

# X86 Memory Management

## Contents of Lecture:

- ❖ General Structure of 8086 Assembly Language Program
- ❖ Assembling, Linking, and Running Programs
- ❖ Running Hello Program on Tasm assembler

## References for Lecture:

*KIP R. IRVINE, Assembly Language for x86 Processors, 7<sup>th</sup> Edition, Chapter 3:  
Assembly Language Fundamentals*

## General Structure of 8086 Assembly Language Program

- ❖ Every programming language has its own programs' structure.
- ❖ In this Assembly Language Programming, A single program is divided into four Segments which are:
  - ✓ Data Segment
  - ✓ Code Segment
  - ✓ Stack Segment
  - ✓ Extra Segment
- ❖ The following simple Assembly program will explain the main structure that you need to follow for all programs.

```
.386                ; 32-bit registers and addresses
.MODEL             ; allow memory models
.STACK             ; stack-size
.DATA              ; start of the data segment
                  ; your variables/constants are defined here
.CODE              ; start of the code segment
START:             ; Indicates the beginning of instructions
    mov ax, @data
    mov ds, ax

                  ; your codes come here

    mov Ah, 4Ch    ; Return to DOS
    int 21h
END START          ;End of instruction
END
```

- ❖ The above code contains the simplified segment **directives** .386, .MODEL, .STACK, .DATA, and .CODE, as well as the END directive.
- ❖ These kinds of statements are called directive statements; they are required to guide the assembler on various aspects of the assembly process, and they are not translated to machine language.
- ❖ Directive tells the assembler necessary information to be considered while assembling and translating the code to machine language.

### .386

- ❖ The .386 directive, which identifies this as a 32-bit program that can access 32-bit registers and addresses.

### .MODEL

- ❖ The .MODEL directive specifies the memory model for an assembler module that uses the simplified segment directives.
- ❖ The .MODEL directive must precede .CODE, .DATA, and .STACK.
- ❖ The format of the .MODEL directive is:

#### **.MODEL memory-model**

- ❖ The memory-model can be SMALL, COMPACT, MEDIUM, LARGE or HUGE.

| Memory-model | Number of code segments | Number of data segments | Number of stack segments |
|--------------|-------------------------|-------------------------|--------------------------|
| SMALL        | <i>One segment</i>      | <i>One segment</i>      | <i>One segment</i>       |
| COMPACT      | More than one segment   | <i>One segment</i>      | <i>One segment</i>       |
| MEDIUM       | <i>One segment</i>      | More than one segment   | <i>One segment</i>       |
| LARGE        | More than one segment   | More than one segment   | <i>One segment</i>       |
| HUGE         | More than one segment   | More than one segment   | More than one segment    |

### .STACK

- ❖ Defines the stack segment and controls the size of the stack.
- ❖ For example:

.STACK 200h

Defines a stack that is 200h bytes long (512 bytes).

- ❖ That's really all you have to do so as far as the stack is concerned; just make sure you've got a .STACK directive in your program, and MASM handles the stack for you.

**.DATA**

- ❖ Marks the start of your data segment.
- ❖ You should place your memory variables and constants in this segment.

**.CODE**

- ❖ Marks the start of your program's code segment.
- ❖ This directive tells TASM exactly which code segment to place your instructions in.
- ❖ **START** is the label used to show the starting point of the code which is written in the Code Segment.

❖ **Statements:**

```
mov ax, @data
mov ds, ax
```

- ✓ After Assuming DATA and CODE Segment, Still it is compulsory to initialize Data Segment to DS register.
- ✓ MOV is a keyword to move the second element into the first element. But we cannot move DATA Directly to DS due to MOV commands restriction, Hence we move DATA to AX and then from AX to DS. *AX is the first and most important register in the ALU unit.*
- ✓ This part is also called INITIALIZATION OF DATA SEGMENT and It is important so that the Data elements or variables in the DATA Segment are made access able.
- ✓ **Other Segments are not needed to be initialized.**

❖ **Statements:**

```
mov ah,4ch
int 21h
```

- ✓ The above two line code is used to exit to dos or exit to operating system (used to terminate the execution of the program).
- ✓ Standard Input and Standard Output related Interrupts are found in INT 21H which is also called as DOS interrupt. It works with the value of AH register, If the Value is 4ch, That means Return to Operating System or DOS which is the End of the program.
- ❖ **END START** is the end of the label used to show the ending point of the code which is written in the Code Segment.

**END**

- ❖ It is a directive to indicate the end of file.

## **Assembling, Linking, and Running Programs**

- ❖ A source program written in assembly language cannot be executed directly on its target computer. It must be translated, or assembled into executable code. In fact, an assembler is very similar to a compiler.
- ❖ The assembler produces a file containing machine language called an object file. This file isn't quite ready to execute. It must be passed to another program called a linker, which in turn produces an executable file. This file is ready to execute from the operating system's command prompt
- ❖ Software tools used for editing, assembling, linking, and debugging assembly language programming are an assembler, a linker, a debugger, and an editor

### **Editor**

- ❖ You need a text editor to create assembly language source files.

### **Assembler**

- ❖ An assembler is a program that converts source-code programs written in assembly language into object files in machine language.
- ❖ Popular assemblers have emerged over the years for the Intel family of processors.
- ❖ These include:
  - ✓ MASM (Macro Assembler from Microsoft)
  - ✓ TASM (Turbo Assembler from Borland)
  - ✓ NASM (Netwide Assembler for both Windows and Linux)
  - ✓ GNU assembler distributed by the free software foundation.

### **Linker**

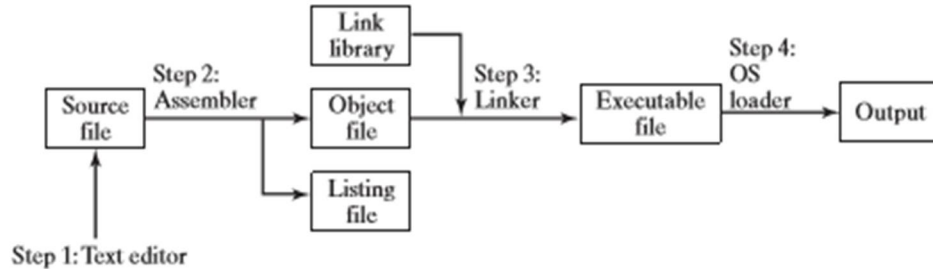
- ❖ A linker is a program that combines your program's object file created by the assembler with other object files and link libraries, and produces a single executable program.
- ❖ You need a linker utility to produce executable files.
  - ✓ Link.exe creates an .exe file from an .obj file.

### **Debugger**

- ❖ A debugger is a program that allows you to trace the execution of a program and examine the content of registers and memory.

### Assemble-Link-Execute cycle:

- ❖ The process of **editing**, **assembling**, **linking**, and **executing** assembly language programs is summarized in following Figure:



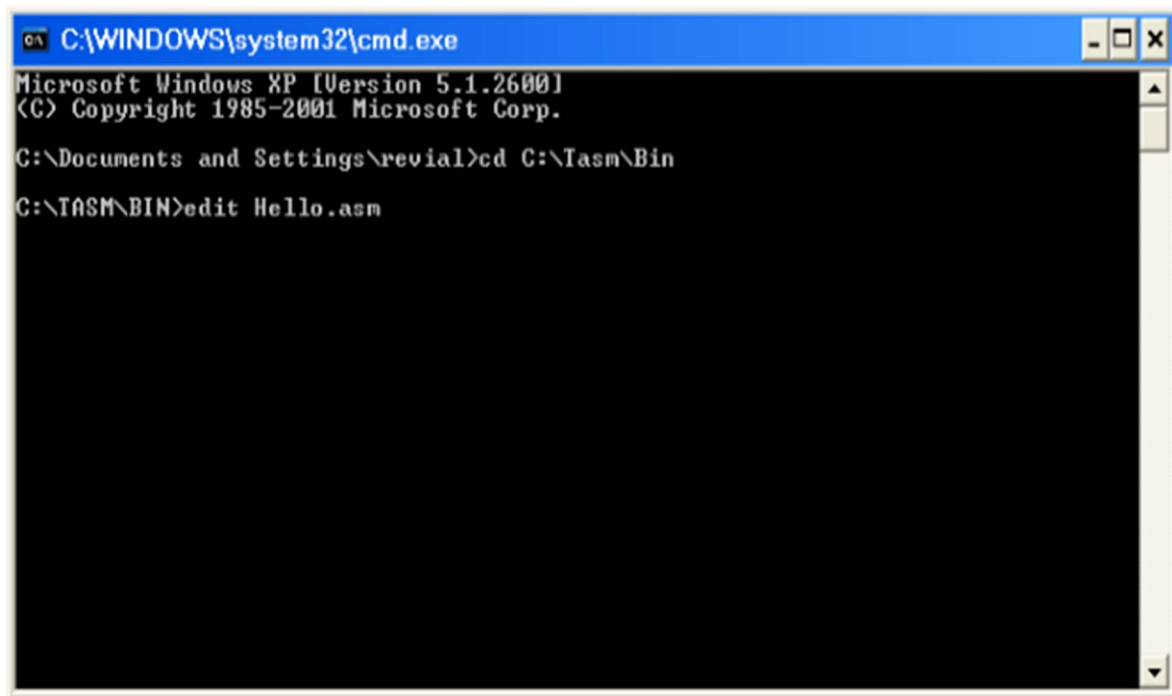
- ❖ Following is a detailed description of each step:
- ✓ **Step 1:** A programmer uses a text editor to create an ASCII text file named the source file.
  - ✓ **Step 2:** The assembler reads the source file and produces an object file, a machine-language translation of the program. Optionally, it produces a listing file. If any errors occur, the programmer must return to Step 1 and fix the program.
  - ✓ **Step 3:** The linker reads the object file and checks to see if the program contains any calls to procedures in a link library. The linker copies any required procedures from the link library, combines them with the object file, and produces the executable file.
  - ✓ **Step 4:** The operating system loader utility reads the executable file into memory and branches the CPU to the program's starting address, and the program begins to execute.

### Running Hello Program on Tasm assembler

Two ways to running assembly programs:

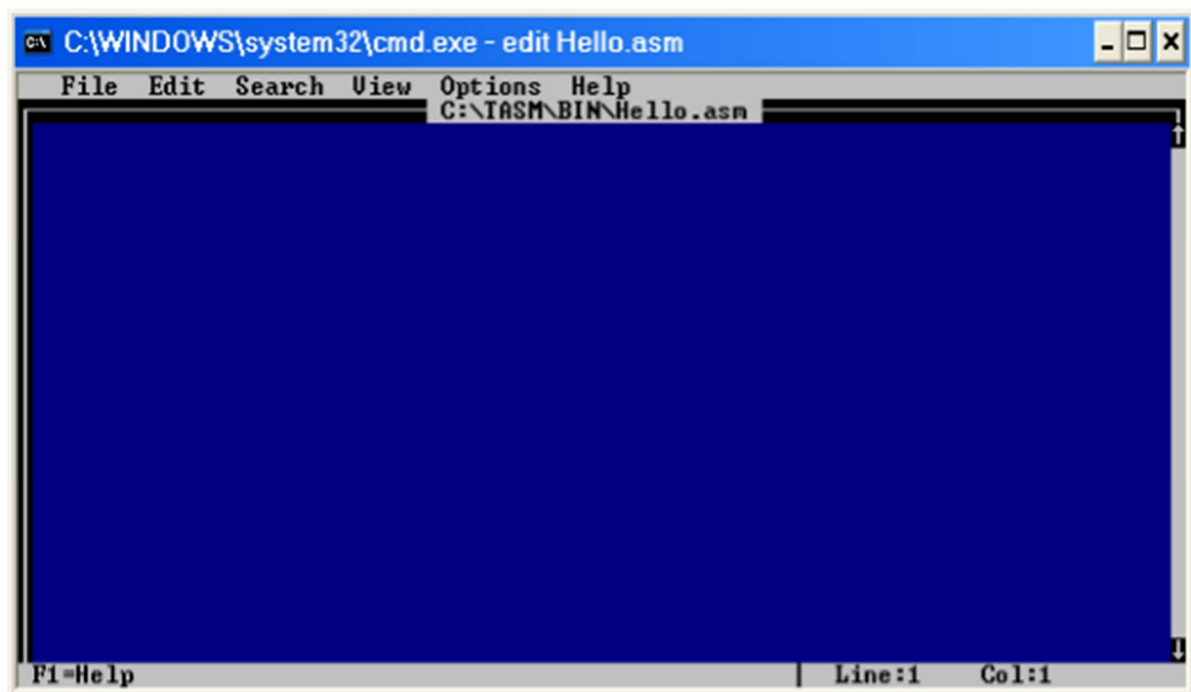
#### First:

1. Click Start (All) Programs Run then write cmd and click OK.
2. Go to directory C:\Tasm\Bin
3. Type the command C:\Tasm\Bin\edit Hello.asm
4. The next screen will open.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\revial>cd C:\Tasm\Bin
C:\TASM\BIN>edit Hello.asm
```



```
C:\WINDOWS\system32\cmd.exe - edit Hello.asm
File Edit Search View Options Help
C:\TASM\BIN\Hello.asm
F1=Help | Line:1 Col:1
```

5. Write the following Hello program

```
.model small
.stack
.data
    msg db "Hello$"
.code
start:
    mov ax,@data
    mov ds,ax

    mov ah,9h
    mov dx,offset msg
    int 21h

    mov ah,4ch
    int 21h
end start
end
```

6. Write C:\Tasm\Bin\tasm hello.asm to create the file hello.obj This file is the machine language for the program.
7. Write C:\Tasm\Bin\tlink hello.obj to create the file hello.exe This file is executable program.
8. Finally, write C:\Tasm\Bin \hello.exe You will show the message hello, world on DOS screen

```
C:\WINDOWS\system32\cmd.exe - tlink Hello
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\revial>cd C:\Tasm\Bin
C:\TASM\BIN>edit Hello.asm
C:\TASM\BIN>tasm Hello
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

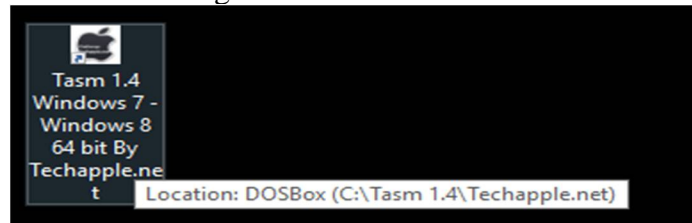
Assembling file: Hello.ASM
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 452k

C:\TASM\BIN>tlink Hello
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International
Warning: DOSSEG directive ignored in module HELLO.ASM
Warning: No stack

C:\TASM\BIN>Hello
hello
C:\TASM\BIN>
```

**Second:**

1. Open the Tasm assembler Program



2. The next screen will open.

```

+=====+
Power Programming Tools 64 Bit By Chaitanya Patel : Techapple.Net
+=====+

Type Proper Command To Perform the Desired Action

Command      .      Action
-----
Edit          -      Open MS-DOS Editor
TASM          -      Compilation
tlink         -      Perform Linking
td            -      Launch Turbo Debugger
Exit          -      Exit Tasm 1.2

For Compiling your files tasm "yourfilename".asm use without quotes
e.g for compiling add.asm command is : tasm add.asm
For Linking and debugging same as 32 bit : tlink,td

Complink,DPMIload and TasmX also available using 32bit commands

C:\TASM>

```

3. Type the command C:\Tasm\edit Hello.asm then press Enter

```

+=====+
Power Programming Tools 64 Bit By Chaitanya Patel : Techapple.Net
+=====+

Type Proper Command To Perform the Desired Action

Command      .      Action
-----
Edit          -      Open MS-DOS Editor
TASM          -      Compilation
tlink         -      Perform Linking
td            -      Launch Turbo Debugger
Exit          -      Exit Tasm 1.2

For Compiling your files tasm "yourfilename".asm use without quotes
e.g for compiling add.asm command is : tasm add.asm
For Linking and debugging same as 32 bit : tlink,td

Complink,DPMIload and TasmX also available using 32bit commands

C:\TASM>edit hallo.asm

```

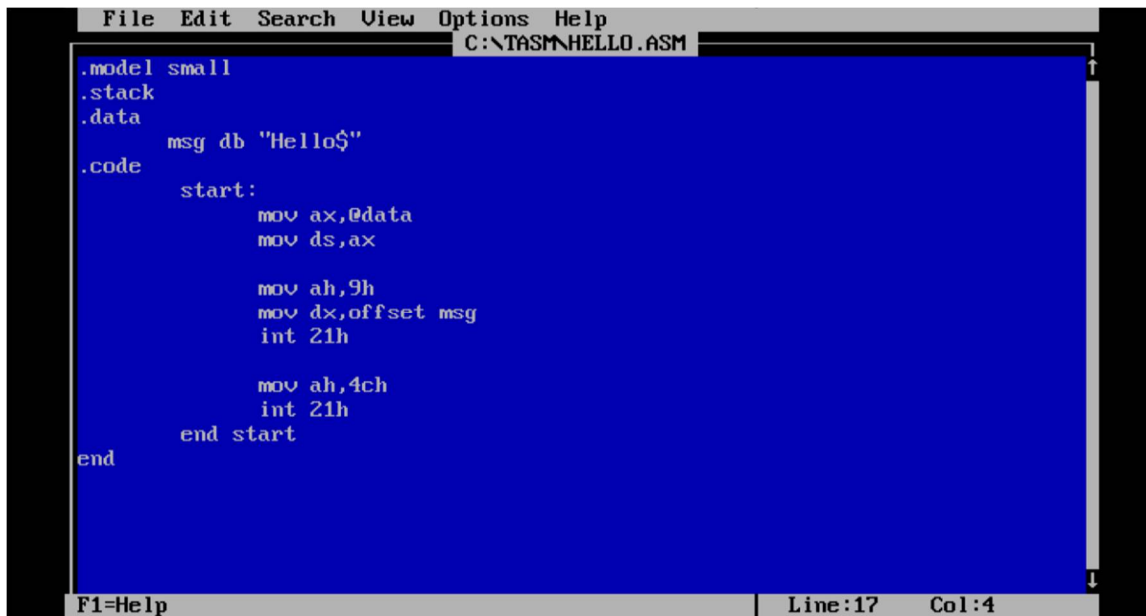


4. The editor will open. Write the following Hello program

```
.model small
.stack
.data
    msg db "Hello$"
.code
    start:
        mov ax,@data
        mov ds,ax

        mov ah,9h
        mov dx,offset msg
        int 21h

        mov ah,4ch
        int 21h
    end start
end
```



```
File Edit Search View Options Help
C:\TASM\HELLO.ASM

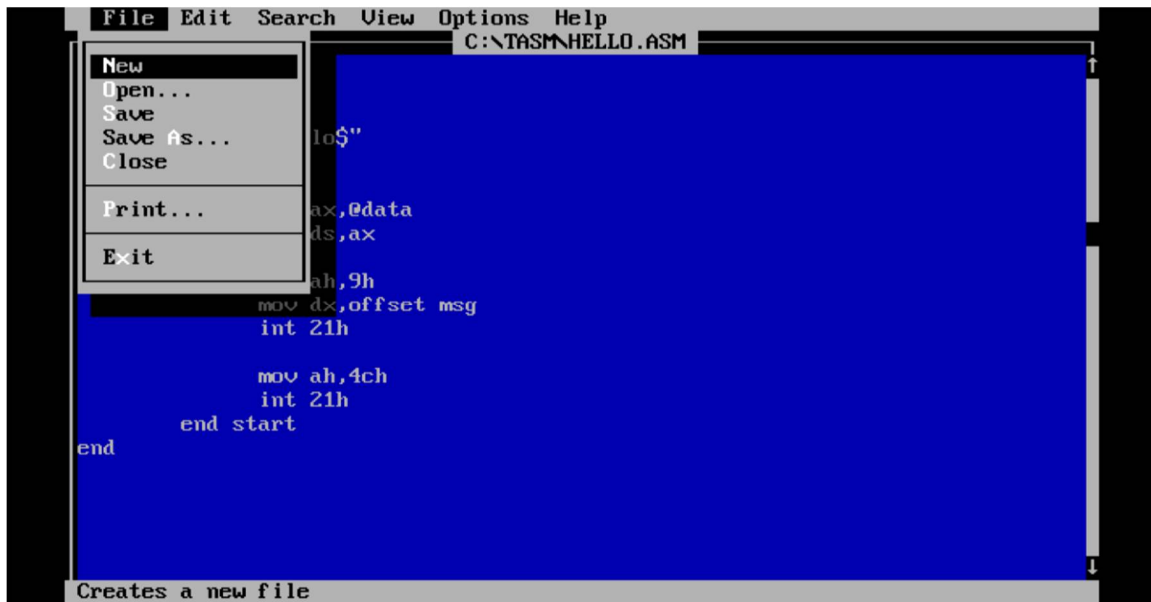
.model small
.stack
.data
    msg db "Hello$"
.code
    start:
        mov ax,@data
        mov ds,ax

        mov ah,9h
        mov dx,offset msg
        int 21h

        mov ah,4ch
        int 21h
    end start
end

F1=Help | Line:17 Col:4
```

5. Save and exit the hello program from File menu.



6. Write `C:\Tasm>tasm hello.asm` to create the file `hello.obj` This file is the machine language for the program.
7. Write `C:\Tasm>tlink hello.obj` to create the file `hello.exe` This file is executable program.
8. Finally, write `C:\Tasm>hello.exe` You will show the message **hello** on DOS screen

