ポリプロセッサシステム PPS-1 の オペレーティング システム

AN OPERATING SYSTEM FOR THE POLY PROCESSOR SYSTEM PPS-1

山内 長承　田中 英彦　元岡 達
Nagatsugu YAMANOUCHI, Hidehiko TANAKA, Tohru MOTO-OKA
東京大学工学部
Tokyo University

1. はじめに

分散処理システムを構成する場合の諸問題を検討する為に，ポリプロセッサシステムによる分散並列処理系のためのオペレーティングシステム PPS-1.OS を試作し検討を行なったので以下に報告する。

我々の本研究に対する興味は，単に複数のプロセッサを管理するオペレーティングシステムの検討を行なうだけではなく，書替え可能な制御記憶をもつマイクロプログラム制御プロセッサを用いて，異質な計算機の複合体 — ポリプロセッサシステムを作り，その上で機能分散による分散処理を行なうことであった。一方オペレーティングシステムのファームウェア化も興味ある点の1つであり，ファームウェア化した際の制御記憶の容量，及びそれによる実行速度の向上などについて興味があった。更に PPS-1 を研究室内計算機網 TECNET [2] に組入れることによって，PPS-1 上だけでなく TECNET 上での分散処理を可能にする計算機網向きオペレーティングシステムの性格を持たせること，その方法についても興味をもっていた。

これらの点を前提に PPS-1.OS を設計し，実装を行なった。以下，その構造，各部の実装手法について説明する。

2. PPS-1 のアーキテクチャ

PPS-1 は当研究室が富士通株式会社と共同開発したポリプロセッサシステムの試作機である。その構成は図1に示すもので，3台のプロセッサユニット（PU）が主記憶を共有するシステムである。

プロセッサユニット（PU）

プロセッサはマイクロプログラム制御方式であり，24ビットの垂直型のマイクロ命令をもつ。3台のプロセッサは各々 8 KW（24ビット＋3パリティビット／W）の書替え可能な制御記憶を有する。マイクロ命令の実行時間はスキップ発生等以外の場合 440 nS，スキップ時 880 nS テーブルブランチ 660 nS などとなっている。

主記憶

主記憶は 16K バイトのバンク4台から成り，合計 64K バイトの大きさをもっている。アクセスタイムは書込み 440 nS，読出し 660 nS であり，1バイト又は2バイトのアクセスが可能である。

入出力装置

低速入出力装置は IOA 及びスイッチマトリクス SWU を介して3台のプロセッサの中の1台と接続され，制御される。現在接続されている装置は，2台のタイプライタ（うち1台は APL 文字用），紙テープリーダ，ラインプリンタ，カードリーダ，キャラクタディスプレイ装置である。

また，各々 9.8 M バイトの容量をもつ磁気ディスクカートリッジは，BFC（Basic File Channel）を介して主記憶及びプロセッサに接続

されている。

通信制御装置は，TECNETに接続する伝送制御手順（ISO 7ビットコードによる拡張モード2重DLE法）をハードウェアで実現したHCCU，及び各種の伝送制御手順をマイクロプログラム制御によって実現できるMCCUの2台が接続されている。[4]

ダイレクトインタフェイスとコンソール

プロセッサ間での情報の授受及び相互診断を可能にする為に，各プロセッサ及びコンソールの間にダイレクトインタフェイスバスが張られている。バスは8ビットの情報を送ることができるほか，他プロセッサの起動，停止などを制御することができる。

尚，PPS-1の設計思想，アーキテクチャの詳細は文献[3]にまとめられている。

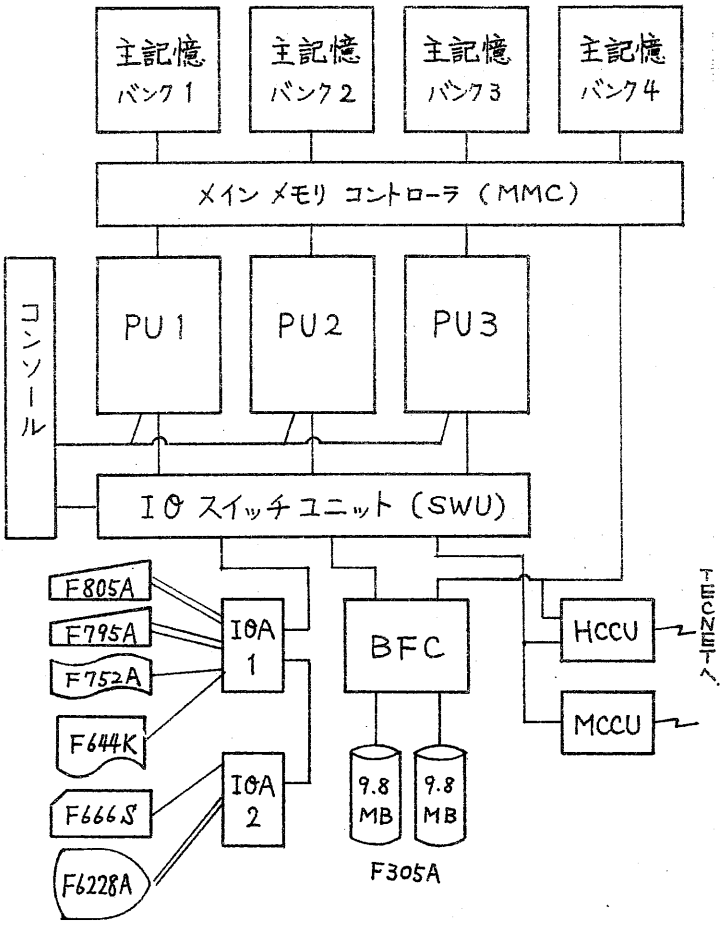


図1．　PPS-1 の アーキテクチャ

3. PPS-1.0Sの構成

全体の構成

全体の論理的な構成は，設計・製作・保守の容易さの点から，図2に示す階層構造とした。すなわちハードウェアから始めて順に機能を追加したバーチャル マシンの階層を設定してゆく手法により，PPS-1.0S全体を見通しのよいものにしている。

マルチプログラミング核

PPS-1.0Sの下ではプログラムはプロセスとして管理スケジュールされる。更にオペレーティングシステム内部の処理もそのほとんどをプロセスとして作り管理することによって，オペレーティングシステムの構造も簡明になる。

プロセスを動かす為に必要最低限のプログラム── ディスパッチャや割込処理プログラムなどは，非プロセスプログラムとして置き，マルチプログラミング核と呼ぶ。この核を環境として生成されたプロセスは，実行優先度に応じてスケジュールされ，プロセッサが割当てられて処理を実行する。

尚このレベルでプロセッサに対するハードウェア割込を吸収し，事象の発生という

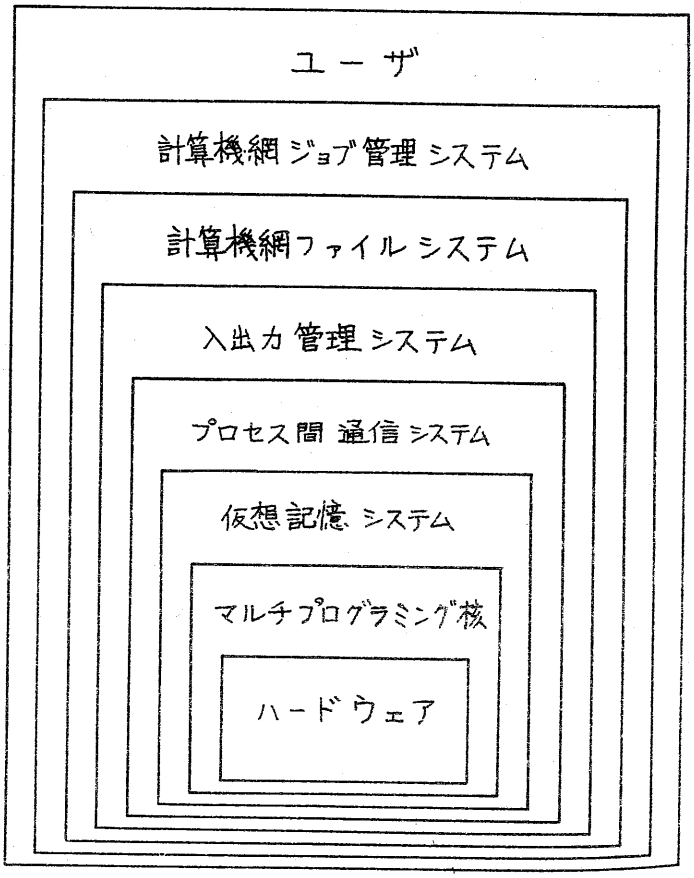


図2．　PPS-1.0S の 構成

形に変換してプロセスに通知する。

仮想記憶

この階層より上位に位置するプロセスが利用する仮想記憶を実現する階層である。プロセスはその属性として使用する仮想空間番号を持っており，これによってプロセスはプロセッサと自分自身のメモリ空間を割当てられたことになる。

プロセス間通信システム

コンカレントプログラムをプロセスとして管理し，それらの間のインタラクションをすべてプロセス間通信という形で管理するのがPPS-1.0S またTECNETのNOS の基本方針である。PPS-1.0S においてもプロセス間通信システムが大きな役割をになっている。すなわちPPS-1内のいかなるプロセスに対する通信（相手プロセスが他プロセッサで実行されるものでも），及びTECNET上のいかなるプロセスに対する通信は，すべてプロセス間通信システムを介して，全く一様な手順で行なうことができる。

入出力管理システム

入出力管理システムは，入出力装置によって異なる制御手順を吸収し，プロセス間通信を用いた簡単なインタフェイスを提供する。

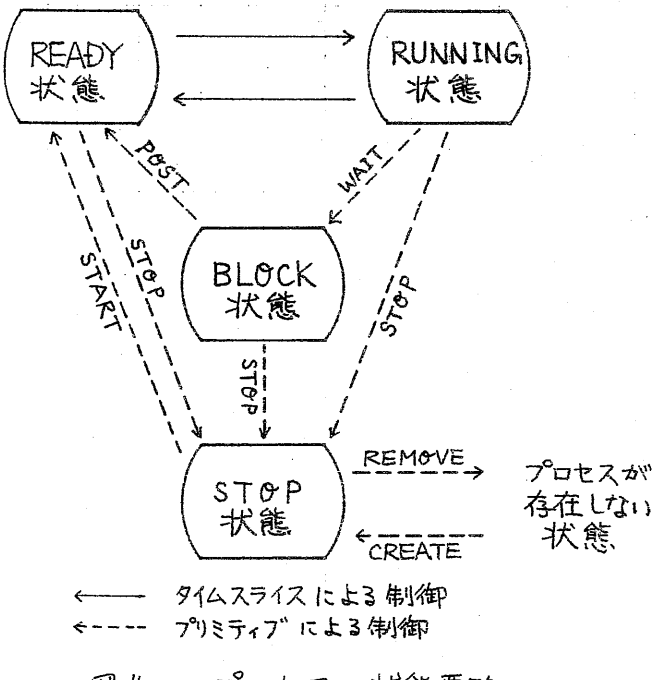


図4. プロセスの状態遷移

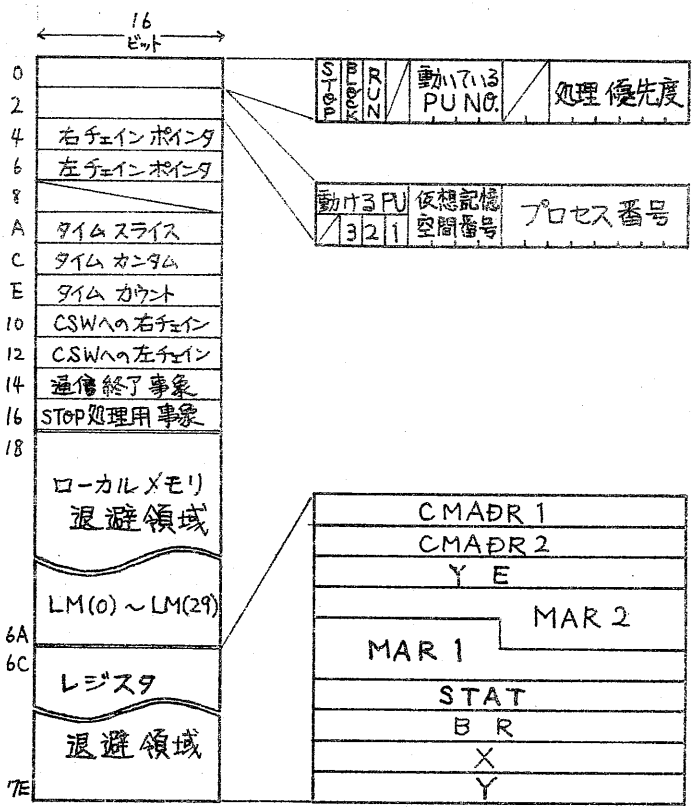


図3. PSWの構成

計算機網ファイルシステム

計算機網に適したファイルシステムを提供する階層である。PPS-1上のファイルだけでなく計算機網上に分散するファイルに対して一様なアクセス手続きを提供し，ユーザは計算機網を意識することなく網上のファイルを利用することができる。

計算機網ジョブ管理システム

計算機網上でのユーザジョブを管理するシステムである。PPS-1上でジョブを実行させる場合だけでなく，網上の他のホストにまたがるジョブを管理することができる。

4. プロセス管理とプロセッサ割当て

プロセス管理

プロセスの状態遷移を図4に示す。RUNNINGはプロセッサが割当てられて実行中の状態，READYはプロセッサ割当てを待っている状態，BLOCKは事象の発生を待っている状態である。

プロセスの状態はプロセスステータスワード（PSW）に表現される。（図3）PSWにはプロセスの状態，レジスタ退避領域などの他，プ

ロセスの属性として動きうるプロセッサ番号，処理優先度などが記述されている。

プロセスの生成・消滅などを制御するプリミティブは図5に示す4種であり各々マイクロプログラムから呼出す手順及び言語PLANからの呼出し手順（第8節参照）がある。

プロセッサの割当て

レディプロセスキューは3台のプロセッサに対応して存在し，キューへの挿入はプロセススケジューラが行ない，キューからの取出しは3台のプロセッサそれぞれの上で動くディスパッチャが行なって実行状態にする。（図6）スケジューリングアルゴリズムは現在次の様な簡単なものとなっている。

レディプロセスキューは各プロセッサ毎にA，Bの2種を設け，Aをリアルタイムプロセス及び非リアルタイムプロセスの中でタイムスライス半ばで中断したものの2種類を収容するキューとして用い，Bは非リアルタイムプロセスが新たにタイムスライスを与えられたもののキューとしている。ディスパッチャはA側キューが空になるまで優先的にAからディスパッチし続ける。

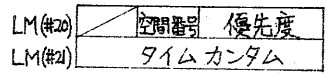
3台のプロセッサの割当てはPSWで指定されている動きうるプロセッサのうち，もっともレディプロセス数の少ないプロセッサに対して割当てる方法によっている。これによって，複数のプロセッサのいずれかで動きうるプロセスはその中から最適なプロセッサを選択して動くため，プロセッサの負荷が3台に分散する一方プロセスをプロセッサに固定的に割当てる（動きうるプロセッサを1つだけ指定する）ことによって機能分散システムを構成することができる。

割込み管理

PPS-1のマイクロプログラムに対するハードウェア割込みは，マシンチェック割込み，TLBフォールト割込み，入出力割込みの3種類があるが，TLBフォールト及び入出力割込は事象の発生として処理プロセスに通知される。

ハードウェアタイマの割込み，プロセッサ間

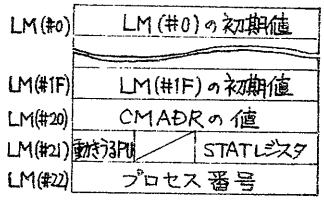
CREATE



BR(L) CREATE

RESULT = CREATE (SPNO, PRIORITY, TQ)

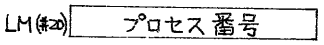
START



BR(L) START

RESULT = START (PRNO, PUNO, STAT, CMADR, LM0, LM1, ----)

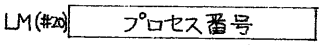
STOP



BR(L) STOP

RESULT = STOP (PRNO)

REMOVE



BR(L) REMOVE

RESULT = REMOVE (PRNO)

マイクロプログラムからのプリミティブの呼出し

PLAN シンタクスからのプリミティブの呼出し

図5. プロセス制御プリミティブ

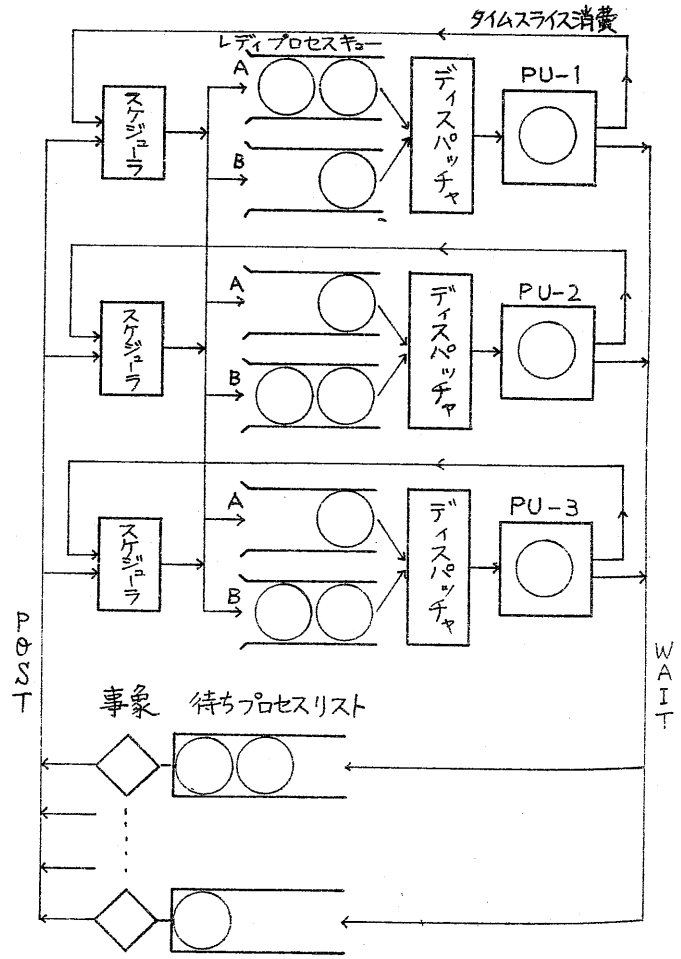


図6. プロセス キュー

の通信のためのダイレクトインタフェイスの割込みはマイクロプログラムに対して割込みを発生しないで，割込みレジスタにフラグビットがセットされるだけである。機械語を設定する場合には機械語レベルでの割込みを起こすことができるが，オペレーティングシステムをマイクロプログラム化した場合これらの割込みの発生は割込みレジスタを参照しない限り知ることはできない。よってプロセスはタイマの間隔より短かい間隔でこのレジスタをチェックし，割込みの有無を確かめる必要がある。

SEND/RECEIVE

LM(#20)	相手ホストNO	相手プロセスNO
LM(#21)	W	S/R ad-u
LM(#22)	メッセージバイト数	
LM(#23)	ad-ℓ	

BR(L) COM

RESULT=SEND(HOST,PRNO,WAIT, MESSBUFFER, LENGTH)

COMWAIT

LM(#21)	W

BR(L) COMWAT

RESULT=COMWAIT(WAIT, PARAMBUFFER)

LM(#20)	相手ホストNO	相手プロセスNO	復帰情報
LM(#21)	S/R		
LM(#22)	メッセージ バイト数		
LM(#23)	ad-ℓ		

COMCANCEL

LM(#20)	相手ホストNO	相手プロセスNO
LM(#21)	S/R	

BR(L) COMCAN

RESULT = COMCANCEL (HOST, PRNO, SR)

図7．プロセス間通信プリミティブ

プロセス間の同期

プロセス間の同期をとる手段としてユーザレベルではプロセス間通信を用いるが，その下位ルーチンとして POST/WAIT を作り，一部のシステムプロセスに使用を認めている。

事象は主記憶上に1ワード（16ビット）のECB（Event Control Word）で表わされ，3つの状態をとる。

- 事象は発生しておらず，待っているプロセスも無い。 ECB＝－1
- 事象は発生したが，待っているプロセスはまだ無い。 ECB＝－2
- 事象はまだ発生しておらず，待っているプロセスがある。 ECB＝待ちプロセスリストへのポインタ

5. プロセス間通信システム

プロセス間通信システムはプロセス間のインタラクションを統一的に扱かうシステムであり，PPS-1内はもとより，TECNET網上のどのプロセスに対する通信も同一の手順で行なうことができる環境を作り出している。

PPS-1 上のプロセス間の通信（ローカル通信）は主記憶上でデータを転送することによって行なう。通信するプロセスが異なるプロセッサに配置されている場合，プロセス間通信システムはデータ転送終了後送受プロセスの存在するプロセッサに，ダイレクトインタフェイスを通して割込んで通知する。すなわちプロセッサ間の通信はプロセス間通信システムの内部に吸収され，プロセスレベルのユーザからはプロセッサ間の通信は意識されない。

TECNET上の他ホスト上のプロセスに対する通信（リモート通信）では，TECNET網上の Host－Host プロトコールで定められた手順により，各メッセージは 250 バイト（メッセージの先頭パケットは 248 バイト）のパケットに分割され，回線上を送られる。[2] これらの機構は一切プロセス間通信システム内に吸収され，プロセスレベルのユーザは意識する必要が無い。

システムはプリミティブ処理ルーチンと，ローカル通信のデータ転送を行なうプロセス（LCP），リモート通信の為の処理プロセス（RCPM），メッセージからパケットへの分解・合成プロセス（RMPT）及び再送処理を行なうプロセス（RACKTC）から成る。

プロセス間通信のプリミティブ COM，COMWAIT，COMCANCEL のマイクロプログラムからの呼出し手順及び PLAN シンタクスからの呼出し手順を図7に示す。

6. 仮想記憶とページ管理，ページングプロセス

PPS-1 では仮想記憶のためのハードウェアとして，各プロセッサ毎に6ワードの TLB

（アドレス変換バッファ）を持ち，主記憶には1ページ（1Kバイト）毎に保護キー及び参照・書換え情報を保持するキーレジスタが備えられている。PPS-1.0Sではこの機構を用いて8Mバイトの仮想記憶を生成している。

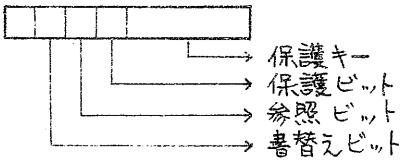


図8. キーレジスタ

ページ管理は次のような方式によっている。

- 1Kバイトを1ページとし，磁気ディスクとのスワッピングはページを単位として行なう。
- 64Kバイトを1論理セグメントとし，ユーザは論理セグメントを単位として仮想空間の獲得と解放を行なう。
- ユーザに割当てる論理空間は最大1Mバイトとする。ユーザは論理セグメントを幾つか獲得し自分の論理空間へはめ込むことによって実際に利用できる。
- 論理空間は15個存在する。すなわち15個のアドレス空間がある。

仮想記憶システムは，各プロセッサ毎に置かれたTLBフォールト割込み処理ルーチンと磁気ディスクへのスワップイン・アウトを管理するページングプロセス（PAGER）から成り，またページングアルゴリズムはキーレジスタ中の参照ビットによる近似LRU法である。

7. 入出力管理

入出力の管理は，すべての入出力装置に対して管理プロセスを置くことにより実現した。すなわちユーザプロセスからは入出力管理プロセス（IOCP）に対し制御用のコマンド及び入出力データをプロセス間通信システムを介して送受することによって利用することができる。これによってTEC-

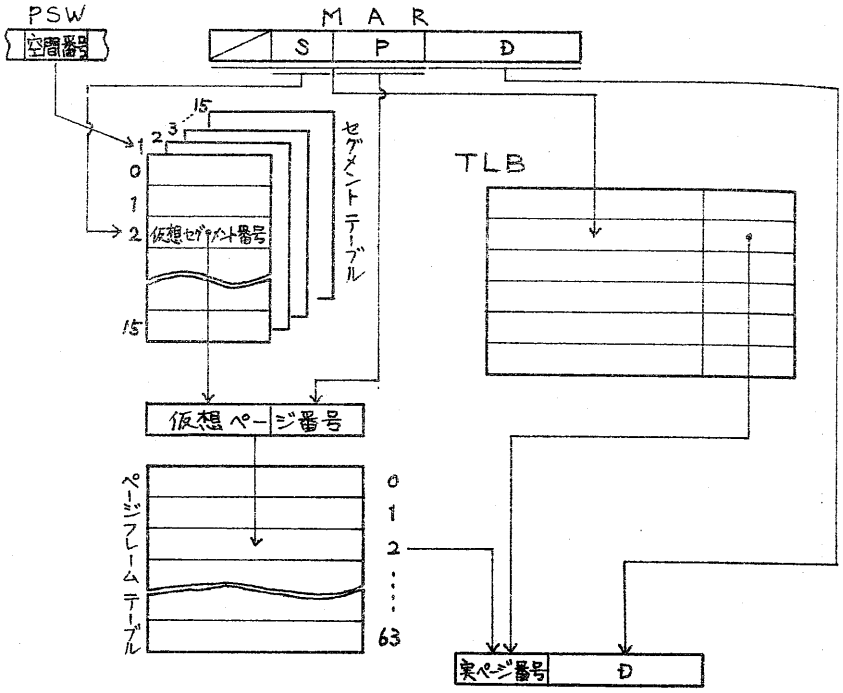


図9. アドレス変換機構

NET網内のいずれのホストに置かれたプロセスからもアクセス可能となっている。その利用手順を図 に示す。ユーザプロセスからIOCPに対して2バイトの制御コマンド及びWRITE時にはそのデータを付したメッセージを送る。入出力が完了すると2バイトの終了ステータス情報及びREAD時にはその入力データがメッセージとしてIOCPから送り返される。

入出力装置へのデータ転送は1バイト単位で行なわなければならないが，入出力管理プロセスに対して1バイト転送終了毎に事象の発生で通知する方法では事象待合わせ処理，事象発生処理，管理プロセス起動処理などのオーバヘッ

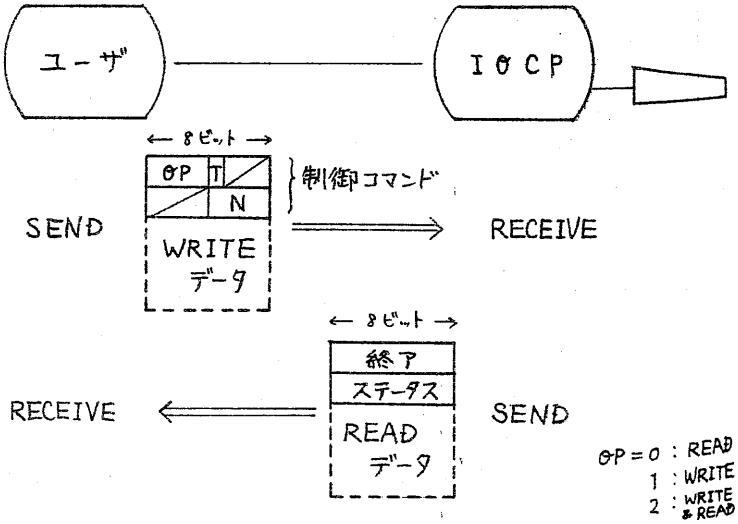


図10. IOCPとのやりとり

ドが大きくなる。そこで入出力データの1ブロック分の転送を入出力割込み処理ルーチンでまとめて管理し，1ブロックの転送終了後入出力管理プロセスを起動（POST）させる方法をとった。また同時に内部コード（ISO7ビットコード）とラインプリンタコード（EBCDIC），カードリーダコード（圧縮カードコード）の間のコード変換も割込処理ルーチンの中で行なっている。

尚，入出力管理プロセスは入出力装置による処理内容の差異があまり無いので，プロシージャはリエントラント構造とし制御記憶上に唯1つ置く。プロセスとしては入出力装置毎に作り各々が異なるレジスタ内容を持っている。

8. 計算機網ジョブ管理システムと計算機網ファイルシステム

計算機網 TECNET と共通インタフェイスを持つ計算機網ファイルシステム及び計算機網ジョブ管理システムは，前節までに述べたファームウェアシステムの上にインプリメントされた。計算機網ファイルシステムについては簡単なものを実装実験し検討を加えているが，詳細については更に検討を重ねた上で別の機会に発表したい。また計算機網ジョブ管理システムについては，その思想，構造，プロトコールに関して本研究会にて「網向きジョブ管理システム」（小田，堀口他）として報告しているので，ここでは実装，特に実装に用いた機械語 PLAN-VM の機構について説明する。

PLAN-VM は高級言語 PLAN [8] を実行するための仮想スタックマシンの機械語であり，PPS-1上ではマイクロプログラムによるインタープリタで解釈・実行される。計算機網ジョブ管理システムなどのシステムプログラムを記述するため，プロセス制御プリミティブ，プロセス間通信プリミディブを発行できるように，PLAN シンタクス及び PLAN-VM 機械語を拡張した。[9]

PLAN で書かれたシステムプログラムはコンパイラによって PLAN-VM 機械語に翻訳される。PLAN-VM 機械語はマイクロプログラムで書かれたインタープリタで実行されるが，プリミティブを利用するために拡張した機械語命令 K-Call に対してはインタープリタの中でプリミティブルーチンを呼出す。その実行の様子を図11に示す。

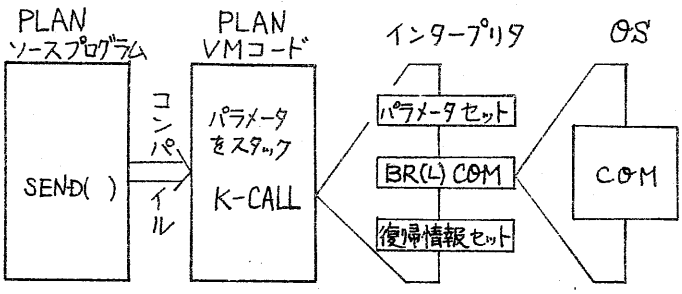


図11. 拡張部の実行の様子

PLANインタープリタは，マイクロプログラムとしては1つであるが，それに対して異なるレジスタ値及び仮想空間を与え異なるプロセスとして登録することによって多重に使うことができる。このとき仮想空間においた各々の機械語のプログラムを PLAN-VM 機械語で書いたプロセスと呼ぶことができる。計算機網ファイルシステム及び計算機網ジョブ管理システムの一部のプロセスはこの形式で作られたプロセスである。

計算機網ジョブ管理システムの起動はマイクロプログラムで書かれた小さなプロセス LOGGER プロセスが行なう。端末から $JOB コマンドが投入されると LOGGER は PLAN-VM で書かれた MJMP をロードしプロセスとして生成する。このようにして起動された計算機ジョブ管理システムは端末からのコマンドに従って，ユーザジョブを実行するジョブプロセスを生成，起動する。

9. 実装実験の結果

3台のプロセッサの制御記憶の配置は図12のようになっている。マルチプログラミング核及びプリミティブ処理ルーチン，TLBフォールト割込み処理ルーチンなどは3台のプロセッサに共通に置かれる。プロセスは，ページングプロセス，プロセス間通信補助プロセス（LCP, RCPM, RMPT, RACKTC）などの核プロセス，入出力管理プロセスなどのシステムプロセスの一部は，機能分散を考えて3台のプロセッサに分散して割当てている。すなわちページ

ングプロセス PAGER をプロセッサ2に置き磁気ディスクを接続し，リモート通信プロセス RCP（RCPM，RMPT，RACKTC）は通信制御装置を接続したプロセッサ1に置く。また入出力管理プロセスはプロセッサ3に置き IOA を接続する。計算機網ジョブ管理システムのプロセスは PLAN-VM インタープリタをプロセッサ1に置いているので，プロセッサ1の上で走ることになる。

残っている領域はユーザ空間として，ユーザ固有のマイクロプログラムをロードして使用することができる。ユーザマイクロプログラムの例としては OKITAC 4300C，FACOM R PDP 11/45 等のエミュレータがある他，PASCAL の VM コードを解釈実行するインタープリタがある。特に PASCAL では PDP 11/45 エミュレータ，PASCAL VM コードインタープリタを用いて Concurrent PASCAL，及び Sequential PASCAL の実装を行ない，Brinch Hansen の Solo Operating System を走らせている。

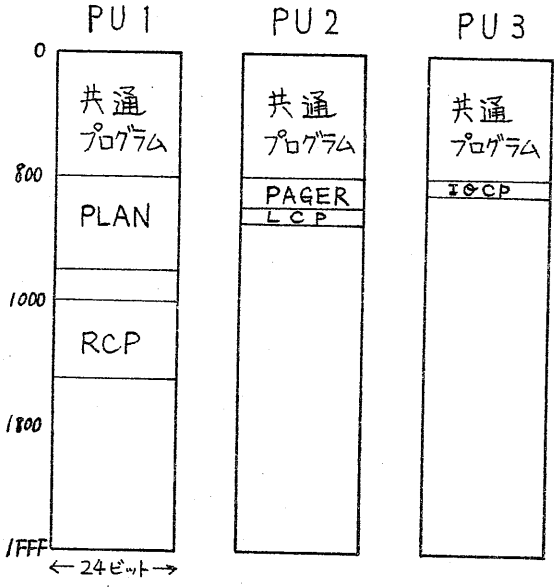


図 12－(1)　制御記憶マップ（全体）

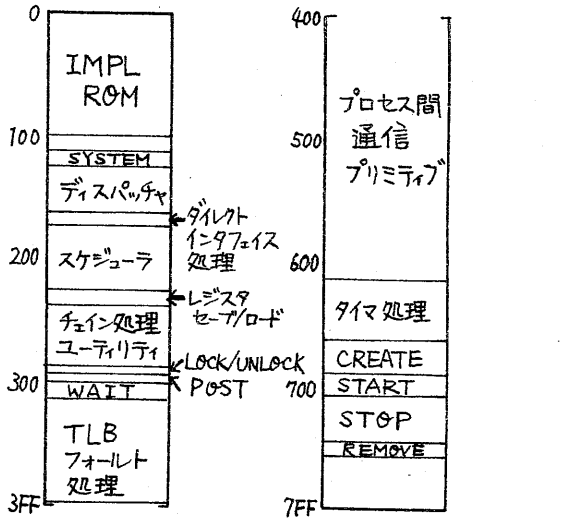


図 12－(2)　制御記憶マップ（共通プログラム）

主記憶は先頭から 8K バイトをシステム領域として確保し，仮想記憶システムの対象外としている。すなわちこの領域はスワッピングの対象にならないのみならず，アドレス変換機構の作用を受けず常にリアルアドレスで参照される。システム領域には核及び核プロセスの使用するテーブル類 — プロセステーブル，PSW，CSW，ページ管理用テーブル等，及び入出力装置のための入出力データバッファ，コード変換テーブル，リモート通信のためのパケットバッファなどが置かれている。(図13) また計算機網ファイルシステム，計算機網ジョブ管理システムの使用するテーブル等は仮想空間上に置いている。

このように先頭からシステム領域を連続して取った場合，主記憶の管理が容易であるが，3台のプロセッサから頻繁にアクセスされるシステム領域がすべて主記憶の第1バンク上に置かれることになり，主記憶アクセスの衝突が多発し，3台のプロセッサに機能分散配置した効果を弱めることが考えられる。この点に関

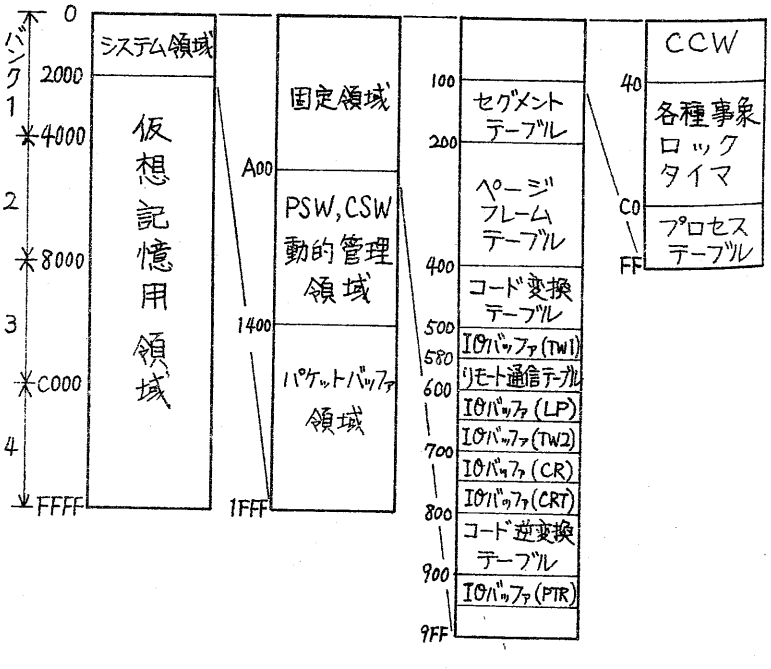


図 13.　主記憶マップ

する詳細な検討は今後に残されている。

各プログラムの実行時間

PPS-1.0Sの各プログラムの実行時間を，表1に示す。

SYSTEM（割込みレジスタチェック）（通常）		1.8 μS
（タイマ割込発生）		20.5 μS
プロセス切換え処理（スケジューラ，ディスパッチャを含む）		380 μS
POST/WAIT	POST	66 μS
	WAIT	180 μS
TLBフォルト割込処理（実際のスワッピング時間を除く）		460 μS
低速入出力装置の割込処理（1バイト転送）		25 μS

表1. 実行時間

これらの値は，簡単なミニコンピュータをPPS-1でエミュレートする場合（PLAN-VM）実行時間が一命令当り10～20μS（平均17μS）かかる事を念頭において評価すると，ファームウェア化した事によるオペレーティングシステム実行時間の短縮は十分効果が出ているものと考えられる。尚，プロセス切換え処理時間が他に比較して大きいのは，ユーザ用ローカルメモリ42語とレジスタ10個の退避復旧に要する時間のためである。

オペレーティングシステムのファームウェア化による効果

オペレーティングシステムの下位部分をファームウェア化することによって，その処理時間が短縮されるほか，次のような効果があることがわかった。

- プリミティブ操作を1つの機械語とすることができた。
- オペレーティングシステムのプログラムがハードウェア的に分離された2階層すなわち制御記憶と主記憶に置かれるため，相互の予期しない干渉，たとえば上位プログラムの暴走による下位システムの破壊などが起こり難い。

しかしながら，オペレーティングシステムをファームウェア化した場合，マイクロ命令で比較的大きなプログラムを組まねばならず，そのプログラミング上次のようなことが起った。

- 大きなマイクロプログラムを，マイクロアセンブラを用いて記述したため，プログラミングの効率が悪い。
  これに対しては，適当な高級言語からマイクロ命令に翻訳するコンパイラ方式が有効であるが，生成したマイクロプログラムの実行効率を十分保証し，更に制御記憶の大きさの制限からマイクロプログラム長を十分圧縮できなければならない，など高度な最適化が必要で，現在適当なものが無い。
- マイクロ命令がハードウェアに密接に関係しているため，命令シーケンスやタイミング等に配慮する必要があり，かなりわずらわしい。
- 実機上でのデバッグには大きな労力を必要とする。マイクロプログラムの開発には適当なシミュレータが必須であるが，オペレーティングシステムのような規模の大きいプログラムをシミュレートする場合，シミュレータの実行効率が問題となる。

オペレーティングシステムのファームウェア化を進めてゆく際には，これらの問題を克服する支援システムを準備することが重要である。

ポリプロセッサシステムを管理するオペレーティングシステムとしての評価

ポリプロセッサシステムを管理するオペレーティングシステムとしてここで実験したようなものが適当であるか否かは，今後更に実験，検討を重ねてゆくに従って明らかになると思われる。現在のところ機能分散による分散処理を，「プロセスに対して動きうるプロセッサを指定する」という方法で実現できること，幾つかのプロセッサ上で動き得るプロセスについては各プロセッサの負荷が平均するようにスケジュールすることで，当初の目標は実現されたと言える。

プロセス間のインタラクションは，プロセス間通信システムによって一元的に取扱うことができるようになり，プログラムの設計製作が容易になった。しかしながら特にローカル通信においては必ず主記憶内でのデータの転送が伴なうため，単純なセマフォを用いたインタラクシ

ョン（PPS-1.0SではPOST/WAIT）に比較すると実行効率は悪い。今後に改良の余地を残している。

またポリプロセッサシステムの特長として，一部プロセッサに障害が発生した場合システムを縮退し運転を続けることが可能である。更に正常なプロセッサを用いて障害の発生したプロセッサを診断することが可能で，PPS-1ではダイレクトインタフェイスに相互診断を支援する機能が用意されている。PPS-1.0Sでは現在のところ，障害発生時のプロセッサ切離し，オペレーティングシステムの自動再配置，プロセッサの相互診断などの処理は行なっていない。プロセッサに障害が発生した場合は制御記憶を入換え，プロセスに対する動きうるプロセッサ指定を変更したデータを用意して縮退運転を行なっている。その際の制御記憶の配置の一例を図14に示す。この場合ファームウェアの内容はほとんど変更せず，単にプロセッサ3に乗っていたIOCPをプロセッサ1に移しただけである。

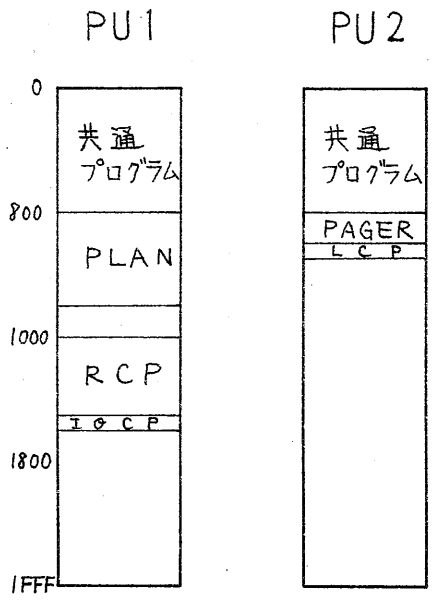


図14. PU3の障害時における縮退運転の制御記憶マップ

10. 結論

この研究の結果

- ポリプロセッサシステムのオペレーティングシステムの構成法としても核とプロセスによる階層構造が有効なこと，
- プロセス間のインタラクションをプロセス間通信システムという形に統一する手法が有効であること，
- ポリプロセッサ上のプロセスのスケジュールは，プロセスの動きうるプロセッサ番号をプロセスの属性として持たせること，プロセスキューをプロセッサ毎に持たせることによって能率良く行なえること，
- 更に上記のスケジューリング機構により各プロセッサに固定的に仕事を割付ける機能分散と各プロセッサに平均に仕事を割当てる負荷分散が，動きうるプロセッサの指定によって容易に実現できること，
- オペレーティングシステムの下位部分のファームウェア化が，実行速度の向上の他，ユーザプログラムとの不必要な干渉の減少という効果をもたらしたこと，
- オペレーティングシステムの下位部分のファームウェア化は機械語レベルでの制御プログラムを簡潔にした反面，大きなマイクロプログラムを作成することに伴うマイクロプログラムの生産性，保守性の悪さという問題が生じ，それらの問題を解決する強力な支援システム（シミュレータやマイクロプログラム用コンパイラなど）が必要であること，

などが分かった。また今後に残された課題としては

- このオペレーティングシステムを機能分散処理システムとした時の性能の評価，
- 複数のプロセスがインタラクトしながら並列に処理を進めるようなユーザジョブを記述するためのプログラミング言語の検討，

などがある。

11. おわりに

本オペレーティングシステムの開発には，他の多くの人々の努力によるところが大きい。特に仲川明和氏（現在富士通勤務）には基本設計とその実装を，埴淵豊氏（富士通）にはデバッグの一部とPASCALの実装を，芝田晃氏（東大工学部）にはリモートプロセス間通信システムの実装を行なっていただいた。ここに記して感謝する。

参考文献

[1] 元岡達 "コンピュータコンプレックスの展望" 情報処理, Vol.15, No.7 1974年7月

[2] 田中英彦, 元岡達 "研究用電子計算機網 TECNET" 電子通信学会電子計算機研究会資料 EC 73-57, 1973年

[3] 元岡達, 山室良夫 "ポリプロセッサシステム PPS-1" 情報処理, Vol.15, No.7 1974年7月

[4] 田中英彦, 海野忍, 山内長承 "PPS-1の通信制御装置" 情報処理学会第16回全国大会論文集 1975年 11月

[5] 元岡達, 田中英彦, 仲川明和 "PPS-1のオペレーティングシステム" 情報処理学会第16回全国大会論文集 1975年 11月

[6] 田中英彦, 堀田敬志 "計算機網ファイルシステム" 情報処理学会第16回全国大会論文集 1975年 11月

[7] 田中英彦, 山内長承 "ネットワークジョブ管理システム" 情報処理学会第17回全国大会論文集 1976年 11月

[8] 武市正人 "ミニ言語のミニコンパイラ" bit, Vol.6, No.8~13, 1974

[9] 山内長承, 田中英彦 "言語PLANの分散処理向きの拡張" 昭和52年度電子通信学会情報部門全国大会論文集 1977年8月

[10] 元岡達, 田中英彦, 仲川明和, 埴淵豊 "PPS-1.OSのファームウェア化とその評価" 情報処理学会第17回全国大会論文集 1976年 11月

[11] 埴淵豊, 田中英彦, 元岡達 "PPS-1ファームウェア技術によるPASCAL・SOLOシステムの移行" 昭和52年度電子通信学会情報部門全国大会論文集 1977年8月

[12] 和賀井フミ子, 田中英彦, 元岡達 "網向きオペレーティングシステムについての一考察" 電子通信学会電子計算機研究会資料 1977年11月

[13] 小田正美, 堀口真志, 黒羽法男, 山内長承, 田中英彦, 元岡達 "網向きジョブ管理システム" 電子通信学会電子計算機研究会資料 1977年11月