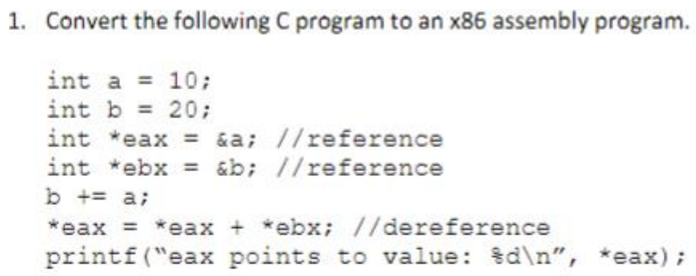
Griffin Pundt

This document will outline the functionality of each .asm file.

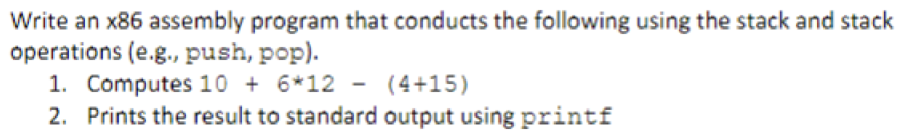
<u>Part1.asm:</u>



1. Convert the following C program to an x86 assembly program.

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

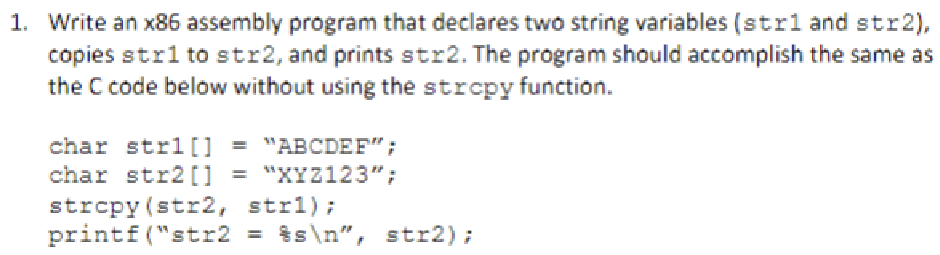
<u>Part2.asm</u>



Write an x86 assembly program that conducts the following using the stack and stack operations (e.g., `push`, `pop`).

1. Computes `10 + 6*12 - (4+15)`
2. Prints the result to standard output using `printf`

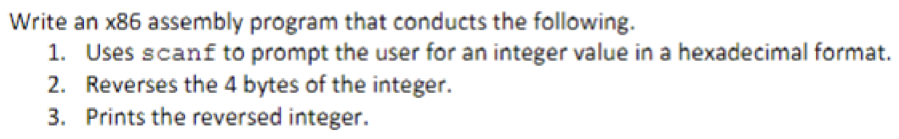
<u>Part3.asm:</u>



1. Write an x86 assembly program that declares two string variables (`str1` and `str2`), copies `str1` to `str2`, and prints `str2`. The program should accomplish the same as the C code below without using the `strcpy` function.

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

<u>Part4.asm</u>



Write an x86 assembly program that conducts the following.

1. Uses `scanf` to prompt the user for an integer value in a hexadecimal format.
2. Reverses the 4 bytes of the integer.
3. Prints the reversed integer.