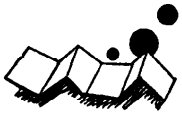
解　説

2. マイクロプログラマブル・アーキテクチャ



2.1 マイクロプログラム制御方式の動向†

富　田　眞　治††

1. はじめに

マイクロプログラム制御方式は1951年ケンブリッジ大学の M. V. Wilkes が EDSAC の設計の経験を生かして提案して以来，着実に発展してきた．1980年ごろまでのマイクロプログラム制御方式については本学会の特集号 (1973)，萩原宏著「マイクロプログラミング」(1977)，相磯，飯塚，坂村編「ダイナミック・アーキテクチャ」(1980) などに詳細にまとめられている[1)~3)]．主要技術は1980年ごろまでにほぼ出尽くした感もあり，また RISC ではマイクロプログラム制御方式を採用しなかったため[42)]，ACM の Sigmicro (マイクロプログラミング専門委員会) の雑誌 (1984) でも "Is Microprogramming Dead?" というショッキングな見出しでマイクロプログラム制御方式の研究現状を悲観的な目で眺めた時期もあった[4),5)]．マイクロプログラム制御方式に関する論文はマイクロプログラミング・ツールなどの一部の分野を除いて確かに減少している．これは，

(a) 研究者が応用指向のアーキテクチャの研究発表を行うことが多くなっている．マイクロプログラム制御方式は基盤技術となっており，それ自体として論じられることが研究発表としては少ない．

(b) VLSI 時代を迎えて，マイクロプログラム制御方式はメーカにとって一つのノウハウとなっており，メーカ側から発表されることが少ない．

などによる．

しかし，マイクロプログラム制御方式には，VLSI 設計との関連で解決すべき多くの問題点が残されている[6),7)]．本稿では，主として1980年以降に提案された種々の方式のうち，水平型マイクロプログラム制御方式から派生した VLIW (Very Long Instruction Word) 方式，分散型マイクロプログラム制御方式，制御記憶の小量化方式について述べる[8)]．

† Trends in Microprogrammed Computers by Shinji TOMITA (Dept. of Information Systems, Interdisciplinary Graduate School of Engineering Sciences, Kyushu University).

†† 九州大学大学院総合理工学研究科

2. 超長形式命令 (VLIW) 型計算機

本章では，VLIW 方式の原理と特徴について述べ，2，3の代表的なシステムを紹介する．また強力なコンパイラによる並列性検出と機械命令への操作の埋込みが性能に重要な役割を果たすので，トレース・スケジューリングと呼ぶ最適化方式について言及する．

2.1 原理と特徴

VLIW 方式では図-1 に示すように，比較的長い命令を多数のフィールドに分割し，おのおののフィールドで多数の演算器，レジスタ，相互結合網，メモリなどが独立に制御される．水平型マイクロ命令形式と比較すると演算装置などが多数装備されており，各フィールドで指定できる機能が高機能化されている．しかし，もともと水平型マイクロ命令形式から派生したものであり，明確な区別があるわけではない[9)]．この方式には次のような特徴があり，表-1 に示すように多数のシステムが研究開発され，一部製品化されている[10)~22)]．

(1) コンパイル時の並列性検出

データ駆動計算機や IBM 360/91 などが実行時に並列演算の可能性を判定するのに対して，VLIW 方式ではコンパイル時になされる．コンパイラはソース・プログラムから並列演算可能なものを抽出して，一つの命令に合成する．並列演算器数に近い並列度が

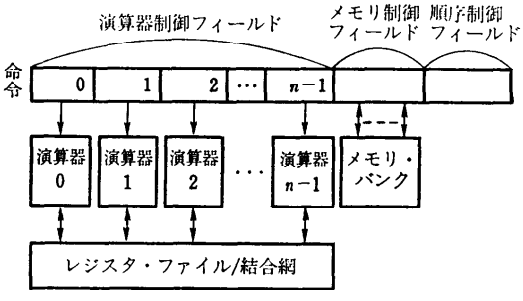


図-1 VLIW 方式

表-1 VLIW 方式の計算機

名称	発表年	開発機関	応用分野	命令ビット長	命令実行サイクル(ナノ秒)	並列ALU演算数	ALU の種類と特徴	文献
AMP	1973	Argonne National Laboratory	リアルタイム処理(グラフィックスなど)	74	430	2	・8ビットおよび16ビット整数演算器	13
QA-1	1976	京都大学	図形・画像・信号処理，高級言語処理	160	350	4	・16ビット整数演算器(乗算器付き)	10
AP-120B	1976	Floating Point Systems	アレイ演算，信号処理(FFT 演算など)	64	167	3	・38ビット浮動小数点加算器および乗算器(パイプライン制御) ・16ビット整数演算器(逆2進変換器付き)	14, 15
Vanguard (1100/60)	1979	Univac	汎用	283	116	2	・36ビット汎用演算器 (MC 10800 を使用)	16
MUNAP	1979	宇都宮大学	非数値処理(記号処理など)	188	550	4	・16ビット非数値処理用プロセッサユニット(2レベルマイクロプログラム制御)	17
QA-2	1983	京都大学	図形・画像・信号処理，高級言語処理	256	600	4	・8/16/24/32ビット整数演算器(乗算器付き)	11, 12, 52
ELI-512	1983	Yale Univ.	科学技術計算	500以上	—	20～30	・32ビット整数演算器 ・64ビット浮動小数点演算器 (Trace Scheduling によるコード・コンパクション)	18
ベクトルプロセッサ	1984	日立製作所	(ミニコン内蔵型)ベクトル演算	96	167	3	・32ビット浮動小数点乗算器(パイプライン制御) ・64ビット浮動小数点加算器(パイプライン制御) ・32ビット整数演算器	19
CYBER-PLUS	1984	CDC	科学技術計算	240	20	8	・8/16/32 ビット整数加算器 (2台)，整数乗算器(1台)，および論理演算器 (2台) ・32/64ビット浮動小数点加算器，乗算器，および除算(平方根演算)器 (パイプライン制御，各1台)	20
MC	1985	松下電器産業無線研究所	グラフィックス	96	250	5	・浮動小数点加算器および乗算器 ・関数発生器(三角関数，平方根，逆数など) ・浮動-固定小数点変換器 ・24ビット汎用演算器	21
SIGHT	1985	NTT	グラフィックス	96	600	3	・浮動小数点演算器，関数テーブル	55
GF-11	1985	IBM	科学技術計算	180	50	5	・浮動小数点乗算器2台 ・浮動小数点加算器2台 ・整数演算器　1台(各 PE 当たり) ・SIMD 型計算機 (576 PE)	56
CHoPP	1987	CHoPP 社	科学技術計算	256	60	8	・浮動小数点演算器4台 ・アドレス計算用整数演算器4台	22

得られるとき，高速処理を達成できる．しかし，並列度が低いときには，演算器制御フィールドなどに遊びができ，命令のビット使用効率が低下する．並列演算の度合はコンパイラの能力と応用自体に含まれる並列性の度合に依存している．

(2) シンプルなハードウェア構成

並列性の検出がコンパイル時になされるので，実行時に複雑な制御（たとえば，データ依存関係の検出など）をする必要がないので，ハードウェア構成を簡潔化することができる．

(3) 低粒度並列処理に適合

VLIW 方式では，演算器間には多数のバスが張られ，それらのバス制御を命令のフィールドで直接制御できるので，通信オーバヘッドを小さくできる．機能レベルの低い演算器によるきめ細かな並列処理（低レベル／低粒度並列処理）に向いている．

(4) 市販のビルディング・ブロックの利用

Weitek 社や TRW 社の浮動小数点演算器やAMD 社や TI 社のビット・スライス・マイクロプロセッサなど，市販されているビルディング・ブロックを利用でき，コスト性能比のよいシステムを構築できる[23]．

超長形式命令に多数の操作が埋め込まれて初めて高速化を達成できるシステムであるので，強力なコンパイラの開発とともに，次に示すような並列操作の埋込

みを容易にするようなハードウェア設計が必要となる[11].

(1) 均質の並列演算機構

複数台の演算器やレジスタの構成が非均質になされていると並列演算の本来可能なものが長形式命令に埋め込めないことが起こるので，可能なかぎり均質な演算構造が必要である．また均質構造はコンパイラのコード生成を容易にする．

(2) 逐次演算の高速化

データ依存関係にある操作をたとえば演算器チェイニングなどの機能を導入して同一命令で指定できるようにすると，高速化を達成でき，また命令のビット使用効率が向上する．

(3) 高機能・構造化順序制御機構

一つの機械命令で同時に多数の演算操作を実行すると，多様なステータスが生成される．VLIW 方式では集中制御方式のもとで実行を行うので，多様なステータスの組み合わせに対する柔軟な分岐制御が必要である[52]．またプログラミングの容易さやプログラムの再利用を可能とするような構造化(マイクロ)プログラミングに対するハードウェア支援が必要である[8),53)]．

2.2 システム例

表-1 に示す代表的なシステム例のうちいくつかのシステムについて紹介にする．

(1) QA-2

QA-2 は個人利用を主体とした高性能な計算機として設計され，図形・画像・音声など多様な情報媒体によるマン・マシン通信のための高速処理機能，各種高級言語に対する効率のよい処理系を提供できる．

QA-2 は 256 ビットの命令形式を有しており，4つの異なる算術論理演算 (ALU) 制御，4つの異なる主記憶アドレスへのアクセス制御，一つの高機能順序制御を同時に指定できる．先に示した設計指針を十分取り入れている．3次元グラフィックスや Prolog 処理系など異なる応用で平均的にみて3台の ALU が並列に動作可能であると報告されている[10)~12)]．

(a) 並列演算装置の構成

並列演算部 (RALU) は同一構造をした4つの算術論理演算装置 (ALU), 6k バイトの大容量共有レジスタ・ファイル，ALU チェイニング網 (ACN) から構成されている．レジスタ・ファイルは各 ALU からの同時アクセスを可能とするため，8重系のECL RAM チップで構成されている．ACN は，ある ALU の出力を直接他の ALU の入力ラッチにセットするための完全網による ALU 結合網である．したがって，$(A+B)*C+D*E$ のような演算を1命令ステップで実行でき，逐次処理の高速化も達成できる．

(b) 高機能・構造化順序制御装置

IF f((A0), …, (A7))
THEN(CALL A or GOTO B or RETURN)
ELSE(CALL C or GOTO D or RETURN)

の形式をした2方向分岐，

CASE((A0), …, (A7))OF

の形式の多方向分岐をサポートしている．ここに，f はユーザ定義可能な論理関数，(Ai) は 256 ビットのステータス・セーブ・レジスタ内の論理変数を表す．

QA-2 では，図-2 に示すように，RAM による制御表を多数用いている．8個の論理変数に対する論理関数は真理値表を蓄える 256 ビットの RAM で実現される．RAM の最上位アドレス (11, …, 1) のみ1，そのほかは0のとき，この真理値表は AND を表し，最下位アドレスのみ 0，そのほかは1のとき OR を表

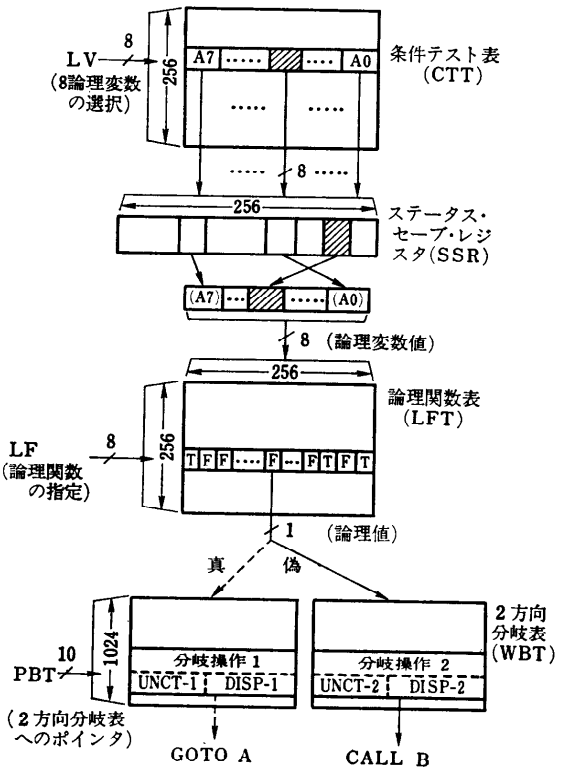


図-2 QA-2 の高機能順序制御方式

す．QA-2 の論理関数表 (LFT) は 256 エントリ用意されており，命令の LF フィールドで選択される．8 個の論理変数は次のようにして求められる．図-2 の条件テスト表 (CTT) には，256 エントリが用意されており，各エントリは 8 個の 8 ビット・フィールド A0, …, A7 からなる．各 Ai はステータス・セーブ・レジスタ (SSR) のアドレスを指定している．したがって，命令の LV フィールドで一つのエントリを選択すると，この A0, …, A7 を用いて，SSR より 8 個の論理変数が同時に読み出される．これらの論理変数値は論理関数表の列アドレスを指定する．

(2) AP-120 B とその後継機

Floating Point Systems 社の AP-120 B はミニコンなどに付加して用いられる商用科学技術用計算機である．最大性能は 12 MFLOPS である[14),15)]．同社の後継機 FPS-164 や最新鋭機 FPS-T シリーズ（2 進 n-キューブ網に最大 16384 台の 16 MFLOPS のプロセッサを結合したマルチプロセッサ・システム）でもほぼ同様なアーキテクチャが採用されている[24)]．浮動小数点乗算回路，加算回路，作業用レジスタとしてのデータパッド (X, Y), 汎用演算装置 (ALU), 定数格納用のテーブル・メモリ (TM), データメモリ (MD) が多重バスで結合され，これらが 64 ビット長の命令で同時に制御される．

(3) CHoPP

CHoPP 社の計算機はコロンビア大学で設計されたもので，16 台構成のマルチプロセッサである[22)]．ハードウェア規模が同一と考えられる 4 プロセッサ構成の CHoPP と 1 プロセッサ構成の Cray X-MP のリバモア・ループに対する性能評価を図-3 に示す．ベクトル化が困難なループに対して CHoPP が優れた性能を示すことが報告されている．

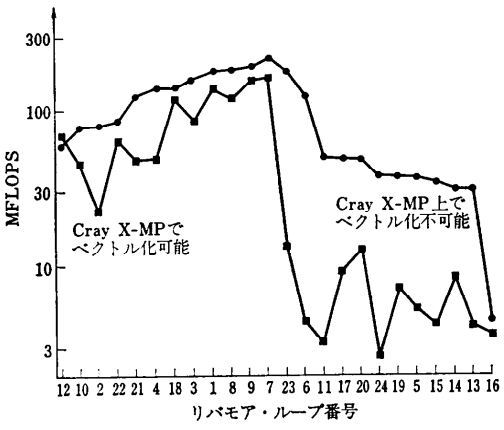


図-3 CHoPP の性能評価[22)]

(4) MC

松下電器無線研究所の MC は 3 次元グラフィックス専用計算機であり，視線探索法による高品質画像の高速生成や実時間で入力されるビデオ画像を球面などにマッピングする特殊効果装置として用いられている[21)]．

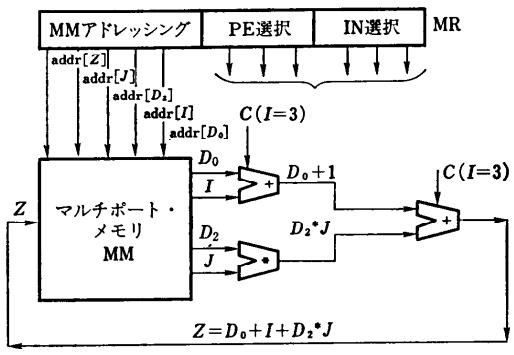
(5) そ の 他

キール大学の P. Marwedel らは VLIW 方式をシリコン・コンパイラ (MIMOLA) のターゲット・アーキテクチャとしている．Fortran で記述された多数の数値計算用サブルーチンを MIMOLA 言語に変換し，それをさらに VLIW 方式のターゲット・アーキテクチャ（112 ビット命令で 5 算術演算，9 メモリポート，1 順序制御回路を制御）にコンパイルした結果，中規模ミニコンの 25 倍の処理性能を達成できたと報告されている[25)]．

パーデュ大学の V. Milutinovic らは VLIW 方式を GaAs アーキテクチャとの関連で検討している．GaAs ではチップ内でのゲート遅延に比較してチップ外での遅延が大きいため，外部のメモリチップからの命令供給速度は GaAs による演算系の速度と比較して非常に遅くなる．このため，チェイニングなどによって複雑な演算を 1 命令で定義し演算時間を長くし，命令供給速度と演算速度のバランスを取っている．図-4 に VM システムを示す[26)]．QA-2 の演算系と類似している．

2.3 最適化コンパイラ

VLIW 方式では，一つの命令にどれだけ多くの基本演算操作を埋め込めるかによって性能が左右され



MM: マルチポート・メモリ
PE: 演算器
IN: 結合網
図-4 VM システムの構成

る．従来，マイクロプログラムの最適化手法の研究としてこの領域の研究が盛んに行われた[27),28)]．最適化手法には，局所最適化と広域最適化の二つの方法がある．基本演算操作の系列で，その出口を除いて分岐操作がなく，かつその入口を除いて外から分岐されることのないものをブロック（またはセグメント）と呼ぶ．局所最適化はこのブロック内でデータの依存関係を調べて並列実行可能な基本演算操作を検出して，VLIW 命令として合成する．広域最適化では，ブロック間で基本演算操作の移動をともなった最適化を進める．しかし，VLIW 型計算機では条件分岐命令の頻度が非常に多く，ブロック長が非常に短いこともあって，最適化によりそれほど大きな効果は得られていない．

エール大学の J. A. Fisher らは条件分岐に非常に偏りのある場合（科学技術計算などの応用）について，トレース・スケジューリング法と呼ぶ興味ある最適化手法を提案している[29),31)]．**図-5** に概要を示す．図(a)～(d)の四角で囲ったものがブロックであり，この中に基本演算操作が並んでいる．プログラムのフロー構造を図(a)に示す．最適化手順は次のようになっている．

（1） このフロー・グラフを解析（サンプル・プログラムの実行などによって）して，図(b)の斜線部に示すパス（これをトレースと呼ぶ）が非常に高い確率で実行されることを調べる．

（2） このトレース内でデータの依存関係に注目して，並列実行可能な演算操作を検出して，逐次，VLIW 命令として合成していく．この過程でトレース内に存在する基本演算操作を移動する．

（3） トレース内での基本演算操作の移動にともなって，図(d)に示すようにトレース外のコードの修正が必要となる（ブックキーピング処理という）．

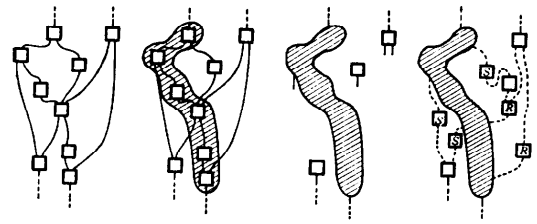
（4） 修正したトレース外のコードに対して上述の手順を繰り返す．

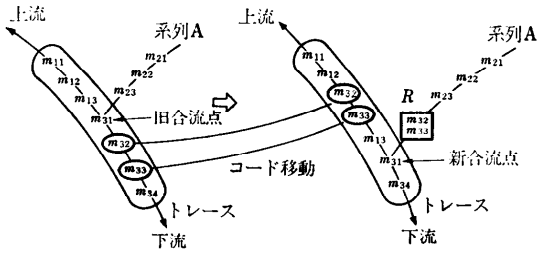
(3)のトレース外のコードの修正は以下のように進められる．

図(e)に示すようにトレース上でコード移動が行われ，$m_{11}, m_{12}, m_{13}, m_{31}, m_{32}, m_{33}, m_{34}$ が $m_{11}, m_{12}, m_{32}, m_{33}, m_{13}, m_{31}, m_{34}$ となったとしよう．このとき，新合流点を，その下流に旧合流点の上流にあるトレース上の演算操作がない点に定める．図(e)の系列Aは新合流点で合流する．しかし，旧合流点の下流にあったもの（ここでは，m_{32}, m_{33}）が新合流点の上流に移動しているので，このままでは系列Aからのパスで m_{32}, m_{33} が実行されないことになる．そこで，系列Aに m_{32}, m_{33} を付加する（図(e)の R）．

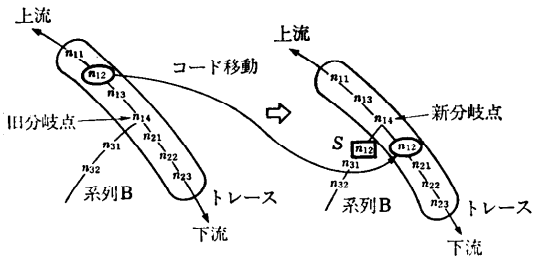
一方，分岐点では図(f)のようになる．旧分岐点の上流にあった演算操作が新分岐点の下流にコード移動されたとき（この場合，n_{12}），系列Bに n_{12} を付加する(図(f)の S)．

このように，トレース・スケジューリング法では，非常に長いトレース上の基本演算操作列を対象として最適化がなされるので，その効果は大きい[30)]．また，B. Su らはループ構造などに対するより効率のよい方式を提案している[32)]．





（e） 合流点での処理



（f） 分岐点での処理

図-5 トレース・スケジューリング

3. 分散形マイクロプログラム制御方式

計算機システムに対する一層の高速化を達成し，また VLSI による高集積化に対処するため，分散型マイ

クロプログラム制御方式が注目されてきた．本章では分散型マイクロプログラム制御方式の原理と特徴および機械命令の実行モデルについて述べ，2，3のシステム例を紹介する．

3.1 原理と特徴

計算機システムの設計にあたっては，高速性，拡張性，柔軟性，さらにVLSIシステムの設計では配線量やピン数なども考慮しなくてはならない．従来のマイクロプログラム制御計算機では，マイクロプログラムを格納する制御記憶(CS)は集中管理されていた．図-6の集中方式-1では，CSから読み出されたマイクロ命令がデコード(D)され，それらが各機能装置(FU)に伝えられている．デコーダ出力の配線数は非常に多くなり，また配線の不規則性も増す．集中方式-2のように各FUに近い所で局所的にデコードすると配線量を減らすことができる．これは特にVLSI設計において重要である．しかし，この方式でもマイクロ命令のバス幅は大きく，また中央の主制御装置(MCU)は各FUのステータス情報に基づいて制御するので伝搬遅延時間が大きくなり，実行サイクルを短くできない．

分散方式-1では各FU近くにそれを制御するマイクロ命令のフィールド部分を格納し，中央制御装置(MCU)からのマイクロアドレスを受けて動作する．この方式ではFUの追加など拡張性があること，マイクロ命令アドレス・バスを利用してバス幅を短くできること，各FUのCS容量を小さくできること(FUが動作する必要がないとき(nop命令の実行)，nop命令をCS内に一つしか格納しなくてもよいようにマッピング回路を装備する)などの特徴がある．先に述べたVLIW方式などに利用できる．

分散方式-2，3は各FU近くに固有の制御装置(SCU)が置かれた構成になっている．方式2，3の違いは中央に主制御装置(MCU)があるかどうかによる．分散方式の特徴は次のようにまとめられる．

(i) 同時並列動作による高速化が可能である．

(ii) 専用高機能装置で特定機能(たとえば浮動小数点演算)に向いたマイクロ命令を採用でき，高速化を達成できる．

(iii) 浮動小数点演算機構，仮想記憶制御機構など外付けチップによる機能拡張が容易である．

(iv) 主制御装置から各機能装置に発せられるマイクロ命令の機能レベルが高くなり，各機能装置の細部を制御する必要がなくなるので，マイクロ命令の語長を短くできる．また，マイクロ命令のデコードを必要な箇所で局所的に行うことができ，配線面積を小さくできる(方式-2)．

(v) 主制御装置からの直接制御やステータス情報の転送が少なくなるので，信号遅延が小さく高速化を達成できる(方式-2)．

(vi) 機能分割して得られる比較的小規模な制御装置をPLAを用いて容易に設計できる．

(vii) 機能装置間の通信オーバヘッドが十分小さくなくてはならない．モジュールの機能レベル(粒度)を最適に設定する必要がある(方式-3)．

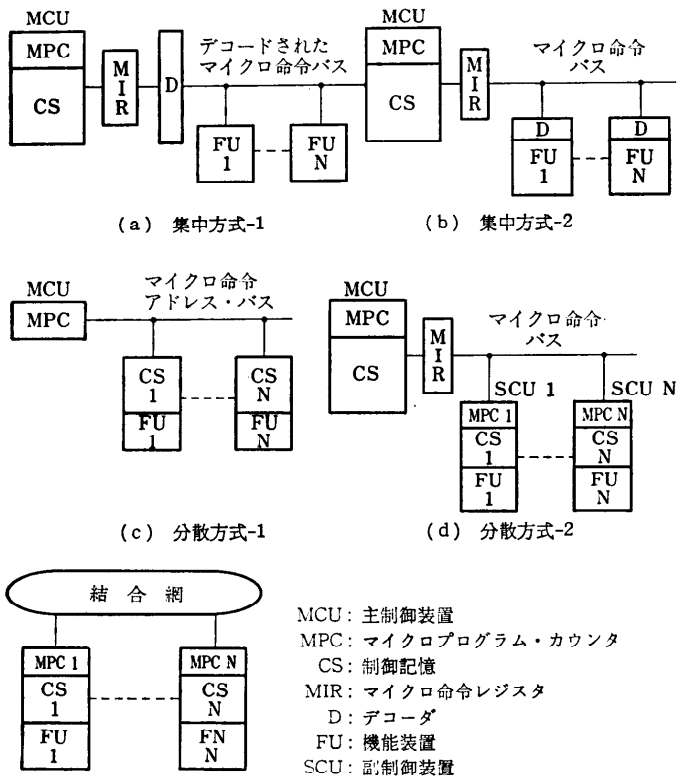
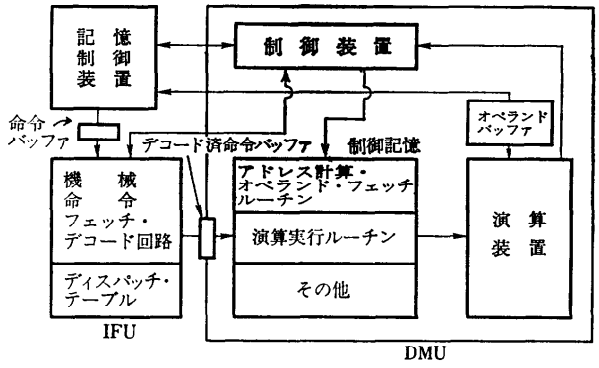


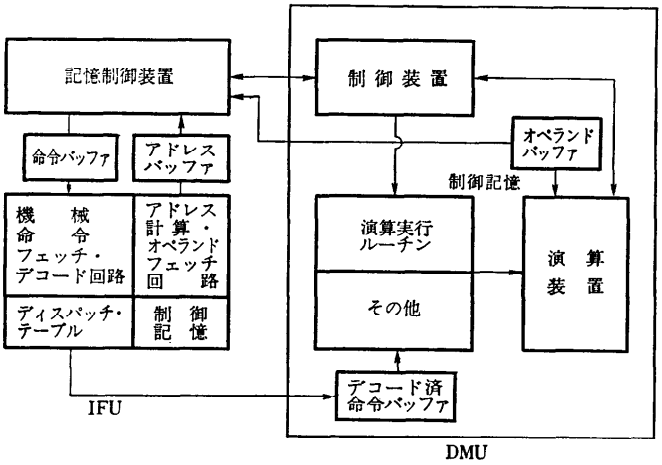
図-6 分散型マイクロプログラム制御方式

3.2 機械命令の実行モデル

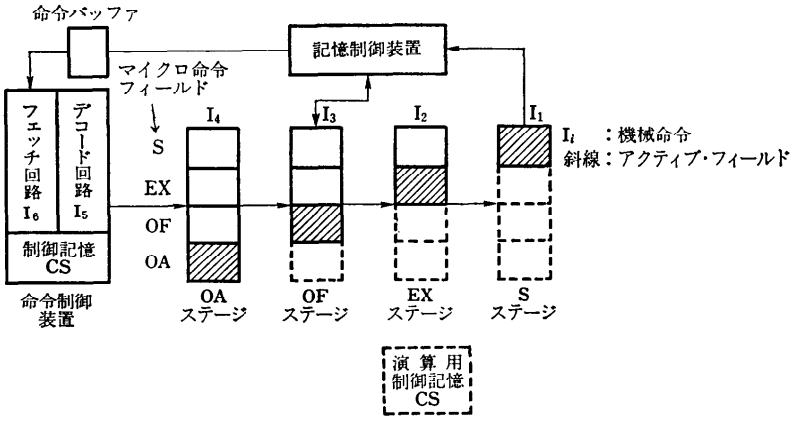
機械命令の実行過程は命令フェッチ



(a) 並列実行方式-1



(b) 並列実行方式-2



(c) パイプライン実行方式

図-7 機械命令実行モデル

(IF), デコード (D), オペランド・アドレス変換 (OA), オペランド・フェッチ (OF), 演算実行 (EX), 結果格納 (S) からなる. 機械命令の実行モデルは次のように分類できる.

(1) 逐次実行方式

IF から S に至るすべての過程が一つの制御装置で制御され, 実行される. 集中型マイクロプログラム制御方式では, これらの過程がすべて一つのマイクロプログラムで実行される. このため機械命令はオーバラップされることなく逐次実行される.

(2) 並列実行方式

図-7 (a) に示すように制御装置を機械命令のフェッチとデコードを担当する装置 (IFU) とオペランド・アクセスと演算実行を担当する装置 (DMU) に分割することができる. IFU は DMU とは独立に動作し, 機械命令をフェッチし, そのデコード結果として DMU の実行すべきマイクロ命令の先頭アドレスとパラメータを出力する. デコードは PLA や ROM など (ディスパッチ・テーブル) を用いてなされる. DMU はこれらを利用して機械命令の実行をマイクロプログラムで行う. 機械命令は機械命令のフェッチ, デコード, マイクロ命令のディスパッチ, マイクロ命令

の実行の各ステージを経てなされる．これらはオーバラップして実行でき，パイプラインを形成できる．マイクロ命令の実行は，オペランド・アドレス計算，論理—物理アドレス変換，オペランド・フェッチ，演算実行，結果の格納などからなり，これらがマイクロ命令で逐次的に実行される．主制御は DMU が握っている．中型汎用計算機や高性能マイクロプロセッサなどはこの方式を採用しているものが多い[34)-36),49),50)]．この集中型マイクロプログラム制御方式に対して，後述の VAX-8600 のようにオペランド・アドレス計算とフェッチを IFU に含め，マイクロプログラムを分散化する方式もある（図(b)）．機械命令の実行過程 IF, D, OA, OF, EX, S に対応したパイプラインを構成できる．IFU と DMU はほぼ対等な制御権をもっている．いずれの場合も IFU と DMU は非同期で動作するため，図(b),(c)に示すように各所にバッファが置かれる．また，浮動小数点演算用のマイクロプログラムを分散化する場合も多い（32 ビット・マイクロプロセッサではコプロセッサの形で一般的になっている[51)]）．

（3） パイプライン実行方式

大型汎用計算機で同期式のパイプライン方式を採用しているものもある．**図-7**(c)の集中型マイクロプログラム制御方式では，命令制御装置で機械命令はマイクロ命令に変換され，マイクロ命令は次々に命令パイプラインを流れていく．マイクロ命令には各ステージを制御するフィールドがあり，各ステージではその情報を取り込んで動作する．通常の機械命令は 1 マイクロ命令に変換されるので，図に示すように命令パイプライン中には異なる機械命令 I_1-I_6 が異なるステージでの処理を受けている．機械命令が複雑な演算命令などで複数のマイクロ命令に対応するときには，命令制御装置の CS からマイクロ命令が供給される．演算装置 (EX) でのステータスに基づいてマイクロ命令の供給（分岐マイクロ命令）がなされる場合があるので，制御が複雑になったり，マイクロ命令が連続的に供給できなくなる場合がある．

分散型マイクロプログラム制御方式では演算装置にもマイクロプログラム制御装置が存在している．整数演算，浮動小数点演算向きにさらに分散化している場合も多い．分岐マイクロ命令の処理は演算装置内で処理でき，たとえば 2 方向分岐の場合には制御記憶の 2 重化方式やディレイド・ジャンプ方式でマイクロ命令の実行遅延を回避できる[8)]．

3.3 システム例

（1） VAX-8600

VAX-8600 では，**図-8** に示すように IBOX, EBOX, FBOX, MBOX から構成され，それぞれに制御記憶が装備され，非同期で動作する[33),49)]．図-6 の分散方式-3 に近い構成となっている．基本サイクルは 80 nsec である．IBOX は機械命令やオペランドのプリフェッチ，デコード，分岐命令の実行処理などを行う．制御記憶は 50 ビット×256 語である．FBOX と EBOX はそれぞれ浮動小数点演算およびそれ以外の演算を分担し，制御記憶はおのおの 48 ビット×512 語，86 ビット×8 k 語である．MBOX は IBOX, EBOX からのメモリ要求を受理し，論理-物理アドレス変換，キャッシュ・メモリ（16 k バイト）の管理を行う．制御記憶は 80 ビット×256 語である．データがキャッシュ・メモリに存在すれば 2 サイクルで供給できる．

IBOX は命令のデコードを行って EBOX にその命令の先頭マイクロ命令アドレス（ディスパッチ先）を与える．同一命令でもアドレッシング・モードが異なればディスパッチ先は異なる．メモリ・オペランドを含む場合には，EBOX の実行途中でオペランド待ちが生ずる．IBOX にはオペランド・アドレス計算回路（マイクロプログラム制御による．汎用レジスタのコピーも存在する）があり，EBOX へディスパッチ情報を送ると同時に MBOX に対してオペランド・アクセスを要求する．MBOX から返されたデータをオペランド・バスを通して EBOX に与える．またオペランドが汎用レジスタを指している場合にはそのアドレスを IBGPR バスで EBOX に転送する．EBOX での処理が終了すると，結果はライト・バスを通して IBOX に与えられ，IBOX 内のレジスタ・コピーに値がセットされたり，メモリ・ライトの場合には MBOX に要求が出される．

8 個のチップで VAX の機械命令のフル実装を行った VAX CHIP SET，1 チップ構成で VAX の機械

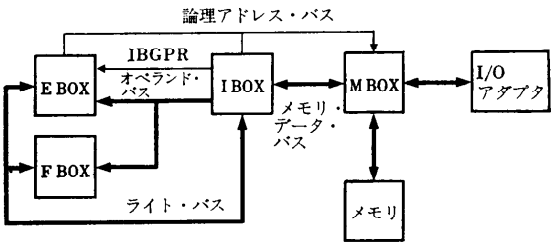


図-8 VAX-8600 のシステム構成

命令のうち175命令（このほかの32命令は浮動小数点演算チップで，59命令はソフトウェア・シミュレーションで実行される）を実行できる MICROVAX-32 でも IBOX, EBOX, MBOX が用いられている．IBOX は機械命令をデコードして，マイクロプログラムのディスパッチ・アドレスを生成する．マイクロプログラムは EBOX に集中化されており，39ビットで 1.6k 語からなる．オペランド・アドレス計算やメモリ・アクセスは EBOX のマイクロプログラムを用いてなされる[34)~36)]．

（2） iAPX 432

Intel 社の iAPX 432 は PLA による小規模順序機械を多用した制御方式となっている（図-6 の分散方式-2 に対応する）．汎用データ・プロセッサ (GDP) は，図-9 に示すように，機械命令デコード・ユニット (IDU), マイクロ命令実行ユニット (MEU) の2チップからなる[37),38)]．

（a） 機械命令デコーダ (ID)

IDU 内の ID は機械命令バッファ，機械命令フィールド抽出器，PLA を用いた順序機械から構成されている．順序機械は各フィールドのデコードにより状態遷移を行い，その過程でデータ参照アドレス生成ユニット (RGU), データ操作ユニット (DMU) の制御に必要なマイクロ命令とオペランドの論理アドレスを逐次 FIFO バッファに格納する．機械命令バッファが空に近づくと ID はマイクロシーケンサ (MIS) に機械命令のビット列のフェッチを要求する．

（b） マイクロシーケンサ (MIS)

MIS はマイクロプログラムの実行の制御を行う装置である（図-6（d）の MCU に対応）．内蔵順序機械は外部入力，内部ステータス，MEU からのステータスのテストを順に行う．外部入力でシステムの初期化が指定されている場合には，初期化マイクロプログラムを，内部ステータスとして ID から機械命令フェッチ要求があればメモリ・アクセス用のマイクロプログラムを，マイクロ命令実行ユニット (MEU) からのステータスがフォールト状態であればフォールト・ハンドリング用のマイクロプログラムを，それぞれ起動する．これらのマイクロプログラムは ROM に格納されている．これらの優先順位の高い処理が終われば，順序機械は ID より与えられたマイクロ命令（16ビット垂直型）を MEU に転送し実行させる．

（c） データ参照アドレス生成ユニット (RGU)

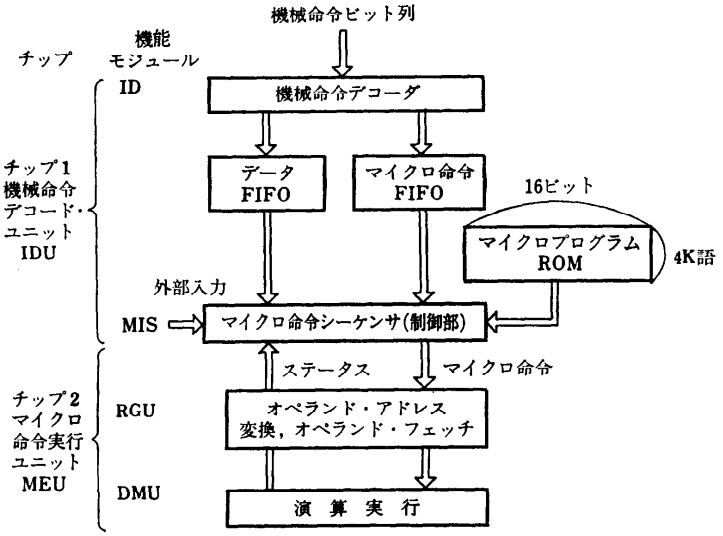
RGU は論理アドレスを物理アドレスに変換し，ケーパビリティに基づくアクセス権のチェックをし，メモリ・アクセスの起動やデータの整列化を行う．PLA による順序機械がマイクロ命令によって起動されRGU を制御する．

（d） データ操作ユニット (DMU)

DMU では，32, 64, 80 ビットの浮動小数点演算などが実行される．MIS から発せられた一つのマイクロ命令によって乗算，除算，平方根などの演算が起動され，その後の多くの処理を専用の PLA 順序機械によって実行できる．

（3） 専用システム

専用システムでは処理内容が明確に分割されることが多く，高速処理を達成するため，それらを独立なプロセッサで実行させることが多い．各プロセッサはマイクロプログラム制御され，相互結合はカスケード（パイプライン）結合などがとられる．図-6 の分散方



ID：機械命令デコーダ，MIS：マイクロ命令シーケンサ(制御部)，
RGU：データ参照アドレス生成ユニット，DMU：データ操作ユニット

図-9 iAPX 432 の構成[37)]

式-3 に対応する．たとえば，3次元図形表示用並列計算機 EXPERTS では SLP, PXP と呼ぶプロセッサがあり，1台の SLP に複数台の PXP がブロードキャスト・バスをとおして結合されている[39]．SLP はホスト計算機から与えられた多角形データ集合から走査線ごとに現れる物体のソートされたセグメント・リストを作成する．リスト処理，ソート処理，微係数計算のための高速化が図られている．走査線を負荷が均等になるように分割してその分割領域に対して各 PXP が隠れ面消去や輝度計算，スムース・シェイディング，境界線平滑化などの画素値計算を行う．このためのハードウェアが強化されている．SLP と PXP のマイクロ命令長はそれぞれ 101 ビット，88 ビットである

また，三菱のシリコン・コンパイラでは，高級言語で記述されたプログラムが図-6 の分散方式-3 に対応するシステムに変換される．各機能装置には小規模な制御装置が付随している[54]．

4. 制御記憶量の小量化方式

制御記憶量の小量化を図る方式はチップ面積やピン数の制約などを考慮にいれる必要のある VLSI 設計でとくに重要となる．本章では2，3の手法について述べる．

4.1 2レベル・マイクロプログラム制御方式

水平型マイクロ命令形式は語長が長く，並列操作の指定は可能であるが，ビット使用効率が悪い．また，水平型マイクロ命令の並列操作の組み合わせパターンには偏りがあり，順序制御指定部以外は全く同一なマイクロ命令が多数現れる．このため全体の制御記憶量が増大する．2レベル・マイクロ命令形式は，原理的には，水平型マイクロ命令の順序制御指定部を垂直型のマイクロ命令で，ALU などへの制御信号の指定部を水平型のナノ命令で指定する．マイクロ命令は制御記憶装置に格納されており，順序制御指定部とナノ命令へのポインタ部から構成される．ナノ命令は多数のマイクロ命令で共有される．

構成方式には古くは Nanodata 社の QM-1 や Burroughs 社の B-700 などの方式がある．マイクロプログラムが ROM に固定される場合には，マイクロ命令のナノ命令へのポインタ部をマイクロ命令自身のアドレスから生成させればより語長の短いマイクロ命令とすることができる．Motorola 社の MC 68000 はこの方式を採用している[8),40),41]．マイクロ命令語長は17ビットで容量は544語，ナノ命令は68ビットで容量は336語であり，平均1.6個のマイクロ命令が同一ナノ命令を共有している．制御記憶とナノ記憶の全ビット数は32096ビットとなり，2レベル方式を採用しない場合には46240ビット必要であるので，必要ビット数は約70%に減少している．また，G.R. Burke や C.A. Papachristou らは3レベル方式や PLA を用いた方式などについて論じている[47),48]．

4.2 制御記憶のオフチップ化と RISC

マイクロプロセッサのチップ内で占める制御記憶の大きさは全チップ面積の 40-50% にも及ぶことがある．これをチップ外に設定できれば内部回路に対する自由度が増し，設計も容易になる．制御記憶の内容を動的に変更して種々の応用への適応性を増すこともできる．しかし，オフチップ化は信号遅延の増大によって実行速度が低下する．また，ピン数の制約によりマイクロ命令形式として語長の短い垂直型を採用することになり，水平型に比べて実行効率が低下する．オフチップ化の例として DEC 社の LSI 11, 東芝の T-88000, 沖電気のマイクロZなどがある．前述の2レベル・マイクロプログラム方式でナノ記憶をオンチップで制御記憶をオフチップで構成することも考えられる．

RISC アーキテクチャは垂直型マイクロプログラムを機械命令プログラムと同一化し，それをオフチップ・メモリに格納する方式とみることができる[42),43]．演算系とキャッシュ・メモリを含む機械命令供給系の速度比に応じて次のような方式が考えられる．

（a） 演算系速度≫機械命令供給系速度のとき

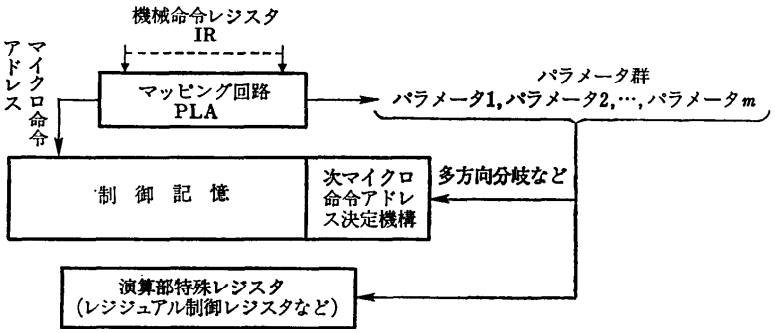
機械命令として機能的に高度なものを用意して，それに対応した複数のマイクロ命令を逐次実行させ，演算系と命令供給系のバランスをとる．演算系の速度を上げるため，水平型マイクロ命令が用いられる．これは CISC アーキテクチャに対応する．

（b） 演算系速度＝機械命令供給系速度のとき

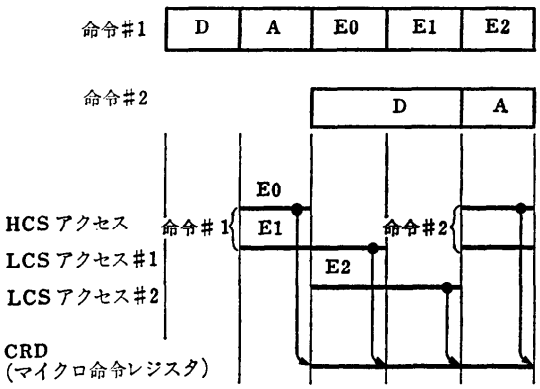
機械命令の機能レベルを上げても効果は得られない．マイクロ命令を機械命令と同格に扱い，オフチップ・メモリから供給することが考えられる．制御記憶の領域を空にできるので，この領域をレジスタに使うかオペランド・キャッシュに使うかの選択の余地が生じる．オペランド・キャッシュが有効に動作しうるほどには現状ではその容量を確保できないので，RISC-1 ではレジスタに利用している[43]．

（c） 演算系速度≪機械命令供給系速度のとき

演算系を多重化してバランスをとる．かっての



(a) ディスパッチの一般的方式



(b) M-760 の方式[45]

図-10 ディスパッチ方式

CDC 6600 や IBM 360/91 が対応する[9].

4.3 マイクロルーチンの共用と高速化

マイクロサブルーチンやレジジュアル制御方式によってマイクロルーチンの共用が可能となり，制御記憶量を減らすことができる．今日のマイクロプログラム制御計算機ではマイクロサブルーチン呼出し機構は一般的になっている．レジジュアル制御方式は，機械命令のパラメータを対応するマイクロルーチンに高速に引き渡す上で重要な役割を果たす．図-10(a)に機械命令とそれを解釈するマイクロ命令とのマッピング（ディスパッチ）機構を示す．マッピング PLA によりマイクロ命令の先頭番地（ディスパッチ・アドレス）や各種パラメータが得られる．マイクロ命令ではレジジュアル制御レジスタを使う共通なテンプレートを用意しておく．たとえば，レジスター-レジスタ演算の機械命令では，第1オペランド (R 1)，第2オペランド (R 2)，操作部 (op) がそれぞれレジジュアル・レジスタに格納される．マイクロ命令ではその操作部，オペランド指定部でこれらのレジジュアル制御レジスタを指定する．このようにして多くのレジスター-レジスタ演算命令は一つのマイクロ命令をテンプレートとして共用できる．

図-10(a) のマッピング回路では制御記憶に対するアドレスをまず生成し，次にマイクロ命令が制御記憶より読み出される構造になっている．マッピング回路自体で最初に実行するマイクロ命令とパラメータ群を直接生成し，以後のマイクロプログラムは制御記憶から読み出す方式もあり，高速化を達成できる (DEC 社の J-11 や日立製作所の前島らの方式[44])．また，図-10(b)に示すように，機械命令の最初のマイクロ命令(ディスパッチ先命令)を容量は小さいが高速な制御装置 (HCS) に格納し高速アクセスし，引き続くマイクロ命令はインタリーブ構成の大容量の低速制御記憶装置(LCS) に置く方式がある(富士通 M-760)[45]．先に述べた制御記憶の分散化の一形態とも考えられる．

4.4 マイクロプログラム・デバッグ支援

マイクロプログラミングはハードウェアの細部を知る必要があり，プログラミングと比較して多大な労力を必要とし，ミスも多い．RAM を制御記憶に使用すると修正は容易である．しかし RAM は ROM に比べて4倍もチップ領域を必要とするので，RAM を全面的に利用することができない．すでに ROM 化されているマイクロプログラムを修正する方式が VAX Chip Set で提案されている[46]．図-11 にパッチャブル制御記憶チップの構成を示す．マイクロ命令アドレスが 14 ビット長で与えられると，14 ビット・アドレスで ROM を，下位 10 ビット・アドレスで RAM が

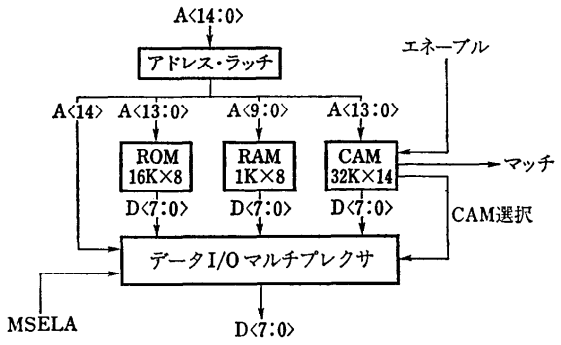


図-11 パッチャブル制御記憶[46]

アクセスされる．同時に 32 語（1語＝14 ビット）の連想記憶 (CAM) がアクセスされる．アドレス情報が連想記憶の内容に一致したとき，MSELA＝1 となり，RAM 出力がマルチプレクサで選択される．RAM 出力はアドレス・ビット 14 (A⟨14⟩) を1とし，以降 RAM 出力がマルチプレクサで選択されるようになっている．アドレスビット14を 0 に設定し直せば ROM 出力が選択される．RAM には修正マイクロプログラムが格納されている．

5. おわりに

1980 年以降に注目されてきたマイクロプログラム制御方式に関する 2，3の技術動向について紹介してきた．VLSI 時代を迎えて，マイクロプログラム制御方式はメーカの実装技術と深く関連するため一層表面に現れない技術となることが懸念される．VLSI 時代のマイクロプログラム制御方式を積極的に公開しあうことが望まれる．

謝辞 九州大学工学部情報工学科助手村上和彰氏，富士通(株)本体事業部北村俊明氏には貴重なご意見をいただいた．記して感謝の意を表します．

参考文献

1) 宇都宮公訓：マイクロプログラミング特集，情報処理，Vol. 14, No. 6 (1973).
2) 萩原 宏，マイクロプログラミング，産業図書 (1977).
3) 相磯，飯塚，坂村：ダイナミック・アーキテクチャ，bit 臨時増刊，共立出版 (1980).
4) Sigmicro Newsletter, Vol. 15, No. 3 (1984).
5) 富田眞治：VLSI とマイクロプログラミング，情報処理学会計算機アーキテクチャ研究会資料，57-2 (1985).
6) Lawson, H.W.: New Direction for Micro-and System Architectures in the 1980 s, Proc. of NCC, pp. 57-62 (1981).
7) Parker, A.C. and Wilner, W.T.: Microprogramming-The Challenge of VLSI, Proc. of NCC, pp. 63-68 (1981).
8) 富田眞治：処理装置の構成（元岡達編，VLSI コンピュータ I，岩波書店）pp. 97-157 (1984).
9) 富田眞治：並列計算機構成論，昭晃堂 (1986).
10) Hagiwara, H., Tomita, S., Oyanagi, S. and Shibayama, K.: A Dynamically Microprogrammable Computer with Low-Level Parallelism, IEEE Trans. C-29, No. 7, pp. 577-595 (1980).
11) 北村，中田，柴山，富田，萩原：ユニバーサル・ホスト計算機 QA-2 の低レベル並列処理方式，情報処理学会論文誌，27 巻，4号，pp. 445-453 (1986).
12) Tomita, S., Shibayama, K., Nakata, T., Yuasa, S. and Hagiwara, H.: A Computer with Low-Level Parallelism QA-2: Its Applications to 3-D Graphics and Prolog/Lisp Machines, Proc. of 13th Int. Symp. on Computer Architecture, pp. 280-289 (1986).
13) Barr, R.G. et al.: A Research-Oriented Dynamic Microprocessor, IEEE Trans. C-22, No. 11, pp. 976-985 (1973).
14) Charlesworth, A.E.: An Approach to Scientific Array Processing: The Architectural Design of the AP-120 B/FPS-164 Family, IEEE Comput., Vol. 14, No. 9, pp. 18-27 (1981).
15) Charlesworth, A.E.: Introducing Replicated VLSI to Super Computing in the FPS-164/MAX Scientific Computer, IEEE Comput., Vol. 19, No. 3, pp. 10-20 (1986).
16) 中西直之：最近のコンピュータ技術とバンガード・システム，インターフェース，6巻，3号，pp. 166-173 (1980).
17) 馬場，石川，奥田：2レベル・マイクロプログラム制御計算機 MUNAP のアーキテクチャ，電子通信学会論文誌，J 64-D 巻，6号，pp. 518-525 (1981).
18) Fisher, J.A.: Very Long Instruction Word Architectures and the ELI-512, Proc. of 10th Int. Symp. on Computer Architecture, pp. 140-150 (1983).
19) 阿部重夫他：スーパミニコン内蔵型ベクトルプロセッサの演算制御方式，情報処理学会論文誌，25 巻，4号，pp. 614-621 (1984).
20) Bongiorno, V.: The Cyberplus Multiparallel Processor System, Proc. of Symp. on Recent Developments in Computing, Processor and Software Research for High-Energy Physics, pp. 321-331 (1984).
21) 日高教行他：マルチコンピュータ画像生成システム MC-1，情報処理学会計算機アーキテクチャ研究会資料 CA 58-5 (1985).
22) Mankovich, T.E., Popescu V. and Sullivan, H.: CHoPP Principles of Operation, Proc. of 2nd Int. Conference on Supercomputing, pp. 2-10 (1987).
23) Leibson, S.H.: 機能や性能の向上したマイクロプログラマブル・プロセッサが相次ぎ登場，日経エレクトロニクス，No. 425, pp. 199-214 (1987).
24) Frenkel, K.A.: Evaluating Two Massively Parallel Machines, CACM, Vol. 29, No. 7, pp. 752-758 (1986).
25) Marwedel, P.: The MIMOLA Design System: Tools for the Design of Digital Processor, Proc. of 21th DA Conference, pp. 587-593 (1984).

26) Milutinovic, V., Lopez-Benitez, N. and Hwang, K.: A GaAs-Based Microprocessor Architecture for Realtime Applications, IEEE Trans. C-36, No. 6, pp. 714-727 (1987).
27) Lewis, T. G. and Shriver, B. D (ed.): Microprogramming Tools and Techniques, IEEE Trans. C-30, No. 7 (1981).
28) 馬場敬信：ファームウェア工学，情報処理，Vol. 20, No. 7, pp. 622-646 (1979).
29) Fisher, J.A.: Trace Scheduling: A Technique for Global Microcode Compaction, IEEE Trans. C-30, No. 7, pp. 478-490 (1981).
30) Nicolau, A. and Fisher, J. A.: Measuring the Parallelism Available for Very Long Instruction Word Architecture, IEEE Trans. C-33, No. 11, pp. 968-976 (1984).
31) Ellis, J.R.: Bulldog: A Compiler for VLIW Architectures, MIT Press (1986).
32) Su, B., Ding, S. and Jin, L.: An Improvement of Trace Scheduling for Global Microcode Compaction, Proc. of 17th Microprogramming Workshop, ACM, pp. 78-85 (1984).
33) DeRosa, J., Glackemeyer, R. and Knight, T.: Design and Implementation of the VAX 8600 Pipeline, IEEE Comput., Vol. 18, No. 5, pp. 38-48 (1985).
34) Johnson, W. N.: VLSI VAX Microcomputer, Proc. of the Compcon Spring, pp. 242-246 (1984).
35) Supnik, B. and Evans, I.: Microvax 32—A Vax Compatible Microprocessor, Proc. of the Compcon Spring, pp. 247-250 (1984).
36) Sherwood, W.: The VLSI VAX Chip Set Microarchitecture, in Microarchitecture of VLSI Computers (P. Antognetti, F. Anceau, J. Vuillemin (ed.), NATO ASI Series, Martinus Nijhoff Publishers (1985)).
37) Bayliss, J.A. et al.: The Instruction Decoding Unit for the VLSI 432 General Data Processor, IEEE Trans. SC-16, No. 5, pp. 531-521 (1981).
38) Budde, D. L. et al.: The Execution Unit for the VLSI 432 General Data Processor, IEEE Trans. SC-16, No. 5, pp. 514-521 (1981).
39) Niimi, H., Imai, Y., Murakami, M., Tomita, S and Hagiwara, H.: A Parallel Processor System for 3-D Color Graphics, ACM SigGraph 84, pp. 67-76 (1984).
40) Stritter, S. and Tredennick, N.: Microprogrammed Implementation of a Single Chip Microprocessor, Proc. of 11th Microprogramming Workshop, pp. 8-16 (1978).
41) Marchal, P., Nicolaidis, M. and Courtois, B.: Microarchitecture of the MC 68000 and Evaluation of a Self Checking Version, in Microarchitecture of VLSI Computers (P. Antognetti, F. Anceau, J. Vuillemin (ed.), NATO ASI Series, Martinus Nijhoff Publishers (1985)).
42) Patterson, D. A. and Sequin, C. H.: A VLSI Risc, IEEE Comput., Vol. 15, No. 9, pp. 8-20 (1982).
43) Hennessy, J. L.: VLSI Processor Architecture, IEEE Trans. C-33, No. 12, pp. 1221-1246 (1984).
44) 前島，桂，安田，木原：高集積マイクロコンピュータに適したマイクロプログラム制御方式，情報処理学会論文誌，23巻，1号，pp. 16-24 (1982).
45) 徳倉紘一他：CPU を1ボード化し，拡張性を高めた中大型コンピュータ M-730/M-760，日経エレクトロニクス，No. 426, pp. 167-191 (1987).
46) Calcagni, R.E. and Scherwood, W.: Patchable Control Store for Reduced Microcode Risk in a VLSI VAX Microcomputer, Proc. of 17th Microprogramming Workshop, pp. 70-76(1984).
47) Burk, G.R.: Control Schemes for VLSI Microprocessor, Proc. of 15th Microprogramming Workshop, pp. 91-95 (1982).
48) Papachristou, C. A.: Hardware Microcontrol Schemes Using PLAs, Proc. of 14th Microprogramming Workshop, pp. 3-16 (1981).
49) 住永洋子：処理性能が 4-6 MIPS まで上がった 32 ビット・スーパミニコン，日経エレクトロニクス，No. 372, pp. 141-160 (1985).
50) Slager, J., Louie, G. and Gindraux, L.: iAPX 286 Microarchitecture to Maximize Performance, ACM Sigmicro Newsletters, Vol. 15, No. 2, pp. 9-36 (1984).
51) 山村 徹：汎用機並の機能・性能を目指す 32 ビット・マイクロプロセッサ，日経エレクトロニクス別冊 32 ビット・マイクロプロセッサ，pp. 42-60 (1987).
52) 柴山，北村，中田，富田，萩原：ユニバーサル・ホスト計算機 QA-2 の高機能順序制御方式，情報処理学会論文誌，27 巻，10 号，pp. 960-969 (1986).
53) Jones, L.H.: Instruction Sequencing in Microprogrammed Computers, NCC, Vol. 44, pp. 91-98 (1975).
54) 平山正治：シリコン・コンパイラ向き VLSI アーキテクチャ，情報処理学会アーキテクチャ・ワークショップ，pp. 31-39 (1984).
55) 吉田，成瀬，高橋，内藤：グラフィックス計算機 SIGHT の基本構成，情報処理学会計算機アーキテクチャ研究会，60-4 (1985).
56) Beetem, J. et al.: The GF-11 Super Computer, Proc. of 12th Int. Symp. on Computer Architecture, pp. 108-115 (1985).

(昭和 62 年 8 月 24 日受付)