Joshua Liu
105136031
CS 33
Discussion 1A

Homework 2
Problems 3.60, 3.63, 3.65

3.60

A.

Program Values	**Register**
x	%rdi
n	%esi
result	%rax
mask	%rdx

Loop long(long x, int n)
X in %rdi, n in %esi

```
loop:
        movl %esi, %ecx         set %ecx = n
        movl $1, %edx           set %edx = 1
        movl $0, %eax           set %eax = 0
        jmp .L2                 goto L2
.L3:
        movq %rdi, %r8          %r8 = x
        andq %rdx, %r8          %r8 = %r8 & %rdx
        orq %r8, %rax           %rax = %rax | %r8
        salq %cl, %rdx          %rdx = %rdx << %cl
.L2:
        testq %rdx, %rdx        if %rdx & %rdx
        jne .L3                  not equal, goto L3
        rep; ret                 return
```

B.
result = 0, in line 4
mask = 1, in line 3

C.
mask != 0, which is seen in line 12 and 13, where %rdx & %rdx sets the zero flag if it is zero and the jump statement jumps if the %rdx is not equal to zero.

D.
mask = mask << n, which is the expression for the next iteration of the for loop. This is seen in line 10.

E.
(mask & %r8) is calculated and stored in %r8. Then, (result = result | %r8) is determined. The for loop updates by shifting the mask to the left by n bits and saving that as the mask, where n is stored in %cl.

F.

```
long loop(long x, int n)
{
	long result = 0;
	long mask;
	for (mask = 1; mask != 0; mask = mask << n) {
		result |= mask & x;
	}
	return result;
}
```

3.63

We investigate the behavior at line 5, where it uses the jump table and long n to determine which case statement it goes into. The default statement is defined before this, where a comparison of jump if greater causes a jump to the default case. The individual cases are determined, and there is only a break statement if there is a “ret” instruction.

```
x in %rdi, n in %rsi
long switch_prob(long x, long n) {
	long result = x;
	switch(n) {
		case 60:
		case 62:
			result = result * 8;
			break;
		case 63:
			result = x;
			result = result >> 3;
		case 64:
			result = x;
			result = result << 4;
			result = result - x;
			x = result;
		case 65:
			x = x * x;
		default:
			result = x + 75;
			break;
	}
	return result;
}
```

We can simplify the function above:

```
x in %rdi, n in %rsi
long switch_prob(long x, long n) {
	long result = x;
```

```
        switch(n) {
                case 60:
                case 62:
                        result *= 8;
                        break;
                case 63:
                        result >>= 3;
                        break;
                case 64:
                        x = (x << 4) - x;
                case 65:
                        x *= x;
                default:
                        result = x + 75;
                        break;
        }
        return result;
}
```

3.65

A.

A[i][j] is in %rdx since it is incremented by 8 every time, and 8 is the amount of memory used by longs(line 6). In the C function, we see that in the inner loop, j is incremented by one every time, which is equivalent to incrementing by 8 bytes in memory.

B.

A[j][i] is in %rax since it is incremented 120 every time(greater than 8, seen in line 7). This means that the array has 120 bytes of memory in it, holding 120/8=15 values for each row and column.

C.

M = 15, since %rax moves 120 in memory each loop. sizeof(long) in x64 is 8 bytes, and 120/8 = 15.