Assembler #4

Re-submit Assignment

Due Aug 2 by 11:59pm **Points** 50 **Submitting** a media recording or 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 FAT16 Date decoding from week 7)

1. Clear the display.
2. Display the string "CS2810 Summer Semester 2017" on row 4 column 33 of the display.
3. Display the string "Assembler Assignment #4" on row 5, column 33 .
4. Display your name on row 6 column 33 of the display.
5. Prompt the user for a FAT16 file date on row 8, column 33 of the display.
6. Allow the user to enter the file date value following the prompt.
7. Decode the file date and display the decoded date on row 10, column 33 in the format:
   November 12th, 2011
   (*Note that you can use FAT16 file date info from the Week 7 Root Area worksheet to test your program)
8. Thoroughly test your project before submitting.

Your Assignment must you an "array" for at least the month's portion. Brute force comparisons will not be accepted.

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 likely be used on this assignment.

CALL - the CALL instruction pushes a return address on the stack and then jumps to the specified label.
Example: CALL DisplayName

RET - The RET instruction jumps to the address on the top of the stack and pops the stack. It is used to provide a return from a CALL.
Example: RET

PUSH - Places the specified register on top of the memory stack.
Example: PUSH EDX

POP - Takes the value from the top of the memory stack and moves it to the specified register and then removes the value from the top of the stack. Example POP EDX

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+0], ax

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.

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

CMP - performs an implied subtraction of the source operand from the destination operand.
Example:
CMP EAX,3

JZ, JNZ, JG, JL, JGE, JLE - are conditional jumps where program execution is transferred to the specified label based on the state of both the zero flag and the sign flag.
Example:
JZ newLocation

JMP - is an unconditional jump. Program execution is transferred to the specified label.
Example:
JMP newLocation

MUL - multiplies the contents of the specified register by the contents of the EAX register and places the result in the EAX register.
Example: MUL BL

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 for 8-bit division. For 32-bit division, the quotient is in AX and the remainder in DX.
Examples:
DIV BL
DIV BX
Note that the dividend is not specified. It is always AX for 8-bit divisors and DX:AX for 16-bit divisors

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