電子計算機の情報処理

京都大学工学部
数理工学教室

萩原

宏*

I 緒言

1. 計算機

昔から実際のいろいろな分野において、計算という仕事はかせないものであり、そのために、大変な労力を費している。そこで、古くからいろいろな計算機械が考案されて、各方面で使われている。すなわち、手近なところでは、ソロバンや計算尺から卓上計算器などがあり、また、パンチカードを使った統計機、会計機あるいは、微分方程式を解くための微分解析機などから、更に、ディジタル型のいわゆる電子計算機にいたるまで、さまざまな種類のものがある。

これらの計算機をその性質から分類すると2つに大別される。その1つはソロバンや卓上計算器のように、数をすべてある単位の整数倍として表現するディジタル型のものであり、他は計算尺のように数を連続的な物理量を使って表わすアナログ型である。アナログ型のものは一般に計算速度は速いが、精度が限られ、精度を上げることがむづかしいのに対して、ディジタル型の場合には桁数さえ増せば精度を上げることは容易であるが、計算速度は速くない。とはいっても、今日の高速の電子計算機では、加減算の速さは $1\mu\text{S}$ 以下であるので、1秒間に百万回以上の加減算を行なうことができる。

2. 自動計算機

計算の手間を省くためには、複雑な一連の計算を連続して自動的に行なうことが望ましい。19世紀の中頃、英国のチャールス・バベッジはこういう自動計算機を考案し、機械的な装置でこれを実現しようとしたが、当時の技術水準ではこれを完成することが出来なかった。その後、1944年に至ってハーバード大学で、リレーを使ってはじめて大型の自動計算機が完成された。しかし、リレーを使っているので、計算速度は、余り速いものではなかった。その後、真空管、トランジス

タ等の電子的な素子が使われるようになって高速の電子計算機として発展して来たのである。

これらの計算機では、電気接点の断続、電圧の正負、あるいは電圧パルスの有無というような電氣的な2値変量を使って、数値を表現し、演算を行なわせているので、安定であり、誤りが起こり難いものとなっている。

また、一連の計算を自動的に行なわせるためには、予め個々の演算を行なわせるための演算指令（命令という）の系列（これをプログラムという）を作っておいて、このプログラムに従って自動的に計算を進めるようにしている。初期の自動計算機では、このプログラムを、パンチテープの上に順番にパンチしておき、これを読みとりながら順次実行して行くものや、配線盤の上に配線しておいてこれに従って順次実行していく方式をとっていたが、これでは、プログラムの変更、交換などに不便であるので、プログラムを構成する命令も数値と同じような形に表現して、計算機内部の記憶装置に格納しておき、このプログラムに従って計算機の制御が行なわれるプログラム内蔵方式が考え出された。この方式はきわめて融通性に富み、使用上も便利であるので、今日の電子計算機は、特殊なものを除いて、すべてこの方式を採用している。

このプログラム内蔵方式によって、電子計算機は、単なる計算機械から、無限の可能性をもった万能機械、人工頭脳と呼ばれるに値する情報処理装置へと発展して来たといえる。

II 情報の表現

1. 情報の単位

われわれが物を考える場合、最も簡単な情報は二者択一的な情報である。たとえば、いま人について考える場合、日本人であるか外国人であるか、すなわち、国籍について考えて、その前者であるとの情報、あるいは男であるか女であるかすなわち性別について考えて、その前者であるとの情報のような、二者択一的なものである。

このような二者択一的な情報を情報の単位として考

Information Processing in Electronic Computer.

※ Hiroshi HAGIWARA : Department of Applied Mathematics and Physics, Kyoto University

えて、それをビット (bit) と呼んでいる。上の例で、日本人の女という情報が与えられれば、それは2ビットの情報と考えることが出来る。

二者択一的な情報に対して、その一方に0、他方に1という記号を割りあてて表現することが出来る。たとえば、上の例で

日本人………0 男………0
外国人………1 女………1

というように割りあてて、国籍、性別の順に0、または1を並べて書くことにすると、01という組み合わせによって日本人の女という情報を表現することができる。10であれば外国人の男という情報を表現している。このような、0、1の組み合わせを2進符号という。すなわち、上の例のように2ビットの情報は2桁の2進符号で表わすことができ、それは $2^2=4$ ケの中の1つを指定する情報である。同様に、一般に n ビットの情報は n 桁の2進符号で表わすことができ、それは 2^n ケの中の1つを指定する情報である。また、このことから2進符号の桁数のことをビット数ということもある。

1ビットの情報をあらわす、0、1の記号を、電気接点の断続、電圧の正負あるいはパルスの有無というような電気的な2値変量に対応させれば、1つのこれらの電気信号で1ビットの情報を表わすことができる。 n ビットの情報は n 桁の2進符号で表わすことができるので、 n ケのこれらの電気信号の組み合わせによって、 n ビットの情報を表わすことができる。

以上は一般的な情報について考えたのであるが、数値についても全く同様に考えられる。すなわち、数を適当な2進符号で表わすことによって、これを電気信号の組み合わせで表現することができるわけである。

数を2進符号で表わすためには2進法を用いるのが都合がよいので、電子計算機で数を取り扱う際には2進法を用いることが多い。もちろん、個々の数字を2進符号で表わして、その組み合わせで10進数を表わすこともある。

2. 数の表現と2進法

たとえば、われわれがふつうに

385

と数を書けば、これを三百八十五と読む。これは、このように書けば

$$3 \times 100 + 8 \times 10 + 5 = 3 \times 10^2 + 8 \times 10 + 5$$

を表わすという約束があるからである。慣用の10進法では10という数が記数法の基数としての特別な意味を

もっている。理論上は、1より大きい任意の正の整数 r を記数法の基数にとることができる。すなわち、 r を定めたとき、任意の正の整数 N はただ一通りに

$$N = a_m \times r^m + a_{m-1} \times r^{m-1} + \dots + a_1 \times r + a_0 \quad (1)$$

と表示できる。ただし、ここで、 $a_m, a_{m-1}, \dots, a_1, a_0$ は、 $0, 1, \dots, r-1$ のいずれかである。これが r 進法である。そうして(1)を

$$N \sim a_m a_{m-1} \dots a_1 a_0 \quad (2)$$

と記すことにする。たとえば10進法の108を7進法で表わせば

$$\begin{aligned} 108 \div 7 &= 15 && \text{余り} 3, \\ 15 \div 7 &= 2 && \text{余り} 1, \\ 2 \div 7 &= 0 && \text{余り} 2, \end{aligned}$$

であるから

$$108(10\text{進法}) = 213(7\text{進法})$$

である。普通(2)の a_m は0でない数字とし、(2)を $m+1$ 桁の数という。

$r=2$ とすれば、(1)又は(2)の a_i は0か1かのいずれかになる。したがって、すべての整数が0、1の二つの記号の組み合わせで表わすことができる。たとえば、0から10までの整数を10進法と2進法で表わすと

10 進	2 進	のようになる。また 108 (10 進) を 2 進法で表わすと、
0	0	$108 \div 2 = 54$ 余り 0
1	1	$54 \div 2 = 27$ 余り 0
2	10	$27 \div 2 = 13$ 余り 1
3	11	$13 \div 2 = 6$ 余り 1
4	100	$6 \div 2 = 3$ 余り 0
5	101	$3 \div 2 = 1$ 余り 1
6	110	$1 \div 2 = 0$ 余り 1
7	111	
8	1000	であるから
9	1001	$108(10\text{進法}) = 1101100$
10	1010	(2進法) となる。

これらの例からもわかるように数を紙の上に書くのには、2進法では数の桁数が多くなってあまり便利ではない。そのために、 $2^3=8$ 進法あるいは $2^4=16$ 進法がよく使われる。これらを使えば2進法3桁あるいは4桁が1桁で表わされるので、10進法とあまり桁数は変わらず、しかも2進法との変換が簡単である。しかし、16進法を用いると10進数の10、11、……、15に相当する6つの記号を0、1、……、9以外の数字として追加しなければならない。このような記号を普通の数字よりも範囲の広い大型の数という意味で King-size number

ということがある。

これまでは整数について述べて来たが、 r 進法の小数も原理は同じことである。いま、10進法で

$$0.924$$

と書けば、これは

$$\frac{9}{10} + \frac{2}{100} + \frac{4}{1000} = 9 \times 10^{-1} + 2 \times 10^{-2} + 4 \times 10^{-3}$$

を表わす。全く同様に $a_{-1}, a_{-2}, \dots$ を 0 から $r-1$ までの整数として、 r 進法で小数

$$0. a_{-1} a_{-2} \dots$$

と書けば、これは

$$a_{-1} \times r^{-1} + a_{-2} \times r^{-2} + \dots$$

を表わす。たとえば、2進法で0.01と書けば、これは $1/4$ すなわち10進法の0.25を表わす。

3. 負数の表示法

これまでは正の数だけを扱って来たが、計算にあたっては負の数も正の数と同じように扱えなければならない。

負の数の表示は、絶対値表示方式と補数表示方式の2つの方式が考えられる。

絶対値表示方式は日常の数の書き方と同じく、+、-の符号を数の大きさを表わす数値につけて表わす方式で、例えば、頭初のビットを符号ビットとし、それが0のとき+、1のとき-とし、他の桁の表わす数を、その数の絶対値として、2進小数ならば

$$0.0110 \text{ は } +6/16$$

$$1.0101 \text{ は } -5/16$$

を表わす。

補数表示は、取扱う数の範囲が制限されているときに、補数を使って負の数を表わす方式である。たとえば10進小数について考えれば、負の数-0.2630を表わすのに、10の補数、すなわち

$$10 - 0.2630 = 9.7370$$

を用いる方法である。2進数の場合には

$$-0.111 = -7/8$$

を表わすのに、2の補数、すなわち $2 = 10$ (2進法)に加えて得る

$$10 - 0.111 = 1.001$$

を用いるのである。

上に述べた補数表示のほかに、9の補数(10進法の場合)、1の補数(2進法の場合)と呼ばれる方法がある。これは、たとえば

$$9.99\dots 9 \text{ あるいは } 1.11\dots 1$$

のように小数点以下に考えている負の数の桁数と同じ

だけ9あるいは1を並べた数に考えている負の数を加えて得る数を用いる方法である。すなわち-0.2630に対しては

$$9.9999 - 0.2630 = 9.7369$$

を、 $-0.111 = -7/8$ に対しては

$$1.111 - 0.111 = 1.000$$

を用いるのである。

これらの負数の表示方法はそれぞれ一長一短があるので、計算機を設計する際にどの長所を生かすかによって、これらの方法のいずれかが用いられる。

4. 数字の表現

既に述べたように情報を電気信号で表わすためには、それを2進符号で表現することが必要である。

0~9の10進数字を2進符号で表現するためには、少くとも4ビット必要である。4ビットの符号は $2^4 = 16$ 通りあるので、それを0~9に割り当てるのには色々な方法が考えられる。その二、三の例を次の第1表にかかげる。

第1表 10進数字の符号化

	2進10進 符 号	5-4-2-1 符 号	2-4-2-1 符 号	3あまり 符 号	5中2 符 号
	8 4 2 1	5 4 2 1	2 4 2 1		7 4 2 1 0
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1	1 1 0 0 0
1	0 0 0 1	0 0 0 1	0 0 0 1	0 1 0 0	0 0 0 1 1
2	0 0 1 0	0 0 1 0	0 0 1 0	0 1 0 1	0 0 1 0 1
3	0 0 1 1	0 0 1 1	0 0 1 1	0 1 1 0	0 0 1 1 0
4	0 1 0 0	0 1 0 0	0 1 0 0	0 1 1 1	0 1 0 0 1
5	0 1 0 1	1 0 0 0	1 0 1 1	1 0 0 0	0 1 0 1 0
6	0 1 1 0	1 0 0 1	1 1 0 0	1 0 0 1	0 1 1 0 0
7	0 1 1 1	1 0 1 0	1 1 0 1	1 0 1 0	1 0 0 0 1
8	1 0 0 0	1 0 1 1	1 1 1 0	1 0 1 1	1 0 0 1 0
9	1 0 0 1	1 1 0 0	1 1 1 1	1 1 0 0	1 0 1 0 0

計算機では情報の伝送時に誤りが生ずることは甚だ困ることであるので、もし誤りが生じたときにはこれを検出することが望ましい。そのために符号に誤り検出のために余分のビットをつけ加えて、数ビットの符号のうち“1”であるビットの数を常に奇数(または偶数)になるように統一しておいて、これが偶数(または奇数)になったときは誤りが生じたとしてこれを検出するパリティ検査方式をとる場合がある。またこのように誤り検出のために特別のビットを加えずに、符

号自体を誤り検出可能なものになっている場合もある。たとえば上の表の5中2符号は、10進数字を表わすのに5ビットを使い、常にその中の2ビットが“1”になる組合わせでもって0から9までの数字を表わしている。

数字の場合と同じようにして文字、記号等も6ビットあるいは8ビットの2進符号を割り当てて表現することが多い。このように文字等を表わすいくつかのビットの集りをバイトと呼ぶことがある。1バイトが8ビットであると、文字を表わすときは1バイトで1文字を、数字だけを表わすときは1バイトで2数字を表わすことができ、2のべき乗であるのが、都合のよいことが多いので、最近よく使われるようになって来た。

Ⅲ 二進情報の論理操作

1. 論理数学

2進符号で表わされた情報の処理を考える場合には、0, 1という2つの値をとる2値変数(論理変数)を取り扱う論理数学あるいはブール代数と呼ばれる数学を利用するのが便利である。

(a) 基本演算

論理数学で用いられる演算を論理演算といっているが、最も広く用いられている基本演算は次の3つである。

(i) 論理積, (ii) 論理和, (iii) 否定

これについて順次説明しよう。

(i) 論理積 2変数の関数で、 $x \wedge y$ あるいは $x \cdot y$ で表わされ、 x および y の両方とも1のときだけ1, それ以外のときは0という値をとる。この関係を表にすると、第2表のようになる。

第2表 論理積の真理値表

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

論理変数や論理関数の値を真理値といい、第2表のように論理変数の真理値のすべての組合わせと、これに対応する論理関数の真理値を表にまとめたものを真理値表といっている。

(ii) 論理和 2変数の関数で、 $x \vee y$ あるいは $x + y$

で表わされ、 x, y の両方とも0のときだけ0, それ以外のとき、すなわち x あるいは y の一方あるいは両方が1のとき1という値をとる。その関係は、第3表の真理値表のようになる。

第3表 論理和の真理値表

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

(iii) 否定 1変数の関数で、 $\bar{x}$ あるいは $\neg x$ で表わされ、 x が0ならば $\bar{x}$ は1, x が1ならば $\bar{x}$ は0という値をとる。

(b) 基本公式

上に述べた3つの演算の間には次のような公式が成立する。これらの公式を用いることによって論理関数を変形したり簡単にしたりすることができる。

(i) 交換則 $A \vee B = B \vee A, A \wedge B = B \wedge A$

(ii) 結合則 $A \vee (B \vee C) = (A \vee B) \vee C,$
 $A \wedge (B \wedge C) = (A \wedge B) \wedge C$

(iii) 分配則 $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C),$
 $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$

(iv) 同一則 $A \vee A = A, A \wedge A = A$

(v) 吸収則 $1 \vee A = 1, 0 \wedge A = 0$

$0 \vee A = A, 1 \wedge A = A$

$A \vee (A \wedge B) = A, A \wedge (A \vee B) = A$

$A \vee (\bar{A} \wedge B) = A \vee B,$

$A \wedge (\bar{A} \vee B) = A \wedge B$

(vi) 相補性 $A \vee \bar{A} = 1, A \wedge \bar{A} = 0$

(vii) 多重否定 $(\bar{\bar{A}}) = A$

(viii) ド・モルガンの定理 $\overline{(A \vee B)} = \bar{A} \wedge \bar{B},$
 $\overline{(A \wedge B)} = \bar{A} \vee \bar{B}$

2. 論理回路

いま、電圧の正、負をそれぞれ論理変数の1, 0に対応させて、上に述べた基本論理演算を実現する電子回路(論理回路)の例をあげよう。

(i) 論理積回路(AND回路)

第1図に示したようなダイオードを使った回路を考えると、ダイオードは一方方向にのみ電流を通し、その抵抗は極めて低く、逆方向に電圧が加わった場合には電流を流さない性質があるので、 x, y が共に正電圧

第 1 図

第 2 図

であれば z は正電圧になるが、もし、 x が負になると、 $+$ の電源から電流が x の方向に向かって流れ抵抗 R の中での電圧降下で z は負になる。このとき y が正であっても、 y につながるダイオードには逆方向の電圧がかかることになって電流は流れない。同様に y が負で、 x が正であっても z は負になる。また、 x, y 共に負であれば両方のダイオードに電流が流れてやはり z は負になる。すなわち、 x, y が共に正すなわち 1 のときのみ z は正すなわち 1 になり、それ以外のときは z は負、すなわち 0 になる。このことは z が x, y の論理積になっていることを示している。そこでこの回路を論理積回路として用いることが出来る。また、論理積の意味から、この回路を AND 回路と呼ぶことも多く、これを記号的に表わす場合には第 1 図の下に示したような記号が使われる。

(ii) 論理和回路

第 2 図に示したようなダイオード回路では、 x あるいは y のいずれか一方あるいは両方が正電圧であれば、ダイオードを通して電流が流れて、抵抗中の電圧降下によって z は正になる。もし、 x, y が共に負電圧であれば、 z は負になる。すなわち、 x, y の両方が負すなわち 0 のときだけ z は負すなわち 0 になり、 x, y の一方あるいは両方が正すなわち 1 のとき z は正すなわち 1 になる。このことは z が、 x, y の論理和になっているので、この回路を論理和回路という。論理和回路は論理和の意味から OR 回路とも呼ばれ第 2 図の下に記号を使って表わされる。

第 3 図

(iii) 否定回路

否定回路はダイオードで構成することはできないので、第 3 図のようなトランジスタ回路を用いる。 x が正であればトランジスタに電流が流れないで、 z は負のある電圧になる。逆に x が負になるとトランジスタに電流が流れて、そのコレクタの電位は上り、したがって z は正のある電圧になる。すなわち、 x が正すなわち 1 であれば、 z は負すなわち 0 となり、 x が負すなわち 0 であれば、 z は正すなわち 1 となり、 z は x の否定となり、この回路で否定操作が行なわれる。否定の意味から、この否定回路は NOT 回路とも呼ばれ、

第3図の下の記事で表わされる。

3. 演算回路

算術演算には加減乗除の四則があるが、乗除算は加減算の繰返しによって行なうことができるので、加減算が基本となる。しかも減算は負数を加えることと等価であるので、演算回路の基本は加算回路である。

(a) 2進法加算器

まず2進法の加算について考えてみよう。2進1桁の数の加算は次の四つの場合がある。

$$\begin{array}{r} 0 \\ +) 0 \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \\ +) 1 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ +) 0 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ +) 1 \\ \hline 10 \end{array}$$

これを表の形にまとめると次のようになる。ここで、 S はその桁の和を、 C は次の桁への桁上げを示す。

第4表

x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

これを、論理変数、 x, y の論理関数、 S, C を表わす真理値表とみることができる。こう考えて、 S, C を x, y の関数として、式の形に書くと、

$$S = \bar{x} \wedge y \vee x \wedge \bar{y}$$

$$C = x \wedge y$$

となる。これを書き下すには、それぞれ、 S, C が1になっている、 x, y の組合わせをみて、その変数の値が0ならば、その変数の否定を、1ならばそのままとて、それらの論理積を作る。そうして、関数の値が1になっているすべての組合わせについて作った論理積の項の論理和として書き下せばよい。すなわち第4表の S についてみると、 $S=1$ になっているのは、 $x=0, y=1$ の組と、 $x=1, y=0$ の組になる。前者については $\bar{x} \wedge y$ を作り、後者については $x \wedge \bar{y}$ を作る。そうしてこれらの論理和として

$$S = \bar{x} \wedge y \vee x \wedge \bar{y}$$

を得る。

これらをそのまま論理回路に書き直すと、第4図のようになる。

第4図

第5図

第5表

x	y	w	S	C
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

また、上の S の式を、既に述べた論理数学の公式を使って変形すると

$$S = (x \vee y) \wedge (\overline{x \wedge y})$$

$$= (x \vee y) \wedge \bar{C}$$

となる。このように変形して、回路図にすると第5図のようになり、第4図にくらべて簡単になる。

これらの回路を半加算器といっているが、これでは、下位の桁からの桁上げがあった場合に、これを加えることができない。そこでこれも考慮して考えるために、下位からの桁上げを w として、真理値表の形にまとめると、第5表のようになる。

S, C を x, y, w の論理関数として書き下し、簡単化すると、

$$C = x \wedge y \vee x \wedge w \vee y \wedge w$$

$$S = (x \vee y \vee w) \wedge (\overline{x \wedge y \wedge w})$$

第 6 図

となる。

これを論理回路の形に書き上げると、第6図のような回路となり、これは全加算器と呼ばれ、2進数1桁の加算を行なうことができる。

(b) 10進法加算器

10進数は、既に述べたように、いろいろな2進符号で表わされるわけであるが、ここでは、第1表の左端に示した8-4-2-1符号で表わされた2進法10進符号について考えてみる。この符号で表わされた10進数1桁を x, y とし、各ビットを $x_8, x_4, x_2, x_1, y_8, y_4, y_2, y_1$ で表わすことにする。

$$\begin{array}{r}
 5 \cdots 0101 \\
 +) 3 \cdots 0011 \\
 \hline
 8 \cdots 1000
 \end{array}
 \quad
 \begin{array}{r}
 5 \cdots 0101 \\
 +) 7 \cdots 0111 \\
 \hline
 1:2 \cdots 1100 \\
 +) 0110 \\
 \hline
 1: 0010
 \end{array}$$

x と y の和が10以下のときは、左のようになって、各数字を表わす4ビットの符号を2進4桁の数とみなして、そのまま加え合わせることによって正しい和を得る。

$10 \leq x+y \leq 15$ のときは、左の例に示すように、2進4桁の数とみなして和を求めたのでは上の桁への桁上げが出ないと同時に、和の4ビットも正しい10進数の和を表わしていない。このためには $6 = (0110)$ を加えて補正を行なえば、正しい和を表わすと同時に上の桁への桁上げも得られる。

$$\begin{array}{r}
 8 \cdots 1000 \\
 +) 9 \cdots 1001 \\
 \hline
 1:7 \quad 1:0001 \\
 +) 0110 \\
 \hline
 1: 0111
 \end{array}$$

$x+y \geq 16$ のときには、上の例に示すように、2進4桁の数とみなして和を求めると、桁上げは生ずるが、和の4ビットは正しい10進数の和を表わしていない。この場合にも $6 = (0110)$ を加えて補正を行なえば正しい和を得ることができる。

このことから、 $10 \leq x+y \leq 15$ の場合、すなわち、第4ビットが1で同時に第2ビットまたは第3ビットが1である場合と、 $x+y \geq 16$ の場合、すなわち、第4ビットの加算器からの桁上げが出る場合には、上位の桁への桁上げを出すと同時に、 $6 = (0110)$ を加えてやる、すなわち、第2ビットと第3ビットに1を加えて補正をすることによって正しい10進数としての和を得る加算器が得られる。そこで、既に述べた半加算器(H. A.) 全加算器(F. A.) を用いてこの加算器を構成すれば、第7図が得られる。

下位の桁からの桁上げ

第 7 図

上位の桁への桁上げ

IV 電子計算機の構成と動作

1. 電子計算機の構成

計算機の主要部を大別すると第8図に示すように、入力装置、出力装置、記憶装置、演算装置および制御装置の5つになる。図において実線は数値や命令の道筋を示し、点線は命令を分解してできる制御信号およびそれに対する応答を示している。

第 8 図

(a) 処理の手順

計算機を使って問題を処理するためには、まずその

処理過程を分解して計算機の命令で書き表わしたプログラムを作ること（プログラミングという）が必要である。できたプログラムは入力装置に応じて紙テープあるいはカードのような適当な媒体にパンチされる。同時に計算に必要なデータも同じ媒体にパンチされる。ここまでは計算機とははなれて全く独立に準備される。計算に必要なプログラムとデータのテープあるいはカードが準備されると、入力装置を通して計算機内部に読込まれて、プログラムおよびデータは記憶装置に格納される。計算開始の信号が与えられると、制御装置はプログラムを構成する命令を一つずつ記憶装置より取り出して来て、これを解読し、記憶装置より必要なデータを取り出して、これを演算装置に送り込んで、命令の指定する演算を行なわせ、あるいは演算結果を記憶装置の中に格納させる。こうして、プログラムの命令を一つずつ実行して最後の結果が得られれば、これはやはりプログラムの指示に従って制御装置から出る信号によって、出力装置から紙の上にタイプされ、あるいはテープやカードにパンチされて外部に取出される。この間、計算機は記憶装置の中に蓄えられているプログラムに従って、全く自動的に制御されて、動作をするのである。

(b) 記憶装置

記憶装置は命令や数値などの情報を保持する装置であるが、情報の単位に応じて多数の場所に分けられ、情報の単位ごとに番号をつけ、これをアドレスあるいは番地と呼んでいる。情報の単位としては、ふつう、命令や数値を一定の長さの2進符号あるいは2進数で表わして、これを語と呼んでいる。そうして、命令を表わす語を命令語、数値を表わす語を数値語と呼び、それらの形式や長さに相関性をもたせて、両者をまったく同様に取扱えるようにしている。

たとえば4000語の記憶容量をもった記憶装置は4000ヶの場所に分けられていて、それぞれ0番地から3999番地までのアドレスがつけられている。計算機内で、命令や数値などの情報を指定するのは、このアドレスによって行なわれる。たとえば、いま100番地に数 x が、101番地に数 y が入って居り、その和を102番地に作るような場合には、100番地の内容と101番地の内容とを加えて、その結果を102番地に格納するというように指定する。

(c) 演算装置

演算装置は各種のレジスタと加算器等を中心とした演算回路などから成り立っている。レジスタは、演算数あるいは演算結果等を一時的に保持する役目をする

ものであり、加算器等の回路と共に、制御装置から出される信号に従って各種の演算操作を行なうものである。

(d) 制御装置

制御装置には記憶装置から取り出された命令を保持する命令レジスタ(OR)、その命令の入っている番地を保持する逐次制御カウンタ(SCC)、命令の解読のためのデコーダその他各種の制御信号の発生回路等があり、プログラムに従って計算機全体の制御を行なう部分である。

(e) 入出力装置

入力装置は機械の外部の情報を内部に読み込むためのもので、紙テープ、カード等の入力媒体を使うものが多く、高速のものは光を使って光電素子によって電気信号に変換している。

また、出力装置は機械内部の情報を外部に取り出すためのもので、紙の上に文字の形にプリントするプリンタおよび、紙テープあるいはカードにパンチするもの等がふつう用いられており、いずれも現在のところ機械的な動作をするもので余り高速ではない。また、ブラウン管の上に文字あるいは図形を表示する表示装置も目的によっては非常に便利であるので最近使われるようになって来た。

2. 電子計算機の動作

計算機は既に述べたように記憶装置に格納されているプログラムに従って全く自動的に順次演算を実行して行くのであるが、そのときの情報の流れを簡単化した計算機を例にとって説明しよう。

まず、この計算機の語の構成を10進6桁の数字と正負の符号からなるものとし、命令語は第9図に示すように、正負の符号を除いて、上位2桁が指令部、下位4桁がアドレス部とする。命令レジスタ(OR)もこの命令語の構成と同じ形になっていて、命令語のそれぞれの部分を保持するものとする。

いま、記憶装置の中に、213番地より順に第6表に示すような命令の系列すなわちプログラムが入っており、550番地以降にはそれぞれ数値が入っているものとする。

第9図

第 6 表

番 地	記 憶 内 容
.....	
213	+150561
214	+110562
215	+100550
216	+820220
217	
.....	
220	+150563
.....	
.....	
550	+688979
.....	
.....	
561	+213814
562	+374062
.....	

第 7 表

命 令	操 作 内 容
15	アドレスで指定された数値を ACC に入れる
11	アドレスで指定された数値を ACC の内容に加える。
10	アドレスで指定された数値を ACC の内容から引く。
82	もし、ACC の内容が負であれば、次にはこの命令のアドレスで指定される番地の命令を実行し、ACC の内容が負でなければ、この命令の入っている番地の次の番地の命令を実行する。

ここに現われる命令は第7表のような操作を行なうものとする。ただし ACC は演算装置内の累算器と呼ばれるレジスタである。

このプログラムの実行に関係のある計算機の部分とその間の関係を第10図に、また各部分の間の情報の流れあるいはその内容の変化の様態を第11図に示す。た

第 10 図

だし、ここで実線は数値の道筋を、点線は制御信号の道筋をそれぞれ示している。

いま、制御装置の逐次制御カウンタの内容が第11図に示すように“0213”となっているとする。主制御回路よりの信号 a によって、これがアドレス選択回路に送られ、記憶装置の213番地が指定され、その内容“+150561”が取出されて、命令レジスタに送られる。この命令レジスタの指令部は命令デコーダによって解釈され、各種の制御信号が発生される。いま、それは“15”であるので、まず、制御信号 b によって命令レジスタのアドレス部“0561”がアドレス選択回路に送られ、記憶装置の561番地が指定され、その内容“+213814”が取出されて、演算装置に送られ、これが累算器に入れられる。これでこの命令の操作が終るので、終了信号 c が主制御回路に送られる。制御信号 d が逐次制御カウンタに加えられて、その内容が1だけ増されて“0214”となる。これで一つの命令の取出しとその実行が一通り終ったことになる。

つぎには逐次制御カウンタの内容が“0214”であるので前と同様にして214番地の内容が取出されて、命令レジスタに送られる。この命令の指令部は“11”であるので、そのアドレスで指定された562番地の内容が取出されて、この内容“+374062”が累算器の内容に加え込まれて、累算器の内容は“+587876”となりこの命令の実行は終了し、終了信号 c が主制御回路に送られ、制御信号 d が、逐次制御カウンタの内容を“1”だけ増す。以下同様にして“215”番地の命令を実行して、550番地の内容“+688979”を累算器の内容より

第 11 図

引いて、累算器の内容は“−101103”となって、この命令を終了し、逐次制御カウンタは1増して“0216”となる。

つぎに216番地の命令を取り出して、その指令部“82”を命令デコーダに送る。この命令は累算器の内容が負であれば、次にはこの命令のアドレスで指定される番地の命令を実行し、負でなければ、この命令の入っている番地の次の番地の命令を実行するのである。このために、命令デコーダからは演算装置に信号が送られ、累算器の符号をしらべる。いまの場合は累算器の符号が負であるので、演算装置からは制御信号eが出されて、命令レジスタのアドレス部の内容“0220”を逐次制御カウンタに送る。したがって、次には220番地の命令を取出して、これを実行することになる。もし、累算器の符号が負でなければ、演算装置からは制御信号eが出されずに、cが出されて、主制御回路より信号dが逐次制御カウンタに送られて、その内容が1だけ増されて、“0217”となり、次には217番地の命令が実行されるようになる。

この82のような命令によって記憶装置に格納されているプログラムの実行順序を変えることができ、計算機の内部条件（いまの場合は累算器の内容が負であるか否か）によって、違ったプログラムの道筋を進ませることができるようになる。このような命令を飛越し命令（あるいは分岐命令）といい、無条件に飛び越す命令と上の例の“82”の命令のように条件によって飛越したり、飛び越さなかったりする命令とがある。この条件付飛越し命令があるために、機械によって演算結果を自動的に判断させて、その結果によって以後の処理内容を変えることができるのであり、自動計算機

には欠かすことのできない命令である。

以上述べたような過程をたどって、計算機の記憶装置に格納されているプログラムの命令を一つずつ順次実行して、演算停止命令がくるか、又は何らかの原因によって演算実行不能の状態となるまで、一連の演算操作を全く自動的に実行して行くのである。

V 結 言

電子計算機では、情報はすべて2進符号で表わされ、その処理操作は、論理和、論理積、否定の論理演算を行なう回路（あるいはその他の論理演算を行なう回路を使う場合もあるが）の組合わせによって行なわれている。また、計算機の制御動作も、上に述べたような方法で行なわれるのであるが、それらも、上記の論理回路を主とし、これにその他の補助回路を組合わせて構成されている。しかも、実行できるのは個々の命令の指定する比較的簡単な操作にすぎないのである。

このような計算機で、複雑な知能的な問題まで処理することができるのは、大量の情報を記憶することができ、高速の自動的な連続処理ができることによるものであり、これを活用するのは、全くそのプログラムの作成にかかっているわけである。すなわち計算機による情報の処理はプログラミングに支配されるということができる。

本文においては、そのプログラミングについて言及することができなかったのは残念であるが、電子計算機の内部における情報の処理の概略について、御理解頂け、何らかの御参考になれば誠に幸いである。