Final Exercise

There are three parts to this exercise. Follow the instructions for each part below and turn in your submission according to the instructions for each exercise.

IMPORTANT:

Make sure that you reference any outside sources that you used to help with your submission. This means that if you find a helpful example on stackexchange, for example, you must reference the webpage that refers to the example.

Additionally, you are required to comment on every instruction in your assembly language programs. Programs without correct or meaningful comments will be graded according to the rubric below.

Grading Rubric:

Each program below is worth 100 points. The following rubric is applied to each part of the exercise:

Part_1:	100
Part_2 and Part_3:	
Assembles without error:	10
Correct comments on each instruction:	10
Correctness of implementation:	80
TOTAL (Each Part):	**100**
Total (3 Parts):	**300**

Final_Part_1

Examine the file titled Final_Part_1.pdf. The file contains a series of assembly instructions, a list of registers and their values, and a small memory snapshot. For each assembly instruction (1-7) in the file, record any changes that will occur to memory or to a register value when the instruction is invoked. The instructions are performed on a little-endian system.

For example, with this instruction:

```
mov eax,0ah
```

The answer should be:

The `eax` register would change from 0xFFCC5665to 0x0000000a.

Note that the instructions are cumulative, so that changes to a register or updates made to data in a memory location persist across subsequent instructions. **Make sure you state the values before and after the instruction for each part.** Be clear when specifying values in either hexadecimal or as bytes in memory by making sure to take the endianness of the value into consideration.

Record the changes for each of the instructions in a text file and turn it in as Final_Part_1.txt.

Final_Part_2

In this exercise you will write an assembly language program that performs the following actions:

- Prompt the user for two integers
- Compare the integers entered by the user
- Display the integers in order from smallest to largest, or, if the integers are equal, state that the two integers are the same value.

For example, the program output might appear as follows:

```
Enter Integer 1: 25
Enter Integer 2: 15

The integers in order are 15 and 25.
```

Or:

```
Enter Integer 1: 456
Enter Integer 2: 456

The integers are equal.
```

The output of your program may be different from this example as long as the requirements stated above are met.

When finished, name your .asm file Final_Part_2.asm and submit the file to D2L.

Final_Part_3

In this exercise you will write an assembly language function that changes the case of each character within a string. The function accepts a pointer to the target string as its only parameter:

```
int SwapCase(char *string);
```

The function must convert each lowercase character to uppercase and each uppercase character to lowercase. Numbers, whitespace characters, and punctuation characters should be ignored.

You are provided with a program that prompts the user for the string and prints the string after the change in case. Here is a sample output from the program:

```
Enter a string:  This is a samPle STRING.

Your string before SwapCase:
     This is a samPle STRING.

Your string after SwapCase:
     tHIS IS A SAMpLE string.
```

When finished, name your .asm file Final_Part_3.asm and submit the file to D2L.