Introduction to MASM

(Microsoft assembler)

To Create Source File:

An editor is a program which allows you to create a file containing the assembly language statements for your program. This file is called a **source file.**

Command to create a source file

D:\5K6\> EDIT *filename*. ASM Enter

The next step is to process the source file with an assembler. When you run the assembler, it reads the source file of your program. On the first pass through the source program, the assembler determines the displacement of named data items, the offset labels, etc. and puts this information in a symbol table. On the second pass through the source program the assembler produces the binary code for each instruction and inserts the offsets, etc. that it calculated during first pass.

D:\5K6\> MASM filename.ASM Enter

With this command assembler generates three files.

1. The first file (X) called the object file, is given the extension .OBJ. The object file contains the binary codes for the instructions and information about the addresses of the instructions.
2. The second file (Y) generated by the assembler is called the assembler list file and is given the extension .LST. The list file contains your assembly language statements, the binary codes for each instruction and the offset for each instruction.
3. The third file (Z) generated by this assembler is called the cross-reference file and is given the extension .CRF. The cross-reference file lists all labels and pertinent information required for cross – referencing

NOTE :

The Assembler only finds syntax errors : It will not tell you whether program does what it is supposed to do. To determine whether your program works, you have to run the program and test it.

Next step is to process the object file with linker.

D:\5K6\> LINK filename .OBJ Enter

To debug

D:\5K6\> DEBUG filename.EXE Enter

ASSEMBLY LANGUAGE PROGRAM DEVELOPMENT TOOLS:

EDITOR: An editor is a program, which allows you to create a file containing the assembly language statements for your program.

ASSEMBLER: An assembler program is used to translate the assembly language Mnemonic instructions to the corresponding binary codes. The second file generated by assembler is called the assembler List file.

LINKER: A Linker is a program used to join several object files in to one large object file. The linkers produce link files with the .EXE extension.

DEBUGGER: If your program requires no external hardware, then you can use a debugger to run and debug your program. A debugger is a program, which allows you to load your object code program into system memory, execute the program, and troubleshoot or "debug" it**.**

ASSEMBLER DIRECTIVES:

An assembler is a program used to convert an assembly language program into the equivalent machine code modules. The assembler decides the address of each label and substitutes the values for each of the constants and variables. It then forms the machine code for mnemonics and data in assembly language program.

Assembler directives help the assembler to correctly understand assembly language programs to prepare the codes. Commonly used assembler directives are DB, DD, DW, DUP, ASSUME, BYTE, SEGMENT, MACRO, PROC, OFFSET, NEAR, FAR, EQU, STRUC, PTR, END, ENDM, ENDP etc. Some directives generate and store information in the memory, while others do not.

DB :- Define byte directive stores bytes of data in memory.

BYTE PTR :- This directive indicates the size of data referenced by pointer.

SEGMENT :- This directive is to indicate the start of the segment.

DUP (Duplicate) :- The DUP directive reserves memory locations given by the number preceding it, but stores no specific values in any of these locations.

ASSUME : - The ASSUME statement is only used with full segment definitions. This statement tells the assembler what names have been chosen for the code, data, extra and stack segments.

EQU : - The equate directive equates a numeric ASCII or label to another label.

ORG : - The ORG (origin) statement changes the starting offset address in a segment.

PROC and ENDP : - The PROC and ENDP directives indicate start and end of a procedure (Sub routine). Both the PROC and ENDP directives require a label to indicate the name of the procedure. The PROC directive, must also be followed with the NEAR or FAR. A NEAR procedure is one that resides in the same code segment as the program. A FAR procedure may reside at any location in the memory system.

MACROS

A macro is a group of instructions that performs one task, just as a procedure. The difference is that a procedure is accessed via a CALL instruction, while a macro is inserted in the program at the point of usage as a new sequence of instructions.

MACRO : - The first statement of a macro is the MACRO directive preceded with name of the macro.

ENDM : - The last statement of a macro is the ENDM instruction. Never place a label in front of the ENDM statement.

PUBLIC &EXTRN : - The public and extern directives are very important to modular programming. We use PUBLIC to declare that labels of code, data or entire segments are available to other program modules. We use EXTRN to declare that labels are external to a module. Without this statement, we could not link modules together to create a program using modular programming techniques.

OFFSET : - Offset of a label. When the assembler comes across the OFFSEToperator along with a label, it first computes the 16 – bit displacement of the particular label, and replaces the string 'OFFSET LABEL' by the computed displacement.

LENGTH : - Byte length of the label. This directive is used to refer to the length of data array or a string.

PROGRAMS

1. Write an ALP and execute progam to add two 8-BIT Numbers

```
ASSUME CS:CODE,
DS:DATA

DATA SEGMENT
NUM1 DB 04H
NUM2 DB 02H
RES DB 00H
DATA ENDS

CODE SEGMENT
START:
MOV AX, DATA
MOV DS, AX
MOV AX, 00H

MOV AL, NUM1
MOV BL, NUM2
ADD AL, BL
MOV RES, AL

INT 03H
CODE ENDS
END START
```

```
-u
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A00000          MOV     AL,[0000]
0745:000B 8A1E0100        MOV     BL,[0001]
0745:000F 02C3            ADD     AL,BL
0745:0011 A20200          MOV     [0002],AL
0745:0014 CC              INT     3
```

Result

Registers

```
AX=0006 BX=0002
```

Data Segment

```
-d ds:0
0744:0000  04 02 06
```

Flag Register

```
NV UP EI PL NZ NA PE NC
```

2. Write an ALP and execute progam to add two 16-BIT Numbers

```
ASSUME CS:CODE, DS:DATA

DATA SEGMENT
NUM1 DW 0004H
NUM2 DW 1234H
RES DW 00H
DATA ENDS

CODE SEGMENT
START:
MOV AX, DATA
MOV DS, AX
MOV AX, 00H

MOV AX, NUM1
MOV BX, NUM2
ADD AX, BX
MOV RES, AX

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A10000          MOV     AX,[0000]
0745:000B 8B1E0200        MOV     BX,[0002]
0745:000F 03C3            ADD     AX,BX
0745:0011 A30400          MOV     [0004],AX
0745:0014 CC              INT     3
```

Result

Registers

```
AX=1238 BX=1234
```

Data Segment

```
-D DS:0
0744:0000  04 00 34 12 38 12
```

Flag Register

```
NV UP EI PL NZ NA PO NC
```

3. Write an ALP and execute progam to add two 32-BIT Numbers

```
ASSUME CS:CODE, DS:DATA

DATA SEGMENT
X1 DW 1234H
X2 DW 2345H
Y1 DW 3456H
Y2 DW 4567H
RES1 DW 00H
RES2 DW 00H
DATA ENDS

CODE SEGMENT
START:
MOV AX, DATA
MOV DS, AX
MOV AX,0H

MOV AX, X1
MOV CX, Y1
MOV BX, X2
MOV DX, Y2
ADD AX, CX
ADC BX, DX
MOV RES1, AX
MOV RES2, BX

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A10000          MOV     AX,[0000]
0745:000B 8B0E0400        MOV     CX,[0004]
0745:000F 8B1E0200        MOV     BX,[0002]
0745:0013 8B160600        MOV     DX,[0006]
0745:0017 03C1            ADD     AX,CX
0745:0019 13DA            ADC     BX,DX
0745:001B A30800          MOV     [0008],AX
0745:001E 891E0A00        MOV     [000A],BX
-U
0745:0022 CC              INT     3
```

Result

Registers

```
AX=468A BX=68AC CX=3456 DX=4567
```

Data Segment

```
-D DS:0
0744:0000  34 12 45 23 56 34 67 45-8A 46 AC 68
```

Flag Register

```
NV UP EI PL NZ NA PE NC
```

4. Write an ALP and execute progam to subtract two 8-BIT Numbers

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
NUM1 DB 0F4H
NUM2 DB 0C2H
RES DB 00H
DATA ENDS

CODE SEGMENT
START:
MOV AX,DATA
MOV DS,AX
MOV AX,00

MOV AL,NUM1
MOV BL,NUM2
SUB AL,BL
MOV RES,AL

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A00000          MOV     AL,[0000]
0745:000B 8A1E0100        MOV     BL,[0001]
0745:000F 2AC3            SUB     AL,BL
0745:0011 A20200          MOV     [0002],AL
0745:0014 CC              INT     3
```

Result

Registers

```
AX=0032 BX=00C2
```

Data Segment

```
-D DS:0
0744:0000  F4 C2 32
```

Flag Register

```
NV UP EI PL NZ NA PO NC
```

5. Write an ALP and execute progam to subtract two 16-BIT Numbers

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
NUM1 DW 56ABH
NUM2 DW 10CFH
RES DW 00H
DATA ENDS

CODE SEGMENT
START:
MOV AX,DATA
MOV DS,AX
MOV AX,00

MOV AX,NUM1
MOV BX,NUM2
SUB AX,BX
MOV RES,AX

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A10000          MOV     AX,[0000]
0745:000B 8B1E0200        MOV     BX,[0002]
0745:000F 2BC3            SUB     AX,BX
0745:0011 A30400          MOV     [0004],AX
0745:0014 CC              INT     3
```

Result

Registers

```
AX=45DC BX=10CF
```

Data Segment

```
0744:0000  AB 56 CF 10 DC 45
```

Flag Register

```
NV UP EI PL NZ AC PO NC
```

6. Write an ALP and execute progam to subtract two 32-BIT Numbers

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
X1 DW 12F4H
X2 DW 5B78H
Y1 DW 5043H
Y2 DW 0AB0DH
RES1 DW 00H
RES2 DW 00H
DATA ENDS

CODE SEGMENT
START:
MOV AX,DATA
MOV DS,AX
MOV AX,00H

MOV AX,X1
MOV BX,X2
MOV CX,Y1
MOV DX,Y2
SUB AX,CX
SBB BX,DX

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A10000          MOV     AX,[0000]
0745:000B 8B1E0200        MOV     BX,[0002]
0745:000F 8B0E0400        MOV     CX,[0004]
0745:0013 8B160600        MOV     DX,[0006]
0745:0017 2BC1            SUB     AX,CX
0745:0019 1BDA            SBB     BX,DX
0745:001B A30800          MOV     [0008],AX
0745:001E 891E0A00        MOV     [000A],BX
-U
0745:0022 CC              INT     3
```

Result

Registers

```
AX=C2B1 BX=B06A CX=5043 DX=AB0D
```

Data Segment

```
-D DS:0
0744:0000  F4 12 78 5B 43 50 0D AB-B1 C2 6A B0
```

Flag Register

```
OV UP EI NG NZ AC PE CY
```

7. Write an ALP and execute progam to multiply two 8-BIT Numbers

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
NUM1 DB 34H
NUM2 DB 0CDH
PROD DW 00H
DATA ENDS

CODE SEGMENT
START:
MOV AX,DATA
MOV DS,AX
MOV AX,00H

MOV AL,NUM1
MOV BL,NUM2
MUL BL
MOV PROD,AL

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A00000          MOV     AL,[0000]
0745:000B 8A1E0100        MOV     BL,[0001]
0745:000F F6E3            MUL     BL
0745:0011 A30200          MOV     [0002],AX
0745:0014 CC              INT     3
```

Result

Registers

```
AX=29A4 BX=00CD
```

Data Segment

```
-D DS:0
0744:0000  34 CD A4 29
```

Flag Register

```
OV UP EI PL NZ NA PE CY
```

8. Write an ALP and execute progam to multiply two 16-BIT Numbers

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
NUM1 DW 1234H
NUM2 DW 5678H
PROD1 DW 00H
PROD2  DW 00H
DATA ENDS

CODE SEGMENT
START:
MOV AX,DATA
MOV DS,AX
MOV AX,00H

MOV AX,NUM1
MOV BX,NUM2
MUL BX
MOV RESULT,AX
MOV RESULT1,DX

INT 3H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A10000          MOV     AX,[0000]
0745:000B 8B1E0200        MOV     BX,[0002]
0745:000F F7E3            MUL     BX
0745:0011 A30400          MOV     [0004],AX
0745:0014 89160600        MOV     [0006],DX
0745:0018 CC              INT     3
```

Result

Registers

```
AX=0060 BX=5678 CX=0029 DX=0626
```

Data Segment

```
-D DS:0
0744:0000  34 12 78 56 60 00 26 06
```

Flag Register

```
OV UP EI PL NZ NA PE CY
```

9. Write an ALP and execute progam to divide two 8-BIT Numbers

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
NUM1 DB 47H
NUM2 DB 22H
QUO DB 00H
REM DB 00H
DATA ENDS

CODE SEGMENT
START:
MOV AX,DATA
MOV DS,AX
MOV AX,00H

MOV AL,NUM1
MOV BL,NUM2
DIV BL
MOV QUO,AL
MOV REM,AH

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A00000          MOV     AL,[0000]
0745:000B 8A1E0100        MOV     BL,[0001]
0745:000F F6F3            DIV     BL
0745:0011 A20200          MOV     [0002],AL
0745:0014 88260300        MOV     [0003],AH
0745:0018 CC              INT     3
```

Result

Registers

```
AX=0302 BX=0022
```

Data Segment

```
-D DS:0
0744:0000  47 22 02 03
```

Flag Register

```
NV UP EI PL ZR NA PE NC
```

10. Write an ALP and execute progam to divide two 16-BIT Numbers

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
NUM1 DW 0ABC4H
NUM2 DW 1232H
QUO  DW 0H
REM  DW 0H
DATA ENDS

CODE SEGMENT
START:
MOV AX,DATA
MOV DS,AX
MOV AX, 0H

MOV DX,00
MOV AX,NUM1
MOV BX,NUM2
DIV BX
MOV QUO,AX
MOV REM,DX

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A10000          MOV     AX,[0000]
0745:000B 8B1E0200        MOV     BX,[0002]
0745:000F F7F3            DIV     BX
0745:0011 A30400          MOV     [0004],AX
0745:0014 89160600        MOV     [0006],DX
0745:0018 CC              INT     3
```

Result

Registers

```
AX=0009 BX=1232 DX=0802
```

Data Segment

```
-D DS:0
0744:0000  C4 AB 32 12 09 00 02 08
```

Flag Register

```
NV UP EI PL ZR NA PE NC
```

11. Write an ALP and execute progam to find the square of a number

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
NUM1 DB 04H
RES DW 00H
DATA ENDS

CODE SEGMENT
START:
 MOV AX,DATA
MOV DS,AX
MOV AX,00H

MOV AL, NUM1
MOV BL, NUM1
MUL BL
MOV RES,AX

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A00000          MOV     AL,[0000]
0745:000B 8A1E0000        MOV     BL,[0000]
0745:000F F6E3            MUL     BL
0745:0011 A30100          MOV     [0001],AX
0745:0014 CC              INT     3
```

Result

Registers

```
AX=0010 BX=0004
```

Data Segment

```
-D DS:0
0744:0000  04 10
```

Flag Register

```
NV UP EI PL NZ NA PE NC
```

12. Write an ALP and execute progam to find the Cube of a number

```
ASSUME
CS:CODE,DS:DATA

DATA SEGMENT
NUM1 DB 04H
RES DW 00H
DATA ENDS

CODE SEGMENT
START:
 MOV AX,DATA
MOV DS,AX
MOV AX,00H

MOV AL, NUM1
MOV BL, NUM1
MUL BL
MUL BL
MOV RES,AX

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A00000          MOV     AL,[0000]
0745:000B 8A1E0000        MOV     BL,[0000]
0745:000F F6E3            MUL     BL
0745:0011 F6E3            MUL     BL
0745:0013 A30100          MOV     [0001],AX
0745:0016 CC              INT     3
```

Result

Registers

```
AX=0040 BX=0004
```

Data Segment

```
-D DS:0
0744:0000  04 40 00
```

Flag Register

```
NV UP EI PL NZ NA PE NC
```

13. Write an ALP and execute progam to exchange two numbers

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
NUM1 DB 04H
NUM2 DB 0A2H
DATA ENDS

CODE SEGMENT
START:
 MOV AX,DATA
MOV DS,AX
MOV AX,00H

MOV AL, NUM1
MOV BL, NUM1
XHCG AX,BX

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80000          MOV     AX,0000
0745:0008 A00000          MOV     AL,[0000]
0745:000B 8A1E0100        MOV     BL,[0001]
0745:000F 93              XCHG    AX,BX
0745:0010 CC              INT     3
```

Result

Registers

```
AX=00A2 BX=0004
```

Data Segment

```
-D DS:0
0744:0000  04 A2
```

Flag Register

```
NV UP EI PL ZR NA PE NC
```

14. Write an ALP and execute progam to find the factorial of a number

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
NUM1 DB 04H
RES DW 00H
DATA ENDS

CODE SEGMENT
START:
 MOV AX,DATA
MOV DS,AX
MOV AX,0001H

MOV BL,NUM1

GO:
MUL BL
DEC BL
JNZ GO
MOV RES,AX

INT 03H
CODE ENDS
END START
```

```
-U
0745:0000 B84407          MOV     AX,0744
0745:0003 8ED8            MOV     DS,AX
0745:0005 B80100          MOV     AX,0001
0745:0008 8A1E0000        MOV     BL,[0000]
0745:000C F6E3            MUL     BL
0745:000E FECB            DEC     BL
0745:0010 75FA            JNZ     000C
0745:0012 A30100          MOV     [0001],AX
0745:0015 CC              INT     3
```

Result

Registers

```
AX=0018 BX=0000
```

Data Segment

```
-D DS:0
0744:0000  04 18 00
```

Flag Register

```
NV UP EI PL ZR NA PE NC
```