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 
-4(\texttt{r[ebp]}), cat is at -8(\texttt{r[ebp]}), and goat is at -12(\texttt{r[ebp]}).

You must show both steps (1) and (2). To receive partial credit, annotate your IA32 code with comments describing which part of the C code you are implementing.

```c
int dog, cat, goat;
dog = 12;
cat = 90;
goat = dog - cat;
while (dog < cat) {
    dog *= 2;
    goat += dog;
}
```
(You may do these on separate pages if space is a concern.)

(1) C goto version
------------------
(2) IA32 Translation
-------------------
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.
func:
  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 func
  addl $4, %esp # at 0x1234
  movl %eax, -4(%ebp)
  movl $0, %eax
  leave
  ret

<table>
<thead>
<tr>
<th>Register</th>
<th>initial value</th>
<th>value at point &quot;A&quot;</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>