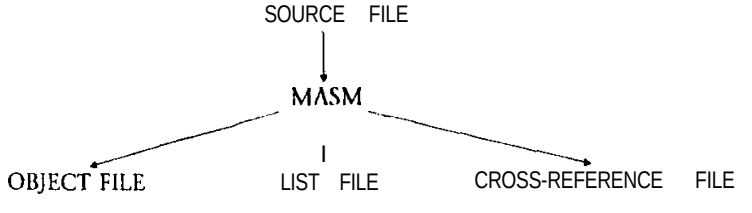
MASM and *LINK* *Options*

MASM

The MASM assembler translates an assembly language source file Into a machine language object file. It generates three files, as shown:



The *object file* contains the machine language translation of the assembly language source code, plus other information needed to produce an executable file.

The list *file* is a text file that gives assembly language code and the corresponding machine code, a list of names used In the program, error messages, and other statistics, It Is helpful in debugging.

The *cross-reference file* lists names used in the program and line numbers where they appear. It makes large programs easier to follow. As generated, it is not readable: the CREF utility program may be used to convert It lo a legible form.

MASM Command Line

For MASM version 5.0, the mast general command line is

```
MASM options source_file,object_file,list_file,cross-
ref_file
```

MASM 4.0 has the same command line, except that the options appear last.

The default extension for the object file is .OBJ, for the listing file it is .LST, and for the cross-reference file it is .CRF.

For example, suppose MASM is on a disk in drive C, source file FIRST.ASM is on a disk in drive A, and C is the logged drive. To create object file FIRST.OBJ, listing file FIRST.LST, and cross-reference file FIRST.CRF on drive A, we could type

```
C>MASM A:FIRST.ASM, A:FIRST.OBJ,A:FIRST.LST,A:FIRST.CRF
```

A simpler way to get the same result is

```
C>MASM A:FIRST,A: ,A: ,A:
```

A semicolon instead of a comma on the MASM command line tells the assembler not to generate any more files. For example, if we type

```
C>MASM A:FIRST,A:;
```

Then MASM will generate only FIRST.OBJ. If we type

```
C>MASM A:FIRST,A:,A:;
```

Then we get FIRST.OBJ, FIRST.LST, but not FIRST.CRF.

It's also possible to let MASM prompt you for the files you want. For example, suppose we want .OBJ and .CRF files only.

```
C>MASM A: FIRST
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

Object filename [FIRST.OBJ]:  A: <Enter>
Source listing [NUL.LST]:     <Enter>
Cross-reference [NUL.CRF]:    A: FIRST <Enter>

 50110 + 231323 Bytes symbol space free
                      0 Warning Errors
                      0 severe  Errors
```

The first response just given means that we accept the name FIRST.OBJ for the object file. The second one means that we don't want a listing file (NUL means no file). The third one means we want a cross-reference file called FIRST.CRF.

Options

The MASM options control the operations of the assembler and the format of the output files. Table D.l gives a llst of some commonly used ones. For a complete list, see the Microsoft Programmer's Guide.

Several options may be specified on a command line. For example,

```
C>MASM /D /W2 /Z /ZI FIRST;
```

Table **D.1** Some **MASM** Options

Option	*Action*
IA	Arrange source segments in alphabetical order.
/C	Create a cross-reference file.
ID	Create pass 1 listing (see below)
/ML	Make names case sensitive.
/R	Accept 8087 floating-point Instructions.
/S	Leave source segments in original order
/W{0|1|2}	Set error level display: (default = 1): 0 = illegal statements 1 = ambiguous or questionable statements 2 = statements that may produce inefficient code
/Z	Display the lines containing errors.
/ZI	Write symbolic information *to* the object file (use with CODEVIEW).

A MASM Demonstration

To show what the MASM output files Look like, the following program SWAP.ASM will be assembled. It swaps the content of two memory words.

Program Listing **PGMD_1.ASM**

```
TITLE PGMD_1: SWAP WORDS
.MODEL SMALL
.STACK 100H
.DATA
WORD1 DW     10
WORD2 DW     20
.CODE
MAIN   PROC
       MOV    AX,@DATA
       MOV    DS,AX
       MOV    AX,WORD1
       XCHG   AX,WORD2
       MOV    WORD1,AX
       MOV    AH, 4CH
       INT    21H
ENDP
       END    MAIN
```

```
C>MASM A:PGMD_1,A:,A:,A:

Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

  47358 + 390893 Bytes symbol space free
                              0 Warning Errors
                              0 severe  Errors
```

The listing file is shown in Figure D.1.

```
C>TYPE A:PGMD_1.LST
```

Down the left side of the listing are the line numbers. Next we have a column of offset addresses (in hex), relative to stack, data, and code segments. Alter that comes the machine code translation (in hex) of the instructions.

Two-Pass Assembly and the SYMBOL TABLE

MASM makes two passes through the source file. On the first pass, MASM checks for syntax errors and creates a *symbol table* of names and their relative locations within a segment. To keep track of locations, it uses a *location counter*. The location counter is reset to 0 at the beginning of a

```
Microsoft (R) Macro Assembler Version 5.10     9/6/91 00:43:35
PGMD_1: SWAP WORDS                                          Page-1
       1                                TITLE PGMD_1:SWAP WORDS
       2                                .MODEL  SMALL
       3                                .STACK  100H
       4                                .DATA
       5  0000  000A                    WORD1   DW      10
       6  0002  0014                    WORD2   DW      20
       7                                        .CODE
       8  0000                          MAIN    PROC
       9  0000  B8 ---- R                       MOV     AX,@DATA
      10  0003  8E D8                           MOV     DS,AX
      11  0005  A1 0000 R                       MOV     AX,WORD1
      12  0008  87 06 0002 R                    XCHG    AX,WORD2
      13  000C  A3 0000 R                       MOV     WORD1,AX
      14  000F  B4 4C                           MOV     AH,4CH
      15  0011  CD 21                           INT     21H
      16  0013                          MAIN    ENDP
      17                                END     MAIN
Microsoft (R) Macro Assembler Version 5.10     9/6/91 00:43:35
PGMD_1: SWAP WORDS                             Symbols-1
Segments and Groups:

                N a m e                 Length   Align   CombineClass
DGROUP. . . . . . . . . .               GROUP
        -DATA. . . . . . . .            0004     WORD    PUBLIC'DATA'
        STACK. . . . . . . .            0100     PARA    STACK   'STACK'
_TEXT . . . . . . . . . .               0013     WORD    PUBLIC'CODE'

Symbols:

                N a m e                 Type     Value   Attr
MAIN. . . . . . . . . . .               N PROC   0000    -TEXT Length = 0013
WORD1 . . . . . . . . . .               L WORD   0000    _DATA
WORD2 . . . . . . . . . .               L WORD   0002    _DATA

@CODE . . . . . . . . . .               TEXT     _TEXT
@CODESIZE . . . . . . . .               TEXT     0
@CPU. . . . . . . . . . .               TEXT     0101h
@DATASIZE . . . . . . . .               TEXT     0
@FILENAME . . . . . . . .               TEXT     PGMD_1
@VERSION. . . . . . . . .               TEXT     510
        17 Source Lines
        17 Total Lines
        20 Symbols
 47358 + 390893 Bytes symbol space free
        0 Warning Errors
        0 Severe Errors
```

segment. When an instruction is encountered, the location counter is increased by the number of bytes needed for the machine code of the instruction. When a name is encountered, It is entered in the symbol table along with the location counter's value. The symbol table appears near the bottom of the .LST file; in the preceding example, the symbols are MAIN, WORD1, and WORD2. The MASM /D option causes the .LST file to include pass 1 error messages. Whether these are actually errors is determined in pass 2.

On the second pass, MASM completes error checking and machine codes the instructions, except for those instructions that refer to names in other object modules. The .LST file is also created.

The reason MASM needs two passes to assemble a program is that some instructions may refer to names that appear later on in the source file. These instructions can be machine-coded only after their relative locations have been determined from the symbol table.

The object file (PGMD_1.OBJ) that MASM creates is not executable. The final addresses of the variables need to be determined by the LINK program (see later description). In the .LST file, these addresses are marked by a "R" (relocatable) symbol (lines 9, 10, 11, 12, 13).

The Cross-Reference File

The cress-reference *file* (here PGMD_1.CRF) contains information on names-where they are defined and the line numbers where they appear in the .LST file. The .CRF file is not printable; the CREF program, on the DOS disk, converts it to a .REF file that has an ASCII format:

```
Microsoft     Cross-Reference   Version   5.10    Fri Sep 06 01:33:52 1991
PGMD_1:  SWAP  WORDS
 Symbol  Cross  Reference          (#  definition,  +  modification)   Cref-1

@CPU . , . . , . . . .       1#
@VERSION . . . . . , , .     1#

CODE . . . . . .             1

D A T A . . . . . .          4
DGROUP . . . . . .           9

MAIN . . . . . .             8#          16          17

STACK. . . . . .             3           #3

WORD1. . . . . .             5#          11          13t
WORD2. . . . . .             6#          12+

_DATA. . . . . .             4#
-TEXT. . . . . .             7#

11  Symbols
```

```
C>CREF A:PGMD_1;

Microsoft (R) Cross-Reference Utility Version 5.10
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

11 Symbols
```

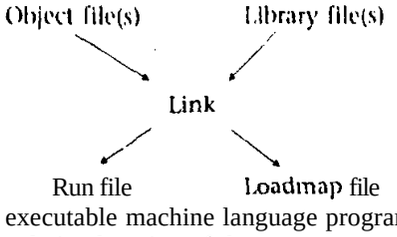
The output is the file PGMD_1.REF, which can be printed by using the TYPE command (Figure D.2).

```
C>TYPE PGMD_1.REF
```

LINK

The job of the LINK program is to link object files (and possibly library files) into a single executable file. To do this, it must resolve reference to names used in one module but defined in another. The mechanism for doing this is explained in Chapter 14. LINK must be used even if there is only one object file.

The input to LINK is one or more object and library files, and the output is a run file and an optional loadmap file, as shown:



The run file is an executable machine language program. The loadmap file gives the size and relative location of the program segments.

LINK Command Line

For LINK version 5.0, the most general command line is

```
LINK options object_file_list,run_file,loadmap_file,li-
brary_list
```

The only option you will be likely to use is /CO, which causes extra information for CODEVIEW lo he included.

'The *object_file_list* is a list of object files to be linked. It begins with the name of the object file containing the main program; the other object files usually contain procedures that arc called by the main program and by each other. The file names are separated by blanks or "+"

The *run_file* has an .EXE extension. It is an executable file unless the program is a .COM format program, in which case one more step is needed to produce an executable file. .COM programs are discussed in Chapter 14.

The *library_list* consists of library files, if any, separated by blanks or "+". Library files usually have a .LIB extension, and they often contain standard routines used by many programs, such as I/O routines. An example appears in Chapter 14.

For example, suppose LINK is on a disk in drive C and the files to be linked are in drive A. The main object file is FIRST.OBJ, other object files are SECOND.OBJ and THIRD.OBJ. To create a run file FIRST.EXE and a loadmap file FIRST.MAP, we could type

```
C>LINK A:FIRST+SECOND+THIRD,A:FIRST,A:FIRST;
```

or just

```
C>LINK A:FIRST+SECOND+THIRD,A:,A:;
```

The semicolon at the end means that there are no library files. As with MASM, it's possible to run LINK interactively:

```
C>LINK FIRST+SECOND+THIRD

Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights
reserved.

Run File [FIRST.EXE]:  <Enter>
List File [NUL.MAP]    A:FIRST <Enter>
Libraries: [.LIB]      <Enter>
```

The first response means that we accept the name FIRST.EXE for the run file. The second response means we want to call the loadmap file FIRST.MAP. The third response means that there are no library files.

A LINK Demonstration

Let's link PGMD_1 above:

```
C>LINK A:PGMD_1,A:,A:;

Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights
reserved.

C>
```

Here is the loadmap file:

```
C>TYPE A:PGMD_1.MAP

 start   stop                    Length  Name          Class
 0000011 0001211 0001311 _TEXT      CODE
 00014H 00017H 00004H -DATA         DATA
 00020H  0011FH  00100H  STACK      STACK

 Origin                          Group
 0001:0                          DGROUP

Program  entry  point  at  0000:0000
```

The file gives the relative size and location of the program segments.