CA 21－1
(1976. 6. 30)

ミニコンピュータによる多重プロセッサ・システム

小林 薫 ・ 重永 実 （山梨大学 工学部）

1.はじめに

近年、処理能力の向上を目的として複数の計算機を結合させる試みが、各所で行なわれてきている。一般にこれらは複合計算機システム[1]と呼ばれ、種々の結合方式と使用されるプロセッサとによって多種多様の形態をとっている。このような複合計算機システムにおける利点はたくさんあるが、その中でも複数の計算機が比較的密に結合したシステムにおいて実現できる並列処理による処理能力の向上は、最も重要な要素であると考えられる。一方においては、マイクロ・コンピュータ等の普及によって、複数のプロセッサを用いて計算機システムを構成するという試みも盛んになってきている。このようなシステムの効率を高めるためには、ハードウェアの構成法のみならずソフト的な並列処理に対する技術も不可欠なものとなる。またOS（Operating System）が大規模化されまた複雑になる中で、これらの構造的アプローチや、この中で使用される各種のプリミティブの研究も盛んに行なわれてきている。特に複数のプロセッサを用いたシステムにおいては、同期の問題や資源管理の問題が重要となり、理論的に考えられた手法を実際に実現し、それを検討してみることも必要であると考えられる。

このような考え方に基づいて、この論文においては、複数のミニコンピュータを共有メモリなどによって結合させ、並列処理による応答時間の短縮を主目的としたミニコンピュータ複合体システムの骨格が完成したので報告する。ここではこれをDMS（Dual Mini-computer System）と呼ぶことにする。ソフト的には、この共有メモリ上に資源の集中管理を行なうモニタを配置し、多重プロセッサによる複数プログラムの並列処理の動作ができるOSの作成・検討を行なった。

ミニコンピュータ複合体システムとしては、C.mmp（カーネギメロン大学[2]）、TMCS-40（東芝[3]）、KOCOS（慶大[4]）、など多くの例が発表され、また既に商品化されているものもある。これらの結合方式としては、〔1〕共有メモリ結合、〔2〕チャネル結合、〔3〕バス結合、などに大きく分類されることができ、それぞれの方式はそれなりにそれぞれの利点をもっていると考えられる。

ここにおけるシステムは、並列処理を重要視しているので、数多くある結合方式の中で最も並列処理を実現しやすいと考えられる共有メモリによる結合方式を採用した。これは複数の同質のミニコンピュータと複数のメモリ・モジュールとがスイッチング装置によって結合されたものを想定するものである。そしてこの場合に対するハードウェアからソフトウェアを通じて生じる各種の問題について検討してみることにする。このような方式の利点は、次のような事柄である。

〔1〕共有メモリの存在によって並列処理を行なう時のプログラムのデータの授受が容易である。

〔2〕モニタなどを共有メモリ上に置き、リエントラント構造にすることによってメモリの節約になる。

〔3〕複数のプロセッサが密に結合されているので、信頼性の向上に対する対策が実現しやすい。

(4) システム資源の集中管理が比較的容易になるので効率の向上が望める。

一方において以下のような問題点も存在する。

(1) スイッチング装置の遅延と、メモリ・モジュールに対するアクセスの競合による遅延とから損失が生じる。

(2) 同期命令であるTS命令が、一般的なミニコンピュータにおいてはハード的に実現するのが困難である。

(3) このような方式のシステムにおける並列処理を制御するモニタの構造や方式が複雑になる。

(4) プロセッサの数が多い場合に生じやすいプロセッサのアイドル状態を少なくする必要がある。

これらの問題点は、プロセッサやメモリ・モジュールの数の増加に伴なって難かしくなると考えられる。したがってここにおけるシステムとしては、これらの数が最小である小規模システムを考えて、並列処理の実験システムを実現した。またこのDMSでは、プロセッサ間の距離が比較的離れているので、それによる時間的な影響も検討してみることにした。

2. ハードウェア構成

現在のDMSは、図1に示すように富士通製のミニコンピュータFACOM U-200を2台使用し、それに16KBのメモリ装置、バス・スイッチング装置(Bus Switching Unit、BSUと略す)、高速チャネル結合装置(U-200 Communication Unit、UCUと略す)を付加することによって形成されている。BSUは、そのメモリ装置が両方の共通バスからアクセスできるようにするための制御装置である。しかし2台のミニコンピュータ間の距離が約33mあるために2号機のミニコンピュータの共通バスとは、80芯(実際に使用されているのは37本で残りはアース線)のケーブルを用いて接続されている。そしてそのBSUから接がれているメモリ装置は、サイクルタイム650nsのコアーメモリ装置である。またUCUは、プロセッサ間の通信のための割込み機能を持っており、データ転送速度は、約300KB/Sである。これもBSUと同様に65芯のケーブルを用いて2つのUCUが接ながれている。U-200は、共通バス方式を用いた制御用のミニコンピュータである。したがってそのプロセッサは、優先処理などを行なうのに十分な割込み機能が完備している。

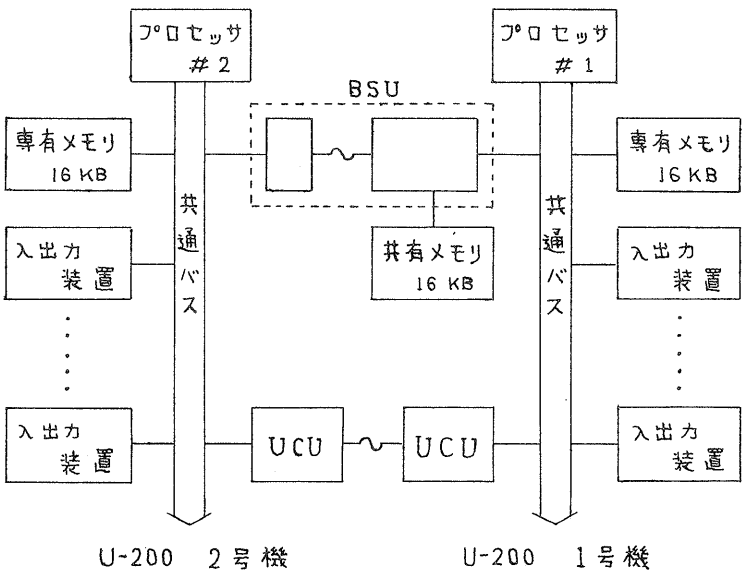


図1 DMSのハードウェア構成

2.1 バス・スイッチング装置（BSU）

BSUの全体の回路は、図2のような構成になっており、全部で7つのモジュールに分けられている。中央の4つのモジュールが実際にスイッチング操作を行なう部分で、メモリ装置と接がれている。両側の3つのモジュールは、共通バスとのバッファ回路、アドレス変換回路、33mのケーブルに対するドライバー回路などから成り立っている。

BSUは、共通バスからの信号をメモリ装置に対する信号に変換するばかりでなく、両方の共通バスから同時に操作された場合の排他制御も行なわなければならない。この同時操作に対する処理回路は重要で、アクセス時間の増加を極力減らして、確実にスイッチング動作が行なわれることが必要となる。DMSにおいてはプロセッサの数が2台であるが、この数が増えた場合の排他制御回路は、個人の飢餓なども考慮しなければならないので複雑なものとなる。このような資源管理を行なう制御回路は、一般にアービター（Arbiter）と呼ばれ各種の回路が考えられてきている[5]。このBSUにおいてはアービターとしては、図3のようなJKフリップ・フロップを使用した競合回路を用いて構成している。この図3において、REQ/P1は、プロセッサ1のある共通バスからの要求信号で、REQ/P2は、プロセッサ2側の要求信号である。SELECTは、メモリ装置の専有権を示す信号である。

またBSU全体の通過時間は、実質的なアクセス時間の増加となりシステム全体の効率に対して大きな影響を与える。したがってこのような装置に対しては、簡潔な回路に高速の素子を使うことが必要となる。このような目的のためにここでは、ショットキ・TTLやTSL（Tri-State Logic）と呼ばれる素子を積極的に使用した。その結果各モジュールの素子数は表1のようになった。

共有メモリに使用したコアーメモリ装置の性能は。サイクルタイム650ns、アクセスタイム280nsである。これにBSUを付加することによって、アクセスタイムは400ns（1号機）、960ns（2号機）に増加した。またこの共有メモリをプロセッサがアクセスする時の共通バス上に送出されているSAVI信号の時間は、830ns（1号機）、1380ns（2号機）、となった。

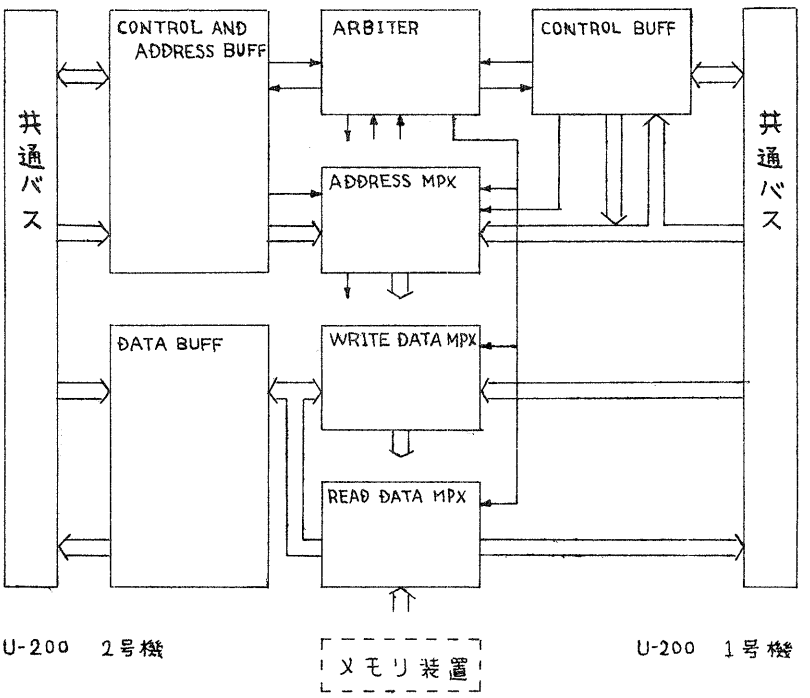


図2 BSUのブロック図

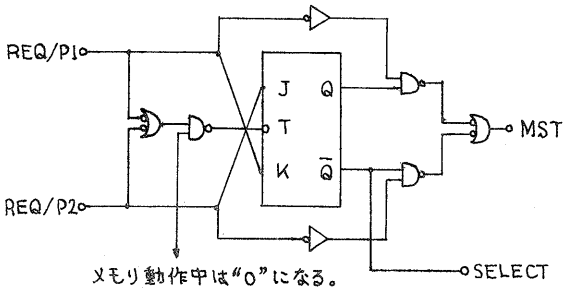


図3　ARBITERの競合回路

表1　各モジュールの素子数

モジュール ＼ 素子	ショットキ・TTL	TSL	一般のTTL
CONTROL BUFF	1	1	7
WRITE DATA MPX	0	10	5
READ DATA MPX	0	0	13
ADDRESS MPX	1	8	3
ARBITER	5	1	9
CONTROL AND ADDRESS BUFF	0	1	11
DATA BUFF	0	0	14

2.2 高速チャネル結合装置（UCU）

DMA方式による、メモリ装置間のデータ転送を行なう装置である。手続きとしては、プログラムによってデータの個数、データの先頭アドレス、コマンドを両方の計算機からセットすることによって転送が開始される。またUCUからプロセッサへの割込みは、他方のプロセッサによってコマンドがセットされた場合のアテンション割込みとデータ転送が終了した時のデバイス・エンド割込みとがある。転送方式は、半二重通信方式で、1語（16ビット）単位の並列転送を行なっている。しかしDMSにおいては、UCUを実際のデータ転送には使わず、割込み信号の転送による機能をプロセッサ間の通信のためにだけ使用している。

2.3 共有メモリの競合

共有メモリに対するアクセスの競合によって起こるプログラムの実行時間の遅れを検討するために、π（3.14……）の計算を行なうプログラムを実行させ、表2のような結果を得た。このプログラムは、固定小数点方式で小数点以下2000桁を計算するもので、命令部約1KB、データ部約4KBからなっている。これを先頭アドレスだけ異なるものを2つローディングして、そしてそれを2つのプロセッサによって同時に実行させ、その計算時間を測定したものである。この時の計算時間の増加に対しては、次の2つの原因がある。

〔1〕共有メモリが専有メモリよりもアクセスタイムが長いため。

〔2〕共有メモリに対するアクセスの競合によりプロセッサが待たされるため。

表2において、片方のプロセッサのみで実行した時の計算時間が、共有メモリを多く用いるにしたがって長くなるのは、〔1〕の原因による遅れである。特に2のプロセッサにおいては、33mのケーブルの遅れによって50%近くも計算時間が増加する。一方、両方のプロセッサで同時に実行した場合には、〔1〕の原因によるものに、〔2〕の原因による遅れが加えられる。特にこの場合には、2つのプロセッサの計算時間が非常に接近してくるという現象が起きてくるので、効率のよいシステムを作るためには、共有メモリをどのように使うかが重要になると考えられる。

表2　πの計算時間　（単位：msec）

プログラムのローディング場所 ＼ 命令部 / データ部		専有 / 専有	専有 / 共有	共有 / 専有	共有 / 共有
片方のプロセッサのみで実行した場合	プロセッサ(#1)	108	113	122	127
	プロセッサ(#2)	108	118	141	151
両方のプロセッサで同時に実行した場合	プロセッサ(#1)		115	143	153
	プロセッサ(#2)		119	146	154

3. ソフトウェア[6][7][8]構成

DMSは、基本的に並列処理システムの実現を目的としているが、実際にこれをある問題のために適用しようとする場合には、いろいろと難かしい問題が生じてくる。特にDMSが多重プロセッサ型であることと、ハード的構成からくる制限が存在することとによって効率のよいシステムを開発するためには、モニタと呼ばれるものが必要となる。このモニタがDMSの各種資源の管理を行ないかつ並列処理に対する機構を備えることによって、実際の問題へのDMSの適用が可能となる。以下においてこのモニタの特徴を示す。

(1). モニタは、共有メモリ上に常駐しリエントラント構造をしているので、2つのプロセッサに対して1つしか存在しない。またOS全体は、階層構造になることを重視した構成を取り入れ可変性とデバッキングの容易さが考慮されている。

(2). 一般に頭初から多重プロセッサが考慮されている大型計算機などにおいては、各種の機能がハードウェアにおいて備わっているが、このDMSで使用したミニコンピュータにおいては、多重プロセッサを実現するための各種の機能が不足している。したがってここでは、ソフト的にこのような機能の追加を行ない、ミニコンピュータによる並列処理システムを可能にしている。

(3). 装置間のデータの授受を行なう共通バスが、DMSにおいては複数個存在し、そしてそれぞれに入出力装置やプロセッサが接続されている。したがって各入出力装置は、同じ共通バスのプロセッサだけからしか操作されることができないという制限が生じる。そのためにDMSのOSには、そのような形態を意識することなしに個々のプログラムから、どの共通バスにある入出力装置でも操作できるような機構が付加されている。

(4). メーカーより提供されるU-200のソフトウェアの中に制御用のCRS(Core Realtime System)と呼ばれているものがある[9]。これはU-200のハードウェア機能を非常に効率よく用いられるように考慮されているので、このDMSにおいてもCRSで用いられている手法の中で有効であると考えられる所は、積極的に導入することにした。

3.1 OSの構造

表3 OSの階層構造

Level	Facility	Macro Operation Name
L5	Device Management	(System Process)
L4	Process Communication Primitives	SM,RM
L3	Process Management	EP,ST,FP
L2	Synchronization Primitives	PO,VO
L1	Dispatcher	DE
L0	Extended Hardware	SL,EL,LC,SI

L0： TS命令に相当する最も低レベルの同期命令とプロセッサ間の通信機構を実現している。

L1： プロセッサのプロセスへの割当てを行なう。その方法は、各プロセッサがPCB群から最も優先度の高いReady状態のプロセスを見つけて実行するという方式をとっている。

L2： プロセス間の同期操作の基本命令となる、セマフォ変数に対するP/V命令を供給する。

L3： プロセスをReady状態やStop状態にする命令などを供給する。

L4： ユーザプロセス間の同期命令となるメッセージの授受によるプロセス間の通信機構を実現する。

L5： 入出力装置の管理を行なうシステム・プロセスである。各入出力装置に対してそれぞれデバイス管理プロセスが割当てられている。

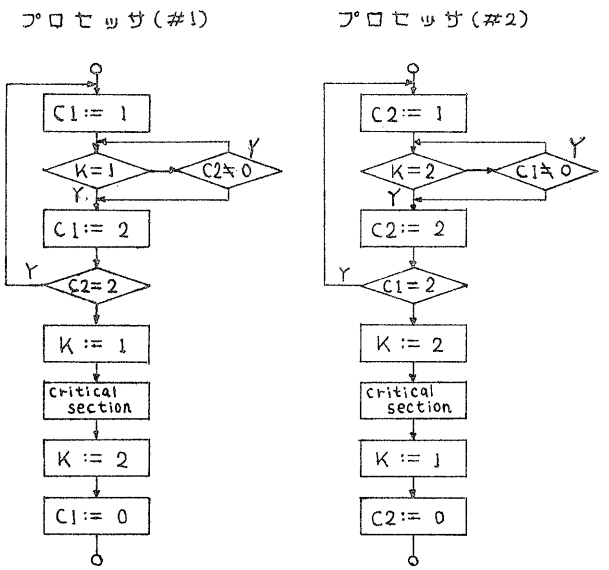
3.2 プロセッサ間の同期

最も低レベルの同期命令としてプロセッサ間の同期命令が必要である。これを実現するためにハード的に機能を追加することも考えられるが、このDMSにおいては図4に示すようなアルゴリズムを用いてソフト的に行なっている[10]。これはKnuth（1966）によって考えられたもので、3つの整数型の変数を用いる必要があり、ここではこれをロック変数と呼ぶことにする。critical sectionの前で行なう操作を、EC命令（Enter C.S.）、後の操作をLC命令（Leave C.S.）として、モニタ・マクロ命令の中に準備されている。またこのアルゴリズムにおいては、1つのロック変数に対する操作が、割込みによって中断されることは許されない。したがってEC命令の最初で自系のプロセッサの割込みを禁止状態にし、LC命令の最後でもとの状態にもどす操作を行なっている。システムで起きるTS命令を使わなければならない同期問題の中で、衝突型の問題に対してこれらの命令を使う場合には、TS命令の1ビットを1つのロック変数に対応させて用いることができる。しかしその他の問題に対しては、ロック変数を用いてTS命令のシミュレーションを行なわなければならない。この命令は当然busy form of waitingになるのでDMSのモニタにおいては、以下のような所に対してのみ使用している。

〔1〕. セマフォ変数に対するP/V命令の非可分操作。

〔2〕. システムの共有資源に対する相互排除。

〔3〕. プロセッサ間の通信機構。



（初期値：C1 = C2 = 0, K = 1）

図4 同期命令のアルゴリズム

3.3 プロセッサ間の通信

DMSにおいては、プロセッサ間の割込みに対する特別のハード的機能はないので、UCUをこの目的のためにのみ使用した。そして実際のデータは、共有メモリのモニタの中に図5のような特定語（32ビット，初期値：0）を1つ設定し授受

することにした。そして図6のような手順で行なっている。

プロセッサ1がプロセッサ2にデータを送る場合。

(1). プロセッサ1は、特定語のOビットが、データの有無のフラグ・ビットになっているので、これを調べてOFFになるまで待つ。

(2). OFFになったら、プロセッサ1はデータ部とモディファイ部に情報を格納する。(この時はすでにフラグ・ビットはON になっている。)

(3). プロセッサ1は、プロセッサ2に対して割込み信号を出す。そうするとプロセッサ2では、外部0割込みが発生し、AKI 命令によってプロセッサ1からの割込みであることがわかる。

(4). プロセッサ2は、特定語のデータを読み取り、フラグ・ビットをOFFにしてモディファイ部に指定された処理を行なう。

特定語をモニタの中で1つしか設けなかった理由は、UCUの機能が半二重通信であること、プロセッサを3台以上に拡張する時にも処理ルーチンが簡単であると考えられることとのためである。しかしこのためにこの通信機構に対するプロセッサ間の同期を考えなければならない。

ここにおける同期問題は、プロセッサ間の相互排除問題とも生産者-消費者問題とも考えることができる。前者の場合には、同期命令を用いて2つのプロセッサがこの通信機構の専有権を獲得するのであると考えられるのであるが、一般的な相互排除問題とは異なりその解放は他のプロセッサにゆだねられてしまっている。また後者であると考えた場合には、バッファは1つで、そのバッファの内容が空の時には決して消費のための処理が生じないということと、2つのプロセッサのどちらもが生産者になりかつ消費者になりうるという形になっていることとが特徴となる。このような同期問題に対して、データを他のプロセッサに送る処理がプロセスからのみではなくモニタの核ルーチンからも使用される可能性があるので、ここではP/V命令ではなく図7のようなロック変数によるTS命令をシミュレートした方法を用いている。

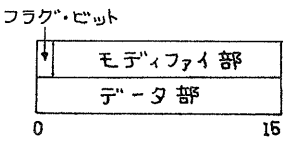


図5 特定語の構造

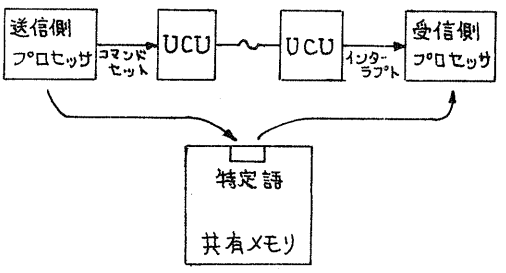


図6 通信機構の手順

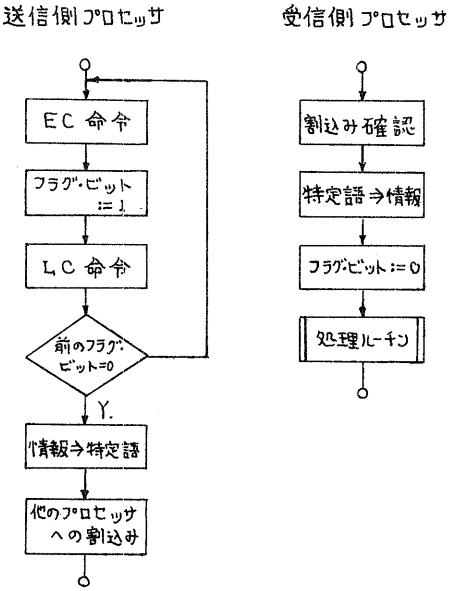


図7 通信機構の同期処理

3.4 プロセスの処理

DMSにおいては、仕事の処理単位をプロセスと呼び多重プログラミング方式によって処理している。プロセスはそれぞれPCB（Process Control Block）と呼ばれるものによって管理され、図8のように状態を遷移する。DMSが制御用システムなどへの適用を目的としているので、5段階の優先度をもった優先処理を行なっている。これによって緊急の処理を行なう必要のあるプロセスに対してプリエンプションを起こさせることが可能となる。プロセッサのプロセスへの割当てを行なう方法は、各プロセッサがPCB群から最も優先度の高いReady状態のプロセスを見つけそして実行するという方式をとっている。

モニタは、核の部分とマクロ命令の部分に分けることができる。したがってプロセッサが核の部分の仕事を実行するときはプロセスとしては実行されない。またマクロ命令の部分は、プロセスから呼ばれた場合にのみプロセスとして実行される。

モニタのリエントラント構造を実現するためには、レジスタの内容などを退避させる作業領域を設けなければならない。したがってここでは、プロセスに対する作業領域はPCBの中に、また核ルーチンの作業領域は各優先レベルを管理するLVT（Level Table）と呼ばれる中にそれぞれ設け、そしてそれをスタックとして使用している。なおこのLVTは各プロセッサごとに専有メモリ上に作られている。

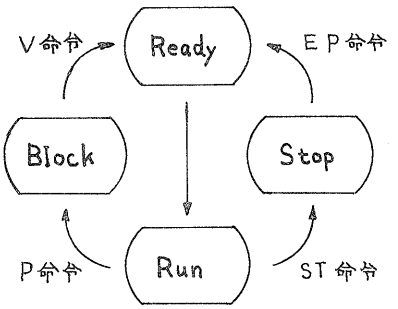


図8　プロセスの状態

3.5 プロセス間の同期と通信

1. P／V命令[11]

プロセス間の各種の同期操作に対する手段として、Dijkstra（1968）提唱のP/V命令を使用している。これは整数値をとるセマフォ変数に対して操作を行なう命令である。しかしDMSにおいてはシステムの混乱を避けるという理由などからユーザ・プロセスでこの命令を使うことは行なっていない。

2. メッセージの授受

システムの重要な機構として、プロセス間のメッセージの授受がある。ユーザ・プロセスでは、メッセージ通信用のマクロ命令であるRM命令（Read Message）とSM命令（Send Message）を用いてプロセス間の同期や通信を行なう。この命令はそれぞれ受信バッファ、相手プロセス名と送信メッセージをパラメータとしてモニタに対して通信要求を出す。実際のメッセージの授受には、システム・バッファと呼ばれる領域がメッセージ用のバッファとして用いられ、メッセージはそれが読まれるべきプロセスのPCBをターミナルとしてリンクされている。そしてプロセスがRM命令を出すと、メッセージ・リンクの最初のメッセージが渡され、またメッセージがない場合には、このプロセスはブロックされる。またSM命令によって送られたメッセージは、メッセージ・リンクの最後尾に加えられ、もしこのプロセスがメッセージ待ちによってブロックされているならば、起動要求が出される。

3.6 入出力管理

一般のミニコンピュータにおける入出力装置に対する操作は、サブルーチンコールのような形式で行なわれている。しかしこのDMSのようなシステムにおいては、共通バスが複数あるのでこのような方法では実現することができない。たとえばあるプロセスを実行しているプロセッサと異なった共通バスにある入出力装置を、このプロセスが直接に操作することは不可能である。したがってDMSでは、あらかじめ1つの入出力装置に対して1つのデバイス管理プロセスを対応させて、基本的にこの入出力装置の操作は、このプロセスにゆだねることにする。そして入出力装置が接続されている共通バスと同じ共通バスにあるプロセッサのみが、これに対応するデバイス管理プロセスへの割当てが可能であることを、このプロセスのPCBに示しておくことにする。

そこでユーザ・プロセスがある入出力装置を使いたい場合には、それに対するコマンドなどを、SM命令によって対応するデバイス管理プロセスに送り、RM命令によって待つ。デバイス管理プロセスは、メッセージを受け取ると各入出力装置に対して準備されているUCB(Unit Control Block)に必要な情報をセットして、装置に対してスタートコマンドを出す。そしてUCBの中にある終了を待つためのセマフォ変数に対してP命令を出して入出力装置の終了を待つ。割込み処理ルーチンにおいては、すべてのデータの処理が終るとこのセマフォ変数に対してV命令が出され、デバイス管理プロセスが再び実行権を得て、終了情報のメッセージがユーザ・プロセスに送られる。ただし入出力装置とユーザ・プロセスのデータの授受は、すべて共有メモリを使用しなければならない。図9はこの操作の流れを示している。

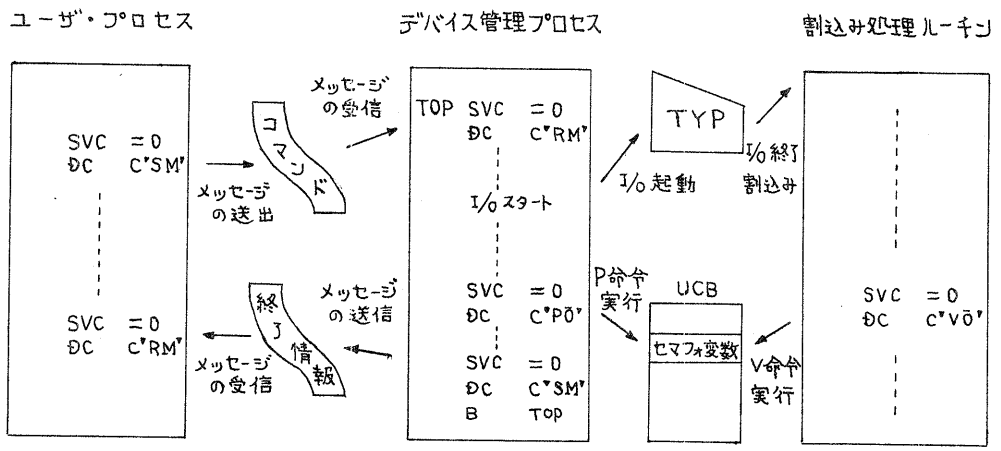


図9 入出力装置の操作

4. おわりに

以上のような機能をもった多重プロセッサによる複数プログラムの並列処理を行なうOSの骨格を完成した。このモニタの大きさは、核の部分が0.85KB、マクロ命令の部分が1.17KBで合計約2KBのものとなった。またモニタの中に準備されているマクロ命令の中で主要な命令についての実行時間の測定を行なったのでその結果を表4に示す。

表4　マクロ命令の実行時間　（単位：msec）

Macro Operation Name		プロセッサ（#1）	プロセッサ（#2）
EC	Enter C.S.	0.33	0.40
LC	Leave C.S.	0.30	0.37
PO	P Operation	0.88	1.0
VO	V Operation	0.88	1.0
SM	Send Message	6.8	8.7
RM	Read Message	5.2	6.3

このような共有メモリ結合によるミニコンピュータ複合体システムにおいては、実際の問題に適用した場合にどの程度の効率の向上が望めるかという疑問に対する考察や実験を行なう必要があると考えられる。またプロセッサやメモリ・モジュールを増やしていった場合には、また異なったいろいろな問題が生じると考えられる。したがって今後はこれらの問題についての検討も行なっていく予定である。

謝辞

本システムの設計・開発にあたり御教示いただいた本学の松原康夫助手、関口芳広助手、二木弘氏に感謝致します。また本研究に協力して頂いた高木道裕君、松土一俊君に謝意を表します。

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，PP 525 ～ 533 （1974）
2). W.A.Wulf and C.G.Bell ： C.mmp - A multi - mini - Processor，Proc. of FJCC，Vol. 39 ，PP 765 ～ 777 （1972）
3). 三木雄作：ミニコン複合体システム TMCS－40，情報処理学会計算機アーキテクチャ研究会資料 75－17 （1975）
4). 上林、徳田、他：KOCOSのアーキテクチャ(1)&(2)，情報処理学会計算機アーキテクチャ研究会資料 75－16 （1975）
5). W.W.Plummer：Asynchronous Arbiters ，IE^3 ，Tr.on C Jan. （1972）
6). J.J.Donovan and S.Madnick ： Operating System ，McGram - Hill （1974）
7). P.B.Hansen ： OPERATING SYSTEM PRINCIPLES ，Prentice - Hall （1973）
8). P.B.Hansen ： The nucleus of a Multiprograming System，CACM ，Vol.13，No.4，PP 238 ～ 241 （1970）
9). FACOM U-200 CRS文法編
10). D.E.Knuth ： Additional Comments on a Problem in a concurrent Programming Control ，CACM ，Vol.9 ，No.5，PP 321 ～ 322 （1966）
11). E.W.Dijkstra ： Co-operating Sequential Processes ，Programming Languages，Academic Press，PP 43 ～ 111 （1968）