CS 31 Homework 5: IA32 Loops and Functions
Due at the start of class Thursday, Oct. 21, 2021 (after Fall Break)

Names and Roles:

Question 1

Convert the following C code fragment to equivalent IA32 assembly code in two steps:
(1) First, translate the loop to its equivalent C goto version
(2) Next, translate your C goto version to IA32, assuming that `dog` is at `r[%ebp] - 4`, `cat` is at `r[%ebp] - 8`, and `goat` is at `r[%ebp] - 12`.
You must show both steps (1) and (2), and to receive partial credit annotate your IA32 code with comments describing which part of the C code you are implementing.

```
        int dog, cat, goat;
        dog = 12;
        cat = 90;
        goat = dog - cat;
        while (dog < cat) {
            dog *= 2;
            goat += dog;
        }
```

```
(2) IA32 Translation
--------------------
```

```
(1) C goto version
------------------
```

Question 2

Trace through the following IA32 code. Show the contents of the given memory and registers **just before the instruction at point A is executed**. Assume the `addl` instruction in `main` that is immediately after the `call` instruction is at memory address `0x1234`. Hints:

- remember to start execution in `main`.
- %esp points to the item on the top of the stack, so a push will grow the top of the stack and then move in the pushed value. A pop will move the value on top of the stack and then shrink the stack.
- The sequence of instructions `leave`; `ret` is equivalent to the sequence `movl %ebp, %esp`; `popl %ebp`; `popl %eip`.

```
foo:
    pushl  %ebp
    movl   %esp, %ebp
    subl   $16, %esp
    movl   8(%ebp), %eax
    addl   %eax, %eax
    movl   %eax, -4(%ebp)
    movl   -4(%ebp), %eax
    leave                   # point A
    ret
main:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $6, -4(%ebp)
    pushl   -4(%ebp)
    call    foo
    addl    $4, %esp        # at addr 0x1234
    movl    %eax, -4(%ebp)
    movl    $0, %eax
    leave
    ret
```

Memory Address	value at point A
0x8880	
0x8884	
0x8888	
0x888c	
0x8890	
0x8894	
0x8898	
0x889c	
0x88a0	
0x88a4	
0x88a8	
0x88ac	
0x88b0	
0x88b4	
0x88b8	
0x88bc	
0x88c0	

Register	Initial	at point A
%eax	2	
%edx	3	
%esp	0x88b0	
%ebp	0x88c0	