1. Given the following code, answer the questions:

```plaintext
main:
.pos 0x0
.irmovl $9, %edx 6 → offset/PC value = 0x0
.irmovl $21, %ebx 6 → offset/PC value = 0x6
.subl %edx, %ebx 2 → offset/PC value = 0xc
.irmovl $128, %esp 6 → offset/PC value = 0xe
.rmmovl %esp, 100(%ebx) 6 → offset/PC value = 0x14
.pushl %edx 2 → offset/PC value = 0x1a
.popl %eax 2 → offset/PC value = 0x1c
.je done 5 → offset/PC value = 0x1e
.call proc 5 → offset/PC value = 0x23
done:
.halt 1 → offset/PC value = 0x28
proc:
.retl 1 → 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? ______________________________ 0x3f31500000
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;
    }
}
```

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.

```text
.irmovl $0,%eax # sum = 0
.irmovl $1,%ebx # num = 1
.irmovl $1000,%ecx # lim = 1000 (constant)
.irmovl $1,%edx # tmp = 1 (constant)
.blue here too with “jl End” instead of jge Loop
Loop:
.addl %ebx,%eax # sum += num
.addl %edx,%ebx # num++
.rmmovl %ecx, %esi # temp place for lim
.subl %ebx, %esi # lim-=num
.jge Loop # loop again
End: halt
```

```text
.irmovl $0,%eax # sum = 0
.irmovl $1,%ebx # num = 1
.irmovl $1000,%ecx # lim = 1000
.irmovl $1,%edx # tmp = 1 (constant)
.blue here too with “jl End” instead of jge Loop
Loop:
.addl %ebx,%eax # sum += num
.addl %edx,%ebx # num++
.subl %edx,%ecx # lim -= 1
.jge Loop # loop again
End: halt
```