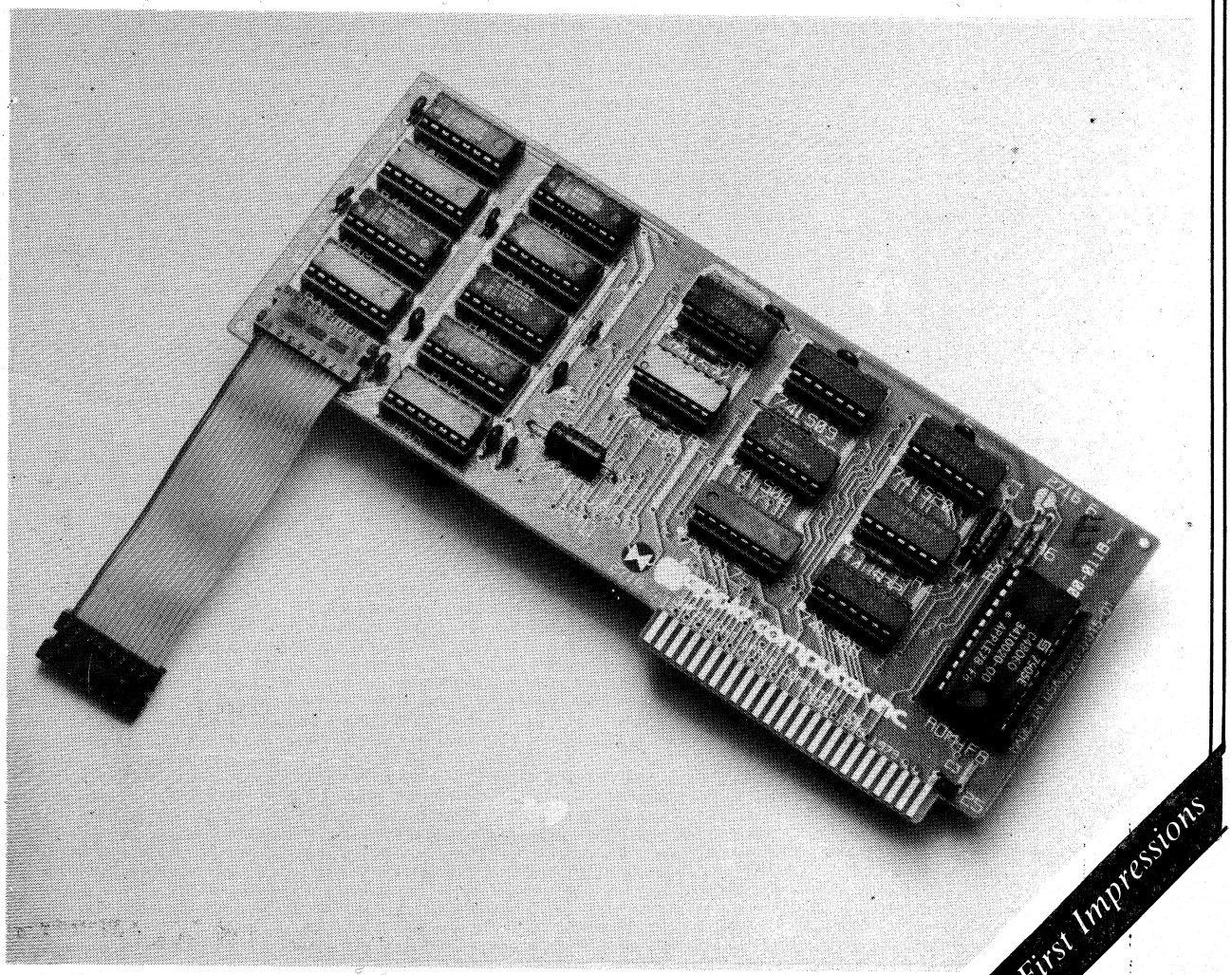
Neal Crook.

November 1979 First Edition 50p

LIVERPOOL SOFTWARE GAZETTE

APPLE PASCAL

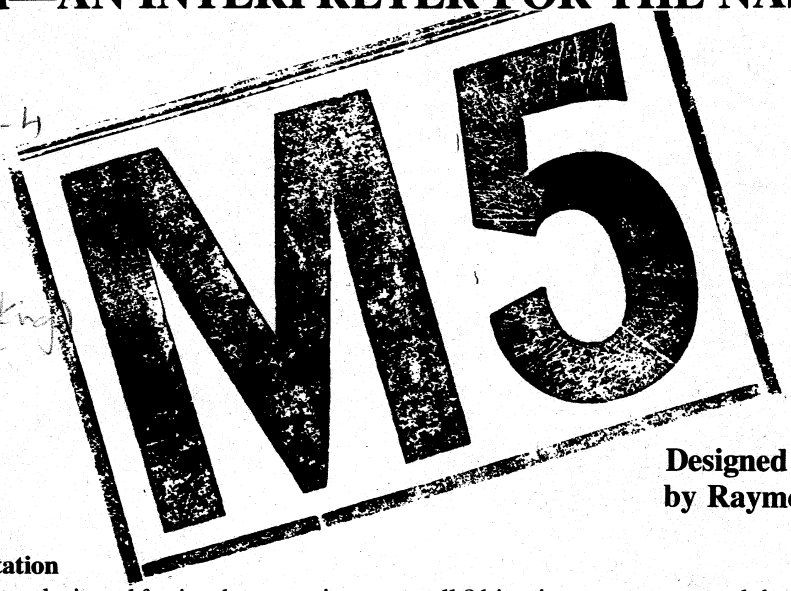


First Impressions

5: A high level language in 3/4 K!

M5 SYSTEM—AN INTERPRETER FOR THE NASCOM ONE

See also
Dr Darks Diary - 4
CT May 79
& Blue file
(disassembly linking)



Designed and Implemented by Raymond Anderson

0.0 The M5 Language

0.1 Nascom Implementation

The M5 interpreter was designed for implementation on small 8 bit microcomputers and the Nascom one standard system was an ideal choice because of its popularity and use of a fairly powerful processor (the Z80).

With only about 940 bytes available to the user, the language had to be compact enough to write decent programs in a small space, and also have a small interpreter to leave the maximum amount of spare memory. A simple editor was almost essential if programs of over about 50 bytes were to be written and debugged easily, and this required about 100 bytes.

The editor, interpreter and command mode are closely linked—for example, program variables are maintained over edits, and resets, and the editor will set up its cursor to inform the user where an error occured.

A compact M5 program can be difficult to follow initially, so error routines which give the exact location and type of a run-time error are included, despite the penalty in RAM usage. (Execution speed is unaffected by error checking).

M5 is a very fast interpreter, although loops are not as fast as in machine code because each loop involves a small search. A well written M5 program will carry out general calculations at about 1/3-1/5 of the speed of machine code. (M5 programs are usually much faster to write and debug of course!)

The user may write programs of about 230 bytes in length—quite large in M5. Overlarge programs may cause trouble when entered, but the most likely indication of an overflow is a lot of garbage appearing on the end of the program when it is listed.

0.2 Introduction

The M5 system is entered by typing EC60 when M5 has been entered into user RAM. The prompt 'M5:' should then appear at the bottom of the screen, indicating that the system is in the command mode. Commands which may be entered now are:

I	Input	a new program and destroy the previous one. System responds with a newline and waits for the user to enter a program. Input is terminated by a semi-colon, which returns the user to the command made.
L	List	the program currently in store and return to command mode.
R	Run	the current program starting at the first symbol, after printing a newline.
E	Edit	the current program, inserting the character pointer at the place the last instruction was executed—or where an error was found. (See section on editor.)
RS	RESET	the Nascom. This will cause a return to Nasbug. However, the current program and value of X will be maintained ready for typing EC60 to resume programming. RESET must also be used to star a looping program.

0.3 Initialisation

When entering M5 for the first time after loading it, it is best to initialise the user work area by entering and running a null program. This is done as follows: (Underlined characters are typed by the system.)

M5:Input
: (I.E. Terminate input after entering nothing!!!)

M5:R (Null program simply results in a carriage return.)

M5: (System is now initialised.)

0.4 Other commands

M5 will respond with a new prompt to any unknown command letter.

0.5 Errors on input

A backspace will delete the last character only when in input mode. It may seem misleading if used to backspace up a line. (Try it and see!)

Backspaces can be inserted into a string in the program by using the INSERT command in EDIT mode.

Semicolons are illegal characters inside an M5 program.

Shift-Backspace is a legal character in strings.

1.0 BASIC M5 LANGUAGE PRINCIPLES

1.1.0 M5 Arithmetic

The basic elements handled in standard M5 are 16 bit unsigned integers, which are adequate for most games and simple simulation or number manipulation. Numbers are in the range 0 - 65535 (decimal) and are modulo 65536 so 65536 seems the same as zero to the language.

Operators permitted in M5 are:

* (multiply) / (divide) + (add) – (subtract) # (-1) & (+1)

the last two are included for faster execution if required, and for compact programming of loop control. (See later).

1.1.1 The Stack

An important aspect of M5 which is quite powerful once it is understood, is its stack based (Reverse polish) expression analysis. This system requires no parentheses and it can be used to evaluate arbitrary expressions quickly. The M5 algebraic system is similar to that found on some calculators and the analogy with a calculator is used in these notes.

1.1.2 The Current Value

On a pocket calculator, the idea of a current value is easy to understand as it appears on the display and is often called "x". In M5 there is also a current value (called "X"), and it is altered only in the following circumstances:

1) If a number appears in the program (not in a string) x takes its value.
2) On encountering an identifier A-Z, x takes the value stored there.
3) On encountering a ? (not after =) x takes its value from the keyboard.
4) After a diadic operator (/ - + *) x becomes the result.
5) If x is incremented or decremented (using & or #).

1.1.3 Variables

As in most other languages, M5 has variables A-Z, and a special one @.
One of these variables becomes current by simply quoting it in the program.
(point 2 above).
X may be stored in asvariable by simply using =k where k is a variable name.
If = ? is used, the current value (x) is displayed as a decimal number on the screen. (This is how numbers are output in M5).
EXAMPLES (These are all legal M5 programs–Try if unsure!)
(i) A What is in location A is now also in x (the current value).
(ii) ABC x takes on the values in A then B then C and keeps the value C.

(iii) 23 x becomes 23.
(iv) 23A x becomes 23 , then x becomes A (i ė. the number in A).
(v) 23 456 x becomes 23 and then x becomes 456.
(vi) A=B x becomes A , then this value is stored in B.
(vii) A=B=C=D x becomes A , then this value is put into B , C and D.
(viii) A=? D=? x becomes A and this is displayed, then x becomes B and is displayed.
(ix) =?=A what is in x (left from last program) is displayed and put in A

N.B. If you want to check what is going on, put the characters: =? in your program and x at these points will be printed.
For neatness and readability use: =? " " which separates No's by a space.
E.G. 23=?" " 1 1 1 1 1 =?". will produce: 00023 11111 as output if run.

1.1.4 Calculating

When a comma is encountered in an M5 program, the value of x is put on the top of the stack– pushing down all other members.
We can represent the stack diagramatically to show what happens.

Imagine the M5 program A,33,,BA where initially A=1 and B=2
step: abcdefgh (could have run 1=A2=3 before)

and follow it step by step:

STEP	SYMBOL	MEANS	x	top–STACK–bottom–>	y (top element of stack)
a	A	load A	1	– – – –	unknown
b	,	push x	1	1 – – –	1
c–d	33	load 33	33	1 – – –	1
e	,	push x	33	33 1 – –	33
f	,	push x	33	33 33 1 –	33
g	B	load B	2	33 33 1 –	33
h	A	load A	1	33 33 1 –	33

Note that the top member of the stack is called y .

So far, we have no means of removing items from the top of the stack. We do this by using operators such as + / * - .

The operators work on x and y and put the result in x, removing y from the stack.
Operators therefore do the following:

Operator	Function	Remarks
#	x := x–1	This is the pound sign on the Nascom
&	x := x+1	Much faster than ,1+ which is equivalent
+	x := x+y	y is lost. Overflow not detected (M5 2.1)
–	x := y–x	y is lost. Underflow not detected.
*	x := x*y	y is lost. Overflowing bits put in @
/	x := y/x	y is lost. Remainder is put in @

EXAMPLES

Program A,B+=? Initially A=1 B=2
step: abcdef

STEP	SYMBOL	MEANS	x	y	Rest of stack	
a	A	load A	1	? – – – –		
b	,	push x	1	1 – – – –		
c	B	load B	2	1 – – – –		
d	+	x:=x+y	3	– – – – –		
e–f	=?	display	3	– – – – –		(3 is displayed on screen 00003).

The program displays the result of A+B

Program to evaluate (2*3) + (7-2) and display it.

Program 2,3*, 7,2- + = ? i.e. add result of 2,3* to 7,2- and display.
step: abcd e fghi j kl

STEP	SYMBOL	MEANS	x	y	Rest of stack	
a	2	load 2	2	? – –		
b	,	push x	2	2 – –		
c	3	load 3	3	2 – –		
d	*	x:=x*y	6	– – –		
e	,	push x	6	6 – –		
f	7	load 7	7	6 – –		
g	,	push x	7	7 6 –		
h	2	load 2	2	7 6 –		
i	–	x:=y–x	5	6 – –		
j	+	x:=x+y	11	– – –		
kl	=?	display	11	– – –		00011 appears on screen – the answer

NOTE The operators # and & only affect x and are equivalent to ,1- and ,1+ (although faster and shorter).

Imagine we want to store the result of multiplying N by M in A.

In Basic this is A=M*N
But in M5 this is M,N*=A

Here are some further examples of expressions:

```
BASIC              M5
=====              ==

Z=N*M*A            N,M*,A*=Z    OR   N,M,A,**=Z
Z=(N+M)*A          N,M+,A*=Z
Z=(N+M)*(A-M)      N,M+,A,M-*=Z
Z=N*N              N,*=Z
Z=N*N*N*N          N,*,*=Z   OR N,,,***=Z    (N.B. M5 ONLY NEEDS TO GET N ONCE )
```

1.2 Getting Data In

Data in M5 is Input from the keyboard. The program requests a number from the keyboard when it encounters a LOAD ? i.e. a ? in the program, not following =.

A number is terminated by any non numeric character. Usually the user will type a space after the number and the program will continue on the same line, otherwise he will use a newline after typing the number.

EXAMPLE ? , ? * = ? will prompt for a number, then another and print the product.

1.3 String print

Any string of characters surrounded by quotes ' " ' is printed to the display exactly as written— including newlines etc.

e.g. "Input the number"
or "NEW
LINE"

N.B. A jump will find labels in a string so beware of using (in a string.

A nicer version of the program above is:

"NUMBER" ?, "TIMES BY"?*" IS "=?

A newline is produced by a newline between quotes.

1.4 Loops and jumps

A way of repeating operations is almost essential in a programming language. In M5 this is done by using using jumps and labels.

A label is represented in M5 by (n where n is any symbol which can be entered at the keyboard.
Examples are: (A (! (1 (.

A jump is represented by)kn where n is a symbol which matches a label, and k is a condition code indicating what condition involving x or x and y must be true for the jump to occur.

Valid condition codes are as follows:

CONDITION CODE CHARACTERS:

Character	Jump occurs if:	Comments:
U	—unconditional—	U stands for unconditional
Z	value of x is 0	7 stands for zero
N	value of x is not 0	N stands for non zero
E	x=y (top 2 on stk)	E stands for equal
X	x≠y	X looks like a notequal sign
L	x < = y	L stands for less than or equal
G	x > = y	G stands for greater than
M	—unconditional—	M is monitor . jump to editor.

EXAMPLES of valid jump symbols are:

)UA)NI (X$)G((Z. matching labels above.

when a jump symbol is reached, the condition indicated by K is tested and if it is found to be true, a jump is made to the first occurence of a label with matching identifier symbol.

EXAMPLES:

```
(i)    2000 (A  "HELLO" #   )NA        prints out "HELLO" 2000 times.

(ii)      0 (A =?  " "  &   )NA        prints out numbers from 0 to 65535
                                       separated by spaces. (Thinks 65536=0 ).

(iii)   (A   )UA                       loops until RESET is pressed.

(iv)   0=N (A N=? &=N , 5555 )GA       prints out numbers from 0 to 5555.
```

2.0 WRITING PROGRAMS

M5 is a powerful language when all its features are properly understood, but it can be a little confusing for the beginner. There is fortunately an easy way of generating programs which can be used until familiarity with M5 is achieved. The method is to write the program in a more standard language and then translate into M5. While this method does not exploit the valuable 'current variable' feature of M5, it will yield workable programs which are easier to follow in many ways. The program can then be optimised when it has started to work.

EXAMPLE: A Program to print a table of squares from 1 to 30.

BASIC	M5
10 PRINT "TABLE OF SQUARES"	"TABLE OF SQUARES "
20 N=0	0=N
30 N=N+1	(B N,1+ = N
40 PRINT N, N*N	N=? " " N,N*=? " "
50 IF N = 20 GOTO 30	N , 20)XB
60 END	)M

NOTE: Newlines in output must be included between quotes in M5 programs. The numbers in M5 are not spaced on output, hence the space in the line equivalent to line 40.

The M5 produced will be completely sound and will run at about the same speed as the tiny Basic program.

If the M5 is optimised, keeping N in "x" as much as possible and using the free layout and the & operator, the speed will be considerably faster, perhaps 4-5 times faster than a fast tiny basic.

Optimised:

```
"TABLE OF SQUARES
" 0=N (B N&=N=? " " ,*=? "
"N,20 )XB )M
```

3.0 THE EDITOR

3.0 Introduction

The M5 Editor is entered by typing E when in the command mode.

The edit prompt of E: will appear when the editor is ready to accept input.

The editor will show the point where the last instruction was executed when it is entered by positioning a cursor at this location. The cursor is a shaded in square which is denoted here by a — (underline).

The cursor indicates the current position of the character pointer, and the character pointed at by the cursor appears at the top right of the screen. All manipulation of text is done relative to this cursor because there are no line numbers in M5.

The character indicating end of file in M5 is a null character which appears as a box when it is pointed at.

A hazard in the M5 interpreter is that the pointer can be moved into the actual M5 Interpreter. A Rule must therefore be: DO NOT use any Delete or insert commands unless you can see where the pointer is positioned.

3.1 Commands

To manipulate the text of a program, the user must be able to position the cursor in the required area and then operate on the text. Commands to move the pointer are as follows:

>	Move cursor forward one place.
<	Move cursor backward one place.
R	Rewind—i.e. move cursor to the start of the file.
N	Move the cursor to the start of the next time (stop at end of prog.)

These commands may be repeated and if followed by a newline, will result in a printout of the text with the cursor in its new position.

EXAMPLE: You have typed in a program as follows:

```
(A "HELLO THERE " N=? " IS N
WHAT NUMBER DO YOU WANT" ; . . . . . etc
```

And you want to move the cursor to the spelling error.

Use: RN i.e. move to start, move down a line, move in 5 characters.

Using a space instead of a newline will not print out the text but will carry out the actions and return the edit prompt.

Once we have moved the prompt to where we want to make adjustments we have commands to delete and insert characters.

D Remove (delete) the character pointed at by the cursor.
The cursor now points to the next character along.

Innnn; Insert the string nnnn before the character pointer.
The terminator is a ; Cursor points to same character.

EXAMPLE: Edit ABCDERTYIJKLMNOP to replace RTY by FGH

ABCDEFRTYJKLMNOP

E:R>>>>>>> Move pointer to start then along 7 characters (to Y?)

ABCDEF–TYIJKLMNOP Character R appears at top R.H. side of screen.

E:D Delete current character.

ABCDEF–YIJKLMNOP T appears at top right.

E:DD Delete two more.

ABCDEF–JKLMNOP 1 appears at top right.

E:IGHI; Insert correct characters.

ABCDEFGHI–KLMNOP string now correct– O still current character.

When editing is complete, the command W is used to return to command mode.

4.0 ERROR MESSAGES

When a large program is written concisely in M5, errors may be difficult to detect so good errr diagnostics at runtime were included.

If a syntax error occurs, one of the following messages will appear:

SYM	FRR	x	The symbol x is not allowed in M5 (except in a string).
10	ERR	x	The symbol x is not a valid identifier, and an attempt was made to copy a value into it. (e.g. =x occurred.)
JID	ERR	x	The label x was not found when a jump occurred to it.
JC	ERR	x	The symbol x occurred in a jump condition position and is not a valid code (one of U A N Z X G E M).
	ERR	x	The symbol x caused an error to occur. (Not one of above.)

In addition to giving the error type, the editing cursor is set up to point at the faulty symbol, so when the editor is entered from the monitor to correct the error, the cursor is in the correct position for amendments.

(N.B. in M6, JID errors are detected before the program starts to execute.)

5.0 SAMPLE PROGRAMS IN M5

```
Number summing program    (A"INPUT A NUMBER"?," THANKS
-----------------------     NOW INPUT 2 MORE NUMBERS"?,"AND"?"GOOD!
                            THEIR SUM IS "+=? "
                            ")NA "THEIR SUM WAS ZERO - TYPE 0 FOR MORE FUN OR
                            1 TO END "?)ZA "GOODBYE!" )M
Factorial of a number:    1=N ? )78 (A  =M , N* =N M# )NA (B N=?
----------------------
M5 24 hour clock:
-----------------         )US (D  N#=N )ND
(N.b. remove all          H=?" HRS "M=?" MINS "S=?" SECS
 spaces for good          " L=N S&=S , T )XD
 timekeeping )              0=S M&=M , T )XD
                            0=M H&=H ,24 )XD
                            0=H )UD
(Start put at end)        (S 1750=L  60=T
                          "SET HRS"?=H"SET MINS"?=M"SECS"?=S"
                          " )UD
```

Note that the main timing loop is at the beginning for higher speed.
1750 is the timekeeping constant. make smaller to speed up clock.

```
Square root of a number:  256=M  ?=N  (1  N,M/ , M )LS +,2/=M )U1
------------------------
                            (S " "M=?" "
Method used is very fast but a little hard to follow.

Prime numbers:              1=T
--------------            (N T&&=T
                            1=G
                          (A G&&=G
                            T,G/,G )GP
                            @ )NA )UN
                          (P T=? "
                            " )UN

This can be compacted to only one line of course, ( a bit baffling though ):

1=T(NT&&=T1=G(AG&&=GT,G/,G)GP@)NA)UN(PT=?" ")UN
```

See disassembly listing in next file.

hexadecimal object code listing 23 MAR 79 14:14

Addr	Bytes								Bytes							
0C50	D6	3F	CD	01	0E	5E	23	56	18	3B	E1	ED	52	EB	18	35
0C60	C3	3E	0E	EF	3F	00	21	00	00	CD	25	0E	CD	14	0E	33
0C70	F8	EB	18	21	62	68	FD	21	0A	0E	AF	FD	46	01	FD	4E
0C80	00	ED	42	38	03	3C	18	F9	09	C6	30	CD	38	01	FD	23
0C90	FD	23	0D	20	E5	DD	23	DD	7E	00	FE	20	28	F7	FE	1F
0CA0	28	F3	FE	3F	28	8D	30	A8	FE	2C	28	30	FE	3D	28	33
0CB0	FE	29	CA	74	0D	FE	23	28	46	FE	26	28	3F	FE	28	28
0CC0	36	FE	20	28	95	FE	2A	28	39	FE	2F	28	56	FE	28	28
0CD0	0E	FE	22	28	6C	B7	CA	3E	0E	C3	54	0D	D5	18	86	DD
0CE0	23	18	82	DD	23	DD	7E	00	D6	3F	28	88	DA	C7	0D	CD
0CF0	01	0E	73	23	72	18	9E	E1	19	EB	18	99	13	18	96	18
0D00	18	93	C1	3E	10	21	C0	00	CB	7A	28	04	09	30	01	13
0D10	3D	28	09	EB	29	EB	29	30	EF	13	18	EC	EB	22	C0	08
0D20	C3	95	0C	42	48	21	C0	00	D1	3E	10	29	EB	29	EB	30
0D30	02	23	B7	ED	42	13	F2	3C	0D	09	CB	83	3D	20	EC	18
0D40	DC	DD	23	DD	7E	00	FE	22	CA	95	0C	B7	CA	3E	0E	CD
0D50	38	01	18	ED	D6	30	FE	0A	30	13	21	00	00	DD	7E	00
0D60	DD	23	CD	14	0E	38	F6	EB	DD	2B	C3	97	0C	EF	53	59
0D70	4D	00	18	57	DD	7E	01	FE	4E	28	31	FE	55	28	5B	FE
0D80	5A	28	23	08	E1	E5	B7	ED	52	08	FE	45	28	24	FE	58
0D90	28	23	FE	4C	28	22	FE	47	28	23	FE	4D	CA	3E	0E	EF
0DA0	4A	00	DD	23	18	25	7A	B3	28	30	18	14	7A	B3	20	2A
0DB0	18	0E	08	18	F3	08	18	F6	08	30	1F	18	03	08	38	1A
0DC0	DD	23	DD	23	C3	95	0C	EF	49	44	00	EF	20	45	52	52
0DD0	20	00	DD	7E	00	CD	3B	01	18	64	DD	4E	02	31	FA	0F
0DE0	21	FE	0E	06	28	7E	23	B8	28	0D	B7	C2	E5	0D	DD	23
0DF0	DD	23	EF	4A	00	18	DD	7E	B9	20	EA	E5	DD	E1	C3	95
0E00	0C	07	4F	06	00	21	BE	08	09	C9	10	27	E8	03	64	00
0E10	0A	00	01	00	D6	30	FE	0A	D0	29	54	5D	29	29	19	5F
0E20	16	00	19	37	C9	CD	3E	00	C3	3B	01	EF	1F	00	21	FD
0E30	0E	23	7E	B7	C8	CD	3B	01	18	F7	AF	77	23	77	EF	1F
0E40	4D	35	3A	00	CD	25	0E	FE	4C	CC	2B	0E	FE	49	CA	D3
0E50	0E	FE	52	20	09	EF	1F	00	DD	21	FD	0E	18	A0	FE	45
0E60	20	DC	DD	E5	E1	4E	36	7F	E5	79	32	F6	0B	CD	25	0E
0E70	E1	71	EF	1F	45	3A	00	CD	25	0E	FE	44	28	3A	FE	1F
0E80	28	E3	FE	3E	20	01	23	FE	3C	20	01	28	FE	52	28	22
0E90	FE	4E	28	34	FE	57	28	A6	FE	49	20	08	CD	25	0E	FE
0EA0	38	28	D4	E5	4E	77	23	79	B7	20	F9	77	23	77	E1	23
0E80	18	EA	21	FF	0E	28	18	BF	E5	DD	E1	DD	7E	01	DD	77
0EC0	00	B7	28	B3	DD	23	18	F3	7E	B7	28	A8	23	FE	1F	20
0ED0	F7	18	A4	EF	6E	70	75	74	1F	00	21	FD	0E	23	CD	25
0EE0	0E	FE	38	CA	3A	0E	77	FE	1D	20	F2	2B	18	F0	D4	

Execute from 0C60. Program starts at 0EFF.