

4ビット・ミニコンピュータについて

東芝 電子計算機事業部

高橋 義造

1. はじめに

超大型計算を主題とするこの研究集会に、超小型計算機の話を持ち出すのは非常に場違いな感があるが、主催者の方の深いお考えを推量するよしもないので、この発表をお引き受けした次第である。そこで本日の発表が超大型計算に関係があるかどうかは考えないことにして、4ビットミニコンピュータの話をして頂くことにする。

コンピュータの大きさをはかるのに、演算速度、主記憶容量、レジスタ数、語長、命令数、入出力機器の性能、チャンネル数、チャンネルのデータ転送速度などの尺度があるが、値段をもつてあるのが最も手取り早い。コンピュータの値段と性能の間には有名な Grosch の法則、即ち

$$\text{Performance} \propto (\text{price})^2$$

があるとされるので値段と性能には一意的な関係がある。

しかしながら、超大型電子計算機の出現、ソフトウェア危機（つまりいかに人員を投入しても、一定限度以上大きなソフトウェアをつくることはできないという警告）、および、ミニコンピュータの発達によって図1のように、A（超大型機）、C（ミニコンピュータ）のような Grosch の法則の成立しない領域があらわれた。

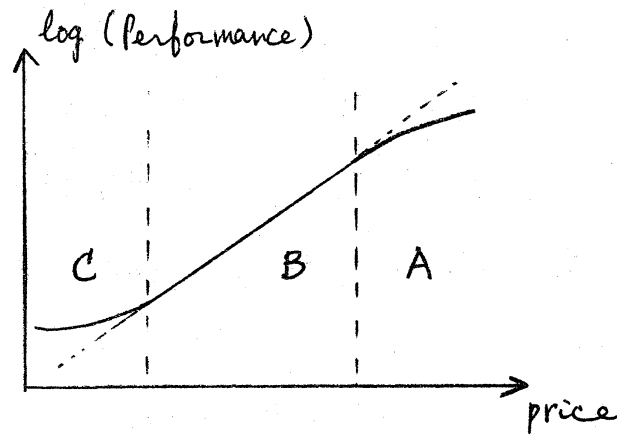


図 1

図1のようにコンピュータはそのスケールを小さくしても、性能は決して減少しない。ミニコンピュータでも使いようによって、中には、小型機に近い効果を出せるわけである。従ってCの領域をさらに左へ広げることにより、非常に小さなミニコンピュータでも予想外の力を発揮する可能性がうかがわれる。ミニコンの特徴を一言でいえば、コンピュータを構成する演算制御装置（CPU）、主記憶装置（メモリ）、周辺装置のうち、CPUとメモリが極端に安いということである。

す、ミニコンの効果を発揮するのは、I/OよりもCPUをよく使う計算を行なう場合である。

2. 4ビットマシンの発想

コンピュータの内部における情報の単位は1か0の値をもつビットであるが、処理される情報の単位はビットの集合である語(word)である。語には命令語とデータ語がある。命令語とデータ語の語長は必ずしも等しくはないが、通常は基本命令の語長と整数データの語長は等しく、命令語の中にこれの整数倍のものがあり、浮動小数点数の語長は整数データの2倍などになっていることが多い。

次に主なコンピュータの基本語長を示す。

CDC-6600	60ビット
----------	-------

IBM 7094	36ビット
----------	-------

IBM S/360	32ビット
-----------	-------

HITAC-5020	48ビット
------------	-------

TOSBAC-3400	24ビット
-------------	-------

また、ミニコンピュータでは次のようなものがある。

PDP-8	12ビット
-------	-------

HITAC-10	16ビット
----------	-------

TOSBAC-40	16ビット
-----------	-------

TOSBAC-10 8ビット

VARIAN 520/i 8ビット

当初は8ビットの語長というのは、とうてい使用にたえないとおもわれていたが、8ビットマシンでもFORTRANコンパイラが実現されており、決して16ビットマシンに劣らない性能をもっている。そこでこれに力を入れて、さらに、語長が短くならないかと考えてみたくなる。

語長が短くなることの利点は、演算回路が簡単になること、デューダが簡単になること、データパスのゲート数が減少することなどのために、CPUのコストが非常に廉くなること、および回路の簡素化の結果として演算の高速化がはかれることである。

それでは、どこまで語長は短かくできるか？ 6ビットあるいは4ビットが考えられる。もっとも1ビットでもよいかもしれないが（Turingマシン）、実際には4ビットぐらいが限度であろう。語長を4ビットにすると $2^4 = 16$ 通りの組合せが可能になり、

1. 0~9の10進数が表現できる。

2. プッシュホンのような127の入力キーでデータが表現できる。

3. デジタルプリックのような簡単な出力装置で

16欄にわたり約16種類の文字を指定できる。

4. load, store など基本命令数16以内で
何とか足りる。

のような条件が満たされる。

このような検討により4ビットマシンが可能であることがわかった。そこで本年1月の箱根でのプログラミング・ミニボジウムの自由討議が話題に上ったところ、東大穂坂教授のミニミニコンピュータ論によって同じような考えのあることを知った。たまたまこの頃米国の半導体メーカー INTEL 社より LSI による4ビットコンピュータ MCS-4 (Micro-Computer とよばれる) が発表され、カタログが配布されはじめた。これは次章にのべるように4ビットのデータ語を取り扱うもので、Cash Register や電卓などの分野に100ドルコンピュータとして売り込まれている。また、前記の自由討議の席で学習院大学米田教授から Sony の Micro-Computer SOBAX ICC-2700 も4ビットマシンであることを教えたいただいた。SOBAX のデータ語は、10進15桁+符号(64ビット)語長であるが、命令語が4ビットである。これに反し INTEL の MCS-4 の命令語の語長は8ビットである。MCS-4 は発表されたばかりでその応用分野はまだ開拓されておらず、効果のほどは未知

数であるが、SOBAXは高級電卓として知られており、愛好者も少なくないようである。

機種 語長	INTEL MCS-4	SOBAX ICC-2700
データ語	4ビット	64ビット
命令語	8ビット	4ビット

それではデータ語と命令語の語長がともに、4ビットの本
当の4ビットマシンは可能であろうか。一つの試案を示すこ
とにするが、その前にMCS-4とSOBAXの紹介を行な
うことにする。

3. INTELの4ビットマシン MCS-4

MCS-4は次の4つの基本的なLSIチップより構成さ
れる。

- | | |
|-------------------------------|------|
| 1. CPU | 4004 |
| 2. ROM (read only memory) | 4001 |
| 3. RAM (random access memory) | 4002 |
| 4. SR (shift register) | 4003 |

1つのCPUにつき16ヶのROMと16ヶのRAMに接続
することができる。SRは入出力を拡張するために必要に依

に使用される。上記の4つの要素は次の図のように接続される。

MCS-4 BASIC SYSTEM

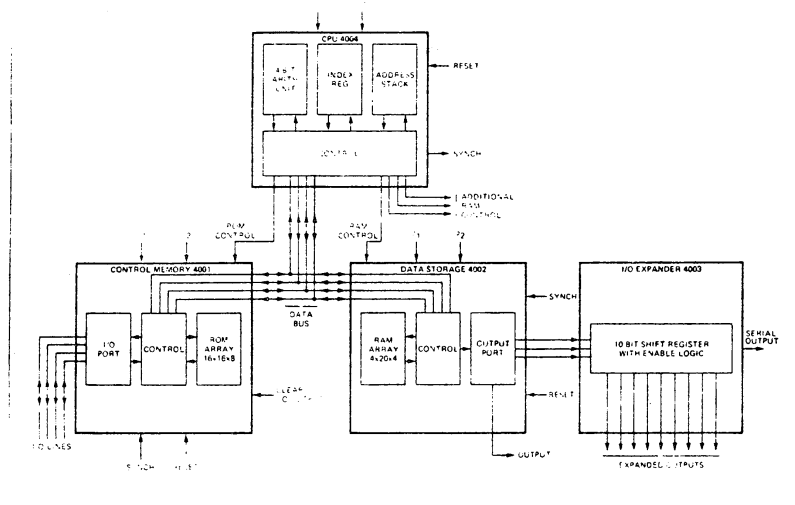


図 2

3.1 ROM 4001

MCS-4のプログラムはROMに記憶される。

4001はmask programmableなROMで142 256×8ビットの情報を記憶させることができる。CPUとつながれている4本のdata bus lineにより、ROMアドレスとデータがやりとりされる。1つのinstruction cycleは10.8 μ sで84のclock cycleより構成されているが、最初の3サイクルの間にCPUよりROMに対して12ビットの(4ビットずつ3回に分けて)ROMのアドレスが送られる。ROMの1語は8ビットで、この中に書き込まれている命令語は次の2サイクルで(4ビットずつ2回に分けて)ROMよりCPUにおくられる。残りの3サイクルでこの命令が解釈されて実行される。ROMにはまた、I/O portがあり、CPUの指示によりdata bus lineを通じてCPUから送られる4ビットの情報を出力したり逆に入力データをCPUにおくる仕事を行う。1つのCPUに最大16つのROMがつながれるので4Kバイト(1バイト=8ビット)の記憶容量がプログラムの記憶に使用される。

3.2 RAM 4002

RAMはデータの記憶、およびデータの出力に使用される。これはそれぞれ4ビットずつの16つのmemory character

と44の status character 51 なる合計 20×4 ビットのレジスタ4451構成されたRAMで1つのCPUには164のRAMが接続されているので、全部で 1280×4 ビットの記憶容量となる。

CPUの SRC (send address to ROM or RAM) 命令を実行すると8ビットのRAM address が2回にわたっておくられて、1つの memory character が指定される。なお、SRC 命令に立って DCL (designate Common and line) 命令によって164の4002のうち44のRAM bank を選択しておかなくてはならない。つぎにWRM, ROM などの命令の実行によりRAMとCPU間のデータの転送が行われる。4002にはまた、4本の出力ラインとエフ output port があり、WMP 命令によりCPUのアキュムレータの内容を出力する事ができる。

3.3 SR 4003

4003 は10ビットのシフトレジスタで4本しかないROMあるいはRAMの入力ラインを10本まで拡張するために使用される。これは10ビットのシフトレジスタで serial input, serial output, parallel output にもつ。

3.4 CPU 4004

CPUには4ビットの adder, 164の4ビットの index

register, 44 の 12 ビットの program counter stack, 8 ビットの instruction register などがある。1 インストラクションサイクルは 10.8 μ s であり, 8 桁の 10 進数の加算は 850 μ s で行われる。4004 には 44 の命令があり, そのうち 5 つは 16 ビットの語長, 他はすべて 8 ビットの語長である。その命令はつぎの通りである。

[Those instructions preceded by an asterisk (*) are 2 word instructions that occupy 2 successive locations in ROM MACHINE INSTRUCTIONS]

MNEMONIC	OPR D ₃ D ₂ D ₁ D ₀	OPA D ₃ D ₂ D ₁ D ₀	DESCRIPTION OF OPERATION
NOP	0 0 0 0	0 0 0 0	No operation.
*JCN	0 0 0 1 A ₂ A ₂ A ₂ A ₂	C ₁ C ₂ C ₃ C ₄ A ₁ A ₁ A ₁ A ₁	Jump to ROM address A ₂ A ₂ A ₂ A ₂ A ₁ A ₁ A ₁ A ₁ (within the same ROM that contains this JCN instruction) if condition C ₁ C ₂ C ₃ C ₄ (1) is true, otherwise skip (go to the next instruction in sequence).
*FIM	0 0 1 0 D ₂ D ₂ D ₂ D ₂	R R R 0 D ₁ D ₁ D ₁ D ₁	Fetch immediate (direct) from ROM Data D ₂ D ₂ D ₂ D ₂ to index register pair location RRR. (2)
SRC	0 0 1 0	R R R 1	Send the address (contents of index register pair RRR) to ROM and RAM at X ₂ and X ₃ time in the instruction cycle.
FIN	0 0 1 1	R R R 0	Fetch indirect from ROM. Send contents of index register pair location 0 out as an address. Data fetched is placed into register pair location RRR at A ₁ and A ₂ time in the instruction cycle.
JIN	0 0 1 1	R R R 1	Jump indirect. Send contents of register pair RRR out as an address at A ₁ and A ₂ time in the instruction cycle.
*JUN	0 1 0 0 A ₂ A ₂ A ₂ A ₂	A ₃ A ₃ A ₃ A ₃ A ₁ A ₁ A ₁ A ₁	Jump unconditional to ROM address A ₃ A ₃ A ₃ A ₃ A ₁ A ₁ A ₁ A ₁ .
*JMS	0 1 0 1 A ₂ A ₂ A ₂ A ₂	A ₃ A ₃ A ₃ A ₃ A ₁ A ₁ A ₁ A ₁	Jump to subroutine ROM address A ₃ A ₃ A ₃ A ₃ A ₁ A ₁ A ₁ A ₁ , save old address. (Up 1 level in stack.)
INC	0 1 1 0	R R R R	Increment contents of register RRR. (3)
*ISZ	0 1 1 1 A ₂ A ₂ A ₂ A ₂	R R R R A ₁ A ₁ A ₁ A ₁	Increment contents of register RRR. Go to ROM address A ₂ A ₂ A ₂ A ₂ A ₁ A ₁ A ₁ A ₁ (within the same ROM that contains this ISZ instruction) if result $\neq 0$, otherwise skip (go to the next instruction in sequence).
ADD	1 0 0 0	R R R R	Add contents of register RRR to accumulator with carry.
SUB	1 0 0 1	R R R R	Subtract contents of register RRR from accumulator with borrow.
LD	1 0 1 0	R R R R	Load contents of register RRR to accumulator.
XCH	1 0 1 1	R R R R	Exchange contents of index register RRR and accumulator.
BBL	1 1 0 0	D D D D	Branch back (down 1 level in stack) and load data DDDD to accumulator.
LDM	1 1 0 1	D D D D	Load data DDDD to accumulator.

Table V - Basic CPU Instruction Set

INPUT/OUTPUT AND RAM INSTRUCTIONS

(The RAM's and ROM's operated on in the I/O and RAM instructions have been previously selected by the last SRC instruction executed.)

MNEMONIC	OPR D ₃ D ₂ D ₁ D ₀	OPA D ₃ D ₂ D ₁ D ₀	DESCRIPTION OF OPERATION
WRM	1 1 1 0	0 0 0 0	Write the contents of the accumulator into the previously selected RAM main memory character.
WMP	1 1 1 0	0 0 0 1	Write the contents of the accumulator into the previously selected RAM output port. (Output Lines)
WRR	1 1 1 0	0 0 1 0	Write the contents of the accumulator into the previously selected ROM output port. (I/O Lines)
WR0 ⁽⁴⁾	1 1 1 0	0 1 0 0	Write the contents of the accumulator into the previously selected RAM status character 0.
WR1 ⁽⁴⁾	1 1 1 0	0 1 0 1	Write the contents of the accumulator into the previously selected RAM status character 1.
WR2 ⁽⁴⁾	1 1 1 0	0 1 1 0	Write the contents of the accumulator into the previously selected RAM status character 2.
WR3 ⁽⁴⁾	1 1 1 0	0 1 1 1	Write the contents of the accumulator into the previously selected RAM status character 3.
SBM	1 1 1 0	1 0 0 0	Subtract the previously selected RAM main memory character from accumulator with borrow.
RDM	1 1 1 0	1 0 0 1	Read the previously selected RAM main memory character into the accumulator.
RDR	1 1 1 0	1 0 1 0	Read the contents of the previously selected ROM input port into the accumulator. (I/O Lines)
ADM	1 1 1 0	1 0 1 1	Add the previously selected RAM main memory character to accumulator with carry.
RD0 ⁽⁴⁾	1 1 1 0	1 1 0 0	Read the previously selected RAM status character 0 into accumulator.
RD1 ⁽⁴⁾	1 1 1 0	1 1 0 1	Read the previously selected RAM status character 1 into accumulator.
RD2 ⁽⁴⁾	1 1 1 0	1 1 1 0	Read the previously selected RAM status character 2 into accumulator.
RD3 ⁽⁴⁾	1 1 1 0	1 1 1 1	Read the previously selected RAM status character 3 into accumulator.

ACCUMULATOR GROUP INSTRUCTIONS

CLB	1 1 1 1	0 0 0 0	Clear both. (Accumulator and carry)
CLC	1 1 1 1	0 0 0 1	Clear carry.
IAC	1 1 1 1	0 0 1 0	Increment accumulator.
CMC	1 1 1 1	0 0 1 1	Complement carry.
CMA	1 1 1 1	0 1 0 0	Complement accumulator.
RAL	1 1 1 1	0 1 0 1	Rotate left. (Accumulator and carry)
RAR	1 1 1 1	0 1 1 0	Rotate right. (Accumulator and carry)
TCC	1 1 1 1	0 1 1 1	Transmit carry to accumulator and clear carry.
DAC	1 1 1 1	1 0 0 0	Decrement accumulator.
TCS	1 1 1 1	1 0 0 1	Transfer carry subtract and clear carry.
STC	1 1 1 1	1 0 1 0	Set carry.
DAA	1 1 1 1	1 0 1 1	Decimal adjust accumulator.
KBP	1 1 1 1	1 1 0 0	Keyboard process. Converts the contents of the accumulator from a one out of four code to a binary code.
DCL	1 1 1 1	1 1 0 1	Designate command line. (See note 1 on page 3.)

NOTES: (1) The condition code is assigned as follows:

C₁ = 1 Invert jump condition C₂ = 1 Jump if accumulator is zero C₄ = 1 Jump if test signal is a 0
 C₁ = 0 Not invert jump condition C₃ = 1 Jump if carry/link is a 1

(2) RRR is the address of 1 of 8 index register pairs in the CPU.

(3) RRRR is the address of 1 of 16 index registers in the CPU.

(4) Each RAM chip has 4 registers, each with twenty 4 bit characters subdivided into 16 main memory characters and 4 status characters. Chip number, RAM register and main memory character are addressed by an SRC instruction. For the selected chip and register, status character locations are selected by the instruction code (OPA).

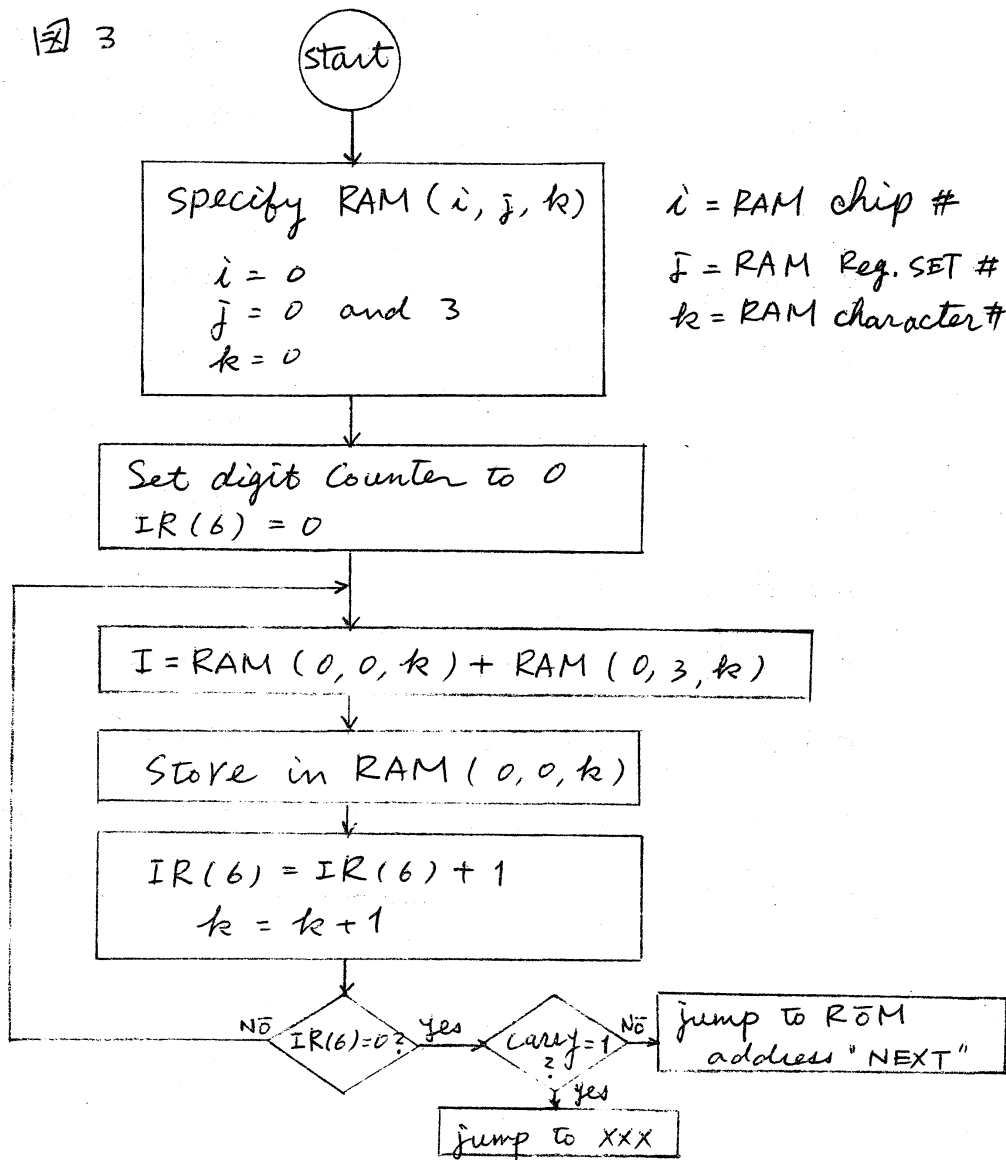
Table V - Basic CPU Instruction Set (Continued)

3.5 プログラム例

RAM 0 の Register 0 と 3 に入った "3" / 6 桁の 10 進数
を計算して Register 0 に入れるプログラムを書いてみる。

RAM register の memory character 0 には 最小桁が
15 には 最大桁が入っているものとする。

図 3



location	instruction word	mnemonic	
000	2000	FIM	0,0
002	2430	FIM	4, '30'
004	D0	LDM	0
005	B6	XCH	6
006	F1	CLC	
007	25	AD1 SRC	4
008	E9	RDM	
009	21	SRC	0
00A	EB	ADM	
00B	FB	DAA	
00C	E0	WRM	
00D	61	INC	1
00E	65	INC	5
010	76 07	ISZ	6, AD1
012	12 15	JCN	2, ***
014	40 1C	JUN	NEXT
015		***	-----

4. SOBAX ICC-2700

SOBAX ICC-2700は高級電卓の1つで、MANUALモードではキーを押すことにより演算が直接行われ、AUTOモードではプログラムメモリに記憶された最大253ステップのプログラムが自動的に実行される。データの記憶には127のデータ・メモリが用意されており、10進15桁+符号のデータが記憶される。データキーより入力されたデータあるいはデータ・メモリより読み出されたデータはレジスタ1に入るとともに数字表示される。このデータについての演算キーがおさめられるか、あるいはプログラムメモリから演算コードが読み出されるとデータはレジスタ2におくられる。そして演算数がキーまたはデータメモリより入力されレジスタ1に入る。そして次に演算キーがおさめられると、先におさめた演算キーに対応して、レジスタ2のデータに対しレジスタ1のデータが演算され、レジスタ1に入り表示される。また、同時に演算数であるレジスタ1にあった数はレジスタ2に移される。15桁の10進数の加減算は5msで行なわれる。

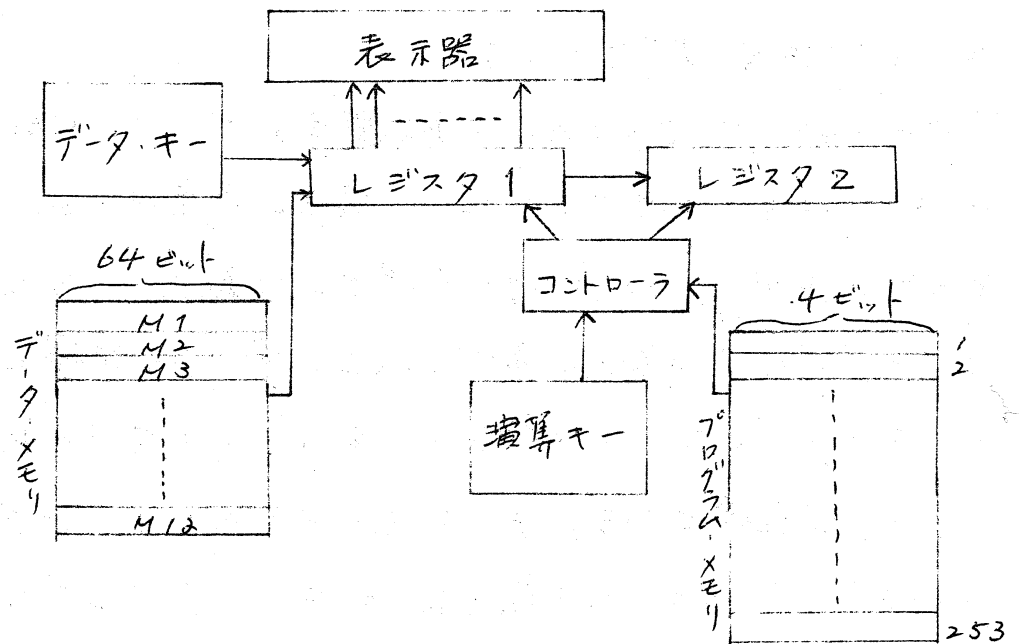


図 4

4.1 レジスタとメモリの語長

レジスタ1, レジスタ2	10進15桁+符号
データメモリ M1~M12	10進15桁+符号
プログラムメモリ	4ビット

4.2 命令語

MANUALモードでは演算命令は演算キーを押すことにより発生される。この演算キーの1つ1つが命令語に対応する。

No.	演算キー	演算
1	+	add
2	-	subtract

No.	演算キー	演算
3	X	multiply
4	÷	divide
5	=	演算の表示
6	√	平方根
7	Min	(register 1) + (データメモリ) → データメモリ
8	Minut	(データメモリ) → register 1
9	Mc	0 → データメモリ
10	M	データメモリのアドレス指定
11	R	(register 1), (register 2)の交換
12	()	数値モードと記号モードの切り換え
13	CHG, SIGN	-(register 1) → register 1
14	S	stop, キーボードよりレジスタ1にデータを输入することが出来る。
15	J	jump
16	End	end of program

これらの演算命令は16ヶあり、4ビットで表現される。

プログラム中に数字データが入るか、これは()と()の間にかきまれた部分、およびM, Jの次の文字として区別される。

4.3 データ語

先にものべたようにSOBAX LC-2700で演算の対象と

なるデータ語は10進数15桁+符号であるが、1つの桁は14種類のデータ・キー、即ち、

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

•, change sign, ←, →

で表わされる4ビットデータである。

4.4 プログラム例

n 個のデータ $x_1, x_2, \dots, x_n$ (n は任意) の平均値を求めるプログラムをかく。計算機が停止する度に、それまでの平均値が表示され次のデータ x_i をデータ・キーから入力し、演算キーを押してスタートするようなプログラムをつくる。

プログラム	説明
S	
M1 MC Min	input x_1
M2 MC	$0 \rightarrow M_2$
() 1 () M3 MC Min	$1 \rightarrow M_3$
M J 1	jump point 1
M3 Mont	$(M_3) \rightarrow \text{reg. 1}$
M2 Min	$(\text{reg. 1}) + (M_2) \rightarrow (M_2)$
Mont $\div$ M1 Mont	
R = S	$(M_1) \div (M_2)$ input x_i
M1 Min	$(\text{reg. 1}) + (M_1) \rightarrow M_1$
- R = J 1	unconditional jump to 1
END	

5. 4ビットマシンの実現

前の2章で2つの実在する4ビットマシンの解説をしたがいずれもデータ語又は命令語のいずれかが4ビットではなかった。そこで命令語とデータ語のいずれもが4ビットであるような4ビットマシンが可能かどうかを考えることとする。

5.1 データ語

データ語の語長は4ビットとする。4ビットのデータは16進表示により

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

A, B, C, D, E, F

のコードで表わされる。

10進数のデータは次のようにあらわされる。

コード	10進データ
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	delimiter for positive number
B	delimiter for negative number
C	decimal point

例

$$123 \longrightarrow 123A$$

$$-56 \longrightarrow 56B$$

$$123.45 \longrightarrow 123C45A$$

$$-3.14 \longrightarrow 3C14B$$

5.2 Xモリ

プログラムおよびデータを記憶させるためのXモリに16
 4のstackを用いる。番号を一度指定すると、次に指定があ
 るまで同じstackが指定されたこととなる。readされたデ
 ータは必ずstackにつまれ、また、stackから読み出され
 たデータはアキュムレータに入るか、あるいはインストラク
 ションレジスタに入る。stackからデータが読まれると、そ
 のstackに最後に入れられたデータが取り出され、stackか
 らは消される。stackにデータがストアされるとこれは一番
 上におかれる。一組の表示器が指定のスタックに接続され
 とそのstackの内容が下を左にして表示される。

5.3 レジスタ

この4ビットマシンでは演算はすべて10進演算で4ビッ
 トのアキュムレータで行なわれる。演算されるデータは16
 4のstack (S₀ ~ S₁₅) より取り出され、演算の結果はアキ
 ュムレータに入る。レジスタとしてはアキュムレータの外

にMQレジスタ(4ビット), キャリーレジスタ(1ビット),
 インストラクションレジスタ, スタックアドレスレジスタが
 ある。stackの長さは16語とする。

命令語はstackから読み出されてIRレジスタに入
 り、デコードされて実行される。

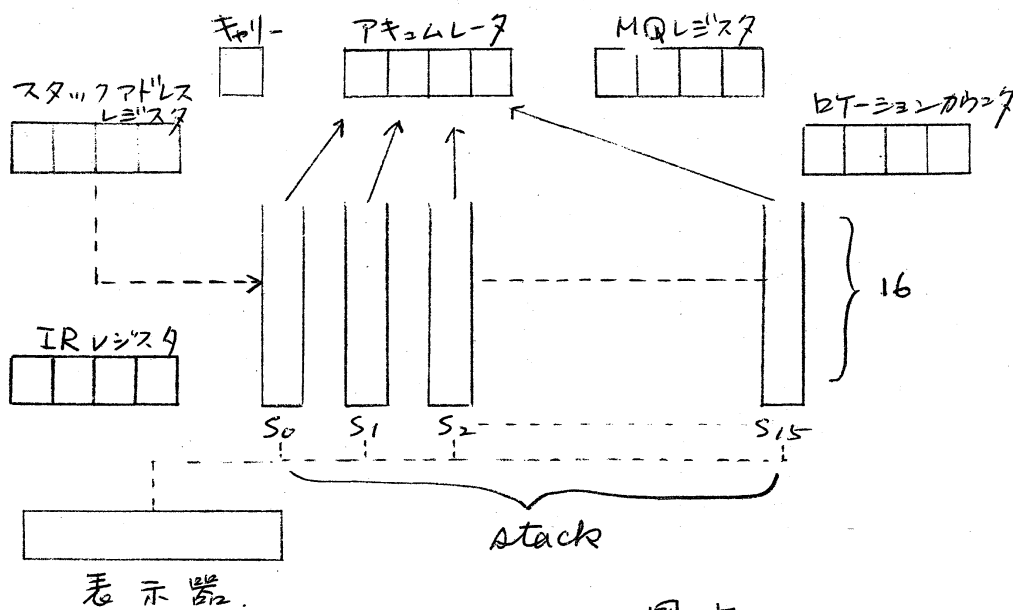


図 5

5.4 入出力装置

入出力装置として下図のような16キーを使用する。

0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

図 6

出力装置

1. stack の内容を表示する 16 桁以上の数字表示装置があり、命令によって 16 個の stack の任意のものに接続する。stack の一番下に入っているデータが一番上に表示されるようにしておく。

コード	表示文字
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	+
B	-
C	.
D	blank

2. 表示装置に表示されたデータはディジタルプリンタにプリントされるようにする。ディジタルプリンタを制御する命令はとくにおかない。

5.5 命令語

この 4 ビットマシンの命令語は 4 ビットより構成される。特定の命令につづく語は operand で、アドレス指定等に使われる数値データとみられる。

命令の種類と命令コードは次の通りである。

mnemonic	命令コード	機能
STOP	0	停止
ADR	1	stackの指定, この命令につづくコードで stackを指定する。
STACK	2	アキュムレータの内容を stackにつむ。 アキュムレータは 0 になる。
ADD	3	スタックの一番上のデータをキャリーとともにアキュムレータに加える。
SUB	4	スタックの一番上のデータをとり, キャリーとともにアキュムレータより引く。
MULT	5	スタックの一番上のデータをとり, アキュムレータの内容にかける結果を MQレジスタとアキュムレータにおく。
DIV	6	スタックの一番上のデータをとり MQレジスタとアキュムレータにある数を割り, 商を MQに余りをアキュムレータにおく。
RIGHT	7	(MQ) → ACC, (Carry) → MQ
LEFT	8	(ACC) → MQ, 0 → ACC
EMPTY	9	stackが空であれば, この命令の次のコードの数だけ命令をスキップする。
JUMP	A	この命令の次のコードで指定されたスタックの, そのスタックの最初のコードの数だけ次の命令にコントロールを移す。
SKIPH	B	(ACC) ≥ 0 のとき, この命令の次のコードの数だけ命令をスキップする。
RESET	C	キャリーをリセットする。
NOP	D	no operation
READ	E	キーよりデータを読み, A又はBがくるまで stackにつまみあげる。
SHOW	F	stackに表示器を接続する。

プログラムは stack 0 から実行される。1つの stack の命令語が処理され、終ると次の stack の処理が行われる。各 stack の最初のデータは必ずスキップされる。このデータは JUMP 命令でこの stack が指定されたときに、その stack に含まれている命令語のうちの何番目から実行に入るかを示す数が入っている。

5.6 プログラム例

キーより 2つの16桁以内の10進数を入力し2つの数の和を表示する。但し10進数の終りには A なるキーを打つものとする。

LOC	S0	S1	S2
0	8	15	0
1	ADR	STACK	3
2	3	ADR	STACK
3	READ	3	ADR
4	ADR	EMPTY	5
5	4	2	EMPTY
6	READ	JUMP	3
7	LEFT	0	ADD
8	ADR	ADR	JUMP
9	3	4	1
A	ADD	EMPTY	ADR
B	ADR	2	3
C	4	JUMP	STACK
D	ADD	0	STOP
E	ADR	ADD	END
F	5	ADR	

