図形情報処理用マイクロプログラム制御計算機

小柳 滋，柴山 潔，富田真治，萩原 宏
（京都大学・工学部・情報工学教室）

1．まえがき

最近集積回路技術の進歩により、ＬＳＩ、ＭＳＩが安価で容易に手に入るようになり、大学、研究所のような所でも計算機を設計、製作することが比較的容易になってきた。　このような状況のもとに我々も図形処理の高速化を達成し、また他の種々の応用にも適用できるようなアーキテクチャを持つ高速な計算機を設計し、製作している。

我々の研究室では、オンライン知能端末の研究としてミニコンピュータを中心とした図形処理システムの研究を行なっている。　更に我々はミニコンシステムを用いて動的な図形処理を行なうことを検討しているが、動的な図形処理では配列演算の処理にぼう大な時間を費し、これをリアルタイムで行なうことはミニコンではほとんど不可能である。　そこで我々は千本のベクトルからなる図形の運動をリアルタイムで処理できる能力をもつ計算機を新たに設計してミニコンと結合して用いることを考えた。

なお、図形処理、画像処理、信号処理等をリアルタイムでおこなう専用マシンは、これまでにもいろいろ開発されている。　図形処理用としては Brown Univ. のＳＩＭＡＬＥ[5]，Argonne Univ. のＡＭＰ[6]等、画像処理用としてはＮＩＨ（National Inst. of Health）のマシン[7]，阪大のパターンプロセッサ[8]等，信号処理用としては慶大のＫＳＰ[2]等があり、各々アーキテクチャ上にユニークな特徴をもち、並列処理、パイプライン処理等を利用して高速化を達成している。

我々はこの計算機を設計するにあたって、これらのマシンと同様に並列処理により高速化をはかり、更に柔軟性のあるアーキテクチャによって種々の応用に役立てることをめざした。

本文では、以下本計算機のアーキテクチャとその特徴、ハードウェア構成、応用の検討について述べる。

2．アーキテクチャとその特徴

本計算機の概略図を図１に示す。　本計算機は４ケのＡＬＵ（算術論理演算装置）によるレジスタ・レベルでの並列処理方式と、ダイナミック・マイクロプログラミング方式により処理の高速化を達成している。　以下、この２つのアーキテクチャ上の特徴について説明する。

2．1　レジスタ・レベルでの並列処理方式

並列処理方式は、共有されるメモリの階層について次の３つのレベルに分類できる。

イ）レジスタ・レベル
ロ）主記憶レベル
ハ）チャネルを通したファイルのレベル

本計算機では、大量均一なデータのシンプルな演算の多いリアルタイム処理を主な応用目的としており、処理自体が本質的な並列性を含んでいる。　従ってイ）のレジスタ・レベルでの並列処理方式を採用し、４ケのＡＬＵがワークレジ

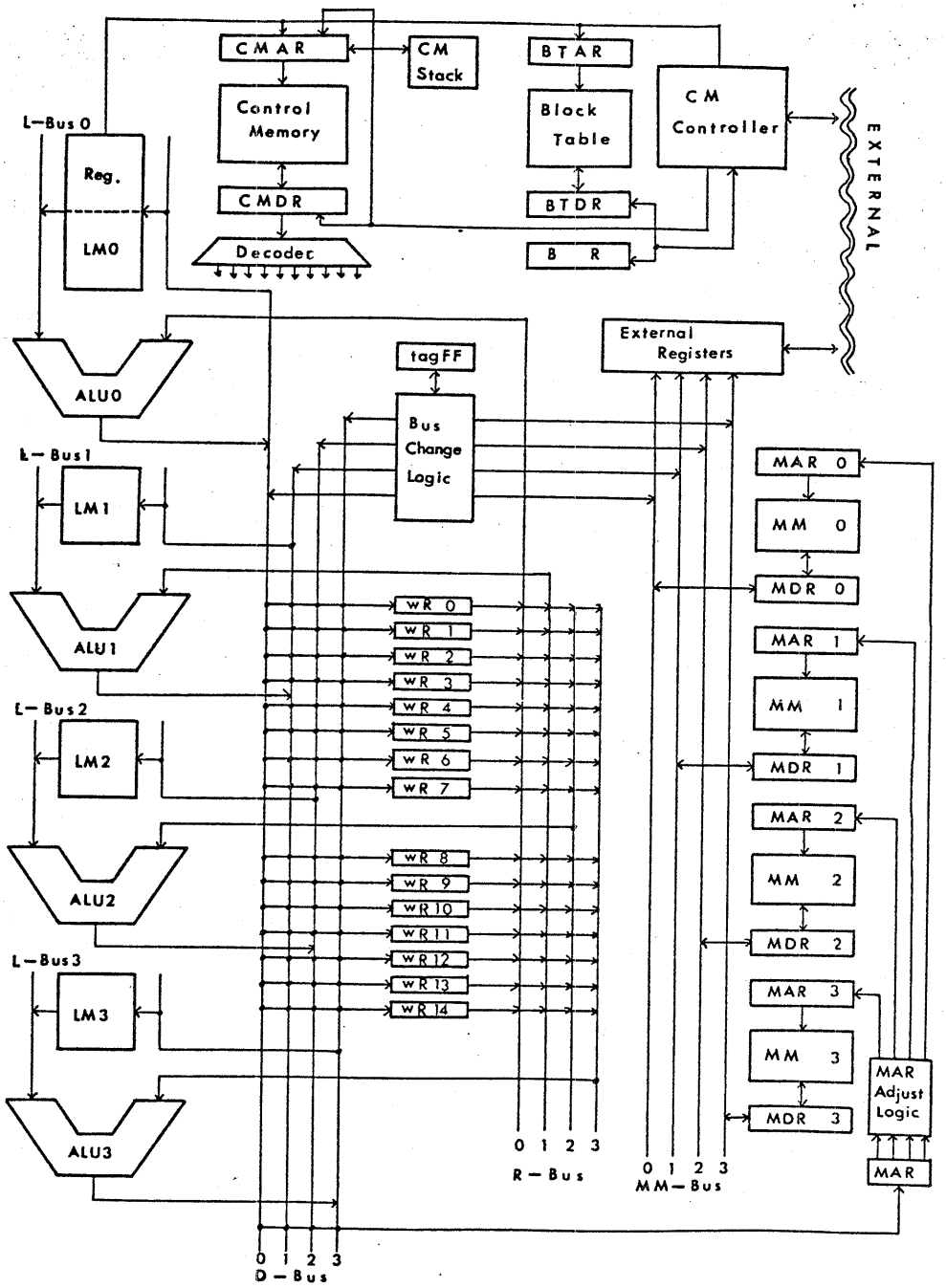


図1. 図形情報処理用マイクロプログラム制御計算機の概略図

スタを共有しつつ並列に動作する構成をとった。　また、4ケのALUを同一マイクロ命令の異なるフィールドで制御するMIMD方式（Multi Instruction Multi Data）を採用した。　マイクロ命令の順序制御は1ケ所に集中しているため、プロセスの同期問題は生じない。　また4ケのALUは同じ、あるいは異なるワークレジスタを同時にアクセスでき、レジスタの競合も生じない構成をとっている。

2．2　ダイナミック・マイクロプログラミング方式

処理の高速化を実現するオ2の方法として、マシン命令のレベルを上げ、計算機を各種応用に適応させることのできる可変構造にするために、書き換え可能な制御記憶をもつマイクロプログラム（以下μPと略す）制御方式を採用した。また並列処理の効果を生かすために、語長160ビットの水平型マイクロ命令形式とした。

1語160ビットは普通の計算機に比べかなりビット数が多く、コスト的に制御記憶の語数を大きくとることができない。　そこで我々はμ命令に2種の形式を設け、並列処理ができないときには1語80ビットとすることにより制御記憶の節約をはかっている。　また制御記憶を仮想記憶方式にして、必要に応じて制御記憶の入れ替えをおこなうことにより、制御記憶の容量を小さくしている。

3．ハードウエア構成

3．1　演算装置

ALUの演算はOP部で指定される。　各ALU毎に異なるOP部が制御するマイクロ命令レベルでのMIMD方式である。　このALUの特徴としては、ハードウェアによる16ビット×16ビットの乗算器が装備されていること、強力なシフト演算機能（1，2，3，4ビット）、ALU入力に対するマスク機能、16ビットの定数発生機能、定数演算機能、4つのALUを連結した最大64ビットまでの倍長演算機能等がある。　表1にOP部の機能を示す。

またALUの入出力に対してバイト毎の操作が容易に行えるようにM1，M2，M3部で図2のようなバイト演算機能をもつ。

上2桁 / 下4桁	φ	1	2	3
φ	NŌP	SUB	SRLN 1	SRAN 1
1	CŌNS	SUBD	〃 2	〃 2
2	MPYI	SUBC	〃 3	〃 3
3	MPYR	SCR	〃 4	〃 4
4	ANĐ	ISB	SLLN 1	SLAN 1
5	ŌR	ISBD	〃 2	〃 2
6	EXŌR	ISBC	〃 3	〃 3
7		SCL	〃 4	〃 4
8	AĐĐ	SRAĐ 1	SRLĐ 1	SRLC
9	AĐĐĐ	〃 2	〃 2	SRAC
A	AĐĐC	〃 3	〃 3	
B	ACR	〃 4	〃 4	
C		SLAĐ 1	SLLĐ 1	SLLC
D		〃 2	〃 2	SLAC
E		〃 3	〃 3	
F	ACL	〃 4	〃 4	IND

表1．OP部の機能

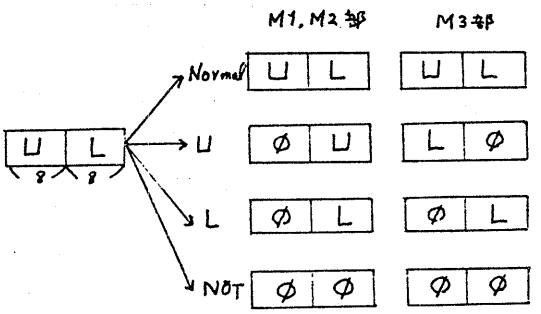


図2．M1，M2，M3部の機能

（略号については付録参照）

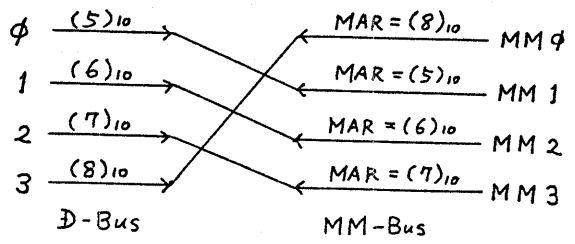
またＡＬＵ演算にはその実行速度にかなりの幅があるため、μ命令のＴＭ部でその終了タイミングを制御し、今後の演算器の付加等のハードウェアの拡張にも対応できるようにしている。

３．２　レジスタとローカルメモリ

水平型μＰ制御方式の利点を生かすために、我々はＡＬＵの演算を３アドレス方式にした。　各ＡＬＵには１５語のローカルメモリ（ＬＭ）が装備され、すべてのＡＬＵから共通にアクセスできる１５語のワークレジスタ（ＷＲ）が装備されている。　ＬＭ，ＷＲとも３１語まで拡張可能である。　また、ＡＬＵφには汎用レジスタ１６語と特殊レジスタ１６語が付加されている。　なおＬＭの一部としてＧＲＡＲ（General Register's Address Register）、ＭＲ（Mask Reg.）、ＥＸＲ((Extended Reg.)16ビット乗算の結果32ビットの上半分又は下半分を格納)がアドレス付けされている。　前述のようにＷＲは４ケのＡＬＵに共有されており、競合が生じないように構成されている。　複数のＡＬＵが同一ＷＲに書込む場合は、ＯＲされた内容が書込まれる。

３．３　主記憶装置

大量データの高速処理をおこなうためにはデータのアクセス時間が大きな問題となる。　そこで我々は主記憶を４バンクに分け、アクセス回数を減らした。主記憶にはサイクルタイム３５０nsのＭＯＳＩＣメモリを用い、各バンクにつき16 bit × 16 k語の大きさである。　我々は種々の応用を考慮して２通りの主記憶のアクセスモード（ノーマルモードとスペシャルモード）を備えた。　ノーマルモードでは４バンクの主記憶を連続したものと考え、指定された番地から始まる連続した４語が、先頭の語がＤバスφにくるように並び換えられてアクセスできる。　この並び換えは図３に示すようにＭＡＲの下位２ビットを用いてBus Change Logicがおこなう。　一方スペシャルモードではこの並び換えがおこなわれず、４バンクを別々のメモリと考えて使うことができる。



MARで10進5番地を指定した時5番地より始まる連続4語が図のようにD-Busにのる。

図３．Bus Change Logicの動作

また、ＭＡＲのセットの方法も次の３通りを考えた。(イ)ＡＬＵφ～３の各々の演算結果が必要とあればＭＡＲφ～３にセットされる。(ロ)ＡＬＵφ～３の内の高々１つの演算結果がすべてのＭＡＲにセットされる。(ハ)ＡＬＵφ～３の内の高々１つの演算結果が選択されたＭＡＲのみにセットされる。

また、外部Ｉ/Ｏとのデータ転送は、主記憶とのRead/Writeと同じレベルで取り扱う。　すなわち、計算機内部においては外部Ｉ/Ｏとのデータ転送の窓口であるＩ/Ｏレジスタが主記憶内でアドレス付けされている。　なお、主記憶装置は異種のメモリによる増設や、他の処理装置とのメモリ共有による競合を考慮に入れ非同期制御方式でアクセスされる。

3．4　タグ・ビット

Wigington のＬＩＳＴプロセッサ[10]、Rice Univ. のＲ－２[11]等で示されているタグマシンは計算機アーキテクチャにおける興味深いアイデアと考え、我々は１語16ビットに２ビットのタグビットを付加した。　このタグビットの制御をμＰでおこなえるように構成し、種々の応用に対してタグを利用できる。　我々はタグビットをマトリクス、トリー、スタック等の処理、デバッグ時の情報等に使用することにより、ソフトウェアの負担を軽減することをめざしている。

3．5　制御記憶

制御記憶はアクセス時間40nsのBipolar IC Memoryで構成し、160 bit×512語の容量を有する。　これは１Ｋ語まで拡張可能である。　制御記憶の容量があまり大きくないので前述のように仮想制御記憶方式を採用し、実行中に主記憶からμＰの入れ替えを行なうようにした。　制御記憶の入れ替えはＣＭ-Controllerが制御する。　この際μ命令は読み出しのみであること、実行されるマシン命令に対応するμＰは必ず用いられること等から、マシン命令毎にそのμＰが制御記憶に格納されているかどうかをチエックする方式を採用した。　この仮想制御記憶方式のパラメータを表２に示す。　なお、これらのパラメータはμＰの動作特性によって将来変更可能なように設計されている。

3．6　Flip-Flop, Test, Branchの制御

Flip-Flop（F.F）にはZero F.F, Carry F.F, Sign F.F, Overflow F.FがＡＬＵφに各々２ケ、ＡＬＵ１～３に各々１ケずつ付加されている。　またDisable F.Fが各ＡＬＵ毎に１ケ付加されており、Disable F.Fが１のとき、対応するＡＬＵの演算は抑止される。　その他いくつかのF.Fがあり、任意のF.Fの出力に応じて２way Branchすることができ、２ケのF.Fの出力を組合せて４way Branchもできる。　また制御記憶のアドレスStackを用いたサブルーチンコール、特殊レジスタの内容による間接Branchもできる。

容　量	512語
mapping	full associative
block数	8ケ
block size	64語
replacement algorithm	LRU

表２．仮想制御記憶のパラメータ

3．7　マイクロ命令形式

前述のように制御記憶の使用効率をよくするためにμ命令に２つの形式、即ち４ケのＡＬＵの並列処理の場合のLong format（160ビット）と、ＡＬＵφのみの演算を制御し、他のＡＬＵを用いないShort format（80ビット）を備えている。　このLong formatとShort formatは制御記憶の１語160ビットの先頭ビットにより区別される。　この２つのμ命令形式を図４に示す。

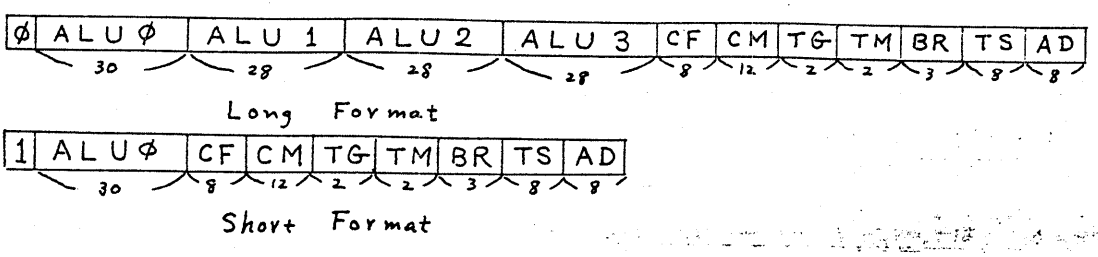


図４．　マイクロ命令の形式

図中、ＣＦ部はＦlip-Flop の制御、ＣＭ部は主記憶の制御、ＴＧ部はタグビットの制御、ＴＭ部はタイミングの指定、ＢＲ部はブランチ形式の制御、ＴＳ部はテスト条件の指定、ＡＤ部は次のμ命令のアドレスの指定をおこなうフィールドである。　また各ＡＬＵの制御部を図５に示す。

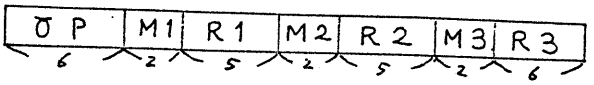


図５．　ＡＬＵの制御部

Ｒ１、Ｒ２、Ｒ３部はそれぞれオ１、オ２、オ３オペランドのアドレスを指定する。　但し、オ１オペランドはＬＭ、オ２オペランドはＷＲと限定される。Ｍ１、Ｍ２、Ｍ３部は前述のようにＡＬＵの入出力に対してバイト処理をおこなう。　なお、ＡＬＵのＬＭには汎用レジスタと特殊レジスタが付加されているので、ＡＬＵのＲ１、Ｒ３部は共に１ビットずつ多くなり、それぞれ６ビット、７ビットとなる。

３．８　マイクロ命令の実行

１マシンサイクルは基本的に７ケのクロック・パルス（$t_0 \sim t_6$）により構成される。　各クロックのパルス幅は２５n.sであり、１マシンサイクルは基本的に３５０n.sである。　ＡＬＵの演算と次のμ命令のfetchはoverlapされている。以下、各クロックの役割を示す。

t_0：制御記憶より読み出されたマイクロ命令をμＩＲ（マイクロ命令レジスタ）にセットし、デコードする。

t_1：ＷＲ、ＬＭを読み出し、それぞれR-Bus、L-Busにのせる。

t_2：R-Bus、L-Busのデータをａｌｕの入力ラッチにセットする。

t_3：ＡＬＵ演算の開始
　このクロックはＴＭ部により制御される。

t_4：ＡＬＵ演算の終了
　次のμ命令のアドレスをセットする。

t_5：ＡＬＵ出力をD-Busにのせる。
　制御記憶にRead信号を送る。

t_6：D-Busの内容をＬＭあるいはＷＲにストアする。
　各種Ｆ.Ｆをセット、リセットする。

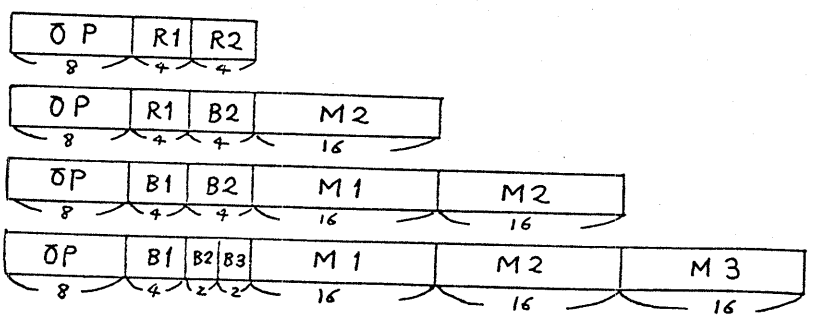


図６．　マシン命令の標準形式

4. 応用の検討

4.1 汎用

本計算機は4ケのALUを使用して、本質的に並列処理できる応用に対して特に効力を発揮するが、汎用計算機として使用することもできる。 現在の所、マシン命令の形式として図6の標準形式を考えている。 もちろんμP制御であるので任意の形式のマシン命令を作ることができる。 この標準形式のマシン命令のfetchをおこなうμPの例を図7に示す。 ここで1つのBoxが1マイクロ命令に対応する。

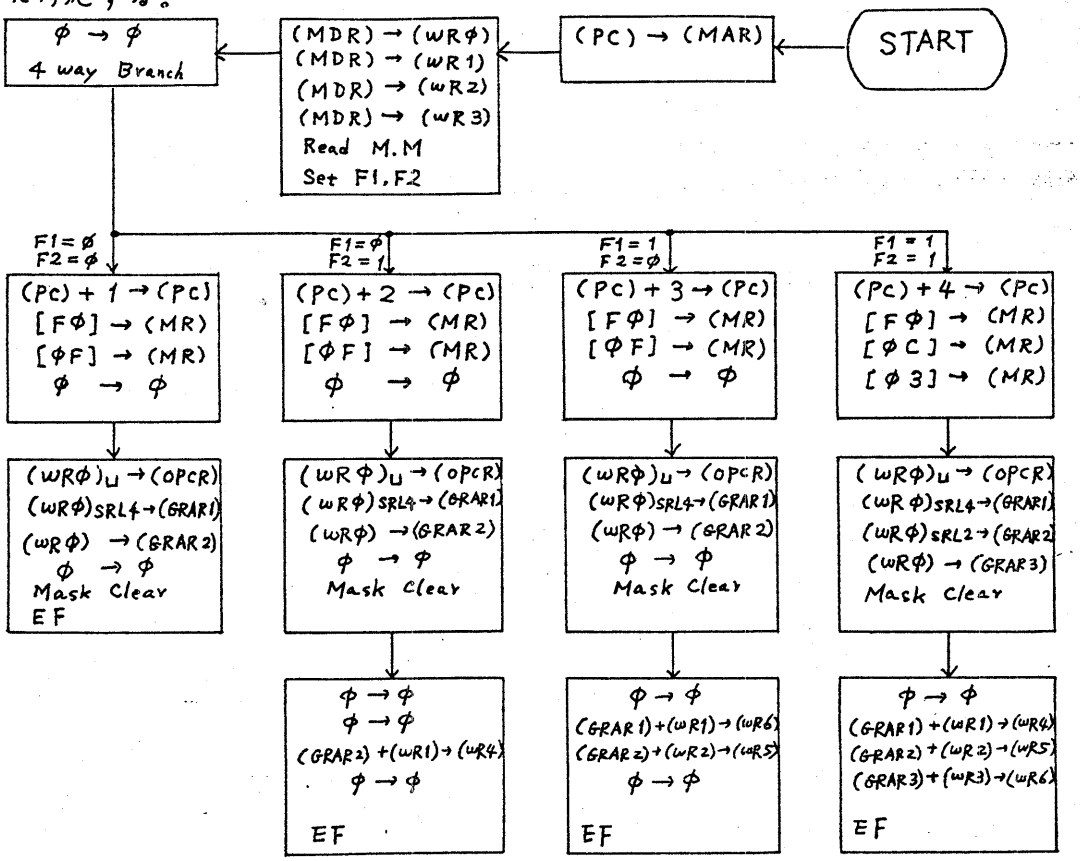


図7. マシン命令fetchのマイクロプログラムの例

4.2 図形処理

図形の運動を処理するには、3次元図形の場合パラメータを導入した4次元の同次座標系を用いると、4×4の変換行列を作用させることで可能である。 すなわち平行移動、拡大、縮小、回転等が次の計算を必要とする。

$$(x, y, z, 1)\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \longrightarrow (x', y', z', 1) \qquad (1)$$

また、Display上で図形の運動がFlickerなしに認識されるためには、30～50回/secの繰返し周波数が必要であり、千本のベクトルより成る図形をリアルタイムで変換するためには、1座標点当り20μsecで処理しなければならない。

(1)の計算は一般に乗算16回、加減算12回を必要とする。座標値x, y, zを16ビットの整数データ、a_{ij}を整数部16ビットと小数部16ビットの32ビットのデータとすると、本計算機ではALU1ケを用いると乗算1回につき4stepを要する。また加減算は48ビット長となるので1回につき3stepを要する。従って本計算機では4ケのALUをfullに使うと座標点当り演算時間が

$$\frac{1}{4}(4\times 16 + 3\times 12) = 25 \ \text{step}$$

を要する。1stepが平均400nsとして他のオーバーヘッドを考慮しても、1座標点の変換が20μsで充分処理可能である。

その他図形の運動処理に必要な機能として、4×4行列の和，差，スカラー倍，積，要素へのアクセス，4次元ベクトルの和，差，スカラー倍，内積，外積等の配列演算も同様に高速化できる。このような演算では4ケのALUはほとんどfullに使用され、ハードウェアによる乗算器、マイクロプログラミング、並列処理の特徴が生かされて、ミニコンと比較して数十倍効率がよくなるであろう。

4．3　画像処理

画像処理においては、ぼう大な画像データに対して比較的単純な処理をおこなうことが多い。リアルタイムでの画像処理においては、特に画像の各点における局所的な（3×3あるいは5×5メッシュ）隣接点間の操作を高速化させる必要がある。我々の計算機を用いてm×n点よりなる画像に対して3×3の局所的な処理をおこなう例を示そう。

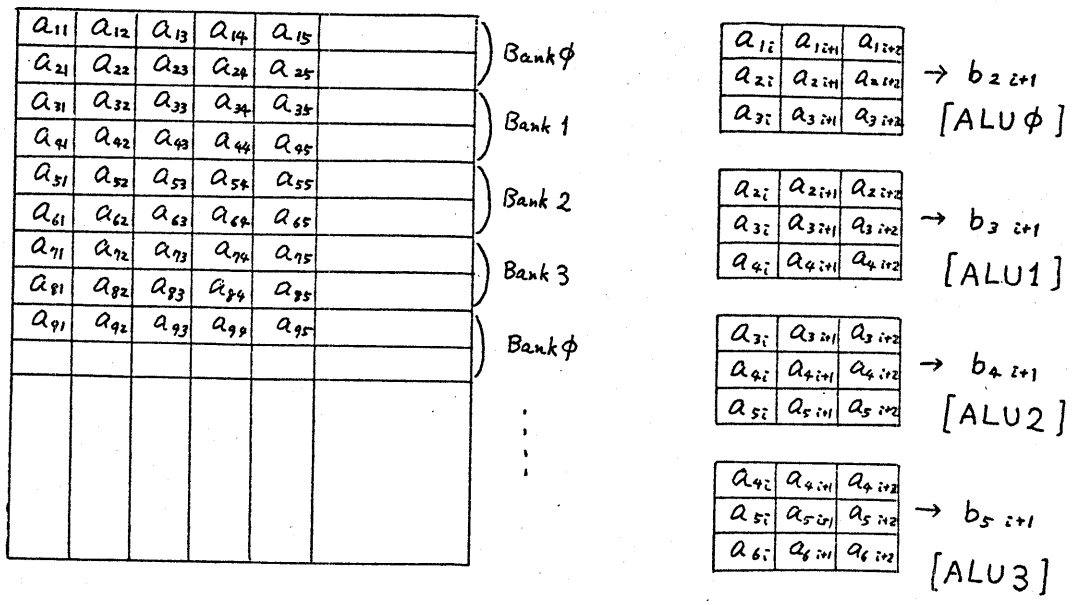


図8．画像処理の例

今，8ビットの濃淡画像の処理とすると，1語に2点が格納できる。図8で$a_{2i-1,j}$と$a_{2i,j}$とが1語に格納されるとする。また主記憶の4ケのBankに図8のように格納されるとする。3×3の局所的な処理をa_{ij}を中心とするメッシュにおこなって、その結果をb_{ij}としよう。本計算機では4ケのALUを用いて図8のようにb_{ij}, $b_{i+1,j}$, $b_{i+2,j}$, $b_{i+3,j}$の処理を同時におこなえる。この時必要なデータは大部分が重複しており、9語のWRに格納される。前述の

ようにＷＲの使用に競合がおこらないので、４ケのＡＬＵはfull に動作する。また、引き続く処理に新たに必要となるデータは３語に格納されており、１回の主記憶アクセスで得られる。　このように演算、主記憶アクセス共に効率よくなり、またμＰも同じ計算の繰返しであるので比較的簡単にできる。　２×２、５×５等の局所的処理についても同様に非常に効率よくできる。

４．４　その他の応用

上記の応用の他にも、次のような応用を検討している。

イ）：tag bit の利用による任意の大きさの配列演算をおこなうこと
ロ）：ＦＦＴ等の信号処理への応用
ハ）：tag bit の利用によるＬＩＳＴ処理への応用
ニ）：Parallel Search，Parallel Sort への応用
ホ）：各種エミュレーションに必要なビット処理，フィールド処理

以上の他にも種々の応用が考えられるであろう。

なお、本計算機がμ命令レベルでのＭＩＭＤ方式を採用している効果について考えてみる。　図形処理，画像処理においては、ほとんどの場合４ケのＡＬＵは同じ演算をおこなっている。　従って図形処理等の専用マシンではＳＩＭＤ方式をとるものが多い。　我々の場合、種々の応用に対してμＰを検討したが、同じμ命令で各ＡＬＵが異なる演算をおこなうことによって効率をよくする例が、かなりの場合生じた。　例えば倍長の乗除算、浮動小数点演算、初等関数の計算、等があり、また図形処理，画像処理においてもアドレス計算と他の計算とが同時にできることがあり、ある程度効果がある。　従って、本計算機の設計方針から考えても、ＭＩＭＤ方式を採用して、より広い応用に対して適用させることが妥当であると考える。

５．　むすび

以上、我々の研究室で現在開発中の計算機のアーキテクチャ，ハードウェア，並びに応用について概説した。

なお、本計算機ではマイクロ命令レベルで並列処理をおこなうため、普通の計算機の場合以上に強力なマイクロプログラムの作成支援システムを考えており、次のような機能をもつマイクロアセンブラの作成を検討している。

イ）：フィールドの数が多いため、フィールド・センシティブな記述ではユーザに負担がかかる。　フィールドを意識せずにマイクロプログラムを記述することを検討している。
ロ）：マイクロプログラムの読み易さ、書き易さのために、高級言語風な記述にしたい。　現在の所、レジスタ・トランスファー・レベルでの記述を検討している。
ハ）：簡単なマクロ機能をもたせる。
ニ）：配列演算等では、特定のフィールドのみが規則的に異なる一連のマイクロ命令が多いため、反復アセンブル機能をもたせる。
ホ）：既に作成されたμＰを会話型で修正できる機能をもたせる。
ヘ）：ハードウェアの変更、追加に対して柔軟に対応できるようにする。
ト）：μＰ作成用の高級言語の設計へのステップとする。

また、ハードウェア設計の検査、μＰのアルゴリズムの検査のためのシミュレータも作成を検討しており、アセンブラ，ローダ等の基本ソフトウェアの作成も

ハードウエアの設計と並行しておこなう予定である。
現在、本計算機の論理設計が完了し、実装設計にとりかかっており、本年度中には完成する予定である。

参考文献

(1) 柴山,小柳 富田,萩原 "図形処理のためのハードウェア構成", 情報処理15回大会予稿 160 1974.12

(2) 柴山,小柳 富田,萩原 "図形処理用マイクロプログラム制御計算機", 通信学会全国大会 予稿 1275 1975.3

(3) 富田 "インテリジェント・サテライト用計算機のアーキテクチャ", 京大工学部情報工学研究談話会 23 1975.3

(4) 柴山,富田 萩原 "高速データ・チャネルによる計算機結合の一方式", 通信学会電子計算機専門委員会資料 EC 74-17 1974

(5) H.Webber "SIMALE",Sigmicro Vol.3,4 1973

(6) V.Tantillo "A research oriented dynamic microprocessor",IEEE Trans. C-22,11 1973

(7) P.Iemkin "Realtime picture processor",1973

(8) 谷内田,富田 辻 "3次元物体の高速パターン処理システム", 信学論(D) Vol. 58-D, No.4 1975

(9) 相磯 他 "高速汎用信号処理装置KSPのマイクロプログラム制御", 情報処理学会 計算機アーキテクチャ研資 74-1 1974

(10) R.Wigington "A machine organization for a general purpose list processor",IEEE Trans.EC 1963

(11) E.Feustel "The Rice reserach computer-A tagged architecture",SJCC p.369 1972

付録

1. 表1の略号の説明

NOP : No OPeration
CONS : 16 bit 定数の生成
MPYI : MultiPly Integer
MPYR : MultiPlY Real
ADDD : Add Double
(下位のALUからケタ上げがくる)
ADDC : Add with initial carry = carry FF
ACL : Add 4 bit Constant from Right input
SUB : (オ1オペランド) − (オ2オペランド)
ISB : (オ2オペランド) − (オ1オペランド)
SRAN : Shift Right Arithmetic (算術長)
SLLD : Shift Left Logical (倍長)
SRLC : Carry FF を用いる Shift Right Logical
IND : Indirect Operation

2. 図2の説明

BTAR : Block Table Address Reg.
BTDR : Block Table Data Reg.
BR : Block Reg.

BTDRとBRの内容を調べてマシン命令に相当するμPが制御記憶に格納されているかどうかを決定する.

3. 図7の説明

F1, F2 : D Bus Φの先頭2ビットが保持される.
[FΦ] : 16進定数 11110000
(MR) : Mask Reg.
(OPCR) : Operation Code Reg.
EF : End of Fetch.
この時 次のμ命令のアドレスはOPCRにより決定される.