグラフ解析機能
  - データフロー関数登録部
  - フローグラフ解析部
  - 不動点検出部
  - クラス階層解析部

- OpenJIT プログラム変換機能
  - AST 変換ルール登録部
  - AST パターンマッチ部
  - AST 変換部

3.2.6 OpenJIT.frontend.util パッケージ

(1) パッケージの構成

OpenJIT.frontend.util パッケージには以下のクラスが含まれる．

- HashtableEntry クラス

  IntKeyHashtable を実現するためのクラス．具体的には IntKeyHashteble の 1 要素であるキーとオブジェクトの組を格納するリストである．

- HashtableEnumerator クラス

  IntKeyHashtable を実現するためのクラス．具体的には IntKeyHashtable に格納されている HashtableEntry オブジェクトを Enumeration として表現する．

- IntKeyHashtable クラス

  java.util.Hashtable クラスでキーとして int を用いるように改良したクラス．

- LookupHashtable クラス

  java.util.Hashtable を拡張し，lookup メソッドを追加したアブストラクトクラス．

- Queue クラス

  オブジェクト単位のキューバッファを実現する．

- QueueEntry クラス

  Queue クラスで要素として用いられるリスト構造のクラス

(2) 実現機能

これらのクラスファイルで実現される機能は以下の通りである．

- OpenJIT バイトコードディスコンパイラ機能
  - バイトコード解析部
  - コントロールグラフ出力部

  - AST 出力部

- OpenJIT クラスファイルアノテーション解析機能
  - アノテーション解析部
  - アノテーション登録部
  - メタクラス制御部

3.2.7 OpenJIT Java Native Code API パッケージ

(1) パッケージの構成

OpenJIT Java Native Code API パッケージには以下の関数が含まれる．

- OpenJIT_compile

  指定されたメソッドをコンパイルする．

- OpenJIT_invoke

  指定されたメソッドをコンパイルし，コンパイルされたネイティブコードを実行する．

- OpenJIT_InitializeForCompiler

  クラスの初期化時に JIT コンパイラのための初期化を行う．

- OpenJIT_CompilerCompileClass

  指定されたクラスのコンパイルを行う．

- OpenJIT_CompilerFreeClass

  指定したクラスをフリーする．

- OpenJIT_PCinCompiledCode

  pc が指定されたメソッドのネイティブコード内にあるかどうか調べる．

- OpenJIT_CompiledCodePC

  Java のスタックフレームと，メソッドからプログラムカウンタの値を得る．

- OpenJIT_CompilerEnable

  JIT コンパイラの動作を可能にする．

- OpenJIT_CompilerDisable

  JIT コンパイラが動作しないようにする．

- OpenJIT_ReadInCompiledCode

  クラスファイルに含まれたアトリビュートを読む．

- java_lang_Compiler_start

  OpenJIT システムの初期化を行う．

(2) 実現機能

これらのファイルで実現される機能は以下の通りである．

- OpenJIT コンパイラ基盤機能
  - OpenJIT 初期化部
  - OpenJIT コンパイラフロントエンド制御部
  - OpenJIT JNI API 登録部

3.3 機能ブロックとの対応

機能ブロックと各パッケージは以下のように対応する．

機能モジュール	OpenJIT パッケージ	OpenJIT.frontend.asm パッケージ	OpenJIT.frontend.flowgraph パッケージ	OpenJIT.frontend.discompiler パッケージ	OpenJIT.frontend.tree パッケージ	OpenJIT.frontend.util パッケージ	OpenJIT Java Native Code API パッケージ	説明
OpenJIT コンパイラ基盤機能	○						○	OpenJIT 全体の基本動作を司る．
OpenJIT バイトコードディスコンパイラ機能				○	○	○		Java のクラスファイルに含まれるバイトコードを，いわゆる discompiler 技術により，コンパイラ向け中間表現に変換する．
OpenJIT クラスファイルアノテーション解析機能				○	○	○		プログラムグラフ (AST) に対して，コンパイル時に適切な拡張された OpenJIT のメタクラスを起動できるようにする．
OpenJIT 最適化機能		○	○	○	○	○		各種解析モジュールおよびプログラム変換モジュールを用い，プログラム最適化を行う．
OpenJIT フローグラフ構築機能			○		○			AST およびコントロールフローグラフを受け取り，対応するデータ依存グラフ，コントロール依存グラフなどのフローグラフを出力する．

(次のページへ続く)

(前のページからの続き)

機能モジュール	OpenJIT パッケージ	OpenJIT.frontend.asm パッケージ	OpenJIT.frontend.flowgraph パッケージ	OpenJIT.frontend.discompiler パッケージ	OpenJIT.frontend.tree パッケージ	OpenJIT.frontend.util パッケージ	OpenJIT Java Native Code API パッケージ	説明
OpenJIT フローグラフ解析機能		○	○		○			フローグラフ構築モジュールで構築されたプログラム表現のグラフに対し，グラフ上の解析を行う．
OpenJIT プログラム変換機能		○			○			OpenJIT フローグラフ解析機能の結果やユーザーのコンパイラのカスタマイゼーションに従って，プログラム変換を行う．
OpenJIT ネイティブコード変換機能	○						○	OpenJIT バックエンド全体の制御を行う．
OpenJIT 中間コード変換機能	○							バイトコードから中間言語への変換を行う．
OpenJIT RTL 変換機能	○							中間言語から RTL への変換を行う．
OpenJIT Peephole 最適化機能	○							RTL を最適化し，無駄な命令を取り除いて最適化を行う．
OpenJIT レジスタ割付機能	○							仮想レジスタから物理レジスタへの割付を行う．

(次のページへ続く)

(前のページからの続き)

機能モジュール	OpenJIT パッケージ	OpenJIT.frontend.asm パッケージ	OpenJIT.frontend.flowgraph パッケージ	OpenJIT.frontend.discompiler パッケージ	OpenJIT.frontend.tree パッケージ	OpenJIT.frontend.util パッケージ	OpenJIT Java Native Code API パッケージ	説明
OpenJIT SPARC プロセッサコード出力モジュール	○						○	RTL からネイティブコードへの変換を行う．
OpenJIT ランタイムモジュール	○						○	ネイティブコードが実行時に呼び出し，ネイティブコードの実行をサポートする．

○は機能モジュールとパッケージが対応していることを示す．

3.3.1 OpenJIT フロントエンドシステム

(1) OpenJIT コンパイラ基盤機能

OpenJIT コンパイラ基盤機能は，以下のパッケージ，クラスに対応する．

- OpenJIT パッケージ
  - OpenJIT.Compile クラス
- OpenJIT Java Native Code API パッケージ

(a) OpenJIT 初期化部

OpenJIT 初期化部は C 言語で記述された，以下の関数に対応する．

- java_lang_Compiler_start()

(b) OpenJIT コンパイラフロントエンド制御部

OpenJIT コンパイラフロントエンド制御部は以下のメソッドに対応する．

- OpenJIT.Compile.compile()

(c) OpenJIT JNI API 登録部

OpenJIT JNI API 登録部は C 言語で記述された，以下の関数に対応する．

- java_lang_Compiler_start()

(2) OpenJIT バイトコードディスコンパイラ機能

OpenJIT バイトコードディスコンパイラ機能は，以下のパッケージ，クラスに対応する．

- OpenJIT.frontend.discompiler パッケージ
  - ASTFactory インターフェース
  - BBABasickBlock クラス
  - BasicBlock クラス
  - BasicBlockAnalyzer クラス
  - BranchTableEntry クラス
  - BytecodeParser クラス
  - CFAConstants インターフェース
  - CFAFlag クラス
  - CFGNode クラス
  - Case クラス
  - ClassSignature クラス
  - ControlFlowAnalyzer クラス
  - ControlFlowGraph クラス
  - DTNode クラス
  - DTVisitor 抽象クラス
  - DefaultASTFactory クラス
  - Discompiler クラス
  - DiscompilerError クラス
  - DominatorTree クラス
  - ExpressionAnalyzer クラス
  - FlowEdge クラス

  - LoopHead クラス
  - Metaclass 抽象クラス
  - MethodInformation インターフェース
  - NameAndType クラス
  - RuntimeMethodInformation クラス
  - Switch クラス
  - SymbolicConstants インターフェース
  - SymbolicStack クラス
  - SymbolicValue クラス
  - VMInstruction クラス

- OpenJIT.frontend.tree パッケージ

  - OpenJIT.frontend.tree.AddExpression クラス
  - OpenJIT.frontend.tree.AndExpression クラス
  - OpenJIT.frontend.tree.ArrayAccessExpression クラス
  - OpenJIT.frontend.tree.ArrayExpression クラス
  - OpenJIT.frontend.tree.AssignAddExpression クラス
  - OpenJIT.frontend.tree.AssignBitAndExpression クラス
  - OpenJIT.frontend.tree.AssignBitOrExpression クラス
  - OpenJIT.frontend.tree.AssignBitXorExpression クラス
  - OpenJIT.frontend.tree.AssignDivideExpression クラス
  - OpenJIT.frontend.tree.AssignExpression クラス
  - OpenJIT.frontend.tree.AssignMultiplyExpression クラス
  - OpenJIT.frontend.tree.AssignOpExpression クラス
  - OpenJIT.frontend.tree.AssignRemainderExpression クラス
  - OpenJIT.frontend.tree.AssignShiftLeftExpression クラス

- OpenJIT.frontend.tree.AssignShiftRightExpression クラス
- OpenJIT.frontend.tree.AssignSubtractExpression クラス
- OpenJIT.frontend.tree.AssignUnsignedShiftRightExpression クラス
- OpenJIT.frontend.tree.BinaryArithmeticExpression クラス
- OpenJIT.frontend.tree.BinaryAssignExpression クラス
- OpenJIT.frontend.tree.BinaryBitExpression 抽象クラス
- OpenJIT.frontend.tree.BinaryCompareExpression クラス
- OpenJIT.frontend.tree.BinaryEqualityExpression クラス
- OpenJIT.frontend.tree.BinaryExpression クラス
- OpenJIT.frontend.tree.BinaryLogicalExpression 抽象クラス
- OpenJIT.frontend.tree.BinaryShiftExpression クラス
- OpenJIT.frontend.tree.BitAndExpression クラス
- OpenJIT.frontend.tree.BitNotExpression クラス
- OpenJIT.frontend.tree.BitOrExpression クラス
- OpenJIT.frontend.tree.BitXorExpression クラス
- OpenJIT.frontend.tree.BooleanExpression クラス
- OpenJIT.frontend.tree.BreakStatement クラス
- OpenJIT.frontend.tree.ByteExpression クラス
- OpenJIT.frontend.tree.CaseStatement クラス
- OpenJIT.frontend.tree.CastExpression クラス
- OpenJIT.frontend.tree.CatchStatement クラス
- OpenJIT.frontend.tree.CharExpression クラス
- OpenJIT.frontend.tree.CheckContext クラス
- OpenJIT.frontend.tree.CodeContext クラス
- OpenJIT.frontend.tree.CommaExpression クラス
- OpenJIT.frontend.tree.CompoundStatement クラス

- OpenJIT.frontend.tree.ConditionVars クラス
- OpenJIT.frontend.tree.ConditionalExpression クラス
- OpenJIT.frontend.tree.ConstantExpression クラス
- OpenJIT.frontend.tree.Context クラス
- OpenJIT.frontend.tree.ContinueStatement クラス
- OpenJIT.frontend.tree.ConvertExpression クラス
- OpenJIT.frontend.tree.DeclarationStatement クラス
- OpenJIT.frontend.tree.DivRemExpression クラス
- OpenJIT.frontend.tree.DivideExpression クラス
- OpenJIT.frontend.tree.DoStatement クラス
- OpenJIT.frontend.tree.DoubleExpression クラス
- OpenJIT.frontend.tree.EqualExpression クラス
- OpenJIT.frontend.tree.ExprExpression クラス
- OpenJIT.frontend.tree.Expression クラス
- OpenJIT.frontend.tree.ExpressionStatement クラス
- OpenJIT.frontend.tree.FieldExpression クラス
- OpenJIT.frontend.tree.FinallyStatement クラス
- OpenJIT.frontend.tree.FloatExpression クラス
- OpenJIT.frontend.tree.ForStatement クラス
- OpenJIT.frontend.tree.GreaterExpression クラス
- OpenJIT.frontend.tree.GreaterOrEqualExpression クラス
- OpenJIT.frontend.tree.IdentifierExpression クラス
- OpenJIT.frontend.tree.IfStatement クラス
- OpenJIT.frontend.tree.IncDecExpression クラス
- OpenJIT.frontend.tree.InlineMethodExpression クラス
- OpenJIT.frontend.tree.InlineNewInstanceExpression クラス

- OpenJIT.frontend.tree.InlineReturnStatement クラス
- OpenJIT.frontend.tree.InstanceOfExpression クラス
- OpenJIT.frontend.tree.IntExpression クラス
- OpenJIT.frontend.tree.IntegerExpression クラス
- OpenJIT.frontend.tree.LengthExpression クラス
- OpenJIT.frontend.tree.LessExpression クラス
- OpenJIT.frontend.tree.LessOrEqualExpression クラス
- OpenJIT.frontend.tree.LocalField クラス
- OpenJIT.frontend.tree.LongExpression クラス
- OpenJIT.frontend.tree.MethodExpression クラス
- OpenJIT.frontend.tree.MultiplyExpression クラス
- OpenJIT.frontend.tree.NaryExpression クラス
- OpenJIT.frontend.tree.NegativeExpression クラス
- OpenJIT.frontend.tree.NewArrayExpression クラス
- OpenJIT.frontend.tree.NewInstanceExpression クラス
- OpenJIT.frontend.tree.Node クラス
- OpenJIT.frontend.tree.NotEqualExpression クラス
- OpenJIT.frontend.tree.NotExpression クラス
- OpenJIT.frontend.tree.NullExpression クラス
- OpenJIT.frontend.tree.OrExpression クラス
- OpenJIT.frontend.tree.PositiveExpression クラス
- OpenJIT.frontend.tree.PostDecExpression クラス
- OpenJIT.frontend.tree.PostIncExpression クラス
- OpenJIT.frontend.tree.PreDecExpression クラス
- OpenJIT.frontend.tree.PreIncExpression クラス
- OpenJIT.frontend.tree.RemainderExpression クラス

  - OpenJIT.frontend.tree.ReturnStatement クラス
  - OpenJIT.frontend.tree.ShiftLeftExpression クラス
  - OpenJIT.frontend.tree.ShiftRightExpression クラス
  - OpenJIT.frontend.tree.ShortExpression クラス
  - OpenJIT.frontend.tree.Statement クラス
  - OpenJIT.frontend.tree.StringExpression クラス
  - OpenJIT.frontend.tree.SubtractExpression クラス
  - OpenJIT.frontend.tree.SuperExpression クラス
  - OpenJIT.frontend.tree.SwitchStatement クラス
  - OpenJIT.frontend.tree.SynchronizedStatement クラス
  - OpenJIT.frontend.tree.ThisExpression クラス
  - OpenJIT.frontend.tree.ThrowStatement クラス
  - OpenJIT.frontend.tree.TryStatement クラス
  - OpenJIT.frontend.tree.TypeExpression クラス
  - OpenJIT.frontend.tree.UnaryExpression クラス
  - OpenJIT.frontend.tree.UnsignedShiftRightExpression クラス
  - OpenJIT.frontend.tree.VarDeclarationStatement クラス
  - OpenJIT.frontend.tree.WhileStatement クラス

- OpenJIT.frontend.util パッケージ

  - HashtableEntry クラス
  - HashtableEnumerator クラス
  - IntKeyHashtable クラス
  - LookupHashtable 抽象クラス
  - Queue クラス
  - QueueEntry クラス

(a) バイトコード解析部

バイトコード解析部は以下のメソッドに対応する．

- BytecodeParser.BytecodeParser()

(b) コントロールグラフ出力部

コントロールグラフ出力部は以下のメソッドに対応する．

- BytecodeParser.parseFrom()
- ControlFlowGraph.createCFG()
- ControlFlowGraph.newBasicBlock()
- BasicBlockAnalyzer.newBasicBlock()
- ExpressionAnalyzer.newBasicBlock()
- ExpressionAnalyzer.ExpressionAnalyzer()
- DominatorTree.DominatorTree()

(c) AST 出力部

AST 出力部は以下のメソッドに対応する．

- Discompiler.discompile()
- BasicBlockAnalyzer.completeValue()
- ExpressionAnalyzer.recoverExpressions()
- ASTFactory.booleanExpression()
- ASTFactory.byteExpression()
- ASTFactory.charExpression()
- ASTFactory.shortExpression()

- ASTFactory.intExpression()
- ASTFactory.longExpression()
- ASTFactory.floatExpression()
- ASTFactory.doubleExpression()
- ASTFactory.stringExpression()
- ASTFactory.nullExpression()
- ASTFactory.arrayAccessExpression()
- ASTFactory.bitNotExpression()
- ASTFactory.convertExpression()
- ASTFactory.exprExpression()
- ASTFactory.fieldExpression()
- ASTFactory.lengthExpression()
- ASTFactory.negativeExpression()
- ASTFactory.notExpression()
- ASTFactory.positiveExpression()
- ASTFactory.postDecExpression()
- ASTFactory.postIncExpression()
- ASTFactory.preDecExpression()
- ASTFactory.preIncExpression()
- ASTFactory.castExpression()
- ASTFactory.commaExpression()
- ASTFactory.instanceOfExpression()

- ASTFactory.addExpression()
- ASTFactory.divideExpression()
- ASTFactory.multiplyExpression()
- ASTFactory.subtractExpression()
- ASTFactory.remainderExpression()
- ASTFactory.andExpression()
- ASTFactory.orExpression()
- ASTFactory.bitAndExpression()
- ASTFactory.bitOrExpression()
- ASTFactory.bitXorExpression()
- ASTFactory.equalExpression()
- ASTFactory.notEqualExpression()
- ASTFactory.shiftLeftExpression()
- ASTFactory.shiftRightExpression()
- ASTFactory.unsignedShiftRightExpression()
- ASTFactory.greaterExpression()
- ASTFactory.greaterOrEqualExpression()
- ASTFactory.lessExpression()
- ASTFactory.lessOrEqualExpression()
- ASTFactory.assignExpression()
- ASTFactory.assignAddExpression()
- ASTFactory.assignBitAndExpression()

- ASTFactory.assignBitOrExpression()
- ASTFactory.assignBitXorExpression()
- ASTFactory.assignDivideExpression()
- ASTFactory.assignMultiplyExpression()
- ASTFactory.assignRemainderExpression()
- ASTFactory.assignShiftLeftExpression()
- ASTFactory.assignShiftRightExpression()
- ASTFactory.assignSubtractExpression()
- ASTFactory.assignUnsignedShiftRightExpression()
- ASTFactory.conditionalExpression()
- ASTFactory.arrayExpression()
- ASTFactory.methodExpression()
- ASTFactory.newArrayExpression()
- ASTFactory.newInstanceExpression()
- ASTFactory.identifierExpression()
- ASTFactory.superExpression()
- ASTFactory.thisExpression()
- ASTFactory.typeExpression()
- ASTFactory.breakStatement()
- ASTFactory.caseStatement()
- ASTFactory.catchStatement()
- ASTFactory.compoundStatement()

- ASTFactory.continueStatement()
- ASTFactory.declarationStatement()
- ASTFactory.doStatement()
- ASTFactory.expressionStatement()
- ASTFactory.finallyStatement()
- ASTFactory.forStatement()
- ASTFactory.ifStatement()
- ASTFactory.returnStatement()
- ASTFactory.switchStatement()
- ASTFactory.synchronizedStatement()
- ASTFactory.throwStatement()
- ASTFactory.tryStatement()
- ASTFactory.varDeclarationStatement()
- ASTFactory.whileStatement()

(3) OpenJIT クラスファイルアノテーション解析機能

OpenJIT クラスファイルアノテーション解析機能は，以下のパッケージ，クラスに対応する．

- OpenJIT.frontend.discompiler パッケージ
  - Annotation クラス
  - AnnotationAnalyzer クラス
  - Metaclass クラス
  - MethodInformation インターフェース
  - RuntimeMethodInformation クラス
- OpenJIT.frontend.tree パッケージ
  - ArrayAccessExpression クラス
  - ArrayExpression クラス
  - AssignExpression クラス
  - AssignOpExpression クラス
  - BinaryAssignExpression クラス
  - BinaryExpression クラス
  - BinaryLogicalExpression 抽象クラス
  - BreakStatement クラス
  - CaseStatement クラス
  - CastExpression クラス
  - CatchStatement クラス
  - CommaExpression クラス
  - CompoundStatement クラス
  - ConditionalExpression クラス
  - ContinueStatement クラス

- ConvertExpression クラス
- DeclarationStatement クラス
- DoStatement クラス
- Expression クラス
- ExpressionStatement クラス
- FieldExpression クラス
- FinallyStatement クラス
- ForStatement クラス
- IdentifierExpression クラス
- IfStatement クラス
- IncDecExpression クラス
- InstanceOfExpression クラス
- LengthExpression クラス
- MethodExpression クラス
- NewArrayExpression クラス
- NewInstanceExpression クラス
- ReturnStatement クラス
- Statement クラス
- SuperExpression クラス
- SwitchStatement クラス
- SynchronizedStatement クラス
- ThisExpression クラス
- ThrowStatement クラス
- TryStatement クラス
- TypeExpression クラス
- UnaryExpression クラス

  - VarDeclarationStatement クラス
  - WhileStatement クラス
- OpenJIT.frontend.util パッケージ
  - LookupHashtable 抽象クラス

(a)　アノテーション解析部

アノテーション解析部は以下のメソッドに対応する．

- AnnotationAnalyzer.readAnnotation()

(b)　アノテーション登録部

アノテーション登録部は以下のメソッドに対応する．

- AnnotationAnalyzer.registerAnnotation()

(c)　メタクラス制御部

メタクラス制御部は以下のメソッドに対応する．

- AnnotationAnalyzer.addMetaobject()
- BinaryAssignExpression.check()
- BreakStatement.check()
- CaseStatement.check()
- CatchStatement.check()
- CommaExpression.check()
- CompoundStatement.check()
- ConditionalExpression.check()
- ContinueStatement.check()

- DeclarationStatement.check()
- DoStatement.check()
- Expression.check()
- ExpressionStatement.check()
- FinallyStatement.check()
- ForStatement.check()
- IfStatement.check()
- IncDecExpression.check()
- MethodExpression.check()
- NewInstanceExpression.check()
- ReturnStatement.check()
- Statement.check()
- SwitchStatement.check()
- SynchronizedStatement.check()
- ThrowStatement.check()
- TryStatement.check()
- WhileStatement.check()
- ArrayAccessExpression.checkValue()
- ArrayExpression.checkValue()
- AssignExpression.checkValue()
- AssignOpExpression.checkValue()
- BinaryExpression.checkValue()

- BinaryLogicalExpression.checkValue()
- CastExpression.checkValue()
- ConditionalExpression.checkValue()
- ConvertExpression.checkValue()
- Expression.checkValue()
- FieldExpression.checkValue()
- IdentifierExpression.checkValue()
- IncDecExpression.checkValue()
- InstanceOfExpression.checkValue()
- LengthExpression.checkValue()
- MethodExpression.checkValue()
- NewArrayExpression.checkValue()
- NewInstanceExpression.checkValue()
- SuperExpression.checkValue()
- ThisExpression.checkValue()
- TypeExpression.checkValue()
- UnaryExpression.checkValue()
- ArrayAccessExpression.checkAssignOp()
- Expression.checkAssignOp()
- FieldExpression.checkAssignOp()
- IdentifierExpression.checkAssignOp()
- ArrayAccessExpression.checkLHS()

- Expression.checkLHS()
- FieldExpression.checkLHS()
- IdentifierExpression.checkLHS()
- ArrayAccessExpression.checkInitializer()
- Expression.checkInitializer()

(4) OpenJIT 最適化機能

OpenJIT 最適化機能は，以下のパッケージ，クラスに対応する．

- OpenJIT.frontend.asm パッケージ
  - Assembler クラス
- OpenJIT.frontend.flowgraph パッケージ
  - ASTTransformer クラス
  - Optimizer クラス

(a) 最適化制御部

最適化制御部は以下のメソッドに対応する．

- Assembler.optimize()
- Optimizer.optimize()

(b) バイトコード出力部

バイトコード出力部は以下のメソッドに対応する．

- Optimizer.generateBytecode()
- AddExpression.code()
- ArrayAccessExpression.codeValue()
- ArrayExpression.codeValue()
- AssignExpression.code()
- AssignExpression.codeValue()
- AssignOpExpression.code()
- AssignOpExpression.codeValue()

- BinaryBitExpression.codeValue()
- BinaryExpression.codeValue()
- BitNotExpression.codeValue()
- BooleanExpression.codeValue()
- BreakStatement.code()
- CatchStatement.code()
- CommaExpression.code()
- CompoundStatement.code()
- ConditionalExpression.codeValue()
- ContinueStatement.code()
- ConvertExpression.codeValue()
- DeclarationStatement.code()
- DoStatement.code()
- DoubleExpression.codeValue()
- Expression.code()
- Expression.codeValue()
- ExpressionStatement.code()
- FieldExpression.codeValue()
- FinallyStatement.code()
- FloatExpression.codeValue()
- ForStatement.code()
- IdentifierExpression.codeValue()

- IfStatement.code()
- InlineMethodExpression.code()
- InlineMethodExpression.codeValue()
- InlineNewInstanceExpression.code()
- InlineNewInstanceExpression.codeValue()
- InlineReturnStatement.code()
- InstanceOfExpresion.code()
- InstanceOfExpression.codeValue()
- IntegerExpression.codeValue()
- LengthExpression.codeValue()
- LongExpression.codeValue()
- MethodExpression.codeValue()
- NegativeExpression.codeValue()
- NewArrayExpression.codeValue()
- NewInstanceExpression.code()
- NewInstanceExpression.codeValue()
- NullExpression.codeValue()
- PostDecExpression.code()
- PostDecExpression.codeValue()
- PostIncExpression.code()
- PostIncExpression.codeValue()
- PreDecExpression.code()

- PreDecExpression.codeValue()
- PreIncExpression.code()
- PreIncExpression.codeValue()
- ReturnStatement.code()
- Statement.code()
- StringExpression.codeValue()
- SuperExpression.codeValue()
- SwitchStatement.code()
- SynchronizedStatement.code()
- ThisExpression.codeValue()
- ThrowStatement.code()
- TryStatement.code()
- VarDeclarationStatement()
- WhileStatement.code()

(5) OpenJIT フローグラフ構築機能

OpenJIT フローグラフ構築機能は，以下のパッケージ，クラスに対応する．

- OpenJIT.frontend.flowgraph パッケージ
  - FlowGraph クラス
  - ControlDependencyGraph クラス
  - DataFlowGraph クラス
  - ClassHierarchyGraph クラス
- OpenJIT.frontend.tree パッケージ
  - Expression クラス
  - Statement クラス

(a) AST 等入力部

AST 等入力部は以下のメソッドに対応する．

- FlowGraph.setAST()
- FlowGraph.setCFG()
- FlowGraph.setCHInfo()

(b) データフローグラフ構築部

データフローグラフ構築部は以下のメソッドに対応する．

- FlowGraph.constructDFG()
- DataFlowGraph.constructGraph()
- DataFlowGraph.seekStatement()
- DataFlowGraph.seekExpression()
- DataFlowGraph.initDataFlow()

(c)　コントロール依存グラフ構築部

コントロール依存グラフ構築部は以下のメソッドに対応する．

- FlowGraph.constructCFG()
- ControlDependencyGraph.constructGraph()
- ControlDependencyGraph.seekStatement()

(d)　クラス階層解析部

クラス階層解析部は以下のメソッドに対応する．

- FlowGraph.analysisCH()
- ClassHierarchyGraph.analysis()

(6) OpenJIT フローグラフ解析機能

OpenJIT フローグラフ解析機能は，以下のパッケージ，クラスに対応する．

- OpenJIT.frontend.flowgraph パッケージ
  - FlowGraph クラス
  - FlowGraphAnalysis クラス
  - DFFunctionRegister クラス
  - FlowGraphAnalyzer インターフェース
  - ReachingAnalyzer クラス
  - AvailableAnalyzer クラス
  - LivenessAnalyzer クラス
  - FixedPointDetector クラス
  - ClassHierarchyAnalyzer クラス
- OpenJIT.frontend.tree パッケージ
  - OpenJIT.frontend.tree.Expression クラス
  - OpenJIT.frontend.tree.Statement クラス

(a) データフロー関数登録部

データフロー関数登録部は以下のメソッドに対応する．

- DFFunctionRegister.register()

(b) フローグラフ解析部

フローグラフ解析部は以下のメソッドに対応する．

- FlowGraphAnalysis.analysis()
- ReachingAnalyzer.analysis()
- AvailableAnalyzer.analysis()
- LivenessAnalyzer.analysis()

(c)　不動点検出部

不動点検出部は以下のメソッドに対応する．

- FixedPointDetector.analysis()

(d)　クラス階層解析部

クラス階層解析部は以下のメソッドに対応する．

- ClassHierarchyAnalyzer.analysis()

(7) OpenJIT プログラム変換機能

OpenJIT プログラム変換機能は，以下のパッケージ，クラスに対応する．

- OpenJIT.frontend.flowgraph パッケージ
  - ASTTransformer クラス
  - FlowGraph クラス
  - FlowGraphAnalysis クラス
- OpenJIT.frontend.tree パッケージ
  - OpenJIT.frontend.tree.Expression クラス
  - OpenJIT.frontend.tree.Statement クラス

(a) AST 変換ルール登録部

AST 変換ルール登録部は以下のメソッドに対応する．

- ASTTransformer.registerRule()

(b) AST パターンマッチ部

AST パターンマッチ部は以下のメソッドに対応する．

- ASTTransformer.match()

(c) AST 変換部

AST 変換部は以下のメソッドに対応する．

- ASTTransformer.transform()

3.3.2 OpenJIT バックエンドシステム

(1) OpenJIT ネイティブコード変換機能

OpenJIT ネイティブコード変換機能は、OpenJIT Java Native Code API パッケージおよび OpenJIT パッケージの以下のクラスファイルに対応する．

- OpenJIT パッケージ
  - OpenJIT.CompilerError クラス
  - OpenJIT.Compile クラス
  - OpenJIT.Constants クラス
  - OpenJIT.ConvertRTL クラス
  - OpenJIT.ExceptionHandler クラス
  - OpenJIT.OptimizeRTL クラス
  - OpenJIT.ParseBytecode クラス
  - OpenJIT.Runtime クラス
  - OpenJIT.Select クラス
  - OpenJIT.Sparc クラス

(a) ネイティブコード変換

ネイティブコード変換部は以下のメソッドに対応する．

- OpenJIT.Compile.compile()

(b) メソッド情報

メソッド情報部は OpenJIT Java Native Code API パッケージの以下の関数に対応する．

- OpenJIT_compile()

(c) バイトコードアクセス

バイトコードアクセス部は以下のメソッドに対応する．

- OpenJIT.Compile.pc2uchar()
- OpenJIT.Compile.pc2signedchar()
- OpenJIT.Compile.pc2signedshort()
- OpenJIT.Compile.pc2signedlong()
- OpenJIT.Compile.pc2ushort()

(d) 生成コードメモリ管理

生成コードメモリ管理部は以下のメソッドに対応する．

- OpenJIT.Compile.NativeCodeAlloc()
- OpenJIT.Compile.NativeCodeReAlloc()
- OpenJIT.Compile.setNativeCode()
- OpenJIT.Compile.getNativeCode()

(2) OpenJIT 中間コード変換機能

OpenJIT 中間コード変換機能は、OpenJIT パッケージの以下のクラスファイルに対応する．

- OpenJIT パッケージ
  - OpenJIT.BBinfo クラス
  - OpenJIT.BCinfo クラス
  - OpenJIT.CompilerError クラス
  - OpenJIT.Compile クラス
  - OpenJIT.Constants クラス
  - OpenJIT.ExceptionHandler クラス
  - OpenJIT.ILnode クラス
  - OpenJIT.LinkedList クラス
  - OpenJIT.ParseBytecode クラス

(a) 中間言語変換

中間言語変換部は以下のメソッドに対応する．

- OpenJIT.ParseBytecode.parseBytecode()

(b) メソッド引数展開

メソッド引数展開部は以下のメソッドに対応する．

- OpenJIT.ParseBytecode.callMethod()

(c) 命令パターンマッチング

命令パターンマッチング部は以下のメソッドに対応する．

- OpenJIT.ParseBytecode.optim_neg()
- OpenJIT.ParseBytecode.optim_lcmp()
- OpenJIT.ParseBytecode.optime_fcmp()

(3) OpenJIT RTL 変換機能

OpenJIT RTL 変換機能は、OpenJIT パッケージの以下のクラスファイルに対応する．

- OpenJIT パッケージ
  - OpenJIT.BBinfo クラス
  - OpenJIT.BCinfo クラス
  - OpenJIT.CompilerError クラス
  - OpenJIT.Constants クラス
  - OpenJIT.ConvertRTL クラス
  - OpenJIT.ExceptionHandler クラス
  - OpenJIT.ILnode クラス
  - OpenJIT.LinkedList クラス

(a) 基本ブロック分割

基本ブロック分割部は以下のメソッドに対応する．

- OpenJIT.ConvertRTL.extractBB()

(b) コントロールフロー解析

コントロールフロー解析部は以下のメソッドに対応する．

- OpenJIT.ConvertRTL.convertRTL()

(4) OpenJIT Peephole 最適化機能

OpenJIT Peephole 最適化機能は、OpenJIT パッケージの以下のクラスファイルに対応する．

- OpenJIT パッケージ
  - OpenJIT.BBinfo クラス
  - OpenJIT.BCinfo クラス
  - OpenJIT.CompilerError クラス
  - OpenJIT.Constants クラス
  - OpenJIT.ILnode クラス
  - OpenJIT.LinkedList クラス
  - OpenJIT.OptimizeRTL クラス
  - OpenJIT.Var クラス

(a) データフロー解析

データフロー解析部は以下のメソッドに対応する．

- OpenJIT.OptimizeRTL.optimizeRTL()

(b) 各種 Peephole 最適化

各種 Peephole 最適化部は以下のメソッドに対応する．

- OpenJIT.OptimizeRTL.eliminateIL()
- OpenJIT.OptimizeRTL.moveUpSV()
- OpenJIT.OptimizeRTL.moveUpSP()

(5) OpenJIT レジスタ割付機能

OpenJIT レジスタ割付機能は、 OpenJIT パッケージの以下のクラスファイルに対応する．

- OpenJIT パッケージ
  - OpenJIT.CompilerError クラス
  - OpenJIT.Constants クラス
  - OpenJIT.RegAlloc$PhyRegs クラス
  - OpenJIT.RegAlloc$VirReg クラス
  - OpenJIT.RegAlloc クラス
  - OpenJIT.Sparc クラス

(a) 仮想レジスタ管理

仮想レジスタ管理部は以下のメソッドに対応する．

- OpenJIT.RegAlloc$VirReg.assign()
- OpenJIT.RegAlloc$VirReg.fill()
- OpenJIT.RegAlloc$VirReg.spill()
- OpenJIT.RegAlloc$VirReg.nouse()
- OpenJIT.RegAlloc$VirReg.invalidate()

(b) 物理レジスタ管理

物理レジスタ管理部は以下のメソッドに対応する．

- OpenJIT.RegAlloc$PhyRegs.fixReg()
- OpenJIT.RegAlloc$PhyRegs.getFixReg()
- OpenJIT.RegAlloc$PhyRegs.unlock()

- OpenJIT.RegAlloc$PhyRegs.save()
- OpenJIT.RegAlloc$PhyRegs.invalidateI()
- OpenJIT.RegAlloc$PhyRegs.invalidateF()
- OpenJIT.RegAlloc$PhyRegs.read()
- OpenJIT.RegAlloc$PhyRegs.write()
- OpenJIT.RegAlloc$PhyRegs.searchVector()
- OpenJIT.RegAlloc$PhyRegs.getTmpReg()
- OpenJIT.RegAlloc$PhyRegs.freeTmpReg()
- OpenJIT.RegAlloc$PhyRegs.isTmpReg()

(c) レジスタ割付

レジスタ割付部は以下のメソッドに対応する．

- OpenJIT.RegAlloc.regAlloc()

(6) OpenJIT SPARC プロセッサコード出力モジュール *

OpenJIT SPARC プロセッサコード出力モジュールは、OpenJIT パッケージの以下のクラスファイルに対応する．

このモジュールは契約の対象外である．

- OpenJIT パッケージ
  - OpenJIT.BCinfo クラス
  - OpenJIT.CompilerError クラス
  - OpenJIT.Compile クラス
  - OpenJIT.Constants クラス
  - OpenJIT.ExceptionHandler クラス
  - OpenJIT.ILnode クラス
  - OpenJIT.LinkedList クラス
  - OpenJIT.OptimizeRTL クラス
  - OpenJIT.RegAlloc クラス
  - OpenJIT.Runtime クラス
  - OpenJIT.Sparc クラス

(7) OpenJIT ランタイムモジュール*

OpenJIT ランタイムモジュールは、OpenJIT Java Native Code API パッケージおよび OpenJIT パッケージの以下のクラスファイルに対応する．

このモジュールは契約の対象外である．

- OpenJIT パッケージ
  - OpenJIT.Runtime クラス

3.4　サブプログラム間の関連（インタフェース）

3.4.1　OpenJIT フロントエンドシステム

(1)　OpenJIT コンパイラ基盤機能

OpenJIT コンパイラ基盤機能を構成するサブプログラムを図 3.2に示す．これらサブプログラム間の関係と，それらを実現するメソッドとの対応は次の通りである．

(1-C1) java_lang_Compiler_start 関数が OpenJIT.Compiler クラスの Compiler メソッドを呼び出す．

(1-C2) OpenJIT.Compiler クラスの Compiler メソッドが OpenJIT.frontend.discompiler.Discomp クラスの Discompiler メソッドを呼び出す．

(1-C3) java_lang_Compiler_start 関数が OpenJIT_InitializeForCompiler 関数を呼び出す．

(1-C4) java_lang_Compiler_start 関数が OpenJIT.Compiler クラスの Compiler メソッドを呼び出す．

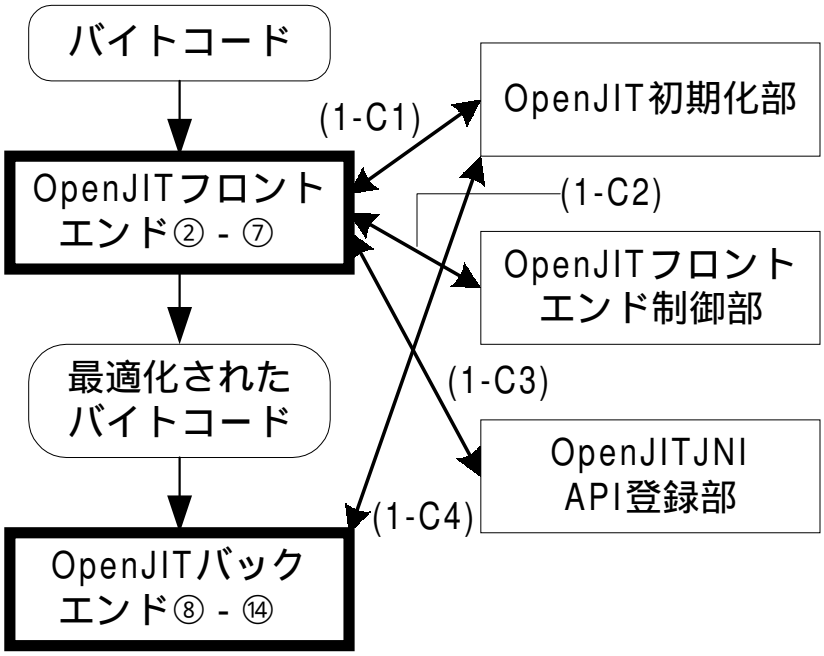

図 3.2: OpenJIT コンパイラ基盤機能

(2) OpenJIT バイトコードディスコンパイラ機能

OpenJIT バイトコードディスコンパイラ機能を構成するサブプログラムを図 3.3に示す．これらサブプログラム間の関係と，それらを実現するメソッドとの対応は次の通りである．

(2-C1) OpenJIT.frontend.discompiler.Discompiler クラスの Discompiler メソッドが，ControlFlowGraph の ControlFlowGraph メソッドを呼び出す．

(2-C2) OpenJIT.frontend.discompiler.Discompiler クラスの Discompiler メソッドが BytecodeParser クラスの BytecodeParser メソッドを呼び出す．

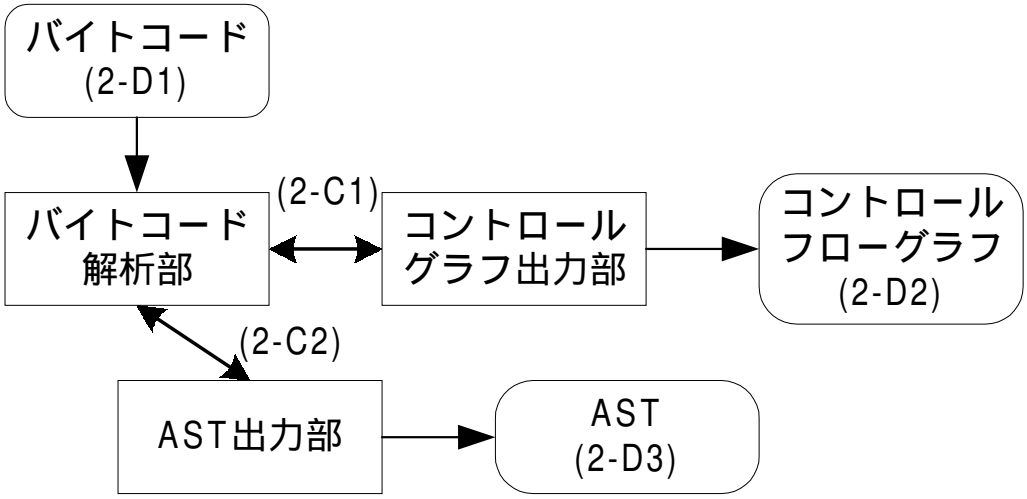

図 3.3: OpenJIT バイトコードディスコンパイラ機能

(3) OpenJIT クラスファイルアノテーション解析機能

OpenJIT クラスファイルアノテーション解析機能を構成するサブプログラムを図3.4に示す．これらサブプログラム間の関係と，それらを実現するメソッドとの対応は次の通りである．

(3-C1) AnnotationAnalyzer クラスの analyze メソッドが AnnotationAnalyzer クラスの addMetaobject メソッドを呼び出す．

(3-C2) AnnotationAnalyzer クラスの analyze メソッドが AnnotationAnalyzer クラスの registerAnnotation メソッドを呼び出す．

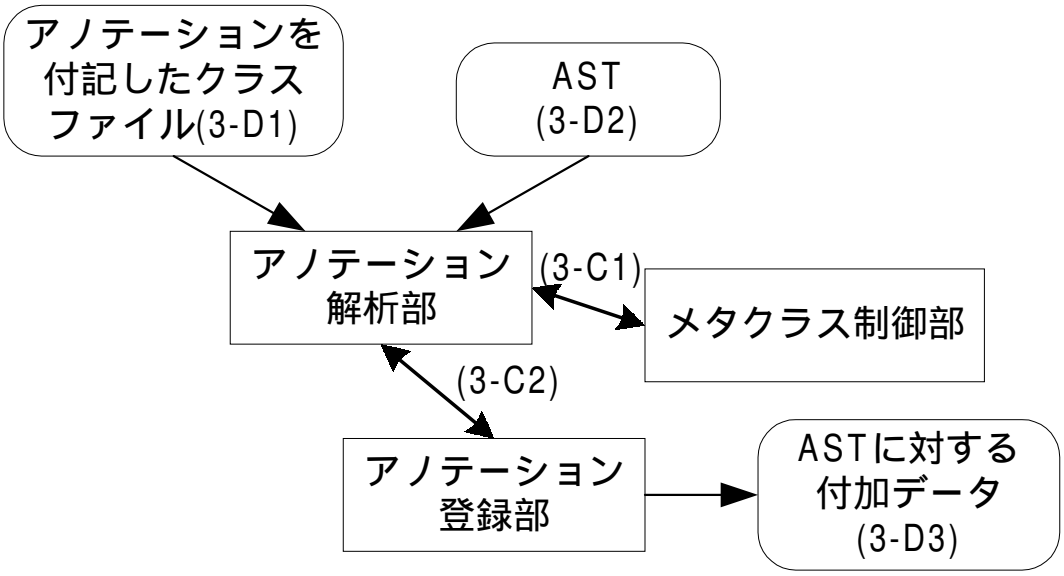

図 3.4: OpenJIT クラスファイルアノテーション解析機能

(4) OpenJIT 最適化機能

OpenJIT 最適化機能を構成するサブプログラムを図 3.5に示す．これらサブプログラム間の関係と，それらを実現するメソッドとの対応は次の通りである．

(4-C1) Optimizer クラスの optimize メソッドが FlowGraph クラスの FlowGraph メソッドを呼び出す．

(4-C2) Optimizer クラスの optimize メソッドが FlowGraphAnalysis クラスの FlowGraphAnalysis メソッドを呼び出す．

(4-C3) Optimizer クラスの optimize メソッドが ASTTransformer クラスの transform メソッドを呼び出す．

(4-C4) Optimizer クラスの optimize メソッドが Statement クラスの code メソッドを呼び出す．

(4-C5) ASTTransformer クラスの transform メソッドが Statement クラスの check メソッドを呼び出す．

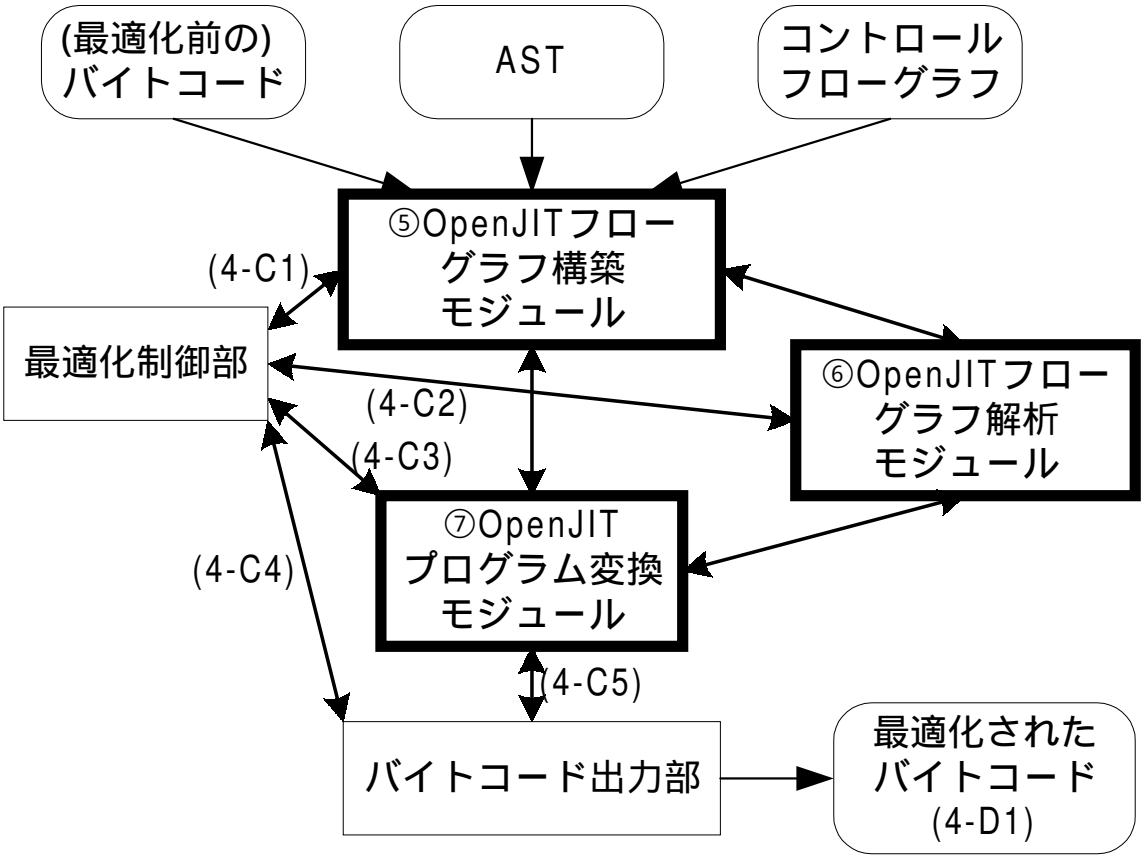

図 3.5: OpenJIT 最適化機能

(5) OpenJIT フローグラフ構築機能

OpenJIT フローグラフ構築機能を構成するサブプログラムを図3.6に示す．これらサブプログラム間の関係と，それらを実現するメソッドとの対応は次の通りである．

(5-C1) OpenJIT.frontend.flowgraph.FlowGraph クラスの constructDFG() メソッドが OpenJIT.frontend.flowgraph.DataFlowGraph クラスの constructGraph() メソッドを呼び出す．

(5-C2) OpenJIT.frontend.flowgraph.FlowGraph クラスの constructCDG() メソッドが OpenJIT.frontend.flowgraph.ControlDependencyGraph クラスの constructGraph() メソッドを呼び出す．

(5-C3) OpenJIT.frontend.flowgraph.FlowGraph クラスの constructCH() メソッドが OpenJIT.frontend.flowgraph.ClassHierarchyGraph クラスの constructGraph() メソッドを呼び出す．

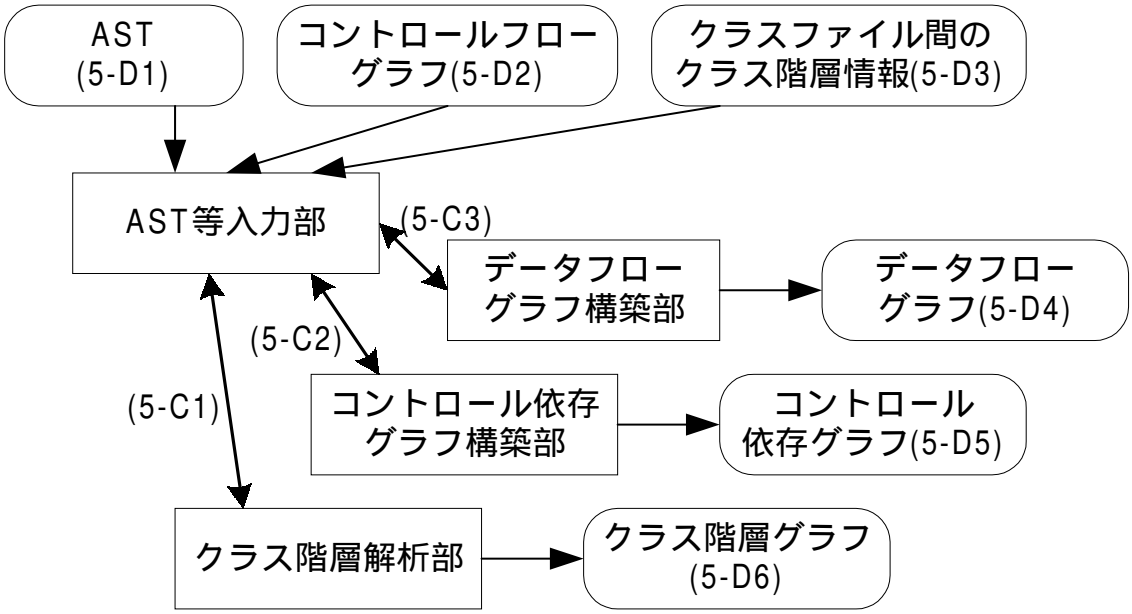

図 3.6: OpenJIT フローグラフ構築機能

(6) OpenJIT フローグラフ解析機能

OpenJIT フローグラフ解析機能を構成するサブプログラムを図3.7に示す．これらサブプログラム間の関係と，それらを実現するメソッドとの対応は次の通りである．

(6-C1) OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラスの analysis() メソッドが OpenJIT.frontend.flowgraph.ClassHierarchyAnalyzer クラスの analysis() メソッドを呼び出す．

(6-C2) OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラスのコンストラクタが OpenJIT.frontend.flowgraph.DFFunctionRegister クラスの register() メソッドを呼び出す．

(6-C3) OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラスの analysis() メソッドが OpenJIT.frontend.flowgraph.FixedPointDetector クラスの analysis() メソッドを呼び出す．

(6-C4) OpenJIT.frontend.flowgraph.ClassHierarchyAnalyzer クラスの analysis() メソッドが OpenJIT.frontend.flowgraph.FixedPointDetector クラスの analysis() メソッドを呼び出す．

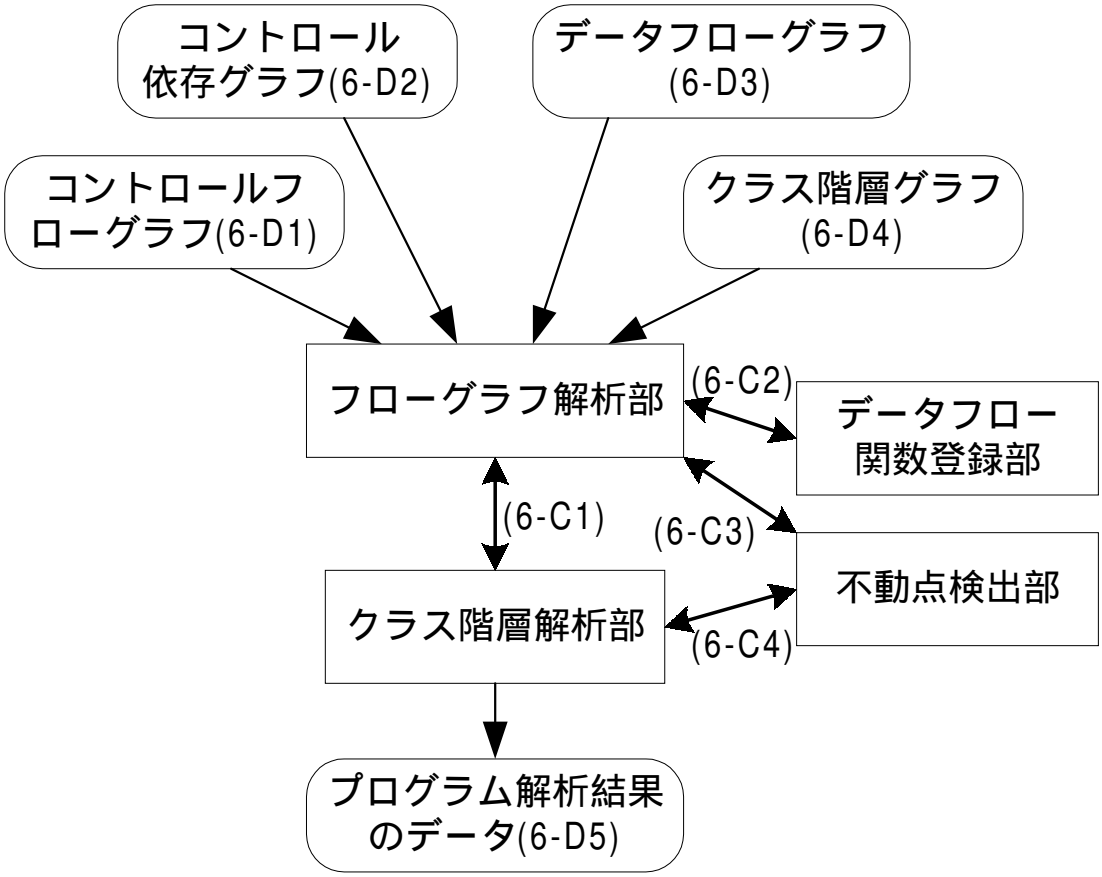

図 3.7: OpenJIT フローグラフ解析機能

(7) OpenJIT プログラム変換機能

OpenJIT プログラム変換機能を構成するサブプログラムを図3.8に示す．これらサブプログラム間の関係と，それらを実現するメソッドとの対応は次の通りである．

(7-C1) OpenJIT.frontend.flowgraph.ASTTransformer クラスの match() メソッドが OpenJIT.frontend.flowgraph.ASTTransformer クラスの register() メソッドを呼び出す．

(7-C2) OpenJIT.frontend.flowgraph.ASTTransformer クラスの match() メソッドが OpenJIT.frontend.flowgraph.ASTTransformer クラスの transform() メソッドを呼び出す．

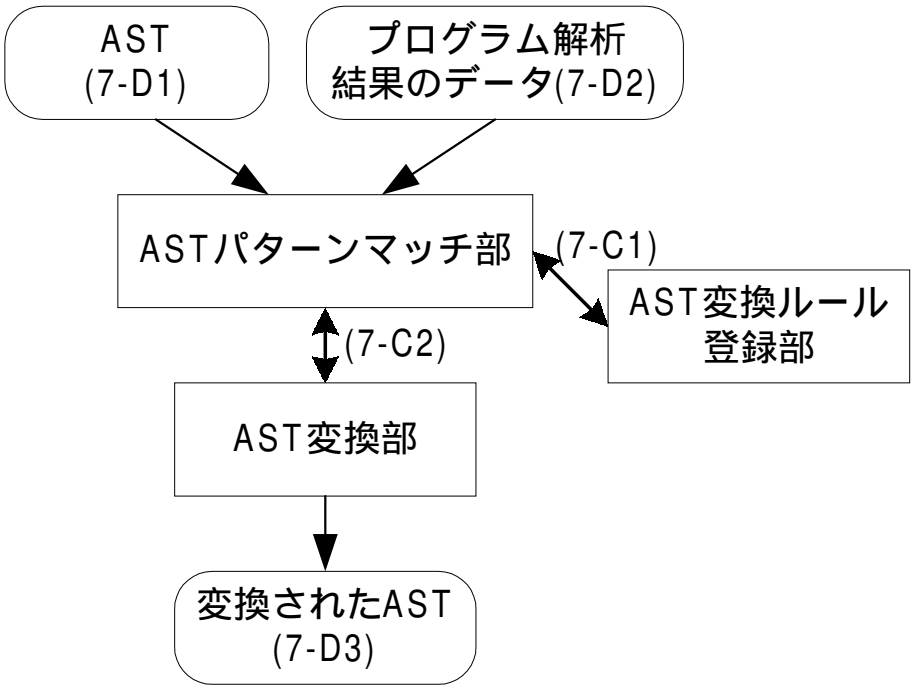

図 3.8: OpenJIT プログラム変換機能

3.4.2 OpenJIT バックエンドシステム

(1) OpenJIT ネイティブコード変換機能

OpenJIT ネイティブコード変換機能を構成するサブプログラムを図 3.9に示す。これらサブプログラム間の関係と、それらを実現するメソッドとの対応は次の通りである。

(8-C1) OpenJIT.Compile クラスの compile メソッドから，OpenJIT.ParseBytecode クラスの parseBytecode メソッドを呼び出す．

(8-C2) OpenJIT.Compile クラスの compile メソッドから，OpenJIT.ConvertRTL クラスの convertRTL メソッドを呼び出す．

(8-C3) OpenJIT.Compile クラスの compile メソッドから，OpenJIT.OptimizeRTL クラスの optimizeRTL メソッドを呼び出す．

(8-C4) OpenJIT Java Native Code API パッケージの OpenJIT_compile 関数から，OpenJIT.Compile クラスの compile を呼び出す．

(8-C5) OpenJIT.ParseBytecode クラスの parseBytecode メソッドから，OpenJIT.Compile クラスの以下のメソッドを呼び出す．

- pc2uchar メソッド
- pc2signedchar メソッド
- pc2signedshort メソッド
- pc2signedlong メソッド
- pc2ushort メソッド

を呼び出す．

(8-C6) OpenJIT.Sparc クラスのメソッドから，OpenJIT.Compile クラスの以下のメソッドを呼び出す．

- NativeCodeAlloc メソッド
- NativeCodeReAlloc メソッド

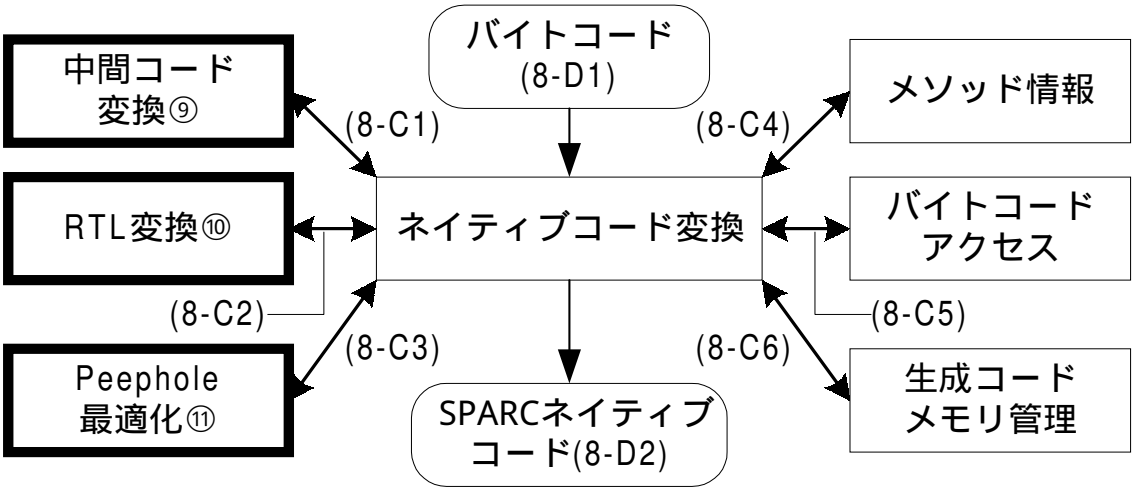

図 3.9: OpenJIT ネイティブコード変換機能

(2) OpenJIT 中間コード変換機能

OpenJIT 中間コード変換機能を構成するサブプログラムを図 3.10に示す。これらサブプログラム間の関係と、それらを実現するメソッドとの対応は次の通りである。

(9-C1) OpenJIT.ParseBytecode クラスの parseBytecode メソッドから，

OpenJIT.ParseBytecode クラスの callMethod メソッドを呼び出す．

(9-C2) OpenJIT.ParseBytecode クラスの parseBytecode メソッドから，

OpenJIT.ParseBytecode クラスの以下のメソッドを呼び出す．

- optim_neg
- optim_lcmp
- optime_fcmp

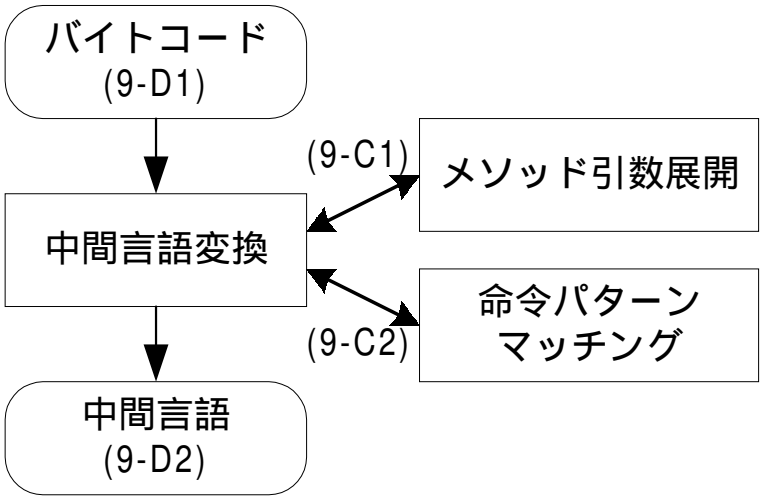

図 3.10: OpenJIT 中間コード変換機能

(3) OpenJIT RTL 変換機能

OpenJIT RTL 変換機能を構成するサブプログラムを図 3.11に示す。これらサブプログラム間の関係と、それらを実現するメソッドとの対応はない。

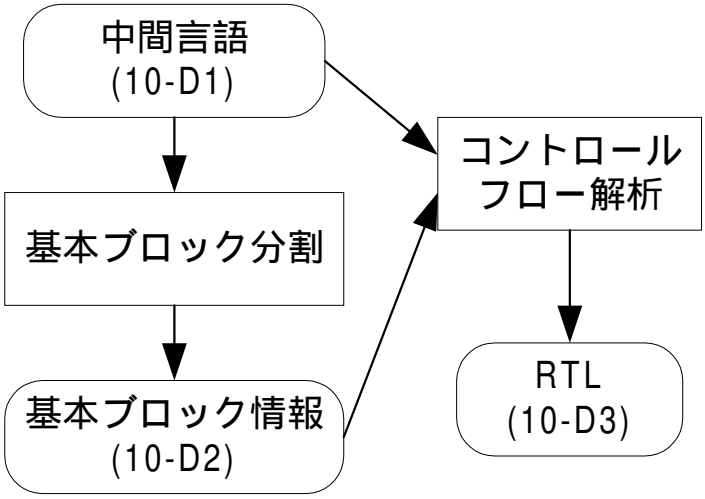

図 3.11: OpenJIT RTL 変換機能

(4) OpenJIT Peephole 最適化機能

OpenJIT Peephole 最適化機能を構成するサブプログラムを図 3.12に示す。これらサブプログラム間の関係と、それらを実現するメソッドとの対応は次の通りである。

(11-C1) OpenJIT.OptimizeRTL クラスの optimizeRTL メソッドから，

OpenJIT.OptimizeRTL クラスの以下のメソッドを呼び出す．

- eliminateIL
- moveUpSV
- moveUpSP

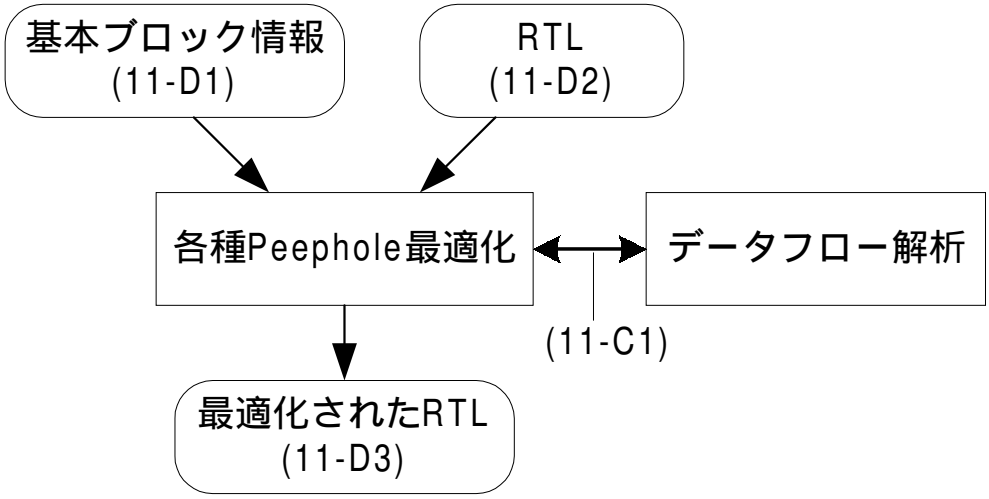

図 3.12: OpenJIT Peephole 最適化機能

(5) OpenJIT レジスタ割付機能

OpenJIT レジスタ割付機能を構成するサブプログラムを図 3.13に示す。これらサブプログラム間の関係と、それらを実現するメソッドとの対応は次の通りである。

(12-C1) OpenJIT.RegAlloc クラスの regAlloc メソッドから，OpenJIT.RegAlloc$VirReg クラスの VirReg メソッド (コンストラクタ) を呼び出す．

(12-C2) OpenJIT.RegAlloc クラスの regAlloc メソッドから，OpenJIT.RegAlloc$PhyRegs クラスの PhyReg メソッド (コンストラクタ) を呼び出す．

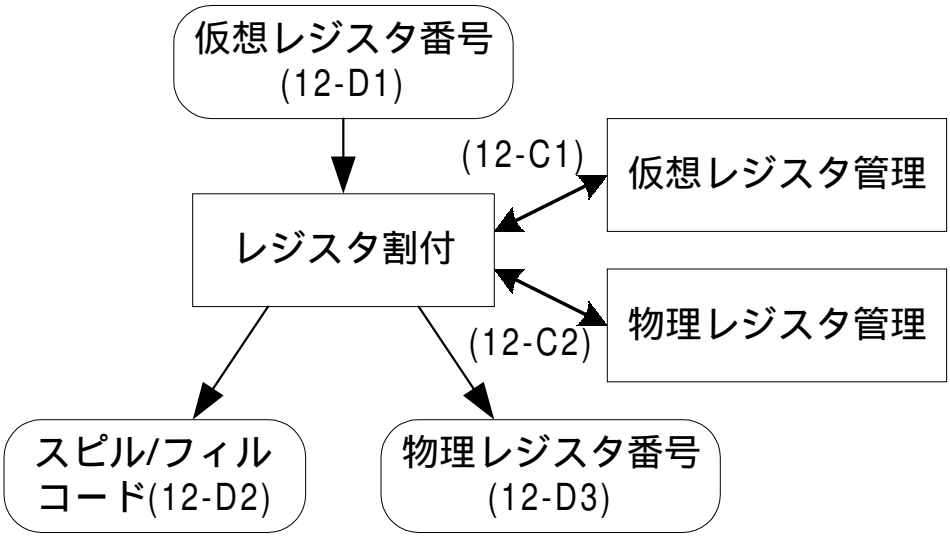

図 3.13: OpenJIT レジスタ割付機能

3.5 全体処理方式

本システムは，JDK 1.1.4 以降の JIT コンパイラに対する標準 API を満たすように作成される．従って，既存の JDK 1.1.4 以降と組み合わせて，JIT コンパイラとして機能する．

第 4 章

プログラム仕様

4.1 OpenJIT パッケージ

- インターフェース
  - OpenJIT.Constants インターフェース
- 抽象クラス
  - OpenJIT.Compile クラス
  - OpenJIT.ConvertRTL クラス
  - OpenJIT.ParseBytecode クラス
  - OpenJIT.RegAlloc クラス
- クラス
  - OpenJIT.BBinfo クラス
  - OpenJIT.BCinfo クラス
  - OpenJIT.Compile クラス
  - OpenJIT.CompilerError クラス
  - OpenJIT.Constants クラス
  - OpenJIT.ConvertRTL クラス

- OpenJIT.ExceptionHandler クラス
- OpenJIT.ILnode クラス
- OpenJIT.LinkedList クラス
- OpenJIT.OptimizeRTL クラス
- OpenJIT.ParseBytecode クラス
- OpenJIT.RegAlloc クラス
- OpenJIT.RegAlloc$PhyRegs クラス
- OpenJIT.RegAlloc$VirReg クラス
- OpenJIT.Runtime クラス (契約の対象外)
- OpenJIT.Select クラス
- OpenJIT.Sparc クラス (契約の対象外)
- OpenJIT.Var クラス

- 例外

  なし .

4.1.1 OpenJIT.Constants インタフェース

(1) 概要

OpenJIT の各クラスで利用する定数を定義したインタフェース .

(2) フィールド (変数)

- public static final char SIGC_VOID

  シグナチャに含まれる文字の void 型を表す.

- public static final char SIGC_BOOLEAN

  シグナチャに含まれる文字の boolean 型を表す.

- public static final char SIGC_BYTE

  シグナチャに含まれる文字の byte 型を表す.

- public static final char SIGC_CHAR

  シグナチャに含まれる文字の char 型を表す.

- public static final char SIGC_SHORT

  シグナチャに含まれる文字の short 型を表す.

- public static final char SIGC_INT

  シグナチャに含まれる文字の int 型を表す.

- public static final char SIGC_LONG

  シグナチャに含まれる文字の long 型を表す.

- public static final char SIGC_FLOAT

  シグナチャに含まれる文字の float 型を表す.

- public static final char SIGC_DOUBLE

  シグナチャに含まれる文字の double 型を表す.

- public static final char SIGC_ARRAY

  シグナチャに含まれる文字の配列型を表す.

- public static final char SIGC_CLASS

  シグナチャに含まれる文字のクラス型の開始を表す.

- public static final char SIGC_METHOD

  シグナチャに含まれる文字の引数の開始を表す.

- public static final char SIGC_ENDCLASS

  シグナチャに含まれる文字のクラス型の終わりを表す.

- public static final char SIGC_ENDMETHOD

  シグナチャに含まれる文字の引数の終わりを表す.

- public static final char SIGC_PACKAGE

  シグナチャに含まれる文字のクラスの階層のセパレータを表す.

- public static final int CONSTANT_UTF8

  constant pool の種類． UTF8

- public static final int CONSTANT_UNICODE

  constant pool の種類． UNICODE

- public static final int CONSTANT_INTEGER

  constant pool の種類． int

- public static final int CONSTANT_FLOAT

  constant pool の種類． float

- public static final int CONSTANT_LONG

  constant pool の種類． long

- public static final int CONSTANT_DOUBLE

  constant pool の種類． double

- public static final int CONSTANT_CLASS

  constant pool の種類． class

- public static final int CONSTANT_STRING

  constant pool の種類． String

- public static final int CONSTANT_FIELD

  constant pool の種類．フィールド

- public static final int CONSTANT_METHOD

  constant pool の種類．メソッド

- public static final int CONSTANT_INTERFACEMETHOD

  constant pool の種類．インタフェース

- public static final int CONSTANT_NAMEANDTYPE

  constant pool の種類．名前

- public static final int ACC_PUBLIC

  JDK におけるクラスの属性． public

- public static final int ACC_PRIVATE

  JDK におけるクラスの属性． private

- public static final int ACC_PROTECTED

  JDK におけるクラスの属性． protected

- public static final int ACC_STATIC

  JDK におけるクラスの属性． static

- public static final int ACC_FINAL

  JDK におけるクラスの属性． final

- public static final int ACC_SYNCHRONIZED

  JDK におけるクラスの属性． synchronized

- public static final int ACC_VOLATILE

  JDK におけるクラスの属性． volatile

- public static final int ACC_TRANSIENT

  JDK におけるクラスの属性． transient

- public static final int ACC_NATIVE

  JDK におけるクラスの属性． native

- public static final int ACC_INTERFACE

  JDK におけるクラスの属性． interface

- public static final int ACC_ABSTRACT

  JDK におけるクラスの属性． abstract

- public static final int ACC_SUPER

  JDK におけるクラスの属性． super

- public static final int opc_nop

  バイトコード命令． nop

- public static final int opc_aconst_null

  バイトコード命令． aconst_null

- public static final int opc_iconst_m1

  バイトコード命令． iconst_m1

- public static final int opc_iconst_0

  バイトコード命令． iconst_0

- public static final int opc_iconst_1

  バイトコード命令． iconst_1

- public static final int opc_iconst_2

  バイトコード命令． iconst_2

- public static final int opc_iconst_3

  バイトコード命令． iconst_3

- public static final int opc_iconst_4

  バイトコード命令． iconst_4

- public static final int opc_iconst_5

  バイトコード命令． iconst_5

- public static final int opc_lconst_0

  バイトコード命令． lconst_0

- public static final int opc_lconst_1

  バイトコード命令． lconst_1

- public static final int opc_fconst_0

  バイトコード命令．fconst_0

- public static final int opc_fconst_1

  バイトコード命令．fconst_1

- public static final int opc_fconst_2

  バイトコード命令．fconst_2

- public static final int opc_dconst_0

  バイトコード命令．dconst_0

- public static final int opc_dconst_1

  バイトコード命令．dconst_1

- public static final int opc_bipush

  バイトコード命令．bipush

- public static final int opc_sipush

  バイトコード命令．sipush

- public static final int opc_ldc

  バイトコード命令．ldc

- public static final int opc_ldc_w

  バイトコード命令．ldc_w

- public static final int opc_ldc2_w

  バイトコード命令．ldc2_w

- public static final int opc_iload

  バイトコード命令．iload

- public static final int opc_lload

  バイトコード命令．lload

- public static final int opc_fload

  バイトコード命令．fload

- public static final int opc_dload

  バイトコード命令．dload

- public static final int opc_aload

  バイトコード命令．aload

- public static final int opc_iload_0

  バイトコード命令．iload_0

- public static final int opc_iload_1

  バイトコード命令．iload_1

- public static final int opc_iload_2

  バイトコード命令．iload_2

- public static final int opc_iload_3

  バイトコード命令．iload_3

- public static final int opc_lload_0

  バイトコード命令．lload_0

- public static final int opc_lload_1

  バイトコード命令．lload_1

- public static final int opc_lload_2

  バイトコード命令．lload_2

- public static final int opc_lload_3

  バイトコード命令．lload_3

- public static final int opc_fload_0

  バイトコード命令．fload_0

- public static final int opc_fload_1

  バイトコード命令．fload_1

- public static final int opc_fload_2

  バイトコード命令．fload_2

- public static final int opc_fload_3

  バイトコード命令．fload_3

- public static final int opc_dload_0

  バイトコード命令．dload_0

- public static final int opc_dload_1

  バイトコード命令．dload_1

- public static final int opc_dload_2

  バイトコード命令．dload_2

- public static final int opc_dload_3

  バイトコード命令．dload_3

- public static final int opc_aload_0

  バイトコード命令．aload_0

- public static final int opc_aload_1

  バイトコード命令．aload_1

- public static final int opc_aload_2

  バイトコード命令．aload_2

- public static final int opc_aload_3

  バイトコード命令．aload_3

- public static final int opc_iaload

  バイトコード命令．iaload

- public static final int opc_laload

  バイトコード命令．laload

- public static final int opc_faload

  バイトコード命令．faload

- public static final int opc_daload

  バイトコード命令．daload

- public static final int opc_aaload

  バイトコード命令．aaload

- public static final int opc_baload

  バイトコード命令．baload

- public static final int opc_caload

  バイトコード命令．caload

- public static final int opc_saload

  バイトコード命令．saload

- public static final int opc_istore

  バイトコード命令．istore

- public static final int opc_lstore

  バイトコード命令．lstore

- public static final int opc_fstore

  バイトコード命令．fstore

- public static final int opc_dstore

  バイトコード命令．dstore

- public static final int opc_astore

  バイトコード命令．astore

- public static final int opc_istore_0

  バイトコード命令．istore_0

- public static final int opc_istore_1

  バイトコード命令．istore_1

- public static final int opc_istore_2

  バイトコード命令．istore_2

- public static final int opc_istore_3

  バイトコード命令．istore_3

- public static final int opc_lstore_0

  バイトコード命令．lstore_0

- public static final int opc_lstore_1

  バイトコード命令．lstore_1

- public static final int opc_lstore_2

  バイトコード命令．lstore_2

- public static final int opc_lstore_3

  バイトコード命令．lstore_3

- public static final int opc_fstore_0

  バイトコード命令．fstore_0

- public static final int opc_fstore_1

  バイトコード命令．fstore_1

- public static final int opc_fstore_2

  バイトコード命令．fstore_2

- public static final int opc_fstore_3

  バイトコード命令．fstore_3

- public static final int opc_dstore_0

  バイトコード命令．dstore_0

- public static final int opc_dstore_1

  バイトコード命令．dstore_1

- public static final int opc_dstore_2

  バイトコード命令．dstore_2

- public static final int opc_dstore_3

  バイトコード命令．dstore_3

- public static final int opc_astore_0

  バイトコード命令．astore_0

- public static final int opc_astore_1

  バイトコード命令．astore_1

- public static final int opc_astore_2

  バイトコード命令．astore_2

- public static final int opc_astore_3

  バイトコード命令．astore_3

- public static final int opc_iastore

  バイトコード命令．iastore

- public static final int opc_lastore

  バイトコード命令．lastore

- public static final int opc_fastore

  バイトコード命令．fastore

- public static final int opc_dastore

  バイトコード命令．dastore

- public static final int opc_aastore

  バイトコード命令．aastore

- public static final int opc_bastore

  バイトコード命令．bastore

- public static final int opc_castore

  バイトコード命令．castore

- public static final int opc_sastore

  バイトコード命令．sastore

- public static final int opc_pop

  バイトコード命令．pop

- public static final int opc_pop2

  バイトコード命令．pop2

- public static final int opc_dup

  バイトコード命令．dup

- public static final int opc_dup_x1

  バイトコード命令．dup_x1

- public static final int opc_dup_x2

  バイトコード命令．dup_x2

- public static final int opc_dup2

  バイトコード命令．dup2

- public static final int opc_dup2_x1

  バイトコード命令．dup2_x1

- public static final int opc_dup2_x2

  バイトコード命令．dup2_x2

- public static final int opc_swap

  バイトコード命令．swap

- public static final int opc_iadd

  バイトコード命令．iadd

- public static final int opc_ladd

  バイトコード命令．ladd

- public static final int opc_fadd

  バイトコード命令．fadd

- public static final int opc_dadd

  バイトコード命令．dadd

- public static final int opc_isub

  バイトコード命令．isub

- public static final int opc_lsub

  バイトコード命令．lsub

- public static final int opc_fsub

  バイトコード命令．fsub

- public static final int opc_dsub

  バイトコード命令．dsub

- public static final int opc_imul

  バイトコード命令．imul

- public static final int opc_lmul

  バイトコード命令．lmul

- public static final int opc_fmul

  バイトコード命令．fmul

- public static final int opc_dmul

  バイトコード命令．dmul

- public static final int opc_idiv

  バイトコード命令．idiv

- public static final int opc_ldiv

  バイトコード命令．ldiv

- public static final int opc_fdiv

  バイトコード命令．fdiv

- public static final int opc_ddiv

  バイトコード命令．ddiv

- public static final int opc_irem

  バイトコード命令．irem

- public static final int opc_lrem

  バイトコード命令．lrem

- public static final int opc_frem

  バイトコード命令．frem

- public static final int opc_drem

  バイトコード命令．drem

- public static final int opc_ineg

  バイトコード命令．ineg

- public static final int opc_lneg

  バイトコード命令．lneg

- public static final int opc_fneg

  バイトコード命令．fneg

- public static final int opc_dneg

  バイトコード命令．dneg

- public static final int opc_ishl

  バイトコード命令．ishl

- public static final int opc_lshl

  バイトコード命令．lshl

- public static final int opc_ishr

  バイトコード命令．ishr

- public static final int opc_lshr

  バイトコード命令．lshr

- public static final int opc_iushr

  バイトコード命令．iushr

- public static final int opc_lushr

  バイトコード命令．lushr

- public static final int opc_iand

  バイトコード命令．iand

- public static final int opc_land

  バイトコード命令．land

- public static final int opc_ior

  バイトコード命令．ior

- public static final int opc_lor

  バイトコード命令．lor

- public static final int opc_ixor

  バイトコード命令．ixor

- public static final int opc_lxor

  バイトコード命令．lxor

- public static final int opc_iinc

  バイトコード命令．iinc

- public static final int opc_i2l

  バイトコード命令．i2l

- public static final int opc_i2f

  バイトコード命令．i2f

- public static final int opc_i2d

  バイトコード命令．i2d

- public static final int opc_l2i

  バイトコード命令．l2i

- public static final int opc_l2f

  バイトコード命令．l2f

- public static final int opc_l2d

  バイトコード命令．l2d

- public static final int opc_f2i

  バイトコード命令．f2i

- public static final int opc_f2l

  バイトコード命令．f2l

- public static final int opc_f2d

  バイトコード命令．f2d

- public static final int opc_d2i

  バイトコード命令．d2i

- public static final int opc_d2l

  バイトコード命令．d2l

- public static final int opc_d2f

  バイトコード命令．d2f

- public static final int opc_i2b

  バイトコード命令．i2b

- public static final int opc_i2c

  バイトコード命令．i2c

- public static final int opc_i2s

  バイトコード命令．i2s

- public static final int opc_lcmp

  バイトコード命令．lcmp

- public static final int opc_fcmpl

  バイトコード命令．fcmpl

- public static final int opc_fcmpg

  バイトコード命令．fcmpg

- public static final int opc_dcmpl

  バイトコード命令．dcmpl

- public static final int opc_dcmpg

  バイトコード命令．dcmpg

- public static final int opc_ifeq

  バイトコード命令．ifeq

- public static final int opc_ifne

  バイトコード命令．ifne

- public static final int opc_iflt

  バイトコード命令．iflt

- public static final int opc_ifge

  バイトコード命令．ifge

- public static final int opc_ifgt

  バイトコード命令．ifgt

- public static final int opc_ifle

  バイトコード命令．ifle

- public static final int opc_if_icmpeq

  バイトコード命令．if_icmpeq

- public static final int opc_if_icmpne

  バイトコード命令．if_icmpne

- public static final int opc_if_icmplt

  バイトコード命令．if_icmplt

- public static final int opc_if_icmpge

  バイトコード命令．if_icmpge

- public static final int opc_if_icmpgt

  バイトコード命令．if_icmpgt

- public static final int opc_if_icmple

  バイトコード命令．if_icmple

- public static final int opc_if_acmpeq

  バイトコード命令．if_acmpeq

- public static final int opc_if_acmpne

  バイトコード命令．if_acmpne

- public static final int opc_goto

  バイトコード命令．goto

- public static final int opc_jsr

  バイトコード命令．jsr

- public static final int opc_ret

  バイトコード命令．ret

- public static final int opc_tableswitch

  バイトコード命令．tableswitch

- public static final int opc_lookupswitch

  バイトコード命令．lookupswitch

- public static final int opc_ireturn

  バイトコード命令．ireturn

- public static final int opc_lreturn

  バイトコード命令．lreturn

- public static final int opc_freturn

  バイトコード命令．freturn

- public static final int opc_dreturn

  バイトコード命令．dreturn

- public static final int opc_areturn

  バイトコード命令．areturn

- public static final int opc_return

  バイトコード命令．return

- public static final int opc_getstatic

  バイトコード命令．getstatic

- public static final int opc_putstatic

  バイトコード命令．putstatic

- public static final int opc_getfield

  バイトコード命令．getfield

- public static final int opc_putfield

  バイトコード命令．putfield

- public static final int opc_invokevirtual

  バイトコード命令．invokevirtual

- public static final int opc_invokespecial

  バイトコード命令．invokespecial

- public static final int opc_invokestatic

  バイトコード命令．invokestatic

- public static final int opc_invokeinterface

  バイトコード命令．invokeinterface

- public static final int opc_xxxunusedxxx

  バイトコード命令．xxxunusedxxx

- public static final int opc_new

  バイトコード命令．new

- public static final int opc_newarray

  バイトコード命令．newarray

- public static final int opc_anewarray

  バイトコード命令．anewarray

- public static final int opc_arraylength

  バイトコード命令．arraylength

- public static final int opc_athrow

  バイトコード命令．athrow

- public static final int opc_checkcast

  バイトコード命令．checkcast

- public static final int opc_instanceof

  バイトコード命令．instanceof

- public static final int opc_monitorenter

  バイトコード命令．monitorenter

- public static final int opc_monitorexit

  バイトコード命令．monitorexit

- public static final int opc_wide

  バイトコード命令．wide

- public static final int opc_multianewarray

  バイトコード命令．multianewarray

- public static final int opc_ifnull

  バイトコード命令．ifnull

- public static final int opc_ifnonnull

  バイトコード命令．ifnonnull

- public static final int opc_goto_w

  バイトコード命令．goto_w

- public static final int opc_jsr_w

  バイトコード命令．jsr_w

- public static final int opc_breakpoint

  バイトコード命令．breakpoint

- public static final int opc_ldc_quick

  バイトコード命令．ldc_quick

- public static final int opc_ldc_w_quick

  バイトコード命令．ldc_w_quick

- public static final int opc_ldc2_w_quick

  バイトコード命令．ldc2_w_quick

- public static final int opc_getfield_quick

  バイトコード命令．getfield_quick

- public static final int opc_putfield_quick

  バイトコード命令．putfield_quick

- public static final int opc_getfield2_quick

  バイトコード命令．getfield2_quick

- public static final int opc_putfield2_quick

  バイトコード命令．putfield2_quick

- public static final int opc_getstatic_quick

  バイトコード命令．getstatic_quick

- public static final int opc_putstatic_quick

  バイトコード命令．putstatic_quick

- public static final int opc_getstatic2_quick

  バイトコード命令．getstatic2_quick

- public static final int opc_putstatic2_quick

  バイトコード命令．putstatic2_quick

- public static final int opc_invokevirtual_quick

  バイトコード命令．invokevirtual_quick

- public static final int opc_invokenonvirtual_quick

  バイトコード命令． invokenonvirtual_quick

- public static final int opc_invokesuper_quick

  バイトコード命令． invokesuper_quick

- public static final int opc_invokestatic_quick

  バイトコード命令． invokestatic_quick

- public static final int opc_invokeinterface_quick

  バイトコード命令． invokeinterface_quick

- public static final int opc_invokevirtualobject_quick

  バイトコード命令． invokevirtualobject_quick

- public static final int opc_invokeignored_quick

  バイトコード命令． invokeignored_quick

- public static final int opc_new_quick

  バイトコード命令． new_quick

- public static final int opc_anewarray_quick

  バイトコード命令． anewarray_quick

- public static final int opc_multianewarray_quick

  バイトコード命令． multianewarray_quick

- public static final int opc_checkcast_quick

  バイトコード命令． checkcast_quick

- public static final int opc_instanceof_quick

  バイトコード命令． instanceof_quick

- public static final int opc_invokevirtual_quick_w

  バイトコード命令． invokevirtual_quick_w

- public static final int opc_getfield_quick_w

  バイトコード命令．getfield_quick_w

- public static final int opc_putfield_quick_w

  バイトコード命令．putfield_quick_w

- public static final int opc_nonnull_quick

  バイトコード命令．nonnull_quick

- public static final int opc_first_unused_index

  バイトコード命令．first_unused_index

- public static final int opc_software

  バイトコード命令．software

- public static final int opc_hardware

  バイトコード命令．hardware

- public static final int TYPE_INT

  バックエンド中間コード，RTL のオペランドの型．int

- public static final int TYPE_LONG

  バックエンド中間コード，RTL のオペランドの型．long

- public static final int TYPE_FLOAT

  バックエンド中間コード，RTL のオペランドの型．float

- public static final int TYPE_DOUBLE

  バックエンド中間コード，RTL のオペランドの型．double

- public static final int TYPE_MASK

  バックエンド中間コード，RTL のオペランドの型を取り出すマスク．

- public static final int TAG_CONST

  バックエンド中間コード，RTL のオペランドのタグ．定数．

- public static final int TAG_STACK

  バックエンド中間コード，RTL のオペランドのタグ．スタック変数．

- public static final int TAG_LOCAL

  バックエンド中間コード，RTL のオペランドのタグ．ローカル変数．

- public static final int TAG_PARAM

  バックエンド中間コード，RTL のオペランドのタグ．パラメタ変数．

- public static final int TAG_MASK

  バックエンド中間コード，RTL のオペランドのタグを取り出すマスク．

- public static final int TAG_SI

  バックエンド中間コード，RTL のオペランドの属性．int のスタック変数．

- public static final int TAG_SF

  バックエンド中間コード，RTL のオペランドの属性．float のスタック変数．

- public static final int TAG_SD

  バックエンド中間コード，RTL のオペランドの属性．double のスタック変数．

- public static final int TAG_VI

  バックエンド中間コード，RTL のオペランドの属性．int のローカル変数．

- public static final int TAG_VF

  バックエンド中間コード，RTL のオペランドの属性．float のローカル変数．

- public static final int TAG_VD

  バックエンド中間コード，RTL のオペランドの属性．double のローカル変数．

- public static final int TAG_PI

  バックエンド中間コード，RTL のオペランドの属性．int のパラメタ変数．

- public static final int TAG_PL

  バックエンド中間コード，RTL のオペランドの属性．long のパラメタ変数．

- public static final int TAG_PF

  バックエンド中間コード，RTL のオペランドの属性．float のパラメタ変数．

- public static final int TAG_PD

  バックエンド中間コード，RTL のオペランドの属性．double のパラメタ変数．

- public static final int TAG_REF

  バックエンド中間コード，RTL のオペランドの属性．参照されていることを表す．

- public static final int CC_A

  バックエンド中間コード，RTL の分岐ノードの種類．絶対分岐．

- public static final int CC_E

  バックエンド中間コード，RTL の分岐ノードの種類．整数比較の結果，等しければ分岐．

- public static final int CC_NE

  バックエンド中間コード，RTL の分岐ノードの種類．整数比較の結果，等しくなければ分岐．

- public static final int CC_L

  バックエンド中間コード，RTL の分岐ノードの種類．整数比較の結果，小さければ分岐．

- public static final int CC_LE

  バックエンド中間コード，RTL の分岐ノードの種類．整数比較の結果，大きくなければ分岐．

- public static final int CC_G

  バックエンド中間コード，RTL の分岐ノードの種類．整数比較の結果，大きければ分岐．

- public static final int CC_GE

  バックエンド中間コード，RTL の分岐ノードの種類．整数比較の結果，小さくなければ分岐．

- public static final int CC_FA

  バックエンド中間コード，RTL の分岐ノードの種類．絶対分岐．

- public static final int CC_FE

  バックエンド中間コード，RTL の分岐ノードの種類．浮動小数比較の結果，等しければ分岐．

- public static final int CC_FNE

  バックエンド中間コード，RTL の分岐ノードの種類．浮動小数比較の結果，等しくなければ分岐．

- public static final int CC_FL

  バックエンド中間コード，RTL の分岐ノードの種類．浮動小数比較の結果，小さければ分岐．

- public static final int CC_FLE

  バックエンド中間コード，RTL の分岐ノードの種類．浮動小数比較の結果，大きくなければ分岐．

- public static final int CC_FG

  バックエンド中間コード，RTL の分岐ノードの種類．浮動小数比較の結果，大きければ分岐．

- public static final int CC_FGE

  バックエンド中間コード，RTL の分岐ノードの種類．浮動小数比較の結果，小さくなければ分岐．

- public static final int CC_NFL

  バックエンド中間コード，RTL の分岐ノードの種類．浮動小数比較の結果，NaN または小さければ分岐．

- public static final int CC_NFLE

  バックエンド中間コード，RTL の分岐ノードの種類．浮動小数比較の結果，NaN または大きくなければ分岐．

- public static final int CC_NFG

  バックエンド中間コード，RTL の分岐ノードの種類．浮動小数比較の結果，NaN または大きければ分岐．

- public static final int CC_NFGE

  バックエンド中間コード，RTL の分岐ノードの種類．浮動小数比較の結果，NaN または小さくなければ分岐．

- public static final int CC_LU

  バックエンド中間コード，RTL の分岐ノードの種類．符合なし整数比較の結果，小さければ分岐．

- public static final int CC_LEU

  バックエンド中間コード，RTL の分岐ノードの種類．符合なし整数比較の結果，大きくなければ分岐．

- public static final int CC_GU

  バックエンド中間コード，RTL の分岐ノードの種類．符合なし整数比較の結果，大きければ分岐．

- public static final int CC_GEU

  バックエンド中間コード，RTL の分岐ノードの種類．符合なし整数比較の結果，小さくなければ分岐．

- public static final int CC_TBLOB

  バックエンド中間コード，RTL の分岐ノードの種類．テーブル分岐でテーブル領域外の分岐．

- public static final int CC_TBLSW

  バックエンド中間コード，RTL の分岐ノードの種類．テーブル分岐．

- public static final int CC_JSR

  バックエンド中間コード，RTL の分岐ノードの種類．JVM の jsr 命令．

- public static final int ARG0

  １番目の引数を表すパラメタ変数の番号．

- public static final int ARG1

  ２番目の引数を表すパラメタ変数の番号．

- public static final int ARG2

  ３番目の引数を表すパラメタ変数の番号．

- public static final int ARG3

  ４番目の引数を表すパラメタ変数の番号．

- public static final int RETI

  関数の戻り値 (int) を格納するパラメタ変数の番号．

- public static final int RETL

  関数の戻り値 (long) を格納するパラメタ変数の番号．

- public static final int RETF

  関数の戻り値 (float,double) を格納するパラメタ変数の番号．

- public static final int METHOD_OFFSET

  配列オブジェクトの大きさが格納されているアドレスのオフセット．JDK の内部構造に依存．

- public static final int METHOD_FLAG_BITS

  配列オブジェクトの大きさを取り出すときのシフト量．JDK の内部構造に依存．

- public static final int FLAG_HAS_INVOKE

  コンパイルのためのフラグ．バイトコードがメソッド呼び出しを含む．

- public static final int FLAG_HAS_EXCEPTION

  コンパイルのためのフラグ．バイトコードが例外を起こす可能性がある．

- public static final int FLAG_HAS_JSR

  コンパイルのためのフラグ．バイトコードが jsr 命令を含む．

- public static final int FLAG_HAS_THROW

  コンパイルのためのフラグ．バイトコードが athrow 命令を含む．

- public static final int FLAG_HAS_CALL

  コンパイルのためのフラグ．バックエンド中間コードが CALL 命令を含む．

- public static final int FLAG_OPTIMIZE_LEAF

  コンパイルのためのフラグ． Leaf 最適化が可能なことを示す．

(3) コンストラクタ

なし．

(4) メソッド

なし．

4.1.2 OpenJIT.LinkedList クラス

(1) 概要

連結リストを構成するための基本クラス．BCinfo, ILnode クラスのスーパークラスである．

(2) フィールド (変数)

- ILnode next

  連結リストをたどる．

(3) コンストラクタ

なし．

(4) メソッド

なし．

4.1.3 OpenJIT.BCinfo クラス

(1) 概要

バイトコードに関する情報を表現する．

(2) フィールド (変数)

- static final int FLAG_BRANCH_TARGET

  分岐先であることを表すフラグ．

- static final int FLAG_CTRLFLOW_END

  制御フローがここで終了することを表すフラグ．

- static final int FLAG_EXCEPTION_HANDLER

  例外ハンドラの先頭命令であることを表すフラグ．

- static final int FLAG_IN_TRY_BLOCK

  例外処理を表すフラグ．

- static final int FLAG_CTRLFLOW_DONE

  フロー解析処理済みであることを表すフラグ．

- static final int FLAG_BEGIN_BASIC_BLOCK

  基本ブロックの先頭命令であることを表すフラグ．

- int flag

  このバイトコード命令についての以下のフラグ値を格納する．

  - FLAG_BRANCH_TARGET
  - FLAG_CTRLFLOW_END
  - FLAG_EXCEPTION_HANDLER
  - FLAG_IN_TRY_BLOCK
  - FLAG_CTRLFLOW_DONE

– FLAG_BEGIN_BASIC_BLOCK

- int stack_delta

  このバイトコード命令の実行前と後で，Java のスタックの深さがどれくらい変化するかを表す．

  OpenJIT 中間言語変換機能によりこの値は設定される．

- int stack_size

  このバイトコード命令を実行する直前の Java のスタックの深さを表す．OpenJIT RTL 変換機能によりこの値は設定される．

- int native_pc

  生成するネイティブコードのアドレス (絶対番地) を保持する．

- BBinfo bb

  基本ブロック情報へのポインタ．基本ブロックの先頭のバイトコード命令でないときは，この値は null である．

(3) コンストラクタ

- public BCinfo()

  機能概要 BCinfo オブジェクトを生成する．

  機能説明 初期化を行う．

  入力データ なし．

  出力データ なし．

  処理方式 native_pc に -1 を設定する．-1 はまだアドレスが確定していないことを意味する．

  エラー処理 なし．

(4) メソッド

なし．

4.1.4 OpenJIT.ILnode クラス

(1) 概要

バックエンド中間コードおよびＲＴＬのノードを表現するクラス．OpenJIT 中間コード変換機能により生成され，OpenJIT ＲＴＬ変換機能によりＲＴＬに変換される．

(2) フィールド (変数)

- static final byte ELIMINATED

  最適化により削除されたことを表すノード種別．

- static final byte NOP

  なにもしない (No OPeration) を表すノード種別．

- static final byte ARRAYCHK

  配列の境界チェックを表すノード種別．

- static final byte DIV0CHK

  ゼロ除算チェックを表すノード種別．

- static final byte BRANCH

  分岐を表すノード種別．

- static final byte CALL

  コールを表すノード種別．

- static final byte RETURN

  リターンを表すノード種別．

- static final byte TBLSW

  テーブル分岐を表すノード種別．

- static final byte RET

  JSR からの復帰を表すノード種別．

- static final byte MOVE

  転送を表すノード種別．

- static final byte ADD

  加算を表すノード種別．

- static final byte ADDCC

  キャリー生成加算を表すノード種別．

- static final byte ADDX

  キャリー付き加算を表すノード種別．

- static final byte AND

  論理積を表すノード種別．

- static final byte LD

  ロード (32bit) を表すノード種別．

- static final byte LDSH

  ロード (16bit) を表すノード種別．

- static final byte LDSB

  ロード (8bit) を表すノード種別．

- static final byte LDUH

  ロード (符合なし 8bit) を表すノード種別．

- static final byte OR

  論理和を表すノード種別．

- static final byte DIV

  除算を表すノード種別．

- static final byte SLL

  左論理シフトを表すノード種別．

- static final byte MUL

  乗算を表すノード種別．

- static final byte SRA

  右算術シフトを表すノード種別．

- static final byte SRL

  右論理シフトを表すノード種別．

- static final byte ST

  ストア (32bit) を表すノード種別．

- static final byte STB

  ストア (8bit) を表すノード種別．

- static final byte STH

  ストア (16bit) を表すノード種別．

- static final byte SUB

  減算を表すノード種別．

- static final byte SUBCC

  キャリー生成減算を表すノード種別．

- static final byte SUBX

  キャリー付き減算を表すノード種別．

- static final byte XOR

  排他的論理和を表すノード種別．

- static final byte IREM

  剰余算を表すノード種別．

- static final byte I2F

  int から float への変換を表すノード種別．

- static final byte I2D

  int から double への変換を表すノード種別．

- static final byte F2I

  float から int への変換を表すノード種別．

- static final byte D2I

  double から int への変換を表すノード種別．

- static final byte F2D

  float から double への変換を表すノード種別．

- static final byte D2F

  double から float への変換を表すノード種別．

- static final byte FNEG

  float や double の値の符号反転を表すノード種別．

- static final byte FABS

  float や double の絶対値を表すノード種別．

- static final byte FSQRT

  float や double の二乗根を表すノード種別．

- static final byte FCMP

  float や double の比較を表すノード種別．

- static final byte CONV

  整数と浮動小数点データの転送を表すノード種別

- private int op_and_tag

  ノード種別とオペランドの型を保持する．

- int dval

  Ｄオペランドの値．

- int lval

  Ｌオペランドの値．

- int rval

  Ｒオペランドの値．

- private static final String name[]

  デバッグ用．ノード種別を java.lang.String で表したもの．

(3) コンストラクタ

- public ILnode()

  機能概要 ILnode オブジェクトを生成する．

  機能説明 ILnode オブジェクトを生成する．

  入力データ なし．

  出力データ なし．

  処理方式 なし．

  エラー処理 なし．

- public ILnode(LinkedList ll)

  機能概要 ILnode オブジェクトを生成し，連結リストにつなぐ．

  機能説明 ILnode オブジェクトを生成し，連結リストにつなぐ．

  入力データ ll(連結リスト)

  出力データ なし．

  処理方式 ll.next に this をセットする．

  エラー処理 なし．

(4) メソッド

- final int op()

  機能概要 ノード種別を得る．

  機能説明 ノード種別を得る．

  入力データ なし．

  出力データ ノード種別．

  処理方式 op_and_tag の上位 8bit を取り出す．

  エラー処理 なし．

- final int dtype()

  機能概要 Ｄオペランドの種別を得る．

  機能説明 Ｄオペランドの種別を得る．

  入力データ なし．

  出力データ Ｄオペランドの種別．

  処理方式 op_and_tag を 16bit 右シフトして 0xff と論理積をとる．

  エラー処理 なし．

- final int ltype()

  機能概要 Ｌオペランドの種別を得る．

  機能説明 Ｌオペランドの種別を得る．

  入力データ なし．

  出力データ Ｌオペランドの種別．

  処理方式 op_and_tag を 8bit 右シフトして 0xff と論理積をとる．

  エラー処理 なし．

- final int rtype()

機能概要 Ｒオペランドの種別を得る．

機能説明 Ｒオペランドの種別を得る．

入力データ なし．

出力データ Ｒオペランドの種別．

処理方式 op_and_tag と 0xff の論理積をとる．

エラー処理 なし．

- final int tagD()

  機能概要 Ｄオペランドのタグを得る．

  機能説明 Ｄオペランドのタグを得る．

  入力データ なし．

  出力データ Ｄオペランドのタグ．

  処理方式 op_and_tag を 16bit 右シフトして TAG_MASK と論理積をとる．

  エラー処理 なし．

- final int tagL()

  機能概要 Ｌオペランドのタグを得る．

  機能説明 Ｌオペランドのタグを得る．

  入力データ なし．

  出力データ Ｌオペランドのタグ．

  処理方式 op_and_tag を 8bit 右シフトして TAG_MASK と論理積をとる．

  エラー処理 なし．

- final int tagR()

  機能概要 Ｒオペランドのタグを得る．

  機能説明 Ｒオペランドのタグを得る．

  入力データ なし．

出力データ Rオペランドのタグ．

処理方式 op_and_tag とTAG_MASK の論理積をとる．

エラー処理 なし．

- final int typeD()

  機能概要 Dオペランドの型を得る．

  機能説明 Dオペランドの型を得る．

  入力データ なし．

  出力データ Dオペランドの型．

  処理方式 op_and_tag を16bit 右シフトして0xff と論理積をとる．

  エラー処理 なし．

- final int typeL()

  機能概要 Lオペランドの型を得る．

  機能説明 Lオペランドの型を得る．

  入力データ なし．

  出力データ Lオペランドの型．

  処理方式 op_and_tag を8bit 右シフトして0xff と論理積をとる．

  エラー処理 なし．

- final int typeR()

  機能概要 Rオペランドの型を得る．

  機能説明 Rオペランドの型を得る．

  入力データ なし．

  出力データ Rオペランドの型．

  処理方式 op_and_tag と0xff の論理積をとる．

  エラー処理 なし．

- public final ILnode insn(int op, int dtype, int dval, int ltype, int lval, int rtype, int rval)

  機能概要 バックエンド中間コードを設定する．

  機能説明 標準的なバックエンド中間コードを設定する．

  入力データ op(ノード種別), dtype(Ｄオペランドの種別), dval(Ｄオペランドの値)，ltype(Ｌオペランドの種別), lval(Ｌオペランドの値)，rtype(Ｒオペランドの種別), rval(Ｒオペランドの値)

  出力データ this

  処理方式 それぞれ 8bit の値である op,dtype,ltype,rtype の値をパックして，32bit の op_and_tag に設定する．dval,lval,rval をそれぞれ this に設定する．

  エラー処理 なし．

- public final ILnode move(int dtype, int dval, int rtype, int rval)

  機能概要 転送命令を表すバックエンド中間コードを設定する．

  機能説明 転送命令を表すバックエンド中間コードを設定する．

  入力データ dtype(Ｄオペランドの種別), dval(Ｄオペランドの値)，rtype(Ｒオペランドの種別), rval(Ｒオペランドの値)

  出力データ this

  処理方式 それぞれ 8bit の値である MOVE,dtype,rtype の値をパックして，32bit の op_and_tag に設定する．dval,rval をそれぞれ this に設定する．

  エラー処理 なし．

- public final ILnode moveStack(int dval, int rval)

  機能概要 スタック転送命令を表すバックエンド中間コードを設定する．

  機能説明 スタック転送命令を表すバックエンド中間コードを設定する．スタック転送命令はオペランドの型が確定していない場合に用いられる．

  入力データ dval(Ｄオペランドの値：スタックの深さ)，rval(Ｒオペランドの値：スタックの深さ)

出力データ this

処理方式 op_and_tag に MOVE,TAG_STACK を設定する．dval,rval を this に設定する．lval には -1 を設定する．

エラー処理 なし．

- public final ILnode call(int func, int arg0)

機能概要 CALL 命令を表すバックエンド中間コードを設定する．

機能説明 CALL 命令を表すバックエンド中間コードを設定する．

入力データ func(ランタイムルーチン)，arg0(constant pool の index)

出力データ this

処理方式 op_and_tag のノード種別に CALL を設定する．lval に arg0, rval に func を設定する．

エラー処理 なし．

- public final ILnode conv(int dtype, int dval, int rtype, int rval)

機能概要 整数と浮動小数点データの転送を表すバックエンド中間コードを設定する．

機能説明 整数と浮動小数点データの転送を表すバックエンド中間コードを設定する．

入力データ dtype(D オペランドの種別), dval(D オペランドの値)，rtype(R オペランドの種別), rval(R オペランドの値)

出力データ this

処理方式 それぞれ 8bit の値である CONV,dtype,rtype の値をパックして，32bit の op_and_tag に設定する．dval,rval をそれぞれ this に設定する．

エラー処理 なし．

- public final ILnode branch(int cond, int target_pc)

機能概要 分岐命令を表すバックエンド中間コードを設定する．

機能説明 分岐命令を表すバックエンド中間コードを設定する．

入力データ cond(分岐条件), target_pc(分岐先 PC)

出力データ this

処理方式 op_and_tag のノード種別に BRANCH を設定する．lval に cond, rval に target_pc を設定する．

エラー処理 なし．

- public final ILnode jsr(int target_pc)

  機能概要 JSR 命令を表すバックエンド中間コードを設定する．

  機能説明 JSR 命令を表すバックエンド中間コードを設定する．

  入力データ target_pc(分岐先 PC)

  出力データ this

  処理方式 op_and_tag のノード種別に BRANCH を設定する．D オペランドの種別として TAG_SI を設定する．lval に CC_JSR, rval に target_pc を設定する．

  エラー処理 なし．

- public final ILnode nop()

  機能概要 NOP を表すバックエンド中間コードを設定する．

  機能説明 NOP を表すバックエンド中間コードを設定する．

  入力データ なし．

  出力データ this

  処理方式 op_and_tag のノード種別に NOP を設定する．

  エラー処理 なし．

- final void remove()

  機能概要 ノード種別を ELIMINATED に変更する．

機能説明 OpenJIT Peephole最適化機能により，Optimize ノード種別をELIMINATEDに変更し，ＲＴＬが不必要になったことを示す．

入力データ なし．

出力データ なし．

処理方式 op_and_tagのノード種別にELIMINATEDを設定する．

エラー処理 なし．

- final void setD(int type, int val)

  機能概要 Ｄオペランドを設定する．

  機能説明 Ｄオペランドを設定する．

  入力データ type(Ｄオペランドの種別), val(Ｄオペランドの値)

  出力データ なし．

  処理方式 op_and_tagのＤオペランドの種別に，typeを設定する．dvalにvalを設定する．

  エラー処理 なし．

- final void setL(int type, int val)

  機能概要 Ｌオペランドを設定する．

  機能説明 Ｌオペランドを設定する．

  入力データ type(Ｌオペランドの種別), val(Ｌオペランドの値)

  出力データ なし．

  処理方式 op_and_tagのＬオペランドの種別に，typeを設定する．lvalにvalを設定する．

  エラー処理 なし．

- final void setR(int type, int val)

  機能概要 Ｒオペランドを設定する．

機能説明 Ｒオペランドを設定する．

入力データ type(Ｒオペランドの種別), val(Ｒオペランドの値)

出力データ なし．

処理方式 op_and_tag のＲオペランドの種別に，type を設定する．rval に val を設定する．

エラー処理 なし．

- final void setConst(int val)

機能概要 定数を設定するＲＴＬに変換する．

機能説明 constant folding 最適化により，定数を転送するＲＴＬに変換する．

入力データ val(定数)

出力データ なし．

処理方式 op_and_tag のノード種別を MOVE に変更し，Ｒオペランドの種別を整数の定数にする．rval に val を設定する．Ｄオペランドに関する情報は保持する．

エラー処理 なし．

- final void referD()

機能概要 Ｄオペランドが参照されているという情報を付け加える．

機能説明 Ｄオペランドが参照されているという情報を付け加える．OpenJIT Peephole 最適化機能において，ＲＴＬを削除してもよいかどうか判断するために用いられる．

入力データ なし．

出力データ なし．

処理方式 Ｄオペランドの種別に TAG_REF を設定する．

エラー処理 なし．

- final boolean isNotReferD()

機能概要 Ｄオペランドが参照されているかどうか調べる．

機能説明 Ｄオペランドが参照されているかどうか調べる．OpenJIT Peephole 最適化機能において，ＲＴＬを削除してもよいかどうか判断するために用いられる．

入力データ なし．

出力データ 参照されていれば true, そうでなければ false を返す．

処理方式 Ｄオペランドの種別に TAG_REF を設定されているか調べる．

エラー処理 なし．

- final boolean isSTORE()

機能概要 ノード種別がストア命令かどうか調べる．

機能説明 ノード種別がストア命令かどうか調べる．ストア命令に関してはＤオペランドの扱いが異なり，Ｄオペランドに値が書き込まれることはなく，読み出しに使われる．これを判断するためにこのメソッドは用いられる．

入力データ なし．

出力データ ストア命令であれば true, そうでなければ false を返す．

処理方式 op_and_tag からノード種別を取り出し，それが ST,STB,STH のどれかであることを調べる．

エラー処理 なし．

- static final String tag2String(int type, int val)

機能概要 オペランドの種別を String に変換する．

機能説明 オペランドの種別を String に変換する．バックエンド中間コードの表示やＲＴＬの表示といったデバッグ用に用いられる．

入力データ type(オペランドの種別), val(オペランドの値)．

出力データ 変換された文字列 (String)．

処理方式 種別を切り出し，StringBuffer buf に append する．

エラー処理 なし．

- public String toString()

  機能概要 String に変換する．

  機能説明 バックエンド中間コードまたはＲＴＬを String に変換する．デバッグ時に用いる．

  入力データ なし．

  出力データ 変換された String．

  処理方式 StringBuffer buf を用意し，static 変数 name を使ってノード種別の String 表現を得て buf に append し，tag2String メソッドを用いて，オペランドを buf に append していき，buf をリターンする．

  エラー処理 なし．

4.1.5 OpenJIT.BBinfo クラス

(1) 概要

基本ブロック情報を表現する．

(2) フィールド (変数)

- BBinfo chain

  BBinfo クラスをつないで，連結リスト構造を作るために用いる．

- int beg

  基本ブロックの開始アドレス (バイトコード上の PC) の値．

- int end

  基本ブロックの終了アドレス (バイトコード上の PC) の値．

(3) コンストラクタ

なし．

(4) メソッド

なし．

4.1.6 OpenJIT.Compile クラス

(1) 概要

OpenJIT コンパイラ基盤機能と OpenJIT ネイティブコード変換機能の中核をなすクラス .

(2) フィールド (変数)

- private int methodblock

  JDK の内部データである methodblock 構造体へのポインタ

- private int constantpool

  JDK の内部データである constantpool へのポインタ

- Class clazz

  現在コンパイル中のメソッドのクラス

- byte signature[]

  現在コンパイル中のメソッドの引数や型

- public byte bytecode[]

  バイトコード列

- public ExceptionHandler exceptionHandler[]

  例外ハンドラに関する情報

- protected int access

  アクセス情報

- protected int nlocals

  ローカル変数の数

- protected int maxstack

  スタックの最大値

- protected int args_size

  引数の数

- protected BCinfo bcinfo[]

  バイトコードに関する情報

- int flags

  コンパイルのための各種フラグ

- protected int code_area

  ネイティブコードの領域の先頭ポインタ

- private int code_size

  ネイティブコードのサイズ

- protected int native_pc

  ネイティブコードのプログラムカウンタ

- private static Select enable

  コンパイルするかどうかの選択

- private static Select disable

  コンパイルするかどうかの選択

- static boolean optimize_stack

  Peeohole 最適化をするかどうかのフラグ

- static boolean optimize_handle

  ハンドル最適化をするかどうかのフラグ

- static boolean debug

  デバッグのためのフラグ

- static boolean debugNative

  デバッグのためのフラグ

- private static final int CONSTANT_POOL_ENTRY_TYPEMASK

  constant pool の型を取り出すためのマスク値

- private static final int CONSTANT_POOL_ENTRY_RESOLVED

  constant pool が解消済みかどうかを調べるためのマスク

(3) コンストラクタ

なし．

(4) メソッド

- public boolean compile()

  機能概要 フロントエンドによって最適化されたバイトコード列から SPARC のネイティブコードを出力する．

  機能説明 バイトコードを入力とし，SPARC ネイティブコードを出力する．OpenJIT 中間コード変換機能，OpenJIT RTL 変換機能，OpenJIT Peephole 最適化機能を制御する．

  入力データ フロントエンド変換後のバイトコード

  出力データ SPARC プロセッサのネイティブコード

  処理方式 クラス変数 enable が null でなければ，enable.match を this を引数として呼び出す．この結果が true でなければ，false をリターンする．

  クラス変数 disable が null でなければ，disable.match を this を引数として呼び出す．この結果が true でなければ，false をリターンする．

  OpenJIT.parseBytecode を呼び出して，バイトコードをバックエンド中間コードに変換する．

  OpenJIT.convertRTL を呼び出して，バックエンド中間コードを RTL(オペランド，基本ブロック情報) に変換する．

  もし optimize_stack の値が true であれば，OpenJIT.optimizeRTL を呼び出して，RTL(オペランド，基本ブロック情報) を RTL(最適化) に変換する．

OpenJIT.regAlloc を呼び出して，仮想レジスタと物理レジスタ番号の対応表を作成する．

OpenJIT.genativeCode を呼び出して，RTL(最適化) を SPARC ネイティブコードに変換する．

エラー処理

- public final int pc2uchar(int pc)

  機能概要 バイトコード列の中の任意の符合なし 8bit データを読み出す．

  機能説明 バイトコード列の中の任意の 1 バイトを符合なし 8bit 整数として読み出し，32bit の整数にして返す．

  主に OpenJIT 中間コード変換機能において，JVM の命令を取り出すために用いられる．

  入力データ pc (バイトコード列のプログラムカウンタ)

  出力データ 32bit の整数

  処理方式 配列 bytecode から，pc をインデックスとして値を取り出す．その値を int にキャストして，0xff と論理積を取った値をリターンする．

  エラー処理 なし．

- public final int pc2signedchar(int pc)

  機能概要 バイトコード列の中の任意の符合付き 8bit データを読み出す．

  機能説明 バイトコード列の中の任意の 1 バイトを符合付き 8bit 整数として読み出し，32bit の整数にしてリターンする．

  主に OpenJIT 中間コード変換機能において，JVM の命令のオペランドを取り出すために用いられる．

  入力データ pc (バイトコード列のプログラムカウンタ)

  出力データ 32bit の整数

  処理方式 配列 bytecode から，pc をインデックスとして値を取り出す．その値を int にキャストしてリターンする．

エラー処理

- public final int pc2signedshort(int pc)

  機能概要 バイトコード列の中の任意の符合付き 16bit データを読み出す．

  機能説明 バイトコード列の中の任意の 2 バイトを符合付き 16bit 整数として読み出し，32bit の整数にしてリターンする．

  主に OpenJIT 中間コード変換機能において，JVM の命令のオペランドを取り出すために用いられる．

  入力データ pc (バイトコード列のプログラムカウンタ)

  出力データ 32bit の整数

  処理方式 配列 bytecode から，pc と pc+1 をインデックスとして値を取り出す．その値を 32bit の整数値に変換してリターンする．

  エラー処理 なし．

- public final int pc2signedlong(int pc)

  機能概要 バイトコード列の中の任意の符合付き 32bit データを読み出す．

  機能説明 バイトコード列の中の任意の 4 バイトを符合付き 32bit 整数として読み出しリターンする．

  主に OpenJIT 中間コード変換機能において，JVM の命令のオペランドを取り出すために用いられる．

  入力データ pc (バイトコード列のプログラムカウンタ)

  出力データ 32bit の整数

  処理方式 配列 bytecode から，pc,pc+1,pc+2,pc+3 をインデックスとして値を取り出す．その値を 32bit の整数値に変換してリターンする．

  エラー処理 なし．

- public final int pc2ushort(int pc)

  機能概要 バイトコード列の中の任意の符合なし 16bit データを読み出す．

機能説明 バイトコード列の中の任意の 2 バイトを符合なし 16bit 整数として読み出し，32bit の整数にしてリターンする．

主に OpenJIT 中間コード変換機能において，JVM の命令のオペランドを取り出すために用いられる．

入力データ pc (バイトコード列のプログラムカウンタ)

出力データ 32bit の整数

処理方式 配列 bytecode から，pc と pc+1 をインデックスとして値を取り出す．その値を 32bit の整数値に変換してリターンする．

エラー処理 なし．

- public final native int ConstantPoolValue(int index)

機能概要 JVM が扱う定数値を読み出す．

機能説明 constant pool の任意の値を読み出す．

主に OpenJIT 中間コード変換機能において，JVM の命令が扱う定数値を取り出すために用いられる．

入力データ index (constant pool のインデックス)

出力データ 定数値 (32bit)

処理方式 このメソッドは C 言語で記述されている．

JDK の内部データにアクセスし，その値を返す．

エラー処理 index が constant pool の境界外を指している場合，OpenJIT .Error("Constant pool index out of boundary") を発生する．

- public final int ConstantPoolAddress(int index)

機能概要 constant pool の値を格納しているアドレスを返す．

機能説明 constant pool の値を格納しているアドレスを返す．

OpenJIT 中間コード変換機能において，JVM の命令が扱う定数値を格納しているアドレスを得るために用いられる．

入力データ index (constant pool のインデックス)

出力データ 定数格納アドレス (32bit)

処理方式 index を 4 倍して変数 constant_pool の値に加えてリターンする．

エラー処理 index が constant pool の境界外を指している場合，OpenJIT .Error("Constant pool index out of boundary") を発生する．

- private final native int ConstantPoolTypeTable(int index)

  機能概要 constant pool の種別，解消済みかを返す．

  機能説明 index で示された constant pool の種別と解消済みかどうかを返す．

  ConstantPoolType メソッド，ConstantPoolTypeResolved メソッドから呼び出される．

  入力データ index (constant pool のインデックス)

  出力データ 32bit データ (うち 1bit は解消済みかどうか，7bit が種別)

  処理方式 このメソッドは C 言語で記述されている．

  JDK の内部データにアクセスし，その値を返す．

  エラー処理 index が constant pool の境界外を指している場合，OpenJIT .Error("Constant pool index out of boundary") を発生する．

- public final int ConstantPoolType(int index)

  機能概要 constant pool の種別を返す．

  機能説明 index で示される constant pool の種別を返す．

  OpenJIT 中間コード変換機能において，JVM の命令が扱う定数の種別を得るために用いられる．

  入力データ index (constant pool のインデックス)

  出力データ constant pool の種別 (int)．とりうる値は 1 ~ 12 で，順に次の種別を示す．

  1. Utf8
  2. Unicode
  3. Integeer

4. Float
5. Long
6. Double
7. Class
8. String
9. Fieldref
10. Methodref
11. InterfaceMethodref
12. NameAndType

処理方式 ConstantPoolTypeTable メソッドを呼び出して，その結果の値に対し CONSTANT_POOL_ENTRY_TYPEMASK (0x7f) と論理積をとりリターンする．

エラー処理 なし．

- public final boolean ConstantPoolTypeResolved(int index)

機能概要 constant pool が解消済みかどうかを調べる．

機能説明 index で示される constant pool の値が解消済みかどうかを調べる．

OpenJIT 中間コード変換機能において，JVM の命令が扱う定数が解消済みかどうかを調べるために用いられる．

入力データ index (constant pool のインデックス)

出力データ 解消済みであれば true，そうでなければ false を返す．

処理方式 ConstantPoolTypeTable メソッドを呼び出して，その結果の値に対し CONSTANT_POOL_ENTRY_RESOLVED (0x7f) と論理積をとる．この値が 0 であれば，false を返し，0 でなければ true を返す．

エラー処理 なし．

- public final native String ConstantPoolClass(int index)

機能概要 constant pool のクラス名を返す．

機能説明 index で示される constant pool のクラス名を返す．

クラスを参照する JVM の命令のクラス名を取り出すために用いる．

入力データ index (constant pool のインデックス)

出力データ クラス名 (java.lang.String)．ただし，クラスのセパレータは'.' ではなく'/' であることに注意すること．

処理方式 このメソッドは C 言語で記述されている．

index が constant pool の境界内であることを確かめる．JDK の内部データにアクセスし，constant pool を解析してクラス名 (char *) を得る．その後，これを java.lang.String クラスに変換してリターンする．

エラー処理 index が constant pool の境界外を指している場合，OpenJIT .Error("Constant pool index out of boundary") を発生する．

また，index で指される constant pool の種別が Methodref, InterfaceMethodRef, FieldRef でない場合は，OpenJIT.CompilerError("invalid constant pool reference") を発生する．

- public final native String ConstantPoolName(int index)

機能概要 constant pool のメソッド名，フィールド名を返す．

機能説明 index で示される constant pool のメソッド名，フィールド名を返す．

OpenJIT 中間コード変換機能において，JVM の命令が参照するメソッド名，フィールド名を得るために用いられる．

入力データ index (constant pool のインデックス)

出力データ メソッド名またはフィールド名 (java.lang.String)．

処理方式 このメソッドは C 言語で記述されている．

index が constant pool の境界内であることを確かめる．JDK の内部データにアクセスし，constant pool を解析してメソッド名またはフィールド名 (char *) を得る．その後，これを java.lang.String クラスに変換してリターンする．

エラー処理 index が constant pool の境界外を指している場合，OpenJIT .Error("Constant pool index out of boundary") を発生する．

また，indexで指されるconstant poolの種別がMethodref, InterfaceMethodRef, FieldRefでない場合は，OpenJIT.CompilerError(”invalid constant pool reference”)を発生する．

- public final native byte[] ConstantPoolMethodDescriptor(int index)

  機能概要 constant poolのメソッドのSignature(引数と返り値の型)を返す．

  機能説明 indexで示されるconstant poolのメソッドのSignature(引数と返り値の型)を返す．

  OpenJIT中間コード変換機能において，JVMの命令が参照するメソッドの引数と返り値の型を得るために用いられる．

  入力データ index (constant poolのインデックス)

  出力データ メソッドの引数と返り値の型 (byte [])

  処理方式 indexがconstant poolの境界内であることを確かめる．JDKの内部データにアクセスし，constant poolを解析してメソッド名またはフィールド名 (char *)を得る．これを，新しくbyteの配列を用意してコピーし，そのbyte配列を返す．

  エラー処理 indexがconstant poolの境界外を指している場合，OpenJIT .Error(”Constant pool index out of boundary”)を発生する．

  また，indexで指されるconstant poolの種別がMethodref, InterfaceMethodRef, FieldRefでない場合は，OpenJIT.CompilerError(”invalid constant pool reference”)を発生する．

- public final native byte ConstantPoolFieldDescriptor(int index)

  機能概要 constant poolのフィールドの型を返す．

  機能説明 indexで示されるconstant poolのフィールドの型を返す．

  OpenJIT中間コード変換機能において，JVMの命令が参照するフィールドの型を得るために用いられる．

  入力データ index (constant poolのインデックス)

出力データ フィールドの型 (byte)．

返り値は次のどれかの値である．

array('['), byte('B'), char('C'), class('L'), float('F'), double('D'), int('I'), long('J'), short('S'), void('V'), boolean('Z')

処理方式

エラー処理 index が constant pool の境界外を指している場合，OpenJIT .Error("Constant pool index out of boundary") を発生する．

- public final native int ConstantPoolFieldOffset(int index)

機能概要 constant pool のフィールドのオフセットを返す．

機能説明 index で示される constant pool のフィールドのオフセットを返す．

OpenJIT 中間コード変換機能において，JVM の命令が参照するフィールドのオフセットを得るために用いる．このオフセットは JDK の内部データ構造において，オブジェクトへのポインタから，フィールドの値をアクセスする際に必要となる．

入力データ index (constant pool のインデックス)

出力データ オフセット値 (int)

処理方式 index が constant pool の境界内であることを確かめる．JDK の内部データにアクセスし，constant pool を解析してフィールドのオフセットを得て返す．

エラー処理 index が constant pool の境界外を指している場合，OpenJIT .Error("Constant pool index out of boundary") を発生する．

- public final native int ConstantPoolFieldAddress(int index)

機能概要 constant pool のフィールドのアドレスを返す．

機能説明 index で示される constant pool の static フィールドのアドレスを返す．

OpenJIT 中間コード変換機能において，JVM の命令が参照する static フィールドの絶対アドレスを得るために用いる．これは JDK の内部データ構造において，クラス変数の値をアクセスする際に必要となる．

入力データ index (constant pool のインデックス)

出力データ 絶対アドレス (int)

処理方式 index が constant pool の境界内であることを確かめる．JDK の内部データにアクセスし，constant pool を解析してフィールドのオフセットを得て返す．

エラー処理 index が constant pool の境界外を指している場合，OpenJIT .Error("Constant pool index out of boundary") を発生する．

- public final native void NativeCodeAlloc(int size)

機能概要 生成するネイティブコードのための領域を確保する．

機能説明 指定されたサイズのメモリ領域を確保する．

OpenJIT SPARC プロセサコード出力から呼び出される．

入力データ size (生成するネイティブコードのサイズ)

出力データ 領域の先頭アドレスを変数 code_area に格納する．変数 code_size に size を格納する．

処理方式 JDK に用意された関数 sysMalloc を呼び出し，領域を確保する．

エラー処理 領域が確保できないとき，OpenJIT .Error("Native code out of memory") を発生する．

- public final native int NativeCodeReAlloc(int size)

機能概要 生成するネイティブコードのための領域を再割り当てする．

機能説明 指定されたサイズのメモリ領域を再割り当てする．

OpenJIT SPARC プロセサコード出力から呼び出され，余分に確保した領域を捨てる．

入力データ size (生成したネイティブコードのサイズ)，変数 code_area (既に確保した領域の先頭アドレス)

出力データ 再割り当てした領域の先頭アドレスを変数 code_area に格納する．変数 code_size に size を格納する．

処理方式 JDK に用意された関数 sysRealloc を呼び出し，領域を再割り当てする．

エラー処理 再割り当てする領域が確保できないとき，OpenJIT .Error(”Native code out of memory”) を発生する．

- public final void genCode(int code)

  機能概要 ネイティブコードを 1 命令生成する．

  機能説明 ネイティブコードに割り当てられた領域に順番に命令を 1 命令だけ書き込む．

  OpenJIT SPARC プロセサコード出力から呼び出され，命令を順に生成する．

  入力データ code (生成する命令)，変数 native_pc

  出力データ 変数 native_pc

  処理方式 setNativeCode メソッドを呼び出し命令を生成した後，変数 native_pc を 1 増やす．

  エラー処理 なし．

- public final native void setNativeCode(int pc, int code)

  機能概要 ネイティブコードを書き込む．

  機能説明 ネイティブコードに割り当てられた領域の任意の番地に命令を 1 命令だけ書き込む．

  genCode メソッド，OpenJIT SPARC プロセサコード出力から呼び出され，命令を書き込む．

  入力データ pc (アドレス，アドレスは絶対番地ではなく，ネイティブコード領域の先頭を 0 番地とする相対アドレス．また，byte 単位でなく word 単位)，code(生成する命令, 32bit)

  出力データ なし．

  処理方式 このメソッドは C 言語で記述されている．変数 code_area を参照し，指定された番地に命令を書き込む．

エラー処理 pc が割り当てられた領域外のとき，OpenJIT .Error(”Native code out of boundary”) を発生する．

- public final native int getNativeCode(int pc)

  機能概要 ネイティブコードを読み出す．

  機能説明 ネイティブコードに割り当てられた領域の任意の番地に命令を 1 命令だけ読み出す．

  OpenJIT SPARC プロセサコード出力から呼び出される．

  入力データ pc (アドレス，アドレスは絶対番地ではなく，ネイティブコード領域の先頭を 0 番地とする相対アドレス．また，byte 単位でなく word 単位)

  出力データ 生成された命令 (32bit)．

  処理方式 このメソッドは C 言語で記述されている．変数 code_area を参照し，指定された番地の命令を読み出す．

  エラー処理 pc が割り当てられた領域外のとき，OpenJIT .Error(”Native code out of boundary”) を発生する．

- public final native String methodName()

  機能概要 コンパイルしているメソッドの名前を返す．

  機能説明 現在コンパイルしているメソッドの名前を返す．

  デバッグやコンパイル時のエラー表示のために用いられる．

  入力データ なし．

  出力データ メソッド名 (java.lang.String)

  処理方式 このメソッドは C 言語で記述されている．

  JDK の内部データにアクセスし，現在コンパイル中のメソッド名 (char *) を得る．それを java.lang.String クラスに変換してリターンする．

  エラー処理 なし．

- public final String className()

機能概要 コンパイルしているクラスの名前を返す．

機能説明 現在コンパイルしているクラスの名前を返す．

デバッグやコンパイル時のエラー表示のために用いられる．

入力データ 変数 clazz

出力データ クラス名 (java.lang.String)

処理方式 変数 clazz に対し，java.lang.Class.getName メソッドを適応し，クラス名を得てリターンする．

エラー処理 なし．

- public final String signatureName()

機能概要 コンパイルしているメソッドのSignature((引数と返り値の型) を返す．

機能説明 現在コンパイルしているメソッドの Signature((引数と返り値の型) を返す．

デバッグやコンパイル時のエラー表示のために用いられる．

入力データ 変数 signature

出力データ Signature(java.lang.String)

処理方式 変数signature(byte配列) をjava.lang.String に変換してリターンする．

エラー処理 なし．

- public final String toString()

機能概要 コンパイルしているメソッドのクラス，メソッド，型を返す．

機能説明 java.lang.Object クラスのメソッドを Override する．

現在コンパイルしているメソッドのクラス，メソッド，型を一つのjava.lang.String にまとめてリターンする．

デバッグやコンパイル時のエラー表示のために用いられる．

入力データ なし．

出力データ クラス，メソッド，型を表す java.lang.String

処理方式 className メソッド, methodName メソッド, signatureName メソッドを呼び出した結果を連結してリターンする．

エラー処理 なし．

- public final String opcName(int pc)

機能概要 バイトコードの命令を JVM の命令文字列に変換する．

機能説明 与えられたバイトコード列の任意の命令を pc をインデックスとして取り出し，JVM の命令文字列 (java.lang.String) に変換する．

デバッグやコンパイル時のエラー表示のために用いられる．

入力データ pc (バイトコード列のプログラムカウンタ)

出力データ JVM の命令を表す java.lang.String

処理方式 pc とバイトコード列から JVM の命令を取出し，JDK に用意されている sun.tools.java.RunTimeConstnats クラスの opcNames を利用して java.lang.String に変換してリターンする．

エラー処理 なし．

4.1.7 OpenJIT.Select クラス

(1) 概要

あるメソッドが与えらたとき，そのメソッドをコンパイルするかどうか選択するために利用するクラス．

(2) フィールド (変数)

- Select next

  OpenJIT .Select クラスのオブジェクトの連結リストを構成する．

- String clazz

  クラス名

- String method

  メソッド名

- String signature

  Signature 名

(3) コンストラクタ

- public Select(String prop)

  機能概要 プロパティから Select オブジェクトの連結リストを生成する．

  機能説明 JDK を起動したときのオプションで与えられるプロパティをもとに，Select オブジェクトの連結リストを生成する．

  入力データ String(プロパティ)．

  プロパティは” クラス名．メソッド名 (Signature 名)” を”:” で区切ったものとして与えられる．

  出力データ なし．

処理方式 prop.indexOf(':') を呼び出すことにより，':' が含まれるか調べる．含まれていれば，prop.substring メソッドにより prop の文字列を分割する．

同様にして，'(' が含まれるか，'.' が含まれるか調べ，文字列を分割し clazz, method, signature のフィールドをセットする．

':' が含まれていた場合は，next フィールドに新しく Select オブジェトを生成してセットし，残りの文字列についてこの処理を繰り返す．

エラー処理 なし．

(4) メソッド

- boolean match(Compile mb)

機能概要 与えれた Compile オブジェクトが，プロパティで指定されたパターンとマッチするか調べる．

機能説明 連結リストである this をたどりながら，与えれた Compile オブジェクトのクラス名，メソッド名，Singature 名とマッチするかどうか調べ，一致するものがあれば true を返し，一致するものがみつからなければ false を返す．

入力データ mb(Compile オブジェクト)

出力データ マッチすれば true，それ以外のときは false を返す．

処理方式 mb.className(), mb.methodName(), mb.signatureName() を呼び出し，Compile オブジェクトのクラス名，メソッド名，Signature 名を得る．

this の連結リストをたどり，一致するものを見つけたら true をリターンする．

最終的に見つからなければ false をリターンする．

エラー処理 なし．

4.1.8 OpenJIT.ParseBytecode クラス

(1) 概要

このクラスは OpenJIT 中間コード変換機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

なし．

(4) メソッド

- private final int ldc(LinkedList ll, int index)

  機能概要 定数のロードに関するバックエンド中間コード列を生成する．

  機能説明 constant pool の型にしたがって，Java スタックに定数を読み込むバックエンド中間コード列を生成する．

  入力データ ll(バックエンド中間コード列の連結リスト), index(constant pool の index)

  出力データ Java スタックに詰まれる定数の大きさを返す．型が double や long のときは 2 ，それ以外は 1 を返す．

  処理方式 OpenJIT .Compile .ConstantPoolType を呼び出し，型を調べる．型に従って，バックエンド中間コードを生成し，ll につなぐ．

  エラー処理 型が以下の値以外のとき， OpenJIT .CompilerError (”invalid ldc constants”) を発生する．

  - CONSTANT_INT
  - CONSTANT_FLOAT
  - CONSTANT_STRING

- CONSTANT_DOUBLE
- CONSTANG_LONG

- private final ILnode array_intro(LinkedList ll, int aref, int index, int size)

  機能概要 配列のアクセスに関するバックエンド中間コード列を生成する．

  機能説明 Java の配列のアクセスには，境界チェックとハンドルの読み出しが必要である．この配列の境界チェックと，配列の要素のアドレス計算，および配列のハンドル読み出しのためのバックエンド中間コード列を生成する．

  また，変数 flags に FLAG_HAS_EXCEPTION を設定する．

  入力データ ll(バックエンド中間コード列の連結リスト), aref(配列の object を示すスタックの深さ), index(配列のインデックスを示すスタックの深さ)，size(配列の要素の大きさ．log_2(バイト数) − 1)

  出力データ 生成したバックエンド中間コード列の最後のバックエンド中間コード

  処理方式 変数 flags に FLAG_HAS_EXCEPTION をセットする．

  配列の境界チェックを行うバックエンド中間コードを生成する．もし，size が 0 より大きければアドレス計算を行うバックエンド中間コードを生成する．ハンドルを読み出すバックエンド中間コードを生成し，そのバックエンド中間コードを返す．

  エラー処理 なし．

- private final int branch(int pc, int offset)

  機能概要 分岐命令に関するバイトコード情報を設定し，分岐先の pc を返す．

  機能説明 分岐命令の分岐先のバイトコードがバイトコード情報を持っているかどうか調べ，持っていない場合は新しくバイトコード情報を生成する．そして，BCinfo.lFLAG_BRACNH_TARGET の情報をセットし，分岐先の pc を返す．

  入力データ pc(分岐元のバイトコードの pc), offset(相対分岐のオフセット)

出力データ 分岐先の pc

処理方式 分岐先の pc を計算し，target_pc にセットする．offset が正のとき，バイトコード情報 bcinfo[target_pc] はまだ設定されていないので，新しく BCinfo object を生成し，配列 bcinfo[target_pc] にセットする．bcinfo[target_pc] の flag に BCinfo.FLAG_BRANCH_TARGET をセットする．target_pc をリターンする．

エラー処理 もし，bcinfo[target_pc] が null のときは，最適化されたバイトコードの途中に飛込む分岐命令として，OpenJIT .CompilerError ("jumped into optimized sequence?") を発生する．

- private final void callMethod(BCinfo bc, int pc, int invoker, int invoker_resolve, int stack, boolean is_static)

機能概要 メソッド呼び出しを行うバックエンド中間コード列を生成する．

機能説明 constant pool からメソッドの signature を読み出し，これにしたがってメソッドの呼び出しを行うときの，引数を設定するバックエンド中間コード列を生成する．

呼び出すメソッドがすでにロードされているかどうか調べ，それに従ってメソッド呼び出しのためのランタイムルーチンを選択し，そのランタイムルーチンを呼び出すバックエンド中間コードを生成する．

その後，メソッドの返り値の型に従って，Java のスタックにメソッドの返り値をセットするバックエンド中間コードを生成する．

入力データ bc(バイトコード情報), pc(バイトコードの pc), invoker(ロードされていないときのランタイムルーチン), invoker_resolve(ロード済みのときのランタイムルーチン)，stack(未使用), is_static(static メソッドかどうか)

出力データ なし．

処理方式 pc+1 から constant pool の index を取り出し，OpenJIT .Compile .ConstantPoolMethodDescriptor を呼び出して sig にセットする．

変数 flags に FLAG_HAS_INVOKE をセットする．

もし，static メソッドでない場合は，第一引数レジスタに Java スタックの 0 をセットするバックエンド中間コードを生成する．

sig を前から順に調べて，引数をセットするバックエンド中間コードを生成していくと共に，引数の数 (args_size) をカウントする．

sig を調べ終わったのち，これまで生成したバックエンド中間コードの補正を行う．オペランドの Java スタックの深さから arigs_size を引いていく．それと同時に，生成したバックエンド中間コード列の連結リストを逆向きにつけ直す．これにより，最終的に生成するネイティブコードの効率が良くなる．

最後に，メソッドの帰り値の型を調べて，Java スタックにメソッドの返り値をセットするバックエンド中間コードを生成する．

さらにバイトコード情報 bc の stack_delta に，args_size からこのメソッド呼び出しによって増減するスタックのサイズの差分をセットする．

エラー処理 読み出したメソッドの signature に解析できない値が存在した場合，OpenJIT .CompilerError (”invalid signature”) を発生する．

- private final boolean optim_neg(int pc)

機能概要 boolean 値の否定を行うバイトコード列のパターンマッチを行う．

機能説明 バックエンド中間コードの最適化を行うために，以下に示すような boolean 値の否定を行うバイトコード列のパターンマッチを行う．

```
0: opc_ifeq, 0, 7,
3: opc_iconst_0,
4: opc_goto, 0, 4,
7: opc_iconst_1
```

入力データ pc(バイトコードの pc)

出力データ パターンにマッチすれば true，マッチしなければ false を返す．

処理方式 与えれた pc から残っているバイトコードが 8 以上あることを確認する．なければ，false を返す．

バイトコードを読み出し，パターンマッチをする．異なれば，false を返す．

もし，このバイトコードの途中に分岐して来ることがあれば，false を返す．

それ以外のとき，true を返す．

エラー処理 なし．

- private final boolean optim_lcmp(int pc)

  機能概要 JVM の long 値の比較命令の最適化を行う．

  機能説明 以下に示すような long 値の比較命令を最適化したバックエンド中間コード列を生成する．

  ```
  0: opc_lcmp
  1: op_if** xxx
  4: ...
  ```

  入力データ pc(バイトコードの pc)

  出力データ 最適化できなかった場合 false を返し，最適化できた場合 true を返す．

  処理方式 与えれた pc から残っているバイトコードが 4 以上あることを確認する．なければ，false を返す．

  もし，このバイトコードの途中に分岐して来ることがあれば，false を返す．

  pc+1 のバイトコード命令を調べる．この命令が

  - opc_ifeq のとき
    上位 32bit の比較，異なれば次のバイトコードへ分岐，下位 32bit の比較，同じであれば xxx へ分岐，
    というバックエンド中間コードを生成
  - opc_ifne のとき
    上位 32bit の比較，異なれば xxx へ分岐，下位 32bit の比較，異なれば xxx へ分岐，
    というバックエンド中間コードを生成

– opc_iflt のとき
上位 32bit の比較，小さければ xxx へ分岐，等しくなければ次へ分岐，
下位 32bit の比較，符合なしで小さければ xxx へ分岐，
というバックエンド中間コードを生成

– opc_ifge のとき
上位 32bit の比較，大きければ xxx へ分岐，等しくなければ次へ分岐，
下位 32bit の比較，等しいか符合なしで多きければ xxx へ分岐，
というバックエンド中間コードを生成

– opc_ifgt のとき
上位 32bit の比較，大きければ xxx へ分岐，等しくなければ次へ分岐，
下位 32bit の比較，符合なしで多きければ xxx へ分岐，
というバックエンド中間コードを生成

– opc_ifle のとき
上位 32bit の比較，小さければ xxx へ分岐，等しくなければ次へ分岐，
下位 32bit の比較，等しいか符合なしで小さければ xxx へ分岐，
というバックエンド中間コードを生成

– それ以外のとき
false をリターン

として，true をリターンする．

エラー処理 なし．

- private final ILnode optim_fcmp(int pc, boolean is_cmpg)

機能概要 JVM の float 値の比較または double 値の比較命令の最適化を行う．

機能説明 以下に示すような float 値または double 値の比較命令を最適化したバックエンド中間コード列を生成する．

```
0: opc_fcmp, opc_dcmp
1: op_if** xxx
4: ...
```

入力データ pc(バイトコードの pc), is_cmpg(NaN の比較のときに大きいと同じとみなす)

出力データ 最適化できなかった場合 null を返し，最適化できた場合は生成したバックエンド中間コードを返す．

処理方式 与えれた pc から残っているバイトコードが 4 以上あることを確認する．なければ，false を返す．

もし，このバイトコードの途中に分岐して来ることがあれば，false を返す．

pc+1 のバイトコード命令を調べる．この命令が opc_ifeq, opc_ifne, opc_iflt, opc_ifge, opc_ifgt, opc_ifle のいずれかであれば，それに応じた分岐を行うバックエンド中間コードを生成し，リターンする．

それ以外のときは null をリターンする．

エラー処理 なし．

- final void parseBytecode() throws CompilerError

機能概要 バイトコードからバックエンド中間コードを生成する．

機能説明 フロントエンドシステムの出力であるバイトコードを順に読み込みながら，バックエンド中間コードを生成する．

バイトコードに特有のパターンをみつけ，最適化を施す．constant pool の値を展開し，バックエンド中間コードのオペランドにする．

また，バイトコード情報も生成する．

入力データ bytecode(バイトコード)

出力データ bcinfo(バイトコード情報，バックエンド中間コード)

処理方式 pc の値を 0 から bytecode.length まで順に増やしながら，バイトコード命令を調べていく．バイトコードは可変長の命令であるため，pc の値はバイトコードの命令長に応じて増やしていく．bcinfo[pc] に新しく BCinfo オブジェクトを生成し，バイトコード情報を加えていく．

opcode はバイトコード命令，bc はバイトコード情報である．

opcode の命令に応じて，bc.stack_delta の値をセットし，pc の値を増やす．

生成するバックエンド中間コードは bc.next に連結リストしてつなげられる．

エラー処理 未知のバイトコード命令を見つけた場合，OpenJIT .CompilerError ("detect unknown bytecode") を発生する．

4.1.9 OpenJIT.ConvertRTL クラス

(1) 概要

このクラスは OpenJIT RTL 変換機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

なし．

(4) メソッド

- private final void extractBB()

  機能概要 基本ブロック情報をバイトコード情報に付加する．

  機能説明 基本ブロック情報 (BBinfo クラス) を生成し，バイトコード情報 (BCinfo クラス) に付加する．

  入力データ bcinfo(バイトコード情報)

  出力データ bcinfo(基本ブロック情報を付加したバイトコード情報)

  処理方式 BBinfo クラスである bb に null を設定する．

  pc に 0 を設定し，pc が bcinfo.length より小さい間，pc を 1 増やしながら次のことを行う．つまり bcinfo をバイトコード列の長さだけ走査する．

  bc に bcinfo[pc] をセットする．

  もし，bc が null であれば，それはバイトコード命令の切目ではなくオペランドであるから，次を調べる．

  もし bc.flag に BCinfo .FLAG_BRANCH_TARGET が立っていなかったら，それは基本ブロックの切目ではないので，次を調べる．

もし，bb が null でなければ，bb.end に pc をセット，bb.chain に BBinfo オブジェクトを生成してセット，さらに bb にセットする．bb が null のときは, bb に BBinfo オブジェクトを生成してセットする．

bc.flag に BCinfo .FLAG_BEGIN_BASIC_BLOCK のフラグを立てる．これによりこのバイトコードが基本ブロックの開始命令であることが定義される．bc.bb に bb をセットし，bb.beg に pc をセットする．

以上の bcinfo の走査が終わり，bb が null でないときは bb.end に bcinfo.length をセットする．

エラー処理 なし．

- private final void convBB(BBinfo bb, byte stackTag[]) throws CompilerError

機能概要 基本ブロック中のバックエンド中間コードを RTL に変換する．

機能説明 基本ブロックにおいて，Java のスタックの深さの動きを解析し，オペランドがスタック表現であるバックエンド中間コードから，レジスタ表現である RTL に変換する．また，バックエンド中間コードにおいてオペランドの型の確定していないものの型を調べ型を設定する．

入力データ bb(基本ブロック情報)，stackTag(Java のスタックに積まれたものの型)，bcinfo(基本ブロック情報を付加したバイトコード情報)

出力データ bcinfo(ＲＴＬ)

処理方式 この convBB メソッドは再帰的に呼び出される．

bb で与えられた基本ブロックの先頭のバイトコード情報を取り出し，bc にセットする．

bc.flag に BCinfo.FLAG_CTRLFLOW_DONE が立っていたら，既に RTL に変換済みであるので，単にリターンする．

bc.flag に BCinfo.FLAG_CTRLFLOW_DONE のフラグを立てる．これによりこの基本ブロックは処理済みという情報が付加される．

new_stack_size に bc.stack_size をセットする．new_stack_size はスタックの深さを表す変数である．

pc に bb.beg をセット (基本ブロックの先頭) して，pc が bb.end(基本ブロックの終わり) より小さい間，pc を 1 増やしながら次のことを行う．

bc に bcinfo[pc] をセットする．もし bc が null であれば pc がバイトコード命令の切目ではないので，次の pc を調べる．

cur_stack_size(バイトコード命令処理前のスタックの深さ) に new_stack_size をセットする． new_stack_size(バイトコード命令処理後のスタックの深さ) に cur_stack_size + bc.stack_delta をセットする． bc.stack_size に cur_stack_size をセットする．ここでバイトコード情報のスタックの深さが確定する．

次に，このバイトコードが展開されたバックエンド中間コードを調べる．

バックエンド中間コードに分岐命令が含まれていたとき，分岐先の基本ブロック情報のフラグを調べる．

もし，フラグに BCinfo .FLAG_CTRLFLOW_DONE が立っていなかったら， convBB メソッドを再起呼び出しする．呼び出す前に分岐先のバイトコード情報のスタックの深さを設定しておく．もしバイトコード命令が jsr(サブルーチン分岐) であれば，スタックの深さを 1 増やす．これは jsr 命令が PC の値をプッシュしてサブルーチンを呼び出すためである． convBB の再起呼び出しには，分岐先の基本ブロック情報と， stackTag を複製したものを引数として渡す．

もしフラグが立っていなくて，無条件分岐命令であれば処理は終了であり，リターンする．

バックエンド中間コードに転送命令が含まれていて，Ｌオペランドの値が - 1 のとき，これは型が未確定を意味する．このとき， stackTag を調べて型を取り出し，Ｄオペランドの型に設定する．

バックエンド中間コードに CALL 命令が含まれていれば， flags(コンパイルフラグ) に FLAG_HAS_CALL のフラグを立てる．

その他のバックエンド中間コードについては，Ｄ，Ｌ，Ｒオペランドを調べ，ＲＴＬへ変換を行う．もしオペランドがスタックであれば，レジスタの番号を設定する．つまりスタックの深さに cur_stack_size と nlocals (ローカル変数の個数) を加える．

さらにＤオペランドについて，バックエンド中間コードがストア命令でなければ stackTag にＤオペランドの型を設定する．

以上を基本ブロック内の pc について繰り返す．

エラー処理 なし．

- final void convertRTL() throws CompilerError

  機能概要 バックエンド中間コードをＲＴＬに変換する．

  機能説明 バイトコード情報の分岐情報をもとに，基本ブロック情報を生成し，バイトコード情報に付加する．

  バイトコードの先頭から，処理の流れを解析してスタックの動きをシミュレートすることで，バックエンド中間コードからＲＴＬへの変換を行う．

  バイトコードの例外処理部には処理の流れが届かないので，これについてもフロー解析を行う．

  入力データ exceptionHandler(例外テーブル) bcinfo(バイトコード情報，バックエンド中間コード)

  出力データ bcinfo(バイトコード情報，基本ブロック情報，ＲＴＬ)

  処理方式 convBB メソッドを呼び出すために stackTag を初期化する．

  例外テーブルの全ての例外ハンドラを調べて，例外処理の開始アドレスのバイトコード情報のフラグに BCinfo .FLAG_BRANCH_TARGET と BCfino . FLAG_EXCEPTION_HANDLER のフラグをセットする．また，開始アドレスのバイトコード情報のバックエンド中間コードに pc の値をスタックにセットする転送命令を挿入する．

  extractBB メソッドを呼び出し，基本ブロック情報を生成する．

  0 番地のバイトコードの基本ブロック情報を取り出し，convBB メソッドを呼び出す．これによりプログラムの先頭からのフロー解析とＲＴＬへの変換を行う．

  例外テーブルの全ての例外処理の開始アドレスについても，基本ブロック情報を取り出し，convBB メソッドを呼び出す．これで，バイトコード全てのＲＴＬ変換が終了する．

  エラー処理 処理終了後，全ての基本ブロック情報を調べて，フロー解析を行っていない基本ブロックが存在していたら，OpenJIT .CompilerError (”found dead code”) を発生する．

4.1.10 OpenJIT.OptimizeRTL クラス

このクラスは OpenJIT Peephole 変換機能を実現する．

(1) 概要

(2) フィールド (変数)

- ILnode thisHandle

  生成するネイティブコードにおいて，オブジェクトのフィールドを参照するときにハンドルを介した間接参照を行うオーバヘッドをなくすために使用する．

- static final boolean debug_optimize

  デバッグ用のクラス変数．これを true にしておけば，デバッグのためのメッセージを出力する．

(3) コンストラクタ

なし．

(4) メソッド

- private static void eliminateIL(ILnode il)

  機能概要 与えられたＲＴＬを不要なものとして削除する．

  機能説明 与えられたＲＴＬが削除可能かどうか調べ，不必要であれば削除する．ただし，ＲＴＬオブジェクトを消去するのではなく，ＲＴＬのノード種別を ELIMINATED に変更する．

  入力データ il(ＲＴＬ)

  出力データ なし．

  処理方式 il が null でなく，かつ il のノード種別が転送命令でかつ，Ｄオペランドがスタック変数であり，Ｄオペランドが参照されていないとき，il.remove() を呼び出して il のノード種別を ELIMIATED に変更 (RTL の削除) する．

エラー処理 なし．

- private static int srcDefTime(ILnode il, Var var[], int callTime)

  機能概要 与えられたＲＴＬのソースオペランドが定義された時刻を返す．

  機能説明 与えられたＲＴＬのソースオペランドが定義された時刻を返す．この時刻はＲＴＬのオペランドを置き換えてよいかどうか判断するために用いられる．

  入力データ il(ＲＴＬ), var(Var オブジェクトの配列：オペランドレジスタ列) callTime(最後に関数呼び出しを行った時刻)

  出力データ 定義された時刻

  処理方式 与えられたＲＴＬが転送命令のとき，その転送元のオペランド (レジスタ) の値がセットされた時刻を返す．ＲＴＬが転送命令でないときは，最大値 0x7fffffff を返す．転送元のオペランドが定数のときは最小値 0 を返す．また，転送元が関数呼び出しの戻り値を表すレジスタのときは，最後に関数呼び出しを行った時刻を返す．

  エラー処理 なし．

- private static boolean moveUpSV(ILnode moveIL, Var var[], int callTime)

  機能概要 スタック変数からローカル変数に転送するＲＴＬを上方に移動する最適化を行う．

  機能説明 与えれたＲＴＬが上方に移動可能かどうか調べ，可能であれば移動先のＲＴＬのＤオペランドを変更し，ＲＴＬを削除する．たとえば

  ```
  stack1 + stack2 -> stack3
  ...
  stack3 -> local0
  ```

  のようなＲＴＬの列があったときに，これを

  ```
  stack1 + stack2 -> local1
  ...
  (削除)
  ```

に変更して最適化を行う．

入力データ moveIL(ＲＴＬ), var(Varオブジェクトの配列：オペランドレジスタ列) callTime(最後に関数呼び出しを行った時刻)

出力データ 最適化できたときtrueを返し，できなかったときfalseを返す．

処理方式 moveILのＲオペランド値をインデックスにしてvarを参照することで，転送元のVarオブジェクトを得てsrc_varにセットする．さらにsrc_varのilフィールドを参照し，転送元のレジスタに値を設定しているＲＴＬを得てsrcILにセットする．

もしsrcILがnullであれば，移動先が見つからないとしてfalseをリターンする．

srcIL.isNotReferD()を呼び，移動先のＲＴＬのＤオペランドが参照されていないか確認する．もし参照されていれば，移動先のＲＴＬのＤオペランドを変更することはできないので，falseをリターンする．

さらに，関数呼び出しにまたがって，ＲＴＬを移動することを避けるため，src_var.defTimeとcallTimeを比較し小さければ，falseをリターンする．

moveILのＤオペランドをインデックスにしてvarを参照し，転送先のVarオブジェクトを得てdst_varにセットする．

転送先のレジスタが使われている時刻が，転送元のレジスタの値が設定された時刻より小さくないことを確認する．src_var.defTimeとdst_var.usrTimeを比較して小さければfalseをリターンする．

また移動先のＲＴＬが転送命令である場合は，ここでは最適化を行わないためfalseをリターンする．

この時点で，ＲＴＬの移動が可能であるとみなし，ＲＴＬの移動・削除を行う．転送元のレジスタの値を設定するＲＴＬのＤオペランドを転送先のレジスタに付け変える．これはsrcIL.setDメソッドを呼び出すことで行う．

そして，転送先レジスタに関するVarオブジェクトのフィールド値を変更する．

moeIL.remove()を呼び出すことによりＲＴＬを削除し，trueをリターンする．

エラー処理 なし．

- private static boolean moveUpSP(ILnode moveIL, Var var[], int callTime, int paramUseTime)

  機能概要 スタック変数からパラメタ変数に転送するＲＴＬを上方に移動する最適化を行う．

  機能説明 与えれたＲＴＬが上方に移動可能かどうか調べ，可能であれば移動先のＲＴＬのＤオペランドを変更し，ＲＴＬを削除する．たとえば

  ```
  stack1 + stack2 -> stack3
  ...
  stack3 -> param1
  ```

  のようなＲＴＬの列があったときに，これを

  ```
  stack1 + stack2 -> param1
  ...
  (削除)
  ```

  に変更して最適化を行う．

  入力データ moveIL(ＲＴＬ), var(Var オブジェクトの配列：オペランドレジスタ列) callTime(最後に関数呼び出しを行った時刻) paramUseTIme(パラメタ変数を最後に使った時刻)

  出力データ 最適化できたとき true を返し，できなかったとき false を返す．

  処理方式 moveIL のＲオペランド値をインデックスにして var を参照することで，転送元の Var オブジェクトを得て src_var にセットする．さらに src_var の il フィールドを参照し，転送元のレジスタに値を設定しているＲＴＬを得て srcIL にセットする．

  もし srcIL が null であれば，移動先が見つからないとして false をリターンする．

  srcIL.isNotReferD() を呼び，移動先のＲＴＬのＤオペランドが参照されていないか確認する．もし参照されていれば，移動先のＲＴＬのＤオペランドを変更することはできないので，false をリターンする．

さらに，関数呼び出しにまたがって，ＲＴＬを移動することを避けるため，src_var.defTime と callTime を比較し小さければ，false をリターンする．

転送元レジスタの定義時刻が paramUseTime より小さい場合に最適化を行わなず，false をリターンする．

また移動先のＲＴＬが転送命令である場合は，ここでは最適化を行わないため false をリターンする．

この時点で，ＲＴＬの移動が可能であるとみなし，ＲＴＬの移動・削除を行う．転送元のレジスタの値を設定するＲＴＬのＤオペランドを転送先のレジスタに付け変える．これは srcIL.setD メソッドを呼び出すことで行う．

moeIL.remove() を呼び出すことによりＲＴＬを削除し，true をリターンする．

エラー処理 なし．

- final void optimizeRTL() throws CompilerError

  機能概要 ＲＴＬを最適化する．

  機能説明 Peephole 最適化を行い冗長なＲＴＬを除去する．また，constant folding 最適化も行う．最適化を行う単位は基本ブロックであり，基本ブロック間にまたがった最適化は行わない．

  入力データ bcinfo(バイトコード情報，基本ブロック情報，ＲＴＬ)

  出力データ bcinfo(バイトコード情報，基本ブロック情報，最適化されたＲＴＬ)

  処理方式 ＲＴＬのオペランドとして使われるレジスタのための，Var オブジェクトを生成し，var にセットする．

  pc の値を 0 から bcinfo.length まで順に増やしながら，バイトコード情報を調べていく．bc い bcinfo[pc] をセットする．bc が null のときは命令の切目ではないため，次を調べる．

  もし，bc.flag に BCinfo .FLAG_BEGIN_BASIC_BLOCK のフラグが立っていれば，基本ブロックの開始であるので，現在時刻である curTime，最終関数呼び出し時刻である callTime, パラメタ使用時刻 paramUseTime を 0 にリセットする．var.reset() を呼びレジスタに関する情報もリセットする．

バイトコード情報 bc に付随するＲＴＬの連結リストを走査する．ＲＴＬノードを一つ処理する度に現在時刻 curTime を１増やす．

ＲＴＬノードが CALL 命令のとき，callTime に curTime をセットし，このＲＴＬノードの処理を中断し次のＲＴＬを処理する．

ＲＴＬノードが転送命令 (MOVE) のとき，ＲＴＬの上方移動の最適化を行う．ただし，バイトコードが例外処理の try ブロックの中にあるときはこの最適化を行わない．転送がスタック変数からローカル変数のときは moveUpSV() を呼び出して最適化を行う．転送がスタック変数からパラメタ変数のときは moveUpSP() を呼び出して最適化を行う．最適化できた場合はこのＲＴＬノードの処理を中断し次のＲＴＬを処理する．

ＲＴＬのＬオペランドの置き換えを行う．これはＬオペランドが転送命令によって設定されているとき，Ｌオペランドをその転送命令のソースレジスタで置き換える処理である．たとえば次のような置き換えである．

```
var1 -> stack1
...
stack1 + stack2 -> var2
```

のようなＲＴＬの列があったときに，これを

```
var1 -> stack1（削除）
...
var1 + stack2 -> var2
```

に変更する．

置き換えが可能であるための条件は，Ｌオペランドがスタック変数でかつ，そのスタック変数の値を設定するＲＴＬが var に登録されている場合である．さらに，置き換えるべきレジスタが定義された時刻が，転送命令より後で，再び書き換えられていないことを確かめなければならない．

これには，まずＬオペランドであるスタック変数に値を設定するＲＴＬを srcIL にセットする．そして srcDefTime(srcIL,var,callTime) を呼び出して定義された時刻を求め，Ｌオペランドが定義された時刻より大きくないことを確認する．大きい場合は置き換え不可能とし，srcIL.referD() を呼び出

し srcIL が示すＲＴＬが削除できないようにする．置き換え可能であった場合，ＲＴＬのＬオペランドを srcIL のＤオペランドで置き換える．

Ｌオペランドがスタック変数かローカル変数であれば，レジスタ var の参照時刻を curTime に更新する．Ｌオペランドがパラメタ変数であれば，paramUseTIme を curTime に更新する．

以上Ｌオペランドの置き換え処理と同様の処理をＲオペランドについても行う．

ＬオペランドとＲオペランドの置き換え処理の後，constant folding 最適化を施す．もし両方のオペランドが定数であれば，ここで予め演算を施してＲＴＬを定数の転送命令に変更する．constant folding を行うＲＴＬのノード種別は DIV0CHK, SLL, ADD, AND, OR である．consant folding の処理が行えないノード種別の場合はエラーとする．

また，特殊なケースとして，Ｒオペランドが定数 0 との AND は，ＲＴＬノードを定数 0 の転送命令に変更する．Ｒオペランドが定数 -1(ALL 1) との AND は単純な転送命令に変更する．

コンパイルしているメソッドが static メソッドでないときは，ハンドル参照除去の最適化を行う．ローカル変数の 0 番は Java において this を示し，バイトコードにおいてアクセスの頻度が高い．さらに this は null pointer でないことが保証されており，この変数については例外処理も必要ない．そこで，ローカル変数の 0 番をベースにしてロードを行う命令をメソッドの入り口に移動して，ロードした結果を特定のマシンレジスタに割り付けておき，後で同じロード命令が現れたら，ロードを行う代りにこのマシンレジスタを使うようにして最適化を行う．

具体的な処理としては，ノード種別が LD でＬオペランドのタグがローカル変数でＬオペランドの値が 0 であるＲＴＬノードを検知し，このＲＴＬノードを this のハンドルに割り付けたマシンレジスタから転送する命令に変更する．

ＲＴＬノードがストア命令のとき，Ｄオペランドに対してもＬオペランドとＲオペランドに施した置き換え処理と同様の処理を行う．

ＲＴＬノードがストア命令でないとき，この時点でＤオペランドが示すレジスタの値は新しく更新される．ここから先の時刻，前の値が参照される

ことはない．したがって，このＤオペランドを前の時刻に設定したＲＴＬについて，削除してよいかを調べる．これは，Ｄオペランドを前の時刻に設定したＲＴＬについてeliminateIL()を呼び出すことによって行う．もしＤオペランドがdoubleの場合は，2つのレジスタを消費しているので，さらにもう一つのＲＴＬについてもeliminateIL()を呼び出し削除する．

Ｄオペランドが示すレジスタについてのVarオブジェクトのフィールドdefTimeをcurTimeに設定し，ilを設定する．

以上のことを1つのバイトコード命令が展開されたＲＴＬノードの連結リストに対して施した後，冗長な命令の削除を行う．JVMはこのバイトコードを実行することにより，bc.stack_deltaで示されるスタックが捨てられる．この捨てられるスタックの変数について，これを定義したＲＴＬについてeliminateIL()を呼び出すことによって，冗長な命令を削除する．

エラー処理　ＲＴＬノードのＬオペランドとＲオペランドが定数で，constant folding の最適化が処理不可能なノード種別を検知したとき，OpenJIT .CompilerError ("constant folding") を発生する．

また，ハンドル参照除去を行う最適化において，ローカル変数の0番をベースにしてロードするＲＴＬのＲオペランドが定数の0でないものを見つけた場合は，OpenJIT .CompilerError ("handle ???") を発生する．

4.1.11 OpenJIT.Var クラス

(1) 概要

OpenJIT Peephole 最適化機能の実現のために使われるクラス．OpenJIT .OptimizeRTL クラスからのみ参照される．ＲＴＬのオペランドのレジスタの定義，参照関係の情報を保持するために用いる．

(2) フィールド (変数)

- int defTime

  レジスタに値が代入された時刻

- int useTime

  レジスタの値が参照された時刻

- ILnode il

  レジスタに値が代入されたときの代入を行うＲＴＬへのポインタ

(3) コンストラクタ

- void Var()

  機能概要 Var オブジェクトを生成し，初期化する．

  機能説明 Var オブジェクトを生成し，初期化する．

  入力データ なし．

  出力データ なし．

  処理方式 フィールドを全て 0 に初期化する．il には null をセットする．

  エラー処理 なし．

(4) メソッド

- final void reset()

機能概要 フィールドの値をリセットする．

機能説明 Peephole 最適化は基本ブロック単位であるため，新しい基本ブロックについえ Peephole 最適化を行う前にリセットする．

入力データ なし．

出力データ なし．

処理方式 フィールドを全て 0 に初期化する．il には null をセットする．

エラー処理 なし．

4.1.12 OpenJIT.RegAlloc クラス

(1) 概要

このクラスは OpenJIT レジスタ割付機能を実現する．

(2) フィールド (変数)

- static final boolean assert

  デバッグのためのクラス変数．

- RegAlloc. VirReg virRegs[]

  仮想レジスタ管理情報

- RegAlloc. PhyRegs iTmp

  物理整数レジスタ管理情報

- RegAlloc. PhyRegs fTmp

  物理浮動小数レジスタ管理情報

- static final int REG_G0

  SPARC のマシンレジスタ番号 0(%g0)，常に値は 0

- static final int REG_G1

  SPARC のマシンレジスタ番号 1(%g1)

- static final int REG_G2

  SPARC のマシンレジスタ番号 2(%g2)

- static final int REG_G3

  SPARC のマシンレジスタ番号 3(%g3)

- static final int REG_O0

  SPARC のマシンレジスタ番号 8(%o0)

- static final int REG_O1

  SPARC のマシンレジスタ番号 9(%o1)

- static final int REG_SP

  SPARC のマシンレジスタ番号 14(%sp)，スタックポインタ．

- static final int REG_O7

  SPARC のマシンレジスタ番号 14(%o7)

- static final int REG_L0

  SPARC のマシンレジスタ番号 16(%l0)

- static final int REG_L7

  SPARC のマシンレジスタ番号 23(%l7)

- static final int REG_I0

  SPARC のマシンレジスタ番号 23(%i0)

- static final int REG_I1

  SPARC のマシンレジスタ番号 23(%i1)

- static final int REG_FP

  SPARC のマシンレジスタ番号 23(%fp)，フレームポインタ．

- static final int REG_I7

  SPARC のマシンレジスタ番号 31(%i7)，リターンアドレス．

- static final int REG_F0

  SPARC の浮動小数点マシンレジスタ番号 0(%f0)

- static final int REG_F1

  SPARC の浮動小数点マシンレジスタ番号 1(%f1)

- static final int REG_INVALID

  無効なレジスタ番号を意味する．

- static final int PARAM_REG_SIZE

  引数の受け渡しに使うレジスタの個数．SPARC は 6 個

- static final int PARAM_REG_OFFSET

  引数の受け渡しに使うレジスタのセーブ領域を表すためのスタックポインタからのオフセット値．SPARC では 68．

- private int param_tag

  引数レジスタのタグを保持する．regAllocFix メソッドで利用．

- private int param_num

  引数レジスタの番号を保持する．regAllocFix メソッドで利用．

- private int param_tmp

  引数レジスタのために一時的に割り当てられた物理整数レジスタ番号を保持する．regAllocFix メソッドで利用．

(3) コンストラクタ

なし．

(4) メソッド

- abstract void emitLD(int tag, int dst, int baseReg, int offset)

  機能概要 ネイティブコードのロード命令を生成する．

  機能説明 物理レジスタのフィルを行う命令を生成するときに用いる．ネイティブコードのロード命令を生成する．

  入力データ tag(レジスタのタグ)，dst(物理レジスタ番号，どのレジスタにデータをロードするか)，baseReg(物理整数レジスタ番号，ベースレジスタ)，offset(ベースレジスタからのオフセット値)

  出力データ ネイティブコードのロード命令．

  処理方式 抽象メソッドであり，OpenJIT .Sparc クラスにて実装される．

エラー処理 なし．

- abstract void emitST(int tag, int src, int baseReg, int offset)

  機能概要 ネイティブコードのストア命令を生成する．

  機能説明 物理レジスタのスピルを行う命令を生成するときに用いる．ネイティブコードのストア命令を生成する．

  入力データ tag(レジスタのタグ)，src(物理レジスタ番号，どのレジスタのデータをストアするか)，baseReg(物理整数レジスタ番号，ベースレジスタ)，offset(ベースレジスタからのオフセット値)

  出力データ ネイティブコードのストア命令．

  処理方式 抽象メソッドであり，OpenJIT .Sparc クラスにて実装される．

  エラー処理 なし．

- final void loadVirReg(int dst, int tag, int virNum)

  機能概要 ネイティブコードのロード命令を生成する．

  機能説明 マシンレジスタに割り付けられなかった仮想レジスタの値を特定の物理レジスタにロードする命令を生成する．

  入力データ dst(物理レジスタ番号，どのレジスタにデータをロードするか)，tag(ロードするレジスタのタグ)，virNum(仮想レジスタ番号)，virRegs(仮想レジスタ管理情報)

  出力データ ネイティブコードのロード命令．

  処理方式 virRegs[virNum] の情報を使ってベースレジスタとオフセットを取り出し，emitLD() を呼び出す．

  エラー処理 なし．

- final void storeVirReg(int src, int tag, int virNum)

  機能概要 ネイティブコードのストア命令を生成する．

  機能説明 特定の物理レジスタをマシンレジスタに割り付けられない仮想レジスタのセーブ領域にストアする命令を生成する．

入力データ src(物理レジスタ番号，どのレジスタにデータをロードするか)，tag(ロードするレジスタのタグ)，virNum(仮想レジスタ番号)，virRegs(仮想レジスタ管理情報)

出力データ ネイティブコードのストア命令．

処理方式 virRegs[virNum] の情報を使ってベースレジスタとオフセットを取り出し，emitST() を呼び出す．

エラー処理 なし．

- final void regAlloc()

機能概要 仮想レジスタ番号に対して，物理レジスタ番号を割り付ける．

機能説明 仮想レジスタ番号に対して，物理レジスタ番号を割り付ける．コンパイル対象のメソッドが leaf 関数かどうか調べ，割り付け方法を変える．仮想レジスタ管理情報と物理整数レジスタ管理情報，物理浮動小数レジスタ管理情報を生成する．

入力データ args_size(引数の数)，maxstack(スタックの最大値)，nlocals(ローカル変数の数), flags(コンパイルフラグ)，thisHandle(ハンドル参照の最適化)

出力データ virRegs(仮想レジスタ管理情報)，iTmp(物理整数レジスタ管理情報)，fTmp(物理浮動小数レジスタ管理情報)

処理方式 メソッドが CALL 命令，例外，JSR 命令を含むかどうか flags を調べる．もし含んでいなくて，使用している仮想レジスタの個数が 6 個以下であれば，leaf 最適化を行う．leaf 最適化とは SPARC プロセッサのレジスタウィンドウの機能を使わないようにして，ネイティブコードを小さくする最適化である．

物理整数レジスタ管理情報を生成し，iTmp にセットする．物理整数レジスタ管理情報を生成するとき，leaf 最適化をするかどうかによって，使えるレジスタが異なる．leaf 最適化をするときは %o0 ~ %o5, %g1 ~ %g4 の物理レジスタが使用可能であり，PhyRegs オブジェクトを生成するときに 0x00003f1e をパラメタとして渡す．leaf 最適化をしないときは %i0 ~ %i5, %l0 ~ %l7, %g1 ~ %g4 の物理レジスタを使用可能とし，PhyRegs オブジェクトを生成するときに 0x3fff001e をパラメタとして渡す．

物理浮動小数レジスタ管理情報を生成し，fTmp にセットする．SPARC プロセッサでは，double の値を保持するために 2 つのレジスタをペアで使うため，物理浮動小数レジスタとして偶数番のレジスタを使用可能とする．また，%f0 レジスタは関数のリターン値として用いるため，レジスタの割り付けを行わない．したがって，PhyRegs オブジェクトを生成するときに，0x55555554 をパラメタとして渡す．

次に，仮想レジスタ管理情報を生成し，virRegs にセットする．nlocals + maxstack + 1 だけの個数の仮想レジスタを用意する．ハンドル参照の最適化を行うときはさらに仮想レジスタを余分に 1 つ用意する．

virRegs の各要素に仮想レジスタの個数だけ VirReg オブジェクトを生成してセットしていく．生成するときのベースレジスタとして，leaf 最適化を行うときはスタックポインタ，leaf 最適化を行わない場合はフレームポインタを使うようにする．

その後，仮想レジスタを物理レジスタを固定的に割り付ける．割り付けるのは整数レジスタだけで，浮動小数レジスタについては固定的な割り付けは行わない．

割り付けは先ずコンパイルするメソッドの引数を優先的に割り付ける．leaf 最適化を行うかどうかによって，引数に使われる物理レジスタが %i0 ~ %i5 と %o0 ~ %o5 と異なる．

次にハンドル最適化を行うときは，これを優先的に割り付ける．

最後に残った仮想レジスタについて，レジスタ番号が小さい順に物理レジスタを割り付ける．

エラー処理 なし．

- final int regNum(int tag, int val)

  機能概要 仮想レジスタ番号から物理レジスタ番号を得る．

  機能説明 仮想レジスタ番号から物理レジスタ番号を得る．

  入力データ tag(仮想レジスタのタグ)，val(仮想レジスタ番号)，virRegs(仮想レジスタ情報)

  出力データ 物理レジスタ番号．

処理方式 tag が定数のとき，val が 0 であれば REG_G0 をリターンする．それ以外のときは REG_INVALID をリターンする．

tag がパラメタ変数のとき，tag の型が整数であれば (REG_O0 + val) をリターンし，そうでなければ val をリターンする．

virRegs[val] を参照し，整数かどうか調べ，割り付けられている物理レジスタ番号をリターンする．

エラー処理 なし．

- final void saveRegs()

機能概要 一時的に割り付けられた物理レジスタをセーブするネイティブコードを生成する．

機能説明 固定的に割り付けられていない物理レジスタのうち，生成したネイティブコードで一時レジスタとして使われたものをメモリ上に待避するネイティブコードを生成する．

入力データ iTmp(物理整数レジスタ管理情報)，fTmp(物理浮動小数レジスタ管理情報)

出力データ セーブするネイティブコード

処理方式 iTmp.save() と fTmp.save() を呼び出す．

エラー処理 なし．

- final void discardStackRegs(int stack_from, int stack_to)

機能概要 仮想レジスタのうち，参照されなくなったレジスタを指定する．

機能説明 saveRegs() では，ネイティブコード上で今後使われることのないレジスタまでセーブを行う．それを避けるために，参照されなくなったレジスタを指定し，不必要なセーブをするネイティブコードが生成されるのを防ぐ．

入力データ stack_from(スタックの深さ)，stack_to(スタックの深さ)，virRegs(仮想レジスタ情報)

出力データ なし．

処理方式 stack_from から stack_to までスタックの深さに対応する仮想レジスタについて，nouse() を呼び出す．

エラー処理 なし．

- final void invalidateRegs()

  機能概要 一時的に割り付けられた物理レジスタを無効化する．

  機能説明 生成するネイティブコード上で，call 命令の直後などはそれまでに割り付けられた物理レジスタの値が無効になる．これを処理するために，一時的に割り付けられた物理レジスタの無効化を行う．

  入力データ iTmp(物理整数レジスタ管理情報)，fTmp(物理浮動小数レジスタ管理情報)

  出力データ なし．

  処理方式 iTmp.invalidateI(), fTmp.invalidateF() を呼び出す．

  エラー処理 なし．

- final int regAllocTmp()

  機能概要 一時的に整数物理レジスタを割り付ける．

  機能説明 生成するネイティブコード上で一時的に使用するレジスタが必要なときに呼ばれ，整数物理レジスタを 1 つ確保してその物理レジスタ番号を返す．

  入力データ iTmp(物理整数レジスタ管理情報)

  出力データ 整数物理レジスタ番号

  処理方式 iTmp.getTmpReg(TYPE_INT, null) を呼び出す．

  エラー処理 なし．

- final void freeTmpReg(int phyNum)

  機能概要 一時的に割り付けられた物理レジスタを解放する．

  機能説明 regAllocTmp() によって割り付けられた整数物理レジスタを解放して，他の用途に使えるようにする．

入力データ phyNum(物理整数レジスタ番号)，iTmp(物理整数レジスタ管理情報)

出力データ なし．

処理方式 iTmp.freeTmpReg() を呼び出す．

エラー処理 なし．

- final int regAllocRead(int tag, int virNum)

機能概要 仮想レジスタ番号から物理レジスタ番号を得る．

機能説明 仮想レジスタの値を読むネイティブコードを生成するために，仮想レジスタ番号から物理レジスタ番号を得る．物理レジスタが割り付けられていない場合は，一時レジスタを確保して，その物理レジスタに対して，値をフィルするネイティブコードを生成する．

入力データ tag(仮想レジスタのタグ)，virNum(仮想レジスタ番号)，virRegs(仮想レジスタ管理情報)，iTmp(物理整数レジスタ管理情報)，fTmp(物理浮動小数レジスタ管理情報)

出力データ 物理レジスタ番号，ネイティブコードのロード命令．

処理方式 tag がパラメタ変数のとき，tag の型が整数であれば (virNum + REG_O0) を返し，そうでなければ virNum を返す．

tag が定数のとき，virNum が 0 であれば REG_G0 を返し，そうでなければエラーを発生する．

仮想レジスタ情報 vr オブジェクトに virRegs[virNum] をセットする．

tag の型が整数のとき，物理レジスタが割り付けられていれば iTmp.read(物理レジスタ番号) を呼び出し，物理レジスタ番号をリターンする．割り付けられていなければ，iTmp.getTmpReg(tag,vr) を呼び出し，一時レジスタを確保する．この一時レジスタの物理レジスタ番号を tr にセットする．vr.assign(TYPE_INT, tr) を呼び，仮想レジスタに物理レジスタ番号を割り当てる．vr.fill(tr) を呼び，値をフィルするネイティブコードのロード命令を生成する．iTmp.read(tr) を呼び，tr をリターンする．

tag の型が浮動小数のとき，物理レジスタが割り付けられていれば fTmp.read(物理レジスタ番号) を呼び出し，物理レジスタ番号をリターンする．割り

付けられていなければ，fTmp.getTmpReg(tag,vr) を呼び出し，一時レジスタを確保する．この一時レジスタの物理レジスタ番号を tr にセットする． vr.assign(tag, tr) を呼び，仮想レジスタに物理レジスタ番号を割り当てる． vr.fill(tr) を呼び，値をフィルするネイティブコードのロード命令を生成する． fTmp.read(tr) を呼び， tr をリターンする．

エラー処理 tagがパラメタ変数で， virNumがRETF,ARG0,ARG1以外のとき， OpenJIT .CompilerError("physical register") を発生する．

tag が定数で virNum が 0 でないとき， OpenJIT .CompilerError("internal error") を発生する．

- final int regAllocWrite(int tag, int virNum)

機能概要 仮想レジスタ番号から物理レジスタ番号を得る．

機能説明 仮想レジスタに値を書き込むネイティブコードを生成するために，仮想レジスタ番号から物理レジスタ番号を得る．物理レジスタが割り付けられていない場合は，一時レジスタを確保して，その物理レジスタ番号をリターンする．

入力データ tag(仮想レジスタのタグ)， virNum(仮想レジスタ番号)，

virRegs(仮想レジスタ管理情報)， iTmp(物理整数レジスタ管理情報)， fTmp(物理浮動小数レジスタ管理情報)

出力データ 物理レジスタ番号， param_tag, param_num, param_tmp

処理方式 tag がパラメタ変数のとき， tag の型が整数，かつ virNum(パラメタ変数の番号) が PARAM_REG_SIZE より小さければ (virNum + REG_O0) を返す． tag の型が浮動小数だったり， virNum が大きいときはレジスタによるパラメタの受け渡しが行えない．このため，一時レジスタを確保してその物理レジスタ番号をリターンする．このとき確保した物理レジスタに関する情報を， param_tag, param_num, param_tmp に記憶しておく．

tag が定数のとき， virNum が 0 であればREG_G0 を返し，そうでなければエラーを発生する．

仮想レジスタ情報 vr オブジェクトに virRegs[virNum] をセットする．

tag の型が整数のとき，物理レジスタが割り付けられていれば iTmp.write (物理レジスタ番号) を呼び出し，物理レジスタ番号をリターンする．割り付けられていなければ，iTmp.getTmpReg(tag,vr) を呼び出し，一時レジスタを確保する．この一時レジスタの物理レジスタ番号を tr にセットする．vr.assign(TYPE_INT, tr) を呼び，仮想レジスタに物理レジスタ番号を割り当てる． vr.lastTag に tag をセットし，整数と浮動小数のどちらが有効なのかを記憶する． iTmp.write(tr) を呼び， tr をリターンする．

tagの型が浮動小数のとき，物理レジスタが割り付けられていればfTmp.write (物理レジスタ番号) を呼び出し，物理レジスタ番号をリターンする．割り付けられていなければ， fTmp.getTmpReg(tag,vr) を呼び出し，一時レジスタを確保する．この一時レジスタの物理レジスタ番号を tr にセットする． vr.assign(tag, tr) を呼び，仮想レジスタに物理レジスタ番号を割り当てる． vr.lastTag に tag をセットし，整数と浮動小数のどちらが有効なのかを記憶する． fTmp.write(tr) を呼び， tr をリターンする．

エラー処理 tagがパラメタ変数で， virNumがRETF,ARG0,ARG1以外のとき， OpenJIT .CompilerError(”physical register”) を発生する．

tag が定数で virNum が 0 でないとき， OpenJIT .CompilerError(”internal error”) を発生する．

- final void regAllocFix()

機能概要 パラメタの受け渡しのために割り付けられた一時レジスタの値を，メモリにストアするネイティブコードを生成する．

また，一つのＲＴＬノードからネイティブコードの生成が終わった後のレジスタ割り付けの後始末を行う．

機能説明 パラメタの数が多くて，レジスタによるパラメタの受け渡しができなかったり，浮動小数のパラメタを受け渡すときは，ネイティブコードのスタック上のメモリにパラメタをストアする必要がある．このためのネイティブコードを生成する．また， SPARC プロセッサでは浮動小数の場合も整数レジスタを使ってパラメタの受け渡しを行うため，浮動小数のデータを整数レジスタに転送するためのネイティブコードを生成する．

ＲＴＬノードからネイティブコードを生成するときに regAllocWrite() を呼ぶが，一時レジスタに値を設定するネイティブコードを生成した後で，パラメタをストアするコードを生成しなければならない．そのため regAllocWrite メソッドでこの処理を行うことはできず，このメソッドを別のメソッドとして用意した．

また，このメソッドは一つのＲＴＬノードからネイティブコードの生成が終わった後，かならず呼ばれる，ここで一時的に割り付けられた物理レジスタの解放を行う．

入力データ param_tag, param_num, param_tmp, iTmp， fTmp

出力データ パラメタをストアするネイティブコード．パラメタをロードするネイティブコード．

処理方式 param_tag が 0 であれば，パラメタのために一時レジスタを使っていないので，何も処理せずリターンする．

パラメタをセーブするスタックポインタからのオフセットを計算して offset にセットする．

param_tag の型に従って以下の処理を行う．

- TYPE_INT のとき
  emit_st(param_tag, param_tmp, REG_SP, offset) を呼ぶ．
- TYPE_FLOAT のとき
  emit_st(param_tag, param_tmp, REG_SP, offset) を呼び， param_num が PARAM_REG_SIZE より小さければ，パラメタレジスタに値を設定するコードを生成するため， emitLD() を呼ぶ．
- TYPE_DOUBLE のとき
  emit_st(param_tag, param_tmp, REG_SP, offset) を呼び， param_num が PARAM_REG_SIZE より小さければ，パラメタレジスタに値を設定するコードを生成するため，emitLD() を呼ぶ．さらに (param_num+1) が PARAM_REG_SIZE より小さければ，パラメタレジスタに値を設定するコードを生成するため， emitLD() を呼ぶ．

次のＲＴＬノードの処理に備えて， param_tag を 0 に初期化する．

iTmp.unlock(), ftmp.unlock() を呼び出し，一時的に割り付けられた物理レジスタの解放を行い，物理レジスタの再利用を可能にする．

エラー処理 なし．

4.1.13 OpenJIT.RegAlloc$VirReg クラス

(1) 概要

OpenJIT.RegAlloc クラスのクラス内クラスであり，仮想レジスタを管理する情報を扱う．一つの仮想レジスタに対して，このクラスの一つのオブジェクトが割り当てられる．

(2) フィールド (変数)

- int regNumI

  整数物理レジスタ番号．

- int regNumF

  浮動小数物理レジスタ番号．

- int lastTag

  現在の型．regNumI と regNumF のどちらが有効か

- int baseReg

  スピル / フィルにおけるベースレジスタ

- int offset

  スピル / フィルにおけるオフセット

(3) コンストラクタ

- public VirReg(int baseReg, int offset)

  機能概要 VirReg クラスのオブジェクトを生成し，初期化を行う．

  機能説明 VirReg クラスのオブジェクトを生成し，物理レジスタを未割り当て状態にし，baseReg と offset を設定する．

  入力データ baseReg(ベースレジスタ)，offset(オフセット)

  出力データ なし．

処理方式 regNumI, regNumF に REG_INVALID を設定し，baseReg, offset フィールドに値を設定する．

エラー処理 なし．

(4) メソッド

- final VirReg assign(int tag, int phyNum)

  機能概要 物理レジスタ番号を割り当てる．

  機能説明 tag の型に応じて，物理レジスタ番号を割り当てる．

  入力データ tag(タグ)，phyNum(物理レジスタ番号)

  出力データ this

  処理方式 lastTag に tag をセットする．

  tag の型が TYPE_INT であれば regNumI に phyNum をセットし，そうでなければ regNumF に phyNum をセットする．

  エラー処理 なし．

- final void fill(int regNum)

  機能概要 レジスタの値をフィルするネイティブコードを生成する．

  機能説明 指定された物理レジスタに仮想レジスタのセーブ領域からロードを行うネイティブコードを生成する．

  入力データ regNum(物理レジスタ番号)

  出力データ ネイティブコードのロード命令．

  処理方式 emitLD(lastTag, regNum, baseReg, offset) を呼び出す．

  エラー処理 なし．

- final void spill(int tag)

  機能概要 レジスタの値をスピルするネイティブコードを生成する．

  機能説明 型を指定して，一時的に割り当てあられた物理レジスタの値を，仮想レジスタのセーブ領域にストアするネイティブコードを生成する．

入力データ tag(型)

出力データ ネイティブコードのストア命令．

処理方式 tagの型がTYPE_INTのとき，emitST(tag, regNumI, baseReg, offset)を呼び出し，スピルコードを生成する．

tagの型がTYPE_FLOATのとき，emitST(tag, regNumF, baseReg, offset)を呼び出し，スピルコードを生成する．

エラー処理 なし．

- final void nouse()

機能概要 この仮想レジスタが使われなくなったことを処理する．

機能説明 不必要な物理レジスタのセーブやスピルを行うネイティブコードの生成を防ぐため，この仮想レジスタがもう必要ないことを登録する．

入力データ iTmp(物理整数レジスタ管理情報)，fTmp(物理浮動小数レジスタ管理情報)

出力データ なし．

処理方式 regNumIが一時レジスタであれば，iTmp.freeTmpReg(regNumI)を呼び出し，一時レジスタを解放する．

regNumFが一時レジスタであれば，fTmp.freeTmpReg(regNumF)を呼び出し，一時レジスタを解放する．

エラー処理 なし．

- final void invalidate()

機能概要 割り当てられた物理レジスタ番号を無効化する．

機能説明 割り当てられた物理レジスタの値が破壊されて，その値を使用できなくなったときに，割り当てられた物理レジスタ番号を無効化する．

入力データ なし．

出力データ なし．

処理方式 lastTagの型がTYPE_INTのとき，regNumIにREG_INVALIDをセットする．それ以外のとき，regNumFにREG_INVALIDをセットする．

エラー処理 なし．

4.1.14 OpenJIT.RegAlloc$PhyRegs クラス

(1) 概要

OpenJIT.RegAlloc クラスのクラス内クラスであり，物理レジスタを管理する情報を扱う．このクラスの一つのオブジェクトで，32 個の物理レジスタを管理する．

(2) フィールド (変数)

- VirReg vregs[]

  仮想レジスタ管理情報へのポインタ．配列の要素の個数は 32 個．

- int init

  利用できる物理レジスタの初期値．32bit の各 1bit を利用し，1 のとき利用できることを表す．

- int free

  新しく割り付けに利用可能な物理レジスタ．32bit の各 1bit を利用．

- int used

  現在使用している物理レジスタ．32bit の各 1bit を利用．

- int dirt

  現在使用していて，値が変更された (dirty な) 物理レジスタ．32bit の各 1bit を利用．

- int lock

  現在使用していて，新しく割り付けに利用することができない物理レジスタ．32bit の各 1bit を利用．

- int tag

  物理レジスタの型．

(3) コンストラクタ

- public PhyRegs(int init, int tag)

  機能概要 phyRegs クラスのオブジェクトを生成し，初期化を行う．

  機能説明 phyRegs クラスのオブジェクトを生成し， vregs 配列の生成，利用できるレジスタの設定，型の設定を行う．

  入力データ init(利用できる物理レジスタ集合の初期値， bit パターン)， tag(型)

  出力データ なし．

  処理方式 フィールド free とフィールド init に init をセットする． vregs に 32 個の仮想レジスタ情報へのポインタを生成する．フィールド tag に tag をセットする．

  エラー処理 なし．

(4) メソッド

- final void fixReg(int regno)

  機能概要 指定された物理レジスタを固定的に割り当てる．

  機能説明 物理レジスタ番号を指定し，その物理レジスタを他の用途に使えないようにする．

  入力データ regno(物理レジスタ番号)

  出力データ なし．

  処理方式 free と init フィールドの指定されたレジスタの bit を 0 にする．

  エラー処理 なし．

- final int getFixReg()

  機能概要 物理レジスタを固定的に割り当てる．

  機能説明 利用できる物理レジスタの中から一つのレジスタを撰び，それを返す．さらに，その物理レジスタを他の用途に使えないようにする．

  入力データ なし．

出力データ 物理レジスタ番号．利用できる物理レジスタが見つからなかった場合，-1 を返す．

処理方式 reg に 0 をセットし，reg を 1 ずつ増やしながら init フィールド (32bit) の各 bit を LSB から調べていき，1 が立っているものを見つける．

見つかれば，free と init フィールドのそのレジスタに対応する bit を 0 にして，reg をリターンする．

見つからなければ-1 をリターンする．

エラー処理 なし．

- final void unlock()

機能概要 一時的に割り当てられた物理レジスタを他の用途に使えるようにする．

機能説明 固定的に物理レジスタが割り当てられていない仮想レジスタには一時レジスタが割り当てられる．この一時レジスタがもう必要なくなったときに，このメソッドを呼び出し，他の用途に使えるようにする．これまでに割り当てた全ての一時レジスタを解放する．

入力データ なし．

出力データ なし．

処理方式 lock に 0 をセットする．

エラー処理 なし．

- final void save()

機能概要 一時的に割り付けられた物理レジスタをセーブするネイティブコードを生成する．

機能説明 一時的に割り付けられた物理レジスタをメモリ上に待避するネイティブコードを生成する．

入力データ なし．

出力データ セーブするネイティブコード

処理方式 init と dirt の各 bit を調べる．dir の bit が 1 であるものは一時レジスタとして割り当てられた後，値が変更されたレジスタを意味している．両方の bit が立っているものについて，その物理レジスタ番号 reg を得て，vregs[reg].spill(tag) を呼び出すことにより，その物理レジスタの値をメモリ上に待避するネイティブコードを生成する．

エラー処理 なし．

- final void invalidateI()

  機能概要 これまでに割り当てを行った一時レジスタ (整数) の割り当てを解除する．

  機能説明 固定的に割り当てられていない整数の物理レジスタで，一時レジスタとして割り当てあられている全てのレジスタについて，割り当てを解除する．

  入力データ なし．

  出力データ なし．

  処理方式 free に init をセットする．used,dirt,lock に 0 をセットする．これに一時レジスタの割り当てを初期状態に戻す．また仮想レジスタ情報の更新も行う．vregs の各要素を調べ null でないものは，その regNumI フィールドに REG_INVALID をセットし，物理レジスタの割り当てを取り消す．

  エラー処理 なし．

- final void invalidateF()

  機能概要 これまでに割り当てを行った一時レジスタ (浮動小数) の割り当てを解除する．

  機能説明 固定的に割り当てられていない浮動小数の物理レジスタで，一時レジスタとして割り当てあられている全てのレジスタについて，割り当てを解除する．

  入力データ なし．

  出力データ なし．

処理方式 free に init をセットする．used,dirt,lock に 0 をセットする．これに一時レジスタの割り当てを初期状態に戻す．また仮想レジスタ情報の更新も行う．vregs の各要素を調べ null でないものは，その regNumF フィールドに REG_INVALID をセットし，物理レジスタの割り当てを取り消す．

エラー処理 なし．

- final void read(int regNum)

  機能概要 指定した物理レジスタを一時レジスタ (読み出し用) として使用し，他の用途に使えないようにする．

  機能説明 一時レジスタを読み出し用に確保した後で，そのレジスタを他の用途に使えないようにロックするために用いる．unlock() を呼び出すことによってこのロックは解除される．

  入力データ regNum(物理レジスタ番号)

  出力データ なし．

  処理方式 regNum に対応する lock の bit を 1 にする．

  エラー処理 なし．

- final void write(int regNum)

  機能概要 指定した物理レジスタを一時レジスタ (書き込み用) として使用し，他の用途に使えないようにする．

  機能説明 一時レジスタを書き込み用に確保した後で，そのレジスタを他の用途に使えないようにロックするために用いる．unlock() を呼び出すことによってこのロックは解除される．また指定されたレジスタに対応する dirt bit を 1 にして，値のセーブが必要なことを記録する．

  入力データ regNum(物理レジスタ番号)

  出力データ なし．

  処理方式 regNum に対応する lock と dirt の bit を 1 にする．

  エラー処理 なし．

- final private int searchVector(int vec)

機能概要 利用可能な物理レジスタ番号を返す．

機能説明 与えられた32bitデータ(bitベクタ)において1が立っているものを見つけ，それに対応する物理レジスタを一時レジスタとして利用するための初期化を行う．また，見つけたレジスタのdirt bitが1であれば，そのレジスタの値をスピルするネイティブコードを生成する．その物理レジスタ番号をリターン値として返す．

入力データ vec(32bitデータ)

出力データ 物理レジスタ番号，スピルを行うネイティブコード

処理方式 regに0をセットし，vecの右論理シフトし，regを1増加させながら，その下位1ビットを調べ，1が立っているものを見つける．

見つかった場合，見つかったbitと同じ桁のused, lockのbitを1にする．freeはそのbitを0にする．dirtの同じbitが1のとき，vregs[reg].spill(tag)を呼び出すことによりスピルを行うネイティブコードを生成する．その後，regをリターンする．

見つからなければ-1をリターンする．

エラー処理 なし．

- final int getTmpReg(int tag, VirReg vr)

機能概要 一時レジスタを確保する．

機能説明 一時的に利用する物理レジスタを確保し，その物理レジスタ番号を返す．

入力データ tag(型)，vr(仮想レジスタオブジェクト)

出力データ 物理レジスタ番号．

処理方式 まず，searchVector(free)を呼び自由に使えるレジスタfreeから物理レジスタを確保してみる．見つからない場合，使用しているがロックされておらず，スピル処理の不必要な(used & (lock) & (dirt))に対してsearchVectorを呼ぶ．これでも確保できなければ(used & (lock))に対してsearchVector を呼ぶ．これで見つからなければエラーとする．

見つかった物理レジスタ番号に対し，もし既に仮想レジスタに割り当てられていれば，前に割り当てられていた仮想レジスタの割り当てを無効にする．

仮想レジスタへのポインタ vregs を更新し，物理レジスタ番号をリターンする．

エラー処理 利用できる物理レジスタがなければ，OpenJIT .CompilerError ("No free temporary register") を発生する．

- final void freeTmpReg(int phyNum)

  機能概要 一時レジスタの割り当てのロックを解除する．

  機能説明 特定の一時レジスタの割り当てのロックを解除し，利用可能にする．

  入力データ phyNum(物理レジスタ番号)

  出力データ なし．

  処理方式 phyNum に対応した lock と dirt の 1bit を 0 にする．

  エラー処理 なし．

- final boolean isTmpReg(int phyNum)

  機能概要 物理レジスタが一時レジスタかどうかをチェックする．

  機能説明 物理レジスタが一時レジスタかどうかをチェックする．

  入力データ phyNum(物理レジスタ番号)

  出力データ 一時レジスタであれば true を返し，そうでなければ false を返す．

  処理方式 phyNum が REG_INVALID のときは false を返す．init の対応する bit が 1 であれば，true を返し．そうでなければ false を返す．

  エラー処理 なし．

4.1.15 OpenJIT.ExceptionHandler クラス

(1) 概要

バイトコードにおける例外処理に関する情報を表すクラスである．

(2) フィールド (変数)

- public int startPC

  例外ブロックの開始番地．Java 言語において try ブロックの先頭を表す．

- public int endPC

  例外ブロックの終了番地．Java 言語において try ブロックの終了を表す．

- public int handlerPC

  例外ハンドラの開始番地．例外が起きたときの飛び先アドレス．

- public int catchType

  例外のクラスを表す constant pool へのインデックス．0 のときはすべての例外を表す．すなわち Java で finally ブロックを指定した場合に相当する．

(3) コンストラクタ

なし．

(4) メソッド

- static void Set(Compile compile)

  機能概要 バイトコードのアドレスからネイティブコードのアドレスに変換する．

  機能説明 ネイティブコードの生成後，startPC, endPC, handlerPC の値ををバイトコードのアドレスから，ネイティブコードのアドレスに変換する．

  入力データ compile(コンパイル情報)

出力データ なし．

処理方式 compile のフィールドである bcinfo を参照し，bcinfo にセットする．また，compile から exceptionHandler フィールドを参照し，例外ハンドラの配列を得る．全ての例外ハンドラに対し，バイトコード情報 bcinfo の native_pc を参照することにより，startPC, endPC, handlerPC をネイティブコードのアドレスに変換する．

エラー処理 なし．

4.1.16 OpenJIT.CompilerError クラス

(1) 概要

OpenJIT システムにおいて，コンパイル中にエラーが起きたときのエラーを表すクラス．

(2) フィールド (変数)

- Throwable e

  エラーや例外を保持する．

(3) コンストラクタ

- public CompilerError(String msg)

  機能概要 CompilerError オブジェクトを生成する．

  機能説明 CompilerError オブジェクトを生成する．

  入力データ msg(エラーメッセージ)

  出力データ なし．

  処理方式 スーパクラス (java.lang.Error) のコンストラクタを呼び出した後，e に this を設定する．

  エラー処理 なし．

- public CompilerError(Exception e)

  機能概要 CompilerError オブジェクトを生成する．

  機能説明 CompilerError オブジェクトを生成する．

  入力データ e(例外)

  出力データ なし．

  処理方式 e.getMessage を呼び出し，例外のメッセージを得て，スーパクラス (java.lang.Error) のコンストラクタを呼び出す．その後,e に this を設定する．

エラー処理 なし．

(4) メソッド

- public void printStackTrace()

  機能概要 Java のスタックの状態を表示する．

  機能説明 Java のスタックの状態を表示する．Throwable クラスの printStackTrace を override したメソッドである．

  入力データ なし．

  出力データ Java のスタックの状態．

  処理方式 スーパクラスの printStackTrace を呼び出す．

  エラー処理 なし．

4.1.17 OpenJIT.Sparc クラス

このクラスは契約の対象外である．

4.1.18 OpenJIT.Runtime クラス

このクラスは契約の対象外である．

4.2 OpenJIT.frontend.asm パッケージ

- インターフェース
  - OpenJIT.frontend.asm.Constants インターフェース
  - OpenJIT.frontend.asm.RuntimeConstants インターフェース
- 抽象クラス

  なし．
- クラス
  - OpenJIT.frontend.asm.ArrayData クラス
  - OpenJIT.frontend.asm.Assembler クラス
  - OpenJIT.frontend.asm.Instruction クラス
  - OpenJIT.frontend.asm.Label クラス
- 例外

  なし．

4.2.1 OpenJIT.frontend.asm.Constants インターフェース

public interface Constants extends RuntimeConstants

(1) 概要

このインターフェースは，OpenJIT.frontend.asmパッケージ，OpenJIT.frontend.tree パッケージで用いる定数を定義する．

(2) フィールド (変数)

- public static final Identifier idAppend = Identifier.lookup(”append”)

  識別子の定義．(append 用)

- public static final Identifier idClassInit = Identifier.lookup("<clinit>")

  識別子の定義．(Class initializer 用)

- public static final Identifier idCode = Identifier.lookup("Code")

  識別子の定義．(コード用)

- public static final Identifier idInit = Identifier.lookup("<init>")

  識別の定義．(initializer 用)

- public static final Identifier idLength = Identifier.lookup("length")

  識別子の定義．(length 用)

- public static final Identifier idNull = Identifier.lookup("")

  識別子の定義．(null 用)

- public static final Identifier idPrint = Identifier.lookup("print")

  識別子の定義．(print 用)

- public static final Identifier idPrintln = Identifier.lookup("println")

  識別子の定義．(println 用)

- public static final Identifier idStar = Identifier.lookup("*")

  識別子の定義．(* 用)

- public static final Identifier idSuper = Identifier.lookup("super")

  識別子の定義．(super 用)

- public static final Identifier idThis = Identifier.lookup("this")

  識別子の定義．(this 用)

- public static final Identifier idToString = Identifier.lookup("toString")

  識別子の定義．(toString 用)

- public static final Identifier idValueOf = Identifier.lookup("valueOf")

  識別子の定義．(valueOf 用)

- public static final Identifier idJavaLang = Identifier.lookup(”java.lang”)

  識別子の定義．(java.lang 用)

- public static final Identifier idJavaLangCloneable = Identifier.lookup(”java.lang.Cloneable”)

  識別子の定義．(java.lang.Clonable 用)

- public static final Identifier idJavaLangError = Identifier.lookup(”java.lang.Error”)

  識別子の定義．(java.lang.Error 用)

- public static final Identifier idJavaLangException = Identifier.lookup(”java.lang.Exception”)

  識別子の定義．(java.lang.Exception 用)

- public static final Identifier idJavaLangObject = Identifier.lookup(”java.lang.Object”)

  識別子の定義．(java.lang.Object 用)

- public static final Identifier idJavaLangRuntimeException = Identifier.lookup(”java.lang.Ru

  識別子の定義．(java.lang.RuntimeException 用)

- public static final Identifier idJavaLangString = Identifier.lookup(”java.lang.String”)

  識別子の定義．(java.lang.String 用)

- public static final Identifier idJavaLangStringBuffer = Identifier.lookup(”java.lang.StringBu

  識別子の定義．(java.lang.StringBuffer 用)

- public static final Identifier idJavaLangThrowable = Identifier.lookup(”java.lang.Throwable

  識別子の定義．(java.lang.Throwable 用)

- public static final Identifier idConstantValue = Identifier.lookup(”ConstantValue”)

  識別子の定義．(ConstantValue 用)

- public static final Identifier idLocalVariables = Identifier.lookup(”LocalVariables”)

  識別子の定義．(LocalVariables 用)

- public static final Identifier idLineNumberTable = Identifier.lookup(”LineNumberTable”)

  識別子の定義．(LineNumberTable 用)

- public static final Identifier idSourceFile = Identifier.lookup("SourceFile")

  識別子の定義．(SourceFile 用)

- public static final Identifier idDocumentation = Identifier.lookup("Documentation")

  識別子の定義．(Documentation 用)

- public static final Identifier idExceptions = Identifier.lookup("Exceptions")

  識別子の定義．(Exceptions 用)

- public static final int F_VERBOSE = 1 << 0

  フラグ．(Verbose 用)

- public static final int F_DUMP = 1 << 1

  フラグ．(Dump 用)

- public static final int F_WARNINGS = 1 << 2

  フラグ．(Warnings 用)

- public static final int F_DEBUG = 1 << 3

  フラグ．(Debug 用)

- public static final int F_OPTIMIZE = 1 << 4

  フラグ．(Optimize 用)

- public static final int F_DEPENDENCIES = 1 << 5

  フラグ．(Dependencies 用)

- public final int M_PUBLIC = ACC_PUBLIC

  モディファイアのビットフィールド．(public 用)

- public final int M_PRIVATE = ACC_PRIVATE

  モディファイアのビットフィールド．(private 用)

- public final int M_PROTECTED = ACC_PROTECTED

  モディファイアのビットフィールド．(protected 用)

- public final int M_STATIC = ACC_STATIC

  モディファイアのビットフィールド．(static 用)

- public final int M_TRANSIENT = ACC_TRANSIENT

  モディファイアのビットフィールド．(transient 用)

- public final int M_SYNCHRONIZED = ACC_SYNCHRONIZED

  モディファイアのビットフィールド．(synchronized 用)

- public final int M_ABSTRACT = ACC_ABSTRACT

  モディファイアのビットフィールド．(abstract 用)

- public final int M_NATIVE = ACC_NATIVE

  モディファイアのビットフィールド．(native 用)

- public final int M_FINAL = ACC_FINAL

  モディファイアのビットフィールド．(final 用)

- public final int M_VOLATILE = ACC_VOLATILE

  モディファイアのビットフィールド．(volatile 用)

- public final int M_INTERFACE = ACC_INTERFACE

  モディファイアのビットフィールド．(interface 用)

- public final int MM_CLASS = M_PUBLIC | M_INTERFACE | M_FINAL | M_ABSTRACT

  モディファイアのマスク．(クラス用)

- public final int MM_FIELD = M_PUBLIC | M_PRIVATE | M_PROTECTED | M_FINAL | M_STATIC | M_TRANSIENT | M_VOLATILE

  モディファイアのマスク．(フィールド用)

- public final int MM_METHOD = M_PUBLIC | M_PRIVATE | M_PROTECTED | M_FINAL | M_STATIC | M_SYNCHRONIZED | M_ABSTRACT | M_NATIVE

  モディファイアのマスク．(メソッド用)

- public static final int TC_BOOLEAN = 0

  タイプコード．(boolean 型)

- public static final int TC_BYTE = 1

  タイプコード．(byte 型)

- public static final int TC_CHAR = 2

  タイプコード．(char 型)

- public static final int TC_SHORT = 3

  タイプコード．(short 型)

- public static final int TC_INT = 4

  タイプコード．(int 型)

- public static final int TC_LONG = 5

  タイプコード．(long 型)

- public static final int TC_FLOAT = 6

  タイプコード．(float 型)

- public static final int TC_DOUBLE = 7

  タイプコード．(double 型)

- public static final int TC_NULL = 8

  タイプコード．(null)

- public static final int TC_ARRAY = 9

  タイプコード．(配列型)

- public static final int TC_CLASS = 10

  タイプコード．(クラスオブジェクト)

- public static final int TC_VOID = 11

  タイプコード．(void 型)

- public static final int TC_METHOD = 12

  タイプコード．(メソッド)

- public static final int TC_ERROR = 13

  タイプコード．(エラー)

- public static final int TM_NULL = 1 << TC_NULL

  タイプマスク．(null)

- public static final int TM_VOID = 1 << TC_VOID

  タイプマスク．(void 型)

- public static final int TM_BOOLEAN = 1 << TC_BOOLEAN

  タイプマスク．(boolean 型)

- public static final int TM_BYTE = 1 << TC_BYTE

  タイプマスク．(byte 型)

- public static final int TM_CHAR = 1 << TC_CHAR

  タイプマスク．(char 型)

- public static final int TM_SHORT = 1 << TC_SHORT

  タイプマスク．(short 型)

- public static final int TM_INT = 1 << TC_INT

  タイプマスク．(int 型)

- public static final int TM_LONG = 1 << TC_LONG

  タイプマスク．(long 型)

- public static final int TM_FLOAT = 1 << TC_FLOAT

  タイプマスク．(float 型)

- public static final int TM_DOUBLE = 1 << TC_DOUBLE

  タイプマスク．(double 型)

- public static final int TM_ARRAY = 1 << TC_ARRAY

  タイプマスク．(配列)

- public static final int TM_CLASS = 1 << TC_CLASS

  タイプマスク．(クラス)

- public static final int TM_METHOD = 1 << TC_METHOD

  タイプマスク．(メソッド)

- public static final int TM_ERROR = 1 << TC_ERROR

  タイプマスク．(エラー)

- public static final int TM_INT32 = TM_BYTE | TM_SHORT | TM_CHAR | TM_INT

  タイプマスク．(32 ビットの整数)

- public static final int TM_NUM32 = TM_INT32 | TM_FLOAT

  タイプマスク．(32 ビットの数値)

- public static final int TM_NUM64 = TM_LONG | TM_DOUBLE

  タイプマスク．(64 ビットの数値)

- public static final int TM_INTEGER = TM_INT32 | TM_LONG

  タイプマスク．(整数)

- public static final int TM_REAL = TM_FLOAT | TM_DOUBLE

  タイプマスク．(実数)

- public static final int TM_NUMBER = TM_INTEGER | TM_REAL

  タイプマスク．(数値)

- public static final int TM_REFERENCE = TM_ARRAY | TM_CLASS | TM_NULL

  タイプマスク．(参照)

- public static final int CS_UNDEFINED = 0

  クラスの状態．(未定義)

- public static final int CS_UNDECIDED = 1

  クラスの状態．(未決定)

- public static final int CS_BINARY = 2

  クラスの状態．(クラスファイル)

- public static final int CS_SOURCE = 3

  クラスの状態．(ソースファイル)

- public static final int CS_PARSED = 4

  クラスの状態．(パーズが済んだ状態)

- public static final int CS_COMPILED = 5

  クラスの状態．(コンパイルされた状態)

- public static final int CS_NOTFOUND = 6

  クラスの状態．(クラスが見つからない)

- public static final int ATT_ALL = -1

  属性．

- public static final int ATT_CODE = 1

  属性．

- public static final int OFFSETBITS = 18

  ファイルの位置を表すオフセットのビットサイズ．

- public static final int MAXFILESIZE = (1 << OFFSETBITS) - 1

  最大のファイルサイズ．

- public static final int MAXLINENUMBER = (1 << (32 - OFFSETBITS)) - 1

  最大の行数．

- public final int COMMA = 0

  オペレータ．(カンマ)

- public final int ASSIGN = 1

  オペレータ．(=)

- public final int ASGMUL = 2

  オペレータ．(*=)

- public final int ASGDIV = 3

  オペレータ．(/=)

- public final int ASGREM = 4

  オペレータ．(%=)

- public final int ASGADD = 5

  オペレータ．(+=)

- public final int ASGSUB = 6

  オペレータ．(-=)

- public final int ASGLSHIFT = 7

  オペレータ．(<<=)

- public final int ASGRSHIFT = 8

  オペレータ．(>>=)

- public final int ASGURSHIFT = 9

  オペレータ．(>>>=)

- public final int ASGBITAND = 10

  オペレータ．(&=)

- public final int ASGBITOR = 11

  オペレータ．(|=)

- public final int ASGBITXOR = 12

  オペレータ．(^=)

- public final int COND = 13

  オペレータ．(**条件式**)

- public final int OR = 14

  オペレータ．(||)

- public final int AND = 15

  オペレータ．(&&)

- public final int BITOR = 16

  オペレータ．(|)

- public final int BITXOR = 17

  オペレータ．(^)

- public final int BITAND = 18

  オペレータ．(&)

- public final int NE = 19

  オペレータ．(!=)

- public final int EQ = 20

  オペレータ．(==)

- public final int GE = 21

  オペレータ．(>=)

- public final int GT = 22

  オペレータ．(>)

- public final int LE = 23

  オペレータ．(<=)

- public final int LT = 24

  オペレータ．(<)

- public final int INSTANCEOF = 25

  オペレータ．(instanceof)

- public final int LSHIFT = 26

  オペレータ．(<<)

- public final int RSHIFT = 27

  オペレータ．(>>)

- public final int URSHIFT = 28

  オペレータ．(>>>)

- public final int ADD = 29

  オペレータ．(+)

- public final int SUB = 30

  オペレータ．(–)

- public final int DIV = 31

  オペレータ．(/)

- public final int REM = 32

  オペレータ．(%)

- public final int MUL = 33

  オペレータ．(*)

- public final int CAST = 34

  オペレータ．(**キャスト式** (x)y)

- public final int POS = 35

  オペレータ．(+x)

- public final int NEG = 36

  オペレータ．(–x)

- public final int NOT = 37

  オペレータ．(!)

- public final int BITNOT = 38

  オペレータ．(!)

- public final int PREINC = 39

  オペレータ．(++x)

- public final int PREDEC = 40

  オペレータ．(– –x)

- public final int NEWARRAY = 41

  オペレータ．(newarray)

- public final int NEWINSTANCE = 42

  オペレータ．(newinstance)

- public final int NEWFROMNAME = 43

  オペレータ．(newfromname)

- public final int POSTINC = 44

  オペレータ．(x++)

- public final int POSTDEC = 45

  オペレータ．(x–)

- public final int FIELD = 46

  オペレータ．(field)

- public final int METHOD = 47

  オペレータ．(**メソッド** x(y))

- public final int ARRAYACCESS = 48

  オペレータ．(arrayaccess x[y])

- public final int NEW = 49

  オペレータ．(new)

- public final int INC = 50

  オペレータ．(++)

- public final int DEC = 51

  オペレータ．(– –)

- public final int CONVERT = 55

  オペレータ．(**暗黙の型変換**)

- public final int EXPR = 56

  オペレータ．((x) **式**)

- public final int ARRAY = 57

  オペレータ．({ x, y, ... })

- public final int GOTO = 58

  オペレータ．(goto)

- public final int IDENT = 60

  value **のトークン．**(**変数**)

- public final int BOOLEANVAL = 61

  value **のトークン．**(boolean **値**)

- public final int BYTEVAL = 62

  value **のトークン．**(byte **値**)

- public final int CHARVAL = 63

  value **のトークン．**(char **値**)

- public final int SHORTVAL = 64

  value のトークン．(short 値)

- public final int INTVAL = 65

  value のトークン．(int 値)

- public final int LONGVAL = 66

  value のトークン．(long 値)

- public final int FLOATVAL = 67

  value のトークン．(float 値)

- public final int DOUBLEVAL = 68

  value のトークン．(double 値)

- public final int STRINGVAL = 69

  value のトークン．(String オブジェクト)

- public final int BYTE = 70

  タイプのキーワード．(byte 型)

- public final int CHAR = 71

  タイプのキーワード．(char 型)

- public final int SHORT = 72

  タイプのキーワード．(short 型)

- public final int INT = 73

  タイプのキーワード．(int 型)

- public final int LONG = 74

  タイプのキーワード．(long 型)

- public final int FLOAT = 75

  タイプのキーワード．(float 型)

- public final int DOUBLE = 76

  タイプのキーワード．(double 型)

- public final int VOID = 77

  タイプのキーワード．(void 型)

- public final int BOOLEAN = 78

  タイプのキーワード．(boolean 型)

- public final int TRUE = 80

  式のキーワード．(true)

- public final int FALSE = 81

  式のキーワード．(false)

- public final int THIS = 82

  式のキーワード．(this)

- public final int SUPER = 83

  式のキーワード．(super)

- public final int NULL = 84

  式のキーワード．(null)

- public final int IF = 90

  文のキーワード．(if)

- public final int ELSE = 91

  文のキーワード．(else)

- public final int FOR = 92

  文のキーワード．(for)

- public final int WHILE = 93

  文のキーワード．(while)

- public final int DO = 94

  文のキーワード．(do)

- public final int SWITCH = 95

  文のキーワード．(switch)

- public final int CASE = 96

  文のキーワード．(case)

- public final int DEFAULT = 97

  文のキーワード．(default)

- public final int BREAK = 98

  文のキーワード．(break)

- public final int CONTINUE = 99

  文のキーワード．(continue)

- public final int RETURN = 100

  文のキーワード．(return)

- public final int TRY = 101

  文のキーワード．(try)

- public final int CATCH = 102

  文のキーワード．(catch)

- public final int FINALLY = 103

  文のキーワード．(finally)

- public final int THROW = 104

  文のキーワード．(throw)

- public final int STAT = 105

  文のキーワード．(stat)

- public final int EXPRESSION = 106

  文のキーワード．(expression)

- public final int DECLARATION = 107

  文のキーワード．(declaration)

- public final int VARDECLARATION = 108

  文のキーワード．(vardeclaration)

- public final int IMPORT = 110

  宣言のキーワード．(import)

- public final int CLASS = 111

  宣言のキーワード．(class)

- public final int EXTENDS = 112

  宣言のキーワード．(extends)

- public final int IMPLEMENTS = 113

  宣言のキーワード．(implements)

- public final int INTERFACE = 114

  宣言のキーワード．(interface)

- public final int PACKAGE = 115

  宣言のキーワード．(package)

- public final int PRIVATE = 120

  モディファイアのキーワード．(private)

- public final int PUBLIC = 121

  モディファイアのキーワード．(public)

- public final int PROTECTED = 122

  モディファイアのキーワード．(protected)

- public final int CONST = 123

  モディファイアのキーワード．(const)

- public final int STATIC = 124

  モディファイアのキーワード．(static)

- public final int TRANSIENT = 125

  モディファイアのキーワード．(transient)

- public final int SYNCHRONIZED = 126

  モディファイアのキーワード．(synchronized)

- public final int NATIVE = 127

  モディファイアのキーワード．(native)

- public final int FINAL = 128

  モディファイアのキーワード．(final)

- public final int VOLATILE = 129

  モディファイアのキーワード．(volatile)

- public final int ABSTRACT = 130

  モディファイアのキーワード．(abstract)

- public final int SEMICOLON = 135

  区切り記号のキーワード．(;)

- public final int COLON = 136

  区切り記号のキーワード．(:)

- public final int QUESTIONMARK = 137

  区切り記号のキーワード．(?)

- public final int LBRACE = 138

  区切り記号のキーワード．({)

- public final int RBRACE = 139

  区切り記号のキーワード．(})

- public final int LPAREN = 140

  区切り記号のキーワード．(“(”)

- public final int RPAREN = 141

  区切り記号のキーワード．(“)”)

- public final int LSQBRACKET = 142

  区切り記号のキーワード．([)

- public final int RSQBRACKET = 143

  区切り記号のキーワード．(])

- public final int THROWS = 144

  区切り記号のキーワード．(throws)

- public final int ERROR = 145

  特殊なトークン．(エラー)

- public final int COMMENT = 146

  特殊なトークン．(コメント)

- public final int TYPE = 147

  特殊なトークン．(型)

- public final int LENGTH = 148

  特殊なトークン．(サイズ)

- public final int INLINERETURN = 149

  特殊なトークン．(inline return)

- public final int INLINEMETHOD = 150

  特殊なトークン．(inline method)

- public final int INLINENEWINSTANCE = 151

  特殊なトークン．(inline newinstance)

- public static final int opPrecedence[] = { 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 12, 13, 14, 15, 16, 17, 18, 18, 19, 19, 19, 19, 19, 20, 20, 20, 21, 21, 22, 22, 22, 23, 24, 24, 24, 24, 24, 24, 25, 25, 26, 26, 26, 26, 26, 26 }

  オペレータの優先順位．

- public static final String opNames[] = { ",", "=", "*=", "/=", "%=", "+=", "-=", "<<=", ">>=", "<<<=", "&=", "|=", "^=", "?:", "||", "&&", "|", "^", "&", "!=", "==", ">=", ">", "<=", "<", "instanceof", "<<", ">>", "<<<", "+", "-", "/", "%", "*", "cast", "+", "-", "!", "~", "++", "--", "new", "new", "new", "++", "--", "field", "method", "[]", "new", "++", "-", null, null, null, "convert", "expr", "array", "goto", null, "Identifier", "boolean", "byte", "char", "short", "int", "long", "float", "double", "string", "byte", "char", "short", "int", "long", "float", "double", "void", "boolean", null, "true", "false", "this", "super", "null", null, null, null, null, null, "if", "else", "for", "while", "do", "switch", "case", "default", "break", "continue", "return", "try", "catch", "finally", "throw", "stat", "expression", "declaration", "declaration", null, "import", "class", "extends", "implements", "interface", "package", null, null, null, null, "private", "public", "protected", "const", "static", "transient", "synchronized", "native", "final", "volatile", "abstract", null, null, null, null, ";", ":", "?", "{", "}", "(", ")", "[", "]", "throws", "error", "comment", "type", "length", "inline-return", "inline-method", "inline-new" }

  オペレータの名前．

(3) コンストラクタ

なし．

(4) メソッド

なし．

4.2.2 OpenJIT.frontend.asm.RuntimeConstants インターフェース

public interface RuntimeConstants

(1) 概要

このインターフェースは，OpenJIT.frontend.asmパッケージ，OpenJIT.frontend.treeパッケージで用いる定数を定義する．

(2) フィールド (変数)

- public static final char SIGC_VOID = 'V'

  シグナチャに含まれる文字の void 型を表す.

- public static final String SIG_VOID = "V"

  シグナチャに含まれる文字の void 型を表す.

- public static final char SIGC_BOOLEAN = 'Z'

  シグナチャに含まれる文字の boolean 型を表す.

- public static final String SIG_BOOLEAN = "Z"

  シグナチャに含まれる文字の boolean 型を表す.

- public static final char SIGC_BYTE = 'B'

  シグナチャに含まれる文字の byte 型を表す.

- public static final String SIG_BYTE = "B"

  シグナチャに含まれる文字の byte 型を表す.

- public static final char SIGC_CHAR = 'C'

  シグナチャに含まれる文字の char 型を表す.

- public static final String SIG_CHAR = "C"

  シグナチャに含まれる文字の char 型を表す.

- public static final char SIGC_SHORT = 'S'

  シグナチャに含まれる文字の short 型を表す.

- public static final String SIG_SHORT = "S"

  シグナチャに含まれる文字の short 型を表す.

- public static final char SIGC_INT = 'I'

  シグナチャに含まれる文字の int 型を表す.

- public static final String SIG_INT = "I"

  シグナチャに含まれる文字の int 型を表す.

- public static final char SIGC_LONG = 'J'

  シグナチャに含まれる文字の long 型を表す.

- public static final String SIG_LONG = "J"

  シグナチャに含まれる文字の long 型を表す.

- public static final char SIGC_FLOAT = 'F'

  シグナチャに含まれる文字の float 型を表す.

- public static final String SIG_FLOAT = "F"

  シグナチャに含まれる文字の float 型を表す.

- public static final char SIGC_DOUBLE = 'D'

  シグナチャに含まれる文字の double 型を表す.

- public static final String SIG_DOUBLE = "D"

  シグナチャに含まれる文字の double 型を表す.

- public static final char SIGC_ARRAY = '['

  シグナチャに含まれる文字の配列型を表す.

- public static final String SIG_ARRAY = "["

  シグナチャに含まれる文字の配列型を表す.

- public static final char SIGC_CLASS = 'L'

  シグナチャに含まれる文字のクラス型の開始を表す.

- public static final String SIG_CLASS = "L"

  シグナチャに含まれる文字のクラス型の開始を表す.

- public static final char SIGC_METHOD = '('

  シグナチャに含まれる文字の引数の開始を表す.

- public static final String SIG_METHOD = "("

  シグナチャに含まれる文字の引数の開始を表す.

- public static final char SIGC_ENDCLASS = ';'

  シグナチャに含まれる文字のクラス型の終わりを表す.

- public static final String SIG_ENDCLASS = ";"

  シグナチャに含まれる文字のクラス型の終わりを表す.

- public static final char SIGC_ENDMETHOD = ')'

  シグナチャに含まれる文字の引数の終わりを表す.

- public static final String SIG_ENDMETHOD = ")"

  シグナチャに含まれる文字の引数の終わりを表す.

- public static final char SIGC_PACKAGE = '/'

  シグナチャに含まれる文字のクラスの階層のセパレータを表す.

- public static final String SIG_PACKAGE = "/"

  シグナチャに含まれる文字のクラスの階層のセパレータを表す.

- public static final int JAVA_MAGIC = 0xcafebabe;

  マジックナンバー .

- public static final int JAVA_VERSION = 45;

  Java のメジャーバージョンナンバー .

- public static final int JAVA_MINOR_VERSION = 3;

  Java のマイナーバージョンナンバー．

- public static final int CONSTANT_UTF8 = 1

  constant pool の種類． UTF8

- public static final int CONSTANT_UNICODE = 2

  constant pool の種類． UNICODE

- public static final int CONSTANT_INTEGER = 3

  constant pool の種類． int

- public static final int CONSTANT_FLOAT = 4

  constant pool の種類． float

- public static final int CONSTANT_LONG = 5

  constant pool の種類． long

- public static final int CONSTANT_DOUBLE = 6

  constant pool の種類． double

- public static final int CONSTANT_CLASS = 7

  constant pool の種類． class

- public static final int CONSTANT_STRING = 8

  constant pool の種類． String

- public static final int CONSTANT_FIELD = 9

  constant pool の種類．フィールド

- public static final int CONSTANT_METHOD = 10

  constant pool の種類．メソッド

- public static final int CONSTANT_INTERFACEMETHOD = 11

  constant pool の種類．インタフェース

- public static final int CONSTANT_NAMEANDTYPE = 12

  constant pool の種類．名前

- public static final int ACC_PUBLIC = 0x00000001

  JDK におけるクラスの属性． public

- public static final int ACC_PRIVATE = 0x00000002

  JDK におけるクラスの属性． private

- public static final int ACC_PROTECTED = 0x00000004

  JDK におけるクラスの属性． protected

- public static final int ACC_STATIC = 0x00000008

  JDK におけるクラスの属性． static

- public static final int ACC_FINAL = 0x00000010

  JDK におけるクラスの属性． final

- public static final int ACC_SYNCHRONIZED = 0x00000020

  JDK におけるクラスの属性． synchronized

- public static final int ACC_VOLATILE = 0x00000040

  JDK におけるクラスの属性． volatile

- public static final int ACC_TRANSIENT = 0x00000080

  JDK におけるクラスの属性． transient

- public static final int ACC_NATIVE = 0x00000100

  JDK におけるクラスの属性． native

- public static final int ACC_INTERFACE = 0x00000200

  JDK におけるクラスの属性． interface

- public static final int ACC_ABSTRACT = 0x00000400

  JDK におけるクラスの属性． abstract

- public static final int ACC_SUPER = 0x00000800

  JDK におけるクラスの属性． super

- public static final int opc_try = -3

  バイトコード命令． try

- public static final int opc_dead = -2

  バイトコード命令． dead

- public static final int opc_label = -1

  バイトコード命令． label

- public static final int opc_nop = 0

  バイトコード命令． nop

- public static final int opc_aconst_null = 1

  バイトコード命令． aconst_null

- public static final int opc_iconst_m1 = 2

  バイトコード命令． iconst_m1

- public static final int opc_iconst_0 = 3

  バイトコード命令． iconst_0

- public static final int opc_iconst_1 = 4

  バイトコード命令． iconst_1

- public static final int opc_iconst_2 = 5

  バイトコード命令． iconst_2

- public static final int opc_iconst_3 = 6

  バイトコード命令． iconst_3

- public static final int opc_iconst_4 = 7

  バイトコード命令． iconst_4

- public static final int opc_iconst_5 = 8

  バイトコード命令．iconst_5

- public static final int opc_lconst_0 = 9

  バイトコード命令．lconst_0

- public static final int opc_lconst_1 = 10

  バイトコード命令．lconst_1

- public static final int opc_fconst_0 = 11

  バイトコード命令．fconst_0

- public static final int opc_fconst_1 = 12

  バイトコード命令．fconst_1

- public static final int opc_fconst_2 = 13

  バイトコード命令．fconst_2

- public static final int opc_dconst_0 = 14

  バイトコード命令．dconst_0

- public static final int opc_dconst_1 = 15

  バイトコード命令．dconst_1

- public static final int opc_bipush = 16

  バイトコード命令．bipush

- public static final int opc_sipush = 17

  バイトコード命令．sipush

- public static final int opc_ldc = 18

  バイトコード命令．ldc

- public static final int opc_ldc_w = 19

  バイトコード命令．ldc_w

- public static final int opc_ldc2_w = 20

  バイトコード命令．ldc2_w

- public static final int opc_iload = 21

  バイトコード命令．iload

- public static final int opc_lload = 22

  バイトコード命令．lload

- public static final int opc_fload = 23

  バイトコード命令．fload

- public static final int opc_dload = 24

  バイトコード命令．dload

- public static final int opc_aload = 25

  バイトコード命令．aload

- public static final int opc_iload_0 = 26

  バイトコード命令．iload_0

- public static final int opc_iload_1 = 27

  バイトコード命令．iload_1

- public static final int opc_iload_2 = 28

  バイトコード命令．iload_2

- public static final int opc_iload_3 = 29

  バイトコード命令．iload_3

- public static final int opc_lload_0 = 30

  バイトコード命令．lload_0

- public static final int opc_lload_1 = 31

  バイトコード命令．lload_1

- public static final int opc_lload_2 = 32

  バイトコード命令．lload_2

- public static final int opc_lload_3 = 33

  バイトコード命令．lload_3

- public static final int opc_fload_0 = 34

  バイトコード命令．fload_0

- public static final int opc_fload_1 = 35

  バイトコード命令．fload_1

- public static final int opc_fload_2 = 36

  バイトコード命令．fload_2

- public static final int opc_fload_3 = 37

  バイトコード命令．fload_3

- public static final int opc_dload_0 = 38

  バイトコード命令．dload_0

- public static final int opc_dload_1 = 39

  バイトコード命令．dload_1

- public static final int opc_dload_2 = 40

  バイトコード命令．dload_2

- public static final int opc_dload_3 = 41

  バイトコード命令．dload_3

- public static final int opc_aload_0 = 42

  バイトコード命令．aload_0

- public static final int opc_aload_1 = 43

  バイトコード命令．aload_1

- public static final int opc_aload_2 = 44

  バイトコード命令． aload_2

- public static final int opc_aload_3 = 45

  バイトコード命令． aload_3

- public static final int opc_iaload = 46

  バイトコード命令． iaload

- public static final int opc_laload = 47

  バイトコード命令． laload

- public static final int opc_faload = 48

  バイトコード命令． faload

- public static final int opc_daload = 49

  バイトコード命令． daload

- public static final int opc_aaload = 50

  バイトコード命令． aaload

- public static final int opc_baload = 51

  バイトコード命令． baload

- public static final int opc_caload = 52

  バイトコード命令． caload

- public static final int opc_saload = 53

  バイトコード命令． saload

- public static final int opc_istore = 54

  バイトコード命令． istore

- public static final int opc_lstore = 55

  バイトコード命令． lstore

- public static final int opc_fstore = 56

  バイトコード命令． fstore

- public static final int opc_dstore = 57

  バイトコード命令． dstore

- public static final int opc_astore = 58

  バイトコード命令． astore

- public static final int opc_istore_0 = 59

  バイトコード命令． istore_0

- public static final int opc_istore_1 = 60

  バイトコード命令． istore_1

- public static final int opc_istore_2 = 61

  バイトコード命令． istore_2

- public static final int opc_istore_3 = 62

  バイトコード命令． istore_3

- public static final int opc_lstore_0 = 63

  バイトコード命令． lstore_0

- public static final int opc_lstore_1 = 64

  バイトコード命令． lstore_1

- public static final int opc_lstore_2 = 65

  バイトコード命令． lstore_2

- public static final int opc_lstore_3 = 66

  バイトコード命令． lstore_3

- public static final int opc_fstore_0 = 67

  バイトコード命令． fstore_0

- public static final int opc_fstore_1 = 68

  バイトコード命令． fstore_1

- public static final int opc_fstore_2 = 69

  バイトコード命令． fstore_2

- public static final int opc_fstore_3 = 70

  バイトコード命令． fstore_3

- public static final int opc_dstore_0 = 71

  バイトコード命令． dstore_0

- public static final int opc_dstore_1 = 72

  バイトコード命令． dstore_1

- public static final int opc_dstore_2 = 73

  バイトコード命令． dstore_2

- public static final int opc_dstore_3 = 74

  バイトコード命令． dstore_3

- public static final int opc_astore_0 = 75

  バイトコード命令． astore_0

- public static final int opc_astore_1 = 76

  バイトコード命令． astore_1

- public static final int opc_astore_2 = 77

  バイトコード命令． astore_2

- public static final int opc_astore_3 = 78

  バイトコード命令． astore_3

- public static final int opc_iastore = 79

  バイトコード命令． iastore

- public static final int opc_lastore = 80

  バイトコード命令．lastore

- public static final int opc_fastore = 81

  バイトコード命令．fastore

- public static final int opc_dastore = 82

  バイトコード命令．dastore

- public static final int opc_aastore = 83

  バイトコード命令．aastore

- public static final int opc_bastore = 84

  バイトコード命令．bastore

- public static final int opc_castore = 85

  バイトコード命令．castore

- public static final int opc_sastore = 86

  バイトコード命令．sastore

- public static final int opc_pop = 87

  バイトコード命令．pop

- public static final int opc_pop2 = 88

  バイトコード命令．pop2

- public static final int opc_dup = 89

  バイトコード命令．dup

- public static final int opc_dup_x1 = 90

  バイトコード命令．dup_x1

- public static final int opc_dup_x2 = 91

  バイトコード命令．dup_x2

- public static final int opc_dup2 = 92

  バイトコード命令．dup2

- public static final int opc_dup2_x1=93

  バイトコード命令．dup2_x1

- public static final int opc_dup2_x2=94

  バイトコード命令．dup2_x2

- public static final int opc_swap=95

  バイトコード命令．swap

- public static final int opc_iadd=96

  バイトコード命令．iadd

- public static final int opc_ladd=97

  バイトコード命令．ladd

- public static final int opc_fadd=98

  バイトコード命令．fadd

- public static final int opc_dadd=99

  バイトコード命令．dadd

- public static final int opc_isub=100

  バイトコード命令．isub

- public static final int opc_lsub=101

  バイトコード命令．lsub

- public static final int opc_fsub=102

  バイトコード命令．fsub

- public static final int opc_dsub=103

  バイトコード命令．dsub

- public static final int opc_imul=104

  バイトコード命令．imul

- public static final int opc_lmul=105

  バイトコード命令．lmul

- public static final int opc_fmul=106

  バイトコード命令．fmul

- public static final int opc_dmul=107

  バイトコード命令．dmul

- public static final int opc_idiv=108

  バイトコード命令．idiv

- public static final int opc_ldiv=109

  バイトコード命令．ldiv

- public static final int opc_fdiv=110

  バイトコード命令．fdiv

- public static final int opc_ddiv=111

  バイトコード命令．ddiv

- public static final int opc_irem=112

  バイトコード命令．irem

- public static final int opc_lrem=113

  バイトコード命令．lrem

- public static final int opc_frem=114

  バイトコード命令．frem

- public static final int opc_drem=115

  バイトコード命令．drem

- public static final int opc_ineg=116

  バイトコード命令．ineg

- public static final int opc_lneg=117

  バイトコード命令．lneg

- public static final int opc_fneg=118

  バイトコード命令．fneg

- public static final int opc_dneg=119

  バイトコード命令．dneg

- public static final int opc_ishl=120

  バイトコード命令．ishl

- public static final int opc_lshl=121

  バイトコード命令．lshl

- public static final int opc_ishr=122

  バイトコード命令．ishr

- public static final int opc_lshr=123

  バイトコード命令．lshr

- public static final int opc_iushr=124

  バイトコード命令．iushr

- public static final int opc_lushr=125

  バイトコード命令．lushr

- public static final int opc_iand=126

  バイトコード命令．iand

- public static final int opc_land=127

  バイトコード命令．land

- public static final int opc_ior=128

  バイトコード命令．ior

- public static final int opc_lor=129

  バイトコード命令．lor

- public static final int opc_ixor=130

  バイトコード命令．ixor

- public static final int opc_lxor=131

  バイトコード命令．lxor

- public static final int opc_iinc=132

  バイトコード命令．iinc

- public static final int opc_i2l=133

  バイトコード命令．i2l

- public static final int opc_i2f=134

  バイトコード命令．i2f

- public static final int opc_i2d=135

  バイトコード命令．i2d

- public static final int opc_l2i=136

  バイトコード命令．l2i

- public static final int opc_l2f=137

  バイトコード命令．l2f

- public static final int opc_l2d=138

  バイトコード命令．l2d

- public static final int opc_f2i=139

  バイトコード命令．f2i

- public static final int opc_f2l=140

  バイトコード命令．f2l

- public static final int opc_f2d=141

  バイトコード命令．f2d

- public static final int opc_d2i=142

  バイトコード命令．d2i

- public static final int opc_d2l=143

  バイトコード命令．d2l

- public static final int opc_d2f=144

  バイトコード命令．d2f

- public static final int opc_i2b=145

  バイトコード命令．i2b

- public static final int opc_i2c=146

  バイトコード命令．i2c

- public static final int opc_i2s=147

  バイトコード命令．i2s

- public static final int opc_lcmp=148

  バイトコード命令．lcmp

- public static final int opc_fcmpl=149

  バイトコード命令．fcmpl

- public static final int opc_fcmpg=150

  バイトコード命令．fcmpg

- public static final int opc_dcmpl=151

  バイトコード命令．dcmpl

- public static final int opc_dcmpg=152

  バイトコード命令．dcmpg

- public static final int opc_ifeq=153

  バイトコード命令．ifeq

- public static final int opc_ifne=154

  バイトコード命令．ifne

- public static final int opc_iflt=155

  バイトコード命令．iflt

- public static final int opc_ifge=156

  バイトコード命令．ifge

- public static final int opc_ifgt=157

  バイトコード命令．ifgt

- public static final int opc_ifle=158

  バイトコード命令．ifle

- public static final int opc_if_icmpeq=159

  バイトコード命令．if_icmpeq

- public static final int opc_if_icmpne=160

  バイトコード命令．if_icmpne

- public static final int opc_if_icmplt=161

  バイトコード命令．if_icmplt

- public static final int opc_if_icmpge=162

  バイトコード命令．if_icmpge

- public static final int opc_if_icmpgt=163

  バイトコード命令．if_icmpgt

- public static final int opc_if_icmple=164

  バイトコード命令．if_icmple

- public static final int opc_if_acmpeq=165

  バイトコード命令．if_acmpeq

- public static final int opc_if_acmpne=166

  バイトコード命令．if_acmpne

- public static final int opc_goto=167

  バイトコード命令．goto

- public static final int opc_jsr=168

  バイトコード命令．jsr

- public static final int opc_ret=169

  バイトコード命令．ret

- public static final int opc_tableswitch=170

  バイトコード命令．tableswitch

- public static final int opc_lookupswitch=171

  バイトコード命令．lookupswitch

- public static final int opc_ireturn=172

  バイトコード命令．ireturn

- public static final int opc_lreturn=173

  バイトコード命令．lreturn

- public static final int opc_freturn=174

  バイトコード命令．freturn

- public static final int opc_dreturn=175

  バイトコード命令．dreturn

- public static final int opc_areturn=176

  バイトコード命令． areturn

- public static final int opc_return=177

  バイトコード命令． return

- public static final int opc_getstatic=178

  バイトコード命令． getstatic

- public static final int opc_putstatic=179

  バイトコード命令． putstatic

- public static final int opc_getfield=180

  バイトコード命令． getfield

- public static final int opc_putfield=181

  バイトコード命令． putfield

- public static final int opc_invokevirtual=182

  バイトコード命令． invokevirtual

- public static final int opc_invokespecial=183

  バイトコード命令． invokespecial

- public static final int opc_invokestatic=184

  バイトコード命令． invokestatic

- public static final int opc_invokeinterface=185

  バイトコード命令． invokeinterface

- public static final int opc_xxxunusedxxx=186

  バイトコード命令． xxxunusedxxx

- public static final int opc_new=187

  バイトコード命令． new

- public static final int opc_newarray=188

  バイトコード命令．newarray

- public static final int opc_anewarray=189

  バイトコード命令．anewarray

- public static final int opc_arraylength=190

  バイトコード命令．arraylength

- public static final int opc_athrow=191

  バイトコード命令．athrow

- public static final int opc_checkcast=192

  バイトコード命令．checkcast

- public static final int opc_instanceof=193

  バイトコード命令．instanceof

- public static final int opc_monitorenter=194

  バイトコード命令．monitorenter

- public static final int opc_monitorexit=195

  バイトコード命令．monitorexit

- public static final int opc_wide=196

  バイトコード命令．wide

- public static final int opc_multianewarray=197

  バイトコード命令．multianewarray

- public static final int opc_ifnull=198

  バイトコード命令．ifnull

- public static final int opc_ifnonnull=199

  バイトコード命令．ifnonnull

- public static final int opc_goto_w=200

  バイトコード命令．goto_w

- public static final int opc_jsr_w=201

  バイトコード命令．jsr_w

- public static final int opc_breakpoint=202

  バイトコード命令．breakpoint

- public static final int opc_ldc_quick=203

  バイトコード命令．ldc_quick

- public static final int opc_ldc_w_quick=204

  バイトコード命令．ldc_w_quick

- public static final int opc_ldc2_w_quick=205

  バイトコード命令．ldc2_w_quick

- public static final int opc_getfield_quick=206

  バイトコード命令．getfield_quick

- public static final int opc_putfield_quick=207

  バイトコード命令．putfield_quick

- public static final int opc_getfield2_quick=208

  バイトコード命令．getfield2_quick

- public static final int opc_putfield2_quick=209

  バイトコード命令．putfield2_quick

- public static final int opc_getstatic_quick=210

  バイトコード命令．getstatic_quick

- public static final int opc_putstatic_quick=211

  バイトコード命令．putstatic_quick

- public static final int opc_getstatic2_quick=212

  バイトコード命令．getstatic2_quick

- public static final int opc_putstatic2_quick=213

  バイトコード命令．putstatic2_quick

- public static final int opc_invokevirtual_quick=214

  バイトコード命令．invokevirtual_quick

- public static final int opc_invokenonvirtual_quick=215

  バイトコード命令．invokenonvirtual_quick

- public static final int opc_invokesuper_quick=216

  バイトコード命令．invokesuper_quick

- public static final int opc_invokestatic_quick=217

  バイトコード命令．invokestatic_quick

- public static final int opc_invokeinterface_quick=218

  バイトコード命令．invokeinterface_quick

- public static final int opc_invokevirtualobject_quick=219

  バイトコード命令．invokevirtualobject_quick

- public static final int opc_invokeignored_quick=220

  バイトコード命令．invokeignored_quick

- public static final int opc_new_quick=221

  バイトコード命令．new_quick

- public static final int opc_anewarray_quick=222

  バイトコード命令．anewarray_quick

- public static final int opc_multianewarray_quick=223

  バイトコード命令．multianewarray_quick

- public static final int opc_checkcast_quick=224

  バイトコード命令．checkcast_quick

- public static final int opc_instanceof_quick=225

  バイトコード命令．instanceof_quick

- public static final int opc_invokevirtual_quick_w=226

  バイトコード命令．invokevirtual_quick_w

- public static final int opc_getfield_quick_w=227

  バイトコード命令．getfield_quick_w

- public static final int opc_putfield_quick_w=228

  バイトコード命令．putfield_quick_w

- public static final int opc_nonnull_quick=229

  バイトコード命令．nonnull_quick

- public static final int opc_first_unused_index=230

  バイトコード命令．first_unused_index

- public static final int opc_software=254

  バイトコード命令．software

- public static final int opc_hardware=255

  バイトコード命令．hardware

- public static final String opcNames[] = { ”nop”, ”aconst_null”, ”iconst_m1”, ”iconst_0”, ”iconst_1”, ”iconst_2”, ”iconst_3”, ”iconst_4”, ”iconst_5”, ”lconst_0”, ”lconst_1”, ”fconst_0”, ”fconst_1”, ”fconst_2”, ”dconst_0”, ”dconst_1”, ”bipush”, ”sipush”, ”ldc”, ”ldc_w”, ”ldc2_w”, ”iload”, ”lload”, ”fload”, ”dload”, ”aload”, ”iload_0”, ”iload_1”, ”iload_2”, ”iload_3”, ”lload_0”, ”lload_1”, ”lload_2”, ”lload_3”, ”fload_0”, ”fload_1”, ”fload_2”, ”fload_3”, ”dload_0”, ”dload_1”, ”dload_2”, ”dload_3”, ”aload_0”, ”aload_1”, ”aload_2”, ”aload_3”, ”iaload”,

"laload", "faload", "daload", "aaload", "baload", "caload", "saload", "istore", "lstore", "fstore", "dstore", "astore", "istore_0", "istore_1", "istore_2", "istore_3", "lstore_0", "lstore_1", "lstore_2", "lstore_3", "fstore_0", "fstore_1", "fstore_2", "fstore_3", "dstore_0", "dstore_1", "dstore_2", "dstore_3", "astore_0", "astore_1", "astore_2", "astore_3", "iastore", "lastore", "fastore", "dastore", "aastore", "bastore", "castore", "sastore", "pop", "pop2", "dup", "dup_x1", "dup_x2", "dup2", "dup2_x1", "dup2_x2", "swap", "iadd", "ladd", "fadd", "dadd", "isub", "lsub", "fsub", "dsub", "imul", "lmul", "fmul", "dmul", "idiv", "ldiv", "fdiv", "ddiv", "irem", "lrem", "frem", "drem", "ineg", "lneg", "fneg", "dneg", "ishl", "lshl", "ishr", "lshr", "iushr", "lushr", "iand", "land", "ior", "lor", "ixor", "lxor", "iinc", "i2l", "i2f", "i2d", "l2i", "l2f", "l2d", "f2i", "f2l", "f2d", "d2i", "d2l", "d2f", "int2byte", "int2char", "int2short", "lcmp", "fcmpl", "fcmpg", "dcmpl", "dcmpg", "ifeq", "ifne", "iflt", "ifge", "ifgt", "ifle", "if_icmpeq", "if_icmpne", "if_icmplt", "if_icmpge", "if_icmpgt", "if_icmple", "if_acmpeq", "if_acmpne", "goto", "jsr", "ret", "tableswitch", "lookupswitch", "ireturn", "lreturn", "freturn", "dreturn", "areturn", "return", "getstatic", "putstatic", "getfield", "putfield", "invokevirtual", "invokenonvirtual", "invokestatic", "invokeinterface", "xxxunusedxxx", "new", "newarray", "anewarray", "arraylength", "athrow", "checkcast", "instanceof", "monitorenter", "monitorexit", "wide", "multianewarray", "ifnull", "ifnonnull", "goto_w", "jsr_w", "breakpoint" }

オペコードの名前．

- public static final int opcLengths[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 2, 3, 3, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

(3)　コンストラクタ

なし．

(4)　メソッド

なし．

4.2.3 OpenJIT.frontend.asm.ArrayData クラス

public final class ArrayData

(1) 概要

行列データを格納するクラス．

(2) フィールド (変数)

- Type type
- int nargs

(3) コンストラクタ

- public ArrayData(Type type, int nargs)

  機能概要 ArrayData オブジェクトを生成する．

  機能説明 ArrayData オブジェクトの生成・初期化を行う．

  入力データ type (型), nargs (次元)

  出力データ なし．

  処理方式 type を this.type に， nargs を this.nargs に格納する．

  エラー処理 特になし．

(4) メソッド

なし

4.2.4 OpenJIT.frontend.asm.Assembler クラス

public final class Assembler implements Constants

(1) 概要

バイトコードの生成を行う．

(2) フィールド (変数)

- int NOTREACHED = 0 (定数)
- int REACHED = 1 (定数)
- int NEEDED = 2 (定数)
- Label first
- Instruction last
- int maxdepth
- int maxvar
- int maxpc

(3) コンストラクタ

- public Assembler()

  機能概要 Assembler オブジェクトを生成する．

  機能説明 Assembler オブジェクトの生成・初期化を行う．

  入力データ なし．

  出力データ なし．

  処理方式 何もしない．

  エラー処理 特になし．

(4) メソッド

- public void add(Instruction inst)

  機能概要 バイトコード命令を追加する．

  機能説明 バイトコード命令を追加する．

  入力データ 追加する命令．

  出力データ なし．

  処理方式 inst が null でなければ，last.nest に inst を格納し，last に inst を格納する．

  エラー処理 特になし．

- public void add(int where, int opc)

  機能概要 バイトコード命令を追加する．

  機能説明 バイトコード命令を追加する．

  入力データ where (行番号), opc (オペコード)

  出力データ なし．

  処理方式 新しい Instruction オブジェクトを生成して，add メソッドを呼び出す．

  エラー処理 特になし．

- public void add(int where, int opc, Object obj)

  機能概要 バイトコード命令を追加する．

  機能説明 バイトコード命令を追加する．

  入力データ where (行番号), opc (オペコード), obj (オペランド)

  出力データ なし．

  処理方式 新しい Instruction オブジェクトを生成して，add メソッドを呼び出す．

エラー処理 特になし．

- void optimize(Environment env, Label lbl)

  機能概要 バイトコード命令を最適化する．

  機能説明 バイトコード命令を最適化し，到達しないコードをマークする．

  入力データ env (環境を格納するEnvironmentオブジェクト), lbl (ラベル)

  出力データ なし．

  処理方式 lbl.pcにREACHED (定数)を代入する．lbl.nextから順にバイトコード命令をたどって，以下の処理を行う．

  inst.pcが，NOTREACHED (定数)ならば，inst.optimizeメソッドを呼び出し，inst.pcにREACHED (定数)を格納する．REACHEDなら，終了．

  inst.opcが，opc_label, opc_deadの場合，inst.pcがREACHEDなら，NOTREACHEDに書き換える．

  inst.opcがopc_ifeq, opc_ifne, opc_ifgt, opc_ifge, opc_iflt, opc_ifle, opc_if_icmpeq, opc_if_icmpne, opc_if_icmpgt, opc_if_icmpge, opc_if_icmplt, opc_if_icmple, opc_if_icmplt, opc_if_acmpeq, opc_if_icmpne, opc_ifnull, opc_ifnonnull, opc_jsrの場合，inst.valueに関してoptimizeメソッドを呼び出す．

  inst.opcがopc_gotoの場合，inst.valueに関してoptimizeメソッドを呼び出し，戻ってきたら終了．

  inst.opcがopc_ret, opc_return, opc_ireturn, opc_lreturn, opc_freturn, opc_dreturn, opc_areturn, opc_athrowの場合，終了．

  エラー処理 特になし．

- boolean eliminate()

  機能概要 到達しないバイトコード命令を削除する．

  機能説明 到達しないバイトコード命令を削除する．

  入力データ なし．

  出力データ 変更があったか否かの真偽値．

処理方式 boolean 型の変数 change を false に初期化する．Instruction prev を first に初期化する．バイトコード列を順番にたどりながら，以下の処理を行う．

inst.pc が NOTREACHED でなければ，prev.next に inst を格納し，prev に inst を格納し，inst.pc に NOTREACHED を格納する．そうでない場合は，change を true にする．

最後に，first.pc に NOTREACHED を格納し，prev.next に null を格納し，change の値を返す．

エラー処理 特になし．

- public void optimize(Environment env)

機能概要 バイトコード命令を最適化する．

機能説明 バイトコード命令を optimize メソッド，eliminate メソッドを呼び出して最適化する．

入力データ env (環境を格納する Environment オブジェクト)

出力データ なし．

処理方式 eliminate メソッドを呼び出し，その戻り値が false になるまで，optimize メソッドを呼び続ける．

エラー処理 特になし．

4.2.5 OpenJIT.frontend.asm.Instruction クラス

public class Instruction implements Constants

(1) 概要

バイトコード命令 1 個を表現するクラス .

(2) フィールド (変数)

- int where
- int pc
- int opc
- Object value
- Instruction next

(3) コンストラクタ

- public Instruction(int where, int opc, Object value)

  機能概要 Instruction オブジェクトを生成する .

  機能説明 Instruction オブジェクトの生成・初期化を行う .

  入力データ where (行番号), opc (オペコード), value (オペランド)

  出力データ なし .

  処理方式 this.where に where, this.opc に opc, this.value に value を格納する .

  エラー処理 特になし .

(4) メソッド

- public int getOpcode()

  機能概要 オペコードのアクセッサメソッド .

機能説明 オペコードとして，opc を返す．

入力データ なし．

出力データ オペコード．

処理方式 opc を返す．

エラー処理 特になし．

- public Object getValue()

  機能概要 オペランドのアクセッサメソッド．

  機能説明 オペランドとして，value を返す．

  入力データ なし．

  出力データ オペランド．

  処理方式 value を返す．

  エラー処理 特になし．

- public void setValue(Object value)

  機能概要 オペランドを設定するメソッド．

  機能説明 与えられたオペランドを value に設定する．

  入力データ value (オペランド)

  出力データ なし．

  処理方式 this.value に value を格納する．

  エラー処理 特になし．

- public String toString()

  機能概要 バイトコード命令を文字列に変換する．

  機能説明 バイトコード命令を文字列に変換する．

  入力データ なし．

  出力データ 文字列．

処理方式 opc が opc_try の場合，“try *value.getEndLabel.().hashCode()*” という形式の文字列を生成して，返す．

opc が opc_dead の場合，“dead” という文字列を返す．

opc が opc_iinc の場合，“*opcNames[opc] value[0], value[1]*” という形式の文字列を生成して，返す．

それ以外の場合，value が null なら，opcNames[opc] の文字列を返す．

value が Label クラスのインスタンスならば，“*opcNames[opc] value.toString()*” という形式の文字列を生成して，返す．

value が Instruction クラスのインスタンスならば，“*opcNames[opc] value.hashCode()*” という形式の文字列を生成して，返す．

value が String クラスのインスタンスならば，“*opcNames[opc]* “*value*” ” という形式の文字列を生成して，返す．

それ以外の場合，“*opcNames[opc] value*” という形式の文字列を生成して，返す．

エラー処理 特になし．

4.2.6 OpenJIT.frontend.asm.Label クラス

public final class Label extends Instruction

(1) 概要

バイトコード命令の内，ラベル命令を表現するクラス．

(2) フィールド (変数)

- static int labelCount = 0
- int ID
- int depth
- FieldDefinition[] locals

(3) コンストラクタ

- public Label()

  機能概要 Label オブジェクトを生成する．

  機能説明 スーパークラスの Instruction クラスで定義されているコンストラクタを用いて，Label オブジェクトの生成・初期化を行う．

  入力データ なし．

  出力データ なし．

  処理方式 super メソッドを引数 0, opc_label, null で呼び出す．labelCount に 1 足して，this.ID に格納する．

  エラー処理 特になし．

(4) メソッド

- Label getDestination()

機能概要 ジャンプ先を取得する．

機能説明 goto 命令へのジャンプを省略し，その goto 命令のジャンプ先のラベルへの直接ジャンプにしてジャンプ先を取得する．無限ループを防ぐために，depth フィールドを用いて段数制限を行う．

入力データ なし．

出力データ ジャンプ先のラベル．

処理方式 ローカル変数 Label lbl に this を格納する．next が null でなく，next が this でなく，depth が 0 の場合，以下の処理を行う．

depth を 1 にする．

next.opc が opc_label の場合，next.getDestination メソッドを呼び出し，戻り値を lbl に格納する．

next.opc が opc_goto の場合，next.value.getDestination メソッドを呼び出し，戻り値を lbl に格納する．

next.opc が opc_ldc, opc_ldc_w の場合，next.value が Integer クラスのインスタンスであれば，変数 inst に next.next を格納する．

inst.opc が opc_label なら，inst.getDestination().next を inst に格納する．

inst.opc が opc_ifeq なら，next.value.intValue メソッドを呼び出し，その戻り値が 0 なら，inst.value を lbl に格納し，そうでなければ，新しいラベル lbl を生成して，lbl.next に inst.next を格納し，inst.next に lbl を格納する．しかる後に，lbl.getDestination メソッドを呼び出し，戻り値を lbl に格納する．

inst.opc が opc_ifne なら，next.value.intValue メソッドを呼び出し，その戻り値が 0 なら，新しいラベル lbl を生成して，lbl.next に inst.next を格納し，inst.next に lbl を格納し，そうでなければ，inst.value を lbl に格納する．しかる後に，lbl.getDestination メソッドを呼び出し，戻り値を lbl に格納する．

depth を 0 にする．

最後に lbl を返す．

エラー処理 特になし．

- public String toString()

  機能概要 Label オブジェクトを文字列にする．

  機能説明 Label オブジェクトを文字列にする．

  入力データ なし．

  出力データ 文字列．

  処理方式 ラベルの文字列として，“$\$ID$: stack=$value$” という形式の文字列を生成して，返す．

  エラー処理 特になし．

4.3 OpenJIT.frontend.discompiler パッケージ

- インターフェース
  - OpenJIT.frontend.discompiler.ASTFactory インターフェース
  - OpenJIT.frontend.discompiler.CFAConstants インターフェース
  - OpenJIT.frontend.discompiler.MethodInformation インターフェース
  - OpenJIT.frontend.discompiler.SymbolicConstants インターフェース
- 抽象クラス
  - OpenJIT.frontend.discompiler.DTVisitor 抽象クラス
  - OpenJIT.frontend.discompiler.Metaclass 抽象クラス
- クラス
  - OpenJIT.frontend.discompiler.Annotation クラス
  - OpenJIT.frontend.discompiler.AnnotationAnalyzer クラス
  - OpenJIT.frontend.discompiler.BBABasicBlock クラス
  - OpenJIT.frontend.discompiler.BasicBlock クラス
  - OpenJIT.frontend.discompiler.BasicBlockAnalyzer クラス
  - OpenJIT.frontend.discompiler.BranchTableEntry クラス
  - OpenJIT.frontend.discompiler.BytecodeParser クラス
  - OpenJIT.frontend.discompiler.CFAFlag クラス
  - OpenJIT.frontend.discompiler.CFGNode クラス
  - OpenJIT.frontend.discompiler.Case クラス
  - OpenJIT.frontend.discompiler.ClassSignature クラス
  - OpenJIT.frontend.discompiler.ControlFlowAnalyzer クラス
  - OpenJIT.frontend.discompiler.ControlFlowGraph クラス
  - OpenJIT.frontend.discompiler.DTNode クラス

- OpenJIT.frontend.discompiler.DefaultASTFactory クラス
- OpenJIT.frontend.discompiler.Discompiler クラス
- OpenJIT.frontend.discompiler.DominatorTree クラス
- OpenJIT.frontend.discompiler.ExpressionAnalyzer クラス
- OpenJIT.frontend.discompiler.FlowEdge クラス
- OpenJIT.frontend.discompiler.LoopHead クラス
- OpenJIT.frontend.discompiler.NameAndType クラス
- OpenJIT.frontend.discompiler.RuntimeMethodInformation クラス
- OpenJIT.frontend.discompiler.Switch クラス
- OpenJIT.frontend.discompiler.SymbolicStack クラス
- OpenJIT.frontend.discompiler.SymbolicValue クラス
- OpenJIT.frontend.discompiler.VMInstruction クラス

• 例外

- OpenJIT.frontend.discompiler.DiscompilerError クラス

4.3.1 OpenJIT.frontend.discompiler.ASTFactory インターフェース

public interface ASTFactory

(1) 概要

このインターフェースは，ディスコンパイルの結果として出力する AST の形式を定めるためのもので，AST の種類ごとにインターフェースメソッドとして用意されたファクトリメソッドを定義することにより，ユーザの求める AST オブジェクトの形式をディスコンパイラの出力として得ることができる．

(2) フィールド (変数)

なし．

(3) コンストラクタ

なし．

(4) メソッド

- public boolean isAssignExpression(Expression exp)

  機能概要 Expression オブジェクト exp が AssignExpression であるかどうかを判定する．

  機能説明 与えられた Expression オブジェクト exp が AssignExpression であるかどうかを判定する機能を，AST の形式に依存しない形でディスコンパイラが使用するためのインターフェース．

  入力データ exp (判定したい式を表す Expression オブジェクト)

  出力データ boolean 値

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression leftValueOf(Expression exp)

  機能概要 Expression オブジェクト exp を BinaryExpression とみなし，その左の部分式を返す．

  機能説明 与えられた Expression オブジェクト exp が BinaryExpression であるとみなしてその左の部分式を取り出す機能を，AST の形式に依存しない形でディスコンパイラが使用するためのインターフェース．

  入力データ exp (取り出したい部分式を含む Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression rightValueOf(Expression exp)

  機能概要 Expression オブジェクト exp を BinaryExpression とみなし，その右の部分式を返す．

  機能説明 与えられた Expression オブジェクト exp が BinaryExpression であるとみなしてその右の部分式を取り出す機能を，AST の形式に依存しない形でディスコンパイラが使用するためのインターフェース．

  入力データ exp (取り出したい部分式を含む Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public boolean isConstantOne(Expression exp)

  機能概要 Expression オブジェクト exp が 1 と等しい定数を表す式であるかどうかを判定する．

  機能説明 与えられた Expression オブジェクト exp が 1 と等しい定数を表す式であるかどうかを判定する機能を，AST の形式に依存しない形でディスコンパイラが使用するためのインターフェース．

  入力データ exp (判定したい式を表す Expression オブジェクト)

  出力データ boolean 値

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public boolean isConstantZero(Expression exp)

  機能概要 Expression オブジェクト exp が 0 と等しい定数を表す式であるかどうかを判定する．

機能説明 与えられたExpressionオブジェクトexpが0と等しい定数を表す式であるかどうかを判定する機能を，ASTの形式に依存しない形でディスコンパイラが使用するためのインターフェース．

入力データ exp (判定したい式を表すExpressionオブジェクト)

出力データ boolean値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public boolean isConstantMinusOne(Expression exp)

機能概要 Expressionオブジェクトexpが-1と等しい定数を表す式であるかどうかを判定する．

機能説明 与えられたExpressionオブジェクトexpが-1と等しい定数を表す式であるかどうかを判定する機能を，ASTの形式に依存しない形でディスコンパイラが使用するためのインターフェース．

入力データ exp (判定したい式を表すExpressionオブジェクト)

出力データ boolean値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression toPreExpression(Expression exp)

機能概要 Expressionオブジェクトexpを等価なPreDecExpressionオブジェクトあるいはPreIncExpressionオブジェクトに作り替える．

機能説明 与えられたExpressionオブジェクトexpを意味的に等価なPreDecExpressionオブジェクトあるいはPreIncExpressionオブジェクトに作り替える機能を，ASTの形式に依存しない形でディスコンパイラが使用するためのインターフェース．

入力データ exp (Pre形式に変更したい式を表すExpressionオブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression toPostExpression(Expression exp)

  機能概要 Expression オブジェクト exp を等価な PostDecExpression オブジェクトあるいは PostIncExpression オブジェクトに作り替える．

  機能説明 与えられた Expression オブジェクト exp を意味的に等価な PostDecExpression オブジェクトあるいは PostIncExpression オブジェクトに作り替える機能を，AST の形式に依存しない形でディスコンパイラが使用するためのインターフェース．

  入力データ exp (Post 形式に変更したい式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression invertCondition(Expression exp)

  機能概要 Expression オブジェクト exp を逆の論理を持つ式を表す Expression オブジェクトに作り替える．

  機能説明 与えられた Expression オブジェクト exp を論理式を表すものと見なし，論理的に逆の意味を表す Expression オブジェクトに作り替えるための機能を，AST の形式に依存しない形でディスコンパイラが使用するためのインターフェース．

  入力データ exp (逆の論理に変更したい式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Identifier identifier(String name)

  機能概要 文字列 name と等しい名前を持つ Identifier オブジェクトを返す．

  機能説明 与えられた String オブジェクト name と等しい名前を持つ Identifier オブジェクトをユーザの要求する形式で作り出すためのファクトリメソッド．

  入力データ name (求める Identifier オブジェクトの持つべき名前)

  出力データ Identifier オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression arrayType(int atype)

  機能概要 VM の命令 newarray で使用されている配列の型を表す整数値に相当する型を表す Expression オブジェクトを返す．

  機能説明 VM の命令 newarray で使用されている配列の型を表す整数値 atype に対応する型を表す式をユーザ定義の AST 表現で得るためにディスコンパイラが使用するファクトリメソッド．

  入力データ atype (VM 命令 newarray のオペランド)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression arrayType(String sig)

  機能概要 VM の命令 anewarray および multianewarray で使用される配列の型を表す文字列に相当する型を表す Expression オブジェクトを返す．

機能説明 VM の命令 anewarray および multianewarray で使用される配列の型を表す文字列を解釈し，その型を表現するオブジェクトをユーザ定義の AST 形式で得るためにディスコンパイラが使用するファクトリメソッド．

入力データ sig (配列の型を表す文字列)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public int arrayDepth(String signature)

  機能概要 配列の型を表す文字列 signature の表す配列の深さを返す．

  機能説明 配列の型を表す文字列 signature を解釈し，その配列の深さを整数値として返す．

  入力データ signature (配列の型を表す文字列)

  出力データ int 値

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression classType(String className)

  機能概要 文字列で表されたクラス名に相当する型を表す Expression オブジェクトを返す．

  機能説明 与えられた文字列 className を完全な (qualified な) クラス名と見なし，そのクラスの型を表すユーザ定義の AST オブジェクトをディスコンパイラが得るために使用するファクトリメソッド．

  入力データ className (完全なクラス名)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression booleanExpression(boolean value)

  機能概要 boolean の値 value を表す BooleanExpression を返す．

  機能説明 boolean の値 value を表す BooleanExpression をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ value (boolean 値)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression byteExpression(byte value)

  機能概要 byte の値 value を表す ByteExpression を返す．

  機能説明 byte の値 value を表す ByteExpression をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ value (byte 値)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression charExpression(char value)

  機能概要 char の値 value を表す CharExpression を返す．

  機能説明 char の値 value を表す CharExpression をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ value (char 値)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression shortExpression(short value)

機能概要 short の値 value を表す ShortExpression を返す．

機能説明 short の値 value を表す ShortExpression をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ value (short 値)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression doubleExpression(double value)

機能概要 double の値 value を表す DoubleExpression を返す．

機能説明 double の値 value を表す DoubleExpression をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ value (double 値)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression floatExpression(float value)

機能概要 float の値 value を表す FloatExpression を返す．

機能説明 float の値 value を表す FloatExpression をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ value (float 値)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression intExpression(int value)

機能概要 int の値 value を表す IntExpression を返す．

機能説明 int の値 value を表す IntExpression をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ value (int 値)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression longExpression(long value)

機能概要 long の値 value を表す LongExpression を返す．

機能説明 long の値 value を表す LongExpression をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ value (long 値)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression stringExpression(String value)

  機能概要 String の値 value を表す StringExpression を返す．

  機能説明 String の値 value を表す StringExpression をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ value (String 値)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression nullExpression()

  機能概要 null を表す式 NullExpression を返す．

  機能説明 null を表す式 NullExpression をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ なし．

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression arrayAccessExpression(Expression right, Expression index)

  機能概要 AST の[] 式ノードを表す Expression オブジェクトを返す．

  機能説明 配列を表す式 right の index 番目の要素を参照することを意味する式をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ right (配列を表す Expression オブジェクト), index (index を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression bitNotExpression(Expression right)

  機能概要 AST の~ 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right のビット毎の否定を表す式をユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ right (演算子を適用する式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression convertExpression(Type type, Expression right)

  機能概要 primitive 型の型変換を表す式を返す．

  機能説明 与えられた式 right が primitive 型の式であると見なし，それを与えられた型に型変換することを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ type (型変換する目的の型を表す Type オブジェクト), right (型変換される式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression exprExpression(Expression right)

機能概要 AST の() 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた式 right に対して () を適用した式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ right (式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression fieldExpression(Expression right, Identifier id)

  機能概要 AST の. 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right の持つ id というフィールドを参照することを意味する式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ right (フィールドを持つ式を表す Expression オブジェクト), id (フィールド名を表す Identifier オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression lengthExpression(Expression right)

  機能概要 AST の.length 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた配列式 right の長さを参照する式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ right (配列式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression negativeExpression(Expression right)

機能概要 AST の単項演算子- ノードを表す Expression オブジェクトを返す．

機能説明 与えられた式 right に単項演算子- を適用することを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ right (単項演算子- を適用する式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression notExpression(Expression right)

機能概要 AST の単項演算子! ノードを表す Expression オブジェクトを返す．

機能説明 与えられた式 right に単項演算子! を適用することを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ right (単項演算子! を適用する式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression positiveExpression(Expression right)

機能概要 ASTの単項演算子+ノードを表すExpressionオブジェクトを返す．

機能説明 与えられた式rightに単項演算子+を適用することを表す式を，ユーザ定義のASTオブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ right (単項演算子+を適用する式を表すExpressionオブジェクト)

出力データ Expressionオブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression postDecExpression(Expression right)

機能概要 ASTの後置演算子--のノードを表すExpressionオブジェクトを返す．

機能説明 与えられた式rightに後置演算子--を適用することを表す式を，ユーザ定義のASTオブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ right (後置演算子--を適用する式を表すExpressionオブジェクト)

出力データ Expressionオブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression postIncExpression(Expression right)

機能概要 ASTの後置演算子++のノードを表すExpressionオブジェクトを返す．

機能説明 与えられた式rightに後置演算子++を適用することを表す式を，ユーザ定義のASTオブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ right (後置演算子++を適用する式を表すExpressionオブジェクト)

出力データ Expressionオブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression preDecExpression(Expression right)

  機能概要 ASTの前置演算子--のノードを表すExpressionオブジェクトを返す．

  機能説明 与えられた式rightに前置演算子--を適用することを表す式を，ユーザ定義のASTオブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ right (前置演算子--を適用する式を表すExpressionオブジェクト)

  出力データ Expressionオブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression preIncExpression(Expression right)

  機能概要 ASTの前置演算子++のノードを表すExpressionオブジェクトを返す．

  機能説明 与えられた式rightに前置演算子++を適用することを表す式を，ユーザ定義のASTオブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ right (前置演算子++を適用する式を表すExpressionオブジェクト)

  出力データ Expressionオブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression castExpression(Expression left, Expression right)

  機能概要 ASTのキャストノードを表すExpressionオブジェクトを返す．

機能説明 与えられた式 right を left で表される型にキャストする式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (キャスト先の型を表す Expression オブジェクト), right (キャストされる式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression commaExpression(Expression left, Expression right)

機能概要 AST の,(カンマ) 演算子のノードを表す Expression オブジェクトを返す．

機能説明 カンマで結ばれた複数の式を順に評価し，最後の評価値を式全体の値とする CommaExpression を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (先に評価される式を表す Expression オブジェクト), right (後に評価され，式全体の値を与える Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression instanceOfExpression(Expression left, Expression right)

機能概要 AST の`instanceof` 演算子ノードを表す Expression オブジェクトを返す．

機能説明 与えられた式 left が型を表す式 right のインスタンスであるかどうかを判定する演算子`instanceof` を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (判定するオブジェクトを表す Expression オブジェクト), right (型を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression addExpression(Expression left, Expression right)

機能概要 AST の+ 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子+ を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression divideExpression(Expression left, Expression right)

機能概要 AST の/ 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子/ を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression multiplyExpression(Expression left, Expression right)

  機能概要 AST の* 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子* を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression subtractExpression(Expression left, Expression right)

  機能概要 AST の- 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子- を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression remainderExpression(Expression left, Expression right)

  機能概要 AST の% 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子% を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression andExpression(Expression left, Expression right)

機能概要 AST の&& 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子& を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression orExpression(Expression left, Expression right)

機能概要 AST の|| 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子| を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression bitAndExpression(Expression left, Expression right)

  機能概要 AST の& 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子& を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression bitOrExpression(Expression left, Expression right)

  機能概要 AST の| 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子| を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression bitXorExpression(Expression left, Expression right)

  機能概要 AST の^ 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子^ を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression equalExpression(Expression left, Expression right)

  機能概要 AST の== 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子== を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression notEqualExpression(Expression left, Expression right)

  機能概要 AST の!= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子!= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression shiftLeftExpression(Expression left, Expression right)

  機能概要 AST の<<式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子<<を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression shiftRightExpression(Expression left, Expression right)

  機能概要 AST の>> 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子>> を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression unsignedShiftRightExpression(Expression left, Expression right)

  機能概要 AST の>>> 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子>>> を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression greaterExpression(Expression left, Expression right)

  機能概要 AST の> 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子> を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression greaterOrEqualExpression(Expression left, Expression right)

  機能概要 AST の>= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子>= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression lessExpression(Expression left, Expression right)

機能概要 AST の<式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子<を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression lessOrEqualExpression(Expression left, Expression right)

機能概要 AST の<= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子<= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression assignExpression(Expression left, Expression right)

機能概要 AST の= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression assignAddExpression(Expression left, Expression right)

機能概要 AST の+= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子+= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression assignBitAndExpression(Expression left, Expression right)

  機能概要 AST の&= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子&= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression assignBitOrExpression(Expression left, Expression right)

  機能概要 AST の|= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子|= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression assignBitXorExpression(Expression left, Expression right)

  機能概要 AST の^= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子^= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression assignDivideExpression(Expression left, Expression right)

  機能概要 AST の/= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子/= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression assignMultiplyExpression(Expression left, Expression right)

  機能概要 AST の*= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子*= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression assignRemainderExpression(Expression left, Expression right)

  機能概要 AST の%= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子%= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression assignShiftLeftExpression(Expression left, Expression right)

  機能概要 AST の<<= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子<<= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression assignShiftRightExpression(Expression left, Expression right)

  機能概要 AST の>>= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子>>= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression assignSubtractExpression(Expression left, Expression right)

  機能概要 AST の-= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子-= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression assignUnsignedShiftRightExpression(Expression left, Expression right)

  機能概要 AST の>>>= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子>>>= を適用した結果を表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression conditionalExpression(Expression cond, Expression left, Expression right)

  機能概要 条件演算子を表す AST ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 cond の値により，二つの式 left および right の適当な方を選択するという意味を表す ConditionalExpression の式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ cond (条件を表す Expression オブジェクト), left (条件が真の場合に選ばれる式を表す Expression オブジェクト), right (条件が偽の場合に選ばれる式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression arrayExpression(Expression args[])

  機能概要 初期設定子付きの配列定義を表す Expression オブジェクトを返す．

  機能説明 初期値を与えられた配列定義を表現する AST ノードを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ args (配列の初期値を表す Expression オブジェクトの配列)

  出力データ Expression

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Expression methodExpression(Expression right, Identifier id Expression args[])

  機能概要 式 right で表されるオブジェクトの持つ id というメソッドを引数を args として呼び出す式を表す Expression オブジェクトを返す．

  機能説明 与えられた式 right の表すオブジェクトの持つ id という名前のメソッドを引数を args として呼び出す式を表現する AST ノードを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ right (メソッドを持つオブジェクトを表す Expression オブジェクト), id (メソッド名を表す Identifier オブジェクト), args (引数を表す Expression オブジェクトの配列)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression newArrayExpression(Expression right, Expression args[])

  機能概要 多次元配列を割り当てる AST の new 演算子ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right で表される型の配列を， args 配列で与えられる要素数を持つように割り当てる AST ノードを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ right (配列の型を表す Expression オブジェクト), args (配列の要素数を表す Expression オブジェクトの配列)

出力データ Expression オブジェクト

処理方式

エラー処理 なし．

- public Expression newInstanceExpression(Expression right, Expression args[])

  機能概要 式 right で表されるクラスのオブジェクトを作る AST の new 演算子ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right で表されるクラスのオブジェクトを作り，引数 args と共にそのコンストラクタを呼ぶことを意味する AST の new 演算子ノードを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ right (クラスを表す Expression オブジェクト), args (コンストラクタの引数を表す Expression オブジェクトの配列)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression identifierExpression(Identifier id)

  機能概要 識別子 id を参照する式を表す Expression オブジェクトを返す．

  機能説明 与えられた識別子 id を参照する式を表現する AST ノードを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ id (識別子を表す Identifier オブジェクト)

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression superExpression()

  機能概要 予約語super に対応する AST ノードを表す Expression オブジェクトを返す．

  機能説明 予約語super に対応する AST ノードを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ なし．

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression thisExpression()

  機能概要 予約語this に対応する AST ノードを表す Expression オブジェクトを返す．

  機能説明 予約語this に対応する AST ノードを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ なし．

  出力データ Expression オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Expression typeExpression(Type type)

  機能概要 型 type を表す AST ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた型 type を表現する AST ノードを表す式を，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ type (型を表す Type オブジェクト)

出力データ Expression オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Statement breakStatement(Identifier lbl)

  機能概要 `break lbl` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられたラベル lbl に対する `break` 文を意味する AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ lbl (`break` 文の対象となるラベルを表す Identifier オブジェクト)

  出力データ Statement オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement caseStatement(Expression expr)

  機能概要 式 expr の値を持つ `case` ラベルを意味する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた式 expr の値を持つ `case` ラベルを意味する AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ expr (`case` ラベルの値を表す Expression オブジェクト)

  出力データ Statement オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Statement catchStatement(Expression texpr, Identifier id, Statement body)

  機能概要 一つのcatch節を表現するASTノードを表すStatementオブジェクトを返す．

  機能説明 与えられた式texprで選択される型の例外をidという名前として捕捉するためのcatch節がbodyという内容を持つものであることを意味するASTノードを，ユーザ定義のASTオブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ texpr (捕捉する例外の型を表すExpressionオブジェクト), id (catch節内における例外の識別子を表すIdentifierオブジェクト), body (catch節の本体を表すStatementオブジェクト)

  出力データ Statementオブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement compoundStatement(Statement args[])

  機能概要 複文を表現するASTノードを表すStatementオブジェクトを返す．

  機能説明 与えられたStatementオブジェクトの配列が一つの複文となることを表すASTノードを，ユーザ定義のASTオブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ args (複文に含まれるそれぞれの文を表すStatementの配列)

  出力データ Statementオブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement continueStatement(Identifier lbl)

  機能概要 `continue lbl` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられたラベル lbl に対する`continue` 文を意味する AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ lbl (`continue` 文の対象となるラベルを表す Identifier オブジェクト)

  出力データ Statement オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement declarationStatement(Expression type, Statement args[])

  機能概要 初期化付きの変数定義を表す Statement オブジェクトを返す．

  機能説明 与えられた式 type で表される型の変数を，配列 args の各文によって初期化する変数定義を表す AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ type (宣言される変数の型を表す Expression オブジェクト), args (変数の初期化を行う文を表す Statement オブジェクトの配列)

  出力データ Statement オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement doStatement(Statement body, Expression cond)

  機能概要 do-while 文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 与えられた式 cond を条件として body を繰り返す do-while 文を意味する AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ body (do-while 文の本体を表す Statement オブジェクト), cond (do-while 文の条件を表す Expression オブジェクト)

出力データ Statement オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Statement expressionStatement(Expression expr)

  機能概要 式文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた式 expr を評価する文を表す AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ expr (式文で評価される式を表す Expression オブジェクト)

  出力データ Statement オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement finallyStatement(Statement body, Statement finalbody)

  機能概要 `finally` 節を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた body が `try` 文を表すものと見なし，その `try` 文に finalbody を持つ `finally` 節が付いていることを表現する AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ body (`try` 文を表す Statement オブジェクト), finalbody (`finally` 節の本体を表す Statement オブジェクト)

出力データ Statement オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Statement forStatement(Statement init, Expression cond, Expression inc, Statement body)

  機能概要 `for` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 初期設定文が init で条件が cond，繰り返しの際に実行される文が inc，そして，本体が body であるような`for` 文を表す AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ init (初期設定文を表す Statement オブジェクト), cond (条件を表す Expression オブジェクト), inc (繰り返しの際に実行される文を表す Statement オブジェクト), body (`for` 文の本体を表す Statement オブジェクト)

  出力データ Statement オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement ifStatement(Expression cond, Statement ifTrue, Statement ifFalse)

  機能概要 `if` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 条件を cond とし，条件が真の場合に実行される文が ifTrue で，条件が偽の場合に実行される文が ifFalse であるような`if` 文を表す AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ cond (条件を表す Expression オブジェクト), ifTrue (条件が真の場合に実行される文を表す Statement オブジェクト), ifFalse (条件が偽の場合に実行される文を表す Statement オブジェクト)

出力データ Statement オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Statement returnStatement(Expression expr)

機能概要 式 expr の値を返す`return` 文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 与えられた式 expr の値を返す`return` 文を表す AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ expr (返り値を表す Expression オブジェクト)

出力データ Statement オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Statement switchStatement(Expression expr, Statement args[])

機能概要 式 expr を評価して分岐する`switch` 文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 与えられた式 expr を評価し， Statement の配列 args 内の適当な`case` 文に分岐する`switch` 文を表す AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ expr (分岐に使用する式を表す Expression オブジェクト), args (`switch` 文の本体を表す Statement オブジェクトの配列)

出力データ Statement オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public Statement synchronizedStatement(Expression expr, Statement body)

  機能概要 `synchronized` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた式 expr に関する`synchronized` 文の本体が body であることを表す AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ expr (同期に使用されるオブジェクトを表す Expression オブジェクト), body (`synchronized` 文の本体を表す Statement オブジェクト)

  出力データ Statement オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement throwStatement(Expression expr)

  機能概要 式 expr で表されるオブジェクトを放出する`throw` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた式 expr で表されるオブジェクト (`throwable`) を例外として放出する`throw` 文を表す AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ expr (放出されるオブジェクトを表す Expression オブジェクト)

  出力データ Statement オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement tryStatement(Statement body, Statement args[])

  機能概要 `try` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 配列として与えられた args のそれぞれの要素が`catch` 節であり，本体が body となっている`try` 文を表す AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ body (`try` 文の本体を表す Statement オブジェクト), args (それぞれが一つの`catch` 節となる Statement オブジェクト配列)

  出力データ Statement オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement varDeclarationStatement(Expression expr)

  機能概要 初期設定付きの変数宣言に現れる，それぞれの初期設定文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 初期設定付きの変数宣言に現れる，それぞれの初期設定文を表す AST ノードを，ユーザ定義の AST オブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

  入力データ expr (初期設定を行う式を表す Expression オブジェクト)

  出力データ Statement オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public Statement whileStatement(Expression cond, Statement body)

  機能概要 `while` 文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 条件がcond，本体がbodyであるwhile文を表すASTノードを，ユーザ定義のASTオブジェクトとして作るためにディスコンパイラが使用するファクトリメソッド．

入力データ cond (条件を表すExpressionオブジェクト), body (while文の本体を表すStatementオブジェクト)

出力データ Statementオブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

4.3.2 OpenJIT.frontend.discompiler.Annotation クラス

public class Annotation implements java.io.Serializable

(1) 概要

このクラスは，クラスファイルに付加されるアノテーション情報の構造を定義するものである．

(2) フィールド (変数)

- Node node アノテーションがアタッチされる AST のノード．
- String name アノテーションに対応するメタクラスの名前．
- Metaclass metaobject アノテーションに対応するメタクラスのインスタンス．

(3) コンストラクタ

- public Annotation(String name)

  機能概要 Annotation オブジェクトを生成する．

  機能説明 Annotation オブジェクトを生成する．

  入力データ name (メタクラスの名前)

  出力データ なし．

  処理方式 this.name を name で初期設定する．

  エラー処理 特になし．

(4) メソッド

なし．

4.3.3 OpenJIT.frontend.discompiler.AnnotationAnalyzer クラス

public class AnnotationAnalyzer

(1) 概要

このクラスは，クラスファイルに付加されたアノテーション情報を読み込み，それを Annotation オブジェクトとして構築する機能を提供するものである．

(2) フィールド (変数)

- private LookupHashtable metaclasses 登録されたメタクラスを保持する．

(3) コンストラクタ

なし．

(4) メソッド

- public Annotation analyze(MethodInformation method)

  機能概要 アノテーション情報を表すオブジェクトを構築する．

  機能説明 アノテーション情報を表すオブジェクトを構築する．

  入力データ method (MethodInformation オブジェクト)

  出力データ Annotation オブジェクト

  処理方式 method の annotation メソッドを呼び出すことにより得られる，原始的なアノテーション情報を保持する byte 配列を引数として，readAnnotation メソッドを呼び出し，続けて addMetaobject メソッドを呼ぶことにより，得られたアノテーション情報に対して適切な Metaclass オブジェクトを格納した Annotation オブジェクトを返す．

  エラー処理 なし．

- public Annotation readAnnotation(byte attribute[])

  機能概要 Annotation オブジェクトを生成する．

  機能説明 原始的なアノテーション情報から，Annotation オブジェクトを生成する．

  入力データ attribute (byte 配列)

  出力データ Annotation オブジェクト

  処理方式 attribute を使って ObjectInputStream を作り，その ObjectInputStream オブジェクトの readObject メソッドを呼び出すことにより，Annotation オブジェクトを生成し，それを返す．

  エラー処理 不正なアノテーション情報が与えられた場合，例外 DiscompilerError が放出される．

- public Annotation addMetaobject(Annotation annotation)

  機能概要 適切なメタオブジェクトを格納する．

  機能説明 アノテーション情報に対して適切な Metaclass オブジェクトを格納する．

  入力データ annotation (Annotation オブジェクト)

  出力データ Annotation オブジェクト

  処理方式 metaclasses フィールドから適切なメタクラスの Class オブジェクトを得，その newInstance メソッドを用いて目的のメタオブジェクトを生成する．生成したメタオブジェクトは annotation の metaobject フィールドに格納し，最後にそれを返す．

  エラー処理 メタクラスが不正なクラスであった場合，例外 DiscompilerError を放出する．

- public Class registerAnnotation(Annotation annotation)

  機能概要 メタクラスを登録する．

  機能説明 アノテーション情報に対して適切なメタクラスを登録する．

入力データ annotation (Annotation オブジェクト)

出力データ Class オブジェクト

処理方式 metaclasses フィールドから，annotation の name フィールドで指定された名前の Class オブジェクトを取り出し，それを返す．

エラー処理 なし．

4.3.4 OpenJIT.frontend.discompiler.BBABasicBlock クラス

public class BBABasicBlock extends BasicBlock implements CFAConstants, SymbolicConstants

(1) 概要

このクラスは，コントロールフロー解析の対象となるコントロールフローグラフノードの形式を定義するものである．

(2) フィールド (変数)

- SymbolicStack contents ベーシックブロック単位のディスコンパイルの結果を格納する．

(3) コンストラクタ

- public BBABasicBlock(int begin, int end, int nway, int context)

  機能概要 コントロールフローグラフのノードを生成する．

  機能説明 コントロールフロー解析用の情報を格納することのできるコントロールフローグラフノードを生成する．

  入力データ begin (ベーシックブロックの先頭の命令のアドレス), end (ベーシックブロックの末尾の命令のアドレス), nway (コントロールフローグラフの分岐数), context (ベーシックブロックの属するコントロールフローグラフの種別)

  出力データ なし．

  処理方式 与えられた引数をそのまま使い，super メソッドを呼び出す．

  エラー処理 なし．

- BBABasicBlock(BBABasicBlock original)

  機能概要 コントロールフローグラフノードのダミー生成する．

機能説明 コントロールフローグラフノードのダミーを生成する．

入力データ original (コピー元となる BBABasicBlock オブジェクト)

出力データ なし．

処理方式 ダミーであることを示す引数と共に super メソッドを呼び出す．

エラー処理 なし．

(4) メソッド

- public boolean isConditionalBranch()

  機能概要 ベーシックブロックの末尾が条件ブランチであるかどうかを判定する．

  機能説明 ベーシックブロックの内容である contents の末尾が条件ブランチ命令であるかどうかを判定する．

  入力データ なし．

  出力データ boolean 値

  処理方式 contents の isLastValueConditionalBranch メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public boolean hasConditionalBranchOnly()

  機能概要 ベーシックブロックの中身が条件ブランチだけであるかどうかを判定する．

  機能説明 ベーシックブロックの中身が条件ブランチだけであるかどうかを判定する．

  入力データ なし．

  出力データ boolean 値

  処理方式 contents の hasConditionalBranchOnly メソッドを呼び出し，その結果を返す．

エラー処理 なし．

- Expression removeBranchCondition()

  機能概要 ベーシックブロック末尾の条件ブランチの条件を Expression オブジェクトとして取り出す．

  機能説明 ベーシックブロックの内容である contents の末尾となっている条件ブランチ命令の，条件を Expression オブジェクトとして取り出す．

  入力データ なし．

  出力データ Expression オブジェクト

  処理方式 contents の pop メソッドを呼び出し，最後の命令から条件を表す Expression オブジェクトを取り出し，それを返す．

  エラー処理 ベーシックブロックの末尾が条件ブランチ命令でない場合，例外 DiscompilerError を放出する．

- void pushConditionalBranch(Expression condition)

  機能概要 末尾に条件ブランチ命令を追加する．

  機能説明 ベーシックブロックの内容である contents の末尾に，与えられた条件による条件ブランチ命令を追加する．す．

  入力データ condition (追加される条件ブランチの条件)

  出力データ なし．

  処理方式 条件ブランチ命令を表す SymbolicValue オブジェクトを作り，そのオブジェクトを引数として contents の push メソッドを呼び出す．

  エラー処理 contents の typeOfLastValue フィールドが symbolic_unknown でない場合，例外 DiscompilerError を放出する．

- public void printContents(IndentedPrintStream out)

  機能概要 ベーシックブロックの内容を表示する．

  機能説明 ベーシックブロック単位のディスコンパイルが済んだ状態で，内容をわかりやすく表示する．

入力データ out (出力先を指定する IndentedPrintStream オブジェクト)

出力データ なし．

処理方式 SymbolicStack オブジェクトである contents の内容を一つずつ取り出し，それを表示する．

エラー処理 なし．

- public String toString()

機能概要 ベーシックブロックを識別する簡単な文字列を出力する．

機能説明 ベーシックブロックを特徴づける先頭アドレスあるいはダミー番号と属するコントロールフローグラフの種別を表す文字列を出力する．

入力データ なし．

出力データ String オブジェクト

処理方式 ダミーのベーシックブロックでない場合先頭アドレス，ダミーの場合はダミーであることを表す文字列を作り，それと contexts[context] を連結した文字列を返す．

エラー処理 なし．

4.3.5 OpenJIT.frontend.discompiler.BasicBlock クラス

public class BasicBlock extends CFGNode

(1) 概要

このクラスは，ディスコンパイルの目的となったメソッドのバイトコードの解析により分割された，それぞれのベーシックブロックを表現するオブジェクトを定義するクラスであり，コントロールフローグラフのノードを表す CFGNode のサブクラスである．

(2) フィールド (変数)

- int begin ベーシックブロックの先頭の命令のアドレスを格納する．
- int end ベーシックブロックの末尾の命令のアドレスを格納する．
- int DONT_CARE ベーシックブロックが不明のコントロールフローグラフに属することを意味する．
- int METHOD_MAIN_STREAM ベーシックブロックがメソッド本体のコントロールフローグラフに属することを意味する．
- int EXCEPTION_HANDLER ベーシックブロックが例外ハンドラのコントロールフローグラフに属することを意味する．
- int JSR_TARGET ベーシックブロックが finally 節のコントロールフローグラフに属することを意味する．
- String[] contexts ベーシックブロックの属するコントロールフローグラフの種別を分かりやすく説明する文字列の配列．
- int context ベーシックブロックの属するコントロールフローグラフの種別を表す整数値．
- BasicBlock jsrTarget ベーシックブロックの末尾の命令が `jsr` 命令である場合に，サブルーチンを表すコントロールフローグラフの先頭ノードを格納する．

(3) コンストラクタ

- public BasicBlock(int begin, int end, int nway, int context)

  機能概要 ベーシックブロックを生成する．

  機能説明 ベーシックブロックを生成する．また，ベーシックブロックはコントロールフローグラフのノードを表す CFGNode のサブクラスであるから，スーパクラス CFGNode のコンストラクタを呼び出し，CFGNode オブジェクトとしての初期設定も行う．

  入力データ begin (ベーシックブロックの先頭の命令のアドレス), end (ベーシックブロックの末尾の命令のアドレス), nway (コントロールフローグラフにおける分岐の数), context (ベーシックブロックの属するコントロールフローグラフの種別)

  出力データ なし．

  処理方式 引数を nway として super メソッドを呼び出す． this.begin を begin で初期化し， this.end を end で初期化し， this.context を context で初期化する．

  エラー処理 負の nway が指定された場合に，例外 DiscompilerError を放出する．

(4) メソッド

- public int begin()

  機能概要 begin の値を返す．

  機能説明 begin の値を返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 begin の値を返す．

  エラー処理 なし．

- public int end()

機能概要 end の値を返す．

機能説明 end の値を返すアクセッサメソッド．

入力データ なし．

出力データ int 値

処理方式 end の値を返す．

エラー処理 なし．

- public void printContents(IndentedPrintStream out)

  機能概要 ベーシックブロックの中身を出力する際に呼ばれるオプションのメソッド

  機能説明 サブクラスに対し，独自部分の出力を追加する機会を与えるためのメソッド．

  入力データ out (出力先を表す IndentedPrintStream オブジェクト)

  出力データ なし．

  処理方式 なし．

  エラー処理 なし．

- public void print(IndentedPrintStream out)

  機能概要 ベーシックブロックの中身を出力する．

  機能説明 ベーシックブロックオブジェクトの中身を読みやすく出力する．

  入力データ out (出力先を表す IndentedPrintStream オブジェクト)

  出力データ なし．

  処理方式 まず toString() の結果を出力し，その後に printContents メソッドを呼び出し，最後に，コントロールフローグラフにおいてそのベーシックブロックから直接フローの渡るノードを出力する．

  エラー処理 なし．

- public String toString()

機能概要 ベーシックブロックを識別する簡単な文字列を出力する．

機能説明 ベーシックブロックを特徴づける先頭アドレスと末尾アドレスおよび属するコントロールフローグラフの種別を表す文字列を出力する．

入力データ なし．

出力データ String オブジェクト

処理方式 begin と end および contexts[context] を組み合わせた文字列を返す．

エラー処理 なし．

4.3.6 OpenJIT.frontend.discompiler.BasicBlockAnalyzer クラス

public class BasicBlockAnalyzer extends ControlFlowGraph implements SymbolicConstants

(1) 概要

このクラスはベーシックブロック単位のディスコンパイル処理 (シンボリック実行) を行い，コントロールフローグラフの各ノードがバイトコードで表現されている ControlFlowGraph よりもより抽象度の高いコントロールフローグラフを構築し，また格納する．

(2) フィールド (変数)

- boolean traceMerge 内部的に，実行過程をトレース (表示) するかどうかを決定する変数．
- ASTFactory astFactory ディスコンパイラ全体の出力結果としての AST の表現方法を決定するファクトリオブジェクト．
- Expression[] localFields 同じ番号を持つローカル変数を表す AST ノードが単一の AST ノードオブジェクトを参照することを確保するための配列．
- IntKeyHashtable classRefsByIndex 同じコンスタントプールインデックスを持つクラスへの参照が単一の AST ノードオブジェクトを参照することを確保するためのハッシュテーブル．
- IntKeyHashtable staticFieldRefs 同じスタティックフィールドへの参照が単一の AST ノードオブジェクトを参照することを確保するためのハッシュテーブル．
- IntKeyHashtable dynamicFieldRefs 同じオブジェクトの同じフィールドへの参照が単一の AST ノードオブジェクトを参照することを確保するためのハッシュテーブル．

- Hashtable arrayRefs dup 命令で複製された配列への参照が単一の AST ノードオブジェクトを参照することを確保するためのハッシュテーブル．
- Hashtable fieldTypes フィールド変数がスタティック変数であるかダイナミック変数であるかを記憶するためのハッシュテーブル．
- Boolean booleanTrue boolean 値 true を持つ Boolean オブジェクトを共用するためのオブジェクト．
- Boolean booleanFalse boolean 値 false を持つ Boolean オブジェクトを共用するためのオブジェクト．
- LookupHashtable classRefsByName 同じ名前を持つクラスへの参照が単一の AST ノードオブジェクトを参照することを確保するためのハッシュテーブル．

(3) コンストラクタ

- public BasicBlockAnalyzer(MethodInformation methodInfo, BytecodeParser bytecodeInfo, ASTFactory astFactory)

  機能概要 ベーシックブロック単位の解析器を生成する．

  機能説明 与えられた methodInfo と bytecodeInfo から得られる情報を元に astFactory で指定する形式でベーシックブロック単位のディスコンパイルを行う解析器を生成する．

  入力データ methodInfo (ディスコンパイルの対象となるメソッドの情報を与える MethodInformation インターフェース), bytecodeInfo (命令単位に解析されたバイトコードを与える BytecodeParser オブジェクト), astFactory (ディスコンパイル結果の出力形式を決定する ASTFactory インターフェース)

  出力データ なし．

  処理方式 methodInfo と bytecodeInfo を引数として super メソッドを呼び出し，this.astFactory を astFactory で初期化する．

  エラー処理 なし．

(4) メソッド

- BasicBlock newBasicBlock(int startPC, int endPC, int nWay, int type)

  機能概要 コントロールフローグラフのノードとして使用されるオブジェクトを生成する．

  機能説明 スーパクラスである ControlFlowGraph クラスの作るコントロールフローグラフの各ノードを，ベーシックブロック単位のディスコンパイルで必要な機能を持つオブジェクトとして構築させるためのファクトリメソッド．

  入力データ startPC (ベーシックブロックの先頭の命令のアドレス), endPC (ベーシックブロックの末尾の命令のアドレス), nWay (ベーシックブロックのコントロールフローグラフにおける分岐の数), type (ベーシックブロックの属するコントロールフローグラフの種別)

  出力データ BasicBlock オブジェクト

  処理方式 メソッドに与えられた引数をそのままコンストラクタの引数として，BBABasicBlock オブジェクトを構築し，それを返す．

  エラー処理 なし．

- void doOnEachBasicBlock(BasicBlock block)

  機能概要 ベーシックブロック単位でのディスコンパイルを行う．

  機能説明 与えられた block が BBABasicBlock オブジェクトであると見なして，ベーシックブロック単位での，ディスコンパイルを行い，結果を block の contents として格納する．

  入力データ block (目的のベーシックブロックを表す BasicBlock オブジェクト)

  出力データ なし．

  処理方式 block の保持する先頭の命令のアドレスから末尾の命令のアドレスまでの範囲の VM 命令を，一つづつ取り出してシンボリック実行を行う．途中経過および結果は SymbolicStack オブジェクトとして記憶しておき，ベーシックブロックの全ての命令のシンボリック実行が終了した時点で block の contents として格納する．

エラー処理 なし．

- private SymbolicValue completeValue(VMInstruction instruction, SymbolicValue value[])

  機能概要 新たなSymbolicValueを生成する．

  機能説明 与えられたinstructionとその時点でのスタックの内容を表す配列valueから，そのinstructionをシンボリック実行した結果を表す新たなSymbolicValueを生成する．

  入力データ instruction (シンボリック実行しようとしている命令を表すVMInstructionオブジェクト), value (現在のシンボリックスタックから，instructionのシンボリック実行に必要な数だけ取り出したSymbolicValueオブジェクトの配列)

  出力データ SymbolicValueオブジェクト

  処理方式 instructionごとに適当なASTノードオブジェクトをastFactoryを使って作り，それを元に新たなSymbolicValueを生成する．

  エラー処理 なし．

- SymbolicValue mergeValue(SymbolicValue predicessor, SymbolicValue successor)

  機能概要 二つのSymbolicValueを合わせたものと等価な意味を持つ一つのSymbolicValueオブジェクトを返す．

  機能説明 SymbolicStack上で連続する二つのSymbolicValueオブジェクトが等価な意味を持つ一つのSymbolicValueオブジェクトで表せないかを調べ，可能であれば組み合わせた一つのSymbolicValueオブジェクトを返す．不可能な場合は，nullを返す．

  入力データ predicessor (SymbolicStack上で先行するSymbolicValueオブジェクト), successor (SymbolicStack上でpredicessorの直後に当たるSymbolicValueオブジェクト)

  出力データ SymbolicValueオブジェクト

処理方式 二つのSymbolicValueオブジェクトの組合わせを分類し，値の必要なAssignExpressionかPost形式の式かPre形式の式である場合には，それぞれにマッチする式に作り替えることにより，一つのSymbolicValueとして表現するオブジェクトを生成する．それ以外の場合はnullを返す．

エラー処理 なし．

- private void allocateArgs(SymbolicValue newArray, int type)

機能概要 ArrayExpressionの初期設定部分を表すExpressionオブジェクト配列をSymbolicValueに格納するための配列を確保する．

機能説明 シンボリック実行中，ArrayExpressionを表すと期待されるSymbolicValueが見つかった際に，初期設定部分を表すためのExpressionオブジェクトの配列をSymbolicValueオブジェクトの中に格納しておくための配列を確保する．

入力データ newArray (ArrayExpressionを表すと期待されるSymbolicValueオブジェクト), type (ArrayExpressionの構築する配列の型を表す整数値)

出力データ なし．

処理方式 newArrayの現在の値から目的のArrayExpressionの配列の長さを計算し，その長さでExpression配列をnewArray.argsとして確保する．また，newArray.arrayTypeとしてtypeを格納する．

エラー処理 なし．

- SymbolicValue resolveArrayExpression(SymbolicValue value)

機能概要 SymbolicValueオブジェクトvalueを適当なArrayExpressionあるいはNewArrayExpressionを表現するSymbolicValueオブジェクトに作り替える．

機能説明 命令のシンボリック実行時にはVM命令newarrayおよびanewarrayがArrayExpressionとNewArrayExpressionのどちらを実現するために現れたの決定できない．しかし，後続の命令のシンボリック実行を続けることにより，やがてそれを決定できる段階に到達する．このメソッドは，その時

点で不明の AST ノードを表していた SymbolicValue オブジェクトを適当な AST ノードを表すための SymbolicValue に作り替えるためのメソッドである．

入力データ value (不明の AST ノードを表している SymbolicValue オブジェクト)

出力データ SymbolicValue オブジェクト

処理方式 value.args と value.arrayType を元に適当な ArrayExpression あるいは NewArrayExpression を作り，それを保持する新たな SymbolicValue オブジェクトを返す．

エラー処理 なし．

- SymbolicValue collectValues(SymbolicStack stack, VMInstruction instruction , int args, boolean resolveArrayExpression)

  機能概要 シンボリック実行に必要な数の SymbolicValue オブジェクトをスタックから取り出す．

  機能説明 instruction のシンボリック実行に必要な数の SymbolicValue オブジェクトを stack から取り出し，必要な数が揃った場合には instruction のシンボリック実行を行い，その結果としての SymbolicValue オブジェクトを返す．足りない場合には，値が不足していることを表す SymbolicValue を返す．

  入力データ stack (シンボリック実行で使用する SymbolicStack オブジェクト), instruction (現在注目している命令を表す VMInstruction オブジェクト), args (instruction がスタックから取り出す値の数), resolveArrayExpression (保留していた ArrayExpression と NewArrayExpression の決定が可能な命令であるかどうか)

  出力データ SymbolicValue オブジェクト

  処理方式 スタックから，SymbolicValue オブジェクトを必要な数だけ取り出す．その際，dup 命令などで複製された複数の SymbolicValue が一つの AST ノードとして表現されるべき値を意味しているかどうかを確認し，必

要なら mergeValue メソッドを呼び出す．必要な数だけ SymbolicValue が取り出せた場合には，更に，completeValue メソッドを呼び出して，instruction に対するシンボリック実行を完了する．

エラー処理 ディスコンパイルが不可能な命令並びを見つけた場合，例外 DiscompilerError を放出する．

- Expression classRef(int index)

  機能概要 index で指定されるクラスへの参照を表す Expression オブジェクトを返す．

  機能説明 ConstantPool への index で指定されるクラスへの参照を表現する AST ノードを表す Expression オブジェクトを返す．

  入力データ index (ConstantPool への index)

  出力データ Expression オブジェクト

  処理方式 まず，index からクラス名を表す文字列を求め，その文字列をキーとして classRefsByName から Expression オブジェクトを求め，それを返す．

  エラー処理 なし．

- Expression localField(int slot)

  機能概要 番号で識別されるローカル変数への参照を表す Expression オブジェクトを返す．

  機能説明 与えられた番号 slot で識別されるローカル変数への参照を表す Expression オブジェクトを返す．

  入力データ slot (ローカル変数番号)

  出力データ Expression オブジェクト

  処理方式 localFields 配列を調べ，オブジェクトが構築済であればそれを返す．新たな番号の変数であれば，その番号に応じた Expression オブジェクトを構築し，localFields 配列に格納した後に，そのオブジェクトを返す．

  エラー処理 なし．

- Expression staticFieldRef(int index)

  機能概要 index で選択されるスタティックフィールドへの参照を表す Expression オブジェクトを返す．

  機能説明 ConstantPool への index で選択されるスタティックフィールドへの参照を表す Expression オブジェクトを返す．

  入力データ index (ConstantPool への index)

  出力データ Expression オブジェクト

  処理方式 同じ index に対する Expression オブジェクトが構築済であれば，それを返す．未知のスタティックフィールドへの参照である場合は，その型記述子とクラス名およびフィールド名を解析し，適当な FieldExpression を構築し，ハッシュテーブルに登録した後に，そのオブジェクトを返す．

  エラー処理 なし．

- Expression dynamicFieldRef(Expression obj, int index)

  機能概要 オブジェクト obj の持つ，index で選択されるフィールドへの参照を表す Expression オブジェクトを返す．

  機能説明 オブジェクト obj の持つ，ConstantPool への index で選択されるダイナミックフィールドへの参照を表す Expression オブジェクトを返す．

  入力データ obj (フィールドを持つオブジェクトを表す Expression オブジェクト), index (ConstantPool への index)

  出力データ Expression オブジェクト

  処理方式 同じ obj と index の組に対する Expression オブジェクトが構築済であれば，それを返す．未知のダイナミックフィールドへの参照である場合は，その型記述子とフィールド名を解析し，適当な FieldExpression を構築し，ハッシュテーブルに登録した後に，そのオブジェクトを返す．

  エラー処理 なし．

- public boolean isStaticField(FieldExpression field)

機能概要 field が参照するフィールド変数がスタティック変数かどうかを返す．

機能説明 フィールド変数への参照を表す FieldExpression オブジェクト field の参照する変数がスタティック変数かどうかを返す．

入力データ field (フィールド変数への参照を表す FieldExpression オブジェクト)

出力データ boolean 値

処理方式 FieldExpression オブジェクトが作られる毎にフィールド変数の記憶クラスが登録されるハッシュテーブル fieldTypes を，field をキーとして引き，その結果を返す．

エラー処理 登録された FieldExpression オブジェクト以外を引数として与えられた場合，例外 DiscompilerError を放出する．

- int symbolicType(int index)

機能概要 index で選択されるフィールド変数の型あるいはメソッドの返し値の型を表す SymbolicValue の型を返す．

機能説明 ConstantPool への index で選択されるフィールド変数の型あるいはメソッドの返し値の型を表す SymbolicValue の型を返す．

入力データ index (ConstantPool への index)

出力データ int 値

処理方式 ConstantPool の index 番目のエントリを調べ，その型記述子からフィールド変数あるいはメソッドの返し値の型を求め，それに対応する整数値を返す．

エラー処理 不正な index と共に呼び出された場合，例外 DiscompilerError を放出する．

- Expression arrayRef(Expression obj, Expression index)

機能概要 配列オブジェクトを表す obj とインデックスを表す index の組により選択される配列への参照を表す Expression オブジェクトを返す．

機能説明 配列オブジェクトを表す Expression オブジェクト obj とインデックスを表す Expression オブジェクト index の組により選択される配列への参照を表す Expression オブジェクトを返す．

入力データ obj (配列オブジェクトを表す Expression オブジェクト), index (インデックスを表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 目的の Expression オブジェクトが構築済かどうかを，obj をキーとしてハッシュテーブル arrayRefs を引くことにより決定し，構築済なら，以前に作った Expression オブジェクトを返す．未構築であれば，obj と index の組に対する ArrayAccessExpression を作り，ハッシュテーブルに登録した後にそれを返す．

エラー処理 なし．

- public void printNode(IndentedPrintStream out, BasicBlock block)

機能概要 コントロールフローグラフのノードの中身を表示する．

機能説明 BasicBlock 単位のディスコンパイルの済んだコントロールフローグラフのノードの中身を表示する．

入力データ out (出力先を指定する IndentedPrintStream オブジェクト), block (表示対象となる BasicBlock オブジェクト)

出力データ なし．

処理方式 BasicBlock オブジェクトの持つ print メソッドを呼び出す．

エラー処理 なし．

4.3.7 OpenJIT.frontend.discompiler.BranchTableEntry クラス

public class BranchTableEntry

(1) 概要

VM 命令の tableswitch および lookupswitch に対応する VMInstruction オブジェクトのキーとジャンプ先の組を表すためのクラスである．

(2) フィールド (変数)

- int key ジャンプ先を選択するキー
- int offset ジャンプ先のアドレスを与えるオフセット値

(3) コンストラクタ

- public BranchTableEntry(int key, int offset)

  機能概要 一つのジャンプエントリを表すオブジェクトを生成する．

  機能説明 tableswitch および lookupswitch 命令のジャンプテーブルの各エントリを表すオブジェクトを生成する．

  入力データ key (マッチキー), offset (ジャンプオフセット)

  出力データ なし．

  処理方式 this.key を key で初期化し，this.offset を offset で初期化する．

  エラー処理 なし．

(4) メソッド

- public int key()

  機能概要 key の値を返す．

  機能説明 key の値を返すアクセッサメソッド．

  入力データ なし．

出力データ int 値

処理方式 key の値を返す．

エラー処理 なし．

- public int offset()

  機能概要 offset の値を返す．

  機能説明 offset の値を返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 offset の値を返す．

  エラー処理 なし．

4.3.8 OpenJIT.frontend.discompiler.BytecodeParser クラス

public class BytecodeParser implements Constants

(1) 概要

このクラスは，ディスコンパイルの対象となるバイト列として表現されているメソッドを，より抽象度の高い VM の命令列に変換して，後のディスコンパイルの下準備を行うと共に，ベーシックブロックの先頭となるべき命令を見つけ出し，コントロールフローグラフの構築の下準備も行う，Java メソッドの命令レベルの解析器を実現するものである．

(2) フィールド (変数)

- MethodInformation method コンパイラ基盤モジュールからディスコンパイルの対象となるメソッドの情報を受け取るために使用するインターフェース．
- VMInstruction[] bytecode バイト列の解析結果として得られる，それぞれが VM の命令に対応する VMInstruction オブジェクトの配列を格納する．
- boolean[] leaderFlag あるアドレスに格納されている命令が，ベーシックブロックの先頭となるべき命令かどうかを格納する．
- ExceptionHandler[] exceptions 例外ハンドラに関する情報を格納する．
- Queue queue ブランチの対象となるアドレスを管理するキューであり，コントロールフローの流れる可能性のある命令を全て解析することを確保するために内部的に使用する．

(3) コンストラクタ

- public BytecodeParser(MethodInformation method)

  機能概要 バイト列を解析し，VM 命令単位のオブジェクト列を生成する．

  機能説明 コンパイラ基盤モジュールからメソッドを表すバイト列を読み込み，VM 命令の配列からなるオブジェクトを生成する．

入力データ method (コンパイラ基盤モジュールからメソッドの情報を読み込むためのインターフェース)

出力データ なし．

処理方式 this.method を method で初期化し，内部変数の初期化を行った後に，アドレス 0 から解析を行う．また，例外ハンドラの内容には，メソッド本来のコントロールフローは到達しないので，独立して解析を行う．

エラー処理 なし．

(4) メソッド

- private void parseFrom(int pc)

機能概要 指示されたアドレスから，命令の解析を行う．

機能説明 引数である pc で指示されたアドレスからバイト列の解析を行い，結果をフィールド変数 bytecode に格納する．

入力データ pc (解析を始めるアドレス)

出力データ なし．

処理方式 コンパイラ基盤モジュールからメソッドの内容を表すバイト列を読み込み，VM の命令を一つずつ解析する．結果として，命令単位に作られるオブジェクト VMInstruction を生成し，命令の先頭アドレスをインデックスとして配列 bytecode に格納する．ブランチ命令など，コントロールフローに影響を与える命令を見つけた場合，その飛び先アドレスがベーシックブロックの先頭となることを記憶しておく．

エラー処理 不正な命令列を見つけた場合，例外 DiscompilerError を放出する．

- void enqueue(int pc)

機能概要 コントロールフローの渡るアドレスをキューに入れる．

機能説明 引数 pc がコントロールフローの移るアドレスであると見なし，そのアドレスから始まる命令列が解析されることを確保する．また，コントロールフローの移るアドレスはベーシックブロックの先頭アドレスとなるので，それを leaderFlag 配列に格納する．

入力データ pc (コントロールフローの移るアドレス)

出力データ なし．

処理方式 pc の値を持つ Integer オブジェクトを引数として，内部フィールド queue の enqueue メソッドを呼び出す．また，pc をインデックスとする leaderFlag 配列の要素を true とする．

エラー処理 なし．

- int dequeue()

機能概要 キューからアドレスを一つ取り出す．

機能説明 キューからアドレスを一つ取り出す．

入力データ なし．

出力データ int 値

処理方式 内部フィールド queue の dequeue メソッドを呼び出す．

エラー処理 キューが空の場合，例外 DiscompilerError を放出する．

- public void print(IndentedPrintStream out)

機能概要 解析したメソッドの命令列を表示する．

機能説明 解析したメソッドの命令列を，ディスアセンブラ風にわかりやすく表示する．

入力データ out (出力先を指定する IndentedPrintStream オブジェクト)

出力データ なし．

処理方式 bytecode 配列を先頭から走査し，それぞれの要素 (VMInstruction オブジェクト) に対して，print メソッドを呼び出す．また，bytecode 配列と同じ長さの配列 leaderFlag の同じインデックス位置の要素が true である場合，同じ位置の bytecode 配列の要素は，ベーシックブロックの先頭となるべき命令であることを意味しているので，その事実を明らかにするためにアドレスを含むラベル風の文字列を出力する．

エラー処理 なし．

4.3.9 OpenJIT.frontend.discompiler.CFAConstants インターフェース

public interface CFAConstants

(1) 概要

このインターフェースは，コントロールフロー解析で使用する定数を定義するためのものである．

(2) フィールド (変数)

- int UNKNOWN_NODE ベーシックブロックが不明のノードであることを示す値．
- int LOOP_HEAD_NODE ベーシックブロックが不明のループ構造のヘッダであることを示す値．
- int IF_NODE ベーシックブロックが if 構造をつくるノードであることを示す値．
- int FOR_NODE ベーシックブロックが for 構造をつくるノードであることを示す値．
- int WHILE_NODE ベーシックブロックが while 構造をつくるノードであることを示す値．
- int DO_NODE ベーシックブロックが do-while 構造をつくるノードであることを示す値．
- int SWITCH_NODE ベーシックブロックが switch 構造をつくるノードであることを示す値．
- int CASE_NODE ベーシックブロックが case ラベルを持っていることを示す値．
- String[] typeOfFlag 整数値で表されたノード種別を分かりやすく表す文字列に変換するための配列．

(3)　コンストラクタ

なし．

(4)　メソッド

なし．

4.3.10 OpenJIT.frontend.discompiler.CFAFlag クラス

public class CFAFlag implements CFAConstants

(1) 概要

CFAFlag はコントロールフロー解析の結果として，コントロールフローグラフのノードが Java 言語レベルの何らかの制御構造をつくるノードであるとわかった場合に，対応するドミネータツリーノードに付けられる情報であり，ディスコンパイルの途中結果を表現する．

(2) フィールド (変数)

- boolean completed 対応する構造に関して，必要な解析が全て終わっているかどうかを表す．
- DTNode next Java 言語レベルでその構造全体の次に制御の移るドミネータツリーノードを保持する．
- BBABasicBlock nextBlock Java 言語レベルでその構造全体の次に制御の移るコントロールフローノードを保持する．
- int type 対応する Java 言語構造の種別を表す．

(3) コンストラクタ

- public CFAFlag(int type)

  機能概要 ある Java 言語構造を表す CFAFlag オブジェクトを生成する．

  機能説明 与えられた種別に対応する Java 言語構造を表す CFAFlag オブジェクトを生成する．

  入力データ type (対応する Java 言語構造の種別)

  出力データ なし．

  処理方式 this.type を type で初期設定する．

  エラー処理 なし．

(4) メソッド

- public boolean completed()

  機能概要 completed を返す．

  機能説明 completed を返すアクセッサメソッド．

  入力データ なし．

  出力データ boolean 値

  処理方式 completed を返す．

  エラー処理 なし．

- public int type()

  機能概要 type を返す．

  機能説明 type を返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 type を返す．

  エラー処理 なし．

- public void setCompleted()

  機能概要 completed を true に設定する．

  機能説明 completed を true に設定するアクセッサメソッド．

  入力データ なし．

  出力データ なし．

  処理方式 completed に true を格納する．

  エラー処理 なし．

- public boolean isBreakTarget()

機能概要 break 文の対象となる構造かどうかを返す．

機能説明 対応する Java 言語構造が break 文の対象となる構造かどうかを返す．

入力データ なし．

出力データ boolean 値

処理方式 false を返す．

エラー処理 なし．

- public void print(IndentedPrintStream out)

  機能概要 コントロールフロー解析の中間結果を表示する．

  機能説明 対応する Java 言語構造に則した形で，コントロールフロー解析の中間結果をわかりやすく表示する．

  入力データ out (出力先を指定する IndentedPrintStream オブジェクト)

  出力データ なし．

  処理方式 なし．

  エラー処理 なし．

- public void print(PrefixedPrintStream out)

  機能概要 コントロールフロー解析の中間結果を表示する．

  機能説明 対応する Java 言語構造に則した形で，コントロールフロー解析の中間結果をわかりやすく表示する．

  入力データ out (出力先を指定する PrefixedPrintStream オブジェクト)

  出力データ なし．

  処理方式 なし．

  エラー処理 なし．

- public String toString()

  機能概要 対応する Java 言語構造を特徴づける簡単な文字列を返す．

機能説明 コントロールフロー解析の中間結果として，対応する Java 言語構造を特徴づける簡単な文字列を返す．

入力データ なし．

出力データ String オブジェクト

処理方式 対応する言語構造を表す文字列と，next に相当するノードを説明する文字列を連結した文字列を返す．

エラー処理 なし．

4.3.11 OpenJIT.frontend.discompiler.CFGNode クラス

class CFGNode

(1) 概要

このクラスは，コントロールフローグラフノードの基本的な構造を定義するものである．

(2) フィールド (変数)

- CFGNode[] successor 直接の後続ノードを格納する配列．
- CFGNode[] zeroWay 行き止まりのノードにおける successor オブジェクトを共有するためのオブジェクト．

(3) コンストラクタ

- public CFGNode(int count)

  機能概要 コントロールフローグラフノードを生成する．

  機能説明 コントロールフローグラフノードを，与えられた数の後続ノードを持つものとして生成する．

  入力データ count (後続ノードの数)

  出力データ なし．

  処理方式 successor に必要な大きさの配列を確保する．

  エラー処理 なし．

- CFGNode()

  機能概要 行き止まりのコントロールフローノードを生成する．

  機能説明 行き止まりのコントロールフローノードを生成する．

  入力データ なし．

出力データ なし．

処理方式 successor を zeroWay で初期化する．

エラー処理 なし．

(4) メソッド

- public void clearSuccessor()

機能概要 後続ノード情報を消去する．

機能説明 コントロールフロー解析の結果，不要となった情報を消去する．

入力データ なし．

出力データ なし．

処理方式 successor に zeroWay を格納する．

エラー処理 なし．

4.3.12 OpenJIT.frontend.discompiler.Case クラス

public class Case extends CFAFlag implements CFAConstants

(1) 概要

このクラスは，コントロールフロー解析の結果として判明したswitch構造の中の，caseラベルが付けられるブロックに対し，caseラベルの内容を格納するためのものである．

(2) フィールド (変数)

- Switch container caseラベルの付けられたブロックを直接に含むswitch構造．
- boolean containsDefault ラベルとしてdefaultラベルを含むかどうかを表す．
- Vector labels 付けられたラベル．

(3) コンストラクタ

- public Case(Switch container)

  機能概要 caseラベルの付けられた構造を表すオブジェクトを生成する．

  機能説明 containerというswitch構造に含まれる，caseラベルの付けられたブロックを表すオブジェクトを生成する．

  入力データ container (caseラベルの付けられた構造を直接に含むswitch構造)

  出力データ なし．

  処理方式 定数CASE_NODEを引数としてsuperメソッドを呼び出し，this.containerをcontainerで初期化する．その後に，ラベルを記憶するための領域を確保する．

  エラー処理 なし．

(4) メソッド

- public void addLabel(int label)

  機能概要 ラベルを追加する．

  機能説明 int 値で表されるラベルを追加する．

  入力データ label (ラベル)

  出力データ なし．

  処理方式 int 値 label を表す Integer オブジェクトを生成し，それを引数として labels の addElement メソッドを呼び出す．

  エラー処理 なし．

- public void addDefault()

  機能概要 default ラベルを追加する．

  機能説明 default ラベルを追加する．

  入力データ なし．

  出力データ なし．

  処理方式 containsDefault に true を格納する．

  エラー処理 なし．

- public String toString()

  機能概要 オブジェクトを説明する文字列を返す．

  機能説明 オブジェクトを説明する文字列を返す．

  入力データ なし．

  出力データ String オブジェクト

  処理方式 `*case*` に続けて，保持しているラベルのリストを連結した文字列を返す．

  エラー処理 なし．

4.3.13 OpenJIT.frontend.discompiler.ClassSignature クラス

public class ClassSignature implements Constants

(1) 概要

このクラスは，VM の命令 anewarray および multianewarray で用いられている配列の型記述子を解析し，ディスコンパイラ内で利用しやすい形式に変換するためのものである．

(2) フィールド (変数)

- Hashtable signatures 同じ型記述子に対する ClassSignature オブジェクトを共用するためのテーブル．
- int dimension 型記述子を解析した結果得られる，配列の次元を格納する．
- char sigc 型を表すための文字を格納する．
- String type 型をわかりやすく表す文字列．

(3) コンストラクタ

- private ClassSignature(String signature)

  機能概要 型記述子の内部表現オブジェクトを生成する．

  機能説明 与えられた signature を型記述子と見なして解析し，結果を内部的な表現形式として生成する．

  入力データ signature (型記述子)

  出力データ なし．

  処理方式 StringTokenizer を用いて signature を解析する．

  エラー処理 なし．

(4) メソッド

- public int dimension()

  機能概要 dimension の値を返す．

  機能説明 dimension の値を返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 dimension の値を返す．

  エラー処理 なし．

- public char sigc()

  機能概要 sigc の値を返す．

  機能説明 sigc の値を返すアクセッサメソッド．

  入力データ なし．

  出力データ char 値

  処理方式 sigc の値を返す．

  エラー処理 なし．

- public String type()

  機能概要 type の値を返す．

  機能説明 type の値を返すアクセッサメソッド．

  入力データ なし．

  出力データ String オブジェクト

  処理方式 type の値を返す．

  エラー処理 なし．

- public static ClassSignature lookup(String signature)

機能概要 型記述子に対応する ClassSignature オブジェクトを返す．

機能説明 与えられた signature を解析した結果得られるものと同じ ClassSignature オブジェクトを返す．

入力データ signature (型記述子)

出力データ ClassSignature オブジェクト

処理方式 引数 signature をキーとしてハッシュテーブル signatures を引き，signature に対する ClassSignature オブジェクトが既に生成済みである場合にはそれを返す．未生成の場合は，signature を引数として ClassSignature オブジェクトを生成し，ハッシュテーブル signatures に登録した後に，それを返す．

エラー処理 なし．

- public void print()

機能概要 デバッグ用の表示を行う．

機能説明 デバッグ用の表示を行う．

入力データ なし．

出力データ なし．

処理方式 dimension と type を表示する．

エラー処理 なし．

4.3.14 OpenJIT.frontend.discompiler.ControlFlowAnalyzer クラス

public class ControlFlowAnalyzer

(1) 概要

このクラスは，ディスコンパイルのために用いられるコントロールフロー解析の機能を提供すると共に，解析結果として得られる構造化された (structured な) コントロールフローグラフを保持するためのものである．

(2) フィールド (変数)

- DTNode methodBody 解析の結果得られたコントロールフローグラフの先頭ノードを保持する．

(3) コンストラクタ

- public ControlFlowAnalyzer(ExpressionAnalyzer method)

  機能概要 構造化されたコントロールフローグラフの解析器を生成する．

  機能説明 各ベーシックブロックの式レベルにまで解析されたコントロールフローグラフを解析し，構造化されたコントロールフローグラフを構築する解析器を生成する．

  入力データ method (解析対象のコントロールフローグラフ)

  出力データ なし．

  処理方式 method を引数として recoverControlStructures メソッドを呼び出す．

  エラー処理 なし．

(4) メソッド

- void recoverControlStructures(ExpressionAnalyzer method)

機能概要 制御構造を解析する．

機能説明 構造化されていない (unstructured な) コントロールフローグラフから，構造化された (structured な) コントロールフローグラフを生成する．

入力データ method (解析対象のコントロールフローグラフ)

出力データ なし．

処理方式 コントロールフローグラフから Java の制御構造を実現している部分グラフを見つけ出し，構造化されたグラフに作り替える．

エラー処理 なし．

- void recoverSwitch()

  機能概要 Java 言語の switch 文に相当する制御構造を復元する．

  機能説明 Java 言語の switch 文に相当する制御構造を復元する．

  入力データ なし．

  出力データ なし．

  処理方式 ドミネータツリーを渡り歩き，switch 文を実現している部分木から制御構造を復元する．

  エラー処理 なし．

- void recoverLoop()

  機能概要 Java 言語のループ構造を復元する．

  機能説明 Java 言語のループ構造を復元する．

  入力データ なし．

  出力データ なし．

  処理方式 ドミネータツリーを渡り歩き，Java 言語のループ構造を実現している部分木から制御構造を復元する．

  エラー処理 なし．

- public void print(IndentedPrintStream out)

機能概要 デバッグ用の表示を行う．

機能説明 デバッグ用に保持しているコントロールフローグラフの表示を行う．

入力データ out (出力先を指定する IndentedPrintStream オブジェクト)

出力データ なし．

処理方式 out を引数として methodBody フィールドの printTree メソッドを呼び出す．

エラー処理 なし．

4.3.15 OpenJIT.frontend.discompiler.ControlFlowGraph クラス

public class ControlFlowGraph implements Constants

(1) 概要

このクラスは，ディスコンパイルで使用するコントロールフローグラフを生成する機能を提供すると共に，構造化されていない (unstructured な) コントロールフローグラフを保持するためのものである．

(2) フィールド (変数)

- boolean traceBlock デバッグ用出力を制御するためのフラグ．
- BytecodeParser bytecodeInfo 対象となるバイトコードの情報．
- MethodInformation methodInfo 対象となるメソッドの情報をコンパイラ基盤モジュールから受け取るためのインターフェース．
- BasicBlock methodBody メソッド本体のコントロールフローグラフの先頭ノード．
- BasicBlock[] exceptionHandler 例外ハンドラのコントロールフローグラフの先頭ノードの配列．
- Vector jsrTarget サブルーチンのコントロールフローグラフの先頭ノード．
- IntKeyHashtable cfgNodes ベーシックブロックの先頭アドレスと対応するコントロールフローグラフノードの組を管理するハッシュテーブル．
- IntKeyHashtable printedNodes デバッグ用の表示を行う際に使用する作業用ハッシュテーブル．

(3) コンストラクタ

- public ControlFlowGraph(MethodInformation methodInfo, BytecodeParser bytecodeInfo)

  機能概要 コントロールフローグラフの構築器を生成する．

  機能説明 コントロールフローグラフの構築を行う下準備を行う．

  入力データ methodInfo (コンパイラ基盤モジュールからメソッドの情報を受け取るためのインターフェース), bytecodeInfo (BytecodeParser が解析したバイトコードの情報)

  出力データ なし．

  処理方式 this.bytecodeInfo を bytecodeInfo で， this.methodInfo を methodInfo でそれぞれ初期化する．

  エラー処理 なし．

(4) メソッド

- public void createCFG()

  機能概要 コントロールフローグラフを構築する．

  機能説明 コントロールフローグラフを構築する．

  入力データ なし．

  出力データ なし．

  処理方式 アドレス 0 からメソッド本体のコントロールフローグラフを構築した後に，例外ハンドラのコントロールフローグラフを構築する．また，サブルーチンのコントロールフローグラフは見つけるたびに構築する．

  エラー処理 なし．

- BasicBlock nodeFrom(int from, int type)

  機能概要 指定したコントロールフローグラフノードを返す．

機能説明 指定したアドレスを先頭とするコントロールフローグラフノードを返す．

入力データ from (目的のコントロールフローグラフノードの先頭アドレス), type (目的のノードの属するコントロールフローグラフ種別)

出力データ BasicBlock オブジェクト

処理方式 from をキーとしてハッシュテーブル cfgNodes を引き，目的のノードが既に構築済であればそれを返す．未構築である場合，from を先頭としてバイトコードを走査し，次にベーシックブロックの先頭となる命令の直前の命令までをベーシックブロックとしてコントロールフローグラフを構築し，それを返す．

エラー処理 type として正当な値が与えられない場合，例外 DiscompilerError を放出する．

- BasicBlock newBasicBlock(int startPC, int endPC, int nWay, int type)

  機能概要 コントロールフローグラフのノードを生成する．

  機能説明 コントロールフローグラフのノードとして使用されるオブジェクトを生成する．

  入力データ startPC (ベーシックブロックの先頭の命令のアドレス), endPC (ベーシックブロックの末尾の命令のアドレス), nWay (コントロールフローグラフの分岐数), type (ベーシックブロックの属するコントロールフローグラフの種別)

  出力データ BasicBlock オブジェクト

  処理方式 BasicBlock オブジェクトを生成し，それを返す．

  エラー処理 なし．

- void doOnEachBasicBlock(BasicBlock block)

  機能概要 コントロールフローグラフの各ノードで処理を行う．

  機能説明 コントロールフローグラフの構築の際に，全てのノードで一度ずつ何らかの処理を行う機能を提供する．

入力データ block (対象となるノード)

出力データ なし．

処理方式 なし．

エラー処理 なし．

- public void print(IndentedPrintStream out)

  機能概要 デバッグ用の表示を行う．

  機能説明 デバッグ用に，コントロールフローグラフを表示する．

  入力データ out (出力先を指定する IndentedPrintStream オブジェクト)

  出力データ なし．

  処理方式 out を引数として printCFG メソッドを呼び出す．

  エラー処理 なし．

- public void printCFG(IndentedPrintStream out)

  機能概要 デバッグ用の表示を行う．

  機能説明 デバッグ用に，コントロールフローグラフを表示する．

  入力データ out (出力先を指定する IndentedPrintStream オブジェクト)

  出力データ なし．

  処理方式 コントロールフローグラフに含まれる全てのノードを一度ずつ表示するために，printAtPC メソッドを呼び出す．

  エラー処理 なし．

- public void printAtPC(IndentedPrintStream out, int pc)

  機能概要 デバッグ用に，部分的なコントロールフローグラフを表示する．

  機能説明 デバッグ用に，先頭アドレスで指定したコントロールフローグラフノードを表示する．

  入力データ out (出力先を指定する IndentedPrintStream オブジェクト), pc (目的の部分的なコントロールフローグラフの先頭ノード)

出力データ なし．

処理方式 目的の部分グラフに含まれるノードの内，まだ表示の行われていないものについて，printNode を呼び出す．

エラー処理 なし．

- public void printNode(IndentedPrintStream out, BasicBlock block)

  機能概要 デバッグ用に，コントロールフローグラフノードの中身を表示する．

  機能説明 デバッグ用に，コントロールフローグラフノードの中身を，逆アセンブラ風に表示する．

  入力データ out (出力先を指定する IndentedPrintStream オブジェクト), block (対象となるコントロールフローグラフノード)

  出力データ なし．

  処理方式 bytecodeInfo の持つ bytecode 配列の内，ベーシックブロックに含まれるアドレス空間に対応する部分に含まれる VMInstruction オブジェクトの print メソッドを順に呼び出す．

  エラー処理 なし．

4.3.16 OpenJIT.frontend.discompiler.Discompiler クラス

public class Discompiler

(1) 概要

このクラスは，OpenJIT のフロントエンドの提供する，Java バイトコードのディスコンパイル機能を実現するためのものである．

(2) フィールド (変数)

- BytecodeParser bytecodeInfo ディスコンパイル対象となるメソッドのバイトコードレベルの情報を保持する．
- ExpressionAnalyzer structurelessCFG ディスコンパイル対象となるメソッドの構造化されていないコントロールフローグラフを保持する．
- ControlFlowAnalyzer structuredCFG ディスコンパイル対象となるメソッドの構造化されたコントロールフローグラフを保持する．
- ASTFactory astFactory ディスコンパイルの結果として求める AST の形式を定めるためのファクトリオブジェクトを保持する．
- DefaultASTFactory defaultASTFactory デフォルトの AST 形式として OpenJIT.frontend.tree パッケージで定める形式の AST オブジェクトを生成するファクトリオブジェクトを保持する．

(3) コンストラクタ

- public Discompiler(MethodInformation method, ASTFactory astFactory)

  機能概要 メソッドのディスコンパイルを行う解析器を生成する．

  機能説明 メソッドのディスコンパイルを行う解析器を生成する．

  入力データ method (ディスコンパイルの対象となるメソッドの情報を得るためのインターフェース), astFactory (ディスコンパイラの結果として求める AST の形式を定めるファクトリオブジェクト)

出力データ なし．

処理方式 this.astFactory を astFactory で初期設定し，bytecodeInfo フィールドや structurelessCFG フィールド，structuredCFG フィールドに保持するオブジェクトを生成する．

エラー処理 なし．

- public Discompiler(MethodInformation method)

機能概要 メソッドのディスコンパイルを行う解析器を生成する．

機能説明 メソッドのディスコンパイルを行う解析器を生成する．

入力データ method (ディスコンパイルの対象となるメソッドの情報を得るためのインターフェース)

出力データ なし．

処理方式 method と defaultASTFactory を引数として this メソッドを呼び出す．

エラー処理 なし．

(4) メソッド

- public Node discompile()

機能概要 メソッドのディスコンパイルを行う．

機能説明 メソッドのディスコンパイルを行う．

入力データ なし．

出力データ Node オブジェクト

処理方式 メソッドのディスコンパイルを行い，結果を AST オブジェクトとして返す．

エラー処理 何らかの理由でディスコンパイルに失敗した場合，例外 DiscompilerError を放出する．

- public void print(IndentedPrintStream out)

機能概要 デバッグ用の表示を行う．

機能説明 デバッグ用に，保持しているフィールド等の情報をわかりやすく表示する．

入力データ out (出力先を指定する IndentedPrintStream オブジェクト)

出力データ なし．

処理方式 out を引数として，bytecodeInfo と structurelessCFG および structuredCFG フィールドの print メソッドをそれぞれ呼び出す．

エラー処理 なし．

4.3.17 OpenJIT.frontend.discompiler.DiscompilerError クラス

(1) 概要

このクラスは，ディスコンパイラモジュールにおいて，ディスコンパイル中に起きたエラーの内容を表すクラス．

(2) フィールド (変数)

- Throwable e エラーや例外を保持する．

(3) コンストラクタ

- public DiscompilerError(String msg)

  機能概要 DiscompilerError オブジェクトを生成する．

  機能説明 DiscompilerError オブジェクトを生成する．

  入力データ msg(エラーメッセージ)

  出力データ なし．

  処理方式 スーパクラス (java.lang.Error) のコンストラクタを呼び出した後，e に this を設定する．

  エラー処理 なし．

- public DiscompilerError(Exception e)

  機能概要 DiscompilerError オブジェクトを生成する．

  機能説明 DisompilerError オブジェクトを生成する．

  入力データ e(例外)

  出力データ なし．

  処理方式 e.getMessage を呼び出し，例外のメッセージを得て，スーパクラス (java.lang.Error) のコンストラクタを呼び出す．その後，e に this を設定する．

エラー処理 なし．

(4)　メソッド

なし．

4.3.18 OpenJIT.frontend.discompiler.DTNode クラス

public class DTNode extends BBABasicBlock

(1) 概要

このクラスは，ディスコンパイラがコントロールフロー解析で用いるドミネータツリーの各ノードの構造を定めるものである．

(2) フィールド (変数)

- DTNode immediateDominator イミディエートドミネータを格納する．
- Vector valets ドミネータツリーの子ノードを格納する．
- CFAFlag flag Java 言語レベルでの構造を表す．
- Case label case ラベルを格納する．
- int dummyNo デバッグ用の内部変数．
- int dummy デバッグ用の内部変数．

(3) コンストラクタ

- public DTNode(int begin, int end, int nway, int context)

  機能概要 ドミネータツリーのノードを生成する．

  機能説明 ドミネータツリーのノードを生成する．

  入力データ begin (ベーシックブロックの先頭の命令のアドレス), end (ベーシックブロックの末尾の命令のアドレス), nway (コントロールフローグラフの分岐数), context (ベーシックブロックの属するコントロールフローグラフの種別)

  出力データ なし．

  処理方式 begin，end，nway，context を引数として super メソッドを呼び出した後，valets の記憶領域を確保する．

エラー処理 なし．

- public DTNode(DTNode original, boolean withValets)

  機能概要 ダミーのノードを生成する．

  機能説明 ドミネータツリーの途中に矛盾なく挟み込むためのダミーのノードを別のノードのコピーとして生成する．

  入力データ original (コピー元となるDTNodeオブジェクト), withValets (子ノードをダミーに移すかどうか)

  出力データ なし．

  処理方式 immediateDominatorをoriginalとし，withValetsが真の場合はvaletsをoriginalから取り込み，originalの子ノードのimmediateDominatorが新しく作られるダミーのノードとなるように参照を付け変える．更に，デバッグ用の情報であるダミー番号を割り当て，dummyに格納する．

  エラー処理 なし．

(4) メソッド

- private int dummyNo()

  機能概要 内部的なデバッグ用の情報を与える．

  機能説明 内部的なデバッグ用の情報を与える．

  入力データ なし．

  出力データ int値

  処理方式 それまでに作られたダミーノードの数を返す．

  エラー処理 なし．

- public void addValet(DTNode valet)

  機能概要 子ノードを追加する．

  機能説明 子ノードを追加する．

入力データ valet (子ノードとなるDTNodeオブジェクト)

出力データ なし．

処理方式 valetを引数として，valetsのaddElementメソッドを呼び出す．

エラー処理 なし．

- public boolean removeValet(DTNode valet)

  機能概要 子ノードを取り除く．

  機能説明 子ノードを取り除く．

  入力データ valet (取り除きたい子ノードを表すDTNodeオブジェクト)

  出力データ boolean値

  処理方式 valetを引数として，valetsのremoveElementメソッドを呼び出す．

  エラー処理 なし．

- public int numberOfValets()

  機能概要 子ノードの数を返す．

  機能説明 子ノードの数を返す．

  入力データ なし．

  出力データ int値

  処理方式 valetsのsizeメソッドを呼び出す．

  エラー処理 なし．

- public void traverse(DTVisitor visitor)

  機能概要 ドミネータツリーを渡り歩く．

  機能説明 ドミネータツリーを渡り歩き，各ノードで何らかの処理を行うという機能を提供する．

  入力データ visitor (各ノードで行う処理を決定するDTVisitorオブジェクト)

  出力データ なし．

処理方式 visitor の parentFirst メソッドを呼び出すことにより，渡り歩きの順序を選択し，適合する traversePF あるいは traverseCF を呼び出す．

エラー処理 なし．

- private void traversePF(DTVisitor visitor)

  機能概要 ドミネータツリーを親を先として渡り歩く．

  機能説明 ドミネータツリーを渡り歩き，各ノードで何らかの処理を行う機能を提供する．その際，親ノードでの処理を先に行う．

  入力データ visitor (各ノードで行う処理を決定する DTVisitor オブジェクト)

  出力データ なし．

  処理方式 まず，this を引数として visitor の visit メソッドを呼び出し，その後に，visitor を引数として各子ノードの traverse メソッドを呼び出す．

  エラー処理 なし．

- private void traverseCF(DTVisitor visitor)

  機能概要 ドミネータツリーを子を先として渡り歩く．

  機能説明 ドミネータツリーを渡り歩き，各ノードで何らかの処理を行う機能を提供する．その際，子ノードでの処理を先に行う．

  入力データ visitor (各ノードで行う処理を決定する DTVisitor オブジェクト)

  出力データ なし．

  処理方式 まず，visitor を引数として各子ノードの traverse メソッドを呼びだし，その後に，this を引数として visitor の visit メソッドをび出す．

  エラー処理 なし．

- public void printTree(PrintStream out)

  機能概要 デバッグ用の表示を行う．

  機能説明 デバッグ用の表示を行う．

  入力データ out (出力先を指定する PrintStream オブジェクト)

出力データ なし．

処理方式 out を引数として IndentedPrintStream オブジェクトを生成し，それを引数として printTree メソッドを呼び出す．

エラー処理 なし．

- public void printTree(IndentedPrintStream out)

  機能概要 デバッグ用の表示を行う．

  機能説明 デバッグ用の表示を行う．

  入力データ out (出力先を指定する IndentedPrintStream オブジェクト)

  出力データ なし．

  処理方式 this ノードを識別する文字列を出力した後に，子ノードに関する出力を行う．

  エラー処理 なし．

- public void printTree(PrefixedPrintStream out, String prefix)

  機能概要 デバッグ用の表示を行う．

  機能説明 デバッグ用のわかりやすい表示を行う．

  入力データ out (出力先を指定する PrefixedPrintStream オブジェクト), prefix (ツリーの親子関係を明らかにするための文字列)

  出力データ なし．

  処理方式 prefix を出力した後に，this ノードを識別する文字列を出力し，更に子ノードに関する出力を行う．

  エラー処理 なし．

- void printValets(PrefixedPrintStream out)

  機能概要 子ノードに関する，デバッグ用の表示を行う．

  機能説明 子ノードに関する，デバッグ用の表示を行う．

  入力データ out (出力先を指定する PrefixedPrintStream オブジェクト)

出力データ なし．

処理方式 子ノードのそれぞれについて，printTree メソッドを呼び出す．

エラー処理 なし．

- public String toString()

  機能概要 ノードを識別する簡単な文字列を返す．

  機能説明 ドミネータツリーノードを識別する簡単な文字列を返す．

  入力データ なし．

  出力データ String オブジェクト

  処理方式 self の toString メソッドの結果と label や flag を説明する文字列を連結して返す．

  エラー処理 なし．

- public boolean doesDominate(DTNode node)

  機能概要 node をドミネートしているかどうかを判定する．

  機能説明 与えられた node をドミネートしているかどうかを判定する．

  入力データ node (ドミネートされているかどうかを判定したいノードを表す DTNode オブジェクト)

  出力データ boolean 値

  処理方式 node の immediateDominator フィールドから親を辿っていき，this に到達すれば true を返し，ルートノードに到達したら false を返す．

  エラー処理 なし．

- public int distanceFromRoot()

  機能概要 ノードのドミネータツリーにおけるルートからの距離を返す．

  機能説明 ノードのドミネータツリーにおけるルートからの距離を返す．

  入力データ なし．

出力データ int 値

処理方式 ルートに辿り着くまでの親ノードの数を数え，それを返す．

エラー処理 なし．

4.3.19 OpenJIT.frontend.discompiler.DTVisitor 抽象クラス

public abstract class DTVisitor

(1) 概要

このクラスは，DTNode クラスの提供する，ドミネータツリーを渡り歩き，各ノードに対して何らかの処理を行うという機能について，各ノードで行うべき処理を定義するための構造を提供する．

(2) フィールド (変数)

- boolean traceVisitor デバッグ用出力の制御を行うフラグ．
- boolean advanced 渡り歩きで進展があったかどうかを表すフラグ．

(3) コンストラクタ

- public DTVisitor()

  機能概要 ドミネータツリーの渡り歩きを行うオブジェクトを生成する．

  機能説明 ドミネータツリーの渡り歩きの際に，各ノードで行う処理を決定するオブジェクトを生成する．

  入力データ なし．

  出力データ なし．

  処理方式 advanced を false で初期設定する．

  エラー処理 なし．

(4) メソッド

- public boolean advanced()

  機能概要 その回の渡り歩きで進歩があったかどうかを判定する．

  機能説明 その回の渡り歩きで進歩があったかどうかを判定する．

入力データ なし．

出力データ boolean 値

処理方式 false を返す．

エラー処理 なし．

- public void setAdvanced()

  機能概要 advanced フラグを設定する．

  機能説明 advanced フラグを設定するアクセッサメソッド．

  入力データ なし．

  出力データ なし．

  処理方式 advanced に true を格納する．

  エラー処理 なし．

- public void clearAdvanced()

  機能概要 advanced フラグをクリアする．

  機能説明 advanced フラグをクリアする．

  入力データ なし．

  出力データ なし．

  処理方式 advanced に false を格納する．

  エラー処理 なし．

- public abstract boolean parentFirst()

  機能概要 渡り歩きの際，親ノードを先に処理するかどうかを決定する．

  機能説明 渡り歩きの際，親ノードを先に処理するかどうかを決定する．

  入力データ なし．

  出力データ boolean 値

処理方式 このメソッドは抽象メソッドであり，実際に使用されるメソッドはサブクラスで定義する．

エラー処理 なし．

- public abstract void visit(DTNode visitTo)

  機能概要 それぞれのノードに対し，行うべき処理を記述するためのアブストラクトメソッド．

  機能説明 それぞれのノードに対し，行うべき処理の内容を記述するためのアブストラクトメソッドを提供する．

  入力データ visitTo (対象となるドミネータツリーノード)

  出力データ なし．

  処理方式 このメソッドは抽象メソッドであり，実際に使用されるメソッドはサブクラスで定義する．

  エラー処理 なし．

4.3.20 OpenJIT.frontend.discompiler.DefaultASTFactory クラス

public class DefaultASTFactory implements ASTFactory, Constants

(1) 概要

このクラスは，ディスコンパイラのデフォルトの出力形式としてOpenJIT.frontend.tree パッケージで定義されている AST の形式を採用するための，ASTFactory インターフェースを実装するものである．

(2) フィールド (変数)

- TypeExpression[] simpleTypes 単純な型を表す TypeExpression を保持する配列．
- LookupHashtable classExpressions クラスを参照するための Expression オブジェクトを保持するための配列．
- LookupHashtable classSignatures 配列の型を表す型記述子の解析結果を保持するための配列．
- Expression booleanTrue boolean 定数 true を表す Expression オブジェクトを保持する．
- Expression booleanFalse boolean 定数 false を表す Expression オブジェクトを保持する．
- LookupHashtable byteConstants byte 定数を表す Expression オブジェクトを保持するためのハッシュテーブル．
- LookupHashtable charConstants char 定数を表す Expression オブジェクトを保持するためのハッシュテーブル．
- LookupHashtable shortConstants short 定数を表す Expression オブジェクトを保持するためのハッシュテーブル．

- LookupHashtable intConstants int 定数を表す Expression オブジェクトを保持するためのハッシュテーブル．
- LookupHashtable longConstants long 定数を表す Expression オブジェクトを保持するためのハッシュテーブル．
- LookupHashtable floatConstants float 定数を表す Expression オブジェクトを保持するためのハッシュテーブル．
- LookupHashtable doubleConstants double 定数を表す Expression オブジェクトを保持するためのハッシュテーブル．
- Expression intConstantOne int 定数 1 を表す Expression オブジェクトを保持する．
- Expression intConstantZero int 定数 0 を表す Expression オブジェクトを保持する．
- Expression intConstantMinusOne int 定数 -1 を表す Expression オブジェクトを保持する．
- Expression longConstantOne long 定数 1 を表す Expression オブジェクトを保持する．
- Expression longConstantZero long 定数 0 を表す Expression オブジェクトを保持する．
- Expression longConstantMinusOne long 定数 -1 を表す Expression オブジェクトを保持する．
- Expression floatConstantOne float 定数 1 を表す Expression オブジェクトを保持する．
- Expression floatConstantZero float 定数 0 を表す Expression オブジェクトを保持する．
- Expression floatConstantMinusOne float 定数 -1 を表す Expression オブジェクトを保持する．

- Expression doubleConstantOne double 定数 1 を表す Expression オブジェクトを保持する．
- Expression doubleConstantZero double 定数 0 を表す Expression オブジェクトを保持する．
- Expression doubleConstantMinusOne double 定数 -1 を表す Expression オブジェクトを保持する．
- LookupHashtable stringConstants 文字列定数を表す Expression オブジェクトを保持するハッシュテーブル．
- Expression nullExpression null 定数を表す Expression オブジェクトを保持する．
- Expression constantSuper super を表す Expression オブジェクトを保持する．
- Expression constantThis this を表す Expression オブジェクトを保持する．

(3) コンストラクタ

なし．

(4) メソッド

- public boolean isAssignExpression(Expression exp)

  機能概要 Expression オブジェクト exp が AssignExpression であるかどうかを判定する．

  機能説明 与えられた Expression オブジェクト exp が AssignExpression であるかどうかを判定する機能を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供する．

  入力データ exp (判定したい式を表す Expression オブジェクト)

  出力データ boolean 値

  処理方式 exp の getOp メソッドを呼び出し，ASSIGN と等しいかどうかを調べる．

エラー処理 なし．

- public Expression leftValueOf(Expression exp)

  機能概要 Expression オブジェクト exp を BinaryExpression とみなし，その左の部分式を返す．

  機能説明 与えられた Expression オブジェクト exp が BinaryExpression であるとみなしてその左の部分式を取り出す機能を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供する．

  入力データ exp (取り出したい部分式を含む Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 exp が BinaryExpression クラスのインスタンスである場合は，left メソッドを呼び出し，その結果を返す．

  エラー処理 exp が BinaryExpression クラスのインスタンスでない場合，例外 DiscompilerError を放出する．

- public Expression rightValueOf(Expression exp)

  機能概要 Expression オブジェクト exp を BinaryExpression とみなし，その右の部分式を返す．

  機能説明 与えられた Expression オブジェクト exp が BinaryExpression であるとみなしてその右の部分式を取り出す機能を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供する．

  入力データ exp (取り出したい部分式を含む Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 exp が UnaryExpression クラスのインスタンスである場合は，right メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public boolean isConstantOne(Expression exp)

機能概要 Expression オブジェクト exp が 1 と等しい定数を表す式であるかどうかを判定する．

機能説明 与えられた Expression オブジェクト exp が 1 と等しい定数を表す式であるかどうかを判定する機能を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供する．

入力データ exp (判定したい式を表す Expression オブジェクト)

出力データ boolean

処理方式 exp の getOp メソッドを呼び出し，exp が定数を表す Expression オブジェクトである場合，その定数が 1 かどうかを返す．それ以外は false を返す．

エラー処理 なし．

- public boolean isConstantZero(Expression exp)

機能概要 Expression オブジェクト exp が 0 と等しい定数を表す式であるかどうかを判定する．

機能説明 与えられた Expression オブジェクト exp が 0 と等しい定数を表す式であるかどうかを判定する機能を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供する．

入力データ exp (判定したい式を表す Expression オブジェクト)

出力データ boolean

処理方式 exp の getOp メソッドを呼び出し，exp が定数を表す Expression オブジェクトである場合，その定数が 0 かどうかを返す．それ以外は false を返す．

エラー処理 なし．

- public boolean isConstantMinusOne(Expression exp)

機能概要 Expression オブジェクト exp が -1 と等しい定数を表す式であるかどうかを判定する．

機能説明 与えられたExpressionオブジェクトexpが-1と等しい定数を表す式であるかどうかを判定する機能を，OpenJIT.frontend.treeパッケージの定めるASTに適合するように提供する．

入力データ exp (判定したい式を表すExpressionオブジェクト)

出力データ boolean

処理方式 expのgetOpメソッドを呼び出し，expが定数を表すExpressionオブジェクトである場合，その定数が-1かどうかを返す．それ以外はfalseを返す．

エラー処理 なし．

- public Expression toPreExpression(Expression exp)

機能概要 Expressionオブジェクトexpを等価なPreDecExpressionオブジェクトあるいはPreIncExpressionオブジェクトに作り替える．

機能説明 与えられたExpressionオブジェクトexpを意味的に等価なPreDecExpressionオブジェクトあるいはPreIncExpressionオブジェクトに作り替える機能を，OpenJIT.frontend.treeパッケージの定めるASTに適合するように提供する．

入力データ exp (Pre形式に変更したい式を表すExpressionオブジェクト)

出力データ Expressionオブジェクト

処理方式 expがAssignExpressionクラスのインスタンスでなければ，pre形式に作り替える方法がないので，nullを返す．次に，expの右辺が左辺のExpressionオブジェクトに定数1を加える式を表している場合はPreIncExpressionオブジェクトに作り替えてそれを返す．また，expの右辺が左辺のExpressionオブジェクトから定数1を減ずる式を表している場合は，PreDecExpressionオブジェクトに作り替えてそれを返す．それ以外の場合には，nullを返す．

エラー処理 なし．

- public Expression toPostExpression(Expression exp)

機能概要 Expression オブジェクト exp を等価な PostDecExpression オブジェクトあるいは PostIncExpression オブジェクトに作り替える．

機能説明 与えられた Expression オブジェクト exp を意味的に等価な PostDecExpression オブジェクトあるいは PostIncExpression オブジェクトに作り替える機能を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供する．

入力データ exp (Post 形式に変更したい式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 exp が AssignExpression クラスのインスタンスでなければ，post 形式に作り替える方法がないので，null を返す．次に，exp の右辺が左辺の Expression オブジェクトに定数 1 を加える式を表している場合は PostIncExpression オブジェクトに作り替えてそれを返す．また，exp の右辺が左辺の Expression オブジェクトから定数 1 を減ずる式を表している場合は，PostDecExpression オブジェクトに作り替えてそれを返す．それ以外の場合には，null を返す．

エラー処理 なし．

- public Expression invertCondition(Expression exp)

機能概要 Expression オブジェクト exp を逆の論理を持つ式を表す Expression オブジェクトに作り替える．

機能説明 与えられた Expression オブジェクト exp を論理式を表すものと見なし，論理的に逆の意味を表す Expression オブジェクトに作り替えるための機能を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供する．

入力データ exp (逆の論理に変更したい式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 与えられた exp の種類に応じて，その逆の論理を持つ式を表す Expression オブジェクトを生成し，それを返す．

エラー処理 exp が論理式以外の式を表す Expression オブジェクトである場合，例外 DiscompilerError を放出する．

- public Identifier identifier(String name)

  機能概要 文字列 name と等しい名前を持つ Identifier オブジェクトを返す．

  機能説明 与えられた String オブジェクト name と等しい名前を持つ Identifier オブジェクトを OpenJIT.frontend.tree パッケージの定める AST に適合するように作り出すためのファクトリメソッド．

  入力データ name (求める Identifier オブジェクトの持つべき名前)

  出力データ Identifier オブジェクト

  処理方式 OpenJIT.frontend.java.Identifier の loopup メソッドを呼び出す．

  エラー処理 なし．

- public Expression arrayType(int atype)

  機能概要 VM の命令 newarray で使用されている配列の型を表す整数値に相当する型を表す Expression オブジェクトを返す．

  機能説明 VM の命令 newarray で使用されている配列の型を表す整数値 atype に対応する型を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供する．

  入力データ atype (VM 命令 newarray のオペランド)

  出力データ Expression オブジェクト

  処理方式 atype が正当な値である場合，atype をインデックスとする配列 simpleTypes の要素を返す．

  エラー処理 atype が不正な値である場合，例外 DiscompilerError を放出する．

- public Expression arrayType(String signature)

  機能概要 VM の命令 anewarray および multianewarray で使用される配列の型を表す文字列に相当する型を表す Expression オブジェクトを返す．

  機能説明 VM の命令 anewarray および multianewarray で使用される配列の型を表す文字列を解釈し，その型を表現するオブジェクトを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供する．

入力データ signature (配列の型を表す文字列)

出力データ Expression オブジェクト

処理方式 signature を引数として，classSignatures の lookup メソッドを呼び出し，その結果を返す．

エラー処理 なし．

- public int arrayDepth(String signature)

  機能概要 配列の型を表す文字列 signature の表す配列の深さを返す．

  機能説明 配列の型を表す文字列 signature を解釈し，その配列の深さを整数値として返す．

  入力データ signature (配列の型を表す文字列)

  出力データ int 値

  処理方式 signature を引数として，ClassSignature の lookup メソッドを呼び出し，その結果として得られる ClassSignature オブジェクトの dimension メソッドを呼び出し，結果を返す．

  エラー処理 なし．

- public Expression classType(String className)

  機能概要 文字列で表されたクラス名に相当する型を表す Expression オブジェクトを返す．

  機能説明 与えられた文字列 className を完全な (qualified な) クラス名と見なし，そのクラスの型を表す AST オブジェクトを OpenJIT.frontend.tree パッケージの定める AST に適合するように提供する．

  入力データ className (完全なクラス名)

  出力データ Expression オブジェクト

  処理方式 className を引数として classExpressions の lookup メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public Expression booleanExpression(boolean value)

  機能概要 boolean の値 value を表す BooleanExpression を返す．

  機能説明 boolean の値 value を表す BooleanExpression を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ value (boolean 値)

  出力データ Expression オブジェクト

  処理方式 value の値に応じて，booleanTrue あるいは booleanFalse のどちらかを返す．

  エラー処理 なし．

- public Expression byteExpression(byte value)

  機能概要 byte の値 value を表す ByteExpression を返す．

  機能説明 byte の値 value を表す ByteExpression を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ value (byte 値)

  出力データ Expression オブジェクト

  処理方式 value を引数として byteConstants の lookup メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public Expression charExpression(char value)

  機能概要 char の値 value を表す CharExpression を返す．

  機能説明 char の値 value を表す CharExpression を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ value (char 値)

  出力データ Expression オブジェクト

  処理方式 value を引数として charConstants の lookup メソッドを呼び出し，その結果を返す．

エラー処理 なし．

- public Expression shortExpression(short value)

  機能概要 short の値 value を表す ShortExpression を返す．

  機能説明 short の値 value を表す ShortExpression を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ value (short 値)

  出力データ Expression オブジェクト

  処理方式 value を引数として shortConstants の lookup メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public Expression doubleExpression(double value)

  機能概要 double の値 value を表す DoubleExpression を返す．

  機能説明 double の値 value を表す DoubleExpression を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ value (double 値)

  出力データ Expression オブジェクト

  処理方式 value を引数として doubleConstants の lookup メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public Expression floatExpression(float value)

  機能概要 float の値 value を表す FloatExpression を返す．

  機能説明 float の値 value を表す FloatExpression を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ value (float 値)

  出力データ Expression オブジェクト

処理方式 value を引数として floatConstants の lookup メソッドを呼び出し，その結果を返す．

エラー処理 なし．

- public Expression intExpression(int value)

  機能概要 int の値 value を表す IntExpression を返す．

  機能説明 int の値 value を表す IntExpression を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ value (int 値)

  出力データ Expression オブジェクト

  処理方式 value を引数として intConstants の lookup メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public Expression longExpression(long value)

  機能概要 long の値 value を表す LongExpression を返す．

  機能説明 long の値 value を表す LongExpression を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ value (long 値)

  出力データ Expression オブジェクト

  処理方式 value を引数として longConstants の lookup メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public Expression stringExpression(String value)

  機能概要 String の値 value を表す StringExpression を返す．

  機能説明 String の値 value を表す StringExpression を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ value (String 値)

出力データ Expression オブジェクト

処理方式 value を引数として stringConstants の lookup メソッドを呼び出し，その結果を返す．

エラー処理 なし．

- public Expression nullExpression()

  機能概要 null を表す式 NullExpression を返す．

  機能説明 null を表す式 NullExpression を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ なし．

  出力データ Expression オブジェクト

  処理方式 nullExpression を返す．

  エラー処理 なし．

- public Expression arrayAccessExpression(Expression right, Expression index)

  機能概要 AST の[] 式ノードを表す Expression オブジェクトを返す．

  機能説明 配列を表す式 right の index 番目の要素を参照することを意味する式を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ right (配列を表す Expression オブジェクト), index (index を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 ArrayAccessExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression bitNotExpression(Expression right)

  機能概要 AST の~ 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた式 right のビット毎の否定を表す式を OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ right (演算子を適用する式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 BitNotExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression convertExpression(Type type, Expression right)

  機能概要 primitive 型の型変換を表す式を返す．

  機能説明 与えられた式 right が primitive 型の式であると見なし，それを与えられた型に型変換することを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ type (型変換する目的の型を表す Type オブジェクト), right (型変換される式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 ConvertExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression exprExpression(Expression right)

  機能概要 AST の () 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right に対して () を適用した式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ right (式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 ExprExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression fieldExpression(Expression right, Identifier id)

機能概要 AST の . 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた式 right の持つ id というフィールドを参照することを意味する式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ right (フィールドを持つ式を表す Expression オブジェクト)，id (フィールド名を表す Identifier オブジェクト)

出力データ Expression オブジェクト

処理方式 FieldExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression lengthExpression(Expression right)

機能概要 AST の `.length` 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた配列式 right の長さを参照する式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ right (配列式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 LengthExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression negativeExpression(Expression right)

機能概要 AST の単項演算子- ノードを表す Expression オブジェクトを返す．

機能説明 与えられた式 right に単項演算子- を適用することを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ right (単項演算子- を適用する式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 NegativeExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression notExpression(Expression right)

  機能概要 AST の単項演算子! ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right に単項演算子! を適用することを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ right (単項演算子! を適用する式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 NotExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression positiveExpression(Expression right)

  機能概要 AST の単項演算子+ ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right に単項演算子+ を適用することを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ right (単項演算子+ を適用する式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 PositiveExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression postDecExpression(Expression right)

  機能概要 AST の後置演算子-- のノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right に後置演算子-- を適用することを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ right (後置演算子-- を適用する式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

処理方式 PostDecExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression postIncExpression(Expression right)

  機能概要 AST の後置演算子++ のノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right に後置演算子++ を適用することを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ right (後置演算子++ を適用する式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 PostIncExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression preDecExpression(Expression right)

  機能概要 AST の前置演算子-- のノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right に前置演算子-- を適用することを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ right (前置演算子-- を適用する式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 PreDecExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression preIncExpression(Expression right)

  機能概要 AST の前置演算子++ のノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right に前置演算子++ を適用することを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ right (前置演算子++ を適用する式を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 PreIncExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression castExpression(Expression left, Expression right)

  機能概要 AST のキャストノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right を left で表される型にキャストする式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (キャスト先の型を表す Expression オブジェクト), right (キャストされる式を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 CastExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression commaExpression(Expression left, Expression right)

  機能概要 AST の,(カンマ) 演算子のノードを表す Expression オブジェクトを返す．

  機能説明 カンマで結ばれた複数の式を順に評価し，最後の評価値を式全体の値とする CommaExpression を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (先に評価される式を表す Expression オブジェクト), right (後に評価され，式全体の値を与える Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 CommaExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression instanceOfExpression(Expression left, Expression right)

機能概要 AST のinstanceof 演算子ノードを表す Expression オブジェクトを返す．

機能説明 与えられた式 left が型を表す式 right のインスタンスであるかどうかを判定する演算子instanceof を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (判定するオブジェクトを表す Expression オブジェクト), right (型を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 InstanceOfExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression addExpression(Expression left, Expression right)

機能概要 AST の+ 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子+ を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 AddExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression divideExpression(Expression left, Expression right)

機能概要 AST の/ 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子/ を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 DivideExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression multiplyExpression(Expression left, Expression right)

  機能概要 AST の* 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子* を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 MultiplyExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression subtractExpression(Expression left, Expression right)

  機能概要 AST の- 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子- を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 SubtractExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression remainderExpression(Expression left, Expression right)

  機能概要 AST の% 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子% を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 RemainderExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression andExpression(Expression left, Expression right)

機能概要 AST の&& 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子& を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 AndExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression orExpression(Expression left, Expression right)

機能概要 AST の|| 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子| を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 OrExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression bitAndExpression(Expression left, Expression right)

  機能概要 AST の& 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子& を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 BitAndExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression bitOrExpression(Expression left, Expression right)

  機能概要 AST の| 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子| を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 BitOrExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression bitXorExpression(Expression left, Expression right)

  機能概要 AST の^ 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子^ を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 BitXorExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression equalExpression(Expression left, Expression right)

機能概要 AST の== 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子== を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 EqualExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression notEqualExpression(Expression left, Expression right)

機能概要 AST の!= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子!= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 NotEqualExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression shiftLeftExpression(Expression left, Expression right)

  機能概要 AST の<<式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子<<を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 ShiftLeftExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression shiftRightExpression(Expression left, Expression right)

  機能概要 AST の>> 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子>> を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 ShiftRightExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression unsignedShiftRightExpression(Expression, Expression)

  機能概要 AST の>>> 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子>>> を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 UnsignedShiftRightExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression greaterExpression(Expression left, Expression right)

  機能概要 AST の> 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子> を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 GreaterExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression greaterOrEqualExpression(Expression left, Expression right)

  機能概要 AST の>= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子>= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

処理方式 GreaterOrEqualExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression lessExpression(Expression left, Expression right)

  機能概要 AST の<式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子<を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 LessExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression lessOrEqualExpression(Expression left, Expression right)

  機能概要 AST の<= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子<= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 LessOrEqualExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression assignExpression(Expression left, Expression right)

  機能概要 AST の= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 AssignExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression assignAddExpression(Expression left, Expression right)

機能概要 AST の+= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子+= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 AssignAddExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression assignBitAndExpression(Expression left, Expression right)

機能概要 AST の&= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子&= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 AssignBitAndExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression assignBitOrExpression(Expression left, Expression right)

  機能概要 AST の|= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子|= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 AssignBitOrExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression assignBitXorExpression(Expression left, Expression right)

  機能概要 AST の^= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子^= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 AssignBitXorExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression assignDivideExpression(Expression left, Expression right)

  機能概要 AST の/= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子/= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 AssignDivideExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression assignMultiplyExpression(Expression left, Expression right)

  機能概要 AST の*= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子*= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 AssignMultiplyExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression assignRemainderExpression(Expression left, Expression right)

  機能概要 AST の%= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子%= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

処理方式 AssignRemainderExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression assignShiftLeftExpression(Expression left, Expression right)

  機能概要 AST の<<= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子<<= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 AssignShiftLeftExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression assignShiftRightExpression(Expression left, Expression right)

  機能概要 AST の>>= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子>>= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 AssignShiftRightExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression assignSubtractExpression(Expression left, Expression right)

  機能概要 AST の-= 式ノードを表す Expression オブジェクトを返す．

機能説明 与えられた二つの式 left および right に対し，二項演算子-= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

出力データ Expression オブジェクト

処理方式 AssignSubtractExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression assignUnsignedShiftRightExpression(Expression left, Expression right)

  機能概要 AST の>>>= 式ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた二つの式 left および right に対し，二項演算子>>>= を適用した結果を表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ left (左辺を表す Expression オブジェクト), right (右辺を表す Expression オブジェクト)

  出力データ Expression オブジェクト

  処理方式 AssignUnsignedShiftRightExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression conditionalExpression(Expression cond, Expression left, Expression right)

  機能概要 条件演算子を表す AST ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 cond の値により，二つの式 left および right の適当な方を選択するという意味を表す ConditionalExpression の式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ cond (条件を表すExpressionオブジェクト), left (条件が真の場合に選ばれる式を表すExpressionオブジェクト), right (条件が偽の場合に選ばれる式を表すExpressionオブジェクト)

出力データ Expressionオブジェクト

処理方式 ConditionalExpressionオブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression arrayExpression(Expression args[])

  機能概要 初期設定子付きの配列定義を表すExpressionオブジェクトを返す．

  機能説明 初期値を与えられた配列定義を表現するASTノードを表す式を，OpenJIT.frontend.treeパッケージの定めるASTに適合するように提供するファクトリメソッド．

  入力データ args (配列の初期値を表すExpressionオブジェクトの配列)

  出力データ Expression

  処理方式 ArrayExpressionオブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression methodExpression(Expression right, Identifier id Expression args[])

  機能概要 式rightで表されるオブジェクトの持つidというメソッドを引数をargsとして呼び出す式を表すExpressionオブジェクトを返す．

  機能説明 与えられた式rightの表すオブジェクトの持つidという名前のメソッドを引数をargsとして呼び出す式を表現するASTノードを表す式を，OpenJIT.frontend.treeパッケージの定めるASTに適合するように提供するファクトリメソッド．

  入力データ right (メソッドを持つオブジェクトを表すExpressionオブジェクト), id (メソッド名を表すIdentifierオブジェクト), args (引数を表すExpressionオブジェクトの配列)

  出力データ Expression

処理方式 MethodExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression newArrayExpression(Expression right, Expression args[])

  機能概要 多次元配列を割り当てる AST の new 演算子ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right で表される型の配列を，args 配列で与えられる要素数を持つように割り当てる AST ノードを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ right (配列の型を表す Expression オブジェクト), args (配列の要素数を表す Expression オブジェクトの配列)

  出力データ Expression オブジェクト

  処理方式 NewArrayExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression newInstanceExpression(Expression right, Expression args[])

  機能概要 式 right で表されるクラスのオブジェクトを作る AST の new 演算子ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた式 right で表されるクラスのオブジェクトを作り，引数 args と共にそのコンストラクタを呼ぶことを意味する AST の new 演算子ノードを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ right (クラスを表す Expression オブジェクト), args (コンストラクタの引数を表す Expression オブジェクトの配列)

  出力データ Expression

  処理方式 NewInstanceExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Expression identifierExpression(Identifier id)

機能概要 識別子 id を参照する式を表す Expression オブジェクトを返す．

機能説明 与えられた識別子 id を参照する式を表現する AST ノードを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ id (識別子を表す Identifier オブジェクト)

出力データ Expression オブジェクト

処理方式 IdentifierExpression オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Expression superExpression()

機能概要 予約語`super` に対応する AST ノードを表す Expression オブジェクトを返す．

機能説明 予約語`super` に対応する AST ノードを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ なし．

出力データ Expression オブジェクト

処理方式 constantSuper を返す．

エラー処理 なし．

- public Expression thisExpression()

機能概要 予約語`this` に対応する AST ノードを表す Expression オブジェクトを返す．

機能説明 予約語`this` に対応する AST ノードを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ なし．

出力データ Expression オブジェクト

処理方式 constantThis を返す．

エラー処理 なし．

- public Expression typeExpression(Type type)

  機能概要 型 type を表す AST ノードを表す Expression オブジェクトを返す．

  機能説明 与えられた型 type を表現する AST ノードを表す式を，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ type (型を表す Type オブジェクト)

  出力データ Expression オブジェクト

  処理方式 TypeExpression オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Statement breakStatement(Identifier lbl)

  機能概要 `break lbl` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられたラベル lbl に対する`break` 文を意味する AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ lbl (`break` 文の対象となるラベルを表す Identifier オブジェクト)

  出力データ Statement オブジェクト

  処理方式 BreakStatement オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Statement caseStatement(Expression expr)

  機能概要 式 expr の値を持つ`case` ラベルを意味する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた式 expr の値を持つ`case` ラベルを意味する AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ expr (`case` ラベルの値を表す Expression オブジェクト)

  出力データ Statement オブジェクト

処理方式 CaseStatement オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Statement catchStatement(Expression texpr, Identifier id, Statement body)

  機能概要 一つのcatch 節を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた式 texpr で選択される型の例外を id という名前として捕捉するためのcatch 節が body という内容を持つものであることを意味する AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ texpr (捕捉する例外の型を表す Expression オブジェクト), id (catch 節内における例外の識別子を表す Identifier オブジェクト), body (catch 節の本体を表す Statement オブジェクト)

  出力データ Statement オブジェクト

  処理方式 CatchStatement オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Statement compoundStatement(Statement args[])

  機能概要 複文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた Statement オブジェクトの配列が一つの複文となることを表す AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ args (複文に含まれるそれぞれの文を表す Statement の配列)

  出力データ Statement オブジェクト

  処理方式 CompoundStatement オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Statement continueStatement(Identifier lbl)

機能概要 `continue lbl` 文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 与えられたラベル lbl に対する `continue` 文を意味する AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ lbl (`continue` 文の対象となるラベルを表す Identifier オブジェクト)

出力データ Statement オブジェクト

処理方式 ContinueStatement オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Statement declarationStatement(Expression type, Statement args[])

機能概要 初期化付きの変数定義を表す Statement オブジェクトを返す．

機能説明 与えられた式 type で表される型の変数を，配列 args の各文によって初期化する変数定義を表す AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ type (宣言される変数の型を表す Expression オブジェクト), args (変数の初期化を行う文を表す Statement オブジェクトの配列)

出力データ Statement オブジェクト

処理方式 DeclarationStatement オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Statement doStatement(Statement body, Expression cond)

機能概要 do-while 文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 与えられた式 cond を条件として body を繰り返す do-while 文を意味する AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ body (do-while 文の本体を表す Statement オブジェクト), cond (do-while 文の条件を表す Expression オブジェクト)

出力データ Statement オブジェクト

処理方式 DoStatement オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Statement expressionStatement(Expression expr)

  機能概要 式文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた式 expr を評価する文を表す AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ expr (式文で評価される式を表す Expression オブジェクト)

  出力データ Statement オブジェクト

  処理方式 ExpressionStatement オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Statement finallyStatement(Statement body, Statement finalbody)

  機能概要 `finally` 節を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた body が `try` 文を表すものと見なし，その `try` 文に finalbody を持つ `finally` 節が付いていることを表現する AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ body (`try` 文を表す Statement オブジェクト), finalbody (`finally` 節の本体を表す Statement オブジェクト)

  出力データ Statement オブジェクト

  処理方式 FinallyStatement オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Statement forStatement(Statement init, Expression cond, Expression inc, Statement body)

機能概要 `for` 文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 初期設定文が init で条件が cond，繰り返しの際に実行される文が inc，そして，本体が body であるような`for` 文を表す AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ init (初期設定文を表す Statement オブジェクト), cond (条件を表す Expression オブジェクト), inc (繰り返しの際に実行される文を表す Statement オブジェクト), body (`for` 文の本体を表す Statement オブジェクト)

出力データ Statement オブジェクト

処理方式 ForStatement オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Statement ifStatement(Expression cond, Statement ifTrue, Statement ifFalse)

  機能概要 `if` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 条件を cond とし，条件が真の場合に実行される文が ifTrue で，条件が偽の場合に実行される文が ifFalse であるような`if` 文を表す AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ cond (条件を表す Expression オブジェクト), ifTrue (条件が真の場合に実行される文を表す Statement オブジェクト), ifFalse (条件が偽の場合に実行される文を表す Statement オブジェクト)

  出力データ Statement オブジェクト

  処理方式 IfStatement オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Statement returnStatement(Expression expr)

  機能概要 式 expr の値を返す`return` 文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 与えられた式 expr の値を返す`return` 文を表す AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ expr (返り値を表す Expression オブジェクト)

出力データ Statement オブジェクト

処理方式 ReturnStatement オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Statement switchStatement(Expression expr, Statement args[])

機能概要 式 expr を評価して分岐する`switch` 文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 与えられた式 expr を評価し，Statement の配列 args 内の適当な`case` 文に分岐する`switch` 文を表す AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ expr (分岐に使用する式を表す Expression オブジェクト)，args (`switch` 文の本体を表す Statement オブジェクトの配列)

出力データ Statement オブジェクト

処理方式 SwitchStatement オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Statement synchronizedStatement(Expression expr, Statement body)

機能概要 `synchronized` 文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 与えられた式 expr に関する`synchronized` 文の本体が body であることを表す AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

入力データ expr (同期に使用されるオブジェクトを表す Expression オブジェクト), body (`synchronized` 文の本体を表す Statement オブジェクト)

出力データ Statement オブジェクト

処理方式 SynchronizedStatement オブジェクトを生成し，それを返す．

エラー処理 なし．

- public Statement throwStatement(Expression expr)

  機能概要 式 expr で表されるオブジェクトを放出する`throw` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 与えられた式 expr で表されるオブジェクト (`throwable`) を例外として放出する`throw` 文を表す AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ expr (放出されるオブジェクトを表す Expression オブジェクト)

  出力データ Statement オブジェクト

  処理方式 ThrowStatement オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Statement tryStatement(Statement body, Statement args[])

  機能概要 `try` 文を表現する AST ノードを表す Statement オブジェクトを返す．

  機能説明 配列として与えられた args のそれぞれの要素が`catch` 節であり，本体が body となっている`try` 文を表す AST ノードを，OpenJIT.frontend.tree パッケージの定める AST に適合するように提供するファクトリメソッド．

  入力データ body (`try` 文の本体を表す Statement オブジェクト), args (それぞれが一つの`catch` 節となる Statement オブジェクト配列)

  出力データ Statement オブジェクト

  処理方式 TryStatement オブジェクトを生成し，それを返す．

  エラー処理 なし．

- public Statement varDeclarationStatement(Expression expr)

  機能概要 初期設定付きの変数宣言に現れる，それぞれの初期設定文を表現する AST ノードを表す Statement オブジェクトを返す．

機能説明 初期設定付きの変数宣言に現れる，それぞれの初期設定文を表すASTノードを，OpenJIT.frontend.treeパッケージの定めるASTに適合するように提供するファクトリメソッド．

入力データ expr (初期設定を行う式を表すExpressionオブジェクト)

出力データ Statementオブジェクト

処理方式 VarDeclarationStatementオブジェクトを生成し，それを返す．

エラー処理 なし．

- public Statement whileStatement(Expression cond, Statement body)

機能概要 `while`文を表現するASTノードを表すStatementオブジェクトを返す．

機能説明 条件がcond，本体がbodyである`while`文を表すASTノードを，OpenJIT.frontend.treeパッケージの定めるASTに適合するように提供するファクトリメソッド．

入力データ cond (条件を表すExpressionオブジェクト), body (`while`文の本体を表すStatementオブジェクト)

出力データ Statementオブジェクト

処理方式 WhileStatementオブジェクトを生成し，それを返す．

エラー処理 なし．

4.3.21 OpenJIT.frontend.discompiler.DominatorTree クラス

public class DominatorTree

(1) 概要

このクラスは，ディスコンパイラ機能がコントロールフロー解析の際に使用するドミネータツリーをコントロールフローグラフから構築し，また，構築したツリーを格納する機能を提供するものである．

(2) フィールド (変数)

- boolean traceEdge デバッグ用の表示を制御するフラグ．
- DTNode root ドミネータツリーのルートとなるコントロールフローグラフノードを保持する．
- Hashtable immediateDominators コントロールフローグラフノードと，そのイミディエートドミネータに対応するコントロールフローグラフノードの組を保持するハッシュテーブル．
- Hashtable treeNodes コントロールフローグラフノードと，それに対応するドミネータツリーノードの組を保持するハッシュテーブル．

(3) コンストラクタ

- public DominatorTree(DTNode root)

  機能概要 ドミネータツリーの構築に必要な情報を生成する．

  機能説明 ドミネータツリーの構築に必要な情報である，コントロールフローグラフの各ノードのイミディエートドミネータの情報を生成し，ドミネータツリー構築の下準備を行う．

  入力データ root (ドミネータツリーのルートとなるべきコントロールフローグラフノード)

  出力データ なし．

処理方式 コントロールフローグラフの各ノードからのびるエッジについて行き先を調査し，それぞれのイミディエートドミネータとなるコントロールフローグラフノードを immediateDominators に格納する．更に，root の immediateDominator が root でない場合，makeTree メソッドを呼び出し，ツリー構造を完成させる．

エラー処理 なし．

(4) メソッド

- public DTNode immediateDominatorOf(DTNode node)

  機能概要 イミディエートドミネータを返す．

  機能説明 指定したノードのイミディエートドミネータとなるコントロールフローグラフノードを返す．

  入力データ node (目的のコントロールフローグラフノード)

  出力データ DTNode オブジェクト

  処理方式 node をキーとしてハッシュテーブル immediateDominators を引き，その結果を返す．

  エラー処理 なし．

- private void makeTree()

  機能概要 ドミネータツリーを構築する．

  機能説明 イミディエートドミネータの情報からドミネータツリーを構築する．

  入力データ なし．

  出力データ なし．

  処理方式 目的のコントロールフローグラフの全てのノードについて，それぞれのイミディエートドミネータに対応するドミネータツリーノードを探し，そのノードの子ノードとして登録することにより，ドミネータツリーを構築する．

  エラー処理 なし．

- public void print(IndentedPrintStream out)

  機能概要 デバッグ用にドミネータツリーを表示する．

  機能説明 デバッグ用にドミネータツリーを表示する．

  入力データ out (出力先を指定する IndentedPrintStream オブジェクト)

  出力データ なし．

  処理方式 treeNodes に含まれる全ての DTNode オブジェクトについて，print メソッドを呼び出す．

  エラー処理 なし．

- public void printImmediateDominators(IndentedPrintStream out)

  機能概要 デバッグ用にイミディエートドミネータの一覧を表示する．

  機能説明 デバッグ用にイミディエートドミネータの一覧を表示する．

  入力データ out (出力先を指定する IndentedPrintStream オブジェクト)

  出力データ なし．

  処理方式 immediateDominators に含まれるノードとそのイミディエートドミネータの組を表示する．

  エラー処理 なし．

- public int distanceFromRoot(CFGNode node)

  機能概要 あるノードのドミネータツリーにおけるルートからの距離を返す．

  機能説明 あるノードのドミネータツリーにおけるルートからの距離を返す．

  入力データ node (目的のコントロールフローグラフノード)

  出力データ int 値

  処理方式 ルートに辿り着くまでの親ノードの数を数え，それを返す．

  エラー処理 なし．

4.3.22 OpenJIT.frontend.discompiler.ExpressionAnalyzer クラス

public class ExpressionAnalyzer extends BasicBlockAnalyzer

(1) 概要

このクラスは，コントロールフローグラフを渡り歩くことにより，複数のベーシックブロックの組合わせが一つの式を表しているようなパターンを見つけ出し，Java 言語の式レベルの構造が，コントロールフローグラフに含まれる各ベーシックブロック内で閉じた状態のコントロールフローグラフに作り替える機能を提供するものである．

(2) フィールド (変数)

- boolean traceCond デバッグ用の表示を制御するフラグ．

(3) コンストラクタ

- public ExpressionAnalyzer(MethodInformation method, BytecodeParser bytecodeInfo, ASTFactory astFactory)

  機能概要 式レベルの構造を解析し終えたコントロールフローグラフを生成する．

  機能説明 式レベルの構造を解析し終えたコントロールフローグラフを生成する．

  入力データ method (コンパイラ基盤モジュールからメソッドの情報を得るためのインターフェース), bytecodeInfo (解析を終えたバイトコード情報), astFactory (ディスコンパイルの結果として得られる AST の構造を決めるファクトリオブジェクト)

  出力データ なし．

  処理方式 method と bytecodeInfo および astFactory を引数として super メソッドを呼び出し，その後に，createCFG メソッドを呼び出して，原始的なコ

ントロールフローグラフを構築する．次に，各コントロールフローグラフについて式レベルの解析を行う．

エラー処理 なし．

(4) メソッド

- void recoverExpressions(DTNode head)

  機能概要 指定したドミネータツリーの式レベルの解析を行う．

  機能説明 指定したドミネータツリーの式レベルの解析を行う．

  入力データ head (ドミネータツリーのルートノード)

  出力データ なし．

  処理方式 ConditionalExpression の解析とショートカットされたブランチの条件式の復元をどちらも進展が見られなくなるまで繰り返し行う．

  エラー処理 なし．

- void deoptimizeConditionalExpression(DTNode node)

  機能概要 最適化などの理由から乱れたコントロールフローグラフを回復する．

  機能説明 最適化などの理由から乱れたコントロールフローグラフを回復する．

  入力データ node (目的のコントロールフローグラフを表すドミネータツリーのルートノード)

  出力データ なし．

  処理方式 ツリーを渡り歩いて復元できるパターンを探し，復元する．

  エラー処理 なし．

- boolean recoverConditionalExpressions(DTNode head)

  機能概要 コントロールフローグラフに含まれる ConditionalExpression 構造を回復する．

  機能説明 コントロールフローグラフに含まれる ConditionalExpression 構造をできるだけ回復する．

入力データ head (コントロールフローグラフの先頭ノードを表すドミネータツリーのルートノード)

出力データ boolean 値

処理方式 ドミネータツリーを渡り歩き，ConditionalExpression を実現するためのパターンを見つけるたびにグラフとツリーの書き換えを行い，ConditionalExpression を回復する．その回の渡り歩きで，回復された ConditionalExpression が存在する場合は true を返し，それ以外では false を返す．

エラー処理 回復不可能な種類のノードを見つけた場合，例外 DiscompilerError を放出する．

- boolean recoverBranchConditions(DTNode head)

機能概要 ショートカットされたブランチの条件式を回復する．

機能説明 コントロールフローグラフに含まれる，ショートカットされたブランチの条件式をできるだけ回復する．

入力データ head (コントロールフローグラフの先頭ノード)

出力データ boolean 値

処理方式 コントロールフローグラフを渡り歩き，ショートカットされた条件ブランチを実現するためのパターンを見つけるたびにグラフの書き換えを行い，ブランチの条件を回復する．その回の渡り歩きで，回復されたブランチの条件式が存在する場合は true を返し，それ以外では false を返す．

エラー処理 なし．

- static Expression snatchBranchCondition(DTNode node)

機能概要 ブランチ命令の条件式を取り出す．

機能説明 指定したコントロールフローグラフノードの末尾の条件ブランチ命令を取り除き，その条件式だけを返す．

入力データ node (目的のコントロールフローグラフノード)

出力データ Expression オブジェクト

処理方式 node.contents の pop メソッドを呼び出すことにより，末尾のブランチ命令を取り出す．次に，その命令に含まれる条件式を取り出し，それを返す．

エラー処理 末尾が条件ブランチ命令でないコントロールフローグラフノードに対して呼び出した場合，例外 DiscompilerError を放出する．

- void append(DTNode predicessor, DTNode successor)

  機能概要 二つのコントロールフローグラフノードを連結する．

  機能説明 先行するベーシックブロックの内容の末尾に後続のベーシックブロックの内容を付け加えることにより，コントロールフローグラフノードを連結する．

  入力データ predicessor (先行するコントロールフローグラフノード), successor (後続のコントロールフローグラフノード)

  出力データ なし．

  処理方式 先行するベーシックブロックと後続のベーシックブロックの内容 (SymbolicStack オブジェクト) を連結する．

  エラー処理 なし．

- void supplyStack(SymbolicStack stack, SymbolicValue value)

  機能概要 シンボリックスタックに値を供給する．

  機能説明 シンボリックスタックに値を供給する．

  入力データ stack (シンボリックスタック), value (供給する値)

  出力データ なし．

  処理方式 スタックの先頭の要素がオペランド不足により未解析の状態になっている場合，その要素に値を供給することによって解析の進展を試みる．

  エラー処理 なし．

- public void printStack(SymbolicStack stack)

  機能概要 デバッグ用に，コントロールフローグラフノードの中身を表示する．

機能説明 デバッグ用に，コントロールフローグラフノードの中身である SymbolicStack の中身を表示する．

入力データ stack (目的の SymbolicStack オブジェクト)

出力データ なし．

処理方式 stack 内の各要素について表示を行う．

エラー処理 なし．

4.3.23 OpenJIT.frontend.discompiler.FlowEdge クラス

public class FlowEdge

(1) 概要

このクラスは，ドミネータツリーの構築時に使用されるコントロールフローグラフのエッジの内部表現形式を定義するものである．

(2) フィールド (変数)

- DTNode from 先行ノードを保持する．
- DTNode to 後続ノードを保持する．

(3) コンストラクタ

- public FlowEdge(DTNode from, DTNode to)

  機能概要 コントロールフローエッジを生成する．

  機能説明 コントロールフローエッジを生成する．

  入力データ from (先行するノード), to (後続のノード)

  出力データ なし．

  処理方式 this.from を from で，this.to を to で初期設定する．

  エラー処理 なし．

(4) メソッド

- public String toString()

  機能概要 エッジを識別する文字列を返す．

  機能説明 エッジを識別する文字列を返す．

  入力データ なし．

出力データ なし．

処理方式 先行するノードと後続のノードのそれぞれを識別する文字列を矢印で連結した文字列を返す．

エラー処理 なし．

4.3.24 OpenJIT.frontend.discompiler.LoopHead クラス

public class LoopHead extends CFAFlag

(1) 概要

このクラスは，コントロールフロー解析の結果として，何らかのループ構造の先頭ノードであると判明したブロックに対して付けられ，ループ構造の先頭であることを示すためのものである．

(2) フィールド (変数)

- int count そのループ構造に関するバックエッジの数を保持する．
- int distance 最後に見つけたバックエッジを持つノードからループ先頭までの距離を保持する．
- DTNode last 最後に見つけたバックエッジを持つノードを保持する．

(3) コンストラクタ

- public LoopHead(DTNode firstFinder, int distance)

  機能概要 ループ構造を示すオブジェクトを生成する．

  機能説明 ループ構造を示すオブジェクトを生成する．

  入力データ firstFinder (最初に見つけたバックエッジを持つノード), distance (最初に見つけたバックエッジを持つノードからループ構造の先頭ノードまでの距離)

  出力データ なし．

  処理方式 定数 LOOP_HEAD_NODE を引数として super メソッドを呼び出し，this.last を firstFinder で，this.distance を distance で，count を 1 で，それぞれ初期設定する．

  エラー処理 なし．

(4) メソッド

- public void update(DTNode newComer, int distance)

  機能概要 ループ末尾候補のノードを更新する．

  機能説明 ループ末尾候補のノードを更新する．

  入力データ newComer (新たに見つかったバックエッジを持つノード), distance (新たに見つかったバックエッジを持つノードからループ先頭ノードまでの距離)

  出力データ なし．

  処理方式 count を 1 増加し，それまでのループ末尾候補のノードよりも新たに見つかったノードの方がループ構造の先頭からの距離が遠かった場合，新たに見つかったノードを新たなループ末尾候補として last に保持する．

  エラー処理 なし．

- public boolean isBreakTarget()

  機能概要 構造が break 文の対象となる構造であるかどうかを判定する．

  機能説明 構造が break 文の対象となる構造であるかどうかを判定する．

  入力データ なし．

  出力データ boolean 値

  処理方式 true を返す．

  エラー処理 なし．

4.3.25 OpenJIT.frontend.discompiler.Metaclass 抽象クラス

このクラスは，クラスファイルに付加されるアノテーション情報に対応するメタクラスのインターフェースを定めるものである．

public abstract class Metaclass

(1) フィールド (変数)

なし．

(2) コンストラクタ

なし．

(3) メソッド

- public void mataInvoke()

  機能概要 メタクラス毎の処理を実行する．

  機能説明 メタクラス毎に定義された，処理を実行する．

  入力データ なし．

  出力データ なし．

  処理方式 このメソッドは抽象メソッドであり，実際に使用されるメソッドはサブクラスで定義する．

  エラー処理

4.3.26 OpenJIT.frontend.discompiler.MethodInformation インターフェース

public interface MethodInformation

(1) 概要

このインターフェースは，ディスコンパイラが必要とするメソッドの情報を受け取るために使用するインターフェースを定義するものである．

(2) フィールド (変数)

なし．

(3) コンストラクタ

なし．

(4) メソッド

- public int nlocals()

  機能概要 メソッドの使用するローカル変数の数を返す．

  機能説明 メソッドの使用するローカル変数の数を返す．

  入力データ なし．

  出力データ int 値

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public boolean isStatic()

  機能概要 メソッドがスタティックメソッドかどうかを返す．

機能説明 メソッドがスタティックメソッドかどうかを返す．

入力データ なし．

出力データ boolean 値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public String thisClassName()

機能概要 メソッドを持つクラスの名前を返す．

機能説明 メソッドを持つクラスの名前を返す．

入力データ なし．

出力データ String オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public String className(int index)

機能概要 コンスタントプールに保持されている特定のクラス名を返す．

機能説明 コンスタントプール内への index で選択される特定のクラス名を返す．

入力データ index (コンスタントプールへの index)

出力データ String オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public int fieldWidth(int index)

機能概要 フィールドの値もしくはメソッドの返し値のワード幅を返す．

機能説明 フィールドの値もしくはメソッドの返し値のワード幅を返す．

入力データ index (フィールドもしくはメソッドを選択する index)

出力データ int 値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public NameAndType nameAndType(int index)

  機能概要 フィールドもしくはメソッドの型情報を返す．

  機能説明 フィールドもしくはメソッドの型記述子を解析した結果としての NameAndType オブジェクトを返す．

  入力データ index (フィールドもしくはメソッドを選択する index)

  出力データ NameAndType オブジェクト

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public int kindOfConstant(int index)

  機能概要 コンスタントプール内に保持されている定数の種別を返す．

  機能説明 コンスタントプール内に保持されている定数の種別を返す．

  入力データ index (定数を保持するコンスタントプールへの index)

  出力データ int 値

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public int constantInt(int index)

  機能概要 コンスタントプールから int 定数を取り出す．

機能説明 コンスタントプールから int 定数を取り出す．

入力データ index (int 定数を保持するコンスタントプールへの index)

出力データ int 値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用される
メソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public float constantFloat(int index)

機能概要 コンスタントプールから float 定数を取り出す．

機能説明 コンスタントプールから float 定数を取り出す．

入力データ index (float 定数を保持するコンスタントプールへの index)

出力データ float 値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用される
メソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public String constantString(int index)

機能概要 コンスタントプールから String 定数を取り出す．

機能説明 コンスタントプールから String 定数を取り出す．

入力データ index (String 定数を保持するコンスタントプールへの index)

出力データ String オブジェクト

処理方式 このメソッドはインターフェースメソッドであり，実際に使用される
メソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public double constantDouble(int index)

機能概要 コンスタントプールから double 定数を取り出す．

機能説明 コンスタントプールから double 定数を取り出す．

入力データ index (double 定数を保持するコンスタントプールへの index)

出力データ double 値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用される
メソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public long constantLong(int index)

機能概要 コンスタントプールから long 定数を取り出す．

機能説明 コンスタントプールから long 定数を取り出す．

入力データ index (long 定数を保持するコンスタントプールへの index)

出力データ long 値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用される
メソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public ExceptionHandler[] exceptionHandler()

機能概要 例外ハンドラテーブルを返す．

機能説明 メソッドの持つ例外ハンドラテーブルを返す．

入力データ なし．

出力データ ExceptionHandler オブジェクトの配列．

処理方式 このメソッドはインターフェースメソッドであり，実際に使用される
メソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public int bytecodeLength()

機能概要 バイトコードの長さを返す．

機能説明 バイトコードの長さを返す．

入力データ なし．

出力データ int 値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public int byteAt(int pc)

  機能概要 バイトコードの特定のアドレスから byte 値を取り出す．

  機能説明 バイトコードの特定のアドレスから byte 値を取り出す．

  入力データ pc (バイトコードのアドレス)

  出力データ int 値

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public int unsignedByteAt(int pc)

  機能概要 バイトコードの特定のアドレスから符合無し byte 値を取り出す．

  機能説明 バイトコードの特定のアドレスから符合無し byte 値を取り出す．

  入力データ pc (バイトコードのアドレス)

  出力データ int 値

  処理方式 このメソッドはインターフェースメソッドであり，実際に使用されるメソッドはこのインターフェースを実装するクラスで定義する．

  エラー処理 なし．

- public int shortAt(int pc)

  機能概要 バイトコードの特定のアドレスから short 値を取り出す．

機能説明 バイトコードの特定のアドレスから short 値を取り出す．

入力データ pc (バイトコードのアドレス)

出力データ int 値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用される
メソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public int unsignedShortAt(int pc)

機能概要 バイトコードの特定のアドレスから符合無し short 値を取り出す．

機能説明 バイトコードの特定のアドレスから符合無し short 値を取り出す．

入力データ pc (バイトコードのアドレス)

出力データ int 値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用される
メソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public int intAt(int pc)

機能概要 バイトコードの特定のアドレスから int 値を取り出す．

機能説明 バイトコードの特定のアドレスから int 値を取り出す．

入力データ pc (バイトコードのアドレス)

出力データ int 値

処理方式 このメソッドはインターフェースメソッドであり，実際に使用される
メソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

- public byte[] annotation()

機能概要 アノテーション情報を取り出す．

機能説明 クラスファイルに保持されているアノテーション情報を取り出す．

入力データ なし．

出力データ byte 配列

処理方式 このメソッドはインターフェースメソッドであり，実際に使用される
メソッドはこのインターフェースを実装するクラスで定義する．

エラー処理 なし．

4.3.27 OpenJIT.frontend.discompiler.NameAndType クラス

public class NameAndType implements Constants

(1) 概要

このクラスは，クラス名とフィールドもしくはメソッドの名前，そして型記述子を解析し，型情報を使用しやすい形に変換する機能を提供するものである．

(2) フィールド (変数)

- byte[] typeVoid void 型を表す型記述子を保持する．
- byte[] typeBoolean boolean 型を表す型記述子を保持する．
- byte[] typeByte byte 型を表す型記述子を保持する．
- byte[] typeChar char 型を表す型記述子を保持する．
- byte[] typeShort short 型を表す型記述子を保持する．
- byte[] typeInt int 型を表す型記述子を保持する．
- byte[] typeLong long 型を表す型記述子を保持する．
- byte[] typeFloat float 型を表す型記述子を保持する．
- byte[] typeDouble double 型を表す型記述子を保持する．
- byte PERIOD_IN_BYTE パッケージの区切りを表す文字を保持する．
- byte SLASH_IN_BYTE クラスファイル中でパッケージの区切りを表す文字を保持する．
- String clazz フィールドもしくはメソッドのクラス名を保持する．
- String name フィールドもしくはメソッドの名前を保持する．
- byte[] signature 解析前の型記述子を保持する．

- boolean isField フィールドの型記述子であるかどうかを表す．
- byte[] type フィールドの型もしくはメソッドの返し値の型を表す記述子を保持する．
- String typeString フィールドの型もしくはメソッドの返し値の型を表す文字列を保持する．
- byte[][] args 引数毎の型を表す記述子を保持する配列．
- String[] argStrings 引数毎の型を表す文字列を保持する配列．
- int numberOfArgs 引数の数を保持する．
- String string 名前と型を Java 言語風に表す文字列を保持する．

(3) コンストラクタ

- public NameAndType(String clazz, String name, byte[] signature)

  機能概要 型記述子の解析結果を生成する．

  機能説明 型記述子の解析結果を生成する．

  入力データ clazz (フィールドもしくはメソッドを持つクラス名), name (フィールドもしくはメソッドの名前), signature (型記述子)

  出力データ なし．

  処理方式 this.name を name で，this.signature を signature で，this.clazz を clazz でそれぞれ初期設定し，次に signature を解析する．

  エラー処理 なし．

(4) メソッド

- public int numberOfArgValues()

  機能概要 引数全体のワード数を返す．

  機能説明 引数全体のワード数を返す．

入力データ なし．

出力データ int 値

処理方式 args.length を返す．

エラー処理 なし．

- public int numberOfArgs()

  機能概要 引数の数を返す．

  機能説明 引数の数を返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 numberOfArgs を返す．

  エラー処理 なし．

- public String name()

  機能概要 フィールドもしくはメソッドの名前を返す．

  機能説明 フィールドもしくはメソッドの名前を返すアクセッサメソッド．

  入力データ なし．

  出力データ String オブジェクト

  処理方式 name を返す．

  エラー処理 なし．

- public byte[] type()

  機能概要 フィールドの型もしくはメソッドの返し値の型を表す型記述子を返す．

  機能説明 フィールドの型もしくはメソッドの返し値の型を表す型記述子を返すアクセッサメソッド．

  入力データ なし．

出力データ byte 配列

処理方式 type を返す．

エラー処理 なし．

- public String typeString()

  機能概要 フィールドの型もしくはメソッドの返し値の型を表す文字列を返す．

  機能説明 フィールドの型もしくはメソッドの返し値の型を表す文字列を返す．

  入力データ なし．

  出力データ String オブジェクト

  処理方式 すでに文字列が作られている場合はそれを返し，まだであれば type を文字列に変換してそれを返す．

  エラー処理 なし．

- public byte[][] args()

  機能概要 引数毎の記述子を返す．

  機能説明 引数毎の記述子を返すアクセッサメソッド．

  入力データ なし．

  出力データ byte 配列の配列

  処理方式 args を返す．

  エラー処理 なし．

- public byte[] arg(int index)

  機能概要 特定の引数の記述子を返す．

  機能説明 特定の引数の記述子を返すアクセッサメソッド．

  入力データ index (引数を選択する index)

  出力データ byte 配列

  処理方式 args[index] を返す．

エラー処理 なし．

- public String argString(int index)

  機能概要 特定の引数の型を表す文字列を返す．

  機能説明 特定の引数の型を表す文字列を返す．

  入力データ index (引数を選択する index)

  出力データ String オブジェクト

  処理方式 すでに文字列に変換されていればそれを返し，まだであればargs[index]を文字列に変換して返す．

  エラー処理 なし．

- public String className()

  機能概要 フィールドもしくはメソッドを持つクラス名を返す．

  機能説明 フィールドもしくはメソッドを持つクラス名を返すアクセッサメソッド．

  入力データ なし．

  出力データ String オブジェクト

  処理方式 clazz を返す．

  エラー処理 なし．

- private String canonClassName(String name)

  機能概要 完全なクラス名を Java 言語風に表現する文字列を返す．

  機能説明 クラスファイルの内部表現で表された文字列を，Java 言語風の表現に変換した文字列を返す．

  入力データ name (クラスファイルの内部表現で表されたクラス名)

  出力データ String オブジェクト

  処理方式 クラスファイルの内部表現で用いられているパッケージの区切り文字‘/’を Java 言語風のパッケージの区切り文字‘.’に変換する．

エラー処理 なし．

- public String toString()

  機能概要 フィールドもしくはメソッドの型記述子を解析した結果を表す文字列を返す．

  機能説明 フィールドもしくはメソッドの型記述子を解析した結果を表す文字列を返す．

  入力データ なし．

  出力データ String オブジェクト

  処理方式 フィールドの型もしくはメソッドの返し値を表す文字列と，その名前，そして，メソッドの場合はそれぞれの引数の型を表す文字列を連結した文字列を返す．

  エラー処理 なし．

- public static String toString(byte[] type)

  機能概要 記述子の表す型を表す文字列を返す．

  機能説明 記述子の表す型を表す文字列を返す．

  入力データ type (記述子)

  出力データ String オブジェクト

  処理方式 与えられた記述子で表された型を表す文字列を返す．

  エラー処理 なし．

- private void parseSignature()

  機能概要 型記述子を解析する．

  機能説明 型記述子を解析し，個々の引数や返し値ごとに分割する．

  入力データ なし．

  出力データ なし．

処理方式 型記述子の先頭から走査し，個々の引数や返し値毎の区切りを見つけ，それぞれを type 配列に格納する．

エラー処理 なし．

- private static byte[] extractAType(byte[] src, int offset)

  機能概要 型記述子の特定の位置から，一つ分の記述子を取り出す．

  機能説明 型記述子の特定の位置から，一つ分の記述子を取り出す．

  入力データ src (型記述子), offset (目的の記述子の先頭位置)

  出力データ byte 配列

  処理方式 offset 位置から一文字ずつ取り出し，個々の記述子の区切りを探し，一つ分の記述子を返す．

  エラー処理 なし．

4.3.28 OpenJIT.frontend.discompiler.RuntimeMethodInformation クラス

public class RuntimeMethodInformation implements MethodInformation, Constants

(1) 概要

このクラスは，ディスコンパイラが必要とするメソッドの情報をコンパイラ基盤機能から受け取るために必要なインターフェースを，実装するものである．

(2) フィールド (変数)

- Compile method ディスコンパイルの対象となるメソッドの情報を与えるコンパイラ基盤モジュールを保持する．
- IntKeyHashtable classNames コンスタントプールへのインデックスとクラス名の組を保持する．
- IntKeyHashtable nameAndTypes コンスタントプールへのインデックスとフィールドまたはメソッドの型情報の組を保持する．
- String thisClassName 目的のメソッドを持つクラス名を保持する．

(3) コンストラクタ

- public RuntimeMethodInformation(Compile method)

  機能概要 ディスコンパイラの要求する情報を与えるインターフェースを生成する．

  機能説明 コンパイラ基盤モジュールからディスコンパイラの要求する情報を取り出すインターフェースを生成する．

  入力データ method (目的のメソッドに関するコンパイラ基盤モジュール)

  出力データ なし．

  処理方式 this.method を method で初期設定し，classNames フィールドと nameAndTypes フィールドに，空のハッシュテーブルを割り当てて格納する．

エラー処理 なし．

(4) メソッド

- public int nlocals()

  機能概要 メソッドの使用するローカル変数の数を返す．

  機能説明 メソッドの使用するローカル変数の数を返す．

  入力データ なし．

  出力データ int 値

  処理方式 method の nlocals メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public boolean isStatic()

  機能概要 メソッドがスタティックメソッドかどうかを返す．

  機能説明 メソッドがスタティックメソッドかどうかを返す．

  入力データ なし．

  出力データ boolean 値

  処理方式 method の access メソッドを呼び出した結果と ACC_STATIC のビット毎論理積が 0 かどうかを返す．

  エラー処理 なし．

- public String thisClassName()

  機能概要 メソッドを持つクラスの名前を返す．

  機能説明 メソッドを持つクラスの名前を返す．

  入力データ なし．

  出力データ String オブジェクト

  処理方式 thisClassName が null でなければそれを返し，null の場合は method の clazz フィールドの getName メソッドを呼び出すことによってクラス名を求め，thisClassName に格納した後にそれを返す．

エラー処理 なし．

- public String className(int index)

  機能概要 コンスタントプールに保持されている特定のクラス名を返す．

  機能説明 コンスタントプール内へのindexで選択される特定のクラス名を返す．

  入力データ index (コンスタントプールへのindex)

  出力データ Stringオブジェクト

  処理方式 indexをキーとしてハッシュテーブルclassNamesを調べ，クラス名が既に格納されていた場合はそれを返す．それ以外の場合はmethodのConstantPoolClassメソッドを呼び出すことによってクラス名を求め，classNamesに格納した後にそれを返す．

  エラー処理 なし．

- public int fieldWidth(int index)

  機能概要 フィールドの値もしくはメソッドの返し値のワード幅を返す．

  機能説明 フィールドの値もしくはメソッドの返し値のワード幅を返す．

  入力データ index (フィールドもしくはメソッドを選択するindex)

  出力データ int値

  処理方式 indexを引数としてnameAndTypeメソッドを呼び出し，目的のフィールドもしくはメソッドの型情報を求め，その情報にしたがってフィールドの値もしくはメソッドの返し値のワード幅を返す．

  エラー処理 なし．

- public NameAndType nameAndType(int index)

  機能概要 フィールドもしくはメソッドの型情報を返す．

  機能説明 フィールドもしくはメソッドの型記述子を解析した結果としてのNameAndTypeオブジェクトを返す．

  入力データ index (フィールドもしくはメソッドを選択するindex)

出力データ NameAndType オブジェクト

処理方式 index をキーとしてハッシュテーブル nameAndTypes を調べ，目的の型情報が既に登録されていた場合はそれを返す．未登録であった場合は，クラス名とフィールドもしくはメソッドの名前，そして，その型記述子を method の適当なメソッドを用いることによって求め，それらを引数として解析結果としての NameAndType オブジェクトを生成する．しかる後に，生成した NameAndType オブジェクトを nameAndTypes に格納した後に，それを返す．

エラー処理 なし．

- public int kindOfConstant(int index)

  機能概要 コンスタントプール内に保持されている定数の種別を返す．

  機能説明 コンスタントプール内に保持されている定数の種別を返す．

  入力データ index (定数を保持するコンスタントプールへの index)

  出力データ int 値

  処理方式 method の ConstantPoolType メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public int constantInt(int index)

  機能概要 コンスタントプールから int 定数を取り出す．

  機能説明 コンスタントプールから int 定数を取り出す．

  入力データ index (int 定数を保持するコンスタントプールへの index)

  出力データ int 値

  処理方式 method の ConstantPoolValue メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public float constantFloat(int index)

  機能概要 コンスタントプールから float 定数を取り出す．

機能説明 コンスタントプールから float 定数を取り出す．

入力データ index (float 定数を保持するコンスタントプールへの index)

出力データ float 値

処理方式 method の ConstantPoolFloatValue メソッドを呼び出し，その結果を返す．

エラー処理 なし．

- public String constantString(int index)

  機能概要 コンスタントプールから String 定数を取り出す．

  機能説明 コンスタントプールから String 定数を取り出す．

  入力データ index (String 定数を保持するコンスタントプールへの index)

  出力データ String オブジェクト

  処理方式 method の ConstantPoolStringValue メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public double constantDouble(int index)

  機能概要 コンスタントプールから double 定数を取り出す．

  機能説明 コンスタントプールから double 定数を取り出す．

  入力データ index (double 定数を保持するコンスタントプールへの index)

  出力データ double 値

  処理方式 method の ConstantPoolDoubleValue メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public long constantLong(int index)

  機能概要 コンスタントプールから long 定数を取り出す．

機能説明 コンスタントプールから long 定数を取り出す．

入力データ index (long 定数を保持するコンスタントプールへの index)

出力データ long 値

処理方式 index を引数として method の ConstantPoolValue メソッドを呼び出した結果を上位 32bit とし，index + 1 を引数として method の ConstantPoolValue メソッドを呼び出した結果を下位 32bit とする long 値を求め，その値を返す．

エラー処理 なし．

- public ExceptionHandler[] exceptionHandler()

機能概要 例外ハンドラテーブルを返す．

機能説明 メソッドの持つ例外ハンドラテーブルを返す．

入力データ なし．

出力データ ExceptionHandler オブジェクトの配列．

処理方式 method の exceptionHandler フィールドを返す．

エラー処理 なし．

- public int bytecodeLength()

機能概要 バイトコードの長さを返す．

機能説明 バイトコードの長さを返す．

入力データ なし．

出力データ int 値

処理方式 method の持つフィールドである bytecode という配列の長さを返す．

エラー処理 なし．

- public int byteAt(int pc)

機能概要 バイトコードの特定のアドレスから byte 値を取り出す．

機能説明 バイトコードの特定のアドレスから byte 値を取り出す．

入力データ pc (バイトコードのアドレス)

出力データ int 値

処理方式 pc を引数として method の pc2signedchar メソッドを呼び出し，その結果を返す．

エラー処理 なし．

- public int unsignedByteAt(int pc)

  機能概要 バイトコードの特定のアドレスから符合無し byte 値を取り出す．

  機能説明 バイトコードの特定のアドレスから符合無し byte 値を取り出す．

  入力データ pc (バイトコードのアドレス)

  出力データ int 値

  処理方式 pc を引数として method の pc2uchar メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public int shortAt(int pc)

  機能概要 バイトコードの特定のアドレスから short 値を取り出す．

  機能説明 バイトコードの特定のアドレスから short 値を取り出す．

  入力データ pc (バイトコードのアドレス)

  出力データ int 値

  処理方式 pc を引数として method の pc2signedshort メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public int unsignedShortAt(int pc)

  機能概要 バイトコードの特定のアドレスから符合無し short 値を取り出す．

機能説明 バイトコードの特定のアドレスから符合無し short 値を取り出す．

入力データ pc (バイトコードのアドレス)

出力データ int 値

処理方式 pc を引数として method の pc2ushort メソッドを呼び出し，その結果を返す．

エラー処理 なし．

- public int intAt(int pc)

機能概要 バイトコードの特定のアドレスから int 値を取り出す．

機能説明 バイトコードの特定のアドレスから int 値を取り出す．

入力データ pc (バイトコードのアドレス)

出力データ int 値

処理方式 pc を引数として method の pc2signedlong メソッドを呼び出し，その結果を返す．

エラー処理 なし．

4.3.29 OpenJIT.frontend.discompiler.Switch クラス

public class Switch extends CFAFlag implements CFAConstants

(1) 概要

このクラスは，コントロールフロー解析の結果として，switch 構造の先頭ノードであると判明したブロックに対して付けられるオブジェクトの構造を定義するものである．

(2) フィールド (変数)

- boolean traceCase デバッグ用の表示を制御するフラグ．
- Expression expression switch 文の分岐を選択する式を表す Expression オブジェクト
- DTNode body switch 構造の本体の先頭を表すノードを保持する．

(3) コンストラクタ

- public Switch(DTNode node, SymbolicValue switchHead)

  機能概要 switch 構造の先頭を表すオブジェクトを生成する．

  機能説明 switch 構造の先頭を表すオブジェクトを生成する．

  入力データ node (switch 構造の先頭となるノード), switchHead (switch 文の分岐を表す SymbolicValue オブジェクト)

  出力データ なし．

  処理方式 現在注目している switch 構造内の case ラベルの付くべきノードにラベルを付け，後のコントロールフロー解析の助けとなるように，先頭ノードが空のノードとなるようにドミネータツリーを作り替える．

  エラー処理 なし．

(4)　メソッド

なし．

4.3.30 OpenJIT.frontend.discompiler.SymbolicConstants インターフェース

public interface SymbolicConstants

(1) 概要

このインターフェースは，SymbolicValue の種別を表す定数を定義するものである．

(2) フィールド (変数)

- int symbolic_unknown 不明の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_void void 型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_byte byte 型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_boolean boolean 型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_char char 型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_short short 型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_int int 型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_long long 型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_float float 型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_double double 型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_object 何らかのオブジェクト型の SymbolicValue オブジェクトであることを表す定数．

- int symbolic_new_array 新しく作られた何らかの配列型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_new_multianewarray 新しく作られた何らかの配列型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_new_object 新しく作られた何らかのクラスのオブジェクト型の SymbolicValue オブジェクトであることを表す定数．
- int symbolic_goto 無条件ブランチを表す SymbolicValue オブジェクトであることを表す定数．
- int symbolic_conditional_branch 条件ブランチを表す SymbolicValue オブジェクトであることを表す定数．
- int symbolic_switch switch 命令を表す SymbolicValue オブジェクトであることを表す定数．
- int symbolic_compare_op 比較命令を表す SymbolicValue オブジェクトであることを表す定数．
- int symbolic_dummy long や double 型の SymbolicValue オブジェクトが本来占めるべきスタックの 2 ワード目であることを表す定数．
- int symbolic_jsr サブルーチンコールを表す SymbolicValue オブジェクトであることを表す定数．
- int symbolic_ret サブルーチンからの復帰を表す SymbolicValue オブジェクトであることを表す定数．
- int symbolic_monitorenter monitorenter 命令を表す SymbolicValue オブジェクトであることを表す定数．
- int symbolic_monitorexit monitorexit 命令を表す SymbolicValue オブジェクトであることを表す定数．
- int symbolic_return return 命令であることを表す SymbolicValue オブジェクトであることを表す定数．

- int symbolic_throw throw 命令であることを表す SymbolicValue オブジェクトであることを表す定数 .

(3) コンストラクタ

なし .

(4) メソッド

なし .

4.3.31 OpenJIT.frontend.discompiler.SymbolicStack クラス

public class SymbolicStack extends Stack implements SymbolicConstants

(1) 概要

このクラスは，ディスコンパイルの際に行われるシンボリック実行の途中経過および結果を格納するためのデータ構造を定義するものである．

(2) フィールド (変数)

- boolean traceStack デバッグ用の表示の制御を行うフラグ．
- int numberOfValues スタックの保持する値の数を保持する．
- int typeOfLastValue スタックに最後に push された値の型を保持する．
- Stack stack スタックの内容を保持する．

(3) コンストラクタ

- public SymbolicStack()

  機能概要 シンボリックスタックを生成する．

  機能説明 シンボリックスタックを生成する．

  入力データ なし．

  出力データ なし．

  処理方式 stack に新たに生成した Stack オブジェクトを格納し，typeOfLastValue を symbolic_unknown で初期設定する．

  エラー処理 なし．

- public SymbolicStack(SymbolicValue value)

  機能概要 最初から値を一つ持っているシンボリックスタックを生成する．

  機能説明 最初から値を一つ持っているシンボリックスタックを生成する．

入力データ value (スタックが最初から持つべき値)

出力データ なし．

処理方式 stack に新たに生成した Stack オブジェクトを格納した後に，value を引数として stack の push メソッドを呼び出す．

エラー処理 なし．

(4) メソッド

- public void push(SymbolicValue value)

  機能概要 スタックに値を push する．

  機能説明 スタックに値を push する．

  入力データ value (スタックに積まれる値)

  出力データ なし．

  処理方式 stack の push メソッドを呼び出し，value が symbolic_long または symbolic_double の値である場合には更に，ダミーの値を積み，最後に，value のフィールドである type を typeOfLastValue に格納する．

  エラー処理 なし．

- public SymbolicValue pop()

  機能概要 スタックから値を pop する．

  機能説明 スタックから値を pop する．

  入力データ なし．

  出力データ SymbolicValue オブジェクト

  処理方式 typeOfLastValue に symbolic_unknown を格納し，stack の pop メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public boolean empty()

機能概要 スタックが空かどうかを判定する．

機能説明 スタックが空かどうかを判定する．

入力データ なし．

出力データ boolean 値

処理方式 stack の empty メソッドを呼び出し，その結果を返す．

エラー処理 なし．

- public Enumeration elements()

  機能概要 スタックの全ての要素を返す．

  機能説明 スタックの全ての要素を返す．

  入力データ なし．

  出力データ Enumeration オブジェクト

  処理方式 stack の elements メソッドを呼び出し，その結果を返す．

  エラー処理 なし．

- public boolean hasValue()

  機能概要 スタックが値を持っているかどうかを判定する．

  機能説明 スタックが値を持っているかどうかを判定する．

  入力データ なし．

  出力データ boolean 値

  処理方式 numberOfValues が 0 以下なら false を返し，その他の場合には true を返す．

  エラー処理 なし．

- public int numberOfValues()

  機能概要 numberOfValues を返す．

  機能説明 numberOfValues を返すアクセッサメソッド．

入力データ なし．

出力データ int 値

処理方式 numberOfValues を返す．

エラー処理 なし．

- public boolean isLastValueConditionalBranch()

  機能概要 スタックに最後に積まれた値が条件分岐を表す値であったかどうかを判定する．

  機能説明 スタックに最後に積まれた値が条件分岐を表す値であったかどうかを判定する．

  入力データ なし．

  出力データ boolean 値

  処理方式 typeOfLastValue が symbolic_conditional_branch かどうかを返す．

  エラー処理 なし．

- public boolean hasConditionalBranchOnly()

  機能概要 スタックに条件分岐を表す値だけが積まれているかどうかを判定する．

  機能説明 スタックに条件分岐を表す値だけが積まれているかどうかを判定する．

  入力データ なし．

  出力データ boolean 値

  処理方式 typeOfLastValue が symbolic_conditional_branch でかつ size メソッドの結果が 1 かどうかを返す．

  エラー処理 なし．

- public void append(SymbolicStack successor)

  機能概要 別の SymbolicStack の中身を連結する．

機能説明 別の SymbolicStack の中身を全て同じ順で，末尾に追加する．

入力データ successor (連結される SymbolicStack オブジェクト)

出力データ なし．

処理方式 successor の要素を一つずつ順に取り出し，ダミー以外の要素について push メソッドを呼び出す．

エラー処理 なし．

- public int typeOfLastValue()

機能概要 typeOfLastValue の値を返す．

機能説明 typeOfLastValue の値を返すアクセッサメソッド．

入力データ なし．

出力データ int 値

処理方式 typeOfLastValue の値を返す．

エラー処理 なし．

4.3.32 OpenJIT.frontend.discompiler.SymbolicValue クラス

public class SymbolicValue implements SymbolicConstants

(1) 概要

このクラスは，ディスコンパイルの際に行われるシンボリック実行の途中経過や結果を表すために使用される SymbolicStack に積まれるオブジェクトの構造を定義するものであり，それぞれのオブジェクトは，AST の部分式あるいは，コントロールフローに影響を与える命令などを表す．

(2) フィールド (変数)

- int type SymbolicValue の種別を表す．
- boolean completed 解析済の SymbolicValue オブジェクトであるかどうかを保持する．
- boolean duplicated VM の dup 命令などによりスタック場でコピーされた値であるかどうかを保持する．
- Object value AST の部分ツリーにまで解析された結果を保持する．
- int width シンボリック実行において，その値がスタックに占めるワード幅を保持する．
- SymbolicValue[] inner オペランド不足当の理由で解析が先伸ばしになっている命令の，現在揃う分のオペランドを保持する．
- int arrayType 新しく作られる配列を表す SymbolicValue オブジェクトについて，配列の型を保持する．
- Expression[] args 初期設定付きの配列を表す SymbolicValue オブジェクトの場合に初期設定の内容を表す AST の部分ツリー群を保持する．
- VMInstruction hint 解析が先伸ばしになっている VM の命令を保持する．

- Expression left 比較命令を保持する SymbolicValue オブジェクトの場合に，比較の左辺を表現する AST の部分ツリーを保持する．
- Expression right 比較命令を保持する SymbolicValue オブジェクトの場合に，比較の右辺を表現する AST の部分ツリーを保持する．
- SymbolicValue dummyValue ダミーの SymbolicValue を保持する．

(3) コンストラクタ

- public SymbolicValue()

  機能概要 ダミーの SymbolicValue オブジェクトを生成する．

  機能説明 ダミーの SymbolicValue オブジェクトを生成する．

  入力データ なし．

  出力データ なし．

  処理方式 width を 0 に，completed を true に，type を symbolic_dummy に，それぞれ初期設定する．

  エラー処理 なし．

- public SymbolicValue(Object value)

  機能概要 AST の部分ツリーを保持する SymbolicValue オブジェクトを生成する．

  機能説明 AST の部分ツリーを保持する SymbolicValue オブジェクトを生成する．

  入力データ value (AST の部分ツリーを表すオブジェクト)

  出力データ なし．

  処理方式 completed を true に，this.value を value で初期設定する．

  エラー処理 なし．

- public SymbolicValue(VMInstruction instruction, SymbolicValue[] material)

機能概要 解析を先伸ばしにする命令に対するSymbolicValueを生成する．

機能説明 解析を先伸ばしにする命令に対するSymbolicValueを生成する．

入力データ instruction (先伸ばしになるVMの命令), material (その時点で揃っている命令のスタックオペランドの配列)

出力データ なし．

処理方式 completedをfalseに，hintをinstructionに，innerをmaterialにそれぞれ初期設定する．

エラー処理 なし．

- public SymbolicValue(Object value, Expression left, Expression right, int type, int width)

  機能概要 部分式を表すASTノードを表現するSymbolicValueオブジェクトを生成する．

  機能説明 部分式を表すASTノードを表現するSymbolicValueオブジェクトを生成する．

  入力データ value (部分式を表すASTノード), left (比較命令の場合の左辺を表すExpressionオブジェクト), right (比較命令の場合の右辺を表すExpressionオブジェクト), type (SymbolicValueの型), width (スタックに占める値のワード幅)

  出力データ なし．

  処理方式 this.valueをvalueで，this.leftをleftで，this.rightをrightで，this.typeをtypeで，this.widthをwidthで，completedをtrueでそれぞれ初期設定する．

  エラー処理 引数として不正な値が与えられた場合，例外DiscompilerErrorを放出する．

(4) メソッド

- public boolean isCompleted()

機能概要 completed を返す．

機能説明 completed を返すアクセッサメソッド．

入力データ なし．

出力データ boolean 値

処理方式 completed を返す．

エラー処理 なし．

- public boolean isProducer()

  機能概要 スタックに値を残す SymbolicValue であるかどうかを判定する．

  機能説明 シンボリック実行の際に，スタックに値を残す SymbolicValue であるかどうかを判定する．

  入力データ なし．

  出力データ boolean 値

  処理方式 width が 0 より大きいかどうかを返す．

  エラー処理 なし．

- public static SymbolicValue dummy()

  機能概要 ダミーの SymbolicValue オブジェクトを返す．

  機能説明 ダミーの SymbolicValue オブジェクトを返す．

  入力データ なし．

  出力データ SymbolicValue オブジェクト

  処理方式 dummyValue を返す．

  エラー処理 なし．

- public String toString()

  機能概要 SymbolicValue を識別する文字列を返す．

  機能説明 SymbolicValue を識別する文字列を返す．

入力データ なし．

出力データ String オブジェクト

処理方式 各フィールドの値を表す文字列を連結した文字列を作り，それを返す．

エラー処理 なし．

- public static String toString(boolean x)

  機能概要 boolean の値を表す文字列を返す．

  機能説明 boolean の値を表す文字列を返す．

  入力データ x (boolean 値)

  出力データ String オブジェクト

  処理方式 x が true なら"true"を返し，それ以外では"false"を返す．

  エラー処理 なし．

4.3.33 OpenJIT.frontend.discompiler.VMInstruction クラス

class VMInstruction implements Constants

(1) 概要

このクラスは，ディスコンパイラで使用されるバイトコードの命令レベルの解析結果を表すために，VM の命令単位に作られるオブジェクトを定義するものである．

(2) フィールド (変数)

- byte opcode 命令のオペコードを保持する．
- short flag 命令やオペランドの種類を区別するフラグ．
- int pcDelta 命令の長さを保持する．
- int BRANCH_SOURCE 分岐の元となる命令であることを表す定数．
- int ESCAPE_SOURCE メソッドのコンテキストからの脱出を表す命令であることを表す定数．
- int INDEX_IS_INTO_CP index に保持されているオペランドがコンスタントプールへのインデックスであることを表す定数．
- int INDEX_IS_INTO_LV index に保持されているオペランドがローカル変数番号であることを表す定数．
- int INDEX_USAGE_MASK index の使用法を表すフラグを選択するマスク．
- int VALUE_IS_IM_VALUE value に保持されているオペランドが即値オペランドであることを表す定数．
- int VALUE_IS_BR_OFFSET value に保持されているオペランドがブランチ命令のオフセットであることを表す定数．
- int VALUE_IS_ARRAYTYPE value に保持されているオペランドが配列の型であることを表す定数．

- int VALUE_IS_NARGS value に保持されているオペランドが invoke_interface 命令の nargs であることを表す定数．
- int VALUE_IS_D_OFFSET value に保持されているオペランドが tableswitch 命令あるいは lookupswitch 命令の default オフセットであることを表す定数．
- int VALUE_IS_DIMENSION value に保持されているオペランドが multianewarray 命令の dimension であることを表す定数．
- int VALUE_USAGE_MASK value の使用法を表すフラグを選択するマスク．
- int MODIFIED_WITH_WIDE 命令が wide 修飾されていることを表す定数．
- int index コンスタントプールへのインデックスあるいはローカル変数番号を表すオペランドを保持する．
- int value index に保持されない種類のオペランドを保持する．
- int misc 解析には使用されないオペランドを保持する．
- BranchTableEntry[] switchTable tableswitch あるいは lookupswitch 命令について，分岐先とキーの組の配列を保持する．
- boolean checkUsageOfSharedField デバッグ用に共用されているフィールドの使用法を確認するかどうかを制御するフラグ．
- VMInstruction[] noOpInsn オペランドのない命令群を保持する配列．
- int[] stackDelta 各命令についてスタック上の値の増減数を保持する．

(3) コンストラクタ

- public VMInstruction()

  機能概要 VMInstruction オブジェクトを生成する．

  機能説明 VMInstruction オブジェクトを生成する．

  入力データ なし．

出力データ なし．

処理方式 flag を 0 で初期設定する．

エラー処理 なし．

- private VMInstruction(int opcode)

  機能概要 VMInstruction オブジェクトを生成する．

  機能説明 オペランドのない命令に対する VMInstruction オブジェクトを生成する．

  入力データ opcode (命令のオペコード)

  出力データ なし．

  処理方式 this.opcode を opcode で，pcDelta を 1 で初期化し，命令ごとに暗黙のオペランドを設定する．

  エラー処理 なし．

(4) メソッド

- public VMInstruction setOpcode(int opcode)

  機能概要 opcode を設定する．

  機能説明 opcode を設定するアクセッサメソッド．

  入力データ opcode (オペコード)

  出力データ VMInstruction オブジェクト

  処理方式 opcode を設定し，this を返す．

  エラー処理 なし．

- public int opcode()

  機能概要 opcode を返す．

  機能説明 opcode を返すアクセッサメソッド．

  入力データ なし．

出力データ int 値

処理方式 opcode を返す．

エラー処理 なし．

- public VMInstruction setPcDelta(int delta)

  機能概要 命令の長さを設定する．

  機能説明 命令の長さを設定するアクセッサメソッド．

  入力データ delta (命令の長さ)

  出力データ VMInstruction オブジェクト

  処理方式 pcDelta に delta を格納し，this を返す．

  エラー処理 なし．

- public int pcDelta()

  機能概要 命令の長さを返す．

  機能説明 命令の長さを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 pcDelta を返す．

  エラー処理 なし．

- public VMInstruction setCPIndex(int index)

  機能概要 コンスタントプールへのインデックスであるオペランドを設定する．

  機能説明 コンスタントプールへのインデックスであるオペランドを設定するアクセッサメソッド．

  入力データ index (コンスタントプールへのインデックス)

  出力データ VMInstruction オブジェクト

  処理方式 this.index に index を格納し，this を返す．

エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public int cpIndex()

  機能概要 コンスタントプールへのインデックスを表すオペランドを返す．

  機能説明 コンスタントプールへのインデックスを表すオペランドを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 index を返す．

  エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public VMInstruction setLVIndex(int index)

  機能概要 ローカル変数番号を表すオペランドを設定する．

  機能説明 ローカル変数番号を表すオペランドを設定するアクセッサメソッド．

  入力データ index (ローカル変数番号)

  出力データ VMInstruction オブジェクト

  処理方式 this.index に index を格納し，this を返す．

  エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public int lvIndex()

  機能概要 ローカル変数番号を表すオペランドを返す．

  機能説明 ローカル変数番号を表すオペランドを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 index を返す．

エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public VMInstruction setArrayType(int arrayType)

  機能概要 配列の型を表すオペランドを設定する．

  機能説明 配列の型を表すオペランドを設定する．

  入力データ arrayType (配列の型を表すオペランド値)

  出力データ VMInstruction オブジェクト

  処理方式 value に arrayType を格納し，this を返す．

  エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public int arrayType()

  機能概要 配列の型を表すオペランドを返す．

  機能説明 配列の型を表すオペランドを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 value を返す．

  エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public VMInstruction setImmediateValue(int value)

  機能概要 即値オペランドを設定する．

  機能説明 即値オペランドを設定するアクセッサメソッド．

  入力データ value (即値オペランドの値)

  出力データ VMInstruction オブジェクト

  処理方式 this.value に value を格納し，this を返す．

エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public int immediateValue()

  機能概要 即値オペランドを返す．

  機能説明 即値オペランドを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 value を返す．

  エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public VMInstruction setBranchOffset(int offset)

  機能概要 ブランチ命令のオフセットを表すオペランドを設定する．

  機能説明 ブランチ命令のオフセットを表すオペランドを設定するアクセッサメソッド．

  入力データ offset (ブランチ命令のオフセット値)

  出力データ VMInstruction オブジェクト

  処理方式 value に offset を格納し，this を返す．

  エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public int branchOffset()

  機能概要 ブランチ命令のオフセットを表すオペランドを返す．

  機能説明 ブランチ命令のオフセットを表すオペランドを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

処理方式 value を返す．

エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public VMInstruction setDefaultOffset(int offset)

  機能概要 default オフセットを表すオペランドを設定する．

  機能説明 default オフセットを表すオペランドを設定するアクセッサメソッド．

  入力データ offset (default オフセットの値)

  出力データ VMInstruction オブジェクト

  処理方式 value に offset を格納し，this を返す．

  エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public int defaultOffset()

  機能概要 default オフセットを表すオペランドを返す．

  機能説明 default オフセットを表すオペランドを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 value を返す．

  エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public VMInstruction setNArgs(int nargs)

  機能概要 nargs を表すオペランドを設定する．

  機能説明 nargs を表すオペランドを設定するアクセッサメソッド．

  入力データ nargs (nargs オペランドの値)

  出力データ VMInstruction オブジェクト

処理方式 value に nargs を格納し，this を返す．

エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public int nargs()

  機能概要 nargs を表すオペランドを返す．

  機能説明 nargs を表すオペランドを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 value を返す．

  エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public VMInstruction setDimensions(int dimensions)

  機能概要 dimensions を表すオペランドを設定する．

  機能説明 dimensions を表すオペランドを設定するアクセッサメソッド．

  入力データ dimensions (dimensions オペランドの値)

  出力データ VMInstruction オブジェクト

  処理方式 value に dimensions を格納し，this を返す．

  エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public int dimensions()

  機能概要 dimensions を表すオペランドを返す．

  機能説明 dimensions を表すオペランドを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

処理方式 value を返す．

エラー処理 オペランドの使用方法が不正である場合は，例外 DiscompilerError を放出する．

- public VMInstruction setGuess(int x)

  機能概要 guess を表すオペランドを設定する．

  機能説明 guess を表すオペランドを設定するアクセッサメソッド．

  入力データ x (guess オペランドの値)

  出力データ VMInstruction オブジェクト

  処理方式 misc に x を格納し，this を返す．

  エラー処理 なし．

- public int guess()

  機能概要 guess を表すオペランドを返す．

  機能説明 guess を表すオペランドを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 misc を返す．

  エラー処理 なし．

- public VMInstruction setCheckNull(int x)

  機能概要 null チェックが必要な命令であるかどうかを表すオペランドを設定する．

  機能説明 null チェックが必要な命令であるかどうかを表すオペランドを設定するアクセッサメソッド．

  入力データ x (null チェックの必要性を表すオペランド値)

  出力データ VMInstruction オブジェクト

処理方式 misc に x を格納し，this を返す．

エラー処理 なし．

- public int checkNull()

  機能概要 null チェックが必要な命令かどうかを表すオペランドを返す．

  機能説明 null チェックが必要な命令かどうかを表すオペランドを返すアクセッサメソッド．

  入力データ なし．

  出力データ int 値

  処理方式 misc を返す．

  エラー処理 なし．

- VMInstruction setBranchSource()

  機能概要 ブランチ元となる命令であることを表すフラグを設定する．

  機能説明 ブランチ元となる命令であることを表すフラグを設定するアクセッサメソッド．

  入力データ なし．

  出力データ VMInstruction オブジェクト

  処理方式 flag と BRANCH_SOURCE とのビット毎論理和を flag に格納し，this を返す．

  エラー処理 なし．

- public boolean isBranchSource()

  機能概要 ブランチ元となる命令であるかどうかを返す．

  機能説明 ブランチ元となる命令であるかどうかを返す．

  入力データ なし．

  出力データ boolean 値

処理方式 flag と BRANCH_SOURCE のビット毎論理積が 0 かどうかを返す．

エラー処理 なし．

- VMInstruction setEscapeSource()

  機能概要 メソッドのコンテキストからの脱出元となる命令であることを表すフラグを設定する．

  機能説明 メソッドのコンテキストからの脱出元となる命令であることを表すフラグを設定するアクセッサメソッド．

  入力データ なし．

  出力データ VMInstruction オブジェクト

  処理方式 flag と ESCAPE_SOURCE のビット毎論理和を flag に格納し，this を返す．

  エラー処理 なし．

- public boolean isEscapeSource()

  機能概要 メソッドのコンテキストからの脱出元のなる命令であるかどうかを返す．

  機能説明 メソッドのコンテキストからの脱出元のなる命令であるかどうかを返す．

  入力データ なし．

  出力データ boolean 値

  処理方式 flag と ESCAPE_SOURCE のビット毎論理積が 0 かどうかを返す．

  エラー処理 なし．

- public void setWideModified()

  機能概要 命令が wide 修飾されていることを表すフラグを設定する．

  機能説明 命令が wide 修飾されていることを表すフラグを設定する．

  入力データ なし．

出力データ なし．

処理方式 flag と MODIFIED_WITH_WIDE のビット毎論理和を flag に格納する．

エラー処理 なし．

- public boolean isWideModified()

  機能概要 命令が wide 修飾されているかどうかを返す．

  機能説明 命令が wide 修飾されているかどうかを返す．

  入力データ なし．

  出力データ boolean 値

  処理方式 flag と MODIFIED_WITH_WIDE のビット毎論理積が 0 かどうかを返す．

  エラー処理 なし．

- public boolean indexAlreadyUsed()

  機能概要 index フィールドがすでに使用されているかどうかを返す．

  機能説明 index フィールドがすでに使用されているかどうかを返す．

  入力データ なし．

  出力データ boolean 値

  処理方式 flag と INDEX_USAGE_MASK のビット毎論理積が 0 かどうかを返す．

  エラー処理 なし．

- public void setIndexIsIntoCP()

  機能概要 index フィールドがコンスタントプールへのインデックスを表すオペランドを保持していることを表すフラグを設定する．

  機能説明 index フィールドがコンスタントプールへのインデックスを表すオペランドを保持していることを表すフラグを設定するアクセッサメソッド．

  入力データ なし．

出力データ なし．

処理方式 flag と INDEX_IS_INTO_CP のビット毎論理和を flag に格納する．

エラー処理 なし．

- public boolean isIndexIntoCP()

  機能概要 index フィールドがコンスタントプールへのインデックスを表すオペランドを保持しているかどうかを返す．

  機能説明 index フィールドがコンスタントプールへのインデックスを表すオペランドを保持しているかどうかを返す．

  入力データ なし．

  出力データ boolean 値

  処理方式 flag と INDEX_IS_INTO_CP のビット毎論理積が 0 かどうかを返す．

  エラー処理 なし．

- public void setIndexIsIntoLV()

  機能概要 index フィールドがローカル変数番号を表すオペランドを保持していることを表すフラグを設定する．

  機能説明 index フィールドがローカル変数番号を表すオペランドを保持していることを表すフラグを設定するアクセッサメソッド．

  入力データ なし．

  出力データ なし．

  処理方式 flag と INDEX_IS_INTO_LV のビット毎論理和を flag に格納する．

  エラー処理 なし．

- public boolean isIndexIntoLV()

  機能概要 index フィールドがローカル変数番号を表すオペランドを保持しているかどうかを返す．

機能説明 index フィールドがローカル変数番号を表すオペランドを保持しているかどうかを返す．

入力データ なし．

出力データ なし．

処理方式 flag と INDEX_IS_INTO_LV のビット毎論理積が 0 かどうかを返す．

エラー処理 なし．

- public boolean valueAlreadyUsed()

  機能概要 value フィールドがすでに使用されているかどうかを返す．

  機能説明 value フィールドがすでに使用されているかどうかを返す．

  入力データ なし．

  出力データ boolean 値

  処理方式 flag と VALUE_USAGE_MASK のビット毎論理積が 0 かどうかを返す．

  エラー処理 なし．

- public void setValueIsImmediate()

  機能概要 value フィールドが即値オペランドを保持していることを表すフラグを設定する．

  機能説明 value フィールドが即値オペランドを保持していることを表すフラグを設定するアクセッサメソッド．

  入力データ なし．

  出力データ なし．

  処理方式 flag と VALUE_IS_IM_VALUE のビット毎論理和を flag に格納する．

  エラー処理 なし．

- public boolean isValueImmediate()

  機能概要 value フィールドが即値オペランドを保持しているかどうかを返す．

機能説明 value フィールドが即値オペランドを保持しているかどうかを返す．

入力データ なし．

出力データ boolean 値

処理方式 flag と VALUE_IS_IM_VALUE のビット毎論理積が 0 かどうかを返す．

エラー処理 なし．

- public void setValueIsBranchOffset()

  機能概要 value フィールドがブランチオフセットを表すオペランドを保持していることを表すフラグを設定する．

  機能説明 value フィールドがブランチオフセットを表すオペランドを保持していることを表すフラグを設定するアクセッサメソッド．

  入力データ なし．

  出力データ なし．

  処理方式 flag と VALUE_IS_BR_OFFSET のビット毎論理和を flag に格納する．

  エラー処理 なし．

- public boolean isValueBranchOffset()

  機能概要 value フィールドがブランチオフセットを保持しているかどうかを返す．

  機能説明 value フィールドがブランチオフセットを保持しているかどうかを返す．

  入力データ なし．

  出力データ boolean 値

  処理方式 flag と VALUE_IS_BR_OFFSET のビット毎論理積が 0 かどうかを返す．

  エラー処理 なし．

- public void setValueIsDefaultOffset()

機能概要 value フィールドが default オフセットを表すオペランドを保持していることを表すフラグを設定する．

機能説明 value フィールドが default オフセットを表すオペランドを保持していることを表すフラグを設定するアクセッサメソッド．

入力データ なし．

出力データ なし．

処理方式 flag と VALUE_IS_D_OFFSET のビット毎論理和を flag に格納する．

エラー処理 なし．

- public boolean isValueDefaultOffset()

機能概要 value フィールドが default オフセットを保持しているかどうかを返す．

機能説明 value フィールドが default オフセットを保持しているかどうかを返す．

入力データ なし．

出力データ boolean 値

処理方式 flag と VALUE_IS_D_OFFSET のビット毎論理積が 0 かどうかを返す．

エラー処理 なし．

- public void setValueIsArrayType()

機能概要 value フィールドが配列の種別を表すオペランドを保持していることを表すフラグを設定する．

機能説明 value フィールドが配列の種別を表すオペランドを保持していることを表すフラグを設定するアクセッサメソッド．

入力データ なし．

出力データ なし．

処理方式 flag と VALUE_IS_ARRAYTYPE のビット毎論理和を flag に格納する．

エラー処理 なし．

- public boolean isValueArrayType()

  機能概要 value フィールドが配列の種別を保持しているかどうかを返す．

  機能説明 value フィールドが配列の種別を保持しているかどうかを返す．

  入力データ なし．

  出力データ boolean 値

  処理方式 flag と VALUE_IS_ARRAYTYPE のビット毎論理積が 0 かどうかを返す．

  エラー処理 なし．

- public void setValueIsNArgs()

  機能概要 value フィールドが nargs オペランドを保持していることを表すフラグを設定する．

  機能説明 value フィールドが nargs オペランドを保持していることを表すフラグを設定するアクセッサメソッド．

  入力データ なし．

  出力データ なし．

  処理方式 flag と VALUE_IS_NARGS のビット毎論理和を flag に格納する．

  エラー処理 なし．

- public boolean isValueNArgs()

  機能概要 value フィールドが nargs オペランドを保持しているかどうかを返す．

  機能説明 value フィールドが nargs オペランドを保持しているかどうかを返す．

  入力データ なし．

  出力データ boolean 値

  処理方式 flag と VALUE_IS_NARGS のビット毎論理積が 0 かどうかを返す．

エラー処理 なし．

- public void setValueIsDimensions()

  機能概要 value フィールドが dimensions オペランドを保持していることを表すフラグを設定する．

  機能説明 value フィールドが dimensions オペランドを保持していることを表すフラグを設定するアクセッサメソッド．

  入力データ なし．

  出力データ なし．

  処理方式 flag と VALUE_IS_DIMENSION のビット毎論理和を flag に格納する．

  エラー処理 なし．

- public boolean isValueDimensions()

  機能概要 value フィールドが dimensions オペランドを保持しているかどうかを返す．

  機能説明 value フィールドが dimensions オペランドを保持しているかどうかを返す．

  入力データ なし．

  出力データ boolean 値

  処理方式 flag と VALUE_IS_DIMENSION のビット毎論理積が 0 かどうかを返す．

  エラー処理 なし．

- public String opcName(int opcode)

  機能概要 インストラクション名を返す．

  機能説明 インストラクション名を返す．

  入力データ opcode (オペコード)

  出力データ String オブジェクト

処理方式 sun.tools.java.RuntimeConstants のフィールド opcNames の opcode でインデックスされる要素を返す．配列の境界を越える opcode の場合，opcode を 10 進数表現に変換した文字列を返す．

エラー処理 なし．

- public void print(IndentedPrintStream out, int pc)

機能概要 オブジェクトの内容をわかりやすく表示する．

機能説明 オブジェクトの内容をわかりやすく表示する．

入力データ out (出力先を指定する IndentedPrintStream オブジェクト) pc (命令の先頭アドレス)

出力データ なし．

処理方式 tableswitch あるいは lookupswitch 命令の場合は printTableInstruction メソッドを呼び出し，それ以外の命令については，printOneLineInstruction メソッドを呼び出す．

エラー処理 なし．

- void printTableInstruction(IndentedPrintStream out, int pc)

機能概要 tableswitch あるいは lookupswitch 命令をわかりやすく表示する．

機能説明 tableswitch あるいは lookupswitch 命令をわかりやすく表示する．

入力データ out (出力先を指定する IndentedPrintStream オブジェクト) pc (命令の先頭アドレス)

出力データ なし．

処理方式 命令を表す文字列を表示した後に，switchTable の内容を表示し，最後に default のジャンプ先アドレスを表示する．

エラー処理 なし．

- void printOneLineInstruction(IndentedPrintStream out, int pc)

機能概要 オブジェクトをわかりやすく表示する．

機能説明 一行で表示できる単純な命令をわかりやすく表示する．

入力データ out (出力先を指定する IndentedPrintStream オブジェクト) pc (命令の先頭アドレス)

出力データ なし．

処理方式 命令を表す文字列と，オペランドを組み合わせてわかりやすく表示する．

エラー処理 なし．

- public String toString()

機能概要 命令を識別する文字列を返す．

機能説明 命令を識別する文字列を返す．

入力データ なし．

出力データ String オブジェクト

処理方式 それぞれのフィールドの内容を連結した文字列を返す．

エラー処理 なし．

4.4 OpenJIT.frontend.flowgraph パッケージ

- インターフェース
  - OpenJIT.frontend.flowgraph.FlowGraphAnalyzer
- 抽象クラス
  - OpenJIT.frontend.flowgraph.Optimizer
- クラス
  - OpenJIT.frontend.flowgraph.FlowGraph
  - OpenJIT.frontend.flowgraph.CHInfo
  - OpenJIT.frontend.flowgraph.ControlDependencyGraph
  - OpenJIT.frontend.flowgraph.DataFlowGraph
  - OpenJIT.frontend.flowgraph.ClassHierarchyGraph
  - OpenJIT.frontend.flowgraph.FlowGraphAnalysis
  - OpenJIT.frontend.flowgraph.DFFunctionRegisiter
  - OpenJIT.frontend.flowgraph.ReachingAnalyzer
  - OpenJIT.frontend.flowgraph.AvailableAnalyzer
  - OpenJIT.frontend.flowgraph.LivenessAnalyzer
  - OpenJIT.frontend.flowgraph.FixedPointDetector
  - OpenJIT.frontend.flowgraph.ClassHierarchyAnalyzer
  - OpenJIT.frontend.flowgraph.ASTTransformer
- 例外

  なし .

4.4.1 OpenJIT.frontend.flowgraph.FlowGraph クラス

public class FlowGraph

(1) 概要

データフローグラフ・コントロール依存グラフ・クラス階層グラフを構築し，構築されたグラフの情報を保持する．

(2) フィールド (変数)

- Node ast
  AST
- Class[] cinfo
  クラスファイル間のクラス階層情報
- ControlFlowGraph cfg
  コントロールフローグラフ
- ControlDependencyGraph cdg
  コントロール依存グラフ
- DataFlowGraph dfg
  データフローグラフ
- ClassHierarchyGraph chg
  クラス階層グラフ

(3) コンストラクタ

- public FlowGraph()

  機能概要 グラフ情報を初期化する．

  機能説明 データフローグラフ・コントロール依存グラフ・クラス階層グラフを初期化する．

  入力データ なし

  出力データ なし

処理方式 データフローグラフの初期化はDataFlowGraphオブジェクトを生成し，フィールドdfgへ代入する．コントロール依存グラフの初期化はControlDependencyGraphオブジェクトを生成し，フィールドcdgへ代入する．クラス階層グラフの初期化はClassHierarchyGraphオブジェクトを生成し，フィールドchgへ代入する．

エラー処理 なし

(4) メソッド

- public void setAST(Node s)

  機能概要 入力であるASTをセットする．

  機能説明 入力であるAST(Nodeオブジェクト)をセットする．

  入力データ s(Nodeオブジェクト)

  出力データ なし

  処理方式 引数sをフィールドastへ代入する．

  エラー処理 なし

- public void setCFG(ControlFlowGraph cfg)

  機能概要 コントロールフローグラフをセットする．

  機能説明 入力であるコントロールフローグラフ(ControlFlowGraphオブジェクト)をセットする

  入力データ cfg(ControlFlowGraphオブジェクト)

  出力データ なし

  処理方式 引数cfgをフィールドcfgへセットする．

  エラー処理 なし

- public void setCHInfo(Class[] info)

  機能概要 入力であるクラス階層情報をセットする．

機能説明 入力であるクラスファイル間のクラス階層情報 (Object オブジェクト) をセットする．

入力データ info(Class オブジェクトの配列)

出力データ なし

処理方式 引数 info をフィールド cinfo へセットする．

エラー処理 なし

- public void constructDFG()

機能概要 データフローグラフを構築する．

機能説明 データフローグラフを構築する．

入力データ なし

出力データ なし

処理方式 DataFlowGraph クラスの constructGraph() メソッドを呼び出すことにより，データフローグラフを構築する．

エラー処理 なし

- public void constructCDG()

機能概要 コントロール依存グラフを構築する．

機能説明 コントロール依存グラフを構築する．

入力データ なし

出力データ なし

処理方式 ControlDependencyGraph クラスの constructGraph() メソッドを呼び出すことにより，コントロール依存グラフを構築する．

エラー処理 なし

- public void constructCH()

機能概要 クラス階層解析を行う．

機能説明 クラス階層解析を行い，クラス階層グラフを構築する．

入力データ なし

出力データ なし

処理方式 ClassHierarchyGraph クラスの constructGraph() メソッドを呼び出すことにより，クラス階層グラフを構築する．

エラー処理 なし

4.4.2 OepnJIT.frontend.flowgraph.CHInfo クラス

class CHInfo

(1) 概要

クラス階層グラフの 1 ノードを表現する．

(2) フィールド (変数)

- Class body
  自分のクラス情報
- CHInfo parent
  スーパークラスのクラス階層グラフのノード
- CHInfo[] child
  子のクラス階層グラフのノード

(3) コンストラクタ

- public CHInfo(Class c)

  機能概要 クラス階層グラフの 1 ノードを初期化する．

  機能説明 自分のクラス情報を与えてクラス階層グラフの 1 ノードを初期化する．

  入力データ c(Class オブジェクト)

  出力データ なし

  処理方式 フィールド body へ引数 c をセットする．

  エラー処理 なし

(4) メソッド

なし

4.4.3 OpenJIT.frontend.flowgraph.ControlDependencyGraph クラス

public class ControlDependencyGraph

(1) 概要

コントロール依存グラフの構築と，構築されたコントロール依存グラフを保持し，フローグラフ解析時に必要な情報を提供する．

(2) フィールド (変数)

- Node ast
  AST
- ControlFlowGraph cfg
  コントロールフローグラフ
- FlowGraph fg
  ControlDependencyGraph 生成元の FlowGraph オブジェクト

(3) コンストラクタ

- public ControlDependencyGraph(FlowGraph fg)

  機能概要 データ依存グラフを初期化する．

  機能説明 フロー解析で用いるデータ依存グラフを生成するためにデータ依存グラフを初期化する．

  入力データ fg(FlowGraph オブジェクト)

  出力データ なし

  処理方式 引数 fg をフィールド fg へセットする．次に，フィールド fg 内のフィールド ast をフィールド ast へセットし，フィールド fg 内のフィールド cfg をフィールド cfg へセットする．

  エラー処理 なし

(4) メソッド

- public void constructGraph()

  機能概要 データ依存グラフを構築する．

  機能説明 フローグラフ解析で用いるデータ依存グラフを構築する．

  入力データ なし

  出力データ なし

  処理方式 フィールドに持つ Node オブジェクト ast を引数として seekStatemnet メソッドを呼び出す．

  エラー処理 なし

- private void seekStatement(Statement s)

  機能概要 入力 s より，コントロール依存グラフ構築のための情報を得る．

  機能説明 入力である式 s を走査し，Expression オブジェクトが含まれていれば，その Expression オブジェクトに対してコントロール依存グラフ構築のための情報を検出・格納する操作を実行させる．

  入力データ s (Statemnet オブジェクト)

  出力データ なし

  処理方式 入力 s のクラスを調べ，各クラスに対応した操作を行う．クラスのフィールドとして Statement オブジェクトを持つ場合は，その Statement オブジェクトに対して seekStatement メソッドを呼び出す．

  エラー処理 なし

4.4.4 OpenJIT.frontend.flowgraph.DataFlowGraph クラス

public class DataFlowGraph

(1) 概要

データフローグラフの構築と，構築されたデータフローグラフを保持し，フローグラフ解析時に必要な情報を提供する．

(2) フィールド (変数)

- Node ast
  AST
- FlowGraph fg
  DataFlowGraph 生成元の FlowGraph オブジェクト
- Hashtable leftexptable
  データフロー解析のための情報を保持する．具体的には式において左辺に現れる変数と行番号の対応の情報を持つ．
- Hashtable rightexptable
  データフロー解析のための情報を保持する．具体的には式において右辺に現れる変数と行番号の対応の情報を持つ．
- Hashtable leftlinetable
  データフロー解析のための情報を保持する．具体的には行番号と式において左辺に現れる変数との対応の情報を持つ．
- Hashtable rightlinetable
  データフロー解析のための情報を保持する．具体的には行番号と式において右辺に現れる変数との対応の情報を持つ．

(3) コンストラクタ

- public DataFlowGraph(FlowGraph fg)

機能概要 データフローグラフの初期化を行う．

機能説明 フローグラフ解析で用いるデータフローグラフを初期化する．

入力データ fg(FlowGraph オブジェクト)

出力データ なし

処理方式 引数 fg をフィールド fg へセットし，フィールド fg のフィールド ast をフィールド ast へセットする．

エラー処理 なし

(4) メソッド

- public void constructGraph()

  機能概要 データフローグラフの構築を行う．

  機能説明 seekStatement メソッドを呼び出すことによりデータフローグラフの構築を行う．

  入力データ なし

  出力データ なし (得られたデータフロー解析のための情報はフィールド leftexptable，rightexptable，leftlinetable，rightlinetable に格納される)

  処理方式 フィールドに持つ Node オブジェクト ast を引数として seekStatemnet メソッドを呼び出す．

  エラー処理 なし

- private void seekStatement(Statement s)

  機能概要 入力 s より，データフローグラフ構築のための情報を得る．

  機能説明 入力である式 s を走査し，Expression オブジェクトが含まれていれば，その Expression オブジェクトに対してデータフローグラフ構築のための情報を検出・格納する操作を実行させる．

  入力データ s (Statemnet オブジェクト)

  出力データ なし

処理方式 入力 s のクラスを調べ，各クラスに対応した操作を行う．クラスのフィールドとして Statement オブジェクトを持つ場合は，その Statement オブジェクトに対して seekStatement メソッドを呼び出す．また，クラスのフィールドとして Expression オブジェクトを持つ場合は，その Expression オブジェクトに対して seekExpression メソッドを呼び出す．

エラー処理 なし

- private void seekExpression(Expression e)

  機能概要 入力 e より，データフローグラフ構築のための情報を得る．

  機能説明 入力である式 e を走査し，データフローグラフ構築のための情報を検出・格納する．

  入力データ e (Expression オブジェクト)

  出力データ なし (得られたデータフローグラフ構築のための情報はフィールド leftexptable，rightexptable，leftlinetable，rightlinetable に格納される)

  処理方式 入力である Expression オブジェクト e より左辺・右辺に現れる変数を見つけ出し，行番号との対応をフィールド leftexptable，rightexptable，leftlinetable，rightlinetable に格納する．入力である Expression オブジェクト e が検査対象として粗ければ，その Expression オブジェクトを分割して再び seekExpression メソッドを呼び出す．また，入力である Expression オブジェクト e が十分細かければ，処理を行わない．

  エラー処理 なし

- private void initDataFlowGraph()

  機能概要 データフロー解析のための情報を初期化する．

  機能説明 フィールドの Node オブジェクト ast に対してデータフロー解析のための情報を与える．

  入力データ なし

  出力データ なし

  処理方式 フィールドの Node オブジェクト ast に対してデータフロー解析のための情報を initFlowInfo メソッドによって与える．与えられる情報はフィー

ルドの Node オブジェクト s に含まれる Node オブジェクトの数 (フィールド stat) および constructFlowInfo メソッドで得られた情報 (フィールド leftexptable , rightexptable , leftlinetable , rightlinetable に格納されている) である .

エラー処理 なし

4.4.5 OpenJIT.frontend.flowgraph.ClassHierarchyGraph クラス

public class ClassHierarchyGraph

(1) 概要

クラス階層解析を行うことによりクラス階層グラフを構築し，構築されたクラス階層グラフを保持する．さらに，フローグラフ解析時に必要な情報を提供する．

(2) フィールド (変数)

- CHInfo chg
  クラス階層グラフ
- class[] cinfo
  クラスファイル間のクラス階層情報．
- FlowGraph fg
  ClassHierarchyGraph オブジェクトの生成元の FlowGraph オブジェクト．

(3) コンストラクタ

- public ClassHierarchyGraph(FlowGraph fg)

  機能概要 クラス階層グラフを初期化する．

  機能説明 クラス階層解析で用いられるクラス階層グラフのフィールドを初期化する．

  入力データ fg(FlowGraph オブジェクト)

  出力データ なし

  処理方式 引数 fg をフィールド fg へセットし，フィールド fg 内のフィールド cinfo をフィールド cinfo へ格納する．

  エラー処理 なし

(4) メソッド

- public void constructGraph()

  機能概要 クラス階層グラフを構築する．

  機能説明 フィールドに持つ情報 (クラスファイル間のクラス階層情報・フローグラフ情報) を用いてクラス階層グラフを構築する．

  入力データ なし

  出力データ なし

  処理方式 クラスファイル間のクラス階層情報に含まれる各クラスについて，そのsuperクラスとimplementクラスなどを列挙し，それらのすべてのsuperクラスを親とした木を構築し，得られたクラス関係の情報に基づき順に子のノードを追加することで，クラス階層グラフを構築する．

  エラー処理 なし

4.4.6 OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラス

public class FlowGraphAnalysis

(1) 概要

フローグラフ解析を制御する．フローグラフ構築で得られたデータフローグラフ，コントロール依存グラフ，クラス階層グラフとコントロールフローグラフを用いて，必要な解析を行う関数を登録し，各種フローグラフ解析 (reaching analysis, available analysis, liveness analysis, fixed point detection) を行い，それらの解析で得られた結果を保持し，プログラム変換時に必要な情報を提供する．

(2) フィールド (変数)

- FlowGraph fg
  構築されたデータフローグラフ・コントロール依存グラフ・クラス階層グラフを格納した FlowGraph オブジェクト．
- Node ast
  AST．フィールド fg のフィールド AST と同じもの．
- ControlFlowGraph cfg
  構築されたコントロールフローグラフ．フィールド fg のフィールド cfg と同じもの．
- ControlDependencyGraph cdg
  構築されたコントロール依存グラフ．フィールド fg のフィールド cdg と同じもの．
- DataFlowGraph dfg
  構築されたデータフローグラフフィールド fg のフィールド dfg と同じもの．
- ClassHierarchyGraph chg
  構築されたクラス階層グラフ．フィールド fg のフィールド chg と同じもの．
- FlowGraphAnalyzer r

データフローグラフ解析の 1 つである到達定義の解析 (reaching analysis) を行うオブジェクト．

- FlowGraphAnalyzer a
  データフローグラフ解析の 1 つである利用可能な式の解析 (available analysis) を行うオブジェクト．

- FlowGraphAnalyzer l
  データフローグラフ解析の 1 つである生きている式の解析 (liveness analysis) を行うオブジェクト．

- FlowGraphAnalyzer f
  フローグラフ解析の 1 つである不動点検出を行うオブジェクト

- FlowGraphAnalyzer h
  クラス階層解析を行うオブジェクト

(3) コンストラクタ

- public FlowGraphAnalysis(FlowGraph fg)

  機能概要 フローグラフ解析情報を初期化する．

  機能説明 フローグラフ解析を行う上で必要となる情報を初期化・セットする．

  入力データ fg(FlowGraph オブジェクト)

  出力データ なし

  処理方式 引数 fg をフィールド fg へセットする．次にフローグラフ解析で用いるフローグラフ解析関数オブジェクト (ReachingAnalyzer, AvailableAnalyzer, LivenessAnalyzer, FixedPointDetector, ClassHierarchyAnalyzer) を生成し，そのうちデータフロー関数にあたるものを DFFunctionRegister オブジェクトへ登録する．また，生成したオブジェクトをフィールド (順に r, a, l, f, c) へ格納する．

  エラー処理 なし

(4) メソッド

- public void analysis()

  機能概要 各種フローグラフ解析を行う．

  機能説明 コンストラクタで生成したフローグラフ解析オブジェクトを用いてフローグラフ解析を行う．

  入力データ なし

  出力データ なし

  処理方式 フィールドに格納されているフローグラフ解析オブジェクトそれぞれ (r, a, l ,f c) について，インターフェース FlowGraphAnalyzer にある analysis() メソッドを呼び出すことによってフローグラフ解析を行う．

  エラー処理 なし

4.4.7 OpenJIT.frontend.flowgraph.DFFunctionRegister クラス

public class DFFunctionRegister

(1) 概要

フローグラフ解析時に解析を行う関数を登録する．

(2) フィールド (変数)

- FlowGraphAnalyzer[] fga
  登録されたデータフロー関数オブジェクト FlowGraphAnalyzer の配列．
- int registered
  登録されたデータフロー関数オブジェクトの数．
- boolean empty
  登録されているデータフロー関数があるかどうか．

(3) コンストラクタ

- public DFFunctionRegister()

  機能概要 データフロー関数登録を初期化する．

  機能説明 データフロー関数の登録を行うフィールド等の初期化を行う．

  入力データ なし

  出力データ なし

  処理方式 フィールド fga を適当な大きさ (デフォルトは 5) の配列で初期化する．次に，登録されているデータフロー関数はオブジェクト生成時にはないので，フィールド empty を true にし，フィールド registered も 0 にする．

  エラー処理 なし

(4) メソッド

- void register(FlowGraphAnalyzer fga)

  機能概要 データフロー関数を登録する．

  機能説明 インターフェースFlowGraphAnalyzerを実装したデータフロー関数を登録する．

  入力データ なし

  出力データ なし

  処理方式 引数fgaをフィールドfgaの配列のフィールドregisteredで指される場所へ格納し，フィールドregisteredを1増やす．

  エラー処理 なし

4.4.8 OpenJIT.frontend.flowgraph.FlowGraphAnalyzer インターフェース

public interface FlowGraphAnalyzer

(1) 概要

フローグラフ解析を行うために必要なメソッド等を定義したインターフェース．フローグラフ解析に用いる関数はこのインターフェースを実装し，登録する必要がある．

(2) フィールド (変数)

なし

(3) コンストラクタ

なし

(4) メソッド

- void analysis()

  機能概要 フローグラフ解析を行う．

  機能説明 フローグラフ解析を行うクラスが実装すべきメソッド．

  入力データ なし

  出力データ なし

  処理方式 なし (実装側で記述される)

  エラー処理 なし

4.4.9 OpenJIT.frontend.flowgraph.ReachingAnalyzer クラス

public class ReachingAnalyzer

(1) 概要

データフロー解析の 1 つである到達定義の解析 (reaching analysis) を行い，その解析結果を保持する．

(2) フィールド (変数)

- FlowGraphAnalysis fga
  解析元の FlowGraphAnalysis オブジェクトを保持する．

- Node ast
  AST

(3) コンストラクタ

- public ReachingAnalyzer(FlowGraphAnalysis fga)

  機能概要 到達定義の解析を行うための初期化を行う．

  機能説明 フローグラフ解析の 1 つである到達定義の解析を行うためのデータ等の初期化を行う．

  入力データ なし

  出力データ なし

  処理方式 引数 fga をフィールド fga へセットし，フィールド fga のフィールド ast をフィールド ast へセットする．

  エラー処理 なし

(4) メソッド

- public void analysis()

機能概要 インターフェース FlowGraphAnalyzer の実装

機能説明 データフローグラフ解析の 1 つである到達定義解析 (reaching analysis) を行う .

入力データ なし

出力データ なし

処理方式 フィールド ast に対してメソッド reachingAnalysis() を呼ぶ .

エラー処理 なし

4.4.10 OpenJIT.frontend.flowgraph.AvailableAnalyzer クラス

public class AvailableAnalyzer

(1) 概要

データフロー解析の 1 つである，利用可能な式の解析 (available analysis) を行い，その解析結果を保持する．

(2) フィールド (変数)

- FlowGraphAnalysis fga
  解析元の FlowGraphAnalysis オブジェクトを保持する．
- Node ast
  AST

(3) コンストラクタ

- public AvailableAnalyzer(FlowGraphAnalysis fga)

  機能概要 利用可能な式の解析を行うための初期化を行う．

  機能説明 フローグラフ解析の 1 つである利用可能な式解析を行うためのデータ等の初期化を行う．

  入力データ なし

  出力データ なし

  処理方式 引数 fga をフィールド fga へセットし，フィールド fga のフィールド ast をフィールド ast へセットする．

  エラー処理 なし

(4) メソッド

- public void analysis()

機能概要 インターフェース FlowGraphAnalyzer の実装

機能説明 データフローグラフ解析の 1 つである利用可能な式 (available analysis 解析を行う .

入力データ なし

出力データ なし

処理方式 フィールド ast に対してメソッド availableAnalysis() を呼ぶ .

エラー処理 なし

4.4.11 OpenJIT.frontend.flowgraph.LivenessAnalyzer クラス

public class LivenessAnalyzer

(1) 概要

データフロー解析の 1 つである，生きている式の解析 (liveness analysis) を行い，そこ解析結果を保持する．

(2) フィールド (変数)

- FlowGraphAnalysis fga
  解析元の FlowGraphAnalysis オブジェクトを保持する．

- Node ast
  AST

(3) コンストラクタ

- public LivenessAnalysis(FlowGraphAnalysis fga)

  機能概要 生きている式の解析を行うための初期化を行う．

  機能説明 データフローグラフ解析の 1 つである生きている式の解析を行うためのデータ等の初期化を行う．

  入力データ なし

  出力データ なし

  処理方式 引数 fga をフィールド fga へセットし，フィールド fga のフィールド ast をフィールド ast へセットする．

  エラー処理 なし

(4) メソッド

- public void analysis()

機能概要 インターフェース FlowGraphAnalyzer の実装

機能説明 データフローグラフ解析の 1 つである生きている式の解析 (liveness analysis) を行う．

入力データ なし

出力データ なし

処理方式 フィールド ast に対してメソッド livenessAnalysis() を呼ぶ．

エラー処理 なし

4.4.12 OpenJIT.frontend.flowgraph.FixedPointDetector クラス

public class FixedPointDetector

(1) 概要

フローグラフ解析の 1 つである，不動点検出 (fixed point detection) を行い，その解析結果を保持する．

(2) フィールド (変数)

- FlowGraphAnalysis fga
  解析元の FlowGraphAnalysis オブジェクトを保持する．
- Node ast
  AST

(3) コンストラクタ

- public FixedPointDetector(FlowGraphAnalysis fga)

  機能概要 不動点検出を行うための初期化を行う．

  機能説明 フローグラフ解析の 1 つである不動点検出を行うためのデータ等の初期化を行う．

  入力データ なし

  出力データ なし

  処理方式 引数 fga をフィールド fga へセットし，フィールド fga のフィールド ast をフィールド ast へセットする．

  エラー処理 なし

(4) メソッド

- public void analysis()

機能概要 インターフェース FlowGraphAnalyzer の実装

機能説明 フローグラフ解析の 1 つである不動点検出 (fixed point detection) を行う．

入力データ なし

出力データ なし

処理方式 フィールド ast に対してメソッド fixedPointDetector() を呼ぶ．

エラー処理 なし

4.4.13 OpenJIT.frontend.flowgraph.ClassHierarchyAnalyzer クラス

public clas ClassHierarchyAnalyzer

(1) 概要

フローグラフ解析と共にクラス階層解析を行い，その解析結果を保持する．

(2) フィールド (変数)

- FlowGraphAnalysis fga
  解析元の FlowGraphAnalysis オブジェクトを保持する．
- Node ast
  AST
- Class[] cinfo
  クラス階層情報
- CHInfo chg
  クラス階層グラフ

(3) コンストラクタ

- ClassHierarchyAnalyzer(FlowGraphAnalysis fga)

  機能概要 クラス階層解析を行うための初期化を行う．

  機能説明 クラス階層解析を行うためのデータ等の初期化を行う．

  入力データ なし

  出力データ なし

  処理方式 引数 fga をフィールド fga へセットし，フィールド fga のフィールド ast をフィールド ast へセットする．

  エラー処理 なし

(4) メソッド

- public void analysis()

  機能概要 クラス階層グラフを構築する．

  機能説明 フィールドに持つ情報 (クラスファイル間のクラス階層情報・フローグラフ情報) を用いてクラス階層グラフを構築する．

  入力データ なし

  出力データ なし

  処理方式 クラスファイル間のクラス階層情報に含まれる各クラスについて，そのsuperクラスとimplementクラスなどを列挙し，それらのすべてのsuperクラスを親とした木を構築し，得られたクラス関係の情報に基づき順に子のノードを追加することで，クラス階層グラフを構築する．

  エラー処理 なし

4.4.14 OpenJIT.frontend.flowgraph.ASTTransformer クラス

public class ASTTransformer

(1) 概要

フローグラフ解析の結果を元に AST レベルでのプログラム変換を行う．

(2) フィールド (変数)

- FlowGraphAnalysis fga
  フローグラフ解析結果を保持している FlowGraphAnalysis オブジェクト
- Node ast
  AST．フィールド fga のフィールド ast と同じもの．
- Vector src
  プログラム変換ルールの変換前の状態
- Hashtable dst
  プログラム変換ルールの変換後の状態

(3) コンストラクタ

- public ASTTransformer(FlowGraphAnalysis fga)

  機能概要 プログラム変換を初期化する．

  機能説明 フローグラフ解析結果とともにプログラム変換の情報を初期化する．

  入力データ fga(FlowGraphAnalysis fga)

  出力データ なし

  処理方式 入力である fga をフィールド fga へ代入する．また，プログラム変換ルールであるフィールド src， dst を null で初期化する．

  エラー処理 なし

(4) メソッド

- public void registerRule(Node src, Node dst)

  機能概要 プログラム変換ルールを登録する．

  機能説明 AST レベルでのプログラム変換を行うための，変換ルールを登録する．

  入力データ src(Node オブジェクト)，dst(Node オブジェクト)

  出力データ なし

  処理方式 引数 src をフィールド src へ追加し，フィールド dst へ引数 src をキーにして引数 dst をハッシュテーブル dst へ追加する．

  エラー処理 なし

- public Node match(Node target)

  機能概要 プログラム変換にマッチするかどうかを調べる．

  機能説明 入力がプログラム変換のルールを検索しマッチするルールがあれば，そのルールで変換しその結果を返す．マッチしなければ null を返す．

  入力データ target(Node オブジェクト)

  出力データ Node オブジェクト

  処理方式 フィールド src 内から引数 target にあたるものを検索し，見つかれば，それをキーにしてハッシュテーブル dst から変換結果を取り出し，それを返す．見つからなければ null を返す．

  エラー処理 なし

- public Node transform(Node target)

  機能概要 プログラム変換を行う．

  機能説明 引数 target 内の引数 pattern を登録されたルールで変換を行う．

  入力データ target(Node オブジェクト)，pattern(Node オブジェクト)

  出力データ Node オブジェクト

処理方式 まず，引数 target 内に引数 pattern が含まれているかをチェックし，含まれていなければ引数 target 自身を返す．次に引数 pattern にマッチする変換ルールを検索し，マッチするものがなければ引数 target 自身を返す．マッチすれば引数 target 内の引数 pattern の部分をマッチしたルールで変換し，その結果を返す．

エラー処理 なし

4.4.15 OpenJIT.frontend.flowgraph.Optimizer 抽象クラス

public abstract class Optimizer

(1) 概要

このクラスは，Java バイトコードの最適化機能を実現するための枠組みを定義するものである．

(2) フィールド (変数)

- ASTTransformer transformer 最適化で使用されるプログラム変換のルールを保持する．

(3) コンストラクタ

- public Optimizer(ASTTransformer transformer)

  機能概要 最適化モジュールを初期化する．

  機能説明 与えられたプログラム変換のルールをもとに，最適化モジュールを初期化する．

  入力データ transformer(ASTTransformer transformer)

  出力データ なし

  処理方式 入力である transformer で this.transformer を初期設定する．

  エラー処理 なし

(4) メソッド

- public abstract byte[] optimize(byte bytecode[], Node ast, ControlFlowGraph cfg)

  機能概要 プログラムを最適化する．

機能説明 バイトコードとAST及びコントロールフローグラフを用いて，プログラムを最適化し，最適化されたバイトコードを返すという機能を実現するためのインタフェースを定義する．

入力データ bytecode (byte配列), ast (Nodeオブジェクト), cfg (ControlFlowGraphオブジェクト)

出力データ byte配列

処理方式

エラー処理 なし

- protected abstract byte[] generateBytecode()

機能概要 最適化されたバイトコードを出力する．

機能説明 最適化されたバイトコードを出力する機能を実現するためのインタフェースを定義する．

入力データ なし

出力データ byte配列

処理方式

エラー処理 なし

4.5 OpenJIT.frontend.tree パッケージ

- インターフェース

  なし．

- 抽象クラス

  - OpenJIT.frontend.tree.BinaryBitExpression 抽象クラス
  - OpenJIT.frontend.tree.BinaryLogicalExpression 抽象クラス

- クラス

  - OpenJIT.frontend.tree.AddExpression クラス
  - OpenJIT.frontend.tree.AndExpression クラス
  - OpenJIT.frontend.tree.ArrayAccessExpression クラス
  - OpenJIT.frontend.tree.ArrayExpression クラス
  - OpenJIT.frontend.tree.AssignAddExpression クラス
  - OpenJIT.frontend.tree.AssignBitAndExpression クラス
  - OpenJIT.frontend.tree.AssignBitOrExpression クラス
  - OpenJIT.frontend.tree.AssignBitXorExpression クラス
  - OpenJIT.frontend.tree.AssignDivideExpression クラス
  - OpenJIT.frontend.tree.AssignExpression クラス
  - OpenJIT.frontend.tree.AssignMultiplyExpression クラス
  - OpenJIT.frontend.tree.AssignOpExpression クラス
  - OpenJIT.frontend.tree.AssignRemainderExpression クラス
  - OpenJIT.frontend.tree.AssignShiftLeftExpression クラス
  - OpenJIT.frontend.tree.AssignShiftRightExpression クラス
  - OpenJIT.frontend.tree.AssignSubtractExpression クラス
  - OpenJIT.frontend.tree.AssignUnsignedShiftRightExpression クラス

- OpenJIT.frontend.tree.BinaryArithmeticExpression クラス
- OpenJIT.frontend.tree.BinaryAssignExpression クラス
- OpenJIT.frontend.tree.BinaryCompareExpression クラス
- OpenJIT.frontend.tree.BinaryEqualityExpression クラス
- OpenJIT.frontend.tree.BinaryExpression クラス
- OpenJIT.frontend.tree.BinaryShiftExpression クラス
- OpenJIT.frontend.tree.BitAndExpression クラス
- OpenJIT.frontend.tree.BitNotExpression クラス
- OpenJIT.frontend.tree.BitOrExpression クラス
- OpenJIT.frontend.tree.BitXorExpression クラス
- OpenJIT.frontend.tree.BooleanExpression クラス
- OpenJIT.frontend.tree.BreakStatement クラス
- OpenJIT.frontend.tree.ByteExpression クラス
- OpenJIT.frontend.tree.CaseStatement クラス
- OpenJIT.frontend.tree.CastExpression クラス
- OpenJIT.frontend.tree.CatchStatement クラス
- OpenJIT.frontend.tree.CharExpression クラス
- OpenJIT.frontend.tree.CheckContext クラス
- OpenJIT.frontend.tree.CodeContext クラス
- OpenJIT.frontend.tree.CommaExpression クラス
- OpenJIT.frontend.tree.CompoundStatement クラス
- OpenJIT.frontend.tree.ConditionVars クラス
- OpenJIT.frontend.tree.ConditionalExpression クラス
- OpenJIT.frontend.tree.ConstantExpression クラス
- OpenJIT.frontend.tree.Context クラス
- OpenJIT.frontend.tree.ContinueStatement クラス

- OpenJIT.frontend.tree.ConvertExpression クラス
- OpenJIT.frontend.tree.DeclarationStatement クラス
- OpenJIT.frontend.tree.DivRemExpression クラス
- OpenJIT.frontend.tree.DivideExpression クラス
- OpenJIT.frontend.tree.DoStatement クラス
- OpenJIT.frontend.tree.DoubleExpression クラス
- OpenJIT.frontend.tree.EqualExpression クラス
- OpenJIT.frontend.tree.ExprExpression クラス
- OpenJIT.frontend.tree.Expression クラス
- OpenJIT.frontend.tree.ExpressionStatement クラス
- OpenJIT.frontend.tree.FieldExpression クラス
- OpenJIT.frontend.tree.FinallyStatement クラス
- OpenJIT.frontend.tree.FloatExpression クラス
- OpenJIT.frontend.tree.ForStatement クラス
- OpenJIT.frontend.tree.GreaterExpression クラス
- OpenJIT.frontend.tree.GreaterOrEqualExpression クラス
- OpenJIT.frontend.tree.IdentifierExpression クラス
- OpenJIT.frontend.tree.IfStatement クラス
- OpenJIT.frontend.tree.IncDecExpression クラス
- OpenJIT.frontend.tree.InlineMethodExpression クラス
- OpenJIT.frontend.tree.InlineNewInstanceExpression クラス
- OpenJIT.frontend.tree.InlineReturnStatement クラス
- OpenJIT.frontend.tree.InstanceOfExpression クラス
- OpenJIT.frontend.tree.IntExpression クラス
- OpenJIT.frontend.tree.IntegerExpression クラス
- OpenJIT.frontend.tree.LengthExpression クラス

- OpenJIT.frontend.tree.LessExpression クラス
- OpenJIT.frontend.tree.LessOrEqualExpression クラス
- OpenJIT.frontend.tree.LocalField クラス
- OpenJIT.frontend.tree.LongExpression クラス
- OpenJIT.frontend.tree.MethodExpression クラス
- OpenJIT.frontend.tree.MultiplyExpression クラス
- OpenJIT.frontend.tree.NaryExpression クラス
- OpenJIT.frontend.tree.NegativeExpression クラス
- OpenJIT.frontend.tree.NewArrayExpression クラス
- OpenJIT.frontend.tree.NewInstanceExpression クラス
- OpenJIT.frontend.tree.Node クラス
- OpenJIT.frontend.tree.NotEqualExpression クラス
- OpenJIT.frontend.tree.NotExpression クラス
- OpenJIT.frontend.tree.NullExpression クラス
- OpenJIT.frontend.tree.OrExpression クラス
- OpenJIT.frontend.tree.PositiveExpression クラス
- OpenJIT.frontend.tree.PostDecExpression クラス
- OpenJIT.frontend.tree.PostIncExpression クラス
- OpenJIT.frontend.tree.PreDecExpression クラス
- OpenJIT.frontend.tree.PreIncExpression クラス
- OpenJIT.frontend.tree.RemainderExpression クラス
- OpenJIT.frontend.tree.ReturnStatement クラス
- OpenJIT.frontend.tree.ShiftLeftExpression クラス
- OpenJIT.frontend.tree.ShiftRightExpression クラス
- OpenJIT.frontend.tree.ShortExpression クラス
- OpenJIT.frontend.tree.Statement クラス

- OpenJIT.frontend.tree.StringExpression クラス
- OpenJIT.frontend.tree.SubtractExpression クラス
- OpenJIT.frontend.tree.SuperExpression クラス
- OpenJIT.frontend.tree.SwitchStatement クラス
- OpenJIT.frontend.tree.SynchronizedStatement クラス
- OpenJIT.frontend.tree.ThisExpression クラス
- OpenJIT.frontend.tree.ThrowStatement クラス
- OpenJIT.frontend.tree.TryStatement クラス
- OpenJIT.frontend.tree.TypeExpression クラス
- OpenJIT.frontend.tree.UnaryExpression クラス
- OpenJIT.frontend.tree.UnsignedShiftRightExpression クラス
- OpenJIT.frontend.tree.VarDeclarationStatement クラス
- OpenJIT.frontend.tree.WhileStatement クラス

- 例外

  なし．

4.5.1 OpenJIT.frontend.tree.BinaryBitExpression 抽象クラス

abstract public class BinaryBitExpression extends BinaryExpression

(1) 概要

このクラスは，BitAndExpression, BitOrExpression, BitXorExpression クラスのスーパークラスであり，ビット演算に共通するコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- BinaryBitExpression(int op, int where, Expression left, Expression right)

  機能概要 BinaryBitExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ op (演算子の種類), where (行番号)，left (式の左辺式の Expression オブジェクト)，right (式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, left.type, left, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 式の型を選択する．

  機能説明 フィールド type (スーパークラス BinaryExpression のフィールドを継承する) を設定し，型変換を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

  出力データ なし．

  処理方式 tm で指定された型をこの式の型に設定し，左辺式 left，右辺値 right の型をそれぞれこの型に変換する．

  エラー処理 特になし．

- public void checkCondition(Environment env, Context ctx, long vset, Hashtable exp, ConditionVars cvars)

  機能概要 式の条件を検査する．

機能説明 式の右辺式，左辺式の評価を行い，それぞれの式をBooleanExpressionに変換する．

入力データ env (環境を格納するEnvironmentオブジェクト), ctx (コンテクストを格納するContextオブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), cvars (条件の評価値オブジェクト)

出力データ なし．

処理方式 左辺式のtrue/falseを検査して，その左辺式をBooleanExpressionに変換する．次に右辺式のtrue/falseを検査して，右辺値をBooleanExpressionに変換する．左辺式，右辺式の検査結果をvsTrue, vsFalseに格納する．最後に，setBitConditionメソッドを引数vsTrue, vsFalseで呼び出して，条件の評価値オブジェクトcvarsを更新する．setBitConditionメソッドは子クラスで定義される．

エラー処理 特になし．

- abstract void setBitCondition(long vsTrue, long vsFalse, ConditionVars cvars)

  機能概要 ビット演算に用いる条件の評価値を設定する．

  機能説明 vsTrue, vsFalseに基づいてcvarsを適切に設定する．

  入力データ vsTrue (左辺式の評価後のvariable set), vsFalse (左辺式の評価後のvariable set), cvars (右辺式の評価後のvariable set)

  出力データ なし．

  処理方式 サブクラスで定義される．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 ビット式の計算をするバイトコードを生成する．

  機能説明 ビット式の計算をするバイトコードを生成する．

  入力データ env (環境を格納するEnvironmentオブジェクト), ctx (コンテクストを格納するContextオブジェクト), asm (コード生成を行うAssemblerオブジェクト)

出力データ なし．

処理方式 left.checkValue を呼び出して左辺式をスタック上に積むコードを生成し，right.checkValue を呼び出して右辺式をスタック上に積むコードを生成する．その後，codeOperation メソッドを呼び出して，スタック上のデータでビット演算を行うコードを生成する．

エラー処理 特になし．

4.5.2 OpenJIT.frontend.tree.BinaryLogicalExpression 抽象クラス

public class BinaryLogicalExpression extends BinaryExpression

(1) 概要

このクラスは，AndExpression, OrExpression クラスのスーパークラスであり，これらに共通するコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public BinaryLogicalExpression(int op, int where, Expression left, Expression right)

  機能概要 BinaryLogicalExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ op (演算子の種類), where (行番号)，left (式の左辺値の Expression オブジェクト)，right (式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, Type.Boolean (定数), left, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 2 項論理演算の式の検査を行う．

機能説明 2 項論理演算の式の検査を行い，条件の評価値を更新する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 ConditionVars オブジェクト cvars を生成する．論理式を評価して，戻り値の真偽値によって cvars を設定するために，checkCondition メソッドを呼び出す．cvars.vsTrue, cvars.vsFalse の論理積を返す．

エラー処理 特になし．

- abstract public void checkCondition(Environment env, Context ctx, long vset, Hashtable exp, ConditionVars cvars)

機能概要 式の条件を検査する．

機能説明 式の条件を検査する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), cvars (条件の評価値オブジェクト)

出力データ なし．

処理方式 サブクラスで定義される．

エラー処理 特になし．

- public Expression inline(Environment env, Context ctx)

機能概要 式の inlining を行う．

機能説明 左辺式，右辺式の inlining を行った BinaryLogicalExpression を返す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 left.inline メソッドを呼び，左辺式の inlining を行って，left に代入する．right.inline メソッドを呼び，右辺式の inlining を行って，right に代入する．this を返す．

エラー処理 特になし．

4.5.3 OpenJIT.frontend.tree.AddExpression クラス

public class AddExpression extends BinaryArithmeticExpression

(1) 概要

このクラスは，AST の＋式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AddExpression(int where, Expression left, Expression right)

  機能概要 AST の＋式ノードを生成する．

  機能説明 スーパークラスの BinaryArithmeticExpression クラスで定義されているコンストラクタを用いて，＋式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (＋式の左辺式の Expression オブジェクト)，right (＋式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ADD(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 ＋式の型を選択する．

  機能説明 フィールド type (スーパークラス BinaryArithmeticExpression のフィールドを継承する) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 左辺式の型が String 型で，かつ右辺式の型が void 型でない場合，および右辺式の型が String 型で，かつ左辺式の型が void 型でない場合，式の型を String 型にする．そうでない場合，super.selectType メソッドを呼び出す．

エラー処理 特になし．

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数の加算を行い，IntExpression オブジェクトを生成する．

  入力データ a (int 型), b (int 型)

  出力データ Expression オブジェクト．

  処理内容 a+b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数の加算を行い，LongExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a+b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(float a, float b)

機能概要 式を評価する．引数がともに float 型の場合．

機能説明 float 型の引数の加算を行い，FloatExpression オブジェクトを生成する．

入力データ a (float 型), b (float 型)

出力データ Expression オブジェクト．

処理内容 a+b を計算し，FloatExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(double a, double b)

機能概要 式を評価する．引数がともに double 型の場合．

機能説明 double 型の引数の加算を行い，DoubleExpression オブジェクトを生成する．

入力データ a (double 型), b (double 型)

出力データ Expression オブジェクト．

処理内容 a+b を計算し，DoubleExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(String a, String b)

機能概要 式を評価する．引数がともに String 型の場合．

機能説明 String 型の引数の join を行い，StringExpression オブジェクトを生成する．

入力データ a (String 型), b (String 型)

出力データ Expression オブジェクト．

処理内容 a+b を計算し，StringExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 右辺式または左辺式が 0 の場合，式を単純化する．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式が 0 の場合，右辺式をこの + 式の値として返す．右辺値が 0 の場合，左辺式をこの + 式の値として返す．それ以外の場合， this オブジェクトを返す．

  エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 Inlining のコストを計算する．

  機能説明 + 式，右辺式，左辺式，のそれぞれの inlining コストを加算して + 式全体の inlining コストを返す．

  入力データ thresh (閾値)

  出力データ inlining のコスト．

  処理方式 + 式の型が基本型の場合， 1，そうでない場合， 12 を + 式のコストとする．この値と， left.costInline(thresh)， right.costInline(thresh) の戻り値を加算して返す．

  エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 + を行うバイトコードを生成する．

  機能説明 各型ごとに適切な + 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_iadd (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- void codeAppend(Environment env, Context ctx, Assembler asm, ClassDeclaration c)

  機能概要 バイトコードを追加する．

  機能説明 + 式をバイトコードにする際，補助的に必要なコード生成を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), c (ClassDeclaration オブジェクト)

  出力データ なし．

  処理方式 + 式の型が String 型の場合，左辺式と右辺式に関してそれぞれ codeAppend メソッドを呼び出す．そうでない場合，super.codeAppend メソッドを呼び出す．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm, ClassDeclaration c)

  機能概要 + 式の値を計算するバイトコードを生成する．

  機能説明 + 式の値を計算するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), c (ClassDeclaration オブジェクト)

  出力データ なし．

  処理方式 + 式の型が String 型の場合，java.lang.StringBuffer クラスの toString メソッドを呼び出すことで式の評価を行う．具体的には，java.lang.StringBuffer

クラスについて opc_new し，opc_dup し，toString メソッドについて invokevirtual するバイトコードを生成する．String 型でない場合は，super.codeValue メソッドを呼び出す．

エラー処理 java.lang.StringBuffer クラスが見つからない場合，および toString メソッドが見つからない場合，コンパイルエラーを表示する．

- public String toPrettyString()

  機能概要 ＋式の中身を出力する．

  機能説明 ＋式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.4 OpenJIT.frontend.tree.AndExpression クラス

public class AndExpression extends BinaryLogicalExpression

(1) 概要

このクラスは，AST の AND 演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AndExpression(int where, Expression left, Expression right)

  機能概要 AST の And 演算子のノードを生成する．

  機能説明 スーパークラスの BinaryLogicalExpression クラスで定義されているコンストラクタを用いて，AND 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (AND 式の左辺式の Expression オブジェクト)，right (AND 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 AND(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- public void checkCondition(Environment env, Context ctx, long vset, Hashtable exp, ConditionVars cvars)

  機能概要 AND 式の条件を検査する．

  機能説明 AND 式の右辺式，左辺式の評価を行い，それぞれの式を BooleanExpression に変換する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), cvars (条件の評価値オブジェクト)

出力データ なし．

処理方式 左辺式の true/false を検査して，左辺式を BooleanExpression に変換する．次に右辺式の true/false を検査して，右辺値を BooleanExpression に変換する． cvars には，条件が静的に評価可能な場合についてのみ，値がセットされる．

エラー処理 特になし．

- Expression eval(boolean a, boolean b)

  機能概要 式を評価する．引数がともに boolean 型の場合．

  機能説明 boolean 型の引数の論理積を行い， BooleanExpression オブジェクトを生成する．

  入力データ a (boolean 型), b (boolean 型)

  出力データ Expression オブジェクト．

  処理内容 a&&b を計算し， BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 左辺式または右辺式が true の場合，式を単純化する．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式が true の場合，右辺式をこの AND 式の値として返す．右辺式が true の場合，左辺式をこの AND 式の値として返す．それ以外の場合，this オブジェクトを返す．

エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

  機能概要 AND 式を実行するバイトコードを生成する．

  機能説明 AND 式を左辺式，右辺式の条件の成立に従って分岐するバイトコードに変換する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

  出力データ なし．

  処理方式 whenTrue が True の場合，新しいラベル lbl2 を生成し，left.codeBranch(env, ctx, asm, lbl2, false), right.codeBranch(env, ctx, asm, lbl, true) を実行した後，ラベル lbl2 を挿入する．whenTrue が False の場合，left.codeBranch(env, ctx, asm, lbl, false), right.codeBranch(env, asm, lbl, false) を実行する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 AND 式の中身を出力する．

  機能説明 AND 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.5 OpenJIT.frontend.tree.ArrayAccessExpression クラス

public class ArrayAccessExpression extends UnaryExpression

(1) 概要

このクラスは，AST の配列アクセス式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- public Expression index
  配列のインデックスを表す式を格納する．

(3) コンストラクタ

- public ArrayAccessExpression(int where, Expression right, Expression index)

  機能概要 AST の [] 式ノードを生成する．

  機能説明 スーパークラスの UnaryExpression で定義されているコンストラクタを用いて，[] 式ノードの生成・初期化を行う．

  入力データ where (行番号), right (配列を表す Expression オブジェクト), index (配列の index を表す Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ARRAYACCESS(定数), where, Type.tError(定数), right で呼び出す．this.index を index で初期化する．

  エラー処理 特になし．

(4) メソッド

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の型を検査する．

機能説明 right が配列型，index が int 型であるかどうか検査する．index は int 型に変換する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 right.checkValue メソッドを呼び出し，right の値を検査する．index.checkValue メソッドを呼び出し，index の値を検査する．次に，index の型が int 型になるように convert メソッドを呼び出す．right の要素の型を得て，type に代入する．index についての条件の評価値を戻り値として返す．

エラー処理 index が null の場合，array.index.required のエラーを出力する．right の型が配列型でない場合，not.array のエラーを出力する．

- public long checkLHS(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 配列式が代入式の左辺にきているかどうかを検査する．

  機能説明 配列式が代入式の左辺にきているかどうかを検査して，条件変数として返す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値．

  処理方式 checkValue(env, ctx, vset, exp) を呼び出す．

  エラー処理 特になし．

- public long checkAssignOp(Environment env, Context ctx, long vset, Hashtable exp, Expression outside)

  機能概要 配列式が += などの代入演算式の左辺にきているかどうかを検査する．

機能説明 配列式が += などの代入演算式の左辺にきているかどうかを検査して，条件の評価値として返す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), outside (右辺式の Expression)

出力データ 条件の評価値．

処理方式 checkValue(env, ctx, vset, exp) を呼び出す．

エラー処理 特になし．

- Type toType(Environment env, Context ctx)

機能概要 配列のエレメントタイプを設定する．

機能説明 右辺のエレメントタイプを求め，それを配列のエレメントタイプとする．このメソッドは再帰的に機能して，最内のエレメントタイプが設定される．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ 型．

処理方式 toType(env, right.toType(env, ctx)) メソッドを呼び出して，その型を返す．

エラー処理 特になし．

- Type toType(Environment env, Type t)

機能概要 配列のエレメントタイプを設定する．

機能説明 配列のエレメントタイプを与えられた型に設定する．

入力データ env (環境を格納する Environment オブジェクト), t (設定する型)

出力データ 型オブジェクト．

処理方式 指定された型をエレメントタイプとする配列型を返す．

エラー処理 特になし．

- public Expression inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 配列式を CommaExpression として inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 配列式を index の評価式と，配列式に分解して CommaExpression を生成する．CommaExpression の inlining を行って，その結果の式を返す．

  エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 右辺の式，インデックスの式の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 right.inlineValue メソッドを呼び出し，右辺式を inlining した結果を right に格納する．index.inlineValue メソッドを呼び出し，インデックスを inlining した結果を index に格納する．その後， this を返す．

  エラー処理 特になし．

- public Expression inlineLHS(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 配列が左辺式として現れたときの inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

処理方式 inlineValue(env, ctx) を呼び出す．

エラー処理 特になし．

- public Expression copyInline(Context ctx)

  機能概要 配列式の inlining されたコピーを作る．

  機能説明 配列式のクローンオブジェクトを作り，その right, index について inlining を行う．

  入力データ ctx (コンテクストを格納する Context オブジェクト)

  出力データ 配列式のオブジェクト．

  処理方式 clone メソッドを呼び出し，ArrayAccessExpression オブジェクト e を作る．right.copyInline メソッドを呼び出し，その結果の式を e.right に格納し，index.copyInline メソッドを呼び出し，その結果の式を e.index に格納する．その後，e を返す．

  エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 inlining のコストを求める．

  機能説明 right, index の inlining のコストを求め，配列式全体の inlining コストを求める．

  入力データ thresh (閾値)

  出力データ inlining のコスト．

  処理方式 right.costInline, index.costInline メソッドを呼び出し，right, index の inlining コストを求める．この 2 つの値と 1 を加えた値を返す．

  エラー処理 特になし．

- int codeLValue(Environment env, Context ctx, Assembler asm)

  機能概要 配列式が左辺式として使われる場合のバイトコードを生成する．

  機能説明 right, index をスタック上に積む，バイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ 消費したスタック量．

処理方式 right.codeValue, index.codeValue を呼び出し，それぞれの値が，スタックに積まれるようにバイトコードを生成する．スタック消費量として 2 を返す．

エラー処理 特になし．

- void codeLoad(Environment env, Context ctx, Assembler asm)

機能概要 配列式が右辺式として使われ，ロードするバイトコードを生成する．

機能説明 エレメントタイプごとに適切なバイトコードを選択して，配列式をスタック上にロードするバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 エレメントタイプが，boolean, byte 型のとき，opc_baload，char 型のとき，opc_caload，short 型のとき，opc_saload，それ以外の型のとき，opc_iaload をオペコードとするバイトコードを追加する．

エラー処理 特になし．

- void codeStore(Environment env, Context ctx, Assembler asm)

機能概要 配列式が左辺式として使われ，ストアするバイトコードを生成する．

機能説明 エレメントタイプごとに適切なバイトコードを選択して，スタックトップの値を配列式にストアするバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 エレメントタイプが，boolean, byte 型のとき，opc_bastore，char 型のとき，opc_castore，short 型のとき，opc_sastore，それ以外の型のとき，opc_iastore をオペコードとするバイトコードを追加する．

エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 配列式が左辺式として使われる場合のバイトコードを生成する．

  機能説明 配列式が左辺式として使われる場合のバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 codeLVvalue メソッドを呼び出し，codeLoad メソッドを呼び出す．

  エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 配列式の中身を出力する．

  機能説明 配列式を読み易く出力する．

  入力データ out (String の出力先の PrintStream)

  出力データ なし．

  処理方式 配列式の中身を PrintStream out に出力する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 配列式の中身を出力する．

  機能説明 配列式を読み易く出力する．

  入力データ なし．

出力データ String 型の文字列．

処理方式 toPrettyString メソッドで right, index をそれぞれ文字列に変換し，“right[index]” という形式の文字列を生成して，返す．

エラー処理 特になし．

4.5.6 OpenJIT.frontend.tree.ArrayExpression クラス

public class ArrayExpression extends NaryExpression

(1) 概要

このクラスは，AST の配列のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public ArrayExpression(int where, Expression args[])

  機能概要 ArrayExpression オブジェクトを生成する．

  機能説明 スーパークラスの NaryExpression のコンストラクタを用いて，ArrayExpression オブジェクトを生成する．

  入力データ where (行番号), args[] (配列オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ARRAY (定数), where, Type.tError (定数), null, args で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の型を検査する．

  機能説明 invalid.array.expr のエラーを出力する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 無条件で invalid.array.expr のエラーを出力し，vset を返す．

エラー処理 なし．

- public long checkInitializer(Environment env, Context ctx, long vset, Type t, Hashtable exp)

  機能概要 配列の初期化を行うように検査する．

  機能説明 配列の初期化を行い，多重配列の場合は再帰的に checkInitializer を呼び出す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), t (エレメントタイプ), exp (ハッシュテーブル)

  出力データ 条件の評価値．

  処理方式 type に t を代入する．t のエレメントタイプを得て，t に格納する．args の各エレメントに関して checkInitializer メソッドを呼び出し，t の型に args の各エレメントの式の型を変換する．その後，vset を返す．

  エラー処理 t で与えられた型が配列型でない場合，invalid.array.init のエラーを出力する．

- public Expression inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 args の各エレメントごとに inlining を行い，args の各エレメントに格納する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

処理方式 argsの各エレメントごとにinlineメソッドを呼び出し，inliningを行った結果をargsの各エレメントに格納する．inliningの結果がnullでない場合は，CommaExpressionオブジェクトを生成して，inliningの結果を格納する．このCommaExpressionオブジェクトにはエレメントごとにinliningを行った式の列が格納される．その後，このCommaExpressionオブジェクトを返す．

エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 式のinliningを行う．

  機能説明 argsの各エレメントごとにinliningを行い，argsの各エレメントに格納する．

  入力データ env (環境を格納するEnvironmentオブジェクト), ctx (コンテクストを格納するContextオブジェクト)

  出力データ inlining後の式．

  処理方式 argsの各エレメントごとにinliningを行い，argsの各エレメントに格納する．その後，thisを返す．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 配列式をローカル変数にロードするバイトコードを生成する．

  機能説明 配列のエレメントタイプに従って，適切なコードを選択して，配列式をローカル変数にロードするバイトコードを生成する．

  入力データ env (環境を格納するEnvironmentオブジェクト), ctx (コンテクストを格納するContextオブジェクト), asm (コード生成を行うAssemblerオブジェクト)

  出力データ なし．

  処理方式 args.lengthをスタックに積むために，opc_ldcをオペコードとするバイトコードを生成する．次に，argsのエレメントタイプを得て，基本型の

場合には，opc_newarray をオペコードとするバイトコードを生成する．エレメントタイプが，クラスや配列の場合には，opc_anewarray をオペコードとするバイトコードを生成する．

その後，配列の内容をストアするために，args の各要素をストアするためのバイトコードを生成する．まず，opc_dup をオペコードとするバイトコードを生成し，要素のインデックスをコンスタントプールからロードしてスタックに積む．次に codeValue メソッドを呼び出して，要素の値をスタック上に積む．その後，エレメントタイプを得て，型が boolean, byte 型の場合は opc_bastore，char 型の場合は opc_castore，short 型の場合は opc_sastore，int 型の場合は opc_istore をオペコードとするバイトコードを生成して，ローカル変数に値を格納する．

エラー処理 特になし．

- public String toPrettyString()

機能概要 配列式の中身を出力する．

機能説明 配列式を読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 toPrettyString メソッドで “{ args }” という形式の文字列を生成して，返す．

エラー処理 特になし．

4.5.7 OpenJIT.frontend.tree.AssignAddExpression クラス

public class AssignAddExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の += 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignAddExpression(int where, Expression left, Expression right)

  機能概要 AST の += 式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，+= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (+= 式の左辺式の Expression オブジェクト)，right (+= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGADD(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 += 式の型を選択する．

  機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承する) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 左辺式の型が String 型の場合，この式の型を String 型にする．それ以外の場合，この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- public int costInline(int thresh)

機能概要 Inlining のコストを計算する．

機能説明 += 式全体の inlining コストを返す．

入力データ thresh (閾値)

出力データ inlining のコスト．

処理方式 += 式の型が基本型の場合，super.costInline(thresh) の戻り値を返す．そうでない場合，25 を返す．

エラー処理 特になし．

- void code(Environment env, Context ctx, Assembler asm, boolean valNeeded)

機能概要 += 式の値を計算するバイトコードを生成する．

機能説明 += 式の値を計算するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), valNeeded (計算後の値をスタック上に残すかどうかを指定するフラグ)

出力データ なし．

処理方式 + 式の型が String 型の場合，java.lang.StringBuffer クラスの toString メソッドを呼び出すことで式の評価を行う．具体的には，java.lang.StringBuffer クラスについて opc_new し，opc_dup し，toString メソッドについて invokevirtual するバイトコードを生成する．さらに，valNeeded が True の場

合には，スーパークラスの codeDup メソッドを呼び出す．String 型でない場合は，super.code メソッドを呼び出す．

エラー処理 java.lang.StringBuffer クラスが見つからない場合，および toString メソッドが見つからない場合，コンパイルエラーを表示する．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 += を行うバイトコードを生成する．

  機能説明 各型ごとに適切な += 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 opc_iadd (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 += 式の中身を出力する．

  機能説明 += 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.8 OpenJIT.frontend.tree.AssignBitAndExpression クラス

public class AssignBitOrExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の &= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignBitAndExpression(int where, Expression left, Expression right)

  機能概要 AST の &= 式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，&= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (&= 式の左辺式の Expression オブジェクト)，right (&= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGBITAND(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 &= 式の型を選択する．

  機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 &= を行うバイトコードを生成する．

  機能説明 各型ごとに適切な &= 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 opc_iand (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 &= 式の中身を出力する．

  機能説明 &= 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.9 OpenJIT.frontend.tree.AssignBitOrExpression クラス

public class AssignBitOrExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の |= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignBitOrExpression(int where, Expression left, Expression right)

  機能概要 AST の |= 式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，|= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (|= 式の左辺式の Expression オブジェクト)，right (|= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGBITOR(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 |= 式の型を選択する．

  機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 |= を行うバイトコードを生成する．

  機能説明 各型ごとに適切な |= 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 opc_ior (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 |= 式の中身を出力する．

  機能説明 |= 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.10 OpenJIT.frontend.tree.AssignBitXorExpression クラス

public class AssignBitXorExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の^=式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignBitXorExpression(int where, Expression left, Expression right)

  機能概要 AST の^=式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，^=式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (^=式の左辺式の Expression オブジェクト)，right (^=式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGBITXOR(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 ^=式の型を選択する．

  機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 ^= を行うバイトコードを生成する．

  機能説明 各型ごとに適切な^= 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 opc_ixor (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 ^= 式の中身を出力する．

  機能説明 ^= 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.11 OpenJIT.frontend.tree.AssignDivideExpression クラス

public class AssignDivideExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の /= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignDivideExpression(int where, Expression left, Expression right)

  機能概要 AST の /= 式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，/= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (/= 式の左辺式の Expression オブジェクト)，right (/= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGDIV(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 /= 式の型を選択する．

  機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 /= を行うバイトコードを生成する．

  機能説明 各型ごとに適切な /= 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 opc_idiv (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 /= 式の中身を出力する．

  機能説明 /= 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.12 OpenJIT.frontend.tree.AssignExpression クラス

public class AssignExpression extends BinaryAssignExpression

(1) 概要

このクラスは，AST の＝式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignExpression(int where, Expression left, Expression right)

  機能概要 AST の＝式ノードを生成する．

  機能説明 スーパークラスの BinaryAssignExpression クラスで定義されているコンストラクタを用いて，＝式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (＝式の左辺式の Expression オブジェクト)，right (＝式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASSIGN(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 値の検査を行う．

  機能説明 右辺式を検査し，左辺式を LHS として検査する．また，右辺式の型を左辺式の型に変換する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 right.checkValue メソッドを呼び出し，戻り値を vset に格納する．left.checkLHS メソッドを呼び出し，戻り値を vset に格納する．次に，left.type を type に格納する．右辺式を type の型に変換するために，convert メソッドを呼び出し，戻り値を right に格納する．最後に vset を返す．

エラー処理 特になし．

- public int costInline(int thresh)

機能概要 Inlining のコストを計算する．

機能説明 = 式全体の inlining コストを返す．

入力データ thresh (閾値)

出力データ inlining のコスト．

処理方式 この式のコスト 2 と super.costInline(thresh) の戻り値を加算して返す．

エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

機能概要 = 式のバイトコードを生成する．

機能説明 右辺式の値をロードし，左辺式のスタック位置にストアする．スタックトップに右辺式を残す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 left.codeLValue メソッドを呼び出し，左辺式の格納されるスタックを確保し，その深さを depth に保存する．right.codeValue メソッドを呼び出し

て，スタックトップに右辺式を置き，codeDup メソッドを呼び出して，スタックトップの値をさらにスタックトップに積む．その後，left.codeStore を呼び出して，スタックトップの値をスタックの depth 位置にコピーする．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 = 式のバイトコードを生成する．

  機能説明 右辺式の値をロードし，左辺式のスタック位置にストアする．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 left.codeLValue メソッドを呼び出し，左辺式の格納されるスタックを確保する．right.codeValue メソッドを呼び出して，スタックトップに右辺式を置き，left.codeStore を呼び出して，スタックトップの値を左辺式の位置にコピーする．

  エラー処理 特になし．

4.5.13 OpenJIT.frontend.tree.AssignMultiplyExpression クラス

public class AssignMultiplyExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の *= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignMultiplyExpression(int where, Expression left, Expression right)

  機能概要 AST の *= 式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，*= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (*= 式の左辺式の Expression オブジェクト)，right (*= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGMUL(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 *= 式の型を選択する．

機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

機能概要 *= を行うバイトコードを生成する．

機能説明 各型ごとに適切な *= 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_imul (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 *= 式の中身を出力する．

機能説明 *= 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.14 OpenJIT.frontend.tree.AssignOpExpression クラス

public class AssignOpExpression extends BinaryAssignExpression

(1) 概要

このクラスは，AST の +=, &=, |=, ^=, /=, *=, %=, <<=, >>=, -=, >>>= 式のノードのスーパークラスであり，これらに共通のコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignOpExpression(int op, int where, Expression left, Expression right)

  機能概要 AssignOpExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryAssignExpression クラスで定義されているコンストラクタを用いて，AssignOpExpression オブジェクトを生成する．

  入力データ op (演算子の種類), where (行番号), left (左辺式の Expression), right (右辺式の Expression)

  出力データ なし．

  処理方式 super(op, where, left, right) を呼び出す．

  エラー処理 特になし．

(4) メソッド

- int getIncrement()

  機能概要 +=, -= の場合に，増加量を求める．

  機能説明 += の場合，右辺式，-= の場合，右辺式の符号反転した値を増加量として返す．

入力データ なし．

出力データ 増加量．

処理方式 左辺式の op が IDENT (定数) に等しく，式の型が int 型で，右辺式の op が INTVAL (定数) であり，かつ left がローカル変数であれば，op が ASGADD に等しければ，右辺の値，ASGSUB に等しければ，右辺の値の符号反転した値を返す．それ以外の場合は NOINC (Integer.MAX_VALUE に等しい定数) を返す．

エラー処理 特になし．

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 値の検査を行う．

  機能説明 右辺式を検査し，左辺式を LHS として検査する．また，右辺式の型を左辺式の型に変換する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値．

  処理方式 left.checkAssignOp メソッドを呼び出し，戻り値を vset に格納する．right.checkValue メソッドを呼び出し，戻り値を vset に格納する．left.type.getTypeMask(), right.type.getTypeMask() から，式の type を選択し，左辺式をこのタイプに変換するために，convert メソッドを呼び出す．最後に vset を返す．

  エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 左辺式，右辺式をそれぞれ inlining した式を返す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 left.inlineValue を呼び出し，その結果を left に格納する．right.inlineValue を呼び出し，その結果を right に格納する．その後， this を返す．

エラー処理 特になし．

- public Expression copyInline(Context ctx)

機能概要 配列式の inlining されたコピーを作る．

機能説明 配列式のクローンオブジェクトを作り，その left, right について inlining を行う．

入力データ ctx (コンテクストを格納する Context オブジェクト)

出力データ 配列式のオブジェクト．

処理方式 clone メソッドを呼び出し， AssignOpExpression オブジェクト e を作る．left.copyInline メソッドを呼び出し，その結果の式を e.left に格納し，right.copyInline メソッドを呼び出し，その結果の式を e.right に格納する．その後， e を返す．

エラー処理 特になし．

- public int costInline(int thresh)

機能概要 inlining のコストを求める．

機能説明 right, index の inlining のコストを求め，式全体の inlining コストを求める．

入力データ thresh (閾値)

出力データ inlining のコスト．

処理方式 getIncrement() を呼び出し，戻り値が NOINC(定数) でなければ 2，そうであれば， 3+super.costInline(thresh) を inlining コストとして返す．

エラー処理 特になし．

- void code(Environment env, Context ctx, Assembler asm, boolean valNeeded)

機能概要 式を実行するバイトコードを生成する．

機能説明 右辺と左辺の 2 項演算を行ってから，左辺に結果を格納するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), valNeeded (計算後の値をスタック上に残すか否かを決定する真偽値)

出力データ なし．

処理方式 getIncrement() を呼び出し，val に格納する．val が NOINC でない場合，左辺式の格納されているローカル変数番号と，val について，op_iinc をオペコードとするバイトコードを生成する．valNeeded フラグが立っていれば，left.codeValue メソッドを呼び出し，left の値をスタックトップに積んで終了．立っていなければそのまま終了．

val が NOINC の場合，left.codeLValue を呼び出して左辺式の領域をスタック上に取り，codeDup を呼び出して左辺式と同じ大きさの領域をスタック上に取る．次に，left.codeLoad を呼び出して left の値でスタックトップを書き換え，codeConversion メソッドを呼び出してそのスタックトップの型が left.type になるように型変換するバイトコードを生成する．さらに，right.codeValue メソッドを呼び出して右辺値をスタックトップに積み，codeOperation メソッドを呼び出して，スタックの先頭に積まれている 2 つの値で演算を行うバイトコードを生成する．計算結果を left の型に変換するために codeConversion メソッドを呼び出す．valNeeded フラグが立っていれば，codeDup を行って，計算結果のスタック領域をコピーする．最後に，left.codeStore を呼び出して，計算結果のスタック領域をローカル変数領域に書き出す．

エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

機能概要 式が左辺式として使われる場合のバイトコードを生成する．

機能説明 式が左辺式として使われる場合のバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 code(env, ctx, asm, true) を呼び出す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 式が右辺式として使われる場合のバイトコードを生成する．

  機能説明 式が右辺式として使われる場合のバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 code(env, ctx, asm, false) を呼び出す．

  エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 式の中身を出力する．

  機能説明 式を読み易く出力する．

  入力データ out (String の出力先の PrintStream)

  出力データ なし．

  処理方式 式の中身を “(op left right)” の形式で PrintStream out に出力する．

  エラー処理 特になし．

4.5.15 OpenJIT.frontend.tree.AssignRemainderExpression クラス

public class AssignRemainderExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の %= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignRemainderExpression(int where, Expression left, Expression right)

  機能概要 AST の %= 式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，%= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (%= 式の左辺式の Expression オブジェクト)，right (%= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGREM(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 %= 式の型を選択する．

機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 %= を行うバイトコードを生成する．

  機能説明 各型ごとに適切な %= 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 opc_irem (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 %= 式の中身を出力する．

  機能説明 %= 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.16 OpenJIT.frontend.tree.AssignShiftLeftExpression クラス

public class AssignShiftLeftExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の <<= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignShiftLeftExpression(int where, Expression left, Expression right)

  機能概要 AST の <<= 式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，<<= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (<<= 式の左辺式の Expression オブジェクト)，right (<<= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGLSHIFT(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 <<= 式の型を選択する．

機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

機能概要 <<= を行うバイトコードを生成する．

機能説明 各型ごとに適切な <<= 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_ishl (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 <<= 式の中身を出力する．

機能説明 <<= 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.17 OpenJIT.frontend.tree.AssignShiftRightExpression クラス

public class AssignShiftRightExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の >>= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignShiftRightExpression(int where, Expression left, Expression right)

  機能概要 AST の >>= 式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，>>= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (>>= 式の左辺式の Expression オブジェクト)，right (>>= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGRSHIFT(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 >>= 式の型を選択する．

機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 >>= を行うバイトコードを生成する．

  機能説明 各型ごとに適切な >>= 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 opc_ishr (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 >>= 式の中身を出力する．

  機能説明 >>= 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.18 OpenJIT.frontend.tree.AssignSubtractExpression クラス

public class AssignSubtractExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の –= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignSubtractExpression(int where, Expression left, Expression right)

  機能概要 AST の –= 式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，–= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (–= 式の左辺式の Expression オブジェクト)，right (–= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGSUB(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 –= 式の型を選択する．

機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

機能概要 –= を行うバイトコードを生成する．

機能説明 各型ごとに適切な –= 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_isub (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 –= 式の中身を出力する．

機能説明 –= 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.19 OpenJIT. frontend. tree. AssignUnsignedShiftRightExpression クラス

public class AssignUnsignedShiftRightExpression extends AssignOpExpression

(1) 概要

このクラスは，AST の >>>= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public AssignUnsignedShiftRightExpression(int where, Expression left, Expression right)

  機能概要 AST の >>>= 式ノードを生成する．

  機能説明 スーパークラスの AssignOpExpression クラスで定義されているコンストラクタを用いて，>>>= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (>>>= 式の左辺式の Expression オブジェクト)，right (>>>= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 ASGURSHIFT(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

機能概要 >>>= 式の型を選択する．

機能説明 フィールド type (スーパークラス AssignOpExpression のフィールドを継承) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 この式の型を tm で指定される型にし，右辺式をこの型に変換する．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

機能概要 >>>= を行うバイトコードを生成する．

機能説明 各型ごとに適切な >>>= 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_iushr (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 >>>= 式の中身を出力する．

機能説明 >>>= 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.20 OpenJIT.frontend.tree.BinaryArithmeticExpression クラス

public class BinaryArithmeticExpression extends BinaryExpression

(1) 概要

このクラスは，AST の +, –, *, / 式のノードのスーパークラスであり，これらに共通のコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public BinaryArithmeticExpression(int op, int where, Expression left, Expression right)

  機能概要 BinaryArithmticExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ op (演算子の種類), where (行番号)，left (式の左辺式の Expression オブジェクト)，right (式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, left.type, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 式の型を選択する．

機能説明 フィールド type (スーパークラス BinaryExpression のフィールドを継承する) を設定し，型変換を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 tm で指定された型をこの式の型に設定し，左辺式 left，右辺値 right の型をそれぞれこの型に変換する．

エラー処理 特になし．

4.5.21 OpenJIT.frontend.tree.BinaryAssignExpression クラス

public class BinaryAssignExpression extends BinaryExpression

(1) 概要

このクラスは，AssignExpression, AssignOpExpression クラスのスーパークラスであり，これらに共通のコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- BinaryAssignExpression(int op, int where, Expression left, Expression right)

  機能概要 BinaryAssignExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ op (演算子の種類), where (行番号)，left (式の左辺値の Expression オブジェクト)，right (式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, left.type, left, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public Expression order()

  機能概要 式を演算子の優先順位に従って並び換える．

  機能説明 この式の優先順位が左辺の式より高い場合，左辺と右辺を入れ換える．

入力データ なし．

出力データ 並び換え後の Expression．

処理方式 スーパークラスの precedence メソッドを呼び，その値が left.precedence() 以上なら，式 left に式 left.right を保持し，元の left.right には再帰的に order メソッドを呼び出した結果を保持する．これを繰り返すことにより，最終的に優先順位順に並び換えた式が this に得られるので， this を返す．

エラー処理 特になし．

- public long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 void 式が含まれるかどうかを検査する．

  機能説明 右辺式，左辺式のそれぞれについて void が含まれるかどうかを検査し，その結果を long で返す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値

  処理方式 スーパークラスの checkValue を引数 env, ctx, vset, exp で呼び出す．

  エラー処理 特になし．

- public Expression inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 inlineValue メソッドを呼び出し，式の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 inlineValue メソッドを引数 env, ctx で呼び出す．

  エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

機能概要 式の inlining を行う．

機能説明 右辺の式，左辺の式をそれぞれ inlining する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 left.inlineLHS メソッドを呼び出し，左辺式を inlining した結果を left に格納する．right.inlineValue メソッドを呼び出し，右辺値を inlining した結果を right に格納する．その後， this を返す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 式の中身を出力する．

機能説明 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.22 OpenJIT.frontend.tree.BinaryCompareExpression クラス

public class BinaryCompareExpression extends BinaryExpression

(1) 概要

このクラスは，GreaterExpression, GreaterOrEqualExpression, LessExpression, LessOrEqualExpression クラスのスーパークラスであり，比較演算に共通するコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public BinaryCompareExpression(int op, int where, Expression left, Expression right)

  機能概要 BinaryCompareExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ op (演算子の種類), where (行番号)，left (式の左辺値の Expression オブジェクト)，right (式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, Type.tBoolean(定数), left, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

機能概要 式の型を選択する．

機能説明 フィールド type (スーパークラス BinaryExpression のフィールドを継承する) を設定し，型変換を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 左辺式 left，右辺式 right の型を tm で指定された型に convert メソッドを呼び出して変換する．

エラー処理 特になし．

4.5.23 OpenJIT.frontend.tree.BinaryEqualityExpression クラス

public class BinaryEqualityExpression extends BinaryExpression

(1) 概要

このクラスは，EqualExpression, NotEqualExpression クラスのスーパークラスであり，等号・不等号演算に共通するコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public BinaryEqualityExpression(int op, int where, Expression left, Expression right)

  機能概要 BinaryEqualityExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ op (演算子の種類), where (行番号)，left (式の左辺値の Expression オブジェクト)，right (式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, Type.tBoolean(定数), left, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

機能概要 式の型を選択する．

機能説明 フィールド type (スーパークラス BinaryExpression のフィールドを継承する) を設定し，型変換を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 左辺式 left，右辺式 right の型を tm で指定された型に convert メソッドを呼び出して変換する．

エラー処理 特になし．

4.5.24 OpenJIT.frontend.tree.BinaryExpression クラス

public class BinaryExpression extends UnaryExpression

(1) 概要

このクラスは，BinaryArithmeticExpression, BinaryAssignExpression, BinaryCompareExpression, BinaryEqualityExpression, BinaryLogicalExpression, BinaryShiftExpression, CastExpression, CommaExpression, ConditionalExpression, InstanceOfExpression クラスのスーパークラスであり，2 項演算に共通するコード生成の機能を実現する．

(2) フィールド (変数)

- Expression left

  左辺式．

(3) コンストラクタ

- public BinaryExpression(int op, int where, Type type, Expression left, Expression right)

  機能概要 BinaryExpression オブジェクトを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されているコンストラクタを呼び出して，BinaryExpression オブジェクトを生成・初期化する．

  入力データ op (演算子の種類), where (行番号), type (型), left (左辺式の Expression), right (右辺式の Expression)

  出力データ なし．

  処理方式 super(op, where, type, right) を呼び出す．left を this.left に格納する．

  エラー処理 特になし．

(4) メソッド

- Expression left()

  機能概要 左辺式のアクセッサメソッド．

  機能説明 左辺式を返す．

  入力データ なし．

  出力データ 左辺式の Expression

  処理方式 left を返す．

  エラー処理 特になし．

- public Expression order()

  機能概要 式を演算子の優先順位に従って並び換える．

  機能説明 この式の優先順位が左辺の式より高い場合，左辺と右辺を入れ換える．

  入力データ なし．

  出力データ 並び換え後の Expression．

  処理方式 スーパークラスの precedence メソッドを呼び，その値が left.precedence() 以上なら，式 left に式 left.right を保持し，元の left.right には再帰的に order メソッドを呼び出した結果を保持する．これを繰り返すことにより，最終的に優先順位順に並び換えた式が this に得られるので， this を返す．

  エラー処理 特になし．

- long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 値を検査する．

  機能説明 右辺式，左辺式の検査を行い，式の型を両辺の型から選択して，設定する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 left.checkValue メソッドを呼び出して，戻り値を vset に格納する．
right.checkValue メソッドを呼び出して，戻り値を vset に格納する．
left.type.getMask() | right.type.getMask() で得られる型に selectType を行う．最後に vset を返す．

エラー処理 特になし．

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 何もせずに this を返す．

  入力データ a (int 型), b (int 型)

  出力データ 評価後の Expression オブジェクト

  処理方式 this を返す．

  エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 何もせずに this を返す．

  入力データ a (long 型), b (long 型)

  出力データ 評価後の Expression オブジェクト

  処理方式 this を返す．

  エラー処理 特になし．

- Expression eval(float a, float b)

  機能概要 式を評価する．引数がともに float 型の場合．

  機能説明 何もせずに this を返す．

  入力データ a (float 型), b (float 型)

出力データ 評価後の Expression オブジェクト

処理方式 this を返す．

エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数がともに double 型の場合．

  機能説明 何もせずに this を返す．

  入力データ a (double 型), b (double 型)

  出力データ 評価後の Expression オブジェクト

  処理方式 this を返す．

  エラー処理 特になし．

- Expression eval(boolean a, boolean b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 何もせずに this を返す．

  入力データ a (boolean 型), b (boolean 型)

  出力データ 評価後の Expression オブジェクト

  処理方式 this を返す．

  エラー処理 特になし．

- Expression eval(String a, String b)

  機能概要 式を評価する．引数がともに String 型の場合．

  機能説明 何もせずに this を返す．

  入力データ a (String 型), b (String 型)

  出力データ 評価後の Expression オブジェクト

  処理方式 this を返す．

  エラー処理 特になし．

- Expression eval()

  機能概要 式を評価する．

  機能説明 eval メソッドを呼び出して式の評価を行う．

  入力データ なし．

  出力データ 評価後の Expression オブジェクト

  処理方式 left.op と right.op が等しくない場合， this を返す．等しい場合，eval(left.value, right.value) を呼び出し，その結果の Expression オブジェクトを返す．

  エラー処理 特になし．

- public Expression inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 左辺式，右辺式の inlining を行い，結果の式を返す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 left.inline メソッドを呼び出し，inlining を行った結果の式を left に格納する．right.inline メソッドを呼び出し，inlining を行った結果の式を right に格納する．left が null になったら，right のみを返す．null でなかったら，“left, right” に相当する CommaExpression を生成して返す．

  エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 左辺式，右辺式の inlining を行い，eval メソッドを呼び出して式全体を評価し，simplify メソッドを呼び出して式の単純化を行う．結果の式を返す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 left.inlineValue メソッドを呼び出し，inliningValue を行った結果の式を left に格納する．right.inlineValue メソッドを呼び出し，inlining を行った結果の式を right に格納する．eval().simplify() メソッドを呼び出して，式全体の評価と単純化を行い，その結果の式を返す．

エラー処理 simplify 時に数値エラーが生じた場合，arithmetic.exception を出力する．

- public Expression copyInline(Context ctx)

機能概要 配列式の inlining されたコピーを作る．

機能説明 配列式のクローンオブジェクトを作り，その left, right について inlining を行う．

入力データ ctx (コンテクストを格納する Context オブジェクト)

出力データ 配列式のオブジェクト．

処理方式 clone メソッドを呼び出し，BinaryExpression オブジェクト e を作る．left.copyInline メソッドを呼び出し，その結果の式を e.left に格納し，right.copyInline メソッドを呼び出し，その結果の式を e.right に格納する．その後，e を返す．

エラー処理 特になし．

- public int costInline(int thresh)

機能概要 inlining のコストを求める．

機能説明 left, right の inlining のコストを求め，配列式全体の inlining コストを求める．

入力データ thresh (閾値)

出力データ inlining のコスト．

処理方式 left.costInline, right.costInline メソッドを呼び出し，left, right の inlining コストを求める．この 2 つの値と 1 を加えた値を返す．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 二項演算を行うバイトコードを生成する．

  機能説明 二項演算を行うバイトコードを生成する．このメソッドはサブクラスでオーバーライドされていない場合に，エラーを出力する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 CompilerError(“codeOperation: *opNames[op]*”) エラーを出力する．

  エラー処理 無条件に，CompilerError(“codeOperation: *opNames[op]*”) エラーを出力する．

- public void codeValue(Environment env, Context ctx, Assembler asm, ClassDeclaration c)

  機能概要 2 項演算式のバイトコードを生成する．

  機能説明 左辺式の値，右辺式の値をスタックに積み，2 項演算を行うバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), c (ClassDeclaration オブジェクト)

  出力データ なし．

  処理方式 type を検査して，型が boolean 型でない場合，left.codeValue メソッド，right.codeValue メソッドを呼び出してスタック上に左辺式と右辺式の値が積まれるようにバイトコードを生成する．次に codeOperation メソッ

ドを呼び出してスタック上に 2 つ値に対する演算を行うバイトコードを生成する．

boolean 型の場合，新しいラベル l1, l2 を生成する．codeBranch メソッドを呼び出し，演算の結果，条件が成立すればラベル l1 にジャンプするバイトコードを生成する．次に，コンスタントプールに格納されている 0 という値を読み出すために opc_ldc をオペコードとするバイトコードを生成し，l2 にジャンプする opc_goto をオペコードとするバイトコードを生成する．この直後にラベル l1 が入るようにし，次にコンスタントプールに格納されている 1 という値を読み出すために opc_ldc をオペコードとするバイトコードを生成し，その直後にラベル l2 が入るようにする．

エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 2 項演算式の中身を出力する．

  機能説明 2 項演算式の中身を PrintStream out に出力する．

  入力データ out (String の出力先の PrintStream)

  出力データ なし．

  処理方式 2 項演算式の中身を “(op left right)” の形式で PrintStream out に出力する．

  エラー処理 特になし．

- String infixForm(String operator)

  機能概要 2 項演算式の中身を出力する．

  機能説明 2 項演算式の中身を文字列形式で出力する．

  入力データ operator (オペレータの文字列)

  出力データ 中置形式での 2 項演算式の文字列．

  処理方式 2 項演算式の中身を “(left op right)” の形式の文字列に変換し，出力する．

  エラー処理 特になし．

4.5.25 OpenJIT.frontend.tree.BinaryShiftExpression クラス

public class BinaryShiftExpression extends BinaryExpression

(1) 概要

このクラスは，ShiftLeftExpression, ShiftRightExpression, UnsignedShiftRightExpression クラスのスーパークラスであり，これらに共通のコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public BinaryShiftExpression(int op, int where, Expression left, Expression right)

  機能概要 BinaryShiftExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを呼び出して，BinaryShiftExpression オブジェクトを生成する．

  入力データ op (演算の種類), where (行番号)，left (左辺式の Expression オブジェクト)，right (右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, left.type, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 BinaryShiftExpression の型を選択する．

機能説明 与えられた型に式の型を設定し，左辺式，右辺式の型を変換する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 tm で与えられる型が long 型の場合， type を Type.tLong (定数) に設定する．そうでない場合， Type.tInt (定数) に設定する． left の型を type に変換するために convert メソッドを呼び出し，その結果を left に格納する． right の型を type に変換するために convert メソッドを呼び出し，その結果を right に格納する．

エラー処理 特になし．

4.5.26 OpenJIT.frontend.tree.BitAndExpression クラス

public class BitAndExpression extends BinaryBitExpression

(1) 概要

このクラスは，AST の & 演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public BitAndExpression(int where, Expression left, Expression right)

  機能概要 AST の & 式ノードを生成する．

  機能説明 スーパークラスの BinaryBitExpression クラスで定義されているコンストラクタを用いて，& 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (& 式の左辺式の Expression オブジェクト)，right (& 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 BITAND(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(boolean a, boolean b)

  機能概要 式を評価する．引数がともに boolean 型の場合．

  機能説明 boolean 型の引数の bit And を行い，BooleanExpression オブジェクトを生成する．

入力データ a (boolean 型), b (boolean 型)

出力データ Expression オブジェクト．

処理内容 a&b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数の bit And を行い，IntExpression オブジェクトを生成する．

  入力データ a (int 型), b (int 型)

  出力データ Expression オブジェクト．

  処理内容 a&b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 int 型の引数の bit And を行い，LongExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a&b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- void setBitCondition(long vsTrue, long vsFalse, ConditionVars cvars)

  機能概要 ビット演算に用いる条件の評価値を設定する．

機能説明 vsTrue, vsFalse に基づいて cvars を適切に設定する．

入力データ vsTrue (左辺式の評価後の variable set), vsFalse (左辺式の評価後の variable set), cvars (右辺式の評価後の variable set)

出力データ なし．

処理方式 cvars.vsTrue と vsFalse の bit Or を取り，その結果と cvars.vsFalse の bit And を取って，その結果を cvars.vsFalse に格納する．cvars.vsTrue と vsTrue の bit Or を取り，その結果を cvars.vsTrue に格納する．

エラー処理 特になし．

- Expression simpify()

機能概要 & 式を単純化する．

機能説明 右辺式ないし左辺式が true の場合，左辺式ないし右辺式が false の場合に式を単純化して返す．

入力データ なし．

出力データ 単純化後の式の Expression オブジェクト．

処理方式 左辺式が true の場合，right を返す．右辺式が true の場合，left を返す．左辺式が false または 0 の場合，“right, left” という CommaExpression オブジェクトを生成し，それに対して simplify メソッドを呼び出した結果を返す．右辺式が false または 0 の場合，“left, right” という CommaExpression オブジェクトを生成し，それに対して simplify メソッドを呼び出した結果を返す．それ以外の場合は this を返す．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

機能概要 & を行うバイトコードを生成する．

機能説明 各型ごとに適切な & 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_iand (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 & 式の中身を出力する．

機能説明 & 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.27 OpenJIT.frontend.tree.BitNotExpression クラス

public class BitNotExpression extends UnaryExpression

(1) 概要

このクラスは，AST の!演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public BitNotExpression(int where, Expression right)

  機能概要 AST の!式ノードを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されているコンストラクタを用いて，!式ノードの生成・初期化を行う．

  入力データ where (行番号)，right (!式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 BITNOT(定数), where, right.type, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 !式の型を選択する．

  機能説明 フィールド type (スーパークラス UnaryExpression のフィールドを継承する) を設定し，型変換を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 右辺式 right の型を tm で指定された型に convert メソッドを呼び出して変換する．

エラー処理 特になし．

- Expression eval(int a)

機能概要 式を評価する．引数が int 型の場合．

機能説明 int 型の引数の bit Not を行い，IntExpression オブジェクトを生成する．

入力データ a (int 型)

出力データ Expression オブジェクト．

処理内容 ~a を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a)

機能概要 式を評価する．引数が long 型の場合．

機能説明 long 型の引数の bit Not を行い，LongExpression オブジェクトを生成する．

入力データ a (long 型)

出力データ Expression オブジェクト．

処理内容 ~a を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression simpify()

機能概要 式の単純化を行う．

機能説明 右辺式の op フィールドが BITNOT(定数) の場合，2 つの BITNOT を相殺して式を単純化する．

入力データ なし．

出力データ Expression オブジェクト．

処理方式 right.op が BITNOT(定数) の場合，right.right を返す．それ以外の場合，this を返す．

エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm, ClassDeclaration c)

機能概要 !式のバイトコードを生成する．

機能説明 右辺式の値をロードし，–1 と xor を取るバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), c (ClassDeclaration オブジェクト)

出力データ なし．

処理方式 right.codeValue メソッドを呼び出し，右辺式の値がスタックトップに積まれるようにバイトコードを生成する．

type が TC_INT に等しい場合，int 型の –1 が格納されているコンスタントプールから値をスタック上に積むバイトコードを生成するために，opc_ldc をオペコードとするバイトコードを生成し，右辺の値と –1 との xor を取るバイトコードを生成するために，opc_ixor をオペコードとするバイトコードを生成する．

そうでない場合，long 型の –1 が格納されているコンスタントプールから値をスタック上に積むバイトコードを生成するために，opc_ldc2_w をオペコードとするバイトコードを生成し，右辺の値と –1 との xor を取るバイトコードを生成するために，opc_lxor をオペコードとするバイトコードを生成する．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 !式の中身を出力する．

  機能説明 !式を前置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 prefixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.28 OpenJIT.frontend.tree.BitOrExpression クラス

public class BitOrExpression extends BinaryBitExpression

(1) 概要

このクラスは，AST の | 演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public BitOrExpression(int where, Expression left, Expression right)

  機能概要 AST の | 式ノードを生成する．

  機能説明 スーパークラスの BinaryBitExpression クラスで定義されているコンストラクタを用いて，| 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (| 式の左辺式の Expression オブジェクト)，right (| 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 BITOR(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(boolean a, boolean b)

  機能概要 式を評価する．引数がともに boolean 型の場合．

  機能説明 boolean 型の引数の bit Or を行い，BooleanExpression オブジェクトを生成する．

入力データ a (boolean 型), b (boolean 型)

出力データ Expression オブジェクト．

処理内容 a|b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数の bit Or を行い，IntExpression オブジェクトを生成する．

  入力データ a (int 型), b (int 型)

  出力データ Expression オブジェクト．

  処理内容 a|b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 int 型の引数の bit Or を行い，LongExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a|b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- void setBitCondition(long vsTrue, long vsFalse, ConditionVars cvars)

  機能概要 ビット演算に用いる条件の評価値を設定する．

機能説明 vsTrue, vsFalse に基づいて cvars を適切に設定する．

入力データ vsTrue (左辺式の評価後の variable set), vsFalse (左辺式の評価後の variable set), cvars (右辺式の評価後の variable set)

出力データ なし．

処理方式 cvars.vsTrue と vsFalse の bit Or を取り，その結果と cvars.vsFalse の bit And を取って，その結果を cvars.vsFalse に格納する．cvars.vsTrue と vsTrue の bit Or を取り，その結果を cvars.vsTrue に格納する．

エラー処理 特になし．

- Expression simpify()

機能概要 | 式を単純化する．

機能説明 右辺式ないし左辺式が false の場合，左辺式ないし右辺式が true の場合に式を単純化して返す．

入力データ なし．

出力データ 単純化後の式の Expression オブジェクト．

処理方式 左辺式が false または 0 の場合，right を返す．右辺式が false または 0 の場合，left を返す．左辺式が true の場合，“right, left” という Comma-Expression オブジェクトを生成し，それに対して simplify メソッドを呼び出した結果を返す．右辺式が true の場合，“left, right” という CommaExpression オブジェクトを生成し，それに対して simplify メソッドを呼び出した結果を返す．それ以外の場合は this を返す．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

機能概要 | を行うバイトコードを生成する．

機能説明 各型ごとに適切な | 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_ior (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 | 式の中身を出力する．

機能説明 | 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.29 OpenJIT.frontend.tree.BitXorExpression クラス

public class BitXorExpression extends BinaryBitXorExpression

(1) 概要

このクラスは，AST の^演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public BitXorExpression(int where, Expression left, Expression right)

  機能概要 AST の^式ノードを生成する．

  機能説明 スーパークラスの BinaryBitExpression クラスで定義されているコンストラクタを用いて，^式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (^式の左辺式の Expression オブジェクト)，right (^式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 BITXOR(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(boolean a, boolean b)

  機能概要 式を評価する．引数がともに boolean 型の場合．

  機能説明 boolean 型の引数の bit Xor を行い，BooleanExpression オブジェクトを生成する．

入力データ a (boolean 型), b (boolean 型)

出力データ Expression オブジェクト．

処理内容 a^b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数の bit Xor を行い，IntExpression オブジェクトを生成する．

  入力データ a (int 型), b (int 型)

  出力データ Expression オブジェクト．

  処理内容 a^b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 int 型の引数の bit Xor を行い，LongExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a^b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- void setBitCondition(long vsTrue, long vsFalse, ConditionVars cvars)

  機能概要 ビット演算に用いる条件の評価値を設定する．

機能説明 vsTrue, vsFalse に基づいて cvars を適切に設定する．

入力データ vsTrue (左辺式の評価後の variable set), vsFalse (左辺式の評価後の variable set), cvars (右辺式の評価後の variable set)

出力データ なし．

処理方式 cvars.vsTrue, cvars.vsFalse の値をそれぞれ nTrue, nFalse に格納する．cvars.vsTrue の値を (nTrue & nFalse) | (vsTrue & nTrue) | (vsFalse & nFalse) で更新する．cvars.vsFalse の値を (nTrue & nFalse) | (vsTrue & nFalse) | (vsFalse & nTrue) で更新する．

エラー処理 特になし．

- Expression simpify()

機能概要 ^ 式を単純化する．

機能説明 右辺式ないし左辺式が true の場合，左辺式ないし右辺式が false の場合に式を単純化して返す．

入力データ なし．

出力データ 単純化後の式の Expression オブジェクト．

処理方式 左辺式が true の場合，right を引数とする NotExpression を生成して返す．右辺式が true の場合，left を引数とする NotExpression を生成して返す．左辺式が false または 0 の場合，right を返す．右辺式が false または 0 の場合，left を返す．それ以外の場合は this を返す．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

機能概要 ^ を行うバイトコードを生成する．

機能説明 各型ごとに適切な^用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_ixor (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 ^ 式の中身を出力する．

機能説明 ^ 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.30 OpenJIT.frontend.tree.BooleanExpression クラス

public class BooleanExpression extends ConstantExpression

(1) 概要

このクラスは，AST の boolean 型ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- boolean value

  真偽値を保持する

(3) コンストラクタ

- public BooleanExpression(int where, boolean value)

  機能概要 BooleanExpression オブジェクトを生成する．

  機能説明 スーパークラスの ConstantExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), value (保持する値)

  出力データ なし．

  処理方式 super メソッドを引数 BOOLEANVAL(定数), where, Type.Boolean (定数) で呼び出し，this.value に value を代入する．

  エラー処理 特になし．

(4) メソッド

- public Object getValue()

  機能概要 値を取り出す．

  機能説明 value を保持する Integer オブジェクトを返す．

入力データ なし．

出力データ 数値オブジェクト．

処理方式 value が True なら 1，Flase なら 0 を保持する Integer オブジェクトを生成して，返す．

エラー処理 特になし．

- public boolean equals(boolean b)

  機能概要 値が与えられた真偽値と一致するか否かを検査する．

  機能説明 値 value が与えられた真偽値と一致するか否かを検査する．

  入力データ b (比較する値)

  出力データ 一致・不一致の真偽値

  処理方式 value==b の結果を返す．

  エラー処理 特になし．

- public boolean equalsDefault()

  機能概要 値が False と一致するか否かを検査する．

  機能説明 値 value を negate した結果を返す．

  入力データ なし．

  出力データ 一致・不一致の真偽値

  処理方式 !value の結果を返す．

  エラー処理 特になし．

- public void checkCondition(Environment env, Context ctx, long vset, Hashtable exp, ConditionVars cvars)

  機能概要 条件を検査する．

  機能説明 value の値に従って，引数の cvars を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), cvars (条件の評価値オブジェクト)

出力データ なし．

処理方式 value が真ならば，cvars.vsFalse に −1, cvars.vsTrue に vset を格納する．偽ならば，cvars.vfFalse に vset, vs.vsTrue に −1 を格納する．

エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

  機能概要 分岐命令のバイトコードを生成する．

  機能説明 分岐命令のバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

  出力データ なし．

  処理方式 whenTrue が value に等しい場合，lbl にジャンプするために，opc_goto をオペコードとするバイトコードを生成する．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 真偽値をスタックトップに積むバイトコードを生成する．

  機能説明 真偽値 value をスタックトップに積むバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

処理方式 value が真なら 1，偽なら 0 をスタックトップに積むために，opc_ldc をオペコードとするバイトコードを生成する．

エラー処理 なし．

- public void print(PrintStream out)

  機能概要 値を読み易く出力する．

  機能説明 与えられた PrintStream に value を true/false で出力する．

  入力データ out (出力先の PrintStream)

  出力データ なし．

  処理方式 value が真なら “true”，偽なら “false” で，out.print メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 真偽値の中身を出力する．

  機能説明 真偽値を true/false で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 value が真なら “true”，偽なら “false” の String を返す．

  エラー処理 特になし．

4.5.31 OpenJIT.frontend.tree.BreakStatement クラス

public class BreakStatement extends Statement

(1) 概要

このクラスは，AST の Break 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Label lbl

  ラベル．

(3) コンストラクタ

- public BreakStatement(int where, Identifier lbl)

  機能概要 BreakStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), lbl (break のラベル)

  出力データ なし．

  処理方式 super メソッドを引数 BREAK (定数), where で呼び出し，this.lbl に lbl を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 Break 文の検査を行う．

  機能説明 Break 文の検査を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 lbl で指定されたコンテクストを取得し，CheckContext destctx に格納する．destctx が null でない場合，destctx.vsBreak & vset を destctx.vsBreak に格納し，-1 を返す．そうでない場合，invalid.break エラーを出力する．

エラー処理 lbl で与えられたコンテクストが存在しない場合に invalid.break エラーを出力する．

- public int costInline(int thresh)

機能概要 Inlining のコストを計算する．

機能説明 Break 文の Inlining のコストとして 1 を返す．

入力データ thresh (閾値)

出力データ Inlining のコスト．

処理方式 1 を返す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

機能概要 Break 文を処理するバイトコードを生成する．

機能説明 Break 文でジャンプした後の CodeContext を生成し，設定した後，Break 文を処理するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 新しい CodeContext オブジェクト newctx を生成し，現在のコンテクストを設定する． destctx に newctx.getBreakContext を呼び出して得られるコンテクストを設定する．次に codeFinally メソッドを呼び出し，destctx に break 文でジャンプする先のラベルを取得する．その後，break で dest.breakLabel にジャンプするバイトコードを生成するために，opc_goto をオペコードとするバイトコードを生成する．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

  機能概要 Break 文を出力する．

  機能説明 Break 文を読みやすくインデントを付けて出力する．

  入力データ out (String の出力先の PrintStream)， indent (インデントの深さ)

  出力データ 特になし．

  処理方式 super.print(out, indent) を呼び出して，スーパークラスで定義された print メソッドで，必要な数のインデントを行う． “break;” の文字列を PrintStream out に出力する．

  エラー処理 特になし．

4.5.32 OpenJIT.frontend.tree.ByteExpression クラス

public class ByteExpression extends IntegerExpression

(1) 概要

このクラスは，AST の byte 型整数ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public ByteExpression(int where, short value)

  機能概要 ByteExpression オブジェクトを生成する．

  機能説明 スーパークラスの IntegerExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), value (保持する値)

  出力データ なし．

  処理方式 super メソッドを引数 BYTEVAL(定数), where, Type.tByte(定数), value で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public void print(PrintStream out)

  機能概要 値を読み易く出力する．

  機能説明 与えられた PrintStream に value を十進数表現で出力する．

  入力データ out (出力先の PrintStream)

出力データ なし．

処理方式 value の十進数表現に b を付けて，out.print メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 整数値の中身を出力する．

  機能説明 整数値を十進数で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 postfixForm メソッドを引数 “b” で呼び出す．

  エラー処理 特になし．

4.5.33 OpenJIT.frontend.tree.CaseStatement クラス

public class CaseStatement extends Statement

(1) 概要

このクラスは，AST の case 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Expression expr
  case 文の条件式．

(3) コンストラクタ

- public CaseStatement(int where, Expression expr)

  機能概要 CaseStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), expr (Case 文の条件式)

  出力データ なし．

  処理方式 super メソッドを引数 CASE (定数), where で呼び出し，this.expr に expr を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 Case 文の検査を行う．

  機能説明 Case 文の検査を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 expr が null でなければ，expr.checkValue メソッドを呼び出して，Case ラベルの検査を行う．expr を int 型に変換するために，convert メソッドを呼び出し，戻り値を expr に格納する．次に，expr.inlineValue メソッドを呼び出して，expr の inlining を行い，戻り値を expr に格納する．vset & D̃EAD_END を計算して，返す．

エラー処理 特になし．

- public int costInline(int thresh)

機能概要 Inlining のコストを計算する．

機能説明 Case 文の Inlining のコストとして 6 を返す．

入力データ thresh (閾値)．

出力データ Inlining のコスト．

処理方式 6 を返す．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

機能概要 Case 文を出力する．

機能説明 Case 文を読み易くインデントを付けて出力する．

入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

出力データ 特になし．

処理方式 super.print(out, indent) を呼び出して，スーパークラスで定義された print メソッドで，必要な数のインデントを行う．expr が null であれば，“default:”，null でなければ，“case *expr*:” の文字列を PrintStream out に出力する．

エラー処理 特になし．

4.5.34 OpenJIT.frontend.tree.CastExpression クラス

public class CastExpression extends BinaryExpression

(1) 概要

このクラスは，AST のキャスト式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public CastExpression(int where, Expression left, Expression right)

  機能概要 キャスト式を表す CastExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号)，left (式の左辺値の Expression オブジェクト)，right (式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 CAST (定数), where, left.type, left, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 キャスト式の検査を行う．

  機能説明 キャスト式の検査を行い，条件の評価値を更新する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 左辺式の型を toType メソッドを呼び出すことにより得，type に格納する．右辺式の条件の評価値を得るために checkValue メソッドを呼び出す．type と右辺式の型が一致しているかどうか検査し，一致していれば，vset を返す．そうでなければ，右辺式を格納する ConvertExpression オブジェクトをコンストラクタを呼び出すことにより，生成し，right に格納し，vset を返す．

エラー処理 特になし．

- public Expression inline(Environment env, Context ctx)

機能概要 キャスト式の Inlining を行う．

機能説明 右辺式の inline メソッドを呼び出して，キャスト式の Inlining を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ Inlining 後の式．

処理方式 right.inline メソッドを呼び出す．

エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

機能概要 キャスト式の Inlining を行う．

機能説明 右辺式の inlineValue メソッドを呼び出して，キャスト式の Inlining を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ Inlining 後の式．

処理方式 right.inlineValue メソッドを呼び出す．

エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 キャスト式の中身を出力する．

  機能説明 キャスト式を読み易く出力する．

  入力データ out (String の出力先の PrintStream)

  出力データ なし．

  処理方式 キャスト式の中身を “(left) right” の形式の文字列で PrintStream out に出力する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 キャスト式の中身を出力する．

  機能説明 キャスト式を読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 toPrettyString メソッドで left を文字列に変換し，“(left) right” という形式の文字列を生成して，返す．

  エラー処理 特になし．

4.5.35 OpenJIT.frontend.tree.CatchStatement クラス

public class CatchStatement extends Statement

(1) 概要

このクラスは，AST の catch 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Expression texpr

  catch 文の型式．

- Identifier id

  catch 文の識別子．

- Statement body

  catch 文の本体．

- LocalField field

  ローカルフィールド．

(3) コンストラクタ

- public CatchStatement(int where, Expression texpr, Identifier id, Statement body)

  機能概要 CatchStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), texpr (catch 文の型式), id (catch 文のアイデンティファイア)，body (catch 文の本体)

  出力データ なし．

処理方式 super メソッドを引数 CATCH (定数), where で呼び出し，this.expr に expr を，this.id に id を，this.body に body を格納する．

エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 Catch 文の検査を行う．

  機能説明 Catch 文の検査を行い，条件の評価値を更新する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値．

  処理方式 texpr.toType メソッドを呼び出して，texpr の型を type に得る．type がクラスオブジェクトでなければ，catch.not.throwable のエラーを返す．クラスオブジェクトの場合，type のクラス定義を読み出す．このクラス定義が，java.lang.throwable クラスのサブクラスでなければ，やはり，catch.not.throwable のエラーを返す．

  クラス定義からフィールドの個数を読み出し，vset | (1 << フィールド数) で vset を更新する．しかる後に，body.check メソッドを呼び出し，その戻り値の条件の評価値を返す．

  エラー処理 throwable でないオブジェクトが id として指定されている場合に，catch.not.throwable のエラーを返す．

- public Statement inline(Environment env, Context ctx)

  機能概要 Catch 文の Inlining を行う．

  機能説明 新しく Context を生成して，body の Inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ Inlining 後の Statement．

処理方式 Context クラスのコンストラクタを呼び出して，新しく Context を生成する．body が null かどうか検査して，null でなければ，body.inline メソッドを呼び出して，body の inlining を行う．最後に，this を返す．

エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

機能概要 catch 文の inlining されたコピーを作る．

機能説明 method inlining のために catch 文のコピーを作って，返す．

入力データ ctx (コンテクストを格納する Context オブジェクト), valNeeded (計算後の値をスタック上に残すか否かを決定する真偽値)

出力データ Inlining 後の Statement．

処理方式 clone メソッドを呼び出し，catch 文のコピーを作り，s に格納する．s.body の inlining を行うために，まず，body が null か否か検査する．body が null でなければ，body.inline メソッドを呼び出し，その結果で body を更新する．最後に s を返す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

機能概要 catch 文を実行するバイトコードを生成する．

機能説明 catch 文を実行するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 まず，opc_pop をオペコードとするバイトコードを生成する．body が null か否か検査して，null でなければ，body.code メソッドを呼び出す．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

  機能概要 catch 文の中身を出力する．

  機能説明 catch 文の中身を読み易く文字列で，PrintStream out に出力する．

  入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

  出力データ なし．

  処理方式 “catch (*texprid*) *body*” という形式で出力できるように，texpr.print メソッド，body.print メソッドを呼び出して，適宜出力する．

  エラー処理 特になし．

4.5.36 OpenJIT.frontend.tree.CharExpression クラス

public class CharExpression extends IntegerExpression

(1) 概要

このクラスは，AST の char 型整数ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public CharExpression(int where, char value)

  機能概要 CharExpression オブジェクトを生成する．

  機能説明 スーパークラスの IntegerExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), value (保持する値)

  出力データ なし．

  処理方式 super メソッドを引数 CHARVAL(定数), where, Type.tChar(定数), value で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public void print(PrintStream out)

  機能概要 値を読み易く出力する．

  機能説明 与えられた PrintStream に value を十進数表現で出力する．

  入力データ out (出力先の PrintStream)

出力データ なし．

処理方式 value の十進数表現に c を付けて，out.print メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 整数値の中身を出力する．

  機能説明 整数値を十進数で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 postfixForm メソッドを引数 “c” で呼び出す．

  エラー処理 特になし．

4.5.37 OpenJIT.frontend.tree.CheckContext クラス

class CheckContext extends Context

(1) 概要

このクラスは，ブロック文のために新しくネストしたコンテクストを保持する機能を実現する．

(2) フィールド (変数)

- long vsBreak

  Break の条件値．

- long vsContinue

  Continue の条件値．

(3) コンストラクタ

- CheckContext(Context ctx, Statement stat)

  機能概要 CheckContext オブジェクトを生成する．

  機能説明 CheckContext オブジェクトを生成し，初期設定を行う．

  入力データ ctx (コンテクストを格納する Context オブジェクト), stat (ネストする Statement オブジェクト)

  出力データ 特になし．

  処理方式 Context クラスのコンストラクタを呼び出すために，super(ctx, stat) を呼び出す．vsBreak, vsContinue を –1 に設定する．

  エラー処理 特になし．

(4) メソッド

なし．

4.5.38 OpenJIT.frontend.tree.CodeContext クラス

class CodeContext extends Context

(1) 概要

このクラスは，ブロック文のために新しくネストしたコンテクストを保持する機能を実現する．

(2) フィールド (変数)

- Label breakLabel

  break のジャンプ先のラベル．

- Label contLabel

  continue のジャンプ先のラベル．

(3) コンストラクタ

- CodeContext(Context ctx, Node node)

  機能概要 CodeContext オブジェクトを生成する．

  機能説明 CodeContext オブジェクトを生成し，初期設定を行う．

  入力データ ctx (コンテクストを格納する Context オブジェクト), node (ネストする Node オブジェクト)

  出力データ 特になし．

  処理方式 Context クラスのコンストラクタを呼び出すために，super(ctx, node) を呼び出す．node.op を調べて，DO, WHITE, FOR, FINALLY, SYNCHRONIZED の場合には，新しくラベルを 2 つ生成して，breakLabel, contLabel にそれぞれ格納する．SWITCH, TRY, INLINEMETHOD, INLINENEWINSTANCE の場合には，新しくラベルを 1 つ生成して，breakLabel に格納する．それ以外の場合には，node が Statement クラスのインスタン

スで，かつ node.labels が null でなければ，新しくラベルを 1 つ生成して，breakLabel に格納する．

エラー処理 特になし．

(4) メソッド

なし．

4.5.39 OpenJIT.frontend.tree.CommaExpression クラス

public class CommaExpression extends BinaryExpression

(1) 概要

このクラスは，AST の,(カンマ) 演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public CommaExpression(int where, Expression left, Expression right)

  機能概要 CommaExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号)，left (式の左辺値の Expression オブジェクト)，right (式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 right が null なら，super メソッドを引数 COMMA (定数), where, Type.tVoid(定数), left, right で呼び出す．そうでなければ，super メソッドを引数 COMMA (定数), where, right.type, left, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 CommaExpression を検査する．

機能説明 CommaExpression の左辺式，右辺式をそれぞれ検査する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 left.check メソッドを呼び出し，戻り値で vset を更新する．right.check メソッドを呼び出し，戻り値で vset を更新する．最後に， vset を返す．

エラー処理 特になし．

- void selectType(Environment env, Context ctx, int tm)

機能概要 式の型を選択する．

機能説明 右辺式の型をこの CommaExpression として返す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 right.type を type に格納する．

エラー処理 特になし．

- Expression simpify()

機能概要 CommaExpression 式を単純化する．

機能説明 左辺式ないし右辺式が null の場合に式を単純化する．

入力データ なし．

出力データ Expression オブジェクト．

処理方式 左辺式が null なら， right を返す．右辺式が null なら， left を返す．それ以外の場合は this を返す．

エラー処理 特になし．

- public Expression inline(Environment env, Context ctx)

機能概要 式の inlining を行う．

機能説明 右辺式，左辺式をそれぞれ inlining して，単純化する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 左辺式が null でなければ，left.inline メソッドを呼び出し，inlining を行った式を left に格納する．右辺式が null でなければ，right.inline メソッドを呼び出し，inlining を行った式を right に格納する．最後に，simplify メソッドを呼び出して，戻り値の式を返す．

エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 右辺式，左辺式をそれぞれ inlining して，単純化する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 左辺式が null でなければ，left.inline メソッドを呼び出し，inlining を行った式を left に格納する．右辺式が null でなければ，right.inlineValue メソッドを呼び出し，inlining を行った式を right に格納する．最後に，simplify メソッドを呼び出して，戻り値の式を返す．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 式のバイトコードを生成する．

  機能説明 式のバイトコードを生成する．スタックトップに右辺式の値を積む．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 左辺式が null でなければ，left.code を呼び出して，左辺式のバイトコードを生成する．右辺式が null でなければ，right.codeValue を呼び出して，右辺式のバイトコードを生成する．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 式のバイトコードを生成する．

  機能説明 式のバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 左辺式が null でなければ，left.code を呼び出して，左辺式のバイトコードを生成する．右辺式が null でなければ，right.code を呼び出して，右辺式のバイトコードを生成する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 カンマ式の中身を出力する．

  機能説明 カンマ式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 “(*left*, *right*)” の形式でカンマ式の中身を文字列に直し，出力する．

  エラー処理 特になし．

4.5.40 OpenJIT.frontend.tree.CompoundStatement クラス

public class CompoundStatement extends Statement

(1) 概要

このクラスは，AST の複合文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Statement args[]
  複合文の各文の配列．

(3) コンストラクタ

- public CompoundStatement(int where, Statement args[])

  機能概要 CompoundStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，生成・初期化を行う．

  入力データ where (行番号), args[] (Statement の配列)

  出力データ なし．

  処理方式 super メソッドを引数 STAT (定数), where で呼び出す．this.args に args を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 複合文の検査を行う．

  機能説明

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値

処理方式 新しく CheckContext newctx を作る．args を順にたどりながら，check メソッドを呼び出す．最後に，ctx.removeAddtionalVars メソッドを呼び出し，戻り値を返す．

エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

機能概要 複合文の inlining を行う．

機能説明 args の各文を順に inlining を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 変数 count を 0 に初期化する．args を順にたどりながら，inline メソッドを呼び出していく．args の要素が CompoundStatement であれば，count にその CompoundStatement の args の要素数を加える．そうでなければ，1 加算する．

count が 0 であれば，null を返す．1 であれば，CompoundStatement に wrap する必要はないので，eliminate メソッドを呼び出し，戻り値を返す．

args のサイズと count が一致しない場合，すなわち args の要素に CompoundStatement を含んでいるか，途中の要素に null を含んでいる場合，args をパッキングしなおして新たな配列 newArgs を生成する．args に newArgs を格納する．

最後に this を返す．

エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

機能概要 コピーした文を Inlining する．

機能説明 コピーした文を Inlining する．

入力データ ctx (コンテクストを格納する Context オブジェクト), valNeeded (評価後の値をスタックに残すかどうかのフラグ)

出力データ inlining された式．

処理方式 clone メソッドを呼び出して，戻り値を s に格納する．arg.length の大きさの Statement クラスの配列を生成して，s.args に格納する．args を順にたどりながら，copyInline メソッドを呼び出し，戻り値を s.args の要素に格納していく．最後に s を返す．

エラー処理 特になし．

- public int costInline(int thresh)

機能概要 inlining のコストを計算する．

機能説明 args の inlining のコストを計算する．

入力データ thresh (閾値)

出力データ inlining のコスト

処理方式 args を順にたどりながら，costInline メソッドを呼び出し，その戻り値を加算していく．最後に総計を返す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

機能概要 複合文を実行するバイトコードを生成する．

機能説明 args をそれぞれ実行するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 args を順にたどりながら，code メソッドを呼び出していく．最後に breakLabel を挿入する．

エラー処理 特になし．

- public boolean firstConstructor()

機能概要 文の最初の処理がコンストラクタ呼び出しか否か検査する．

機能説明 文の最初の処理がコンストラクタ呼び出しか否か検査して，真偽値を返す．

入力データ なし．

出力データ 真偽値．

処理方式 args.length が正，かつ args[0].firstConstructor メソッドを呼び出し，戻り値が真ならば，true，それ以外なら，false を返す．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

機能概要 複合文を出力する．

機能説明 Break 文を読み易くインデントを付けて出力する．

入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

出力データ 特になし．

処理方式 以下のような形式で，PrintStream out に出力する．

```
{
    statement0
    statement1
    statement2
    ...
}
```

エラー処理 特になし．

4.5.41 OpenJIT.frontend.tree.ConditionVars クラス

class ConditionVars

(1) 概要

このクラスは，条件の評価値を保持するために用いられる．

(2) フィールド (変数)

- long vsTrue
  True 分岐のための Variable set
- long vsFalse
  False 分岐のための Variable set

(3) コンストラクタ

なし．

(4) メソッド

なし．

4.5.42 OpenJIT.frontend.tree.ConditionalExpression クラス

public class ConditionalExpression extends BinaryExpression

(1) 概要

このクラスは，AST の条件演算式 (?:) のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Expression cond
  条件式の Expression．

(3) コンストラクタ

- public ConditionalExpression(int where, Expression cond, Expression left, Expression right)

  機能概要 ConditionalExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを用いて，条件式ノードの生成・初期化を行う．

  入力データ where (行番号)，cond (条件式の Expression オブジェクト), left (式の左辺式の Expression オブジェクト)，right (式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 COND (定数), where, Type.tError (定数), left, right で呼び出す．this.cond に cond を格納する．

  エラー処理 特になし．

(4) メソッド

- public Expression order()

機能概要 式を演算子の優先順位に従って並び換える．

機能説明 この式の優先順位が左辺の式より高い場合，左辺と右辺を入れ換える

入力データ なし．

出力データ 並び換え後の Expression．

処理方式 スーパークラスの precedence メソッドを呼び，その値が cond.precedence() 以上なら，式 cond に式 cond.right を保持し，元の cond.right には再帰的に order メソッドを呼び出した結果を保持する．これを繰り返すことにより，最終的に優先順位順に並び換えた式が this に得られるので，this を返す．

エラー処理 特になし．

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

機能概要 式の検査を行う．

機能説明 条件式の型を boolean 型に変換する．左辺式の型に従って式の型を設定し，必要があれば変換を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値

処理方式 left.checkValue メソッドの戻り値と right.checkValue メソッドの戻り値の and を取り，vset に格納する．cond の型を boolean 型に変換するために，convert メソッドを呼び出し，その戻り値を cond に格納する．

left.type と right.type が一致すれば，left.type を type に代入する．そうでない場合，tm が TM_DOUBLE であれば，type に Type.tDouble(定数) を格納する．tm が TM_FLOAT であれば，type に Type.tFloat(定数) を格納する．tm が TM_LONG であれば，type に Type.tLong(定数) を格納する．tm が TM_CHAR であれば，type に Type.tChar(定数) を格納する．tm が TM_SHORT であれば，type に Type.tShort(定数) を格納する．tm が TM_BYTE であれば，type に Type.tByte(定数) を格納する．tm がそれ以外であれば，type に Type.tInt (定数) を格納する．

左辺式の型を type に変換するために convert メソッドを呼び出し，その戻り値を left に格納する．右辺式の型を type に変換するために convert メソッドを呼び出し，その戻り値を right に格納する．

最後に vset を返す．

エラー処理 特になし．

- public long check(Environment env, Context ctx, long vset, Hashtable exp)

機能概要 条件式の検査を行う．

機能説明 条件式の検査を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値

処理方式 cond.checkValue メソッドを呼び出し，その戻り値を vset に格納する．

cond の型を boolean 型に変換するために convert メソッドを呼び出し，戻り値を cond に格納する．

left.check メソッド，right.check メソッドを呼び出し，その戻り値の and を取り，その結果を返す．

エラー処理 特になし．

- Expression simpify()

機能概要 条件式の単純化を行う．

機能説明 cond が true または false のとき，式を単純化する．

入力データ なし．

出力データ 単純化後の式を返す．

処理方式 cond が true であれば，left を返す．cond が false であれば，right を返す．そうでなければ，this を返す．

エラー処理 特になし．

- public Expression inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式の inlining を行う．左辺式が null の場合に単純化を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 左辺式を inlining するために left.inline メソッドを呼び出し，戻り値を left に格納する．右辺式を inlining するために right.inline メソッドを呼び出し，戻り値を right に格納する．

  左辺式，右辺式がともに null なら， cond.inline メソッドを呼び出し，戻り値を返す．

  左辺式が null なら， left に right を格納し， right には null を格納する． cond に対する条件を反転するために， NotExpression を新たに生成して cond に格納する．

  cond を inlining するために， cond.inlineValue メソッドを呼び出し，戻り値を cond に格納する．最後に simplify メソッドを呼び出し，戻り値を返す．

  エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 cond.inlineValue メソッドを呼び出して戻り値を cond に格納する． left.inlineValue メソッドを呼び出して戻り値を left に格納する． right.inlineValue メソッドを呼び出して戻り値を right に格納する．最後に simplify メソッドを呼び出して戻り値を返す．

エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 条件式の inlining のコストを計算する．

  機能説明 条件式の inlining のコストを計算する．

  入力データ thresh (閾値)

  出力データ inlining のコスト．

  処理方式 cond, left, right の inlining のコストをそれぞれ costInline メソッドを呼び出して計算する．それらの総和に 1 を足したものを条件式の inlining コストとして返す．

  エラー処理 特になし．

- public Expression copyInline(Context ctx)

  機能概要 コピーされた条件式の inlining を行う．

  機能説明 条件式のコピーを作り，それに対して inlining を行った結果を返す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 clone メソッドを呼び出して，条件式のコピーを作り，ローカル変数 e に格納する．cond.inlineValue メソッドを呼び出して戻り値を e.cond に格納する．left.inlineValue メソッドを呼び出して戻り値を e.left に格納する．right.inlineValue メソッドを呼び出して戻り値を e.right に格納する．最後に e を返す．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 条件式を実行するバイトコードを生成する．

  機能説明 適切なラベルをいれて条件式を実行するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 新しいラベル l1, l2 を生成する．cond の条件が成立しなかったらラベル l1 に飛ぶバイトコードを生成するために，cond.codeBranch メソッドを呼び出す．左辺式のバイトコードを生成するために left.codeValue メソッドを呼び出す．ラベル l2 にジャンプするために opc_goto をオペコードとするバイトコードを生成する．

次に，ラベル l1 を挿入する．右辺式のバイトコードを生成するために right.codeValue メソッドを呼び出す．最後にラベル l2 を挿入する．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

機能概要 条件式を実行するバイトコードを生成する．

機能説明 適切なラベルをいれて条件式を実行するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 新しいラベル l1, l2 を生成する．cond の条件が成立しなかったらラベル l1 に飛ぶバイトコードを生成するために，cond.codeBranch メソッドを呼び出す．左辺式のバイトコードを生成するために left.code メソッドを呼び出す．

right が null でない場合，ラベル l2 にジャンプするために opc_goto をオペコードとするバイトコードを生成する．次にラベル l1 を挿入する．右辺式のバイトコードを生成するために right.code メソッドを呼び出す．最後にラベル l2 を挿入する．

right が null の場合，ラベル l1 を挿入する．

エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 条件式の中身を出力する．

  機能説明 条件式を読みやすいフォーマットで出力する．

  入力データ out (String の出力先の PrintStream)

  出力データ なし．

  処理方式 条件式を “(COND *out left right*)” という形式で出力する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 条件式の中身を出力する．

  機能説明 条件式を読みやすいフォーマットで出力する．

  入力データ なし．

  出力データ 文字列．

  処理方式 条件式を “(COND *out left right*)” という形式の文字列に変換し，返す．

  エラー処理 特になし．

4.5.43 OpenJIT.frontend.tree.ConstantExpression クラス

class ConstantExpression extends Expression

(1) 概要

このクラスは，BooleanExpression, DoubleExpression, FloatExpression, IntegerExpression, LongExpression, NullExpression, StringExpression クラスのスーパークラスであり，これらに共通のコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public ConstantExpression(int op, int where, Type type)

  機能概要 ConstantExpression オブジェクトを生成する．

  機能説明 スーパークラスの Expression クラスのコンストラクタを用いて，ConstantExpression オブジェクトを生成する．

  入力データ op (演算子の種類), where (行番号), type (定数値の型)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, type で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public boolean isConstant()

  機能概要 定数値か否かを答える．

  機能説明 定数値か否かを答える．常に True を返す．

  入力データ なし．

出力データ 真偽値 .

処理方式 True を返す .

エラー処理 特になし .

4.5.44 OpenJIT.frontend.tree.Context クラス

public class Context implements Constants

(1) 概要

このクラスは，メソッドのコンテクストを保持する機能を実現する．

(2) フィールド (変数)

- Context prev

  ネストの外側のコンテクスト．

- Node node

  ノード情報．

- int varNumber

  ローカル変数の個数．

- LocalField locals

  ローカル変数フィールド．

- FieldDefinition field

  フィールドの定義を格納する．

(3) コンストラクタ

- public Context(FieldDefinition field)

  機能概要 Context オブジェクトを生成する．

  機能説明 Context オブジェクトを生成する．field のみが与えられる場合．

  入力データ field (フィールドの定義)

  出力データ なし．

  処理方式 this.field に field を格納する．

エラー処理 特になし．

- Context(Context ctx, Node node)

  機能概要 Context オブジェクトを生成する．

  機能説明 Context オブジェクトを生成する．ctx と node が与えられる場合．

  入力データ ctx (コンテクストを格納する Context オブジェクト), node (ノード情報)

  出力データ なし．

  処理方式 this.prev に ctx, this.node に node, this.locals に ctx.locals, this.varNumber に ctx.varNumber, this.field に ctx.field を格納する．

  エラー処理 特になし．

(4) メソッド

なし．

4.5.45 OpenJIT.frontend.tree.ContinueStatement クラス

public class ContinueStatement extends Statement

(1) 概要

このクラスは，AST の continue 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Identifier lbl

  continue 文のラベル．

(3) コンストラクタ

- public ContinueStatement(int where, Identifier lbl)

  機能概要 ContinueStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), lbl (continue のラベル)

  出力データ なし．

  処理方式 super メソッドを引数 CONTINUE (定数), where で呼び出し，this.lbl に lbl を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 Continue 文の検査を行う．

  機能説明 Continue 文の検査を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 lbl で指定されたコンテクストを取得し，CheckContext destctx に格納する．destctx が null でない場合，desstctx.node.op が FOR，DO，WHILE のいずれかであれば，destctx.vsContinue & vset を destctx.vsContinue に格納し，-1 を返す．

これ以外の場合は，invalid.continue エラーを出力する．

エラー処理 lbl で与えられたコンテクストが存在しない場合に invalid.continue エラーを出力する．

- public int costInline(int thresh)

機能概要 Inlining のコストを計算する．

機能説明 Continue 文の Inlining のコストとして 1 を返す．

入力データ thresh (閾値)

出力データ Inlining のコスト．

処理方式 1 を返す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

機能概要 Continue 文を処理するバイトコードを生成する．

機能説明 Continue 文でジャンプした後の CodeContext を生成し，設定した後，Continue 文を処理するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 lbl から Continue Context を取得して，CodeContext destctx に格納する．codeFinally メソッドを呼び出す．最後に，destctx.contLabel にジャンプするために，opc_goto をオペコードとするバイトコードを生成する．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

機能概要 Continue 文を出力する．

機能説明 Continue 文を読み易くインデントを付けて出力する．

入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

出力データ 特になし．

処理方式 super.print(out, indent) を呼び出して，スーパークラスで定義された print メソッドで，必要な数のインデントを行う．“continue;” の文字列を PrintStream out に出力する．

エラー処理 特になし．

4.5.46 OpenJIT.frontend.tree.ConvertExpression クラス

public class ConvertExpression extends UnaryExpression

(1) 概要

このクラスは，型変換を行う式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public ConvertExpression(int where, Type type, Expression right)

  機能概要 ConvertExpression オブジェクトを生成する．

  機能説明 スーパークラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号)，type (Type オブジェクト)，right (式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 CONVERT (定数), where, type, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 値の検査を行う．

  機能説明 値の検査を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 right.checkValue メソッドを呼び出し，その戻り値を返す．

エラー処理 特になし．

- Expression simpify()

機能概要 式の単純化を行う．

機能説明 右辺式を type で与えられる型の式に単純化する．

入力データ なし．

出力データ 単純化後の式．

処理方式 right.op が BYTEVAL(定数), CHARVAL(定数), SHORTVAL(定数), INTVAL(定数) の場合，type.getTypeCode メソッドを呼び出し，その戻り値が，TC_BYTE であれば，新しく ByteExpression オブジェクトを生成して返す．TC_CHAR であれば，新しく CharExpression オブジェクトを生成して返す．TC_SHORT であれば，新しく ShortExpression オブジェクトを生成して返す．TC_INT であれば，新しく IntExpression オブジェクトを生成して返す．TC_LONG であれば，新しく LongExpression オブジェクトを生成して返す．TC_FLOAT であれば，新しく FloatExpression オブジェクトを生成して返す．TC_DOUBLE であれば，新しく DoubleExpression オブジェクトを生成して返す．

right.op が LONGVAL(定数), の場合，type.getTypeCode メソッドを呼び出し，その戻り値が，TC_BYTE であれば，新しく ByteExpression オブジェクトを生成して返す．TC_CHAR であれば，新しく CharExpression オブジェクトを生成して返す．TC_SHORT であれば，新しく ShortExpression オブジェクトを生成して返す．TC_INT であれば，新しく IntExpression オブジェクトを生成して返す．TC_FLOAT であれば，新しく FloatExpression オブジェクトを生成して返す．TC_DOUBLE であれば，新しく DoubleExpression オブジェクトを生成して返す．

right.op が FLOATVAL(定数), の場合， type.getTypeCode メソッドを呼び出し，その戻り値が， TC_BYTE であれば，新しく ByteExpression オブジェクトを生成して返す． TC_CHAR であれば，新しく CharExpression オブジェクトを生成して返す． TC_SHORT であれば，新しく ShortExpression オブジェクトを生成して返す． TC_INT であれば，新しく IntExpression オブジェクトを生成して返す． TC_LONG であれば，新しく LongExpression オブジェクトを生成して返す． TC_DOUBLE であれば，新しく DoubleExpression オブジェクトを生成して返す．

right.op が DOUBLEVAL(定数), の場合， type.getTypeCode メソッドを呼び出し，その戻り値が， TC_BYTE であれば，新しく ByteExpression オブジェクトを生成して返す． TC_CHAR であれば，新しく CharExpression オブジェクトを生成して返す． TC_SHORT であれば，新しく ShortExpression オブジェクトを生成して返す． TC_INT であれば，新しく IntExpression オブジェクトを生成して返す． TC_LONG であれば，新しく LongExpression オブジェクトを生成して返す． TC_FLOAT であれば，新しく FloatExpression オブジェクトを生成して返す．

それ以外の場合， this を返す．

エラー処理 特になし．

- public boolean equals(int i)

  機能概要 与えられた値と right が一致するか否か検査する．

  機能説明 与えられた値と right が一致するか否か検査して，真偽値を返す．

  入力データ i (比較する値)

  出力データ 真偽値．

  処理方式 right の値と i が一致するか否か検査して，真偽値を返す．

  エラー処理 特になし．

- public boolean equals(boolean b)

  機能概要 与えられた値と right が一致するか否か検査する．

機能説明 与えられた値とrightが一致するか否か検査して，真偽値を返す．

入力データ b (比較する値)

出力データ 真偽値．

処理方式 rightの値とbが一致するか否か検査して，真偽値を返す．

エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 型変換を行うバイトコードを生成する．

  機能説明 型変換を行うバイトコードを生成する．

  入力データ env (環境を格納するEnvironmentオブジェクト), ctx (コンテクストを格納するContextオブジェクト), asm (コード生成を行うAssemblerオブジェクト)

  出力データ なし．

  処理方式 right.codeValueメソッドを呼び出し，rightを評価してその値をスタックトップに積むバイトコードを生成する．コード変換用のバイトコードを生成するために，codeConversionメソッドを呼び出す．

  エラー処理

- public void print(PrintStream out)

  機能概要 式を読み易く出力する．

  機能説明 与えられたPrintStreamに式を適切なフォーマットで出力する．

  入力データ out (出力先のPrintStream)

  出力データ なし．

  処理方式 “(*type*) *right*” という形式の文字列を生成して，outに出力する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 式を読み易く出力する．

機能説明 式を適切なフォーマットの文字列に変換して，返す．

入力データ なし．

出力データ 文字列．

処理方式 “$($ $type$ $)$ $right$” という形式の文字列を生成して，返す．

エラー処理 特になし．

4.5.47 OpenJIT.frontend.tree.DeclarationStatement クラス

public class DeclarationStatement extends Statement

(1) 概要

このクラスは，AST の宣言文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- int mod

  モディファイア．

- Expression type

  型式．

- Statement args[]

  引数の文の配列．

(3) コンストラクタ

- public DeclarationStatement(int where, int mod, Expression type, Statement args[])

  機能概要 DeclarationStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，DeclarationStatement オブジェクトを生成する．

  入力データ where (行番号), mod (モディファイア), type (型式), args[] (引数の配列)

  出力データ なし．

  処理方式 super メソッドを引数 DECLARATION (定数), where で呼び出す．this.mod に mod, this.type に type，this.args に args を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 宣言文の検査を行う．

  機能説明

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値

  処理方式 type.toType メソッドを呼び出し，戻り値をローカル変数 Type t に格納する．args を最初から順にたどりながら，args[i].checkDeclaration メソッドを呼び出して，各宣言文の検査を行い，戻り値を vset に格納していく．

  最後に vset を返す．

  エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

  機能概要 宣言文の inlining を行う．

  機能説明 宣言文の args を順に inlining する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 int 型の変数 n を 0 に初期化する．args を最初からたどりながら，以下の処理を行う．

  args[i] を inlining するために，args[i].inline メソッドを呼び出し，戻り値を args[i] に格納する．その結果，args[i] が null でなければ，n を 1 加算する．

  最後に，n が 0 であれば，null を返し，そうでなければ，this を返す．

  エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 宣言文を実行するバイトコードを生成する．

  機能説明 宣言文の args を順に実行するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 args を最初から順にたどりながら，以下の処理を行う．

  args[i] が null でなければ，args[i] の処理を行うバイトコードを生成するために，args[i].code メソッドを呼び出す．

  エラー処理 特になし．

- public void print(PrintStream out, int indent)

  機能概要 宣言文を出力する．

  機能説明 宣言文を読みやすくインデントをつけて出力する．

  入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

  出力データ 特になし．

  処理方式 “declare ” という文字列を PrintStream out に出力する．super.print(out, indent) を呼び出して，スーパークラスで定義された print メソッドで，必要な数のインデントを行う．

  “*type args[0], args[1], args[2], ... ;*” という形式の文字列を PrintStream out に出力する．

  エラー処理 特になし．

4.5.48 OpenJIT.frontend.tree.DivRemExpression クラス

public class DivRemExpression extends BinaryArithmeticExpression

(1) 概要

このクラスは，DivideExpression, RemainderExpression クラスのスーパークラスであり，これらに共通するコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public DivRemExpression(int op, int where, Expression left, Expression right)

  機能概要 DivRemExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryArithmeticExpression クラスで定義されているコンストラクタを用いて，式ノードの生成・初期化を行う．

  入力データ op (演算子の種類), where (行番号)，left (式の左辺式の Expression オブジェクト)，right (式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- public Expression inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 剰余算の式を inlining して，その結果を返す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 左辺式，右辺式がともに byte, int, short, long 型 (整数型) の場合，右辺式を inlining にしてその結果を right に代入する．right が定数かつ非零のとき，左辺式を inlining し，その結果の式を返す．right が不定または零のとき，左辺式を inlining してその結果を left に代入する．この式を eval メソッドで評価した式オブジェクトに対して simply メソッドを呼び，0 除算が起きれば，エラーが出力される．

右辺式または左辺式が整数型でない場合，スーパークラスの inline メソッドを呼び，その結果の式を返す．

エラー処理 inlining の結果，0 除算が行われた時はエラーを出力する．

4.5.49 OpenJIT.frontend.tree.DivideExpression クラス

public class DivideExpression extends DivRemExpression

(1) 概要

このクラスは，AST の / 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public DivideExpression(int where, Expression left, Expression right)

  機能概要 AST の / 式ノードを生成する．

  機能説明 スーパークラスの DivRemExpression クラスで定義されているコンストラクタを用いて，/ 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (/ 式の左辺式の Expression オブジェクト)，right (/ 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 DIV(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数の除算を行い，IntExpression オブジェクトを生成する．

  入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理内容 a/b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数の除算を行い，LongExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a/b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(float a, float b)

  機能概要 式を評価する．引数がともに float 型の場合．

  機能説明 float 型の引数の除算を行い，FloatExpression オブジェクトを生成する．

  入力データ a (float 型), b (float 型)

  出力データ Expression オブジェクト．

  処理内容 a/b を計算し，FloatExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数がともに double 型の場合．

機能説明 double 型の引数の除算を行い，DoubleExpression オブジェクトを生成する．

入力データ a (double 型), b (double 型)

出力データ Expression オブジェクト．

処理内容 a/b を計算し，DoubleExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression simpify()

機能概要 式の単純化を行う．

機能説明 右辺式が 0 または 1 の場合，式を単純化する．

入力データ なし．

出力データ Expression オブジェクト．

処理方式 右辺式が 0 の場合，0 除算のエラーを出力する．右辺式が 1 の場合，左辺式をこの式の値として返す．それ以外の場合，この式を返す．

エラー処理 0 除算が起きた時，ArithmeticException を投げる．

- void codeOperation(Environment env, Context ctx, Assembler asm)

機能概要 / を行うバイトコードを生成する．

機能説明 各型ごとに適切な / 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_idiv (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 / 式の中身を出力する．

  機能説明 / 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.50 OpenJIT.frontend.tree.DoStatement クラス

public class DoStatement extends Statement

(1) 概要

このクラスは，AST の Do 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Statement body

  do 文の本体．

- Expression cond

  do 文の条件式．

(3) コンストラクタ

- public DoStatement(int where, Statement body, Expression cond)

  機能概要 DoStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), body (Statement オブジェクト), cond (条件式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 DO (定数), where で呼び出し，this.body に body，this.cond に cond を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 Do 文の検査を行う．

  機能説明 Do 文の検査を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値．

  処理方式 新しく CheckContext オブジェクトを生成し，これを newctx に格納する．body を検査するために body.check メソッドを呼び出す．

  body を通過するか，body の途中で continue 文に到達したら，cond に落ちるのでその条件を検査する．cond.checkCondition メソッドを呼び出して，その戻り値を ConditionVars cvars に格納する．cond の型を boolean 型に変換するために，covert メソッドを呼び出す．

  cond を評価した結果 false になるか，body の途中で break 文に到達したら，ループから抜けるので，その条件を検査する．newctx.vxBreak & cvars.vsFalse を vset に格納し，コンテクストから vset を取り除くために ctx.removeAddionalVars メソッドを呼び出す．その戻り値を返す．

  エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

  機能概要 Do 文の inlining を行う．

  機能説明 body, cond の inlining を行うことで，Do 文の inlining を行って，inlining 後の文を返す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ Inlining 後の文オブジェクト．

処理方式 body の inlining を行うために，まず，body が null か否か検査する．body が null でなければ，body.inline メソッドを呼び出し，その結果で body を更新する．cond の inlining を行うために cond.inlineValue メソッドを呼び出して，その結果で cond を更新する．最後に this を返す．

エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

  機能概要 Do 文のコピーを作り，inlining を行う．

  機能説明 Do 文のコピーを作り，body, cond の inlining を行うことで，Do 文の inlining を行って，inlining 後の文を返す．

  入力データ ctx (コンテクストを格納する Context オブジェクト), valNeeded (計算後の値をスタック上に残すかどうかを指定するフラグ)

  出力データ Inlining 後の文オブジェクト．

  処理方式 clone メソッドを呼び出して Do 文のコピーを作り，これを s に格納する．body の inlining を行うために，まず，body が null か否か検査する．body が null でなければ，body.inline メソッドを呼び出し，その結果で s.body を更新する．cond の inlining を行うために cond.inlineValue メソッドを呼び出して，その結果で s.cond を更新する．最後に s を返す．

  エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 Inlining のコストを計算する．

  機能説明 Do 文の Inlining のコストとして，条件式とボディのそれぞれの inlining コストの和を計算して返す．

  入力データ thresh (閾値)．

  出力データ Inlining のコスト．

  処理方式 cond.costInline(thresh) + body.costInline(thresh) + 1 を計算して，返す．

  エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 Do 文を実行するバイトコードを生成する．

  機能説明 Do 文を実行するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 新しいラベル l1 を生成し，ラベルを挿入する．現在の CodeContext を生成して newctx に格納する． body が null でなければ， body.code メソッドを呼び出して， body 部を実行するバイトコードを生成する． newctx.contLabel を挿入する． cond 式を評価して，それが成立すれば l1 にジャンプするコードを生成するために， cond.codeBranch メソッドを呼び出す．最後に， newctx.breakLabel を挿入する．

  エラー処理 特になし．

- public void print(PrintStream out, int indent)

  機能概要 Do 文の中身を出力する．

  機能説明 Do 文の中身を読み易く文字列で， PrintStream out に出力する．

  入力データ out (String の出力先の PrintStream)， indent (インデントの深さ)

  出力データ なし．

  処理方式 “do *body* while *cond*;” という形式で出力できるように，まず， “do ” を out に出力し， body.print メソッドを呼び出し， “while ” を out に出力し， cond.print メソッドを呼び出す．

  エラー処理 特になし．

4.5.51 OpenJIT.frontend.tree.DoubleExpression クラス

public class DoubleExpression extends ConstantExpression

(1) 概要

このクラスは，AST の double 型浮動少数点数ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- double value

  浮動少数点数値を保持する

(3) コンストラクタ

- public DoubleExpression(int where, double value)

  機能概要 DoubleExpression オブジェクトを生成する．

  機能説明 スーパークラスの ConstantExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), value (保持する値)

  出力データ なし．

  処理方式 super メソッドを引数 DOUBLEVAL(定数), where, Type.tDouble (定数) で呼び出し，this.value に value を代入する．

  エラー処理 特になし．

(4) メソッド

- public Object getValue()

  機能概要 値を取り出す．

  機能説明 value を保持する Double オブジェクトを返す．

入力データ なし．

出力データ 数値オブジェクト

処理方式 value を保持する Double オブジェクトを生成して，返す．

エラー処理 特になし．

- public boolean equals(int i)

  機能概要 値が与えられた整数値と一致するか否かを検査する．

  機能説明 値 value が与えられた整数値と一致するか否かを検査する．

  入力データ i (比較する値)

  出力データ 一致・不一致の真偽値

  処理方式 value==i の結果を返す．

  エラー処理 特になし．

- public boolean equalsDefault()

  機能概要 値が 0 と一致するか否かを検査する．

  機能説明 値 value が 0 と一致するか否かを検査する．

  入力データ なし．

  出力データ 一致・不一致の真偽値

  処理方式 value==0 の結果を返す．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 浮動少数点数値をスタックトップに積むバイトコードを生成する．

  機能説明 浮動少数点数値 value をスタックトップに積むバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 スタックトップに value を積むために，opc_ldc2_w をオペコードとするバイトコードを生成する．

エラー処理 なし．

- public void print(PrintStream out)

  機能概要 値を読み易く出力する．

  機能説明 与えられた PrintStream に value を十進数表現で出力する．

  入力データ out (出力先の PrintStream)

  出力データ なし．

  処理方式 value の十進数表現に D を付けて，out.print メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 浮動少数点数値の中身を出力する．

  機能説明 浮動少数点数値 value を十進数で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 Double.toString メソッドを呼び出して，value を文字列に変換し，文字列 “D” と join して返す．

  エラー処理 特になし．

4.5.52 OpenJIT.frontend.tree.EqualExpression クラス

public class EqualExpression extends BinaryEqualityExpression

(1) 概要

このクラスは，AST の == 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public EqualExpression(int where, Expression left, Expression right)

  機能概要 AST の == 式ノードを生成する．

  機能説明 スーパークラスの BinaryEqualityExpression クラスで定義されているコンストラクタを用いて，== 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (== 式の左辺式の Expression オブジェクト)，right (== 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 EQ(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数で == 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理内容 a==b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数で == 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a==b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(float a, float b)

  機能概要 式を評価する．引数がともに float 型の場合．

  機能説明 float 型の引数で == 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (float 型), b (float 型)

  出力データ Expression オブジェクト．

  処理内容 a==b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数がともに double 型の場合．

機能説明 double 型の引数で == 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (double 型), b (double 型)

出力データ Expression オブジェクト．

処理内容 a==b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(boolean a, boolean b)

機能概要 式を評価する．引数がともに boolean 型の場合．

機能説明 boolean 型の引数で == 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (boolean 型), b (boolean 型)

出力データ Expression オブジェクト．

処理内容 a==b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression simpify()

機能概要 式の単純化を行う．

機能説明 左辺式が定数で，右辺式が定数でない場合に式の単純化を行う．

入力データ なし．

出力データ Expression オブジェクト．

処理方式 左辺式が定数で，右辺式が定数でない場合，右辺式と左辺式を入れ換えた EqualExpression を生成して返す．そうでない場合， this を返す．

エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

機能概要 == 式の条件分岐を行うバイトコードを生成する．

機能説明 各型ごとに適切な条件分岐用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

出力データ なし．

処理方式 1. left.codeValue メソッドを呼び出し，左辺式をスタックトップに積むバイトコードを生成する．

2. 左辺式の型を検査し，boolean または int 型の場合は，右辺式が 0 であれば 7 へ．非零であれば，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．さらに whenTrue が True/False に従って，opc_if_icmpeq, opc_if_icmpne をオペコードとするバイトコードを生成して終了．
3. 左辺式の型が long の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_lcmp をオペコードとするバイトコードを生成する．7 へ．
4. 左辺式の型が float の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_fcmpl をオペコードとするバイトコードを生成する．7 へ．
5. 左辺式の型が double の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_dcmpl をオペコードとするバイトコードを生成する．7 へ．
6. 左辺式の型が，クラスオブジェクト，配列オブジェクトの場合，右辺式が 0 であれば，whenTrue の True/False に従って，opc_ifnull, opc_ifnonnull をオペコードとするバイトコードを生成して，7 へ．非零であれば，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．さらに whenTrue の True/False に従って，opc_if_acmpeq, opc_if_acmpne をオペコードとするバイト

コードを生成して，7へ．

7. whenTrue の True/False に従って，opc_ifeq, opc_ifne をオペコードとするバイトコードを生成して終了．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 == 式の中身を出力する．

  機能説明 == 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.53 OpenJIT.frontend.tree.ExprExpression クラス

public class ExprExpression extends UnaryExpression

(1) 概要

このクラスは，AST の () で囲まれた式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public ExprExpression(int where, Expression right)

  機能概要 ExprExpression オブジェクトを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されているコンストラクタを呼び出して，ExprExpression オブジェクトを生成・初期化する．

  入力データ where (行番号), right (右辺式の Expression)

  出力データ なし．

  処理方式 super メソッドを引数 EXPR(定数), where, right.type, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public void checkCondition(Environment env, Context ctx, long vset, Hashtable exp, ConditionVars cvars)

  機能概要 式の条件を検査する．

  機能説明 右辺式の検査を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), cvars (条件の評価値オブジェクト)

出力データ なし．

処理方式 right.checkCondition メソッドを呼び出す． type に right.type を格納する．

エラー処理 特になし．

- void selectType(Environment env, Context ctx, int tm)

機能概要 式の型を選択する．

機能説明 type を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 type に right.type を格納する．

エラー処理 特になし．

- Expression simpify()

機能概要 式の単純化を行う．

機能説明 右辺式に単純化する．

入力データ なし．

出力データ 単純化された式．

処理方式 right を返す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 () 式の中身を出力する．

機能説明 () 式の中身を読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 “($right$)” という形式の文字列を生成して，返す．

エラー処理 特になし．

4.5.54 OpenJIT.frontend.tree.Expression クラス

public class Expression extends Node

(1) 概要

このクラスは，AST のすべての式のノードのスーパークラスであり，式の評価に関係するコード生成の機能を実現する．

(2) フィールド (変数)

- public Type type

  式の型．

(3) コンストラクタ

- Expression(int op, int where, Type type)

  機能概要 Expression オブジェクトを生成する．

  機能説明 スーパークラスの Node クラスで定義されているコンストラクタを用いて，Expression オブジェクトの生成・初期化を行う．

  入力データ op (オペレータ), where (行番号), type (型)

  出力データ なし．

  処理方式 super メソッドを引数 op, where で呼び出す．this.type に type を格納する．

  エラー処理 特になし．

(4) メソッド

- int precedence()

  機能概要 オペレータの優先順位を求める．

  機能説明 opPrecedence[] で定義されている優先順位を返す．

入力データ なし．

出力データ 優先順位．

処理方式 op が opPrecedenc.length より小さい場合，opPrecedence[op] を返す．
そうでない場合，100 を返す．

エラー処理 特になし．

- public Expression order()

機能概要 式を演算子の優先順位に従って並び換える．

機能説明 式を演算子の優先順位に従って並び換える．この場合は，不要．

入力データ なし．

出力データ 並び換え後の Expression．

処理方式 this を返す．

エラー処理 特になし．

- public boolean isConstant()

機能概要 定数か否かを答える．

機能説明 定数か否かを答える．この場合は，常に false．

入力データ なし．

出力データ 真偽値．

処理方式 false を返す．

エラー処理 特になし．

- public Object getValue()

機能概要 value のアクセッサメソッド．

機能説明 value のアクセッサメソッド．この場合は，常に null．

入力データ なし．

出力データ オブジェクト．

処理方式 null を返す．

エラー処理 特になし．

- public boolean equals(int i)

  機能概要 式が与えられた値と等しいか否かを検査する．

  機能説明 式が与えられた値 (int 型) と等しいか否かを検査する．この場合は，常に false．

  入力データ i (int 型)

  出力データ 真偽値．

  処理方式 false を返す．

  エラー処理 特になし．

- public boolean equals(boolean b)

  機能概要 式が与えられた値と等しいか否かを検査する．

  機能説明 式が与えられた値 (boolean 型) と等しいか否かを検査する．この場合は，常に false．

  入力データ b (boolean 型)

  出力データ 真偽値．

  処理方式 false を返す．

  エラー処理 特になし．

- public boolean equals(Identifier id)

  機能概要 式が与えられた値と等しいか否かを検査する．

  機能説明 式が与えられた値 (Indentifier) と等しいか否かを検査する．この場合は，常に false．

  入力データ id (Identifier オブジェクト)

  出力データ 真偽値．

処理方式 false を返す．

エラー処理 特になし．

- public boolean equalsDefault()

  機能概要 式が既定の静的な値 (0) と等しいか否かを検査する．

  機能説明 式が既定の静的な値 (0) と等しいか否かを検査する．この場合は，常に false．

  入力データ なし．

  出力データ 真偽値．

  処理方式 false を返す．

  エラー処理 特になし．

- Type toType(Environment env, Context ctx)

  機能概要 式を型に変換する．

  機能説明 式を型に変換する．この場合は，変換できないため，invalid.type.expr エラーを出力する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ 型．

  処理方式 invalid.type.expr エラーを出力し，Type.tError (定数) を返す．

  エラー処理 特になし．

- public boolean fitsType(Environment env, Type t)

  機能概要 式が与えられた型に変換可能かどうかを検査する．

  機能説明 式が与えられた型に変換可能かどうかを検査する．式の型と与えられた型のクラスの階層情報から決定する．

  入力データ env (環境を格納する Environment オブジェクト), t (型)

  出力データ 真偽値．

処理方式 env.isMoreSpecific メソッドを呼び出し，this.type が t と同じタイプか，あるいはサブクラスタイプであれば，true を返す．そうでなければ，false を返す．

エラー処理 特になし．

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の値を検査する．

  機能説明 式の値を検査する．この場合は，vset を返すのみ．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値．

  処理方式 vset を返す．

  エラー処理 特になし．

- public long checkInitializer(Environment env, Context ctx, long vset, Type t, Hashtable exp)

  機能概要 式の初期化の検査を行う．

  機能説明 式の初期化の検査を行うために，checkValue メソッドを呼び出す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), t (エレメントタイプ), exp (ハッシュテーブル)

  出力データ 条件の評価値．

  処理方式 checkValue メソッドを呼び出し，戻り値を返す．

  エラー処理 特になし．

- public long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の検査を行う．

機能説明 式の検査を行う．サブクラスでこのメソッドが定義されていない場合，エラーを出力する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値

処理方式 CompilerError("check failed") を throw する．

エラー処理 無条件で CompilerError("check failed") を throw する．

- public long checkLHS(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 左辺式を検査する．

  機能説明 左辺式を検査する．サブクラスでこのメソッドが定義されていない場合は，エラーを出力する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値

  処理方式 invalid.lhs.assignment エラーを出力する． type に Type.tError (定数) を格納する． vset を返す．

  エラー処理 無条件で invalid.lhs.assignment エラーを出力する．

- public long checkAssignOp(Environment env, Context ctx, long vset, Hashtable exp, Expression outside)

  機能概要 代入演算式の検査を行う．

  機能説明 代入演算式の検査を行う．サブクラスでこのメソッドが定義されていない場合は，エラーを出力する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), outside (右辺式の Expression)

出力データ 条件の評価値．

処理方式 outside が IncDecExpression のインスタンスでなければ，invalid.arg エラーを出力する．そうでなければ，invalid.lhs.assignment エラーを出力する．type に Type.tError(定数) を格納し，vset を返す．

エラー処理 outside が IncDecExpression のインスタンスなら，invalid.lhs.assignment エラー，そうでなければ，invalid.arg エラーを出力する．

- public ConditionVars checkCondition(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の条件を検査する．

  機能説明 式の条件を検査して，ConditionVars オブジェクトを返す．ConditionVars オブジェクトは，条件が true になるときの変数の集合，および条件が false になるときの変数の集合を表す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), cvars (条件の評価値オブジェクト)

  出力データ 条件の評価値オブジェクト．

  処理方式 新しく ConditionVars オブジェクト cvars を生成し，checkCondition メソッドを呼び出す．cvars を返す．

  エラー処理 特になし．

- public void checkCondition(Environment env, Context ctx, long vset, Hashtable exp, ConditionVars cvars)

  機能概要 式の条件を検査する．

  機能説明 式の条件を検査する．式の値の検査を行い，その結果を cvars の vsTrue, vsFalse の両方に格納する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), cvars (条件の評価値オブジェクト)

出力データ なし．

処理方式 checkValue メソッドを呼び出し，その戻り値を cvars.vsTrue, cvars.vsFalse に格納する．

エラー処理 特になし．

- Expression eval()

  機能概要 式を評価する．

  機能説明 式を評価する．

  入力データ なし．

  出力データ 評価後の式．

  処理方式 this を返す．

  エラー処理 特になし．

- Expression simplify()

  機能概要 式の単純化を行う．

  機能説明 式の単純化を行う．

  入力データ なし．

  出力データ 単純化を行った式．

  処理方式 this を返す．

  エラー処理 特になし．

- public Expression inline(Environment env, Context ctx)

  機能概要 式の右辺値の inlining を行う．

  機能説明 式の右辺値の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

処理方式 null を返す．

エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 this を返す．

  エラー処理 特になし．

- public Expression inlineLHS(Environment env, Context ctx)

  機能概要 式の左辺値の inlining を行う．

  機能説明 式の左辺値の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 null を返す．

  エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 式の inlining のコストを求める．

  機能説明 式の既定の inlining のコストを 1 として，1 を返す．

  入力データ thresh (閾値)

  出力データ inlining のコスト．

  処理方式 1 を返す．

エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

  機能概要 分岐命令のバイトコードを生成する．

  機能説明 分岐命令のバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

  出力データ なし．

  処理方式 type が boolean 型でなければ，CompilerError(“codeBranch”) を throw する．

  条件式の値を計算し，スタックトップに積むバイトコードを生成するために，codeValue メソッドを呼び出す．whenTrue が true なら，opc_ifne をオペコードとするバイトコードを生成する．false なら，opc_ifeq をオペコードとするバイトコードを生成する．

  エラー処理 type が boolean 型でなければ，CompilerError(“codeBranch”) を throw する．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 式の値を計算し，スタックトップに積むバイトコードを生成する．

  機能説明 式の値を計算し，スタックトップに積むバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 type が booleann 型の場合，2 つの新しいラベル l1, l2 を生成する．

条件が成立すれば，l1 にジャンプするバイトコードを生成するために，codeBranch メソッドを呼び出す．

次に，opc_ldc をオペコードとするバイトコードを生成し，スタック上に値 0 を積み，l2 に無条件ジャンプするために opc_goto をオペコードとするバイトコードを生成する．次に，ラベル l1 を挿入し，opc_ldc をオペコードとするバイトコードを生成し，スタック上に値 1 を積み，ラベル l2 を挿入する．

type が boolean 型でない場合，CompilerError(“codeValue”) を throw する．

エラー処理 type が boolean 型でない場合，CompilerError(“codeValue”) を throw する．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 式を実行するバイトコードを生成する．

  機能説明 式を実行するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 式の計算値をスタックに積むために，codeValue メソッドを呼び出す．

  メソッド type.getType() の戻り値が，TC_VOID (定数) の場合，終了．

  メソッド type.getType() の戻り値が，TC_DOUBLE (定数)，または TC_LONG の場合，opc_pop2 をオペコードとするバイトコードを生成して，終了．

  メソッド type.getType() の戻り値が，それ以外の場合，opc_pop をオペコードとするバイトコードを生成して，終了．

  エラー処理 特になし．

- int codeLValue(Environment env, Context ctx, Assembler asm)

  機能概要 式が左辺式として使われる場合のバイトコードを生成する．

機能説明 式が左辺式として使われる場合のバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 CompilerError(“invalid lhs”) を throw する．

エラー処理 無条件に，CompilerError(“invalid lhs”) を throw する．

- void codeLoad(Environment env, Context ctx, Assembler asm)

機能概要 式が右辺式として使われ，ロードするバイトコードを生成する．

機能説明 式が右辺式として使われ，ロードするバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 CompilerError(“invalid load”) を throw する．

エラー処理 無条件で，CompilerError(“invalid load”) を throw する．

- void codeStore(Environment env, Context ctx, Assembler asm)

機能概要 式が左辺式として使われ，ストアするバイトコードを生成する．

機能説明 式が左辺式として使われ，ストアするバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 CompilerError(“invalid store”) を throw する．

エラー処理 無条件で，CompilerError(“invalid store”) を throw する．

- void codeAppend(Environment env, Context ctx, Assembler asm, ClassDeclaration c)

  機能概要 バイトコードを追加する．

  機能概要 補助的なバイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), c (ClassDeclaration オブジェクト)

  出力データ なし．

  処理方式 式の計算値がスタックに積まれるように，codeValue メソッドを呼び出す．opc_invokevirtual をオペコードとするバイトコードを生成する．

  エラー処理 特になし．

- void codeDup(Environment env, Context ctx, Assembler asm, int items, int depth)

  機能概要 スタック上のデータの複製を行うバイトコードを生成する．

  機能説明 スタックの depth の深さの位置にある，items 個のデータを複製するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), items (コピーするアイテムの数), depth (コピーするアイテムのスタックの深さ)

  出力データ なし．

  処理方式 items が 0 の場合，終了．

  items が 1 の場合，depth が 0 なら，opc_dup をオペコードとするバイトコードを生成して，終了．depth が 1 なら，opc_dup_x1 をオペコードとするバイトコードを生成して，終了．depth が 2 なら，opc_dup_x2 をオペコードとするバイトコードを生成して，終了．

  items が 2 の場合，depth が 0 なら，opc_dup2 をオペコードとするバイトコードを生成して，終了．depth が 1 なら，opc_dup2_x1 をオペコードとす

るバイトコードを生成して，終了．depth が 0 なら，opc_dup2_x2 をオペコードとするバイトコードを生成して，終了．

これ以外の場合は，CompilerError(“can’t dup: *items*, *depth*”) を throw する．

エラー処理 複製できない items, depth のペアの場合は，CompilerError(“can’t dup: *items*, *depth*”) を throw する．

- void codeConversion(Environment env, Context ctx, Assembler asm, Type f, Type t)

  機能概要 型変換用のバイトコードを生成する．

  機能説明 引数で与えられた型 f から型 t への変換を行うバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), f (変換前の型), t (変換後の型)

  出力データ なし．

  処理方式 f.getTypeCode メソッドを呼び出し，戻り値を from に格納する．

  t.getTypeCode メソッドを呼び出し，戻り値を to に格納する．

  to の値に従って，以下の処理を行う．

  - to が TC_BOOLEAN (定数) に等しい場合，from が TC_BOOLEAN (定数) に等しければ，終了．そうでなければ，“CompilerError(”codeConversion: ” + from + ”, ” + to)” を throw する．
  - to が TC_BYTE (定数) に等しい場合，from が TC_BYTE (定数) に等しければ，終了．そうでなければ，codeConversion(env, ctx, asm, f, Type.tInt) を呼び出し，opc_int2byte をオペコードとするバイトコードを生成して終了．
  - to が TC_SHORT (定数) に等しい場合，from が TC_SHORT (定数) に等しければ，終了．そうでなければ，codeConversion(env, ctx, asm, f, Type.tInt) を呼び出し，opc_int2short をオペコードとするバイトコードを生成して終了．

- to が TC_INT (定数) に等しい場合，from の値に従って，以下の処理を行う．
  - from が TC_BYTE (定数), TC_CHAR (定数), TC_SHORT (定数), TC_INT (定数) のいずれかであれば，終了．
  - from が TC_LONG (定数) であれば，opc_l2i をオペコードとするバイトコードを生成して終了．
  - from が TC_FLOAT (定数) であれば，opc_f2i をオペコードとするバイトコードを生成して終了．
  - from が TC_DOUBLE (定数) であれば，opc_d2i をオペコードとするバイトコードを生成して終了．
  - これ以外の場合，“CompilerError(”codeConversion: ” + from + ”, ” + to)” を throw する．
- to が TC_LONG (定数) に等しい場合，from の値に従って，以下の処理を行う．
  - from が TC_BYTE (定数), TC_CHAR (定数), TC_SHORT (定数), TC_INT (定数) のいずれかであれば，opc_i2l をオペコードとするバイトコードを生成して，終了．
  - from が TC_LONG (定数) であれば，終了．
  - from が TC_FLOAT (定数) であれば，opc_f2l をオペコードとするバイトコードを生成して終了．
  - from が TC_DOUBLE (定数) であれば，opc_d2l をオペコードとするバイトコードを生成して終了．
  - これ以外の場合，“CompilerError(”codeConversion: ” + from + ”, ” + to)” を throw する．
- to が TC_FLOAT (定数) に等しい場合，from の値に従って，以下の処理を行う．
  - from が TC_BYTE (定数), TC_CHAR (定数), TC_SHORT (定数), TC_INT (定数) のいずれかであれば，opc_i2f をオペコードとするバイトコードを生成して，終了．

    * from が TC_LONG (定数) であれば，opc_l2f をオペコードとするバイトコードを生成して終了．
    * from が TC_FLOAT (定数) であれば，終了．
    * from が TC_DOUBLE (定数) であれば，opc_d2f をオペコードとするバイトコードを生成して終了．
    * これ以外の場合，“CompilerError(”codeConversion: ” + from + ”, ” + to)” を throw する．
  - to が TC_DOUBLE (定数) に等しい場合，from の値に従って，以下の処理を行う．
    * from が TC_BYTE (定数), TC_CHAR (定数), TC_SHORT (定数), TC_INT (定数) のいずれかであれば，opc_i2d をオペコードとするバイトコードを生成して，終了．
    * from が TC_LONG (定数) であれば，opc_l2d をオペコードとするバイトコードを生成して終了．
    * from が TC_FLOAT (定数) であれば，opc_f2d をオペコードとするバイトコードを生成して終了．
    * from が TC_DOUBLE (定数) であれば，終了．
    * これ以外の場合，“CompilerError(”codeConversion: ” + from + ”, ” + to)” を throw する．
  - to が TC_CLASS (定数) に等しい場合，from の値に従って，以下の処理を行う．
    * from が TC_NULL (定数) であれば，終了．
    * from が TC_CLASS (定数), TC_ARRAY (定数) であれば，型 f から型 t への暗黙のキャストが有効か否かを検査するために，env.implicitCast(f, t) を呼び出し，true なら終了．false なら，実行時に検査を行うために，opc_checkcast をオペコードとするバイトコードを生成する．
    * これ以外の場合，“CompilerError(”codeConversion: ” + from + ”, ” + to)” を throw する．

– to が TC_ARRAY (定数) に等しい場合，from の値に従って，以下の処理を行う．

* from が TC_NULL (定数) であれば，終了．
* from が TC_CLASS (定数), TC_ARRAY (定数) であれば，型 f から型 t への暗黙のキャストが有効か否かを検査するために，env.implicitCast(f, t) を呼び出し，true なら終了．false なら，実行時に検査を行うために，opc_checkcast をオペコードとするバイトコードを生成する．
* これ以外の場合，“CompilerError(”codeConversion: ” + from + ”, ” + to)” を throw する．

エラー処理 型変換できないすべての場合において，“CompilerError(”codeConversion: ” + from + ”, ” + to)” を throw する．

- public boolean firstConstructor()

  機能概要 式中，最初に行うことがコンストラクタの呼び出しか否かを検査する．

  機能説明 式中，最初に行うことがコンストラクタの呼び出しか否かを検査する．この場合は，false を返す．

  入力データ なし．

  出力データ 真偽値．

  処理方式 false を返す．

  エラー処理 特になし．

- public Expression copyInline(Context ctx)

  機能概要 コピーされた式の inlining を行う．

  機能説明 コピーされた式の inlining を行う．

  入力データ ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

処理方式 clone メソッドを呼び出し，戻り値を返す．

エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 式を読み易く出力する．

  機能説明 与えられた PrintStream に式のオペコードを文字列で出力する．

  入力データ out (出力先の PrintStream)

  出力データ なし．

  処理方式 opNames[op] を PrintStream out に出力する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 式を読み易く出力する．

  機能説明 式のオペコードを文字列に変換して，返す．

  入力データ なし．

  出力データ 文字列．

  処理方式 “(Expression{ *type* })” という形式の文字列を生成して，返す．

  エラー処理 特になし．

4.5.55 OpenJIT.frontend.tree.ExpressionStatement クラス

public class ExpressionStatement extends Statement

(1) 概要

このクラスは，AST の式で文として機能するもののノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Expression expr

  文として機能する式．

(3) コンストラクタ

- public ExpressionStatement(int where, Expression expr)

  機能概要 ExpressionStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), expr (式オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 EXPRESSION(定数), where で呼び出す．this.expr に expr を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の検査を行う．

  機能説明 式 expr の検査を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 expr.check メソッドを呼び出し，戻り値を返す．

エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式 expr の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の文．

  処理方式 expr が null でなければ，expr.inline メソッドを呼び出し，戻り値を expr に格納する．expr が null であれば，null を返し，そうでなければ this を返す．

  それ以外の場合は null を返す．

  エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

  機能概要 コピーされた式の inlining を行う．

  機能説明 式 expr のコピーを作り，inlining を行う．

  入力データ ctx (コンテクストを格納する Context オブジェクト), valNeeded (計算後の値をスタック上に残すかどうかを指定するフラグ)

  出力データ inlining 後の文．

  処理方式 clone メソッドを呼び出してコピーを作り，s に格納する．expr.copyInline メソッドを呼び出し，戻り値を s.expr に格納する．s を返す．

  エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 inlining のコストを求める．

  機能説明 式 expr の inlining のコストを求め，inlining コストを求める．

  入力データ thresh (閾値)

  出力データ inlining のコスト．

  処理方式 expr.costInline メソッドを呼び出し，戻り値を返す．

  エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 式を実行するバイトコードを生成する．

  機能説明 式 expr を実行するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 expr.code メソッドを呼び出す．

  エラー処理 特になし．

- public boolean firstConstructor()

  機能概要 expr の最初がコンストラクタ呼び出しか否か検査する．

  機能説明 expr の最初がコンストラクタ呼び出しか否か検査する．

  入力データ なし．

  出力データ 真偽値．

  処理方式 expr.fistConstructor メソッドを呼び出し，戻り値を返す．

  エラー処理 特になし．

- public void print(PrintStream out, int indent)

機能概要 文を出力する．

機能説明 文を読みやすくインデントを付けて出力する．

入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

出力データ 特になし．

処理方式 super.print(out, indent) を呼び出して，スーパークラスで定義された print メソッドで，必要な数のインデントを行う．expr が null でなければ，expr.print メソッドを呼び出し，expr を文字列にしたものを出力する．それ以外の場合，“<empty>” の文字列を out に出力する．最後に “;” を出力する．

エラー処理 特になし．

4.5.56 OpenJIT.frontend.tree.FieldExpression クラス

public class FieldExpression extends UnaryExpression

(1) 概要

このクラスは，“object.field” のように，フィールドを指すオペレータのノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Identifier id

  変数フィールドの識別子．

- FieldDefinition field

  フィールドの定義．

- boolean parsed = false

  真偽値．

(3) コンストラクタ

- public FieldExpression(int where, Expression right, Identifier id)

  機能概要 FieldExpression オブジェクトを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されているコンストラクタを用いて，FieldExpression オブジェクトを生成する．FieldExpression オブジェクトを生成する．right と id が与えられる場合．

  入力データ where (行番号), right (右辺式), id (識別子)

  出力データ なし．

  処理方式 super メソッドを引数 FIELD (定数), where，Type.tError (定数), right で呼び出す．this.id に id を格納する．

  エラー処理 特になし．

- public FieldExpression(int where, Context ctx, FieldDefinition field)

  機能概要 FieldExpression オブジェクトを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されているコンストラクタを用いて，FieldExpression オブジェクトを生成する．FieldExpression オブジェクトを生成する．ctx と field が与えられる場合．

  入力データ where (行番号), ctx (コンテクストを格納する Context オブジェクト), field (フィールドの定義)

  出力データ なし．

  処理方式 super メソッドを引数 FIELD (定数), where，field.getType(), field.isStatic() が true なら null, false なら新しく ThisExpression オブジェクトを生成して，呼び出す．this.id に field.getName(), this.filed に field を格納する．

  エラー処理 特になし．

(4) メソッド

なし．

4.5.57 OpenJIT.frontend.tree.FinallyStatement クラス

public class FinallyStatement extends Statement

(1) 概要

このクラスは，AST の Finally 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Statement body

  本体の文．

- Statement finalbody

  finally ブロック本体の文．

- boolean finallyFinishes

  finalbody が return しないかどうかのフラグ．

(3) コンストラクタ

- public FinallyStatement(int where, Statement body, Statement finalbody)

  機能概要 FinallyStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，FinallyStatement オブジェクトを生成・初期化する．

  入力データ where (行番号), body (本体の文), finalbody (finally ブロックの文)

  出力データ なし．

  処理方式 super メソッドを引数 FINALLY (定数), where で呼び出す．this.body に body, this.finalbody に finalbody を格納する．

  エラー処理 特になし．

(4)　メソッド

なし．

4.5.58 OpenJIT.frontend.tree.FloatExpression クラス

public class FloatExpression extends ConstantExpression

(1) 概要

このクラスは，AST の long 型浮動少数点数ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- float value

  浮動少数点数値を保持する

(3) コンストラクタ

- public FloatExpression(int where, long value)

  機能概要 Eloatxpression オブジェクトを生成する．

  機能説明 スーパークラスの ConstantExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), value (保持する値)

  出力データ なし．

  処理方式 super メソッドを引数 FLOATVAL(定数), where, Type.tFloat (定数) で呼び出し，this.value に value を代入する．

  エラー処理 特になし．

(4) メソッド

- public Object getValue()

  機能概要 値を取り出す．

  機能説明 value を保持する Float オブジェクトを返す．

入力データ なし．

出力データ 数値オブジェクト

処理方式 value を保持する Float オブジェクトを生成して，返す．

エラー処理 特になし．

- public boolean equals(int i)

  機能概要 値が与えられた整数値と一致するか否かを検査する．

  機能説明 値 value が与えられた整数値と一致するか否かを検査する．

  入力データ i (比較する値)

  出力データ 一致・不一致の真偽値

  処理方式 value==i の結果を返す．

  エラー処理 特になし．

- public boolean equalsDefault()

  機能概要 値が 0 と一致するか否かを検査する．

  機能説明 値 value が 0 と一致するか否かを検査する．

  入力データ なし．

  出力データ 一致・不一致の真偽値

  処理方式 value==0 の結果を返す．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 浮動少数点数値をスタックトップに積むバイトコードを生成する．

  機能説明 浮動少数点数値 value をスタックトップに積むバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 スタックトップに value を積むために，opc_ldc2_w をオペコードとするバイトコードを生成する．

エラー処理 なし．

- public void print(PrintStream out)

  機能概要 値を読み易く出力する．

  機能説明 与えられた PrintStream に value を十進数表現で出力する．

  入力データ out (出力先の PrintStream)

  出力データ なし．

  処理方式 value の十進数表現に F を付けて，out.print メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 浮動少数点数値の中身を出力する．

  機能説明 浮動少数点数値 value を十進数で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 Float.toString メソッドを呼び出して，value を文字列に変換し，文字列 “F” と join して返す．

  エラー処理 特になし．

4.5.59 OpenJIT.frontend.tree.ForStatement クラス

public class ForExpression extends Statement

(1) 概要

このクラスは，AST の For 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Statement init

  for 文の初期化文．

- Expression cond

  for 文の条件式．

- Expression inc

  for 文の 1 反復ごとに評価する式．

- Statement body

  for 文の本体．

(3) コンストラクタ

- public ForStatement(int where, Statement init, Expression cond, Expression inc, Statement body)

  機能概要 For オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，For オブジェクトを生成・初期化する．

  入力データ where (行番号), init (初期化文), cond (条件式), inc (増減式), body (for 文のボディ)

  出力データ 特になし．

処理方式 super メソッドを引数 FOR (定数)，where で呼び出す．this.init, this.cond, this.inc, this.body をそれぞれ init, cond, inc, body に更新する．

エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 for 文の検査を行う．

  機能説明 for 文の検査を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値

  処理方式 初期化文 init が null でなければ，init.check メソッドを呼び出して，init の検査を行う．

  条件式 cond が null でない場合，cond.checkCondition メソッドを呼び出して，cond の検査を行う．cond の型を boolean 型にするために convert メソッドを呼び出し，戻り値を cond に格納する．body.check メソッドをメソッドを呼び出し，body の検査を行う．inc が null でなければ，inc.check メソッドを呼び出して，inc の検査を行う．

  cond が null の場合，body.check メソッドを呼び出して，body の検査を行う．inc が null でなければ，inc.check メソッドを呼び出して，inc の検査を行う．

  最後に vset を返す．

  エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式の inlining を行う．init, cond, inc, body をそれぞれ必要に応じて inlining を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 init が null でない場合，Statement の配列 body に init, this を格納し，init を null に格納し，body を含む CompoundStatement オブジェクトを生成する．このオブジェクトに対して，simplify メソッドを呼び出した結果の文を返す．

そうでない場合，cond が null でなければ，code.inlineValue メソッドを呼び出して，その戻り値を cond に格納する．body が null でなければ，body.inline メソッドを呼び出して，その戻り値を body に格納する．inc が null でなければ，inc.inline メソッドを呼び出して，その戻り値を inc に格納する．最後に this を返す．

エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

機能概要 コピーされた式の inlining を行う．

機能説明 コピーされた式の inlining を行う．init, cond, inc, body をそれぞれ必要に応じて inlining を行う．

入力データ ctx (コンテクストを格納する Context オブジェクト), valNeeded (計算後の値をスタック上に残すかどうかを指定するフラグ)

出力データ inlining 後の式．

処理方式 clone メソッドを呼び出して，this のコピーを生成して，s に格納する．init が null でなければ，init.copyInline を呼び出して，その戻り値を s.init に格納する．cond が null でなければ，cond.copyInline を呼び出して，その戻り値を s.cond に格納する．body が null でなければ，body.copyInline を呼び出して，その戻り値を s.body に格納する．inc が null でなければ，inc.copyInline を呼び出して，その戻り値を s.inc に格納する．最後に s を返す．

エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 Inlining のコストを計算する．

  機能説明 init, cond, body, inc の inlining のコストを足し合わせてコストを計算して，返す．

  入力データ thresh (閾値)

  出力データ inlining のコスト．

  処理方式 init.costInline メソッド，cond.costInline メソッド，body.costInline メソッド，inc.costInline メソッドをそれぞれ呼び出して，その戻り値の総和に 2 を加えたものをコストとして返す．

  エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 For 文を実行するバイトコードを生成する．

  機能説明 For 文を実行するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 新しい CodeContext を生成して，newctx に格納する．init が null でなければ，init.code メソッドを呼び出して，init を実行するバイトコードを生成する．

  新しいラベル l1，l2 を生成する．l2 にジャンプするために opc_goto をオペコードとするバイトコードを生成する．

  ラベル l1 を挿入する．body が null でなければ，body.code メソッドを呼び出して，body を実行するバイトコードを生成する．

  newctx.contLabel を挿入する．inc が null でなければ，inc.code メソッドを呼び出して，inc を実行するバイトコードを生成する．

  l2 を挿入する．cond が null でなければ，code の条件式を評価した結果が真なら l1 にジャンプするバイトコード命令を生成するために，

cond.codeBranch メソッドを呼び出す．cond が null の場合は，l1 に無条件でジャンプするために，opc_goto をオペコードとするバイトコードを生成する．

newctx.breakLabel を挿入する．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

  機能概要 For 文を出力する．

  機能説明 For 文を読み易くインデントを付けて出力する．

  入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

  出力データ 特になし．

  処理方式 “for (*init* ; *cond* ; *inc*) *body*” という形式の文字列を生成して，PrintStream out に出力する．

  エラー処理 特になし．

4.5.60 OpenJIT.frontend.tree.GreaterExpression クラス

public class GreatorOrEqualExpression extends BinaryEqualityExpression

(1) 概要

このクラスは，AST の > 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public GreaterExpression(int where, Expression left, Expression right)

  機能概要 AST の > 式ノードを生成する．

  機能説明 スーパークラスの BinaryEqualityExpression クラスで定義されているコンストラクタを用いて，> 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (> 式の左辺式の Expression オブジェクト)，right (> 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 GT(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数で > 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理内容 a>b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数で > 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a>b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(float a, float b)

  機能概要 式を評価する．引数がともに float 型の場合．

  機能説明 float 型の引数で > 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (float 型), b (float 型)

  出力データ Expression オブジェクト．

  処理内容 a>b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数がともに double 型の場合．

機能説明 double 型の引数で > 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (double 型), b (double 型)

出力データ Expression オブジェクト．

処理内容 a>b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 左辺式が定数で，右辺式が定数でない場合に式の単純化を行う．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式が定数で，右辺式が定数でない場合，右辺式と左辺式を入れ換えた LessExpression を生成して返す．そうでない場合， this を返す．

  エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

  機能概要 > 式の条件分岐を行うバイトコードを生成する．

  機能説明 各型ごとに適切な条件分岐用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

  出力データ なし．

  処理方式 1. left.codeValue メソッドを呼び出し，左辺式をスタックトップに積むバイトコードを生成する．

2. 左辺式の型を検査し，int 型の場合は，右辺式が 0 であれば 6 へ．非零であれば，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．さらに whenTrue が True/False に従って，opc_if_icmpgt, opc_if_icmple をオペコードとするバイトコードを生成して終了．
3. 左辺式の型が long の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_lcmp をオペコードとするバイトコードを生成する．6 へ．
4. 左辺式の型が float の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_fcmpl をオペコードとするバイトコードを生成する．6 へ．
5. 左辺式の型が double の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_dcmpl をオペコードとするバイトコードを生成する．6 へ．
6. whenTrue の True/False に従って，opc_ifgt, opc_ifle をオペコードとするバイトコードを生成して終了．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 > 式の中身を出力する．

  機能説明 > 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.61 OpenJIT.frontend.tree.GreaterOrEqualExpression クラス

public class GreatorOrEqualExpression extends BinaryEqualityExpression

(1) 概要

このクラスは，AST の >= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public GreaterOrEqualExpression(int where, Expression left, Expression right)

  機能概要 AST の >= 式ノードを生成する．

  機能説明 スーパークラスの BinaryEqualityExpression クラスで定義されているコンストラクタを用いて，>= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (>= 式の左辺式の Expression オブジェクト)，right (>= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 GE(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

機能説明 int 型の引数で >= 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理内容 a>=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

機能概要 式を評価する．引数がともに long 型の場合．

機能説明 long 型の引数で >= 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (long 型), b (long 型)

出力データ Expression オブジェクト．

処理内容 a>=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(float a, float b)

機能概要 式を評価する．引数がともに float 型の場合．

機能説明 float 型の引数で >= 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (float 型), b (float 型)

出力データ Expression オブジェクト．

処理内容 a>=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数がともに double 型の場合．

  機能説明 double 型の引数で >= 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (double 型), b (double 型)

  出力データ Expression オブジェクト．

  処理内容 a>=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 左辺式が定数で，右辺式が定数でない場合に式の単純化を行う．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式が定数で，右辺式が定数でない場合，右辺式と左辺式を入れ換えた LessOrEqualExpression を生成して返す．そうでない場合，this を返す．

  エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

  機能概要 >= 式の条件分岐を行うバイトコードを生成する．

  機能説明 各型ごとに適切な条件分岐用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

出力データ なし．

処理方式　1. left.codeValue メソッドを呼び出し，左辺式をスタックトップに積むバイトコードを生成する．

2. 左辺式の型を検査し，int 型の場合は，右辺式が 0 であれば 6 へ．非零であれば，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．さらに whenTrue が True/False に従って，opc_if_icmpge, opc_if_icmplt をオペコードとするバイトコードを生成して終了．

3. 左辺式の型が long の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_lcmp をオペコードとするバイトコードを生成する．6 へ．

4. 左辺式の型が float の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_fcmpl をオペコードとするバイトコードを生成する．6 へ．

5. 左辺式の型が double の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_dcmpl をオペコードとするバイトコードを生成する．6 へ．

6. whenTrue の True/False に従って，opc_ifge, opc_iflt をオペコードとするバイトコードを生成して終了．

エラー処理 特になし．

- public String toPrettyString()

機能概要 >= 式の中身を出力する．

機能説明 >= 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.62 OpenJIT.frontend.tree.IdentifierExpression クラス

public class IdentifierExpression extends Expression

(1) 概要

このクラスは，AST の識別子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- public Identifier id

  識別子．

- FieldDefinition field

  識別子に対応するフィールドの定義．

(3) コンストラクタ

- public IdentifierExpression(int where, Identifier id)

  機能概要 IdentifierExpression オブジェクトを生成する．

  機能説明 スーパークラスの Expression クラスで定義されているコンストラクタを用いて，IdentifierExpression オブジェクトを生成する．

  入力データ where (行番号), id (識別子)

  出力データ なし．

  処理方式 super メソッドを引数 IDENT (定数), where, Type.tError (定数) で呼び出す．this.id に id を格納する．

  エラー処理 特になし．

(4) メソッド

- public boolean equals(Identifier id)

機能概要 与えられた識別子と値が同じかどうかを検査する．

機能説明 与えられた識別子と値が同じかどうかを検査し，真偽値を返す．

入力データ id (識別子)

出力データ 真偽値．

処理方式 this.id.equals(id) メソッドを呼び出し，戻り値を返す．

エラー処理 特になし．

- private long assign(Environment env, long vset)

  機能概要 アイデンティファイアに値を代入する．

  機能説明 アイデンティファイアが local フィールドを指していれば，代入を行う．

  入力データ env (環境を格納する Environment オブジェクト), vset (条件式の評価値)

  出力データ 条件式の評価値

  処理方式 field.isLocal メソッドを呼び出した結果，field が local フィールドであれば，vset $|= 1 <<$ field.number とする．また，field.writecount を 1 加算する．

  field.isFinal メソッドを呼び出した結果，field が final フィールドであれば，assign.to.final エラーを出力する．

  vset を返す．

  エラー処理 final フィールドに値を代入しようとした場合は，assign.to.final エラーを出力する．

4.5.63 OpenJIT.frontend.tree.IfStatement クラス

public class IfStatement extends Statement

(1) 概要

このクラスは，AST の If 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Expression cond

  if 文の条件式．

- Statement ifTrue

  if 文の条件が true の時に実行する文．

- Statement ifFalse

  if 文の条件が false の時に実行する文．

(3) コンストラクタ

- public IfStatement(int where, Expression cond, Statement ifTrue, Statement ifFalse)

  機能概要 If オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを呼び出して，If オブジェクトの生成・初期化を行う．

  入力データ where (行番号), cond (条件式), ifTrue (True 節), ifFalse (False 節)

  出力データ なし．

  処理方式 super メソッドを引数 IF (定数), where で呼び出す．this.cond に cond，this.ifTrue に ifTrue，this.ifFalse に ifFalse を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 If 文の検査を行う．

  機能説明

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値

  処理方式 条件式 cond の検査を行うために，cond.checkCondition メソッドを呼び出す．cond の型を boolean 型に変換するために convert メソッドを呼び出す．次に，True 節 ifTrue の検査を行うために，ifTrue.check メソッドを呼び出す．ifFalse が null でなければ，False 節 ifFalse の検査を行うために，ifFalse.check メソッドを呼び出す．

  エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

  機能概要 If 文の inlining を行う．

  機能説明

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の文．

  処理方式 条件式 cond の inlining を行うために，cond.inlineValue メソッドを呼び出して，戻り値を cond に格納する．

  cond が true の場合，True 節 ifTrue を inlining するために，ifTrue.inline メソッドを呼び出す．ifTrue 節を CompoundStatement に変換するために，eliminate メソッドを呼び出し，戻り値を返す．

  cond が false の場合，ifFalse が null かどうか検査する．null でなければ，ifFalse を inlining するために，ifFalse.inline メソッドを呼び出す．ifFalse

節を CompoundStatement に変換するために，eliminate メソッドを呼び出し，戻り値を返す．

cond がこれら以外の場合，ifTrue が null でなければ，True 節 ifTrue を inlining するために，ifTrue.inline メソッドを呼び出して，戻り値を ifTrue に格納する．ifFalse が null でなければ，False 節 ifFalse を inlining するために，ifFalse.inline メソッドを呼び出して，戻り値を ifFalse に格納する．

ifTrue, ifFalse ともに null であれば，cond を文として実行する ExpressionStatement オブジェクトを生成し，inline メソッドを呼び出す．この戻り値を CompoundStatement に変換するために，eliminate メソッドを呼び出し，戻り値を返す．

ifTrue が null であれば，cond の符号反転式を作るために NotExpression を生成して，cond に格納する．cond を条件式，ifFalse を True 節とする新しい IfStatement オブジェクトを生成し，この式を CompoundStatement にするために，eliminate メソッドを呼び出し，戻り値を返す．

最後に，this を返す．

エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

  機能概要 コピーされた式の inlining を行う．

  機能説明 If 文のコピーを作り，inlining を行う．

  入力データ ctx (コンテクストを格納する Context オブジェクト), valNeeded (計算後の値をスタック上に残すかどうかを指定するフラグ)

  出力データ inlining 後の文．

  処理方式 clone メソッドを呼び出して，If 文のコピーを生成し，s に格納する．cond を inlining するために，cond.copyInline メソッドを呼び出し，戻り値を s.cond に格納する．ifTrue が null でなければ，ifTrue 文を inlining するために，ifTrue.copyInline メソッドを呼び出し，戻り値を s.ifTrue に格納する．ifFalse が null でなければ，ifFalse 文を inlining するために，ifFalse.copyInline メソッドを呼び出し，戻り値を s.ifFalse に格納する．最後に s を返す．

エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 Inlining のコストを計算する．

  機能説明 cond, ifTrue, ifFalse の inlining のコストを足し合わせてコストを計算して，返す．

  入力データ thresh (閾値)

  出力データ inlining のコスト．

  処理方式 cond.costInline メソッド，ifTrue.costInline メソッド，ifFalse.costInline メソッドをそれぞれ呼び出して，その戻り値の総和に 1 を加えたものをコストとして返す．

  エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 If 文を実行するバイトコードを生成する．

  機能説明 If 文を実行するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 新しく CodeContext を生成し，newctx に格納する．新しくラベル l1 を作る．cond の式を評価して，成立しなかったら l1 にジャンプするバイトコードを生成するために，cond.codeBranch メソッドを呼び出す．ifTrue の文を実行するバイトコードを生成するために，ifTrue.code メソッドを呼び出す．

  ifFalse が null でない場合，新しくラベル l2 を生成する．ラベル l2 に無条件でジャンプするバイトコードを生成するために，opc_goto をオペコードとするバイトコードを生成する．ラベル l1 を挿入する．次に，ifFalse の文

を実行するバイトコードを生成するために，ifFalse.code メソッドを呼び出す．ラベル l2 を挿入する．

そうでない場合，ラベル l1 を挿入する．

最後に newctx.breakLabel を挿入する．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

  機能概要 If 文を出力する．

  機能説明 If 文を読み易くインデントを付けて出力する．

  入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

  出力データ 特になし．

  処理方式 “if *cond* *ifTrue* else *ifFalse*” という形式の文字列を生成して，PrintStream out に出力する．

  エラー処理 特になし．

4.5.64 OpenJIT.frontend.tree.IncDecExpression クラス

public class IncDecExpression extends UnaryExpression

(1) 概要

このクラスは，PostDecExpression, PostIncExpression, PreDecExpression, PreIncExpression のスーパークラスであり，これらに共通するコード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public IncDecExpression(int op, int where, Expression right)

  機能概要 IncDecExpression オブジェクトを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されてるコンストラクタを用いて，IncDecExpression オブジェクトを生成・初期化する．

  入力データ op (演算子の種類), where (行番号), right (右辺式),

  出力データ なし．

  処理方式 super メソッドを引数 op, where, right.type, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の検査をする．

  機能説明 式の検査をする．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値

処理方式 right.checkAssignOp メソッドを呼び出して，右辺式の検査を行う．right.type が数値型であれば，right.type を type に格納する．そうでない場合，invalid.arg.type エラーを出力する．

エラー処理 右辺式が数値型でない場合に invalid.arg.type エラーを出力する．

- public long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式を検査する．

  機能説明 void 式を検査する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値

  処理方式 checkValue メソッドを呼び出し，戻り値を返す．

  エラー処理 特になし．

- public Expression inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 inlineValue メソッドを呼び出して，戻り値を返す．

  エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

機能概要 式の inlining を行う．

機能説明 式の inlining を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 right.inlineValue メソッドを呼び出し，戻り値で right を更新する．this を返す．

エラー処理 特になし．

- void codeIncDec(Environment env, Context ctx, Assembler asm, boolean inc, boolean prefix, boolean valNeeded)

  機能概要 式を実行するバイトコードを生成する．

  機能説明 式を実行するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), inc (加算か減算かのフラグ), prefix (前置演算子か後置演算子かのフラグ), valNeeded (計算後の値をスタック上に残すかどうかを指定するフラグ)

  出力データ なし．

  処理方式 right.op が IDENT(定数) であり，type が int 型の場合，valNeeded が true かつ prefix が false ならば，right.codeLoad を呼び出す．

  opc_iinc をオペコードとするバイトコードを生成する．valNeeded が true かつ prefix が true ならば，right.codeLoad を呼び出して，戻る．

  そうでない場合，right.codeLValue メソッドを呼び出し，codeDup を呼び出し，right.codeLoad を呼び出す．valNeeded が true かつ prefix が false なら，codeDup を呼び出す．type の型を得て，それが byte 型なら，opc_ldc をオペコードをするバイトコード，opc_iadd ないし opc_isub をオペコードとするバイトコード，opc_int2byte をオペコードとするバイトコードを順次生成する．

それが short 型なら，opc_ldc をオペコードをするバイトコード，opc_iadd ないし opc_isub をオペコードとするバイトコード，opc_int2short をオペコードとするバイトコードを順次生成する．

それが char 型なら，opc_ldc をオペコードをするバイトコード，opc_iadd ないし opc_isub をオペコードとするバイトコード，opc_int2char をオペコードとするバイトコードを順次生成する．

それが int 型なら，opc_ldc をオペコードをするバイトコード，opc_iadd ないし opc_isub をオペコードとするバイトコードを順次生成する．

それが long 型なら，opc_ldc2_w をオペコードをするバイトコード，opc_ladd ないし opc_lsub をオペコードとするバイトコードを順次生成する．

それが float 型なら，opc_ldc をオペコードをするバイトコード，opc_fadd ないし opc_fsub をオペコードとするバイトコードを順次生成する．

それが double 型なら，opc_ldc2_w をオペコードをするバイトコード，opc_dadd ないし opc_dsub をオペコードとするバイトコードを順次生成する．

valNeeded が true かつ prefix が true なら，codeDup を呼び出す．

最後に，right.codeStore を呼び出す．

エラー処理 特になし．

4.5.65 OpenJIT.frontend.tree.InlineMethodExpression クラス

public class InlineMethodExpression extends Expression

(1) 概要

このクラスは，AST のインライン展開されたメソッド呼び出しの式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- FieldDefinition field

  メソッドのフィールドの定義．

- Statement body

  メソッド本体．

(3) コンストラクタ

- public InlineMethodExpression(int where, Type type, FieldDefinition field, Statement body)

  機能概要 InlineMethodExpression オブジェクトを生成する．

  機能説明 スーパークラスの Expression クラスで定義されているコンストラクタを用いて，InlineMethodExpression オブジェクトの生成・初期化を行う．

  入力データ where (行番号), type (型), field, body (文本体)

  出力データ なし．

  処理方式 super メソッドを引数 INLINEMETHOD(定数), where, type で呼び出す．this.field に field，this.body に body を格納する．

  エラー処理 特になし．

(4) メソッド

- public Expression inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 body.inline メソッドを呼び出し，戻り値を body に格納する．body が null であるか，body.op が INLINERETURN (定数) の場合には null，そうでない場合には this を返す．

  エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 body.inline メソッドを呼び出し，戻り値を body に格納する．body.op が INLINERETURN (定数) の場合には，body.expr を返す．そうでない場合には this を返す．

  エラー処理 特になし．

- public Expression copyInline(Context ctx)

  機能概要 コピーされた式の inlining を行う．

  機能説明 コピーされた式の inlining を行う．

  入力データ ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 clone メソッドを呼び出し，戻り値を e に格納する．body が null でなければ，body.copyInline メソッドを呼び出し，戻り値を e.body に格納する．最後に e を返す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 式を実行するバイトコード生成する．

  機能説明 式を実行するバイトコード生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 codeValue メソッドを呼び出す．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 式を実行するバイトコード生成する．

  機能説明 式を実行するバイトコード生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 body を実行するバイトコードを生成するために，body.code メソッドを呼び出す．breakLabel を挿入する．

  エラー処理 特になし．

- public void print(PrintStream out)

機能概要 式を出力する．

機能説明 式を読み易く出力する．

入力データ out (String の出力先の PrintStream)

出力データ 特になし．

処理方式 “(*op*”, “ body)” という形式の文字列を生成して，PrintStream out に出力する．

エラー処理 特になし．

4.5.66 OpenJIT.frontend.tree.InlineNewInstanceExpression クラス

public class InlineNewInstanceExpression extends Expression

(1) 概要

このクラスは，AST のインライン展開された newInstance の式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- FieldDefinition field

  変数フィールドの定義．

- Statement body

  本体．

(3) コンストラクタ

- public InlineNewInstanceExpression(int where, Type type, FieldDefinition field, Statement body)

  機能概要 InlineNewInstanceExpression オブジェクトを生成する．

  機能説明 スーパークラスの Expression クラスで定義されているコンストラクタを用いて，InlineNewInstanceExpression オブジェクトを生成する．

  入力データ where (行番号), type (型), field (変数フィールドの定義), body (本体の文)

  出力データ なし．

  処理方式 super メソッドを引数 INLINENEWINSTANCE (定数), where, type で呼び出す．this.field に field，this.body に body を格納する．

  エラー処理 特になし．

(4) メソッド

- public Expression inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 inlineValue メソッドを呼び出し，戻り値を返す．

  エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 body が null でなければ，body を inlining するために，body.inline メソッドを呼び出す．

  body が null でなく，body.op が INLINERETURN (定数) の場合，body を null にする．

  this を返す．

  エラー処理 特になし．

- public Expression copyInline(Context ctx)

  機能概要 メソッドインライン展開用に式のコピーを作る．

  機能説明 式のコピーを作る．body をインライン展開した式をコピーに格納する．

入力データ ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 コピーを作るために，clone メソッドを呼び出し，InlineNewInstance-Expression オブジェクト e を作る．body.copyInline メソッドを呼び出し，戻り値を e.body に格納する．e を返す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

機能概要 newInstance の処理を行うバイトコードを生成する．

機能説明 newInstance の処理を行うバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 field.getClassDeclaration メソッドを呼び出し，得られたクラス定義に対応するオブジェクトを生成するために，opc_new をオペコードとするバイトコードを生成する．

body が null でなければ，field から引数の配列の最初の要素を取り出して，LocalField v に格納する．また，新しく CodeContext オブジェクトを生成して，newctx に格納する．

v の指すローカルフィールドに値を格納するために，opc_astore をオペコードとするバイトコードを生成する．次に，body を実行するバイトコードを生成するために，body.code メソッドを呼び出す．最後に，break の処理のために，newctx.breakLabel を挿入する．

エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

機能概要 newInstance の処理を行うバイトコードを生成する．

機能説明 newInstance の処理を行うバイトコードを生成する．最後に，値がスタックトップに積まれた状態にする．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 field.getClassDeclaration メソッドを呼び出し，得られたクラス定義に対応するオブジェクトを生成するために，opc_new をオペコードとするバイトコードを生成する．

body が null でなければ，field から引数の配列の最初の要素を取り出して，LocalField v に格納する．また，新しく CodeContext オブジェクトを生成して，newctx に格納する．

v の指すローカルフィールドに値を格納するために，opc_astore をオペコードとするバイトコードを生成する．次に，body を実行するバイトコードを生成するために，body.code メソッドを呼び出す．break の処理のために，newctx.breakLabel を挿入する．スタックトップに値をロードするために，opc_aload をオペコードとするバイトコードを生成する．

エラー処理 特になし．

- public void print(PrintStream out)

機能概要 式の中身を出力する．

機能説明 式の中身を読み易く文字列で，PrintStream out に出力する．

入力データ out (String の出力先の PrintStream)

出力データ なし．

処理方式 “(opNames[op] = *body*)” という形式の文字列を生成して，PrintStream out に出力する．

エラー処理 特になし．

4.5.67 OpenJIT.frontend.tree.InlineReturnStatement クラス

public class InlineReturnStatement extends Statement

(1) 概要

このクラスは，AST の inline return 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Expression expr

  戻り値の式．

(3) コンストラクタ

- public InlineReturnStatement(int where, Expression expr)

  機能概要 InlineReturnStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，InlineReturnStatement オブジェクトの生成・初期化を行う．

  入力データ where (行番号), expr (戻り値の式)

  出力データ なし．

  処理方式 super メソッドを引数 INLINERETURN (定数), where で呼び出す．this.expr に expr を格納する．

  エラー処理 特になし．

(4) メソッド

- Context getDestination(Context ctx)

  機能概要 break の後のコンテクストを取得する．

  機能説明 break の後のコンテクストを取得する．

入力データ ctx (コンテクストを格納する Context オブジェクト)

出力データ コンテクスト．

処理方式 ctx を後ろから順にたどっていって，ctx.node が null でなく，ctx.node.op が INLINEMETHOD (定数) または INLINENEWINSTANCE (定数) である最初の ctx を返す．

最後まで見つからなかった場合，null を返す．

エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

機能概要 文の inlining を行う．

機能説明 expr の inlining を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の文．

処理方式 expr が null でなければ，expr.inlineValue メソッドを呼び出し，戻り値を expr に格納する．

最後に，this を返す．

エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

機能概要 コピーされた文の inlining を行う．

機能説明 inline return 文のコピーを作り，それに対して inlining を行う．

入力データ ctx (コンテクストを格納する Context オブジェクト), valNeeded (計算後の値をスタック上に残すか否かを決定する真偽値)

出力データ Inlining 後の Statement．

処理方式 clone メソッドを呼び出し，InlineReturnStatement オブジェクトのコピーを作り，s に格納する．

expr が null でなければ，expr.inlineValue メソッドを呼び出し，戻り値を expr に格納する．

最後に s を返す．

エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 inlining のコストを計算する．

  機能説明 inlining のコストを計算する．

  入力データ thresh (閾値)

  出力データ inlining のコスト．

  処理方式 1+ expr.costInline(thresh) を求めて，返す．

  エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 文を実行するバイトコードを生成する．

  機能説明 expr を計算するバイトコードと，Caller に戻るためのバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 expr が null でなければ，expr.codeValue メソッドを呼び出して，expr を計算して，スタックトップに計算値に積んだ状態にするバイトコードを生成する．

  getDestination メソッドを呼び出して，戻り先のコンテクストを得る．このコンテクストの breakLabel にジャンプするために，opc_goto をオペコードとするバイトコードを生成する．

  エラー処理 特になし．

- public void print(PrintStream out, int indent)

  機能概要 文の中身を出力する．

  機能説明 文の中身を読み易く文字列で，PrintStream out に出力する．

  入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

  出力データ なし．

  処理方式 indent×4 個の空白文字を out に出力する．“inline-return *expr*;” という形式の文字列を生成して，PrintStream out に出力する．

  エラー処理 特になし．

4.5.68 OpenJIT.frontend.tree.InstanceOfExpression クラス

public class InstanceOfExpression extends BinaryExpression

(1) 概要

このクラスは，AST の instanceof 演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- InstanceOfExpression(int where, Expression left, Expression right)

  機能概要 InstanceOfExpression オブジェクトを生成する．

  機能説明 スーパークラスの BinaryExpression クラスで定義されているコンストラクタを用いて，InstanceOfExpression オブジェクトの生成・初期化を行う．

  入力データ where (行番号), left (左辺式), right (右辺式)

  出力データ なし．

  処理方式 super メソッドを引数 INSTANCEOF (定数), where, Type.tBoolean, left, right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の検査を行う．

  機能説明

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 左辺式を検査するために，left.checkValue メソッドを呼び出し，戻り値を vset に格納する．右辺式の型式を生成するために，新しく TypeExpression オブジェクトを生成し，right に格納する．

env.explicitCast(left.type, right.type) メソッドを呼び出して，左辺式が右辺式 (型式) に変換可能かどうか検査する．変換不可能であれば，invalid.instanceof エラーを出力する．

最後に vset を返す．

エラー処理 instanceof 式として不正な場合，即ち，左辺式が右辺式の型式に明示的なキャストで変換不能な場合，invalid.instanceof エラーを出力する．

- public Expression inline(Environment env, Context ctx)

機能概要 instanceof 式の inlining を行う．

機能説明 instanceof 式の左辺式の inlining を行い，左辺式を返す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の左辺式．

処理方式 left.inline メソッドを呼び出し，戻り値を返す．

エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

機能概要 instanceof 式の inlining を行う．

機能説明 instanceof 式の左辺式の inlining を行い，this を返す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の instanceof 式．

処理方式 left.inline メソッドを呼び出し，戻り値を left に格納する．その後，this を返す．

エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 instanceof 式を実行するバイトコードを生成する．

  機能説明 instanceof 式を実行するバイトコードを生成する．(スタックトップに式の計算結果を残す場合)

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 左辺式の計算値をスタックトップに積むバイトコードを生成するために，left.codeValue メソッドを呼び出す．

  opc_instanceof をオペコードとするバイトコードを生成する．

  エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

  機能概要 instanceof 式が条件文の中で使われる場合のバイトコードを生成する．

  機能説明 instanceof 式が条件文の中で使われる場合のバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

  出力データ なし．

処理方式 codeValue メソッドを呼び出し，instanceof 式の評価結果をスタックトップに積むバイトコードを生成する．whenTrue が true なら，opc_ifne をオペコードとするバイトコードを生成する．whenTrue が false なら，opc_ifeq をオペコードとするバイトコードを生成する．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 左辺式の計算を行うバイトコードを生成する．

  機能説明 instanceof 式の左辺式の計算を行うバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 left.code メソッドを呼び出す．

  エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 instanceof 式を出力する．

  機能説明 instanceof 式を読みやすい形式で PrintStream に出力する．

  入力データ out (String の出力先の PrintStream)

  出力データ 特になし．

  処理方式 “(instanceof *left right*)” という形式の文字列を作り，PrintStream out に出力する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 instanceof 式の中身を出力する．

  機能説明 instanceof 式を十進数で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを引数 “instanceof” で呼び出す．

エラー処理 特になし．

4.5.69 OpenJIT.frontend.tree.IntExpression クラス

public class IntExpression extends IntegerExpression

(1) 概要

このクラスは，AST の int 型整数ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public IntExpression(int where, int value)

  機能概要 IntExpression オブジェクトを生成する．

  機能説明 スーパークラスの IntegerExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), value (保持する値)

  出力データ なし．

  処理方式 super メソッドを引数 INTVAL(定数), where, Type.tInt(定数), value で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public boolean equals(Object obj)

  機能概要 与えられた式との等価性を検査する．

  機能説明 与えられた式との等価性を検査し，真偽値を返す．

  入力データ obj (比較する式)．

出力データ 真偽値．

処理方式 obj が IntExpression オブジェクトであれば，obj.value と value が一致するかどうか検査し，真偽値を返す．それ以外の場合は，false を返す．

エラー処理 特になし．

- public int hashCode()

  機能概要 ハッシュコードを返す．

  機能説明 switch 文で必要になるハッシュコードを返す．

  入力データ なし．

  出力データ ハッシュ値．

  処理方式 value を返す．

  エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 値を読み易く出力する．

  機能説明 与えられた PrintStream に value を十進数表現で出力する．

  入力データ out (出力先の PrintStream)

  出力データ なし．

  処理方式 out.print メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 値を読み易く出力する．

  機能説明 与えられた value を十進数表現で出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 postfixForm メソッドを引数 “” で呼び出す．

  エラー処理 特になし．

4.5.70 OpenJIT.frontend.tree.IntegerExpression クラス

public class IntegerExpression extends ConstantExpression

(1) 概要

このクラスは，AST の整数ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- int value

  整数値を保持する

(3) コンストラクタ

- public IntegerExpression(int op, int where, Type, type, int value)

  機能概要 IntegerExpression オブジェクトを生成する．

  機能説明 スーパークラスの ConstantExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ op (演算子の種類), where (行番号)，type (保持する値の型), value (保持する値)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, type で呼び出し，this.value に value を代入する．

  エラー処理 特になし．

(4) メソッド

- public boolean fitsType(Environtment env, Type t)

  機能概要 与えられた型と数値の型が一致するかどうかを検査する．

機能説明 与えられた型に value をキャストとして元の value と値が一致するか否かを検査する．

入力データ env (環境を格納する Environment オブジェクト), t (型オブジェクト)

出力データ 型の一致・不一致の真偽値

処理方式 t の型を取得し，型が byte, short, char の場合， value==(byte)value, value==(short)value, value==(char)value の結果を返す．それ以外の場合， super.fitsType(env, t) を呼び出す．

エラー処理 特になし．

- public Object getValue()

機能概要 値を取り出す．

機能説明 value を保持する Integer オブジェクトを返す．

入力データ なし．

出力データ 数値オブジェクト

処理方式 value を保持する Integer オブジェクトを生成して，返す．

エラー処理 特になし．

- public boolean equals(int i)

機能概要 値が与えられた整数値と一致するか否かを検査する．

機能説明 値 value が与えられた整数値と一致するか否かを検査する．

入力データ i (比較する値)

出力データ 一致・不一致の真偽値

処理方式 value==i の結果を返す．

エラー処理 特になし．

- public boolean equalsDefault()

機能概要 値が 0 と一致するか否かを検査する．

機能説明 値 value が 0 と一致するか否かを検査する．

入力データ なし．

出力データ 一致・不一致の真偽値

処理方式 value==0 の結果を返す．

エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 整数値をスタックトップに積むバイトコードを生成する．

  機能説明 整数値 value をスタックトップに積むバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 スタックトップに value を積むために，opc_ldc をオペコードとするバイトコードを生成する．

  エラー処理 なし．

- public String toPrettyString()

  機能概要 整数値の中身を出力する．

  機能説明 整数値を十進数で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 Integer.toString メソッドを呼び出して，value を文字列に変換して返す．

  エラー処理 特になし．

- public String postfixForm(String operator)

  機能概要 後置記法形式で出力する．

機能説明 値を演算子とともに後置記法形式で出力する．

入力データ operator (演算子の文字列)

出力データ String 型の文字列．

処理方式 Integer.toString メソッドを呼び出して，value を文字列に変換し，operator 文字列と join して得られた文字列を返す．

エラー処理 特になし．

4.5.71 OpenJIT.frontend.tree.LengthExpression クラス

public class LengthExpression extends UnaryExpression

(1) 概要

このクラスは，AST の単項演算子 Length の式ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public LengthExpression(int where, Expression right)

  機能概要 LengthExpression オブジェクトを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されているコンストラクタを用いて，Length 式ノードの生成・初期化を行う．

  入力データ where (行番号)，right (Length 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 LENGTH(定数), where, type.tInt(定数), right で呼び出す．

  エラー処理 なし．

(4) メソッド

- long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の型を検査する．

  機能説明 式が配列であるか否かを検査する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 right.checkValue メソッドを呼び出し，式 rihgt を検査する．right.type が配列型でなければ，invalid.length エラーを出力する．最後に，vset を返す．

エラー処理 式が配列型でなければ，invalid.length エラーを出力する．

- public void codeValue(Environment env, Context ctx, Assembler asm)

機能概要 length 式の計算をするバイトコードを生成する．

機能説明 length 式の計算をするバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 右辺式をスタックトップに積むバイトコードを生成するために，right.codeValue メソッドを呼び出す．次に，opc_arraylength をオペコードとするバイトコードを追加する．

エラー処理 特になし．

- public String toPrettyString()

機能概要 length 式の中身を出力する．

機能説明 length 式を後置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 postfixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.72 OpenJIT.frontend.tree.LessExpression クラス

public class LessExpression extends BinaryEqualityExpression

(1) 概要

このクラスは，AST の < 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public LessExpression(int where, Expression left, Expression right)

  機能概要 AST の < 式ノードを生成する．

  機能説明 スーパークラスの BinaryEqualityExpression クラスで定義されているコンストラクタを用いて，< 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (< 式の左辺式の Expression オブジェクト)，right (< 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 LT(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数で < 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理内容 a<b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数で < 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a<b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(float a, float b)

  機能概要 式を評価する．引数がともに float 型の場合．

  機能説明 float 型の引数で < 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (float 型), b (float 型)

  出力データ Expression オブジェクト．

  処理内容 a<b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数がともに double 型の場合．

機能説明 double 型の引数で < 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (double 型), b (double 型)

出力データ Expression オブジェクト．

処理内容 a<b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 左辺式が定数で，右辺式が定数でない場合に式の単純化を行う．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式が定数で，右辺式が定数でない場合，右辺式と左辺式を入れ換えた GreaterExpression を生成して返す．そうでない場合， this を返す．

  エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

  機能概要 < 式の条件分岐を行うバイトコードを生成する．

  機能説明 各型ごとに適切な条件分岐用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

  出力データ なし．

  処理方式 1. left.codeValue メソッドを呼び出し，左辺式をスタックトップに積むバイトコードを生成する．

2. 左辺式の型を検査し，int 型の場合は，右辺式が 0 であれば 6 へ．非零であれば，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．さらに whenTrue が True/False に従って，opc_if_icmplt, opc_if_icmpge をオペコードとするバイトコードを生成して終了．
3. 左辺式の型が long の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_lcmp をオペコードとするバイトコードを生成する．6 へ．
4. 左辺式の型が float の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_fcmpl をオペコードとするバイトコードを生成する．6 へ．
5. 左辺式の型が double の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_dcmpl をオペコードとするバイトコードを生成する．6 へ．
6. whenTrue の True/False に従って，opc_iflt, opc_ifge をオペコードとするバイトコードを生成して終了．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 ＜式の中身を出力する．

  機能説明 ＜式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.73 OpenJIT.frontend.tree.LessOrEqualExpression クラス

public class LessOrEqualExpression extends BinaryEqualityExpression

(1) 概要

このクラスは，AST の <= 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public LessOrEqualExpression(int where, Expression left, Expression right)

  機能概要 AST の <= 式ノードを生成する．

  機能説明 スーパークラスの BinaryEqualityExpression クラスで定義されているコンストラクタを用いて，<= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (<= 式の左辺式の Expression オブジェクト)，right (<= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 LE(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数で <= 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理内容 a<=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数で <= 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a<=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(float a, float b)

  機能概要 式を評価する．引数がともに float 型の場合．

  機能説明 float 型の引数で <= 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (float 型), b (float 型)

  出力データ Expression オブジェクト．

  処理内容 a<=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数がともに double 型の場合．

機能説明 double 型の引数で <= 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (double 型), b (double 型)

出力データ Expression オブジェクト．

処理内容 a<=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 左辺式が定数で，右辺式が定数でない場合に式の単純化を行う．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式が定数で，右辺式が定数でない場合，右辺式と左辺式を入れ換えた GreaterOrEqualExpression を生成して返す．そうでない場合，this を返す．

  エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

  機能概要 <= 式の条件分岐を行うバイトコードを生成する．

  機能説明 各型ごとに適切な条件分岐用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

  出力データ なし．

処理方式　1. left.codeValue メソッドを呼び出し，左辺式をスタックトップに積むバイトコードを生成する．

2. 左辺式の型を検査し，int 型の場合は，右辺式が 0 であれば 6 へ．非零であれば，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．さらに whenTrue が True/False に従って，opc_if_icmple, opc_if_icmpgt をオペコードとするバイトコードを生成して終了．

3. 左辺式の型が long の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_lcmp をオペコードとするバイトコードを生成する．6 へ．

4. 左辺式の型が float の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_fcmpl をオペコードとするバイトコードを生成する．6 へ．

5. 左辺式の型が double の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_dcmpl をオペコードとするバイトコードを生成する．6 へ．

6. whenTrue の True/False に従って，opc_ifle, opc_ifgt をオペコードとするバイトコードを生成して終了．

エラー処理 特になし．

- public String toPrettyString()

機能概要 <= 式の中身を出力する．

機能説明 <= 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.74 OpenJIT.frontend.tree.LocalField クラス

public class LocalField extends FieldDefinition

(1) 概要

このクラスは，AST のローカル変数フィールドのノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- int number

  ローカル変数番号．

- int readcount

  ローカル変数を読んだ回数．

- int writecount

  ローカル変数に書いた回数．

- LocalField prev

  前のローカル変数を指すポインタ．

(3) コンストラクタ

- public LocalField(int where, ClassDefinition clazz, int modifiers, Type type, Identifier name)

  機能概要 LocalField オブジェクトを生成する．

  機能説明 スーパークラスの FieldDefinition クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), clazz (ClassDefinition オブジェクト), modifiers (修飾子), type (型), name (変数の名前)

  出力データ なし．

処理方式 super メソッドを引数 clazz, modifiers, type, name, new ClassDeclaration[0], null で呼び出す．

エラー処理 特になし．

(4) メソッド

- public boolean isLocal()

  機能概要 ローカル変数であるかどうかを返す．

  機能説明 true を返す．

  入力データ なし．

  出力データ 真偽値．

  処理方式 true を返す．

  エラー処理 特になし．

- public boolean isUsed()

  機能概要 使われたかどうかを返す．

  機能説明 readcount, writecount を見て使われたかどうかを返す．

  入力データ なし．

  出力データ 真偽値．

  処理方式 readcount が 0 かつ writecount が 0 のとき，false を返す．それ以外のとき false を返す．

  エラー処理 特になし．

- public Node getValue(Environment env)

  機能概要 値のアクセッサメソッド．

  機能説明 ローカル変数の値を取り出す．

  入力データ env (環境オブジェクト)

  出力データ 値の式．

処理方式 getValue メソッドを呼び出し，その戻り値を返す．

エラー処理 特になし．

4.5.75 OpenJIT.frontend.tree.LongExpression クラス

public class LongExpression extends ConstantExpression

(1) 概要

このクラスは，AST の long 型整数ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- long value (整数値を保持する)

(3) コンストラクタ

- public LongExpression(int where, long value)

  機能概要 LongExpression オブジェクトを生成する．

  機能説明 スーパークラスの ConstantExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), value (保持する値)

  出力データ なし．

  処理方式 super メソッドを引数 LONGVAL(定数), where, Type.tLong (定数) で呼び出し，this.value に value を代入する．

  エラー処理 特になし．

(4) メソッド

- public Object getValue()

  機能概要 値を取り出す．

  機能説明 value を保持する Long オブジェクトを返す．

  入力データ なし．

出力データ 数値オブジェクト

処理方式 value を保持する Long オブジェクトを生成して，返す．

エラー処理 特になし．

- public boolean equals(int i)

  機能概要 値が与えられた整数値と一致するか否かを検査する．

  機能説明 値 value が与えられた整数値と一致するか否かを検査する．

  入力データ i (比較する値)

  出力データ 一致・不一致の真偽値

  処理方式 value==i の結果を返す．

  エラー処理 特になし．

- public boolean equalsDefault()

  機能概要 値が 0 と一致するか否かを検査する．

  機能説明 値 value が 0 と一致するか否かを検査する．

  入力データ なし．

  出力データ 一致・不一致の真偽値

  処理方式 value==0 の結果を返す．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 整数値をスタックトップに積むバイトコードを生成する．

  機能説明 整数値 value をスタックトップに積むバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

処理方式 スタックトップに value を積むために，opc_ldc2_w をオペコードとするバイトコードを生成する．

エラー処理 なし．

- public void print(PrintStream out)

  機能概要 値を読み易く出力する．

  機能説明 与えられた PrintStream に value を十進数表現で出力する．

  入力データ out (出力先の PrintStream)

  出力データ なし．

  処理方式 value の十進数表現に L を付けて，out.print メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 整数値の中身を出力する．

  機能説明 整数値を十進数で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 Long.toString メソッドを呼び出して，value を文字列に変換し，文字列 “L” と join して返す．

  エラー処理 特になし．

4.5.76 OpenJIT.frontend.tree.MethodExpression クラス

public class MethodExpression extends NaryExpression

(1) 概要

このクラスは，AST のメソッド呼び出しのノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Identifier id

  メソッドの識別子．

- ClassDefinition clazz

  メソッドのクラスの定義．

- FieldDefinition field

  メソッドのフィールドの定義．

(3) コンストラクタ

- public MethodExpression(int where, Expression right, Identifier id, Expression args[])

  機能概要 MethodExpression オブジェクトを生成する．

  機能説明 スーパークラスの NaryExpression クラスで定義されているコンストラクタを用いて，MethodExpression オブジェクトを生成・初期化する．

  入力データ where (行番号), right (右辺式), id (識別子), args[] (引数の式の配列)

  出力データ なし．

  処理方式 super メソッドを引数 METHOD (定数), where, Type.tError (定数), right，args で呼び出す．this.id に id を格納する．

  エラー処理 特になし．

- public MethodExpression(int where, Expression right, FieldDefinition field, Expression args[])

  機能概要 MethodExpression オブジェクトを生成する．

  機能説明 スーパークラスの NaryExpression クラスで定義されているコンストラクタを用いて，MethodExpression オブジェクトを生成・初期化する．

  入力データ where (行番号), right (右辺式), filed (フィールドの定義), args[] (引数の式の配列)

  出力データ なし．

  処理方式 super メソッドを引数 METHOD (定数), where, field.getType().getReturnType(), right，args で呼び出す．this.id に field.getName(), this.field に field，this.clazz に field.getClassDefinition() を格納する．

  エラー処理 特になし．

(4) メソッド

なし．

4.5.77 OpenJIT.frontend.tree.MultiplyExpression クラス

public class MultiplyExpression extends BinaryArithmeticExpression

(1) 概要

このクラスは，AST の * 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public MultiplyExpression(int where, Expression left, Expression right)

  機能概要 AST の * 式ノードを生成する．

  機能説明 スーパークラスの BinaryArithmeticExpression クラスで定義されているコンストラクタを用いて，* 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (* 式の左辺式の Expression オブジェクト)，right (* 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 MUL(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数の乗算を行い，IntExpression オブジェクトを生成する．

  入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理内容 a*b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数の乗算を行い，LongExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a*b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(float a, float b)

  機能概要 式を評価する．引数がともに float 型の場合．

  機能説明 float 型の引数の乗算を行い，FloatExpression オブジェクトを生成する．

  入力データ a (float 型), b (float 型)

  出力データ Expression オブジェクト．

  処理内容 a*b を計算し，FloatExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数がともに double 型の場合．

機能説明 double 型の引数の乗算を行い，DoubleExpression オブジェクトを生成する．

入力データ a (double 型), b (double 型)

出力データ Expression オブジェクト．

処理内容 a*b を計算し，DoubleExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression simpify()

機能概要 式の単純化を行う．

機能説明 右辺式または左辺式が 1 の場合，式を単純化する．

入力データ なし．

出力データ Expression オブジェクト．

処理方式 左辺式が 1 の場合，右辺式をこの * 式の値として返す．右辺値が 1 の場合，左辺式をこの * 式の値として返す．それ以外の場合，this オブジェクトを返す．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

機能概要 * を行うバイトコードを生成する．

機能説明 各型ごとに適切な * 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_imul (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 + 式の中身を出力する．

  機能説明 + 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.78 OpenJIT.frontend.tree.NaryExpression クラス

public class NaryExpression extends UnaryExpression

(1) 概要

このクラスは，AST の多項演算子の式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- public Expression args[]

  引数の式の配列．

(3) コンストラクタ

- NaryExpression(int op, int where, Type type, Expression right, Expression args[])

  機能概要 NaryExpression オブジェクトを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されているコンストラクタを用いて，NaryExpression オブジェクトの生成・初期化を行う．

  入力データ op (オペレータの種類), where (行番号), type (型), right (最も右辺の式), args (最も右辺の式を除いた式の配列)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, type, right で呼び出す．this.args に args を格納する．

  エラー処理 特になし．

(4) メソッド

- public Expression copyInline(Context ctx)

機能概要 メソッドインライニング用に多項演算式のコピーを生成する．

機能説明 メソッドインライニング用に多項演算式のコピーを生成する．

入力データ ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining をした多項演算式．

処理方式 clone メソッドを呼び出して，NaryExpression オブジェクトのコピー e を作る．right が null でなければ，右辺式の inlining を行うために，right.copyInline メソッドを呼び出し，戻り値を e.right に格納する．

args.length の大きさの Expression オブジェクトの配列を生成して，e.args に格納する．args を inlining するために，args[i].copyInline メソッドを呼び出して，戻り値を e.args[i] に格納する．

最後に e を返す．

エラー処理 特になし．

- public int costInline(int thresh)

機能概要 式の inlining のコストを計算する．

機能説明 right のコスト，args のコストを加算して，総コストを求める．

入力データ thresh (閾値)

出力データ inlining のコスト．

処理方式 right.costInline メソッドを呼び出して，右辺式の inlining コストを求め，それに 3 を加えたものを変数 cost に格納する．

args[i].costInline メソッドを呼び出して，cost に加算していく．cost が thresh を越えたら，処理を打ち切る．

最後に，cost の値を返す．

エラー処理 特になし．

- public void print(PrintStream out)

機能概要 多項演算式の中身を出力する．

機能説明 多項演算式を読み易く出力する．

入力データ out (String の出力先の PrintStream)

出力データ なし．

処理方式 多項演算式の中身を “(*opNames[op] right args[0] args[1] args[2] ...*)” という形式の文字列を生成して，PrintStream out に出力する．

エラー処理 特になし．

4.5.79 OpenJIT.frontend.tree.NegativeExpression クラス

public class NegativeExpression extends UnaryExpression

(1) 概要

このクラスは，AST の単項演算子 – のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public NegativeExpression(int where, Expression right)

  機能概要 AST の単項演算子 – の式ノードを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されているコンストラクタを用いて，– 式ノードの生成・初期化を行う．

  入力データ where (行番号)，right (– 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 NEG(定数), where, right.type, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 – 式の型を選択する．

  機能説明 フィールド type (スーパークラス UnaryExpression のフィールドを継承する) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式の型情報)

出力データ なし．

処理方式 tm で指定された型をこの式の型に設定し，右辺式 right の型をこの型に変換する．

エラー処理 特になし．

- Expression eval(int a)

機能概要 式を評価する．引数が int 型の場合．

機能説明 int 型の引数の符号反転を行い，IntExpression オブジェクトを生成する．

入力データ a (int 型)

出力データ Expression オブジェクト．

処理内容 –a を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

機能概要 式を評価する．引数が long 型の場合．

機能説明 long 型の引数の符号反転を行い，LongExpression オブジェクトを生成する．

入力データ a (long 型)

出力データ Expression オブジェクト．

処理内容 –a を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(float a, float b)

機能概要 式を評価する．引数が float 型の場合．

機能説明 float 型の引数の符号反転を行い，FloatExpression オブジェクトを生成する．

入力データ a (float 型)

出力データ Expression オブジェクト．

処理内容 –a を計算し，FloatExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数が double 型の場合．

  機能説明 double 型の引数の符号反転を行い，DoubleExpression オブジェクトを生成する．

  入力データ a (double 型)

  出力データ Expression オブジェクト．

  処理内容 –a を計算し，DoubleExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 右辺の式が単項演算式 – の場合，式を単純化する．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 right の式が単項演算式 – の場合，2 つの – を相殺して，right.right の式を返す．それ以外の場合，this オブジェクトを返す．

  エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 – を行うバイトコードを生成する．

  機能説明 各型ごとに適切な – 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 right の式をスタックに積むコードを生成するために，right.codeValue メソッドを呼び出す．opc_ineg (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 – 式の中身を出力する．

  機能説明 – 式を前置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 prefixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.80 OpenJIT.frontend.tree.NewArrayExpression クラス

public class NewArrayExpression extends NaryExpression

(1) 概要

このクラスは，AST の newarray 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public NewArrayExpression(int where, Expression right, Expression args[])

  機能概要 NewArrayExpression オブジェクトを生成する．

  機能説明 スーパークラスの NaryExpression クラスで定義されているコンストラクタを用いて，NewArrayExpression オブジェクトを生成する．

  入力データ where (行番号), right (右辺式), args[] (引数の式の配列)

  出力データ なし．

  処理方式 super メソッドを引数 NEWARRAY (定数), where. Type.tError (定数), right, args で呼び出す．

  エラー処理 特になし．

(4) メソッド

なし．

4.5.81 OpenJIT.frontend.tree.NewInstanceExpression クラス

public class NewInstanceExpression extends NaryExpression

(1) 概要

このクラスは，AST の newInstance 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- FieldDefinition field

  変数フィールドの定義．

(3) コンストラクタ

- public NewInstanceExpression(int where, Expression right, Expression args[])

  機能概要 NewInstanceExpression オブジェクトを生成する．

  機能説明 スーパークラスの NaryExpression クラスで定義されているコンストラクタを用いて，NewInstanceExpression オブジェクトを生成する．

  入力データ where (行番号), right (右辺式), args (newInstance の引数)

  出力データ なし．

  処理方式 super メソッドを引数 NEWINSTANCE (定数), where, Type.tError (定数), right, args で呼び出す．

  エラー処理 特になし．

- public NewInstanceExpression(int where, FieldDefinition field, Expression args[])

  機能概要 NewInstanceExpression オブジェクトを生成する．

  機能説明 スーパークラスの NaryExpression クラスで定義されているコンストラクタを用いて，NewInstanceExpression オブジェクトを生成する．

入力データ where (行番号), field (フィールドの定義), args (newInstance の引数)

出力データ なし．

処理方式 super メソッドを引数 NEWINSTANCE (定数), where, field.getType(), null, args で呼び出す．

エラー処理 特になし．

(4) メソッド

- int precedence()

機能概要 優先順位を返す．

機能説明 最下位の優先順位を返す．

入力データ なし．

出力データ 優先順位の値．

処理方式 100 を返す．

エラー処理 特になし．

4.5.82 OpenJIT.frontend.tree.Node クラス

public class Node implements Constants, Cloneable

(1) 概要

このクラスは，AST の全てのノードのスーパークラスであり，全てに共通するコード生成の機能を実現する．

(2) フィールド (変数)

- int op
- int where

(3) コンストラクタ

- Node(int op, int where)

  機能概要 Node オブジェクトを生成する．

  機能説明 this.op, this.where フィールドを初期化する．

  入力データ op (オペレータの種類), where (行番号)

  出力データ なし．

  処理方式 this.op に op, this.where に where を格納する．

  エラー処理 特になし．

(4) メソッド

- public int getOp()

  機能概要 オペレータのアクセッサメソッド．

  機能説明 op フィールドをアクセスする．

  入力データ なし．

出力データ オペレータ．

処理方式 op を返す．

エラー処理 特になし．

- public int getWhere()

  機能概要 行番号のアクセッサメソッド．

  機能説明 where フィールドをアクセスする．

  入力データ なし．

  出力データ 行番号．

  処理方式 where を返す．

  エラー処理 特になし．

- public Expression convert(Environment env, Context ctx, Type t, Expression e)

  機能概要 式の型変換を行う．

  機能説明 式の型変換を行う．暗黙の型変換に相当する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), t (変換する型), e (変換する式)

  出力データ 変換後の式．

  処理方式 e.type が t と等しい場合，すでに型が同じなので，e を返す．

  e.fitsType(env, t) メソッドを呼び出し，戻り値が true なら，ConvertExpression オブジェクトを生成して，返す．env.explicitCast メソッドを呼び出して，戻り値が true なら，explicit.cast.needed エラーを出力する．

  それ以外の場合，incompatible.type エラーを出力する．

  エラー処理 型変換ができない場合に，incompatible.type エラーを出力する．型変換できるが明示的な型変換が必要な場合に，explicat.cast.needed エラーを出力する．

- public void print(PrintStream out)

機能概要 ノードの中身を出力する．

機能説明 ノードの中身を出力する．

入力データ out (String の出力先の PrintStream)

出力データ なし．

処理方式 CompilerError(“print”) を throw する．

エラー処理 無条件で，CompilerError(“print”) を出力する．

- public Object clone()

  機能概要 Node オブジェクトのクローンを作る．

  機能説明 スーパークラスの clone メソッドを呼び出して，Node オブジェクトのクローンを生成する．

  入力データ なし．

  出力データ クローンオブジェクト．

  処理方式 super.clone メソッドを呼び出し，戻り値を返す．

  エラー処理 特になし．

- public String toString()

  機能概要 ノードの中身を文字列に直す．

  機能説明 ノードの中身を文字列に直す．

  入力データ なし．

  出力データ 文字列．

  処理方式 新しく ByteArrayOutputStream bos を生成する．print(new PrintStream(bos)) メソッドを呼び出す．bos.toString メソッドを呼び出し，戻り値を返す．

  エラー処理 特になし．

4.5.83 OpenJIT.frontend.tree.NotEqualExpression クラス

public class NotEqualExpression extends BinaryEqualityExpression

(1) 概要

このクラスは，AST の != 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public NotEqualExpression(int where, Expression left, Expression right)

  機能概要 AST の != 式ノードを生成する．

  機能説明 スーパークラスの BinaryEqualityExpression クラスで定義されているコンストラクタを用いて，!= 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (!= 式の左辺式の Expression オブジェクト)，right (!= 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 NE(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数で != 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理内容 a!=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数で != 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a!=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(float a, float b)

  機能概要 式を評価する．引数がともに float 型の場合．

  機能説明 float 型の引数で != 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (float 型), b (float 型)

  出力データ Expression オブジェクト．

  処理内容 a!=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数がともに double 型の場合．

機能説明 double 型の引数で != 演算を行い，BooleanExpression オブジェクトを生成する．

入力データ a (double 型), b (double 型)

出力データ Expression オブジェクト．

処理内容 a!=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(boolean a, boolean b)

  機能概要 式を評価する．引数がともに boolean 型の場合．

  機能説明 boolean 型の引数で != 演算を行い，BooleanExpression オブジェクトを生成する．

  入力データ a (boolean 型), b (boolean 型)

  出力データ Expression オブジェクト．

  処理内容 a!=b を計算し，BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 左辺式が定数で，右辺式が定数でない場合に式の単純化を行う．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式が定数で，右辺式が定数でない場合，右辺式と左辺式を入れ換えた NotEqualExpression を生成して返す．そうでない場合，this を返す．

  エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

機能概要 != 式の条件分岐を行うバイトコードを生成する．

機能説明 各型ごとに適切な条件分岐用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

出力データ なし．

処理方式 1. left.codeValue メソッドを呼び出し，左辺式をスタックトップに積むバイトコードを生成する．

2. 左辺式の型を検査し，boolean または int 型の場合は，右辺式が 0 であれば 7 へ．非零であれば，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．さらに whenTrue が True/False に従って，opc_if_icmpne, opc_if_icmpeq をオペコードとするバイトコードを生成して終了．
3. 左辺式の型が long の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_lcmp をオペコードとするバイトコードを生成する．7 へ．
4. 左辺式の型が float の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_fcmpl をオペコードとするバイトコードを生成する．7 へ．
5. 左辺式の型が double の場合，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．次に opc_dcmpl をオペコードとするバイトコードを生成する．7 へ．
6. 左辺式の型が，クラスオブジェクト，配列オブジェクトの場合，右辺式が 0 であれば，whenTrue の True/False に従って，opc_ifnonnull, opc_ifnull をオペコードとするバイトコードを生成して，7 へ．非零であれば，right.codeValue メソッドを呼び出し，右辺式をスタックトップに積むバイトコードを生成する．さらに whenTrue の True/False に従って，opc_if_acmpne, opc_if_acmpeq をオペコードとするバイトコー

ドを生成して，7へ．

7. whenTrue の True/False に従って，opc_ifne, opc_ifeq をオペコードとするバイトコードを生成して終了．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 != 式の中身を出力する．

  機能説明 != 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.84 OpenJIT.frontend.tree.NotExpression クラス

public class NotExpression extends UnaryExpression

(1) 概要

このクラスは，AST の!式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public NotExpression(int where, Expression right)

  機能概要 NotExpression オブジェクトを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されているコンストラクタを用いて，NotExpression オブジェクトの生成・初期化を行う．

  入力データ where (行番号), right (右辺式)

  出力データ なし．

  処理方式 super メソッドを引数 NOT (定数), where, Type.tBoolean (定数), right で呼び出す．

  エラー処理 特になし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 !式の型を選択する．

  機能説明 右辺式の型を boolean 型にする．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

出力データ なし．

処理方式 右辺式の型を boolean 型にするために，convert メソッドを呼び出し，戻り値を right に格納する．

エラー処理 特になし．

- public void checkCondition(Environment env, Context ctx, int vset, Hashtable exp, ConditionVars cvars)

  機能概要 !式の条件を検査する．

  機能説明 式の右辺式の検査を行い，右辺式の型を boolean 型に変換する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), cvars (条件の評価値オブジェクト)

  出力データ なし．

  処理方式 右辺式の検査を行うために，right.checkCondition メソッドを呼び出す．右辺式を boolean 型に変換するために，convert メソッドを呼び出し，戻り値を right に格納する．

  true と false の条件を入れ換えるために，cvars.vsFalse の値と cvars.vsTrue の値を入れ換える．

  エラー処理 特になし．

- Expression eval(boolean a)

  機能概要 !式の評価を行う．

  機能説明 !a を計算して，BooleanExpression オブジェクトを返す．

  入力データ a (boolean 型の値)

  出力データ Expression オブジェクト．

  処理方式 !a を計算し，その値を持つ BooleanExpression オブジェクトを生成して，返す．

エラー処理 特になし．

- Expression simpify()

  機能概要 !式の単純化を行う．

  機能説明 !の相殺，条件の反転により，式の単純化を行って，返す．

  入力データ なし．

  出力データ 単純化を行った式．

  処理方式 right.op が NOT の場合，!演算子が相殺するので，right.right を返す．

  right.op が EQ, NE, LT, LE, GT, GE 以外の場合，this を返す．

  right.left.type が real 型の場合，negate することができないので，this を返す．

  right.op が EQ の場合，NotEqualExpression を生成して，返す．

  right.op が NE の場合，EqualExpression を生成して，返す．

  right.op が LT の場合，GreaterOrEqualExpression を生成して，返す．

  right.op が LE の場合，GreaterExpression を生成して，返す．

  right.op が GT の場合，LessOrEqualExpression を生成して，返す．

  right.op が GE の場合，LessExpression を生成して，返す．

  それ以外の場合，this を返す．

  エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

  機能概要 !式を実行するバイトコードを生成する．

  機能説明 whenTrue を反転して，codeBranch メソッドを呼び出す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

出力データ なし．

処理方式 !whenTrue の時に lbl にジャンプするバイトコードを生成するために，
right.codeBranch(env, ctx, asm, lbl, !whenTrue) を呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 ! 式の中身を出力する．

機能説明 ! を前置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.85 OpenJIT.frontend.tree.NullExpression クラス

public class NullExpression extends ConstantExpression

(1) 概要

このクラスは，AST の Null ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public NullExpression(int where)

  機能概要 NullExpression オブジェクトを生成する．

  機能説明 スーパークラスの ConstantExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号)

  出力データ なし．

  処理方式 super メソッドを引数 NULL(定数), where, Type.tNull (定数) で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public boolean equals(int i)

  機能概要 値が与えられた整数値と一致するか否かを検査する．

  機能説明 与えられた整数値が 0 と一致するか否かを検査する．

  入力データ i (比較する値)

出力データ 一致・不一致の真偽値

処理方式 i==0 の結果を返す．

エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 null 値をスタックトップに積むバイトコードを生成する．

  機能説明 null 値をスタックトップに積むバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 スタックトップに null 値を積むために，opc_aconst_null をオペコードとするバイトコードを生成する．

  エラー処理 なし．

- public void print(PrintStream out)

  機能概要 値を読み易く出力する．

  機能説明 与えられた PrintStream に null を出力する．

  入力データ out (出力先の PrintStream)

  出力データ なし．

  処理方式 out.print メソッドで “null” を呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 null 値の中身を出力する．

  機能説明 null 値を文字列で読み易く出力する．

  入力データ なし．

出力データ String 型の文字列．

処理方式 文字列 “null” を返す．

エラー処理 特になし．

4.5.86 OpenJIT.frontend.tree.OrExpression クラス

public class OrExpression extends BinaryLogicalExpression

(1) 概要

このクラスは，AST の OR 演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public OrExpression(int where, Expression left, Expression right)

  機能概要 AST の Or 演算子のノードを生成する．

  機能説明 スーパークラスの BinaryLogicalExpression クラスで定義されているコンストラクタを用いて，Or 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (Or 式の左辺式の Expression オブジェクト)，right (Or 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 OR(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- public void checkCondition(Environment env, Context ctx, long vset, Hashtable exp, ConditionVars cvars)

  機能概要 OR 式の条件を検査する．

  機能説明 OR 式の右辺式，左辺式の評価を行い，それぞれの式を BooleanExpression に変換する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル), cvars (条件の評価値オブジェクト)

出力データ なし．

処理方式 左辺式の true/false を検査して，左辺式を BooleanExpression に変換する．次に右辺式の true/false を検査して，右辺値を BooleanExpression に変換する． cvars には，条件が静的に評価可能な場合についてのみ，値がセットされる．

エラー処理 特になし．

- Expression eval(boolean a, boolean b)

機能概要 式を評価する．引数がともに boolean 型の場合．

機能説明 boolean 型の引数の論理和を行い， BooleanExpression オブジェクトを生成する．

入力データ a (boolean 型), b (boolean 型)

出力データ Expression オブジェクト．

処理内容 a||b を計算し， BooleanExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression simpify()

機能概要 式の単純化を行う．

機能説明 左辺式または右辺式が true または false の場合，式を単純化する．

入力データ なし．

出力データ Expression オブジェクト．

処理方式 左辺式が true または右辺式 false の場合，左辺式をこの OR 式の値として返す．左辺式が false または右辺式が true の場合，右辺式をこの AND 式の値として返す．それ以外の場合， this オブジェクトを返す．

エラー処理 特になし．

- void codeBranch(Environment env, Context ctx, Assembler asm, Label lbl, boolean whenTrue)

  機能概要 OR 式を実行するバイトコードを生成する．

  機能説明 OR 式を左辺式，右辺式の条件の成立に従って分岐するバイトコードに変換する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), lbl (ジャンプ先のラベル), whenTrue (条件成立時にジャンプするか，非成立時にジャンプするかのフラグ)

  出力データ なし．

  処理方式 whenTrue が True の場合， left を評価して true であれば lbl にジャンプするバイトコードを生成するために，left.codeBranch メソッドを呼び出す．次に right を評価して true であれば lbl にジャンプするバイトコードを生成するために，right.codeBranch メソッドを呼び出す．

  whenTrue が False の場合，新しいラベル lbl2 を生成する． left を評価して true であれば lbl2 にジャンプするバイトコードを生成するために，left.codeBranch メソッドを呼び出す．次に right を評価して false であれば lbl にジャンプするバイトコードを生成するために， right.codeBranch メソッドを呼び出す．最後にラベル l2 を挿入する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 OR 式の中身を出力する．

  機能説明 OR 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.87 OpenJIT.frontend.tree.PositiveExpression クラス

public class NegativeExpression extends UnaryExpression

(1) 概要

このクラスは，AST の単項演算子 + のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public PositiveExpression(int where, Expression right)

  機能概要 AST の単項演算子 + の式ノードを生成する．

  機能説明 スーパークラスの UnaryExpression クラスで定義されているコンストラクタを用いて，+ 式ノードの生成・初期化を行う．

  入力データ where (行番号)，right (+ 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 POS(定数), where, right.type, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- void selectType(Environment env, Context ctx, int tm)

  機能概要 + 式の型を選択する．

  機能説明 フィールド type (スーパークラス UnaryExpression のフィールドを継承する) を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式の型情報)

出力データ なし．

処理方式 tm で指定された型をこの式の型に設定し，右辺式 right の型をこの型に変換する．

エラー処理 特になし．

- Expression simpify()

機能概要 式の単純化を行う．

機能説明 + 式は単に冗長なので，右辺式のみに単純化する．

入力データ なし．

出力データ Expression オブジェクト．

処理方式 right を返す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 + 式の中身を出力する．

機能説明 + 式を前置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 prefixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.88 OpenJIT.frontend.tree.PostDecExpression クラス

public class PostDecExpression extends IncDecExpression

(1) 概要

このクラスは，AST の後置演算子– –のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public PostDecExpression(int where, Expression right)

  機能概要 AST の後置演算子– –の式ノードを生成する．

  機能説明 スーパークラスの IncDecExpression クラスで定義されているコンストラクタを用いて，後置演算子– –式ノードの生成・初期化を行う．

  入力データ where (行番号)，right (後置演算子– –式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 POSTDEC(定数), where, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 後置演算子– –式の値を計算するバイトコードを生成する．

  機能説明 後置演算子– –式の値を計算するバイトコードを生成する．計算後の値をスタックトップに保持する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 codeIncDec(env, ctx, asm, false, false, true) を呼び出す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 後置演算子 – – 式の値を計算するバイトコードを生成する．

  機能説明 後置演算子 – – 式の値を計算するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 codeIncDec(env, ctx, asm, false, false, false) を呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 – – 式の中身を出力する．

  機能説明 – – 式を後置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 postfixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.89 OpenJIT.frontend.tree.PostIncExpression クラス

public class PostIncExpression extends IncDecExpression

(1) 概要

このクラスは，AST の後置演算子 ++ のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public PostDecExpression(int where, Expression right)

  機能概要 AST の後置演算子 ++ の式ノードを生成する．

  機能説明 スーパークラスの IncDecExpression クラスで定義されているコンストラクタを用いて，後置演算子 ++ 式ノードの生成・初期化を行う．

  入力データ where (行番号)，right (後置演算子 ++ 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 POSTINC(定数), where, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 後置演算子 ++ 式の値を計算するバイトコードを生成する．

  機能説明 後置演算子 ++ 式の値を計算するバイトコードを生成する．計算後の値をスタックトップに保持する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 codeIncDec(env, ctx, asm, true, false, true) を呼び出す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 後置演算子 ++ 式の値を計算するバイトコードを生成する．

  機能説明 後置演算子 ++ 式の値を計算するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 codeIncDec(env, ctx, asm, true, false, false) を呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 ++ 式の中身を出力する．

  機能説明 ++ 式を後置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 postfixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.90 OpenJIT.frontend.tree.PreDecExpression クラス

public class PreDecExpression extends IncDecExpression

(1) 概要

このクラスは，AST の前置演算子– –のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public PreDecExpression(int where, Expression right)

  機能概要 AST の前置演算子– –の式ノードを生成する．

  機能説明 スーパークラスの IncDecExpression クラスで定義されているコンストラクタを用いて，前置演算子– –式ノードの生成・初期化を行う．

  入力データ where (行番号)，right (前置演算子– –式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 PREDEC(定数), where, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 前置演算子– –式の値を計算するバイトコードを生成する．

  機能説明 前置演算子– –式の値を計算するバイトコードを生成する．計算後の値をスタックトップに保持する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 codeIncDec(env, ctx, asm, false, true, true) を呼び出す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

機能概要 前置演算子 – – 式の値を計算するバイトコードを生成する．

機能説明 前置演算子 – – 式の値を計算するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 codeIncDec(env, ctx, asm, false, true, false) を呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 – – 式の中身を出力する．

機能説明 – – 式を前置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 prefixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.91 OpenJIT.frontend.tree.PreIncExpression クラス

public class PreIncExpression extends IncDecExpression

(1) 概要

このクラスは，AST の前置演算子 ++ のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public PreIncExpression(int where, Expression right)

  機能概要 AST の前置演算子 ++ の式ノードを生成する．

  機能説明 スーパークラスの IncDecExpression クラスで定義されているコンストラクタを用いて，前置演算子 ++ 式ノードの生成・初期化を行う．

  入力データ where (行番号)，right (前置演算子 ++ 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 PREINC(定数), where, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 前置演算子 ++ 式の値を計算するバイトコードを生成する．

  機能説明 前置演算子 ++ 式の値を計算するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 codeIncDec(env, ctx, asm, true, true, true) を呼び出す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 前置演算子 ++ 式の値を計算するバイトコードを生成する．

  機能説明 前置演算子 ++ 式の値を計算するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 codeIncDec(env, ctx, asm, true, true, false) を呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 ++ 式の中身を出力する．

  機能説明 ++ 式を前置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 prefixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.92 OpenJIT.frontend.tree.RemainderExpression クラス

public class RemainderExpression extends DivRemExpression

(1) 概要

このクラスは，AST の % 演算式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public RemainderExpression(int where, Expression left, Expression right)

  機能概要 AST の % 式ノードを生成する．

  機能説明 スーパークラスの DivRemExpression クラスで定義されているコンストラクタを用いて，% 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (% 式の左辺式の Expression オブジェクト)，right (% 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 REM(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数の演算を行い，IntExpression オブジェクトを生成する．

  入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理内容 a%b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

機能概要 式を評価する．引数がともに long 型の場合．

機能説明 long 型の引数の演算を行い，LongExpression オブジェクトを生成する．

入力データ a (long 型), b (long 型)

出力データ Expression オブジェクト．

処理内容 a%b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(float a, float b)

機能概要 式を評価する．引数がともに float 型の場合．

機能説明 float 型の引数の演算を行い，FloatExpression オブジェクトを生成する．

入力データ a (float 型), b (float 型)

出力データ Expression オブジェクト．

処理内容 a%b を計算し，FloatExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(double a, double b)

機能概要 式を評価する．引数がともに double 型の場合．

機能説明 double 型の引数の演算を行い，DoubleExpression オブジェクトを生成する．

入力データ a (double 型), b (double 型)

出力データ Expression オブジェクト．

処理内容 a%b を計算し，DoubleExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 % を行うバイトコードを生成する．

  機能説明 各型ごとに適切な % 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 opc_irem (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 % 式の中身を出力する．

  機能説明 % 式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.93 OpenJIT.frontend.tree.ReturnStatement クラス

public class ReturnStatement extends Statement

(1) 概要

このクラスは，AST の return 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- public Expression expr

  戻り値の式．

(3) コンストラクタ

- public ReturnStatement(int where, Expression expr)

  機能概要 ReturnStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，ReturnStatement オブジェクトを生成・初期化する．

  入力データ where (行番号), expr (戻り値の式)

  出力データ なし．

  処理方式 super メソッドを引数 RETURN (定数)，where で呼び出す．this.expr に expr を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 return 文の検査を行う．

  機能説明 return 文の戻り値の式 expr について検査を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 expr が null でなければ，式 expr の検査を行うために，expr.checkValue メソッドを呼び出す．-1 を返す．

エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

機能概要 return 文の inlining を行う．

機能説明 return 文の expr 部分の inlining を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の文．

処理方式 expr が null でなければ，expr を inlining するために expr.inlineValue メソッドを呼び出し，戻り値を expr に格納する．

this を返す．

エラー処理 特になし．

- public int costInline(int thresh)

機能概要 return 文の inlining のコストを計算する．

機能説明 return 文の inlining のコストを計算する．

入力データ thresh (閾値)

出力データ inlining のコストを計算する．

処理方式 expr.costInline メソッドを呼び出し，expr 部分の inlining のコストを求め，その値に 1 を加えて返す．

エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

機能概要 コピーされた return 文の inlining を行う．

機能説明 メソッドインライン展開用のコピーされた return 文の inlining を行う．

入力データ ctx (コンテクストを格納する Context オブジェクト), valNeeded (戻り値が必要か否かのフラグ)

出力データ inlining 後の式．

処理方式 expr が null でなければ，expr を inlining するために expr.inlineValue メソッドを呼び出し，戻り値を e に格納する．expr が null ならば，e は null とする．

valNeeded が false で，e が null でない場合，e を含む ExpressionStatement オブジェクトと，InlineReturnStatement オブジェクトを生成し，この 2 つのオブジェクトを要素として持つ，Statement オブジェクトの配列 body[] を生成する．body[] を含む CompoundStatement オブジェクトを生成して，返す．

それ以外の場合，e を含む InlineReturnStatement オブジェクトを生成して，返す．

エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

機能概要 return 文を実行するバイトコードを生成する．

機能説明 return 文を実行するバイトコードを生成する．finally ブロックの処理を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 expr が null の場合，finally ブロックを実行するバイトコードを生成するために，codeFinally メソッドを呼び出す．次に，opc_return をオペコードとするバイトコードを生成する．

expr が null でない場合，式 expr を計算して，スタックトップに結果を保持するバイトコードを生成するために，expr.codeValue メソッドを呼び出す．次に，codeFinally メソッドを呼び出す．最後に，opc_ireturn+expr.type.getTypeCode を計算して，その戻り値をオペコードとするバイトコードを生成する．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

  機能概要 return 文の中身を出力する．

  機能説明 return 文の中身を読みやすくインデントをつけて出力する．

  入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

  出力データ 特になし．

  処理方式 super.print(out, indent) を呼び出して，スーパークラスで定義された print メソッドで，必要な数のインデントを行う．“return *expr*” の文字列を PrintStream out に出力する．

  エラー処理 特になし．

4.5.94 OpenJIT.frontend.tree.ShiftLeftExpression クラス

public class ShiftLeftExpression extends BinaryShiftExpression

(1) 概要

このクラスは，AST の << 演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public ShiftLeftExpression(int where, Expression left, Expression right)

  機能概要 AST の << 式ノードを生成する．

  機能説明 スーパークラスの BinaryShiftExpression クラスで定義されているコンストラクタを用いて，<< 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (<< 式の左辺式の Expression オブジェクト)，right (<< 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 LSHIFT(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数が共に int 型の場合．

  機能説明 int 型の引数の左シフトを行い，IntExpression オブジェクトを生成する．

入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理方式 a<<b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数の左シフトを行い，LongExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理方式 a<<b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 左辺式または右辺式が 0 の場合，式を単純化する．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式または右辺式が 0 の場合，left を返す．それ以外の場合は this を返す．

  エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 << を行うバイトコードを生成する．

機能説明 各型ごとに適切な << 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_ishl (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 << 式の中身を出力する．

機能説明 << 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.95 OpenJIT.frontend.tree.ShiftRightExpression クラス

public class ShiftRightExpression extends BinaryShiftExpression

(1) 概要

このクラスは，AST の >> 演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public ShiftRightExpression(int where, Expression left, Expression right)

  機能概要 AST の >> 式ノードを生成する．

  機能説明 スーパークラスの BinaryShiftExpression クラスで定義されているコンストラクタを用いて，>> 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (>> 式の左辺式の Expression オブジェクト)，right (>> 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 RSHIFT(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数が共に int 型の場合．

  機能説明 int 型の引数の右シフトを行い，IntExpression オブジェクトを生成する．

入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理方式 a>>b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数の右シフトを行い，LongExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理方式 a>>b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 左辺式または右辺式が 0 の場合，式を単純化する．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式または右辺式が 0 の場合，left を返す．それ以外の場合は this を返す．

  エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 >> を行うバイトコードを生成する．

機能説明 各型ごとに適切な >> 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_ishr (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 >> 式の中身を出力する．

機能説明 >> 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.96 OpenJIT.frontend.tree.ShortExpression クラス

public class ShortExpression extends IntegerExpression

(1) 概要

このクラスは，AST の short 型整数ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public ShortExpression(int where, short value)

  機能概要 ShortExpression オブジェクトを生成する．

  機能説明 スーパークラスの IntegerExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), value (保持する値)

  出力データ なし．

  処理方式 super メソッドを引数 SHORTVAL(定数), where, Type.tShort(定数), value で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public void print(PrintStream out)

  機能概要 値を読み易く出力する．

  機能説明 与えられた PrintStream に value を十進数表現で出力する．

  入力データ out (出力先の PrintStream)

出力データ なし．

処理方式 value の十進数表現に s を付けて，out.print メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 整数値の中身を出力する．

  機能説明 整数値を十進数で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 postfixForm メソッドを引数 “s” で呼び出す．

  エラー処理 特になし．

4.5.97 OpenJIT.frontend.tree.Statement クラス

public class Statement extends Node

(1) 概要

このクラスは，BreakStatement クラス，CaseStatement クラス，CatchStatement クラス，CompoundStatement クラス，ContinueStatement クラス，DeclarationStatement クラス，DoStatement クラス，ExpressionStatement クラス，FinallyStatement クラス，ForStatement クラス，IfStatement クラス，InlineReturnStatement クラス，ReturnStatement クラス，SwitchStatement クラス，SynchronizedStatement クラス，ThrowStatement クラス，TryStatement クラス，VarDeclarationStatement クラス，WhileStatement クラスのスーパークラスであり，これらに共通するコード生成の機能を実現する．

(2) フィールド (変数)

- public static final long DEAD_END = (long)1 << 63
- Identifier labels[] = null
- int[] r_gen

  データフロー解析−到達定義の解析で得られた情報 gen。
- int[] r_kill

  データフロー解析−到達定義の解析で得られた情報 kill。
- int[] r_in

  データフロー解析−到達定義の解析で得られた情報 in。
- int[] r_out

  データフロー解析−到達定義の解析で得られた情報 out。
- int[] a_gen

  データフロー解析−利用可能な式の解析で得られた情報 gen。

- int[] a_kill

  データフロー解析-利用可能な式の解析で得られた情報 kill。

- int[] a_in

  データフロー解析-利用可能な式の解析で得られた情報 in。

- int[] a_out

  データフロー解析-利用可能な式の解析で得られた情報 out。

- int[] l_def

  データフロー解析-生きている式の解析で得られた情報 def。

- int[] l_use

  データフロー解析-生きている式の解析で得られた情報 use。

- int[] l_in

  データフロー解析-生きている式の解析で得られた情報 in。

- int[] l_out

  データフロー解析-生きている式の解析で得られた情報 out。

- int fwhere

  データフロー解析のトップレベルの Statement 内での行番号。

- int blocks

  データフロー解析のトップレベルの Statement 内に含まれる Statement オブジェクトの数。

- int size

  データフロー解析の結果を格納するために必要となる配列のサイズ。

- Hashtable leftexptable

  データフロー解析のための情報を保持する。具体的には式において左辺に現れる変数と行番号の対応の情報を持つ。

- Hashtable rightexptable

  データフロー解析のための情報を保持する。具体的には式において右辺に現れる変数と行番号の対応の情報を持つ。

- Hashtable leftlinetable

  データフロー解析のための情報を保持する。具体的には行番号と式において左辺に現れる変数との対応の情報を持つ。

- Hashtable rightlinetable

  データフロー解析のための情報を保持する。具体的には行番号と式において右辺に現れる変数との対応の情報を持つ。

(3) コンストラクタ

- Statement(int op, int where)

  機能概要 Statement オブジェクトを生成する．

  機能説明 スーパークラスの Node クラスで定義されているコンストラクタを用いて，オブジェクトの生成・初期化を行う．

  入力データ op (文の種類), where (行番号)

  出力データ なし．

  処理方式 super メソッドを引数 op, where で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public void setLabel(Environment env, Expression e)

  機能概要 文にラベルを設定する．

  機能説明 文に与えられたラベル式をラベルとして設定する．

  入力データ env (環境を格納する Environment オブジェクト), e (ラベル式)

  出力データ なし．

処理方式 e.op が IDENT (定数) に等しいか否か検査する．等しくなければ，invalid.label エラーを出力する．

新しい Indentifier オブジェクトの配列 (サイズ 1) を生成して，labels に格納する．labels[0] に e.id を格納する．

エラー処理 e がラベル式でないときに，invalid.label エラーを出力する．

- public long checkMethod(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 文を検査する．

  機能説明 メソッド内を検査する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値

  処理方式 check メソッドを呼び出し，戻り値を vset に格納する．ctx.field.getType().getReturnType() メソッドを呼び出して得られる return の型が void 型でなく，かつ vset & DEAD_END が 0 (文の最後まで return 文が現れない) の場合，return.required.at.end エラーを出力する．そうでない場合は，vset を返す．

  エラー処理 return type が void 型以外の場合にメソッドの最後まで return 文がなければ，return.required.at.end エラーを出力する．

- long checkDeclaration(Environment env, Context ctx, long vset, int mod, Type t, Hashtable exp)

  機能概要 文を検査する．

  機能説明 宣言文の検査を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値

処理方式 CompilerError(“checkDeclaration”) を throw する．

エラー処理 無条件で，CompilerError(“checkDeclaration”) を throw する．

- long check(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 文を検査する．

  機能説明 宣言文の検査を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値

  処理方式 CompilerError(“check”) を throw する．

  エラー処理 無条件で，CompilerError(“check”) を throw する．

- long reach(Environment env, long vset)

  機能概要

  機能説明

  入力データ env (環境を格納する Environment オブジェクト), vset (条件式の評価値)

  出力データ 条件の評価値

  処理方式 vset & DEAD_END (定数) が 0 でなければ，vset &= ~DEAD_END をして，vset の DEAD_END フラグをオフにする．

  vset を返す．

  エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

  機能概要 文の inlining を行う．

  機能説明 文の inlining を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 this を返す．

エラー処理 特になし．

- public Statement eliminate(Environment env, Statement s)

  機能概要 この文を削除する．

  機能説明 この文を削除する．ラベルを付けていない文のみ可能．

  入力データ env (環境を格納する Environment オブジェクト), s (対象の文)

  出力データ 削除後の Satement オブジェクト．

  処理方式 s が null でなく，かつ labels が null でない場合，s を要素とする Statement の配列オブジェクト args に格納する．args を Statement 列とする CompoundStatement オブジェクトを生成して，s に格納する．labels を s.labels に格納する．

  s を返す．

  エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 文を実行するバイトコードを生成する．

  機能説明 文を実行するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 CompilerError(“code”) を throw する．

  エラー処理 無条件で CompilerError(“code”) を throw する．

- void codeFinally(Environment env, Context ctx, Assembler asm, Context stopctx, Type save)

  機能概要 finally 文を呼ぶためのバイトコードを生成する．

  機能説明 break, continue, return によって，finally ブロックを実行する必要がある場合に，それを実行するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト), stopctx (停止するコンテクスト), save (保存しておく型)

  出力データ なし．

  処理方式 まず，boolean 型の変数 haveCleanup, haveNonLocalFinally を false にする．

  ctx を後ろから順にたどっていって，そのコンテクストが stopctx に等しいか，あるいは先頭に達するまで以下の処理を行う．

  ctx.node が null であれば，続行． ctx.node.op が SYNCHRONIZED (定数) であれば， haveCleanup を true にする．そうでなく， ctx.node.op が FINALLY (定数) であれば， havaCleanup を true にする． c.node を FinallyStatement st に格納し， st.finallyFinishes が false なら， haveNonLocalFinally を true にし， break する．

  次に， haveCleanup が false なら，以下の処理は不要なので， return ．

  save が null でない場合， haveNonLocalFinally が false なら，戻り値を保存しておくために， opc_istore をオペコードとするバイトコード生成する．true なら，save の型が void 型の時は break ，double, long 型の時は opc_pop2 をオペコードとするバイトコードを生成し，それ以外の型の時は opc_pop をオペコードとするバイトコードを生成する．

  再度 ctx を後ろから順にたどっていって，そのコンテクストが stopctx に等しいか，あるいは先頭に達するまで以下の処理を行う．

  ctx.node が null であれば，続行． ctx.node.op が SYNCHRONIZED (定数) であれば， opc_jsr をオペコードとするバイトコードを生成する．FINALLY (定数) であれば， c.node を FinallyStatement st に格納し， st.finallyFinishes

が true の時には，opc_jsr をオペコードとするバイトコードを生成し，そうでない時には，opc_goto をオペコードとするバイトコードを生成する．

最後に，戻り値をレジスタからスタックに書き戻すために，opc_iload をオペコードとするバイトコードを生成する．

エラー処理 特になし．

- public boolean hasLabel (Identifier lbl)

  機能概要 ラベルの検査を行う．

  機能説明 この文が与えられたラベルを持っているかどうかを検査する．

  入力データ lbl (検査するラベル)

  出力データ 真偽値．

  処理方式 this.labels を Identifier の配列オブジェクト labels[] に格納する．labels が null でなければ，labels の各要素が lbl に等しいか否か検査し，等しければ true を返す．

  false を返す．

  エラー処理 特になし．

- public boolean firstConstructor()

  機能概要 文の最初の処理がコンストラクタ呼び出しか否か検査する．

  機能説明 文の最初の処理がコンストラクタ呼び出しか否か検査する．

  入力データ なし．

  出力データ 真偽値．

  処理方式 false を返す．

  エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

  機能概要 文の inlining を行う．

  機能説明 コピーされた文の inlining を行う．

入力データ ctx (コンテクストを格納するContextオブジェクト), valNeeded (式を評価した後の値がスタック上に必要かどうかのフラグ)

出力データ inliningされた文．

処理方式 cloneメソッドを呼び出し，戻り値を返す．

エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 Inliningのコストを計算する．

  機能説明 Inliningのコストを計算する．

  入力データ thresh (閾値)

  出力データ Inliningのコスト．

  処理方式 threshを返す．

  エラー処理 特になし．

- public void initFlowInfo(int num, Hashtable left, Hashtable right, Hashtable leftl, Hashtable rightl)

  機能概要 データフロー解析のための情報を初期化する。

  機能説明 データフロー解析のための情報を得て、自分のフィールドにセットする。また、データフロー解析の結果を格納する領域を確保する。

  入力データ num(トップレベルの Statement オブジェクトに含まれる Statement オブジェクトの数)、 left(左辺に現れる変数と行番号の対応の情報)、 right(右辺に現れる変数と行番号の対応の情報)、 leftl(行番号と左辺に現れる変数との対応の情報)、 rightl(行番号と右辺に現れる変数との対応の情報)

  出力データ なし

  処理方式 データフロー解析に必要な情報を自分のフィールド (引数の順にblocks、 eftexptable、 rightexptable、 leftlinetable、 rightlinetable) にセットし、データフロー解析の結果を格納する配列 (r_gen、 r_kill、 r_in、

r_out、 a_gen、 a_kill、 a_in、 a_out、 l_def、 l_use、 l_in、 l_out) を確保する。

エラー処理 なし

- public int[] reachingAnalysis(int[] in)

  機能概要 到達定義の解析を行う。

  機能説明 自分自身についてデータフロー解析の 1 つである到達定義の解析を行う。

  入力データ in(前の Statement の out)

  出力データ 解析後の out

  処理方式 Expression オブジェクトを持たない Statement のサブクラスのためのメソッドであるため、入力がそのまま出力となる。 Expression オブジェクトを持つ Statement のサブクラスはこのメソッドをオーバーライドする。

  エラー処理 なし

- public int[] availableAnalysis(int[] in)

  機能概要 利用可能な式の解析

  機能説明 自分自身についてデータフロー解析の 1 つである利用可能な式の解析を行う。

  入力データ in(前の Statement の out)

  出力データ 解析後の out

  処理方式 Expression オブジェクトを持たない Statement のサブクラスのためのメソッドであるため、入力がそのまま出力となる。 Expression オブジェクトを持つ Statement のサブクラスはこのメソッドをオーバーライドする。

  エラー処理 なし

- public int[] livenessAnalysis(int[] out)

  機能概要 生きている式の解析

機能説明 自分自身についてデータフロー解析の 1 つである生きている式の解析を行う。

入力データ out(次の Statement の in)

出力データ 解析後の in

処理方式 Expression オブジェクトを持たない Statement のサブクラスのためのメソッドであるため、入力がそのまま出力となる。Expression オブジェクトを持つ Statement のサブクラスはこのメソッドをオーバーライドする。

エラー処理 なし

- public int[] detectFixedPoint(int[] in)

機能概要 不動点の検出を行う。

機能説明 自分自身について不動点の検出のための解析と不動点の検出を行う。

入力データ in(次の Statement の in)

出力データ 解析後の out

処理方式 まず、自分のフィールドに Statement オブジェクトもしくは Statemnet オブジェクトの配列を持っていれば、それらすべての Statement オブジェクトに対して detectFixedPoint() メソッドを呼び出し、それらの解析結果を得た上で自分の Statemnet オブジェクトについて不動点の検出のための解析と不動点の検出を行い、得られた結果を返す。

エラー処理 なし

- protected void r_calcOut()

機能概要 到達定義の解析の out を計算する。

機能説明 フィールド r_gen、 r_kill、 r_in を用いて r_out を計算する。

入力データ なし

出力データ なし (計算結果は r_out に格納される)

処理方式 $r_out = r_gen \cup (r_in - r_kill)$ を計算する。

エラー処理 なし

- protected void a_calcOut()

  機能概要 利用可能な式の解析の out を計算する。

  機能説明 フィールド a_gen、 a_kill、 a_in を用いて a_out を計算する。

  入力データ なし

  出力データ なし (計算結果は a_out に格納される)

  処理方式 $a_out = a_gen \cup (a_in - a_kill)$ を計算する。

  エラー処理 なし

- protected void l_calcIn()

  機能概要 生きている式の解析の in を計算する。

  機能説明 フィールド l_def、 l_use、 l_out を用いて l_in を計算する。

  入力データ なし

  出力データ なし (計算結果は l_in に格納される)

  処理方式 $l_in = l_def \cup (l_out - l_use)$ を計算する。

  エラー処理 なし

- protected void set(int[] a, int where)

  機能概要 ビットを立てる。

  機能説明 配列 a の where 番目のビットを立てる。

  入力データ a(配列)、 where(ビットを立てる場所)

  出力データ なし

  処理方式 $a[where >> 5]| = 1 << (where - ((where >> 5) << 5))$ を計算する。

  エラー処理 なし

- protected void reset(int[] a, int where)

  機能概要 ビットを降ろす。

  機能説明 配列 a の where 番目のビットを降ろす。

入力データ a(配列)、 where(ビットを降ろす場所)

出力データ なし

処理方式 $a[where >> 5]| = 1 << (where - ((where >> 5) << 5))$ を計算する。

エラー処理 なし

- protected int[] cap(int[] a, int[] b)

機能概要 論理積を求める。

機能説明 2 つのビット配列 (実際には int の配列) を得て、 2 つの配列の論理積を求め、結果のビット配列を返す。

入力データ a(配列)、 b(配列)

出力データ 配列

処理方式 2 つの配列より、それぞれの同じ添字の要素について論理積を求め、結果の配列の同じ添字の要素として格納し、その配列を返す。

エラー処理 なし

- protected int[] cup(int[] a, int[] b)

機能概要 論理和を求める。

機能説明 2 つのビット配列 (実際には int の配列) を得て、 2 つの配列の論理和を求め、結果のビット配列を返す。

入力データ a(配列)、 b(配列)

出力データ 配列

処理方式 2 つの配列より、それぞれの同じ添字の要素について論理和を求め、結果の配列の同じ添字の要素として格納し、その配列を返す。

エラー処理 なし

- protected int[] minus(int[] a, int[] b)

機能概要 ビット演算におけるマイナス求める。

機能説明 2 つのビット配列 (実際には int の配列) を得て、 2 つの配列の差を求め、結果のビット配列を返す。

入力データ a(配列)、 b(配列)

出力データ 配列

処理方式 2 つの配列より、それぞれの同じ添字の要素について差を求め、結果の配列の同じ添字の要素として格納し、その配列を返す。具体的には $a \cap \not b$ を計算する。

エラー処理 なし

- protected int[] cloneInfo(int[] a)

機能概要 配列データをコピーする。

機能説明 データフロー解析の情報が入った配列を新たな別の配列としてコピーする。

入力データ a(配列)

出力データ 配列

処理方式 データフロー解析の情報が入った配列を受け取り、そのすべての要素を新たな別の配列にコピーし、その配列を返す。

エラー処理 なし

- protected boolean compare(int[] a, int[] b)

機能概要 2 つの配列が等しいかどうかを調べる

機能説明 データフロー解析の情報が入った 2 つの配列を受け取り、その配列の要素が等しいかどうかを調べる。

入力データ a(配列)、 b(配列)

出力データ boolean 値

処理方式 データフロー解析の情報が入った 2 つの配列を受け取り、各添字の要素の値が等しいかどうかを調べ、 1 つでも等しくないものがあれば false を返し、すべて等しければ true を返す。

エラー処理 なし

- public void setFWhere(int where)

  機能概要 行番号をセットする。

  機能説明 トップレベルの Statement 内での行番号をセットする。

  入力データ where(int)

  出力データ なし

  処理方式 where を受け取り、フィールド fwhere にセットする。

  エラー処理 なし

- public int getFWhere()

  機能概要 行番号を返す。

  機能説明 トップレベルの Statement 内での行番号を得る。

  入力データ なし

  出力データ where(int)

  処理方式 フィールド fwhere の値を返す。

  エラー処理 なし

- public String toBitString(int[] a)

  機能概要 データフロー解析の情報を見易い文字列にする。

  機能説明 データフロー解析の情報受け取り、それを見易い文字列へ変換し、その文字列を返す。

  入力データ a(配列)

  出力データ 文字列 (String オブジェクト)

  処理方式 データフロー解析の情報が入った配列 a を受け取り、それを 2 進数表記した文字列へ変換し、その文字列を返す。

  エラー処理 なし

- void printIndent(PrintStream out, int indent)

機能概要 インデントを出力する．

機能説明 与えられたインデントの深さに従った空白を出力する．

入力データ out (出力する PrintStream), indent (インデントの深さ)

出力データ なし．

処理方式 indent × 4 個の空白文字を out に出力する．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

  機能概要 文を出力する．

  機能説明 文をインデントを付けて出力する．

  入力データ out (出力する PrintStream), indent (インデントの深さ)

  出力データ なし．

  処理方式 各ラベルごとに，“*label* : ” という文字列を生成して，out に出力する．

  エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 文を出力する．

  機能説明 文を出力する．

  入力データ out (出力する PrintStream)

  出力データ なし．

  処理方式 print(out, 0) メソッドを呼び出す．

  エラー処理 特になし．

4.5.98 OpenJIT.frontend.tree.StringExpression クラス

public class StringExpression extends ConstantExpression

(1) 概要

このクラスは，AST の文字列ノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- String value (文字列を保持する)

(3) コンストラクタ

- public StringExpression(int where, String value)

  機能概要 StringExpression オブジェクトを生成する．

  機能説明 スーパークラスの ConstantExpression クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), value (保持する文字列)

  出力データ なし．

  処理方式 super メソッドを引数 STRINGVAL(定数), where, Type.tString (定数) で呼び出し，this.value に value を代入する．

  エラー処理 特になし．

(4) メソッド

- public Object getValue()

  機能概要 文字列を取り出す．

  機能説明 value を保持する String オブジェクトを返す．

  入力データ なし．

出力データ 文字列オブジェクト

処理方式 value を返す．

エラー処理 特になし．

- public int hashCode()

  機能概要 value のハッシュ値を求める．

  機能説明 value のハッシュ値と値 3213 の Xor を求めて，返す．

  入力データ なし．

  出力データ ハッシュ値

  処理方式 value.hashCode メソッドを呼び出し，戻り値と値 3213 の Xor を求めて返す．

  エラー処理 特になし．

- public boolean equals(Object obj)

  機能概要 文字列が与えられた文字列と一致するか否かを検査する．

  機能説明 文字列 value が与えられた式オブジェクトの value と一致するか否かを検査する．

  入力データ obj (比較する式オブジェクト)

  出力データ 一致・不一致の真偽値

  処理方式 obj が StringExpression のインスタンスの場合，String クラスのメソッド value.equals(obj.value) を呼び出して，文字列の一致・不一致の検査して，その結果を返す．obj が StringExpression のインスタンスでなければ，False を返す．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

  機能概要 文字列オブジェクトをスタックトップに積むバイトコードを生成する．

機能説明 文字列オブジェクト value を wrap する this オブジェクトをスタックトップに積むバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 スタックトップに this を積むために，opc_ldc をオペコードとするバイトコードを生成する．

エラー処理 なし．

- public void print(PrintStream out)

機能概要 値を読み易く出力する．

機能説明 与えられた PrintStream に value を出力する．

入力データ out (出力先の PrintStream)

出力データ なし．

処理方式 value に対して out.print メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 文字列の中身を出力する．

機能説明 文字列 value にダブルクォーテーションを付けた文字列を出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 value の左右にダブルクォーテーションを join した文字列を返す．

エラー処理 特になし．

4.5.99 OpenJIT.frontend.tree.SubtractExpression クラス

public class SubtractExpression extends BinaryArithmeticExpression

(1) 概要

このクラスは，AST の – 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public SubtractExpression(int where, Expression left, Expression right)

  機能概要 AST の – 式ノードを生成する．

  機能説明 スーパークラスの BinaryArithmeticExpression クラスで定義されているコンストラクタを用いて，– 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (– 式の左辺式の Expression オブジェクト)，right (– 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 SUB(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数がともに int 型の場合．

  機能説明 int 型の引数の減算を行い，IntExpression オブジェクトを生成する．

  入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理内容 a–b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数の減算を行い，LongExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理内容 a–b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(float a, float b)

  機能概要 式を評価する．引数がともに float 型の場合．

  機能説明 float 型の引数の減算を行い，FloatExpression オブジェクトを生成する．

  入力データ a (float 型), b (float 型)

  出力データ Expression オブジェクト．

  処理内容 a–b を計算し，FloatExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression eval(double a, double b)

  機能概要 式を評価する．引数がともに double 型の場合．

機能説明 double 型の引数の減算を行い，DoubleExpression オブジェクトを生成する．

入力データ a (double 型), b (double 型)

出力データ Expression オブジェクト．

処理内容 a–b を計算し，DoubleExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 右辺式または左辺式が 0 の場合，式を単純化する．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式が 0 の場合，右辺式を符号を反転させた式 NegativeExpression オブジェクトを生成して返す．右辺式が 0 の場合，左辺値をこの – 式の値として返す．それ以外の場合，this オブジェクトを返す．

  エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 – を行うバイトコードを生成する．

  機能説明 各型ごとに適切な – 用オペコードを選択して，バイトコードを追加する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 opc_isub (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

  機能概要 –式の中身を出力する．

  機能説明 –式を中置記法で読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 infixForm メソッドを呼び出す．

  エラー処理 特になし．

4.5.100 OpenJIT.frontend.tree.SuperExpression クラス

public class SuperExpression extends Expression

(1) 概要

このクラスは，AST の super オブジェクトのノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- LocalField field

(3) コンストラクタ

- public SuperExpression(int where)

  機能概要 SuperExpression オブジェクトを生成する．

  機能説明 スーパークラスの Expression クラスで定義されたコンストラクタを用いて，生成・初期化を行う．

  入力データ where (行番号)

  出力データ なし．

  処理方式 super メソッドを引数 SUPER (定数), where, Type.tObject (定数) で呼び出す．

  エラー処理 特になし．

(4) メソッド

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の検査を行う．

  機能説明

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 ctx.field のクラス定義からスーパークラス定義を得て，superClass に格納する．ctx.field が static であるか，variable であるか，superClass が null であれば，undef.var エラーを出力し，type に Type.tError(定数) を格納して，vset を返す．

vset の LSB が立っていなければ，access.inst.before.super エラーを出力し，type に Type.tError(定数) を格納して，vset を返す．

superClass の型を得て，type に格納する．ctx.getLocalField メソッドを呼び出して，this のローカル変数フィールドを field に格納する．field.readcount を 1 加算する．最後に vset を返す．

エラー処理 必要に応じて，undef.var エラー，access.inst.before.super エラーを出力する．

- public void codeValue(Environment env, Context ctx, Assembler asm)

機能概要 super 文のバイトコードを生成する．

機能説明 super 文のバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 field.number 番目のローカル変数をスタックトップに積むために，opc_aload をオペコードとするバイトコードを生成する．

エラー処理 特になし．

- public String toPrettyString()

機能概要 super の中身を出力する．

機能説明 super の文字列を出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 “super” の文字列を返す．

エラー処理 特になし．

4.5.101 OpenJIT.frontend.tree.SwitchStatement クラス

public class SwitchStatement extends Statement

(1) 概要

このクラスは，AST の switch 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- public Expression expr

  switch 文の条件式

- public Statement args[]

  switch 文の本体の文の配列．

(3) コンストラクタ

- public SwitchStatement(int where, Expression expr, Statement args[])

  機能概要 SwitchStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，SwitchStatement オブジェクトを生成・初期化する．

  入力データ where (行番号), expr (条件式), args[] (本体の文の配列)

  出力データ なし．

  処理方式 super メソッドを引数 SWITCH (定数)，where で呼び出す．this.expr に expr，this.args に args を格納する．

  エラー処理 特になし．

(4) メソッド

なし．

4.5.102 OpenJIT.frontend.tree.SynchronizedStatement クラス

public class SynchronizedStatement extends Statement

(1) 概要

このクラスは，AST の synchronized 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- public Expression expr

  同期式．

- public Statement body

  同期文の本体．

(3) コンストラクタ

- public SynchronizedStatement(int where, Expression expr, Statement body)

  機能概要 SynchronizedStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，SynchronizedStatement オブジェクトの生成・初期化を行う．

  入力データ where (行番号), expr (同期式), body (同期文の本体)

  出力データ なし．

  処理方式 super メソッドを引数 SYNCHRONIZED (定数), where で呼び出す．this.expr に expr，this.body に body を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

機能概要 synchronized 文の検査を行う．

機能説明 synchronized 文の検査を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件式の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値

処理方式 同期式 expr を検査するために，expr.checkValue メソッドを呼び出し，戻り値を vset に格納する．

expr を java.lang.Object 型に変換するために，convert メソッドを呼び出し，戻り値を expr に格納する．

同期文本体 body を検査するために，body.check メソッドを呼び出し，戻り値を vset に格納する．

エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

機能概要 synchronized 式の inlining を行う．

機能説明 synchronized 式の inlining を行う．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 body が null でなければ，body.inline メソッドを呼び出して，戻り値を body に格納する．

body が null なら，expr を式とする ExpressionStatement オブジェクトを生成し，そのオブジェクトに対して，inline メソッドを呼び出し，戻り値を返す．

expr.inlineValue メソッドを呼び出し，戻り値を expr に格納し，this を返す．

エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

  機能概要 コピーされた synchronized 式の inlining を行う．

  機能説明 CompilerError を出力する．

  入力データ ctx (コンテクストを格納する Context オブジェクト), valNeeded (計算後の値をスタック上に残すかどうかを指定するフラグ)

  出力データ inlining 後の式．

  処理方式 CompilerError(“copyInline”) を出力する．

  エラー処理 無条件に，CompilerError(“copyInline”) を出力する．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 synchronized 文を実行するバイトコードを生成する．

  機能説明 synchronized 文を実行するバイトコードを生成する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

  出力データ なし．

  処理方式 同期式 expr を計算して，スタックトップに積むバイトコードを生成するために，expr.codeValue メソッドを呼び出す．

  新しい Context オブジェクト ctx を生成する．Object 型のローカル変数フィールド f1，int 型のローカル変数フィールド f2 を生成する．f1, f2 のローカル変数番号を格納する Integer オブジェクト num1, num2 を生成する．

  新しくラベル endLabel を生成する．新しく TryData オブジェクト td を生成し，td.add(null) を呼び出す．

  expr の式にロックをかけるために，opc_astore をオペコードとするバイトコード，opc_aload をオペコードとするバイトコード，opc_monitorenter をオペコードとするバイトコードを生成する．

  body を実行するために，新しい CodeContext オブジェクト bodyctx を生成し，opc_try をオペコードとするバイトコードを生成し，body.code メ

ソッドを呼び出す．最後に bodyctx.breakLabel と，td の endLabel を挿入する．

body の実行の後の始末をするために，opc_aload をオペコードとするバイトコード，opc_monitorexit をオペコードとするバイトコード，endLabel にジャンプする opc_goto をオペコードとするバイトコードを生成する．

catch 用のコードを生成するために，td の CatchData のラベルを挿入する．さらに，opc_aload をオペコードとするバイトコード，opc_minotrexit をオペコードとするバイトコード，opc_athrow をオペコードとするバイトコードを生成する．

finally 用のコードを生成するために，bodyctx.contLabel を挿入する．さらに，opc_astore をオペコードとするバイトコード，opc_aload をオペコードとするバイトコード，opc_minitorexit をオペコードとするバイトコード，opc_ret をオペコードとするバイトコードを生成する．

最後に，endLabel を挿入する．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

  機能概要 synchronized 文を出力する．

  機能説明 synchronized 文を読みやすく，インデントをつけて出力する．

  入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

  出力データ 特になし．

  処理方式 indent × 4 個の空白文字を出力する．“synchronized *expr body*” という形式で PrintStream out に出力するために，expr.print メソッド，body.print メソッドを適宜呼び出す．

  エラー処理 特になし．

4.5.103 OpenJIT.frontend.tree.ThisExpression クラス

public class ThisExpression extends Expression

(1) 概要

このクラスは，AST の this 式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- LocalField field

  this 式 (ローカル変数)

(3) コンストラクタ

- public ThisExpression(int where)

  機能概要 ThisExpression オブジェクトを生成する．

  機能説明 スーパークラスの Expression クラスで定義されているコンストラクタを用いて，ThisExpression オブジェクトの生成・初期化を行う．

  入力データ where (行番号)

  出力データ なし．

  処理方式 super メソッドを引数 THIS (定数), where, Type.tObject で呼び出す．

  エラー処理 特になし．

- public ThisExpression(int where, Context ctx)

  機能概要 ThisExpression オブジェクトを生成する．

  機能説明 スーパークラスの Expression クラスで定義されているコンストラクタを用いて，ThisExpression オブジェクトの生成・初期化を行う．

  入力データ where (行番号), ctx (コンテクスト)

出力データ なし．

処理方式 super メソッドを引数 THIS (定数), where, Type.tObject で呼び出す．ctx.field.isMethod メソッドを呼び出し，true ならば，field に ctx.getLocalField(idThis) メソッドの戻り値を格納する．また，field.readcount を 1 加算する．

エラー処理 特になし．

(4) メソッド

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 式の値を検査する．

  機能説明 式の値を検査する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値．

  処理方式 ctx.field.isStatic メソッドの戻り値が true なら，undef.var エラーを出力する．vset の LSB が立っていなければ，access.inst.before.super エラーを出力する．

  ctx.field.getClassDeclaration().getType メソッドの戻り値を type に格納する．ctx.field.isMethod メソッドの戻り値が true なら，ctx.getLocalField (idThis) の戻り値を field に格納して，field.readcount を 1 加算する．

  最後に vset を返す．

  エラー処理 undef.var エラー，および access.inst.before.super エラーを出力する．

- public Expression inline(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 null を返す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の式．

処理方式 null を返す．

エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 式の inlining を行う．

  機能説明 式の inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 field が null でなければ，field.getValue メソッドを呼び出し，戻り値が null でなければ戻り値を返す．それ以外の場合は，this を返す．

  エラー処理 特になし．

- public Expression copyInline(Context ctx)

  機能概要 コピーされた式の inlining を行う．

  機能説明 clone メソッドで式をコピーし，その式に対して inlining を行う．

  入力データ ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 clone メソッドを呼び出して，this をコピーして，ThisExpression e に格納する．field が null なら，ctx.getLocalField(idThis) メソッドの戻り値を e.field に格納し，e.field.readcount を 1 加算する．

  最後に e を返す．

  エラー処理 特になし．

- public void codeValue(Environment env, Context ctx, Assembler asm)

機能概要 this 式を実行するバイトコードを生成する．

機能説明 this 式を実行するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_aload をオペコードとするバイトコードを生成する．

エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 this 式を文字列で出力する．

  機能説明 this 式を文字列で PrintStream out に出力する．

  入力データ out (出力先の PrintStream)

  出力データ なし．

  処理方式 “this#*field.hashCode()*” の形式で，out に対して出力する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 this 式を文字列で出力する．

  機能説明 this 式を文字列に変換して返す．

  入力データ なし．

  出力データ 文字列．

  処理方式 “this” という文字列を返す．

  エラー処理 特になし．

4.5.104 OpenJIT.frontend.tree.ThrowStatement クラス

public class ThrowStatement extends Statement

(1) 概要

このクラスは，AST の throw 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- public Expression expr

  throw するオブジェクトの式．

(3) コンストラクタ

- public ThrowStatement(int where, Expression expr)

  機能概要 ThrowStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，ThrowStatement オブジェクトを生成・初期化する．

  入力データ where (行番号), expr (throw するオブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 THROW (定数), where で呼び出す．this.expr に expr を格納する．

  エラー処理 特になし．

(4) メソッド

なし．

4.5.105 OpenJIT.frontend.tree.TryStatement クラス

public class TryStatement extends Statement

(1) 概要

このクラスは，AST の Try 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- public Statement body

  try 文の本体の文．

- public Statement args[]

  try 文の引数の文の配列．

(3) コンストラクタ

- public TryStatement(int where, Statement body, Statement args[])

  機能概要 TryStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，TryStatement オブジェクトを生成・初期化する．

  入力データ where (行番号), body (try 文の本体), args[] (引数の文の配列)

  出力データ なし．

  処理方式 super メソッドを引数 TRY (定数), where で呼び出す．this.body に body, this.args に args を格納する．

  エラー処理 特になし．

(4) メソッド

なし．

4.5.106 OpenJIT.frontend.tree.TypeExpression クラス

public class TypeExpression extends Expression

(1) 概要

このクラスは，AST の型式のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public TypeExpression(int where, Type type)

  機能概要 TypeExpression オブジェクトを生成する．

  機能説明 スーパークラスの Expression クラスで定義されているコンストラクタを呼び出して，TypeExpression オブジェクトの生成・初期化を行う．

  入力データ where (行番号), type (型)

  出力データ なし．

  処理方式 super メソッドを引数 TYPE (定数), where, type で呼び出す．

  エラー処理 特になし．

(4) メソッド

- Type toType(Environment env, Context ctx)

  機能概要 特定の型に変換する．

  機能説明 type を返す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ 型．

処理方式 type を返す．

エラー処理 特になし．

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 型式の検査を行う．

  機能説明 型式は値を持たないので，invalid.term エラーを出力する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値．

  処理方式 invalid.term エラーを出力し，vset を返す．

  エラー処理 無条件に invalid.term エラーを出力する．

- public void print(PrintStream out)

  機能概要 型式の中身を出力する．

  機能説明 型式を読み易く出力する．

  入力データ out (String の出力先の PrintStream)

  出力データ なし．

  処理方式 type.toString メソッドの戻り値を out に出力する．

  エラー処理 特になし．

- public String toPrettyString()

  機能概要 型式の中身を出力する．

  機能説明 型式を読み易く出力する．

  入力データ なし．

  出力データ String 型の文字列．

  処理方式 type.toString メソッドの戻り値を返す．

  エラー処理 特になし．

4.5.107 OpenJIT.frontend.tree.UnaryExpression クラス

public class UnaryExpression extends Expression

(1) 概要

このクラスは，ArrayAccessExpression クラス, BinaryExpression クラス, BitNotExpression クラス, ConvertExpression クラス, ExprExpression クラス, FieldExpression クラス, IncDecExpression クラス, LengthExpression クラス, NaryExpression クラス, NegativeExpression クラス, NotExpression クラス, PositiveExpression クラスのスーパークラスであり，これらのクラスに共通するコード生成の機能を実現する．

(2) フィールド (変数)

- public Expression right

  右辺式．

(3) コンストラクタ

- UnaryExpression(int op, int where, Type type, Expression right)

  機能概要 UnaryExpression オブジェクトを生成する．

  機能説明 スーパークラスの Expression クラスで定義されているコンストラクタを用いて，UnaryExpression オブジェクトの生成・初期化を行う．

  入力データ op (オペレータ), where (行番号), type (型), right (右辺式)

  出力データ なし．

  処理方式 super メソッドを引数 op, where, type で呼び出す．this.right に right を格納する．

  エラー処理 特になし．

(4) メソッド

- public Expression right()

機能概要 右辺式のアクセッサメソッド．

機能説明 右辺式 right をアクセスする．

入力データ なし．

出力データ 式オブジェクト．

処理方式 right を返す．

エラー処理 特になし．

- public Expression order()

  機能概要 オペレータの優先順位に従って，式を並び換える．

  機能説明 この式の優先順位が左辺の式より高い場合，左辺と右辺を入れ換える．

  入力データ なし．

  出力データ 並び換え後の Expression．

  処理方式 スーパークラスの precedence メソッドを呼び，その値が right.precedence() 以上なら，UnaryExpression e に式 right を保持し，right には e.right を保持する．e.right には再帰的に order メソッドを呼び出した結果を保持する．これを繰り返すことにより，最終的に優先順位順に並び換えた式が this に得られるので，this を返す．

  エラー処理 特になし．

- void selectType(Environment env, Context ctx, int tm)

  機能概要 式の型を選択する．

  機能説明 単項演算式の型は指定できないので，エラーを出力する．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), tm (右辺式と左辺式の型情報)

  出力データ なし．

  処理方式 CompilerError(“selectType: *opNames[op]*”) を throw する．

エラー処理 無条件に，CompilerError(“selectType: *opNames[op]*”) を throw する．

- public long checkValue(Environment env, Context ctx, long vset, Hashtable exp)

  機能概要 単項演算式の検査を行う．

  機能説明 単項演算式の検査を行う．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

  出力データ 条件の評価値．

  処理方式 右辺式の検査を行うために，right.checkValue メソッドを呼び出し，戻り値を vset に格納する．右辺式の型にこの単項演算式の型を合わせるために，selectType メソッドを呼び出す．最後に，vset を返す．

  エラー処理 特になし．

- Expression eval(int a)

  機能概要 単項演算式の評価を行う．引数が int 型の場合．

  機能説明 何もせずに，this を返す．

  入力データ a (int 型)

  出力データ Expression オブジェクト．

  処理方式 this を返す．

  エラー処理 特になし．

- Expression eval(long a)

  機能概要 単項演算式の評価を行う．引数が long 型の場合．

  機能説明 何もせずに，this を返す．

  入力データ a (long 型)

  出力データ Expression オブジェクト．

処理方式 this を返す．

エラー処理 特になし．

- Expression eval(float a)

  機能概要 単項演算式の評価を行う．引数が float 型の場合．

  機能説明 何もせずに，this を返す．

  入力データ a (float 型)

  出力データ Expression オブジェクト．

  処理方式 this を返す．

  エラー処理 特になし．

- Expression eval(double a)

  機能概要 単項演算式の評価を行う．引数が double 型の場合．

  機能説明 何もせずに，this を返す．

  入力データ a (double 型)

  出力データ Expression オブジェクト．

  処理方式 this を返す．

  エラー処理 特になし．

- Expression eval(boolean a)

  機能概要 単項演算式の評価を行う．引数が boolean 型の場合．

  機能説明 何もせずに，this を返す．

  入力データ a (boolean 型)

  出力データ Expression オブジェクト．

  処理方式 this を返す．

  エラー処理 特になし．

- Expression eval(String a)

機能概要 単項演算式の評価を行う．引数がString型の場合．

機能説明 何もせずに，thisを返す．

入力データ a (String型)

出力データ Expressionオブジェクト．

処理方式 thisを返す．

エラー処理 特になし．

- Expression eval()

機能概要 単項演算式の評価を行う．

機能説明 単項演算式の評価を行う．

入力データ なし．

出力データ 評価後の式．

処理方式 right.opがBYTEVAL(定数), CHARVAL(定数), SHORTVAL(定数), INTVAL(定数)の場合，right.valueをIntegerExpressionとして，evalメソッドを呼び出し，戻り値を返す．

right.opがLONGVAL(定数)の場合，right.valueをLongExpressionとして，evalメソッドを呼び出し，戻り値を返す．

right.opがFLOATVAL(定数)の場合，right.valueをFloatExpressionとして，evalメソッドを呼び出し，戻り値を返す．

right.opがDOUBLEVAL(定数)の場合，right.valueをDoubleExpressionとして，evalメソッドを呼び出し，戻り値を返す．

right.opがBOOLEANVAL(定数)の場合，right.valueをBooleanExpressionとして，evalメソッドを呼び出し，戻り値を返す．

right.opがSTRINGVAL(定数)の場合，right.valueをStringExpressionとして，evalメソッドを呼び出し，戻り値を返す．

エラー処理 特になし．

- public Expression inline(Environment env, Context ctx)

機能概要 単項演算式の右辺式の inlining を行う．

機能説明 単項演算式の右辺式の inlining を行い，inlining 後の右辺式を返す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ inlining 後の右辺式．

処理方式 right.inline メソッドを呼び出し，戻り値を返す．

エラー処理 特になし．

- public Expression inlineValue(Environment env, Context ctx)

  機能概要 単項演算式の inlining を行う．

  機能説明 単項演算式の inlining を行い，inlining 後の式を返す．

  入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

  出力データ inlining 後の式．

  処理方式 right.inlineValue メソッドを呼び出し，戻り値を right に格納する．eval メソッドを呼び出し，戻り値に対して simplify メソッドを呼び出し，戻り値を返す．simplify メソッド中で数値演算のエラーが生じた場合には，arithmetic.exception エラーを出力する．

  エラー処理 inlining の結果，数値演算エラーが生じたら，arithmeritc.exceptin エラーを出力する．

- public Expression copyInline(Context ctx)

  機能概要 コピーされた単項演算式の inlining を行う．

  機能説明 単項演算式のコピーを作り，そのコピーに関して inlining を行う．

  入力データ env (環境を格納する Environment オブジェクト)

  出力データ inlining 後の式．

  処理方式 clone メソッドを呼び出し，単項演算式のコピーを作り，UnaryExpression e に格納する．right が null なら，e をそのまま返す．

right が null でない場合，right.copyInline メソッドを呼び出し，戻り値を e.right に格納して，e を返す．

エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 inlining のコストを計算する．

  機能説明 右辺式の inlining のコストに 1 を加えてコストを計算する．

  入力データ thresh (閾値)

  出力データ inlining のコスト．

  処理方式 1 + right.costInline(thresh) を計算して，返す．

  エラー処理 特になし．

- public void print(PrintStream out)

  機能概要 単項演算式の中身を出力する．

  機能説明 単項演算式を読み易い形式で PrintStream に出力する．

  入力データ out (String の出力先の PrintStream)

  出力データ なし．

  処理方式 単項演算式を “(*opNames[op] right*)” という形式で，PrintStream out に出力する．

  エラー処理 特になし．

- String prefixForm(String operator)

  機能概要 前置記法で単項演算式を文字列に変換する．

  機能説明 前置記法で単項演算式を文字列に変換する．

  入力データ operator (オペレータの文字列)

  出力データ 文字列．

  処理方式 “(*operator right*)” という形式の文字列を生成し，返す．

エラー処理 特になし．

- String postfixForm(String operator)

  機能概要 後置記法で単項演算式を文字列に変換する．

  機能説明 後置記法で単項演算式を文字列に変換する．

  入力データ operator (オペレータの文字列)

  出力データ 文字列．

  処理方式 “(*right operator*)” という形式の文字列を生成し，返す．

  エラー処理 特になし．

4.5.108 OpenJIT.frontend.tree.UnsignedShiftRightExpression クラス

public class UnsignedShiftRightExpression extends BinaryShiftExpression

(1) 概要

このクラスは，AST の >>> 演算子のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

なし．

(3) コンストラクタ

- public UnsignedShiftRightExpression(int where, Expression left, Expression right)

  機能概要 AST の >>> 式ノードを生成する．

  機能説明 スーパークラスの BinaryShiftExpression クラスで定義されているコンストラクタを用いて，>>> 式ノードの生成・初期化を行う．

  入力データ where (行番号)，left (>>> 式の左辺式の Expression オブジェクト)，right (>>> 式の右辺式の Expression オブジェクト)

  出力データ なし．

  処理方式 super メソッドを引数 URSHIFT(定数), where, left, right で呼び出す．

  エラー処理 なし．

(4) メソッド

- Expression eval(int a, int b)

  機能概要 式を評価する．引数が共に int 型の場合．

機能説明 int 型の引数の >>> を行い，IntExpression オブジェクトを生成する．

入力データ a (int 型), b (int 型)

出力データ Expression オブジェクト．

処理方式 a>>>b を計算し，IntExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

エラー処理 特になし．

- Expression eval(long a, long b)

  機能概要 式を評価する．引数がともに long 型の場合．

  機能説明 long 型の引数の >>> を行い，LongExpression オブジェクトを生成する．

  入力データ a (long 型), b (long 型)

  出力データ Expression オブジェクト．

  処理方式 a>>>b を計算し，LongExpression クラスのコンストラクタを呼び出し，生成したオブジェクトを返す．

  エラー処理 特になし．

- Expression simpify()

  機能概要 式の単純化を行う．

  機能説明 左辺式または右辺式が 0 の場合，式を単純化する．

  入力データ なし．

  出力データ Expression オブジェクト．

  処理方式 左辺式または右辺式が 0 の場合，left を返す．それ以外の場合は this を返す．

  エラー処理 特になし．

- void codeOperation(Environment env, Context ctx, Assembler asm)

  機能概要 >>> を行うバイトコードを生成する．

機能説明 各型ごとに適切な >> 用オペコードを選択して，バイトコードを追加する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 opc_iushr (定数) に type.getTypeCodeOffset() の戻り値を加えた値をオペコードとして，asm オブジェクトの add メソッドを呼び出す．

エラー処理 特になし．

- public String toPrettyString()

機能概要 >>> 式の中身を出力する．

機能説明 >>> 式を中置記法で読み易く出力する．

入力データ なし．

出力データ String 型の文字列．

処理方式 infixForm メソッドを呼び出す．

エラー処理 特になし．

4.5.109 OpenJIT.frontend.tree.VarDeclarationStatement クラス

public class VarDeclarationStatement extends Statement

(1) 概要

このクラスは，AST の変数宣言文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- public LocalField field

  宣言している変数のローカルフィールド．

- public Expression expr

  初期化式．

(3) コンストラクタ

- public VarDeclarationStatement(int where, Expression expr)

  機能概要 VarDeclarationStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，VarDeclarationStatement オブジェクトを生成・初期化する．

  入力データ where (行番号), expr (初期化式)

  出力データ なし．

  処理方式 super メソッドを引数 VARDECLARATION (定数), where で呼び出す．this.expr に expr を格納する．

  エラー処理 特になし．

- public VarDeclarationStatement(int where, LocalField field, Expression expr)

機能概要 VarDeclarationStatement オブジェクトを生成する．

機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，VarDeclarationStatement オブジェクトを生成・初期化する．

入力データ where (行番号), field (変数のローカルフィールド), expr (初期化式)

出力データ なし．

処理方式 super メソッドを引数 VARDECLARATION (定数), where で呼び出す．this.field に field, this.expr に expr を格納する．

エラー処理 特になし．

(4) メソッド

なし．

4.5.110 OpenJIT.frontend.tree.WhileStatement クラス

public class WhileStatement extends Statement

(1) 概要

このクラスは，AST の While 文のノードを表現するとともに，コード生成の機能を実現する．

(2) フィールド (変数)

- Expression cond
  条件式．
- Statement body
  While ブロックの文．

(3) コンストラクタ

- public WhileStatement(int where, Expression cond, Statement body)

  機能概要 WhileStatement オブジェクトを生成する．

  機能説明 スーパークラスの Statement クラスで定義されているコンストラクタを用いて，ノードの生成・初期化を行う．

  入力データ where (行番号), cond (条件式)， body (While ブロックの式)

  出力データ なし．

  処理方式 super メソッドを引数 WHILE (定数), where で呼び出し，this.cond に cond，this.body に body を格納する．

  エラー処理 特になし．

(4) メソッド

- long check(Environment env, Context ctx, long vset, Hashtable exp)

機能概要 While 文の検査を行う．

機能説明 While 文の条件式，ボディをそれぞれ検査し，ループの脱出条件を設定する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), vset (条件の評価値), exp (ハッシュテーブル)

出力データ 条件の評価値．

処理方式 新しく CheckContext オブジェクトを生成し，これを newctx に格納する．まず，cond を検査する．cond 式に現れるどの変数が true ないし false を返す値を保持しているかを決定するために，cond.checkCondition メソッドを呼び出し，その結果の条件の評価値を cvars に格納する．また，条件式 cond の型を convert メソッドを呼び出すことで，boolean 型に変換する．

次に body を検査するために，cond が true になったと仮定して，body.check メソッドを呼び出す．

while ループは cond の検査の結果が false になるか，break 文に到達すると終了するため，newctx.vxBreak & cvars.vsFalse の計算結果を引数として，ctx.removeAdditionalVars メソッドを呼び出す．最後に，その戻り値の条件の評価値を返す．

エラー処理 特になし．

- public Statement inline(Environment env, Context ctx)

機能概要 While 文の Inlining を行う．

機能説明 条件式，ボディをそれぞれ Inlining して，その結果の文を返す．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト)

出力データ Inlining 後の文オブジェクト．

処理方式 cond の inlining を行うために cond.inlineValue メソッドを呼び出して，その結果で cond を更新する．body の inlining を行うために，まず，body が null か否か検査する．body が null でなければ，body.inline メソッドを呼び出し，その結果で body を更新する．最後に this を返す．

エラー処理 特になし．

- public int costInline(int thresh)

  機能概要 Inlining のコストを計算する．

  機能説明 While 文の Inlining のコストとして，条件式とボディのそれぞれの inlining コストの和を計算して返す．

  入力データ thresh (閾値)．

  出力データ Inlining のコスト．

  処理方式 cond.costInline(thresh) + body.costInline(thresh) + 1 を計算して，返す．

  エラー処理 特になし．

- public Statement copyInline(Context ctx, boolean valNeeded)

  機能概要 While 文の inlining されたコピーを作る．

  機能説明 method inlining のために while 文のコピーを作って，返す．

  入力データ ctx (コンテクストを格納する Context オブジェクト), valNeeded (計算後の値をスタック上に残すか否かを決定する真偽値)

  出力データ While 文オブジェクト．

  処理方式 clone メソッドを呼び出し，While 文のコピーを作り，s に格納する．s について inlining を行うために，s.cond の inlining を行うために cond.inlineValue メソッドを呼び出して，その結果で cond を更新する．s.body の inlining を行うために，まず，body が null か否か検査する．body が null でなければ，body.inline メソッドを呼び出し，その結果で body を更新する．最後に s を返す．

  エラー処理 特になし．

- public void code(Environment env, Context ctx, Assembler asm)

  機能概要 While 文を実行するバイトコードを生成する．

機能説明 While 文を実行するバイトコードを生成する．

入力データ env (環境を格納する Environment オブジェクト), ctx (コンテクストを格納する Context オブジェクト), asm (コード生成を行う Assembler オブジェクト)

出力データ なし．

処理方式 まず，現在の CodeContext を生成して newctx に格納する．newcontLabel にジャンプするバイトコードを生成するために，opc_goto をオペコードとするバイトコードを生成する．次に新しいラベル l1 を生成して，l1 をジャンプコードの直後に配置する．

body が null でなければ，body.code メソッドを呼び出して，body のコードを実行するバイトコードを生成する．

newctx.contLabel のラベルを挿入し，cond の条件式が成立すれば l1 にジャンプするバイトコードを生成するために，cond.codeBranch メソッドを呼び出す．最後に，newctx.breakLabel を挿入する．

エラー処理 特になし．

- public void print(PrintStream out, int indent)

機能概要 While 文の中身を出力する．

機能説明 While 文の中身を読み易く文字列で，PrintStream out に出力する．

入力データ out (String の出力先の PrintStream)，indent (インデントの深さ)

出力データ なし．

処理方式 “while *cond body*;” という形式で出力できるように，まず，“while ” を out に出力し，cond.print メソッド，body.print メソッドを呼び出す．

エラー処理 特になし．

4.6 OpenJIT.frontend.util パッケージ

- インターフェース

  なし．

- 抽象クラス

  - OpenJIT.frontend.util.LookupHashtable 抽象クラス

  なし．

- クラス

  - OpenJIT.frontend.util.HashtableEntry クラス
  - OpenJIT.frontend.util.HashtableEnumerator クラス
  - OpenJIT.frontend.util.IntKeyHashtable クラス
  - OpenJIT.frontend.util.Queue クラス
  - OpenJIT.frontend.util.QueueEntry クラス

- 例外

  なし．

4.6.1 OpenJIT.frontend.util.HashtableEntry クラス

class HashtableEntry

(1) 概要

IntKeyHashtable を実現するためのクラス．具体的には IntKeyHashteble の 1 要素であるキーとオブジェクトの組を格納するリストである．

(2) フィールド (変数)

- int key

  キー

- Object value
  キーに対応づけられたオブジェクト

- HashtableEntry next
  次の HashtableEntry

(3) コンストラクタ

なし

(4) メソッド

- protected Object clone()

  機能概要 要素の複製を作る．

  機能説明 IntKeyHashtable の要素である HashtableEntry の自分自身の複製を作る．

  入力データ なし

  出力データ なし

  処理方式 新しく HashtableEntry オブジェクトを生成する．そして，その新しい HashtableEntry オブジェクトに自分自身の情報 (key， value， next) をセットする． next が null でなければ， next に対してもこのメソッド (clone()) が呼び出され，その結果が格納される．

  エラー処理 なし

4.6.2 OpenJIT.frontend.util.HashtableEnumerator クラス

class HashtableEnumerator implements Enumeration

(1) 概要

IntKeyHashtable を実現するためのクラス．具体的には IntKeyHashtable に格納されている HashtableEntry オブジェクトを Enumeration として表現する．

(2) フィールド (変数)

- boolean keys
  Enumeration オブジェクトとして表現する時に，キーを用いるか，オブジェクトを用いるかを指定する．
- int index
  格納されている HashtableEntry の数
- HashtableEntry[] table
  Enumeration オブジェクトとして表現される HashtableEntry の配列
- HashtableEntry entry
  次に返すべき HashtableEntry オブジェクト

(3) コンストラクタ

- HashtableEnumerator(HashtableEntry[], boolean)

  機能概要 IntKeyHashtable のための Enumeration を実現する．

  機能説明 HashtebleEntry オブジェクトの配列 table[] と Enumaration におけるオペレーションでキーとオブジェクトどちらを返すかを指定する keys を受け取り，初期化する．

  入力データ table[](HashtableEntry)， keys(boolean)

  出力データ なし

処理方式 入力 table[]，keys をフィールドに代入し，table[] のサイズをフィールド index に格納する．

エラー処理 なし

(4) メソッド

- public boolean hasMoreElements()

  機能概要 インターフェース Enumeration のメソッド hasMoreElements() を実装する．

  機能説明 インターフェース Enumeration のメソッド hasMoreElements() の実装．まだ要素があれば Boolean 値 True を返し，なければ Boolean 値 False を返す．

  入力データ なし

  出力データ boolean 値

  処理方式 配列 table[] の index 番目の要素を entry へ代入する．つぎに entry が null であるかどうかを調べ，null でなければ true を返す．null であったら index の値を 1 減らし，同様の操作を繰り返す．もし index が 0 であれば要素がないので false を返す．

  エラー処理 なし

- public Object nextElement()

  機能概要 インターフェース Enumeration のメソッド nextElement() の実装．

  機能説明 格納されている次の要素を返す．

  入力データ なし

  出力データ Object

  処理方式 entry が null であるかどうかをチェックし，null であれば index の値を減らして次の null でない要素を見つけて entry へ代入する．この操作の後でも entry が null であったら，NoSuchElementException を発行する．entry が null でない場合，entry を新しく生成した HashtableEntry オブジェクト

へ代入し，entry の値をフィールド next で指される HashtableEntry オブジェクトへ更新する．次に，keys をチェックし，true であればフィールド key を Integer オブジェクトとして返す．false であれば，フィールド value にあるオブジェクトを返す．

エラー処理 格納されている要素がない場合，NoSuchElementException を発行する．

4.6.3 OpenJIT.frontend.util.IntKeyHashtable クラス

public abstract class IntKeyHashtable implements Cloneable, Serializable

(1) 概要

java.util.Hashtable クラスでキーとして int を用いるように改良したクラス．

(2) フィールド (変数)

- HashtableEntry[] table
  要素である HashtableEntry の配列を格納する．
- int count
  ハッシュテーブルの数
- int threshold
  テーブルを再構築するかどうかの閾値
- float loadFactor
  ハッシュテーブルのロードファクター
- long serialVersionUID
  シリアライズしたときの ID

(3) コンストラクタ

- public IntKeyHashtable(int initialCapacity, float loadFactor)

  機能概要 IntKeyHashtable オブジェクトを初期化する．

  機能説明 テーブルのキャパシティとロードファクターをもらって IntKeyHashtable オブジェクトを初期化する．

  入力データ initialCapacity(int)，loadFactor(float)

  出力データ なし

処理方式 まず，引数が正当な値であるかをチェックする．引数 initialCapcity が 0 未満であるか，引数 loadFactor が 0.0f 未満であれば IllegalArgumentException() を発行する．次に，引数 loadFactor を同名のフィールドへ代入する．さらに HashtableEntry オブジェクトの配列 table を大きさを引数 initialCapacity として生成する．最後にフィールド threshold を initialCapacity と loadFactor の積に設定する．

エラー処理 ．引数 initialCapcity が 0 未満であるか，引数 loadFactor が 0.0f 未満であれば IllegalArgumentException() を発行する．

- public IntKeyHashtable(int initialCapacity)

機能概要 IntKeyHashtable オブジェクトを初期化する．

機能説明 テーブルのキャパシティをもらって IntKeyHashtable オブジェクトを初期化する．

入力データ initialCapacity(int)

出力データ なし

処理方式 引数 initialCapacity とデフォルトのロードファクター 0.75f を引数として 2 引数のコンストラクタを呼び出す．

エラー処理 なし

- public IntKeyHashtable()

機能概要 IntKeyHashtable オブジェクトを初期化する．

機能説明 デフォルトのキャパシティとロードファクターで IntKeyHashtable オブジェクトを初期化する．

入力データ なし

出力データ なし

処理方式 デフォルトのキャパシティ 101 とロードファクター 0.75f を引数として 2 引数のコンストラクタを呼び出す．

エラー処理 なし

(4) メソッド

- public int size()

  機能概要 Hashtable のキーの数を返す．

  機能説明 Hashtable に格納されているキーの数を返す．

  入力データ なし

  出力データ int

  処理方式 フィールド count の値を返す．

  エラー処理 なし

- public boolean isEmpty()

  機能概要 Hashtable が空であるかを調べる．

  機能説明 Hashtable が空であれば true を，そうでなければ false を返す．

  入力データ なし

  出力データ boolean

  処理方式 count==0 の結果を返す．

  エラー処理 なし

- public Enumeration keys()

  機能概要 キーを列挙したオブジェクトを返す．

  機能説明 Hashtable に含まれているキーを Enumeration オブジェクトとして返す．

  入力データ なし

  出力データ Enumeration オブジェクト

  処理方式 フィールド table と boolean 値 true を引数として HashtableEnumerator オブジェクトを生成し，そのオブジェクトを返す．

  エラー処理 なし

- public Enumeration elements()

  機能概要 オブジェクトを列挙したオブジェクトを返す．

  機能説明 Hashtable に含まれているオブジェクトを Enumeration オブジェクトとして返す．

  入力データ なし

  出力データ Enumeration オブジェクト

  処理方式 フィールド table と boolean 値 false を引数として HashtableEnumerator オブジェクトを生成し，そのオブジェクトを返す．

  エラー処理 なし

- public boolean contains(Object value)

  機能概要 オブジェクトが含まれているかをチェックする．

  機能説明 引数である value が Hashtable 内で何らかのキーとマップされていれば true を，そうでなければ false を返す．

  入力データ value(Object オブジェクト)

  出力データ boolean

  処理方式 フィールド table 内の HashtableEntry の value について順に，引数 value と等しいかどうかをメソッド equals() を用いて判定し，等しければ true を，すべて調べても等しいものが見つからなければ false を返す．

  エラー処理 引数 value が null であれば，NullPointerException を発行する．

- public boolean containsKey(int key)

  機能概要 キーが含まれているかをチェックする．

  機能説明 引数である key が Hashtable 内でオブジェクトとマップされていれば true を，そうでなければ false を返す．

  入力データ key(int)

  出力データ boolean

処理方式 フィールド table 内の HashtableEntry の key について順に，引数 value と等しいかどうかを調べ等しければ true を，すべて調べても等しいものが見つからなければ false を返す．

エラー処理 なし

- public Object get(int key)

機能概要 オブジェクトを得る．

機能説明 Hashtable 内で引数 key でマップされているオブジェクトを返す．

入力データ key(int)

出力データ Object オブジェクト

処理方式 フィールド table 内の HashtableEntry の key について順に，引数 value と等しいかどうかを調べ等しければそのオブジェクト value を，すべて調べても等しいものが見つからなければ null を返す．

エラー処理 なし

- protected void rehash()

機能概要 テーブルを再構築する．

機能説明 ハッシュテーブルを再構築する．

入力データ なし

出力データ なし

処理方式 ハッシュテーブルの内容を，より大きな容量をもつハッシュテーブルにハッシュし直す．ハッシュテーブルにあるキーの数がハッシュテーブルの容量と負荷係数を超えると，このメソッドが自動的に呼び出される．

エラー処理 なし

- public Object put(int key, Object value)

機能概要 指定された key を，指定された value にマップする．

機能説明 Hastable に指定された key を，指定された value にマップする．

入力データ key(int)，value(Object)

出力データ Object オブジェクト

処理方式 引数 value が null かどうかを調べ，null であれば NullPointerException を発行する．Hashtable 内に既に引数 key にマップされたものが在れば，引数 value に置き換え，古いオブジェクトを返す．フィールド count がフィールド threshold を超えていれば，メソッド rehash() を呼び出し，同じ引数 put を再び呼び出す．対応するキーがなければ，新たに HashtableEntry オブジェクトを生成し，そのフィールドに引数 key，value を格納し，生成した HashtableEntry オブジェクトをフィールド table に格納する．

エラー処理 引数 value が null であれば，NullPointerException を発行する．

- public Object remove(int key)

機能概要 指定された key でマップされるオブジェクトを削除する．

機能説明 Hashtable 内で指定された key でマップされるオブジェクトがあれば削除する．

入力データ key(int)

出力データ Object オブジェクト

処理方式 キーでマップされるオブジェクトを探し，あればそのキーとオブジェクトのマッピングを消し，フィールド count を減らし，マッピングの消されたオブジェクトを返す．キーに一致するものがなければ null を返す．

エラー処理 なし

- public void clear()

機能概要 空にする．

機能説明 Hashtable を空にする．

入力データ なし

出力データ なし

処理方式 フィールド table を初期化し，フィールド count を 0 にする．

エラー処理 なし

- public Object clone()

  機能概要 オブジェクトの複製を作る．

  機能説明 IntKeyHashtable 自身の複製を生成して返す．

  入力データ なし

  出力データ Object オブジェクト

  処理方式 IntKeyHashtable オブジェクトを新たに生成し，フィールド table に格納されているすべての HashtableEntry オブジェクトの複製を新たに生成した IntKeyHashtable オブジェクトのフィールド table へ格納する．生成された IntKeyHashtable オブジェクトを返す．

  エラー処理 なし

- public String toString()

  機能概要 String オブジェクトへ変換する．

  機能説明 Hashtable 内の情報を見易い形の String オブジェクト変換する．

  入力データ なし

  出力データ String オブジェクト

  処理方式 Hashtable 内に格納されているすべてのキーとオブジェクトについてそれぞれのマッピングを示す見易い文字列を生成し，String オブジェクトとして返す．

  エラー処理 なし

4.6.4 OpenJIT.frontend.util.LookupHashtable 抽象クラス

public abstract class LookupHashtable extends Hashtable

(1) 概要

java.util.Hashtable を拡張し，lookup メソッドを追加したアブストラクトクラス．

(2) フィールド (変数)

なし

(3) コンストラクタ

- public LookupHashtable(int initialCapacity, float loadFactor)

  機能概要 コンストラクタ

  機能説明 LookupHashtable オブジェクトを生成する．

  入力データ initialCapacity(int)，loadFactor(float)

  出力データ なし

  処理方式 スーパークラス Hashtable のコンストラクタを呼び出す．

  エラー処理 なし

- public LookupHashtable(int initialCapacity)

  機能概要 コンストラクタ

  機能説明 LookupHashtable オブジェクトを生成する．

  入力データ initialCapacity(int)

  出力データ なし

  処理方式 スーパークラス Hashtable のコンストラクタを呼び出す．

  エラー処理 なし

- public LookupHashtable()

機能概要 コンストラクタ

機能説明 LookupHashtable オブジェクトを生成する．

入力データ なし

出力データ なし

処理方式 なし (暗黙的にスーパークラス Hashtable のコンストラクタを呼び出す)

エラー処理 なし

(4) メソッド

- public Object lookup(Object key)

機能概要 key にマップされているオブジェクトを返す．

機能説明 key にマップされているオブジェクトを返すが，取り出されたオブジェクトは再び格納される．

入力データ key(Object オブジェクト)

出力データ Object オブジェクト

処理方式 Hashtable より get() メソッドを呼び出して引数 key に対応するオブジェクトを取り出し，put() メソッドで再び Hashtable に格納し，取り出したオブジェクトを返す．

エラー処理 なし

4.6.5 OpenJIT.frontend.util.Queue クラス

public class Queue

(1) 概要

オブジェクト単位のキューバッファを実現する．

(2) フィールド (変数)

- QueueEntry head
  キューの先頭
- QueueEntry tail
  キューの最後
- int count
  キューに入っている QueueEntry の数

(3) コンストラクタ

- public Queue()

  機能概要 Queue クラスを初期化する．

  機能説明 Queue クラスを初期化する．

  入力データ なし

  出力データ なし

  処理方式 フィールド count を 0 にし，フィールド head， tail を null にする．

  エラー処理 なし

(4) メソッド

- public int count()

  機能概要 現在のキューの長さを返す．

機能説明 現在キューに格納されているオブジェクトの数を返す．

入力データ なし

出力データ int

処理方式 フィールド count の値を返す．

エラー処理 なし

- public void enqueue(Object value)

  機能概要 キューに追加する．

  機能説明 引数 value で示されるオブジェクトをキューに追加する．

  入力データ value(Object オブジェクト)

  出力データ なし

  処理方式 引数 value を引数として QueueEntry オブジェクトを生成し，キューリストの最後に追加する．

  エラー処理 なし

- public Object dequeue()

  機能概要 キューから取り出す．

  機能説明 キューの中の先頭のオブジェクトを取り出す．

  入力データ なし

  出力データ Object オブジェクト

  処理方式 キューが null でなければフィールド head で指される QueueEntry のフィールド value で指されるオブジェクトを返す．キューが null であれば，null を返す．

  エラー処理 なし

- public boolean hasMoreElements()

  機能概要 キューが空かどうかを調べる．

機能説明 キューが空でなければ(キューに要素があれば)true を返し，空であれば false を返す．

入力データ なし

出力データ boolean

処理方式 $\mathrm{count} > 0$ の結果を返す．

エラー処理 なし

4.6.6 OpenJIT.frontend.util.QueueEntry クラス

class QueueEntry

(1) 概要

Queue クラスで要素として用いられるリスト構造のクラス

(2) フィールド (変数)

- Object value
  キューに格納されているオブジェクト

- QueueEntry next
  次の QueueEntry

(3) コンストラクタ

- QueueEntry(Object value)

  機能概要 QueueEntry オブジェクトを生成する．

  機能説明 オブジェクト value を値としてもつ QueueEntry オブジェクトを生成する．

  入力データ value(Object オブジェクト)

  出力データ なし

  処理方式 引数 value をフィールド value へ格納し，フィールド next を null で初期化する．

  エラー処理 なし

(4) メソッド

なし

4.7 OpenJIT Java Native Code API パッケージ

4.7.1 OpenJIT Java Native Code API パッケージ

(1) 概要

本パッケージは JDK システムと OpenJIT システムのインタフェースを提供する．

Sun の JDK においては，Java Native Code API(Application Programmer's Interface) というコンパイラに対するインタフェースが用意されている。この API は JVM のインタプリタにネイティブコード生成を組み込むために用意されたものである。このパッケージはそのインタフェースを定義するものである．

このパッケージは C 言語で記述している．

(2) 変数

- sys_mon_t *_compile_lock

  コンパイルの同期を行うためのロック変数．

  OpenJIT コンパイラは並列動作が可能である．すなわち，同時に複数のメソッドをコンパイルすることができる．しかし，同じメソッドを重複してコンパイルすることを避けるために同期しなければならない．これを実現するために用いられる．

- ClassClass *OpenJITClass

  JIT コンパイルのためのベースのクラスとなる JDK のクラスの内部構造へのポインタ．通常は OpenJIT. Sparc クラスへのポインタとなる．

  java_lang_Compiler_start() において初期化され，その後値が変更されることはない．

- ClassClass *OpenJITExceptionClass

  OpenJIT. ExceptionHandler クラスを表した JDK のクラスの内部構造へのポインタ．

java_lang_Compiler_start() において初期化され，その後値が変更されることはない．

(3) 関数

- void OpenJIT_compile(ExecEnv *ee, struct methodblock *mb)

  機能概要 指定されたメソッドをコンパイルする．

  機能説明 指定されたメソッドに対し，ベースとなる Java のコンパイルオブジェクトを生成し，compile メソッドを呼び出すことで，バイトコードからネイティブコードへのコンパイルを行う．また，JDK のメソッド情報の内部データを変更し，コンパイルされたネイティブコードが呼び出されるようにする．

  入力データ ee(JDK の実行環境構造体へのポインタ)，mb(JDK のメソッド情報へのポインタ)

  出力データ ネイティブコード

  処理方式 まず，コンパイルのロックをかけて，一つのスレッドでしか次の処理を実行できないようにする．これによって同期がとられた処理の中で，指定されたメソッドが現在コンパイル中であるか調べる．コンパイル中であればロックを外しリターンする．コンパイル中でなければ，mb->CompildedCodeFlags にコンパイル中であることを示すフラグをたてる．そして，mb->invoker, mb->CompiledCode を変更して，今後このメソッドが呼び出されたときには JDK のインタプリタが呼び出されるようにする．これにより Java 言語で記述された OpenJIT システム自身のコンパイルが可能になる．以上を行ったのち，コンパイルのロックを外す．

  メソッドのバイトコードを調べ，このサイズが 1 バイトで，かつ return 命令のときは，何も行わないメソッドであるので，以後このメソッドが呼び出されるときに，特別に用意した nullMethod という関数を呼ぶように設定し，リターンする．

  コンパイルオブジェクトを生成し，そのフィールドを初期化する．コンパイルオブジェクトは OpenJIT . Compile のサブクラスであり，与えられ

たJDKのメソッド情報mbをもとに，methodblock, constantpool, access, nlocals, maxstack, args_size, clazz のフィールドを設定する．さらに，signature, bytecode, exceptionHandler, attributeに対しても，Javaの配列オブジェクトを生成し，その値を設定する．

exceptionHandlerに関してはOpenJIT .ExceptionHandlerクラスのオブジェクト配列であり，OpenJITExceptionClassを使ってその要素であるオブジェクトを生成する．生成したオブジェクトの各フィールド, startPC, endPC, handlerPC, catchTypeをJDKのメソッド情報mbを元に初期化する．

attributeに関しては，事前にOpenJIT_ReadInCompiledCode関数が呼ばれたときにmb->CompiledCodeInfoにコピーされていたデータを，さらにここでコピーする．その後mb->CompiledCodeInfoをフリーする．これはattributeデータをJavaの中から参照できるようにするためである．

そして，環境構造体eeから得られたJavaのスタックフレームを拡張する．これは元のJavaのスタックフレームを破壊しないためである．OpenJITコンパイラはJavaの処理として呼び出されるが，この処理はJavaの本来の実行とは関係ないため，本来の処理のデータを破壊してしまう．これを防ぐためにJavaのスタックフレームを拡張が必要となる．

do_execute_java_method_vararg関数を呼び出して，コンパイルオブジェクトのcompileメソッドを呼び出す．これにより実際のバイトコードからネイティブコードへのコンパイル処理が行われる．

コンパイルに成功した場合，今後このメソッドを呼び出したときにネイティブコードが起動するように，JDKのメソッド情報mbを更新する．また，mb->CompiledCodeInfoにネイティブコードの終了アドレスをセットする．さらに，exceptoinHandlerの各要素を調べて，ネイティブコードのための例外処理用のテーブルを作成して，生成したネイティブコードの領域の最後に付加する．

そして，生成したネイティブコードの領域の命令キャッシュのフラッシュを行う．

Javaのスタックフレームを元に戻し，処理が終了する．

エラー処理 メモリ不足などによりオブジェクトの生成に失敗した場合，コンパイルを行わず，今後このメソッドが呼び出されたときにはJDKのインタプリタが呼び出されるようにする．また，何らかのエラーが起きてコンパイルに失敗した場合も同様である．

- bool_t OpenJIT_invoke(JHandle *o, struct methodblock *mb, int args_size, ExecEnv *ee)

  機能概要 指定されたメソッドをコンパイルし，コンパイルされたネイティブコードを実行する．

  機能説明 この関数はJDKのメソッド情報内にある，メソッド呼び出しのエントリポイントmb->invokerにセットされるべき関数である．JDKのインタプリタから呼び出される．指定されたメソッドをコンパイルし，コンパイルされたネイティブコードを実行する．コンパイルに失敗した場合は，JDKのインタプリタ内のメソッド起動ルーチンが呼び出される．

  入力データ o(オブジェクトへのポインタ)，mb(JDKのメソッド情報へのポインタ)，args_size(引数の数)，ee(JDKの実行環境構造体へのポインタ)

  出力データ ネイティブコード

  処理方式 OpenJIT_Compile(ee, mb)を呼び出した後，mb->invoker(o, mb, args_size, ee)を呼び出す．

  エラー処理 なし．

- void OpenJIT_InitializeForCompiler(ClassClass *cb)

  機能概要 クラスの初期化時にJITコンパイラのための初期化を行う．

  機能説明 この関数は，JDKが新しくクラスをロードしたときに，JITコンパイラのための初期化を行う．Java Native Code APIのフック関数の一つである．

  クラスに含まれるコンパイル可能な(nativeやabstractでない)全てのメソッドの起動関数を再設定し，メソッド起動時にJITコンパイラが呼び出されるようにする．

入力データ cb(JDK のクラス情報へのポインタ)

出力データ なし．

処理方式 クラスに含まれる全てのメソッドを走査する．このメソッド情報 (mb) を調べ，

属性が abstract なものについては何も行わない．

属性が native なものについては，mb->CompiledCode をランタイムルーチンの dispatchLazyNativeMethod にセットする．それ以外は dispatchJIT-Compiler をセットする．mb->CompiledCode はネイティブコードが実行されているときに，その中からメソッド呼び出しを行ったときに呼び出されるエントリポイントである．

また，native メソッドでないものについては mb->inovker を OpenJIT_invoke にセットする．これによりメソッド起動時に OpenJIT_invoke が呼び出されるようになる．

mb->CompiledCodeFlags を初期化する．

エラー処理 なし．

- void OpenJIT_CompilerCompileClass(ClassClass *cb)

機能概要 指定されたクラスのコンパイルを行う．

機能説明 この関数は，JDK において定義されている java.lang.Compiler クラスの compileClass メソッドを実現したものである．

指定されたクラスのコンパイル可能な全てのメソッドのコンパイルを行う．

入力データ cb(JDK のクラス情報へのポインタ)

出力データ なし．

処理方式 JDK のクラス情報へのポインタ cb 内の全てのメソッドを走査する．そのメソッドのうち，既にコンパイルされている，abstract，native なメソッドについて OpenJIT_compile 関数を呼び出す．

エラー処理 なし．

- void OpenJIT_CompilerFreeClass(ClassClass *cb)

機能概要 指定したクラスをフリーする．

機能説明 この関数はJDKシステムから呼び出される．Java Native Code APIのフック関数の一つである．

指定されたクラスに関してJITコンパイルで確保した資源(ネイティブコードや例外テーブルのメモリ領域)の解放を行う．

入力データ cb(JDKのクラス情報へのポインタ)

出力データ なし．

処理方式 JDKのクラス情報へのポインタcb内の全てのメソッドを走査する．そのメソッドのうち，コンパイルされているものについて，もしバイトコードのサイズが1でなかったら，ネイティブコード領域を解放する．

バイトコードのサイズが1のものは，コンパイルされたコードのエントリポイントはnullMethodとなっているため，解放を行わない．

エラー処理 なし．

- bool_t OpenJIT_PCinCompiledCode(unsigned char *pc, struct methodblock *mb)

機能概要 pcが指定されたメソッドのネイティブコード内にあるかどうか調べる．

機能説明 この関数はJDKシステムから呼び出される．Java Native Code APIのフック関数の一つである．

pcが指定されたメソッドのネイティブコード内にあるかどうか調べる．

入力データ pc(プログラムカウンタ)，mb(JDKのメソッド情報へのポインタ)

出力データ pcが指定されたメソッドのネイティブコード内にあれば1をそうでなければ0を返す．

処理方式 pcがmb->CompiledCodeより大きくて，mb->CompiledCodeInfoより小さいことを調べる．

エラー処理 なし．

- unsigned char *OpenJIT_CompiledCodePC(JavaFrame *frame, struct methodblock *mb)

機能概要 Java のスタックフレームと，メソッドからプログラムカウンタの値を得る．

機能説明 この関数は JDK システムから呼び出される．Java Native Code API のフック関数の一つである．

Java のスタックフレームと，メソッドからプログラムカウンタの値を得る．

入力データ JavaFrame(Java のフレーム)，mb(JDK のメソッド情報へのポインタ)

出力データ プログラムカウンタ

処理方式 mb->CompiledCode を返す．

エラー処理 なし．

- void OpenJIT_CompilerEnable()

機能概要 JIT コンパイラの動作を可能にする．

機能説明 この関数は JDK システムから呼び出される．Java Native Code API のフック関数の一つである．

JIT コンパイラの動作を可能にする．

入力データ なし．

出力データ なし．

処理方式 なし．

エラー処理 なし．

- void OpenJIT_CompilerDisable()

機能概要 JIT コンパイラが動作しないようにする．

機能説明 この関数は JDK システムから呼び出される．Java Native Code API のフック関数の一つである．

JIT コンパイラが動作しないようにする．

入力データ なし．

出力データ なし．

処理方式 なし．

エラー処理 なし．

- void OpenJIT_ReadInCompiledCode(void *context, struct methodblock *mb, int attribute_length, unsigned long (*get1byte)(), unsigned long (*get2bytes)(), unsigned long (*get4bytes)(), void (*getNbytes)())

  機能概要 クラスファイルに含まれたアトリビュートを読む．

  機能説明 この関数はJDKシステムから呼び出される．Java Native Code APIのフック関数の一つである．

  JDKシステムによってクラスのロード時に呼び出され，クラスファイルに含まれたアトリビュートを読む．OpenJITシステムではメソッドのコンパイル時にこのデータを読むため，このアトリビュートデータを保存しておき，コンパイル時に読めるようにする．

  入力データ context(クラスのロードコンテキスト)，mb(JDKのメソッド情報へのポインタ) attribute_length(アトリビュートのサイズ), get1byte(アトリビュートを1byte読む関数へのポインタ), get2bytes(アトリビュートを2byte読む関数へのポインタ), get4bytes(アトリビュートを4byte読む関数へのポインタ), getNbytes(アトリビュートをNbyte読む関数へのポインタ)

  出力データ なし．

  処理方式 アトリビュートのサイズだけメモリ領域を確保し，そこにgetNbyteを使ってアトリビュートを読み出す．mb->CompiledCodeInfoにそのメモリ領域のアドレスをセットする．

  エラー処理 なし．

- void java_lang_Compiler_start(void *** linkVector)

  機能概要 OpenJITシステムの初期化を行う．

  機能説明 この関数は，JDKにおいて定義されているjava.lang.Compilerクラスのstartメソッドを実現したものである．

環境変数 JAVA_COMPILER を設定して，JDK を起動することによりこのメソッドが呼び出される．

OpenJIT システムが利用するクラスの初期化を行い，JIT コンパイラが動作するように Java Native Code API のフック関数を設定する．

さらにこのメソッドが呼び出される前に，すでにロードされたクラスについても，JIT コンパイルされるように設定を行う．

入力データ linkVector(Java Native Code API の定義のためのリンクベクタ)

出力データ なし．

処理方式 _compile_lock のための領域を確保して，ロック変数として登録する．

環境変数 OPENJIT_COMPILER が定義されていれば，それをコンパイルのベースクラスとする．定義されていなければ OpenJIT. Sparc クラスをコンパイルのベースクラスとする．コンパイルのベースクラスは OpenJIT .Compile クラスのサブクラスである必要がある．

コンパイルのベースクラスをロードし，その JDK のクラスの内部構造へのポインタを OpenJITClass にセットする．

OpenJIT .ExceptionHandler クラスをロードし，その JDK のクラスの内部構造へのポインタを OpenJITClassExceptionHandlerClass にセットする．

linkvector に対して，

OpenJIT_InitializeForCompiler,

OpenJIT_CompilerFreeClass,

OpenJIT_CompilerCompileClass,

OpenJIT_SignalHandler,

OpenJIT_PCinCompiledCode,

OpenJIT_CompiledCodePC,

OpenJIT_CompiledFramePrev,

OpenJIT_CompilerEnable,

OpenJIT_CompilerDisable,

OpenJIT_ReadInCompiledCode,

OpenJIT_InitializeForCompiler,

OpenJIT_CompilerFreeClass,

OpenJIT_CompilerCompileClass,

OpenJIT_SignalHandler(OpenJIT ランタイムモジュールの関数),

OpenJIT_PCinCompiledCode,

OpenJIT_CompiledCodePC,

OpenJIT_CompiledFramePrev(OpenJIT ランタイムモジュールの関数),

OpenJIT_CompilerEnable,

OpenJIT_CompilerDisable,

OpenJIT_ReadInCompiledCode,

のフック関数を登録する．

すでにロードされている全てのクラスに対し，OpenJIT_InitializeCompiler を呼び出す．

エラー処理 コンパイルベースクラス，OpenJIT . ExceptionHandler クラスのローディングに失敗したとき，”Can’t find class” のメッセージを出力し，OpenJIT システムの動作を行わないようにする．

第 5 章

入出力仕様

5.1 OpenJIT フロントエンドシステム

5.1.1 概要

OpenJIT フロントエンドシステムで用いられる入出力データを小項目単位で説明する．

5.1.2　入出力データ仕様

(1)　OpenJIT コンパイラ基盤機能

OpenJIT コンパイラ基盤機能を構成するサブプログラムを図 5.1に示す．これらサブプログラム間で用いられる入出力データない．

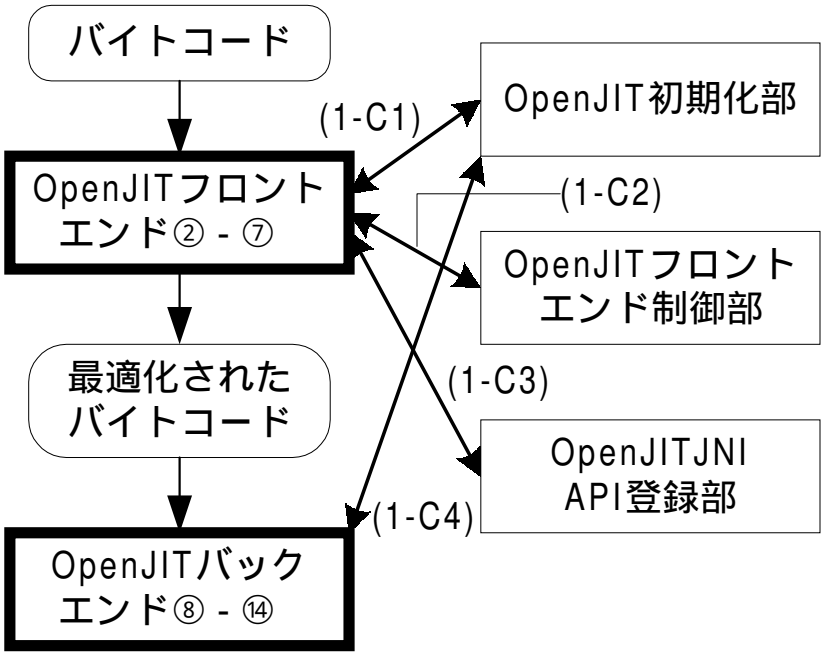

図 5.1: OpenJIT コンパイラ基盤機能

(2) OpenJIT バイトコードディスコンパイラ機能

OpenJIT バイトコードディスコンパイラ機能を構成するサブプログラムを図 5.2に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(2-D1) バイトコード OpenJIT.Compile クラスの bytecode フィールドがこれに相当する．次のように宣言されている．

```
public byte bytecode[];
```

(2-D2) コントロールフローグラフ OpenJIT.frontend.discompiler.ControlFlowGraph クラスがこれを定義する．

(2-D3) AST AST は，以下のクラス群が定義する種々のツリーノードが再帰的に組合わさったオブジェクトとして定義されている．

- OpenJIT.frontend.tree.AddExpression クラス
- OpenJIT.frontend.tree.AndExpression クラス
- OpenJIT.frontend.tree.ArrayAccessExpression クラス
- OpenJIT.frontend.tree.ArrayExpression クラス
- OpenJIT.frontend.tree.AssignAddExpression クラス
- OpenJIT.frontend.tree.AssignBitAndExpression クラス
- OpenJIT.frontend.tree.AssignBitOrExpression クラス
- OpenJIT.frontend.tree.AssignBitXorExpression クラス
- OpenJIT.frontend.tree.AssignDivideExpression クラス
- OpenJIT.frontend.tree.AssignExpression クラス
- OpenJIT.frontend.tree.AssignMultiplyExpression クラス
- OpenJIT.frontend.tree.AssignOpExpression クラス
- OpenJIT.frontend.tree.AssignRemainderExpression クラス
- OpenJIT.frontend.tree.AssignShiftLeftExpression クラス
- OpenJIT.frontend.tree.AssignShiftRightExpression クラス

- OpenJIT.frontend.tree.AssignSubtractExpression クラス
- OpenJIT.frontend.tree.AssignUnsignedShiftRightExpression クラス
- OpenJIT.frontend.tree.BinaryArithmeticExpression クラス
- OpenJIT.frontend.tree.BinaryAssignExpression クラス
- OpenJIT.frontend.tree.BinaryBitExpression 抽象クラス
- OpenJIT.frontend.tree.BinaryCompareExpression クラス
- OpenJIT.frontend.tree.BinaryEqualityExpression クラス
- OpenJIT.frontend.tree.BinaryExpression クラス
- OpenJIT.frontend.tree.BinaryLogicalExpression 抽象クラス
- OpenJIT.frontend.tree.BinaryShiftExpression クラス
- OpenJIT.frontend.tree.BitAndExpression クラス
- OpenJIT.frontend.tree.BitNotExpression クラス
- OpenJIT.frontend.tree.BitOrExpression クラス
- OpenJIT.frontend.tree.BitXorExpression クラス
- OpenJIT.frontend.tree.BooleanExpression クラス
- OpenJIT.frontend.tree.BreakStatement クラス
- OpenJIT.frontend.tree.ByteExpression クラス
- OpenJIT.frontend.tree.CaseStatement クラス
- OpenJIT.frontend.tree.CastExpression クラス
- OpenJIT.frontend.tree.CatchStatement クラス
- OpenJIT.frontend.tree.CharExpression クラス
- OpenJIT.frontend.tree.CheckContext クラス
- OpenJIT.frontend.tree.CodeContext クラス
- OpenJIT.frontend.tree.CommaExpression クラス
- OpenJIT.frontend.tree.CompoundStatement クラス
- OpenJIT.frontend.tree.ConditionVars クラス

- OpenJIT.frontend.tree.ConditionalExpression クラス
- OpenJIT.frontend.tree.ConstantExpression クラス
- OpenJIT.frontend.tree.Context クラス
- OpenJIT.frontend.tree.ContinueStatement クラス
- OpenJIT.frontend.tree.ConvertExpression クラス
- OpenJIT.frontend.tree.DeclarationStatement クラス
- OpenJIT.frontend.tree.DivRemExpression クラス
- OpenJIT.frontend.tree.DivideExpression クラス
- OpenJIT.frontend.tree.DoStatement クラス
- OpenJIT.frontend.tree.DoubleExpression クラス
- OpenJIT.frontend.tree.EqualExpression クラス
- OpenJIT.frontend.tree.ExprExpression クラス
- OpenJIT.frontend.tree.Expression クラス
- OpenJIT.frontend.tree.ExpressionStatement クラス
- OpenJIT.frontend.tree.FieldExpression クラス
- OpenJIT.frontend.tree.FinallyStatement クラス
- OpenJIT.frontend.tree.FloatExpression クラス
- OpenJIT.frontend.tree.ForStatement クラス
- OpenJIT.frontend.tree.GreaterExpression クラス
- OpenJIT.frontend.tree.GreaterOrEqualExpression クラス
- OpenJIT.frontend.tree.IdentifierExpression クラス
- OpenJIT.frontend.tree.IfStatement クラス
- OpenJIT.frontend.tree.IncDecExpression クラス
- OpenJIT.frontend.tree.InlineMethodExpression クラス
- OpenJIT.frontend.tree.InlineNewInstanceExpression クラス
- OpenJIT.frontend.tree.InlineReturnStatement クラス

- OpenJIT.frontend.tree.InstanceOfExpression クラス
- OpenJIT.frontend.tree.IntExpression クラス
- OpenJIT.frontend.tree.IntegerExpression クラス
- OpenJIT.frontend.tree.LengthExpression クラス
- OpenJIT.frontend.tree.LessExpression クラス
- OpenJIT.frontend.tree.LessOrEqualExpression クラス
- OpenJIT.frontend.tree.LocalField クラス
- OpenJIT.frontend.tree.LongExpression クラス
- OpenJIT.frontend.tree.MethodExpression クラス
- OpenJIT.frontend.tree.MultiplyExpression クラス
- OpenJIT.frontend.tree.NaryExpression クラス
- OpenJIT.frontend.tree.NegativeExpression クラス
- OpenJIT.frontend.tree.NewArrayExpression クラス
- OpenJIT.frontend.tree.NewInstanceExpression クラス
- OpenJIT.frontend.tree.Node クラス
- OpenJIT.frontend.tree.NotEqualExpression クラス
- OpenJIT.frontend.tree.NotExpression クラス
- OpenJIT.frontend.tree.NullExpression クラス
- OpenJIT.frontend.tree.OrExpression クラス
- OpenJIT.frontend.tree.PositiveExpression クラス
- OpenJIT.frontend.tree.PostDecExpression クラス
- OpenJIT.frontend.tree.PostIncExpression クラス
- OpenJIT.frontend.tree.PreDecExpression クラス
- OpenJIT.frontend.tree.PreIncExpression クラス
- OpenJIT.frontend.tree.RemainderExpression クラス
- OpenJIT.frontend.tree.ReturnStatement クラス

- OpenJIT.frontend.tree.ShiftLeftExpression クラス
- OpenJIT.frontend.tree.ShiftRightExpression クラス
- OpenJIT.frontend.tree.ShortExpression クラス
- OpenJIT.frontend.tree.Statement クラス
- OpenJIT.frontend.tree.StringExpression クラス
- OpenJIT.frontend.tree.SubtractExpression クラス
- OpenJIT.frontend.tree.SuperExpression クラス
- OpenJIT.frontend.tree.SwitchStatement クラス
- OpenJIT.frontend.tree.SynchronizedStatement クラス
- OpenJIT.frontend.tree.ThisExpression クラス
- OpenJIT.frontend.tree.ThrowStatement クラス
- OpenJIT.frontend.tree.TryStatement クラス
- OpenJIT.frontend.tree.TypeExpression クラス
- OpenJIT.frontend.tree.UnaryExpression クラス
- OpenJIT.frontend.tree.UnsignedShiftRightExpression クラス
- OpenJIT.frontend.tree.VarDeclarationStatement クラス
- OpenJIT.frontend.tree.WhileStatement クラス

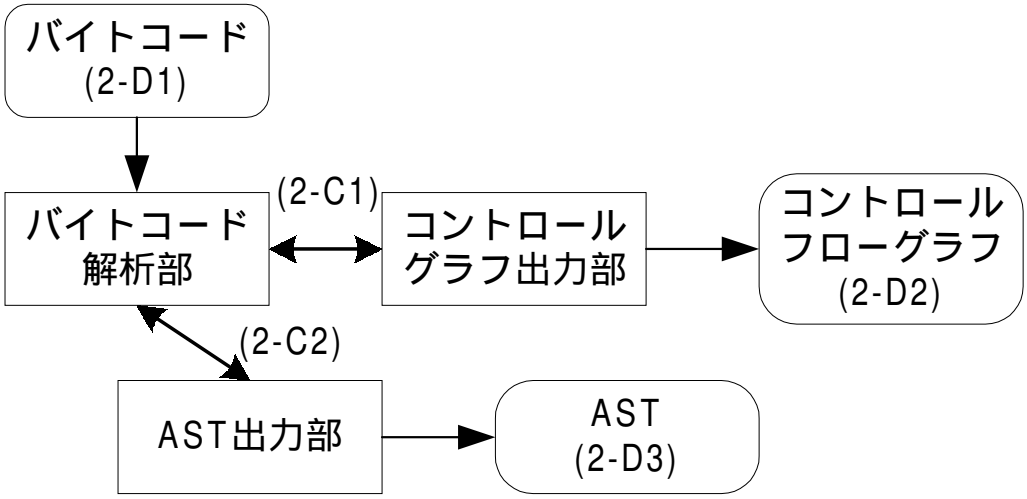

図 5.2: OpenJIT バイトコードディスコンパイラ機能

(3) OpenJIT クラスファイルアノテーション解析機能

OpenJIT クラスファイルアノテーション解析機能を構成するサブプログラムを図5.3に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(3-D1) アノテーションを付記したクラスファイルOpenJIT.frontend.discompiler.RuntimeMethodInformation クラスがこれを定義する．

(3-D2) AST (2-D3) を参照．

(3-D3) AST に対する付加データ OpenJIT.frontend.discompiler.Annotation クラスがこれを定義する．

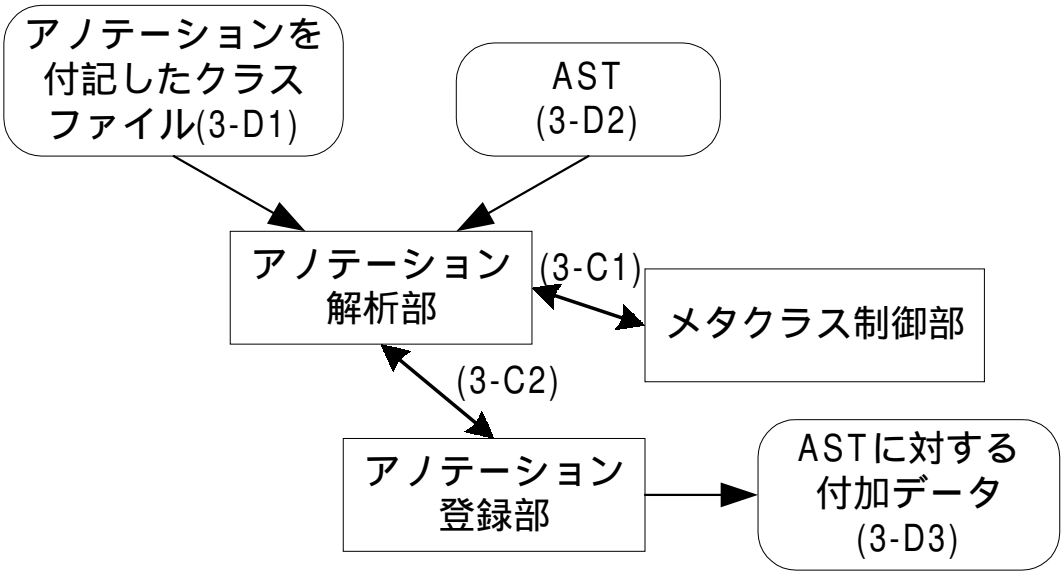

図 5.3: OpenJIT クラスファイルアノテーション解析機能

(4) OpenJIT 最適化機能

OpenJIT 最適化機能を構成するサブプログラムを図 5.4に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(4-D1) 最適化されたバイトコード OpenJIT.Compile クラスの bytecode フィールドがこれに相当する．次のように宣言されている．

```
public byte bytecode[];
```

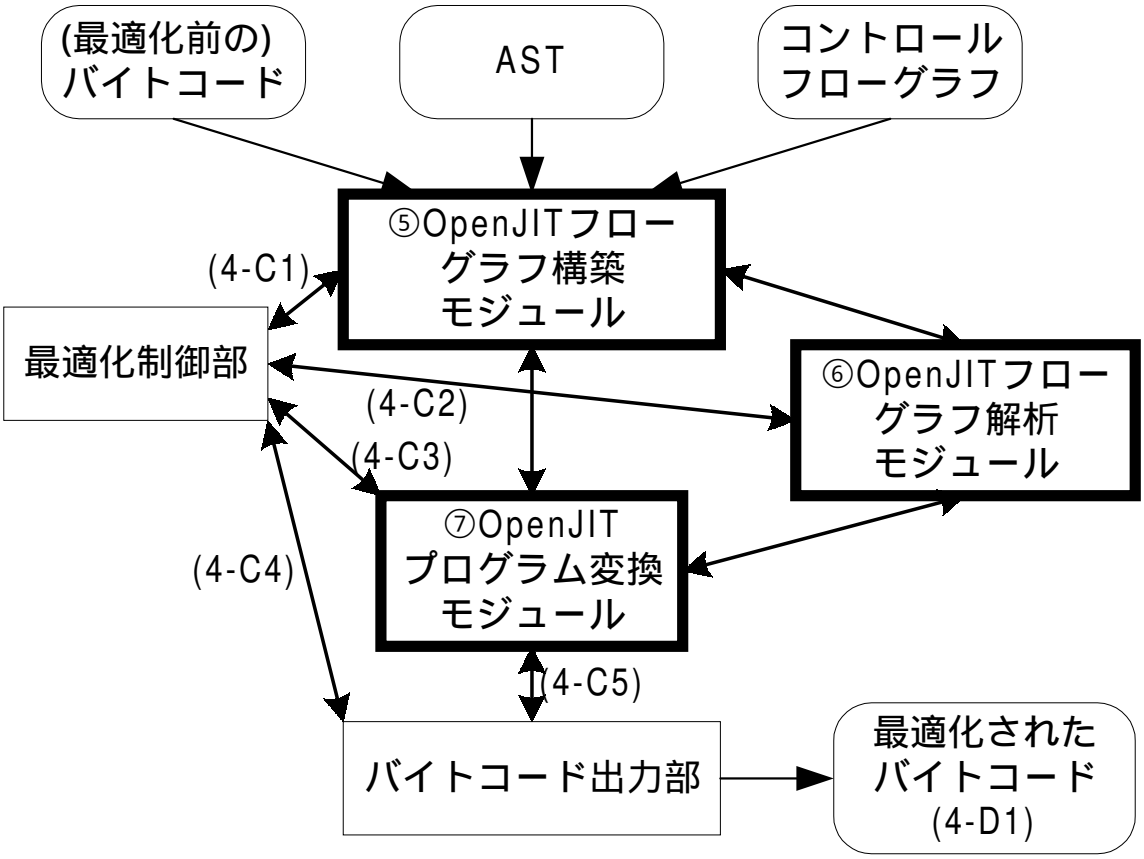

図 5.4: OpenJIT 最適化機能

(5) OpenJIT フローグラフ構築機能

OpenJIT フローグラフ構築機能を構成するサブプログラムを図 5.5に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(5-D1) AST

(2-D3) を参照．

(5-D2) コントロールフローグラフ

OpenJIT.frontend.flowgraph.FlowGraph クラスの cfg フィールドがこれに相当する．これは次のように宣言されている．

```
ControlFlowGraph cfg;
```

(5-D3) クラスファイル間のクラス階層情報

OpenJIT.frontend.flowgraph.FlowGraph クラスの cinfo フィールドがこれに相当する．これは次のように宣言されている．

```
Class[] cinfo;
```

(5-D4) データフローグラフ

OpenJIT.frontend.flowgraph.FlowGraph クラスの dfg フィールドがこれに相当する．これは次のように宣言されている．

```
DataFlowGraph dfg;
```

(5-D5) コントロール依存グラフ

OpenJIT.frontend.flowgraph.FlowGraph クラスの cdg フィールドがこれに相当する．これは次のように宣言されている．

```
ControlDependencyGraph cdg;
```

(5-D6) クラス階層グラフ

OpenJIT.frontend.flowgraph.FlowGraph クラスの chg フィールドがこれに相当する．これは次のように宣言されている．

```
ClassHierarchyGraph chg;
```

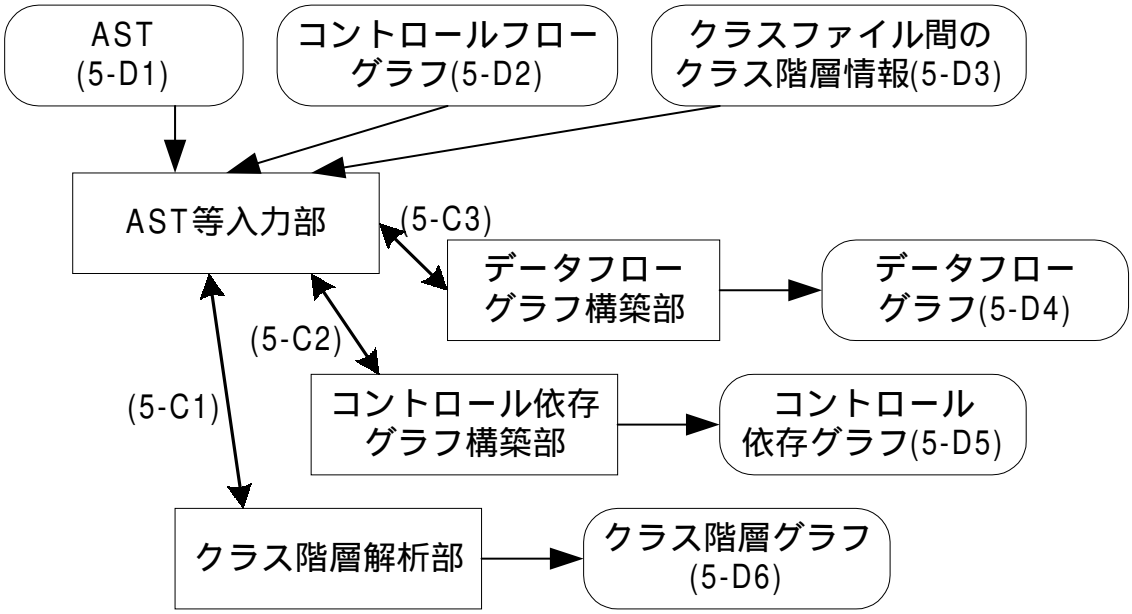

図 5.5: OpenJIT フローグラフ構築機能

(6) OpenJIT フローグラフ解析機能

OpenJIT フローグラフ解析機能を構成するサブプログラムを図 5.6に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(6-D1) コントロールフローグラフ

OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラスの cfg フィールドがこれに相当する．これは次のように宣言されている．

```
ControlFlowGraph cfg;
```

(6-D2) コントロール依存グラフ

OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラスの cdg フィールドがこれに相当する．これは次のように宣言されている．

```
ControlDependencyGraph cdg;
```

(6-D3) データフローグラフ

OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラスの dfg フィールドがこれに相当する．これは次のように宣言されている．

```
DataFlowGraph dfg;
```

(6-D4) クラス階層グラフ

OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラスの chg フィールドがこれに相当する．これは次のように宣言されている．

```
ClassHierarchyGraph chg;
```

(6-D5) プログラム解析結果のデータ

OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラス自身がこれに相当する．これは次のように宣言されている．

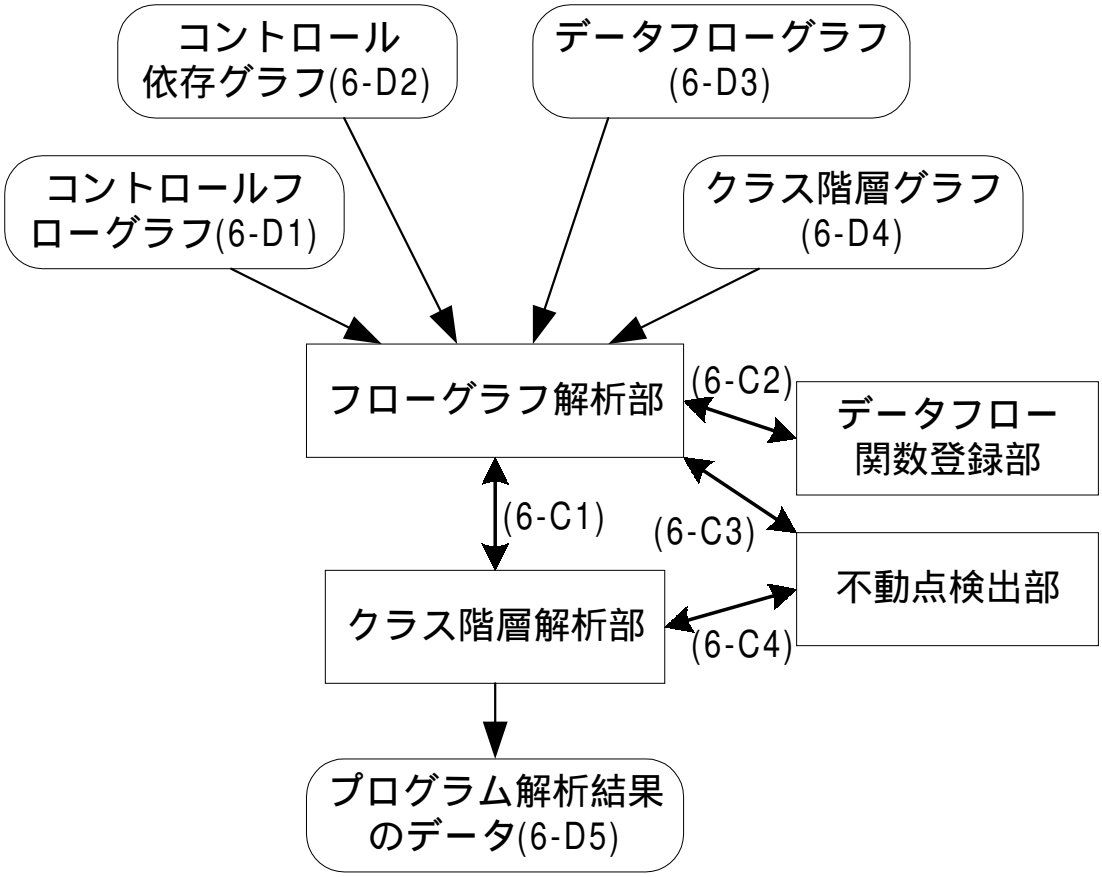

図 5.6: OpenJIT フローグラフ解析機能

(7) OpenJIT プログラム変換機能

OpenJIT プログラム変換機能を構成するサブプログラムを図 5.7に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(7-D1) AST

OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラスの ast フィールドがこれに相当する．これは次のように宣言されている．

```
Node ast;
```

(7-D2) プログラム解析結果のデータ

OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラスの fga フィールドがこれに相当する．これは次のように宣言されている．

```
FlowGraphAnalysis fga;
```

(7-D3) 変換された AST

OpenJIT.frontend.flowgraph.FlowGraphAnalysis クラスのメソッド transform() の返り値がこれに相当する．このメソッドは次のように宣言されている．

```
 \item public Node transform(Node target)
```

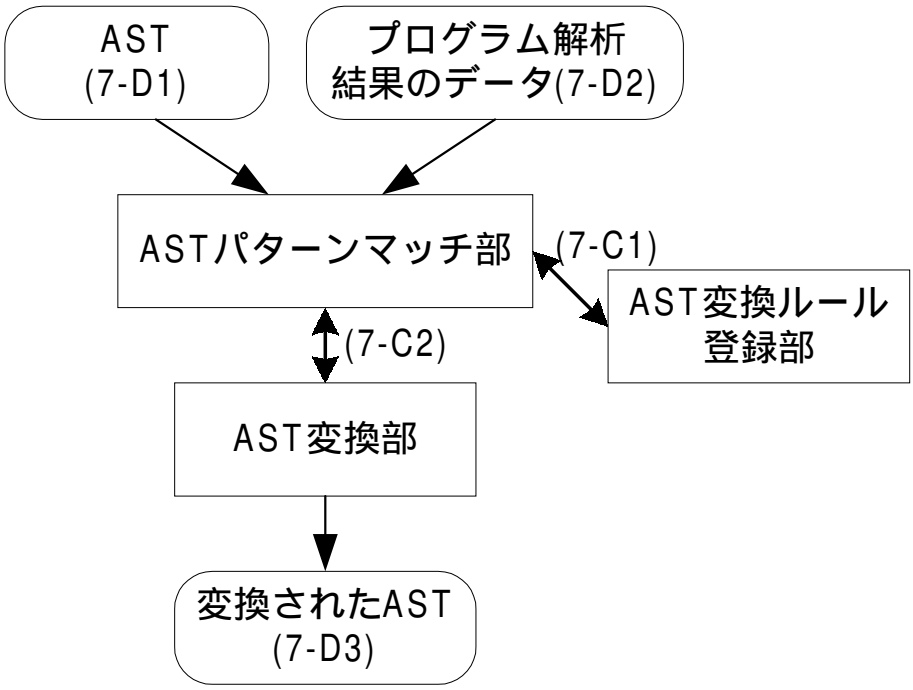

図 5.7: OpenJIT プログラム変換機能

5.2 OpenJIT バックエンドシステム

5.2.1 概要

OpenJIT バックエンドシステムで用いられる入出力データを小項目単位で説明する．

5.2.2　入出力データ仕様

(1)　OpenJIT ネイティブコード変換機能

OpenJIT ネイティブコード変換機能を構成するサブプログラムを図 5.8に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(8-D1) バイトコード

OpenJIT.Compile クラスの bytecode フィールドがこれに相当する．次のように宣言されている．

```
public byte bytecode[];
```

(8-D2) SPARC ネイティブコード

Sparc Version 8 の機械語命令．1 命令が必ず 4byte の固定長であり，4byte 単位にアラインメントされている必要がある．

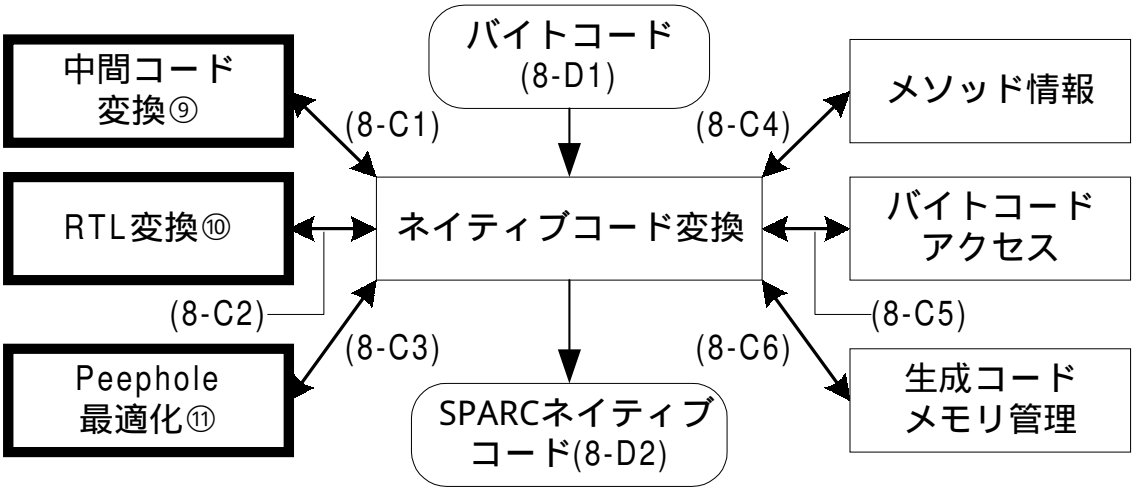

図 5.8: OpenJIT ネイティブコード変換機能

(2) OpenJIT 中間コード変換機能

OpenJIT 中間コード変換機能を構成するサブプログラムを図 5.9に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(9-D1) バイトコード

OpenJIT.Compile クラスの bytecode フィールドがこれに相当する．次のように宣言されている．

```
public byte bytecode[];
```

(9-D2) 中間言語

実体は OpenJIT.Compile クラスの bcinfo フィールドである．次のように宣言されている．

```
protected BCinfo bcinfo[];
```

次のクラスから構成される．

- OpenJIT.BCinfo クラス
- OpenJIT.ILnode クラス
- OpenJIT.LinkedList クラス

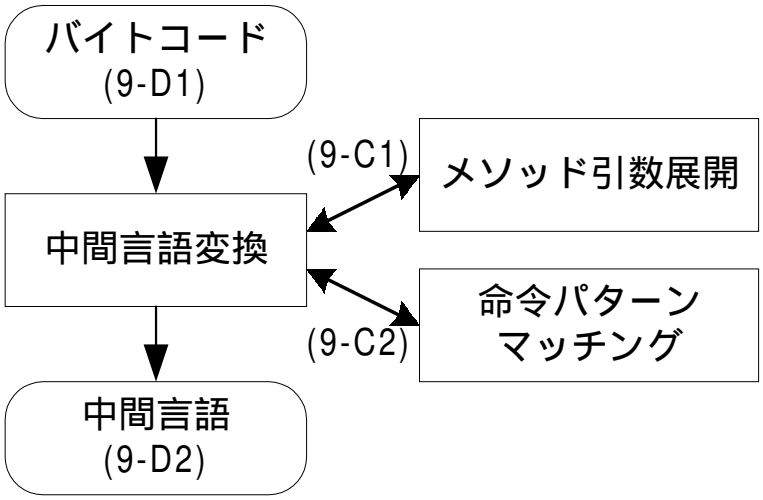

図 5.9: OpenJIT 中間コード変換機能

(3) OpenJIT RTL 変換機能

OpenJIT RTL 変換機能を構成するサブプログラムを図 5.10に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(10-D1) 中間言語

実体は OpenJIT.Compile クラスの bcinfo フィールドである．次のように宣言されている．

```
protected BCinfo bcinfo[];
```

次のクラスから構成される．

- OpenJIT.BCinfo クラス
- OpenJIT.ILnode クラス
- OpenJIT.LinkedList クラス

(10-D2) 基本ブロック情報

実体は OpenJIT.BCinfo クラスの bb フィールドである．次のように宣言されている．

```
BBinfo bb;
```

次のクラスから構成される．

- OpenJIT.BBinfo クラス

(10-D3) RTL

実体は OpenJIT.Compile クラスの bcinfo フィールドである．次のように宣言されている．

```
protected BCinfo bcinfo[];
```

次のクラスから構成される．

- OpenJIT.BCinfo クラス
- OpenJIT.ILnode クラス
- OpenJIT.LinkedList クラス

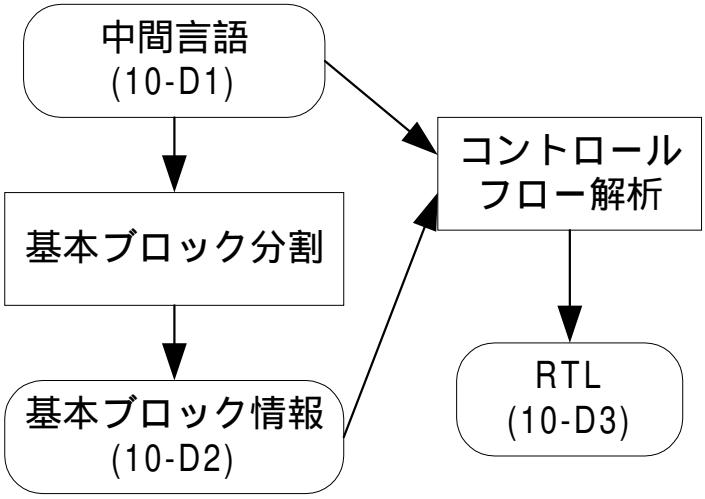

図 5.10: OpenJIT RTL 変換機能

(4) OpenJIT Peephole 最適化機能

OpenJIT Peephole 最適化機能を構成するサブプログラムを図 5.11に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(11-D1) 基本ブロック情報

実体は OpenJIT.BCinfo クラスの bb フィールドである．次のように宣言されている．

```
BBinfo bb;
```

次のクラスから構成される．

- OpenJIT.BBinfo クラス

(11-D2) RTL

実体は OpenJIT.Compile クラスの bcinfo フィールドである．次のように宣言されている．

```
protected BCinfo bcinfo[];
```

次のクラスから構成される．

- OpenJIT.BCinfo クラス
- OpenJIT.ILnode クラス
- OpenJIT.LinkedList クラス

(11-D3) 最適化された RTL

実体は OpenJIT.Compile クラスの bcinfo フィールドである．次のように宣言されている．

```
protected BCinfo bcinfo[];
```

次のクラスから構成される．

- OpenJIT.BCinfo クラス
- OpenJIT.ILnode クラス
- OpenJIT.LinkedList クラス

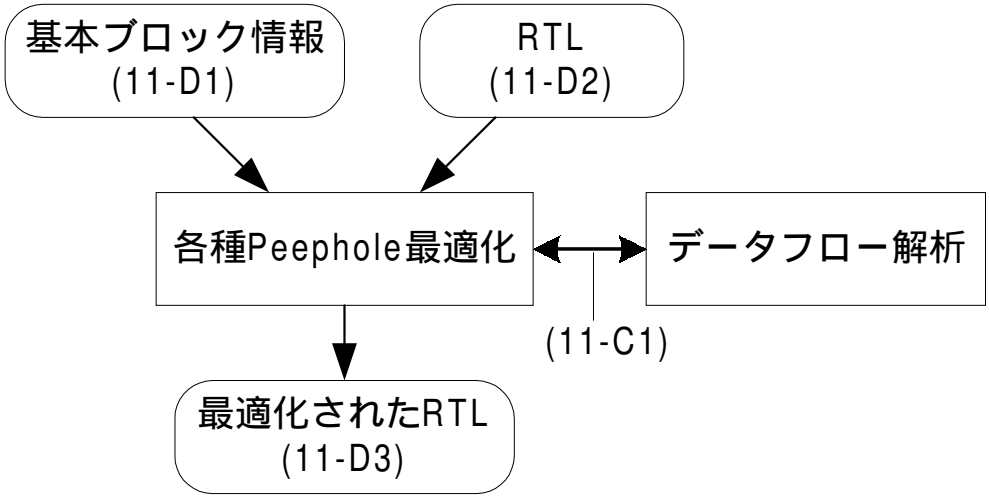

図 5.11: OpenJIT Peephole 最適化機能

(5) OpenJIT レジスタ割付機能

OpenJIT レジスタ割付機能を構成するサブプログラムを図 5.12に示す．これらサブプログラム間で用いられる入出力データの一覧を示す．

(12-D1) 仮想レジスタ番号

実体は OpenJIT.RegAlloc クラスの virReg フィールドである．次のように宣言されている．

```
VirReg virRegs[]
```

次のクラスから構成される．

- OpenJIT.RegAlloc$VirReg クラス

(12-D2) スピル / フィルコード

Sparc Version 8 のロードやストアを行う機械語命令．1 命令が必ず 4byte の固定長であり，4byte 単位にアラインメントされている必要がある．

スピルコードはアセンブラで次のフォーマットになる．

```
st <register>, [%sp + <offset>]
```

フィルコードはアセンブラで次のフォーマットになる．

```
ld [%sp + <offset>], <register>
```

<register> はスピル / フィルを行うレジスタ，<offset> は 4 の倍数値である．

(12-D3) 物理レジスタ番号

実体は OpenJIT.RegAlloc クラスの iTmp, fTmp フィールドである．次のように宣言されている．

```
PhyRegs iTmp;
PhyRegs fTmp;
```

次のクラスから構成される．

- OpenJIT.RegAlloc$PhyRegs クラス

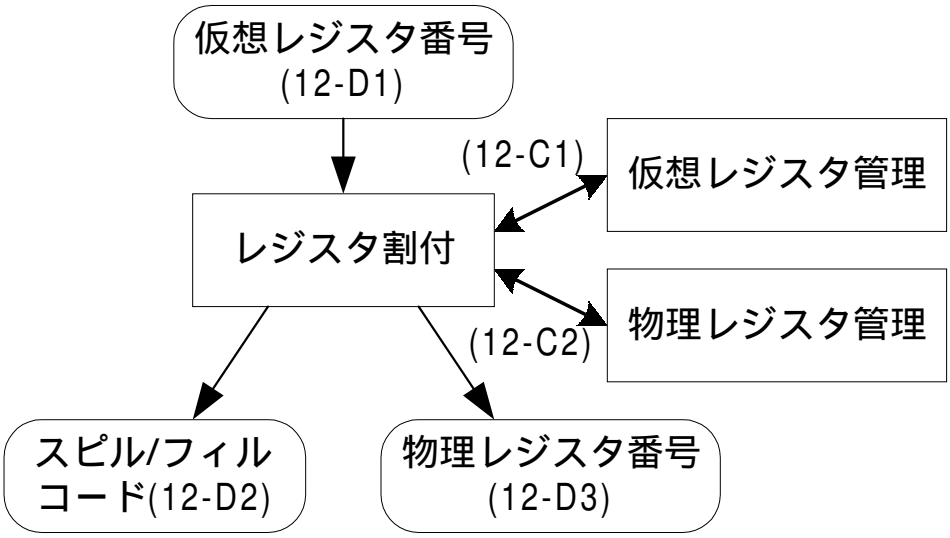

図 5.12: OpenJIT レジスタ割付機能

第 6 章

ファイル仕様

6.1 概要

ファイル仕様としては以下のものがある．

- Java Class ファイル

6.2　物理ファイル仕様

- Java Class ファイル

  The Java Virtual Machine Specification[1] 参照 .

6.3　物理データベース仕様

本システムでは該当しない．

第 7 章

内部データ仕様

サブプログラム間での内部データの受渡しに関しては，第 4 章に全て記述した．

第 8 章

システムの性能及び容量

OpenJIT フロントエンドシステム，OpenJIT バックエンドシステムは，以下の性能・容量上の仕様を満たすものとする．

処理容量 JDK 1.1.4 以降の Java VM に準ずる．

処理時間 JDK 1.1.4 以降の Java VM の 10 倍以内の時間で，ユーザプログラムの実行が完了するものとする．

参考文献

[1] Sun Microsystems, Inc. “The Java Virtual Machine Specification”, 1996.