Chapter 11 Assembler r86

The r86 is an assembler for 8086, 80186, and 80286. The r86 aims at assembling of codes generated by the C compiler. Therefore, the r86 is not complicated as MASM and is very simple. Additionally, a new feature is provided that converts a jump command, which cannot reach, into long jump command automatically, which is not supported by MASM.
This chapter describes how to operate the r86.

11.1 Operation

A r86 command has the following format.

r86 [option] input file

If an extension of the input file is omitted, ".r86" is put temporarily. The r86 reads the input file and creates ".obj" files having the same name in the same directory. These files have a format which is to be linked with MS-LINK.
The following options are provided.

-o X	An object is made in file X.
-p X	Listing is made in file X. If this option is omitted, no listing is made.
-m X	A module name of the object is set to X.
-q	All jump commands are converted to long jump commands to reduce the assembling time. If this option is not put, jump commands are assembled to those as short as possible.

11.2 Language specifications

This section describes how to write the r86 input language.

11.2.1 Overview

The following shows an example of the r86 assembly language.

```
CGROUP      GROUP     TEXT

DATA        DSEG
ignore_case:          db          0

TEXT        CSEG
            EXTRN     getch

getc::
            CALL      getch
            CMP       [ignore_case].B, 0
            JE        _1
            CMP       AL, ‘a’
            JB        _1
            CMP       AL, ‘z’
            JA        _1
                      SUB         ‘a’ - ‘A’
            _1:
            RET

            END
```

As shown in the above example, Intel’s standard mnemonic is applied to machine language commands in the same manner as MASM. However, this assembler does not use types. Therefore, [ ] must be put to refer to a memory. Additionally, if it is not clear whether a byte-operand or word-operand is used, a type specified operator such as .B or .W is absolutely necessary.
Pseudo commands to define segments are also different from those in MASM.

11.2.2 Names

A name is also called “symbol” that defines an address or absolute value used in the program and used to refer to them.
The following characters are used to specify a name.

a - z A - Z 0 - 9 _ . $ @ ?

However, a number cannot be put at the beginning of a name. For user defined names, upper and lower cases are distinct. However, upper and lower cases are not distinct for keywords such as commands.

11.2.3 Numeric constants

A numeric constant begins with numbers, and then alphanumeric characters follow them. The base notation is determined by the last character as shown below.

h, H	Hexadecimal notation
o, O, q, Q	Octal notation
b, B	Binary notation
0 - 9	Decimal notation

The following example shows that 26 in decimal are expressed by several base notations.

```
01ah    26      32Q     32o     00011010B
```

11.2.4 Character constants

A character constant contains an ASCII character code, which is a specific character sting enclosed by ‘ or “.
‘ and “ can also be used as character constants by enclosing them using ‘ or “.

```
‘”hello”, said he.’
“””hello””, said he.”
“I’m Stallman.”
‘I’’m Stallman.’
```

If a character constant is used in the expression, a character string with more than 3 characters cannot be written. Such character string is allowed only in the operand of the DB pseudo command.

11.2.5 Location counter

Symbol “$” shows a current value of the location counter.

```
xyz:    dw      1234h
xyz_lo  equ     $-2     ;=xyz
xyz_hi  equ     $-1     ;=xyz + 1
        jump    $
```

The jump command located at the last jumps to the jump command itself.

11.2.6 Expressions

An expression is that names, numeric constants, and/or character constants (called primary) are combined by operators and can be put in an operand of the command. The following operators can be used.

```
( )        [ ]
HIGH    LOW     SEG      unary + -
*   /   %   >>   <<
+   -
==   !=   <=   >=   <   >
~
&
|   ^
:
.B   .W   .D   .Q   .T
```

Operators at upper positions have higher precedence.
Therefore, x * y + z == 4 & (5|6) equals (((x * y) + z) == 4) & (5|6).

The following describes each operator.

(x)	This gathers expressions. The value is x.
x.B	Byte type is put on x.
x.W	Word type is put on x.
x.D	Double-word type is put on x.

x.Q	Quad type is put on x.
x.T	10-byte type is put on x.
x:y	Segment base of y is changed to x. x must be a segment register.
[x]	Contents of memory specified by x are shown.
x [y]	This is the same as [(x) + (y)].
x | y	Logical OR of x and y
x ^ y	Exclusive logical OR of x and y
x & y	Logical AND of x and y
~x	NOT of x (one's complement)
x == y	When x equals y, the value of this expression is 0FFFFh, otherwise it is 0.
x != y	When x does not equal y, the value of this expression is 0FFFFh, otherwise it is 0.
x <= y	When x equals or is smaller than y, the value of this expression is 0FFFFh, otherwise it is 0.
x >= y	When x equals or is bigger than y, the value of this expression is 0FFFFh, otherwise it is 0.
x < y	When x is smaller than y, the value of this expression is 0FFFFh, otherwise it is 0.

x > y	When x is bigger than y, the value of this expression is 0FFFFh, otherwise it is 0.
x + y	Sum of x and y
x - y	y is subtracted from x.
x * y	x is multiplied by y.
x / y	x is divided by y.
x % y	Reminder after x is divided by y.
x >> y	x is shifted right by y bits.
x << y	x is shifted left by y bits.

+x	x is used for a value of this expression.
-x	Two's complement of x is a value of this expression.
HIGH x	This is the same as (x) >> 8.
LOW x	This is the same as (x) & 0FFh.
SEG x	Segment base of x

These operations are calculated by unsigned 16-bit integer.

11.2.7 Statements

This assembler handles one line as a statement. The statement has either of the following formats.

```
name            command [operand] [; comment]
[name : [:]]    command [operand] [; comment]
```

Putting ":" after the name may define the current value of the location counter to this name. A name defined in such manner is called a label. If two ":" exist, this label is defined as an external name and can be referred to from other files.

The command is either pseudo command or machine language command and must satisfy the conditions for the name.
Each command has different operand format. However, several expressions are separated by a comma in most operands.

A comment begins with ";" and finishes at the end of the line. Any character string can be put in the comment. Additionally, a comment can be written in a line having no commands.

11.2.8 Pseudo commands

This assembler provides the following pseudo commands.

```
GROUP
CSEG/DSEG/SSEG/ESEG
DB/DW/DD
RS
EQU
```

EXTRN

PUBLIC
EVEN
END
.8086/.8087/.186/.286c/.286p/.287

11.2.8.1 GROUP
The GROUP pseudo command has the following format.

group name GROUP segment name [, segment name…]

A group name must be a name. Segment names written in the operand of the GROUP pseudo command are gathered and a group name is put on it.

11.2.8.2 CSEG/DSEG/SSEG/ESEG
The CSEG pseudo command has the following format.

[segment name] CSEG [align type] [combination type] [‘class’]

The same format is applied to DSEG, SSEG, and ESEG. These commands specify which segment the subsequent part is located at.
This designation is valid until next CSEG, DSEG, SSEG, or ESEG command appears.

Among four commands, only default segment register is different when referring to the data in the segment. CSEG, DSEG, SSEG, and ESEG use CS, DS, SS, and ES registers as a default segment register, respectively.

A segment name must be a name. This name becomes a name of the segment and can be referred to in the GROUP command or expression. If a segment name appears in the expression, the expression becomes the base value of the segment. If the segment name is omitted, the following names are specified.

CSEG CODE
DSEG DATA
ESEG EXTRA
SSEG STACK

The align type is specified when the start address of the segment is started from a certain boundary.
The following align types are provided.

PAGE	Adjusted to 256-byte boundary.
PARA	Adjusted to 16-byte boundary.
WORD	Adjusted to 2-byte boundary.
BYTE	No boundary is adjusted.

The default align type of CSEG is BYTE and that of others is WORD.
The combination type specifies a segment align method. The following combination types are provided.

PUBLIC	Segments having the same name are combined sequentially.
MEMORY	This is the same as PUBLIC.
STACK	This is almost the same as PUBLIC. However, the initial value of the stack pointer in the EXE file is set at the address next to the final address of the segment specified by the stack pointer.
COMMON	This overlaps segments with the same name and allocates them.
LOCAL	This does not combine segments having the same name.

If the align type is omitted, it is interpreted as PUBLIC.
The class name specifies a class belonging to the segment. When the class name is omitted, the following names are specified.

CSED	CODE
DSEG	DATA
ESEG	EXTRA
SSEG	STACK

For details of segment allocation, see the manual for MS-LINK.

11.2.8.3 DB
The DB pseudo command has the following format.

```
[label]   DB   expression [, expression …]
```

A long character constant can be written in the expression of the operand. The DB pseudo command embeds a value of the given expression in the object in units of bytes (8 bits).

11.2.8.4 DW

The DW pseudo command has the following format.

```
[label]  DW  expression [, expression …]
```

The DW pseudo command embeds a value of the given expression in the object in units of words (16 bits).

11.2.8.5 DD

The DD pseudo command has the following format.

```
[label]  DD  expression [, expression …]
```

The DW pseudo command embeds a value of the given expression in the object in units of double-words (32 bits). However, only address information can be written in this command. 32-bit integer cannot be written. For address information, the offset and segment are generated in that order.

11.2.8.6 RS

The RS pseudo command has the following format.

```
[label]  RS  expression
```

The RS pseudo command allocates a memory area for the number of bytes in the given expression. This area is not initialized.

11.2.8.7 EQU

The EQU pseudo command has the following format.

```
name  EQU  expression
```

The EQU pseudo command defines a value of the expression as a name.

11.2.8.8 EXTRN

The EXTRN pseudo command has the following format.

```
EXTRN  name  [, name]
```

The EXTRN pseudo command recognizes two names as an external name defined by other files.
When referring to the name, putting two # may obtain the same effect as that of the EXTRN declaration.

11.2.8.9 PUBLIC

The PUBLIC pseudo command has the following format.

PUBLIC name [, name]

The PUBLIC pseudo command recognizes two names as an external name defined in this file.
When defining the label, putting two : may obtain the same effect as that of the PUBLIC declaration.

11.2.8.10 EVEN

The EVEN pseudo command has the following format.

EVEN

The EVEN pseudo command does nothing if the current value of the location counter is even, and keeps a one-byte area if it is odd.

11.2.8.11 END

The END pseudo command has the following format.

END

The END pseudo command shows the last of the file. If a line exists after the END pseudo command, relevant error is reported.

11.2.8.12 .8086/.8087/.186/.286c/.286p/.287

These pseudo commands change operable command sets.

.8086	Changes to a mode in which only 8086 commands are interpreted.
.8087	Changes to a mode in which only 8087 commands are interpreted.
.186	Changes to a mode in which 8086 and 8087 commands are interpreted.
.286c	Changes to a mode in which only non-protect 8086 commands are interpreted. This command functions in the same manner as .186.
.286p	Changes to a mode in which all 80286 commands are interpreted.
.287	Changes to a mode in which all 80287 commands are interpreted.

11.2.8.13 Line number control

The line number control command has the following format.

line number “file name”

A line number must be expressed in decimal. This command informs a next line number and file name to the assembler. According to this information, the assembler shows an error message. The pre-processor mainly generates this command.

11.2.8.14 Debug line number information

The debug line number information command has the following format.

% line number

A line number must be expressed in decimal. When the assembler encounters this command, it generates the line number information in the object. This information is used by the symbolic debugger.

11.2.9 List of machine language commands

The mnemonics of machine language commands are almost the same as those defined by Intel. However, the following are different.

r86	Intel	
JMPF label	JMP label	(Jumps to other segments.)
CALLF label	CALL label	(Calls other segment.)
RETF	RET	(Returns to other segment.)
MOVSB	MOVS AL, AL	
MOVSW	MOVS AX, AX	
LDSB	LODS AL	
LODSW	LODS AX	
STOSB	STOS AL	
STOSW	STOS AX	
CMPSB	CMPS AL	
CMPSW	CMPS AX	
SCASB	SCAS AL	
SCASW	SCAS AX	

Additionally, the r86 can specify the shift cycle twice or more such as SHL AX, 4 even in the 8086 mode. SHL AX, 1 is generated four times. This also applies to SHR, SAL, ROL, ROR, RCL, and RCR.

All machine language commands are listed in the following. Symbols used in the description have the following meanings.

acc	AX or AL register
mem	Memory is referred to.
reg	AX, BX, CX, DX, SI, DI, BP, SP, AL, AH, BL, BH, CL, CH, DL, DH
reg16	AX, BX, CX, DX, SI, DI, BP, SP
seg	CS, DS, SS, ES
imm	Constant or jump destination

The following shows the list of the machine language commands to be used in the 8086, 80186, and 80286 modes.

AAA	
AAD	
AAM	
AAS	
ADC	acc, imm
ADC	mem, reg
ADC	reg, reg/mem
ADC	reg/mem, imm
ADC	acc, imm
ADD	mem, reg
ADD	reg, reg/mem
ADD	reg/mem, imm
AND	acc, imm
AND	mem, reg
AND	reg, reg/mem
AND	reg/mem, imm
CALL	imm
CALL	reg/mem
CALLF	imm
CALLF	imm, imm
CALLF	reg/mem
CBW	
CLC	
CLD	
CLI	
CMC	
CMP	acc, imm
CWD	
DAA	
DAS	
DEC	reg/mem
DEC	reg16
DIV	reg/mem
ESC	imm, reg/mem
HLT	
IDIV	reg/mem
IMUL	reg/mem
IN	acc, dx
IN	acc, imm
INC	reg/mem
INC	reg16
INT	imm
INTO	
IRET	
JA	imm
JAE	imm
JB	imm
JBE	imm
JC	imm
JCXZ	imm
JE	imm
JG	imm
JGE	imm
JL	imm

CMP mem, reg
CMP reg, reg/mem
CMP reg/mem, imm
CMPSB
CMPSW

JLE imm
JMP imm
JMP reg/mem
JMPF imm
JMPF imm, imm

JMPF reg/mem
JMPS imm
JNA imm
JNAE imm
JNB imm
JNBE imm
JNC imm
JNE imm
JNG imm
JNGE imm
JNL imm
JNLE imm
JNO imm
JNP imm
JNS imm
JNZ imm
JO imm
JP imm
JPE imm
JPO imm
JS imm
JZ imm
LAHF
LDS reg16, mem
LEA reg16, mem
LES reg16, mem
LODSB
LODSW
LOOP imm
LOOPE imm
LOOPNE imm
LOOPNZ imm
LOOPZ imm

MOV acc, mem
MOV mem, acc
MOV mem, reg
MOV reg, imm
MOV reg, reg/mem
MOV reg/mem, imm
MOV reg/mem, seg
MOV seg, reg/mem
MOVSB
MOVSW
MUL reg/mem
NEG reg/mem
NOP
NOT reg/mem
OR acc, imm
OR mem, reg
OR reg, reg/mem
OR reg/mem, imm
OUT dx, acc
OUT imm, acc
POP mem
POP reg16
POP seg
POPF
PUSH mem
PUSH reg16
PUSH seg
PUSHF
RCL reg/mem, cl
RCL reg/mem. imm
RCR reg/mem, cl
RCR reg/mem, imm
RET

RET imm

TEST reg, reg/mem

```
RETF
RETF    imm
ROL     reg/mem, cl
ROL     reg/mem, imm
ROR     reg/mem, cl
ROR     reg/mem, imm
SAHF
SAL     reg/mem, cl
SAL     reg/mem, imm
SAR     reg/mem, cl
SAR     reg/mem, imm
SBB     acc, imm
SBB     mem, reg
SBB     reg, reg/mem
SBB     reg/mem, imm
SCASB
SCASW
SHL     reg/mem, cl
SHL     reg/mem, imm
SHR     reg/mem, cl
SHR     reg/mem, imm
STC
STD
STI
STOSB
STOSW
SUB     acc, imm
SUB     mem, reg
SUB     reg, reg/mem
SUB     reg/mem, imm
TEST    acc, imm
TEST    mem, reg
TEST    reg/mem, imm
WAIT
XCHG    acc, reg
XCHG    mem, reg
XCHG    reg, reg/mem
XLAT
XOR     acc, imm
XOR     mem, reg
XOR     reg, reg/mem
XOR     reg/mem, imm
```

The following shows the list of the machine language commands to be used in the 80186, and 80286 modes.

```
BOUND reg, mem
ENTER  imm, imm
INSB
INSW
IMUL    reg16, imm
IMUL    reg16, reg16/mem, imm
LEAVE
OUTSB
```

```
OUTSW
PUSH    imm
PUSHA
POPA
RCL     reg/mem, imm
RCR     reg/mem, imm
ROL     reg/mem, imm
ROR     reg/mem, imm
SAL     reg/mem, imm
SAR     reg/mem, imm
SHL     reg/mem, imm
SHR     reg/mem, imm
```