CS 240 SI Worksheet #12
Valerie Zhao
3/16/17

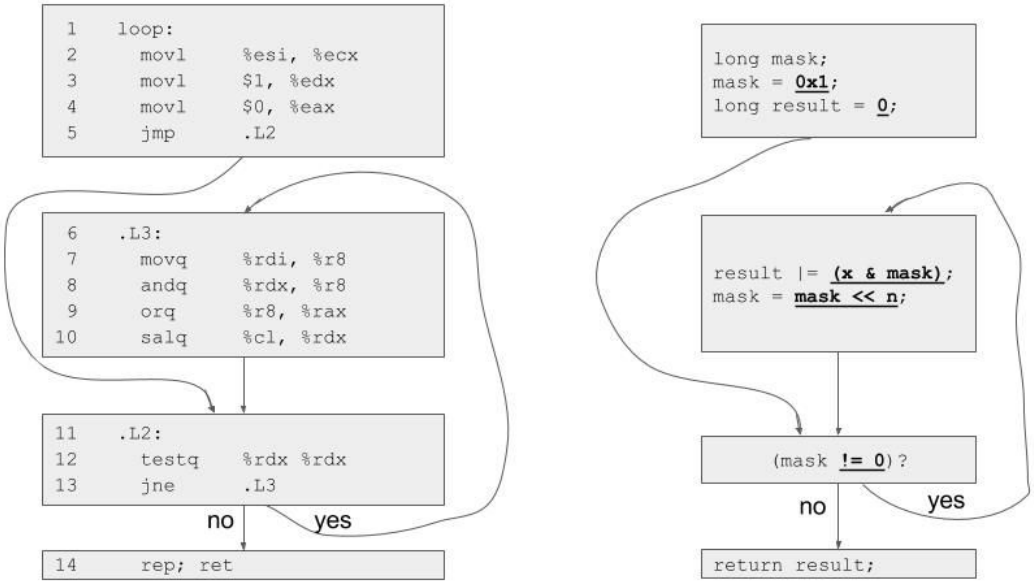
x86 (Part 1)

For the following assembly code:

```
 1    loop:
 2      movl     %esi, %ecx
 3      movl     $1, %edx
 4      movl     $0, %eax
 5      jmp      .L2
 6    .L3:
 7      movq     %rdi, %r8
 8      andq     %rdx, %r8
 9      orq      %r8, %rax
10      salq     %cl, %rdx
11    .L2:
12      testq    %rdx %rdx
13      jne      .L3
14      rep; ret                // ignore rep, it's not important
here
```

(Adapted from *Computer Systems: A Programmer's Perspective,* 3rd ed*,* Problem 3.60.)

1. Construct its control flow diagram:



2. Answer the following questions:
   a. Which registers hold program values `x`, `n`, `result`, and `mask`?
      x = %rdi n = %esi, and copied into %ecx (which %cl is a part of)
      result = %rax (which %eax is a part of)
      mask = %rdx (because it was initialized to be 1, then involved in "andq")

   b. What are the initial values of `result` and `mask`?
      result = 0 mask = 1

   c. What is the test condition for `mask`?
      if mask != 0, jump back to .L3

   d. How does `mask` get updated?
      mask is shifted left (salq) by n (%cl) each time (in other words, multiplied by 2^n each time); the for loop stops when the 1 in mask has been shifted past the most significant bit.
      Note: %cl is the rightmost byte of n, but since we're shifting something (which is stored in 64 bits) by n each time, the compiler recognizes that n must be small enough (less than 64) to be represented by 8 bits, otherwise the shifts would have unpredictable behaviors (and if n does end up being bigger than 63, then it's okay for the program to behave unpredictably).

   e. How does `result` get updated?
      result = result | (mask & x) (andq, orq, .L3)

(Adapted from *Computer Systems: A Programmer's Perspective,* 3rd ed, Problem 3.60.)

3. Fill in the C code generated by compiling the above assembly code:

```
1    long loop(long x, int n)
2    {
3          long result = 0;
4          long mask;
5          for (mask = 0x1; mask != 0; mask = mask << n) {
6               result |= (x & mask);
7          }
8          return result;
9    }
```

(Adapted from *Computer Systems: A Programmer's Perspective,* 3rd ed*,* Problem 3.60.)