DECUS PROGRAM LIBRARY

DECUS NO.	8-569
TITLE	FLIT ASSEMBLER
AUTHOR	Gary R. Smith
COMPANY	Submitted by: George E. Ott University of Wisconsin Madison, Wisconsin
DATE	July 12, 1972
SOURCE LANGUAGE	MACRO

The FLIT Assembler

TABLE OF CONTENTS

Abstract

The FLIT Assembler produces a binary object tape on a high-speed paper tape punch and a listing on a 33 ASR teletype from PDP-8 assembly language source tape read on a high-speed reader.

FLIT has the following major advantages over other assemblers:

1. Literals and off-page linkages are automatically generated.
2. The source tape is read rapidly and reliably, reducing assembly time.
3. Line numbers appear on the listing, simplifying use of the Symbolic Editor.
4. Tabulations become 8-space fields in the listing just as with the Symbolic Editor.
5. The symbol table has room for at least 348_{10} user symbols.

FLIT does not recognize macros, floating-point or double-precision numbers, or Boolean operators. A few other minor source language differences exist between FLIT and other assemblers.

I. Introduction

The FLIT (for Fast Line-numbering Interrupting Tabulating) Assembler reads from a high-speed paper tape reader a source tape containing PDP-8 assembly language and punches on a high-speed paper tape punch a binary object tape and types a listing and symbol table on a 33 ASR teletype. The major advantages of the FLIT Assembler over PAL III and MACRO-8 are that literals and links are generated and the paper tape is read rapidly and reliably.

During pass 1 the assembler reads the source tape and generates a symbol table. During pass 2 the source tape is read again and a binary tape punched and/or a listing typed.

The FLIT Assembler performs roughly the same task as the other PDP-8 assemblers. This document describes the differences between the FLIT Assembler and the other assemblers. Information about the other PDP-8 assemblers is contained in the Digital Equipment Corporation book Programming Languages.

II. Source Language Differences

A. Character Set

Characters read by the FLIT Assembler are normally checked for validity. If an invalid characte. is read FLIT types an IC error message, then halts, displaying the bad character in the accumulator. The operator must set the switch register to a valid character and depress CONTINUE.

Characters which are not checked for validity are those in a comment (between a slash and a carriage return or semi-colon) and the first character following a double quote.

The following characters are always ignored:

- leader/trailer (code 0)
- line feed (code 212)
- rubout (code 377)
- any character except the end characters beyond the 72nd in a statement

The following characters always cause a statement to end:

- carriage return (code 215, see note)
- semi-colon (code 273)

The following characters are valid:

A-Z, 0-9, space, ", $, (,), [,], *, +, -, period, comma, /, =, tab (code 211, see note), form (code 214, see note)

All other characters are invalid.

Note: Codes 215, 211, and 214 are changed to 336, 334, and 337, respectively, after their validity is checked. This prevents their confusion with codes 315, 311, and 314 in the 6-bit symbol table format.

B. Statements
Statements representing machine instructions are written just as for PAL III and MACRO-8, that is:

```
     label,  instruction operand          /comment
```

Any one or more of these statement elements may constitute a statement. Spaces and/or tabs must separate the instruction and operand.

C. Literals
A current-page literal (indicated by a left parenthesis) or a zero-page literal (indicated by a left bracket) may be used as the operand in a statement. A literal may take the form of any expression consisting of constants (numerical and ASCII code generated by a ") and variables combined by + and - signs. A literal may also take the form of an instruction which may include a literal. Literals may be nested in this way up to a level of four.

The FLIT Assembler forms the value of a literal and then searches for that value in the zero-page literal buffer. If the value is found its address is used in place of the literal even if a current-page literal was specified. Otherwise the storing of literals and the assignment of addresses proceeds as with MACRO-8 except that FLIT will never dump out a literal buffer before the logical end of a page. The zero-page literal and link buffer can store up to 64_{10} words, the current-page buffer can store up to 84_{10} words. FLIT does not remember the address of the last literal stored on a page after the literal buffer of that page has been output. Therefore setting the location counter to a new page always causes FLIT to assign the first literal to page address 177. Thus any literal previously stored on that page would be overlaid.

D. Off-page Linkages
If the address of an operand is not on page zero or the current page then an off-page linkage is required and FLIT sets the indirect bit. The off-page address is then treated just like the value of a current-page literal. All the statements about literals in the previous paragraph also apply to links since common buffers are used to store literals and links.

E. Direct Assignment Statements
Any expression, but <u>not</u> an instruction with an indirect or non-zero page address, may be used to the right of an equals sign provided all symbols used have been defined. The following are valid statements:

```
A=3
B=A+1
MINUSC=-"C
EXIT=JMP+400
```

The following statements are invalid:

```
EXIT=JMP I 0
WAIT=JMP .    (would generate WAIT=5000+current location counter)
```

F. The FIELD Statement

Upon encountering a FIELD statement FLIT outputs the current-page literal and link buffer. If a valid non-negative expression less than 10_8 follows FIELD and the expression (the new field number) is different from the old field number, then FLIT outputs the zero-page literal and link buffer, the field setting code, and an origin of 200. In all other cases FLIT outputs only the origin of 200.

G. The PAGE Statement

A PAGE statement with a valid non-negative expression less than 40_8 sets the location counter to 200_8 times the expression. A PAGE statement without an expression sets the location counter to the beginning of the next page following the one the location counter was pointing to. In particular, the statements

```
*377
A,      0
        PAGE
B,
```

will give B the value 600 because the location counter was incremented to 400 at the end of processing the statement labeled A. No PAGE statement is necessary if code is to continue onto a new page and no literals or links were used on the old page.

H. The TEXT Statement

The TEXT statement causes characters between the delimiters to be stripped to 6 bits and stored two to a word with a zero stop code at the end, just as with MACRO-8. However, any number of spaces and/or tabs may precede the first occurrence of the delimiter character which may be any of the valid FLIT characters (not ignored or end characters) given under II.A. Character Set. But if invalid characters are to appear between delimiters of a TEXT statement, the delimiters must be slashes so that FLIT does not try to check the characters for validity. Appearance of an end character (carriage return or semi-colon) before the second occurrence of the delimiter is an error and the end character will be treated as if it were the delimiter.

I. The BLOCK Statement

The BLOCK statement is used by FLIT to reserve several words of memory by placing zeros in them. This statement is similar to the BLOCK statement in SABR. An expression following BLOCK is treated as a positive number indicating how many zeros are to be output. If no expression or one with a zero value follows BLOCK, the statement does not output any zeros but defines a label, if any, to be equal to the current location counter. Examples of the use of the BLOCK statement follow:

```
          BLOCK      /HAS NO EFFECT ON THE ASSEMBLY
     A,   BLOCK      /DEFINES A=
     B,   BLOCK 0    /B=
     C,              /C=D
     D,   BLOCK 2    /ASSEMBLES 2 ZEROS WITH D
                     /  THE LABEL OF THE FIRST ONE
```

Note: If many locations need to be reserved an asterisk should be used to change the location counter. If these locations need to be initialized to zero the program should contain a loop to do it.

J. Other Source Language Differences
1. FLIT does not recognize macros.
2. FLIT does not generate double-precision or floating-point numbers.
3. FLIT does not have EXPUNGE or FIXTAB statements.
4. FLIT does not recognize the pseudo-op Z.
5. FLIT does not recognize ! or & as operators.

III. Other Processing Differences

In addition to the differences in processing source language statements mentioned above, the FLIT Assembler has the following processing differences:

A. FLIT uses interrupts to switch efficiently between the three activities of reading, assembling, and punching. During a pass 2 when a binary tape but no listing is being output, FLIT is usually assembling a statement while reading the next one and while punching data from the previous one. This use of interrupts allows much more efficient and reliable use of the high-speed reader.

B. Tabulations in the source statements are expanded to 8-character fields during listing of the statements (same format as the Symbolic Editor).

C. Form characters (code 214) are used just as in the Symbolic Editor to delimit blocks of source statements. The FLIT Assembler uses these characters in generating line numbers which are typed between the assembled code and the source statement on the listing. These line numbers will be most helpful in using the Symbolic Editor if editing operations (e.g., inserting, changing, deleting) are done from the end of a block of statements towards the beginning.

D. A binary tape and a listing can be produced simultaneously during pass 2. See V. Operating Procedures.

E. The format of the symbol table is different than that for PAL III or MACRO-8. See IV. Symbol Table.

F. Many assembly errors cause a HLT (7402) to be assembled instead of an erroneous instruction. Such an instruction will not then damage the program if the operator forgets to patch in the correction.

G. A PAUSE statement in the middle of a source tape will usually not cause the loss of any information on the tape, certainly not if the blank tape associated with a form character follows the PAUSE. Just depress CONTINUE to go on assembling after the PAUSE.

H. Different error messages are used by FLIT. See VI. Error Diagnostics.

IV. Symbol Table

A. Format of the Symbol Table

Each symbol or pseudo-op in the table requires $\lfloor \frac{n}{2} \rfloor$+2 words, where $\lfloor \frac{n}{2} \rfloor$ means the greatest integer in $\frac{n}{2}$ and n is the number of characters in the symbol or pseudo-op, but 6-character symbols require 4 words. The characters of a symbol are stripped to 6 bits and words of 2 characters are formed by shifting the first character 6 bits to the left and negating the whole word. The format of one- through six-character symbols is shown below.

Word 1	$-(C1 \times 100_8)$
Word 2	value of symbol

Word 1	$-(C1 \times 100_8 + C2)$
Word 2	0
Word 3	value of symbol

Word 1	$-(C1 \times 100_8 + C2)$
Word 2	$-(C3 \times 100_8)$
Word 3	value of symbol

Word 1	$-(C1 \times 100_8 + C2)$
Word 2	$-(C3 \times 100_8 + C4)$
Word 3	0
Word 4	value of symbol

Word 1	$-(C1 \times 100_8 + C2)$
Word 2	$-(C3 \times 100_8 + C4)$
Word 3	$-(C5 \times 100_8)$
Word 4	value of symbol

Word 1	$-(C1 \times 100_8 + C2)$
Word 2	$-(C3 \times 100_8 + C2)$
Word 3	$-(C5 \times 100_8 + C6)$
Word 4	value of symbol

The current end of the symbol table is indicated by a zero in the word following the last symbol. No bits are needed for system flags during assembly, but bit 0 of word 1 is complemented during symbol table typeout to indicate that that symbol has not yet been typed. (That bit is always one for user symbols since they must begin with a letter.)

Pseudo-ops are indicated by their position at the beginning of the table. Instead of a value pseudo-ops have in the last word the address of the routine which processes the pseudo-op.

B. Contents of the Symbol Table

The pseudo-ops are stored in the following order in the symbol table (read down columns and from left to right):

tab	+	"	FIELD
space	-	BLOCK	PAUSE
carriage return	[	*	$
/	;	.	)
,	form	=	]
I	PAGE	DECIMAL	
(	TEXT	OCTAL	

The permanent symbols are stored in the following order:

TAD	SPA	HLT	OSR	TSF	PPC
DCA	SMA	RAL	GLK	TCF	PLS
CLA	CLL	RAR	NOP	TPC	CDF
CMA	CML	RTL	ION	TLS	CIF
JMP	SZL	RTR	IOF	RSF	RDF
ISZ	SNL	IAC	KSF	RRB	RIF
JMS	STL	AND	KCC	RFC	RIB
SZA	STA	SKP	KRS	PSF	RMF
SNA	CIA	LAS	KRB	PCF	

V. Operating Procedures

To assemble a program using the FLIT Assembler follow the instructions below:

1. Load FLIT (into field 0 only) with the Binary Loader.
2. Set the switch register to 0200.
3. Depress LOAD ADDRESS.
4. Put the source tape in the high-speed reader.
5. Set switches 0 and 1 to select the pass and output desired:

switch 0	switch 1	action
down	down	pass 1
up	down	pass 2 with binary tape punched on high-speed punch (turn this device on)
down	up	pass 2 with listing typed on 33 ASR teletype
up	up	pass 2 with both binary tape and listing

6. Depress START.
7. When FLIT halts (after reading a PAUSE statement), place the next tape in the reader and depress CONTINUE.
8. When FLIT halts at the end of a pass, proceed to the next pass by beginning at step 4 above (or equivalently, at step 2).

The alphabetized symbol table is typed at the end of the listing.

A binary tape and a listing can be made in any order or simultaneously. Pass 2 may be entered as many times as necessary after pass 1 has been completed once.

Another program may be assembled after one has been completed by putting the other source tape in the reader, setting switches 0 and 1 to 0(down) and depressing START.

Any pass may be restarted at any time by returning to step 2 above.

VI. Error Diagnostics

The format of error messages is:

EC AT SYMBOL + DSPL

where EC is the error code (see below), SYMBOL is the last label before the error, and DSPL is the octal displacement from that label to the value of the location counter when the error message was typed.

Processing continues after all error messages except after an IC (illegal character) error when a valid character can be entered and after an ST (symbol table) error when a halt occurs and the source tape must be modified and the FLIT Assembler restarted.

Error Code	Meaning
IC	Illegal character: set the switch register to a valid character and depress CONTINUE.
ST	Symbol table overflow: the program must be modified or divided so there are fewer or shorter symbols.
RD	Reader or confusing statement or redefinition error. In pass 2 a symbol is redefined to have a value different from its preceding definition. The new definition replaces the old one. This error could be caused by improper loading or reading of a paper tape or by a confusing statement which caused the FLIT Assembler to adjust the location counter differently in passes 1 and 2. This message could also be caused by using an equals sign to set a symbol to two or more values during one assembly; this message should then be viewed as a warning.
ZE	Zero page exceeded. An attempt was made to overlay an instruction on page 0 with a literal or vice versa.
PE	Current, non-zero page exceeded. Same as ZE except for a non-zero page.
II	Illegal indirect. An I was given in an instruction which had an off-page address.
DT	Duplicate tag. In pass 1 a tag was already in the symbol table; the symbol retains its old value. In pass 2 an RD error would be generated and the symbol would be set to the new value.

SN	Syntax error. A statement had an improper format: missing or extra operands or spaces or tabs or in a TEXT statement an end character preceded the second occurrence of the delimiter.
US	Undefined symbol. In pass 2 a symbol was encountered which was not defined during pass 1.
IO	Illegal operand. The operand n of a PAGE statement was not in the range $0 \leq n < 40_8$, or of a FIELD statement not in the range $0 \leq n < 10_8$.
IN	Invalid number. An 8 or 9 was given in an octal number or a number would have required more than 12 bits.