6.1100 Spring 2024 Miniquiz #4

Please submit your answers on Gradescope by March 7th, 2024, 11:59pm.

Name:

Email:

1. x86 Introduction

Ben Bitdidle is trying to find his favorite number in a list of numbers. He has started writing a x86 assembly program to do this. He has completed most of his program:

```
find_magic_num(unsigned int*, int):
    pushq   %r15
    pushq   %r14
    pushq   %r13
    pushq   %r12
    pushq   __#1__
    movl    $-1, %ebx
    __#2__  %esi, %esi
    jle     .LBB2_5
    movq    %rdi, %r14
    movl    %esi, %r15d
    xorl    %r12d, %r12d
    xorl    %r13d, %r13d
.LBB2_3:
    addl    %r12d, %r13d
    movl    (%r14,%r12,4), __#3__
    callq   popcnt(unsigned int)
    _______#4_______
    je      .LBB2_4
    incq    %r12
    cmpq    %r12, %r15
    jne     .LBB2_3
    jmp     .LBB2_5
.LBB2_4:
    movl    %r13d, %ebx
```

(Continued on the next page.)

```
    .LBB2_5:
        movl    %ebx, %eax
        popq    %rbx
        _______#5_______
        retq
```

But as you can see, some parts of his program are incomplete. He would like your help to complete it! The final program should be equivalent to the following C code:

```
int find_magic_num(unsigned int* y, int N) {
    int x = 0;
    for (int i = 0; i < N; i++) {
        x += i;
        if (popcnt(y[i]) == 4) {
            return x;
        }
    }
    return -1;
}
```

He wants his code to follow the x86 calling convention and was told that the `popcount()` also follows this convention. He also knows that **#1** should be a register, **#2** should be a x86 instruction name, **#3** should be a register, **#4** should be *one* x86 instruction, and **#5** should be *four* x86 instructions.

#1:

#2:

#3:

#4:

#5:

2. Stacks and Addressing

Excited by the success of his magic numbers program, he wants to move on to more advanced programs that use the stack. He is writing a program that is equivalent to the following C function:

```
 1    int histogram(unsigned int* x, int N) {
 2        int freq[100];
 3        for (int i = 0; i < N; i++) {
 4            if (x[i] > 98) {
 5                freq[99]++;
 6            } else {
 7                freq[x[i]]++;
 8            }
 9        }
10        return 0;
11    }
```

This is what he has written so far:

```
 1   histogram(unsigned int*, int):
 2       testl   %esi, %esi
 3       jle     .LBB3_4
 4       subq    $280, %rsp
 5       leaq    268(%rsp), %rax
 6       movl    %esi, %ecx
 7       xorl    %edx, %edx
 8   .LBB3_2:
 9       movl    ____#1____, %esi
10       cmpq    $99, %rsi
11       leaq    ____#2____, %rsi
12       cmovaeq %rax, %rsi
13       incl    (%rsi)
14       incq    %rdx
15       cmpq    %rdx, %rcx
16       jne     .LBB3_2
17       addq    ____#3____, %rsp
18   .LBB3_4:
19       xorl    %eax, %eax
20       retq
```

(Questions are on the next page.)

a. Which instruction allocates space on the stack? (Use the line numbers)

b. How is instruction on line 10 being used in the code? Which other instruction makes use of the change in the status register, and how?

c. What C code line(s) does assembly line 13 best correspond to?

d. What should he put in place of **#1**?

e. What should he put in place of **#2**? (Note: you can assume red zones)

f. What should he put in place of **#3**?