FIFO キューを同期手段とする並列プログラムについて (III)†

——実行管理機構——

有田 五次郎†† 末吉 敏則††

高多重並列処理の効率を低下させる要因に同期のオーバヘッドがある．われわれはこの問題を解決する一つの手段としてFIFOキューを同期手段とする待ちなし並列プログラム(self synchronizing parallel program, SPP) の概念を提案した．本論文では SPP の実行管理機構について考察し，SPP を実行する高多重並列計算機のプロセッサアーキテクチャを提案する．まず最初に準備として SPP の概念について概説し，ここで検討の対象とするシステムのモデルを示す．SPP はデータの依存関係に従って木構造グラフで表現された並列プログラムであり，メモリ共有型の多重プロセッサシステムの上で動作する．次に SPP を実行する計算機システムのオペレーティングシステムについて考察して，回線結合の複合計算機システム上での SPP の実行管理ソフトウェアの構成を示し，最後に SPP の実行管理機構をハードウェアでもつプロセッサの構造を示す．ここに提案するタスク管理ハードウェアは単一プロセッサにおけるマルチプログラミングシステムの構築にも有用であり，OS の機能のハードウェア化と考えることができる．

1. はじめに

FIFO キューを同期手段とする待ちなし並列プログラム(self synchronizing parallel program, 以後 SPP と書く) の概念と，一般のプログラムから SPP を得る手順とについてはすでに述べた[1),2)]．木構造グラフで表現される SPP は，各プロセッサが先着順 (first come first served, 以後 FCFS と書く) に各オペレーションを実行するとき自然に同期がとれる並列プログラムになっている．高多重並列処理においては，多重化の効率を上げるため高速な同期手段を必要とするが，割込み機構を用いてソフトウェアで実装されている従来の同期操作では速度の点で十分でない．SPP の同期機構である FCFS は FIFO メモリを用いてハードウェアで容易に実現でき，SPP は高多重並列処理における同期問題を解決する一つの手段となる．

FIFO メモリを用いた SPP の同期機構の概要についてはすでに述べた[1)]．これは，異なるプロセッサによって実行されるタスクの起動を意味する並列分岐命令と，現在実行中のタスクの完了をプロセッサに知らせるタスク切換え命令とを実現するハードウェアである．一つの SPP だけを実行するのであればこれで十分であるが，実際には SPP の実行を管理する管理プログラム (OS) が必要になり，システムは OS のための機能をもたなければならない[3)]．これは通常の計算機が利用者の目的プログラムの実行とは直接関係のない割込み機構やメモリ保護機構，OS のための各種特権命令をもっていることに対応する．

本論文では OS の機能まで含めた SPP の実行管理機構について考察し，SPP を実行する高多重並列計算機のプロセッサアーキテクチャを提案する．2章ではまず準備として SPP の概要と，ここで取り扱うシステムのモデルとについて説明する．次に3章では，SPP の実行管理ハードウェアに必要な機能を調べるため，回線結合の複合計算機上で SPP を実行する分散型 OS の概念を示し，最後に4章でこれらの機能を実現するハードウェアの構成を示す．

2. システムモデル

本章では準備として，SPP の概念とここで考察の対象としているシステムのモデルとについて概説する．

2.1 SPP とその実行管理機構

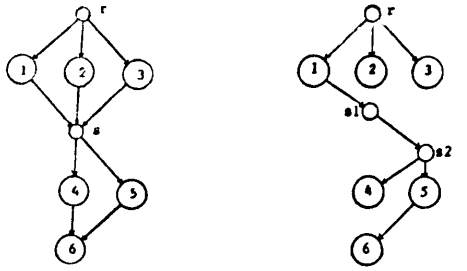
図1 (a)の逐次的プログラムはデータの依存関係を考慮して(b)のようなグラフで表現でき，適当な変換操作[2)]によって(b)から(c)を得ることができる．ここで点はオペレーションを，枝は制御の移動を表し，同一列のオペレーションは同一プロセッサで実行され

† On a Parallel Program with Synchronizing Mechanism Using FIFO Queue (III)—Execution Control Mechanism— by ITSUJIRO ARITA and TOSHINORI SUEYOSHI (Department of Computer Science and Communication Engineering, Faculty of Engineering, Kyushu University).

†† 九州大学工学部情報工学科

```
①  SX=SIN(X)
②  SY=SIN(Y)
③  CXY=COS(X+Y)
④  Z1=CXY*(SX+SY)
⑤  Z2=SQRT(SX*SX+SY*SY)
⑥  Z=Z1/Z2
```

(a) Sequential program



(b) Parallel program (c) Self synchronizing parallel program

図 1 プログラム例
Fig. 1 Example program.

ることを意味する．したがって，異なる列の点に向かう枝は異なるプロセッサで遂行されるオペレーションの起動，すなわち並列分岐を表す．異なるプロセッサ上のオペレーションは並列実行が可能であるから(b)，(c)は並列プログラムを表現している．

図1(c)のように表現されたプログラムの各列が異なるプロセッサのメモリに分散格納され，各プロセッサが FCFS でそのプロセッサ上のオペレーションを実行しているとすると，(c)の実行結果は(a)の実行結果と等しくなる．このような木構造グラフで表現される並列プログラムを SPP と呼ぶ．SPP は同期操作をもっておらず，同期は FCFS という逐次操作に還元されている．

SPP において，一度制御が渡ると中断されずに実行される同一プロセッサ上のオペレーションの集合をタスクと呼ぶ．タスクは SPP の実行の単位であり，FCFS による SPP の実行管理機構は以下のように構成される．

各プロセッサはそのプロセッサ上のタスクの実行順序を制御するため，実行すべきタスクの入口番地を保持する FIFO キューとその上の二つの操作とをもつ．

(1) タスク切換え (exchange task, EXT)

一つのタスクの実行が完了するとプロセッサは自分のキューから次に実行すべきタスクを取り出し，その入口番地に制御を渡す．このときキューが空であればプロセッサはアイドル状態になり，次に実行すべきタスクがキューに登録されるのを待つ．

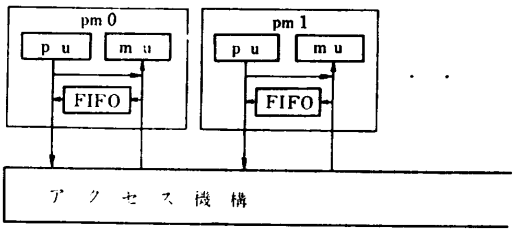
(2) 並列分岐 (parallel branch, PB)

他のプロセッサに割り当てられたタスクの起動は，そのタスクが割り当てられているプロセッサのキューにそのタスクの入口番地を登録することによって実現される．タスクを起動した側のプロセッサは引き続き実行を続けるので，この操作は並列分岐となる．

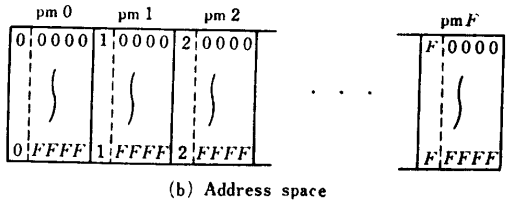
タスクは最後に実行される命令として EXT 命令を含み，タスク間の実際の制御の移動は，実行中のタスクが発行する EXT 命令によって起こる．しかし FIFO キューによる実行制御機構はタスクから見えないので，SPP の各タスクは他のタスクが発行した PB 命令によって起動されると考えてよい．

2.2 システムモデル

SPP を実行する並列計算機のシステムモデルを図2に示す．このシステムは同図(a)に示すように，プロセッサユニット (以後 pu と書く，以下同様) とメモリユニット (mu) とから成るプロセッサモジュール (pm) をアクセス機構によって結合した，モジュラ型のメモリ共有型並列計算機である．各モジュールはシステムアドレス (SAD) と呼ばれるモジュール番号で識別される．システム内のアドレス空間は図2(b)に示すように，SAD とモジュール内のロケーション (MAD) との2次元アドレスで表され，単一のアドレス空間を構成する．各 pu は全メモリを参照することができるが，命令フェッチはモジュール内の mu からしか行わない．すなわち，各 mu は各 pu のローカルメモリであると同時に，他の pu から参照可能なグローバルメモリでもある．



(a) System configuration



(b) Address space

図 2 システムモデル
Fig. 2 System model.

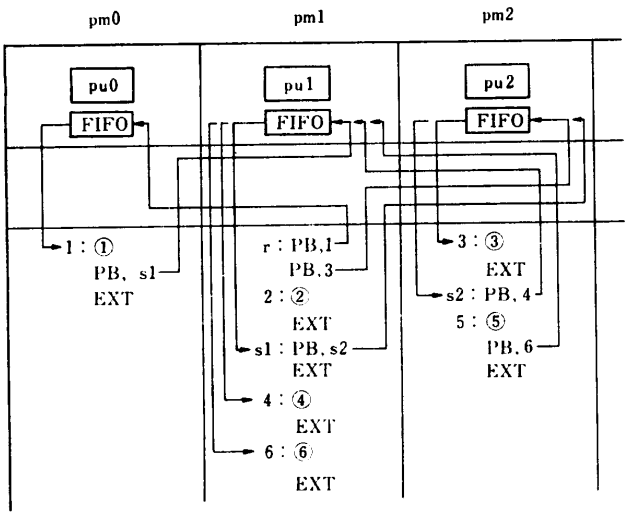


図 3 タスクの配置と制御の移動
Fig. 3 Task allocation and control flow.

SPP の各タスクは各 mu に分散配置され，おのおのの pu によって並列に実行される．このとき FIFO キューで実行順序が制御される．図1(c)のプログラムの各タスクの配置とタスク間の制御の移動の様子とを**図3**に示す．このプログラムの実行は pm1 の r: から始まる．なお各 pu はアドレス (SAD, MAD) によって全メモリを参照することができる．したがってデータはどの mu にあってもよいが，アクセス機構におけるデータアクセス競合を少なくするためには，そのデータをアクセスする頻度が最も高いプログラムが存在する pm に置くのがよい．

図2(a)においてアクセス機構がバス切換え回路である場合は，アドレス空間が SAD，すなわちプロセッサ台数分だけ拡張された通常のバス結合メモリ共有型多重プロセッサシステムとなる．この場合はデータアクセス速度が高速なので FIFO キューをハードウェアでもつ必要がある．並列実行されるタスクは，かなり実行時間の短いものでよい．

一方，アクセス機構が回線網のハードウェアと通信制御のソフトウェアとから構成されているときは，このシステムは通常の回線結合の複合計算機となる．この場合はデータアクセス速度が遅いので FIFO キューをソフトウェアで実現してもよいが，並列実行されるタスクは十分実行時間の長いものである必要がある．

いずれの場合についても並列処理の効率の上からはアクセス機構の物理的構造が問題になるが，ここではこの問題については言及しない．

3. 分散型 OS

本章では SPP の実行管理ハードウェアに要求される機能を調べるため，回線結合の複合計算機上にソフトウェアで実現される SPP の実行管理機構—複合計算機の分散型 OS の基本機能，について考察する．

3.1 複合計算機の分散型ソフトウェア

回線結合の複合計算機システムのソフトウェアを，その OS の構造によって次の二つの型に分類する．

第1は，各計算機が独立な，完結した OS をもち，OS の機能の一つとしてプロセス間通信の機能をもつもので，ここではこれを対称型ソフトウェアと呼ぶ．ネットワークアーキテクチャはこの代表的なもので，異なる計算機上にある並行プロセスはおのおの独立であり，その間のデータの授受および同期はメッセージ交換の形で行われる．

第2は，各計算機が独立した OS をもたず全体として一つの OS をもつもので，これを分散型ソフトウェアと呼ぶ．この型の複合計算機では全システムが単一のシステムとして扱われる．一つのプロセスは並列実行可能なタスクに分割され，各計算機に分散配置されて実行される．また，異なる計算機間でのデータの参照はタスク間のメッセージ交換ではなく，データ参照を必要とするタスクが相手計算機のメモリ領域を，アドレスを指定して直接読み書きすることによって実現される．

分散型ソフトウェアはメモリアクセスが回線を経由してソフトウェアで行われる点を除き，メモリ共有型多重プロセッサシステムのソフトウェアとまったく同一である．SPP はオペレーションおよびデータが全システムに分散配置された並列プログラムであり，各計算機上に配置されたデータを自由に参照できなければならない．したがって SPP を実行する複合計算機システムのソフトウェアは分散型ソフトウェアとなる．

分散型ソフトウェアにおける管理プログラムを分散型 OS と呼ぶ．分散型 OS の基本機能を以下に示す．これらは括弧内に示すような，メモリ共有型多重プロセッサシステムのハードウェアに要求される機能に対応する．

(1) データ共有（共有メモリ）
他計算機のメモリの読み書き．

(2) 制御の移動 (割込み)
他計算機上のプログラムの起動.
(3) 排他制御 (テスト・アンド・セット)
メモリ領域のロック/アンロック.

(1)は共有データの参照, (2)および(3)は同期のための機能であり, (2)は主としてオペレーション間の直接の同期に, (3)は共有データを介した同期に使用される. 回線結合の複合計算機システムではシステム内の任意の計算機同士が回線経由で通信することができる. この通信機能を使用して, 上記の三つの機能はソフトウェアまたはファームウェアで実装される.

3.2 ジョブの形態

一つの SPP の実行をプロセスと呼ぶ. プロセスにはプロセス番号 (PID) が与えられる. ジョブは一つのプロセス, あるいはいくつかのプロセスのつながりとして実行される. ジョブの実行形態はシステムの使用目的によって異なり, 次のように分類できる.

(1) シングルプロセス・マルチタスク (SPMT)
(2) マルチプロセス・マルチタスク (MPMT)
(3) マルチプロセス・シングルタスク (MPST)

これら三つの処理形態を**図4**に示す. SPMT は高速処理を必要とする専用システム向きの実行形態であるが, タスク分割の大きさによってはプロセッサの遊び時間が多くなる. またデータ転送速度が遅い場合はデータ転送によるオーバヘッドがシステム効率に直接影響する.

MPMT は複数のプロセスを同時に実行しようとするもので, 通常のマルチジョブと同じ形態である. もちろん各ジョブは並列プログラム作成の手間および並列プログラミングシステムのオーバヘッドに耐えられるだけ十分に実行時間の長いものでなければならない. この場合はマルチジョブの場合と同様に, 各プロセスが実行を完了するまでの時間は長くなるが, データ転送のオーバヘッドおよびプロセッサの遊び時間はほとんどなくなり, システムの使用効率はよくなる.

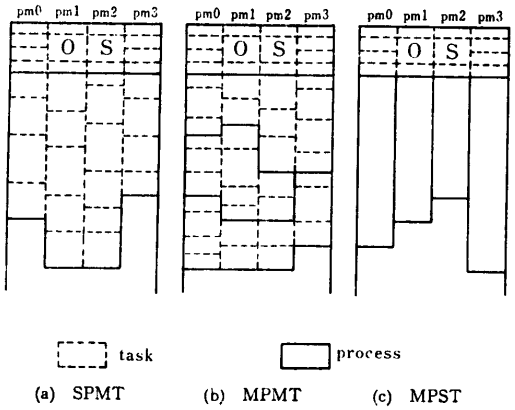


図4 分散型 OS におけるジョブ処理の形態
Fig. 4 The types of job processing forms under a distributed operating system.

MPST は各プロセッサを分割して使用する実行形態で, ジョブは OS の機能を使用する場合以外は単一プロセッサ上で動作する. 複合計算機によるバッチ処理の負荷分散や複合マイクロコンピュータによるマルチユーザシステム等はこれに相当する. この場合でも OS の機能を共有することができるので各計算機は完結した OS をもっている必要はなく, 全体として一つの OS をもてばよい.

3.3 基本マクロ命令

分散型 OS の基本マクロとその意味を**表1**に示す. (1)～(4)をアクセス制御マクロ, (5)～(10)をタスク管理マクロ, (11)～(13)をシステム管理マクロと呼ぶ. 利用者は(1)～(4)および(7), (8)のマクロを用いて並列プログラムを記述する. その他は OS 用のマクロであり以下のような目的に使用される.

ENABLE マクロはプロセスの実行を許可するために, また DISABLE マクロはエラー等の発生によりプロセスの実行が継続できない場合にそのプロセスに属する全タスクの実行を停止するために用いられる. SET PID マクロおよび SET KEY マクロはそれぞれ, プロセス発生時にプロセス番号および記憶保護キーを設定するために使用され, INTERRUPT マクロは実行中のタスクと無関係な OS の処理ルーチンを実行するために使用される. なお RESET マクロはハードウェア, ソフトウェアの障害によりシステム全体が閉塞状態になることを防ぐために用意されており, 任意のプロセッサから他のプロセッサの初期化を行うことができる.

3.4 分散型 OS の核

アクセス制御マクロ, タスク管理マクロ, システム管理マクロの処理ルーチンを分散型 OS の核 (カーネル) と呼ぶ. (8), (9), (11)以外の各マクロの実行は二つのプロセッサにまたがって行われる. したがってカーネルは二つの部分から成り, マクロを発行した側での処理をマスタ処理, マスタから処理を依頼される側の処理をスレーブ処理と呼ぶ. **図5**に示すように各プロセッサのカーネルは相手側のカーネルと通信しながらマクロの処理を行う.

カーネルのデータ構造とマクロとの関係を以下に示

表 1 分散型 OS のマクロ命令
Table 1 Macro instructions of distributed OS.

	マクロ	パラメータ	機能
1	READ	SAD, MAD, BUF, LENGTH	SAD, MAD で指定したアドレスから BUF で指定した自分のアドレスに LENGTH で指定した長さのデータを読み込む.
2	WRITE	SAD, MAD, BUF, LENGTH	BUF で指定した自分のアドレスから SAD, MAD で指定したアドレスに LENGTH で指定した長さのデータを書き込む.
3	LOCK	SAD, MAD, KEY	SAD, MAD で指定したアドレスの内容を読み出し KEY に与え，同時にその内容のビットを全部 1 にする.
4	UNLOCK	SAD, MAD	SAD, MAD で指定したアドレスの内容を 0 にする.
5	ENABLE	SAD, PID	SAD で指定したシステム上で PID で指定したプロセスに属するタスクの実行を可能にする.
6	DISABLE	SAD, PID	SAD で指定したシステム上で PID で指定したプロセスに属するタスクの実行を禁止し，もしそのタスクが実行中なら中止される.
7	PARALLEL BRANCH	SAD, MAD	SAD, MAD で指定したアドレスを入口とするタスクの起動を要求する.
8	EXCHANGE TASK		実行中のタスクの完了を示し，新しいタスクの実行開始を要求する．タスクがない場合には，起動待ちになる.
9	CHECK TASK		タスクがない場合，このマクロ命令の次のロケーションを実行する点を除き，(8)と同じ動作をする.
10	INTERRUPT	SAD, MAD	SAD, MAD で指定したアドレスを入口とするルーチンの起動を要求する．実行中のタスクは中断され，指定されたルーチンが実行される.
11	SET PID	PID	PID で指定した値にプロセス番号を設定する.
12	SET KEY	SAD, KAD, KEY	SAD, KAD で指定したメモリ保護キーレジスタに KEY の値を設定する.
13	RESET	SAD	SAD で指定されたシステムをリセットする.

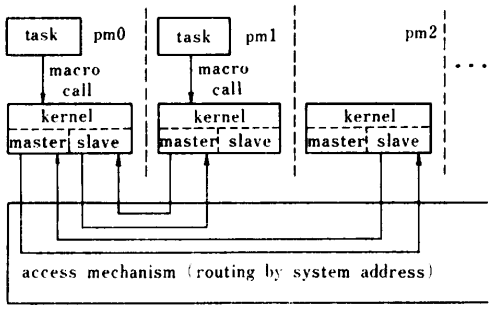


図 5 カーネル間の通信
Fig. 5 The communications between kernels.

す．なおこれらのデータはメモリ中の制御表であるが次章との関係でレジスタとして説明する．

(1) PIDR：プロセス id レジスタ

実行中のタスクが属すプロセスのプロセス番号を保持しているレジスタ．この値は SET PID マクロおよび EXCHANGE TASK マクロで設定され，PARALLEL BRANCH マクロおよび READ/WRITE, LOCK/UNLOCK マクロで相手システムに送出される．

(2) PKEY：プロテクションキーレジスタ

メモリの一定領域ごとのプロテクションキーを保持するレジスタ．キーはプロセス番号であり SET KEY マクロで設定される．アクセス制御マクロの実行時にマクロを発行したタスクのプロセス番号と比較され，一致しなければメモリ保護侵害とする．プロセス番号 0 のプロセスは OS のプロセスとし，これについてはチェックしない．

(3) EPIR：実行可能プロセスレジスタ

このプロセッサ上で実行が許可されているプロセスのプロセス番号を保持しているレジスタ．ENABLE マクロでセットされ，DISABLE マクロでリセットされる．PIDR がセットされたときおよび EPIR がリセットされたときは EPIR と PIDR が比較され，プロセス番号が EPIR にない場合はそのタスクの実行を放棄する．

(4) SSAR：システムアドレスレジスタ

このプロセッサのシステムアドレスを保持しているレジスタ．PARALLEL BRANCH マクロでマクロ命令の出所システムアドレスとして相手側に送出される．

(5) RSAR：戻り先システムアドレスレジスタ

EXCHANGE TASK マクロで後述する TQUE からセットされる．PARALLEL BRANCH マクロの出所システムアドレスであり，次の RMAR とともにパラメータの受渡しおよびサブルーチンリンクの構成に使用される．

(6) RMAR: 戻り先ロケーションレジスタ

RSAR と同様 PARALLEL BRANCH マクロの出所ロケーション.

(7) TQUE: タスク管理用 FIFO キュー

プロセス番号, PARALLEL BRANCH の行先ロケーション, 出所システムアドレス, 出所ロケーションを保持しているキュー. PARALLEL BRANCH マクロによってこれらのデータがキューに登録され, EXCHANGE TASK マクロによって取り出される.

(8) IQUE: 割込み管理用 FIFO キュー

INTERRUPT マクロの割込み先ロケーション, 出所システムアドレス, 出所ロケーションを保持するキュー. このキューに INTERRUPT マクロでデータが登録されると割込みとみなされタスクの実行は中断される.

システムの初期状態では各計算機上でリセットルーチンが起動され, 初期タスクが実行される. 初期タスクは OS の初期設定タスクまたは EXCHANGE TASK 1命令から成るタスクである. OSの初期設定タスクは基本マクロを使用して必要なタスクを各計算機に配分し, それらを起動することによってシステムを立ち上げる. なお, リセットルーチンはカーネルの初期設定を行うもので RESET マクロによっても実行される.

4. 実行管理ハードウェア

本章では, 前章の考察に基づいた SPP の実行管理ハードウェア, すなわち SPP を実行する並列計算機のプロセッサアーキテクチャについて述べる.

4.1 プロセッサモジュールの構造

図6にプロセッサモジュールの構成を示す. CPU はタスク管理命令をもった中央処理装置, TMU はタスク管理用レジスタ, FIFO キューなどを含むタスク管理ユニットで, これらについては後述する. バスは内部バス(Fバス)と外部バス(Sバス)とからなる. Fバスに接続されたファームウェアモジュール FMU はプロセッサモジュールにローカルな入出力装置やメモリで, 割込み処理ルーチンや入出力ルーチン等のシステムルーチン, 演算ルーチン等の共用ルーチンはこのメモリ上に置かれる. SMU は主記憶でありこの領域はSバス経由で他のプロセッサから参照可能である.

4.2 プロセッサ

CPU は通常の LSI マイクロプロセッサにタスク管理機能を増強したものである. したがってどの LSI

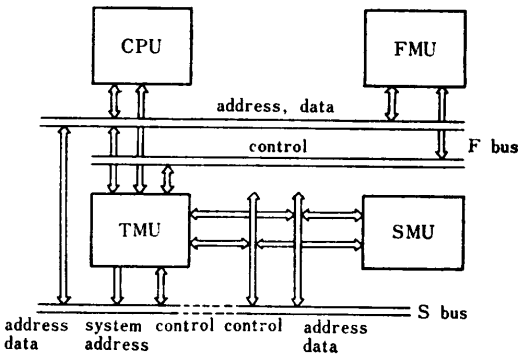


図 6 プロセッサモジュールの構成
Fig. 6 Configuration of a processor module.

プロセッサについて考えてもよいが, ここでは一例としてZ8000 マイクロプロセッサ[4]と類似した内部アーキテクチャをもつ16ビットマイクロプロセッサを考える.

4.2.1 アドレッシングモード

このプロセッサは F/N の二つのアドレッシングモードをもつ. Fモードはファームウェアモードで, 出力されるアドレスがそのままファームウェア領域をアクセスする. Nモードはノーマルモードで, システム内の全メモリ空間をアクセスする. このとき, 出力されたアドレスの上位3ビットはシステムアドレスレジスタの選択に使用され, アドレス空間の拡張とプログラムの再配置すなわちプロセッサ割当の変更とが可能になる. アドレッシングモードは, 命令フェッチの場合は後述するフラグアンドコントロールワード中に, オペランドアクセスの場合は命令中に指定され, ステータス情報としてバス上に出力される.

4.2.2 制御レジスタ・汎用レジスタ

プログラムステータスワード (PSW) はプログラムカウンタ (PC) とフラグアンドコントロールワード (FCW) とから成り, 前章に述べた SSAR および PIDR は PSW の一部として構成されている. **図7**

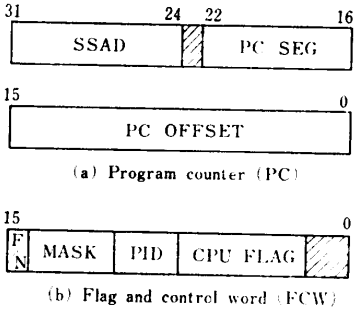


図 7 プログラムステータスワード
Fig. 7 Program status word (PSW).

に PSW の形式を示す．SSAD はこのプロセッサモジュールのシステムアドレスを表す．SSADはシステムリセット時にのみ設定され，以後は不変である．PC の 0～22 ビットはプログラムカウンタであり，23ビットのアドレス空間を指定する．ただし PC OFFSET 部から PC SEG 部への桁上りは起こらない．

FCW の F/N ビットは前述のアドレッシングモードで，PC のアドレスモードを決定する．PID は実行中のタスクのプロセス id を示すとともに CPU の実行モードを決める．すなわち PID が 0 であれば CPU はスーパバイザモードにあり特権命令の実行が許される．また MASK および CPU FLAG はそれぞれ割込みマスクおよび CPU が設定する状態フラグである．なお，タスク管理ユニットからの割込み実行要求に対するマスクも MASK に含まれている．MASK の他のビット，CPU FLAG およびニュープログラムステータスエリアポインタ等の形式は Z8000 と同一である．

汎用レジスタの構成と使用法も Z8000 と同一である．ただし，主としてスタックポインタとして使用される2ワード汎用レジスタ RR 14 は8個あり，PID の値によって一つが選択される．また RR 12 は，前章に述べた RMAR としてタスク管理命令で使用される．

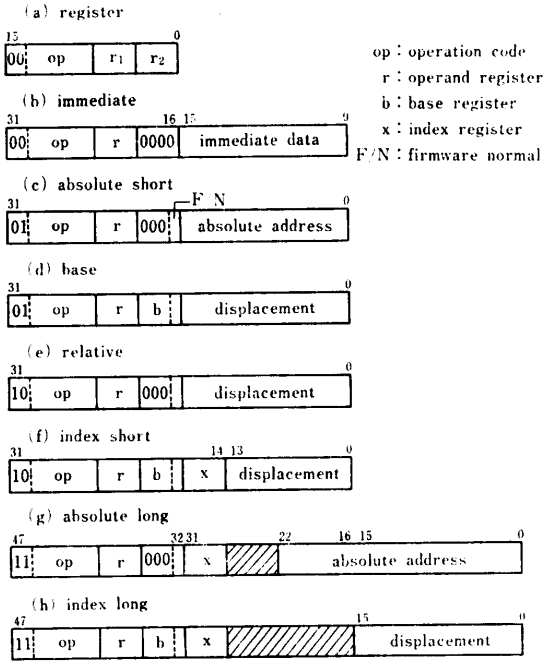


図 8 命令の一般形
Fig. 8 General instruction format.

4.2.3 命令の形式

命令の一般形式を図8に示す．操作部の上位2ビットとレジスタ部の値とによって命令長およびアドレス形式が決まる．ベースレジスタには RR 0 を除く2ワードレジスタ7個を使用し，ベース指定部の上位3ビットで指定を行う．ベース指定部の最下位ビットはアドレッシングモードの指定に使用する．

アドレスは7ビットのセグメントアドレスと16ビットのオフセットとから成り，アドレス修飾の結果はセグメント部に桁上りしない．またノーマルモードではセグメント部の上位3ビットがシステムアドレスレジスタの選択に使用される．

4.2.4 基本バスオペレーション

基本バスオペレーションにはメモリアクセスと IO アクセスとがある．割込み応答およびタスク管理レジスタアクセスには IO アクセスシーケンスが使用され，これらはステータス情報によって区別される．また，バスの排他的使用を行うため $\overline{\text{BUSHLD}}$ 信号があり，バスを排他的に使用する場合にこの信号が出力される．

4.3 タスク管理ユニット

図9はタスク管理ユニットの構成である．この機能および動作をレジスタを中心として以下に示す．

4.3.1 アドレス拡張およびSバス制御

SADR はアドレス拡張のための8ビット64語のレジスタファイルであり，プロセス id ごとに $SADR_{0\text{-}7}$ の8個のレジスタ群から構成されている．アドレッシングモードが N モードであれば，PID とアドレスの上位3ビットとによって SADR が選択され，システムアドレスとしてSバスに出力される．なお各プロセスごとに，$SADR_0$ は自分のシステムアドレスを保持しており，また $SADR_7$ は前章に述べた RSAR として使用される．

Sバス制御は CPU のアクセスシーケンスに従い，Sバスの獲得，データ転送，Sバスの解放を行う．

4.3.2 タスクの実行許可および禁止

EPIR は図10(a)に示すような8ビットのレジスタで，各ビット位置はプロセス番号に対応している．各ビットは対応するプロセスの実行の許可・禁止を意味する．EPIR の値は CPU が出力する PID と比較され，対応ビットが1でない場合は状態レジスタにフラグが立ち，タスク管理例外トラップの要因となる．

EPIR への値の設定はSバス経由で行われ，これを行う命令は特権命令である．

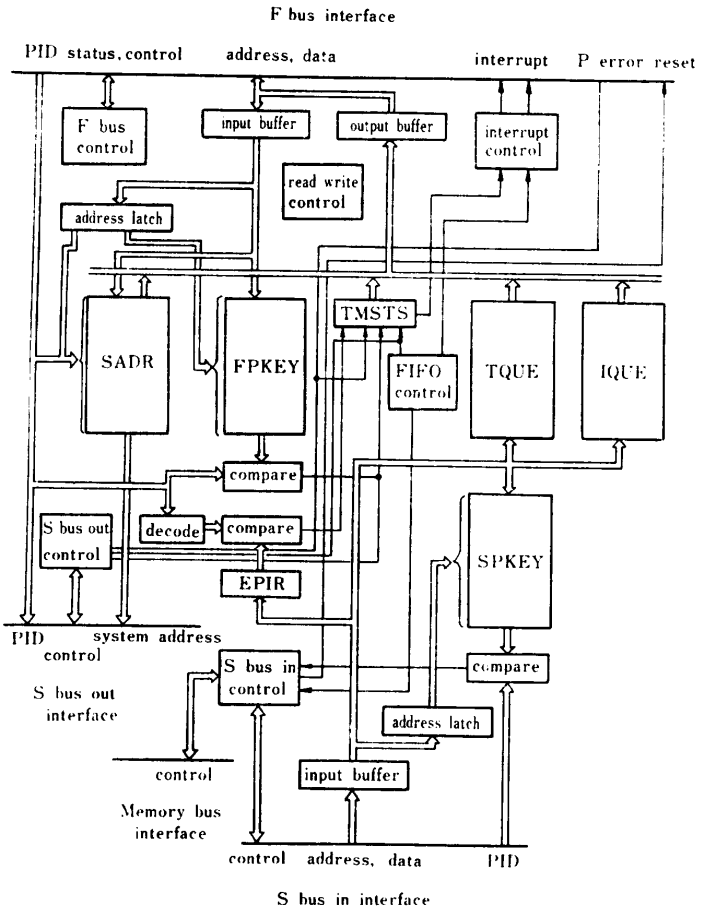


図 9 TMU の構成
Fig. 9 Configuration of TMU.

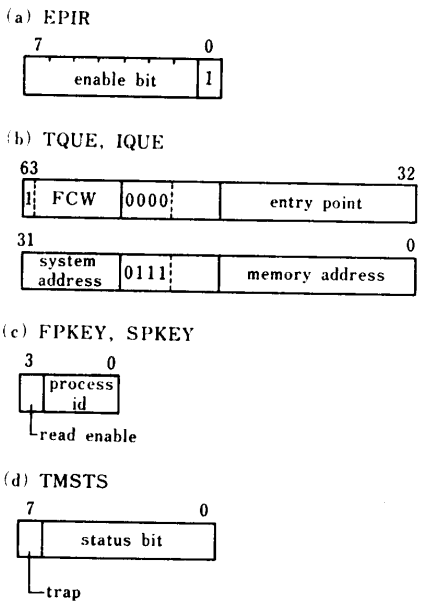


図 10 レジスタの形式
Fig. 10 Register format.

4.3.3 タスク実行および割込み実行制御

TQUE, IQUE はそれぞれタスク実行制御，割込み実行制御のための FIFO のキューで，各エントリはいずれも図10(b)の形式をもつ．これらは並列分岐命令および割込み実行命令によってSバス経由で書き込まれ，タスク切換え命令および割込み実行応答シーケンスによって読み出される．TQUE と IQUE の違いは，IQUE がデータを保持しているときには CPU に対して割込み実行要求の割込み信号を出すこと，および IQUE の FCW 部を CPU が使用しないことの2点である．

なお，キューが空のときの読出しおよびキューがフルのときの書込みに対しては，アクセスした側の状態レジスタに状態を表示し，タスク管理例外トラップの要因とする．

4.3.4 記憶保護

FPKEY はファームウェアメモリの，SPKEY はシステムメモリの保護キーを保持する図10(c)に示すような4ビットのレジスタ群である．記憶保護は 16 kB ごとに行う．FバスおよびSバスのアドレスビットはそれぞれ 23 ビットおよび 20 ビットである．したがって，FPKEY は 512 語，SPKEY は 64 語から成る．キーの下位3ビットはこの領域にアクセスすることを許されているプロセスのプロセス番号を表す．メモリアクセス時にはアドレスの上位ビットによって対応するレジスタが選択され，読み出された内容が PID と比較されて，一致しなければメモリ保護侵害とする．なお，PID が0の場合はスーパバイザモードであって，メモリ保護侵害とはしない．またキーの最上位ビットは読出し許可ビットで，このビットが1の場合は PID が一致しなくても読出しアクセスのみを可とする．これにより共用ルーチンの使用，プロセス間のデータ相互参照が可能となる．

FPKEY への値の設定はFバス経由で，SPKEY への値の設定はSバス経由で行われ，これを実行する命令は特権命令である．

4.3.5 タスク管理例外

TMSTS は図10(d)に示すような，タスク管理ユニットの状態レジスタである．タスク管理例外には，

メモリパリティ，記憶保護侵害，不許可タスク実行，TQUE フル，TQUE エンプティ，IQUE フル，IQUE エンプティがあり，これらは TMSTS に表示されると同時にタスク管理例外トラップを発生させる．なお図9には表示していないが，TMU はリセットフリップフロップをもち，Sバス経由でこれがセットされると CPU に対しリセット信号を出す．

4.4 タスク管理命令

タスク管理のために用意されている命令を以下に示す．これらの命令の詳細については省略する．

(1) pararell branch 並列分岐
(2) interrupt execute 割込み実行
(3) exchange task タスク切換え
(4) set SADR SADR への値の設定
(5) load control register タスク管理レジスタへの値の設定
(6) store control register タスク管理レジスタの値の読出し

(5)および(6)は特権命令である．

5. むすび

以上，FIFO キューを同期手段とする待ちなし並列プログラムを実行する計算機のオペレーティングシステムについて考察し，待ちなし並列プログラムの実行管理機構をソフトウェアおよびハードウェアによって実現する方法を示した．本文中にも指摘したように，このようなシステムが効率よく動作するためにはアクセス機構の構造が問題になる．これについては並列処理システムにおけるアクセス競合問題として別に検討している．

なおここに提案しているタスク管理ハードウェアは，単一プロセッサシステムの上でマルチプログラミングシステムを構築する場合にも十分有効なものと考えられる．

謝辞 本論文をまとめるに当たり九州大学工学部牛島和夫教授および荒木啓二郎助手には多くのご助言をいただいた．ここに記して謝意を表す．

参考文献

1) 有田五次郎：FIFO キューを同期手段とする並列プログラムについて(I)—待ちなし並列プログラム—，情報処理学会論文誌，Vol. 24, No. 2, pp. 221-229 (1983).
2) 有田，荒木：FIFO キューを同期手段とする並列プログラムについて(II)—並列プログラムの待ちなし変換—，同上，pp. 230-237 (1983).
3) 有田：FIFO キューを同期手段とする待ちなし並列プログラムとその実行管理機構について，情報処理学会第23回プログラミングシンポジウム予稿集，pp. 21-32 (1982).
4) シャープ(株)：Z8000 データマニュアル(1980).

(昭和58年3月1日受付)
(昭和58年5月10日採録)