Lab 1: Visual Studio (Visual C++ Express) + MASM

Requirements:

1. Run the debuggers and turn in an MS-WORD or PDF file that contains three snapshots of using the debugger.
    a. Snapshot 1: the window for monitoring registers (32-bit).
    b. Snapshot 2: the window for monitoring variables and registers in a watch window (32-bit).
    c. Snapshot 3: the window for monitoring memory locations (32-bit).
2. Use the debugger to verify the little endian order
3. Please submit on Canvas by 11:59pm 9/10/2013.

Assignments:

Part I

You will use one of the following software applications for programming in this class:

- Visual Studio 2010/2008
- Visual C++ Express 2010 /2008

Note you will need to register after installation if you'd use Visual C++ Express. To register, go to Help→Register Product. Then follow the instructions for registering.

Part II

You can download the link libraries and examples at http://kipirvine.com/asm/examples/index.htm. Based on the version of your Visual Studio or Visual C++, choose "Example programs and link libraries for Visual Studio …". The download is a zip file. Please unzip the content to c:\irvine.

Part III

The easiest way to get started is to use an example project you just downloaded. It can be found in "C:\Irvine\Examples\Project_sample".

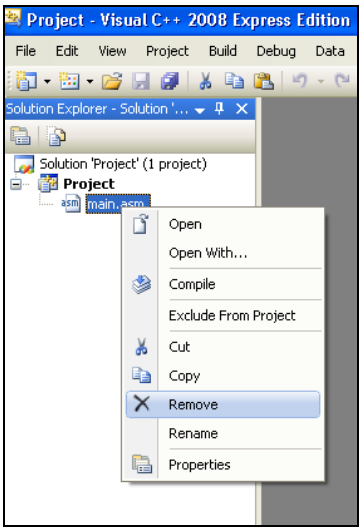
You could also set up a project, in Visual Studio/C++, from scratch. The instructions can be found at

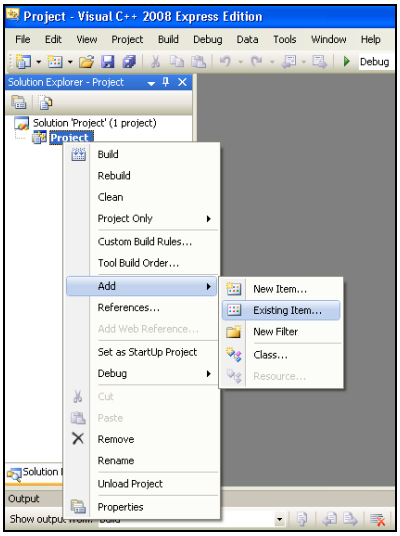
- Visual Studio/C++ 2008: http://kipirvine.com/asm/gettingStartedVS2008/index.htm
- Visual Studio/C++ 2010: http://kipirvine.com/asm/gettingStartedVS2010/index.htm

Part IV

Now please copy hello32irvine.asm to the project directory. Replace main.asm in the project with hello32irvine.asm by doing the following:

- Right-click "main.asm" and choose "Remove" in the floating window.
- Right-click on "Project" and choose "Add→Existing Item".
- Select "hello32irvine.asm" and click "OK"





Part V

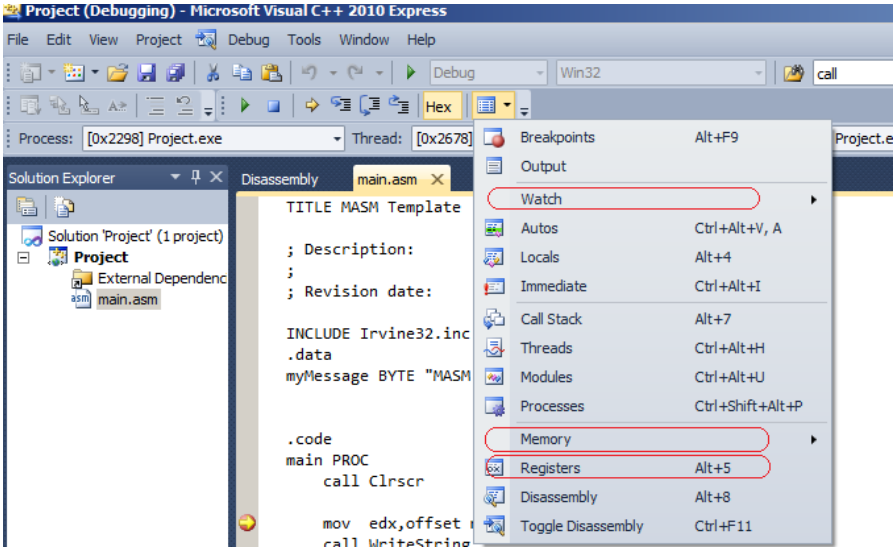
The next important task is to learn how to use the debugger. Assembly programs are more difficult to understand than high level language programs. Using debugger can greatly improve the efficiency of the process of locating and fixing errors. Additionally, knowing how to use debuggers can improve your abilities of working independently. If you have ever used any debugger, you will find this one very similar. All the debuggers have some common functions as follows:

Functions	**Hot Keys**
Set a breakpoint	move the cursor to the line and press F9
Run to a breakpoint	F5
Step through your program (run your program one line at a time)	F10
Step into a function/procedure	F11
Step out of a function/procedure	Shift+F11
Terminate the debugging mode	Shift+F5
Run your program without debugging	Control+F5

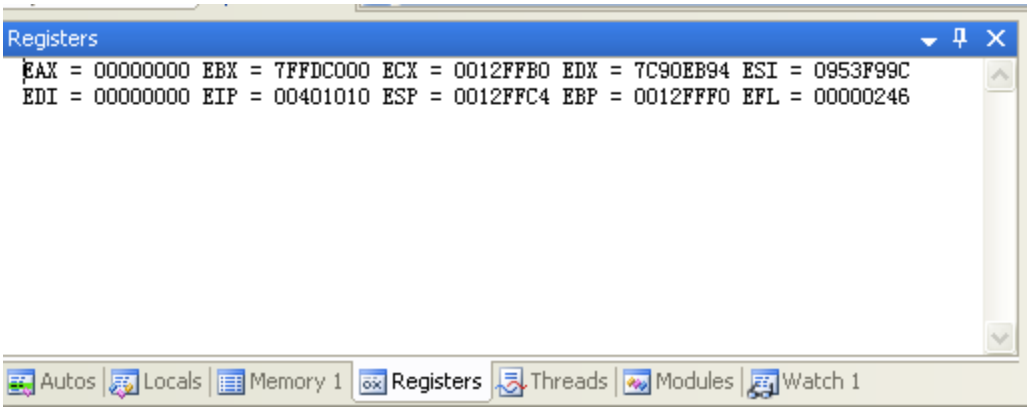
In the debugging mode, you can monitor the value of registers, variables, and memory locations. Please use hello32irvine.asm, enclosed in the lab1.zip, for this part. First you need to start the debugger by pressing F10 and turn on the following windows:

1. Select Debug->Windows->Watch->Watch 1 (**Ctrl-Alt-W, 1**)
2. Select Debug->Windows->Memory->Memory 1 (**Ctrl+Alt+M, 2**)
3. select Debug->Windows->Registers (**Ctrl+Alt+G**)

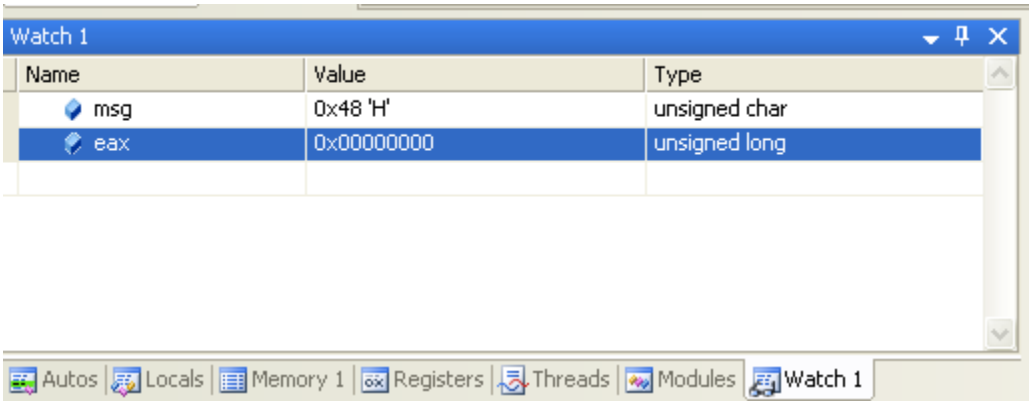
Or you may have to use the tool bar as follows: (Some installations do not show some of these menu items. If that is the case, use the hot keys.)



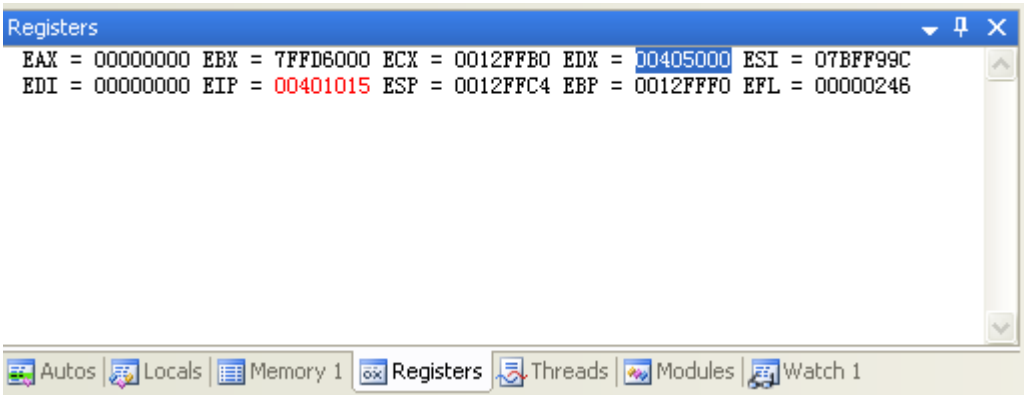
These actions will create three windows at the bottom. Please select "Registers". The window shows the values of registers in hexadecimal.

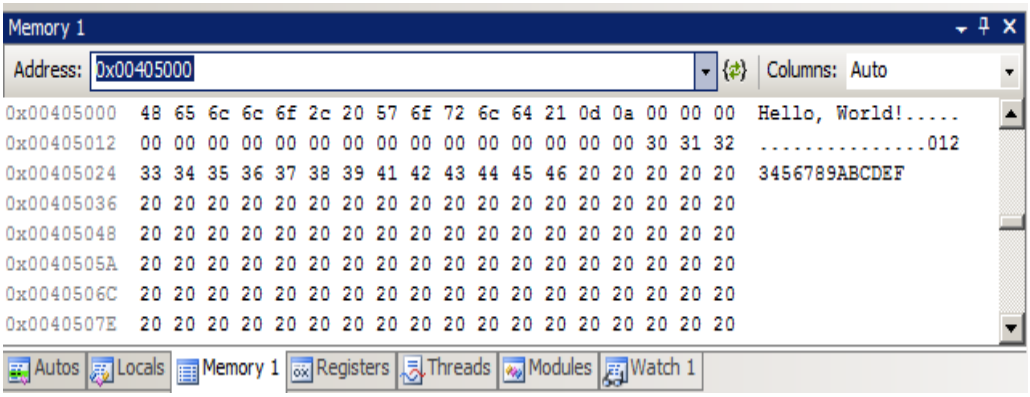


Please select "Watch 1". The window may be empty. Double-click the first entry under "name" and input "msg" and press enter. Then input a register's name, say eax. As you can see, the window shows the values. For "msg" which is an array variable, the debugger can only show the first element.



Before discussing monitoring memory, please step-trace your program and stop at "call WriteString". Then, select tab "Registers" at the bottom and copy the value of edx. Now select tab "Memory 1" and input the value of edx in the field, labeled "Address:". Note the value must be preceded by "0x" to indicate it is a hexadecimal value.





Now, the major functions of the debugger have been introduced.

Part VI

In this part, please verify the data's little endian order in memory for the following variables using the debugger.

- dVariable DWORD 11223344h
- bVariable BYTE 11h, 22h
- wVariable WORD 3344h

Attach the screenshot of debugger and find a way to **explain the order of the bytes in the snapshot**.