MGP-21 のブートストラップ*

松永 均†
富山県立大学‡

MPG-21, product of Mitusbishi Precision, is a small computer in era late 1960s. This paper explains that's boot strapping.

1 始めに

私が入学した 1966 年ころ京都産業大学には 3 台のコンピュータがあった。(3 台コンピュータがあるから、略称をさんだいというのだというジョークがあった。) そのうちの一台がこの MGP-21 である。ちなみにあとの 2 台は TOSBAC-3400/model 40 と Gamma-10 である。本稿はこの MPG-21 の bootstrap について述べるものである。

2 特徴

MGP-21 は三菱プレシジョン社製の汎用超小型電子計算機である。オペレーション説明書の表紙裏には MGP-21 の特徴として

- 小型で床面積が少なくてすむ
- 特殊な温度調節や不要である
- 消費電力僅少
- オールトランジスタ化
- 豊富なアクセサリ
- 大きな記憶容量 (4096 語)
- 操作方法が簡単

といったことが書かれている。が、使ってみた立場から見た特徴は

- メインメモリがディスクである
- このため、オペランドによって演算速度が違う
- レジスタの内容をランプなどではなく、別売の 3 現象ディスプレイスコープを使って表示していた。
- 1 ワードは 32 bit であったが、LSB が word mark として使われてしまうので実質は 31 bit しか使えなかった。
- ビット単位のシフト命令は無く、データのシフトはダミーの入力装置から文字を入力するか (4 または 6 bit 左シフト) か、乗除算を使うしかなかった。
- 命令語のニーモニックは命令語のコードに 01 を加えた文字コードそのものである。
- アドレス等は 16 進数で表すのであるが、これに 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, f, g, j, k, q, w を使った。

といったところであろうか。

*Bootstrap of MGP21
†MATZNAGA Hitoshi
‡Toyama Prefectural University

3 ハードウェア

MGP-21 で使っていた文字コードは次のとおり。

bit	bit 56 (Uc Lc)			
1234	10	01	00	11
0000	)0	Zz		Space
0001	L1	Bb	Lower Case(LC)	_ -
0010	*2	Yy	Upper Case(UC)	= +
0011	"3	Rr	Color Shift	:;
0100	△4	Ii	Car.Return	?/
0101	%5	Dd	Back Space	[.
0111	$6	Nn	Tap	],
1000	@8	Mm		Vv
1001	(9	Pp	Cond Stop	Oo
1010	Ff	Uu		
1011	Gg	Tt		
1100	Jj	Hh		
1101	Kk	Cc		
1110	Qq	Aa		
1111	Ww	Ss		

見ればわかる通り、文字コードはアルファベット順にはなっていない。また Upper Case と Lower Case の切換えがあり、あるコードが何を表しているかは、そのときの状態による。L の小文字は数字の 1 と同一視されている。なお、このコード表の上位 2bit が 10 である列でアドレスなどの数値を、01 である列で命令を表すが、読込みルーチンは下位 4bit しか取り込まないので、見た目のわかりやすさ以上の区別は無い。

- メインメモリ: 32 track 128 sector のハード disk(論理的に 64 track 64 sector として扱う)
- 固定小数点演算
- 32bit/word (但し LSB は word mark として使われるので常に 0)

3.1 インストラクションセット

命令語の構成は次のとおり。

0	1 ~ 11	12 ~ 15	16,17	18 ~ 23	24 ~ 29	30,31
符号 bit		命令コード		トラック	セクタ	

命令を分類すると次のようになる。

- 演算命令
  - 加算 A(Add)
  - 減算 S(Subtract)
  - 乗算
    - 普通乗算 M(Multiply)
    - N 型乗算 N(N-Multiply)
  - 除算
- 論理演算命令
  - 抜取 E(Extarct)
  - 計算コース変更
    - 条件付き順序変更 T(Test)
    - 無条件順序変更 U(Unconditional Transfer)
- 停止およびスィッチテスト
  - 停止およびスィッチテスト Z(Zero)
- 記憶関係
  - 取り出し B(Bring)
  - 格納
    - 格納 H(Hold)
    - 格納 C(Clear)
- 番地部分修正
  - 番地部分修正 Y
  - 帰路設定命令 R(Set Return Address)
- 入出力操作命令
  - 入力
    - 6bit I(Input)
    - 4bit -I
  - 出力
    - 6bit O(Output)
    - 4bit -O
- TC スィッチテストおよびオーバーフローテスト
  - TC スィッチテスト -T
  - オーバーフローテスト -Z

これをコード順に並べなおすと次のとおり。

Symbol	Command	Binary	Hexadecimal	Decimal
Z	Stop, Sense and Transfer	0000	0	0
B	Bring	0001	1	1
Y	Store Address	0010	2	2
R	Set Return Address	0011	3	3
I	Inuput. Shift Left	0100	4	4
D	Divide	0101	5	5
N	Multiply, Save Right	0110	6	6
M	Multiply, Save Left	0111	7	7
P	Print or Punch	1000	8	8
E	Extract	1001	9	9
U	Unconditional Transfer	1010	f	10
T	Conditional Transfer	1011	g	11
H	Hold	1100	j	12
C	Clear	1101	k	13
A	Add	1110	q	14
S	Subtract	1111	w	15

これらの命令の実行には

Instruction	Optimum	Non-Optimum
Bring, Add, Subtract, Hold, Clear, Extract, Set Return Address, Store Address, Shift	7.26 ms	58.11ms
N-Multiply, M-Multiply, Divide	58.11 ms	108.96 ms
Unconditional Transfer, Conditional Transfer	1.59 ms	Each sector beyond optimum adds .40 ms
Sense	7.26ms or 14.52ms	

と、数ミリ秒から100ミリ秒程度の時間が必要である。メモリがディスクなのでオペランドによって実行時間が変わるが、命令によってオペランドの最適範囲が次のように違う。

Instruction	Distance from Instruction Location to Optimum Operand in Word-Times
Bring, Add, Subtract, Hold, Clear, Extract, Set Return Address, Store Address	2 through 16
N-Multiply	2 through 81
M-Multiply	2 through 79
Divide	2 through 78
Unconditional Transfer, Conditional Transfer	4 ore more if transfer is active
Others	always optimum

4 ブートストラップ

ここでは、オペレーションを表わすのに次のような略記を使用する。

MIM. マニュアルインプットモード
OOM. ワンオペレーションモード
NM. ノーマルモードにする
FC. フィルクリアスィッチを押す
EX. エグゼキュートスィッチを押す
SR. スタートリードキーを押す
IO. アイオースィッチを押す

4.1 bootstrap

まず、紙テープに

```
C002
-I0200
C003
C005
C004
-I0200
U002
Z000
```

のコードをパンチしておく。各行の末尾には stop code をパンチしておく。こうしておくと、Start Read Key を押したときに OKI typer の紙テープから文字が読み込まれ、stop code の前までがアキュムレータに入るのである。

次の一連の操作を行なうと、上記の紙テープに打たれた内容が0番地から4番地までに、しまわれ、2番地から実行が始まる。

```
MIM.
IO.
SR.             C000        → Acc
FC.                         → R
SR.             0000        → Acc
OOM. EX. MIM.               → 0000 番地
SR.             C000        → Acc
FC.                         → R
SR.             800I0200
OOM. EX. MIM.               → 0002 番地
SR.             C005
OOM. EX. MIM.               → 0003 番地
SR.             800I0200
OOM. EX. MIM.               → 0004 番地
SR.             U0008       → Acc
FC.                         → R
SR.             EXECUTE     → Acc
OOM.EX.         0002        → C
NM.
```

やっていることをおおざっぱに言えば

- Clear 命令を命令レジスタに入れる。
- 命令をアキュムレータに読み込む。
- 命令レジスタにある Clear 命令を実行する
- という全体を繰り返して、メモリの先頭に入れ、最後に Transfer 命令で、入れた命令に制御を移す。

0番地の内容は無意味なので、これを省いた次のようなバージョンもあった。実際にはもっと短いバージョンがあったはずなのであるが、資料を失ったため、本稿では記述できなかった。

```
MIM.
SR.              000C0008    → Acc
FC.              (C002)      → R
SR.              800I200     → Acc
OOM. EX. MIM.    (-I0200)    → 0002 番地
SR.FC.SR.        C005
OOM. EX. MIM.                → 0003 番地
SR.FC.SR.        800I0200
OOM. EX. MIM.                → 0004 番地
SR.FC            U0008       → R
SR.              EXECUTE     → Acc
OOM.EX.          0002        → C
NM.
```