オペレーティング・システム 50－7
（1991. 3. 15）

研究総説：並列コンピュータとOS

曽和将容
名古屋工業大学工学部

著者の研究室では，①マルチマイクロコンピュータ，②マクロデータフローコンピュータ，③マシン語レベルデータフローコンピュータとそのOS，④（パラレル）コントロールフローコンピュータ，⑤ハイブリッド並列コンピュータ，ユニバーサルコンピュータ，⑥リソースドリブンコンピュータ，⑦PNスーパスカラコンピュータ，⑧定並列コントロールフローコンピュータ，⑨BSCPコンピュータの研究を行ってきた．これらの研究から得られたノイマンコンピュータから並列処理コンピュータに通ずる基本概念であるコントロールフローという考え方，また，処理の基本概念をうまく表すデータフロー処理とそのOSの基本的な考え方について述べる．

PARALLEL COMPUTERS AND DATAFLOW OS

Masahiro Sowa

Department of Electrical Engineering and
Computer Science, Nagoya Institute of
Technology

Gokiso, Syouwaku 466, Nagoya, Japan

This paper describes the basic ideas of parallel computers and a data flow OS researched in our laboratory. The researched computers are 1. Multi-microcomputer, 2. Macro-dataflow computer, 3. Dataflow computer -DFNDR-, 4. Parallel control flow computer, 5. Hybrid parallel computer, 6. Resource driven computer 7. PN super-scalar computer, 8. Constant parallelism computer, 9. BSCP computer. This paper also describes the relation of a Neumann computer with the parallel computers.

1．まえがき

本研究室で研究を行った並列コンピュータは以下のようなものである．

①マルチマイクロコンピュータ

マクロプロセッサチップを８個使って構成したプロシジャレベルで平行処理を行うコンピュータ．

②マクロデータフローコンピュータ

①の実行制御をデータフロー原理によって行った並列コンピュータ．

③マシン語レベルデータフローコンピュータ

マッチングユニットとして中心にマルチポート連想メモリをもつ機械語レベルで並列処理を行うデータフローコンピュータ

④（パラレル）コントロールフローコンピュータ

データフローコンピュータの経験から創造されたコントロール制御による並列コンピュータ．フレキシビリティがあるが故にデータフローコンピュータより将来性があると考えられている．

⑤ハイブリッド並列コンピュータ

ユニバーサルフローコンピュータ

データフローコンピュータとノイマンコンピュータを組合せ，データフローコンピュータの柔軟性の無さとシリアル処理の遅さを解決しようとしたコンピュータ．後に，パラレルコントロールフロー処理が可能に改良され，データフロー，ノイマン処理，パラレルコントロールフロー処理が可能な”ユニバーサルフローコンピュータとなった．

⑥リソースドリブンコンピュータ

ハイブリッドコンピュータの概念をもっと一般化し，命令の実行が計算資源がすべて到着したときに始まると言う原理に沿って行うコンピュータ．②～⑤までのコンピュータはこの一つの概念で統一的に把握することが出来る．

⑦ＰＮスーパスカラコンピュータ

ノイマンコンピュータの命令を転送，演算，分岐などのように機能別にわけ，この機能別命令間の並列性をコンパイル時に抽出して並列実行する比較的小並列，細粒度の並列コンピュータ．

パラレルノイマンコンピュータという名前で研究されてきたが，４年後に表れたスーパスカラコンピュータと良く似た原理であるのでこの名前にかえられた．静的に並列度を検出するので，スーパスカラコンピュータより性能が良いことを，シミュレーション結果が表している．

⑧定並列コントロールフローコンピュータ

④の（パラレル）コントロールフローコンピュータのプログラムの実行形態を定並列にしたもの．並列処理にありがちのオーバヘッドが非常に少ないという特徴を持っている．太いノイマン処理コンピュータとも呼ばれる．

⑨ＢＳＣＰコンピュータ

並列コンピュータの要素プロセッサ間のデータ通信の効率化をめざして開発されたコンピュータ．このデータ通信を放送プログラムとリスニングプログラムの相互動作で行う．

以後これらの研究から生まれ整理されてきたコントロールフローの基本概念とノイマンコンピュータの関係，データフローコンピュータとそのＯＳの研究結果，また，それぞれのコンピュータの特徴について述べる．

2．資源駆動計算モデル

データ駆動，コントロール駆動，要求駆動など種々の計算モデルが考えられているが，資源駆動とはこれらを統一的に捕らえることが出来るモデルである．

計算資源駆動とは，図１に示すように，計算に必要なものすべてを資源と考え，この資源がすべて揃ったとき，計算が始まるとする考え方である．この計算資源が揃うA1やA2，A3のような場所のことをアクタと呼んでいる．たとえば，いま，計算を行うのに必要な資源のうち，データだけがアクタに揃っていないとすると，この計算はデータが到着したとき始まる．計算の実行が終わると，これら資源のうち原則としてどれか１個を出力する．この最後まで揃わない資

源のことを，最欠乏資源と呼んでいる．計算に必要な資源にはデータのほか，命令，プロセッサ，計算要求，計算許可（または，コントロール），メッセージやメモリ等あらゆる物が考えられる．最欠乏資源は，必ずしも１個である必要はないし，入力と出力が同種類の資源である必要もない．

３．データフロー処理とコントロールフロー処理

資源駆動で処理すべきデータを最欠乏資源としてプログラム上を流せば，そのプログラムはデータフロー処理用のプログラムとなる．データを出力するアクタとそのデータを入力するアクタの間には，生産者と消費者の関係がある．この関係をデータの従属性という．すなわち，データフロープログラム（ＤＦプログラムと呼ぶ）はデータの従属性にそって書かれ，このプログラム上をデータが流れることによって，データが加工され，目的のデータがえられる計算方法である．

図２（ａ）は，

y = (ab + cd)(e - f)

を計算するＤＦプログラムである．アクタの中に書かれた＊や＋などの記号は命令のファンクションを表す．アーク上の黒丸はデータを表し，このデータはデータトークン，または簡単にトークンと呼ばれる．アクタは必要なデータがすべて揃ったとき命令の実行を始め，命令の実行が終わると結果のデータ出力する．これをアクタの発火規則という．同図（ｂ）は（ａ）のデータフロープログラムをそのままコントロールプログラム（ＣＦプログラムグラフと呼ぶ）に書き換えたものである．命令の位置やそのつながりかたは，同図（ａ）のままである．データフローの場合は，プログラム上をトークンと呼ばれるデータが流れたが，コントロールフロープログラム上を流れるのはコントロール（実行許可権）である．このコントロールはコントロールトークンと呼ばれる．プログラムのアクタ，たとえば，Ａ1の

Ｍ　ａ，ｂ，ｖ

は，ａ番地とｂ番地の内容の積を求め，その結果をｖ番地へ格納することを意味する．アクタＡ5は，アクタＡ3とＡ4からコントロールトークンが出力されたとき実行可能となり，Ａ５が実行を完了するとコントロールトークンをその出力アークに出す．コントロールトークンの動きとアクタの発火規則は，データフロー処理と同じであるが，コントロールフロー処理では，アクタの中に演算子とそのオペランド，すなわち，メモリ変数またはメモリ番地を書かなくてはならない．同図（ａ）のプログラムは，（ｂ）のデータフロープログラムをそのままコントロールフロー処理になおしたものであるので，データの従属性はデータフロープログラムの時のまま残されている．たとへば，アクタＡ1でｖに格納されたデータがアクタＡ3で使われている．しかし，たとへば，アクタＡ1とＡ2間のアークによって決められている従属性はコントロール従属性と呼ばれる物であり，この従属性は，アクタＡ1の次にＡ2を実行しなければならないと言うことを表しているだけである．したがってコントロールの従属性は，データの従属性とは無関係に決められるものである．データの従属性はアクタ内のメモリ変数によって表されている．したがって，メモリ変数によって表されているデータ従属性をまもってコントロール従属性を決める限り，そのプログラムによって計算される結果は正しいものとなる．

４．コントロールフロー処理とノイマン処理

コントロール従属性を，”アクタＡ3は，少なくともアクタＡ1の後で実行されなければならない”という従属性と考えると，図２（ｂ）のコントロールフロープログラムは，同図（ｃ）のように書き換えることが出来る．このプログラムのアクタＡ4をＡ3とＡ5の間に入れると同図（ｄ）のようになり，直列処理のプログラムが得られる．これらのプログラムは，同図（ａ）の

プログラムとは大きく形が異なっているが，データフロープログラムにあったデータの従属性は，そのまま命令のメモリ変数として保たれている．（d）のプログラムに次の二つの高速化手法を導入すると，ノイマン処理となる．

①プログラムカウンタの導入

同図（d）の直列コントロールフロー処理プログラムを図3（a）のようにならべるとプログラムがわかりやすくなる．このように並べると同図（b）のようにアクタ間のアークを省略しても命令の実行順序は一目でわかる．命令に実行順に1，2，3，・・・と番号（アクタの名前に相当）をつけておき，1個のカウンタを用意し，ひとつの命令の実行が終わる毎にカウンタの値を1だけ増せばスケジューリングができる．実際には，命令が格納されているメモリの番地を命令の名前としているので，その命令の実行が終わったときその命令を構成している語数だけ増やすことになる．このカウンタがプログラムカウンタである．カウンタは構造が簡単であり，そのインクリメント非常に高速である．

2）メモリの階層化

計算機は大量のメモリを必要とする．一般にメモリの容量とそのアクセスの速さとはその構造から反比例する．それゆえ，メモリを階層化して高速で小量のメモリと低速であるが大要量のメモリを組合せ，疑似的に高速で大量のメモリを用意することが考えられてきた．ノイマンコンピュータはこの階層化を，累算器，またはアキュムレータとして実現している．図3（b）にレジスタを導入したプログラムを示す．破線で囲った命令が，アキュムレータ導入のため新たに追加されたロードやストアなどのアキュムレータ操作命令である．この処理がノイマン処理である．このプログラムでは命令数が9と同図（a）の5に比べて倍近く増えているが，レジスタにロードしたデータが長く使えれば使えるほどその処理速度が増えることになる．最近では，この考え方をもっとすすめたロードストアアーキテクチャがＲＩＳＣプロセッサなどでは取り入れられている．

5．自由並列処理の問題点

図2（a）（b）（c）のようにプログラムの並列度が変わるプログラムを自由並列度プログラムといい，このような処理を自由並列処理という．

図（b）のアクタはアセンブリ言語風には

Ｍ　a，b，v；A3．Ｌ．2

と表現される．セミコロン後のA3．Ｌは，このアクタの実行終了後にトークンを出力する送出先のアーク（A3の左入力アーク）を示し，2はA3が2本の入力アークを持っていることをあらわしている．命令を実行するためのスケジューリングは次のように行われる．トークンはその行き先アクタ名（出力アーク）をつけたトークンパケットと呼ばれるパケット形式で送られてくる．たとえば，アクタA1からアクタA3の左側入力アークに送られるトークンパケットを

＜A3．Ｌ＞＜ｔ．2＞

の形式で表す．A7は送り先アクタ名，Ｌは左側入力アーク，ｔはトークンを表し，＜ｔ．2＞の2は，A3の命令を実行するには2個のトークンを必要とすることを表す．A3のアクタは，

＜A3．Ｌ＞＜ｔ．2＞

＜A3．Ｒ＞＜ｔ．2＞

の2個のトークンパケットを受け取り，これらから

＜A3＞＜ｔ．Ｌ＞＜ｔ．Ｒ＞＜2＞

の形をしたパケットを構成し，必要なトークンが2個が到着していることを調べることによって，実行可能かどうかを決定する．この操作は実行可能命令検出処理と呼ばれている．

並列コンピュータ内では，多くのアクタの実行が同時に行われる．それゆえ，トークンパケットが多くのアクタから送出されており，その中から，送り先アクタ名が同じトークンパケットを捜し出し集めると言う操作が必要になる．並列コンピュータでは，このようなことが同時

に多数のトークンパケットに対してなされなければならないので，このためにマルチポート連想メモリが必要になる．マルチポート連想メモリはハードウエアが非常に複雑であるし，また，オーバヘッドも大きい．これが自由並列コンピュータのハードウエアを複雑にし，パフォーマンスを下げるので，このことが自由並列処理の大きな問題点である．

並列プログラムでは，プロシジャやサブルーチンが複数の箇所から同時に呼ばれることがある．この場合これらはお互いに干渉を起こす．同様なことはループ処理の場合もおこる．この干渉を避けるには，アクタがどのプロシジャ呼び出しに関連して実行されているかを区別する必要がある．この区別はインスタンスの区別とよばれ，トークンに色，または，タグをつけ，プロシジャを呼ぶ毎にその色またはタグを付け換えることによってトークンを区別し，それと同時にその色に応じたデータ格納領域を用意することによって行われている．このようなインスタンスの区別手続きは，オーバヘッドの大きな要因となる．

コントロールフロー処理では，メモリアクセスの回数が大きくなり，これがコントロールフローコンピュータを高速にすることを妨げる．上で述べたように，コントロールフロー処理の命令は基本的に３アドレス命令となる．それゆえ，アキュムレータ，または，汎用レジスタを用いたノイマン処理よりも命令長が長くなり，その分だけ命令フェッチ時にメモリアクセスをする回数が増える．また，実行時には計算するためのデータ２個をメモリから取り出し，その結果をメモリに格納するので３回のメモリアクセスが起こる．アクタは，トークンの送り先アクタ指定部分を持っている（a11，a12等）．これも命令長を長くし，フェッチ時のメモリアクセス回数を増やす．このように命令実行スケジューリング，インスタンスの区別，メモリアクセス回数の多さが自由並列コントロールフローコンピュータの大きな問題点であった．

６．定並列（ＣＰ）コントロールフロー計算機

コンスタント・パラレリズム計算機は，並列処理の並列度を自由に変化させることをあきらめて処理の並列度を一定に制限し，その代償として，命令実行スケジューリングの高速化，メモリアクセス量の減少化，インスタンスの非区別化を行って，コンピュータ構造の簡単化と処理の高速化をはかろうとするものである．このような処理方式をConstant Parallelisum control flow processing，簡単にＣＰ処理，定並列処理と呼んでいる．

①基本原理

図４（ａ）のようなプログラムを，コンスタント・パラレリズム・コントロールフロープログラム（定並列コントロールフロープログラム，以後簡単に定並列プログラム）と呼ぶ．アクタの発火規則はコントロールフロー処理の場合と同じで，アクタのアークのそれぞれに，コントロールトークンが到着したとき発火可能（実行可能）となる．このプログラムで横に並んだアクタがほぼ同時に実行されると考えると，定並列プログラムによる処理は，これらの横に並んだアクタ群単位で逐次的に実行されるシリアル処理であると考えることが出来る．一方，ノイマン処理は，同図（ｂ）に示すように，一アクタ単位の逐次処理である．したがって，この処理を太い逐次処理と考えて，俗に”太いノイマン処理”と呼んでいる．定並列プログラムが太い逐次処理を表すと考えると，ノイマンコンピュータと同様にインスタンスの区別が不必要となり，その分オーバヘッドが減る．なんとならば，逐次処理では，プロシジャを同時に呼び出すことが無いからである．また，アクタを実行順にメモリに格納しておけば（分岐命令を除く），実行スケジューリングにカウンタを用いることが出来る．

各列のアクタはその列内で順序づけされている．この場合，あるアクタの実行が終わってト

ークンを別の列のアクタに送るとき，その送り先をアクタ名で指定しなくともそのアクタの属している列の名前で指定できる．列の本数はアクタの数より格段に少ないのが一般的である．したがって，トーク送り先指定に必要なビット数が少なくなり，それゆえメモリアクセス回数が減り高速化がはかれる．

②ＣＰ処理のアクタ実行

定並列処理でもしＶＬＩＷコンピュータのように，横に並んだアクタ群を単位として逐次処理をするようにコンパイル時に決めると，次のアクタ群の実行を始めるまえに，実行中のアクタ群内の統べてのアクタの実行が終わっていることが必要になる．この場合，アクタ群は，そのなかの一番遅いアクタの実行が終了したとき実行の終了となるので，アクタ群の実行時間は，その中に含まれる一番時間のかかるアクタの実行時間となってしまう．この欠点を持ち込まないために本コンピュータでは，同一スレッドのアクタ実行スケジューリングは，ノイマンコンピュータのプログラムカウンタと同様な方法で，スレッド間のアクタ実行スケジューリングは，トークンカウンタと呼ばれる特定スレッドからのトークンの入力数を記憶するカウンタを用いて行う．この方法では，他のスレッドからアークが入力されていないアクタは，プログラムカウンタの値がそのアクタをさしていれば，そのアクタは実行可能なアクタである．他のスレッドからアークが入力されているアクタは，そのアクタをプログラムカウンタの値がさしたとき，アークを入力しているスレッドに対するトークンカウンタが１以上で，すでにトークンが送られていることを示しておれば実行可能である．トークンカウンタの値が０の時は，アクタは実行可能でないので，この値が１になるまで命令の実行は遅らされる．

プログラムカウンタのカウントアップは高速であるし，トークンカウンタのカウントアップの高速であるので，この方式の命令の実行スケジューリングは超高速である．

７．ＰＮスーパスカラコンピュータ

①基本原理

パラレルノイマンコンピュータとは，ノイマンコンピュータの命令を，演算操作，キャッシュ操作，分岐操作などの機能別に分類し，これらの命令を種類ごと並列に実行するコンピュータである．このコンピュータは，もとのプログラムはノイマンプログラムのシリアルプログラムであり，これを機能別に３～７程度に定並列化するので，計算モデルとしては定並列コントロールフローコンピュータでもある．

(a+2)が正ならば(a+b)*(1-b)を計算し，(a+b)が負ならば(a+b)*(1+b)を計算するＰＮプログラムは図５のようになる．

④簡単な実行例

図５のＰＮプログラムは，図６に示されるように実行される．ここで種類別命令間の矢印はトークンの流れを表す．

３種類の命令が並列に実行されているが，ここで注目すべきことは，ノイマンプログラムでは，MUL R2,R3,R4 と MOV R4,m1 の後の６番目と７番目で実行されていた比較命令と分岐命令（CMP R3,0; BGT ESC）が，最初から実行され始めていることである．このように，各命令は，他の命令の実行に無関係にフェッチされ解釈されて，その後，もし，他の命令との関係がでてきたときには，その命令からのトークンの到着を待つという形で実行を wait する．この簡単な例では，実行時間は，ノイマンコンピュータの実行時間に比べて 1／2 の実行時間となっている．

８．マクロデータフローコンピュータ

本データフローコンピュータは，ノードとよばれるプロシジャ程度の大きさのプログラムを単位として，データ駆動原理にしたがってプログラムを実行するコンピュータである．プログ

ラムは複数個のモジュールがあつまったモジュールという単位で取り扱われる．ノード間のデータフローは次のようなモニタコール命令を置くことによって定義される．

SD(CM, TM, DN, TB, TA, Chain)

RD(CM, TM, DN, TB, TA, Chain)

ＳＤは，データを送信するためのモニタコールであり，ＲＤはデータを受信するためのモニタコールである．ＣＭはそのモニタコールを行ったモジュールプログラム名である．ＴＭは目標モジュールプログラム名で，ＳＤの場合は送信先，ＲＤの場合は送信元モジュールプログラム名となる．ＤＮは，データに名をつけた場合の識別に用いられる．ＴＢ，ＴＡはデータエリアを指定するもので，それぞれデータの転送バイト数と先頭アドレスを示す．Ｃｈａｉｎは，いくつかのモニタコールを同時にモニタに知らせて，一括して処理を行うために必要となる．

本システムはＭＣ６８００マイクロプロセッサチップを要素プロセッサとして最大８台構成で実現され，並列処理制御に対するデータフロー制御の適応性を劇的なフィーリングで示した．実際には，データ駆動のためのデータの送受とマッチングに１０～２０ｍｓの時間が必要であり，そのため，一つのノードの実行時間がこの時間を無視できるぐらいに長い必要性を示した．このことから最粒度のデータフロー処理には，本質的な合理的で高速なデータの送受と，命令スケジューリング機構の開発の必要性を示した．

９．マシン語レベルデータフローコンピュータ

本コンピュータDFNDRは図７のように，トークンメモリＴＭ，プログラムメモリＰＭ，ファンクショナルユニットＦＵ，ホストコンピュータＨＣよりなっている．ＴＭは，プログラム上を流れるデータであるトークンパケットを格納するマッチングメモリであり，相互排除，マルチポート連想メモリの機能を持つ．プログラムメモリＰＭは，ノードパケット，したがって，データフロープログラムグラフを格納するメモリで，リードに関してマルチポートメモリとなっている．このアーキテクチャでは，ＴＭが大きな役割をしており，プログラム上を流れているデータのうち同じアクタに行くデータを集め，そのアクタを実行するに必要なデータが集まったかどうかを識別し，実行可能であればそれを実行可能アクタ用データ集合として（完全トークンパケットＣＴＰと呼ぶ）格納する．ＦＵはこのＣＴＰを取り出し，このＣＴＰで示されるアクタをＰＭから取り出して実行する．実行結果のデータは，そのデータの行き先が付加されたトークンパケットという形でＴＭに送られる．このことがすべてのＦＵで繰り返され，データフロープログラムの実行が並列に進んでいく．

１０．プログラムカウンタを持ったデータフローコンピュータ

プログラムカウンタは，直列処理の命令を高速に実行スケジューリングする．それゆえ，データフローコンピュータも直列処理に限れば，プログラムカウンタを導入でき，高速処理が出来る．すなわち，データフローコンピュータで並列処理をあきらめ，データフローコンピュータを直列処理専用にすれば，ノイマンコンピュータより高速なコンピュータが得られるはずである．理論的な研究によれば，データフローコンピュータにプログラムカウンタを導入することにより４０％の直列処理の高速化が得られるという結果が得られている．このコンピュータは，ノイマンコンピュータより速い．

しかしながら，データフロー処理ではデータ従属性のある処理しかプログラムが書けないし，直列にデータ従属性のある処理はほとんどないので，直列データフローコンピュータを汎用コンピュータとして使う方法は実用的ではない．

１１．ハイブリッドデータフローコンピュータ

データフローコンピュータにノイマン処理を導入すると，ノイマンコンピュータの直列処理の高速性によって並列コンピュータの直列処理

の遅さがカバーされる．また，ノイマン処理は，コントロールフロー処理を計算原理にしているので大きなフレキシビリティを持っており，このためデータフロー処理では出来なかったデータ依存性の無い処理が出来るなど，データフローコンピュータに大きな柔軟性を付け加えることが出来る．

たとえば，レジスタの内容をｎ回デクリメントするプログラムをデータフローコンピュータとノイマンコンピュータで実行すると，ノイマンコンピュータの方が３．４３倍も速くなる．

ノイマン処理が不得意とする並列処理を，データフローコンピュータに行わせ，データフローコンピュータが不得意とする処理をノイマンコンピュータに行なわせようというのが，ハイブリッドコンピュータである．

１２．ユニバーサルコンピュータ

データフローコンピュータは，並列処理が出来るが柔軟性に欠ける．コントロールフローコンピュータは，柔軟性があるがそれだけにサイドエフェクトがあり，また遅い．ノイマンコンピュータは，直列処理が速く柔軟性があるが並列処理が出来ない．それぞれのコンピュータは，それぞれ特徴と欠点を持っている．これらを組み合わせ，お互いの特徴を生かそうというコンピュータがユニバーサルフローコンピュータである．

一般的に言えばコンピュータは，図８に示すようにどの命令を実行するかを決めるスケジューラと，その命令を実行するプロセッサ，命令やデータを格納するメモリ，これらを接続するバスよりなっている．ノイマンコンピュータでは，プログラムカウンタがスケジューラ，プログラムカウンタを除いたＣＰＵがプロセッサである．メモリはメモリであり，データとプログラムを格納する．データフローコンピュータでは，スケジューラはトークンメモリであり，スケジューリングをすると同時にデータをも格納する．プロセッサは複数のプロセッサ，メモリはプログラムのみを格納するメモリである．コントロールフローコンピュータでは，スケジューラはトークンメモリ，プロセッサは複数のプロセッサ，メモリは，プログラムとデータを格納するメモリである．したがって，たとえばスケジューラを，コントロールフローコンピュータのスケジューリングをすると同時に，プログラムカウンタのようなスケジューリングを行えるように構成し，ついでに計算の途中結果を格納できるように構成すれば，３種類の処理をミックスして実行できるコンピュータが構成できる．

ユニバーサルコンピュータのハードウェアのユニットは，ほとんどデータフローコンピュータのハードウエアに含まれている．したがって，データフローコンピュータにプログラムカウンタを追加し，データフローコンピュータのメモリに，プログラムとともにデータを格納出来るようにすれば，これだけでユニバーサルフローコンピュータが構成できる．

単純な並列演算はデータフロー処理で，並列履歴処理や並列構造体処理はコントロールフロー処理で，シリアル処理や決定的処理，リアルタイム処理，ノイマンコンピュータとのコンパチビリティをとる仕事は，ノイマン処理部分でというのが，ユニバーサルコンピュータにおける処理割当の基本戦略である．

１３．データフローＯＳ（ＳＳＳ－ｄ）

①　まえがき

コンピュータを並列処理コンピュータとして完成させるためには、その上で走る並列システムプログラムを開発しなくてはならない。われわれは，データフローコンピュータを例にとって，sss-d (space sharing system for dataflow computer)と呼ばれる並列オペテーティングシステムの研究を行ってきた．すなわち，１）割込みをデータフローコンピュータでどのように扱うか２）割込みに関連の深い入出力機構に関する研究，３）ファイルなどの資源をどのよ

うに処理するかという資源管理に関する研究，4）sss-dの基本ファイルシステムの理論的構築，5）プロセス管理とプロセス間通信に関しての考察を行ってきた．

このような考察からデータフローコンピュータのプログラムをオペレーティングシステムを通して実行する場合には，単独で実行するときは異なり，大きな問題を生じることが解ってきた．すなわち，データフローコンピュータが，マルチプログラミングのようなＯＳを持つと，それは多くのユーザーから使われる共有資源としての性格を持つようになり，この場合には，”実行の永久遅延”と呼ばれる現象が発生する．

② 割り込みと入出力

ノイマンコンピュータでは，割り込みが起こると現在実行しているプログラムの実行を止めて，割り込みプログラムを実行するという実行制御を行ったが，データフローコンピュータのような並列処理コンピュータは多くのプロセッサを持っているので，割り込みが起こったとき割り込みプログラムを実行することはするが，必ずしも実行中のプログラムの実行をやめるとは限らない．本研究では，データフローコンピュータにおける割り込みを次のように分類して定義してきた．

. 割り込み：任意のプログラム実行中に，緊急に
  処理されるべき事象が発生したとき，その事象を優先して処理すること．
. 停止割り込み：実行中のプログラムの実行を停止して，割り込みプログラムを実行する割り込み．
. 通常割り込み：実行中のプログラムの実行を中止せず，割り込みプログラムを実行する．

データフローコンピュータでは，命令の実行開始は，データであるトークンをフェッチすることによりなされる．それゆえ，優先してプログラムを実行させるには，トークンを優先的にフェッチする機構を設けなければならない．このため，トークンに優先順位を知らせるためのタグをつけることを提案した．また，実行中のプログラムを停止させるには，プログラムを区別する機構が必要である．そのため，トークンにプロセス識別用のタグをつけることを提案した．

入出力に関する命令もデータフロー原理によって行われ，入出力は専用プロセッサ(I/Oプロセッサ）によって行われるとすると，何等かの方法で入出力用のトークンと一般処理用のトークンを区別する必要が生じる．これは，トークンにＩ／Ｏ識別用のタグをつけるか，Ｉ／Ｏ用のトークン格納場所を変えるかすることによって行われる．

③ 資源管理

資源管理を実現する場合，少なくとも履歴処理の実現，共有データの実現，共有データへのアクセスの平等性の３項目を可能にしなくてはならない．

１）履歴処理の実現

データフローコンピュータは，原理的にメモリを持たない．それゆえ，履歴処理，即ち，過去の実行結果を記憶し，その記憶内容によって現在の実行結果が影響を受けると言う処理を行うには，記憶機能を実現しなくてはならない．記憶機構は，プログラムによってループを構成するか，メモリを導入するかの２種類の方法によって実現できる．メモリを導入した場合には，１つの番地の内容をポインタ変数などの手段により２度以上アクセスすることになるため，次に述べる共有データともなり，副作用を生じる危険性が出て来る．

２）共有データの実現

データフローコンピュータでは，基本的にデータを共有すると言う概念が無い．データフローコンピュータでは，データであるトークンに色，または，タグをつけてインスタンスを区別する．１）で述べた履歴処理におけるデータは，共有データの１種ではあるが，この場合の共有

データは，同じインスタンスの中だけで有効である．従って，一般的な共有データを実現するには，異なったインスタンス間で共有できるデータの実現が必要である．本研究では，この共有データの実現を，a）色の階層化，特にスーパーカラーの導入，b）単色プロシジャによって行えることを提案してきた．

a）色の階層化

データフローコンピュータにおけるトークンの色に強弱の順位をつけ，この色の強弱によって，アクタが発火するかどうかを決定する方法である．本来アクタは，同じ色のトークンが揃ったときに発火するが，色の階層化を導入した場合には，ある基準のアーク上にあるトークンの色より弱い色のトークンが，他の必要なアーク上にきたとき発火する．この方法と，上で述べたループによる履歴処理を用いると，一世代前のトークンを送ったプログラムと現在のプログラム間でデータアを共有出来る．メモリを用いた履歴処理と色の階層化を用いると，多くのプロシジャ間でデータの共有を行うことが出来る．

スーパカラーとは，一番強い色のことである．この色を持つトークンは，全てのトークンと反応することが出来るので，一つのデータフローコンピュータ内のプロシジャ全てに共通な共有データが実現できる．

b）単色プロシジャ

プロシジャ呼び出しが起こったとき，常に一定の色をトークンに割り当てるプロシジャである．色が単色なので，あらゆるプロシジャから入力されたトークンが共有データのそれとマッチングすることが出来る．単色プロシジャ内では，トークンの追越しは，禁止されなければならない．

3）共有データへの排他処理と包含処理

データフローのアクタは，もともと排他処理の性質を持っている．たとえば，1）や2）の履歴処理や共有データ用の処置を行わない一般のデータフロー処理は，データ共有の概念を持たないために排他的であるし，たとえ，1）や2）の処理を導入した場合でも，入力アークに置かれるトークンを常に1個になるようにした場合には，排他制御となる．例えば，ループ法におけるループ上のトークンを1個に限定すれば，その処理は，自動的に排他的になる．

包含処理とは，排他処理とは逆の処理で，1つのデータを複数個のプログラムが同時に共有することである．包含処理は，スーパーカラーか上位の色を持つ複数個のポインタが，共有メモリ上の同一番地のデータを指すようにすることにより実現出来る．

4）共有データの平等アクセス．

共有データの平等アクセスは，ＦＩＦＯ処理が基本である．直列処理の場合は，同時に要求が到着することはないが，並列処理では，例えば，１，０００個以上のアクセス要求が，同時か，または，同時と見なされうる時間間隔で到着すると言うことが起こりうる．この時，到着順序を判別しＦＩＦＯ処理をすることは難しい．即ち，最短到着時間間隔とそれを認識する時間との関係で，もし，前者のほうが，後者より短い場合には，２つの要求の順序を決定できない．もし，この様な到着が次から次へと起こった場合，順序を決定できない要求が大量にたまり，たまった要求は，要求到着時間に関する情報を持っていないので，順序付けは不可能となる．そのためにａ）番号付け法，ｂ）タイムスタンプ法，ｃ）ブロックキュー法，ｄ）並列キュー法が考えられた．

ａ）番号法

これは，到着した要求に，到着順に番号をつける方法である．この方法は，番号付時間より到着時間間隔が大きいことがわっかている場合に有効である．

ｂ）タイムスタンプ法

この方法は，要求の発生時刻を要求トークンにつけ，共有メモリアクセス側では，到着している要求の中から，発生時刻の速いものをさがしサービスする方法である．データフローコン

ピュータは，非決定的処理を行うので，要求発生時刻と，その要求の到着時刻は，必ずしも比例関係にあるとは限らないが，この方法では，要求発生時刻順に近い処理が得られる．要求発生時刻の追加は，各々のプロセッサによって行われるので並列処理となり，並列処理結果の直列処理への集中が起こら無いので，この方法は現実的である．

ｃ）ブロックキュー法

到着要求のうち，ある時間間隔内に到着する要求，または，ある一定個数の要求をグループ化し，このグループ間で，ＦＩＦＯ処理を行う方法である．この方法の順序づけはグループ間だけでよいので，順序づけ回数が減り，順序づけの速度が遅くても問題ではなくなる．この方法では，グループ内の平等処理が守れないが，グループ間では守れる．

ｄ）並列キュー法

一般のキュー（シリアルキュー）を複数個並べ，複数要求の発生に対応させる方法である．

④　プロセス管理

ａ）状態遷移

ノイマンコンピュータでは，プロセスは基本的に３つの状態（実行可，実行中，待ち合わせ）を取ることが出来た．sss-dでも基本的には同じであるが，並列処理コンピュータの場合には，プロセッサの数が多いのでそれらの状態の中身は異なる．sss-dのプロセスの状態は以下のようである．

・実行可　：新たにプロセスが生成されるか、または、待ち合わせ状態からの復帰により、実行されるのを待っているプロセスの状態。

・実行中　：そのプロセスの実行を開始するのに必要な資源が割付けられ、実行を開始し，その実行を継続中の状態である．プロセスの実行は，多くのプロセッサで行われる故，プロセスが純粋に計算だけのプロセスとして実行されている場合と，計算とそれ以外の処理，たとえば，I/o処理とが同時に行われているている場合とがある．前者を純計算処理，後者を混合処理と呼ぶことにする．

・待合せ　：入出力などの待合せのため、プロセスが完全に停止している状態である。

ノイマン処理ではプロセッサが一つ故、実行中にあるプロセスはただ一つであったが、データフロー処理ではプロセッサが複数個であるので、多くのプロセスが実行中状態になることが出来る。

ｂ）プロセスの開始と消滅，停止制御

プロセスの実行開始は，プロセスにプロセス実行開始のための初期値データ（または，初期値トークン）を与えることによって出来る．ところがプロセスの終了に関しては少し工夫がいる．と言うのは，データフローコンピュータでは，プログラムの終了を確認する方法が確立されていない．それゆえ，プロセスの終了をＯＳが知るためには，プロセスの終了時に，それを知らせるための特殊トークンを出力するプログラムを，プロセスの最後に付加しなくてはならない．

プロセスの遷移において，実行中から待ち合わせへの遷移は，プロセスの停止という形を取る．ところがデータフローコンピュータでは，プログラムの実行を停止することは難しい．なぜならば，データフローのプログラムはトークンによって駆動されるので，プロセスを停止させるには，そのプロセスに関係するトークンを総て引き上げなければならない．このような制御は時間もかかるし，このような機能をデータフロープログラムによってデータフローコンピュータに組み込むことは，難しい．

プロシジャを停止するよい方法は，全てのプロセッサに指令を出し，そのプロセスの実行を停止させることである．そのためには，データフローコンピュータに，プロセッサ間通信と次に述べるプロセスの識別機構を用意しなければならない．

ｃ）プロセスの識別機構

sss-dは、並列プログラムを基本とする多重プ

ログラミング方式をとっている．そのため、同時に複数のプロセスが実行され，しかもそのプロセスの一つ一つが多くのプロセッサによって実行される．このようなプロセスを実行中にしたり，待ち合わせ中にしたりと言うように制御するには，少なくともプロセスを識別できなくてはならない．

d）優先順位処理

ノイマン処理におけるプロセスの優先順位処理とは、プロセスが実行可状態から実行中状態に遷移するとき，どのプロセスを先にするかと言う実行開始の順位であった．しかしながら、データフロー処理では複数のプロセスが実行中状態にあることが出来，また，実行中状態にある各プロセスは，複数個のプロセッサによって実行されるので，ノイマン処理の場合の実行開始順序による優先処理に加えて、プロセスに何個のプロセッサを割り当てるかと言う優先順位の決め方がある。すなわち、前者は、実行開始時間に関する優先順位処理であり、後者は、実行プロセッサの数，すなわち，実行速度による優先順位処理である。実行速度とは，単位時間に実行するプリミティブな命令の数として定義される．この数による優先順位を可能にするためには，プロセッサに、どのプロセスのアクタを優先的に実行するかを決めるの情報を与えなければならない．実行開始優先順位は，プロセス実行開始順位の表を作り，それに従って，プロセスを実行開始するのに必要な初期値データをマッチングメモリ，もしくは，トークンメモリに格納することによって行われる．実行速度優先順位は，実行開始時にプロセスにプロセッサを割り付けることによって行うことが出来る．プロセッサにどのプロセスを実行すべきかの情報を与えるには，トークンカラーと実行すべきプロセスとの関係を示す情報を，関連するプロセッサに知らせることによって可能である．

実行速度優先順位におけるプロセッサ割当において，１つのプロセスに多くのプロセッサを割り当て，実行速度をあげようとしても必ずしも期待どうりの結果が得られるとは限らない．ここで言うところの速度とは，アクタを並列に行うことによる単位時間当りの実行量であり，もし，プログラムに無限の並列度があればそれは可能であるが，プログラムの並列度が限られている場合には，そのプロセスに幾ら多くのプロセッサを割り付け，実行速度を増やそうとしても実行速度は増えない．もしこの時，一つのプロセッサを一つのプロセス専用に割り当てることにすると，並列度が小さいプロセスに多くのプロセッサを割り当てた場合には，多くのプロセッサが遊んでしまうことになる．このようなことを避けるためには，一つのプロセッサに多くのプロセスを割り当てることが必要になる．一つのプロセッサに複数個のプロセスを割り当てると，プロセッサに，その内のどれを先に実行すべきかを指示しなくてはならない．すなわち，プロセスにプロセッサを割り付けるとき，プロセッサ内でどの程度の実行優先順位をもって割り付けるかを決定しなければならない．以上の事より，実行速度優先順位は，プロセスに割り当てるプロセッサの数と，そのプロセスに割り当てられたプロセッサ内での実行に対する強さによって決定される．この強さのことを，ここでは，実行強度と呼ぶ．

⑤永久遅延

データフローコンピュータ上で単独のプログラムを実行するときは次のような問題を生じないが、データフローコンピュータをオペレーティングシステムを通して見たとき、これは，多くのユーザーから使われる共有資源としての性格を持ってくる。この場合永久遅延が発生する．

いま、２入力アクタにおいて，１Ｌ，２Ｌ，３Ｌのトークンが左側の入力アークに，１Ｒ，２Ｒ，３Ｒのトークンが右側入力アークに到着していると仮定し，アクタは，たとえば，（１Ｌ，２Ｌ）の様なトークンの組合せに対して実行されるとする（この様な，アクタ実行に必要なトークン対のことを，必要トークンとよぶ）．

すると、一時には、（１Ｌ、１Ｒ）または（２Ｌ、２Ｒ）、（３Ｌ、３Ｒ）のうちの１組の必要トークンに対してアクタの実行が行われ、３組の必要トークンに対して実行が終った時、そのアクタに対する総ての実行を終了する．いま、次から次へと必要トークンが到着しており、アクタの入力アーク上に常に複数個の必要トークンがあると仮定する．データフロー処理では，これらのトークンの内，通常任意の必要トークン（したがって、ランダムに取りだされた必要トークン）に関するアクタに対して実行が行われるので、たとえば、（２Ｌ、２Ｒ）に対するアクタ実行のまえに、後から到着した必要トークンに対するアクタの実行が次から次へと行われ、（２Ｌ、２Ｒ）に対するアクタ実行が永久に待たされることがおこりうる。たとえ永久遅延に致らなくとも、かなり長期間の遅延が発生する可能性がある。

１４．むすび

以上われわれの並列処理コンピュータとＯＳに関する研究について述べてきた．この他に，関連研究として並列言語に関する研究がある．研究の結果生まれてきた言語は，単一代入規則を基本原理とした並列言語Ｐと，並列オブジェクト指向言語ＩＭＰＯＯＬである．Ｐには後に，シリアル処理の書き易さと現在の手続き型言語とのコンパチビリティをめざして，ノイマン処理の導入がはかられた．

参考文献
1）曽和，早川；高レベルデータフローコンピュータシステムーーGMMCSーー，情報処理学会計算機アーキテクチャ研究会，36-2,1979.
2）Sowa,Ohkubo:"Data-driven control of multi-microprocessor system,"Trans. of IECE Japan,Vol.E67,No.2,1984.
3）Sowa,M.,Murata,T.,"A dataflow computer architecture with program and token memories,"Proceedings of the 1980 Asilomar Conference on Circuits,Systems, and Computers,IEEE ,Nov.1980. or IEEE Trans. on Computers,Vol.C-31,No.9, 1982.
4）曽和，ラモス：データフローコンピュータDFNDRにおけるプロシジャコールのインプリメンテーション，電子通信学会電子計算機研究会，EC83-16，1983.
5）曽和，"データフローマシンと言語"昭晃堂，1986年.
6）曽和；Controll flow parallel computer，情報処理学会計算機アーキテクチャ研究会，48-2，1983.
7）Ramos,F.D.,M.Sowa,"Design of assembly level language for controll flow parllel computer,"情報処理学会ソフトウエア基礎論，7-5，1984.
8）Sowa,M.,Ramos, F.D., "Performance of a simulated control flow parallel computer,IEEE 6th Computer Science Conference,1986.
9）曽和，菊池；コントロールフローパラレルコンピュータ，電子通信学会電子計算機研究会，EC84-15，1984.
10）曽和，青木；コントロールフローコンピュータの実現結果について，昭和60年電子通信学会総合全国大会，1714，1985.
11）穂坂；コントロールフロー処理におけるプロシジャコールの実現法，群馬大学昭和60年度卒業論文，SLL850064，1985.
12）市川；データフロー，コントロールフロープログラム変換，昭和60年度群大卒業論文，SLL-850063，1985.
13）野口；データフロー，コントロールフロープログラム変換と変数域の最適化，昭和61年度群大卒業論文，SLL860101，1986.
14）Sowa;Speed-up in dataflow computer, IEEE EUROCON84,1984.
15）Sowa;"A method for speeding up serial processing in dataflow computers by means of a program counter,IEE The computer Journal,Vol.30,No.4,1987.
16）曽和；ユニバーサルフローコンピュータアーキテクチャ，情報処理学会アーキテクチャワークショプ イン ジャパン，1984-11.
17）曽和，青木；ユニバーサルコンピュータの実現，東北大学通研シンポジューム，I-5，1985-7.
18）Sowa,M.:"Dataflow computer DFNDR-2 and its extension,"Journal of Informention processing,Vol.10,No.4,1987.
19）田崎，"PNコンピュータの概要と原理"群馬大学SLﾚﾎﾟｰﾄ，SLL86031,1986-10.
20）田島，"PNコンピュータのためのCompilarの設計（基本編）"群馬大学SLﾚﾎﾟｰﾄ，SLL860034,1986-10.
21）曽和；PN（parallel Neumann）コンピュータの提案，情報処理学会コンピュータアーキテクチャシンポジューム，pp109-114，1988-5.
22）曽和：コンスタンスパラレリズムコントロールフロー計算機の提案，電子情報通信学会コンピュータシステム研究会，CPS89-6，1989.
23）林，曽和；データフロー用並列オペレーティングシステムSSSの資源管理に関する考察，情報処理学会，オペテーティングシステム研究会，33-8，1986-12.
24）林；データフローコンピュータ用並列オペレーティングシステムSSSに関する基礎研究，群馬大学大学院修士論文，SLL-86009 0，1986.
25）曽和；データフローコンピュータにおけるデッドロックの発生ーー永久遅延問題ーーー，情報処理学会計算機アーキテクチャ64-2，1987.
26）Zhong, Y.,Sowa,M.:"Towads an implicitly parallel object-oriented language,IEEE COMPSAC'87,1987.

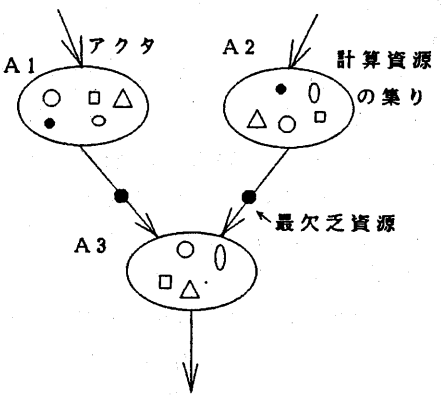
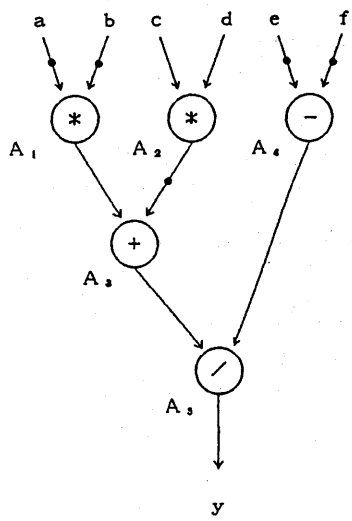
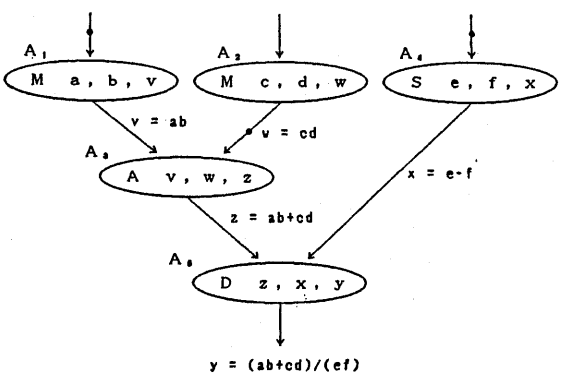


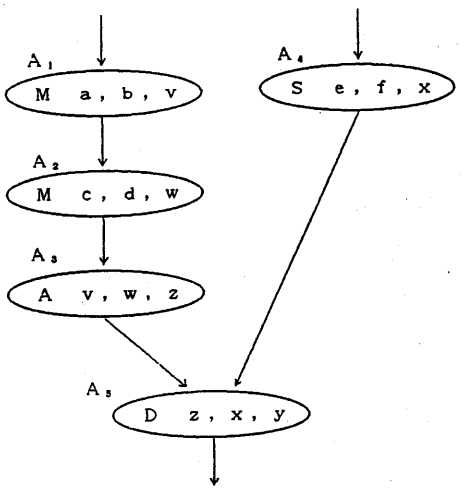
図1．資源駆動原理



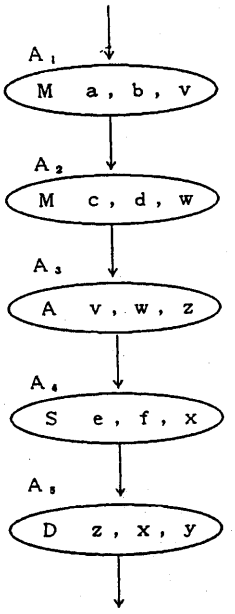
（a）DATAFLOW PROGRAM



（b）CONTROL FLOW PROGRAM

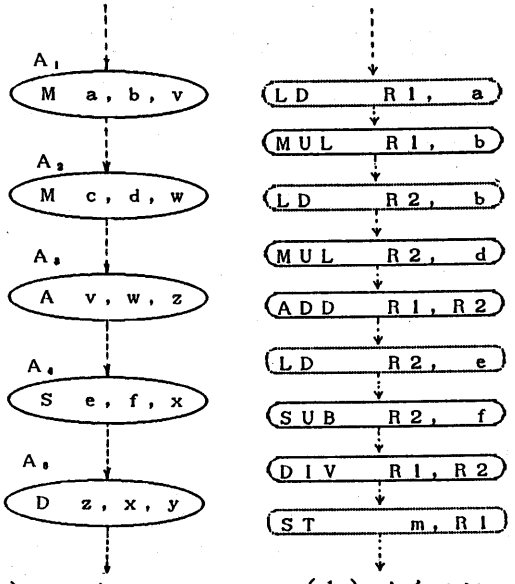


（c）CONTROL FLOW PROGRAM



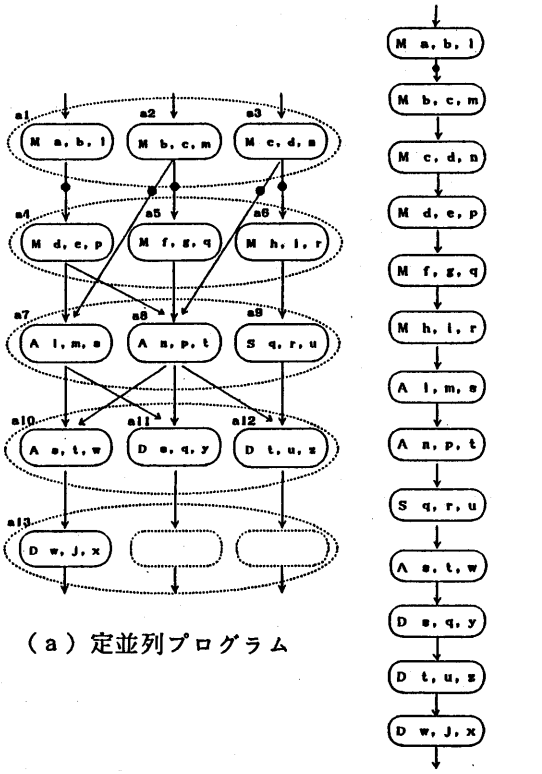
（d）SERIAL CONTROL PROGRAM

図2．データフローとコントロールフロー



（a）コントロールフロー　（b）ノイマン

図3．ノイマンプログラム



（a）定並列プログラム

（b）ノイマンプログラム

図4．定並列とノイマンプログラム

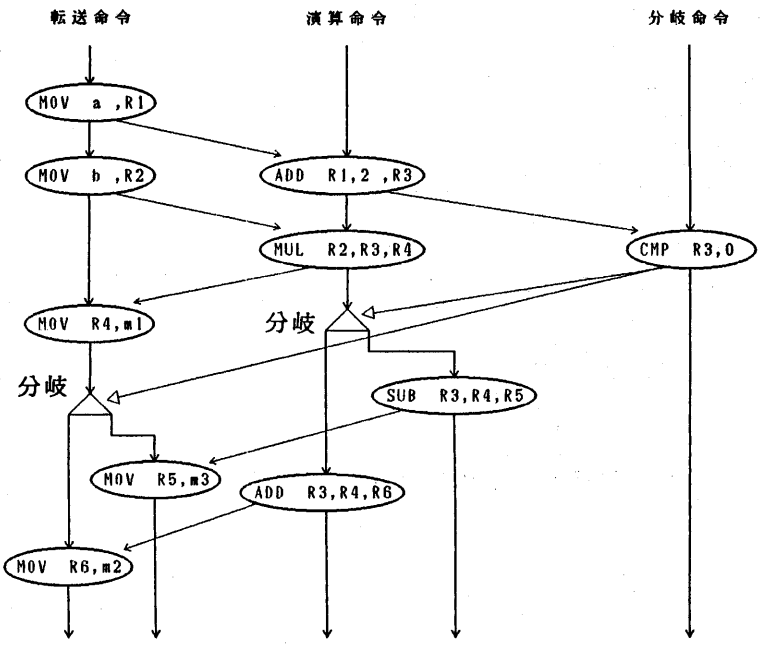


図5．ＰＮプログラム

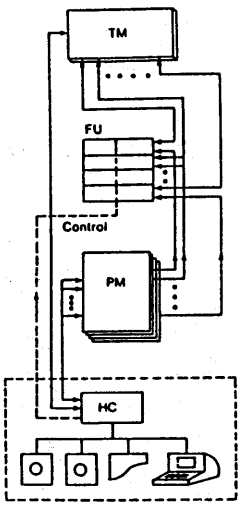


図7．マシン語レベルデータフローコンピュータ

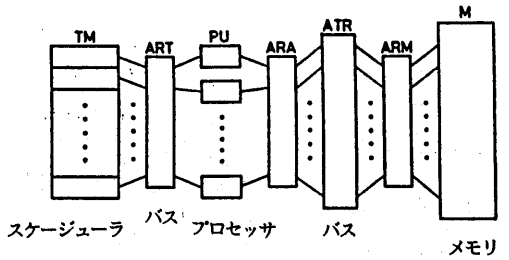


図8．コンピュータの一般構成

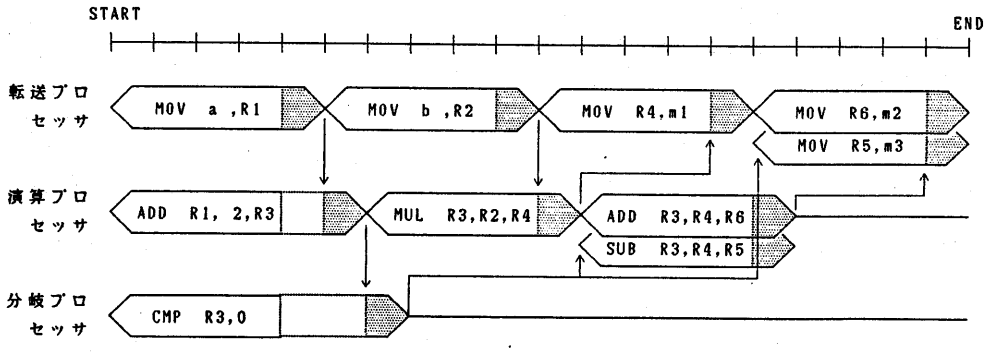


図6．ＰＮプログラムの実行