Question 1

Convert the following C code fragment to equivalent IA32 assembly code in two steps:
(1) First, translate the if-else to its equivalent C goto version
(2) Next, translate your C goto version to IA32, assuming that `dog` is at `Reg[%ebp] - 4`, `cat` is at `Reg[%ebp] - 8`, and `goat` is at `Reg[%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. We are showing the declarations for the variables below (assume there is other code that initializes them before the code fragment you are to translate).

```
// variable declarations
int dog, cat, goat;
// ...
// convert this fragment:
if((dog > cat)) {  // (1) C goto version
    dog = goat + cat;
    goat = cat*4;
} else if (dog > goat){
    goat = dog;
    dog = goat*2;
}
cat = goat + dog;
```

```
// IA32 Translation
// variable declarations
int dog, cat, goat;
// ...
// convert this fragment:
if((dog > cat)) {  // (1) C goto version
    dog = goat + cat;
    goat = cat*4;
} else if (dog > goat){
    goat = dog;
    dog = goat*2;
}
cat = goat + dog;
```
Question 2

Trace through the following IA32 code. Show the contents of the given memory and registers right 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 # 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
```

<table>
<thead>
<tr>
<th>Register</th>
<th>Initial</th>
<th>at A</th>
</tr>
</thead>
<tbody>
<tr>
<td>%eax</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>%edx</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>%esp</td>
<td>0x88b0</td>
<td></td>
</tr>
<tr>
<td>%ebp</td>
<td>0x88c0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>at A value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8880</td>
<td></td>
</tr>
<tr>
<td>0x8884</td>
<td></td>
</tr>
<tr>
<td>0x8888</td>
<td></td>
</tr>
<tr>
<td>0x888c</td>
<td></td>
</tr>
<tr>
<td>0x8890</td>
<td></td>
</tr>
<tr>
<td>0x8894</td>
<td></td>
</tr>
<tr>
<td>0x8898</td>
<td></td>
</tr>
<tr>
<td>0x889c</td>
<td></td>
</tr>
<tr>
<td>0x88a0</td>
<td></td>
</tr>
<tr>
<td>0x88a4</td>
<td></td>
</tr>
<tr>
<td>0x88a8</td>
<td></td>
</tr>
<tr>
<td>0x88ac</td>
<td></td>
</tr>
<tr>
<td>0x88b0</td>
<td></td>
</tr>
<tr>
<td>0x88b4</td>
<td></td>
</tr>
<tr>
<td>0x88b8</td>
<td></td>
</tr>
<tr>
<td>0x88bc</td>
<td></td>
</tr>
<tr>
<td>0x88c0</td>
<td></td>
</tr>
</tbody>
</table>