C-03

リターゲッタブル・コンパイラのための命令パターン生成
Instruction Pattern Generation for Retargetable Compiler

岸本 充司† 石浦 菜岐佐† 今井 正治††
Atsushi Kishimoto Nagisa Ishiura Masaharu Imai

1 はじめに

近年, 特定用途専用の信号処理プロセッサ (DSP) の利用の拡大に伴い, プログラムと DSP アーキテクチャ記述からコード生成が行えるリターゲッタブル・コンパイラの研究がその重要性を増してきている.

リターゲッタブル・コンパイラの命令選択フェーズ (機械独立な中間コードに命令の割り当てを行う) では, プロセッサで実行可能な命令の集合についての情報が必要である. 一般にこの情報は各命令で実行可能な演算のパターンを木やグラフの形で表現した「命令パターン」の集合として与えられるが, 人手による記述が必要なものが多く, これがコンパイラのリターゲッティング作業を煩雑にしている. Dortmund 大学の RECORD [1, 2] では, プロセッサのハードウェア記述から命令セットの木パターンを自動抽出できる. しかし, そのためにはデータパスの詳細な記述が必要である.

これに対し本研究では, プロセッサの各命令の動作記述からコンパイラ用の命令パターンを抽出する方法を提案する. ASIP (Application Specific Instruction set Processor) 設計システム ASIP Meister [3] のプロセッサ動作記述から, プロセッサが実行可能な全ての命令パターン木を抽出する. 本稿ではさらに, 1 つの命令の記述がコンパイラに対する複数の命令パターンを含む場合に, これらを抽出する手法を提案する.

2 プロセッサ動作記述からの命令パターン木生成

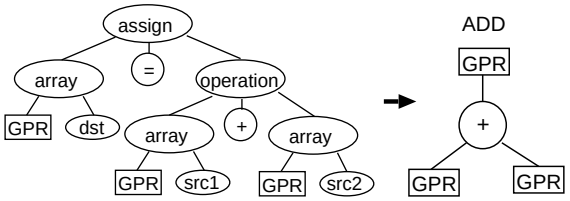
ASIP Meister の動作記述では, 各命令の動作を C-like な構文で記述する. 図 1 (a) は加算命令 ADD の動作記述例である. dst, src1, src2 は命令のオペランドフィールドを表す. 動作記述は, 構文解析後図 1 (b) のような解析木に変換される. 命令選択フェーズで必要となるのは図 1 (c) のような木構造の命令パターンである. これは構文木を走査して構築することができる.

図 2 (a) はプレディケート付命令の動作記述例である. この場合には, 図 2 (c) のようなプレディケート付加算命令 ("+*") の命令パターンを生成する必要がある. これは, 構文木中の if 文の then 節に代入文がある構造を認識することにより抽出できる.

†関西学院大学 理工学部 情報科学科
††大阪大学 大学院情報科学研究科 情報システム工学専攻

```
behavior ADD (dst, src1, src2) {
  GPR[dst] = GPR[src1] + GPR[src2];
}
```

(a) ADD の動作記述例

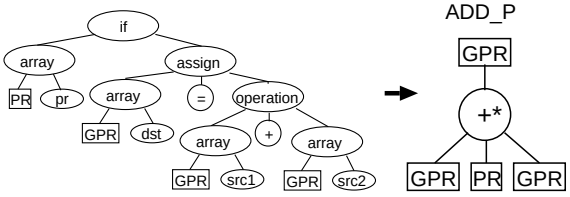


(b) 構文木 (c) 命令パターン

図 1: ADD の命令パターン生成

```
behavior ADD_P (pr, dst, src1, src2) {
  if (PR[pr]) {
    GPR[dst] = GPR[src1] + GPR[src2];
  }
}
```

(a) ADD の動作記述例



(b) 構文木 (c) 命令パターン

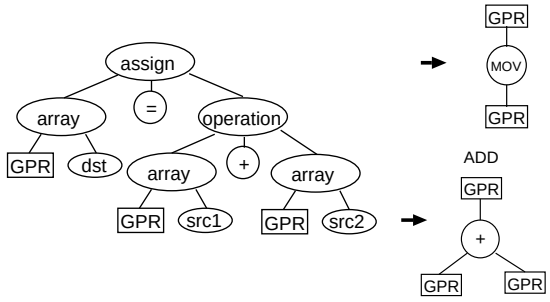
図 2: プレディケート付 ADD 命令のパターン生成

3 複数の命令パターン生成

3.1 設計上の命令とコンパイラに必要な命令の差

プロセッサ設計上は 1 命令であっても, コンパイラではそれを複数の命令として扱わなければならない場合がある.

例えば, 汎用レジスタ間のデータ転送命令は, 独立した命令として設計されるのではなく, 即値 0 やゼロレジスタを用いた加算命令で実現されることが多い. この場合, 単純に ADD 命令の記述から加算の命令パターンの抽出を行っただけでは, コンパイルに必要なデータ転送命令が欠如することになってしまう. そこで, 図 3 のように ADD 命令から通常の加算命令のパターンに加え, src2



(a) 構文木　　(b) 命令パターン

図 3: ADD からの複数命令パターンの生成

を 0 にして*データ転送を行う命令パターンを生成する必要がある.

また, プレディケート付命令の場合は, プレディケート修飾を持つものと持たないものを独立した命令として設計するのではなく, 図 4 のようにプレディケート修飾なしが特殊ケースであるような 1 つの命令として設計するのが一般的である. この場合には 1 つの命令記述から 2 通りの命令パターンを抽出する必要がある.

3.2 複数の命令パターン生成手法

このような複数命令パターンの抽出は, 命令から制御可能な部分を判別し, そこへ構文木が縮約されるような値の代入を行うことにより実現できる.

1. 解析木中の命令から制御可能な部分と代入可能な値の集合を抽出する.
2. 制御可能部分に値の代入を行うことにより構文木を縮約する.
3. 縮約した構文木を命令パターンに変換する.
4. パターンの重複を取り除く.

例えば, 図 4 の動作記述を解析した結果得られる, 図 5 の解析木において, pr, src1, src2 が命令から制御可能である. pr にはプレディケートの真偽で 0 と非 0 を設定できる. 0 と 非 0 をそれぞれ代入して構文木の縮約を行うと図 6 (a) のような ADD と図 6 (b) のようなプレディケート付の ADD 命令のパターンが得られる. さらに, 図 4 の記述から図 6 の (a) と (b) のそれぞれの src1, src2 にも 0 と非 0 が代入でき, 最終的に, MOV と プレディケート付 MOV を含む 4 つの命令パターンが抽出できる.

4 むすび

本研究では, 命令の動作記述からプロセッサで実行可能な命令パターンを抽出する手法と, 1 命令からコンパイラに必要な複数の命令パターンを生成する手法を提案した. 今後は, 提案手法の実装およびスケジューリング,

*どのレジスタがゼロレジスタの機能を持つかはプロセッサ記述中に宣言されている

```
if (pr == 0) {
  GPR[dst] = GPR[src1] + GPR[src2];
}
else {
  if (PR[pr]) {
    GPR[dst] = GPR[src1] + GPR[src2];
  }
}
```

図 4: 複数命令パターンを含む動作記述の例

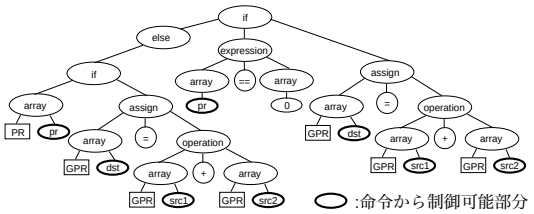
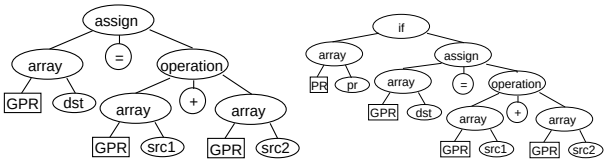


図 5: 複数命令パターンを含む命令の構文木構造



(a) pr==0 の時　　(b) pr!=0 の時

図 6: 1 命令からの複数命令の生成

バインディングなどのバックエンド部分の実装を行う予定である. 課題として, 命令の木パターン以外 (非巡回有向グラフ) への拡張やコンディションコード命令への対応などが挙げられる.

謝辞

本研究に際し, 御討論頂き, また, 種々の面々でお世話になりました大阪大学の武内良典先生をはじめ大阪大学今井研究室諸氏, および関西学院大学石浦研究室の関係諸氏に感謝致します.

参考文献

[1] R. Leupers and P. Marwedel: *Retargetable Compiler Technology for Embedded Systems*, Kluwer Academic, 2001.

[2] R. Leupers and P. Marwedel: "Instruction Selection for Embedded DSPs with Complex Instructions," in *Proceedings of the Conference on European Design Automation*, pp. 200–205, 1996.

[3] 今井正治, 武内良典: "コンフィギュラブル・プロセッサ開発システム ASIP Meister," 情報処理学会関西支部 支部大会 講演論文集, p. 97, 2001.