1. In the below code, written in intel syntax, how many arguments does this subroutine have?

```
push ebp
mov ebp, esp
sub esp, 8
mov DWORD PTR [ebp-4], ecx
mov DWORD PTR [ebp-8], edx
mov edx, DWORD PTR [ebp-4]
mov eax, DWORD PTR [ebp-8]
add edx, eax
mov eax, DWORD PTR [ebp+8]
add edx, eax
mov eax, DWORD PTR [ebp+12]
add edx, eax
mov eax, DWORD PTR [ebp+16]
add eax, edx
leave
ret 12
```

---

2. Using an assembler show and contrast the differences between the following code once it is compiled. Do optimizations change the ways the constructs work? Include source code and assembly.

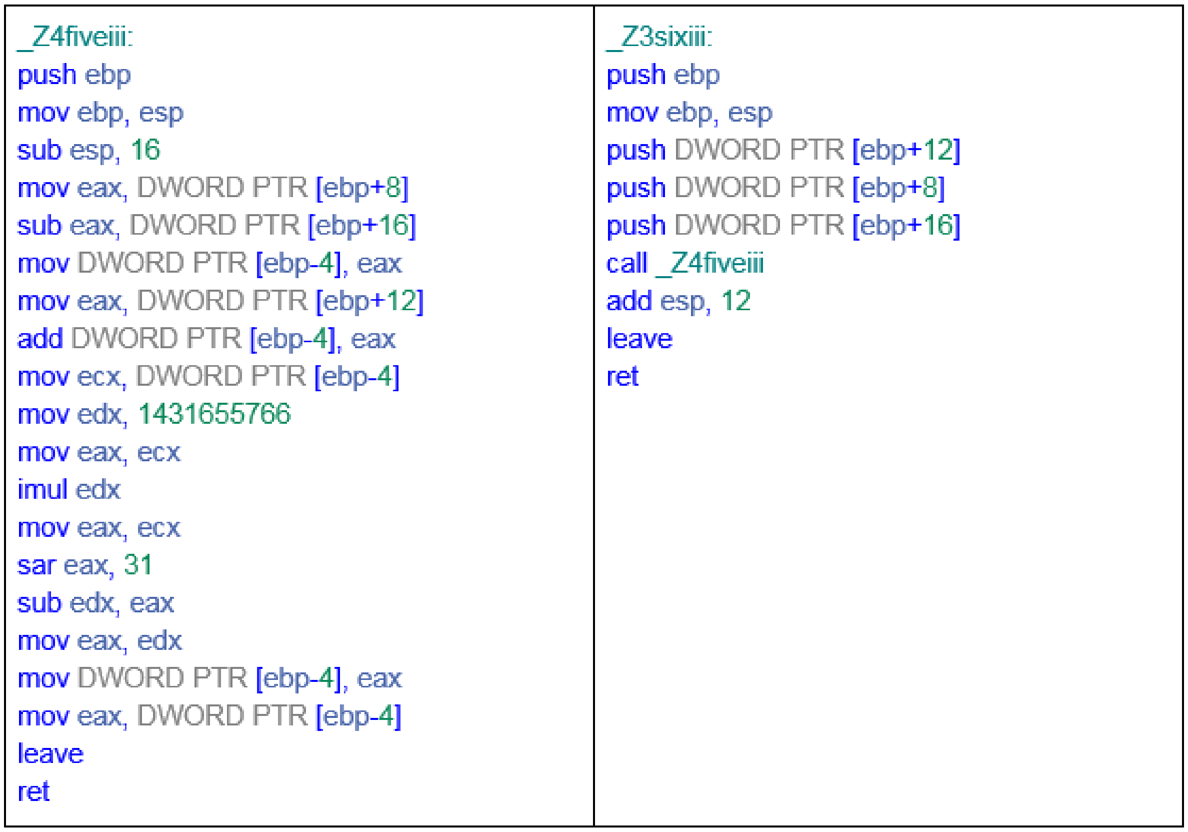
- The c++ post-fix increment (i++) compares to i = i + 1.
- The ternary operator compared to if and else.
- The while, for, and do while loops.

---

3. In the below code in AT&T Syntax, In the below basic block of assembly, what value is returned by this subroutine when the last instruction executes.

```
push %ebp
mov %esp, %ebp
mov 8(%ebp),%ecx
mov12(%ebp),%eax
sub %eax, %ecx
mov %ecx, %eax
pop %ebp
Ret
```

4. In the below code, How many arguments does function Five and function Six have?



```
_Z4fiveiii:
push ebp
mov ebp, esp
sub esp, 16
mov eax, DWORD PTR [ebp+8]
sub eax, DWORD PTR [ebp+16]
mov DWORD PTR [ebp-4], eax
mov eax, DWORD PTR [ebp+12]
add DWORD PTR [ebp-4], eax
mov ecx, DWORD PTR [ebp-4]
mov edx, 1431655766
mov eax, ecx
imul edx
mov eax, ecx
sar eax, 31
sub edx, eax
mov eax, edx
mov DWORD PTR [ebp-4], eax
mov eax, DWORD PTR [ebp-4]
leave
ret
```

```
_Z3sixiii:
push ebp
mov ebp, esp
push DWORD PTR [ebp+12]
push DWORD PTR [ebp+8]
push DWORD PTR [ebp+16]
call _Z4fiveiii
add esp, 12
leave
ret
```

5. Using the same Assembly code as question (4) What is accomplished logically by the mangled “six” function.

---

6. Given the instruction: cmp a, b;
Give values for a and b to set all possible combinations of EFLAGS that can be set from a cmp instruction. Construct a table with the values of a and b and what flags were set.

7. Calculate the value of the eax register when the main function returns.

```
main:
push ebp
mov ebp, esp
sub esp, 32
mov DWORD PTR [ebp-24], 123434
mov DWORD PTR [ebp-20], 9000
mov DWORD PTR [ebp-16], 2243244
mov DWORD PTR [ebp-12], 34250234
mov DWORD PTR [ebp-8], 234234
mov DWORD PTR [ebp-4], 0
lea eax, [ebp-24]
push eax
call _Z1fPi
add esp, 4
cwde
leave
ret
```

```
_Z1fPi:
push ebp
mov ebp, esp
sub esp, 16
mov DWORD PTR [ebp-8], 0
jmp .L2
.L3:
mov eax, DWORD PTR [ebp-8]
lea edx, [0+eax*4]
mov eax, DWORD PTR [ebp+8]
add eax, edx
mov eax, DWORD PTR [eax]
mov edx, eax
movzx eax, WORD PTR [ebp-2]
add eax, edx
mov WORD PTR [ebp-2], ax
add DWORD PTR [ebp-8], 1
.L2:
mov eax, DWORD PTR [ebp-8]
lea edx, [0+eax*4]
mov eax, DWORD PTR [ebp+8]
add eax, edx
mov eax, DWORD PTR [eax]
test eax, eax
jne .L3
movzx eax, WORD PTR [ebp-2]
leave
ret
```

8. If the following program was ran with arguments 10 and 9, what would be the return value?

```
_Z1fPiS_:
push ebp
mov ebp, esp
sub esp, 16
mov DWORD PTR [ebp-4], 9000
mov eax, DWORD PTR [ebp+12]
mov eax, DWORD PTR [eax]
imul edx, eax, 9000
mov eax, DWORD PTR [ebp+8]
mov DWORD PTR [eax], edx
mov eax, DWORD PTR [ebp+8]
mov eax, DWORD PTR [eax]
mov edx, 9000
sub edx, eax
mov eax, DWORD PTR [ebp+12]
mov DWORD PTR [eax], edx
leave
ret
```

```
main:
lea ecx, [esp+4]
and esp, -16
push DWORD PTR [ecx-4]
push ebp
mov ebp, esp
push ebx
push ecx
sub esp, 16
mov ebx, ecx
mov eax, DWORD PTR [ebx+4]
add eax, 4
mov eax, DWORD PTR [eax]
sub esp, 12
push eax
call atoi
add esp, 16
mov DWORD PTR [ebp-12], eax
mov eax, DWORD PTR [ebx+4]
add eax, 8
mov eax, DWORD PTR [eax]
sub esp, 12
push eax
call atoi
add esp, 16
mov DWORD PTR [ebp-16], eax
sub esp, 8
lea eax, [ebp-16]
push eax
lea eax, [ebp-12]
push eax
call _Z1fPiS_
add esp, 16
mov edx, DWORD PTR [ebp-12]
mov eax, DWORD PTR [ebp-16]
add eax, edx
lea esp, [ebp-8]
pop ecx
pop ebx
pop ebp
lea esp, [ecx-4]
ret
```

Graduate Students:
Produce an assembly file (-s in gcc and .s file) which computes the 13th fibonacci number consisting of no registers. The filed turned in should be compilable assembly code using standard g++ and each instruction of the assembly file should not contain standard process registers (EAX, EBX, ECX, EDX, ESI, EDI) The exception is you are allowed to use EBP and ESP in this program. You should not use the smaller size aliases of these registers
i.e. You should not use AL in place of EAX,.