VLIWプロセサにおけるロードモジュール圧縮手法

5N－4

岩田 靖　安里 彰　木村 康則
新情報富士通研

1 はじめに

VLIW方式のプロセッサが実行する命令コードには，使用しない実行ユニットに対するnop命令が必ず含まれる．これによりコードサイズが膨大になるという問題がある．この問題を緩和するために，nop命令を可能な限り省いてロードモジュールを作成し，実行時に動的にnop命令を挿入することにより，正しいVLIW命令に展開する方法がある．この方法を命令圧縮と呼ぶ．

本稿では著者らが開発しているVLIW方式のジオメトリプロセッサであるProcyonで採用したロードモジュールの圧縮方式の定量的評価を行なうとともに，現在までに提案されている圧縮方式において，その圧縮率とハードウエア量の比較検討を行なう．

2 命令圧縮方式

2.1 Procyonの命令圧縮方式[1]

Procyonにおける1ワードのVLIW命令は並列に実行される4個の命令エレメントから構成される．各命令エレメントは30ビットの長さを持つ．Procyonの圧縮方式を長短命令方式と呼ぶ．Procyonでは，4ビットのフラグと30ビットの命令エレメント2個が連続した64ビットのブロック単位に命令が並んでいる．長短命令方式は，1ワードのVLIW命令に含まれている有効命令数に応じて長命令と短命令を使い分ける．更に，後続のVLIW命令の4個の命令エレメントがすべてnop命令(全nop命令)であることを表現するために暗黙nopフラグを導入している．Procyonにおける長命令，短命令，暗黙nopフラグを次に示す．

(1) 長命令 ブロック2個で形成され，それらに含まれる4個の命令エレメントが順にスロットA～Dで実行される．

(2) 短命令 ブロック1個で形成され，その中の2個の命令エレメントにnopが2個付加されて4スロットで実行される．nopが付加されるスロットは${}_6C_2 = 6$通りの短命令がある．

(3) 暗黙nop ブロックのフラグの1ビットをONにすることで，後続のVLIW命令が全nop命令であることを表現する．4ビットのフラグのうち3ビットが，長命令と6種の短命令の識別に用いられ，残りの1ビットが暗黙nopの表現に使用される．

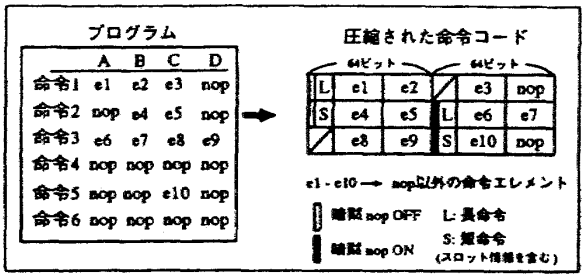

図1: Procyonの方式による命令圧縮の例

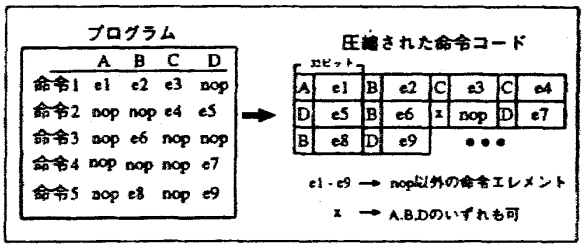

図2: pos方式による命令圧縮の例

本方式によって命令圧縮を行なった例を図1に示す．命令1,3が長命令，命令2,5が短命令，命令4,6が暗黙nopになる．

2.2 pos方式

Procyon方式に比べてアグレシブな方式として，pos方式を考案し，比較対象に選んだ．pos方式は，有効命令エレメントの先頭に，それが置かれるスロットを識別する2ビットのフラグを付加する方式である．

pos方式では4個以上連続したnopエレメントを除去するとそれに続く有効命令の展開位置が定まらなくなるので，このような場合は最低1個のnopは残さねばならない．それ以外のケースでは全てのnopを除去することが可能である．

pos方式によって命令圧縮を行なった例を図2に示す．圧縮後のコード中の唯一のnopは，圧縮前にnopが5個連続していたために除去されずに残されている．

2.3 pos+sw方式[2]

pos+sw方式は，命令エレメントが置かれるスロットを表すpos方式と同様の2ビットのフラグと，VLIW命令の切替えを表す1ビットのフラグを併せて用いる方式

Load Module Compaction Method for VLIW Processor
Yasushi Iwata, Akira Asato, Yasunori Kimura
RWCP Multi-Processor Computing Fujitsu Laboratory

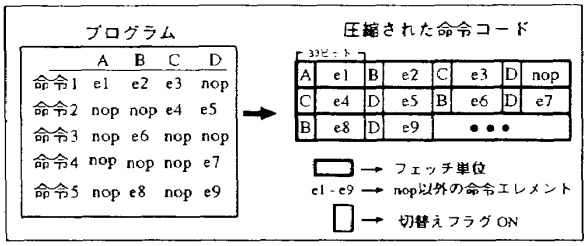

図 3: pos+sw 方式による命令圧縮の例

である．この方法は，文献 [2] を参考にして導出したものである．pos+sw 方式では，pos 方式で nop を置かざるを得なかった図 2のケースでも，切替えフラグによって nop を除去できる．ただし pos + sw 方式では，同一 VLIW 命令中の命令エレメントをフェッチ単位にまたがって置くことを禁止する．

図 2と同じプログラムを pos + sw 方式で圧縮した例を図 3に示す．図中の nop は，命令 2 に含まれる有効命令 e4,e5 をフェッチ単位にまたがって置けないので，それらを同じフェッチ単位に含めるために置かれている．

3 評価

本節では，上に示した各圧縮方式において，Procyon で使用する実際のプログラムの圧縮率と，圧縮されたロードモジュールを実行時に展開する回路規模を評価する．

3.1 圧縮率

各圧縮方式の圧縮率を評価するにあたり，本稿では，式 (1) で圧縮率を定義する．

$$\text{圧縮率} = \frac{\text{圧縮後の命令エレメント数}}{\text{圧縮前の命令エレメント数}} \qquad (1)$$

評価対象のベンチマークプログラムは，Procyon で実行する実プログラムのうち，アセンブリ言語で記述したプログラム群 (10 本) と C 言語で記述したプログラム群 (10 本) を用いた．表 1において，その結果を示す．

表 1 によると，平均の圧縮率がもっとも良いものは pos 方式および pos+sw 方式であり，Procyon 方式はそれらよりも 5% 程圧縮率が低いことがわかる．

3.2 ハードウエア量

各圧縮方式で圧縮されたロードモジュールを実行するためには，プロセッサ内で動的に圧縮されたロードモジュールを展開して実行しなければならない．これは，プロセッサに実現された展開回路で行なう．この展開回路部分の複雑さを評価するために，ハードウエア記述言語 VHDL を用いて各圧縮方式を展開する回路を記述・論理合成を行ない，さらに実際にモジュールレイアウトを行なって，展開回路の回路規模を評価した．表中の BC 数とはベーシックセル (2 入力 NAND ゲート) を意味する．

表 1: 圧縮率と展開回路のハードウエア規模

圧縮率	Procyon	pos	pos+sw
アセンブラ	50%	47%	52%
C 言語	42%	34%	30%
平均	46%	41%	41%
BC 数	936	1909	1890
ネット数	537	973	952
面積比	1.00	5.61	4.88

3.3 圧縮率とハードウエア量のトレードオフの評価

圧縮率とハードウエア量のトレードオフの評価を行なうために，圧縮によって節約されたコードサイズと，それを展開するために必要なハードウエア量の比の値を，各圧縮方式のトレードオフを示す値として導入する．この比の値はすなわち，展開回路の単位面積あたりのコード削減量の比率を表すことになり，この値が大きいほど，効率が良いことになる．

表 2から，単位ハードウエアあたりの効率は 4 つの方式の中で Procyon 方式がもっとも良く，pos 方式や pos + sw 方式の 4.5 倍から 5.1 倍の値になっている．

表 2: 展開回路の単位面積当たりのコード削減量比

Procyon	pos	pos+sw
1.00	0.194	0.223

4 まとめ

本稿では VLIW 方式のロードモジュール圧縮方式として，Procyon に実現した長短命令方式を，その圧縮率と展開回路のハードウエア量の観点から定量的に評価した．さらに，圧縮率の観点からよりアグレッシブな方法として，pos 方式と pos + sw 方式を規定し，同様に評価し比較を行なった．その結果，Procyon で採用した長短命令方式は，展開回路の単位面積あたりの命令コード削減量が最も大きく，長短命令方式が効率のよい圧縮方式であることを明らかにした．

参考文献 [1] 岩田ほか: ジオメトリプロセサ Procyon - コンパクション方式 - 第 55 回情報処理学会全国大会講演集 (1), pp.46-47 (Sep.1997). [2] Ray Simar,「米 TI の DSP,MPU に先駆けて最大 8 命令の VLIW を採用」日経エレクトロニクス，97 年 6 月 16 日号，no. 691, p135.