1. Given the following code, answer the questions:

```assembly
main:
    .pos 0x0
    irmovl $9, %edx  ; offset/PC value = 0x0
    irmovl $21, %ebx  ; offset/PC value = 0x6
    subl %edx, %ebx   ; offset/PC value = 0xc
    irmovl $128, %esp ; offset/PC value = 0xe
    rmmovl %esp, 100(%ebx) ; offset/PC value = 0x14
    pushl %edx        ; offset/PC value = 0x1a
    popl %eax         ; offset/PC value = 0x1c
    je done           ; offset/PC value = 0x1e
    call proc         ; offset/PC value = 0x23
done:
    halt              ; offset/PC value = 0x28
proc:
    ret               ; offset/PC value = 0x29
```

Stack (if needed) could start at 0x2a + room to manage stack info

What is the instruction encoding for the irmovl $21, %ebx statement? __________________________ 0x3f0f31500000

What is the instruction encoding for the popl %eax statement? ______________________________ 0xb00f

What is the value of %ebx after the program is executed? ____________________________ 0xc

2. Given the following C code, write the Y86 code:

```c
void main() {
    // adds the numbers from 1 to 1000
    int sum = 0;
    int lim = 1000;
    for (num=1; num <= lim; num++){
        sum += num;
    }
}
```

Need to convert the condition as it compares to zero, so num<=lim

is the same as

lim-num >= 0 for true condition
lim-num < 0 for false condition

However, if do the subtract

subl %ebx,% ecx

to set the condition codes so you can test the condition and determine whether or not to jump, then the value of the limit changes. This is why you have to have a temp place for the limit when you are subtracting.

There are **multiple** ways to do this, however, will the values of sum, num and lim all be the same at the end of the loop? Also, compare with y86loop.ys in slides.
 irmovl $0,%eax  # sum = 0
 irmovl $1,%ebx  # num = 1
 irmovl $1000,%ecx  # lim = 1000 (constant)
 irmovl $1,%edx  # tmp = 1 (constant)
 rrmovl %ecx, %esi  # temp place for lim
 subl %ebx, %esi  # lim-num
 Loop:
    jl End  # check condition
    addl %ebx, %eax  # sum += num
    addl %edx, %ebx  # num++
    rrmovl %ecx, %esi  # temp place for lim
    subl %ebx, %esi  # lim-num
    jmp Loop  # loop again
End: halt

 irmovl $0,%eax  # sum = 0
 irmovl $1,%ebx  # num = 1
 irmovl $1000,%ecx  # lim = 1000 (constant)
 irmovl $1,%edx  # tmp = 1 (constant)
 rrmovl %ecx, %esi  # temp place for lim
 subl %ebx, %esi  # lim-num
 jge Loop  # check condition
End: halt