CSEC 201, LAB #9: BASIC ASSEMBLY CODING	DUE: 11/3/2021 @ 7 PM

Learning Outcome: Students will gain experience in using a debugger to read registers and memory and writing assembly code.

1. (25 points) Referencing and dereferencing

   Convert the following C program to x86 assembly.

```
int a = 10;
int b = 20;
int *eax = &a;
int *ebx = &b;
b += a;
*eax = *eax + *ebx;
printf("eax points to %d\n", *eax);
```

2. (25 points) Stack operations

   Compute `10 + 6*12 - (4+15)` and print the result to standard output.

   Here are the laws of computation you must respect:

   1) Immediate operands can only be used with `push`.
      For example, `push 5` is allowed but `add eax,5` is not.
   2) The `mov` instruction will not be used
   3) The `ecx` and `edx` registers will not be used.
   4) There will be no variables declared in the `.data` section except the one to be passed into `printf`.

3. (25 points) String copy

Write an x86 assembly program that declares two string variables str1 and str2, copy str2 to str1, and print str2 as follows:

```
char str1[] = "ABCDEF";
char str2[] = "XYZ123"
strcpy(str2, str1);
printf("str2 = %s\n", str2);
```

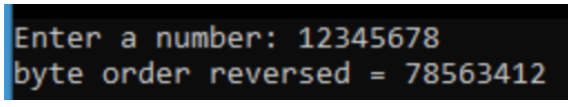
Note: You will not be using the built-in `strcpy` function. You may assume that the length of both `str1` and `str2` is 6 (because we have not learned repetition yet)

4. (25 points) Standard input

   Use `scanf` to prompt the user to get an integer value in a hexadecimal format.

   Reverse the 4 bytes of the integer.

   Print the reversed integer.



CSEC 201, LAB #9: BASIC ASSEMBLY CODING	DUE: 11/3/2021 @ 7 PM

Deliverables:

1. You must submit the working code to the GitHub Assignment10 repository before 11/3/2021 at 7 PM.

CSEC 201, LAB #9: BASIC ASSEMBLY CODING	DUE: 11/3/2021 @ 7 PM

NAME:

Each signature is worth 1/N of your lab grade where N is the number of signatures.

- **Student could understand how referencing and dereferencing work in assembly.**

- **Student used push and pop to compute expressions using a limited number of registers.**

- **Student could understand how string copy works.**

- **Student used the scanf function and could manipulate byte orders in memory.**