2C－2

ＶＬＩＷ計算機における条件実行アーキテクチャの評価とコンパイラの役割

境　隆二、竹内　陽一朗、石川　禎
（株）東芝　情報・通信システム技術研究所

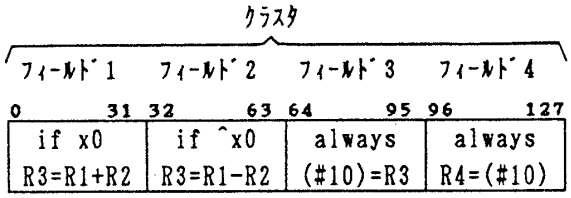
1．はじめに

ＶＬＩＷ方式の計算機ではコンパイラが命令を並列にスケジュールする。このとき、条件分岐命令が障害となってスケジュール対象の命令が限られた範囲に制約され、十分な並列度が思うように得られないことがある。

ＶＬ２０００シリーズでは、条件実行アーキテクチャを採用し、この問題を克服している。本稿では、この条件実行アーキテクチャの評価結果と、このアーキテクチャを有効に使うためのコンパイラの最適化技法について述べる。

2．アーキテクチャ

ＶＬ２０００シリーズでは、１ワード（３２ビット）１命令のＲＩＳＣ命令セットを４つのフィールドに指定して１つのＶＬＩＷ命令を構成している（図１）。また、命令取り出し、デコード＋レジスタ読みだし、演算、結果格納、の４段パイプライン構成で、分岐命令の遅延サイクルは１である。



ﾌｨｰﾙﾄﾞ：１語長の命令指定領域
ｸﾗｽﾀ：４ﾌｨｰﾙﾄﾞからなる命令語の集まり

図１　ＶＬＩＷ命令の構成

さらに、ＶＬＩＷ命令を構成する各単位命令毎に、実行、非実行を決定するフラグを指定することが出来る（図１）。フラグは複数あり、比較命令などで実行時に値が設定される。フラグにより実行条件が指定された命令を条件実行命令と呼び、このようなアーキテクチャを条件実行アーキテクチャと呼ぶ。

Evaluation of conditional execution on VLIW
RyujiSAKAI, YoichiroTAKEUCHI, TadashiISHIKAWA
TOSHIBA Corporation

3．条件実行命令の適用

条件実行アーキテクチャの効果を発揮するには、コンパイラが条件実行命令を有効に使って最適化を行う必要がある。以下にソフトウェアブースティングと条件実行命令展開について説明する。

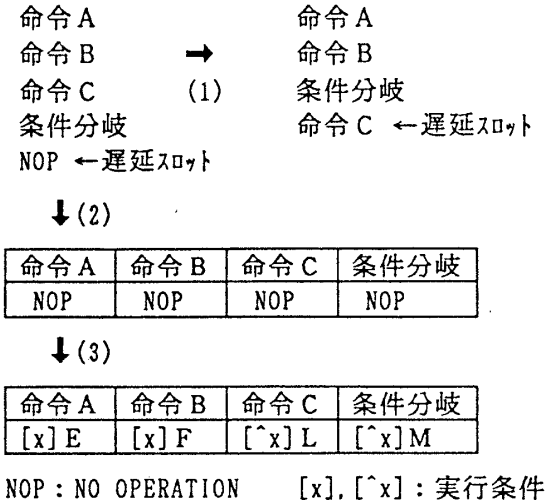
3.1 ソフトウェアブースティング

遅延スロット最適化とは、コンパイラが分岐命令の遅延スロットに命令を配置することで、命令の実行効率を向上させようというものである（図２の(1)）。

ＶＬＩＷ方式の場合、遅延スロットに配置できる命令が、より以前の命令の空きフィールドに並列に並べられる可能性が高く、遅延スロットが有効に使えない（図２の(2)）。

ソフトウェアブースティングとは、上記のような場合でも、遅延スロットや空きフィールドが有効に使えるように、分岐先や次のブロックから命令をもってきて、配置する最適化である。条件分岐の場合、もってきた命令を条件実行命令にすることで、この最適化が可能となる（図２の(3)）。

ソフトウェアブースティングは、遅延スロットのみではなく、可能な限り命令を押し上げることで、より効率のよいオブジェクトを生成する。



NOP：NO OPERATION　[x]，[^x]：実行条件

図２　ｿﾌﾄｳｪｱﾌﾞｰｽﾃｨﾝｸﾞ

3．2 条件実行命令展開

if-then-elseなどで、条件成立、非成立のときのそれぞれの処理に対して、対応する命令を条件実行命令とすることで、条件分岐命令を使わない命令列に展開する最適化を条件実行命令展開という。この最適化を行うことで、命令のスケジュール範囲が広くなり、並列度が向上する（図３）。

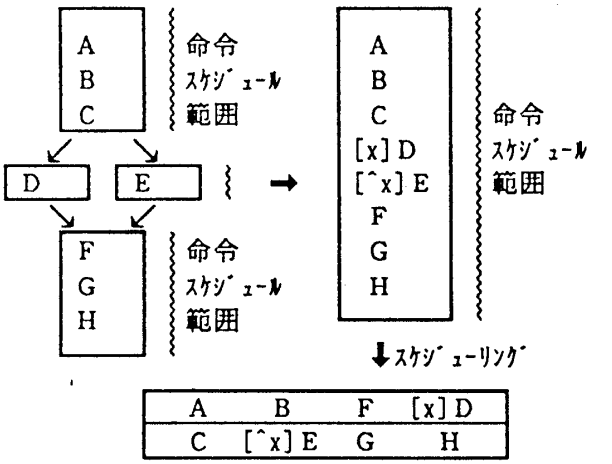


図３ 条件実行命令展開

4．評価方法

コンパイラの最適化処理で、条件実行命令を使って最適化したオブジェクトと、条件実行命令を使わないで最適化したオブジェクトの実行性能を比較した。

5．評価結果

	Livermore		Specint
	kernel16	kernel24	compress
性能向上率	12.7%	29.0%	13.2%

	ｺﾝﾊﾟｲﾗ最適化		その他のﾍﾞﾝﾁﾏｰｸ	
	スカラ	並列化	ntt	dhryston
性能向上率	15.3%	11.4%	15.4%	7.2%

リバモアカーネルのカーネル２４は、条件実行展開最適化の効果が発揮される典型的な例である（図４）。カーネル１６（モンテカルロ法）は、条件分岐が多数あるので、ソフトウェアブースティングの効果が大きい。

```
      DO 24  L= 1,Loop
             m= 1            条件実行命令になる
      DO 24  k= 2,n                ↓
           IF( X(k).LT.X(m)) m= k
   24 CONTINUE
```

図４ カーネル２４

実用性を見るために、Specベンチマークのcompressや、コンパイラの最適化処理をセルフコンパイルして評価した。compressやコンパイラの最適化処理はほとんどが整数演算で、条件分岐も比較的多いため、十数％の効果が得られた。

dhrystoneでの効果が小さいのは、他の最適化により分岐命令の頻度が減少したためと考えられる。

6．まとめ

ＶＬ２０００シリーズでは、分岐の遅延サイクルが１であるため、分岐のオーバーヘッドは比較的小さい。ソフトウェアブースティングと条件実行命令展開で、十数パーセントも性能が向上したということは、このアーキテクチャの有効性を十分に実証している。また、より分岐の遅延サイクルの大きいマシンに対しては、より大きな効果が期待できる。

7．今後の課題

ソフトウェアブースティングは、命令を一度スケジュールしてみないと、どの命令の組み合わせを条件実行命令にしてブースティングすれば最適なのか判断が難しい。現在の命令スケジューラは、基本ブロックのサイクル数が出来るだけ短くなるようにスケジュールし、そのあとでブースティング最適化を行っているが、ブースティング効果は命令をスケジュールしたときのブロックの境界部分の命令の並び具合で大きく左右される。

条件実行アーキテクチャをより有効に使うためには、上記のような問題点を解決し、さらに進んだコンパイラの最適化処理を開発する必要がある。

参考文献

石川禎，竹内陽一朗
「ＶＬＩＷアーキテクチャの実現」
情報処理学会第４６回全国大会（1993）