BASIC 言語による COMET シミュレータ

内野俊治・岩城常夫

COMET simulater using Basic language

Toshiji UCHINO and Tsuneo IWAKI

Abstract

This paper describes a simulation program for COMET, a simplified imaginary computer with only 23 machine code instructions. The original program written in the personal computer Basic language is published from a book company, and our efforts have been directed to fit the original one to be used in our machine language training course. The new program has a feature that permits single step execution of each instruction with the display of extensive associated information, such as contents of all general registers, value of effective address and its content as required, before and after the execution of each instruction. Also it permits to dump each content of memory starting from specified address on the display in the form of deassembled instruction or, alternatively, data (as decimal number plus corresponding character), the selection of which is decided automatically by the program, plus in the form of straight hexa-decimal number simultaneously. This permits easy and spontaneous evaluation of memory contents. The new program has been actually used in the first training course, and favorably appreciated by the students.

1. まえがき

昭和60年4月，筆者の一人が当大学に奉職したとき，近く情報工学科ができるので，そこでコンピュータの機械語の講義を担当し，できればコンピュータを使った演習も加えるようにとのことであった。しかし当時は，どんなCPUの機械語をコンピュータで演習するようにしたらよいか，とくにそれに用いる適当な計算センター用のプログラムを一体どうしたらよいかが分からず，永い間，最も頭の痛い問題として続くことになった。

結局，講義は，これを立案された中村教授の御意見等に従って，情報処理技術者試験に使用される仮想コンピュータCOMETの機械語を取り上げることになった。そこでCOMETに関する参考書を集めてみると，そのいくつかの中に，パーソナルコンピュータを使ったCOMET用のシミュレータプログラムが掲載されていることが分かった[1)~3)]。これらはBASIC言語で書かれており，実際に打ち込んでRUNしてみると，それぞれに特色があり，演習用としても充分使用できるのではないかと思われた。

一方，本学の方もパーソナルコンピュータ導入の機運が進み，適当なプログラムさえあれば講義が始まる63年4月頃までには，対象になる学生一人一人がそれぞれ自分のコンピュータを専有し，自由に演習できるという見通しが得られるようになった。

そこでこれらの参考書の中から教科書[1)]を選定し，それに付録としてつけられているCOMETシミュレータを演習用に使おうと決心した。ところが，講義も間近になって，真剣にこれを使用した演習を考えてみると，下記の理由で多少手を加えた方が一層，目的によく合うのではないかという気がしてきた。

原プログラムは，情報処理技術者試験のために自ら積極的にこの本を購入して自習する読者が，自分で作成したプログラムを確認するのに用いるのをその目的としている。このような読者に対してはシミュレータプログラム自体を解読することもよい勉強になるので，徒らに複雑な長いプログラムよりも，基本機能が完備していて簡潔で解読しやすいプログラムの方が望ましい。この点，原プログラムは著者のこのような意図を充分満足していると思われる。一方，我々は，機械語の講義と演習を通して，コンピュータの機械語を

昭和63年9月28日受理

なるべく多くの学生に分かりやすく理解させることを大きな目的の一つにしている。このためにはプログラム自体の複雑さは問われず，プログラムの実行が上述のような目的にできるだけ適していることの方が望ましい。

結局，このような観点から，我々は原プログラムに手を加え，できるだけこの目的に近ずけるような努力をすることになった。これらの変更は，表示形式とか取扱い易さとか便利さとかといったむしろ副次的な点に関するもので，プログラムの骨格ともいうべき基本部分は，ほとんど原プログラムによっている。しかし，まるっきり他人の手になる借り物を不満を感じながら使うよりも，できれば自分たちの手をかけて気にいるようにしたものを用いる方が講義や演習に対する情熱もそれだけ増すものである。このような内容で，貴重な紙面を穢すのは甚だ申し訳ないが，一つの努力の経験として発表させていただく次第である。

2. COMETシミュレータの概要

最初にこのシミュレータの概要を説明する。このシミュレータはBASIC言語で書かれていて，アセンブラとシミュレータが一体となっている。

プログラムをRUNすると，Fig. 1に示すメニュー画面が表示され，[1]から[8]まである中の一つの機能を選択することを要求する。各機能の内容は以下の通りである。

Assem[1]：COMETのアセンブラ言語(CASL)で書かれたソースプログラムをアセンブルしてCOMETのメモリに格納する機能。

Mem[2]：COMETのモメリの指定したアドレスの内容を表示し，また書き換える機能。

Reg[3]：COMETの汎用レジスタの内容を表示する機能。

Break[4]：COMETの実行するプログラムにブレークポイントを設定し，またそれを解除する機能。

Exec[5]：COMETに機械語のプログラムを実行させる機能。

Cont[6]：COMETに機械語のプログラムを中断した所から再開させる機能。

Dump[7]：COMETのメモリ内容を指定したアドレスから読みだして種々の形式で表示する機能。

End[8]：COMETシミュレータからBASICへ制御を戻す機能。

さて，Fig. 1のメニュー画面が表示されたら，最初は必ず，[1][↵]の入力によってAssem[1]の機能を選択する。これにたいしてプログラムはアセンブルすべきソースプログラムのファイル名の入力を要求する。

CASLで書いたソースプログラムは，あらかじめ，BASICのエディタを利用してBASICのコメント文の形で作成し，適当なファイル名をつけてアスキイセーブしておく必要がある。

プログラムの要求に応じて，あらかじめセーブしてあるソースプログラムのファイル名を入力すると，このファイルを読みだしながら次々の命令を機械語に変換し，2回のファイル読みだし（2パス）処理によって機械語プログラムを完成し，これをCOMETのメモリ中に格納し，さらにソースプログラムに用いた各ラベルにたいするアドレスの一覧表を表示して，最初のメニュー画面に戻る。

さて，COMETがこのプログラムを実行するにあたって，プログラム中に設定または変更すべきデータがある場合には，次に，Mem[2]の機能を選択してCOMETのメモリ内データの設定または変更を行う。またディバッグ等のため，プログラムの実行をある命令で一時停止したい場合には，Break[4]の機能を用いてその命令のラベルにブレークポイントを設定する。

以上のようなメニュー画面の各機能は，それぞれの処理が終了すると自動的に，またはSTOPキイを押すことでもとのメニュー画面に戻り，再び次の任意の機能の選択を新たらしく要求するようになっている。

さて，いよいよExec[5]の機能を選択してCOMETに機械語のプログラムを実行させることになるが以下が我々が主として手を加えた部分になるので，節を改めて説明することにする。

```
Assem[1], Mem[2], Reg[3], Break[4], Exec[5], Cont[6], Dump[7], End[8]
機能 [1-8] ?
```

Fig. 1.　CRT display of function menu

3. 手を加えた主な部分

3.1 Exec[5] および Cont[6] の機能

前述のメニュー画面の表示にたいして，[5]↵と入力すると，COMET の機械語プログラムの実行処理に入いる。

これには大きく分けて，

（A） 途中経過を全く表示せず（但しプログラム中に出力[OUT]命令があるときは，この出力を表示）機械語プログラムの終了（EXIT 命令）までを最高速度で実行するモード。

（B） 進行キイ（例えばスペースキイ）を押すごとに一命令ずつ実行して途中経過を表示しつつステップ実行するモード。

（C） 上述の（B）と同じ内容を表示しつつ連続的に実行する連続経過表示モード。

これら各モードはさらに，印字しますかの質問に対して[Y]↵と応答することによって，CRT 表示と同じものをプリンタに出力することもできる。

これらの指定は Fig. 2 のフローチャートに従って行われる。入口名 ？ にたいしては，機械語プログラムの開始番地のラベルを入力するが，最初からスタートするプログラムについては単に↵を入力するだけでよい。以下の質問に対する[N]↵（No の意味）の応答は N がディフォールトとなっていて単に↵と省略できるので，通常のプログラムを（A）の最高速で実行する場合には簡単に↵を 3 回入力すればよいようにしてある。

次に，（B）のステップ実行のモードを選択する場合には，1 ステップずつ実行しますか[Y/YS/N] ？ の質問に対して[Y]または[Y][S]で応答する。

さて，このプログラムの特徴の一つは，機械語プログラム進行過程の表示形式を以下のようにしたことである。

Fig. 3 に原プログラムのプログラム実行途中における表示形式を示す。

Fig. 4 に[Y]で応答したときの本プログラムの対応する部分の表示形式を示す。図で分かるように以下のような変更が加えられている。

（ I ） OP コードの変更（これについては 3.3 で述べる。）

（ II ） 各機械命令を OP コード部分ばかりではなく命令全体をディアセンブルして表示。

（III） 汎用レジスタの内容ばかりでなく，実効アドレスが必要な命令（POP, RET 以外の命令）にたいしては実効アドレスを，また実効アドレスの内容までも必要とする命令（LD, ST, ADD, SUB, AND, CPA 等）にたいしては実効アドレスの内容の表示を追加。

（IV） 表示データはすべて（16 進数表示欄を除き）10 進数に変換して表示。

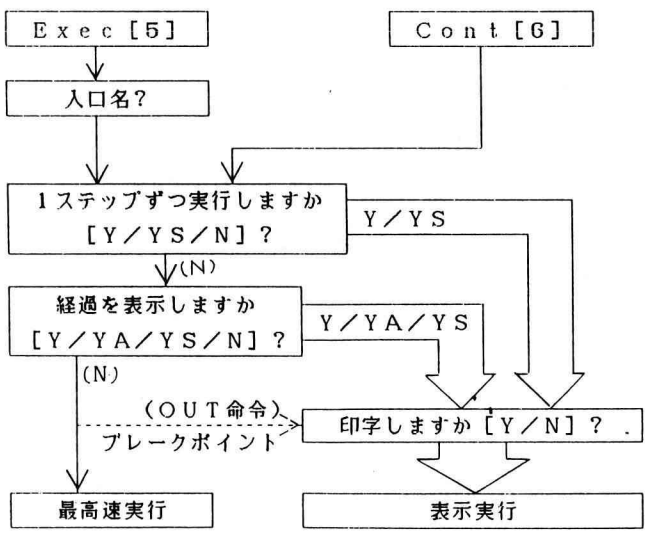

Fig. 2. Flowchart illustrating the sequence of executing mode selection

*1	*2	*3	*4	*5	*6	*7	*8	*9
000C	1200 0006	JNZ	0000	0000	0000	0000	0000	M
0006	0300 0012	ADD	0000	0000	0000	0000	0000	M
0008	0A11 0001	LEA	004B	0000	0000	0000	0000	P
000A	0810 0013	CPA	004B	0001	0000	0000	0000	P
000C	1200 0006	JNZ	004B	0001	0000	0000	0000	M
0006	0300 0012	ADD	004B	0001	0000	0000	0000	M
0008	0A11 0001	LEA	0096	0001	0000	0000	0000	P
000A	0810 0013	CPA	0096	0002	0000	0000	0000	P

(注)
*1 命令アドレス. *2 機械語命令の16進表示.
*3 OPコード. *4 ～ *8 GR0～GR4の内容.
*9 FRの内容.

Fig. 3. CRT display of single step execution mode of original program

*1	*2	*3	*4	*5	*6	*7	*8	*9	*10	*11
12	#6200 #0006	JNZ 6	0	0	0	0	0	M	6	
6		AFTER EXECUT.	..	..	..	..	..	.	6	
6	#2000 #0012	ADD 0, 18	0	0	0	0	0	M	18	75
8		AFTER EXECUT.	75	..	..	..	..	P	18	75
8	#1211 #0001	LEA 1, 1,1	75	0	0	0	0	P	1	
10		AFTER EXECUT.	..	1	..	..	..	P	1	
10	#4010 #0013	CPA 1, 19	75	1	0	0	0	P	19	2
12		AFTER EXECUT.	..	..	..	..	..	M	19	2
12	#6200 #0006	JNZ 6	75	1	0	0	0	M	6	
6		AFTER EXECUT.	..	..	..	..	..	.	6	
6	#2000 #0012	ADD 0, 18	75	1	0	0	0	M	18	75
8		AFTER EXECUT.	150	..	..	..	..	P	18	75
8	#1211 #0001	LEA 1, 1,1	150	1	0	0	0	P	2	
10		AFTER EXECUT.	..	2	..	..	..	P	2	
10	#4010 #0013	CPA 1, 19	150	2	0	0	0	P	19	2

(注)
*3 ディアセンブルした機械語命令. *10 実行アドレス.
*11 実行アドレスの内容. 他はFig. 3と同じ.

Fig. 4. CRT display of single step execution mode specified by "Y" input

さらにこれらの表示は，各機械語命令の実行前における値を表示して停止するようにし，進行キーを押すごとに，この機械語命令の実行によって更新されるべきデータだけをAFTER EXECUT.の行に表示して次の命令に進み，その命令と実行前におけるデータとを表示した状態でこの命令に停止するようにした。また，CRT面上における表示は欄ごとに適当な色分けを行ってなるべく一目で見分けられ易いようにした。

これらの変更は，初心者でも各機械語命令の実行によるCOMETの動作をできるだけ分かり易く理解できるようにするためのものである。

表示形式をこのようにすると，それだけ処理時間が長

```
Assem[1], Mem[2], Reg[3], Break[4], Exec[5], Cont[6], Dump[7], End[8]
機能 [1-8] ? 5
入口名 ? MULT1
1ステップづつ実行しますか [Y/YS/N]? YS
印字しますか [Y/N]?

  Ø #12ØØ #ØØØØ  LEA Ø,   Ø       Ø  ..  ..  ..  .. Z    Ø
  2 #121Ø #ØØØØ  LEA 1,   Ø      ..   Ø  ..  ..  .. Z    Ø
  4 #64ØØ #ØØØA  JMP      1Ø     ..  ..  ..  ..  .. .    1Ø
 1Ø #4Ø1Ø #ØØ13  CPA 1,   19     ..  ..  ..  ..  .. M   19   2
 12 #62ØØ #ØØØ6  JNZ       6     ..  ..  ..  ..  .. .    6
  6 #2ØØØ #ØØ12  ADD Ø,   18     75  ..  ..  ..  .. P   18  75
  8 #1211 #ØØØ1  LEA 1,   1,1    ..   1  ..  ..  .. P    1
 1Ø #4Ø1Ø #ØØ13  CPA 1,   19     ..  ..  ..  ..  .. M   19   2
 12 #62ØØ #ØØØ6  JNZ       6     ..  ..  ..  ..  .. .    6
  6 #2ØØØ #ØØ12  ADD Ø,   18    15Ø  ..  ..  ..  .. P   18  75
  8 #1211 #ØØØ1  LEA 1,   1,1    ..   2  ..  ..  .. P    2
 1Ø #4Ø1Ø #ØØ13  CPA 1,   19     ..  ..  ..  ..  .. Z   19   2
 12 #62ØØ #ØØØ6  JNZ       6     ..  ..  ..  ..  .. .    6
 14 #11ØØ #ØØ14  ST  Ø,   2Ø     ..  ..  ..  ..  .. .   2Ø 15Ø
 16 #9ØØØ #ØØØØ  EXIT

Assem[1], Mem[2], Reg[3], Break[4], Exec[5], Cont[6], Dump[7], End[8]
機能 [1-8] ?
```

Fig. 5. CRT display of single step execution mode specified by “YS” input

くなる筈であるが，もともとこのような表示形式を用いる実行が要求される場合には，人間が各表示内容を理解できる以上にいくら速くしても意味がないので，上の変更による処理時間の増加は，実際の使用上全くディメリットにならない程度ですんでいる。

さて，一命令ずつの実行を分かり易く表示するには上述の表示が適しているが，プログラムの流れを CRT 画面で追う場合には，同じ画面中にできるだけ多くの命令の実行結果が見やすい形で表示されていることの方が望ましい。そこで本プログラムでは，前述の 1 ステップずつ実行しますか ? にたいする応答として上述の Y ↵ のほかに，Y S ↵ の応答を追加した。これは各機械語命令ごとに命令とその実行結果だけを簡潔に表示するものである。この場合におけるメニュー画面の選択から，前と同じプログラムを最初から最後まで実行して再びメニュー画面に戻るまでの CRT 表示を Fig. 5 に示す。

一般に従来のプログラムでは，各命令ごとにその実行前の状態が表示され，ある命令の実行結果は次の命令の行で表示されるのが普通であるが，本プログラムのように実行アドレスの内容までも表示しようとする場合には，この方法では最も重要な ST（ストア）命令を適切に表示することができなくなる（つまり，ストアすべき実効アドレスは通常次の命令には現れないので，ストアで更新されたメモリの内容が全く表示されないことになる）。このため一方だけの表示の場合には，処理時間さえ問題にならなければ，このように各命令の実行後のデータを表示する方が有効と思われる。また，ここでの表示は重複する内容を省略してできるだけプログラムの流れを見易くなるようにしている。

さて次に，(B) の連続経過表示モードは，以下のような表示を行う。

プログラム側からの「経過を表示しますか [Y/YA/YS/N] ?」の質問に対して，Y A ↵ で応答すると，前述の Fig. 4 に示した表示を行間をつめて連続表示しつつプログラムが連続して進行する。

Y S ↵ で応答すると，前述の Fig. 5 の連続表示でプログラム進行する。

また，Y ↵ で応答すると，前述の Fig. 4 の実行前の表示行だけを連続表示する。

このように，各表示形式のいずれでも自由に選択できるようにした。これらの表示形式においては，前述のプログラムの進行キイは，プログラムの進行を任意

の命令で一時停止するのに使用できる。一時停止したプログラムをそこから再開するにはもう一度このキイを押せばよい。また一時停止した状態におけるメモリ内容等の各種データを調べる場合にはSTOPキイを押してメニュー画面に戻し，後述のDump[7]の機能等を用いて自由に行うことができる。

このようにしてメニュー画面まで戻したプログラムを中断した所から再開するには，Cont[6]の機能を選択する。これによりFig. 2に示すように，「入口名 ?」に対する入力をバイパスして（この場合の開始番地は当然プログラムを中断した所となるのでこの指定は不必要である），上述のExec[5]で述べた任意の表示形式を，この点から新しく指定しなおしてプログラムを再開することができる。

また，プログラム途中における表示形式の変更は，その点にブレークポイントを設定し，プログラムをブレークポイントで停止させてメニュー画面に戻し，Cont[6]の機能で再開させることによっても自由に行うことができ，表示形式の見易さ，および選択の自由さと相まって，ディバック等に便利に使用できる。

3.2 Dump[7]の機能

COMETのメモリ内容にアクセスするために既述のMem[2]の機能があるが，本プログラムではメモリ内容の読みだしの機能を強化するためDump[7]の機能を追加した。

これは，指定した任意のアドレスからのCOMETメモリの内容を読みだして，これを

（A） ディアセンブルして命令として，または10進数に変換してデータとして表示。

（B） 16進数としてそのまま表示。

（C） 上記（A）の中の10進数データを，対応するコードをもつ文字として同時に表示。

の3通りの表示を，Fig. 6に示すように並列に同時に行うようにしたものである。

特に断わらなければ（A）は指定されたアドレスからのメモリ内容を2ワードずつディアセンブルして命令として表示する。もし対応するOPコードがなくて命令としてディアセンブルできない場合には，自動的にこれをデータとみて，1ワードずつ10進数に変換して表示するとともに，データの値が16進数の20から5Fの範囲にあるときには対応するコードをもつ文字に変換して並列に表示する（この範囲外の場合には・を表示）。

このようにすると，任意のデータがある場合には，データを命令として表示するおそれがあるので，（A）をすべて10進数として表示すための指定も設けてある。しかし，2バイトで構成されるデータの中で命令となるのは比較的絶対値の大きな特別な値をもつ約30個の数（3.3参照）に限られるので，間違いの起こる確率は非常に小さく（0.05%），この自動切り替え表示機能は便利に使用できる。

また，アセンブル時にラベルをつけたアドレスは，ラベルも同時に表示するようにしている。

```
開始番地－〔終了番地〔／IorD〔／P〕〕？ 0-

 0 MULT1  LEA 0,     0      #1200 #0000
 2        LEA 1,     0      #1210 #0000
 4        JMP        10     #6400 #000A
 6 M1     ADD 0,     18     #2000 #0012
 8        LEA 1,      1,1   #1211 #0001
10 M2     CPA 1,     19     #4010 #0013
12        JNZ         6     #6200 #0006
14        ST  0,     20     #1100 #0014
16        EXIT              #9000 #0000
18 M           75     K     #004B
19 N            2     .     #0002
20 P          150     .     #0096
21              0     .     #0000

開始番地－〔終了番地〔／IorD〔／P〕〕？
```

Fig. 6. CRT display of COMET memory contents as shown using Dump [7] function

Table 1. Instruction format and examples of assumed OP codes of COMET
（OP の数値は 16 進表示）

0	4	8	12	16	31 ←— ビット番号		
第　語　語				第 2 語	命令語とアセンブラとの対応		
OP		GR	XR	adr	アセンブラ命令		意　味
主 OP	副 OP						
0	0				未使用		
1	0				LD	GR, adr, XR	load
	1				ST	GR, adr, XR	store
	2				LEA	GR, adr, XR	load effective address
2	0				ADD	GR, adr, XR	add arithmetic
	1				SUB	GR, adr, XR	subtract arithmetic
3	0				AND	GR, adr, XR	and
	1				OR	GR, adr, XR	or
	2				EOR	GR, adr, XR	exclusive or
4	0				CPA	GR, adr, XR	compare arithmetic
	1				CPL	GR, adr, XR	compare logical
5	0				SLA	GR, adr, XR	shift left arithmetic
	1				SRA	GR, adr, XR	shift right arithmetic
	2				SLL	GR, adr, XR	shift left logical
	3				SRL	GR, adr, XR	shift right logical

表示の制御は以下のように行なっている。メニュー画面から 7 ↵ でこの機能が選択されると，Fig. 6 の 1 行目の ？ までに示す質問が表示される。これに対して 1 0 ↵ と入力すると，10 番地からの内容がスペースキイを押すごとに 1 行ずつ表示され，1 0 − ↵ と入力すると，10 番地からこのプログラムに割り当てられた最終番地まで表示が連続して進行する（但しスペースキイによって進行の停止・再開は自由にできる）。1 0 − 2 0 ↵ で 10 番地から 20 番地の間を連続表示して停止する。／D を指定すると前述の（A）をデータ表示とし，／／P を指定すると CRT 表示と共にプリンタから印刷結果が出力される。

Fig. 6 はこの機能を用いて，プログラムとデータを含む領域を 0 − ↵ として表示したものを示す。

3.3　OP コードの変更

COMET, CASL 仕様書の参考資料として定義されている命令語の構成の一部を，Table 1 に示す。本プログラムの機械語はこれに従うこととした。　これに対して原プログラムでは，ほぼ Table 2 に示すような 1 から始まる連続した数字を OP コードに割り当てている。

この結果，ON OP GOTO ＊(OP=1 に対する処理ルーチン開始番地のラベル)，＊(OP=2 に対する処理ルーチン開始番地のラベル)，＊(OP=3 に対する処理ルーチン開始番地のラベル)，……というタイプの BASIC のジャンプ命令が，COMET の機械語命令デコーダとして効果的に利用できる。

本プログラムもこの特徴を利用するため，シミュレータが命令実行のために COMET のメモリから読みだした Table 1 に示す機械語の OP を，直ちに対応する Table 2 のコードに変換して上述のデコーダを用いるという方法をとった。この変換には，Table 1 の OP コードを，その 16 進数の 1 桁目（下位桁）と 2 桁目（上位桁）とを異なる次元とした 2 次元配列として取扱かい，Table 3 に示すような変換マトリクス OPMTRX（OPH, OPL）を作り，Table 1 で書いた OP コードの値を OP1, Table 2 で書いた対応する OP

Table 2. Examples of continuously allocated OP codes

OP	アセンブラ命令	OP	アセンブラ命令	OP	アセンブラ命令
01	LD	0A	CPL	13	JMP
02	ST	0B	SLA	14	PUSH
03	LEA	0C	SRA	15	POP
04	ADD	0D	SLL	16	CALL
05	SUB	0E	SRL	17	RET
6	AND	0F	JPZ	18	EXIT
07	OR	10	JMI	19	IN
08	EOR	11	JNZ	1A	OUT
09	CPA	12	JZE		

Table 3. Transformation matrix OPMTRX (OPH, OPL)

OPH \ OPL	0	1	2	3	4
1	(LD) 01	(ST) 02	(LEA) 03	— 00	— 00
2	(ADD) 04	(SUB) 05	— 00	— 00	— 00
3	(AND) 06	(OR) 07	(EOR) 08	— 00	— 00
4	(CPA) 09	(CPL) 0A	— 00	— 00	— 00
5	(SLA) 0B	(SRA) 0C	(SLL) 0D	(SRL) 0E	— 00
6	(JPZ) 0F	(JMI) 10	(JNZ) 11	(JZE) 12	(JMP) 13
7	(PUSH) 14	(POP) 15	— 00	— 00	— 00
8	(CALL) 16	(RET) 17	— 00	— 00	— 00
9	(EXIT) 18	(IN) 19	(OUT) 1A	— 00	— 00

コードの値をOP2とすると，

```
OPL=OP1 AND & H0F
OPH=(OP1 AND & HF0)¥16
IF OPH<0 THEN OPH=OPH+16
OP2=OPMTRX(OPH, OPL)
```

の命令群でTable 1のOP1からTable 2のOP2に変換して用いている。このような直接的な（サーチを用いない）変換方法をとることによって，この変換にともなう時間の遅れを，最高速実行の場合にも実質的に無視できる程度にすることができた。

4. あ と が き

以上のように，このプログラムは情報処理技術者試験対策用というよりむしろ，COMET という簡単なコンピュータを用いて，使用者がコンピュータの機械語による動作をできるだけ易しく理解できるようにすることを目標としたものである。このため各機械語命令の実行段階における表示をできるだけ分かり易く丁寧にし，また必要な場合にはメモリ内の情報を容易に自由に知ることができるようにした。この他にもこの目的に対し少しでも使用し易くなると考えた点はできるだけ手を加えるようにしたつもりである。

これらによって我々の目標は一応達せられらと考えている。ただしこのために原プログラムの約 1.5 倍の長さになり，原プログラムのもつ解読の容易さという特徴をそれだけ低下させることになった。

このプログラムは実際に機械語の講義の演習用として使用されたが，最後に行った無記名のアンケート調査によると，過半数の学生がこれによる演習に興味をもち，またこのプログラムの性能に満足しているという結果が得られた。

終わりに，これを用いた演習の実現に関連して種々御援助をいただいた情報工学科木名瀬，中村，北條の各教授，徳弘助手ならびにシステム工学科山田科長，田口助教授，平山（弘）講師，佐々木助手，および原プログラムの変更使用およびその発表を了承された著者ならびに出版社[1] の方々に厚く感謝致します。

参 考 文 献

1） 玉井　浩，「明解 COMET & CASL」，サイエンス社
2） 杉原敏夫，「COMET と CASL」，オーム社
3） 下條哲司，「CASL」，オーム社