

目 次

第 I 部	コンパイラの概要	2
1.	はじめに	4
1.1	コンパイラとは	4
1.2	変換系と通訳系	7
2.	コンパイラの簡単な例	10
2.1	後置記法	10
2.2	スタック	12
2.3	簡単なコンパイラの例	18
2.4	コンパイラの論理的構造	21
2.5	コンパイラの物理的構造	25
第 II 部	コンパイラの構成	31
3.	文法と言語	33
3.1	バックス記法	33
3.2	構文図式	35
3.3	文法と言語の形式的定義	37
3.4	解析木	41
3.5	あいまいな文法	42
4.	字句解析	45
4.1	文字読みとり	45
4.2	字句読み取り	46

4.3	正規表現と有限オートマトン	55
4.3.1	正規表現	55
4.3.2	正規表現から非決定性有限オートマトンへ	57
4.3.3	非決定性有限オートマトンから決定性有限オートマトンへ	60
4.3.4	有限オートマトンの状態数の最小化	62
4.3.5	字句解析器生成系	64
4.4	字句読み取りプログラムの例	65
4.4.1	浮動小数点定数の読み取り	65
4.4.2	コメントの読み取り	73
4.4.3	最長一致と最短一致	75
5.	構文解析	79
5.1	構文解析手法の簡単な歴史	79
5.2	下向き構文解析法	81
5.2.1	下向き構文解析法とその問題点	81
5.2.2	LL(1) 文法	86
5.2.3	再帰的下向き構文解析プログラム	91
5.2.4	文法から再帰的下向き構文解析プログラムへ	94
5.2.5	非再帰的下向き構文解析法	96
5.2.6	あいまいな文法の扱い	98
5.3	LR 構文解析法	101
5.3.1	上向き構文解析法	101
5.3.2	LR 構文解析の概略	102
5.3.3	SLR(1) 構文解析	107
5.3.4	LR(1) 構文解析	110
5.3.5	LALR(1) 構文解析	113
5.3.6	あいまいな文法の扱い	115
5.3.7	正規右辺文法の LR 構文解析	116
5.4	演算子順位構文解析法	117
5.4.1	演算子順位文法	117
5.4.2	演算子順位行列による構文解析	119
5.4.3	演算子順位関数	123
5.5	構文解析法の選択	127
5.6	構文解析器生成系	129

6. 意味解析	131
6.1 意味解析とは	131
6.2 記号表	132
6.2.1 記号表の情報	132
6.2.2 記号表の探索	133
6.2.3 ブロック構造と記号表	138
6.3 属性文法	141
6.3.1 属性文法の簡単な例	141
6.3.2 属性文法の定義	146
6.3.3 属性評価器	150
6.3.4 1パス型属性文法	153
6.3.5 属性文法の拡張と応用	162
7. 誤りの処理	166
7.1 誤りの発見	166
7.1.1 構文上の誤り	166
7.1.2 意味上の誤り	167
7.2 誤りの情報の出力	169
7.3 誤りの修復	170
7.4 正常処理への復帰	174
8. 実行時記憶域と仮想マシン	181
8.1 実行時記憶域	181
8.1.1 基本データ型の表現	182
8.1.2 配列型と構造体型の表現	182
8.1.3 クラスの表現	185
8.1.4 手続きの使う記憶域	193
8.2 仮想マシンと通訳系	200
8.2.1 仮想マシンとは	200
8.2.2 仮想マシンの命令	200
8.2.3 仮想マシン語への変換	210
8.2.4 仮想マシンの実現 (通訳系)	210
9. 目的コードの生成	213

9.1	中間語と機械語	213
9.2	式のコード生成	218
9.2.1	オペランド参照のコード	218
9.2.2	算術式のコード	220
9.3	文のコード生成	227
9.3.1	分岐文のコード	228
9.3.2	手続き呼出しのコード	232
9.3.3	例外処理のコード	235
9.4	コード生成器生成系	236
9.4.1	木のパターンマッチングによるコード生成	237
9.4.2	構文解析法によるパターンマッチングでのコード生成	240
9.4.3	ダイナミックプログラミングによるコード生成	243
9.4.4	コード生成器作成時のダイナミックプログラミング	248
9.4.5	JIT 向きの方法	253
9.4.6	SSA グラフと二次計画法による命令選択	256
9.4.7	リターゲッタブルコード生成の実装	260
9.5	実行時コード生成	262
9.5.1	動的コード生成	263
9.5.2	JIT コンパイラ	266

第 III 部 目的コードの最適化 270

10.	最適化とは	272
10.1	最適化の例 (行列の掛け算の例)	274
10.1.1	配列要素の配置と番地計算	275
10.1.2	最適化の例 1 (スカラ計算機の場合)	276
10.1.3	最適化の例 2 (アクセス効率とループ変換)	279
10.1.4	最適化の例 3 (ベクトル計算機の場合)	280
10.1.5	最適化の例 4 (並列計算機の場合)	282
11.	最適化の方法	285
11.1	命令の実行回数を減らす	286
11.1.1	共通部分式の削除	286

11.1.2	定数の畳み込み	287
11.1.3	命令のループの外への移動	287
11.1.4	部分冗長性の除去	288
11.1.5	ループの変換	289
11.1.6	式の性質を利用して実行を省略する	291
11.1.7	無用命令の削除	292
11.1.8	複写の削除	292
11.1.9	手続き呼出しの特殊化	293
11.1.10	オブジェクトのインライン割当て	294
11.1.11	判定の置き換え	295
11.1.12	のぞき穴式最適化	295
11.2	より速い命令の利用	296
11.2.1	レジスタ割付け	296
11.2.2	特殊な命令の利用	297
11.2.3	メモリアクセス順序の最適化 (メモリ階層の有効利用)	298
11.2.4	演算の強さの軽減	299
11.3	並列度を上げる	300
11.3.1	命令レベル並列実行	301
11.3.2	データ並列実行	302
11.4	最適化の方法の適用順序	309
12.	制御とデータの流れの解析	314
12.1	制御フローグラフ	314
12.2	データの流れの解析	316
12.2.1	共通部分式の削除 (基本ブロックと拡張基本ブロック内)	317
12.2.2	変数の使用に対応する定義の解析	320
12.2.3	データの流れの等式の一般形	325
12.2.4	共通部分式の削除 (大域的)	326
12.2.5	ループ	328
12.2.6	ループの外への移動	332
12.2.7	変数の生と死の解析	334
12.2.8	無用命令の削除	335
12.2.9	定数伝播と畳み込み	335
12.2.10	部分冗長性の除去	336

12.2.11	手続き間解析	345
12.2.12	ポインタ解析	356
12.2.13	最適化の各操作の適用順序	365
13.	静的単一代入形式 (SSA 形式)	370
13.1	SSA 形式の概要	370
13.2	SSA 形式の求め方	372
13.2.1	流れグラフから支配境界を求める	375
13.2.2	支配境界に ϕ 関数を挿入する	377
13.2.3	変数の名前替えをする	379
13.2.4	SSA 形式から通常の形式に戻す	380
13.3	制御依存グラフとプログラム依存グラフ	389
13.4	SSA 形式での最適化のアルゴリズム	392
13.4.1	無用命令の削除	392
13.4.2	共通部分式の削除	392
13.4.3	ループの外への移動	400
13.4.4	帰納変数	400
13.4.5	演算の強さの軽減と判定の置き換え	402
13.4.6	定数伝播	405
13.4.7	部分冗長性の除去	409
13.4.8	ポインタ解析	411
14.	命令スケジューリング (並列実行)	415
14.1	基本ブロック内の命令スケジューリング	416
14.2	基本ブロックにまたがった命令スケジューリング	419
14.2.1	トレーススケジューリング	420
14.2.2	パーコレーションスケジューリング	420
14.2.3	スーパーブロックスケジューリング	423
14.2.4	ハイパブロックスケジューリング	424
14.3	ソフトウェアパイプラインニング	425
14.3.1	モジュロスケジューリング	426
14.3.2	ソフトウェアパイプラインニングにおけるレジスタ割付け	428
14.3.3	ループの終了判定命令	431
14.4	条件文がある場合のソフトウェアパイプラインニング	435

14.4.1	階層的縮約	437
14.4.2	条件変換	438
14.4.3	逆条件変換	440
14.5	整数線形計画法によるソフトウェアパイプラインニング	442
15.	レジスタ割付け	446
15.1	簡単な割付け	447
15.2	生存区間を使ったレジスタ割付け	450
15.2.1	区間グラフ	450
15.2.2	循環区間グラフ	453
15.2.3	接続グラフ	456
15.2.4	螺旋グラフ	460
15.2.5	階層的区間グラフ	461
15.3	生存区間の干渉グラフを使ったレジスタ割付け	463
15.3.1	基本アルゴリズム	464
15.3.2	生存区間の分割	468
15.3.3	生存区間の合併	472
15.3.4	SSA形式の利用	474
15.3.5	レジスタの特異性への対処	477
15.4	整数線形計画法によるレジスタ割付け	481
15.5	二次計画法によるレジスタ割付け	483
16.	配列要素の依存解析	488
16.1	データの依存関係	490
16.1.1	フロー依存, または真の依存	490
16.1.2	逆依存	490
16.1.3	出力依存	491
16.1.4	入力依存	492
16.2	1重ループ内の依存関係とベクトル化	493
16.2.1	多重ループの最内側ループのベクトル化	495
16.2.2	単純変数の配列化	496
16.2.3	条件変換	497
16.2.4	漸化関係, 総和などの特殊パターン	497
16.3	多重ループにおける依存関係	499

16.4	データ依存関係の解析法	502
16.4.1	1重のループでのデータ依存解析	502
16.4.2	2重ループでのデータ依存解析の例	506
16.4.3	一般的なデータ依存解析	511
17.	ループ変換	515
17.1	ループ変換の条件	515
17.2	各種のループ変換	517
17.2.1	ループ分配	517
17.2.2	ループ融合	519
17.2.3	ループ交換	520
17.2.4	ループ逆転	524
17.2.5	ループ傾斜	525
17.2.6	波頭変換	528
17.2.7	ループ細分	530
17.2.8	ループマイル化	531
17.2.9	ループ展開	533
17.2.10	ループ合体	535
17.2.11	ループつぶし	536
17.2.12	ループ皮むき	536
17.3	ループ変換の適用順序	537
18.	データ分散と通信	539
18.1	各プロセッサでの私物化	540
18.2	配列分散の解析	541
18.3	計算分散の解析	544
18.4	通信の解析	545
18.5	通信の最適化	550
18.6	データ分散の自動化	555
18.6.1	セグメント分け	557
18.6.2	データ分散の候補	557
18.6.3	データ分散の仕方	559
18.6.4	データ分散と計算分散	560
18.6.5	データの再分散	565

18.7	配列へのアクセスが規則的でない場合	565
19.	データの流れの解析の別法	569
19.1	Datalog 言語	569
19.2	二分決定グラフ BDD	573
19.3	文脈自由言語到達性	577
19.4	種々のポインタ解析	579
19.4.1	フィールドの考慮	579
19.4.2	C のポインタ解析	582
19.4.3	メソッド呼び出しの考慮	583
20.	オブジェクト指向言語での最適化	588
20.1	メソッド呼出しの高速化	588
20.1.1	仮想メソッド呼出しとインターフェース呼出し	588
20.1.2	脱仮想化	590
20.2	例外検査の除去	591
20.2.1	配列の範囲検査	592
20.2.2	null 検査	594
20.2.3	例外検査の順序	596
20.3	脱出解析	597
20.4	例外ハンドラの見つけ方	599