モニターシステムの設計と運用について*

高田　勝**　牛島和夫**　有田五次郎**

1. はじめに

東京大学に全国共同利用の大型計算機センターが設置され，わが国の大学においても大型計算機を普通に使用できるようになった．しかし，この計算センターをより有効に使用するためには，大型計算機センターはもとより，各地方の大学計算センターにとっても，またセンター相互間でも解決しなければならない問題は非常に多い．従来，九州大学中央計数施設においては，OKITAC-5090 H を用いて，ALGOL を運営の中心言語にし，load and go 方式（操作は閉鎖式）によって計算業務を行なってきた．しかし，計算需要の増加および東京大学に設置された大型計算機の共同利用のために新たに増加した仕事に対処するため，計算機による仕事の監視プログラム（モニター）の開発が必要となった．

われわれに課せられた条件は，次のようなものである．

（1） おもな仕事は従来の ALGOL，東大大型機の共同利用のための FORTRAN 言語の事前検査，カードの作表，複製である．特にカード関係の仕事は専用の会計機を備えればよいのであるが，これは経済的に不可能なので，OKITAC-5090 H を利用せざるを得ない．しかし従来のような処理方式では機械時間の浪費である．

（2） 兼任の操作員（元来はキーパンチャー）による操作，また夜間の無人運転が可能であるように，ハードウェアのエラー，プログラムによる偶発的事故に対して，完全な処理プログラムを持つこと．

（3） センター運営上不可欠なアカウント（使用時間，使用印刷用紙枚数など）がとれること．

（4） マルチプログラム方式を採用して，入出力の同時並行処理を行なうこと．

（5） 仕様の全く異なる既製のコンパイラー，ユティリティルーチンがわずかの変更で使用できること．

（6） モニターが，いままでユーザーの使用していた領域を侵さないこと．

（7） 紙テープ（ALGOL）とカードの二つの入力媒体の処理をしなければならない．

（8） ハードウェアーの条件として下記の機能を持つ：

記憶容量：8 K語（1語 42 ビット），磁気ドラム：12 K 語，磁気テープ：4台（1チャンネル），各種の割込みの機能，チャンネルによるマルチプロセスの機能．

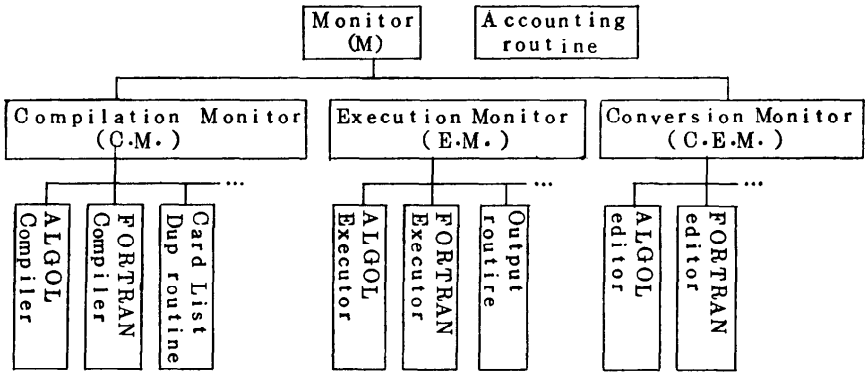
2. モニターシステムの概要

前記諸条件を勘案して設計されたわれわれのモニターシステムは，コントロールカードシステムによる磁気テープベースのバッチ処理方式をとり，その構成は親モニター（monitor）とコンパイルモニター（compilation moniter），実行モニター（excecution monitor），変換編集モニター（conversion monitor）からなり，各モニターの監視下に各種コンパイラ，エクゼキュータなどがある．それを**第1図**に示す．

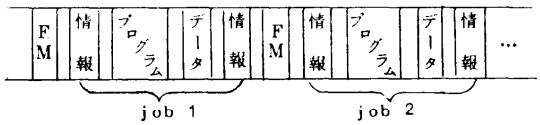
コンパイルモニターは ALGOL，FORTRAN の翻訳，エラーチェック，実行可能なプログラムへの変換，データカードの処理を行ない，プログラムの種類，メモリーレイアウト・アカウント情報などをともに磁気テープにおさめ，実行可能な job の列を磁気テープ1に作る（**第2図**参照）．プログラムおよび/またはデータにエラーがあると，その job は磁気テープに入らない．実行モニターはコンパイルモニターによって作られた job の列を磁気テープから読み出し，プログラムの種類に応じてエクゼキュータ（Executor）を呼び出して実行を行ない，磁気テープ2にアウトプットファイルを作製する．変換編集モニターは実行モニターによって作られたアウトプットファイルを2進/10 進変換し，高速度印刷装置またはカードせん孔機上のイメージに編集して，磁気テープ3におさめる．ここで作製された結果は，次のバッチの計算実行時に実行モニ

* Moniter System at Kyushu University Computation Center by Masaru Takata, Kazuo Ushijima and Ituziro Arita (Computation Center, Kyushu University)

** 九州大学中央計数施設



第1図　モニターの構成



第2図　磁気テープ1上の情報

ターの中のプログラムユニット o によって，計算実行と並行して，高速度印刷装置またはカードせん孔機に出力される．

これはチャンネルの機能により，1ブロック印字するのに本体の演算時間は数クロックしか必要としない．

親モニターは磁気テープの管理，サブモニター間の制御の受け渡し，操作員のコンソールからの指示（バッチの打切り，再開など）の判断を行なう．エラー処理ルーチンは各サブモニターが必要にして十分なだけ持っている．

3. モニターシステムの詳細

3.1 記号の定義

われわれのモニターシステムを説明するために記号を用いる記述を試みる．以下に行なう記号の定義で，大文字は，情報を示し，小文字は操作を示すものとする．

S^j: ソースプログラム，ソースデック

P^j: コンパイルされてできたオブジェクトプログラム

D^j: データデック

R^j: 入力データファイル

W^j: 出力リストおよび出力様式（format）

E^j: W^j を変換および編集した結果

O^j: E^j を出力機器に出した最終結果

j: 第 j 番目の job に関することを示す．一つの job に着目しているときは省略する．

m: 親モニターの機能を示す演算子

cm: コンパイルモニターの機能を示す演算子

em: 実行モニターの機能を示す演算子

cem: 変換編集モニターの機能を示す演算子

${}_ic$: i 種のコンパイラを示す演算子

${}_il$: i 種のローダを示す演算子

${}_ie$: i 種の変換編集プログラムを示す演算子

o: 出力のプログラムを示す演算子

${}_ip$: i 種のオブジェクトプログラム（情報 P^j がローダー ${}_il$ により実行可能になったもの）と i 種のエグゼキュータを示す演算子

${}_iK$: i 種のプログラム言語を示すコントロールカード，ただし ${}_0K$ は ＊EOF カード．

${}_iI$: i 種のプログラム言語であることを示す情報，ただし ${}_0I$ はアカウントデータ

i: i 種のプログラム言語に関することを示す．

M_k: モニターの切り換えの情報（各 M_k は後述）

${}_x(X)_a \Rightarrow (Y)_b{}_y$: a の上の情報 X に x が作用して b 上の情報 Y にかえ，演算子も y に変化することを示す．

さらに，一般に

${}_x(X)_{a,b,\cdots} \Rightarrow (Y)_{c,d,\cdots}, (Z)_{e,f,\cdots}\cdots$ y: a または b または…上の情報 X に x が作用して，c または d または…上の情報 Y にかえ，かつ e または f または…上の情報 Z にかえ演算子も y にかえることを示す．

3.2 マクロにみたモニターシステムの機能

仕事の流れに着目するとわれわれのシステムは次のように書くことができる.

① $cm\ (S^1D^1S^2D^2\cdots S^ND^N)_{\mathrm{CR,PTR}} \Rightarrow (P^1R^1P^2R^2\cdots P^NR^N)_{\mathrm{MT1}}em$

② $em(P^1R^1P^2R^2\cdots P^NR^N)_{\mathrm{MT1}} \Rightarrow (W^1W^2\cdots W^N)_{\mathrm{MT2}}cem$

③ $cem(W^1W^2\cdots W^N)_{\mathrm{MT2}} \Rightarrow (E^1E^2\cdots E^N)_{\mathrm{MT3}}cm$

④ $o(E^1E^2\cdots E^N)_{\mathrm{MT3}} \Rightarrow (O^1O^2\cdots O^N)_{\mathrm{LP,CP}}em$

ここで一つの job は S^j と D^j からなり N 個で一つのバッチを構成する. したがって, われわれのモニターシステムは, $S^1D^1S^2D^2\cdots S^ND^N$ として与えられた系列を最終的に $O^1O^2\cdots O^N$ として出力するのであるが, 仕事の流れからみると, 上の四つの段階に大別して設計されているのである. 4段階の変換のうち ④ はチャンネルの機能を用いて ② と並行して行なわれるものであるが, 衛星計算機をもつシステムでは, ② が主計算機, ① ③ ④ は衛星計算機の受け持つ仕事である. すなわち ① は ② における入力ファイルを作るという意味で衛星計算機のすべき役割であるが $S \Rightarrow P$ というコンパイルが行なわれているのは, ① を受け持つのが実は主計算機であり, 入力時間内にコンパイルも終了できるためである. ③ ④ については, 一般の衛星計算機をもつシステムでは $e^1o^1e^2o^2\cdots e^No^N$ の順で行なわれるものと思われるが, われわれのシステムでは $e^1e^2\cdots e^No^1o^2\cdots o^N$ の順になっている.

3.3 モニターシステムの機能

前節では, 仕事の流れに着目して, 仕事が四つの段階に大別されることを述べたが, ここでは, **3.1** 節の記号を用いて, モニターシステムをその機能の側から詳述する.

まずモニターシステムへの入力となる一つの job は Id カード M_0, プログラム言語の種類を示すコントロールカード ${}_iK$ につづいて S^jD^j, 最後に ＊EOF を示す ${}_0K$ からなる. また一つのバッチの最後にはバッチの終を示す情報 M_1 (＊RUN カード) をそえる. したがってモニターシステムに対する入力系列は, 実際には次のようにあらわせる.

$$M_0{}^1{}_{i_1}K^1S^1D^1{}_0K^1M_0{}^2{}_{i_2}K^2S^2D^2{}_0K^2\cdots M_0{}^i{}_{i_j}K^jS^jD^j{}_0K^i\cdots M_0{}^N{}_{i_N}K^NS^ND^N{}_0K^NM_1$$

そしてこの系列を処理するシステムは次のように記述できる.

⑤
$m(M_0)_{\mathrm{CR}} \Rightarrow cm$ (1)
$m(M_1)_{\mathrm{CR}} \Rightarrow (M_2)_{\mathrm{MT1}}em$ (2)

⑤
$m(M_2)_{\mathrm{core,manual,CR}} \Rightarrow em$ (3)
$m(M_3)_{\mathrm{core,manual,CR}} \Rightarrow cem$ (4)

⑥
$cm({}_iK)_{\mathrm{CR}} \Rightarrow ({}_iI)_{\mathrm{MT1}}{}_ic$ (1)
${}_ic(SD)_{\mathrm{CR,PTR}} \Rightarrow (PR)_{\mathrm{MT1}},(E)_{\mathrm{MT3}}cm$ (2)
$cm({}_0K)_{\mathrm{CR}} \Rightarrow ({}_0I)_{\mathrm{MT1}}m$ (3)

⑦
$em({}_iI)_{\mathrm{MT1}} \Rightarrow ({}_iI)_{\mathrm{MT2}}{}_il$ (1)
${}_il(P)_{\mathrm{MT2}} \Rightarrow {}_ip$ (2)
${}_ip(R)_{\mathrm{MT1}} \Rightarrow (W)_{\mathrm{MT2}}em$ (3)
$o(E)_{\mathrm{MT3}} \Rightarrow (O)_{\mathrm{LP,CP}}em$ (4)
$em(M_2)_{\mathrm{MT1}} \Rightarrow (M_3)_{\mathrm{MT2}},(M_3)_{\mathrm{core}}m$ (5)
$em(M_4)_{\mathrm{manual}} \Rightarrow (M_2)_{\mathrm{CP}},(M_3)_{\mathrm{MT2}}m$ (6)

⑧
$cem({}_iI)_{\mathrm{MT2}} \Rightarrow {}_ie$ (1)
${}_ie(W)_{\mathrm{MT2}} \Rightarrow (E)_{\mathrm{MT3}}cem$ (2)
$cem({}_0I)_{\mathrm{MT2}} \Rightarrow ({}_0I)_{\mathrm{CP}},({}_0I)_{\mathrm{MT3}}cem$ (3)
$cem(M_3)_{\mathrm{MT2}} \Rightarrow m$ (4)
$cem(M_5)_{\mathrm{manual}} \Rightarrow (M_3)_{\mathrm{CP}}m$ (5)

モニターははじめ親モニターの制御のもとにあるので, 入力系列への操作は ⑤-(1) からはじまり ⑥-(1), (2) によりコンパイルが実行され, ⑥-(3) によって, ＊EOF カードが読まれると job の区切りの情報を磁気テープ1上に書いて制御を再び親モニターにもどす. 第 N 番目の job までこのような処理をくり返し, 第 $N+1$ 番目としてカードリーダ上にバッチの区切りの情報 M_1 を読むと ⑤-(2) により磁気テープ上にバッチの区切りの情報 M_2 を書いて制御を実行モニターにうつす. 次に ⑦-(1), (2) により i 種のローダによりオブジェクトプログラムが磁気テープ1より主記憶装置内に格納され, ⑦-(3) により計算が開始されて計算結果を磁気テープ2に出力する. このように次々計算を実行し ⑤-(2) で書いた M_2 に出会うと ⑦-(5) により制御は親モニターにもどる. 親モニターは ⑦-(5) で主記憶装置上に残された情報 M_3 により, ⑤-(4) によって編集変換モニターに制御を渡す. ここで ⑧-(1), (2) により W (2進情報) が E (文字情報) に変換編集されて磁気テープ3に格納される. ⑧-(3) では job の区切りでアカウント情報がオンラインのカードパンチにせん孔される. このように続けて, ⑦-(5) で磁気テープ2に書かれた M_3 に出会うと ⑧-(4) により制御は親モニターにもどる. さて ⑧-(2) で編集された E は, 別のバッチにおける ⑦-(3) の実行中に割込みによって ⑦-(4) として実行されるのであるが, ⑦-(3) の ${}_ip$ は, 割込みによって結果的に m 個の副演算子に分割されるのでこれを

$$_{i}p = {}_{i}p_1 {}_{i}p_2 \cdots {}_{i}p_m$$

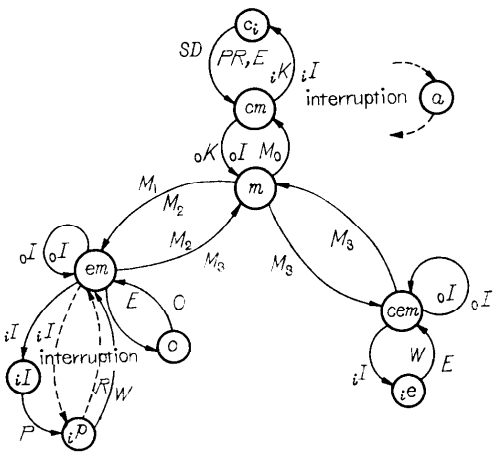
のようにあらわすと

$$⑨\begin{cases} {}_{i}p_k \Rightarrow em \quad (割込み) \\ em \Rightarrow o \\ o \Rightarrow em \\ em \Rightarrow {}_{i}p_{k+1} \end{cases}$$

のようになる．

なお実行モニターの実行打切りを操作員が操作卓から行なうものが ⑦-(6) で M_4 はそのとき操作員が与える情報を意味する．また編集変換モニターの実行打切りを操作員が操作卓から行なうのが ⑧-(5) で M_5 はそのとき操作員が与える情報である．それぞれの打切り時にカードパンチ上に出力された ⑦-(6) の M_2 情報，あるいは ⑧-(5) の M_3 情報を用いて，それぞれの処理の続きを実行するには，それぞれ ⑤-(3) または ⑤-(4) によりカードリーダから再開の指示情報を与えることになる．

われわれはここにあげたアルゴリズムを流れ図のかわりにしてモニタープログラムを作成したのであるが，この ⑤ ⑥ ⑦ ⑧ の関係をステートダイアグラムの形に表現したものが**第3図**である．この図で (a) は



第3図　ステートダイアグラム

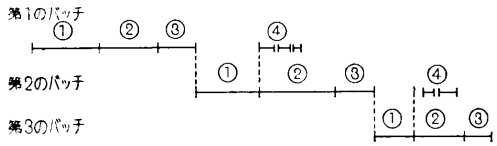
単位時間ごとの割込みを示し，CPU 占有時間の計数を行なっている．

なお状態の遷移によって呼び出される各システムプログラムは 12K 語のドラムに格納されている．

4. 多重プログラムの実行

前節で ② と ④ の列が多重に行なわれていることを述べたが，② の実行中にモニターが主記憶装置に余裕のあることを知れば，割り込みによって ④ を行なう．もちろんこのとき磁気テープ3に編集されている E は一つ前のバッチの E である．E はすでに編集が終っているから ④ のプログラム（詳しくは ⑦-(4)）としては，磁気テープからの読み出しと印刷命令の発令だけであとは各チャンネルが計算機本体と並列に実行してくれる．

印刷命令を出すと直ちに計算機本体は ② の実行のつづきにもどる．チャンネルから出力終了の信号がくれば再び割込みが生じて磁気テープ3から E の読み出し，印刷命令の発令，② の実行への復帰ということをつづける．このように ④ のプログラムは数ステップの命令と入出力用のバッファーだけからなっている．このようにして，**第4図**に示すように，印刷装置



第4図　計算の重ね合わせ

の動く時間を ② の実行と全く並列にできるので，全体としてこの部分に衛星計算機を増設したと同じ効果を期待している．

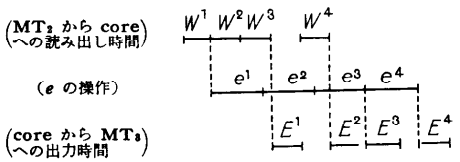
次に ① のプログラムが多重にならないのは次の理由による．

1) カードと紙テープが混在しているため，たとえば一つの S^j が数本の紙テープに分れる時は，どうしても計算機を一時停止してテープのかけかえを要する．多重プログラムにしておくととまらなければならないところで割込みを生じて計算機が走り出してしまうことがある．

2) 8K メモリーではコンパイルに精いっぱいで別のプログラムの領域が見出せない．

$$W^1 W^2 \cdots W^N \Rightarrow E^1 E^2 \cdots E^N$$

(MT_2)　　(MT_3)



第5図　変換編集モニター

さらに ③ のプログラムを特に独立させている理由は OKITAC-5090 H 程度の速度の計算機では，**第5図**に示すように変換および編集の時間で大部分の計算時間をおおうことができる．実際には W^j, E^j は適当なバッファリングによりさらに細分されているので，($W^j=W_1^jW_2^j\cdots W_{n_j}^j$, $E^j=E_1^jE_2^j\cdots E_{m_j}^j$)，第5図での入出力のためにはみだす頭（$W_1^1$）としっぽ（$E_{m_N}^N$）はわずかである．

計算機が多重入出力処理，割込み機能，多重プログラムの能力をもっていても記憶容量が小さいため能力を十分に発揮できないこともありうる．われわれの場合でも，もし記憶容量にもう少し余裕があれば，入出力用のバッファを複数個増やすことによって，ほとんど実効計算ばかりで計算時間（CPUT）をうめることが可能となるはずである．

5. 種々の利用法

モニターによるバッチ処理の概要は以上述べたとおりであるが，実際の運用上は種々な要求により以下に述べるような便宜が考えられている．

（1） 実際に一つの job に着目すると実効計算時間は，load and go 方式より短かくなるが，計算機にかけてから結果を得るまでの時間が長くなる．そこで load and go（特急処理）の要求に応えるために，コントロールカード ＊EXECUTE を設けておく．このコントロールカードがくると各磁気テープをそのままの状態に保って直ちに $S^jD^j\Rightarrow P^jR^j$, $R^j\Rightarrow W^j$, $W^j\Rightarrow E^j$, $E^j\Rightarrow O^j$ が行なわれる．したがって特急通過後の各磁気テープの $j+1$ 番目の情報は $j-1$ 番目の情報のあとにつづくことになる．

（2） コンパイルのみの要求を認めて ① のみの操作によりターンアラウンドタイムを向上させる．

（3） S^jD^j にエラーのあるものは実際 ②，③，④ は空になる．

（4） カードの作表や複製はコンパイルのみの要求とほぼ同じで ① においてのみ処理される．

（5） 1日の仕事の打切りと翌日の仕事の操作指示カードの出力は，操作員からの割込みにより，その事情は**3.3**節に述べた ⑦-(6) または ⑧-(5) のようになっている．

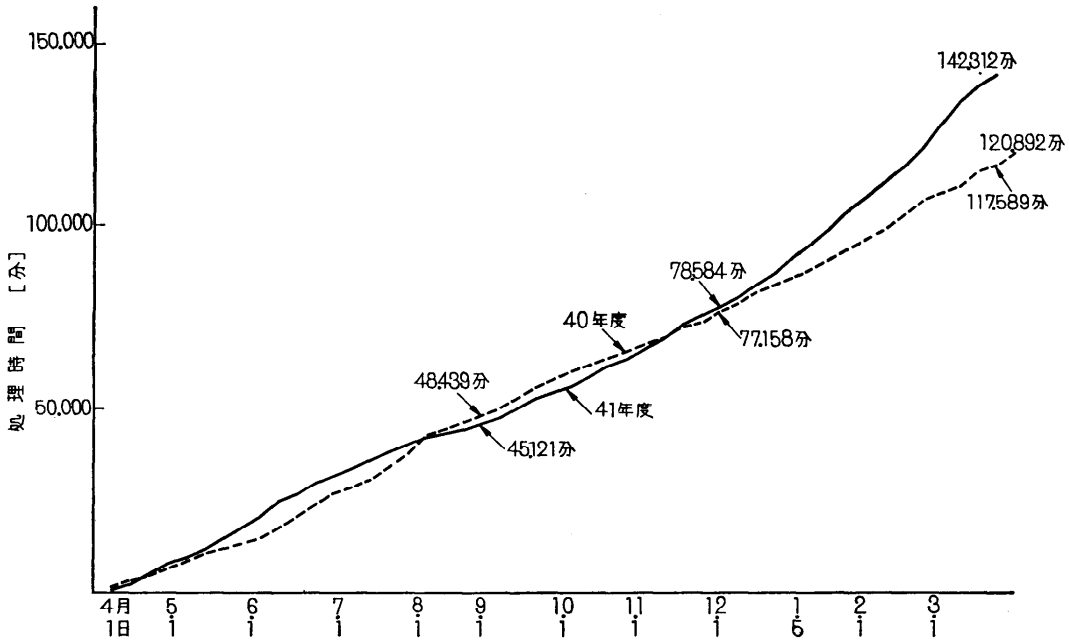
（6） 操作員の勤務時間の関係で，**3.2**節に述べた仕事の流れは，必ずしも第4図のような順序で行なわれるわけではない．すなわち九大計算センターにおいて，1日あたりの計算処理件数は，平常時 70～80 件，ピーク時 150 件程度であるが，これらの job を平常時は，2～3 バッチ，ピーク時は 4～5 バッチに分割して行なう．**3.2**節で4段に分けた仕事のうちもっとも人手を要するのは ① の仕事である．したがって操作員の人数の多い午後5時までに第1バッチの ①，第2バッチの ①，第3バッチの ①，…が行なわれ，操作員の数の減った時間外にそれぞれの ②，③，④ が行なわれることが多い．このような事情からも (1) に述べたように，単一の job に着目するならば，load and go 方式の場合よりもターンアラウンドタイムが長くなることがありうる．なお (2)，(3)，(4) に述べたように，コンパイルのみの job，S^jD^j にエラーのある job，カードの作表，複製などは，②，③，④ が空になるので ① だけの操作で利用者に返却されるので，これらの場合のターンアラウンドタイムはかなり短縮されている．

6. モニターシステムの運用による統計

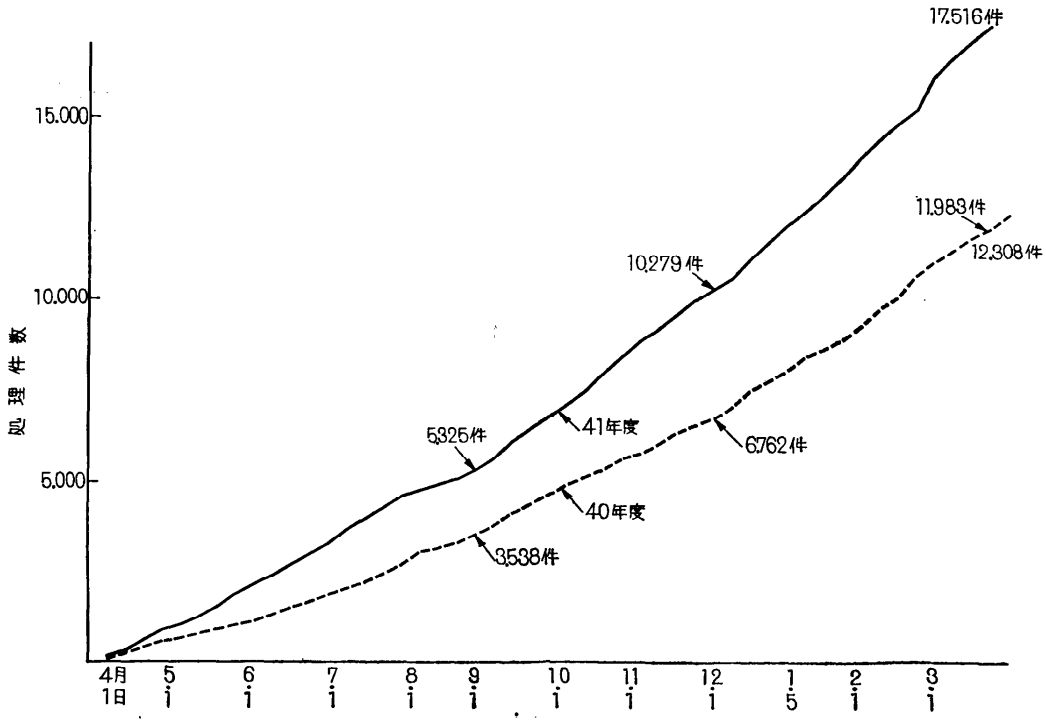
これまでの各節において，九州大学モニターシステムの設計思想と構成を述べたが，以下の各節においては，このモニターシステムを実際に運用した結果得られた統計資料のうち，特に今後の計算機システムの計画，運用などに関係すると思われる結果について報告したい．まず運用による効果は，load and go 方式をとっていた 40 年度（40年4月～41年3月）とモニターシステムによる一括処理方式を採用した 41 年度

第1表 計算時間と処理件数の比較

項目	計算時間〔分〕			処理件数〔件〕			平均処理時間〔分〕	
期日	40年度	41年度	41年度/40年度	40年度	41年度	41年度/40年度	40年度	41年度
9月1日現在	48,439	45,121	0.93	3,538	5,325	1.50	13.69	8.47
12月1日現在	77,158	78,584	1.02	6,762	10,279	1.52	11.41	7.65
3月23日現在	117,589	142,312	1.21	11,983	17,516	1.46	9.81	8.12



第6図　累積処理時間



第7図　累積処理件数

(41 年4月～42年3月) の計算処理時間，処理件数，1件あたりの平均処理時間などを比較してみると明らかになる (41 年度は3月 23 日現在)．**第6図**は有料計算時間を累積したものであり，**第7図**は処理件数を同様に累積した．図中9月1日，12月1日，3月23日現在の数字は**第1表**のようになる．これらによれば，計算時間はほとんど差がないのに (12月1日現在)，処理件数は約 45% という大幅の増加をみている．これはモニターによる一括処理をするようになって，各入出力チャンネルの並行動作をより効率的に使用することが可能になったためのもので，経済的な効果は一目瞭然である．**4** 節に述べたように記憶容量にもう少し余裕があれば，チャンネルによる多重処理がさらに増加して，計算時間はこのままでも処理件数を幾分増加させることができるのではないかと思うが，現状ではこれ以上無理であろう．なお41年度12月以降の計算時間の急激な増加は41年度12月より計算機の夜間運転を経常化した効果である．

さて一般に多重プログラムの実行で，主記憶装置内に複数個のプログラムが共存することになると，各プログラムの相互作用を防ぐ必要がある．またある優先度の高いプログラムによって記憶容量の大部分が占有されると多重プログラムの実行は不可能になる．そこで多重プログラムが実際にどのくらい動きうるかを知りたいという動機から，モニターランで一括処理されている job の占有容量と，その占有時間に関するデータを約7週間分集めてみた．この結果は科学技術計算についてはかなり一般性があると思われる．

3.2 節に述べたように①でコンパイルされたオブジェクトプログラムを，バッチにしてテープにファイルし，②で次々に実行してゆく．その際 job i について，定数の領域 (c_i 語)，プログラムの領域 (p_i 語)，データの領域 (v_i 語)，全領域 (m_i 語)，と計算時間 (t_i 秒) を job の終了ごとに出力させる (ここで $m_i=c_i+p_i+v_i$)．モニターの占有容量は 1K 語なので m_i は 7K 語まで使用可能である．②と並列に動きうる④の列 (1K 語) は②の実行を最優先にしている関係上 $m_i>6$K 語となると動き得なくなる．このような条件のもとで計算実行時間 (t_i) と占有容量の関係 (m_i) および c_i, v_i をしらべてみた．まず相関表を作って各度数を記録してみる．これを累積してまとめると**第2表**が得られる．

この表から計算時間の長いものが増加しても (2) 欄の占有容量 (m_i) の平均はそう増加せず，時間と容量の相関係数は 0.2 程度で入力プログラムの大きさは実行時間のメドになっていないことが明らかになる．ところで m_i は，c_i, p_i, v_i などの条件がまざり合っているので，t_i と c_i, p_i, v_i それぞれについて相関係数を計算したものが**第3表**である．ここでも各容量と時間との相関関係はあまりないといってよい．

第3表 相 関 係 数

	t	m	v	p
m	0.181			
v	0.141	0.900		
p	0.165	0.694	0.310	
c	0.072	0.310	0.267	0.747

さて第2表で多重プログラムの実行可能性を論ずるのは，m_i の平均では十分でなく，m_i 語の job が t_i 秒記憶装置を占有しているので，t_i の重みつき平均 ($\sum m_i t_i / \sum t_i$) をとる必要がある (第2表の第(7)欄)．さらに**第4表**に c, p, v についても同じ観点でまとめてある．結局，常駐モニターの 1K 語を含めて時間の重みつき平均で約 4K 語の記憶容量が利用

第2表 計算時間と占有容量の関係

	(1) 件数 (%)	(2) 占有容量の平均	(3) 占有容量の標準偏差	(4) 総計時間 [秒] (%)	(5) 計算時間の平均	(6) 計算時間標準偏差	(7) 重みつき平均容量	(8) 時間と容量の相関係数
1 分以内	949(54.3)	1,581.3	1,690.6	13,560(2.2)	14.3	15.2	1,948.0	0.204
2 〃	1,105(63.3)	1,723.2	1,771.5	27,490(4.4)	24.9	30.4	2,259.4	0.248
5 〃	1,293(74.1)	1,811.4	1,795.8	66,132(10.6)	51.1	72.3	2,321.8	0.201
10 〃	1,430(81.9)	1,982.3	1,809.6	128,000(20.5)	89.5	139.9	2,423.4	0.191
20 〃	1,618(92.7)	1,947.4	1,788.9	296,620(47.6)	183.3	297.6	2,423.5	0.164
30 〃	1,668(95.5)	1,957.6	1,785.1	375,699(60.3)	225.2	379.5	2,384.4	0.142
40 〃	1,700(97.5)	1,975.4	1,790.7	447,532(71.8)	263.3	466.1	2,465.7	0.155
100 〃	1,744(99.8)	2,020.1	1,827.5	606,464(97.3)	347.7	716.4	2,768.7	0.199
全 体	1,746(100.0)	2,019.3	1,826.6	623,298(100.0)	357.0	766.4	2,729.7	0.181

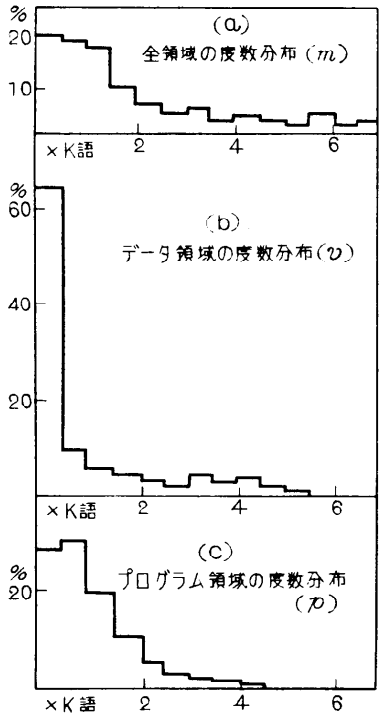
第4表　占有領域の統計

	平　均	標準偏差	最　大	重みつき平均	標準偏差
m	2019.3	1826.6	7154	2729.7	1970.9
v	902.6	1381.7	6337	1319.9	1701.5
p	1059.1	807.7	6860	1345.7	759.8
c	57.6	41.4	297	64.0	37.2

されており，残り 4K 語は空いているので，多重プログラムは記憶容量に関する限り十分実行可能であるといえる．

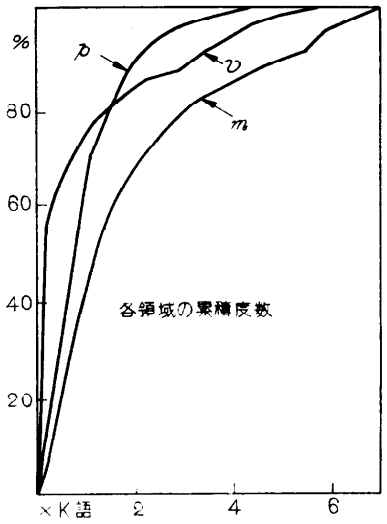
7. 占有容量の語数の分布

今度は計算時間を無視して各 job の占有容量の語数の分布をしらべてみる(**第8図**)．これを累積したも

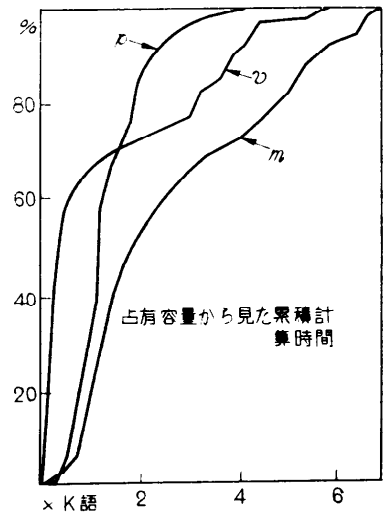


第8図　各領域の度数分布

のが**第9図**である．さらに前節で述べたように m_i が t_i 秒占有されるのであるから，**第10図**に占有容量からみた累積計算時間のグラフを付け加えた．さて記憶装置の保護などの関係から記憶装置の使用法について，READ 可能 (R)，WRITE 可能 (W)，実行可能 (X) のようにあらわすと，われわれの c, p, v は c: R, p: R, X, v: R, W のような性格付けができる．



第9図　各領域の累積度数



第10図　占有容量からみた累積計算時間

以下に統計からみた各領域の性質を考察する．

(1)　常数領域

第4表から最高297語，平均58語であって予想よりも非常に小さい．このデータはコンパイラの設計(定数表の大きさ)などの基礎資料となる．この領域自身が非常に小さく (R) という性格からプログラム領域と一緒にして論じてもさしつかえないと思われる．

(2)　プログラム領域とデータ領域

データ領域の 65% が512語以内におさまるという

ことは驚異である．データ領域は，宣言文によって生成されるもので，array の宣言などで自由に大きくできるが，一方プログラム領域は，たとえ問題向き言語といっても，実行の手順を人間が記述するといった意味で機械語であり，人間の努力には限りがあるので第9図で p 曲線はたち上りは遅いが早く飽和する．そしてプログラムとデータの曲線は1,500語のところで交っている．プログラム用言語がもっと進歩すれば，p 曲線は v 曲線的になるのではないだろうか．次にページメモリーの大きさについて一言するならば，いま1ページを512語にとるとデータ領域は全 job の 65%，プログラム領域は 29%，さらに 1024 語にとると，それぞれ 74%，59% が1ページ内におさまってしまうのである．

（3） 全領域

5K 語～6K 語のところに見られる小さな山は m_i <7K 語の制限によるものと思われる．次にモニターの容量が 1K 語から 2K 語になったとき拒否される job は全体の 4.3%，3K 語では 10.5%である．4K 語でも全体の 84% の計算が処理できる．これを計算時間からみると，モニター 2K 語で 7.8%，3K 語で 19.3% の計算ができなくなる．多重プログラムの実行可能性はモニター 2K 語の場合と同じだから，全体の 7.9%の時間だけ多重に実行することができないことになる．(**3.2** 節④の tape to printer プログラムの CPU 占有時間は数クロックしか必要としない)．われわれがこの計算機にモニターを設計するとき 8K 語は非常に小さいと感じ，モニターを 1K 語以内ときめ 7K 語を利用者にあけわたした．さらに ALGOL コンパイラについてはそのオブジェクトプログラムをできるだけサブルーチン化してコンパクトにすることにつとめた．この結果添字付変数の番地計算の時間が馬鹿にならなくなったが，これをオープンルーチンにすれば p はふえても t は短かくなるはずで，第9図からそれは許されるようである．

8. おわりに

この研究は，東京大学に全国共同利用の大形計算機センターが設置され，電子計算機の全国的な総合使用法を研究するために昭和40年，41年度文部省科学研究費が交付された．これに全国8大学の計算センター関係者が参加し，その総合研究の一環としておこなわれた．この論文ではまずわれわれのモニターシステムの設計思想を述べ，システム構成の記述について一つの方法を示したつもりである．従来ソフトウェアシステムの記述については，言葉による記述に頼るほかなく，何らかのシステム記述の方法の必要性を痛感していた．しかしわれわれの採用した方法は，全く試みにすぎず，われわれのシステムは記述できていても，必ずしも一般性があるわけでなく，さらに一般的なシステム記述の方法について研究を進める必要があると思われる．

次にモニターシステムの運用に関する統計については，従来この種の統計はあまり発表されておらず，この機会に発表することは無意味ではないと考えている．

計算機を利用するものの要求は“より早く，より広く”といわれるが，最後に述べた占有容量の分布をみると，さらに大容量の計算機でも第8図でのすそ野が拡がるだけで，同じ傾向があらわれるものと思われる．むしろ計算機の時分割，多重プログラミングによる利用のため，記憶装置の中にどれだけ複数個のプログラムが入って動きうるかが問題になってくるであろう．

最後にこの統計数字は，バッチを処理するモニター側からみた統計であって，人間対機械の関係からいって，本当に job の結果が人間に満足を与えているかどうかは別の問題であることに注意する必要がある．

参考文献

1) モニターシステムについて，計数施設ニュース，VOL. 3 No. 5, pp. 1～16, (1966.3) 九州大学中央計数施設
2) 高田，牛島，有田：九州大学モニタープログラムについて，昭和41年度情報処理学会予稿集 (1966.12)
3) 牛島，有田：電子計算機の使用時間と記憶容量の利用について，昭和41年度情報処理学会大会予稿集 (1966.12)
4) 牛島，有田：ALGOL の入出力実行命令について，第6回プログラミングシンポジウム報告集 (1965.1)

（昭和42年4月1日受付）