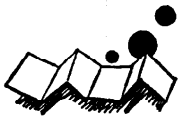
解　説

2. マイクロプログラマブル・アーキテクチャ



2.3 汎用計算機におけるマイクロプログラム技術†

金田三郎†† 竹田克己†††
森末秀雄†††† 発田　弘†††††

1. はじめに

英国ケンブリッジ大学の M. V. Wilkes によって1951年にはじめて提案されたマイクロ・プログラム方式は多くの秀れた特徴を備えている．たとえば，

（1） 制御論理の設計がプログラミングの形で実現できるため融通性が高く，開発がやりやすい．

（2） 演算回路と制御回路が明確に分離されるので設計がわかりやすくなる．

（3） 機械命令の設計に自由度が高くなる．機能の変更などもマイクロ・プログラムの変更として比較的容易に実現できる．

（4） 複雑な機能もマイクロ・プログラムのステップ数を増やすのみで安価に実現できる．

（5） ハードウェアの細部を制御しているので故障診断などにも利用できる．

などである．このために最近ではほとんどの計算機がマイクロプログラム方式で実現されているが，その背景には，

（1） LSI 技術の進歩でマイクロプログラム用の高速メモリが安価に利用できるようになった．

（2） 論理回路が LSI 化され，その変更が困難になった．

（3） ハードウェアの信頼性への要求が厳しくなり，故障診断などの機能が必須になった．

（4） ハードウェア機能への要求が高度化し，複雑な機能を実現する必要が増大した．

などの要因もあると思われる．

一方，マイクロプログラム方式は，

（1） 結線論理 (Wired Logic) に比較して処理速度が遅くなる．

（2） マイクロプログラムの作成にはハードウェアの高度の知識が必要である．

などの傾向をもっているので，計算機の設計上種々の工夫が必要になるし，マイクロプログラムの開発に関しても一般のプログラムの開発とは異なった配慮が必要である．

現在の汎用計算機におけるマイクロプログラム技術では以前から提案されていながらハードウェア技術の制約などで実現できなかったものが実用化されており，分散型マイクロプログラム方式，キャッシュ方式，多階層方式などを採用して，高速化，並列化，大容量化をはかっている．特に高性能を要求される大型／超大型計算機で多くの工夫がなされており，マイクロプログラム制御と結線論理の長所をうまく活かしている．

以下に商用の汎用計算機を中心にマイクロ・プログラム技術の現状を紹介する．

2. マイクロプログラムを用いた計算機の構成

マイクロプログラム方式を用いた汎用大型計算機として M-280 H (日立) を例に，まずマイクロ・プログラム技術全般について現状を紹介する．

2.1 M-280 H 処理装置の構成

図-1 に示すように，M-280 H 処理装置は，以下の装置より構成される．

（1） 演算処理装置 (Basic Processing Unit, BPU)

命令で指定された演算を実行する装置であり，機能的にさらに以下のユニットに分けられる．

① 命令制御ユニット (Instruction Unit, IU)

命令の読み出しと解読を行う．

† Microprogramming in general purpose computers by Saburo KANEDA (COMPUTER & SYSTEM ARCHITECTURE DEPT.), Katsumi TAKADA (Kanagawa Works, Hitachi Ltd.), Hideo MORISUE (Computer Engineering Division, NEC Corporation) and Hiroshi HATTA (NEC Corporation).

†† 富士通(株)電算機開発部
††† (株)日立製作所神奈川工場
†††† 日本電気(株)コンピュータ技術本部
††††† 日本電気(株)情報処理製品計画本部

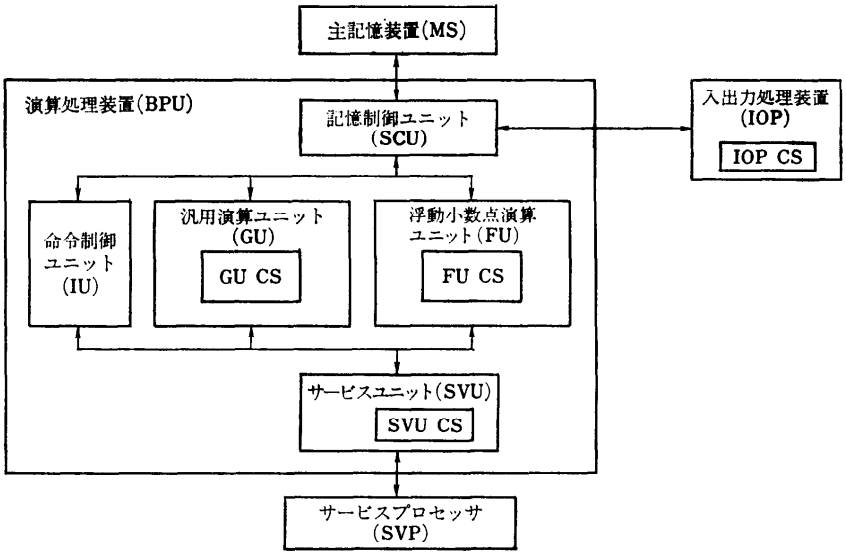


図-1 M-280 H 処理装置の構成

② 汎用演算ユニット (General Arithmetic Unit, GU)

下記③の FU 以外の演算を実行する.

③ 浮動小数点演算ユニット (Floating Point Arithmetic Unit, FU)

浮動小数点演算と固定小数点乗除算を実行する.

④ 記憶制御ユニット (Storage Control Unit, SCU)

主記憶装置に対するデータの読み出しと書き込みの制御を行う.

⑤ サービスユニット (Service Unit, SVU)

演算処理装置と後述の SVP との間の, 保守制御関係のインタフェースを制御する.

(2) 入出力処理装置 (Input Output Processor, IOP)

入出力装置と主記憶装置との間のデータ転送を制御する.

(3) 主記憶装置 (Main Storage, MS)

プログラムとデータを格納する.

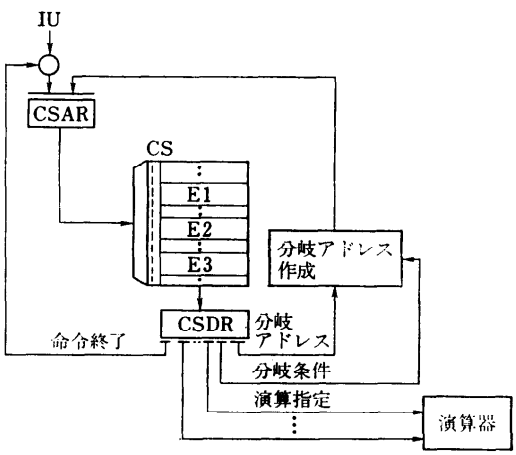
(4) サービスプロセッサ (Service Processor, SVP)

計算機の操作や保守に必要な制御を行う.

以上の装置の内, マイクロプログラムが制御するのは GU, FU, SVU, IOP である. 特に SVU をマイクロプログラム制御にしたのは, 多様化するマン-マシンインタフェースの制御に柔軟性をもたせるためである. その他の IU, SCU, MS の制御はハードウェアの結線論理で行い, SVP の制御はマイクロプロセッサで行っている. マイクロプログラムは, 制御記憶と呼ぶ高速で書き替え可能な記憶装置内に格納されている. M-280 H では, マイクロプログラム制御ユニットの内部に専用の制御記憶を置き, 制御記憶とそれが制御するユニットの間の信号伝播時間を短縮して, 特に GU と FU の高速処理を可能にしている.

2.2 マイクロプログラム制御方式

マイクロプログラム制御部は, 図-2 に示すような装置からなる. マイクロプログラムは, 一連のマイク



CS: Control Storage
CSAR: Control Storage Address Register
CSDR: Control Storage Data Register

図-2 マイクロプログラム制御部

ロ命令語の流れとして実行されるが，そのマイクロ命令語は制御記憶 (CS) 内に格納される．マイクロ命令語を読み出すためのアドレスは CSAR に設定され，読み出されたマイクロ命令語は CSDR に格納される．CSDR はいくつかのフィールドに分割されていて，各フィールドで指定された動作は，1基本マシンサイクル内で並列に実行される．次に GU を例として，マイクロプログラムの動作を説明する．GU のマイクロプログラムは，IU が起動する．すなわち，IU は主記憶から命令語を読み出し，その命令コードに対応する制御記憶のアドレスを CSAR にセットする．この CSAR の値で，命令の第1実行サイクルのマイクロ命令語 E1 が CSDR に読み出される．この CSDR の各フィールドで演算器などを制御するとともに，次のマイクロ命令語 E2 のアドレスを CSAR にセットする．この E2 のアドレスは，E1 の分岐アドレスフィールドの値そのままのこともあるが，分岐条件のテスト結果を反映して一部修整されることもある．以上のような手順で，E2，E3 のマイクロ命令語を実行していくが，E3 の1フィールドで命令の実行終了が指定されると，IU は次の命令の先頭マイクロ命令語のアドレスを CSAR にセットし，次命令の実行を開始する．

マイクロプログラムは，フロッピディスク上に格納されていて，処理装置の電源投入時に，フロッピディスクから制御記憶内に転送される．

2.3 マイクロ命令語の構成と機能

(1) GU，FU のマイクロ命令語

GU，FU のマイクロプログラムは，前述のように IU によって起動され，機械語命令やオペレーティングシステム高速化支援のためのファームウェアなどを実行する．また，GU のマイクロプログラムは，処理装置の障害箇所の指摘（マイクロ診断）にも用いられる．機械語命令の内，浮動少数点演算命令と固定少数点乗除算命令は FU が実行し，その他の命令は GU が実行するが，IU は命令解読後に GU と FU を両方同時に起動する．したがって，GU または FU の一方は演算を実行するが，他の一方は次に起動されるまでアイドルループすることになる．従来技術では，IU は一旦 GU を起動し，GU が浮動少数点演算または固定少数点乗除算であることを判定すると，GU のマイクロプログラムが FU のマイクロプログラムを起動していた．GU と FU の同時起動は，この従来技術の起動時の処理オーバヘッドをなくし，処理速度の向上を可能にしている．

表-1 GU マイクロ命令語の構成

フィールドの種類	機　　能
演算器制御	並列加算器，直列加算器，シフターなどの演算器を制御する．
ワークレジスタ制御	マイクロプログラムがワークとして使用するレジスタの入出力を制御する．
レジスタ間データ転送制御	各種のレジスタ間のデータ転送を制御する．
レジスタメモリ制御	汎用レジスタ，浮動少数点レジスタの読み出し，書き込みを制御する．
メモリリクエスト制御	主記憶への読み出し，書き込みを制御する．
分岐条件	分岐するために必要な条件を示す．
分岐アドレス	分岐先のマイクロ命令語の番地を示す．
リトライ制御	命令語を再試行するために必要なデータの退避などを指示する．
リテラル	演算器に入力するデータ値を指定する．
命令の終了	1命令語の終了を指示する．

GU，FU のマイクロ命令語は，機械語命令などの高速処理を要するものを実行するので，マイクロ命令語内に多数のフィールドを用意し，1基本マシンサイクル内での並列動作の多重度を高めて，処理の高速化を実現している．表-1 に GU のマイクロ命令語の構成の概略を示すが，FU のマイクロ命令語もほぼ同様の構成である．

(2) SVU のマイクロ命令語

SVU は，サービスプロセッサ (SVP) と処理装置本体間のインタフェース動作を制御する．SVP は，オペレータや保守員のコンソール操作などの比較的高速性を要求されない処理を行うので，SVU のマイクロ命令語は，並列実行可能な1語中のフィールド数を減らし，マイクロ命令語長を小さくして，制御記憶の容量節減を図っている．

(3) 入出力処理装置 (IOP) のマイクロ命令語

IOP は，チャネルプロセッサ，チャネル制御部，チャネル部の3部分からなるが，これらの3部分が階層的に制御を渡しながら一連の入出力動作を制御する．3部分はおのおの独立のマイクロプログラムで制御される．

チャネルプロセッサは，GU によって起動され，入出力命令と入出力割り込みの処理を行う．チャネル制御部はチャネルプロセッサによって起動され，入出力装置の起動と終結の処理を行う．チャネルプロセッサとチャネル制御部のマイクロ命令語は，表-1 の GU のマイクロ命令語と同様の構成をもつが，GU ほどの高速処理は要求されないため，並列実行可能なフィールド数を減らして制御記憶の小容量化を図っている．

チャネル部は，標準的な入出力インタフェースで入出力装置と接続され，データ転送の制御を行う．チャネル部のマイクロ命令語は，データ転送が主な機能であり，並列動作も少ない．したがって1語中のフィールド数を減らして制御記憶の小容量化を図っている．

3. 分散マイクロプログラム方式

マイクロプログラムを複数の制御対象の近くにそれぞれ独立に設けた制御記憶に分散配置するマイクロプログラム制御方式である．

被制御ユニットの近傍に制御記憶を配置することによりマイクロプログラム・アクセスを高速化し，さらに各制御対象ごとのマイクロ命令を専用化，特殊化することにより処理の高速化，および並列動作制御によるパイプラインの使用効率向上を図るのが目的であるが，論理ゲートとともに RAM を内蔵した LSI 化装置で本方式を採用するとチップクロス（2チップにまたがるパス）を削減することも可能であり，VLSI 化向きの技術といえる．

各制御記憶のマイクロプログラムの起動は機械命令のデコードの結果によってハードウェアで行われるのが普通であるが，アドレス変換制御のマイクロプログラムのように TLB（アドレス変換バッファ）にヒットしなかった場合に起動されるといったように，特定の事象の発生によって起動されるものもある．

起動後の各制御記憶のマイクロプログラム間の同期制御は，大別すると，データ同期と事象同期の二つの方法がある．データ同期は，データの確定待ちにより同期をとる方法であり，特定レジスタに対応してデータ確定情報を設けて制御する方法がよく採用される．主記憶読み出しデータと演算との同期や，演算と演算結果の主記憶書き込みの同期などに利用される．一方，事象同期は，特定事象または状況の発生待ちにより同期をとる方法であり，命令終了の同期などに利用される．いずれの同期方法にしても，処理の性質上必要最小限の同期制御が行われる以外は各制御記憶のマイクロプログラムは独立して処理を遂行し，処理の並列度が高められる．

各制御記憶のマイクロプログラム制御系は同一周期のクロックで動くものが多いが，データパスの論理段数を減らして整数分周期のクロックで動作させて高速化を図るものもある．

前述した M-280 H においても GU，FU，SVU，IOP は独立したマイクロプログラムで制御されているが，以下に他の事例を紹介しよう．

3.1 M-780（富士通）

FACOM M-780 CPU（M-780）は命令制御部・演算部・記憶制御部からなり立っており，命令制御部と演算部で，独立したマイクロプログラム制御を採用している．さらに，命令制御部は，制御記憶を分割しハードウェア設計上の柔軟性とハードウェア資源を効率的に動作させる高速性を同時に実現している．

命令制御部の制御記憶は，

① ファーストフロー CS (FCS)
② ネクストDサイクル CS (NDCS)
③ ネクストアドレス CS (NACS)
④ マルチフロー CS (MCS)
⑤ サブオペ CS (SCS)

の5種類に分割されている．このような分割を採ったのは，それぞれの制御記憶を機能により特殊化し，語長・語数・実装位置を最適化することによって，高速化を図るためである．これは，高速 RAM & 論理 LSI をマスタ・スライス方式にしたことで，実現できた．

各制御記憶について説明を加える前に，前提となるパイプライン処理について述べる[1]．命令は，命令実行パイプラインを1回または複数回通って処理される．これは，マイクロプログラムの各マイクロステップに対応しており，それぞれを『フロー』と呼んでいる．パイプラインは，D/A/T/B/E/W の6サイクルからなっており，それぞれ，命令デコード，オペランド・アドレス計算，TLB とバッファ・タグ読み出し，バッファ読み出し，演算，書き込みを行う．

① ファーストフロー CS (FCS)

命令の最初のフローで読み出され，このフローのAサイクル以降の制御に使用される．

② ネクストDサイクル CS (NDCS)

複数フローで処理される命令の第2フロー以降のDサイクルを制御する．

③ ネクストアドレス CS (NACS)

マイクロプログラムのシーケンス制御を行う．

④ マルチフロー CS (MCS)

FCS で制御された第1フローに続く第2フロー以降のAサイクル以降を制御するのに使用される．語長・語数とも最も多い．

⑤ サブオペ CS (SCS)

命令コードの次に，1バイトのサブ命令コードをもつ2バイト・コードの命令を処理する場合に使う．サ

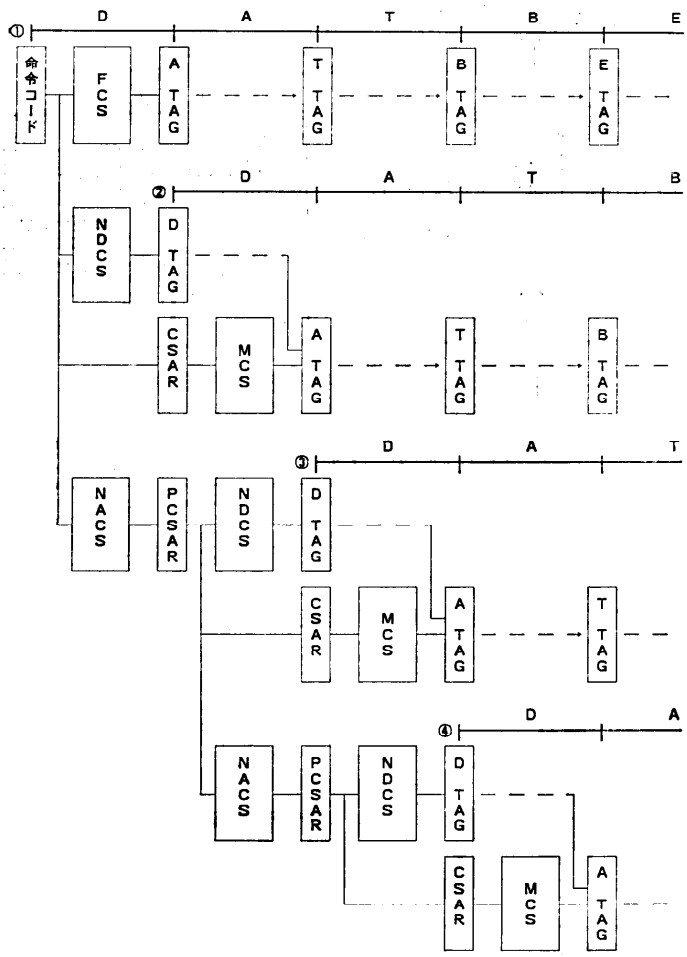


図-3 制御記憶のタイムチャート（2フロー以上の命令の場合）

ブ命令コードにより NACS の出力アドレスを修飾することができる．

図-3 に複数フローで実行される命令での制御記憶の動きを示す．また，**表-2** に各制御記憶の語長・語数を示す．

3.2 ACOS 1500（日本電気）

表-2 各制御記憶の語長・語数

	語長（ビット）	語　数
FCS	128	256
NDCS	48	2048
NACS	16	2048
MCS	144	2048
SCS	64	256

大型コンピュータ ACOS 1500 では，**図-4** のように制御記憶として次の四つが存在する．第一は，演算処理装置全体にわたる制御と，基本演算器，浮動小数点加減算器，十進／可変長データ演算器を制御する主制御記憶（MCS）である．第二は，命令制御部の命令解読，アドレス計算，オペランド先取りを制御する命令制御記憶（PCS），第三は，バッファ制御部のアドレス変換を制御するアドレス変換制御記憶（ACS），第四は，乗除算器，二進／十進変換器を制御する演算制御記憶（ECS）である．[3]

なお中大型コンピュータ ACOS 430，610，630 でも中央処理装置内部のすべての論理構造を制御する主制御記憶（CS）と，浮動小数点演算を高速に実行する高速科学演算機構（オプション）を制御する副制御記憶（HCS）の二つの制御記憶を用いている．このシステムでは，高速科学演算機構は中央処理装置の基本クロックの半分の周期で動作する．

4. 制御記憶構成方式

市場ニーズの多様化に呼応した追加機能の増加にともなってマイクロプログラム容量も増加の傾向にあり，また性能向上のためにも制御記憶の構成方法に工夫が必要となっているので，これに関した技術を紹介する．

4.1 アクセスタイムの異なる記憶素子による構成

高速 SRAM と低速 SRAM を混在させて制御記憶を構成し，制御記憶アドレスに応じて読み出しのためのダミーサイクルを発生させる制御を行う方式である．性能とコストとのトレードオフにより機能別マイクロプログラムの対性能影響度に応じて使い分けるのが目的である．たとえば ACOS 3300 では，25 ナノ秒，45 ナノ秒の 2 種類の RAM で制御記憶を構成している．

4.2 キャッシュ記憶を用いた構成

主記憶とキャッシュ記憶の関係と同様に，マイクロプログラムを実行するときに対象マイクロプログラ

ム・ルーチンを制御記憶キャッシュへ動的に格納する方式で制御記憶キャッシュ内に格納したマイクロプログラムを実行する．マシン命令の処理など高速性を要求されるマイクロプログラム・ルーチンは制御記憶キャッシュ上に常駐させ，さほど高速性を要求されない処理，たとえば，リセット処理，デバッグ関連処理などのマイクロプログラム・ルーチンは，通常，制御記憶キャッシュ上には存在していない．実際に実行されるときだけ制御記憶キャッシュ上に転送されて，各処理が実行される．対象となるデータがマイクロプログラムであるため，既存のキャッシュ記憶とは以下の点で異なる．

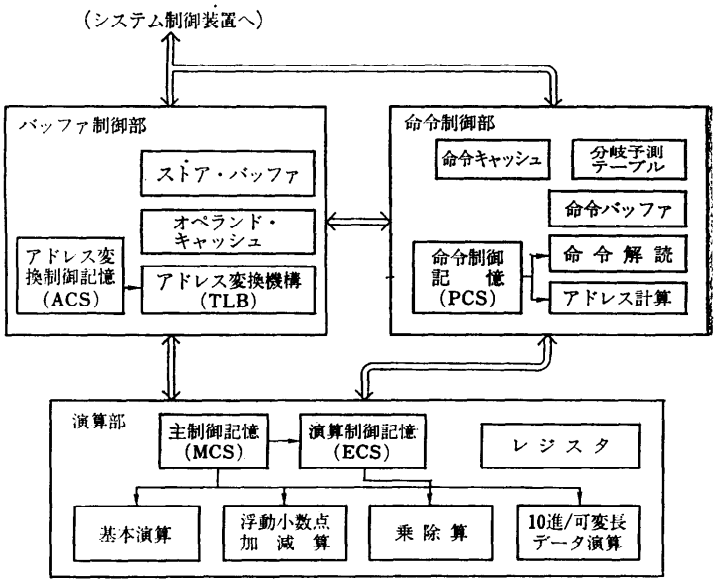


図-4　ACOS 1500 演算処理装置の構成

① 転送ブロック容量は一般に既存のキャッシュ記憶に比べてかなり大きい．また，高速記憶の有効活用を促進するために，マイクロプログラムの機能属性に応じて数種のブロック単位のうちから最適な一つを選択する可変ブロック構造をとるものもある．たとえば ACOS 1500 では，32，64，128，256 ワードの 4 種類の可変ブロック構造を採用しており，ブロック長の選択は同じく主記憶上に存在する管理テーブルに表現されている．

② キャッシュ更新アルゴリズムは利用頻度によらずマイクロプログラムの機能属性に応じた高速化の必要度に依存した優先順位制御による．たとえば ACOS 1500 ではマイクロプログラム・アドレスの若番地優先で更新管理を行っているが，若番地に高速化が必要なマイクロプログラム・モジュールを割り当てておき，装置の初期設定フェーズで初期化マイクロプログラムが優先順位の高いブロックを次々に実行することによりこれらのブロック内のマイクロプログラムを制御記憶キャッシュ上で常駐化することによって，ブロックロードによる性能低下を最小限に抑えつつキャッシュの有効活用を図っている．

次に，FACOM M-730 の事例を紹介する．[2]

M-730 の制御記憶のアドレス空間は，64 K ワード（1 ワードは 32 ビット）から構成される．64 K ワードの制御記憶のアドレス空間全体は主記憶上の一部に割り当てられている．このエリアはソフトウェアからはアクセスできない．

制御記憶キャッシュの大きさは 16 K ワードであり，二つのエリアに分割されて管理される．前半の 8 K ワードのエリアはレジデント・エリア，後半の 8 K ワードのエリアはトランジェント・エリアと呼ぶ．制御記憶のアドレス空間と制御記憶キャッシュの関係を図-5 に示す．

レジデント・エリアには，制御記憶のアドレス空間の最下位 8 K ワードのマイクロプログラムを常時格納しておく．ここに格納される主なマイクロプログラム・ルーチンには，マシン命令処理，動的アドレス変換処理などがある．

トランジェント・エリアには，制御記憶のアドレス空間の残りの 56 K ワードのマイクロプログラムを格納する．ここに格納される主なマイクロプログラム・ルーチンには，リセット処理，デバッグ関連処理などがある．このトランジェント・エリアへのマイクロプログラムの格納は，32 ワードを 1 ブロックとして，ブロックごとに管理されている．つまり，制御記憶キャッシュの 1 ブロックごとに，その内容の有効／無効性，対応する制御記憶アドレスを管理情報として，CPU 内に保持している．CPU ハードウェアはこの管理情報をもとに，主記憶上に格納されているマイクロプログラムの必要なブロックを必要なときにトランジェント・エリアに転送する．

このキャッシュ方式の制御記憶を利用して，M-730

では，IMPL (Initial Micro Program Loading) 時にマシン命令処理，割り込み処理などMシリーズ・アーキテクチャを実現するマイクロプログラムをローディングする前に，マイクロ診断用のマイクロプログラムを制御記憶キャッシュに転送して，CPU ハードウェアの初期診断を行っている．

4.3 複数バンクによる構成

制御記憶を複数バンクに分けて構成し，マイクロプログラムの条件分岐で同時に複数の分岐先マイクロプログラムステップを読み出すことにより条件分岐での回路遅延を改善することを目的とする構成方式である．分岐方向は2または4方向が多いので，4バンク構成にしている．たとえば ACOS 2000, 750 などがこの構成である．

5. 二階層制御記憶方式

従来のマイクロプログラム方式には，1語のビット長を多くし，多数の回路を同時に制御することによって高性能を実現させた方式(水平型マイクロ命令)と，加減算・シフト・分岐などの基本的な制御手順を一単位として一つのマイクロ命令とした方式（垂直型マイクロ命令）がある．これらから，システムの性能・コスト・テクノロジにより，最適な方式が選択されてきた．

水平型マイクロ命令は高速だが，高コストであるという短所，垂直型マイクロ命令は低コストであるが，非高速性という短所がそれぞれあった．両者の短所を補い最適な性能・コストを得るために考えられたのが二階層制御記憶方式である．

事例として FACOM M-330 E を紹介する．

M-330 E の二階層制御記憶の上位層は垂直型マイクロプログラムで，基本的な動作を制御し，下位層は水平型マイクロプログラムで，より高速なきめ細かい制御を行う．垂直型マイクロプログラムはコントロールストレージと呼ばれる制御記憶に格納され，単にマイクロプログラムと呼ばれる．また，水平型マイクロプログラムは，ピコストレージと呼ばれる制御記憶に格納され，ピコプログラムと呼ばれる（図-6）．

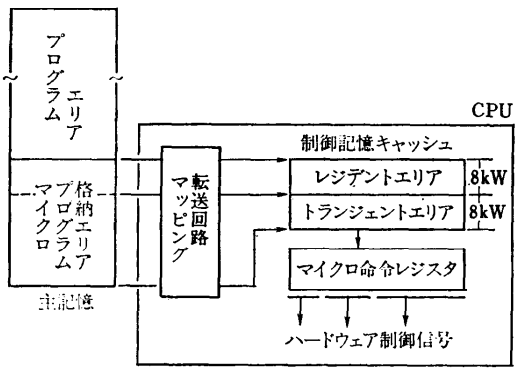


図-5 M-730 のキャッシュ方式の制御記憶

主記憶から読み出されたマシン命令は，マイクロプログラム，およびピコプログラムによって実行される．図-6 をもとにマシン命令実行の方式を概説する．マイクロプログラムは，マシン命令読み出しのマイクロ命令を指示する．これにより，ピコプログラムが走行して，命令バッファ上に読み出されているマシン命令（通常，マシン命令は先取りされて命令バッファ上に格納されている）を，命令レジスタにセットし，そのマシン命令の命令コードの値により決められたマイクロプログラム，ピコプログラムを起動する．起動されたマイクロプログラムとピコプログラムは，命令コードに応じた処理を行う．

また，マシン命令の実行中にプログラム割り込み条件や，入出力割り込み条件が発生した場合には，ハードウェアにより割り込み処理ルーチンのマイクロプログラムが起動され，そのルーチンの中で割り込み原因が調べられ，必要な処理がなされる．

そのほかに，M-330 E の CPU マイクロプログラムはシステムに接続されているチャネル共通の処理，たとえば CCW (Channel Command Word) の読み出しなども行う．これにより，各チャネルの負担の軽減，小型化を実現している．

6. ハイブリッド・マイクロプログラム

マイクロプログラムの処理手順の一部分を機械命令プログラムで記述して実行するマイクロプログラム実行制御方式である．

マイクロプログラマブル・プロセッサでは，使用頻度の高い基本命令について水平型マイクロプログラムの特徴が最大限生かせるようにハードウェア構造を最適化するので，マイクロプログラムのダイナミック・ステップ数はきわめて少ない．これに対して初期設定，構成制御命令，データハンドリングを反復するリスト処理命令，プロセス制御機能などのシーケンシャルな要素を含む処理では，このように高度に水平化されたマイクロプログラムを使用しても並列性を高めたプログラミングはできずにステップ数が増加するので制御記憶の利用効率が著しく低下することになる．

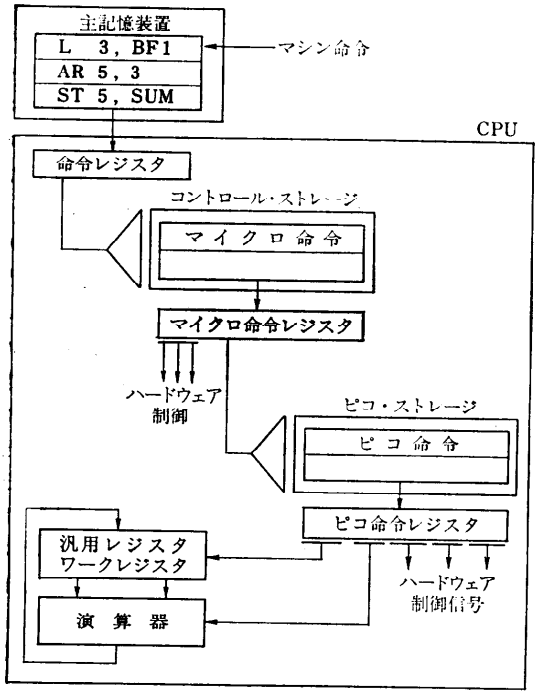


図-6　M-330 E の二階層制御記憶

本実行制御方式は，基本命令からなるソフトウェア・プログラムで該当処理手順を構成し，マイクロプログラムがこれを高速にコール／リターンできる制御方式であり，このソフトウェア・プログラムは主記憶上のハードウェア領域に格納するので制御記憶の利用効率の改善が図れるが，同時に高度な並列処理ハードウェアを利用できるので性能改善を図ることもできる．

7. 内蔵型専用機のマイクロプログラム技術

汎用計算機だけでなく専用機においてもマイクロプログラム方式が採用されることが多いので一例としてM-280 H（日立）の内蔵アレイプロセッサを紹介する．

M-280 H は汎用の計算機であるが，科学技術計算に高い比重を占めるベクトル演算を高速に処理する内蔵アレイプロセッサ (Integrated Array Processor, IAP) を備えている．M-280 H の IAP は，内積演算などのベクトル演算用のベクトル命令を設定し，これをマイクロプログラムと専用演算器（FU 内の演算器）を使用して高速化するものである．

IAP は，次のように動作する．すなわち，IU はベクトル命令を解読すると GU のマイクロプログラムを起動する．GU のマイクロプログラムは，マイクロ命令語で FU を起動し，FU でベクトル演算，GU でスカラー演算を同期的に実行し，一つの演算結果を GU でまとめる方式で演算を実行する．

従来機種の IAP は，GU のマイクロ命令語に追加した IAP フィールドで制御するものであった．この場合は，ベクトル命令以外では IAP フィールドは意味がなくノーオペレーション指定されるのみなので，制御記憶が無駄に使われるという欠点があった．M-280 H は，GU と FU のマイクロプログラムでベクトル命令を実行する方式でこの点を改善している．

8. マイクロプログラムの開発環境，支援システム

開発期間の短縮化と開発効率の改善を目標に，マイクロプログラムの開発環境は年ごとに強化されてきている．各社の開発支援システムのポイントを以下に紹介する．

8.1 日本電気

① マイクロプログラムをフローチャート形式で入力・編集しソース・イメージを作成できる．本ツールはパーソナル・コンピュータ (PC) で運用でき，論理設計のペーパレス化，修正／保守作業の容易化が推進できる．

② マイクロプログラミング上の制約事項を完全にチェックできる．1ステップの並列動作間のみならず前後ステップ間の制約事項についてもチェック可能である．本ツールも PC で運用でき，設計品質の向上が推進できる．

③ マイクロプログラムの各ステップの番地割り付けを機械化した．

④ マイクロプログラム・シミュレータにより論理検証が可能である．本ツールは PC で運用でき，マイクロプログラムのソース・イメージまたはアセンブル後のパターン・イメージのどちらを入力とすることも可能である．ハードウェア・モデルは論理記述で別に入力する必要がある．

⑤ 超高速ハードウェア・シミュレータによりハードウェアのプロダクト・イメージ（製造用論理記述ファイル）上でマイクロプログラムの論理検証が可能である．

図-7 にマイクロプログラム開発全体の流れを示す．

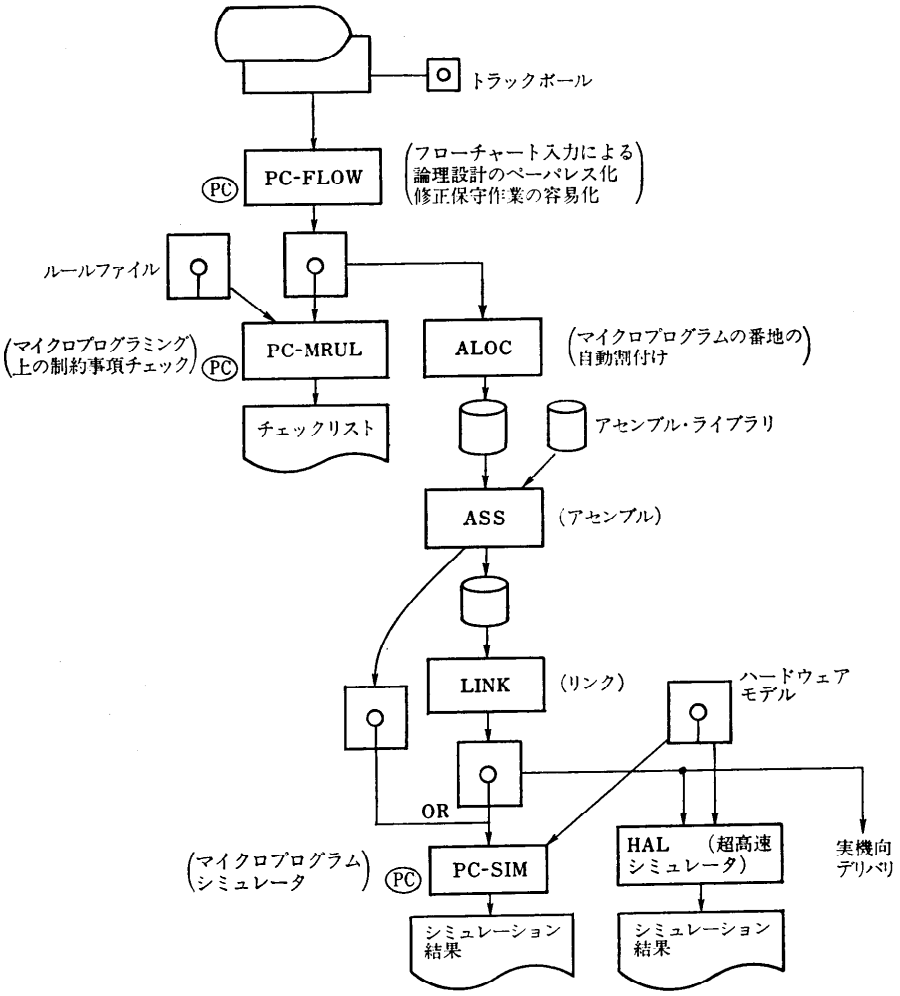


図-7 マイクロプログラム開発の流れ

8.2 日　　立

① Mシリーズの計算機上で動作するマイクロプログラムアセンブラを用意している．これは，図-8 に示すような出力を作成して，マイクロプログラムの作成を支援するものである．

② マイクロプログラムは，シミュレーションによってもデバッグできる．そのほかに，実計算機上でマイクロプログラムを走らせてその正当性をテストするために，マイクロプログラムの走行履歴を記録する機構や，指定番地のマイクロ命令語実行時に，計算機を停止させる機構などを用意している．

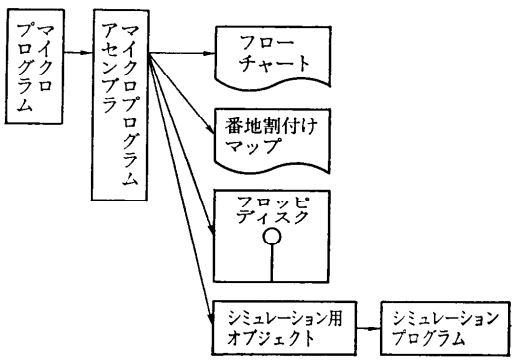


図-8 マイクロプログラム作成支援システム

8.3 富士通 (M-730 の場合)

① M-730 のマイクロプログラムの開発には，M

シリーズ上で動作する汎用のマイクロアセンブラとシミュレータが使用された.

② M-730 にはマイクロプログラム・デバッグ機能として，以下のように機能が設けられている.

●指定した任意のマイクロプログラムのアドレスでCPU ハードウェアのクロックを停止する機能.

●クロックの停止直前までに実行されたマイクロ命令のアドレスの履歴を保持する機能.

●マイクロプログラムが使用するレジスタなど各種ハードウェア状態を読み出したり，変更する機能.

●マイクロプログラムを1ステップずつ実行する機能.

9. おわりに

マイクロ・プログラム方式は実用化当初は中小型計算機での利用が中心であったが，LSI 技術の進歩で大型計算機でも利用されるようになり，最近では小型から超大型まですべての計算機に採用されている．この傾向は今後とも続くであろう．それは特に，VLSI 時代をむかえてマイクロプログラム方式がハードウェア開発効率の改善や設計ミスの救済等の面からも必須になっている点からも予想できることである．マイクロプログラム用の記憶素子に RAM を用いると電源投入時にフロッピディスクから毎回ロードしなければならないなどの不便もあるが電気的に書き換え可能な ROM (EPROM) の高速化やバッテリィ・バックアップ付の RAM などの技術進歩でもっと使いやすくかつコスト低減できる可能性もある．また，マイクロ・プログラム用メモリの大容量化で現在のソフトウェア (たとえば OS) の一部または全部をマイクロプログラム化して処理の高速化を狙うことは以前から多くの人が期待しているところである.

参 考 文 献

1) 槌本，他：CPU を1ボードに実装した大型コンピュータ M-780，日経エレクトロニクス，No. 396, pp. 179-209 (1986. 6. 2).

2) 徳倉，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. 167-192 (1987. 7. 27).

3) 馬場，他：2レベルのキャッシュやパイプライン処理の工夫で速度を上げた大型コンピュータ ACOS 1500，日経エレクトロニクス，No. 373, pp. 233-279 (1985. 7. 15).

(昭和62年8月26日受付)