

Assembly Language Lab # 1

Introduction

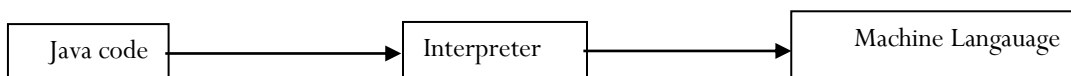
Introduction to Assembly Language

Objective:

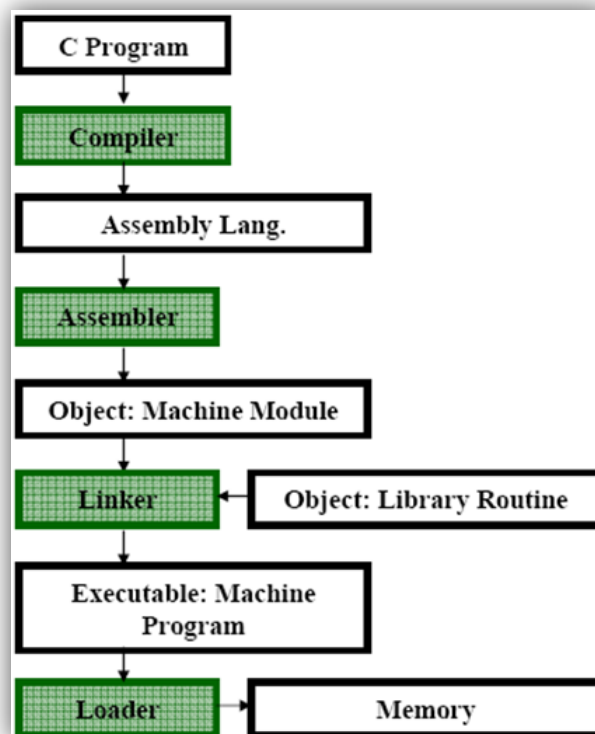
To be familiar with Assembly Language,

Introduction:

- ❖ Machine language can be made directly from java code using interpreter .



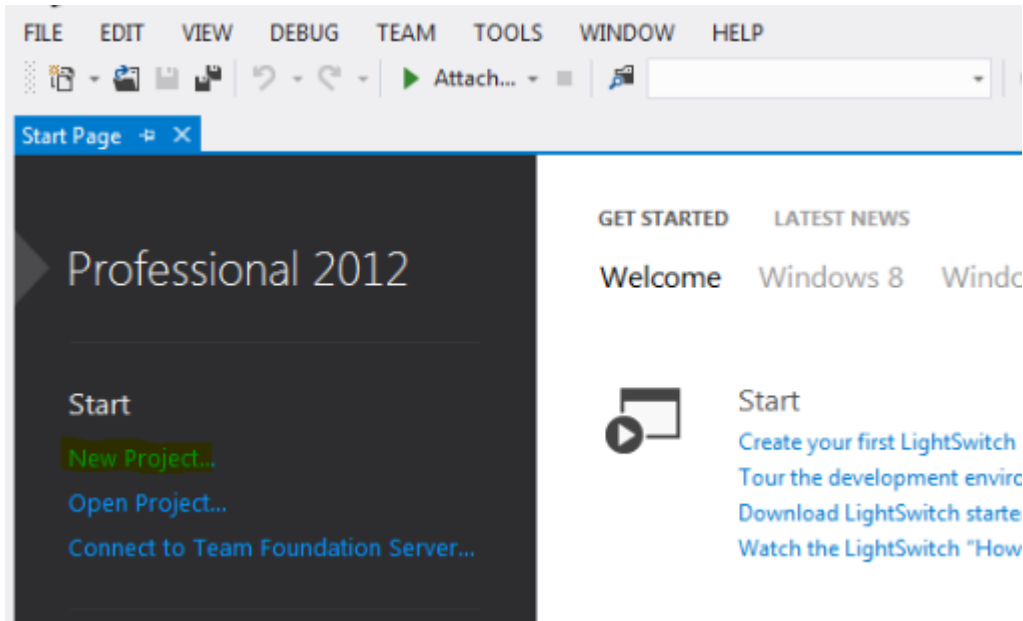
- ❖ C ,C++ code is executed faster than Java code ,because they transferred to assembly language before machine language .



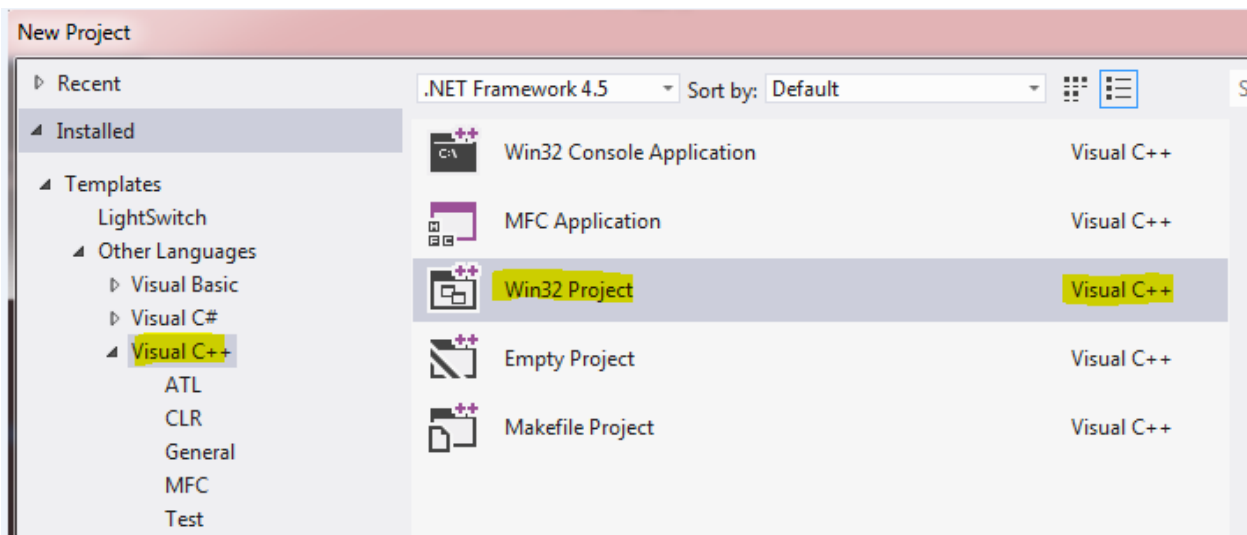
❖ Visual Studio 2012

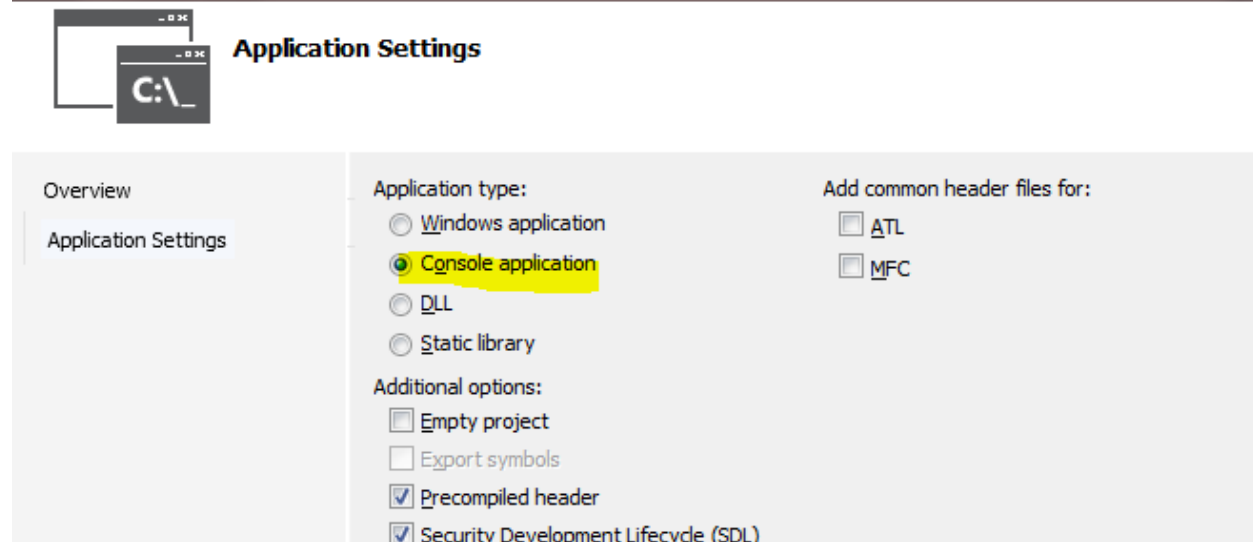
To convert C++ program to assembly language.

1. From **File** menu >> choose **new** >> then choose **project**.
Or from the **start** page choose **new project**.



2. Then the new project window will appear,
 - choose visual C++ and win32 console application
 - The project name is welcome:





3. Write C++ Program that print "Welcome all to our assembly Lab \n"

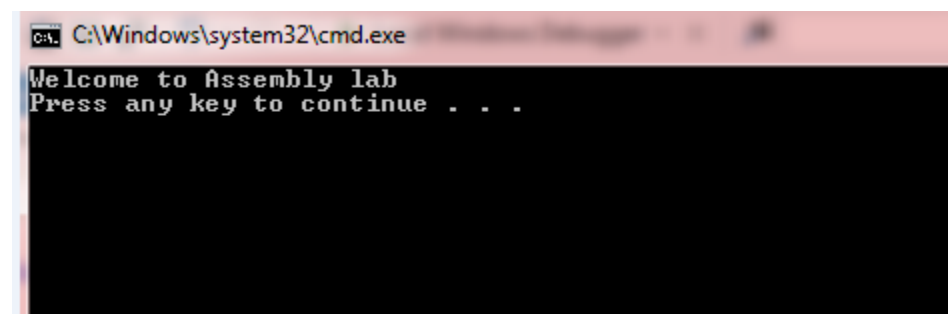
```
(Global Scope)
// Welcome_lab.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{
    printf("Welcome to Assembly lab \n");
    return 0;
}
```

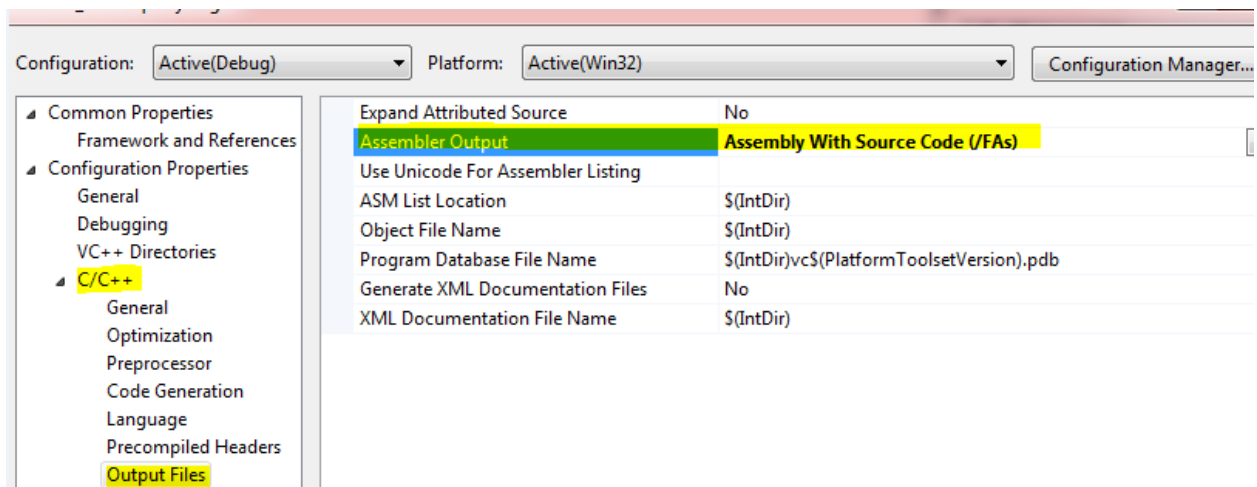
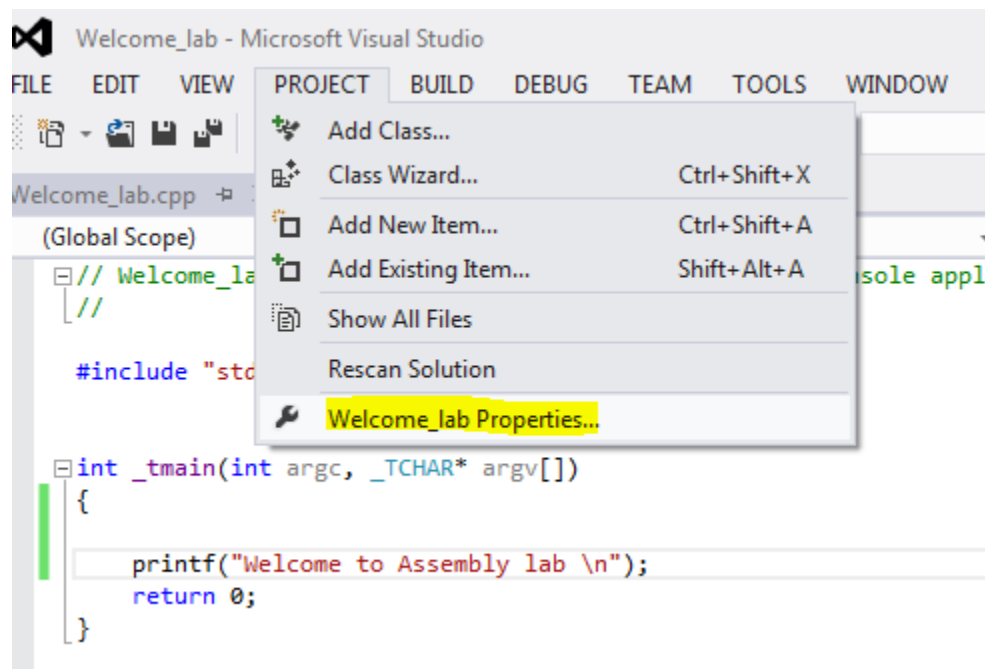
4. To run the project, do the following two steps in order:

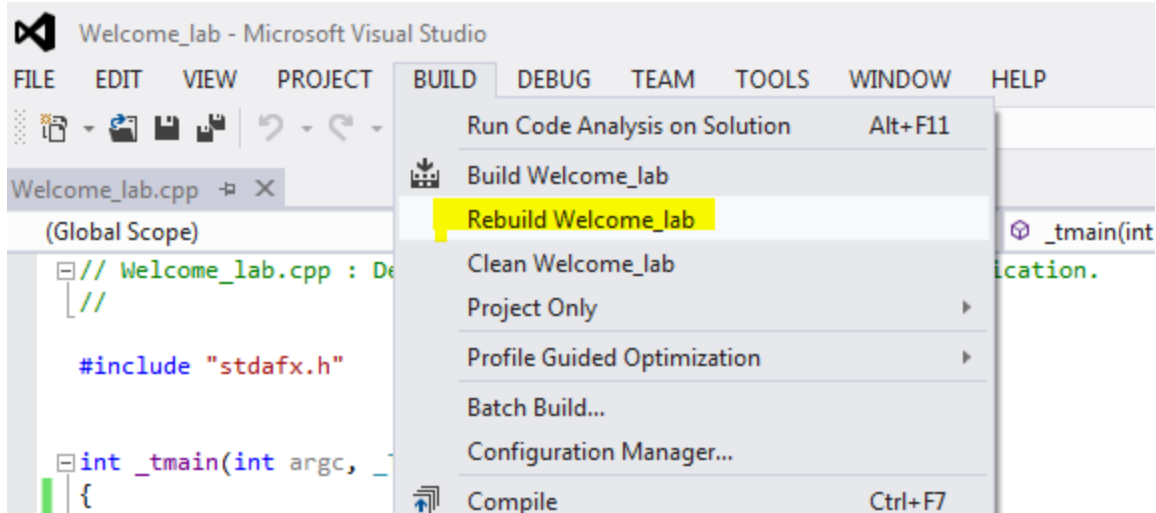
- From **build** menu choose **build Welcome**.
- From **debug** menu choose **start without debugging**.

The output is

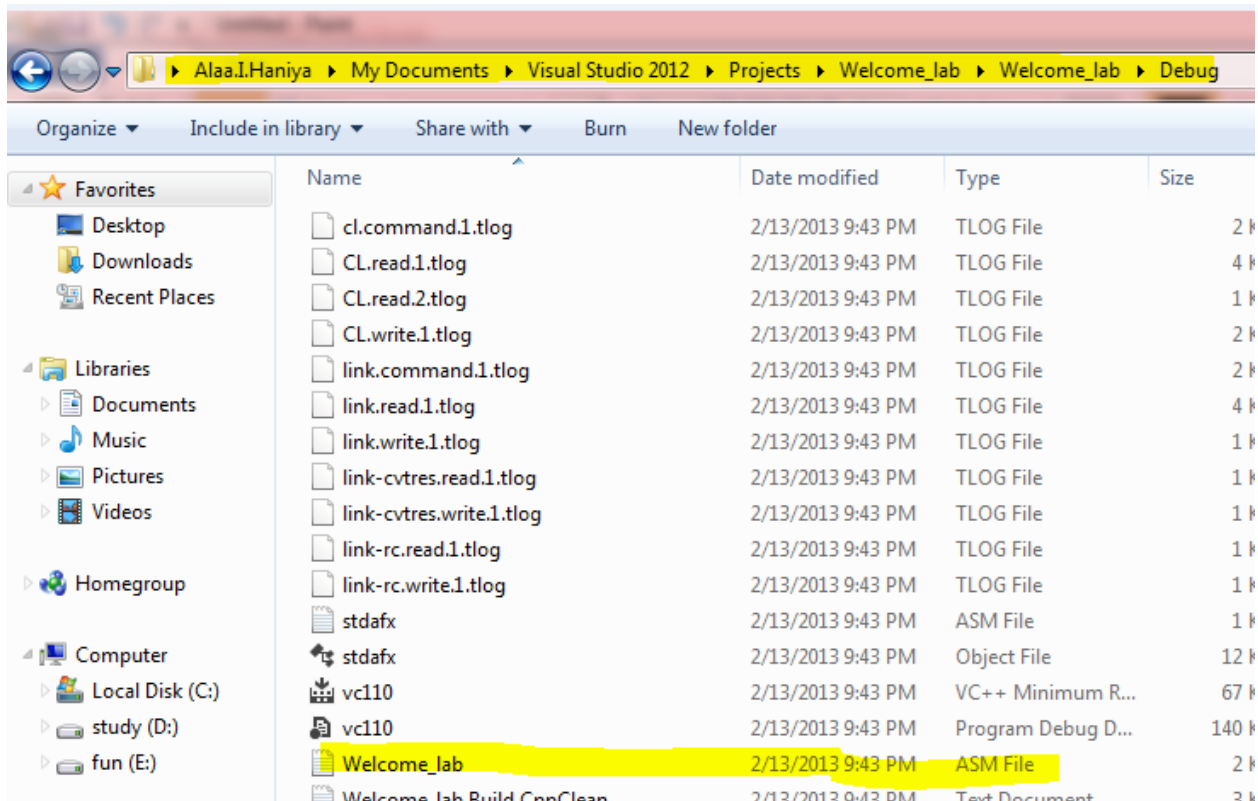


5. To convert C++ code to Assembly code we follow these steps:





We will find the Assembly code on the project folder we save in (*Visual Studio 2012\Projects\Welcome\Welcome\Debug*), named as **Welcome.asm**



Part of the code:

```
52 ; 9 :  
53 ; 10 :    printf("Welcome to Assembly lab \n");  
54  
55     mov esi, esp  
56     push OFFSET ??_C@_OBK@OEAIJFJJ@Welcome?5to?5Assembly?5lab?5?6?$AA@  
57     call DWORD PTR __imp__printf  
58     add esp, 4  
59     cmp esi, esp  
60     call __RTC_CheckEsp  
61  
62 ; 11 :    return 0;  
63
```

❖ *Assembly Language:*

Assembly Language is a programming language that is very similar to machine language, but Uses symbols instead of binary numbers. It is converted by the assembler (e.g. Tasm and Masm) Into executable machine-language programs

❖ *Assembly Language Tools:*

Software tools are used for editing, assembling, linking, and debugging assembly language programming. You will need an assembler, a linker, a debugger, and an editor.

1. *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), and GNU assembler distributed by the free software foundation.

2. *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.

3. *Debugger*

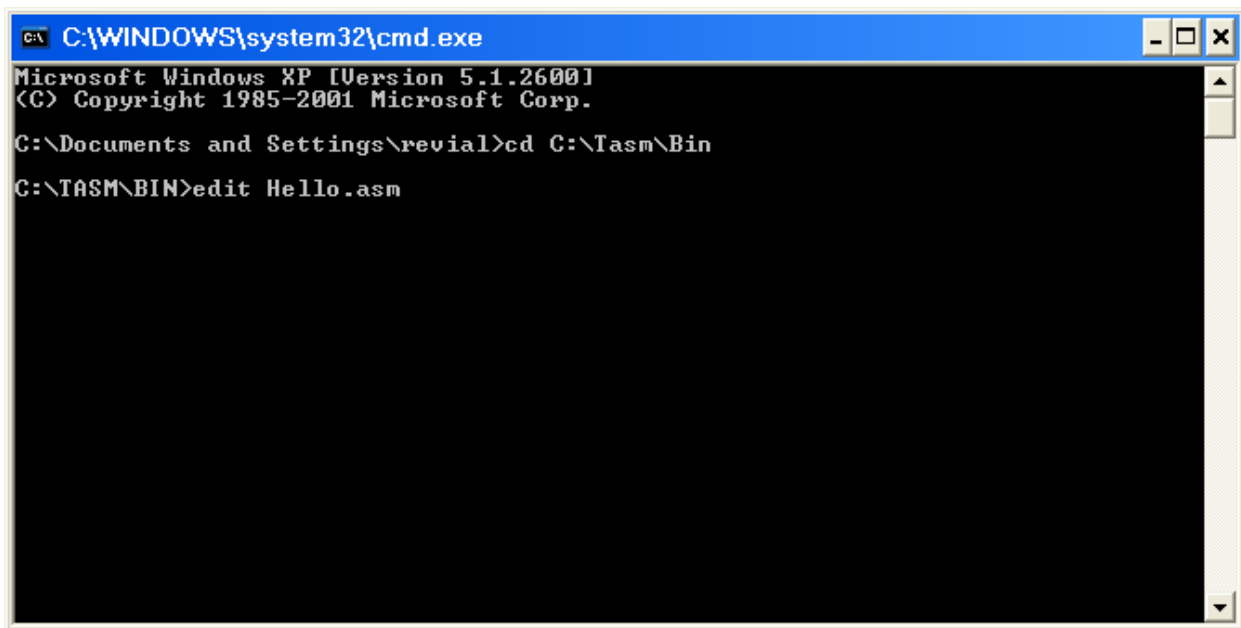
A debugger is a program that allows you to trace the execution of a program and examine the content of registers and memory.

4. *Editor*

You need a text editor to create assembly language source files. MASM6.15 has its own editor or you can use for example Notepad++.

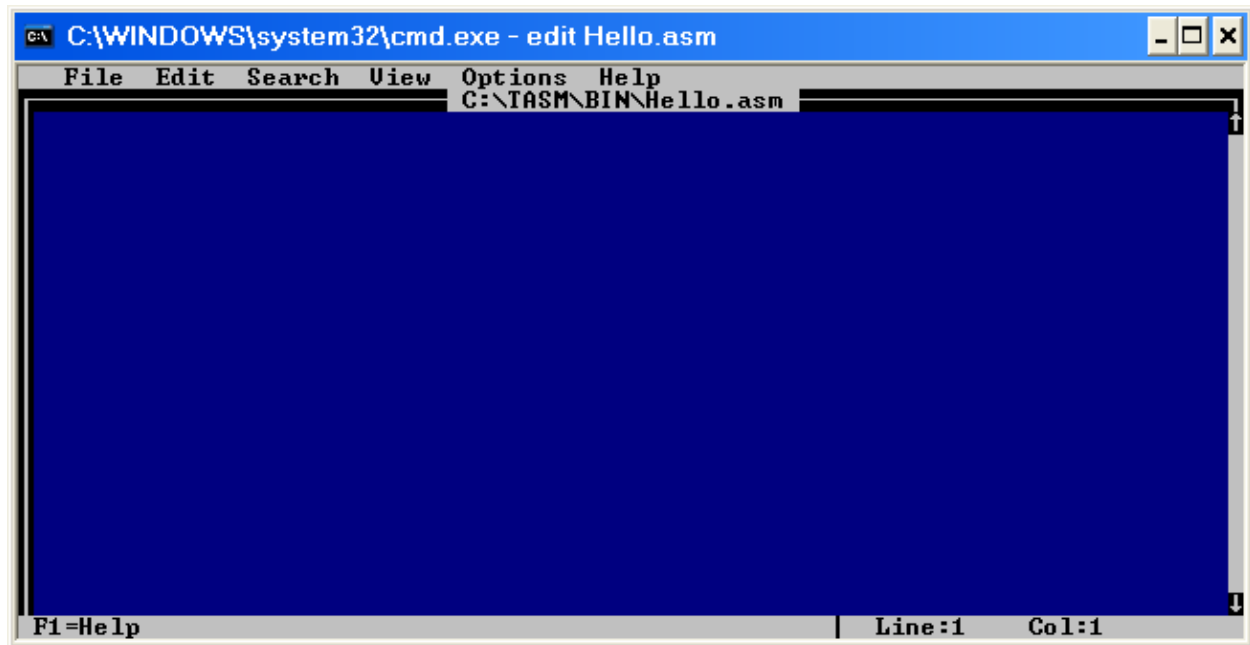
❖ *Running Hello Program on Tasm assembler:*

- 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- A blue screen will open.



```
C:\> 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
```

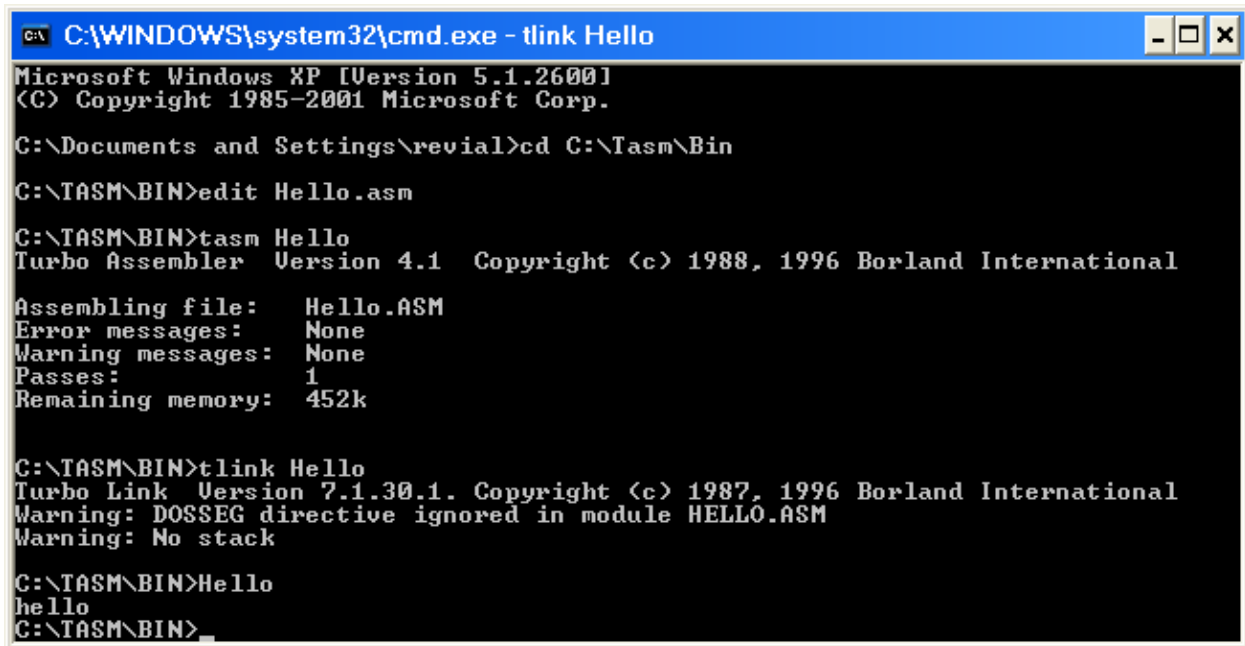



5- Write the following Hello program

A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\command.com". The window has a menu bar with "File", "Edit", "Search", "View", "Options", and "Help". The status bar at the bottom shows "C:\TASM\BIN\Hello.asm". The main area is a blue screen with white text showing assembly code:

```
Dosseg
.model small
.data
message db "Hello to My first Assembly Lab", "$"
.code
main:
    mov ax, @data
    mov ds, ax
    mov ah, 9
    mov dx, offset message
    int 21h
    mov ah, 4ch
    int 21h
end main_
```

- 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

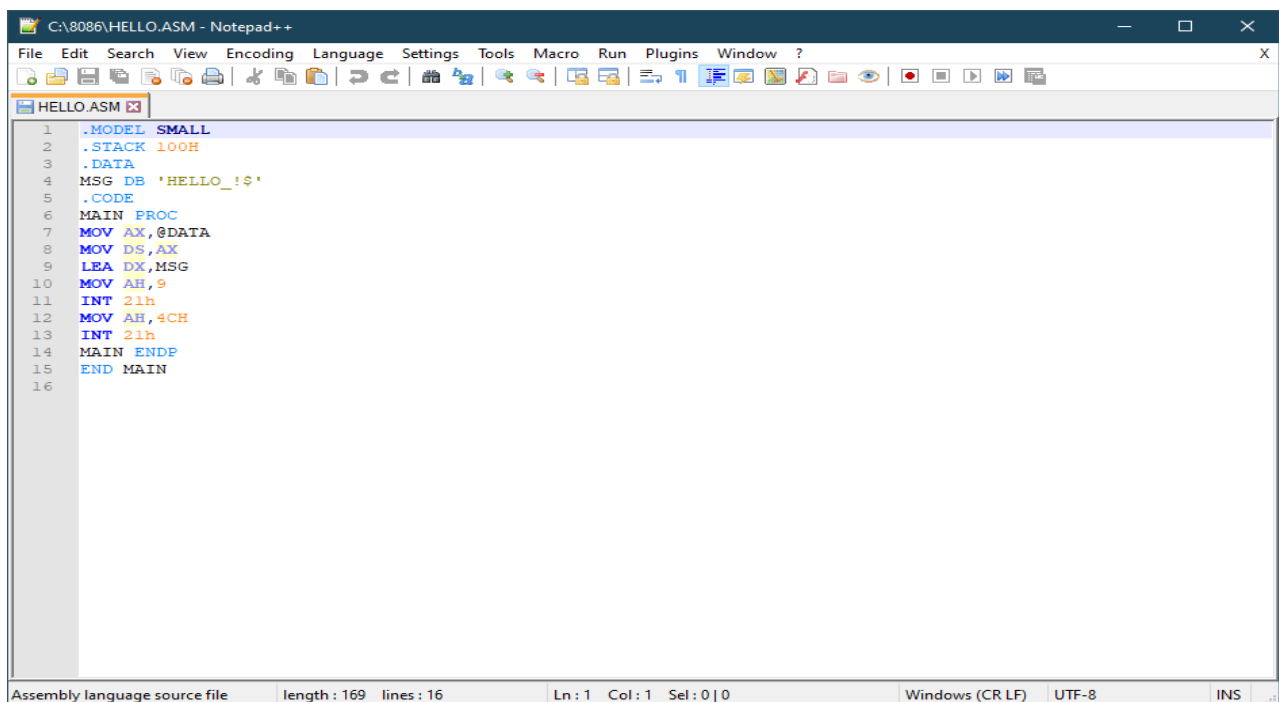
C:\TASM\BIN>tlink Hello
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International
Warning: DOSSEG directive ignored in module HELLO.ASM
Warning: No stack

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

I prefer a light weight and good editor like Notepad++7 with beautiful syntax highlighting.

So, we will use this editor for program writing.

You can download it at <https://notepad-plus-plus.org/downloads/v7.0/>



```
C:\8086\HELLO.ASM - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
HELLO.ASM
1  .MODEL  SMALL
2  .STACK 100H
3  .DATA
4  MSG DB 'HELLO_!$'
5  .CODE
6  MAIN PROC
7  MOV AX,0DATA
8  MOV DS,AX
9  LEA DX,MSG
10 MOV AH,9
11 INT 21h
12 MOV AH,4CH
13 INT 21h
14 MAIN ENDP
15 END MAIN
16

Assembly language source file length : 169 lines : 16 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
```

❖ How to Debugging Assembly Language programs:

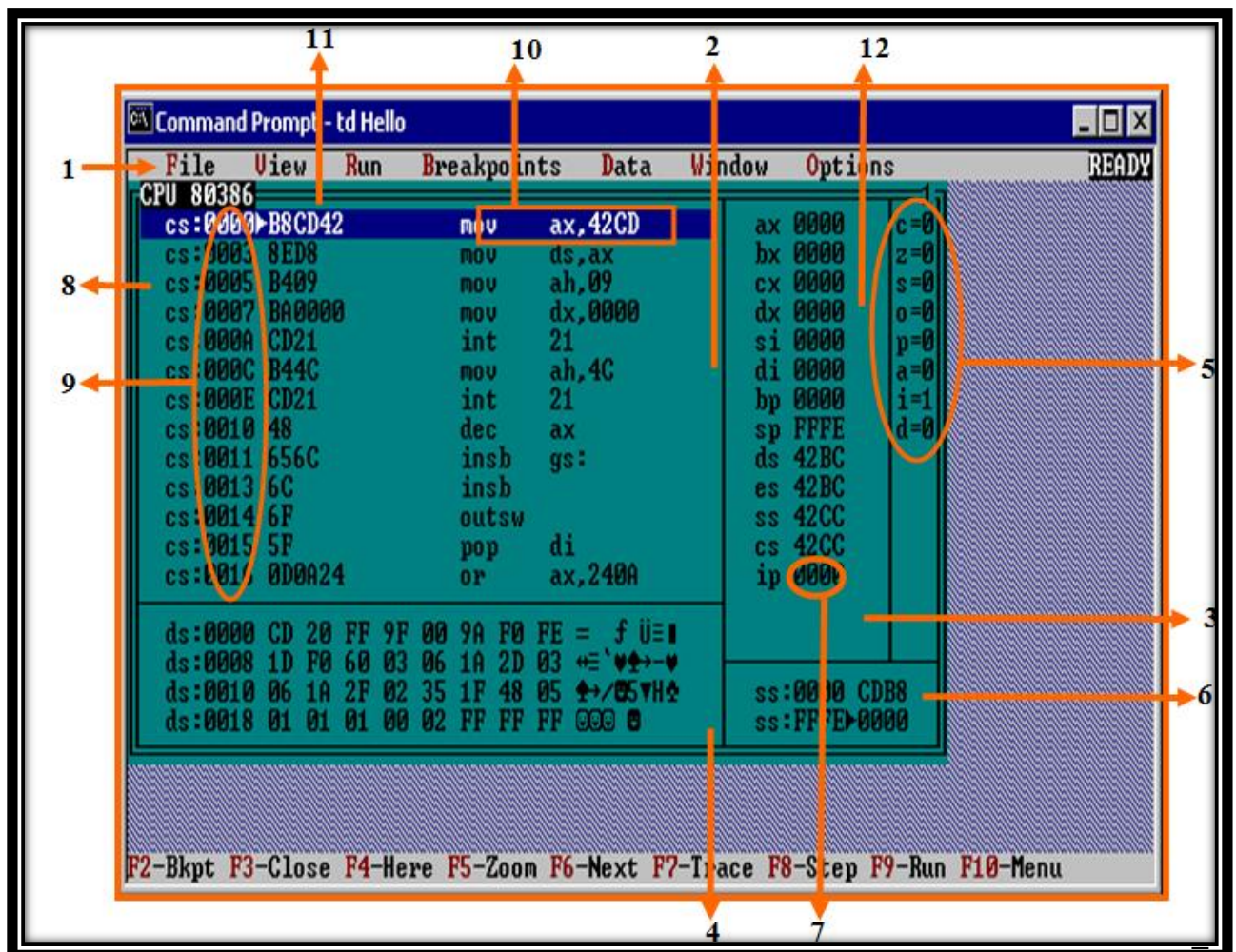
Using Tasm Turbo Debugger:

The Turbo Debugger is a program that allows you to single-step your program (that means run it line-by-line while you watch what happens). You can observe the registers, the memory dump, individual variables, flags, and the code as you trace through your program.

Also it is used to debug errors that have to be made by logic reasons.

After you write your program you can use assembly turbo debugger by follow the following:

C:\Tasm\Bin\td Hello



Number	Description
1	Indicate to the menu bar of turbo debugger
2	Indicate to the region contain Code pane
3	Indicate to the region contain Register pane
4	Indicate to the region contain Data pane
5	Indicate to the region contain Flag pane
6	Indicate to the region contain Stack pane
7	Indicate to the instruction pointer (IP) it contains the offset address of the instruction will be execute.
8	Indicate to Code register that have value of (42CC) and we get it from register pane.
9	The offset address of each instruction
10	This statement tell the assembler to put (@data) default offset address in AX and this value from figure equal to (42CD)
11	indicate to the machine language of statement and from figure it is equal to (B8CD42)
12	This column is the values of Registers.

❖ *Debugging Assembly Language programs*

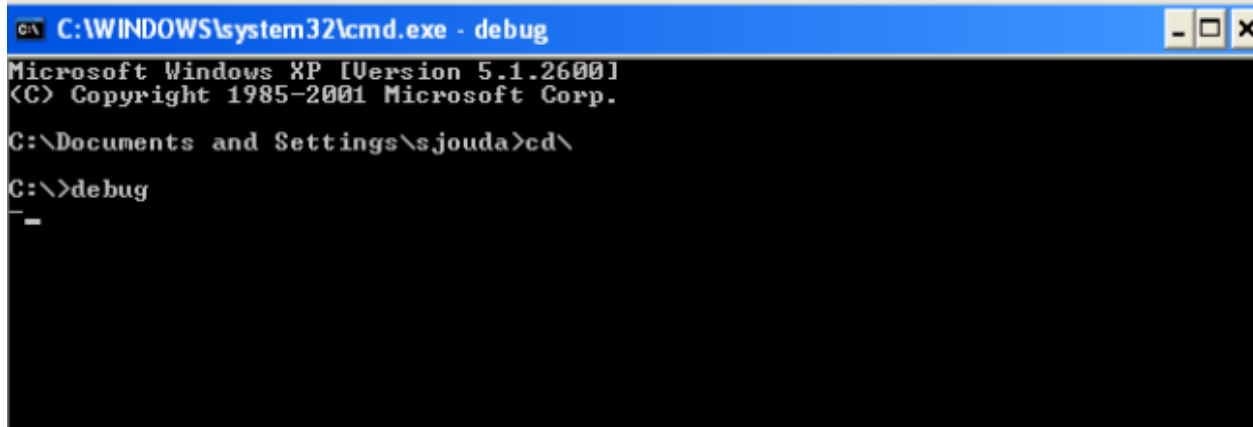
For Debugging you can use:

Debug hello.exe

COMMAND	SYNTAX	FUNCTION	EXAMPLE
Register	R [Register Name]	Examine or modify the contents of an internal register of the CPU	-R AX (AX reg.) -RF ZR (zero flag)
Dump	D [Start Addr] [End Addr]	Display the contents of memory locations specified by Address	-D DS:100 200 -D start-add end-add
Enter	E [Address] [Data]	Enter or modify the contents of the specified memory locations	-E DS:100 22 33 -E address data data
Fill	F [Start Addr] [End Addr] [Data]	Fill a block of memory with data	-F DS:100 120 22
Assemble	A [Starting address]	Convert assembly lang. instructions into machine code and store in memory	-A CS:100 -A start-address
Un-assemble	U [Starting Address]	Display the assembly instructions and its equivalent machine codes	-U CS:100 105 -U start-add end-add
Trace	T [Address][Number]	Line by line execution of specific number of assembly lang. instructions	-T=CS:100 -T=starting-address
Go	G [Starting Address] [Breakpoint Add.]	Execution of assembly language instructions until Breakpoint address	-G=CS:100 117 -G=start-add end-add

❖ *Invoking Debug*

To invoke the DEBUG program, a user opens command prompt window and enters the following:



```
C:\WINDOWS\system32\cmd.exe - debug
Microsoft Windows XP [Version 5.1.2600]
Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\sjouada>cd\
C:\>debug
```

A (Assemble)

Assemble a program into machine language. Command formats:

A

A *address*

If only the offset portion of *address* is supplied, it is assumed to be an offset from CS. Here are examples:

Example	Description
A 100	Assemble at CS:100h.
A	Assemble from the current location.
A DS:2000	Assemble at DS:2000h.

When you press Enter at the end of each line, Debug prompts you for the next line of input. Each input line starts with a segment-offset address. To terminate input, press the Enter key on a blank line. For example:

```
-a 100
5514:0100 mov ah,2
5514:0102 mov dl,41
5514:0104 int 21
5514:0106
```

D (Dump)

The D command displays memory on the screen as single bytes in both hexadecimal and ASCII. Command formats:

D
D *address*
D *range*

If no address or range is given, the location begins where the last D command left off, or at location DS:0 if the command is being typed for the first time. If *address* is specified, it consists of either a segment-offset address or just a 16-bit offset. *Range* consists of the beginning and ending addresses to dump.

Example	Description
D F000:0	Segment-offset
D ES:100	Segment register-offset
D 100	Offset

Q (Quit)

The Q command quits Debug and returns to DOS.

R (Register)

The R command may be used to do any of the following: display the contents of one register, allowing it to be changed; display registers, flags, and the next instruction about to be executed; display all eight flag settings, allowing any or all of them to be changed. There are two command formats:

R
R *register*

Here are some examples:

Example	Description
R	Display the contents of all registers.

T (Trace)

The T command executes one or more instructions starting at either the current CS:IP location or at an optional address. The contents of the registers are shown after each instruction is executed. The command formats are:

```
T
T count
T =address count
```

Where *count* is the number of instructions to trace, and *address* is the starting address for the trace. Examples:

Example	Description
T	Trace the next instruction.
T 5	Trace the next five instructions.
T =105 10	Trace 16 instructions starting at CS:105.

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
E:\>debug hello.exe
-r
AX=FFFF BX=0000 CX=0016 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0000 NV UP EI PL ZR NA PE NC
0744:0000 B84507          MOV     AX,0745
-t
AX=0745 BX=0000 CX=0016 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0003 NV UP EI PL ZR NA PE NC
0744:0003 8ED8          MOV     DS,AX
-t
AX=0745 BX=0000 CX=0016 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0745 ES=0734 SS=0743 CS=0744 IP=0005 NV UP EI PL ZR NA PE NC
0744:0005 B409          MOV     AH,09
-d0
0745:0000 68 65 6C 6C 6F 24 00 FC-26 80 0E 05 00 01 E8 90 hello$.&.....
0745:0010 3B E8 75 3B E8 B4 3B E9-E3 04 BF 3C 01 57 8B D8 ;.u;...<.W..
0745:0020 E8 79 EE 26 C7 06 0A 00-01 00 EB 13 BF 69 01 EB .y.&.....i..
0745:0030 08 BF 5A 02 EB 03 BF CE-02 57 8B D8 E8 5D EE E8 ..Z.....W...l..
0745:0040 2A DB 5F 06 8E C3 53 FF-D7 5B 07 72 0A 26 A3 0A *._...S...[.r.&..
0745:0050 00 26 89 16 0C 00 C3 26-89 1E 06 00 E8 D5 06 F8 .&....&.....
0745:0060 C3 E8 D9 05 E8 55 2E 26-A1 0A 00 E8 6E 3A 0C 20 ....U.&....n:.
0745:0070 26 A2 00 00 33 C0 26 A3-02 00 26 A3 00 20 FD 0B &...3.&...&...
-q
```