並列処理プロセッサMELPIP-1の構成と制御方式について

A Parallel Processor for Concurrent Micro-Process

平山正治[+]　房岡璋[+]　金森唯[+]　朱雀二朗[++]
Masaharu HIRAYAMA　Akira FUSAOKA　Tadashi KANAMORI　Jiro Sujaku
[+]三菱電機株式会社中央研究所　[++]同コンピュータシステム工場
[+] Central Research Laboratory, Mitsubishi Electric Co.Ltd.　[++] Computer System Works

1. はじめに

LSI技術の進展にともない，小規模なプロセッサを複数個結合したマルチ・プロセッサ・システムが，汎用大型計算機では十分なコスト・パーフォーマンスが得られていない分野に有効となる可能性があり，多くの研究が行なわれている.[1)~3)] これらのマルチ・プロセッサ・システムを用いて並列処理を行なう場合，対象とする問題のデータ構造やアルゴリズムの性質に依存してプログラムの構成が決まってくるため，容易に問題に適合しうるようなハードウェア構成と制御方式が必要となる．　とりわけ，1つのアルゴリズムを並列のプロセスに分解して実行するような場合では，できるだけきめの細かいプロセスに分解して，そのアルゴリズムのもつ並列性をよりよく反映するとともに，同期や通信といった相互作用を効率よく実施してオーバヘッドを減少する事が望ましい.

このような観点から，マイクロ・プログラム・レベルでのプロセスの相互作用により，問題に適合した並列処理を実施する事を目的として，並列処理プロセッサ　MELPIP-1　の開発を行なっている．　マイクロ・プログラム・レベルで並列プロセスを実施している V.R.Lesser のマシン[4)] では，均質なプロセッサを複合化した構成になっており，プロセス全体の管理やプロセス間の通信が複雑であるように思われる．MELPIP-1ではプロセス処理を1つのプロセッサで集中して，しかもハードウェアによって高速に処理している事を特徴としている.

本稿ではMELPIP-1の基本構成を主として報告する.

2. MELPIP-1の概要

2.1 システム構成

MELPIP-1システムは，図1に示すように，基本的には4台の演算プロセッサ（AP），各1台の制御プロセッサ（CP），メモリ・プロセッサ（MP），およびホスト・インターフェイス（HI）で構成されている．　これらのプロセッサはコントロール・バスとデータ・バスによって接続され相互に情報（データ）の送受を行ないながら，各々独立した動作をしている．このように機能的に異なったプロセッサの複合化システムにしたのは，汎用プロセッサにくらべてハードウェア量が少なくてすむ事と，コントロールを一箇所に集中する事により，並行プロセスの同期が効率的に行なえると考えたからである.

演算プロセッサ（AP）はCPから1個のマイクロ命令を受けとり，この命令を実行し，結果のステータス情報をCPに返すというサイクルをくりかえす．　APはマイクロ・プログラムのシーケンス制御機能をもたず，命令の実行終了毎にCPに次のマイクロ命令の要求を出す．APは4台で構成されているが，1台で独立した2本のマイクロ・プログラムを時分割で実行するようにしているので，システム全体で8本

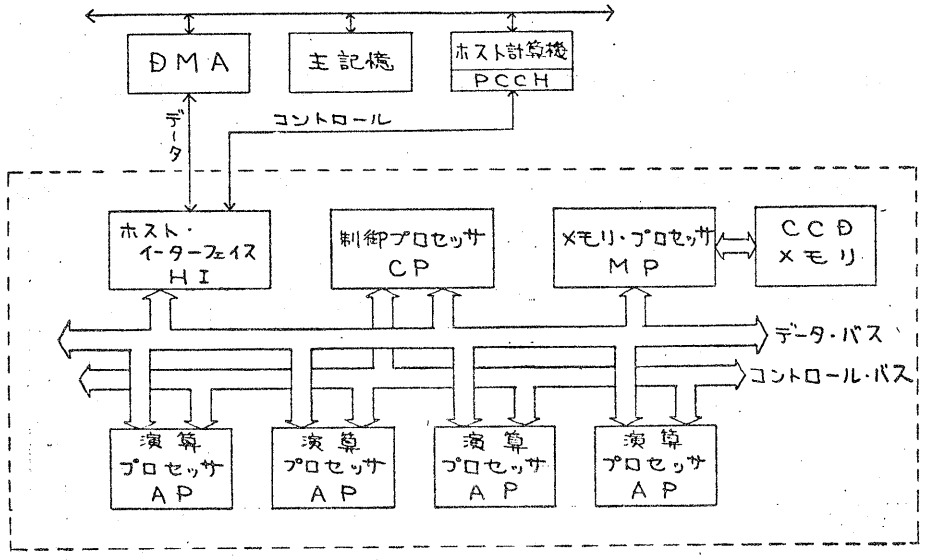


図1　システム構成

のマイクロ・プログラムの同時実行が可能となる．

制御プロセッサ（CP）はAPの要求に応じて，CP内にある制御記憶より次に実行するマイクロ命令を読み出してAPに送るというシーケンス制御を行なう．　また後述するように，マイクロ・プロセス間の同期などプロセスの制御を行なっている．

メモリ・プロセッサ（MP）はAPからの主記憶リクエストに応じて主記憶データのR／Wをコントロールする．MELPIP-Iの主記憶はCCDメモリと高速RAMで階層的に構成する事により，大きな容量の主記憶を容易に実現できるように試みた．

コントロール・バスはCPとAPを結合するバスであり，マイクロ命令と命令実行結果のステータス情報の転送に使われる．

データ・バスはMELPIP-Iのすべてのプロセッサを結合するバスであり，主記憶データをはじめ，各プロセッサのメモリ，レジスタの転送に使われる．

MELPIP-Iはホスト・インターフェイス（HI）を経由してホスト計算機と（他の入出力装置と同様の方法で）接続している．　ホスト計算機は，入出力機能をもたないMELPIP-Iの代わりに入出力動作を行なって，ユーザとのインターフェイスをとるソフトウェアと，MELPIP-Iで処理するプログラムを管理するソフトウェアを実行する．

2.2　アーキテクチャの特徴

MELPIP-Iは，種々の問題領域における並列処理を効率よく実行する目的のために，融通性やマイクロ・プログラム・レベルでのきめの細かい並列処理機能などを配慮したアーキテクチャになっている．

(1) 可変語長処理

ユーザの種々のアプリケーションにおいて，自由なデータ構造を利用できるために，マイクロ・プログラム・レベルで対象とするデータの長さを変えうる可変語長処理を行なっている．加えて，APはパイプラインによって可変長データを処理しているので，高速の多語長演算が可能になっている．

(2) 制御ストリームの可変性

語長を可変としたのと同様に，容易に最も適した並列処理アルゴリズムを実現するために，マイクロ・タスクとマイクロ・プロセスという考え方を導入し，これらを制御するための機能をハードウェアで用意している．

マイクロ・タスクとはMELPIP-Iにホスト計算機から依頼される仕事の単位である。 MELPIP-Iでは同時に複数のマイクロ・タスクの存在を許すが，これらの間でインタアクションをもつ事はできない．

マイクロ・プロセスとは1つのAPで連続実行される一連のマイクロ命令の系列である．同一のマイクロ・タスクに属するマイクロ・プロセスは主記憶等のデータの共有が可能であり，互いに同期をとりながら並列処理を行なう事ができる．

このように，ユーザがマイクロ・プログラムの実行をプロセスという形でとらえ，MELPIP-Iで用意されているプロセス制御命令を使って，問題に最も適した制御機造を実現できるという可変制御ストリームの機能をもっている．

(3) プロセス間通信機能

同一のマイクロ・タスクに属する複数のマイクロ・プロセスは，プロセスの動作と共通データ送受の両方において通信する機能が必要である。 MELPIP-Iでは主記憶をマイクロ・タスクで管理しており，各マイクロ・プロセスは同じ論理空間をもつため主記憶データはすべて共有できる。 プロセス間の動作の通信（同期）はプロセス制御命令の中のSYNC（Syncronize）命令を使って，2つのプロセスが互いに同期をとりながら処理を進める事ができる．

(4) 階層構造の主記憶

MELPIP-Iでは複数のマイクロ・タスクが同時に存在するため大容量主記憶を必要とするが，これを安価に実現するためにCCD素子を利用している。 しかしCCD素子はアクセス時間が長く，ランダム・アクセス・メモリとしては使用できないため，高速のMOSメモリによる主記憶バッファを設け，セット・アソシアティブ方式のキャシュ・メモリと同様の方式で高速大容量主記憶を構成している．

2.3 システムの動作

MELPIP-Iは演算専用プロセッサであり入出力の機能をもっていないため，各アプリケーションにおける入出力動作，およびユーザとのインターフェイスはすべてホスト計算機が行なう方式を採っている。 ホスト計算機にはMELPIP-Iでのマイクロ・タスクを管理するOSがあり，MELPIP-Iを使用するユーザはこのOSにジョブの起動と実行を依頼する。 この時，ユーザがOSに渡す情報には以下のものがある．

① マイクロ・プログラム
② 処理の対象となるデータ
③ ホスト計算機で実行するソフトウェア

ホスト計算機のOSはこれらの情報を受取ると，マイクロ・プログラムをCPの制御記憶に，処理されるデータをMPを経由してMELPIP-Iの主記憶に，ホスト計算機で実行するユーザのソフトウェアをホストの主記憶に，それぞれ格納する。 以上の準備した後，OSはHIを経由してCPにマイクロ・タスクを起動する．

マイクロ・タスクが起動された時，CPは自動的に1個のマイクロ・プロセス（これをルート・プロセスと呼ぶ）を生成し，APを割りあて，マイクロ・プログラムの実行を開始する．これ以降はユーザが作ったマイクロ・プログラムの中で新しいプロセスを生成して並列処理を進めていく。 マイクロ・プロセスはプロセスを終了するマイクロ命令を実行すると消滅し，ルート・プロセスが終了した時はマイクロ・タスクの終了とみなし，CPはホスト計算機に報告する．

MELPIP-Iにおいて実行するマイクロ・プロセスは，入出力動作等を行なう時にホスト計算機にあるソフトウェアを起動して処理を行なわせる事ができる。 また，このソフトウェア動作の終了は，ホスト計算機からの割込みという形で，MELPIP-Iのマイクロ・プロセスに伝えられる．

MELPIP-Iシステムを利用する時のシステム動作の概要を図2に示す．

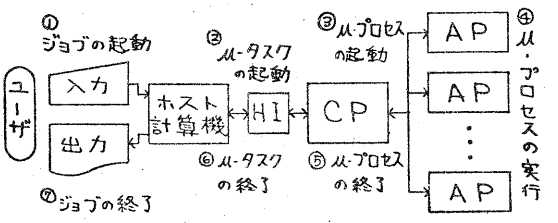


図2 システム動作の概要

3. ハードウェア構成

MELPIP-1は7台のプロセッサと2本の共通バスで構成されたシステムであり，以下ではこれらの基本的な構成要素について述べる．

3.1 演算機能

MELPIP-1において，データを処理するマイクロ命令はすべて演算プロセッサ（AP）で実行される． APの内部構成を図3に示す．

APの内部には1ブロック 128語（8bit/語）のレジスタ・ファイルが8ブロックあり，このうちの半分（4ブロック）が1つのマイクロ・プロセスにおいて使用できる． これらのレジスタ・ファイルはマイクロ命令の指定によって，演算装置の入力，出力となるほかに，主記憶リクエスト等を行なう場合のアドレス・レジスタ，データ・レジスタとして使用される．

APでの演算は，シフタ，SPL，ALUと呼ぶ，2段構成，各8ビット巾の演算装置で行なわれ，これらは以下に示す機能をもっている．

シフタ…… 0～4ビット左右ロジカルシフト
　　0～4ビット左右ローテートシフト
SPL …… 3ビット・デコード
　　上位の連続する0，または1のビットの個数カウント
　　上位ビットのマスク
ALU ……加減算，キャリ付加減算（＋，－）
　　論理演算（∨，∧，∀）
　　1ビット乗除算（＊，／）

APと他のプロセッサとの間で，以下に示すデータ転送機能をもっている．

- 主記憶リクエスト（MP）
- 制御記憶リクエスト（CP）
- A－A間通信（他のAP）

以上の他に，APは次のようなデータ処理能力をもっている．

- レジスタのベース・レジスタ修飾
- レジスタの間接指定
- 8ビット・カウンタ（2個）
- 8，4ビット定数との演算
- サブルーチンのリンク用スタック
- ベース・レジスタのインクリメント
- 命令リピート機能

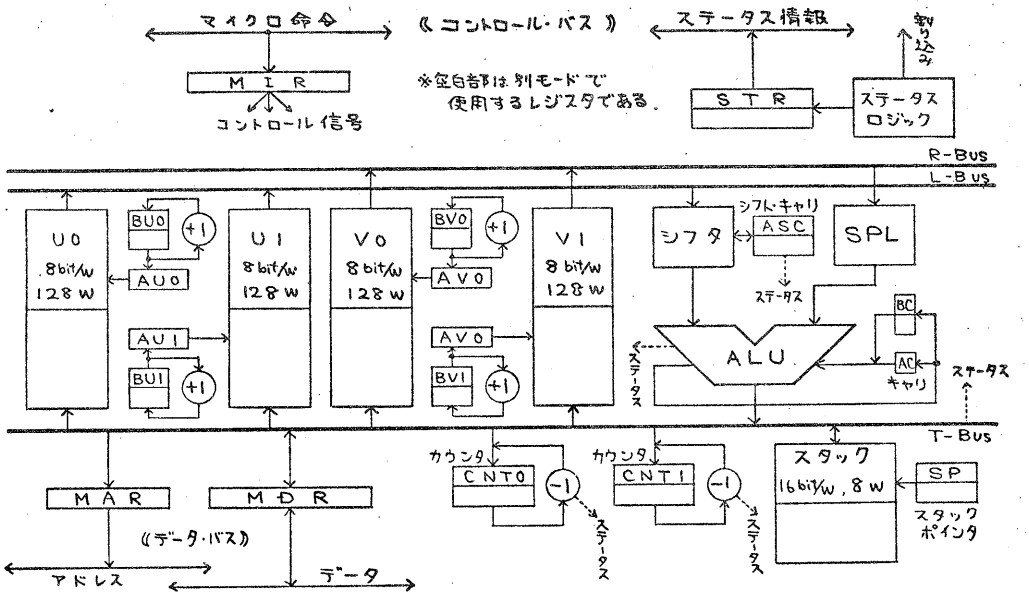


図3 演算プロセッサの構成

《可変長データ処理》

MELPIP-1では可変長データの取扱いを容易にするために，個々のマイクロ命令のフィールドでデータ長を指定する形式を採っている．すなわち，各アプリケーションに最も適したデータ長の演算を自由に組みあわせる事ができるように，各命令毎に対象とするデータ長を1～16バイトの範囲で指定する．

この可変語長データはAPにおいてパイプラインで高速に処理される．パイプラインにおける各ステージの動作は以下のようになる．

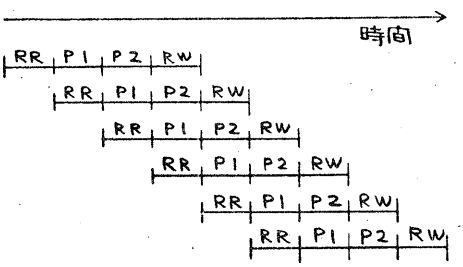
Stage 1 --- レジスタ・ファイルからの読出し（2つのレジスタ・ブロックより各1バイトのデータ）
Stage 2 --- 演算処理1（シフタ，SPLの演算）
Stage 3 --- 演算処理2（ALUの演算）
Stage 4 --- レジスタ・ファイルへの書込み（1バイトのデータ）

以上の動作が各マイクロ命令のバイト・カウント・フィールドで指定されるバイト数だけ，次々に処理される．（このようすを図4に示す）従って，各ステージの実行時間をΔTとすればBバイトの処理には（B＋3）ΔT 時間を要する．MELPIP-1ではΔTを70 nsecとしているので語長と処理時間の関係は次のようになる．

1バイト	280	nsec
2 〃	350	〃
4 〃	490	〃
8 〃	770	〃
16 〃	1330	〃

《APの時分割使用》

MELPIP-1では物理的に4台のAPで構成されているが，同時に8本のマイクロ・プログラムが動作できる機能をもっている．これは1台のAPが2セットのリソース（レジスタ・ファイル等，図では空白部で示す）をもち，1個のマイクロ命令の実行が終了してから次のマイクロ命令がCPから送られてくるまでの時間を使って，もう一本のマイクロ・プログラムの実行を行なっている．すなわち2本のマイクロ・プログラムが1命令ずつ交互に1台のAPを使って動作している．



※バイト数が6の時の例

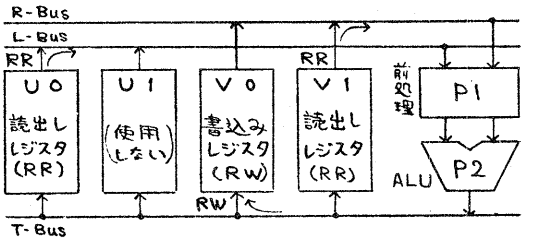


図4 パイプライン動作

3.2 制御機能

MELPIP-1の制御機能は制御プロセッサ（CP）によって行なわれる．

CPの機能は2つあり，1つはシーケンス・コントローラの機能であり，他の1つはプロセス制御を行なう機能である．

前者は各APからの命令終了割込みを受けつけ，その時のステータス情報によって次に実行すべきマイクロ・プログラムのアドレスを決定し，CPの内部にある制御記憶からマイクロ命令を読出してAPに送出する．この動作もパイプラインによって高速に処理され，4台のAPからのリクエストに対して，できるだけ割込みペンディング状態をなくし，APが最大のスピードで実行できるように考慮されている．

後者は，制御記憶から読出されたマイクロ命令であった時，またはHIよりマイクロ・タスクの起動・終了，マイクロ・プロセスへの割込みが発生した時，およびハードウェアで検知するエラーが発生した時にプロセスのふるまいに

に関する制御を行なう．

CP　　は，図5に示すように，各処理部と制御記憶，PAR（Process Activation Record の略号），その他制御レジスタで構成される．

制御記憶はマイクロ・プログラムを格納するためのメモリで，32 bit/語，最大 64 K語（現在の実装は 16 K語である）の全体がRAMで構成される書込み可能制御記憶（WCS）である．

PARはマイクロ・プロセスに関するすべての制御情報を格納したレジスタであり，8台のAPに対応して 104 bit/語のPARが8語ある．

MELPIP-1ではマイクロ・プロセスを生成する時，適当なAPを割当て，そのAPに対応するPARにそのマイクロ・プロセスの情報が格納される．　以後，マイクロ・プロセスが消滅するまで，プロセス制御動作はPARの内容の参照，変更によって行なわれる．

表1にPARの内容を示す．

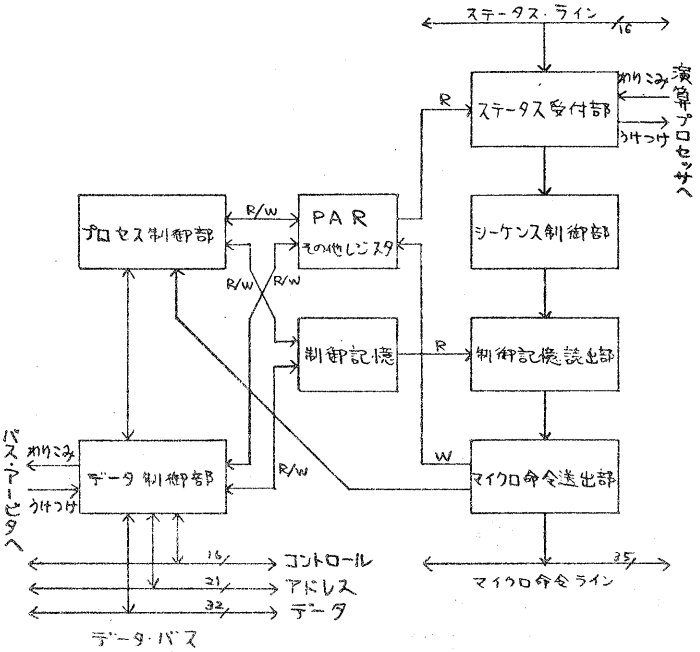


図5　制御プロセッサの構成

3.3　メモリ・コントロール

MELPIP-1の主記憶は各プロセッサの共有メモリであり，各タスクの主記憶データを保護するために，タスク毎に独立した主記憶論理空間を割当てている．　メモリ・プロセッサ（MP）は，タスク毎のアドレス空間の管理と，前述したCCDとMOSによる階層構造の主記憶の制御を行なうプロセッサである．

MPは，図6に示すように，本来の主記憶であるCCDメモリのほかに，主記憶バッファ・メモリ，タグ・メモリ，マップ・メモリ，CCDバッファ・メモリの各リソースをもち，各プロセッサ（AP，HI）からの主記憶リク

表1　PARの内容

略号	名称	ビット長	内容
TN	タスク番号	3	マイクロ・タスクの番号を示す．（ホスト計算機で指定する）
PN	プロセス番号	3	マイクロ・プロセスの番号を示す．（ユーザがプログラムで指定する）
SN	先頭セグメント番号	8	制御記憶に格納されるマイクロ・プログラムの先頭セグメント番号を示す．
SS	セグメント・サイズ	8	〃　　大きさを示す．
IE	割込みルーチン・アドレス	16	ホスト計算機からの割込みに対する処理ルーチンの先頭アドレスを示す．
PS	プロセス・ステータス	13	プロセスを制御するための各種制御情報を示す．
SC	シーケンス制御	5	シーケンス・コントロールを行なうためにマイクロ命令を一部を格納する．
CC	コンディション・コード	8	マイクロ命令の実行結果を示すコンディション・コードを格納する．
PC	プログラム・カウンタ	16	マイクロ・プログラム・カウンタで，各タスクの論理アドレスで示す．
CCS	CC退避領域	8	ホスト計算機からの割込みがあった時にCCを格納する
PCS	PC　〃	16	〃　　PC　〃

エストを受付けて，主記憶データのR／Wを制御している．

CCDメモリは最大容量1Mバイト（現在の実装は512Kバイトである）で，1Kバイトを単位としてブロック化されている．主記憶バッファ・メモリはCCDメモリの1部を格納しておく高速メモリで，容量64Kバイト（64ブロック）である．タグ・メモリは主記憶バッファ・メモリにCCDメモリのどのブロックが格納されているかを示すメモリで，20bit/語，32語で構成されている．マップ・メモリはタスク毎の論理アドレスを物理アドレスに変換するためのメモリで，8bit/語，256語で構成されている．

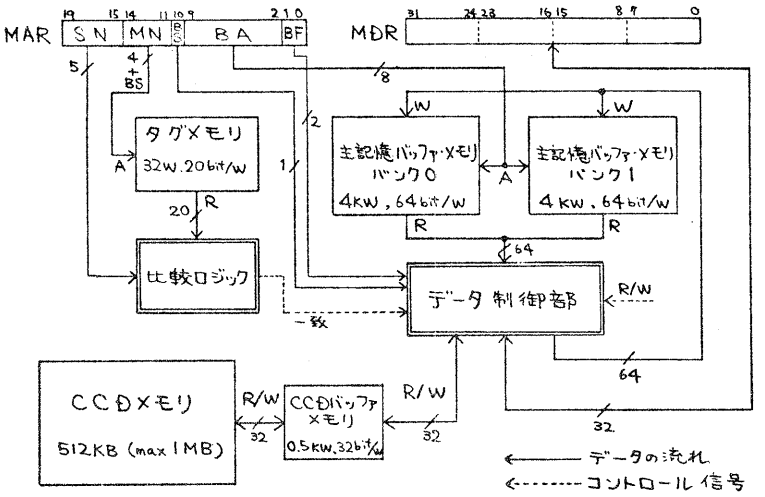


図6　メモリ・プロセッサの構成

《主記憶階層化の方式》

MELPIP-1で実施している主記憶の階層化はセット・アソシアティブ方式のキャシュ・メモリと同様の方式を採用している．すなわち，図7に示すように，CCDメモリ，主記憶バッファ・メモリとも1Kバイト単位にブロック化し，セット数を32，コンポーネント数を2として構成している．従って，同一セットに属する32個のCCDメモリのブロックのうち，2個だけが主記憶バッファ・メモリにコピーされ，この2つのブロック番号がタグ・メモリに保持される．

主記憶リクエストの時，論理アドレスのMNとBSで選択されたセットのタグ・メモリの内容が参照され，この値によって対応する主記憶ブロックが索引される．（図6，図7参照）

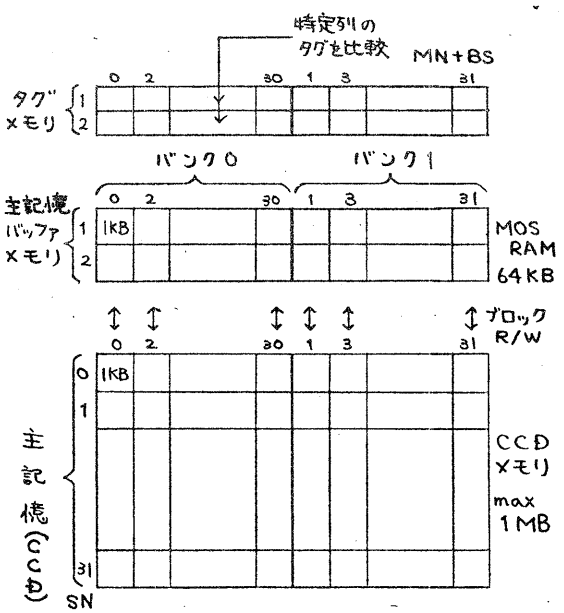


図7　主記憶階層化の方式

また，CCDメモリを起動し，主記憶バッファ・メモリとの転送を行なっている間でも，他のプロセッサからの主記憶リクエストを円滑に行うため，主記憶バッファ・メモリを2バンク構成にするとともに，2ブロック分のCCDバッファ・メモリを置き，CCDとのデータ転送による主記憶占有時間を極力減らすように考慮されている．

3.4 バス構造

MELPIP-1では2本の共通バス（データ・バス，コントロール・バス）によって各プロセッサ間が結合され，これを介するデータ転送に従って各プロセッサが非同期に動作している．

コントロール・バスはCPとAPを結合し，CPからAPへマイクロ命令を伝えるライン（35本），APからCPへ命令実行後のステータス情報を伝えるライン（16本），および，タイミングの制御ライン（2本）の計53本のラインで構成されている．（図8参照）

以上の共通バスの他に，各APからCPへマイクロ命令の実行終了割込みをおこす割込みライン（INT）と，CPから各APへ割込みが受付けられた事を伝える受付けライン（ACK）がある．マイクロ命令の実行を終了したAPは割込みラインによってCPに知らせ，CPがこの割込みを受付けた時にAPはCPにステータス情報を送る事が許される．

CPからAPへのマイクロ命令の送出はCPがマイクロ命令と共にAPを識別するためのヘッダ（3ビット）を付けて送出する．

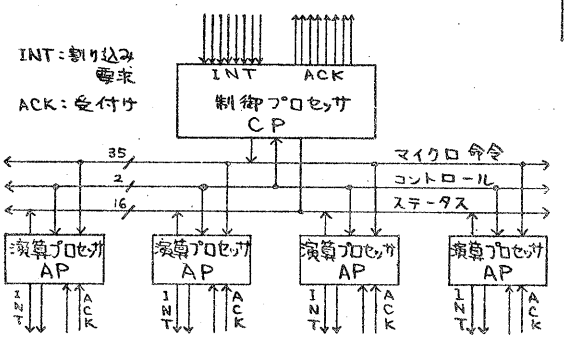


図8　コントロール・バスの構成

データ・バスは各プロセッサ間の種々のデータ転送を行なうためのバスであり，すべてのプロセッサと結合している．データ・バスは，相手プロセッサとそのプロセッサ内のレジスタ，およびレジスタ・アドレスを指定するアドレス・ライン（21本），実際のデータが送受されるデータ・ライン（32本），およびデータ・バスの転送を制御するためのコントロール・ライン（16本）の計56本で構成されている．（図9参照）

以上の共通バスの他に各プロセッサからデータ・バスの使用要求を出すライン（REQ）と，その要求が受けつけられた事を伝える受付けライン（ACK）がデータ・バス・アービタに接続されている．データ・バス・アービタはデータ・バスの使用権を制御する独立した装置であるが，ハードウェア的には小規模なもので，MPの中に実装している．

データ・バスの使用の要求をもったプロセッサは，データ・バス・アービタに対して使用要求の割込みをおこし，これが受付けられるとこのプロセッサがバス・マスタとなり，任意のプロセッサを指定してデータ転送を行なう．

データ・バスを使った転送は14種あり，この概要を表2に示す．

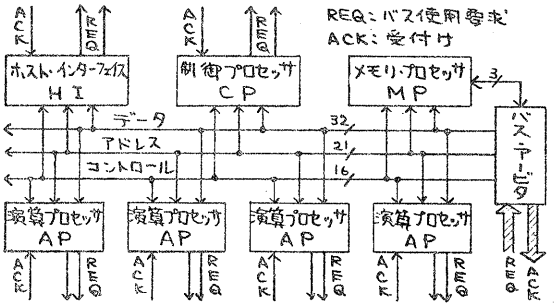


図9　データ・バスの構成

表2　データ・バス転送の概要

転送の種類	対象	データバス	アドレス・バス
システム・リセット	HI → all	—	—
CCD転送終了	バスアービタ → all	—	—
主記憶リクエスト	AP HI → MP	32,16,8	主記憶アドレス
A-A 間通信	AP HI → AP	16，8	レジスタ・アドレス
制御記憶リクエスト	AP HI → CP	32	制御記憶アドレス
CPレジスタ	HI → CP	32,16,8	レジスタ・アドレス
タスク起動依頼	HI → CP	32	—
ホスト計算機からの割込み	HI → CP	—	プロセス番号
タスク強制終了	HI → CP	—	—
マップ・メモリ	HI → MP	8	マップ・アドレス
タグ・メモリ	HI → MP	20	タグ・アドレス
ホスト計算機への割込み	CP → HI	—	割込情報
タスク異常終了報告	CP → HI	—	エラー表示レジスタ
タスク終了報告	CP → HI	—	終了情報

4. マイクロ・プログラム

4.1 マイクロ命令の概要

MELPIP-1のマイクロ命令は1語32ビットで構成され，基本的に4ビットずつ8個のフィールドに分かれている．各フィールドの意味は，表3に示す11種の命令のタイプによって意味や組合わせが異なるが，基本的には以下のようになる．

- 演算コード・フィールド（OP1～OP5）
  3，2，1オペランド命令において演算の種類を指定する．プロセス制御命令，ブランチ命令において命令の機能を指定する．
- レジスタ・ブロック・フィールド
  （RGS1～RGS7）
  演算の対象となるレジスタ・ブロックを指定する．また，動作の詳細を指定する場合もある．
- バイト数フィールド（BCNT）
  演算の対象となるデータの長さをバイト数で指定する．
- レジスタ・フィールド
  （LRG，RRG，TRG）
  演算の対象となるレジスタのアドレスを指定する．（ブロック内アドレス）
- サブ・ファンクション・フィールド（SF）
  ALUを使う演算と並列して動作できる各種サブ・ファンクションを指定する．
- ブランチ・フィールド（BR）
  マイクロ命令のシーケンス・コントロール・フィールドであり，各種条件つきスキップ機能を指定する．
- ブロック数フィールド（BRCT）
  主記憶リクエスト，A－A間通信の時の転送ブロック数を指定する．
- アドレス，データ・レジスタ・フィールド
  （ARG，DRG）
  主記憶，制御記憶リクエスト，A－A間通信の時のアドレス・レジスタ，データ・レジスタを指定する．

以上の他に，8 bit Emit，4 bit Emitや各命令毎に特殊な指定を必要とするものがある．

表3 マイクロ命令の概要

マイクロ命令	機能	フィールド分割 OP (31-28)	RGS (27-24)	BCNT (23-20)	LRG (19-16)	RRG (15-12)	TRG (11-8)	SF (7-4)	BR (3-0)
プロセス制御命令	各種プロセス制御を行なう	0000	0 OP5	（各命令によって異なる）					
ブランチ命令	各種ブランチを行なう	0000	1 OP4	〃					
1オペランド命令	LRG ㊥ Emit → LRG Emit ㊥ RRG → RRG	0001	00 0/1 RGS7	BCNT	LRG / 4 bit Emit	4 bit Emit / RRG	OP3	SF	BR
制御記憶Test & Set命令	制御記憶のデータに対してTest & Setを行なう	0001	01 RGS6 0	1000	ARG	0000	0000	SF	BR
制御記憶リクエスト命令	制御記憶リクエストを行なう	0001	01 RGS5	0000	ARG	DRG	000 MOD2	SF	BR
主記憶Test & Set命令	主記憶のデータに対してTest & Setを行なう	0001	10 RGS6 0	1000	ARG	0000	0000	SF	BR
主記憶リクエスト命令	主記憶リクエストを行なう	0001	10 RGS5	0 BRCT	ARG	DRG	00 MOD1	SF	BR
A-A間通信命令	A-A間通信を行なう	0001	11 RGS4	0 BRCT	LRG	RGAD		SF	0 PN
2オペランド命令	LRG ㊥ RRG → LRG or RRG	0010	0 RGS3	BCNT	LRG	RRG	OP2	SF	BR
定数発生命令	8 bit Emit → TRG	0010	1 RGS2	BCNT	8 bit Emit		TRG	SF	BR
3オペランド命令	LRG ㊥ RRG → TRG	OP1	RGS1	BCNT	LRG	RRG	TRG	SF	BR

《特徴的なマイクロ命令》

◎プロセス制御命令

MELPIP-1のプロセス制御命令は，表4に示すように14種あり，これらはすべてCPのハードウェアによって実行される．MELPIP-1のユーザはこれらのマイクロ命令を自由に使って，問題に適応した並列処理アルゴリズムを記述する事ができる．

表4　プロセス制御命令

名称	ニーモニック	機能
Fork & Dispatch	FAD	プロセス番号PNのプロセスを生成しPCで示される番地から実行を開始する(PN,PCは命令で指定)
Vector Fork & Dispatch	FADV	プロセス番号がPNからPK個のプロセスを生成し，それぞれPCで示される番地から実行を開始する(PN,PK,PCは命令で指定)
Synchronize	SYNC	PNで示されるプロセスと同期をとる．(PNは命令で指定)
Vector Synchronize	SYNCV	PNで示される連続するPK個のすべてのプロセスおよび親プロセスと同期をとる(PN,PKは命令で指定)
Terminate	TERM	この命令を実行したプロセスを終了する．
Vector Terminate	TERMV	ベクトルモードの実行を終了する．
Wait	WAIT	Host Interfaceからの割り込みをまつwait状態に移る．
Check Wait	CWAIT	PNで示される相手プロセスがtrace wait状態になるのを待つwait状態に移る(PNは命令で指定)
Continue	CONT	PNで示される相手プロセスのtrace wait状態をぬけ出させる．(PNは命令で指定)
Trace Flag Set/Reset	TRSR	PNで示される相手プロセスのtraceフラグをセット/リセットする．(PNは命令で指定)
Interrupt Routine Declare	IRD	プロセスの割り込み処理ルーチンの先頭番地を宣言する．
Interrupt Enable Flag S/R	IEFSR	割り込み可能フラグをセット/リセットする．(Host Interfaceからの割り込みを許可/禁止する)
Interrupt to Host Interface	ITH	Host Interfaceに割り込み情報(命令で指定)を伝え，ホスト計算機に何らかの処理を依頼する．
Return from Interrupt	RTI	割り込み処理ルーチンの最後にあって，元のプログラム(割り込まれたプログラム)の実行に戻る．

◎ブランチ命令

MELPIP-1のブランチ機能は，各命令毎のブランチ・フィールドで指定する条件付きスキップ動作の他に，表5に示すような各種の特殊ブランチ命令が用意されている．これらの命令を使う事により，全アドレス空間への直接ブランチ，コンデション・コードによる条件付きブランチ，レジスタの値による間接ブランチ，レジスタの値によるインデックス・ブランチ，およびサブルーチンとのリンクを行なう事ができる．

表5　ブランチ命令

名称	ニーモニック	機能
Branch & Link	BAL	現在の番地PCをスタックにプッシュし，NPCで示される番地にブランチする．(NPCは命令で指定)
Return from Subroutine	RTS	スタックをポップし取り出した番地へ戻る．
Branch on Condition Set	BCS	制御プロセッサ内のCondition Codeのうち Maskで選択されるビットがすべてもセットされていればADへブランチし，そうでなければ次の命令を実行する
Branch on Condition Reset	BCR	制御プロセッサ内のCondition Codeのうち Maskで選択されるビットがすべてリセットされていればADへブランチし，そうでなければ次の命令を実行する
Index Branch U	IDXU	Uレジスタ RGの下位CNビットをADにかえた番地にブランチする．(RG,CN,ADは命令で指定)
Index Branch V	IDXV	Vレジスタ RGの下位CNビットをADにかえた番地にブランチする(RG,CN,ADは命令で指定)
Indirect Full Branch	IFB	連続した2ワードのレジスタRG,RG+1の示す番地へブランチする．(RGは命令で指定)

◎Test & Set 機能

主記憶リクエストと制御記憶リクエストにはTest & Set命令があり，主記憶と制御記憶にあるデータに対して，複数のプロセスからの共有と相互排斥を容易に行なえる．

◎リエントラント・サブルーチン

ブランチ命令の中のBAL命令は戻り番地をAP内のスタックに格納してブランチするため，1個のサブルーチンを複数のプロセスで同時に利用するリエントラントなサブルーチンを作る事ができる．

◎トレース機能

プロセス制御命令の中のTRSR，CWAIT，CONT命令を使う事により，他のプロセスの動作を強制的に止めたり，1命令実行する事に停止させて刻々の情報をトレースする事ができ，デバッグ時やプログラム動作解析を行なう時に有効である．

4.2 並列プログラムの例

マイクロ・タスクがMELPIP-Iに発生してから，いくつかの並行プロセスを発生しながら処理を進めていき，処理が終わるという生成から消滅までの過程を例によって示す．

例1　並行処理（MIMD）

ホスト計算機からの依頼によって発生したマイクロ・タスクはCPによってルート・プロセス（PN＝0）を発生し，実行開始する．ルート・プロセスはFAD命令によってPN＝1，2の子プロセス，PN＝3の孫プロセスを発生していく。これらのプロセスはそれぞれ別のAPが割りあてられて並行して処理を進めていくが，相手プロセスを指定してSYNC命令を実行すると相手プロセスが自プロセスを指定してSYNC命令を実行するまで動作を停止する．

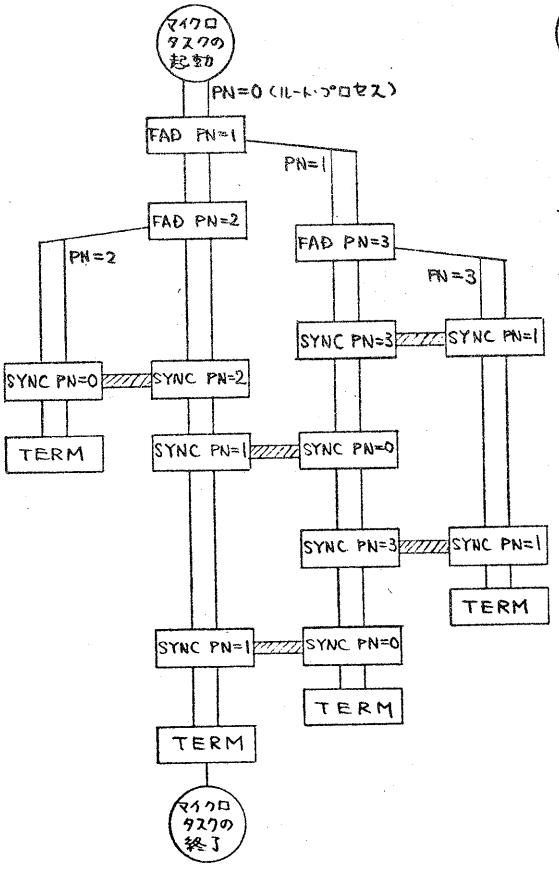


図10　並列処理の例

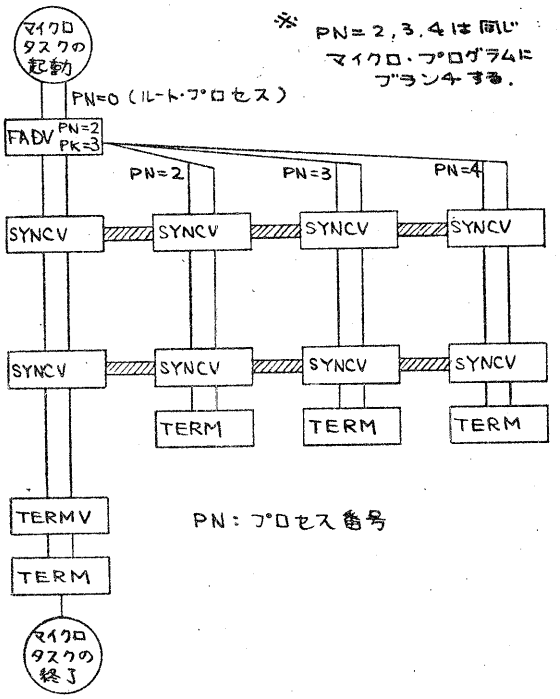


図11　ベクトル処理の例

同期がとられると両プロセスとも動作を再開し，再び並行して処理を行なう．　プロセスが動作を終了する時TERM命令を実行し，プロセスを消滅させるが，ルート・プロセスがTERM命令を実行した時はマイクロ・タスクの終了としてホスト計算機に伝えられる．（図10参照）

例2　ベクトル処理（SIMD）

MIMDの環境はFAD命令，SYNC命令によって実現しうるが，アレー処理のようにデータだけが異なり同じ命令を実行するプロセスに対してはFADV命令，SYNCV命令が用意されている．　これらの命令は，オペランドで指定するPN，PKにより，プロセス番号PNから（PN＋PK－1）個のプロセスを1度に発生して同じマイクロ・プログラムを実行させたり，これらのすべてのプロセス，および親プロセスの間で同期をとる機能がある．

例3　逆行列の計算例

MELPIP-1において，N次の実行列Aの逆行列をGauss - Jordan法で求める場合のプログラム例を示す．

Gauss - Jordan法では，行列Aと単位行列ENを並べた行列をBとし，第kステップではBの第（k＋1）列から第（k＋N）列までのN列からなる行列$A^{(k)}$についてBの（k,k）要素b_{kk}をピボットとする掃出しを行なってBを変更していくという操作を，kについて1からNまで行なうとBの右半分にAの逆行列が求まるという性質を利用している．

これをMELPIP-1で並列化して行なうために，$A^{(k)}$をN行，N／8列の8個のブロックに分割し，各ブロックの掃出しを1つのマイクロ・プロセスが受けもつようにしてベクトル処理を行なうマイクロ・プログラムの構成にする．

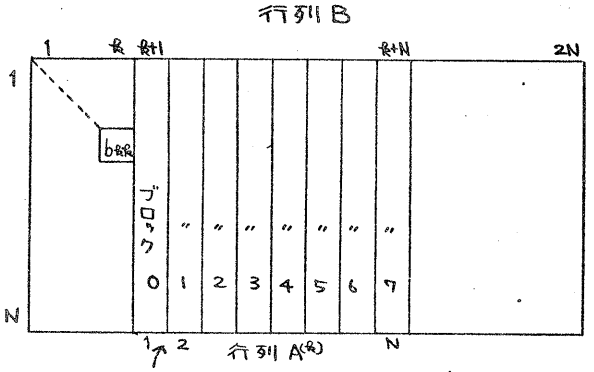


図12　アルゴリズムの説明

このマイクロ・タスクが起動されると，ルート・プロセスはすぐにFADV命令により7個の子プロセスを生成する．　子プロセスはBAL命令によってブロックの掃出しを行なうサブルーチンに入るが，ルート・プロセスが各定数を子プロセスに伝え，ルート・プロセス自身もこのサブルーチンに入るまでSYNCV命令の所で同期まちになる．　以後，各ステップ毎に全体で同期をとりながら，Nステップまでを実行して逆行列が得られる．　但し，子プロセスはこのサブルーチン内で終了するが，ルート・プロセスだけは結果の出力等の後処理をした後，マイクロ・プロセスの実行を終了する．　図12に並列化したアルゴリズムの説明図を，図13，図14にプログラムの概略フローチャートを示す．

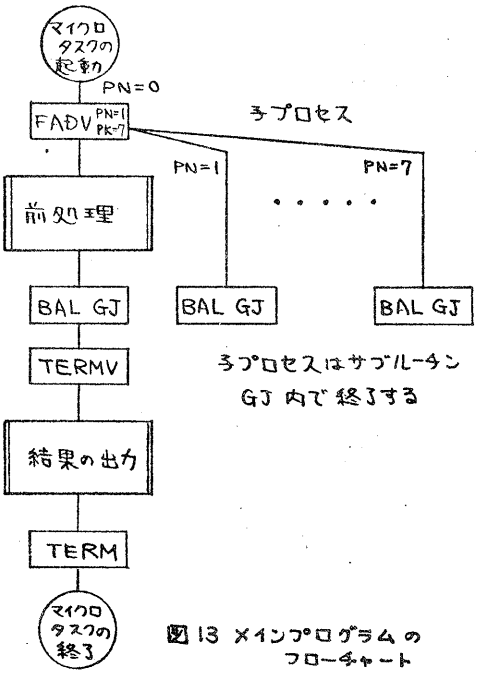


図13　メインプログラムのフローチャート

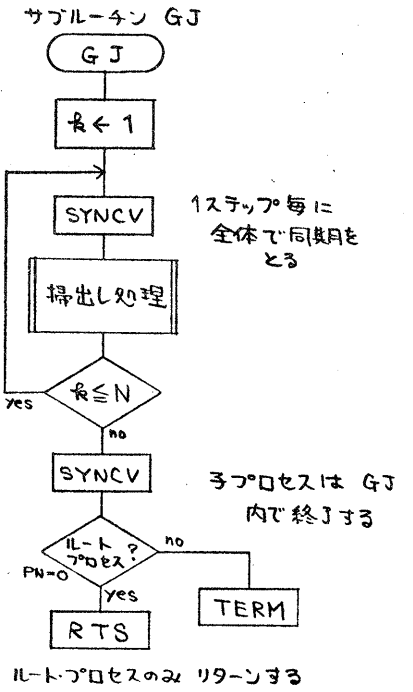


図14　サブルーチンのフローチャート

5. おわりに

MELPIP-Iは現在開発中であるが，CCDメモリを除いて大型カード（280×300mm）22枚で構成され，ロジックは主としてショットキーTTL，FPLAを，また主記憶バッファ・メモリや制御記憶には平均アクセス時間が70 nsecの高速MOSメモリを使用している．FPLAは設計労力，素子数，消費電力等の減少を狙って使用したものであり，ランダム・ロジックの70％程度をFPLAで実現した．また効果として，1石あたり10～20石のSSI／MSIを吸収でき，全体の素子数は1/2～1/3になっている．

ソフトウェアのサポート体制はMELPIP-I用のクロス・マイクロ・アセンブラがMELCOM 7500で動作している．現在，ホスト計算機において動作する管理プログラムを開発中である．

MELPIP-Iの用途は，並列処理，アレイ処理，工業プロセスの制御用等，並列性が有効な諸分野であり，これらに実験的に適用し，その効果を評価したいと考えている．

今後の問題点としては，ユーザのマイクロ・プログラミングを容易にするために，並列プロセスの動作記述が可能な高級言語や，多重マイクロ・プログラムを制御するファームウェアのモニタ等について検討を進めたいと考えている．

謝辞

MELPIP-Iの開発にあたり御指導，御協力頂いている三菱電機コンピュータシステム工場開発部の米沢部長，武蔵部長代理，橋本部長代理，長沢課長代理に感謝します．

参考文献

1) L.C.Widdoes, "The MINERVA Multi-Microprocessor," 3rd Annual Symposium on Computer Architecture, pp. 34－39 Jan. 1976

2) K.Murakami, S.Nishikawa, and M.Sato, "Poly-Processor System Analysis and Design", 4th Annual Symposium on Computer Architecture, pp. 49－56, 1977

3) 岡田，田島，森，"新しい可変構造プロセッサアレイ"，信学技報，EC 76－81，1977

4) V.R.Lesser, "Dynamic Control Structure and Their Use in Emulation", Ph.D. dissertation, Rep. CS309, Computer Science Dep., Stanford Univ., Stanford, CA, Sept. 1972

5) 房岡，平山，朱雀，"多重マイクロプログラム制御方式による並列演算プロセッサ"，昭和52年度情報処理学会全国大会講演論文集，pp. 433－434，講演番号 220

6) 房岡，平山，金森，"CCDを用いた主記憶階層化の試み"，昭和52年度電気関係学会関西支部連合大会講演論文集，pp. G220