電　子　計　算　機

オリエンテーション

財団法人 日本情報処理開発センター

6807

電　子　計　算　機

オリエンテーション

財団法人 日本情報処理開発センター

目　　次

I　電子計算機のプログラミング

I　電子計算機のプログラミング

ま　え　が　き

電子計算機の威力が世間に喧伝されてすでに久しい。しかしどちらかといえばその表面的効果のみがクローズ・アップされて，前準備にどのような陰の苦労があったかという事が，ついないおざりにされがちである。その一つにプログラミングがある。電子計算機は何ら自発的意志を持たぬ単なる機械にすぎないから，さまざまな名記手順を人間がこと細かに組み立てて，それを計算機に与えてやらねばならない。その手順書がプログラム（Program）であり，プログラムを作ることをプログラミング（Programming）という。

1.　命令とその表現法

夫々の計算機は，いくつづつかの固有の命令というものを持っている。それらの命令の種類や機能は計算機毎に異り，３０前後の命令しか持たぬものもあり，中には３００近い命令を持つものもある。しかし，ごく基本的なものとして表1.1にあげた様なものは，どの計算機も持っている命令だと思ってよいだろう。プログラムとはこれらの命令が順序正しく並べられたものである。

さて，表1.2の２重ワクの中は，４命令からなる，ごく簡単なプログラムである。

表1.1　代表的な命令

命令の種類	機　能　の　説　明
Load	記憶装置から演算装置に移す
Addition	加える
Subtruct	引く
Multiply	かける
Divide	割る
Store	記憶する
Read	読む
Write	書く
Jump	とぶ
Jump Plus (minus)	+(−)ならとぶ
Stop	とまる

表1.2

操作部	番地部	説　　明
３０	１００	(100)→AR
２０	１０１	(AR)+(101)→AR
２１	１０２	(AR)−(102)→AR
１１	２００	(AR)→200

３０　１００は記憶装置１００番地の内容を演算レジスタ（以後略してＡＲという）に持って来るロード（ＬＯＡＤ）命令である。説明文中(n)はｎの内容を意味する。次の２０　１０１はアッド（ＡＤＤ）命令で，その時のＡＲの内容に１０１番地の内容を加える。２１　１０２はサブトラクト（ＳＵＢ）命令で，ＡＲの内容から１０２番地の内容を引く。最後の１１　２００は，ＡＲの計算結果を２００番地にストア（ＳＴＯＲＡ）する命令である。つまりこれは（１００）＋（１０１）−（１０２）の演算を行い，結果を２００番地に入れるプログラムである。このように一つの命令は，３０（ＬＯＡＤ），２０（ＡＤＤ），２１（ＳＵＢ），１１（ＳＴＯＲ）などの様に，その機

能を表わす操作部と，その操作の対象となるものが，どこにあるかという記憶装置の番地を示す番地部との2つの部分から成り立っている。

ここで問題になるのは操作部の表現法である。表1.3に示すように，機種によって命令の表現法がことなり，その上，表現と内容とが全く無関係であるから，憶えにくいばかりでなく，間違いもおこし易いので，最近は第4列目の様に，ある程度その機能を表現できる記号命令を使うのが普通になってきた。また番地部に関しても，１００とか２００とか実際の番地を直接表示せずＤＡＴＡとかＫＥＩＳＵとか記憶場所に，適当な名前をつけて使う方が，プログラムがわかり易く，かつ間違いも少なくなる。そういった表現で表1.2のプログラムを書き直してみると表1.4の様になる。

表1.3　命令のさまざまな表現法

機　　能	機種A	機種B	機種C	記号命令
Load	30	03	500	LOAD
Add	20	02	400	ADD
Aub	21	04	402	SUB
Mult	32	12	200	MULT
Divide	17	16	220	DIV
Store	11	38	601	STOR
Write	61	82	640	WRITE
Read	66	37	540	READ

表1.4

LOAD	A
ADD	B
SUB	C
STOR	RESULT

2. 簡単なプログラム例

「カードにパンチされているデータを一枚づつ読んで，その中の最少のものをプリントせよ。なおデータの中に0はないものとし，データの尽きた印として最後のカードには0がパンチされているものとする」。

まず処理の手順を図（図1.1）で示そう。これをフローチャートと呼び，プログラムの処理手順を表現するには大へん有効な方法である。先ずこのフローチャートを説明しよう。

(1) 最初のデータを読み込んで，とりあえずそれまでの最小値とみなし，MIN(minmum の意）と名付けた場所にしまっておく。

(2) 次のデータを読み込んでＤＡＴＡに入れる。

(3) 今読んだものが0ならもうデータは終ったわけで最後のプリントの仕事にゆく。

(4) 今読んだものと，すでにＭＩＮに入っているものとくらべてみる。ＭＩＮの中身の方が小さければ次のデータを読む(2)へ戻る。

(5) 今読んだものの方が小さければ，ＭＩＮの中身を入れかえる。そして次のデータを読む(2)へ戻る。

(6) ＭＩＮをプリントする。

(7) 停止

このプログラムは表1.5のようになる。

表1.5　例題のプログラム

	READ	MIN	(1) 最初のカード上のデータをＭＩＮに入れる。
L2	READ	DATA	(2) 次のカードを読みデータをＤＡＴＡに入れる。
	LOAD	DATA	(3) データが０ならＬ１へとぶ。０でなければ次の命令へ
	JZ	L1	
	SUB	MIN	(4) ＤＡＴＡ－ＭＩＮの計算を行う。その結果が＋ならＬ２へとび，＋でなければ次の命令へ
	JP	L2	
	LOAD	DATA	(5) ＤＡＴＡの内容をＭＩＮに入れかえる。
	STOR	MIN	
	J	L2	Ｌ２へとべ
L1	WRITE	MIN	(6) ＭＩＮの内容をプリントする。
	HALT		(7) 停　止
	START		このプログラムの最初から実行を開始する。

READ Nはカードから読み込んだデータをNに入れる命令で，最後のWRITE MINはＭＩＮのプリンターにプリントする命令である。

ＪＺはJump Zeroの略でＡＲの内容，ここではＤＡＴＡが０の場合は，番地部で示すＬＩにとび，ゼロでなければ次の命令に移るという条件付飛越命令である。

同じようにＪＰはJump Plusの略でＡＲの内容が＋ならば，つまりこの場合は，DATA－ＭＩＮの計算結果が＋ならば，番地部で示すＬ２へとび，＋でなければ次の命令に移るという命令である。Ｌ１およびＬ２はラベル（Label）と呼び，立て札の役目をする。この例のように，いくつかの命令をとび越えたり，元に戻ったりする場合に必要な立て札である。ＪＰやＪＺの番地部は，今までと異なり，記憶装置の番地ではなく，行く先を示している事に注意しなければならない。ＨＡＬＴは停止命令である。最後のＳＴＡＲＴはこのプログラムの実行を指令する。

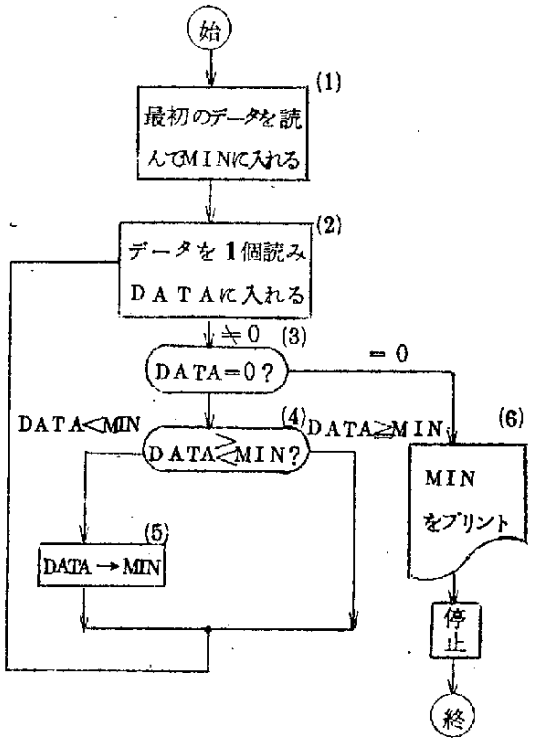


図1.1　例題のフローチャート

一般にプログラムそのものも，いろいろなデータと同じように記憶装置に記憶されるので，この

プログラムがカードやテープにパンチされて，入力装置から読み込まれ，記憶装置にすべて一旦記憶されてからＳＴＡＲＴの指令で実行が開始される事となる。一つの命令を１ステップと呼び，ステップ数によって，プログラムの規模の目安がつく事になっている。

3. プログラミング言語

表1.2のようなプログラムを機械語によるコーディング，またはマシーン・コーディング（machine coding）と言い，表1.4のように記号命令や記号番地を使ったプログラムを記号語によるコーディング，またはシンボリック・コーディング（Symbolic coding）と呼ぶ。この両者を比較すると後者の方がわかり易く，間違いも少ないが，どちらも命令という形に分解しなければならないという点では大差はない。そこで，命令という考え方を無視して自然語に近い表現や，算術式の形のままでプログラムが書ければもっと便利であろうという事になる。例えば，表1.4のプログラムの代りに，

RESULT＝A＋B－C

と書けばよい。また，プログラムは算術式ばかりというわけにはゆかないから，

IF DATA＞MIN THEN GO TO LABEL, ELSE MOVE DATA TO MIN;
LABEL, ……

という様に，日常の英文に近い表現が許される。こういった形でプログラムを作る事を自動コーディング，またはオートマティック・コーディング（Automtic Codig）という。

さて以上の語をまとめてみると，プログラミングには，

マシン　コーディング
シンボリック　コーディング
オートマティック　コーディング

の３通りの言語による方法があることがわかった。

ところが困った事に計算機そのものはやはり機械語しか理解できない。したがって，元はどんな言語で書いてあっても，それが計算機の中で実行される時点では，必ず機械語になおされていなければならない。即ちいつかそれを機械語に翻訳しなければならないのである。そしてその役目を果たすのが翻訳プログラムである。シンボリックコーディングのプログラムを機械語に翻訳するものをアセンブラー（assemblor），オートマティックコーディングのものを機械語に翻訳するプログラムをコンパイラー（Compiler）と呼ぶ。これらの翻訳プログラムを作る仕事というのは，なかなか大変なもので，とくに大規模なコンパイラーになると数万ステップにもなる場合がある。その上，メモリーの容量や，処理速度という点でもそれ相応の要求が出て来ることになる。

4. 国際共通語

自然語に近いプログラミング言語の出現によって，プログラミングの作業が，かなり容易になったと同時にもう一つの大きな利点は国際共通語への道がひらけた事である。

計算機が変わればプログラムも全部変るという事は，大へん厄介なことである。プログラム

を通しての技術の交流という点でも大きなマイナスである。この原因は計算機の機能が夫々異なること，特に命令というものが全くまちまちであるからであるが，それなら世界中の計算機の命令を全部統一してしまえばよいのではないかという意見も出るだろうが，現在まだまだ，あらゆる面で発展途上にある電子計算機に対して命令体系を固定化してしまうということは大きな技術的障害となることは明らかである。そこで内部的ではなく，外部的に，言いかえればハードウェア的にではなく，ソフトウェア的にプログラム言語の方を統一しようという事になった。それも機械語や記号命令のレベルでは無理で，オートマテックコーディングのレベルの言葉を共通なものとし，個々の機械語に翻訳する翻訳プログラム即ちコンパイラーを計算機毎に夫々作ればよいということになった。現在国際的に共通語となっているものがいくつかあって，そのうち技術計算用には，

ALGOL(Algorithmic Language)，FORTRAN(Formula Translator)，事務計算用にはCOBOL(Common Business Oriented Language) があるが，夫々の言語について例題を中心に簡単に説明しておこう。

4.1 ALGOL

この言語はヨーロッパを中心とした学者グループによって提案され１９５８年，スイスのチューリッヒで開かれた国際会議で最初の版がきめられてから，１９６０年に入出力とある一部の細かい点を除いて殆ど仕様がきまったのでＡＬＧＯＬ　６０と呼ばれる様になった。その後さらに１９６５年に入出力仕様も決定された。

ＡＬＧＯＬの目的は勿論この言語で書いたプログラムが，計算機ごとに作られているＡＬＧＯＬコンパイラーによって計算機に直接受けつけられるということもさることながら，そもそもこの言語が作られた動機は，プログラム手順や解析のアルゴリズムをこの言語で公表するためのPublication language としての目的があったとも言われる。その意味からして同じ技術用のＦＯＲＴＲＡＮに比べるとより文章的な部分が多いようである。

以下いくつかの例題を通してＡＬＧＯＬを紹介してゆこう。

「例題１．　１００個のデータを読んで，その中の最小のものをプリントせよ。」

アンダーラインのあるものはＡＬＧＯＬできめられた単語である。；で区切られた一つずつの言葉や算出をステートメント（statement）と呼び，プログラムの一単位とする。endの直前のステートメントには；がいらない。ＡＬＧＯＬでは一区切りのプログラムをブロック（block）と呼び，begin と end でくくる。普通のプログラムはいくつかのブロックからなり立つが，この例題のように簡単なものは１つのブロックでプログラムを完結している。

```
begin real A,MIN;
      integer N;
      inreal (A);
      MIN:=A; N:=1;
   L1:inreal (A);
      if MIN > A then MIN:=A;
      if N=99 then go to PRINT;
      N:=N+1; go to L1;
PRINT: outreal(MIN)
end
```

real A, MIN; integer N; はこのプログラムの中にはA，MIN，N という名の変数を使っていることを宣言しており，ＡＬＧＯＬではプログラムの冒頭に以下で使うすべての変数の名前をこのような形で宣言しなければならない。このうち integer とは0, 1, 2,……等の整数のことで，real とは整数以外の実数で，

例えば1.5，－0.00328，0.476×10^{10} などはすべてrealである。先ず最初に読んだAをMIN(minimum 最小の意)とし，同時に個数Nを1とする。ここまでは準備である。MIN:=Aは A → MINといったような意味で，N：＝1も1→N即ち，Nを1とする意味である。次のL1：はレーベル（label）と呼び，立て札の役目をする。L1のレーベルについたinreal(A) のステートメントからgo to L1; の間が99回繰返される。if MIN>A then MIN:=A; は殆んど文章の通りでMIN>AならAが新しいMIN になるわけであるからMIN:=Aが行われ，そうでない時は次のステートメントへ行く，次のifステートメントはNが99になったか否かをしらべるもので99になればもうデータは尽きたわけであるからgo to PRINT;でPRINT; というレーベルのついたステートメントへゆく。そこでMIN をプリントして終る。Nがまだ99にならなければN：＝N＋1でNを1つ上げgo to L1; でL1に戻る。

「例題2. A，B夫々200個ずつのデータが記憶されている。夫々対応する順序に掛け合わせ，2乗した和を作れ。」

この例は $SUM=\sum_{i=1}^{200}(A_iB_i)^2$ を計算せよということである。

```
begin real SUM;
      real array A, B[1:200];
      integer  I;
      SUM=0;
      for I:=1  step 1 until 200 do
          SUM:=SUM+(A[I]×B[I])↑2
end
```

real array A, B[1:200]; という宣言はAとBが共にrealで夫々1から200まで200個のグループになっていることを示す。仮にAが10ケ，Bが50ケならreal array A[1:10], B[1:50]; と別々に書けばよいし，マトリックスのように2次元のものは例えば30×40ならA[1:30, 1:40]又，3次元で5×10×20ならA[1:5, 1:10, 1:20] のように表現すればよい。また添字の下限は必ずしも1でなくてもA[−10:10] という表現もゆるされる。for I:=1 step 1 until 200 dもほぼ文章通りの意味で，Iが1から200まで1おきに以下のステートメント，ここではSUM:=SUM+(A[I]×B[I]↑2を実行せよということである。( )↑2は($)^2$ のことである。

「例題3. 10行7列の行列Aと7行5列の行列Bの積C＝ABをつくれ。Cは10行5列となり，各要素は $c_{ik}=\sum_{j=1}^{7}a_{ij}b_{jk}$ $\begin{matrix}1\leqq j\leqq 10\\1\leqq k\leqq 5\end{matrix}$ となる。」

```
begin real array A[1:10, 1:7], B[1:7, 1:5], C[1:10, 1:5],
      integer I, J, K;
      for I:=1 step 1 until 10 do
          for k:=1 step 1 until 5 do
      begin C[I, K]:=0;
          for J:=1 step 1 until 7 do
          C[I, K]:=C[I, K]+A[I, J]×B[J, K]
      end
end
```

forのループ内に２つ以上のステートメントが入る場合はこの例のようにそれをbegin …… end でくるむ。又forは何重かに重ねて使うこともできる。

「例題４　２次方程式 $ax^2+2bx+c=0$ の２根を計算せよ。」

```
begin real A, B, C, D, E, F, X1, X2;
      D:=B↑2−A×C;
      if D≧0 then
      begin E:=−B/A; F:=sqrt(D)/A;
            X1:=E+F; X2:=E−F;
      end else
      begin X1:=−B/A, X2:=sqrt(−D)/A
      end
end
```

複素根の場合はＸ１が実数部，Ｘ２が虚数部を表わす。if ステートメントは，

if 条件式 then begin条件が満足したときのステートメント群 end else begin 条件が不満足のときのステートメント群 end;

という形をとり，条件のいかんにかかわらず次は；以下へゆく。then 以下又はelse 以下のステートメントが１個のみのときはbegin，end はいらない。sqrt(D) はＤの平方根をとるということで，関数$\sqrt{x}$，sin$x$，cos$x$，ln$x$，arctan$x$，exp$x$ 等は，そのまま書けばよいことになっている。

この計算をサブルーチンとみなして係数ａ，ｂ，ｃを１組ずつ読み込んで２次方程式を解き，結果をプリントする仕事を１０組引き続いてやるプログラムは次のようにすることができる。

```
begin real a, b, c, d, x1, x2;
Procedure QUAD(A, B, C, D, X1, X2);
```

```
        value A, B, C
        real  A, B, C, D, X1, X2;
        begin  real E, F;
          D:=B↑2−A×C;
        if D≥0  then
          begin E:=−B/A; F:=sqrt(D)/A;
                X1:=E+F; X2:=E−F
                end      else
          begin X1:=−B/A; X2:=sqrt(−D)/A
          end
        end;
          for I:=1 step 1 until 10 d
          begin inreal(a, b, c);
          QUAD(a, b, c, d, x1, x2);
          if α ≥0 then outstring("X1□□□□□X2")
                  else outstring("Real□□□□□Image");
          outreal (x1, x2)
        end
end
```

サブルーチンは，

```
procedure subroutine name(parameter list);
        parameter の type declaration
 begin
        procedure body
 end
```

という形になる。parameter を必要としない場合はその type declaration も必要ない。メインプログラムでの計算結果が値で渡される変数名を value で指定する。又この例のE，Fのようにその procedure の中でのみ必要な変数は procedure body の先頭で宣言する。メインプログラムの中にある outstring で"　　"内は character string であり，"　　"内の文字がそのままプリントされるものとする。□はスペースである。

4.2 FORTRAN

FORTRANとはFormula translatorの略といわれ，1957年IBM 704のために開発されたプログラム言語であるが，その後IBMの他の機種（7090，7040等）にも採用され，IBM系以外の計算機にも広く利用されるようになった。

ここではＡＬＧＯＬとの比較の意味も含めて同じ例題で説明しよう。

「例題１.　１００個のデータを読んで，その中の最小のものをプリントせよ。

```
      READ(u, 20)A
      XMIN=A
          N = 1
   10 READ(u, 20)A
      IF(XMIN.GT.A)XMIN=A
      IF(N.GE.99)GOTO11
      N=N+1
      GO  TO  10
   11 WRITE(u, 30)XMIN
      STOP
   20 FORMAT (F10.3)
   30 FORMAT (F15.3)
      END
```

ＦＯＲＴＲＡＮでは単一変数(single variable)については原則として宣言を必要としない。変数名がＩ，Ｊ，Ｋ，Ｌ，Ｍ，Ｎのいずれかから始まるものはinteger，他はreal ということになっている。ただし特にＲＥＡＬとかINTEGERか，type を宣言すれば頭文字にこだわらずに使うこともできる。この例では特に宣言をしていないのでＡ，ＸＭＩＮは共にreal，Ｎはintegerである。

ＩＦステートメントは，

ＩＦ（論理式）ステートメント１

という形をとり，（　）内の論理式が成り立てば，右のステートメント１を実行し，成り立たなければ，ステートメント１を実行せずに次へ行く。

XMIN.GT.A は XMIN ＞ Ａの意味であり，N.GE.99 は Ｎ ≧ 99 の意味である。

左側の１０，１１等の数字はステートメント番号(statement number) と呼び，これはレーベルで，普通５桁以内の整数である。元来ＦＯＲＴＲＡＮのプログラムはカードから読むのが普通で，ステートメントのカード上のコラム（column）の指定は図4.1のようにきめられている。

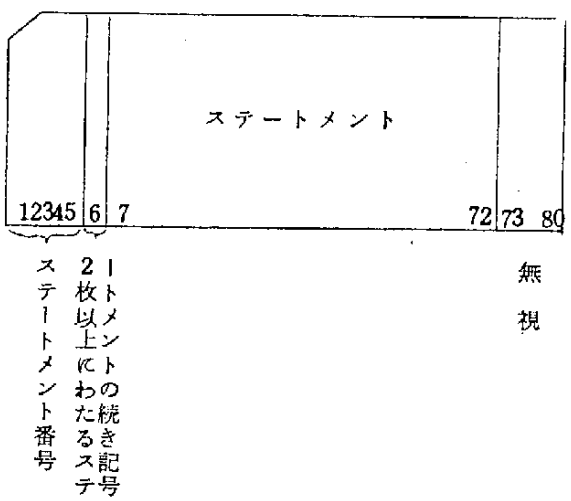


図4.1　ＦＯＲＴＲＡＮのカードフォーマット

FORTRANの入出力は一般的に

READ(u, n)入力リスト, WRITE(u, n)出力リスト

という形をとり, uは入出力の機器の指定, nはその入出力データのフォーマット(FORMAT)指定のステートメント番号である。

ここでは仮に, 入力はカード出力はプリンターとすると, 先ず入力は,

```
    READ(u, 20)A
      ⋮
20  FORMAT(F10.3)
```

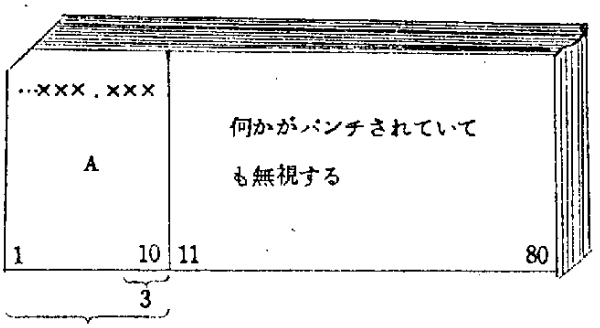


図 4.2　例題1のデータカード

という形で, これは図4.2に示すように, Aという変数がカードの1colから10colまでに実数値としてパンチされ, 小数以下は3桁であることを示している。

次に出力のWRITEステートメントのフォーマット, FORMAT(F15.3)はプリンター上に(15桁)と実数で小数点以下3桁の形でプリントされる。整数部はこの範囲内なら何桁になってもよい。符号もその桁数の中に入る。FORMATの指定はFORTRANプログラムの中でもかなり厄介なところで, 文字, 数字(real, integerの区別もある)ブランク, レコードの区切り(1レコードはカードなら1枚, プリンターなら1行, 磁気テープなら1回でread, writeする単位)等の指定が行われる。ENDは1つのプログラムの区切りに必ずつける。

「例題2.　A, B夫々200個ずつのデータが記憶されているときSUM= $\sum_{i=1}^{200}(A_i B_i)^2$ を計算せよ。」

```
    DIMENSION A(200), B(200)
      SUM = 0.0
    DO 100 I = 1,200
100 SUM = SUM+(A(I)*B(I))**2
    END
```

FORTRANはarrayに限り宣言が必要である。DIMENSIONステートメントをそれに使う。この例ではA, Bとも2次元の200個ずつのarrayであることを宣言している。Cが5×10の2次元ならC(5, 10)というような形で書く。FORTRANは原則として3次元までゆるされる。DOステートメントは一般に左のように書き, nはステートメント番号で, IがLからMまでNおきにnまでの間のステートメント群を反復実行せよということになる。Nが1ならこの例のよ

```
DO  n  I=L, M, N
       ⋮
n  ————————   } DO の range
```

うに省略してもよい。D＊＊2はD^{2}のことで，一般にD＊＊EはD^{E}を示す。

「例題3.　10行7列の行列Aと7行5列の行列Bの積Cを作れ，Cの各要素は，

$c_{ik} = \sum_{j=1}^{7} a_{ij} b_{jk}$　　$1 \leq j \leq 10$　　$1 \leq k \leq 5$　　でCは10行5列となる。

```
      DIMENSION A(10, 7), B(7, 5), C(10, 5)
        DO    500    I = 1, 10
        DO    500    K = 1, 5
         C(I, K)=0.0
        DO    500    J = 1, 7
500     C(I, K)=C(I, K)+A(I, J)*B(J, K)
        END
```

「例題4.　10組の2次方程式$ax^2 + 2bx + c = 0$　の係数を1組ずつ読んで2根を計算しプリントせよ。」

サブプログラム

```
SUBROUTINE QUAD(AA,BB,CC,DD,XX1,XX2)
    DD=BB**2-AA*CC
    IF(DD.LT.0.0) GO TO 10
20  E=-BB/AA
    F=SQRT(DD)/AA
    XX1=E+F
    XX2=E-F
    RETURN
10  XX1=-BB/AA
    XX2=SQRT(-DD)/AA
    RETURN
    END
```

メインプログラム

```
    DO 1 I=1, 10
    READ(u, 100)A, B, C
    CALL QUAD(A, B, C, D, X1, X2)
    IF(D.LT.0.0) GO TO 2
    WRITE(u, 200) X1, X2
    GO TO 1
  2 WRITE(u, 300) X1, X2
  1 CONTINUE
    STOP
100 FORMAT(3F10.5)
200 FORMAT(4HX1 □□, F10.3, 7H□□□
    X2 □□, F10.3)
300 FORMAT(6HREAL □□, F10.3, 10H□
    □□ IMAGE □□, F10.3)
    END
```

FORTRANではブルーチンの働きをSUBROUTINEサブプログラムと，FUNCTIONサブプログラムの2つの形で使いわけ，前者は多価の結果を得るもの，後者は1価のものとする。サブプログラムもメインと同様，ENDで区切った1つのまとまったプログラムの形をとる。メインからサブプログラムを呼ぶにはSUBROUTIN 型ならこの例のようにCALL

name(parameter list) で呼び，FUNOTION 型なら算術式の右辺に左辺＝……
fanotion name(parameter list)…… の形で書き込んでしまってよい。又サブプログラムからメインに帰るにはいずれもRETURNステートメントを使う。D.LT.0.0 はD＜0.0の事である。またDO の rangeの最後にCONTINUE ステートメントをつける事もある。FORMATの中で $\overbrace{nH\times\times\times\cdots\cdots\times\times}^{n\text{ケ}}$ とあるのはH以後のnケの文字がそのままプリントされる。

4.3 COBOL

事務計算と技術計算の大きな違いの一つは事務計算の入出力データ構成の複雑さである。ALGOLやFORTRANでは不充分なこれらの機能をCOBOLではかなり重視してとりあつかっている。1959年アメリカの国防省が中心となり，多くの計算機メーカが加わってCOBOL言語を作る作業が開始された。そして1960年にCOBOL 60，同61年にCOBOL 61，1963年初頭にCOBOL 61 EXTENDED が発表され現在に至っている。

COBOLのプログラムは次の4つの部門にわけて書く。

○ IDENTIFICATION DIVISION

プログラム名，プログラマー名，日付等プログラムの見出しを書く。

○ ENVIRONMENT DIVISION

使用する機械の機能，即ちメモリーサイズ，入出力装置の種類，台数や夫々に適当な名前をつけたりする。

○ DATA DIVISION

そのプログラムにあらわれるすべてのデータの性質，即ちファイルやレコードの構成，次元の指定，数値の型，桁数等を指定する。

○ PROCEDURE DIVISION

せまい意味のプログラムで，演算手続きを書く。

COBOLの特長は次のような点にある。

(1) ハードウェアとの関連を明記してあるので，新らしい計算機におきかえる時に，プログラムの変更の労力はENVIRONMENT DIVISIONのみに集中できるといわれる。

(2) プログラムがそのまま文書として利用できる。

(3) 事務計算特有の入出力の複雑さを充分考慮に入れてある。

(4) 反面，要素が多すぎてプログラムが冗長になりやすい。

すべての機能を含むというわけにはいかないが，簡単な例題を1つあげておこう。

「例題　図4.3の給与計算の入力データをもとに，失業保険，差引給与額を計算して出力テープを作れ。」

入力データ

給与入力									
表示項目				支給項目			控除項目		
所属	番号	氏名	職位	基本給	特別手当		消費組合	雑控除計	
(5)	(5)	(21)	(3)	(6)	(3)	(6)	(6)	(6)	(06)

出力データ

給与出力												
表示項目				支給項目			控除項目					差引支給額
所属	番号	氏名	職位	基本給	特別手当	支給計	消費組合	雑控外計	失業保険料	控除計		
(5)	(5)	(21)	(3)	(6)	(6)	(6)	(6)	(6)	(4)	(6)		(6)

図 4.3　給与計算のデータの例

この手順を流れ図に書くと図 4.4 のようになる。

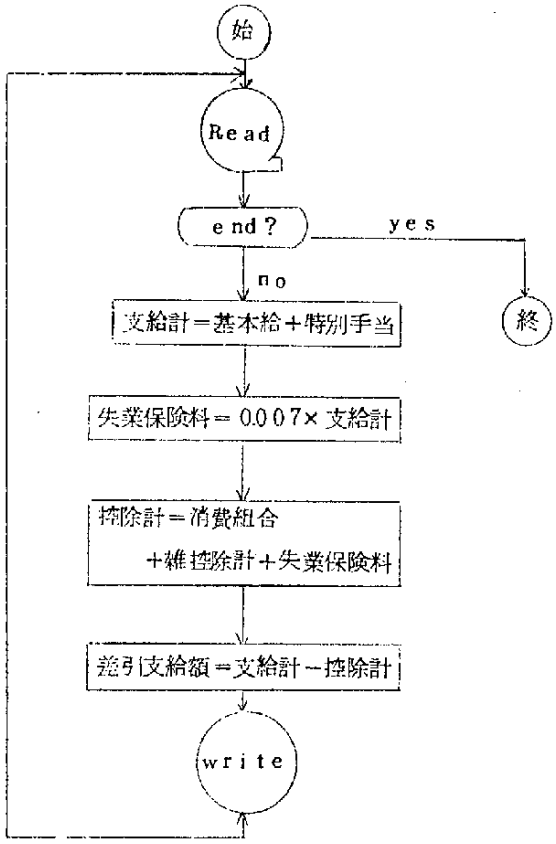


図 4.4　COBOL の例題の流れ図

これをＣＯＢＯＬで書くと次のようになる。

```
IDENTIFICATION  DIVISION.

PROGRAM-ID.          KYUYO-KEISAN.
AUTHOR.          HASIMOTO-N.
DATE-WRITTEN.          1963-9-14
REMARKS.          "REIDAI".
```

```
ENVIRONMENT  DIVISION.

CONFIGURATION SECTION.
SOURCE-COMPUTER.    JIPDAC.
OBJECT-COMPUTER.    JIPDAC.
INPUT-OUTPUT SECTION.
FLLE-CONTROL.
SELECT KYUYO-INPUT-FILE ASSIGN TO TAPE.
SELECT KYUYO-OUTPUT-FILE ASSIGN TO TAPE.

DATA  DIVISION.

FILE  SECTION.
FD KYUYO-INPUT-FILE  BLOCK CONTAINS 10 RECORDS.
   LABEL RECORD IS STANDARD
   DATA RECORD IS KYUYO-INPUT.
   01  KYUYO-INPUT.
       02  HYOZI-KOMOKU.
           03  SYOZOKU  PICTURE IS 9(5).
           03  BANGO    PICTURE IS 9(5).
           03  SIMEI    PICTURE IS A(21).
           03  SYOKUI   PICTURE IS 9(3).
       02  SIKYU-KOMOKU.
           03  KIHON-KYU          PICTURE IS S9(6)V.
           03  TOKUBETU-TEATE     PICTURE IS S9(6)V.
           03  SIKYU-KEI          PICTURE IS S9(6)V.
       02  KOZYO-KOMOKU.
           03  SYOHI-KUMIAI          PICTURE IS S9(6)V.
           03  ZATU-KOZYO-KEI        PICTURE IS S9(6)V.
           03  SITUGYO-HOKEN-RYO     PICTURE IS S9(4)V.
           03  KOZYO-KEI             PICTURE IS S9(6)V.
       02  SASIHIKI-SIKUY  PICTURE IS S9(6)V.
FD KYUYO-OUTPUT-FILE  BLOCK CONTAINS 10 RECORDS
   LABEL RECORD IS STANDARD
   DATA RECORD IS KYUYO-OUTPUT.
   01  KYUYO-OUTPUT       PICTURE IS X(80).
```

```
CONSTANT  SECTION.
77  KEISU PICTURE IS SV999     VALUE IS  007.

PROCEDURE  DIVISION.

HAZIME.      OPEN INPUT KYUYO-INPUT-FILE
    OUTPUT KYUYO-OUTPUT-FILE.
YOMU.  READ KYUYO-INPUT-FILE  AT AND GO TO OWARI.
    COMPUTE   SIKYU-KEI = KIHON-KYU + TOKUBETU-TEATE.
    COMPUTE   SITUGYO-HOKEN-RYO = KEISU * SIKYU-KEI.
    COMPUTE   KOZYO-KEI=SYOHI-KUMIAI+ZATU-KOZYO-KEI+SITUGYO-HOKEN-RYO.
    COMPUTE   SASIHIKI-SIKYU = SIKYU-KEI - KOZYO-KEI.
    WRITE KYUYO-OUTPUT       FROM KYUYO-INPUT  GO TO YOMU.
OWARI.    CLOSE KYUYO-INPUT-FILE KYUYO-OUTPUT-FILE.
    STOP RUN.
```

アンダーラインのあるものはCOBOL固有の単語である。ENVIRONMENT DIVISIONでわかるように(Compile)する機械(SOURCE-COMPUTER)とオブジェクトプログラムの入る機械(OBJECT-COMPUTER)とが違ってもよい。

事務計算で扱われるデータの構造は，ファイル(file)，レコード(record)，項目(item)という順に細分類され，更に項目には集団項目(group item)とか基本項目(elementary item)とかいくつかのレベルがあり得る。COBOLのDATA DIVISIONでは，このデータの構造を最高49レベルまで使用することができる。この例ではファイルとして

FD	ファイル
01	レコード
02	集団項目
〃	⋮
49	基本項目

FD KYUYO-INPUT-FILEとFD KYUYO-OUTPUT-FILE，レコードとして01 KUYO-INPUTと01 KUYO-OUT-PUT，集団項目として02 HYOZI-KOMOKU など，基本項目として03 SYOZOKUなどの3つのレベルにわけてある。常に最低のレベルが基本項目となり，これにはPICTUREとしてデータの形を表示する必要がある。PICTUREの意味は9(5)は5桁の数字で99999と同じである。A(21)は21字の文字，S9(6)Vは符号のついた6桁の数字で小数点が最後につく，同様にS9V999は小数点が上から2桁目にある4桁の符号付数字である。

この例では入力データの一部に結果を書き込んで出力データとする形をとっているので，ファイルの構造は入出力ともに同じであり，入力ファイルについてのみ明細を表示してある。こういう場合はPROCEDURE DIVISIONの中で演算結果をWRITE KYUYO-OUTPUT FROM KYUYO-INPUTという形で出力テープに書きなおす。

5. プログラミングの範囲と手順

プログラミングという仕事の範囲を厳密に規定することは，なかなかむずかしいが，通常次の4つの段階に分けて考えている。

5.1 問題の解析

計算機の持っている機能は，加減乗除とか簡単な比較，判別等のごく基本的なものだけであるからいかにむずかしい問題も究極的には，これ等の基本的機能に分解されねばならない。

"どこそこの会社では事務のシステムをすべて計算機で処理している"というようなことがいわれるが，実はその複雑な帳簿の整理や，伝票の流れをすべて計算機の能力に合わせて編成し直し，結局は四則演算とか比較，分類等に分解しつくして，それ等の複合体としての作業を流しているのである。また，仮に選挙の予想のような問題をとりあげるとしても，どのようなデータと解析式が必要であるかという検討をまず行なわなければならないし，また物理的工等的な現象が，微分方程式や，積分形で表わされたとしても，そのままの形では計算機にかけられないから，それ等の式を四則演算に分解する数値解析(numerical analysis)の仕事が必要となる。

また，すべての情報が計算機の受け付け得る数字，文字または特殊記号で表現されねばならぬということで，情報の記号化，計数化という問題も重要である。このように解析(analysis)の作業はその対象によって多少異なるが，大きく分類すると次のような要素に分けられるだろう。

(1) あらゆる要素を記号化，数量化する。

(2) 問題を規定している規則性や関係式を導き出す。

(3) 方程式や関係式を四則演算に分解する。

(4) 必要なデータの収集

5.2 流れ図の作成

仕事の手順を図的に表現するということは，問題の流れを手取り早くつかむ上にも，また間違いの発生を防ぎかつ訂正をたやすくする上からも是非必要なことである。普通，大まかなブロックダイアグラム(block diagram)と，その細部に説明する意味のくわしいフローチャート(flow chart)と二段構えに作ることが望ましいとされている。また図5.1に示すように作業と図形とを対象させておくとなおよいだろう。

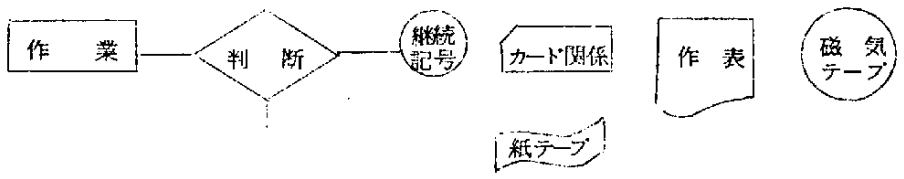


図5.1　フローチャート表現の一例

5. 3　コーディング

フローチャートで表わされた手順を直接計算機に読ませる言葉に翻訳するのがコーディング（coding）である。この仕事は計算機ごとにすべて異なるし，また機械コーディング（machine coding），記号コーディング（symbolic coding），自動コーディング（automatic coding）のいずれを選ぶかということによって異なる。この作業は，とくに不注意によるミスの介入しやすいところで，例えば，機械コーディングの際に０２と２０を取り違えても大きな間違いとなるし，記号の場合でも命令の区切りを示す『，』が１つ抜けていても機械は停ってしまうか，またはそのまま間違った操作をして以後に影響を与える。あくまでも相手は機械であるということを忘れてはならない。

解析→流れ図→コーディングという実際の手順とは逆にプログラミングをマスターするにはまずコーディングの勉強から始めるのが最も能率がよい。

5. 4　ディバッキング

一通り出来上ったプログラムを間違いがないかどうかを調べる仕事をディバッキング（debugging）という。主として計算機を使って間違いをみつけるが，その方法が上手か下手かがプログラマーの熟練の一つのバロメーターともなっている。紙の上では充分検討を加え，もう間違いはないとかなり自信のある場合でも計算機で実行させてみるとなおかなりの間違いが発見できるというのがプログラミングの宿命のようなものであって，このディバッキングの仕事に要する時間や労力は決して軽く考えてはいけない。

6. ソフトウェアとは

電子計算機の能力は（ハードウェア）＋（ソフトウェア）であるといわれる。ハードウェア（Hardware）という言葉は昔からあった。これは計算機そのもので日本語では「金物」といわれる。これに対してソフトウェア（Software）という言葉は電子計算機固有のもので「利用技術」などと訳すことがある。広い意味では，何もかも一切のプログラムを含めてソフトウェアという場合もあるが，一般には，ある特定の一ユーザのみに必要なプログラムではなくて，少くとも幾つかのユーザに共通して使われるプログラムを総称する場合が多い。

そういった意味から一通りのソフトウェアは計算機メーカ側で整備して，ハードウェアとともにユーザに提供し，ユーザはハード，ソフト両者を含めたものがその計算機の能力であるという感覚で使用するというのが普通である。

さて，このソフトウェアの種類や機能は，計算機の規模，使用目的などにより違いもあり，また時代と共にその内容や分類も変化してゆくものであるから一概には言えないが，さしあたり表6.1のように分類してみる事が出来る。

表6.1　ソフトウェア一覧表

(1)　技術計算用ライブラリ
1.　線型計算
連立一次方程式，逆行列，行列演算，固有値

2. 代数方程式
3. 数値積分，数値微分
4. 補間法
5. 微分方程式，積分方程式
6. 関　数

   $\sin x$, $\cos x$, e^x, $\log x$, $\tan^{-1} x$, Bessel 等
7. 関数近似

(2) オペレーティング・システム

1. 制御プログラム
2. 言語処理プロセッサー

   各種コンパイラ，アセンブラ，ゼネレータ，インタプリタ
3. サービス・プログラム

   入出力処理プログラム

   ソーティング・プログラム

   ファイル処理プログラム

   ライブラリー編集プログラム

   ディバッグ　プログラム

(3) アプリケーションプログラム

1. オペレーションズ　リサーチ用プログラム
2. シミュレーション用プログラム
3. 統計計算用プログラム
4. 経営管理用プログラム

   日程計画，生産管理，在庫管理，販売管理，需要予測等
5. 工学用プログラム

   土木建築，原子力，機械工学，化学工業，電力，造船等
6. 情報検索用プログラム

6.1　技術計算用ライブラリ

技術計算で日常さかんに使われるようなプログラムは，

個々の使用者が，その都度作製するという重複をさけるため，共通性のあるものを予め一通り誰かが作っておいて，他の人はそれを信用して利用しようというのが目的である。この性質からも当然，プログラムとしてのすぐれた特長を幾つか兼ねて備えていなければならない。例えば，

(1) ステップ数をできるだけ少なくする工夫がしてあること。

(2) 演算時間が早いこと。

(3) 計算誤差が少なく数値解析的に吟味されること。

(4) 一般性があること。（例えば連立方程式なら何元でも解けるように）

(5) 使い易いこと。（特にパラメータの種類や書き方があまり複雑でないこと）

(6) 使用者の誤りが指適できること。(データ数の不足やパラメータの書き方の不備など)

以上の(1)～(6)はお互いに矛盾する条件もあり，なかなかむずかしい注文であるが，これらは必ずしもライブラリープログラム(Library program)に限ったわけではなく，何回も繰返し使用するもの，あるいは多くの人が共通に利用するプログラムは以上のような点の吟味が充分されていることがのぞましい。

ライブラリープログラムは普通，磁気テープにまとめて保管されるが，同時に次のような内容をもった詳細な説明書をつけておく必要がある。

題名，整理番号，目的，語数，精度，時間，使用上の注意(丁寧に)，解析法，流れ図，コーディングシート，作成者名，年月日等

6.2 オペレーティング システム

最近の計算機は高速，大容量，多岐の入出力装置，多重処理など，ハードウェアの機能がますます増大しつつある。このような機能の向上はまことに結構なことではあるが一方これを使う立場からいうと，これらの機能を充二分に無駄なく発揮させるにはどうしたらよいかと苦労することになる。

また計算機が高速になってくると1日にかける仕事の種類も相当増えてくる。1つの仕事を計算機にかけるにはプログラムを読ませたり，磁気テープを取り変えたり，指定通りスイッチをセットしたりするいろいろな作業が付随することが常である。1日にかける仕事の量が増えるということは，それだけ手作業の仕事も増えるということで例えば1日8時間稼動時間があったとしても，そのうち3時間はオペレータの手作業の時間であるというようなことにもなりかねない。これでは甚だ能率が悪いので一日又は半日分の仕事をあらかじめあたかも一つの仕事のように編集してしまい，最初一度オペレータがスタートさせるとあとは次から次と自動的に仕事を処理して途中に入力を介入させないですませるような自動運転が必要になる。

このようにハードウェアの機能を助けて，それをより有効に生かし，機械の稼動効率をあげるためのソフトウェアをオペレーティングシステム(operating system略してOS)と呼んでいる。

オペレーティング システムの役目は次のようなものがある。

1. 異なった言語によるプログラムの連続処理

   OSはいろいろなアセンブラ，コンパイラ等を全部支配下においているので，どんな言語で書かれたプログラムがきても，その都度必要な翻訳プログラムを自由に呼び出してきて使うことができる。

2. プログラム エラーの検出とディバック機能

   人力の介入を許さないということは，プログラマーが自分で計算機を操作しながらディバッキングするという事も不可能になることである。そのためにはプログラムの誤りを検出し，どんな誤りかという情報を出し，プログラマーに訂正の手がかりをあたえる手段が講じてなければならない。また，自由に呼び出せる強力なディバックプログラムが内蔵されている必要もある。

3. ライブラリの管理

ライブラリの追加，変更にも応じられる様に管理を行い，いろいろなジョブが要求するライブラリをとり出してきて，うまく連結して使うことができる様にする。

4. 時間の管理

内蔵の時計で時間の管理を行い，時間によって仕事を中断したり，切りかえたりする。

5. 入出力制御

チャンネルやユニットの管理，割りあて，入出力割込みの処理等を行なう。

6. 多重処理スケジューリング

仕事の優先順位にしたがって処理順序のスケジューリングを行ない，なるべく効率のよい多重処理を実行する。

7. メモリーの割りあて

各ジョブにメモリーを割りあてる。とくに複数個のジョブが同時に働く様なシステムでは，外部記憶も含めたスワッピング（Swapping）の管理も必要となる。

8. セグメンテーション

一つのジョブが，いくつかのセグメント（segment）に分かれている場合は，セグメント間のコントロールの受け渡しが必要となる。

9. ファイルの管理

プログラムファイル，データファイルを含め，ファイルの作成，更新，保管等一切の管理を行なう。

10. エラーの処理

できるだけ計算機を停めないという方針で，予期しないハードウェアのエラーに対しても，できる限り自己回復の処置が必要である。

オペレーティング　システムの管理下で仕事を行う場合は，プログラム本体の他に，ＯＳに対するいくつかの指定が必要になる。例えば使用する言語の種類，ライブラリーの名前，翻訳のみか，演算も行うか，演算続行希望時間，使用する磁気テープの本数，夫々の仕事の始まりや終了の表示等，いくつかの項目を，定められた規則にしたがって指定しなければならない。これらをコントロール　パラメータと呼ぶ。

6.3 アプリケーションプログラム

計算機のアプリケーションの範囲は無限に広い。したがってアプリケーションプログラム（application program）の数にも限りがないわけであるが，とくに，これを時代と共にその内容や種類の変化が甚だしい。したがってアプリケーションプログラムとは，これこれあると限定するのがむずかしいが，現在比較的広く使われている分野を表6.1にあげておいた。特に工学分野では専門家のグループによる共通に使用できる言語やプログラムの開発がさかんである。今後はどの分野においても，アプリケーションプログラムの種類も質も急速に向上してゆくことが期待される。

Ⅱ　電子計算機のハードウェア

Ⅱ　電子算機のハードウェア

1.　電子計算機の概路

計算機は取扱われる情報の形で分類すると次の2種類になる。

計数型計算機（Digital Computer）

相似型計算機（Analogue Computer）

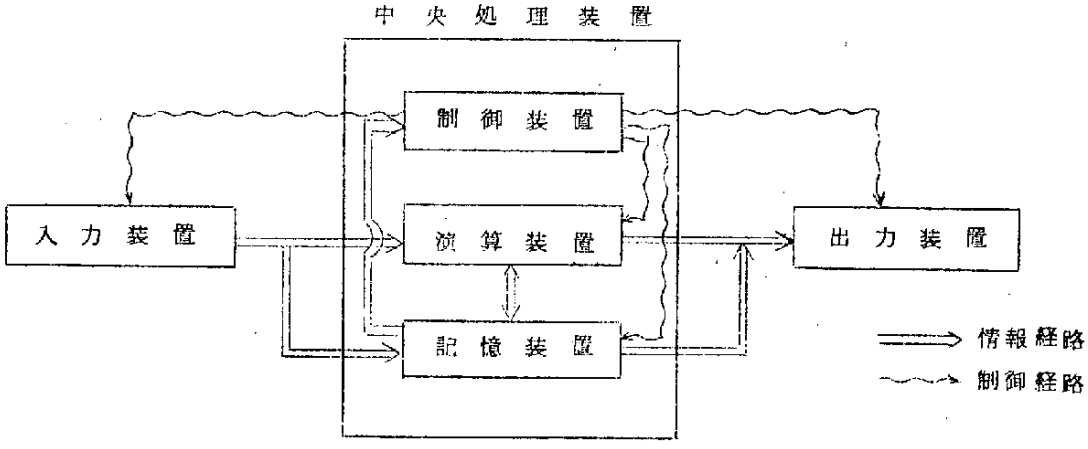
前者に於ては物体の高さを表現するときは数字を用いて2 4.5 6とか，ソロバンの玉等で表現し，敢者に於てはその物体の高さと等長又は相似な棒，線，ひも等で表現する。即ち前者は数で情報を表現し，後者は物理量で表現する。以上のように表現された情報を電子の作用を使って取扱い，処理するのが電子計算機である。

電子計算機ではないが物理量を使って計算するものに身近なものとして計算尺があり，数を使って計算するものにソロバン，卓上計算機，電動計算機がある。

ここでは計数型電子計算機について述べるが以後電子計算機又は計算機と呼ぶことにする。

1.1　電子計算機の構成とその機能

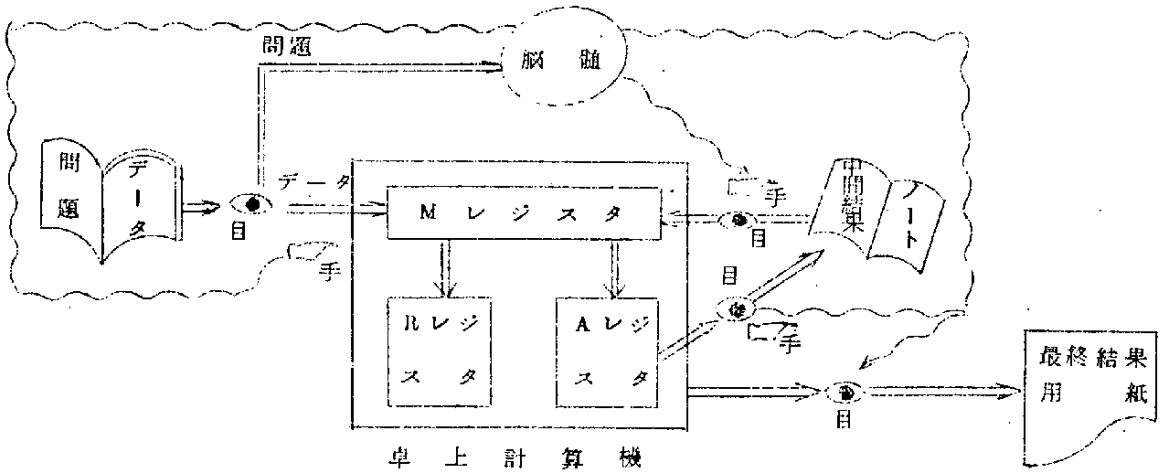
（a）構　成



（b）機　能

機能については卓上計算器を使用して人間が計算する場合を図示して上の構成図と対応させて説明する。

人間が目を通して問題を読み取る。このとき目が構成図の入力装置に相当するわけである。この読み取った内容によって計算法を考えて手順を組立て，それに従って置数レジスタ（Mレジスタ）に数を目で読み取って手でセットし，これを結果レジスタ（アキュムレータ又はAレジスタ）に移し計算する。



このMレジスタからAレジスタに移す回数や桁移動は回転数レジスタ（Rレジスタ）に出てくるし，加減算ならそのたびにMレジスタの数を新しい数にセットしなおして計算する。

このMレジスタ，Aレジスタ，Rレジスタは演算装置に相当する。一連の計算を完了するとその結果を中間結果ノートに移して計算器では別の項目の計算をする。このときの中間結果ノートが記憶装置に相当する。これを何回か重ね中間結果ノートに書き取っておいてその結果と合わせて最後の計算をして答を出し，これを解答して最終結果用紙に書き取る。

この書き取るときの手が出力装置になるわけである。

このとき脳髄のすることは問題を解く方法を考え出すことと，それによりどの部分をどう動かせばよいかを定め，そのように動かすべく手に指示を与え，その動作が考えている通りになっているかどうかを目を通して監視することである。このとき脳髄が制御装置に相当するわけである。

電子計算機による計算は以中のことより分るように人間が卓上計算器で記憶されていたことがエレクトロニックなディジタル素子（真空管，ダイオード，トランジスタ，パラメトロン等）によって行われているのである。

1.2 電子計算機に於ける数の表現方法

電子計算機で数を表現する場合の基礎となるものは次のようなものである。

電圧が正か負か，パルスが有か無か，位相が0（ゼロ）相かπ（パイ）相か，孔が有か無か，電流が流れているかいないか，というものでこれらを数として扱うときは"1"と"0"に対応させて取扱う。このように1つのもので2つの状態を有するものが1単位となり，これをいくつかある規格に従って順序よく並べて1つの情報を表現する。このときこの2つの状態を2進要素といい，この要素を持つ素子を2進素子，この進号を2進信号という。2進信号によって作り出される数を2進数という。これが電子計算機に於ける数を扱うときの最小単位であって，これを2進1桁即ちビット（Bit- Binary Digit）と云う。このビットを順序良く並べ，それに2^n（n＝0，1，2，……）という荷重（Weight）を付ける

ことによって我々が使用する数を表現するのである。この２進数を色々使用して数字，文字等の符号を作成する。これを符号化（Coding）と呼んでいる。そしてその符号をコード（Code）という。

２進法の計算機（Binary Computer）に於いてはそのまま2^nという荷重をその並べたビットにつけて数を表現するので，１桁というものは"０"と"１"しかなく，"２"というときは１つ上の位に桁が上り，我々が一般に使用している"９"から"１０"になって桁上りを生ずる即ち10^nという荷重を桁につけて表現する１０進数（Decimal Number）と異るので取扱いが不便である。そこでこの２進数を使って我々が日常使っている１０進数を作り，これを使用すると便利である。この１０進数を使った計算機（Decimal Computer）に使用されている符号表を次にあげる。

	純2進化10進符号				3あまり符号				2 out of 符号					bi-quinary 符号						
＼	8	4	2	1	8	4	2	1	7	4	2	1	0	5	0	4	3	2	1	0
0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	0	0	0	0	1
1	0	0	0	1	0	1	0	0	0	0	0	1	1	0	1	0	0	0	1	0
2	0	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1	0	0
3	0	0	1	1	0	1	1	0	0	0	1	1	0	0	1	0	1	0	0	0
4	0	1	0	0	0	1	1	1	0	1	0	0	1	0	1	1	0	0	0	0
5	0	1	0	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1
6	0	1	1	0	1	0	0	1	0	1	1	0	0	1	0	0	0	0	1	0
7	0	1	1	1	1	0	1	0	1	0	0	0	1	1	0	0	0	1	0	0
8	1	0	0	0	1	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
9	1	0	0	1	1	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0

１０進法の計算機では符号表のような符号を１桁として，これを何桁か並べ数値，文字又は計算機に計算手順を与える場合の命令として使用するのである。

1.3　電子計算機に於ける語の構成

電子計算機に於いて内部で情報を取扱うときは通常これを一定の長さに区切り，これを単位として扱い，語（Word）と呼んでいる。このような方式を固定長語方式（Fixed word Length）と呼ぶ。語には計算に使用する数値を表わす数値語と計算方法を示すプログラムの構成単位となる命令語，文字を表わす文字語とがある。これらの語は電子計算機の記憶装置の中では形式，長さ等に相関性を持たせ，技術的には全く同等に取扱えるようにしてある。又取扱う情報の単位に数字１桁あるいは文字１字を使い，１語の長さを自由に変えられるようにしたものもある。このような方式を可変長語方式（Variable Word Length）と呼んでいる。

1.4　電子計算機に於ける数値，文字，命令の表現方法

２進法計算機の場合の数値の表現

（１語＝符号＋１２ビット）

(注)……この場合の符号は＋，－のことです。

〔例〕　10進数で－1881,＋1881という数値の表現

10進数	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	正負
－1881	0	1	1	1	0	1	0	1	1	0	0	1	1
＋1881	0	1	1	1	0	1	0	1	1	0	0	1	0

負数の表現には上記のような方法と次のような方法がある。

1	0	0	0	1	0	1	0	0	1	1	1	1

このような方法を補数表示方法という。

上記のように計算機では数値を取扱う場合，数＋ビットで1つの単位にして表わす。これを数値語と呼んでいる。

10進法に於ける数値の表現は1.2に於いて図で示したような符号を1桁としてこれを並べて1つの語を構成する。

●〔例〕　純2進化10進符号を使用した場合

（1語＝符号＋6桁）

荷重	10^{5}	10^{4}	10^{3}	10^{2}	10^{1}	10^{0}	正負
1	0	0	1	1	1	0	0
2	1	0	1	0	0	1	0
3	1	0	1	0	0	0	0
4	0	0	0	1	0	0	0

左の表は＋607912という数値を表わしている。

負数即ち－607912という数値を表現するにも2進法のときのように2種類ある。

1	0	0	1	1	1	0	1
2	1	0	1	0	0	1	0
4	1	0	1	0	0	0	0
8	0	0	0	1	0	0	0

絶対値表示表

1	1	1	0	0	0	0	1
2	1	0	1	0	0	0	0
4	0	0	0	0	0	0	0
8	0	1	0	0	1	1	0

補数表示法

2進法計算機に於いて文字を表現する場合は普通1語を6ないし8ビットで区切って使用し，これで1文字を表現する。即ち6ないし8ビットの組合わせで数　，英字，カナ文字，特殊文字を表現するのである。

10進法計算機に於ては1語を2桁で区切ってそれで1文字を表現するのが普通である。即ち0から99までの数　，英字，カナ文字，特殊文字にあてるわけである。この文字で構成された語を文字語と呼んでいる。

命令の表現としては数値，文字に使用する語をそのまま命令語に使用するのであって，その命令語の形式としては一般的に次のようになっている。

命　令　部	アドレス部

命令部とは我々の言葉に相当するものが入るところで，この言葉を機械語（Machine Language）といっている。これはやはり数で表現され，この言葉は普通1つの計算機で数十ないし数百種持っていて機械ごとに特有のものである。

アドレス部とはその言葉に従って操作される意味によって文字，数値のありか又は記憶する所，順序等を指定するもので，我々が"どこへ行く"というときの"どこ"に相当するものである。このとき"行く"という言葉が機械語に相当するのである。

この言葉を命令といい，これとアドレス（番地）を1つにして1個の命令語を構成し，これを計算内容に従って幾つか組合せて一連の計算の手順を作りあげるのである。

1.5 計算機内部に於ける情報の検査方法

電子計算機の中にはビットがたくさんつまっていて，これが高速に飛びまわっていると考えられる。そしてこのビットが記憶装置と演算装置，制御装置，入出力装置間を往来して一連の計算が行われるので，その間で何かの衝げきによってビットが消えたり又は余分に附加されたりする恐れがある。このようなことは電子計算機に於いては絶対許されないことで，このようなときは何かの方法で検出する必要が生じる。このために桁単位，文字単位，又は語単位に1ビット又は数ビット余分にその単位中で"1"になっているビットが奇数又は偶数になるという規則を作り，その規則に従って附加し，その情報を移動した場合にその規則に従っているかどうかを検査するようにしている。この方法をパリティ・チェック方式（parity Check）といい，このビットをパリティ・チェック・ビットと呼ぶ。

又この他に検査の方法として附号それ自身にその機能を持たせた，例えば1,2に述べた10進附号に於ける2 out of 5，bi-quinary等がそれである。

1.6 計算機内部に於ける情報の伝送方式

電子計算機内部で情報を移動する場合の伝送法には次のようなものがある。即ち情報を表わすビットが1本の伝送線路によって一定時間間隔で伝送される直列方式と各ビットごとに別々の伝送線路を設け，一時に1語の情報を伝送する並列方式，又これらの中間を取ったもの，即ち語を数ビットずつに区切って（例えば10進数1桁，文字ごと），このブロックは並列にそしてブロックごとには直列に伝送される直並列方式等がある。10進法表示のときは直並列，2進小型では直列，大型では並列が多い。

1.7 電子計算機の基本回路

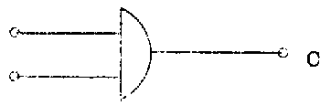
これには最も基本となるゲート回路（Gate）とフリップ・フロップ回路（Flip-Flop）がある。そしてこれらを組合せて色々の回路を作成するのである。

実際の回路はここでは省くことにする。ここではどのようなものがあるかについて述べる。

ゲート回路これは論理回路といい，ブール代数を基として作成された回路で，これには論理和回路（OR）論理積回路（AND），否定回路（NOT），禁止回路（INHIBIT）があり，又フリップ・フロップ回路にはダイナミック回路とスタテック回路がある。次に各々について説明する。

論理積回路とは2個以上の入力端子を持ち，全ての入力端子に所定の入力が加えられた場合のみ出力端子に出力が現われる回路で，次にその記号と真理表及び論理式を示す。

記　号　　A，B……入力端子



真　理　表

A	0	0	1	1
B	0	1	0	1
C	0	0	0	1

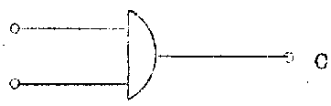
論　理　式

C＝A・B　又はC＝A⌒B

真理表の"1"は信号あり，"0"は信号なし。以後信号の有無はこのように示す。

論理和回路とは2個以上の入力端子を持ち，全ての入力端子に入力が加えられない場合以外は出力端子に出力が現われる回路をいい，論理積回路のように記号，真理表，論理式を示すと次のようになる。

記　号



真　理　表

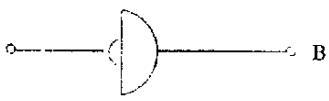
A	0	0	1	1
B	0	1	0	1
C	0	1	1	1

論　理　式

C＝A＋B 又はC＝A⌣B

否定回路とは入力端子に入力が加えられた場合出力端子に出力が現れず，入力が加えられない場合に出力が現われるという，入力と逆のものが出力に現われる回路といい，記号，真理表，論理式は次のようになる。

記　号



真理表

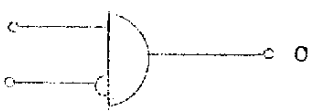
A	0	1
B	1	0

論　理　式

B＝～A 又はB＝$\overline{A}$

禁止回路とは論理積回路と否定回路を組合せて作成するもので，否定回路の方の端子に入力があった場合出力端子には絶対に出力が現れず，否定回路の方の端子に入力がなく，その他全てに入力があったときのみ出力端子に出力が現れる回路をいい，その記号，真理表，論理式は次のようになる。

記　号



真　理　表

A	0	0	1	1
B	0	1	0	1
C	0	0	1	0

論理式

C＝A・$\overline{B}$

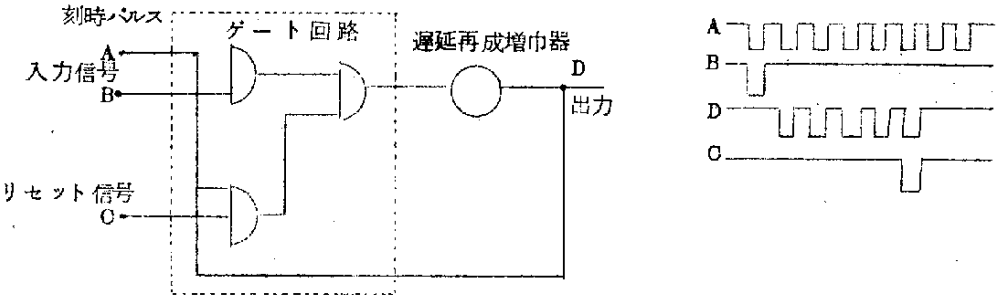
フリップ・フロップ回路は符号化された数を1ビット記憶する回路で，演算の際一時的に数を記憶しておく計数器や又は色々な制御のためにある一定時間信号を保持しておくときなどに使用するもので，これにはパルスを保持するダイナミック・フリップ・フロップ回路と

電圧の状態を保持するスタテック・フリップ・フロップ回路の2種類がある。

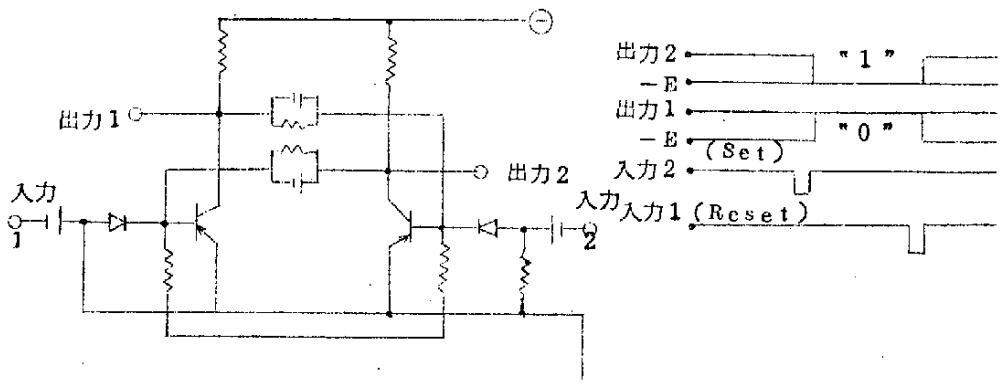
その回路構成を簡単に述べると，ダイナミック・フリップ・フロップ回路はゲート回路とパルス遅延再成増巾器よりなり，刻時パルス（クロックパルス）に同期あせてパルスを循環させる回路で，禁止信号によりこれを消去する。この回路ではゲート回路としてダイオード，遅延再成増巾器としてトランス，コンデンサ，トランジスタが各々1個ずつ使用されている。スタテック・フリップ・フロップ回路は2個の2進素子（真空管又はトランジスタ）を使用して，常にどちらか一方がオンのとき他方は必ずオフにあり，次の入力が加わるまでその状態を持続し，次の入力が加わった時に反転即ちそれまでオンのものがオフに又オフのものがオンになる回路である。それでこの2個の素子のどちらか一方の出力を"1"ときめれば，それが記憶された信号となり，他が"0"となり，直接否定信号も得られるわけである。

以上述べたのが電子計算機の基本回路でこれらを色々組合せて制御回路，演算回路を構成するのである。

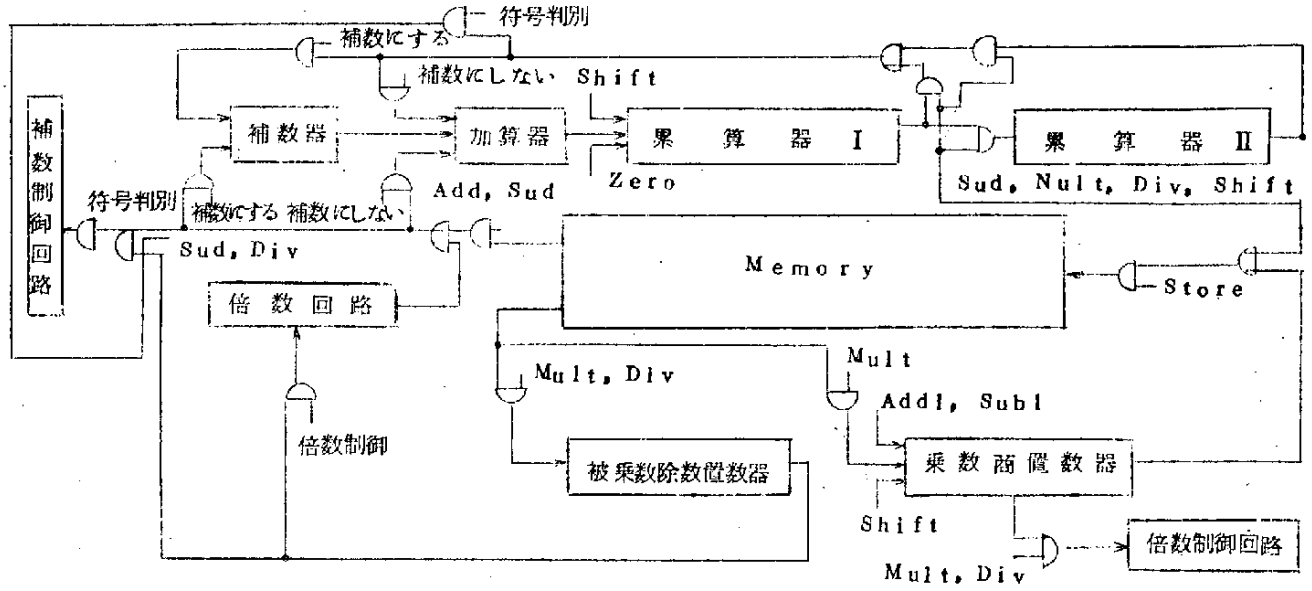
動的フリップ・フロップ回路



動的フリップ・フロップ回路



2. 演算装置及び演算方法



2.1 加　　算

2進法に於ける加算の原則として次のようなものがある。

0＋0＝0，1＋0＝0＋1＝1，1＋1＝0と桁上り。

〔例〕　01100　　左の演算は10進法の12＋8＝20を2進法で計算したもの

＋）01000　　で，12は通常累算器1（Accumulator I）に入っており，

10100　　8はMemory に入っており答が累算器に出る。

2.2 減　　算

計算機には減算回路を設けていないため減算は何かの方法で行う必要がある。この方法として補数法がある。これに使われるのが補数回路（Complementer）である。この回路には0進法，2進法とも2種類ある。

次に2進法に於けるこの2種類を使った計算を示す。

〔例〕　10進法に於いて9－5＝4は2進法では次のようになる。

2の補数のとき

01001　　01001

－）00101 ……→ 補数変換 →＋）11011

100100

↓

最上位からのあふれ

結果の最上位桁（Most Significant Digit）よりあふれが出るとこの結果は正であることを意味し，そのあふれを取除くとそれが結果になる。この場合の結果は10進法の＋4ということになる。

1の補数のとき

```
  01001                      01001
-)00101 ……→ 補助変換 →   +)11010
                           100010
                       +)        1   あふれの加算
                           00100
```

この場合補数を加えて結果ができ，最上位にあふれが出るとこの結果は正であり，そのあふれを結果の最下位桁（Least Significant Digit）にもう一度加える必要がある。そしてでき上ったのが最終結果である。

結果の最上位にあふれが出ない場合2の補数も1の補数も負数を表わし，その結果をもう一度補数に変換して結果を出す必要がある。

10進法の減算に於いて2進法の場合と同様2種類の補数がある。即ち2進法の場合の1の補数に相当するものとして9の補数，2の補数に相当するものには10の補数があり，10進法の場合の演算方式も2進法と全く同じである。補数を作る場合元の数を真数という。

○ 補数の作り方

a） 2進法

- 2の補数

  真数を最下位桁より見ていって最初に有効数字（"1"）が現れるまでそのままの状態にし，最初に現われた"1"の次の桁より"1"は"0"に"0"は"1"に変換する。これを語の最上位まで行う。

- 1の補数

  真数の各桁の"1"を"0"に"0"を"1"に変換する。

b） 10進法

- 10の補数

  真数を最下位桁より見ていって最初に有効数が現れるまでそのままの状態にし，最初の有効数字を"10"より減じてその残りの数をその桁にあて，次の桁よりは，"9"よりその現われた数を減じてその残りの数を対応する桁にあて，これを語の最上位まで行う。

- 9の補数

  真数の各桁の数各々を"9"より減じ，その残りの数を対応する桁にあてる。

2.3 乗　　算

桁単位に最下位桁より被乗数を乗数の各桁の数だけ桁移動（Shift）を繰返しながら最上位桁まで加算を繰返して計算する。

〔例〕　10進法の4×5＝20を2進法で計算する。

```
        0 0 1 0 0 ……被乗数
  ×)    0 0 1 0 1 ……乗　数
  ---------------
        0 0 1 0 0
      0 0 0 0 0
    0 0 1 0 0          } 桁移動を示す。
  0 0 0 0 0
+)0 0 0 0 0
  ---------------
  0 0 0 0 1 0 1 0 0
```

乗算の場合は結果の桁数はもとの桁数の最大2倍になる。それで結果全部を出すため累算器が2個ある。乗算では前もって被乗数をある置数器，ここでは被乗数除数置数器（Multiplicand and Divisor Register）に入れておき，記憶装置（Memory）の内容（これが乗数である）とかけ合せるわけである。乗算は加算を繰返すわけであるがその回数は乗数できめる。この乗数は実際にはMemoryから一度乗数商置数器（Multiplier and Quotient Register）に入り，ここの各桁が"0"になるまで加算を行い，"0"になったら桁移動し乗数全体が"0"になるまで繰返すわけで，結果が累算器ⅠⅡに出る。又10進法の計算器に於いては計算速度を上げるため倍数回路というものを使用して一度に被乗数の何倍かを加えるようにし，加算の回数を少くしている。

2.4　除　　算

乗算に於いては最下位桁より桁移動しながら加算を繰返して結果を出したが，除算は反対に最上位桁より桁移動しながら減算（補数変換して加える）を繰返して行うのである。この場合結果即ち商は減算した回数で減算した残りが剰余である。

補数又は真数を加えたりしたとき"1"を計数したり"1"を引いたりして順次桁移動を行って出来た

0 0 1 0 0

が商になるわけである。除算に於いては被除数は前もって累算器ⅠⅡに入っていなければならない。それでMemoryの内容で割るわけであるが，このときMemoryの内容が一時被乗数除数置数器にセットされ，この内容が引かれ（補数にして加える），そしてその回数が乗数商置数器にでき，剰余が累算器に残るわけである。この場合も10進法では計算速度を上げるため倍数回路が使われている。

〔例〕　10進法の21÷5＝4剰余1の計算を2進法2の補数を使用して計算する。

	[0]0 0 0 0 1 0 1 0 1	(1)	被除数を1桁ずらしておく。最上位の"0"がなくなる。
＋）	1 1 0 1 1	(2)	除数の補数を加え"1"を計数する。
	1 1 1 0 0	(3)	桁上りなし。
＋）	0 0 1 0 1	(4)	除数の真数を加え前に計数したものより"1"を引く。
	[1 0]0 0 0 1 0	(5)	桁移動してはみ出しの"1"と最上位の"0"がなくなる又同時に計数桁の桁移動も行う。
＋）	1 1 0 1 1	(6)	(2)と同じ動作
	1 1 1 0 1	(7)	(3)と同じ
＋）	0 0 1 0 1	(8)	(5)と同じ動作
	[1 0]0 0 1 0 1	(9)	(5)　〃
＋）	1 1 0 1 1	(10)	(2)　〃
	[1]0 0 0 0 0	(11)	桁上りある，あふれの"1"がなくなる。
＋）	1 1 0 1 1	(12)	(2)と同じ動作
	1 1 0 1 1	(13)	(3)と同じ
＋）	0 0 1 0 1	(14)	(4)と同じ動作
	[1 0]0 0 0 0 0	(15)	(5)　〃
＋）	1 1 0 1 1	(16)	(2)　〃
	1 1 0 1 1	(17)	(3)と同じ
＋）	0 0 1 0 1	(18)	(4)と同じ動作
	[1 0]0 0 0 0 1	(19)	(5)　〃
＋）	1 1 0 1 1	(20)	(2)　〃
	1 1 1 0 0	(21)	(3)と同じ
＋）	0 0 1 0 1	(22)	(4)と同じ動作
	[1]0 0 0 0 1	(23)	あふれの"1"がなくなる。これが剰余である。

3. 制御装置及び主制御装置

3.1 制御装置

制御装置は計算機の動作を決定する命令置数器(Order Register)をはじめ，解読器(Decoder)，符号器(Coder)，計数器(Counter)，選択回路(Selector Circuit)一致回路(Coincidence Circuit)，主制御回路(Main Control Circuit)，制御パルス発生器(Control pulse Generator)等より構成されている。

計算機で計算していく上にはこの制御装置は欠くことのできないもので，これは我々の脳に相当する働きをするもので，それぞれの制御は我々が作ったプログラムというもので行われる。このプログラムを実行するに当り，このプログラムを機械にどのような方式で与えるかということより見るとこれに2種類ある。

即ちプログラムを一担最初に記憶装置に記憶しておく内部記憶プログラム方式(Stored Program System)と計算機の外部に配線盤(パッチボード)にプログラムを配線し又は特殊なカード及び紙テープにプログラムをさん孔してこれらを機械の外部にセットして機械に

指令を与える外部プログラム方式とがある。

ここでは内部記憶プログラム方式を中心に述べる。

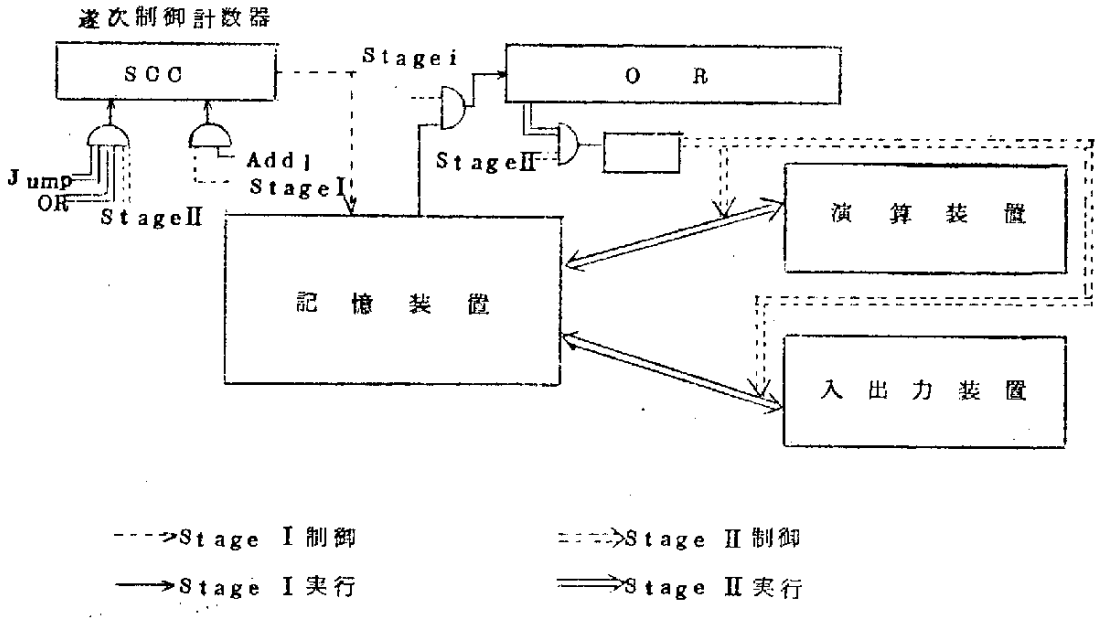
制御装置を大きく分けるとプログラムの実行順序を制御するもの即ちサイクルを制御するものと命令夫々によって制御されるものの２種類ある。ここでは前者について説明する。これが主制御である。

これは内部記憶プログラム方式に於いて，前もって貯えられていた命令がどのような順序で実行されていくかということを制御するものである。

これは命令語の形によって異る，即ち命令語は前に述べたように命令部と番地部に分れている。この場合この番地部が機械によって１つとは限らず，２つ又は３つというものもあり，その使われ方によっても又呼び方が異る。

１つのものは１アドレス方式と云われ，２つのものは１＋１アドレス方式及び２アドレス方式といわれ，３つのものは２＋１アドレス方式及び３アドレス方式と呼ばれる。又インデックス・レジスタ（Index Register）というものがあって，これによってアドレスを変化できるものは$1/2$というものを付けて$1\frac{1}{2}$アドレス方式などと呼んでいる。

まず１アドレス方式について説明する。（１０進法）



上図で説明すると，まずＳＣＣ（Sequence Control Counter）に０１００という数が入っていたとする。そして記憶装置の０１００というところには０２０５００と入っているとする。０２というのはこの機械では"加える"という意味の機械語である。計算開始指令が人間により与えられたとすると，そのときのＳＣＣの内容は０１００であるので，記憶装置の０１００番地の内容０２０５００がＯＲに持って来られ，その後ＳＣＣにＩが加えられる。この段階をStageIという。これからわかるようにＳＣＣとは次に行おうとする命令の入っ

ているところを示す計数器である。このためこの内容を変えることにより色々枝分れさせて計算したり，同じ通路を何回も通させることができる。この作用を行うのが飛越し（Jump）命令である。

StageIが終ると次に命令の実行段階に入るわけである。そしてこの段階で命令が解読され実行される。これがStageII である。このStageIIではORの内容に従って色々な制御がそれぞれの制御装置を通して実行される。この例ではORに０２０５００と入るわけで，02というのが解読器により"加えよ"と判断され，それに関する制御装置に信号が送られる。それと同時に番地部の０５００も解読され５００番地にある内容が演算装置に加算器を通して送られるのである。そして完全に加算が終了すると主制御装置に終了信号が帰ってきてStageIIが終了し，Stageに移るわけである。このStageI と StageIIが交互に繰返されて一連の計算が行われるわけである。

次に１＋１アドレス方式について説明すると，これは１アドレス方式の場合のＳＯＣがないものと思ってよい。即ち２番目のアドレスによって次に行う命令のありかを示すわけである。

２アドレス方式に於いてはＳＯＣがあり，これで次に命令のありかを指示することは１アドレス方式と同じであるが，実行段階に於いてアドレス２つでその操作されるところを指示するわけである。これを１アドレス方式で行うと２つの命令が必要になるわけである。２＋1アドレス方式，３アドレス方式も同じように考えればよい。

4. 記憶装置

記憶は電子計算機には欠かせないものである。この記憶装置は用途によって２種類に分けることができる。

即ち内部記憶装置（計算手順及びすぐ使用するデータ，中間結果の記憶に使用）と外部記憶装置（内部記憶装置の補助的な役目をするもの，又はファイルの動きをするもの）と分けられる。この場合前者は一般に書き込み読み出しが高速に行われるものが要求され，後者は速度よりも容量の非常に多いものが要求される。

一般に記憶装置とは読み書きができるもの，書いたものをいつまでも保持できるもの，忘れる即ち消すことのできるもの，位置の選択のできるものでなければならない。選択のために使用するのが番地である。記憶装置に於てはこの書込み読み出しの場合にその場所を選択し，書込み読み出しを行うまでに時間を要するものと要しないものがある。これが記憶装置の速さになり，この時間を呼び出し時間（Access Time）という

4.1 磁気ドラム記憶装置

これはＡℓ，黄銅系の円筒の表面に磁性体（酸化鉄粉 Fe_2O_3 ）を塗布し，この表面に円筒軸に沿って高透磁率のフェライト鉄心又はパーマロイ薄鉄板を積み重ねて作った鉄心にコイルを巻いた磁気ヘッドと呼ばれるものを多数並べ，円筒を高速度で回転させ，その巻線に電流を流し，ヘッドの磁心にできた磁束によって磁性体の表面に小さな磁石を作成する，この状態が記憶した状態であり，読み出しとはその記憶されたもの（小さな磁石がヘッドの下を通ったとき磁束がヘッドの鉄心を通る，そうするとそれに巻いてあるコイルに起電力が誘

起され電圧が生じる。これを読み取り信号として使用するのである。ヘッド1つについての円周上の信号記録範囲をトラックといい，これを数個まとめてバンドという。

この種の記憶装置で1周にヘッドが1個の場合は，同じ場所に書いたりその場所から読んだりするとき，1周に一度の機会しかない。そのため最大1周するに要する時間が必要である。これが前に述べたアクセスタイムというものである。これの小さいものは内部記憶装置に，大きいものは外部記憶装置として補助的なものに使用される。

ドラムの表面はそれぞれ区切られていて，その1つ1つに番地というものがあり，この番地によってその表面のどこかを示すわけで，これは我々の土地に於ける番地と全く同じ働きをすると考えればよい。この番地の広さは語が単位になるもの，桁が単位になるもの及び字が単位になるものがあり，語が単位になるものを固定長語方式，桁，字が単位になるものを可変長語方式という。

このドラムの記録密度は通常3～4 bit／㎜である。通常高速ドラムではその回転速度は3,000～20,000 rpmで，その容量は30,000～200,000 bitであり，低速ドラムでは3,000 rpm以下，その容量は200,000～30,000,000 bit程度である。

4.2　磁気テープ記憶装置

この装置はテープレコーダを思い出していただければよい。しかしテープレコーダより装置は大変複雑で又記録テープ自体も良質でなければならない。

磁記テープはプラスチックベース（アセテート又はマイラー）上に磁性材料を塗布したもので，直経30　程度のリールに巻いて使用するもので，その長さは750 m程度で巾は1／2 インチ，3／4 インチ，1インチ等がある。このテープを磁気テープハンドラーという書込み読み出しヘッドのついた装置にかけてテープを駆動し読み出し書込みを行うのである。磁気テープは常にテープを動かしているものではなく，書いたり読んだりするときのみ走らすので，テープを走らせたり停止させたりする機能が必要である。このときテープに急激な力が加わる恐れがある。そこでこの力をなくするためテープをためておく必要がある。このためておく方法にテンションアーム方式と真空方式（バキュームコラム方式）がある。一般にテープの駆動はキャプスタンローラーというものを一定速度で回転させておき，これに読み出し書込み命令により計算機より信号を与え，ピンチローラーを動作させその摩擦によって走らせ，テープ速度が一定になったところで始めて書いたり読んだりするわけである。このとき加速時間（Start Stop Time)はむだになり何も書かれないところとなる。これをIRG（Inter Record Gap)といい，通常10～50㎜ある。又テープの先端と後端には始めと終りを示すマーク（透明，又は箔）がある。

磁気テープのヘッドは通常巾方向に7～9個あり，それぞれ書込ヘッド読み出しヘッドが何㎜か離れて別々についている。1対のヘッドで読み書きできる範囲をトラックという。テープ上の記録密度は少ないもので4～10 bit／㎜，多いもので10～50 bit／㎜にも及ぶ。そしてテープの速度（読み書き時）は1～4 m／sec 程度である。以上のことより磁気テープに於ける伝送速度が計算できると思う。

この磁気テープ装置は逐次式に書いたり読んだりするので，飛び越して読んだり前の方の

データを読んだりするのに非常に時間がかかり遅い。しかし容量は無限に取れるので外部記憶装置として用いられ，ファイルとしては最適である。

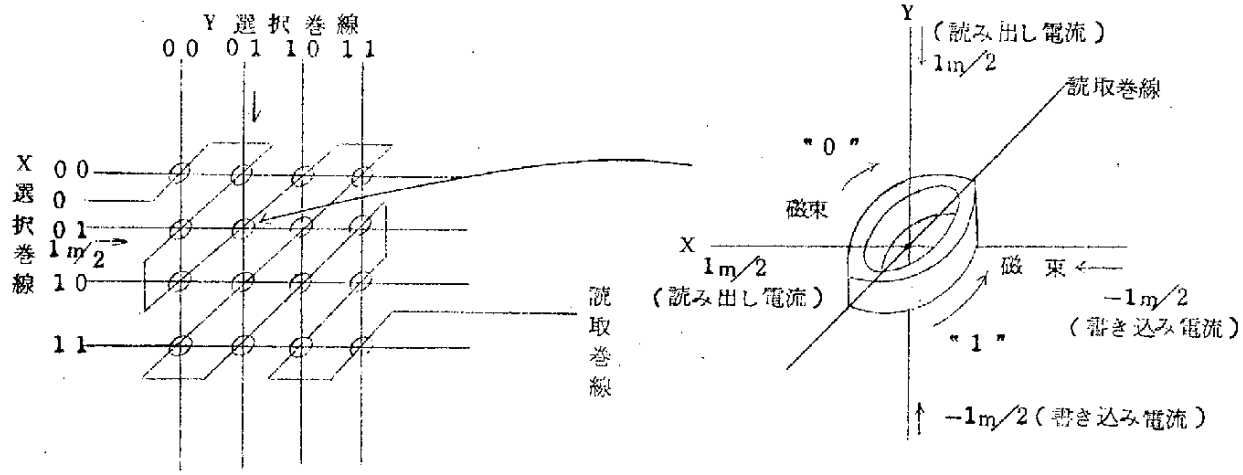
テープレコーダに於いては連続信号を記載するのであるが，計算機では継続信号を記録する。このためテープ自体に非常に良質のものが要求されるわけである。

4.3 ディスク気憶装置

これはレコード盤のような円板の表裏面に磁性材を塗布し，これを積み重ねその間に書込み読み出しのためのヘッドが入るようにして回転させて使用するもので，この円板の面を記録体に使う。その方法は磁気ドラムと全く同じである。この円板は取りはずせるもの(磁気ディスクパック記憶装置)と固定のもの(一般的な磁気ディスク記憶装置)がある。又ヘッドが1～3本のアームの先端についていて，それが上下左右に動くものと櫛状に固定されていて前後に動くものとがある。円板に記録するのであるから同心円的に記録されるもので，この円板に1つのヘッドで記録される範囲をトラックといい，1つの面にはこのトラックが多数あるわけである。一般にこの装置の速度はドラムより遅いが容量は多く又磁気テープよりアクセスタイムは速いが容量は少ない。即ち，磁気ディスクパック記憶装置で100万字～1000万字磁気ディスク装置で1000万字～数億字である。これも外部記憶装置に使用される。

4.4 磁気コア記憶装置

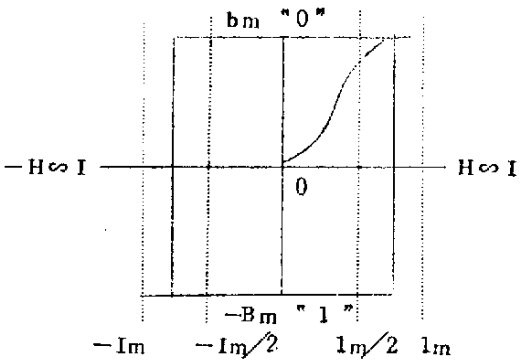
この書込み読み出し方法は前の3つとは非常に異る。即ち前のものは記録媒体を動かすことによって，その面に小さな磁石を作ることにより記録し，書いたり，読んだりしている。これに使用している記録媒体は角形ヒステレシス特性(Hysterecis)を持つ外径0.5～2.0㎜のフェライト環状磁心を用い，これを図のように網目状に並べ，これに縦横斜に線を通し，この磁心の磁化状態により1つの磁心を1ビット記憶として使用するもので，記憶装置として大変すぐれたものである。その速度は非常に速く現在すべての内部記憶装置に使用されている。その速度は数 μsec 程度である。



4. 5　その他の記憶装置

4. 5. 1　磁性薄膜記憶装置（ワイヤメモリー・スポット形磁性薄膜記憶装置）

これは原理は磁気コア・記憶装置と同様の方式によって記憶するものでその記憶素子として磁気コアのかわりにパーマロイの薄い膜を使用するものでワイヤメモリーは金属線にパーマロイも電着して作りスポット形磁性薄膜記憶装置はガラスのような基板にパーマロイの蒸気をスポット状に電着して作ったものでこれは非常に小さく又スケッチ速度は磁気コアの $\frac{1}{10}$ から $\frac{1}{100}$ なので最近非常に注目されている。これは主記憶装置に使用される。



前項図の黒くつぶしたコア（0101）に信号を書き込むとする。このときX選択巻線とY選択巻線に単独では磁化の方向を変えるに足らないが2つ重なると磁化の方向を変えるような電流即ちヒステレシス曲線からわかるように，磁化の方向を変える電流とはIm及び−Imのことでこれの半分即ちIm／2及び−Im／2 の電流のことである。そして書き込むということは"1"の状態にするのであるから，図よりそれぞれの巻線に−Im／2の電流を流せば交点のコア　は−Imという電流が流れるため，前に"0"の状態にあったものが"1"の状態になって書き込まれるわけである。これが書込みで読み出しは書込みと反対方向に電流を流すわけである。そうすると今まで"1"の状態に大きく変化するので，磁束の変化によって読取巻線に起電力が発生する。その起電力を信号として読み取るわけである。しかしこのとき記憶していた"1"が"0"に変化する。そのため内容が読み出すたびに破壊される。それで読み出したときに信号のあったコアだけにもう一度書込む操作が必要である。又書き込みに於いてもその前に一度コア全体を消去する必要がある。このようにこの種の方法では2回の操作が必要になるわけで，この操作の時間のみが速度に関係するのであるから速度が速いのである。この方法を電流一致方式という。この他に記憶状態が破壊されない方法もあるがここでは省略することにする。

以上述べた何く各種の記憶装置があるがこの記憶装置に要求される条件を述べると

1. 価格が安く，小形でかつ消費電力の少ないもの
2. 動作速度（アクセスタイム・サイクルタイム）の速いもの
3. 動作の安定性，信頼性が高く寿命が長いこと

が考えられこれらの条件を多く満しているものが使用され又開発されるわけである。

5. 入出力装置

5.1 入力装置

計算機にデータ及びプログラム等を入れる手段になるものとして一般に次のようなものがある。

a) 紙テープ（巾 22.5 mm, 25.4 mm）

b) カード （80欄, 90欄, 縦 3¼ インチ, 横 7/8 インチ）

c) 印刷物

上記の印刷物以外のものは長方形又は円形の孔をある規則（コード）に従って専用の機械であけ，これを図A，Bのように光を通して光電管又はフォトトランジスタによって電気信号に変えるものと，ブラシによって導通させて電気信号に変えるものがある。これらを読み取る機械もそれぞれ専用のものがあり，一般的に光によって読み取るものは高速である。

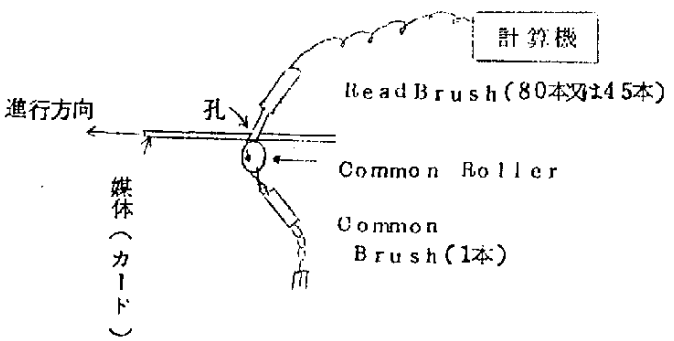
紙テープを読むものは紙テープ読取機といい，この読取高速は200～1200 ch/sec 程度である。

カードを読むものはカード読取機と呼ばれ，その速度は200～1500 Cards/min でこれには光によるもの，ブラシによるものがあるが，高速の部類に入るものは殆んど光によるものである。カードの場合は1枚で最大80字扱えるものと90字扱えるものの2種類ある。

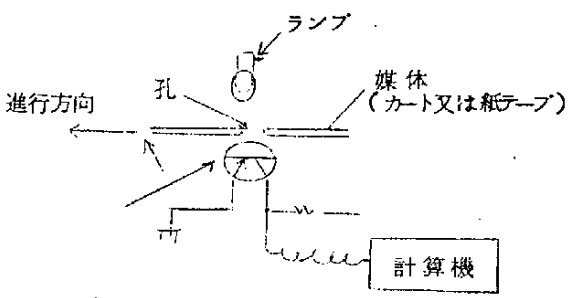
又この他に光によって印刷物を又磁気ヘッドによって磁気インクで書いた字を読み取る光学文字読取機（Optical Character Reader）及び磁気インク文字読取機（Magnetic Ink Character Reader）又，鉛筆や特殊なペンで指定個所にマークをし（縦又は横の線）それを光によって読み取る光学マーク読取機（Optical Mark Reader）等があるがここでは省略することにする。

勿論この入力装置に最も簡単なものとして鍵盤読み取りがある。即ちタイプライタ等の鍵盤をそのまま使用するわけである。

図A ブラシ方式



図B 光方式



光電管又はホトトランジスタ
カードの場合80個又は45個
紙テープの場合6個又は8個

5.2　磁気カード記憶装置

これは記憶の方式は磁気ドラム・ディスクのように磁性体に記録するもので読み出し書き込みは全く同じであるが媒体がカード状になっておりこの表面に磁性体を塗布してこの表面に記憶する方式であり非常に沢山の容量を取ることが出来る。

5.3　出力装置

これは計算機で計算した結果を我々が目で読めるような状態にする装置で，この手段としてすぐ読める印刷物にするものと，後でもう一度機械で使用するようなデータを出す即ち紙テープ又はカードに孔をあけるというような2種類がある。

印刷物として出すものには1字づつ印刷するものと1行を一度に印刷するものの2種類がある。前者を頁式プリンタ，後者を高速製表機又はラインプリンタといい，その速度は前者で５００～６００ch/min，後者で1行に最大１００～１５０chで200~1500line/minである。

ラインプリンタの方式としては活字が円筒上に配列されているドラム方式，ベルト又はチェーン上に配列されているベルト方式，チェーン方式等がある。

紙テープに孔をあけるものとして紙テープさん孔機があり，これには低速と高速があり，低速のものは５００～６００ch/minで，高速のものは３０００～１２００ch/minである。

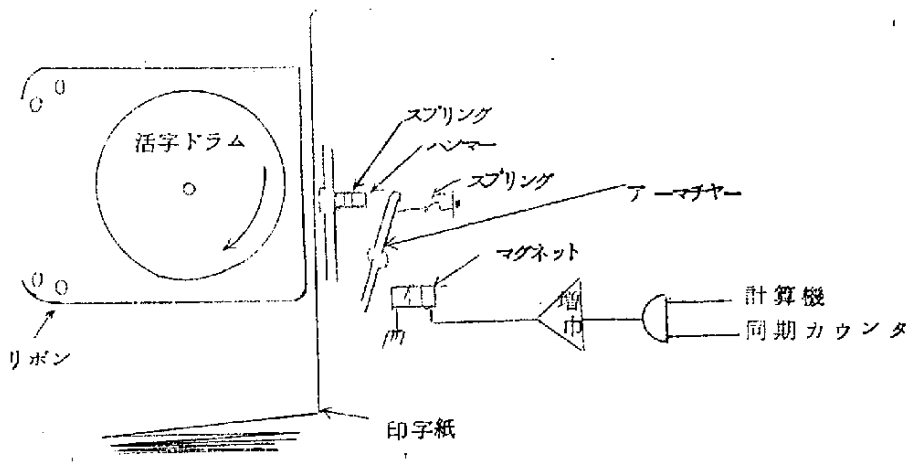
カード孔をあけるものとしてカード穿孔機があり，その速度は１００～５００Cards/minである。

この他に鍵盤，頁式プリンタ，低速度紙テープ読取機，低速度紙テープさん孔機のついた万能入出力装置と呼ばれるものもある。

5.4　固定記憶装置

これは読み取り専門の記憶装置で読み取り速度の非常に速いこと，大容量に出来て小型でかつ安いことが要求される。

次にラインプリンタの印字機構の一例として，その模式図をあげておく。



Ⅲ　電子計算機の現状と将来

Ⅲ　電子計算機の現状と将来

1.　電子計算機の現状と将来

はじめに

電子計算機産業はまだ若い産業である。１９４６年，世界で最初の電子計算機ＥＮＩＡＣ（ペンシルバニア大学）が開発されてからわずか２０年であるが，その間における技術の進歩には実にめざましいものがある。

技術進歩のテンポが速いことと研究開発のために莫大な投資を伴なうことは，この産業の大きな特長であって，８～９年まえにすでに真空管から固体回路（トランジスタ，パラメトロンなど）という２つの世代の機械が交代している。しかも，現在では超小形集積回路のように進歩した構成素子が組みこまれた，第３世代の電子計算機が出現した。

いっぽう，電子計算機の利用普及も急速である。最初の業務用電子計算機が米国人口統計局に設置されたのは１９５１年で，それ以来まだ１５年を経過したにすぎない。また量産の技術が電子計算機の生産に適用されたのは1,９５３～１９５４年であるといわれている。しかるに，現在，世界ではすでに５０,０００～５５,０００台以上（推計）が設置され，広範な分野で利用されている（表１参照）。

わが国においても，開発に着手してまだ１０年にすぎないが，その間メーカー各社の努力と政府の振興策によりめざましい発展を示している。また，各企業における電子計算機導入の気運も昭和３６年頃から急激に高かまり，現在では推定3,６００台が設置されている。

このような市場の拡大に伴ない輸入も増加しているが，生産の伸びが著しいので輸入依存率は低減しつつある。しかし，研究開発体制の強化，生産体制の合理化などについては，まだ十分とはいえない状況で，国際競争力についても依然として輸入制限体制下にあり，早急な国際競争力の強化を迫まられている。

しかも，第三世代への機種交換期を迎えて，わが国電子計算機産業の進むべき方向が，あらためて問題とされるにいたった。

表１　各国の電子計算機設置状況

１９６７年６月末

国　名	台　数
アメリカ	３２,５００
西ドイツ	3,３００
日本	2,７００
イギリス	2,２００
フランス	1,９５０
イタリア	1,３００

ＩＮＴＥＲ－ＮＡＴＩＯＮＡＬ
ＤＡＴＡ　ＣＯ.

1.1　電子計算機産業の世界の動き

電子計算機産業の世界的動向に目を向けた場合，まず注意をひくのはＩＢＭの存在である。ＩＢＭは世界市場の７０％を制し，完全に主導権を手中におさめているようである。1９６７年における海外を含めた総売上げは５３億ドル（１兆9,０８０億円）を越えており，これはわが国電子工業の全生産額（１９６７年度１兆円）を上回わる額である。さらに注目すべきことは，ここ数年来，１０％内外の高度成長を続けるとともに，売上高利益率も常に１０％

を上回わっていることである。また研究開発への投資も，わが国電子計算機生産額の数倍という規模に達しており，世界の電子計算機企業はＩＢＭの動向を無視した経営方針を立てることは不可能になっている。

このように，いわば独走体制に近いＩＢＭに対して，世界の電子計算機企業は懸命の巻返えしを策しつつある。米国でもＵＮＩＶＡＣをはじめとして，ＲＣＡ，ＣＤＣ，Honeywell，GE，NCR，Burroughsなどの諸企業がＩＢＭを追い，激しい競争を展開している。ＣＤＣは合併を続け急速に事業の拡大をし，ここ７年間に売上げを約２００倍に増やしたといわれており，またＧＥも電子計算機事業に本腰を入れはじめ，Bull(仏)，Olivetti(伊)を支配下におさめて世界的規模で巻返えしを図っている。ＵＮＩＶＡＣ（スペリーランド）は電子計算機メーカーの名門であり，その歴史は古く，従来からＩＢＭに対抗する企業の雄として存在し，軍関係に多くの顧客をもち，計算機システムの設計についてもＩＢＭと異なる行き方をしている（表２参照）。

表２　米国電算機市場のシェア

単位　１００万ドル

メーカー	市場シェア	設置金額
ＩＢＭ	７０％	１３,３００
ユニバック	7	1,３３０
ハネウェル	5.5	1,０４５
ＣＤＣ	4.3	８１７
ＲＣＡ	3.2	６０８
ＧＥ	3.2	６０８
バロウス	2.7	５１３
ＮＣＲ	1.7	３２３
ＴＯＴＡＬ	9７.6	１９,０００

（出）Computer INDUSTRY SURVEY TOTALが100にならないのはまだほかにメーカーがあるため（1968年末調）

イギリスでは，乱立状態にあった企業が，激しい国際競争に対処するため次第に集約化され，現在ではＩＣＴとEE−Leo Marconiの２企業にまとまり，ＩＢＭ，その他の米国企業にたちむかいつつある。

ＩＢＭを追う企業としては，以上の他にフランスのBull，西ドイツのTelefunken，オランダのPhillips，イタリアのOlivettiなどの企業があるが，すでに述べたように，Bullは資金難におちいりＧＥの支配下に入った。

最近の企業活動の特色は，ＩＢＭに対抗して，できるだけ市場シェアを確保し，今後の国際競争に生き残り，１９７０年以後における世界の電子計算機企業の位置を獲得する点にある。

1.2　わが国における電子計算機産業のあゆみ

1.2.1　開発研究時代

わが国における電子計算機の開発研究は，主として大学および官公立の研究所を中心に行なわれた。昭和３１年，富士写真フィルム（株）で真空管を使用した"FUJIC"が完成したが，わが国における電子計算機の最初である。この電子計算機は，同社の技術陣がレンズ設計の計算のために自家製作したものであり，真空管式の電子計算機で実用に供されたものとしては，わが国唯一のものであるといえる。

また，昭和３３年には東京大学が東京芝浦電気（株）の協力を得て真空管式の電子計算機"ＴＡＣ"を試作した。しかし，このような真空管を使用した電子計算機は生産に移さ

れることなく，後述するトランジスタまたはパラメントロンを使用した電子計算機の開発生産に移行した。

トランジスタを使用した最初の電子計算機は，昭和３１年通産省工業技術院電気試験所で完成した。"ETL−markⅢ" で，つづいて昭和３２年，"ETL−markⅣ" が製作された。この電子計算機の完成によって，製造各社のトランジスタ式電子計算機の製造計画は具体化した。

この技術指導をうけて，昭和３３年には，商品としての第１号機である日本電気（株）の "NEAC−2201"が完成した。ETL−markⅣの技術は，その後日立製作所をはじめ各製造会社の電子計算機に影響を及ぼした。

さらに，電気試験所においては，その後 "ETL−markⅤ"（昭和３５年）を設計したが，この計算機は（株）日立製作所で製作され，商品化された。

一方，わが国の技術をほこるパラメントロンが，昭和２７年東京大学において，その原理が発見されてから，これを電子計算機の回路に使用する研究が進められた。昭和３３年には同大学でパラメントロンを使用した "PC−Ⅰ" が完成した。その後，本機の性能向上について検討が加えられ，昭和３５年に富士通信機製造（株）はよって "PC−Ⅱ" が完成した。

なお，パラメトロン式電子計算機は，日本電信電話公社電気通信研究所においても別途研究が進められ，昭和３２年にはパラメントロン式電子計算機の第１号機である "MUSASHINO−Ⅰ"が作られた。その後改良されて昭和３５年には "MUSASHINO−IB" が完成した。この MUSASHINO−IB は電気通信研究所の設計になるものを富士通信機製造（株）が製造にあたった。

このように，わが国における電子計算機の製造は，電気試験所，大学あるいは電気通信研究所といった研究機関での開発研究を基礎として出発したといえる。

1.2.2　商品化の時代

昭和３３年から４年間はわが国電子計算機の商品化の時代である。

総合電機メーカおよび通信機メーカが相続いて試作に乗出し，昭和３３年～３４年にかけて，日本電気（株），富士通信機製造（株），沖電気工業（株），三菱電機（株），松下通信工業（株）が加わり，電子計算機メーカは７社を数えるに至った。

商品化の時代になると，研究所の基礎技術に各メーカの技術が加わり，それぞれ事務機としての電子計算機の生産が進められた。

一方，政府においても，電子工業の振興をはかるために，昭和３２年に電子工業振興臨時措置法を制定し，電子計算機生産の振興が国の大きな政策としてとりあげられた。

また政府の振興方策に呼応して，わが国電子工業界に指導的地歩をしめる有力メーカによって，社団法人日本電子工業振興協会が設立された。この協会は，電子技術の向上，電子機器の開発普及，電子工業の合理化等により電子工業を振興し，日本経済の発展に寄与することを目的としている。特に電子計算機に関しては，電子計算機センターを設けて国産化とその普及に貢献してきた。

さらに国産電子計算機販売促進の一環として，昭和３６年に日本電子計算機（株）が設

立された。これは，ＩＢＭ，レミントン・ユニバックその他海外の多くの電子計算機メーカが実施しているレンタル制度に対抗するもので，国産電子計算機メーカ７社が共同出資（現在は松下通信工業を除く６社）して作った会社である。出資会社はこの会社に現金販売し，ユーザーはこの会社にレンタル料を支払う仕組みである。

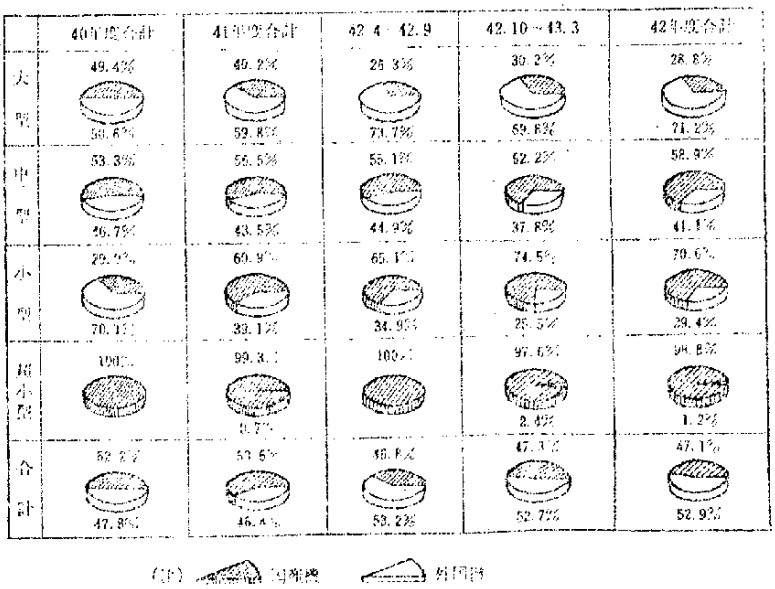
フランスのマシン・ブル社，イタリーのオリベッティ社など外国の有力な企業の電子計算機生産部門が危機に直面しＧＥに吸収合併されたが，その大きな理由としては，レンタル制による資金難があげられている。この事実からしても，国産電子計算機メーカが共同でこのようなレンタル会社を設立したことは，わが国電子計算機産業の発展のために大いに意義のあることである。

その後電子計算機の普及にともない，通産省，郵政省と関係団体の協力によって４２年１２月財団法人日本情報処理開発センターが設立された。

1.3　わが国の電子計算機産業の現状

わが国の電子計算機の生産は昭和３３年ごろから始ゆられ，企業化が軌道にのったのは昭和３５～３６年のことである。それ以来の成長は図１の推移にみるごとく順調に拡大を続けている。とくに国産機のレンタル制が確立された昭和３６年以降，いよいよ本格的生産段階に入った。

型別からみた40年～42年上期における国産機・外国機の比率金額



（注）　国産機　　外国機

現在，電子計算機を生産している主な企業は，通信機メーカーの富士通，日本電気，沖電気工業と総合電機メーカーの日立製作所，三菱電機，東芝の６社である。通信機３社は電子計算機が身近かな分野の事業であり，その研究開発が今後の企業成長につながる立場でもある。そのため開発には大いに力を入れている。大手総合電機メーカーの場合は，電子工業の結晶といわれる電子計算機をわがものにすることで，総合電機メーカーとしての地位をさら

に高めようとするものである。

いっぽう政府機関や企業における電子計算機の利用状況をみると，昭和３２年にはわずか３台であったものが，４３年６月現在約3,５６０台に達した。うち国産機の占める割合は台数で６９％，金額で４７％となっているが，輸入依存率は漸次低減しつつある（図１参照）。

いままでの経過のうち，特筆すべきものとして次の事項があげられる。

(1) 電子工業振興臨時措置法の制定（昭和３２年）

(2) ＩＢＭと国産機メーカーとの基本特許契約の締結（昭和３５年）

(3) 日本電子計算機株式会社（レンタル会社）の設立（昭和３６年）

(4) 技術提携（昭和３６年～３９年）日立－ＲＣＡ，三菱－ＴＲＷ，（ＴＲＷはバンカー・ラモに吸収され，さらに，電子計算機部門はＧＥに合併された）日電－Honeywell，沖－UNIVAC，東芝－ＧＥ

これらの措置によって，わが国電子計算機産業のレベルが大幅に引き上げられたことは疑いのないところである。しかし海外，特に米国の電子計算機産業は，これを凌ぐ成長を遂げており，このような状況から，いまもって競争力が十分であるとはいいがたい。今後とも，諸外国の動向に対処して，国際競争力を強化するための強力な施策を早急に推進することが迫られている。

1.4 急速化しつつある技術革新

1.4.1 第３世代への移行

電子計算機をめぐる技術革新のテンポはきわめて急速である。昭和３９年４月，ＩＢＭ３６０の発表を契機として，電子計算機産業における超小形集積回路や，その他の超小形部品の利用がいよいよ本格化し，第３世代への動きが活発になった。また，ＲＣＡのSpectra７０など，他の企業からも相ついで高性能の新機種が発表され，世界の電子計算機業界は急激な機種交代期に入った。

小形化された半導体集積回路を基礎に設計された計算機は，現在までの計算機よりさらに高速で，信頼度が高く計算機のサイズを縮小することが可能となり，かつ原価も長期的にみれば，第２世代の計算機よりも安くなるみこみである。

さらに，最近では，データー伝送をともなう実時間処理（online realtime process）や広範な情報の多重処理の問題がとりあげられるようになり，計算機の適用範囲はますます広がりつつある。このような応用面の拡大や計算機本体の性能の向上に伴なって，従来の入出力装置の高速化はもちろんのこと，新しい入出力機器として光学式文字読取機やディスプレイ（表示）装置の結合など，人間と電子計算機が自由に能率よい情報の連絡ができるような方向に努力が払われるであろう。また，実時間処理や多重処理の発展につれて，データ伝送系が利用されるようになり，これに伴なう方式や関連機器の開発が進むものと思われる。

1.4.2 ファミリーの形式

第３世代の電子計算機の特長は，超小形集積回路の使用だけでなく，Compatible な計算機（ファミリーズ）の発展がある。Compatibilityとは，一連の計算機群が同一の

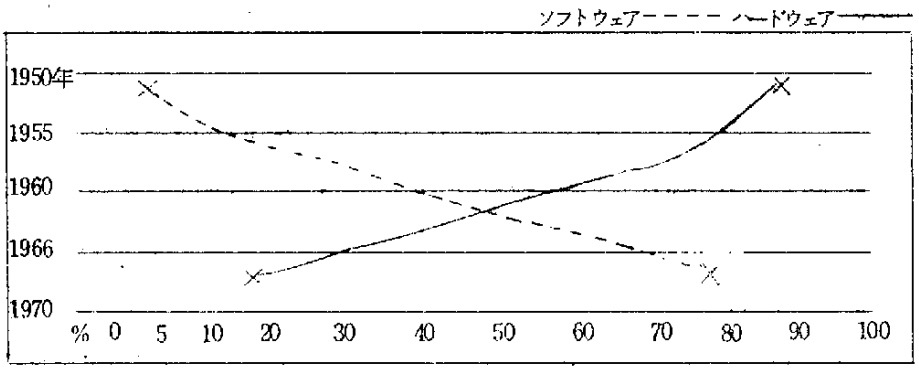
プログラム方式をもち，さらに使用者の要求に応じて任意のシステムが選択でき，また，応用分野の拡大に伴ない広い用途に応じうるようなシステムである。つまり，使用者は，より大きなセントラル・プロセッサや，より高速の周辺機器に仕事を拡大する場合，広い範囲にまたがるいろいろな周辺機器からの選択ができるし，また一層大きな力をもつプロセッサにたやすく切り換えることができる。

初期のファミリーズの1つとして，GE400シリーズがあげられるが，最近ではこの他にIBM360，RCA Spectra70, Honeywell H200, GE 600, CDC 3000・6000, UNIVAC 9000 シリーズなどがファミリー機種に加わった。

わが国でも，すでにTOSBAC5400シリーズ（東芝），NEAC2200シリーズ（日電）FACOM230シリーズ（富士通），HITAC8000シリーズ（日立），MELCOM3100シリーズ（三菱），OUK9000シリーズ（沖ユニバック）が発表され生産に入っている。

このような方向は，計算機の大量生産を容易にするもので，さらに価格対性能化（Performance per Cost）の大幅な向上にもつながるものである。

アメリカ市場におけるソフトウェアとハードウェアの比重



注：コンピュータ投資総額（価格）を100%としてそのうちに占める両者の比率
出所：Electronics News, 1966年11月2日号

表3により電子計算機システムのコストの配分をみると，今後は本体のシステム全体に占めるコストの割合は漸次低下し，その反面，入出力装置，ディジタル・ファイル，イメージ・ファイル，データ伝送などの占める割合が相対的に増加することを示している。さらに，1963年の計算機システムの構成と1972年におけるそれとは，相当異なることを物語るものである。つまり，それぞれの機能をもつ機器（入出力機器，中央処理装置，記憶装置，データ伝送装置など）を要求に適合できるよう自由に組わせてシステムが合成でき，応用面に適応性をもつ性格を強めつつある。

1.4.3 新しいプログラム言語

ソフトウェアの領域においても重要な進歩がみられる。これは，ビジネス用にはCOBOLとか科学計算用にはFORTRAN, ALGOL とか，異なる仕事のパターンに応じて，異なったプログラミング・ランゲージを使用する困難性を取り除くためのものである。仕事のいかんを問わず，また，プログラマが特定の計算機の性能に関係なくその仕事をプログラムできるように，機械から独立した言語としての特長が現われてきた。

電子計算機を有効に利用するためには，数多くのソフトウェアを整備する必要があり，ソフトウェアの開発は，電子計算機産業育成のために不可欠な因子である。ところが，ソフトウェアの開発は，これに携わる人の質と数に負うところが大きく，さらに莫大な費用を必要とするもので，電子計算機開発におけるこれらの比重は今後ますます増大する傾向にある。

1.5 電子計算機産業の課題

経済規模の拡大に伴なう産業活動の活発化，各種管理事務の複雑化，ならびに適用分野の多様化と相まって，電子計算機の需要は今後とも増大を続けるものとみられる。

このような需要の動向に対応して，電子計算機技術も急速な進歩を遂げ，広い用途に応じうるような融通性と普遍性をもち，しかも，同一プログラム方式の一連の機種の開発をねらいにしたシステムの設計が急速に普及するであろう。

わが国の電子計算機業界が，このような情勢のなかで企業活動を続けるためには，非自由化品目として輸入数量制限などによる保護育成体制の存続されている間に，十分な技術開発力を形成し，企業基盤の強化をはかることが不可欠である。つまり機種交代と技術革新という両面の技術的激動期に直面して，これに必要な研究開発投資は膨大となり，しかも技術的陳腐化の速度と相まって，その回収期間はますます短縮せざるをえなくなるが，電子計算機業界はこれに耐えるだけの力を持たなければならない。

2. 電子計算システム

2.1 データ処理システムについて

2.1.1 データ処理の考え方

電子計算機によるデータ処理は，必要なときに，要求に合った情報を，必要とする人に提供する手段である。しかし，その前提として，電子計算機システムの採用においては，マネージメントの要請に焦点を合せ，その目的をはっきりさせておく必要があり，これにもとづいて，処理システム――提供する情報（report），処理速度（dateの収集も含めて），処理サイクル，その他手続など――が決められる。

① 電子計算システムが提供する情報

従来それが作られていたからという理由だけで，今後もそれを続けなければならないというものではない。何のために必要か，どのような情報を盛りこむべきか，つまり，どのような情報を提供すれば，マネージメントの要請に合うかを充分検討する。むやみにリポートを作るのがよいことではない。

② インプット・データ

どのようなデータをどの時点で集めるかは報告書の内容によって決められるものである。しかし，基本的な業務活動を混乱させるような収集方法はさけるべきである。

電子計算機によるデータ処理システムはインプットにはじまりインプットに終るとまでいわれている。

③ 処理速度

一般的にいって，処理速度（応答時間）を短かくすると費用が増加するから，適用業務の要求に適したシステムを設計することが重要である。

つまり，スピードが要求される場合はハイ・スピードで処理できるようなシステム・デザインが必要であるが不必要な速さを要求して多くの費用をかけるのは無意味である。

④ 処理サイクル

データ処理のサイクルを管理サイクルと一致させることも重要である。たとえば，在庫記録の更新は，毎日 up to date に行なうことも可能であるが，更新の頻度の決定は，電子計算機の能力によるのではなく，管理の目的から即時（real time），日単位あるいは週単位というように決めるべきである。

2.1.2 一括処理と実時間処理

処理システムを大別すると，一括処理（batch process）と実時間処理（real time process）に分けることができる。

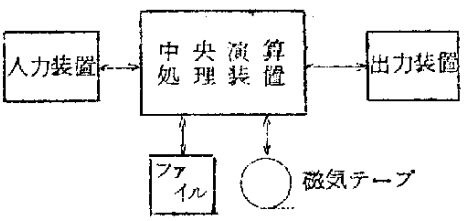


図2 Batch Process のマシンシステム

① 一括処理（Batch Process）

生産管理，販売管理，購売事務，給与計算，料金計算，経理，各種統計，技術計算などにみられる最も一般的な処理方法で，データをある周期でまとめて計算機に投入し，一括して処理する。

② 実時間処理（real time Process）

座席予約，バンキング業務などにみられるシステムで，応答に即時性が要求される場合に使用される。ただし，要求される応答速度は業務によって異なる。

通常の場合，real time process はデータの発生場所と電子計算機の処理装置が直結され，on line で情報の受授が行なわれる。これが on line real time process である。

現在の電子計算機は標準化されたそれぞれの機能をもつ機器の組合せにより，要求される機能に適合できるよう組合せて，機械システムが合成でき，応用面に適応性をもつ性格をつよめつつある。また，時分割方式，大容量のファイル，より速い処理能力，man machine Comunication system などの発展によって，今まで，即時把握の困難な情報の処理が Batch Process に割りこむといった方法で，この二つの process は融合される方向に進むであろう。

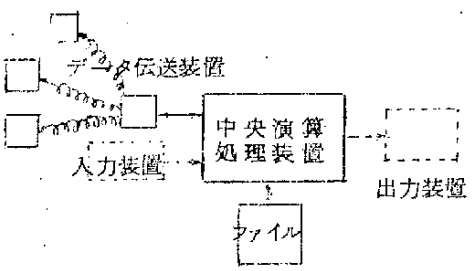


図3 on line real timc process のマシンシステム

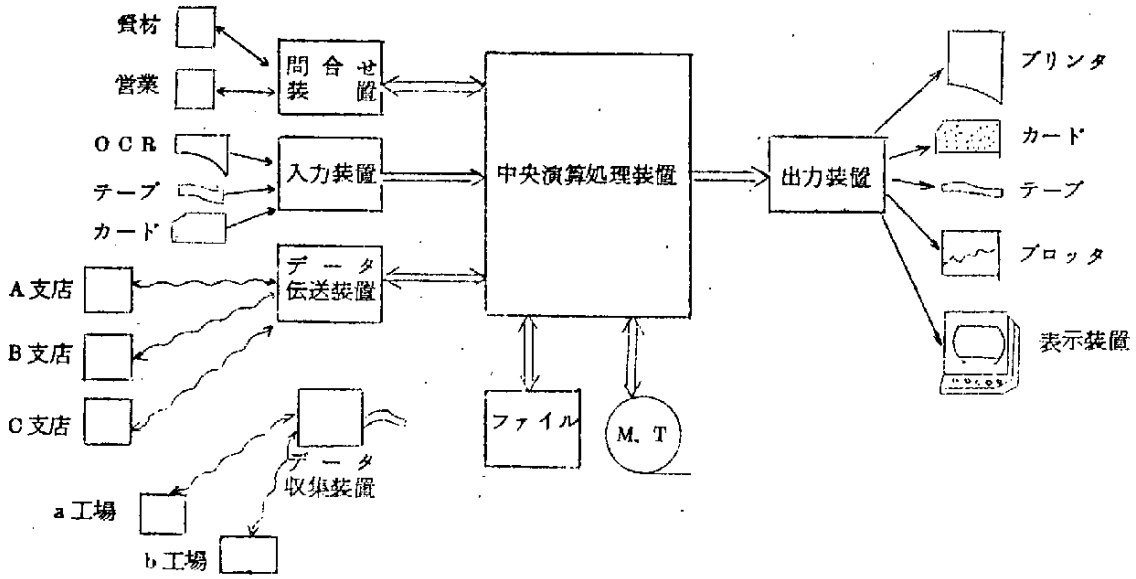


図 4　応用面に対する適応性をつよめつつあるマシンシステム

2. 2　電子計算機の構成

電子計算機は入力装置（input），記憶装置（memory），制御装置（control），演算装置（arithmetic）および出力装置（output）で構成されている。これらの装置の関係についてのべると，ある型式にととのえられた各種の情報（データとかプログラム）は入力装置を介して記憶装置に貯えられる。情報のうちプログラムは記憶装置から逐次，制御装置に送られて解読され，指令となって他の4つの装置を制御する。データは，制御装置からの指令によって，記憶装置から演算装置に送られる。このうちの必要な処理されたデータが，出力装置によってプリントされるか，またはカードとかテープなどに穿孔されるわけである。

ところで，電子計算機の改良，発展をみると，より大容量の記憶装置，より速い処理機能の開発に加えて，入出力，付属装置等についても，著しい進展が期待できる。

すなわち，磁気テープ装置の読込み，読出し時間のスピード化および大容量化から，更に入出力装置の高速化などがそれである。

図 5　電子計算機の基本的な構造

殊に最も顕著な業績は，大容量でスピードの速い，比較的安価なランダム・アクセスの記憶装置，同一型の標準活字ならば，活字をそのまま走査できる読取機などである。それらと同様に重要なものとして，データ収集装置，問合せ装置などがある。これらの発展は，今後ますます計算機の利用促進に寄与することであろう。

2. 2. 1　入出力装置

電子計算機はプログラムとかデータを，機械の中におさめてしまえば，あとは高速度で自動的に処理できるが，情報の出し入れに時間がかかったのでは計算機の能力を半減することになる。入出力装置はインプット，アウトプットすべき情報の媒体により異なり，表

4に示したようなものがある。

表4　入　出　力　装　置

装置の区分	装　置　名	速　　度	記　録　密　度
入力装置（in put）	カード読取機	200~1,600枚/分	80, 90字/枚
	紙テープ読取機（光電式）	200~1,000字/秒	10字/25mm
	電動タイプのキー	手　動	
	（磁気テープ装置）	10,000~150,000字/秒	100~1,000字/25mm
出力装置（out put）	カード穿孔機	100~500枚/分	80, 90字/枚
	紙テープ穿孔機	10~200字/秒	10字/25mm
	電動タイプライタ	10字/秒	
	ライン・プリンタ	200~1,500行/分	120字/行
	（磁気テープ装置）	10,000~150,000字/秒	100~1,000字/25mm

これ以外にも，マグネティックインクによって文字を記録し，それを読みとる装置とかタイプされた文字を光学的に読みとる装置など特殊なものがあるが，ここでは省略して，ごく一般的なものをあげた。どのような入出力装置も機械的な動きがともなうために，中央処理装置の処理速度とはバランスしないのが普通であるが，入力と出力装置によって，システム全体の速度の低下を避けるためにいろいろな工夫がされてきた。その1つに，大量のデータのインプットとかアウトプットには，できるだけ磁気テープを使う方法がある。磁気テープはインプット，アウトプットの手段としては最高の媒体で他の媒体（カードとか紙テープなど）を使う場合に比較してシステム全体の速度を上げることができる。他の方法としては入出力機構と中央処理機構との同時操作がある。これは入出力と記憶装置の間に緩衝用の記憶装置を置き，記憶装置は直接入出力装置とデータのやりとりをしないで，この緩衝用記憶装置（buffer storage）を通して行なう。緩衝用記憶装置と記憶装置の間のデータの移動は，電子的な速度であるから，直接入出力装置と往復するよりよりもはるかに短時間ですみ，入出力装置と緩衝用装置との間のデータの移動は演算に並行して行なわれる。

2.2.2　記　憶　装　置

入力装置で読んだ情報を記憶して，その情報のうち指令となるべきプログラムを制御装置に供給するとともに必要なデータを演算装置に伝達して，その演算結果を以後の計算のために，あるいは出力装置に送り出すまで一時的に貯える装置である。記憶容量は装置を大きくすれば増加するが，一般に容量が大きくなれば，記憶内容を取り出したり記憶させるために書込んだりするのに時間がかかる。この時間を待ち合せ時間（access time）といい，その大小によって高速記憶装置と低速記憶装置に分けることができる。現在最も多く使用されている記憶装置には，磁気コアー（magnetic core），磁気ドラム（magnetic drum），磁気ディスク（magnetic disk），および磁気テープ（magnetic tape）がある。アクセスタイムは磁気コアーが最も短かく，ここにあげた順序にしたがって長くなっている。記憶容量はこの逆で，磁気テープが最も大である。ただし，これらの順序は常識的なものであって必らずしもこのとおりであるとは限らない。記憶装置はその機能に

よって，主記憶装置と補助記憶装置がある。前者には磁気コア，高速磁気ドラム（容量の小さいもの）が後者には低速磁気ドラム（容量の大きいもの），磁気ディスク，磁気テープが使用されている。

第5　記憶装置比較表

記憶のタイプ		計算機の大きさ	大体の容量（桁）	平均アクセスタイム
主記憶	磁気コア	小形	2千～ 2万	½～10μs
	磁性薄膜メモリ	中形	2万～ 10万	
		大形	10万～100万	
補助記憶	磁気ドラム	（ある計算機の例）	2,400万（1台）	35～375ms
	磁気ディスク		*1,200万（1台）	70～850ms
	磁気テープ		500万～2,000万（1巻）	

＊1台で5,000万桁のものもある。

補助記憶装置には単なる補助的な役目をするものと，ファイルの役目をはたす2種類に分けることができる。ファイルとして使用される場合は，特に大容量のものが望まれ，一つの装置で数百万から数千万桁も記憶できる。計算機の規模によって相違はあるが，このような記憶装置を数台から十数台連動可能なものがある。現在，ファイルに使用されているものは，磁気テープ，磁気ディスク，大容量の磁気ドラムがある。通常，磁気テープは逐次呼出式であり，あとの2つ（これ以外にもあるが）は前者に比較して即時呼出式（ランダム・アクセス・メモリ）である。

最近では，新らしいタイプの記憶装置として，磁性薄膜メモリ（thin film memory）が開発され，アクセス・タイムも10億分の1秒（ナノセカンド）の単位にまで達している。記憶装置は語（word）とよばれる多くの区域に分かれていて，容量を表現する単位となる。おのおの語には10桁前後の数と＋－の符号を入れることができそれぞれ番地（address）がつけられている。たとえば2,000語の記憶装置であると0000～1999までの番地をもっていることになる。ところで1語の桁数を固定すると無駄なこともあるので，1桁を単位として，データの桁数によって任意の語の長さを決定する方法（バリアブル・レングス）もある。いずれにしても記憶装置と他の装置との情報のやりとりは，語を単位として行なわれ，一度記憶された情報は半永久的に消えないで，何度でも必要に応じて取出すことができる。しかし，ある情報を新たに書きこむと，以前のものは消失して新らしい情報に置き換わる。

2.2.3　制御と演算装置

制御装置は指令レジスタ（instruction register）と制御計数器（control counter）から成り，記憶装置に貯えられている情報のなかのプログラムから次々に命令を取出し，それを解読して，他の装置をコントロールする。制御計数器は，次に実行すべき命令が記憶されている番地を指定する。指令レジスタは，制御計数器によって指示された番地の内容を取出し，それを命令として実行する。

演算装置は制御装置からの指令によって働らく。最も基本的な演算装置は，累算器（accumulator），被乗数と除数レジスタ（multiplic and divisor register），乗数と商レジスタ（multiplier and quotient register）の3つのレジスタで構成されている。演算装置は記憶装置と情報のやりとりができるが，加減算の場合には累算器のみが使用され，乗除算では被乗数・除数レジスタ，乗数・商レジスタおよび累算器が用いられる。累算器は第1累算器（Acc Ⅰ）と第2累算器（Acc Ⅱ）に分けられて，それぞれ記憶装置1語分の情報を入れることができる。他の2つのレジスタも同じ容量である。しかしどの計算機も被乗数・除数レジスタと乗数・商レジスタをもっているとは限らず，累算器がその機能を兼ね備えていることもあるし，また，累算器が表面に現われていない計算機もある。

図6　制御装置と演算装置

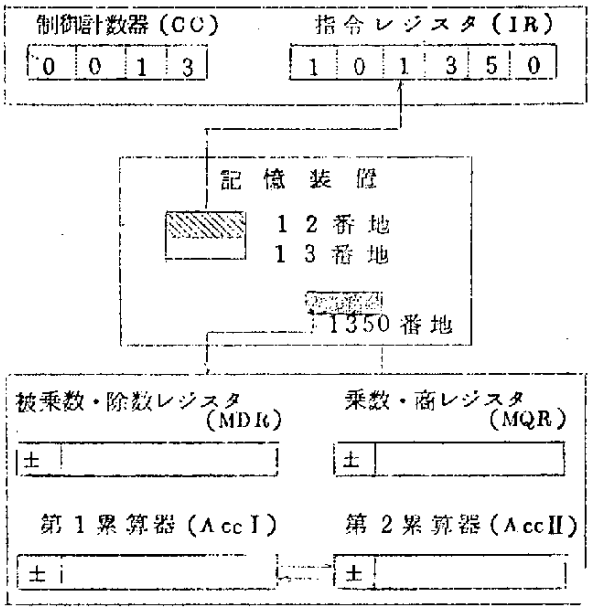


図6で制御計数器の示す数"0013"は次の命令が入って記憶装置の番地を指し，指令レジスタの内容はいま実行しつつある命令で，この命令の実行が終ると13番地の内容が指令レジスタに入る。このとき，制御計数器には自動的に1が加えられて"0014"となる。つぎはすでに指令レジスタにある命令が解読されて，それが実行される。このとき，制御計数器の内容はすでに"0014"となっているから；14番地の内容が指令レジスタに移されて，まえと同じような操作が繰返される。

表6　演算速度（1例）

演算の種類	計算機A	計算機B	計算機C
加減算	44μs	24μs	8μs
乗算	440μs	229μs	24μs
除算	1090μs	550μs	42μs

2.3　電子計算機の能力と限界

電子計算機は，膨大な量のデータを迅速・正確に処理するだけでなく，プログラムを与えておけば，その範囲内で，人間の行なってきた定型的な判断を代行する。

電子計算機のもつ機能を列記すると，つぎのとおりである。

① 処理手順（プログラム）を機械自身で記憶できる。

② 処理すべきデータや結果を記憶できる。

③ プログラムにしたがって自動的にデータを処理する。

④ 高速で，しかも正確である。

⑤ 定型的な判断業務が可能である。

⑥　融通性がある。

これらの機能をうまく発揮させれば，一般的には，①データーの処理時間が大幅に短縮され，正確になるとともに，②データー処理の多角性が大となり，一つのデータをさまざまに利用できるほか，関連するデータの相関々係など，各種各様の処理操作が一つのシステム内で容易になり，③外部条件に対する応答性が改善され，フィード・バックが速くなる。

以上のように，電子計算機はすばらしい能力をもっている。しかし，本当の意味では，まだ「考える機械」にはなっていない。人間が適切に機械をセットし，正しいデータとプログラムを与えたときに，はじめてすばらしい能力を発揮する。機械そのものが自分で勝手に理論づけたり，独自の判断をしたり，直観的な結論を下したり，プログラムされた以上のことをやることはできない。したがって，その特質とするところは，プログラムによって明確な基準を与えれば，それにしたがって，自動的に，しかも高速で，多種多様な仕事ができるという融通性である。

この「明確な基準」の与え方が電子計算機の応用面における大きな課題であり，有効に作用するか否かは，この機械を中心としたデータ処理システム（ＥＤＰＳ）を設計し，運用する人間の力にかかっている。

2.3.1　高速である

電子計算機の特長の一つは高速性である。基本的な数値演算とか2数の比較などは秒当り数千回～数万回の速度で行なわれる。しかし，この速度は内部処理に関するもので，データを読込んだり結果を外に出す速度はそれほど速くない。データを計算機に読みとらせる時には入力装置を通して行なわれ，処理結果は中央演算処理装置から出力装置に送られるので，これらの速度が重大な問題となる。つまりデータの収集方式，入力装置，出力装置の速度が，ある程度まで全体のシステムの速度を定めるといってよい。

2.3.2　自動的である

もう一つの特長は，プログラムにしたがって自動的にデータを処理する能力である。しかし，計算機に与えるプログラムは詳細で，しかも，完全なものでなければならない。もし誤った手順を教えこまれても「考える力」がないため，その通りに実行して，自分でそれを訂正するようなことはない。プログラム（コーデング）をつくるにはまず与えられた問題を基本的な単純な操作に分解することが必要で，しかも計算機にわかる言葉，つまり命令コードに変えなければならない。プログラムをつくるには相当な労力を要するが，一たび完全なプログラムが与えられるならば，あとは人手をかけずに自動的に作業をすすめることができる。ところで実際の仕事においては，命令の数が数百～数千を越えるようなプログラムもまれではない。ところが，計算機の命令は，命令コードという特殊な言葉――通常，マシンランゲージ（機械語）とよばれている――で計算機に供給してやらねばならない。この機械語は人間の言葉とは似ても似つかないものであるから，プログラムをつくることは面当な作業であり，誤りを起しやすい作業である。これを解決するために自動プログラミング，アプリケーション・プログラム，シミュレーション用プログラム，ユーテリティ・プログラムなどが開発されている。

2.3.3 判断する能力がある

電子計算機には判断する能力がある。この典型的な例は，二つの数のどちらが大きいかを比較し，以後の処理について，いくつかの手順の中から一つを選ぶことである。これらの判断は全て人によって前もってプログラムされたものである。

2.3.4 融通性がある

他の特質は融通性である。プログラムを変えることによって，命令コードの組合せで表現される操作ならば，どの仕事にも向く融通性がある。しかし，この融通性は能率よく処理できるか否かの問題とは別である。能率の良し悪しについては，その業務の処理に適した性能と構成であるかどうかによって相異がでてくる。

電子計算機を採用するには，その能力と限界をよく検討して，業務に適合した装置を選ぶ必要があることは当然である。

2.4 EDPシステムの方向

2.4.1 量から質への移行

事務の能率化は，電子計算機の利用によって著しい進歩をみせているが，最初の段階では，基本的な業務活動に包含される作業事務を中心に導入される傾向がある。これは，日常の基本的な業務活動を遂行する上で，大量のデータを速く正確に処理することが要求されており，人手ではとうていさばききれないからである。

このような適用方法は，全体計画をたてることなく，当面する対象業務を個々にとり上げるため，比較的短期間にEDP化を進めることができる。

ところで，経営におけるEDPシステム本来の目標は，経営の意志決定に寄与するマネジメント・インフォーメーションの提供にあるといわれているが，①そのもとになるデータ処理の合理化なくして意志決定資料の提供はあり得ない。しかし，②大量事務のEDP化のみに電子計算機導入の効果を期待するならば，事務処理の高速化，正確性，人員の増加の防止に効果は現われるとしても，一部の企業を除外して，EDP導入の費用に見合った効果を得ることは相当困難であろう。したがって，③経営のEDP化にあたっては，大量の事務データ処理を基礎とした量的側面の解決から質への移行を期待している。

量・質両面へのEDPの適用は，情報の発生場所でとらえた正確なままのデータを材料に，関連する業務の一貫処理を可能にする総合システムの設計が必要である。

2.4.2 総合処理システム

企業の成長にともなって，informationの量はますます多くなり，それに加えて人手中心の事務処理では多くの無駄がある。たとえば，各部門で似かよった資料を重複して作るとか，本来は必要でないにもかかわらず，それを作らなければ目的とする結果が得られないといった中間的な帳票がある。また，管理用の資料にしても，その内容について充分検討せず，ただ慣習にしたがって，業務活動の結果を個々に集計したにすぎないものもある。さらに，その資料が必要なときに間に合はないような場合，過去にこういったことがあったということを知るに止まって，それによって適切な処置を講ずることはできない。

「今までの報告書は，取引の結果つまり過去の状態を示すものであったが，これからは

過去の結果とともに将来の方向を指し示すものでなければならない」といわれている。これは，経営管理者にとって，将来の方向をどのように決定するかが最大の関心事であることをあらわしている。

企業の規模が大きくなり，企業間の競争が激しくなればなるほど，企業発展のためには科学的な経営が要求される。企業内の事務処理を簡素化し，能率を向上させるとともに，経営管理面の効果を増大させるためには，電子計算機の処理能力を背景として，例外管理の考えをとり入れた総合処理システムの確立が望ましい。

One job one Record.
From order entry to balancesheet.
Report for exception basis.

電子計算機による総合処理システムによってつぎのことが可能となる。

① データの重複や欠如が避けられる。

② 実質的にはデータ量が減少し，中間的な処理が排除され，大幅に事務処理が簡素化される。

③ 関連する業務のデータが一括して計算機の中に貯蔵されるので多角的な利用が可能となり，経営管理上必要なinformationをそれぞれのデシジョンポイントに適時フィード・バックできる。

④ 業務の進行に応じて，常に計画と実績の比較が行なわれ，状態と傾向，目標に対する達成状況が明確になり割当てられた責任が効果的に実行されているかどうかの監視とコントロール，さらに時機に応じたダイナミックな計画が可能となる。

⑤ 例外管理の導入が容易となる。

例外管理というのは，計画値と実際活動のinformationを対比することによって，決められた基準から外れたもの，たとえば，資材の最低在庫基準を割って発注を要するもの，販売達成率の低いもの，その他計画面，実施面に何らかの手をうつ必要のある事項のみを報告する方法である。この例外管理の方法を採用すれば，経営者とか管理者は，貴重な時間の大部分を問題解決のために使うことができ，問題をさがしだすのに時間を費さなくて済むようになり，定型的な業務から解放され，より重要な管理業務に専念できる。

しかし，このようなシステムを有効に運用するには，業務活動のinformationを迅速，正確に収集するだけでなく，計画そのものが信頼できるもので，しかもup to dateでなければならない。

2.4.3 総合処理システムの一例

商品販売活動に例をとると，毎日の受注，販売，受注，仕入など販売活動に関連するデータを，その発生場所から直接収集して電子計算機に投入する。もちろん販売活動から発生した原始データのみでなく，販売計画高とか商品の在庫基準，仕入基準などを与えることによって，それぞれのデータが多角的に生かされ，さまざまな相関（状態と傾向，目標に対する達成状況）などをみることができる。

① 受注，販売，発注，仕入，受注残，発注残，在庫など実績の迅速正確な把握。

② 販売実績にもとづき，必要な時点で自動的に請求書の発行が可能。

③ 販売（出荷），受注残，在庫量，在庫基準，仕入基準の総合により，適確な発注指示。

④ 売掛金の回収情報との関連で，売掛金の回収促進および売掛金の管理。

⑤ 実績にもとづき，経理情報の提供が可能。

⑥ 販売実績の分析，販売計画との相関により迅速で，適切な販売の管理。

⑦ 業務活動の実績と外部情報により，時機に応じたダイナミックな計画が可能。

以上に別記したように，総合処理システムは，受注，販売事務，発注，仕入事務，在庫管理，請求書発行事務，売掛金管理，販売，仕入関係の経理などが一貫処理され，さらに基準から外れたもの，つまり管理のポイントがタイムリーに把握できるため，偏在在庫とか品切れによる機会損失を未然に防ぐことができる。

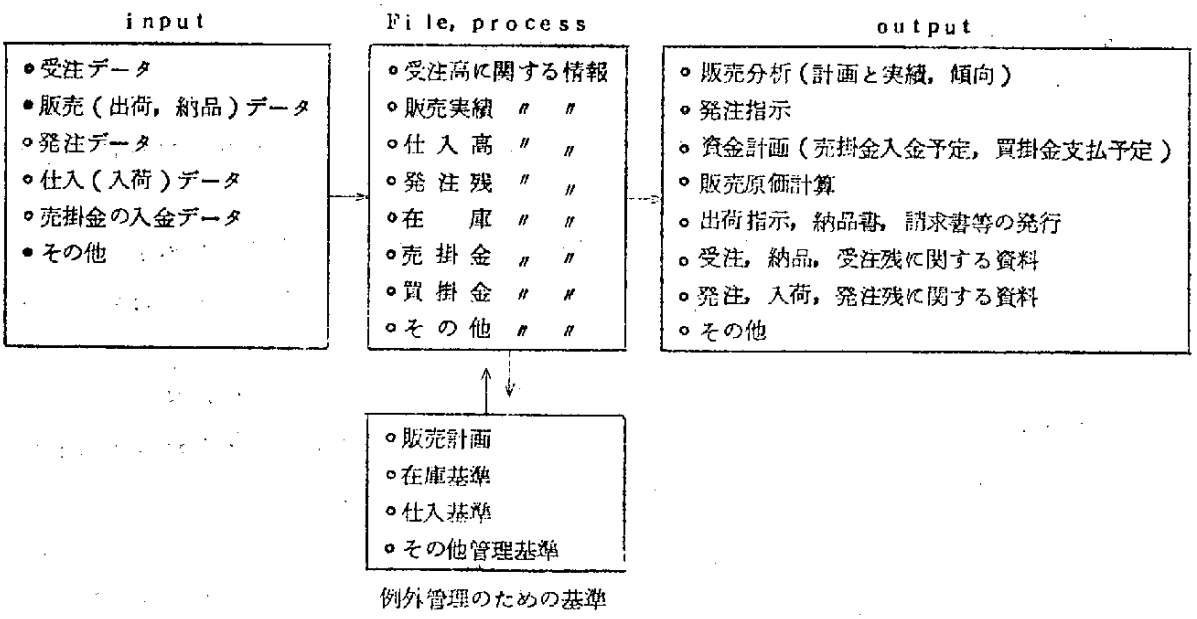


図7　販売におけるデータ処理システム

2.5　システム設計のすすめ方

2.5.1　社内の空気づくり

ＥＤＰシステムは特定の作業のみを対象としたものではなく，全社的な問題である。そのためにＥＤＰ担当部門のみで計画をすすめるのはむずかしいことである。もし，よい計画ができたとしても関係部門では新らしいシステムへ移行するために必要な知識もないし，心構えもできていない。このようにして出来た計画を実施に移すには多くの障害がある。このようなことから，担当部門は，経営者の理解と関係部門の積極的な協力を得るためのＰＲと社内教育を行なうとともに，場合によっては関係部門の長による委員会の設置も必要であろう。

2.5.2　目的をハッキリさせる

そのシステムにはどのような使命と役割があるのか，なぜそのシステムが必要なのかといったポリシーが確立されなければならない。

システム設計に入るためには，設計すべきシステムに与えられている目的を解明するとともに，対象業務の範囲もはっきりさせておく必要がある。長い時間と貴重な労力をかけても使命をはたしていなければ無意味な仕事をしていることになる。その意味では，この段階は以後のシステム設計のすべての前提となるものである。

一般に目的としては，二つの性格のものが存在しなければならない。つまり，このシステムによって成し遂げなければならない結果の状態と，課せられた達成目標を実現するためにどのような過程を選択すべきかを検討するに十分な条件である。

2.5.3 システムの構成

電子計算機の利用効果はシステム設計の良し悪しにかかっている。もし電子計算機が効果的に活用されていないとすれば，それはシステム設計に原因がある。

システム設計の手順は固定しているものではない。目的，対象範囲のそれぞれの立場や条件によって，いろいろな手順が考えられる。しかし，これらを通してシステム設計には一つの段階があると考えられる。

① 現状調査

目標達成に必要な要素，基準となる数量関係，それらの要素相互間の順序関係および処理上の問題点，例外処理，各単位業務間の関係について現状を調査し，対象とする業務の輪郭を明らかにする。

② 基本構想の設定

目標を達成するために必要な諸機能を摘出し，それらの機能相互間の連結状態を設定するとともに，機能の発揮を分担する要素（人間 ── 機械）と，その場所を指定するとともに合理的な相互間の関連を設定する。さらに，機能，要素，場所などの特性に基づいて組織体系における位置を指定し，人間相互間における責任権限と状報伝達の経過を示す。

③ 詳細設計

システム設計の第3段階は，前段階で構成された基本構想を，いろいろ変化する各環境条件の下で，好ましいと評価される方法で作動させるための詳細な計画であり，プログラムにつながるものである。

作成資料の内容とフォームの設計，データの収集方法，投入形式，伝票・コードの設計，ファイルの設計に基づいてプログラムが作られる。また，機械処理部分のみでなく手作業部分の業務との関連を考えてその合理化を計る必要がある。

④ 手続書の作成と移行計画

最適のシステムが構成されたとしても，ルール通りにオペレーションが実施されなければ，システム設計の具体的な成果を最大限に発揮させることはできない。そこで，対象業務，作業方法，作業手順，所要時間などについて指示し指導するためのプロセジュアが必要になる。

プログラム作成が完了すれば，ＥＤＰへの移行が行なわれるわけであるが，そのため

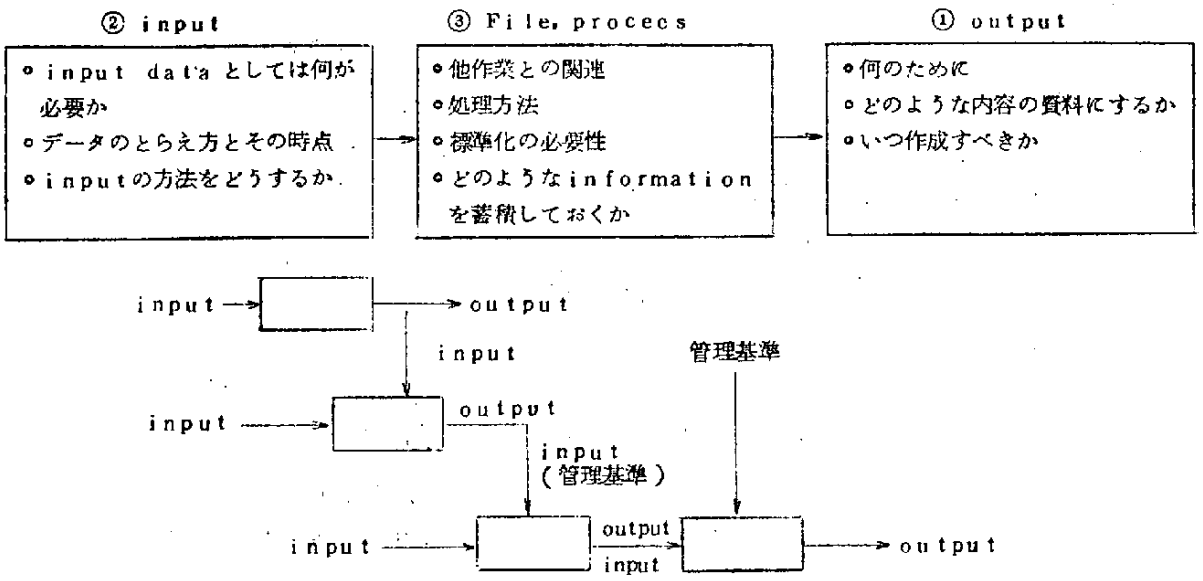


図 8　機能相互間の連結

にはファイルの切換えテスト・ラン，現場の訓練，機械処理に伴う諸規定類の改定，どういう順序で切換えていくかのタイム・スケジュール，人員配置などを事前に計画しておく。

2. 5. 4　設備計画

設備計画は電子計算機，付属機器の選定発注，設置場所，電源とか空調などの設備，什器・備品などその範囲は広いが，最もむずかしいのが計算機の機種決定である。

システムを事前に設計することによって，どの程度の大きさの計算機でよいか，またその構成をどうすればよいかが決められる。

米国の政府では「電子計算機の選定と導入はシステムの明細に基づいて行なわれなければならない」というポリシーがある。

2. 5. 5　組織と要員計画

これは簡単にいえば，ＥＤＰシステムをどのように運営するかの問題で，企業内における位置づけと推進母体を何にするか。また，電子計算機を中心とした新らしいシステムをうまく管理し，動かしていくためには，要員は何人位必要で，どのような順序で教育をすすめればよいかといったような問題である。

電子計算機の利用効果はシステム・アナリシス，システム・デザイン，プログラミング，オペレーションに左右されるが，これは電子計算機要員の質によって決定づけられる。要員の育成を軽視すると，与えられた使命と役割をはたすことは困難であろう。

オリエンテーション

（禁無断転載）

発行者　財団法人　日本情報処理開発センター
東京都港区芝公園21号地1－5　機械振興会館
TEL(434)8211(代)

印刷所　有限会社　盛光印刷所
東京都千代田区飯田橋4の6の3
TEL(264)1851(代)