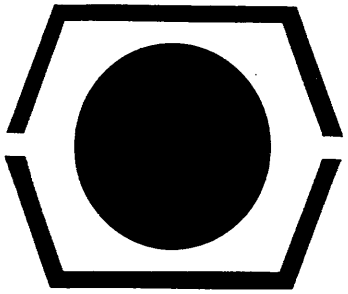

DECUS
PROGRAM LIBRARY

DECUS NO.	8-91
TITLE	Micro-8: An On-Line Assembler
AUTHOR	K. F. Kinsy, State University of New York at Geneseo, and M. E. Nordberg, Jr., Cornell University
COMPANY	
DATE	September 27, 1967
FORMAT	

Micro - 8: An On-Line Assembler

K.F. Kinsey

Department of Physics
State University of New York at Geneseo
Geneseo, N. Y.

and

M.E. Nordberg, Jr.

Laboratory of Nuclear Studies
Cornell University
Ithaca, N. Y. 14850

Abstract

Micro - 8 is a short assembler program for the PDP-8 computer that translates typed mnemonic instructions into the appropriate binary code and places them in specified memory locations immediately ready to function. It processes the typed instructions by a table-lookup procedure. It is especially useful for programs of less than one page which are to be run immediately. Only octal (not symbolic) addresses may be specified, but the user has control of the zero page and indirect addressing bits. An octal typeout routine permits examination of any memory location.

Often the need arises in on-line use of a computer for short programs or short sub-programs to used immediately. For example, one may wish to write a simple interrupt routine to check out a piece of hardware before it is used with a full-fledged data analysis program. In cases where this can be done in a half a dozen steps or less, the switch registers may be used to enter binary codes directly and in cases where a few pages are required, one might as well use Macro - 8. For programs intermediate in length, between a few commands and a few pages of commands, the need arises for a quick way to use mnemonic instructions; Micro - 8 satisfies this need. It eliminates the necessity of memorizing or looking up binary codes which especially helps those who use the computer only occasionally. It is somewhat similar to DDT-8 but shorter and does not use page 0 or the high pages so that it is useful in conjunction with interrupt routines in page 0 and with the floating-point packages.

The format of a Micro - 8 statement is the same as that of Macro - 8, except that symbolic addressing is not permitted. The assembler contains an address counter which controls the address into which a valid command is inserted. The instruction codes are translated from symbolic to binary by searching through two symbol tables containing the operate and the memory reference instructions.

The address counter specifies the address into which the next binary instruction or constant is to be inserted or read out. It can be affected in three ways. 1) By typing a "*" as the first character of a line, followed by four octal digits,

the address is set to the value of the four digits:

*1234

the program spaces the typewriter and awaits the next insertion. This control is used to initialize the address counter.

2) By typing a space as the first character of a line, the program types out the current value of its address counter in octal followed by a space and awaits the next insertion:

␣1234

This control is used to sequentially enter instructions or constants, since the address counter is automatically incremented after entry.

3) By typing a "/" as the first character of a line, the program types out the value of the address counter plus one followed by a space and awaits the next insertion:

/1235

This control is used to sequentially print out the contents of the specified address since the octal print does not automatically increment the address counter.

After the address counter has been typed, there are 3 basic modes of operation of Micro - 8. It is ready to receive a symbolic instruction, a 4-digit octal entry, or a two-character entry followed by a space. Instruction codes may be generated by typing the appropriate symbolic instruction code. Micro - 8 considers all instruction codes to be 4 characters long, left justified. Thus, the code for halt is HLT␣. No code added by a user may have a space as its third character.

Instructions are grouped into operate (micro-programmable) instructions and memory reference instructions.

Micro-programmable instructions. On receiving the 3 character code and space, the computer will generate a second space and expect a further instruction. Up to three instructions may be micro-programmed on one line. The line will be terminated on receipt of the third instruction or of a space instead of an instruction code:

*0200 CLA␣

␣0201 CMA␣ IAC␣␣

␣0202 CLL␣ CLA␣ RTL␣

The corresponding instructions are combined by the inclusive OR operation. Table I lists the location, octal content and symbolic content of the operate instructions. Note that Micro - 8 does not distinguish logically inconsistent micro-instructions. The address counter is automatically incremented after completion of the entry. The program then outputs a carriage return and line feed.

Memory reference instructions. These instructions are looked up in a separate table which includes the usual 6 memory reference instructions and the 6 floating point memory reference instructions. See Table II. The user may specify an indirect bit, (by typing I) and/or a zero page reference (by typing Z), and must give a 3 character octal address. The first of this address is masked to provide the 7-bit binary address: Thus, 177, 377, 577, and 777 will all assemble to the same 7-bit address: 177. The following formats are all acceptable:

```
*0200 TAD␣I Z 101        1501
␣0201 TAD␣␣ Z 033        1033
␣0202 TAD␣I ␣ 422        1622
␣0203 TAD␣ I 322         1722
␣0204 TAD␣Z 777          1177
␣0205 TAD␣ 333           1333
```

In place of an instruction code, an octal 4-digit number may be typed. This number is entered into the address specified by the address counter and the address is incremented. The computer will output a carriage return and line feed. This mode of operation is useful in order to enter a constant, an index, an indirect address or an instruction such as seldom used I/O instructions which have not been entered in the symbol table.

The third type of entry is two characters followed by a space. The contents of the address specified by the address counter are set to the two characters in 6-bit form. For example:

```
*0345 AB
```

then C(0345) = 0102.

Any characters are acceptable except that the first may not be an octal number, a space or an "=". The address counter is incremented and the computer will output a carriage return and line feed. This mode of entry may be used to expand the symbol table.

The symbol table of operate instructions starts in 4041 and extends to 4200. Location 3570 contains the number LAST + 1=4201 To add to the symbol table the instruction; ADC1 = 6402, one would type the following:

```
*4201 AD␣       0104      (First part of code)
␣4202 C1␣       0361      (Second part of code)
␣4203 6402      6402      (Instruction in octal)
*3570 4204
```

In case of mistakes in instruction codes such as HTL␣ when HLT␣ was intended or in case of the invalid numbers 8 or 9, the program spaces twice, types two question marks, a carriage return and line feed. The address counter is not incremented and the program awaits the first character on the line which is usually a space to produce a typeout of the current value of the address counter. For example:

```
*0200 HTL␣ ??
␣0200 HLT␣
␣0201 7409  ??
```

The contents of the address specified by the address counter may be examined by typing "=" instead of an instruction or other entry. The contents are typed out in octal. Since after examining the contents one may wish to change them, the value of the address counter is not incremented. For example:

```
*3570 = 4201
␣3570 4204
```

would set the number LAST + 1 = 4204. On the other hand one may wish to examine successive addresses and the process of typing the "=" followed by four digits would be a nuisance. Therefore, if one types a "/" as the first character of the line, the address counter is incremented before it is typed out. For example:

```
*0200 = 7770      (index variable)
/0201 = 2200      (ISZ 200)
/0202 = 5201      (JMP 201)
/0203 = 7402      (HLT)
```

The user should be warned that Micro - 8 is quite capable of modifying and therefore destroying itself.

Table I

Basic Operate Instruction Symbol Table

Location	Symbolic Contents	Octal Contents	Location	Symbolic Contents	Octal Contents	Location	Symbolic Contents	Octal Contents
4041	NO	1617	4102	CL	0314	4143	IO	1117
4042	P␣	2040	4103	A␣	0140	4144	N␣	1640
4043		7000	4104		7200	4145		6001
4044	IA	1101	4105	ST	2324	4146	IO	1117
4045	C␣	0340	4106	A␣	0140	4147	F␣	0640
4046		7001	4107		7240	4150		6002
4047	RA	2201	4110	HL	1014	4151	KS	1323
4050	L␣	1440	4111	T␣	2440	4152	F␣	0640
4051		7004	4112		7402	4153		6031
4052	RT	2224	4113	OS	1723	4154	KC	1303
4053	L␣	1440	4114	R␣	2240	4155	C␣	0340
4054		7006	4115		7404	4156		6032
4055	RA	2201	4116	SK	2313	4157	KR	1322
4056	R␣	2240	4117	P␣	2040	4160	S␣	2340
4057		7010	4120		7410	4161		6034
4060	RT	2224	4121	SN	2316	4162	KR	1322
4061	R␣	2240	4122	L␣	1440	4163	B␣	0240
4062		7012	4123		7420	4164		6036
4063	CM	0315	4124	SZ	2332	4165	TS	2423
4064	L␣	1440	4125	L␣	1440	4166	F␣	0640
4065		7020	4126		7430	4167		6041
4066	CM	0315	4127	SZ	2332	4170	TC	2403
4067	A␣	0140	4130	A␣	0140	4171	F␣	0640
4070		7040	4131		7440	4172		6042
4071	CI	0311	4132	SN	2316	4173	TP	2420
4072	A␣	0140	4133	A␣	0140	4174	C␣	0340
4073		7041	4134		7450	4175		6044
4074	CL	0314	4135	SM	2315	4176	TL	2414
4075	L␣	1440	4136	A␣	0140	4177	S␣	2340
4076		7100	4137		7500	4200		6046
4077	ST	2324	4140	SP	2320			
4100	L␣	1440	4141	A␣	0140			
4101		7120	4142		7510			

Table II

Memory Reference Instruction Symbol Table

Location	Symbolic Contents	Octal Contents	Location	Symbolic Contents	Octal Contents
3731	AN	0116	3753	FA	0601
3732	D␣	0440	3754	DD	0404
3733		0000	3755		1000
3734	TA	2401	3756	FS	0623
3735	D␣	0440	3757	UB	2502
3736		1000	3760		2000
3737	IS	1123	3761	FM	0615
3740	Z␣	3240	3762	PY	2031
3741		2000	3763		3000
3742	DC	0403	3764	FD	0604
3743	A␣	0140	3765	IV	1126
3744		3000	3766		4000
3745	JM	1215	3767	FG	0607
3746	S␣	2340	3770	ET	0524
3747		4000	3771		5000
3750	JM	1215	3772	FP	0620
3751	P␣	2040	3773	UT	2524
3752		5000	3774		6000