講座

電子計算機とその応用(2)

電子計算機のプログラミング入門

清水留三郎*

1. 計算機の構成と命令

1.1 はじめに

計算機は加減乗除という最も基本的な演算しかできないのであって，初等関数の計算などの数学的問題を解くには，その問題を四則の組合わせで表現することが必要である。このことは，例えば積分をするのに，連続な変数の値を離散的な値に置きかえ，積分を和分として計算しなければならないことを意味している。

計算機は計算手が卓上計算機を作って問題を解くのと全く同じ方法で問題を解くのである。基本的な算術演算を指定する命令を，つぎつぎに計算手に与えて問題を解くことができるのであるが，計算機の方では，ある特定の計算をするための命令 (instruction, order) の系列をプログラムという。計算手は与えられた命令を過去の経験によって改善することができるのであるが，計算機は与えられた命令を馬鹿正直に実行するだけである。

1.2 計算機の構成

プログラムを作るには，計算機のおもな要素と相互の関係を簡単に知って置かねばならない。プログラマーに関係の深い要素として，記憶装置，演算装置および入出力装置がある。また人間の頭脳の役割をする制御装置がある。それらの関係は図—1に示してある。

(1) 入力装置

a) ある媒体から一連の数字を読み，

b) それらの数字を計算機の適当な場所（普通は記憶装置）へ移す。

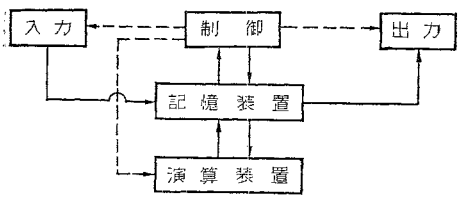
(2) 記憶装置

a) 指定された数字を受取って保存し，

b) 指定された数字を適当な場所（演算装置，制御装置，出力装置）に移す。

記憶装置は何個かの部分に分けられている。そのいずれかを指定できるように，各部分には番地 (address) がつけられている。各番地の内容を語といい，例えば n 番地の内容は (n) と表わされる。

図—1 計算機内の情報の流れ図



(3) 演算装置

a) 2個の数値を受け取って，

b) 単純な算術演算や論理演算を実行し，

c) その結果を適当な場所（普通は記憶装置）に移す。

演算装置の中心は，ソロバンの役割をする。accumulator (Acc) である。Acc は乗算の結果の2語の長さの積を収容できるように，番地の2倍の長さをもっている。演算装置はまた，乗数あるいは被乗数，除数を置くための multiplier (ormultiplicand) divisor register (MDR) をもっていることもある。

計算機に関してプログラマーが知って置かねばならぬことのうちおもなものは次のことがらである。

(1) 計算機が実行できる命令の種類と，それらを指定する符号 (operation code)。

(2) 計算機がある命令を実行した後，どのようにして次の命令を決めるか。

(3) 計算に必要な数値はどのような形で計算機に入れるか，またそれらの数値は計算機内でどのように表現されているか。例えば2進法であるか，10進法であるか，あるいは，固定小数点であるか，浮動小数点であるかということ。

以下では東大工学部の TAC に従って説明する。

1.3 命　　令

命令は演算を指定する演算部と，番地を指定する番地部から成っている。番地部が記憶装置のある1つの番地だけを指定する形を one-address 型という。このような型では m 番地の命令を実行すると，それが飛躍命令であるときを除いて，次の命令として $(m+1)$ を取り出してくる。

最も大切な命令のいくつかを書くと，

A	n	Add $(Acc)+(n)\rightarrow(Acc)$.
S	n	Subtract $(Acc)-(n)\rightarrow(Acc)$.
T	n	Transfer $(Acc)\rightarrow(n)$, $0\rightarrow(Acc)$.
U	n	$(Acc)\rightarrow(n)$, (Acc) は変化しない。
H	n	$(n)\rightarrow(MD)$
V	n	Verfielfachen $(Acc)+(n)\times(MD)\rightarrow(Acc)$
D	n	Divide $(Acc)+(n)\div(MD)\rightarrow(Acc)$
Z	～	停止（番地部不要）

1.4 簡単な例題

いくつかの簡単な計算をするのにどのような命令が必要か考えよう。計算に使われる数値はすでに記憶装置に入っているとする。また Acc は初めは払われており，問題の終りで払われていなければならないことにする。

例題 1：$(10)=x$, $(14)=y$ のとき，$x+y$ を計算して8番地に入れよ。

番地	命令	(Acc)
100	A 10	x
101	A 14	$x+y$
102	T 8	0

* 東京大学工学部応用物理学教室　森口研究室

例題 2：(10)＝a，(11)＝b，(20)＝x，(30)＝y のとき，ax を 12 番地に，$y(ax+by)$ を 13 番地に入れよ。

番地	命令	(*Acc*)	注
100	*H* 20	——	$x\rightarrow(MD)$
101	*V* 10	ax	
102	*U* 12	ax	$ax\rightarrow(12)$
103	*H* 30	ax	$y\rightarrow(MD)$
104	*V* 11	$ax+by$	
105	*T* 0	0	$ax+by\rightarrow(0)$
106	*V* 0	$(ax+by)\cdot y$	
107	*T* 13	0	$(ax+by)y\rightarrow(13)$

注 1：*H*-命令で *MDR* に入れられた内容は，次の *H*-命令がくるまで不変であるから，103 の命令で入れられた値 yは，104 と 106 の *V*-命令で使われる。

注 2：被乗数は記憶装置の内容でなければならないから，*Acc* にできた $ax+by$ は，それに yを掛ける前に，一たん記憶装置に移さなければならない。この目的に使われる番地を作業用番地（working storage）という。

1.5 飛躍命令

次の命令を命令群の特定のところから取出す操作を飛躍という。飛躍ののち命令は，他の飛躍命令に来るまで，引続く番地から取られる。この飛躍には無条件飛躍と，計算の過程で得られるある量の値に従って飛躍する条件付飛躍がある。飛躍命令に次の 3 種がある（まだ他にあるが，後に紹介する）。

F *n*	(*n*) を次の命令として取る。
G *n*	(*Acc*)<0 ならば，⇨*n* そうでなければ，引続く番地へ進む。
E *n*	(*Acc*)≧0 ならば，⇨(*n*) そうでなければ，引続く番地へ進む。

Fn は無条件飛躍であり，*En* と *Gn* は条件つき飛躍である。*En* と *Gn* は表裏の関係にある。

もしn番地に *Fn* という命令があると，一たんその命令が実行されると，飛躍が永久にくり返されて機械は停止状態になる。これを dynamic stop といい，普通の *Z*-命令の代りに使われることもある。

例題 3：x＝(4) の絶対値を 4 番地に入れよ。

番地	命令	(*Acc*)	注
100	*S* 4	$-x$	
101	*G* 103	$-x$	$-x<0$ ならば飛ぶ
102	*T* 4	0	$x\leq 0$ のとき，$-x\rightarrow(4)$
101→103	つ づ き		

もしxが負ならば，101 番地の条件つき飛躍命令のとき，*Acc* の内容は正であるから，102 番地に進んで $|x|$ を 4 番地に入れる。x が正ならば，飛躍命令のとき，*Acc* は負であるから，103 番地へ飛ぶ。このときは 4 番地にはすで $|x|$ にが入っているから，何もしないでよい。

例題 4：(4)<0 である。(4) に (0) を加えることを，その結果が正になるまでくり返し，結果を (4) に入れよ。(4)＝$-x(x>0)$，(0)＝y とする。

番地	命令	(*Acc*)	注
100	*A* 4	$-x$	
102→101	*A* 0	$-x+y$，$-x+2y$,…	
102	*G* 101		(*Acc*)<0 ならばとぶ
103	*T* 4		

1.6 B-レジスター

一群の命令の一部を少しずつ変更して，何回かくり返す場合がよくある。そのときの変更は，ある命令の番地について施される。

例題 5：(10)＝a_0，(11)＝a_1,…，(19)＝a_9；(20)＝x，多項式 $a_0x^9+a\cdot x^8+\cdots+a_9$ を計算して，0 番地に入れよ。

多項式は次の式によって計算する。

$$[\{(a_0x+a_1)x+a_2\}x+a_3]x+\cdots$$

従ってxを掛けては a_i を加えるという一対の計算を，必要な回数だけくり返す。xをくり返し掛けるから，xは *MDR* に入れておく。この演算を n 回くり返したときの中間結果を S_n とすると，

$$S_{n+1}=xS_n+a_n$$

最終結果は 0 番地に入れるのであるから，S_n は 0 番地に入れておく。a_i を加える命令の番地部は，くり返しごとに変更しなければならないが，B-レジスターを使ってこれを行なうことができる。B-レジスターは一般にはインデックス・レジスターと呼ばれている。

B-レジスターには次の命令によって，1 個の数を入れておくことができる。

Bm	B-レジスターに数 m を入れよ。 (m 番地の内容ではないことに注意)

どんな命令の番地部にも，この B-レジスターの内容を加えるように指定することができる。この指定は演算符号の後に *S* を書いて行なう。もし B-レジスターに 11 が入っているとすると，

命令	実行命令	注
V 0	*V* 0	xを掛ける。
AS 0	*A* 11	a_1 を加える。
T 0	*T* 0	$S_1=xS_0+a_1$

一番左の列のように書いた命令は，2 列目の命令と全く同じ効果をもつ。B-レジスターの内容を 1 だけ増す命令を行なったあとで同じ命令群を実行すると，2 行目の命令は，

A 12 | a_2 を加えよ。

と同等になる。この命令の変更は記憶装置から制御装置に移されるときに行なわれるのであって，記憶装置にある命令 *AS* 0 は *A* 11 や *A* 12 に書きかえられはしない。B-レジスターの内容は次の命令によってふやせる。

BSm | B-レジスターの内容を m だけふやせ。

B-レジスターの内容から m を引きたい場合もよくある。これは命令の番地部の後に *S* を書いて示す。従って

BSmS | B-レジスターの内容を *m* だけ減らせ。

という命令と，これと同様な

B mS | B-レジスターに数-*m* を入れよ。

という命令がある。以上のようにして番地部の変更を行なうことができることがわかったが，適当な回数だけ命令群をくり返すために回数を数える必要がある。これも次の条件つき飛躍命令を使って，B-レジスターで数えることができる。

J n | (*B*)≠0 ならば *n* 番地へ飛べ。(*B*)=0 ならば次の番地へ進む。

この命令でB-レジスターの内容が0でなければいつでも飛ぶのであるから，回数の数え方には，初めに正の数を入れて置いて，減らして0にするか，負の数を入れて置いて，ふやして0にするかの2通りある。この5例題は後者の場合であるので，プログラムは次のようになる。

100	*H* 20		x
101	*A* 10		a_0
102	*B* 9*S*		$-9\rightarrow(B)$
107→103	*BS* 1		$(B)+1\rightarrow(B)$
104	*T* 0		
105	*V* 0		$S_{n+1}=xS_n+a_n$
106	*AS* 19		
107	*J* 103		

もし $a_0,a_1,\cdots,a_9$ が逆の順に記憶されていて，(10)=a_9，(11)=a_8,…であるときは，へらしながら数えなければならないので，次のようになる。

101	*A* 19		
102	*B* 9		$9\rightarrow(B)$
103	*BS* 1*S*		(*B*) を1だけへらす
107→104	*T* 0		
105	*V* 0		$S_{n+1}=xS_n+a_n$
106	*AS* 10		
107	*J* 104		

2. 例　　題

2.1 平 方 根

(300)=a の平方根 $\sqrt{a}$ を求めよ。

Newton の近似公式 $r_{i+1}=\frac{1}{2}\left(r_i+\frac{a}{r_i}\right)$によって計算する。250 番地に $\frac{1}{2}$ を入れて置く。

100	*A*	300	a	
101	*R*	1	$\times 2^{-1}$	$r_1=\frac{1+a}{2}(r_0=1)$
102	*A*	250	$+2^{-1}$	
110→103	*T*	301		$r_i\rightarrow(301)$
104	*H*	301	r_i	
105	*D*	300	a	$r_{i+1}-r_i=\frac{1}{2}\left(\frac{a}{r_i}-r_i\right)$
106	*S*	301	$-r_i$	
107	*R*	1	$\times 2^{-1}$	
108	*E*	111		$r_{i+1}-r_i\geqq 0$ ならば収束
109	*A*	301	$+r_i$	
110	*F*	103		
108→111	*A*	301		

2進法の計算機では 2^{-m} 倍は m 桁右へずらすことによって実現できる。桁移動に関する命令には次の2つがある。

R m	Right shift (*Acc*) を右へ *m* 桁ずらせ。
L m	Left shift (*Acc*) を左へ *m* 桁ずらせ。

2.2 積　　和

(300)=a_1，(301)=a_2,……，(319)=a_{20}；(400)=b_1，(401)=b_2,……，(419)=a_{20} のとき，積和 $S=\sum_{i=1}^{20}a_ib_i$ を求めよ。

これを

H	300	a_1
V	400	$\times b_1$
H	301	a_2
V	401	b_2
⋮		⋮
H	319	a_{20}
V	419	b_{20}

とくり返すことの代りに B-レジスターを使って次のようにプログラムする。

100	*B*	20*S*	$-20\rightarrow(B)$
104→101	*BS*	1	$(B)+1\rightarrow(B)$
102	*HS*	319	$a_i\rightarrow(MD)$
103	*VS*	419	$S_{n+1}=S_n+a_i\times b_i$
104	*J*	101	(*B*)=0 になるまでくり返す

3. サブルチーン

3.1 相対番地

プログラムを書くときに，一つのまとまった命令群はその番地部を書くときに，命令群の最初の番地を起点にして書くのが大変便利である。このように記憶装置につけられている実際の番地を意味するのではなく，特定の番地からの相対的な増加量を表わすものを相対番地といい，前者を絶対番地という。いままではすべて絶対番地で話を進めてきたわけである。

番地	命令
100	*S* 4
101	*G* 103
102	*T* 4

番地	命令	注
θ	*S* 4	
$\theta+1$	*G* 3θ	$G(\theta+3)$
$\theta+2$	*T* 4	

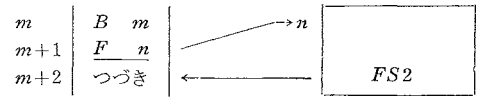
相対番地は，絶対番地と区別するために，番地部のあとに θ を書く。上の2つの命令群は全く同等である。

3.2 サブルチーン

計算機は前にも述べたように四則だけしかできない。よく必要になる初等関数などは四則の組合わせで求めるのであるが，このための命令群は誰かがそれを一度作ったら，それを他の誰でもが自由に使えて，一々作る必要のないようにしておくのが便利である。そのために便利なようにまとめたものをサブルチーンという。これに対して，個々の問題特有の計算過程を与える命令群を主ル

チーン（master routine or main routine）という。サブルチーンとして必要な資格には，主ルチーンのどこからそのサブルチーンに飛び込んできても，飛んで来た所の続きへうまく戻ることである。このために主ルチーンでは B-レジスターに命令自身の入っている番地を入れてから，サブルチーンへ飛び込む。サブルチーンでは (B) によって戻るべき場所を知る。このための手続きをつなぎ（linkage）という。

〔主ルチーン〕　〔サブルチーン〕



初等関数のサブルチーンでは主ルチーンで変数 x を Acc に入れて置いてサブルチーンに飛ぶと，Acc に関数 $f(x)$ が入った状態で戻ってくるのが普通である。

例題　平方根サブルチーン

0θ	U	0	$x\rightarrow(0)$
1	R	1	
2	A	13θ	$r_1=\frac{1+a}{2}$
$10\theta\rightarrow 3$	T	40	$r_i\rightarrow(40)$
4	H	40	
5	D	0	
6	S	40	$r_{i+1}-r_i=\frac{1}{2}\left(\frac{x}{r_i}-r_i\right)$
7	R	1	
8	E	11θ	$r_{i+1}-r_i\geqq 0$ ならば収束
9	A	40	
10	F	3θ	
$8\theta\rightarrow 11$	A	40	
12	FS	2	
13	‖I	0	定数 2^{-1}

この例のようにサブルチーンは相対番地で書かれている。そうしておけば，この命令群を 100 番地から入れようが，200 番地から入れようが，適当な計算をしてくれるが，絶対番地で書いてあったらそのようには行かない。例えば 8θ の条件付飛躍命令 $E11\theta$ を E111 と絶対番地に変えたものを 200 番地から入れたのでは，平方根を正しく計算しない。

このサブルチーンを 100 番地から入れ，それを使って，$(300)=x$ の平方根 $\sqrt{x}$ を計算するには，主ルチーンを次のように書けばよい。

200	A	300
201	B	201
202	F	100

参　考　書

1) Booth, K.H.V.: Programming for an Automatic Digital Calculator, Academic Press, 1958.
2) McCracken, D.D.: Digital Computer Programming, John Wiley, 1957.
3) Wilkes, M.V., Wheeler, D., Gill, S.: The Preparation of Program for an Electronic Digital Computer, 2nd ed., Addison-Wesley, 1957.
4) 渋谷政昭：計算機のプログラミング，東洋経済，1959.

1) は London の Birkbeck College にある計算機 APEXC をもとにして書かれている。2) は架空の計算機 TYDAC を考えて説明している。3) は Cambridge Univ. に昔あった EDSAC の言葉でプログラミングを基礎から説いた好著。4) は国産のリレー計算機 FACOM-128B について例題本位にプログラミングを解説している。自動計算機で使われている数値計算の手法の参考になる。

（原稿受付：1960.10.24）

書　評

東京の水道

佐藤志郎著　都政通信社刊

著者は 30 余年の長きにわたって水道事業に終始したその豊富な経験と知識をもとにして，大東京の活力源ともいうべき水道について，その生い立ちから，推移，さらには今後の諸問題について，明快なる筆をすすめている。

著者はそのはしがきの中で「東京 23 区の人口のうち未給水人口は 150 万人もある。近年簡易水道が非常に普及して相当辺鄙な農山村でさえ水道の恩恵に浴しているのに日本の首都のまん中に水道のない巨大な田舎があるとはいったいどうしたことか……」と。

試みにおもなる項目をひろってみると，本書は 3 編よりなっているが，まず第 1 編では上水道についてのべている。すなわち本格的な水道ができたのは明治も半ばすぎであるがそれ以前は飲料水をどうして得ていたであろうか。しばしば都内の工事現場から木製の管が発掘されることがある。これはすなわち江戸住民の給水本管であると，江戸時代の水道についても興味深い解説がつけられている。

さらに，明治，大正，昭和と 3 代にわたる水道事業の計画，工事の施工状況，経営の状況のそれぞれについて詳細にのべている。このうち関東大震災による水道施設の被害資料は今後の水道施設の耐震工法にしさするところが多いと思われる。

戦後における水道技術の進歩はめざましく，長沢浄水場において，日本初めてのマイクロストレーナー（回転式微細網ろ過装置）の採用，2 階槽沈殿池，あるいは集中管理方式の採用などについて詳細にのべてあり，今後の水道に参考にすべき点が多くあると思われる。また著者はダム築造の権威として小河内ダム完成に半生をかけ，その経過について表裏をつぶさにのべていることは，興味深いところである。

第 2 編では下水道について明治 10 年コレラ大流行による下水道施設の必要性が痛感せられ，いわゆる神田下水なる分流式下水道が実施され，その後合流式下水道に切りかえられ，関東大震災をはさんで大正，昭和と下水道事業の変せんについて詳細にのべられてゐる。

さらに第 3 編ではお水道，下水道，工業用水道の諸問題点について言及してあり，水道技術者はもちろん，水道経営にたずさわっている人々，一般土木技術者にとっても興味あるものと思われる。

著者：正員　東京都水道局長
B 6 判　771ページ，　定価 800 円
昭. 35. 6. 30 発行
都政通信社：東京都港区芝松本町 58
Tel　(451)　2669・5068
振替東京　24290