

— CONTENTS —

INTRODUCTION	1
ASSEMBLY LANGUAGE RULES	3
Characters	4
Line	5
Label Symbols	5
Constants	6
ASSEMBLY LISTING AND ASSEMBLER MESSAGES	7
Definition Condition Messages	8
Error Messages	8
ASSEMBLER DIRECTIVES	10
ENT (entry)	10
EQU (equate)	11
ORG (origin)	12
DEFB n (define byte)	13
DEFB 'S', DEFB "S" (define byte)	13
DEFW nn' (define word)	14
DEFM 'S', DEFM "S" (define message)	14
DEFS nn' (define storage)	15
SKP n (skip n lines)	16
SKP H (skip home)	16
END (end)	16
MESSAGE TABLE	17

1	INTRODUCTION
3	ASSEMBLY LANGUAGE RULES
4	Characters
5	Line
5	Label Symbols
6	Constants
7	ASSEMBLY LISTING AND ASSEMBLER MESSAGES
8	Definition Condition Messages
8	Error Messages
10	ASSEMBLER DIRECTIVES
10	EVT (entry)
11	EQ (equates)
12	ORG (origin)
13	DEFB n (define byte)
13	DEFB 'S', DEFB "S" (define byte)
14	DEFW n (define word)
14	DEFW 'S', DEFW "S" (define message)
15	DEFS n (define storage)
16	SKP n (skip n lines)
16	SKP H (skip pages)
16	END (end)
17	MESSAGE TABLE

INTRODUCTION

The assembler translates a source file written in assembly language to generate a relocatable binary file; the source file is one which has been generated and edited by the text editor, and the relocatable binary file is an intermediate file between the source file and object file. It is possible to link several relocatable files by the linker.

The assembly source file is coded in assembly language. It consists of labels, mnemonic operations codes, assembler directives, comments and an end directive; these are arranged according to the rules of the assembler. The source file edited by the editor is written in ASCII code. The assembler translates the source file to generate a relocatable file and outputs messages which indicate definition conditions and syntax errors. These messages are included in the assembly listing which is displayed on the CRT or printed on the printer.

The following FDOS commands activate the assembler.

- **ASM SAMPLE**

Activates the assembler. The assembler translates source file SAMPLE.ASC and generates relocatable file SAMPLE.RB.

- **ASM SAMPLE, \$LPT/L, \$CRT/E**

Activates the assembler. The assembler translates source file SAMPLE.ASC, generates relocatable file SAMPLE.RB, prints the assembly listing on the printer and display only erroneous lines and external reference lines of the CRT screen.

- **ASM/N SAMPLE, \$SOA/L**

Activates the assembler. The assembler translates source file SAMPLE.ASC and outputs the assembly listing to serial output port A (\$SOA), but does not generate a relocatable file since global switch/N is specified.

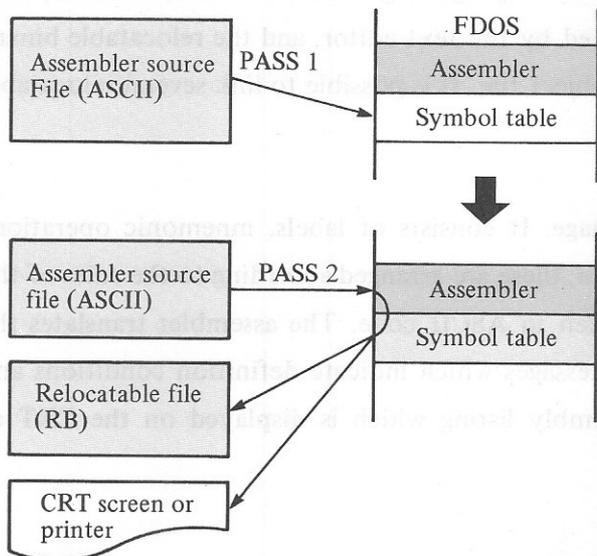
- **ASM SAMPLE, \$FD3; SAMLIST/L**

Activates the assembler. The assembler translates source file SAMPLE.ASC, generates relocatable file SAMPLE.RB and outputs the assembly listing in the same form as that printed on the printer to SAMLIST.ASC on FD3 in ASCII code.

- **ASM SAMPLE, \$LPT/L, \$4000**

Activates the assembler. The assembler translates source file SAMPLE.ASC, generates relocatable file SAMPLE.RB and prints the assembly listing on the printer with a bias of \$4000 added to the relocatable address. Relocatable file is not affected by the bias of \$4000.

The assembler basically uses a 2-pass system. A pass is the process in which the assembler reads a source file from its beginning to end. The following shows operation of the assembler with the 2-pass system.



During pass 1, the assembler stores label symbols according to the assembler rules in the symbolic label table. Label symbols help the operator to read and understand the program easily.

During pass 2, the assembler generates a relocatable file with reference to the symbol table generated during pass 1, then outputs the assembly listing (on the CRT or printer).

The relocatable file and the assembly listing do not occupy space in RAM, which is only used by the symbol table. Therefore, the size of the source file to be assembled is not limited by the amount of RAM.

The following program list will help you understand the function of the assembler. This program is only for reference and has no meaning.

```

** Z80 ASSEMBLER SA-7201 <A> PAGE 01      ??/??/?
01 0000      ;
02 0000      ; SAMPLE LIST
03 0000      ;
04 2000      ORG      2000H
05 2000 3E33  LD      A, '3'
06 2002 FE43  CP      43H
07 2004 FE43  CP      'C'
08 2006 FE15  CP      'H'
09 2008 22    DEFB   ""
10 2009 27    DEFB   ""
11 200A 43    DEFB   'C'
12 200B 12    DEFB   'H'
13 200C 16151211 DEFMB  'CH↓↑→←'
14 2010 1314
15 2012 7E    LD      A, (HL)
16 2013 7E    LD      A, M      ; M may be used in place of (HL).
17 2014      ;
18 2014      ;
19 2014 P     XYZ:   EQU   10
20 2014 C32120 JP      ABC+XYZ      ; Relocatable address ± EQU defined aymbol value.
21 2017 C30A00 ABC:   JP      XYZ
22 201A C31420 JP      ABC-3
23 201D C30A00 JP      10           ; Absolute address 10
24 2020 C32A20 JP      +10         ; Relative address 2AH (20H+10)
25 2023 2100D0 LD      HL, D000    ; Handled as a hexadecimal number.
26 2026 213930 LD      HL, 12345
27 2029 212120 LD      HL, ABC+XYZ
28 202C 3E0D  LD      A, XYZ+3    ; EQU defined label value ± numerica data
29 202E 3EFF  LD      A, -1       ; Negative value is converted into one's complement.
30 2030 21FFFF LD      HL, -1
31 2033 21F0FF LD      HL, -10H
32 2036 C33520 JP      -1
33 2039
34 2039 CD4A20 CALL   ZZZ
35 203C CD5420 CALL   ZZZ+10
36 203F CD4B20 CALL   ZZZ+XXX
37 2042 21FFFF LD      HL, -XXX
38 2045 21FEFF LD      HL, -XXX-XXX
39 2048 4920  DEFWB  ZZZ-XXX
40 204A 00    ZZZ:   NOP
41 204B P     XXX:   EQU   1
42 204B      END

** Z80 ASSEMBLER SA-7201 <A> PAGE 02      ??/??/?
ABC 2017 XXX 0001 XYZ 000A ZZZ 204A      ; Indicates the contents of the symbol table.
  
```

ASSEMBLY LANGUAGE RULES

The source program must be coded according to assembly language rules. This paragraph describes the structure of the source program and the assembly language rules.

The assembly source program consists of the following.

Z80 instruction mnemonic codes

Label symbols

Comments

Assembler directives
(Pseudo instructions)

Definition directives

Entry directives

Skip directives

End directive

Comments may be used as needed by the programmer; they have no effect on execution of the program and are not included in the relocatable file.

All assembly source programs must be ended with the assembler directive END.

Z80 instruction mnemonic codes from the body of the assembly source program. These are explained in a separate volume.

A mnemonic code consists of an op-code of up to 4 characters, separators (space, comma, etc.) and operands.

A label symbol symbolically represents an address or data. A label symbol is either placed in the label column and separated from the following instruction with a colon (:), or placed in an operand.

The first 6 characters of a label symbol are significant and the 7th and following characters (if used) are ignored. Therefore, ABCDEFG and ABCDEFH are treated as the same label symbol.

Alphanumerics are generally used for label symbols, but any characters other than those used for separators and special symbols may be used.

Comments are written between the separator " ; " and a CR code; these have no influence on program execution.

Assembler directives will be explained later in this manual. These are written in the same column as the Z80 instruction mnemonic codes.

An END directive is one of the assembler directives; all assembly source programs must end with this directive.

-Characters-

Characters which are used in an assembly source program are alphanumeric, special symbols and other characters. The special symbols have functional meanings. (Separators, `[CR]`, `[SPACE]`, etc.)

1) Alphabetic characters: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

These characters are used to represent symbols and instruction mnemonic codes. A ~ F are also used for representing hexadecimal values. Further, D is used to indicate decimal and H is used to indicate hexadecimal.

2) Numerics: 0 1 2 3 4 5 6 7 8 9

These are used to represent constants and symbols. Whether a constant is a hexadecimal number or a decimal number is determined according to the rules of constants.

3) Space

Spaces are treated as separators except when they are used in comments. They perform the tabulation function on the assembly listing when they are placed between op-code and operand or between operand and comment as shown below:

```

Example:   OR [SP] FOH [SP]; A<-X0
           XYZ : PUSH [SP] AF
           ADD [SP] HL, BC [SP]; BC = COUNT
           ↓
           OR   FOH   ; A<-X0
XYZ:       PUSH  AF
           ADD  HL, BC ; BC = COUNT
           ↑    ↑
           Tab set Tab set
    
```

} Editor list

} Assembly listing

4) Colon " : "

A colon behaves as a separator when it is placed between a label symbol and an instruction. It performs the tabulation function on the assembly listing.

```

Example:   START: LD   SP, START
           MAIN:  ENT
           ↑      ↑
           Tab set Tab set
    
```

An address is assigned to the label symbol even if no instruction follows. (See the paragraph on symbols.)

```

Example:   ENTRY:           ← "ENTRY" is assigned the same address as "TOP0".
           TOPO:  PUSH  HL
    
```

5) Semicolon " ; "

A semicolon represents the beginning of a comment. None of the characters between a semicolon and a `[CR]` code have any influence on execution of the program. The semicolon is placed at the top of a line or the beginning of a comment column.

```

Example:   ;
           ; SAMPLE PROGRAM
           ;
           CMMNT: ENT           ; COMMENT
                                   Comment column
    
```

} All lines are comments.

6) Carriage return (`CR`)

A carriage return code represents the end of a line.

7) Other special symbols: + - ' () ,

All these are special symbols used in instruction statements.

8) Other symbols

Other characters are not generally used, although they may be used as symbol labels or in the comment column.

—Line—

Each line of a source program is formed of alphanumeric and symbols, and is ended with a carriage return. Except for comments, each line includes only one of the Z80 instructions, an assembler directive, an end statement or an empty statement for a skip.

Components on each line are arranged according to the tab settings when it is listed. (See the assembly listing on page 7.)

—Label Symbols—

All characters other than special symbols may be used for label symbols, but generally alphanumeric are used. Each label symbol can consist of up to 6 characters; the 7th and following characters, if used, are ignored by the assembler.

Example:	Correct	ABC	START	BUFFER	50STEP	
	Incorrect	(ABC)	,HL	IY+3	XYZ+3	← Special characters are used.
		COMPARE0] The following label are treated as the same label symbol "COMPAR".
		COMPARE1				

Assembler directive EQU defines data (1 byte or 2 bytes) for a label symbol and assigns it to the label.

Example:	ABC:	EQU	3
	CR:	EQU	0DH
	VRAM0:	EQU	D000H

Assembler directive ENT defines a label symbol as a global symbol. A colon (:) placed between a label symbol and a following instruction defines the label symbol as a relocatable instruction address.

Example:	RLDR:	ENT
	RLDR0:	PUSH HL

When a label symbol is referenced (that is, when it is used as an operand), the assembler first searches the symbol table for the specified label symbol; if it is not found, the assembler treats it as hexadecimal data. For example, when CALL ABC is encountered, the assembler searches the symbol table for ABC; if it is not found, the assembler treats it as 0ABCH and calls address 0ABC.

A label symbol used as an operand must be defined in the assembly source program unit in which it is used, or must be defined as a global symbol in another assembly source program unit. Otherwise, it is converted into binary and left undefined.

A label symbol which has once been defined cannot be defined again.

ASSEMBLY LISTING AND ASSEMBLER MESSAGES

The assembly listing is output to the CRT screen or printer when an FDOS system command ASM is executed with \$CRT/L or \$LPT/L specified as an argument. Examining the assembly listing is one of the most important procedures in assembly programming since this is when a check is made for errors in the source program.

The assembler translates the specified source program and outputs the assembly listing, which includes line numbers, relative addresses, relocatable binary codes, assembler messages and the source program list (including label symbols, Z80 instruction mnemonic codes and comments). The assembly listing is pages every 60 lines.

The comment column is displayed when the number of characters per line is set to 80, but is not displayed when it is set to 40.

The assembly listing format is shown below. Tabs are set at the beginnings of labels, op-codes, operands and comment columns.

Line number	Relative address	Relocatable binary code	Assembler message	Label	Op-code	Operand	Comment
Z80 ASSEMBLER SA-7201 <A> PAGE 01 ???/??/? ? This message is output at the top of each page.							
01	0000						
02	0000						; ASSEMBLER LIST SAMPLE
03	0000						
04	0000	P		LETNL:	EQU	0006H	
05	0000	P		MSG:	EQU	0015H	
06	0000						
07	0000			START:	ENT		; ENTRY FROM UNIT#1
08	0000			MAIN:	ENT		; ENTRY FROM UNIT#2
09	0000	310000			LD	SP, START	; INITIAL STACK POINTER
10	0003	210000	E		LD	HL, TEMPO	
11	0006	DD210000	E		LD	IX, TEMP1	
12	000A	DD360000	EE	MAIN0:	LD	(IX+CONST0), CONST1	
13	000E	00	Q		XOA	A	; A<-00
:	:	:	:	:	:	:	:
47	005A	1A		MAIN7:	LD	A, (DE)	
48	005B	B7			OR	A	
49	005C	2000	V		JR	NZ, COMP	
50	005E	EB		MAIN8:	EX	DE, HL	; EXCHANGE DE, HL
Z80 ASSEMBLER SA-7201 <A> PAGE 02 ???/??/? ? A new page is started when the number of lines on the preceding page reaches 60.							

Errors detected during assembly and definition conditions are indicated with assembler messages.

—Definition Condition Messages—

E (External)

This message indicates that an external symbol reference is being made; i.e., the label symbol by the operand is not defined in the assembly source program unit assembled.

The label symbol indicated must be defined as a global symbol in another assembly program unit for linkage with the current unit by the linker. (See "Assembler Directive ENT" on page 10.)

An undefined byte of data is treated as "00"; 2 undefined bytes of data (or an address) are uncertain.

```
Example:  E    LD      B, CONST0
           |
           |——— The byte of data "CONST0" is not defined in the program unit.
           E    CALL    SORT
           |
           |——— Address SORT is not defined in the program unit.
           EE   BIT     TOP, (IY+FLAG)
           |
           |——— The byte of data "FLAG" is not defined in the program unit.
           |
           |——— The byte of data "TOP" is not defined in the program unit.
```

P (Phase)

This message indicates that the label symbol is defined by an EQU statement with a constant value assigned. A label symbol indicated by this message can be referenced from an external file. In this case, however, the program unit including the EQU statement must be loaded before the other program units which are to be linked with it.

The P message is displayed when a label symbol different from those stored in the symbol table during PASS 1 is found.

```
Example:  P    LETNL:  EQU  0762H
           P    DATA1: EQU  3
           |
           |——— Indicates that LETNL and DATA1 are defined by EQU.
```

The P message is displayed in the relocatable binary code column rather than in the assembler message column.

—Error Messages—

C (illegal Character error)

This message indicates that an illegal character has been used as an operand.

```
Example:  C JP  +1000-3
```

F (Format error)

This message indicates that the instruction format is incorrect.

N (Non label error)

This message indicates that ENT or EQU has no label symbol.

```
Example:  N    _____ EQU  0012H
           |
           |——— No label symbol
```

L (erroneous Label error)

This message indicates that an illegal label symbol is used.

Example: L JR XYZ

↑ XYZ is not defined in the current source program.

No externally defined global symbol can be used as an operand of the JR or DJNZ commands.

The L message is displayed if such a label symbol is specified.

M (Multiple label error)

This message indicates that a label symbol is defined two or more times.

Example: M ABC: LD DE, BUFFER

}

M ABC: ENT

↑ Indicates that ABC is defined more than once.

O (erroneous Operand)

This message indicates that an illegal operand has been specified.

Q (Questionable mnemonic)

This message indicates that a mnemonic code is incorrect.

Example: Q CAL XYZ

CALL XYZ is correct.

Q PSH B

PUSH BC is correct.

S (String error)

This message indicates that single quotation mark(s) are omitted from a DEFM statement.

Example: S DEFM GAME OVER

DEFM 'GAME OVER' is correct.

V (Value over)

This message indicates that the value of the operand is out of the prescribed range.

Example: V LD A, FF8H

V SET 8, A

V JR -130

ASSEMBLER DIRECTIVES

Assembler directives (also sometimes referred to as "pseudo instructions") control assembly, but are not converted into machine language. However, in the DEFB, DEFW and DEFM directives, their operands are sometimes converted into machine language.

—ENT (entry)—

This assembler directive defines a label symbol as a global symbol. Label symbols which are referenced by two or more programs when multiple programs are linked must be defined by the entry directive.

Label symbols defined by the entry directive are included in the relocatable file so that the linker can identify them. The symbolic debugger can perform symbolic addressing using these label symbols.

Label symbols which are not defined by the entry directive contribute only to assembly of the current source program unit, and are not included in the relocatable file output by the assembler. However, labels defined by the EQU directive are exceptions since they are defined as global symbols and entry definition is not necessary.

The example below shows label symbols being referenced between program units GAUSS-MAIN and GAUSS-SR. The E message in the assembler message column indicates that a label symbol which is not defined in the current program unit is being referenced externally.

Program unit 1 "GAUSS-MAIN"	; GAUSS-MAIN	
	;	
	MAIN0: ENT	← Entry definition of label symbol MAIN0
	: :	
	CALL CMLPX	
	: :	
	CALL CMLPX+2	← No offset can be added to a label symbol which is defined externally.
: :		
END	← END is always required at the end of a program unit.	

Program unit 2 "GAUSS-SR"	; GAUSS-SR	
	;	
	CMLPX: ENT	← Entry definition of label symbol CMLPX
	: :	
	RET	
	: :	
	JP MAIN0	
: :		
END		

-EQU (equate)-

This assembler directive defines a label symbol with a numeric value (or address) assigned. The numeric value must be a decimal or hexadecimal constant. Any numeric value can be added to or subtracted from a label symbol once it is defined with a numeric value assigned; this allows a new label symbol to be defined.

The label symbol used as an address in the operand is generally treated as a relative address. However, when a specific address is assigned to the label symbol with an EQU directive, the address is not changed during assembly.

The EQU directive also defines a label symbol as a global symbol. A label defined by the EQU directive can be referenced by an external program unit. **However, program units including such directives must be loaded before other program units to be linked.**

The following example illustrates use of the EQU directive to define label symbols as monitor subroutine addresses and I/O port numbers for a specific device. The P messages indicate that the EQU directives define the label symbols as global symbols.

```

** Z80 ASSEMBLER SA-7201 <A> PAGE 01          ??/??/??
01 0000          ;
02 0000          ; MONITOR SUBROUTINE
03 0000          ;
04 0000 P        BRKEY:   EQU    001EH
05 0000 P        GETKY:   EQU    001BH
06 0000 P        PRNTS:   EQU    000CH
07 0000 P        PRNT:    EQU    0012H
08 0000 P        MSG:     EQU    0015H
09 0000 P        NL:      EQU    0009H
10 0000 P        LETNL:   EQU    0006H
11 0000 P        GETL:    EQU    0003H
12 0000          SKP      3

16 0000          ;
17 0000          ; SET PORT#: PRINTER
18 0000          ;
19 0000 P        POTFE:   EQU    FEH
20 0000 P        POTFF:   EQU    POTFE+1    ; POTFF is defined with FF (hexadecimal)
21 0000          ;                          assigned.
22 0000 P        CON1:    EQU    1
23 0000 P        CON2:    EQU    2
24 0000 P        CON3:    EQU    CON1+CON2  ; This results in assigned of 3 to CON 3. In this
                                          ; case, CON1 and CON2 must be defined in
                                          ; advance.

```

—ORG (origin)—

This assembler directive determines the object program loading address. For example, when
 ORG 2000H
 is placed at the beginning of the program to be assembled, the assembler assembles the program with a loading address of 2000H specified.

When a relocatable binary file generated with the loading address specified with the ORG directive is linked with other programs by the linker, the loading address specified with the ORG directive is effective and that specified with the linker is not.

When relocatable files with loading addresses specified with ORG directives are linked, or when more than one ORG directives is used in a program, the loading addresses specified must not overlap and must appear in the sequential order.

When a relocatable file with a loading address specified with an ORG assembler directive is converted into a system file using the LINK/S command, the specified loading address is ignored.

```

** Z80 ASSEMBLER SA-7201 <ORG> PAGE 01          ???/??/??
01 0000          ;TYPE COMMAND
02 0000          ;
03 2000          ORG    2000H
04 2000          .TYPE: ENT
05 2000 116220   LD     DE,SWTBL ; DE: = SWITCH TABLE
06 2003 CD0000   E      CALL  ?GSW   ; CHECK GLOBAL SWITCH
07 2006 D8      RET     C
08 2007 CD0000   E      CALL  C&L1  ; SELECT CRT OR LPT
09 200A CD0000   E      CALL  ?SEP   ; CHECK SEPARATOR
10 200D D8      RET     C
11 200E FE2C    CP     2CH     ; SEPARATOR = " , " ?
12 2010 3E03    LD     A,3     ; 3 IS ERR CODE
13 2012 37      SCF
14 2013 C0      RET     NZ     ; NO, ERR RETURN
15 2014 CD0000   E      TYPE0: CALL  ?LSW   ; CHECK LOCAL SWITCH
16 2017 D8      REC     C
17 2018 3E08    LD     A,8     ; 8 IS ERR CODE
18 201A 37      SCF
19 201B C0      RET     NZ     ; ERROR, LSW EXIST
20 201C 0E80    LD     C,128   ; LU#: = 128
21 201E D9      EXX
22 201F 0604    LD     B,4     ; DEFAULT MODE = ASC
23 2021 D9      EXX
  :      :      :      :      :
55 2062 88      SWTBL:  DEFB  88H     ; /P
56 2063 FF      DEFB  FFH     ; END OF SWTBL
57 2064          BUFFER:  DEFS  128   ; 128 BYTE BUFFER
58 20E4          END

```

```

** Z80 ASSEMBLER SA-7021 <ORG> PAGE 02          ???/??/??
.TYPE 2000  BUFFER 2064  SWTBL 2062  TYPE0 2014  TYPE10 203C
.TYPE20 2048  TYPEER 2058

```

—DEFB n (define byte)—

This directive sets constant n (1 byte) in the address of the line on which the directive is specified. A label symbol defined with a constant (1 byte) assigned may be used in place of n.

This directive (as well as DEFW and DWFDM) is used to form message data or a graphic data group for a code conversion table or other table.

The following example forms the message "ERROR" in ASCII code. Since it uses ODH as an end mark, monitor subroutine 06B3H can be used to output the message.

```
13 1FF3 B7          OR      A
14 1FF4 CA0000     E      JP      Z, READY
15 1FF7 110020     LD      DE, MESGO
16 1FFA CD1500     CALL   MSG
17 1FFD C30000     E      JP      MAIN2
18 2000 P          MSG:    EQU    06B5H
19 2000           ;
20 2000           ; MESSAGE GROUP
21 2000           ;
22 2000           MESGO:  ENT      ;"ERROR"
23 2000 45        DEFB   45H
24 2001 52        DEFB   52H
25 2002 52        DEFB   52H
26 2003 4F        DEFB   4FH
27 2004 52        DEFB   52H
28 2005 0D        DEFB   0DH
```

—DEFB 'S', DEFB "S" (define byte)—

This directive sets the ASCII code corresponding to the character enclosed in single or double quotation marks in the address of the line on which the directive is specified.

Since this directive converts characters to ASCII code, the above example can be rewritten as follows.

```
21 2000           MESGO:  ENT      ;"ERROR"
22 2000 45        DEFB   'E'
23 2001 52        DEFB   'R'
24 2002 52        DEFB   'R'
25 2003 4F        DEFB   'O'
26 2004 52        DEFB   'R'
27 2005 0D        DEFB   0DH
28 2006 16        DEFB   '␣'
29 2007 13        DEFB   '␣'
30 2008 0D        DEFB   0DH
31 2009 27        DEFB   "' '
32 200A 22        DEFB   '" "'
```

—DEFW nn' (define word)—

This directive sets n' in the address of the line on which the directive is specified and n in the following address; in other words, it sets two bytes of data. A label symbol may be used in place of nn'.

```

39 5FF1          CMTD:  ENT          ; COMMAND TABLE
40 5FF1 41      DEFB   41H
41 5FF2 0053    DEFW   CMDA
42 5FF4 42      DEFB   42H
43 5FF5 1E53    DEFW   CMDB+3
44 5FF7 53      DEFB   53H
45 5FF8 0000    E      DEFW   CMDS
46 5FFA 0D      DEFB   0DH
47 5FFB          CONST0: ENT
48 5FFB 0F01    DEFW   010FH
49 5FFD          CONST1: ENT
50 5FFD 660D    DEFW   0D66H

```

—DEFM 'S', DEFM "S" (define message)—

This directive sets the character string enclosed in single or double quotation marks in ASCII code in addresses starting at that of the line on which the directive is specified. The number of characters must be within the range from 1 to 64. On the assembly listing, codes for 4 characters are output on each line.

The example on the preceding page can be written as follows with this directive.

```

21 2000          MESGO:  ENT          ; "ERROR"
22 2000 4552524F DEFM   ' ERROR '
23 2004 52      DEFB   0DH
24 2005 0D      DEFB   0DH
25 2006 16134142 DEFM   '  AB '
26 200A 0D      DEFB   0DH
27 200B 41274247 DEFM   " A' B' C' "
28 200F 4327    DEFB   0DH
29 2011 0D      DEFB   0DH

```

—DEFS nn' (define storage)—

This directive reserves nn' bytes of memory area starting at the address of the line on which the directive is specified.

This directive adds nn' to the reference counter contents; the contents of addresses skipped are not defined.

The following example reserves buffer areas.

02	4BB8	TEMP0:	ENT		; BUFFER A
03	4BB8		DEFS	1	
04	4BB9	TEMP1:	ENT		; BUFFER B
05	4BB9		DEFS	2	
06	4BBB	TEMP2:	ENT		; BUFFER C
07	4BBB		DEFS	2	
08	4BBD	TEMP3:	ENT		; BUFFER D
09	4BBD		DEFS	128	
10	4C3D	BFFR:	ENT		; BUFFER E
11	4C3D		DEFS	A	
12	4C47	BUFFER:	ENT		; BUFFER F
13	4C47		DEFS	2	

The addresses are increased by amounts corresponding to the values indicated by the respective DEFS statements.

—SKP n (skip n lines)—

This directive advances the assembly listing by n lines to make the list easy to read.

```
30          COMMON: ENT          ; NORMAL RETURN
31 3BB8 AF   XOR      A          ; A<--00
32 3BB9 32B84B LD      (TEMPO), A ; CLEAR CMD BUFFER
33 3BBC 110020 LD      DE, MESGO ; "READY"
34 3BBF C9   RET
35 3BC0          SKP      3
```

} 3 line feeds are made.

```
39 3BC0          ;
40 3BC0          ; ABNORMAL RETURN
41 3BC0          ;
42 3BC0          ABNRET: ENT     ; SET INVALID MODE
```

—SKP H (skip home)—

This directive advances the page during output of the assembly listing.

—END (end)—

This directive declares the end of the source program. All source programs must be ended with this directive. Assembly operation is not completed if this directive is omitted.

The assembly outputs

```
END?
```

when it reads a source file which doesn't include an END directive.

MESSAGE TABLE

Definition status message	Meaning	Example
E (External)	Indicates that a label symbol is being referenced externally; that is, the label is not defined in the current source program unit.	<pre>E LD B, CONST0 ↑——The data byte "CONST0" is undefined. E CALL SORT ↑——The address "SORT" is undefined. EE BIT TOP, (IY+FLAG) ↑——The data byte "FLAG" is undefined. ↑——The data byte "TOP" is undefined.</pre>
P (Phase)	Defines a label symbol with a constant assigned. This message is also output when a label symbol is encountered during pass 2 which was not encountered during pass 1.	<pre>P LETNL : EQU 0762H P DATA1 : EQU 3 ↑——LETNL and DATA1 are defined by EQU. The P message is displayed in the relocatable binary code column rather than in the assembler message column.</pre>

Error message	Meaning	Example
C (illegal Character error)	Indicates that an illegal character is used in the operand.	<pre>C JP +1000-3</pre>
F (Format error)	Indicates that the instruction format is incorrect.	
N (Non label error)	Indicates that no label symbol is specified for ENT or EQU.	<pre>N EQU 0012H ↑——No label symbol</pre>
L (erroneous Label error)	Indicates that an illegal label symbol is used.	<pre>L JR XYZ ↑——XYZ is not defined in the current program. No externally defined global symbol can be used as the operand of a JR or DJNZ command. If such a label symbol is specified, the L message is displayed.</pre>
M (Multiple label error)	Indicates that a label symbol is defined two or more times.	<pre>M ABC : LD DE, BUFFER ? M ABC : ENT ↑——ABC is defined twice.</pre>
O (erroneous Operand)	Indicates that an illegal operand is specified.	
Q (Questionable mnemonic)	Indicates that the mnemonic code is incorrect.	<pre>Q CAL XYZ CALL XYZ is correct.</pre>
S (String error)	Indicates that single or double quotation mark(s) are omitted.	<pre>S DEFM GAME OVER DEFM 'GAME OVER' is correct.</pre>
V (Value over)	Indicates that the value of the operand is out of the prescribed range.	<pre>V LD A, FF8H V SET 8, A V JR -130</pre>
END?	Indicates that the END directive is missing from the source program.	

Note: Refer to the System Error Messages in the System Command manual for other system errors.

MESSAGE TABLE

Error message	Meaning	Example
END?	Indicates that the END directive is missing from the source program.	
V (Value over)	Indicates that the value of the operand is out of the prescribed range.	V LD A, FFH V SET R, A V JR, LSH
S (String error)	Indicates that single or double quotation mark(s) are omitted.	DETM GAME OVER DETM GAME OVER, A
Q (Questionable mnemonic)	Indicates that the mnemonic code is incorrect.	CALL XYZ CALL XYZ, A
O (Operand)	Indicates that an illegal operand is specified.	
M (Multiple label error)	Indicates that a label symbol is defined two or more times.	M ARG, ENT M ARG, LD, BUFFER ABC defined twice
L (Erroneous label error)	Indicates that an illegal label symbol is used. No externally defined global symbol can be used as the operand of a JR or DJNZ command. If an illegal label symbol is specified, the L message is displayed.	JR XYZ XYZ is not defined in the current program.
N (Non label error)	Indicates that no label symbol is specified for ENT or EQU.	ENT 0012H No label symbol
F (Format error)	Indicates that the instruction format is incorrect.	
C (Illegal Character error)	Indicates that an illegal character is used in the operand.	C JP, +1000-3
P (Pass)	label symbol is encountered during pass 2 which was not encountered during pass 1. This message is also output when a label symbol is assigned. Defines a label symbol with a constant.	LETN EQU 0552H DATA EQU 3 LETN and DATA are defined by EQU. The P message is displayed in the relationship matrix code column rather than in the assembler message column.
E (Error)	Indicates that a label symbol is being referenced externally, that is, the label is not defined in the current source program unit. The data byte "TOP" is undefined. The data byte "FLAG" is undefined. The address "SORT" is undefined. The label "CALL SORT" is undefined. The label "B.CONSTO" is undefined.	LD, B, CONSTO CALL SORT BIT TOP, (Y+FLAG) The data byte "TOP" is undefined. The data byte "FLAG" is undefined.

Note: Refer to the System Error Messages in the System Command manual for other system errors.