Assembler #2

Re-submit Assignment

Due Jul 19 by 11:59pm **Points** 50 **Submitting** a file upload **File Types** asm and exe

At the end of this assignment, upload both your assembler source: **CS2810-Assembler-Template.asm** and the executable file **CS2810-Assembler-Template.exe.**
(note that the .asm file is in your project folder and the .exe in the debug folder within the project)

a. Download the assignment project template zip file from 2015 project template: **https://weberstate.box.com/v/2810-asm-vs2015 (https://weberstate.box.com/v/2810-asm-vs2015)**
   2017 project template: **https://weberstate.box.com/v/2810-VS2017-template (https://weberstate.box.com/v/2810-VS2017-template)**
b. Unzip the project folder to a location on your hard drive.
c. Start Visual Studio and open the VS solution contained in the project folder.
d. When the project has started, open the assembler source code file, CS2810-Assembler-Template.asm
e. Modify the program source code to do the following:
   (Basically, you are writing a program to do the FAT16 Root Area time decoding from week 7)

1. Clear the display.
2. Display the string "CS 2810 Spring Semester 2018" on row 15 column 20 of the display.
3. Display the string "Assembler Assignment #2" on row 16, column 20.
4. Display your name on row 17, column 20 of the display.
5. Prompt the user for FAT16 file time in hex format and allow the user to enter it on row 19, column 20.
6. Decode the entered file time and convert it to ASCII for display.
7. Display the converted file time on row 20, column 20.
8. Thoroughly test your project before submitting.

The Irvine library contains a number of Procs and Macros that simplify I/O in MASM.

On this assignment you will use three of the Procs in the library:

- **Clrscr** - clears the command window.
- **Gotoxy** - positions the cursor at the row and column specified in register sections DH and DL
- **WriteString** - displays the string at the address specified in the EDX register. The string must be 0 terminated.
- **ReadHex** - allows the user to enter a hex value which is converted to binary and placed in EAX

The sample program shows how to call the Procs.
Note that with any Procs that require values to be in registers, the registers **MUST** be set **PRIOR** to calling the Proc.

The following MASM instructions will also be used on this assignment.

MOV - the MOV instruction copies from register to register, register to memory and memory to register.
Sample formats are:
MOV ECX, EAX
MOV word ptr [displayTime], ax

ROR - the Rotate Right command shifts the bits of a register to the right. As bits are shifted out on the right, they rotate around and are placed back into the left. An example is ROR AX,8.

SHR - The Shift Right command shifts the bits of a register to the right. As bits are shifted out to the right, zeros are placed in the left. An example is SHR AX, 11

SHL - The Shift Left command shifts the bits of a register to the left. As bits are shifted out to the left, zeros are brought in on the right. The effect of the command SHL AX,1 is that of multiplying the AX register by 2.

DIV - The Divide instruction allows for 8, 16 and 32 bit division. In this assignment you will use AX register as the dividend and use an 8-bit register half as the divisor. After the division, the quotient will be in AL and the remainder in AH.
An example is: DIV BL
Note that the dividend is not specified. It is always AX for 8-bit division.

ADD - The add instruction does just what it specifies. In this assignment you will use a form of the add wherein a literal is added to a register. Example: ADD AX, 3030h

AND - This instruction performs a bit-by-bit logical AND between a register and a literal with the result of the AND placed in the specified register. It is used to mask (ie. strip out) bits from a register.

Examples:

AND AX,1111100000000000b

AND AX,F800h

The command window is typically 80 columns (characters) wide and 25 rows (lines) deep.