計算機アーキテクチャ　104−11
システムソフトウェアと
オペレーティング・システム　62−11
（１９９４．　１．　２８）

ソフトウェアパイプラインによる条件分岐の際のハザードの影響の削減

安倍 正人　岡部 公起　根元 義章

東北大学大型計算機センター
〒980 仙台市青葉区片平 2-1-1
東北大学大型計算機センター
Tel.022-221-3380 Fax.022-263-9202

従来の計算機では、ほとんどの場合１つの命令を複数のステップに分け、それらをハードウェアで自動的に連続してパイプライン処理することにより高速化を図っている。しかし、この方式では、条件分岐の際のハザードの影響が避けられず、条件分岐の多いアプリケーションでは実質的に高速化が図れない場合が多い。そこで、本論文では、ハードウェアによるパイプライン計算機と同様に、１つの命令を複数のステップからなる部分に分けるものの、それらをハードウェアによってではなく、マイクロ命令として直接ソフトウェアで実行することにより、条件分岐におけるハザードの影響を減少させる超細粒度計算機を提案する。そして、幾つかのベンチマークプログラムにより、提案方式の有効性を述べる。

Decrease of the Effect of Harzard Due to the Conditional Branch by Use of Software Pipeline

Masato Abe　Kouki Okabe　Yoshiaki Nemoto

Computer Center, Tohoku University
980 Computer Center, Tohoku University, Sendai, Japan
Tel.022-221-3380 Fax.022-263-9202

In the conventional pipeline architexture, an instruction is divided into a series of many segments, and they are executed continuously by hardware to realize a high speed computer. However, the architecture does not work well in case of conditional branch. Therefore, we propose a new architecture, in which one conventional instruction is divided into several micro-instructions, and they are executed by software not but the hardware. It is found from some C programs that the proposed architecture offers us a twice faster computer in case that a program has many condition branches.

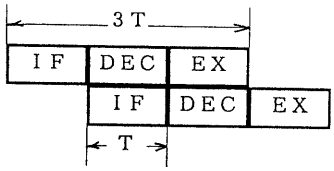
1 まえがき

最近、計算速度の高速化を図るための手法として、複数の命令を1ステップで実行できるスーパースカラ計算機や、1つの命令を従来の計算機と比べて多くの段数のパイプラインで実行するスーパーパイプライン計算機が実用化されてきている。〔1〕しかし、これらの方式では、いずれもハードウェアでパイプライン処理するため、条件分岐の際のハザードの影響が避けられず、〔2〕アプリケーションによっては実質的に高速化が図れない場合が多い。そこで、本論文では、ハードウェアによるパイプライン処理と同様に1つの命令を複数のステップに分けるものの、それらをハードウェアでパイプライン処理するのではなく、マイクロ命令として直接ソフトウェアで制御することにより、条件分岐の際のハザードの影響を減少させる方式を提案する。この方式は、従来の命令より細かい粒度の命令を用いるということなので、本論文では超細粒度計算機と呼ぶことにする。

本論文では、更に、提案方式におけるハザードの影響の減少とパイプラインの段数の関係を明らかにするとともに、我々の研究室で手書き文字の認識〔3〕のために用いているプログラムのうち、もっとも時間がかかっている部分をベンチマークプログラムとして用い、提案方式の有効性を確かめる。

2 ハードウェアによるパイプライン処理計算機におけるハザードの影響

パイプライン計算機とは、1つの命令を実行するのに、例えば図1に示すように、命令のフェッチ（ＩＦ）、デコード（ＤＥＣ），実行（ＥＸ）という3つの部分（段数）に分け、それぞれを並列に実行することにより、多数の命令を処理する場合に実質的に命令の実行時間を（1／パイプラインの段数）にする計算機である。従来は、この段数が3から5が主流であったが、最近では、パイプラインの段数を更に多くして高速化を図るスーパーパイプライン計算機も実用化されてきている。



IF : Instruction Fetch
DEC : Decode
EX : Execution

図1: パイプライン計算機における命令実行の流れ

しかし、このパイプライン処理方式は、条件分岐の際のハザードが問題となる。例えば、図2のCのプログラムのアセンブラ出力コードを図3に示す。ただし、用いた計算機はＨＰ９０００モデル７３０で、コンパイルオプションは（－Ｓ　－Ｏ）である。この計算機は、1命令を1ステップで実行可能なので、条件分岐に起因するハザードの影響や後で述べるデータが揃うまでの時間待ちに起因するハザードの影響がなければ基本ループを1回処理するのに、条件が成立したときに7ステップ、成立しないときに6ステップかかるはずである。しかし、このプログラムを実際に実行して時間を計測した結果、条件が成立するしないにかからず基本ループ1回を処理するのに１０ステップかかった。この場合の条件分岐に関するハザードの影響とは、ハードウェアによりパイプライン処理する計算機において、条件判断が終了するまで次に実行すべき命令のフェッチが行えないために生じる。最近では、予測分岐用のハードウェア機構を持つ計算機もあるが、条件の成立、不成立の確率が五分五分であるようなアプリケーションでは必ずしも有効でないという問題がある。

```
double a[1000];
double sum;
main()
{
   int i,j,k;
   for(i=0;i<1000;i++) {
      if(a[i]<0.0)
         sum -= a[i];
      else
         sum += a[i];
   }
}
```

図 2: 条件分岐の際のハザードの影響調査のためのCプログラム

3 マイクロ命令によるソフトウェアパイプライン処理

本論文で提案するソフトウェアパイプライン処理計算機を従来のハードウェアパイプライン処理計算機と図2に示したプログラムの基本ループの部分を用いて比較する。ただし、ハードウェアによるパイプライン計算機のパイプラインの段数は以下に示すように、5と仮定する。

(イ) メモリからのデータ入力 (a〔i〕のロード)
- (1) 命令のフェッチ
- (2) 命令のデコード
- (3) アドレス"&a〔i〕"の計算
- (4) 計算されたアドレスのメモリユニットへの転送
- (5) a〔i〕のレジスタへのロード

(ロ) 四則演算
- (1) 命令のフェッチ
- (2) 命令のデコード
- (3) レジスタから演算器へのデータ入力
- (4) 演算
- (5) 演算結果のレジスタへの出力

```
$00000003
   COPY        %r24,%r25
   LDI         -1000,%r29
$00000006
   FLDDS,MA    8(0,%r25),%fr5
   FCMP,DBL,<  %fr5,%fr0
   FTEST
   B,N         $00000007
   FSUB,DBL    %fr4,%fr5,%fr4
   OR,TR       %r0,%r0,%r0
$00000007
   FADD,DBL    %fr4,%fr5,%fr4
$00000008
$00000004
   ADDIB,<,N   1,%r29,$00000006+4
   FLDDS,MA    8(0,%r25),%fr5
$00000005
   ADD         %r26,%r31,%r26
$00000001
   LDO         1(%r31),%r31
   COMB,<,N    %r31,%r23,$00000003+4
   COPY        %r24,%r25
```

図 3: Cプログラムのアセンブラ出力コード

(ハ) 比較
- (1) 命令のフェッチ
- (2) 命令のデコード
- (3) レジスタから演算器へのデータ入力
- (4) 比較演算
- (5) 演算結果によるフラッグ設定

(ニ) レジスタからメモリへの出力
- (1) 命令のフェッチ
- (2) 命令のデコード
- (3) 出力アドレス"&sum"の計算
- (4) 計算されたアドレスと出力データのメモリユニットへの転送
- (5) 書き込み

（ホ）条件分岐

（1）命令のフェッチ

（2）命令のデコード

（3）条件フラッグのチェック

（4）分岐アドレスの計算

（5）分岐アドレスのプログラムカウンタ（ＰＣ）への書き込み

従来のハードウェアによるパイプライン処理計算機で、（イ）に示したメモリからレジスタへのデータの入力を実行した場合の処理の流れを図6に示す。また、図2のプログラムの基本ループを実行した場合の処理の流れを図4に示す。図4において、Ｗｄの部分はデータ待ちに起因するハザードで、例えば比較命令中のＷｄはデータａ〔ｉ〕のロードが完了するまで、ａ〔ｉ〕と値0との比較の実行が待たされるために生じる。一方、Ｗｊは条件ＪＵＭＰに起因するハザードで、条件フラッグのチェックが終了するまで、次に実行すべき命令のフェッチが行えないために生じる。

上記の2つのハザードの原因のうち、Ｗｄの削減は困難であるが、Ｗｊの削減は本論文で提案するように、パイプライン処理をソフトウェアで行うことにより削減できる。すなわち、

（1）最近一般的に用いられるようになってきたスーパースカラ方式により、複数の命令を1ステップで同時に実行できるアーキテクチャを取ることにより、条件分岐のあとで実行される”ａ〔ｉ〕＋ｓｕｍ”の計算を条件分岐の前に他の処理と並列に行う。

（2）演算結果のメモリへの出力の処理は条件分岐の前に開始しておき、最後の書き込み動作のみを条件分岐のあとに行う。

上述の（1）の手法はバーコレーション〔1〕と呼ばれ，最適化の手法の1つとして良く知られている方法である。しかし、（2）の手法はハードウェアによるパイプライン処理計算機では不可能であり、次に示すように、パイプラインの各ステップをソフトウェアで処理するという本論文で提案する方式でしか実現できない。

我々が提案するマイクロ命令を用いたソフトウェアパイプライン処理計算機では、例えば（イ）に示した”ａ〔ｉ〕のロード”命令の場合、（イ）の（3）から（5）がそれぞれ別々のマイクロ命令であり、図7に示す順番で3つの命令として実行される。

なお、この方式では、各マイクロ命令を直接別々にプログラムで動作させる必要があるため、必然的に従来のハードウェアパイプライン処理計算機に比べて、プログラムメモリの幅が非常に大きくなる。しかし、このことは逆に、プログラムメモリの幅に関して従来の計算機とのオブジェクトの互換性維持というしがらみを断ち切ることができる自由が得られるとも考えられる。この場合には、すべてのマイクロ命令について（2）のデコードの部分を省略すること、すなわちデコード済の命令をプログラムメモリに格納しておくことが出来、パイプラインの段数を2とすることができる。DEFがそれを表している。

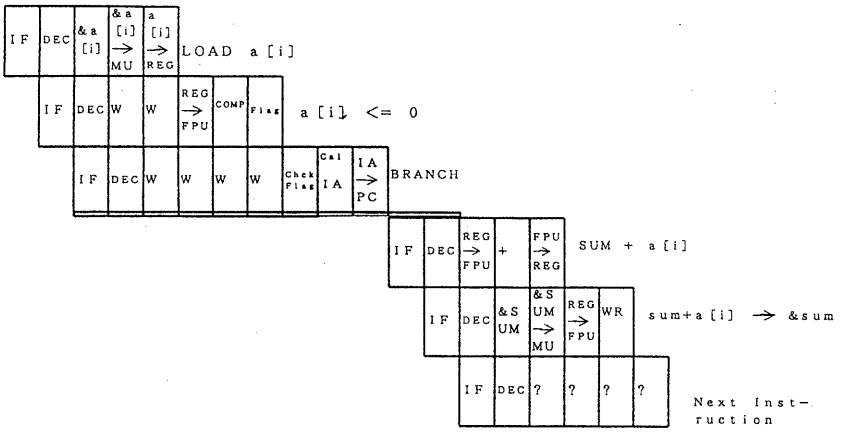


図 4: 条件分岐のあるプログラムに対するハードウェアパイプライン計算機の処理の流れ

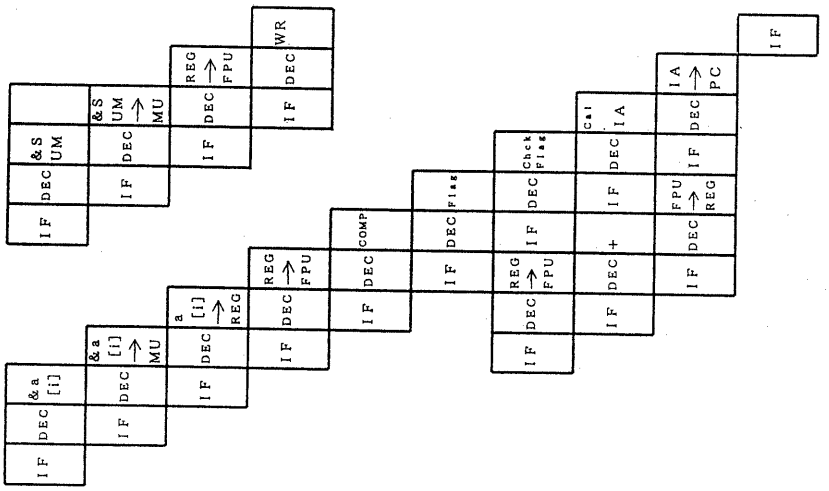
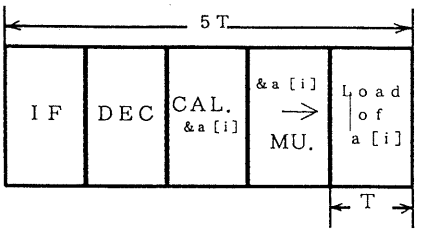
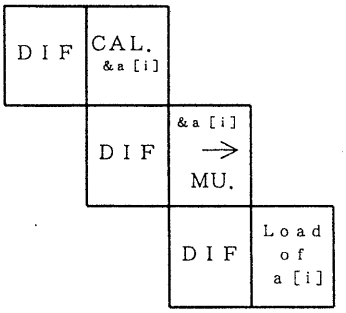


図 5: 条件分岐のあるプログラムに対するソフトウェアパイプライン計算機の処理の流れ



IF : Instruction Fetch
DEC : Decode
CAL &a[i] : Calculation of address a[i]
&a[i] -> MU : Transfer address a[i] to Memory Unit
Load of a[i]: Load of a[i]

図 6: ハードウェアによるパイプライン処理計算機で、メモリからレジスタへのデータの入力を実行した場合の処理の流れ



DIF : Decoded Instruction Fetch
CAL &a[i] : Calculation of address a[i]
&a[i] -> MU : Transfer address a[i] to Memory Unit
Load of a[i]: Load of a[i]

図 7: ソフトウェアによるパイプライン処理計算機で、メモリからレジスタへのデータの入力を実行した場合の処理の流れ

図 6, 7を見ただけでは、わずか 1 ステップ減少しただけで、7の方がプログラムメモリの幅が多く必要なだけコストパフォーマンスが悪いように思われる。しかし、図 2のプログラムにおいて、基本ループ全体を見たときには、デコード部を入れた場合 (図 7 でも、従来のハードウェアパイプライン処理計算機の結果 (図 6) に比べて 3 ステップ実行時間が短縮されている。一般的に、複数命令の同時実行が可能な場合には、本論文で提案する手法を用いることにより得られる条件分岐 1 回あたりの実行ステップの減少数ΔNは次のように表される。

$$\Delta N = Nh - Nm \quad (1)$$

ただし、Nhはハードウェアパイプライン処理計算機のパイプラインの段数、Nmは提案したソフトウェアパイプライン計算機におけるマイクロ命令のパイプラインの段数である。

4 ベンチマークプログラムによる性能評価

我々の研究室では、手書き文字の認識に関する研究も行っている。このためのプログラムのなかで特に時間がかかっている部分は、

(1) 入力文字と辞書中の文字とのマッチングと類似度 (距離) の計算
(2) 3000種の文字のなかから、距離が小さい (類似度が高い) 候補30 の選択
(3) 30の候補の類似度の高い順での並べ替え

の3つの部分であるが、その基本ループは全く同一で、いずれも比較演算すなわち条件分岐が必要であり、しかも条件が成立するかしないかが予測できない。この基本ループについて、提案するソフトウェアによるパイプライン計算機のパイプラインの段数を 2 とし、ハードウェアによるパイプライン計算機のパイプラインの段数を変化させ

て、基本ループを１回計算する時間を机上で計算し、次式で評価した。

$$C = Ts/Th \quad (2)$$

ただし、Ｔｓ：提案したソフトウェアによるパイプライン計算機での実行時間、Ｔｈ：ハードウェアによるパイプライン計算機での実行時間である。

ここで、ハードウェアパイプラインの段数Ｎｓが３から７の場合において、各段の処理時間はＴ／Ｎｓ（Ｔ：命令のフェッチからその命令の実行完了までの時間）と仮定し、また処理の内容は以下のように仮定した。

Ｎｓ＝３のときは、ＩＦ，ＤＥＣ，ＥＸの３つに分かれ、ＥＸで命令が実行される。

Ｎｓ＝４のときは、ＩＦ，ＤＥＣ，ＥＸ，ＭＡの４つに分かれ、ＥＸでメモリアドレスの計算、四則演算等が行われ、ＭＡではメモリからレジスタへのデータのロード、演算結果のレジスタへの退避、演算結果としてのＦＬＡＧの設定、あるいはレジスタからメモリへのデータの書き込みが実行される。

Ｎｓ＝５のときは、ＩＦ，ＤＥＣ，ＥＸ，ＭＡ、ＷＢの５つに分かれ、ＥＸでメモリアドレスの計算、四則演算等が行われ、ＭＡではメモリからレジスタへのデータのロード、演算結果のレジスタへの退避、演算結果としてのＦＬＡＧの設定、あるいはレジスタからキャッシュメモリへのデータの書き込みが実行される。またＷＢでは、主記憶への書き戻しが実行される。

Ｎｓ＝６のときは、ＩＦ，ＤＥＣ，ＥＸ１，ＥＸ２，ＭＡ、ＷＢの６つに分かれ、ＥＸ１でメモリアドレスの計算、ＥＸ２で四則演算等が行われ、ＭＡではメモリからレジスタへのデータのロード、演算結果のレジスタへの退避、演算結果としてのＦＬＡＧの設定、あるいはレジスタからキャッシュメモリへのデータの書き込みが実行される。またＷＢでは、主記憶への書き戻しが実行される。

Ｎｓ＝７のときは、ＩＦ，ＤＥＣ，ＥＸ１、ＥＸ２，ＥＸ３,，ＭＡ、ＷＢの７つに分かれ、ＥＸ１でレジスタからのデータのロード、ＥＸ２でメモリアドレスの計算、ＥＸ３で四則演算等が行われ、ＭＡではメモリからレジスタへのデータのロード、演算結果のレジスタへの退避、演算結果としてのＦＬＡＧの設定、あるいはレジスタからキャッシュメモリへのデータの書き込みが実行される。またＷＢでは、主記憶への書き戻しが実行される。

前述した条件分岐を含む文字認識プログラムの基本ループ１回を処理するのに要する時間を図８に示す。ただし、パイプラインのピッチはＴ／Ｎｓとした。図８から分かるように、ハードウェアにより自動的にパイプライン処理する従来の方式では、条件分岐に起因するハザードのために、ソフトウェアによりパイプライン処理する方式と比較して２倍程度時間が多く必要であり、また、パイプラインの段数が変化してもその傾向は変わらないことが分かる。以上の結果から、ソフトウェアによるパイプライン処理により、条件分岐に起因するハザードの影響を大幅に軽減できることが明らかとなった。

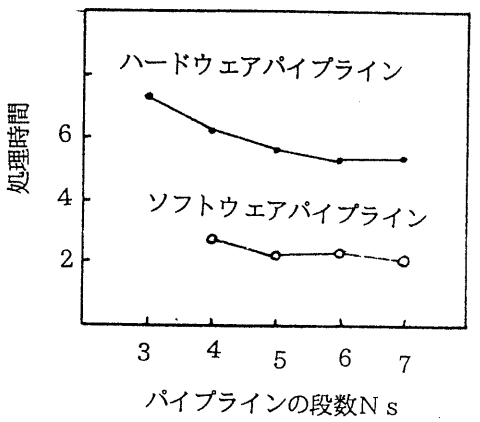


図 8: ：パイプラインの段数Ｎｓを変化させて計算した文字認識プログラムの基本ループ１回を処理するのに要する時間

5　結論

計算速度の高速化を図るためにパイプライン技術が広く用いられており、今後もパイプラインの段数の増加が予想される。しかし、現状ではいくらパイプラインの段数を増加させても条件分岐に起因するハザードのために期待したほどの効果が得られないという問題がある。このハザードの影響を減らすために、本論文ではソフトウェアによるパイプライン処理計算機を提案した。これは、従来のパイプライン計算機で1つの命令を複数のステップに分割し、各ステップをハードウェアで自動的に連続して処理していたと同じことを各ステップをマイクロ命令としてソフトウェアで実行すること、および条件分岐後に実行していたアドレス計算や四則演算等のステップを条件分岐前に実行してしまい、最後のレジスタあるいはメモリへの書き込みのステップだけを条件分岐後に実行することにより、高速化を図る手法である。

本論文ではさらに、我々の研究室で手書き文字の認識に用いているプログラムをベンチマークプログラムとして用い、従来のハードウェアによるパイプライン処理計算機と提案したソフトウェアによるパイプライン処理計算機との比較を行った。その結果、提案した方式により演算速度が2倍程度高速化されることが分かった。