Section #5

What are the difference between DEBUG and MASM?

DEBUG	MASM (Microsoft Macro Assembler)
DOS program used to assemble assembly code into machine code.	DOS program used to assemble assembly code into machine code.
Instructions must contain addresses.	Write Instructions directly without addresses.
Assume all numbers are hexadecimal.	Assume all numbers are decimal.
Hexa numbers shouldn't contain h and not begging with numbers. MOV ah, 3A >> true MOV ax, AE h >> true MOV ax, AE >> true	Hexa numbers must contain h and begging with numbers. MOV ah, 3A h >> true MOV ax, AE h >> false MOV ax, 0AE h >> true
DEBUGER work as interpreter and compiler.	Assembler work as a compiler.

Note: *Assembler convert all numbers into hex finally before execution.*

What are the main steps used for creating an assembly program using MASM?

1- Use an editor to write your assembly program (e.g. print.asm)

2- Assemble your asm program using MASM (MASM print.asm) >> print.obj

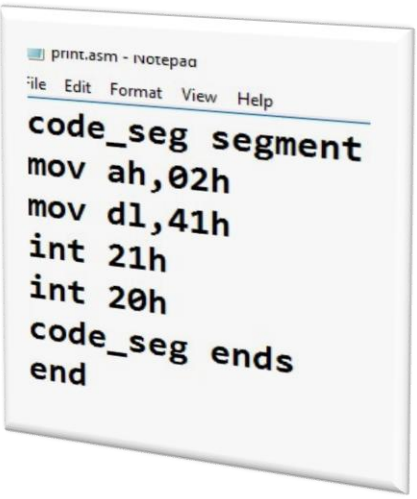
3- Linking your file using linker (link print.obj) >> print.exe

4- If you want to create .com program use EXE2BIN program (EXE2BIN print.exe) >> Print .com

Define each of the following:

- **Object file:** Intermediate file which contains our machine language program Generated by MASM program.
- **Linker:** A DOS program which introduced an exe file from object file.
- **EXE2BIN:** A DOS program used to convert exe file into .com file.

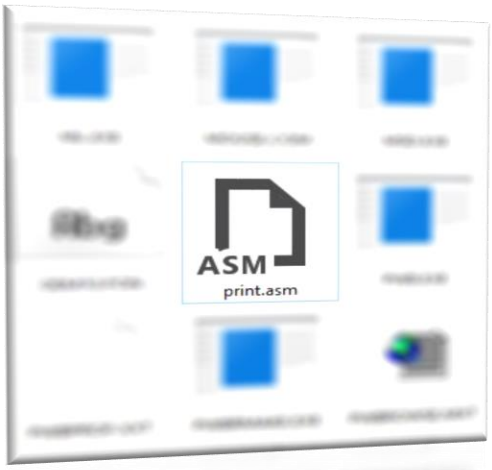
Example for creating an assembly program using MASM?



Save as

Print.ASM

>>>>>>>

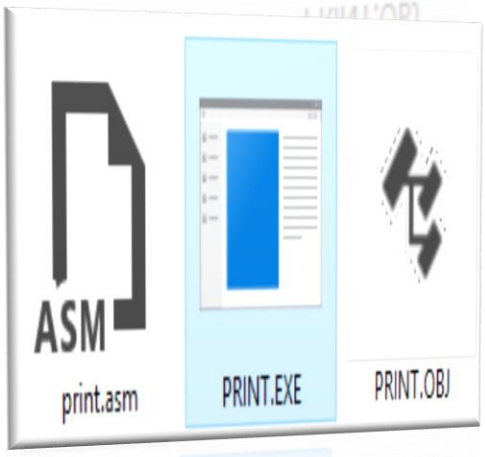


MASM print.asm

>>>>>>>

Link print.obj

>>>>>>>



If you want to create .com program use EXE2BIN (e.g. EXE2BIN print.exe print.com).

Describe the syntax of assembly instruction?

Assembly language instructions Consists of four fields:

[Label] mnemonic [operands] [;comment]

- Label: allows the program to refer to a line of code by name.
  (can't exceed 31 character, and followed by " **:** ")
- Mnemonic (instruction) & operands: perform the real work of the program.
  (e.g. MOV ax, 6421) [MOV >> mnemonic] [ax, 6421 >> operands]
- Comments: are optional and begins with " **;** "

What are Pseudo-Instructions (Directives)?

- Used by assembler to organize the program.
- Gives directions to the assembler about how to translate the Assembly language instructions into machine code.
- Do not generate any machine code.
- Examples: DB, END, ENDP, .MODEL, .CODE

What are .model Directive?

- Used to select size of memory model.

.MODEL SMALL	Uses 64 KB for code, and another 64 KB for data.
.MODEL MEDIUM	Data must fit into 64 KB, but code can exceed 64 KB.
.MODEL COMPACT	Data can exceed 64 KB, but code must fit into 64 KB.
.MODEL HUGE	Both data and code can exceed 64 KB
.MODEL TINY	Data and code must fit into 64 KB, used with .com files.

Write short notes about each directive?

► **.Stack:** Marks the beginning of stack segment**.**
- » Define storage for the stack**.**
- » Corresponding to "SS" Stack segment.
- » **Ex: .stack 64** >> Reserves 64 bytes of memory for the stack.

► **.Data:** marks the beginning of data segment.
- **Ex:**

```
.data
Data1 DB 52H
Data2 DB 29H
```

- » **DB:** (Define byte) 1 byte.
- » **DW:** (Define Word) 2 bytes.
- » **DD:** (Define Double Word) 4 bytes.
- » **DQ**: (Define Quad Word) 8 bytes.
- » **DT**: (Define Ten Bytes) 10 bytes. (BCD)
- » **ORG:** (Origin) Used to indicate the beginning of offset.
- » **DUP:** (Duplicate) Used to duplicate a given number of characters.
- » **EQU:** (Equate): define a constant value without occupying in memory.

► **.Code:** marks the beginning of code segment.
- » **PROC directive:** is a group of instructions designed to accomplish a specific function.
- » A code segment may consists of only one procedure or several procedures.
- » Every procedure must contain a name and end with ENDP directive.
- » The PROC and ENDP statements must contain the same label.
- » The PROC directive may have FAR or NEAR.
- » **FAR**: Pass control between procedures in different segment.
  - (CS, IP) CS changes IP changes.
- » **NEAR**: pass control between procedures in the same code segment.
  - (CS, IP) CS not changes, IP changes.

Write MASM program used to sum of two numbers.

```
.MODEL SMALL
.STACK 64

.DATA
        DATA1 DB 52H
        DATA2 DB 29H
        SUM DB ?


.CODE
MAIN PROC FAR
        MOV AX, @DATA
        MOV DS, AX
        MOV AL, DATA1
        MOV BL, DATA2
        ADD AL, BL
        MOV SUM, AL
        MOV DL, SUM
        MOV AH, 02H
        INT 21 H
        MOV AH, 4CH
        INT 21H

MAIN ENDP
END
```

Exercises

Q1: What is the purpose of the pseudo-Instructions?

- Gives directions to the assembler about how to translate the Assembly language instructions into machine code.
- Used by assembler to organize the program.

Q2: Write Assembly language program with the following characteristics:

- ❖ Data item named HIGH_DAT, which contains 95.
- ❖ Instructions to move HIGH_DAT to AH, BH and DL.
- ❖ A program entry point named START.

```
.MODEL SMALL
.STACK
.DATA
        HIGH_DAT DB 95h
.CODE
        START PROC FAR
        MOV AX, @DATA
        MOV DS, AX
        MOV AH, HIGH_DAT
        MOV BH, HIGH_DAT
        MOV DL, HIGH_DAT
START ENDP
END
```

Q3: Find the errors in the following?

```
.MODEL ENORMOUS
.STACK
.CODE
.DATA
MAIN PROC FAR
        MOV AX, DATA
        MOV DS, @DATA
        MOV AL, 34H
        MOV AL, 4FH
        MOV DATA1, AL

START ENDP
END
```

The correct >>

```
.MODEL SMALL
.STACK
.DATA
DATA1 DB ?
.CODE
MAIN PROC FAR
        MOV AX, @DATA
        MOV DS, AX
        MOV AL, 34H
        MOV AL, 4FH
        MOV DATA1, AL
MAIN ENDP
END
```

Q4: The DB Directive is always used for ASCII strings longer than 2 bytes.

Q5: write two method to filling six memory locations with FF h?

```
.data
        Data1 DB 0FFh, 0FFh, 0FFh, 0FFh, 0FFh, 0FFh
        Or
        DATA2 DB 6DUP (0FFh)
```

Q6: How many bytes are defined by the following?

DATA_1 DB 6DUP (4 DUP (0FF H))

6 * 4 = 24 byte

Q7: Do the following data segment definitions results in the same storage in bytes at offset 10 H and 11H? If not, explain why?

ORG 10H	**ORG 10H**
DATA_1 DB 72H	**DATA_1 DW 7204H**
DATA_2 DB 04H	

NO because:

0010 72H 0011 04H	0010 04H 0011 72H

Q8: The DD directive is used to allocate memory locations that are *4* bytes in length.

Q9: The DQ directive is used to allocate memory locations that are *8* bytes in length.

Q10: state briefly the purpose of the ORG directive.

Used to indicate the beginning of offset where we will start storing data.

Q11: what are the advantage in using the EQU directive to define a constant value?

Define a constant value without storing in memory.

Can be used in Code Segment.

Q12: How many bytes are set aside by each of the following directives?

(a) ASC_DATA DB '1234' >> 4 Bytes.
(b) HEX_DATA DW 1234H >> 2 Bytes.

Q13: Using assembler to write a program used for printing ASCII.

```
.model small
.stack 100h
.data
.code
Main    Proc
        mov ax, @data
        mov ds, ax
        mov ah, 2
        mov cx, 256
        mov dl, 0
Print:
        int 21h
        inc dl
        dec cx
        jnz Print
        mov ah, 4ch
        int 21h
Main   EndP
End
```