シミュレーションによる大型制御用計算機構成の評価と検討†

福永　泰††　坂東忠秋††　川本幸雄††
奥田健三†††　加藤　猛††††　井手寿之††††

近年，高速処理性能を達成するために，制御用計算機にも種々の並列処理が導入される傾向にある．パイプライン処理は，並列処理方式としては，性能第一を目指す大型計算機に，従来より広く使用されている方式であるが，規模の小さい制御用計算機に導入する場合には性能／コストの点を十二分に考慮する必要がある．

パイプライン処理を有効に機能させるには，十分なメモリスループットを確保する必要があり，このためのメモリバス幅，メモリのインタリーブの構成方式，命令スタックのサイズの3点を，相互に関連性を持つ要素と考え，分析評価を行った．

まず，この3点を考慮した 16 ビット計算機の最適構成を求めるため，シミュレーションプログラムを新たに開発し，このシミュレーションプログラムにより，32ビットのメモリバス，2ウェイインタリーブのメモリ，4語の命令スタックがパイプライン制御に最適であることを明らかにした．

これ等の最適化により，処理装置の性能は，パイプライン制御を実施しない計算機の約2倍で，S-TTL を用いた場合，1 MIPS の高速処理性を実現できることを明らかにした．

1. はじめに

大規模化傾向にある計算制御システムのニーズに対応して，制御用計算機は，高速処理，大容量メモリ化の方向を指向している．このために，従来は，大型計算機の領域にしか適用されていなかった各種の並列処理，および先行制御方式が，高速処理性を実現するため随所に採用されてきている．しかし，並列処理や先行制御方式を，性能／コストの要求が厳しい制御用計算機に導入するためには，計算機システムを解析し，システムネックを解消する，いわゆるバランスのとれたシステム構成のための技術として評価する必要がある．したがって，新しいハードウェアを設計するにあたり，実際のシステムにおいて，上記処理性能向上策が，どの程度の効果を有するのかを事前に定量的に評価し，適切な処理性能が得られるハードウェアの論理構造を決定することが，必要不可欠である．

我々が検討の対象とした制御用計算機は，多くの命令実行処理において，アドレス演算に使用されるレジスタと，命令実行処理に用いられるレジスタが明確に分離している．そこで，これに着目して，パイプライン処理の導入を図ったが，この高速化に伴って，最適なメモリスループットを評価する必要性が生じた．

こうしたメモリインタフェースの評価を行った論文には，プリフェッチを行う計算機に関する論文[1]や，パイプライン処理，マルチプロセッサに関する論文[2)~5)]が発表されている．しかし，これらの論文では，命令の実行過程をモデル化しているため，実際の処理装置の動作とは異なった点がある．たとえば，3）では1命令実行を1回のフェッチサイクルと1回のメモリ参照で定義しており，5）ではパイプライン処理の1セグメントから必ず1回のメモリ起動が発生するものとしている．

ところが，実際の計算機においては，各命令によってメモリアクセスの頻度が異なり，上記仮定が十分に満たされているとは言えない．我々が考察したハードウェアにおいても，命令の解読部では1語アクセスのものと2語アクセスのものがあり，実行部では，命令の種類により0語から8語アクセスまで分布している．

そのため，ハードウェアの動作を詳細に評価し，最適構成を求めるには，現実の動作を正確に評価するのが難しい解析的手法よりもシミュレーションが最適であると考え，シミュレーションプログラムを開発し，

† Evaluation of Large Scale Control Computer Architecture through Simulation by YASUSHI FUKUNAGA, TADAAKI BANDOH, YUKIO KAWAMOTO, KENZO OKUDA (Hitachi Research Laboratory, Hitachi Ltd.), TAKESHI KATO, and JUSHI IDE (Omika Works, Hitachi Ltd.).

†† 日立製作所日立研究所
††† 日立製作所日立研究所（現在 宇都宮大学工学部）
†††† 日立製作所大みか工場計算制御設計部

（1） パイプライン処理を行った場合の効果，並びにパイプライン処理を効率よく行うための(2)命令語先き読み用スタック，(3)メモリインタフェースの方式検討とその評価を行った．

本論文では，ハードウェアの概要と，各種ハードウェア構成のシミュレーションによる評価結果について述べる．

2. 処理装置構成と検討課題

2.1 処理装置全体構成

考察する制御用計算機のレジスタは，専用レジスタ方式をとっているため，命令実行に用いるレジスタと，アドレス計算に用いるレジスタが分離している．このため，アドレス計算用の演算器と，命令実行用の演算器を設けることにより，ジェネラルレジスタ方式をとっている計算機よりも容易にパイプライン処理を導入できる．

ところがパイプライン処理の導入により，命令解読部である I UNIT と，命令実行部である E UNIT から，個別にメモリ要求が出力されることになり，メモリ要求の頻度が従来よりも増大し，これに対応するメモリスループットが必要となる．このため，命令語先き読み用スタック，メモリインタリーブ，およびメモリバス幅の増大で対処する方式を検討した．以上を前提とした，処理装置の概略構成を図1に示す．

図1のメモリ制御ユニット (MCU) は，命令先き読み用スタック，命令実行時のメモリ要求，入出力装置からのメモリ要求に対し，メモリ要求の選択，メモリの起動，データ転送の制御を行う．

スタックは，先行して命令語の読み出しを行い，最大 $2n$ 語 ($n=1,2,3,\cdots\cdots$) の命令を格納する．

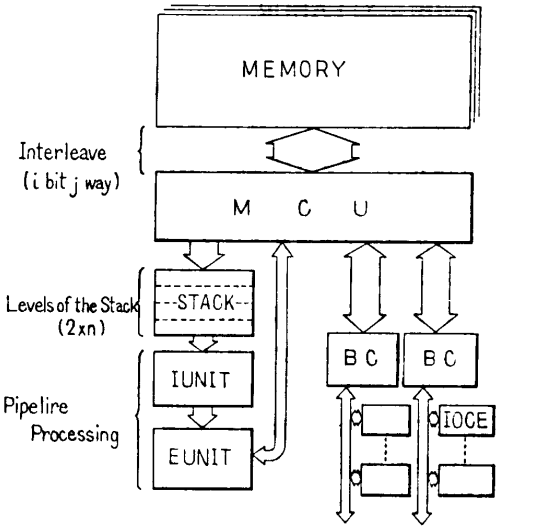
I UNIT はスタックより命令を取り出し，命令の解読，並びに実効アドレスの演算を行う．

E UNIT は，マイクロプログラム制御により，I UNIT から送られた実効アドレスで指示されるメモリとのデータ転送，並びに命令の実行を行う．この時，I UNIT は，次の命令の解読，および実効アドレスの演算を行う，いわゆるパイプライン処理方式を採用している．

以上のハードウェア構成で，処理率向上に対して考慮すべき項目としては，

（1） パイプライン制御の構成方法，およびその評価

（2） 命令用スタックの構成方法，およびその評価



MCU : Memory Control Unit
BC : Bus Controller
I UNIT: Instruction Unit　E UNIT : Execution Unit
IOCE : Input Output Control Electronics

図 1 処理装置ハードウェア構成
Fig. 1 The CPU hardware structure.

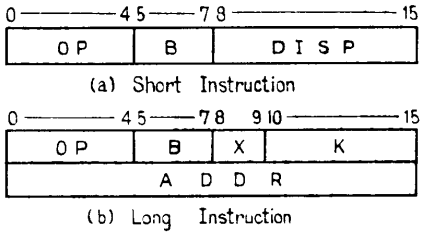


図 2 命令語フォーマット
Fig. 2 Instruction format.

（3） メモリインタリーブ数と，バス幅の構成，およびその評価

の3項目があげられる．

2.2 パイプライン制御の導入

パイプライン制御導入の難易度は命令語のアーキテクチャに強く影響され，各ステージで使用するレジスタをはじめとする資源が明確に分離しているものほどパイプライン制御を導入しやすい．

考察対象の制御用計算機の命令語は前述したように，各ステージで使用するレジスタが分離されている．そのフォーマットを図2で示す．オペランドを必要とする命令語の実効アドレス (EA) の演算は，16ビットの短語命令(a)では，

$$EA=(B)+DISP$$

で求め，一方，32 ビットの長語命令(b)では，

EA=(B)+(X)+ADDR

で求める．ここで，Bはベースレジスタ番号を示し，Xはインデックスレジスタ，DISP, ADDR はアドレス情報である．

こうした実効アドレスの演算，並びに命令コードの解読が I UNIT で行われるが，この時使用されるベースレジスタやインデックスレジスタとは別に，命令実行時にはアキュムレータが使用される構造をとっている．そこで，I UNIT より与えられた実効アドレス，および命令処理マイクロプログラムの開始アドレスを元に，E UNIT で命令の実行が行われる間に，I UNIT で次の命令の解読が行われる2段階のパイプライン制御が可能となる．

E UNIT 実行開始と同時に，一般には I UNIT で次の命令の解読が行われるが，次のような制限を受ける．

（1） 命令実行時に，I UNIT 中にある資源（主にレジスタ）を使用する命令は，その使用が終了するまで，I UNIT の動作を開始できない．

（2） 命令実行時に，プログラムカウンタの内容を変更する命令は，変更終了まで，I UNIT 動作を開始できない．

こうした制限を考察した上でのパイプライン処理の効果，およびその評価については，第4章で記述する．

2.3 命令用スタックの構成

命令用スタックは，I UNIT から 16 ビット単位で順次読み出せるように制御され，その層数は，$2n$ 語（1語=16 ビット）持つと仮定される．

スタックの層数が少ないと，I UNIT からスタックを参照しても，所望の命令が入っている確率が小さくなり，I UNIT からのアクセスタイムが長くなる．一方，スタックを無効とするジャンプ命令の頻度が比較的多いため，層数を多くすると，あらかじめスタックに読み込んでいても，使用されずに捨てられる命令語が増加し，その分，メモリを占有し，他のアクセスを妨害するため，かえって性能を低下させることになる．このことから，スタックの最適層数を求める必要がある．

次にスタックの制御方式について述べる．I UNIT からのスタック読み出しが終了した時点で，スタックは1語分進められる．

スタックに空がある場合のメモリへの要求方法は，メモリバス幅が 16 ビットの場合と，32 ビットの場合とでは異なり，前者の場合は，1度のメモリ参照で，1語のデータが読み出されるので，スタック中に空があれば，スタックからメモリ要求が出力される．一方後者の場合は，1度のメモリ参照で，2語のデータが読み出されるため，スタック中に2語分の空が出来て始めてメモリ要求が出力される．

また，ジャンプ命令等で，プログラムカウンタの変更があった場合，スタック内をすべて空にするように制御される．

2.4 メモリインタフェース

メモリスループットを増大させる方法としては，

（1） メモリバス幅の増大

（2） インタリーブ数の増大

（3） メモリアクセスタイム，サイクルタイムの減少

の3つの手法が考えられる．

（1）に対しては，考慮する処理装置内バスが 16 ビットであることから，16 ビット，32 ビット，64 ビット等の構成が考えられるが，実装上の問題から，16 ビットと 32 ビットに関して比較検討を行う．

（2）に対しては，インタリーブを行うメモリそれぞれに対して独立にアドレス線やデータ線を接続する，いわゆる空間分割の方式と，信号線は同一でも，その信号線を時間で分割して使用する時分割の方式が考えられる．前者の場合は，インタリーブを行うメモリに対して，個別の信号線が必要であるため，常時，メモリ起動をかけることが可能であるが，信号線の数が倍増する欠点を有する．後者の場合は，信号線上で干渉があるため，性能がそれだけ低下するが信号線の数は少なくてすむ．これより，時分割方式に対するシミュレーションを行い．どの程度，メモリスループットに影響があるかを確認する必要がある．

（3）に対しては，使用できる素子の速度に依存するため，検討項目から除外した．

処理装置から見た場合のメモリインタフェースを図3に示す．処理装置からメモリ要求 (MEX) が出力されると，メモリ制御ユニット (MCU) での選択時間 (t_{st}) を経過した後，アドレスバスを占有する (t_{absy})．対応するメモリが Busy でなければ，メモリは起動さ

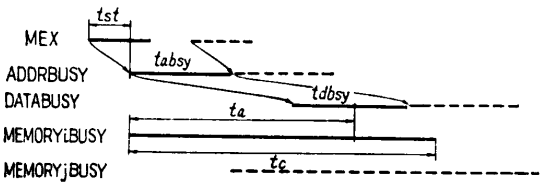


図 3 メモリインタフェースタイムチャート
Fig. 3 Memory interface time chart.

れ，アクセス時間 (t_a) 経過後，メモリから処理装置に応答が返る．メモリはデータを出力してからもしばらくの間 Busy 状態を保つ (t_c)．データバスは応答が返る前から Busy 状態となり，応答が返った後も一定時間 Busy 状態が続く (t_{absy})．

他のメモリ要求がある場合には，次にすぐ，図の破線で示されるようにメモリが起動できるので，独立に動作可能なメモリ数を M，メモリサイクルタイムを t_c，1回の転送に対し，アドレスバス，データバスが Busy である時間をそれぞれ t_{absy}，t_{dbsy} とすると，インタリーブ数 N は，

$$N=\min(M,\ t_c/\max(t_{absy},\ t_{dbsy}))$$

で表わすことができる．

3. シミュレータ入力データの考察

本論文で述べるシミュレータは，第2章で述べたハードウェア構造をシミュレートしているが，その出力結果が信頼たるには，与えられる入力データが実際のシステム動作を充分に反映しているかどうかを評価しておく必要がある．

入力データ中，第1に問題となるのは，実行する命令ストリングの指定方法である．これに対しては，制御用計算機のオンライン用のオペレーティングシステムプログラム全体のリストより，各命令の使用頻度を計算して求めたスタティックな命令頻度表を元に図4に示すように乱数発生により決定する方式をとった．すなわち，0～100% の値の乱数を発生させ，累積頻度表によりその値に該当する命令語を次に実行する命令と定めた．

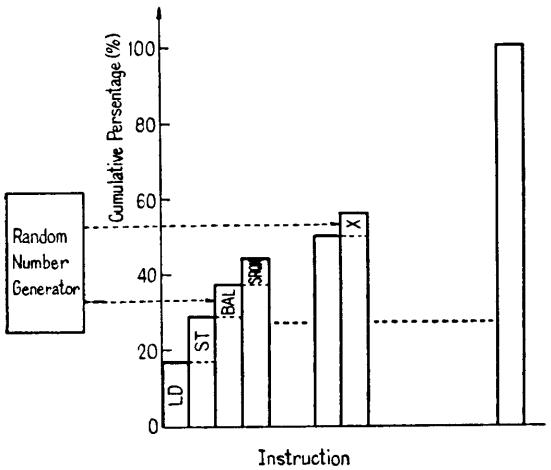


図 4　実行命令の選択方法
Fig. 4　Selection of execution instruction.

実際に実行する場合のダイナミックな命令頻度と，リスト上に分布するスタティックな命令頻度の間には，大きな相関があることが7）で明らかにされているために，トレースデータを用いるのではなく，本方式を採用した．実際に約500命令のダイナミックな命令ストリングを使用したものと比較しても，平均命令実行時間で1% 弱の差しかないことを確かめることにより，乱数発生方式の妥当性を検証している．

一方，1命令の実行については，マイクロプログラムの実行ステップを，

（1）　メモリ読み出しを含むマイクロプログラムステップ

（2）　メモリ書き込みを含むマイクロプログラムステップ

（3）　処理装置内部のハードウェアだけで動作するマイクロプログラムステップ

の3種に分けて，その実行順序に従って定義できるようにした．(1)，(2)に対しては，メモリの衝突によって待ちが発生するが，待ちがない場合は，それぞれ 500 ns，360 ns で動作が終了するものとし，(3)に対しては，1ステップ 160 ns で動作が終了するものとした．メモリサイクルを含む1ステップを上記のような値としたのは，メモリとして，アクセスタイム 340 ns，サイクルタイム 450 ns のメモリを用いることを考慮し，それに処理装置部におけるオーバヘッドを加算したためである．

一方，I UNIT の実行については，スタックに命令語がある場合，短語命令で 200 ns，長語命令で 400 ns 実行できると仮定した．

また，メモリインタフェースについては，アドレスバス，データバスの占有時間を両者共 200 ns とした．

これらの値は，S-TTL (Schottky Transistor-Transistor Logic) を用いてハードウェアを構成した場合の標準的な値である．

4. シミュレーション結果

4.1 パイプライン制御の効果

2.2 でパイプライン処理を制限する命令のタイプを示したが，その中で，(1)で示した命令語に対しては，さらに(1)-(a) E UNIT の実行が，I UNIT の資源を使用し終わってからも続行される場合と，(1)-(b)続行されない場合とがあり，前者の場合は残った E UNIT 実行時間の間はパイプライン処理が有効と

表 1 パイプライン処理の命令別分類

Table 1 Instruction grouping on pipeline processing type.

No.	Instruction Type	Typical Instruction	I-E Interaction	Percentage
1	Full Pipeline Processing	Load, Add, And, Or	t → I EXECUTE / E EXECUTE	53%
2	Partial Pipeline Processing	Shift	t → I EXECUTE / E EXECUTE	15%
3	No Pipeline Processing	Data Transfer Between I and E UNITS	t → I EXECUTE / E EXECUTE	10%
4	No Pipeline Processing (PC Change)	Branch	t → I EXECUTE / E EXECUTE	22%

なり，後者の場合では，パイプライン処理がまったくできない．そこで，最初からパイプライン処理が可能なもの，および上記(1)-(a)，(1)-(b)，さらに(2)に分けて，I UNIT, E UNIT の実行過程，および実行頻度を % で示すと，**表1**のようになる．

表1より，パイプライン処理は実行頻度で約 70% の命令に対して有効となる．

Type 3 の命令に対しては，次命令のアドレス演算で使用するレジスタと，実行中の命令で使用するレジスタが異なる場合は，パイプラインを実施させ，より効率を向上させる事が考えられる．しかしながら，本方式は，20～40 ns 平均命令実行時間を増加できるが，それ以上にハードウェア増加をまねき，コスト上の欠点が大きいことが明らかとなり検討から除外した．

さて，上記頻度でパイプライン処理が可能なハードウェアに対して，平均命令実行時間をどの程度減少させる効果があるかを示したものが**表2**である．

表2は，スタックを4語，独立に動作できるメモリが4個，メモリバス幅が 32 ビットの場合に，出現頻度の高い命令に対してパイプライン制御がどれだけ効果を上げているかを示したものである．

表1に示した命令の分類の内，(1)，(2)に属するものは，300 ns～400 ns 実行時間を短縮しており，全体として 300 ns 弱実行時間の短縮が可能である．これは，割合で示すと，約 25% の処理率向上である．

パイプライン処理によるハードウェアの増加は IC 数にして約90個であり，パフォーマンス向上率として，IC 1 個当りの性能向上率をとると，I UNIT, E UNIT 分割によるパイプライン処理の効果は 0.28%/IC となる．

表 2 パイプライン処理の効果

Table 2 The effect of the pipeline processing.

No.	a) Instruction	b) Execution Number	c) Mean time with Pipeline Processing	d) Mean time without Pipeline Processing	d)—c)
1	LD (S)	165	558 (ns)	972 (ns)	414 (ns)
	ST (S)	143	457	810	353
	LD (L)	40	608	996	388
	ST (L)	24	485	848	363
	S (S)	20	562	972	410
2	SRQN (S)	84	1360	1714	354
	SRQF (S)	46	1360	1693	333
	LDB (L)	41	1233	1627	394
	STB (L)	37	1628	1993	365
	STR (L)	30	639	920	281
3	TTR (S)	29	574	574	0
	LDR (S)	27	564	564	0
4	BAL (L)	101	1174	1267	93
	BC (L)	59	1283	1316	33
	B (S)	50	1092	1207	115
Σ		1000	918	1203	285

(S): Short Instruction
(L): Long Instruction

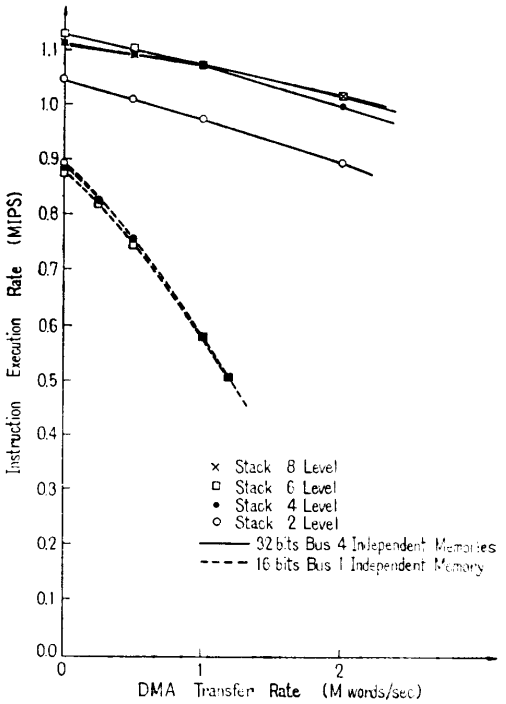


図 5 メモリ構成に対するスタックの効果

Fig. 5 Effect of stack depending on memory structure.

4.2 命令用スタックの効果

図5にメモリスループットが大きい場合，すなわち

32 ビットバス幅，独立に動作可能なメモリが 4 個の時のスタックの層数による処理性の変化と，メモリスループットが小さい場合，すなわち16ビットバス幅，独立に動作可能なメモリが1個の場合のスタックの層数による処理率の変化を併せて示す．

メモリスループットが大きい場合，スタックは層数2個よりも，4個以上の方が6%ほど処理率が向上し，その差異は顕著なものがあるが，4個以上にすると，差はほとんど無いといえる．これは，独立に動作可能なメモリが2や1の場合でも，32 ビットバス幅の時は同じ傾向を示す．

一方，スタック4層によるハードウェアの増加は，IC 数にして約25個であるため，IC 1個当りの性能向上率は 0.24%/IC となる．

ところが，バス幅が 16 ビットと小さくなり，メモリスループットが減少すると図に示すように，スタックによる処理率の違いはほとんどなくなり，むしろスタックの層数が多い方が処理率が落ちるような傾向を示す．これは，独立に動作するメモリが2個に増加しても同様で，4個に増加して始めて，4層スタックが2層スタックよりも処理性能が向上する結果が得られた．

以上より，メモリスループットに見合った最適なスタック層数が存在することが明らかとなり，たとえば32 ビットバス幅の時は4層が最適であることが判明した．

4.3 メモリインタリーブの効果

図6に，スタック4層の時のメモリバス幅を変化させた場合，および，独立に動作可能なメモリの個数を変化させた場合の処理率の関係を示す．

32 ビット幅と 16 ビット幅のバスでは，独立動作可能なメモリが同一数である場合を比較すると，その差異は顕著である．これより，処理装置は 16 ビット幅ではあるが，スタックのような，メモリバスと処理装置間にバス幅の効率的な変換器を設けることのみでも，32 ビットバス幅メモリインタフェースは，その処理率を大幅に向上できることが明らかになった．

また，32 ビットで，独立に動作可能なメモリが1個の場合の処理率と，16 ビットで，独立動作可能なメモリが4個の場合の処理率が，DMA 転送の少ない時，ほぼ同一である結果が得られていることから，この両者のメモリスループットはほぼ同一であるとみなすことができる．見方を変えると，16 ビットの場合は独立動作可能なメモリ数を4個にしても，32 ビットインタリーブ無の場合のスループットしか得られないので，実質的には2ウェイインタリーブの効果しかないと言える．

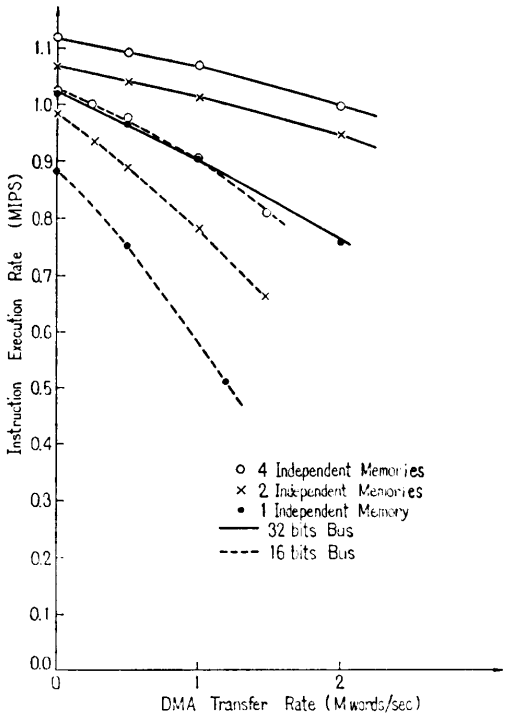


図 6 メモリ構成に対する処理率の変化
Fig. 6 Thoughput depending on memory structure.

一方，メモリスループットによって，DMA 転送が入った場合の処理率の低下の傾きにも特徴がある．すなわち，メモリスループットの大きい，32 ビットの独立に動作可能なメモリが4や2の場合には，DMA 転送が増加しても，処理率の低下は少ない．ところが，メモリスループットが更に低下すると，DMA 転送による処理率低下は急に大きくなる．これは次のように分析できる．各命令平均のメモリアクセスは，我々が考察している計算機では，1命令当り，平均約 1.3 回の命令語のアクセスと，約 0.7 回のデータのアクセスがある．命令のアクセスについては，32 ビットバス幅では，命令語のメモリに対するアクセス回数は約半分に減少できるから 0.7 回のアクセスとなる．DMA 転送が2M語／秒の時でも，平均命令実行数は約1MIPS (Million Instructions per Second) であるので，命令実行によるメモリアクセスは 0.7+0.7=1.4M 回，DMA は1M回で，合計すると1秒間に 2.4M 回のメモリアクセスがある．一方，メモリはサイクルタイム 450 ns であるので，2つのメモリが休みなく動作

すると，4.4M 回のメモリアクセスができる．

これより上記回数のメモリアクセスが起こっても，メモリが Busy となっているのは 60% 以下であり，メモリ要求の回数が少々増加してもメモリが Busy となる割合が増加するだけで，処理率の低下に影響する割合は小さい．

一方，独立に動作できるメモリが1つの場合や，メモリバス幅が 16 ビットの場合は，メモリが Busy となっている割合が 70% を越えるため，メモリ要求の回数の増加がそのまま処理率の低下に結びつく．

メモリインタフェースに対する性能向上率を求めると，16 ビットから，32 ビットバス幅に広げることにより，DMA 転送の入らない場合で約 10%，DMA 転送が1M語／秒の時，約 20% の向上が得られ，増加 IC は約 60 個であることから性能向上率は 0.17～0.33%/IC となる．

一方，インタリーブに対して，32 ビットで独立に動作できるメモリが1の場合と，2の場合で，やはり約 10～20% の向上が得られ，増加 IC は約 40 個であるので，性能向上率は 0.25～0.5%/IC となる．

5. むすび

計算機性能向上のため，パイプライン処理を導入し，パイプライン処理の効果，並びに，パイプラインを有効に動作させるためのスタック，およびメモリインタフェースの定量的評価を行った．その結果，命令先き読み用スタックに対しては，4層のスタックで必要十分であること，メモリバス幅を 32 ビットとし，2ウェイのインタリーブを行うことにより処理率の向上を図れることを明らかにした．

一方，ハードウェアの増加量も考慮し，総合的に判断すると，パイプライン処理，メモリインタフェースの高速化，スタックの順に処理率向上に効果のあることを明らかにした．

最後に，本研究を進めるにあたり，御指導，御鞭撻いただいた日立製作所中央研究所神内俊郎主任研究員，大みか工場桑原洋部長，日立研究所高砂常義所長，平沢宏太郎主任研究員に深謝します．

参考文献

1) Burnett, G. J., Coffman, E. G.: A Study of Interleaved Memory Systems, 1970 Spring Joint Computer Conference AFIPS, pp. 467-474 (1970).
2) Ravi, C.V.: On the Bandwidth and Interference in Interleaved Memory Systems, IEEE trans. Computers, Vol. C-21, No. 8, pp. 899-901 (1972).
3) Sastry, K. V., Kain, R. Y.: On the Performance of Certain Multiprocessor Computer Organizations, IEEE trans. Computers, Vol. C-24, No. 11, pp. 1066-1074 (1975).
4) Bhandarkar, D. P.: Analysis of Memory Interference in Multiprocessors, IEEE trans. Computers, Vol. C-24, No. 9, pp. 897-908 (1975).
5) Briggs, F. A., Davidson, E. S.: Organization of Semiconductor Memories for Parallel-Pipelined Processors, IEEE trans. Computers, Vol. C-26, No. 2, pp. 162-169 (1977).
6) Strecker, W. D.: Cache Memories for PDP-11 Family Computer, IEEE Computer Architecture, pp. 155-158 (1976).
7) Alexander, W. G., Wortman, D. B.: Static and Dynamic Characteristics of XPL Programs, IEEE Computer, Vol. 8, No. 11, pp. 41-46 (1975).

(昭和54年1月18日受付)
(昭和54年9月20日採録)