バッファ・メモリを有するパイプライン制御計算機の性能評価について†

金沢正憲†† 北川一†† 萩原宏†††

計算機におけるプログラムの実行速度の高速化は，素子の動作の速さのほかに，パイプライン制御方式とバッファ・メモリ方式によりもたらされている．

ここでは，パイプライン制御計算機の命令処理時間が，バッファ・メモリのミス率，ストア・バッファ・レジスタの数，および，主記憶の構成により受ける影響をシミュレーション法によって検討した．シミュレーション・プログラムへの入力データは，実際の計算機システムから測定によって得られた値を用いた．バッファ・メモリのミス率に関しては，メモリ・アクセスの種別に従って3つに分類して定義し，ハードウェア・モニタによって測定されたデータを用いた．命令処理の分岐命令等による遅れに対しては「ダミー命令」なる概念を導入した．これらに基づいて命令列（入力データ）を作成することにより，シミュレータは，バッファ・メモリを単に1つのリソースとして取り扱うことができ，かつ，逐次的に命令処理をシミュレートするだけの簡単なモデルとすることができた．シミュレーション・ランの高速化を図ることもできた．

シミュレーションの結果，メモリの構成による命令処理時間の影響が明らかになった．特に，メモリの構成，速度ならびにミス率にほとんど関係なく，ストア・バッファ・レジスタの必要な個数を求めることができた．

1. はじめに

計算機のプログラム実行速度を高速化するために，パイプライン制御方式とバッファ・メモリ（以後 BM と略す）が有効であることは，最近の高速計算機をみれば明らかである．このような高速計算機の命令処理の効率については今までにいくつか解析されている．文献1),2)ではメモリ・アクセスのうち，命令の読み出しについては十分考慮されているが，データの読み出し，および，書き込みについてはあまり検討されていない．さらに，メモリでのアクセスの競合については全く取り扱われていない．実際にカーネル・プログラムを実行させて測定する方法は[3]，システムの構成等に関して検討できる範囲が自ら限定される．

しかし，命令処理の効率は，パイプライン制御方式，BM，主記憶等の相互作用によって定まるものであり，それらの動作を総合的に解析する必要がある．

ここでは，パイプライン制御の中で最も一般的な命令の先廻り制御による命令の流れと，BM および主記憶の動作をモデル化した．実際のシステムとして FACOM 230-75（以後モデル 75 と呼ぶ）を取り上げ，プログラム実行速度と，BM，ストア・バッファ・レジスタ*（以後 SBR と略す），および，主記憶との関係をシミュレーションにより評価した．このシミュレーションでは，BM のミス率はハードウェア・モニタによる測定データを用い，パイプライン制御下での命令処理と各レベルの記憶装置におけるアクセスの競合などの時間的要素に係る動作を主としてシミュレーション・プログラムで実現した．その結果，シミュレーション・モデルが簡単化されるとともに，シミュレーション・ランが高速化された．

2. ハードウェア・モニタによる測定

CPU や BM の動作を正確に測定するには，ソフトウェア・モニタでは不可能であり，ハードウェア・モニタでも CPU のマシン・サイクルの速さで動作を追従できるものでなければならない．モデル 75 には，これを満足するハードウェア・モニタ[5]（以後 CPA と呼ぶ）が用意されている．

モデル75は 2CPU 密結合構成が可能で，CPU 間で直接的に情報を交換するためのダイレクト・インタフェースと呼ばれる機能がある．CPA はこの機能を用いて 1CPU 構成のモデル 75 と接続されるので，ただ単にモデル75の動作状況を測定するだけでなく，モデル 75 で実行中のプログラム（たとえば，ソフトウェア・モニタ）がダイレクト・インタフェース用命

† Performance Evaluation of a Pipelined Processor with Buffer Memory by MASANORI KANAZAWA, HAJIME KITAGAWA (Data Processing Center, Kyoto University) and HIROSHI HAGIWARA (Department of Information Science, Kyoto University).

†† 京都大学大型計算機センター

††† 京都大学工学部情報工学教室

* ストア命令における格納動作のおいてきぼり制御のために用意された記憶レジスタ．

令を発して，CPA の測定データを読み出すことができる．

測定の対象としたシステムは，京都大学大型計算機センターのモデル 75 (1 CPU) バッチ処理システム*で，ジョブは FORTRAN による科学技術計算が 90% 以上を占めている．また，測定は通常のサービス時に行い，測定データの偏りを少なくするため，期間を空けて測定した．このうち，シミュレーションで用いる測定データを以下に示す．

2.1 ミス率の測定

CPU からメモリ**へのアクセスは，命令の読み出し，データの読み出し，データの書き込みの 3 つに分類される．したがって，ミス率に関しても，この 3 つのアクセスに対応させて考える必要があり，次のような 3 種類のミス率を定義する．

$$\left.\begin{aligned} P_X &= f_X/F_X, \\ P_R &= f_R/F_R, \\ P_W &= f_W/F_W, \end{aligned}\right\} \qquad (1)$$

ただし，

F_X: 命令の読み出し回数，

f_X: 命令の読み出しにおいて BM 上になかった回数，

F_R: データの読み出し回数，

f_R: データの読み出しにおいて BM 上になかった回数，

F_W: データの書き込み回数，

f_W: データの書き込みにおいて BM 上に対応するブロックがなかった回数．

CPA によって，10 秒間の上記の各回数を測定し，式(1)の各値を求めるが，連続的に測定して求めた各々の確率分布を**図 1**，**図 2** に示す***．なお，平均値は，$P_X=0.0457$, $P_R=0.0434$, $P_W=0.135$ であった．

図 1，図 2 より，P_X と P_R はほとんど同じ分布である．一方，P_W はスケールが異なり約 3 倍の値になっているが，ほぼ同じ形状である．そこで，P_X, P_R, P_W のそれぞれの相関係数を求めると，P_X と P_R では 0.89，P_R と P_W では 0.82，P_W と P_X では 0.77 であった．すなわち，各値は共に大きな，または，小さな値になることが多いことになる．P_X と P_R との相関が高いことは BM へロードされる扱いが同一であるため当然であると考えられるが，ストア・スルー方式のためデータの書き込みだけは扱いが全く異なるにもかかわらず，P_R または P_X と P_W との相関も比較的高いことが判った．

そのほかの測定については，**5.1** で測定値のみ示す．

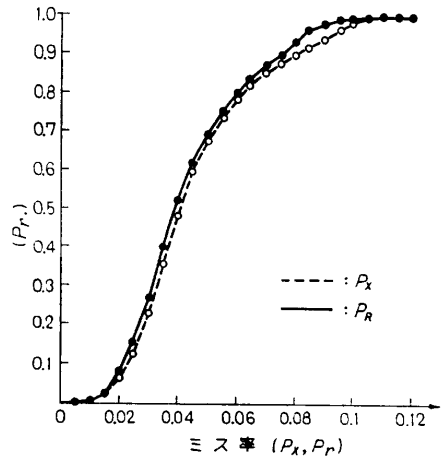


図 1 P_X, P_R の確率分布
Fig. 1 Probability distributions of P_X and P_R.

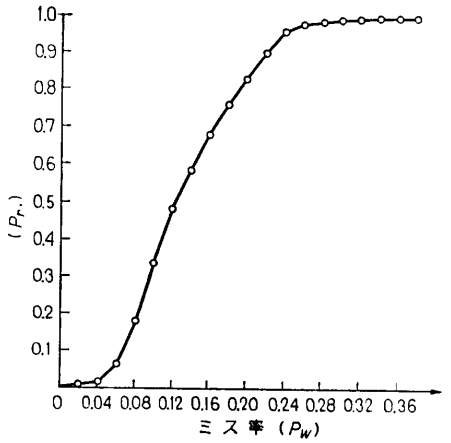


図 2 P_W の確率分布
Fig. 2 Probability distribution of P_W.

3. シミュレーション・モデル

3.1 構　成

CPU, BM, SBR, 主記憶間の情報の流れから見た構成は，**図 3** のようにモデル化できる．図中には各部分間での情報の流れる方向とアクセス単位の 2 語の転送所要時間を示す．なお，τ は単位時間とする．

(1) BM の構成

---

* モデル 75 の BM は，容量 4 K 語，ブロックの大きさ 8 語，構成は 4 ロー×128 カラム，セット・アソシアティブ方式，LRU 置換法，ストア・スルー方式であり，最もよく採用されている方式の BM である[4]．

** BM と SBR と主記憶の総称として，ここでは，「メモリ」を用いる．

*** 4 回延べ 17 時間弱の測定データをまとめたもので 1 回目と 4 回目は約 9 箇月間の開きがある．この測定では，CPU がアイドルの時は各値を数えない．なお，命令の読み出しに関しては後述の 3.2 の(1)を参照せよ．

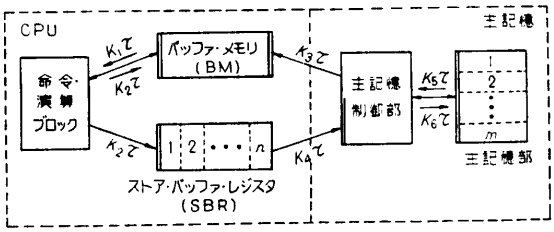


図 3 モデル・システムの構成
Fig. 3 Configuration of Modeling System.

BM はモデル 75 と同じ構成*とする．ミス率に関しては，CPA により得られた測定データを用いる．したがって，BM のモデルは，容量に関して考慮する必要がなく，単一のリソースとして簡単に考えられる．しかし，現実の動きは十分に反映されている．

（2） SBR の構成

ストア・スルー方式では，データの格納時，対応するブロックが BM 上にあるか否かにかかわらず，主記憶に必ず書き込むため，ストア命令の処理で BM の速さを活かせられない．この解決のために，SBR が用意されている．ストア命令は SBR に格納することで処理の完了とし，実際に主記憶へ書き込むまで待たない，おいてきぼり制御のため，実行が速くなる．

SBR は図3のようにレジスタ（長さは BM へのアクセス単位長に同じ）が n 個分用意されていて，$K_2\tau$ 時間で最も主記憶制御部寄りの空いている箇所へデータが格納される．SBR も一種の BM と考えられるが，CPU から SBR の内容は読み出せない．そのため，SBR 中のデータに対応するブロックのロード要求に対しては，ブロック・ロードの再試行，または，ロード要求の一時抑止が行われる**．この状況が生じるのは，ブロックをロード要求し未だ BM へ転送完了していない間に，そのブロックへデータの書き込みが生じた場合と，BM にないブロックへの書き込み要求後，そのデータが主記憶制御部へ転送完了していない間に当該ブロックのロード要求が出た場合である．ここでは，上に述べた BM にない同一ブロックへのアクセスの重なりは考えない（仮定1）．

（3） 主記憶の構成

主記憶は制御部分と記憶部分とからなり，アクセスの競合を減少させるため，バンクという独立に動作する単位を m 個持つ（m ウェイのインタリーブ）．バンクでのアクセス単位は2語で，1ビット誤りの自動訂正をハミング・コードにより行う．主記憶の種類については次の2つを考える．

（i） S-A 型（破壊読み出し型）

主記憶から読み出すと，主記憶上のそのデータが壊れるために直ちに書き込む必要のある記憶装置で，モデル 75 の主記憶はこれに当る．

（ii） S-B 型（非破壊読み出し型）

主記憶へのすべてのアクセスは同一時間要する記憶装置で，読み出しデータの再書き込みを必要としない．

（4） 書き込みにおける BM と SBR の関係

モデル 75 では，BM と SBR へ個々別々にデータが格納される．格納データに対応するブロックが BM 上にある場合，格納データに BM 上のデータをマージしアクセス単位の2語にデータを拡張することを考える．格納時間は $(K_1+K_2)\tau$ とし，主記憶が S-B 型のものを M-B 型と呼ぶ*．

3.2 命令の処理と情報の流れ

パイプライン制御計算機では命令の処理をいくつかのフェーズに区分し，各フェーズでは1つの命令のみが処理されるが，各フェーズごとに次から次へと命令が処理されるため，複数の命令が同時に実行される．このフェーズの分け方は計算機によって異なるが，ここでは，命令の読み出し，命令解読とアドレス計算，データの読み出し，演算実行（演算またはデータの格納）の4つのフェーズに区分する．なぜならば，モデル 75 に準拠したモデルにするとともに**，メモリと命令の処理の速さとの関係を求めるためのモデルであるので，メモリ・アクセスの種別によって分けることが必要と考えられるからである．また，この区分けは一般的と思われる．

4つのフェーズにおける命令の処理の様子を**図4**に例示する．図中の破線部は，そのフェーズでの処理は

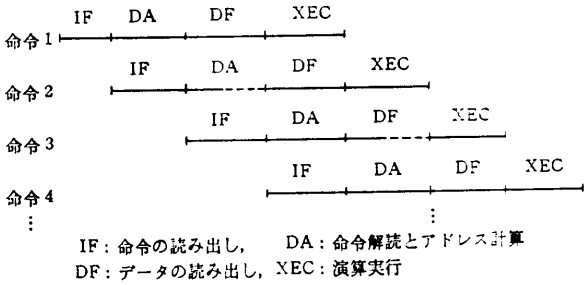


図 4 命令の流れ
Fig. 4 Pipelined Instruction Execution.

* p. 528 の脚註(*)参照．
** SBR の内容が読み出せる場合は，この動作は考えなくてよい．

* 主記憶が S-A 型のものは，マージの効果が期待できないので，ここでは取り上げない．
** モデル 75 は仮想記憶方式でないためアドレス変換は考えない．

完了したが，先行する命令が前のフェーズにて処理未完でインタロックをかけているため，その処理の完了を現フェーズ内で待っている状態を示す．

（1） 命令の読み出し（フェーズ1）

図3に示したように命令・演算ブロックからの命令読み出し要求に対し，BM 上で見つかれば $K_1\tau$ 時間要して転送される．この時，他フェーズから BM にアクセスしていれば待たされる．

BM 上で見つからなければ，読み出すべき命令を含む1ブロックを $K_5\tau$ 時間費し主記憶部から主記憶制御部へ転送する．続いて，主記憶制御部から $K_3\tau$ 時間費し BM へ転送されるが，1ブロックを1度に転送できないので，BM のアクセス単位（2語）ごとに分け，少しずつ (τ) 遅れて転送される．この転送時に，BM をバイパスして求める命令が命令・演算ブロックへも送られる．この動作では，主記憶部からの読み出しと BM への転送で待ちが生じる（図3の2重線部参照）．

モデル 75 ではもう1つの命令読み出しのルートがある．BM へのアクセスは2語単位で，2語が一度に読み出されて命令・演算ブロックに転送される．命令の読み出しで，もし直前に読み出された2語の後の1語が読み出すべき命令ならば，メモリに読み出し要求を出さないで，命令・演算ブロックに一時保持されたものを用いる．このような命令読み出しの方法は多くの計算機で行われ，その影響は小さくないため，モデルに採り入れる．なお，所要時間は τ とする．

（2） 命令解読とアドレス計算（フェーズ2）

このフェーズはどの命令も同じであるので，所要時間も一定 (τ) とする．先行する命令の演算実行により，データのアドレスを計算し直す必要の生じる場合があるが，ここでは考慮しない（仮定2）．

（3） データの読み出し（フェーズ3）

データの読み出しは命令の読み出しと同様である．ただし，2語を一度に読み出しても必要な部分以外は捨てられる．また，データをメモリから読み出さない命令でも，このフェーズを飛び越すことはない．

（4） 演算実行（フェーズ4）

ストア以外の命令は，その命令ごとに定まった時間を要して実行される．

ストア命令は，SBR に，BM に対応するブロックがある場合は BM にも，データを格納して処理は終了する．その後，SBR から $K_4\tau$ 時間で主記憶制御部へ，さらに，$K_6\tau$ 時間で主記憶部へ書き込まれ，実際のストア動作が完了する．この動作では，SBR と BM への格納と主記憶への転送で待ちが生じる（図3の2重線部参照）．なお，格納データ長が主記憶のアクセス単位（2語）より短い場合，まず主記憶部から読み出しを行い，2語にまとめ，エラー自動訂正コード(ハミング・コード)化して書き込む必要がある．

モデル 75 では1語長の1アドレス命令がほとんどで，格納データも1語長以下が多い[6]．そこで，モデルでは命令はすべて1語長の1アドレス命令で，格納データは1語長以下と仮定する（仮定3）．

上に述べたように，次から次へと遅滞なくアドレス順に命令を処理するが，分岐命令のような命令の実行順序をアドレス順と異なるようにする命令が実行されると，命令処理において遅れが生じる．この遅れとモデルでの扱いについては次章で述べる．

4. シミュレータと入力データ

4.1 シミュレータ

シミュレータは実行すべき命令を次々と入力データとして読み込み，前述の命令処理の流れに沿ってシミュレートするが，同時に動作する構成要素が4つのフェーズと3種のメモリと数多くあり，かつ，同程度の時間を要するため，シミュレーション時計を単位時間ずつ進めるタイム・スライス方式とした．命令処理の各フェーズの制御はカスケード的に行い，フェーズの区分に対して柔軟性を持たせる構造とした．また，メモリに関してはアクセスにおける競合に主眼を置いた．

分岐命令等による命令処理の遅れをシミュレータで考慮すると，シミュレータは複雑化する．そこで，入力データである命令列に **4.3** で述べる工夫をし，シミュレータは命令処理の遅れに配慮しないで作成できるようにした．

4.2 入力データ

実行される命令列はシミュレータへの入力データとして与えられるが，命令ごとに以下に示す項目を含む．

- 命令のタイプ（**表1**のクラスおよび付加情報），
- 命令のアドレス（BM 上にあるか否か，または命令・演算ブロックにすでにあるかの区別を含む），
- データのアドレス（読み出しか格納かの区別，および BM 上にあるか否かを含む），
- 演算実行時間（フェーズ4の時間で，ストア命令はシミュレーションにより定まる）．

表 1 命令のクラス分け
Table 1 Classes of Instructions.

クラス	タイプ	データの読み出し	データの書き込み	演算時間（単位：τ）
1	ロード（置数）	○	×	1
2	ロード（置数）	×	×	1
3	ストア（格納）	×	○	$1+K_2$
4	固定小数点加減算	○	×	1
5	固定小数点加減算	×	×	1
6	比較	○	×	2（成立） 3（不成立）
7	無条件分岐	×	×	2
8	条件付分岐	×	×	2（不成立） 3（成立）
9	浮動小数点加減算	○	×	4
10	浮動小数点乗算	○	×	6
11	浮動小数点除算	○	×	15
12	固定小数点乗算	○	×	5
13	固定小数点除算	○	×	26
14	シフト	×	×	4
15	論理演算	○	×	1
16	レジスタを使用しない	×	×	1
17	その他	×	×	1

注 ・○は有，×は無を示す．
・インデックス・レジスタ関係の命令は各クラス内に分けて含めた．

4.3 命令処理の遅れについて

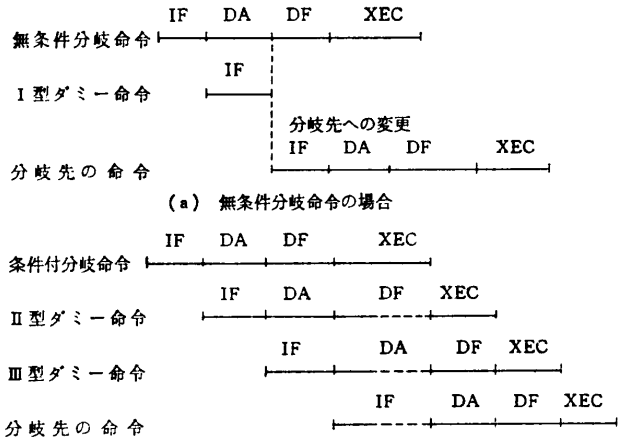
分岐命令等による命令処理の遅れについて簡単に説明し，シミュレータと入力データとでいかに機能分担してシミュレートするかを該当する命令ごとに述べる．特に，命令処理を連続した流れとするために挿入するダミー命令について述べる．

（1） 無条件分岐命令

図5（a）に示すように，無条件分岐命令がフェーズ2に入った時点で命令読み出しアドレスの変更が判明し，直ちに分岐先アドレスから命令が読み出される．無条件分岐命令のアドレス順で次の命令は，フェーズ1は通常の処理がなされるが，フェーズ2以降は処理されない．これに対応する命令をⅠ型ダミー命令と呼び，無条件分岐命令の次の入力データとして挿入される．シミュレータではⅠ型ダミー命令のフェーズ1完了時点で，ほかのフェーズの状態に関係なく，フェーズ1を空き状態にする．

（2） 条件付分岐命令

条件付分岐命令がフェーズ2で解読された時点から，分岐と非分岐の両ケースに対応する命令が並列的に読み出される．ここでは，実行されない命令列関連のメモリからの読み出しはないと仮定する（仮定4）．非分岐の場合を予測して，フェーズ2,3の処理を先廻りして進めていくことから，条件不成立の時は通常の非分岐命令と同様に考えればよい．条件成立時は，図5



（a） 無条件分岐命令の場合

（b） 条件付分岐命令の場合

図 5 分岐命令による遅れとダミー命令
Fig. 5 Instruction branch delays and dummy instructions.

（b）に示すように，条件付分岐命令のアドレス順で次の命令が読み出され，続いて分岐先命令が読み出される．ところが，分岐先命令がフェーズ2に入るのは条件付分岐命令がフェーズ4を完了した直後である．

アドレス順で次の命令は，命令・演算ブロックにすでに保持されていて，データをメモリから読み出さず，演算実行時間が最小（τ）の命令（Ⅱ型ダミー命令と呼ぶ）と設定する．分岐先命令は，命令のアドレスの項目のみ分岐先アドレスでほかはⅡ型と同じ命令（Ⅲ型ダミー命令と呼ぶ），および，命令のアドレスの項目は命令・演算ブロックにすでに保持されているが，ほかは分岐先命令に対応する命令に分けて設定する．シミュレータでは，Ⅱ型ダミー命令がフェーズ2から3へ移る時点で，Ⅲ型ダミー命令がフェーズ1を終了していなければ，その終了時刻を単位時間（命令・演算ブロックに保持されている命令の読み出し時間）だけ早める．

このようにして，条件付分岐命令と分岐先命令とのフェーズの遅れをダミー命令で補償する方法を考えた．そして，Ⅲ型ダミー命令の読み出しにおけるBMでの待ちや主記憶からの読み出しによる必要以上の遅れに対して，シミュレータが遅れ（単位時間）を取り戻すようにした．この結果，命令処理の流れの制御に関して，シミュレータは分岐命令も非分岐命令と同等に扱え，単純化された．

（3） 比較命令

比較命令で条件不成立時，アドレス順で次の命令の実行がスキップされる．スキップされる命令は，演算実行時間が単位時間になる以外は通常と同じ命令（Ⅳ

表 2 京大結合ミックス
Table 2 Kyoto University concatenate instruction mix.

I_j \ I_i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	京大ミックス
1	0.164	0.022	0.206	0.075	0.121	0.104	0.055	0.028	0.037	0.064	0.020	0.020	0.000	0.008	0.000	0.0	0.056	0.228
2	0.127	0.135	0.086	0.078	0.017	0.221	0.057	0.027	0.027	0.0	0.0	0.182	0.006	0.033	0.000	0.0	0.003	0.052
3	0.450	0.083	0.132	0.010	0.118	0.084	0.088	0.026	0.001	0.0	0.0	0.001	0.000	0.003	0.001	0.000	0.005	0.198
4	0.065	0.006	0.474	0.000	0.110	0.040	0.022	0.265	0.0	0.0	0.0	0.010	0.0	0.008	0.001	0.0	0.0	0.030
5	0.117	0.018	0.294	0.006	0.196	0.247	0.081	0.039	0.0	0.0	0.0	0.0	0.0	0.002	0.000	0.000	0.0	0.113
6	0.082	0.030	0.113	0.002	0.045	0.135	0.545	0.012	0.001	0.000	0.000	0.0	0.0	0.005	0.026	0.001	0.003	0.119
7	0.314	0.041	0.188	0.012	0.126	0.145	0.127	0.035	0.001	0.001	0.0	0.000	0.000	0.004	0.000	0.004	0.001	0.131
8	0.319	0.039	0.223	0.000	0.099	0.060	0.139	0.073	0.038	0.004	0.0	0.0	0.0	0.003	0.001	0.001	0.001	0.036
9	0.134	0.025	0.531	0.0	0.020	0.000	0.004	0.111	0.145	0.008	0.021	0.0	0.0	0.0	0.0	0.0	0.0	0.018
10	0.255	0.005	0.459	0.0	0.003	0.000	0.002	0.0	0.245	0.031	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.016
11	0.014	0.0	0.913	0.003	0.000	0.001	0.0	0.003	0.066	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.005
12	0.006	0.0	0.056	0.025	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.0	0.899	0.0	0.0	0.008	0.015
13	0.0	0.391	0.104	0.0	0.096	0.0	0.002	0.001	0.0	0.0	0.0	0.0	0.0	0.105	0.0	0.0	0.302	0.001
14	0.072	0.033	0.082	0.209	0.464	0.038	0.017	0.004	0.0	0.0	0.0	0.009	0.016	0.057	0.000	0.0	0.000	0.020
15	0.0	0.0	0.870	0.0	0.002	0.002	0.057	0.020	0.0	0.011	0.0	0.0	0.020	0.008	0.011	0.0	0.0	0.004
16	0.168	0.140	0.052	0.0	0.041	0.001	0.481	0.037	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.081	0.0	0.001
17	0.295	0.620	0.005	0.0	0.030	0.001	0.045	0.004	0.0	0.0	0.0	0.0	0.0	0.000	0.0	0.0	0.0	0.015

型ダミー命令と呼ぶ) とするだけでよい．なお，ストア命令ならば，非ストア命令にする．

5. シミュレーションの実行と結果

5.1 入力データ（命令列）の作成

命令の実行順序は遷移確率である結合ミックス[6] (ただし，命令のクラス分けは表1，値は表2を参照)，分岐命令の分岐先アドレスはアドレス・トレーサで得られた確率分布[7]に従うものとし，乱数を発生し，4.3 で述べたダミー命令を挿入しながら，4.2 で示した項目からなる命令列を作成した*．BM のミス率に関しては，10 秒間ごとの測定でかなりの範囲に分布することとアクセス種別ごとの相関が高いことから，各値共に，平均値 (Case 1)，最良の状態 (Case 0，各値は 0)，きわめて悪い状態 (Case 2，$P_X=P_R=0.1$，$P_W=0.3$) の3ケースを考えた．

5.2 シミュレーションの実行とパラメータ

図3のパラメータのうち，K_5, n (SBR の数)，m (インタリーブのウェイ数) を変え，K_6 は主記憶が S-A 型の時 $2K_5+1$，S-B 型の時 K_5+1 とした．そのほかは $K_1=1, K_2=K_3=2, K_4=6$ と一定にした**．

シミュレーションの実行では，1,000 命令（ダミー命令は除く）実行ごとに命令処理時間率***を求め，その平均値の信頼区間幅が危険率5%のもとで，平均値の5%以内になればシミュレーション・ランを終了させた[8]．

5.3 シミュレーションの結果

シミュレーションにより，命令処理時間率とメモリの関係を検討するが，図6は Case 1，図7は Case 0，図8は Case 2 における結果を示す*．これらの結果

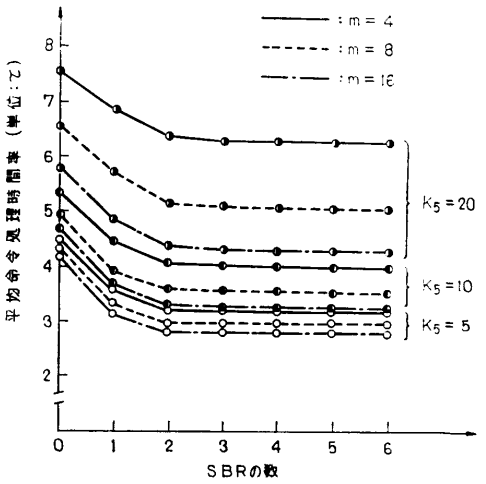


図 6 平均命令処理時間率 (Case 1)
Fig. 6 Average execution time rate (Case 1).

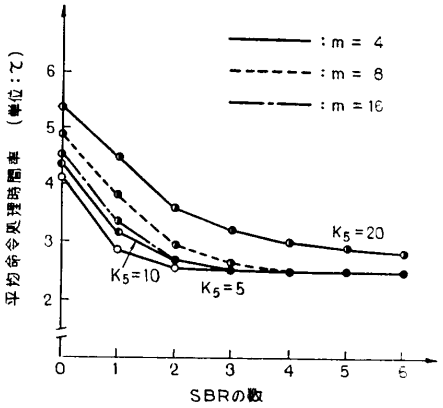


図 7 平均命令処理時間率 (Case 0)
Fig. 7 Average execution time rate (Case 0).

* 比較命令の条件成立率は 0.450 (CPA による測定)，条件付分岐命令の条件成立率は 0.524 (トレーサによる測定)．

** モデル 75 を目安とし，変化させたパラメータに対しては $K_5=5$，$n=3$ が対応し，m は 4,8,16 が構成により選べる．

*** k 個の命令実行時間を k で割った値．

* Case 0 では，K_5 が 5,10 の場合，m による差はほとんどなかったため図示されていない．

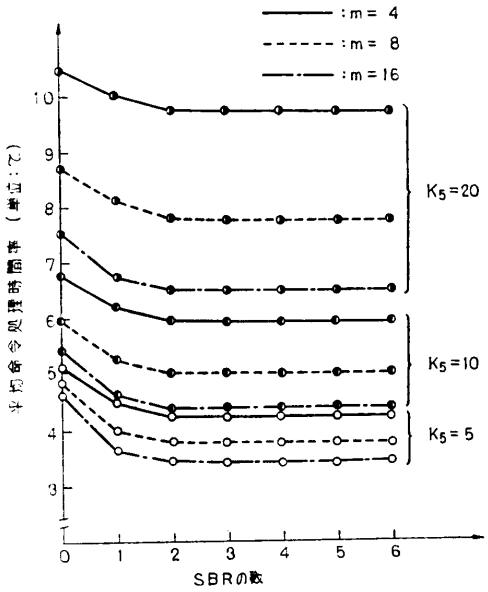


図 8 平均命令処理時間率 (Case 2)
Fig. 8 Average execution time rate (Case 2).

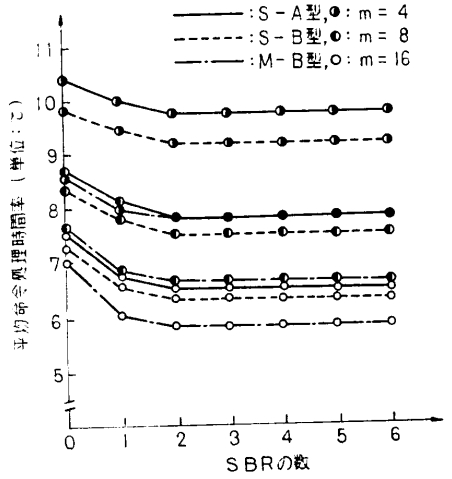


図 9 主記憶の型と平均命令処理時間率
Fig. 9 Average execution time rate effected by main memory types.

から次のことが判った.

- m を大きくすることは，BM のミス率の増加，および，K_5 の増大に対して，命令処理時間率の向上により大きな効果をもたらす.
- SBR による効果は，Case 0 の K_5 が 20 で m が 4 または 8 の場合を除けば，2 個までである.
- m が小さい時より大きい時の方が，SBR による効果がより大きい.

結局，我々のシステムの利用環境では，主記憶の構成に関係なく，SBR の数は 2 個で十分であり，ミス率の増大と主記憶の相対的速度の低下には，インタリーブのウェイ数の増大が有効であることが判った.

表 3 シミュレーション結果と測定値の比較
Table 3 Comparison of Simulation results with Measured data.

ケース	シミュレーション結果	測定値
Case 0	2.4	2.4
Case 1	3.2	3.3
Case 2	4.2	4.4

ただし，m=4, K_5=5, n=3, τ=90 ナノ秒.

図 9 に，S-A 型，S-B 型，M-B 型による違いの大きかった Case 2 の K_5 が 20 の場合を示す．一般に S-B 型は S-A 型より優れている．しかし，図示してないが，ほかの場合の差はシミュレーションの誤差範囲内であった．M-B 型は K_5 が大きい時に多少効果があったが，小さい時にはかえって低下させることもあった．すなわち，S-A 型と S-B 型の間には有意差がみられなく，M-B 型はデータの格納時間をほかの型と同一にできない限り意味がないことが判った.

シミュレーションの結果と，同じ命令の遷移確率からなるテスト・プログラム[6]を実行させて得られた結果を比較すれば，**表 3** のようになった．仮定 1 から 4 による影響は比較的小さいと考えられる.

6. まとめ

パイプライン制御計算機のプログラム実行時間が，メモリの構成と速さによって受ける影響を調べるためシミュレーション・モデルを作成し，評価を行った．我々のシステムの利用環境では，主記憶の構成や速さに関係なく，SBR の数は 2 個で十分であることが判った．対象とした計算機はバッチ処理専用システムであるが，会話型処理を行うオンライン・システムにした場合，BM のミス率の増大が予想される．その場合，プログラムの実行速度を早めることに関して，主記憶のインタリーブのウェイ数を増大させることの効果が明らかになった.

シミュレーション・モデルの作成において命令処理の過程を一般的にモデル化することを考えた．更に，命令の実行順序の変更（分岐）に伴う遅れに対し，ダミー命令挿入方式を考え，すべて非分岐命令として単純に扱えるよう工夫をした．BM のミス率や命令の実行順序等に関しては測定データを用いているので，実際的なシミュレーション結果を効率良く得ることができた．また，ミス率に関して，アクセス種別ごとに定義する必要性を指摘できた.

ほかの利用環境（異なる命令の遷移確率）でも，同

様なある一定の SBR の数とインタリーブの有効性が見られると考える．また，ダミー命令挿入方式はパイプライン制御計算機の一般的なシミュレーションに適用できると考える．

最後に，CPA による測定でのご協力，並びに，ご助言を戴いた富士通(株)三輪修氏はじめ関係者各位に深く感謝します．なお，本研究の一部は，京都大学大型計算機センターの開発計画の下で行われた．

参考文献

1) Rau, R. B. et al.: The Effect of Instruction Fetch Strategies upon the Performance of Pipelined Instruction Units, Proc. 4th Annual Symposium on Computer Architecture, pp. 80-89 (1977).
2) Lang, D. E. et al.: A Modeling Approach and Design Tool for Pipelined Central Processors, Proc. 6th Symp. on Computer Architecture, pp. 122-129 (1979).
3) Peuto, B. L. et al.: An Instruction Timing Model of CPU Performance, Proc. 4th Symp. on Computer Architecture, pp. 165-178 (1977).
4) 長島他：キャッシュ記憶，情報処理，Vol. 21, No. 4, pp. 332-340 (1980).
5) 乾他：FACOM 230-75 性能測定器，情報処理学会第 15 回大会講演論文集 (1974).
6) 金沢他：結合ミックスによる CPU の性能評価，情報処理，Vol. 14, No. 9, pp. 715-718 (1973).
7) 中村他：バッファ・メモリ方式のシミュレーション，情報処理，Vol. 15, No. 1, pp. 26-33 (1974).
8) 金沢他：計算機システムのシミュレーションについて，情報処理，Vol. 13, No. 10, pp. 648-691 (1972).
9) 三輪他：FACOM 230-75 の方式，FUJITSU, Vol. 25, No. 7, pp. 85-108 (1974).

(昭和 55 年 11 月 25 日受付)
(昭和 56 年 5 月 20 日採録)