COSC2406 Lab 3 – Introduction to Assembly Language

Coding: submit all asm files to the LMS.
Tutorial: Command Prompt Assembly! Create a file called Hello.asm with the following program:

```
INCLUDE Irvine32.inc
.data
hello BYTE "Hello Assembly! ",0

.code
main PROC
        MOV EDX, OFFSET hello           ;Print the Hello Message
        CALL WriteString
        CALL CRLF
        EXIT
main ENDP
END main
```

Open the Visual Studio Developer Console and change to the directory with the file using the command **cd <path>**.

Then, assemble the program using Visual Studio Developer console using the command:
ml /nologo /c /I C:\Irvine Hello.asm

Then, link the generated .obj file with the command:
link /NOLOGO /SUBSYSTEM:CONSOLE /LIBPATH:C:\Irvine irvine32.lib kernel32.lib user32.lib Hello.obj

Finally, execute compiled file using one of these commands:
Hello.exe OR ***./Hello.exe***. If one doesn't work, try the other.

Modify the string in the file to say "Hello Assembly World!" instead of "Hello Assembly!", then assemble and link the file in one step using the following command:
ml /nologo /I C:\Irvine Hello.asm /link /NOLOGO /SUBSYSTEM:CONSOLE /LIBPATH:C:\Irvine irvine32.lib kernel32.lib user32.lib

Then, you can run the compiled file using the methods we did before.

**** Note **** If you unzipped your Irvine files to a different directory than C:/Irvine, use that path instead. If you get an error and you did unzip to C:/Irvine, try using C:\Irvine\Irvine instead. If this works, you can move the files from C:\Irvine\Irvine to C:\Irvine, or continue using C:\Irvine\Irvine when C:\Irvine is referenced.

COSC2406 Lab 3 – Introduction to Assembly Language

Tutorial / Solo: Using Visual Studio to Assemble. Create an assembly project in Visual Studio as shown, create a new assembly file called SubTwo and type in the program on page 3. Build the file using Build > Build Solution (or, use Ctrl + Shift + B), then run the solution without debugging using Debug > Run Without Debugging (or, use Ctrl + F5 or Ctrl + FN + F5). Try the program with positive values. It should work correctly, displaying the result of num2 - num1. Run it again, but use a negative number. You should notice that the program now displays +0 where the negative value was entered. Change two lines in the reading part of the code only (so, don't change DWORD to SDWORD) to fix the issue. Hint: look at which function is reading the number. What type of value does it read, and what type of value do we need? Bigger Hint: Look to where we're writing the number; change Write to Read.

Tutorial: Making Setup Easier By Creating a Project Template. Once your project is created and configured properly, you can save it as a project template that you can use to set up further projects without needing to go through the setup process again.

Solo: Assembly Practice. Write an assembly program to read in an unsigned decimal value and convert it to binary and hexadecimal. You will need the WriteBin and WriteHex functions, as well as the ReadDec function. Use the previous code to guide you to input the value. If you don't get to this part, that's perfectly fine! For this lab, only the first two parts count towards your grade. In the future, all parts will contribute, so you will need to get used to working solo.

```
INCLUDE Irvine32.inc
.data
prompt1 BYTE "Enter the first number: ",0
prompt2 BYTE "Enter the second number: ",0
num1 DWORD ?
num2 DWORD ?
answer DWORD ?
answer1 BYTE " - ",0
answer2 BYTE " = ", 0

.code
main PROC
        MOV EDX, OFFSET prompt1         ;Prompt for the first number
        CALL WriteString
        CALL ReadDec
        MOV num1, EAX

        MOV EDX, OFFSET prompt2         ;Prompt for the second number
        CALL WriteString
        CALL ReadDec
        MOV num2, EAX

        SUB EAX, num1                   ;Calculate num2 – num1
        MOV answer, EAX

        MOV EAX, num2                   ;Print the answer
        CALL WriteInt
        MOV EDX, OFFSET answer1
        CALL WriteString
        MOV EAX, num1
        CALL WriteInt
        MOV EDX, OFFSET answer2
        CALL WriteString
        MOV EAX, answer
        CALL WriteInt
        CALL CRLF
        EXIT
main ENDP
end main
```