

Names and lab sections:

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: a `push` grows the top of the stack and inserts the pushed value. A `pop` copies the value on top of the stack, then shrinks the stack.
- The sequence of instructions `leave; ret` is equivalent to the sequence `movl %ebp, %esp; popl %ebp; popl %eip`.

	memory address	value at point A
<code>func:</code>		
<code> pushl %ebp</code>		
<code> movl %esp, %ebp</code>	0x8880	
<code> subl \$16, %esp</code>		
<code> movl 8(%ebp), %eax</code>	0x8884	
<code> addl %eax, %eax</code>		
<code> movl %eax, -4(%ebp)</code>	0x8888	
<code> movl -4(%ebp), %eax</code>		
<code> leave # point A</code>	0x888c	
<code> ret</code>		
<code>main:</code>	0x8890	
<code> pushl %ebp</code>		
<code> movl %esp, %ebp</code>	0x8894	
<code> subl \$16, %esp</code>		
<code> movl \$6, -4(%ebp)</code>	0x8898	
<code> pushl -4(%ebp)</code>		
<code> call func</code>	0x889c	
<code> addl \$4, %esp # at addr 0x1234</code>		
<code> movl %eax, -4(%ebp)</code>	0x88a0	
<code> movl \$0, %eax</code>		
<code> leave</code>	0x88a4	
<code> ret</code>		
	0x88a8	
	0x88ac	
	0x88b0	
	0x88b4	
	0x88b8	
	0x88bc	
	0x88c0	

register	initial value	value at point A
<code>%eax</code>	2	
<code>%edx</code>	3	
<code>%esp</code>	0x88b0	
<code>%ebp</code>	0x88c0	