Problem 3.56

1 // %esi = x

2 // %ebx = n

3 // %edi = -1, result =-1

4 // %edx = 1, mask = 1

5 //.L2

6 // %eax = %edx =mask,

7 // %eax = %esi & %eax = x &mask

8 // %edi = %eax ^ %edi = (x &mask)^ (result)

9 // %ecx = %ebx = n

10// %edx = %edx << %cl, which %cl is the low byte of %ecx = mask<<n

11// test %edx

12// if mask != 0, go to .L2 loop

13// %eax = %edi = result

1 int loop(int x, int n)

2 {

3 int result = -1;

4 int mask;

5 for (mask = 1; mask !=0 ; mask = mask<<n) {

6 result ^= x & mask ;

7 }

8 return result;

9 }

1. %esi holds x, %ebx holds n, %edi holds result, %edx holds mask
2. initial value of result is -1, initial value of mask is 1
3. mask != 0
4. left shift n
5. result = result^(x&mask)
6. as above red words.

Problem 3.62

1 //.L3

2 // %eax =(%ebx), t = A[i][j]

3 //%edx =(%esi + 4\* %ecx), A[i][j] = A[j][i],

4 //(%esi + 4\* %ecx) = %eax, A[j][i] = t, we know here %ecx is j

5// %ecx = %ecx +1, j = j +1

6//(%ebx) = %edx

7//%ebx = %ebx +52, M is 52/4 = 13 because we know the size of bytes is 4 from line3

8// compare %edi and %ecx, %edi is i

9// if %ecx < %edi, goto .L3

1. Value of M is 13
2. %edi holds i, because it campare to j and there is no change in the inner loop; %ecx holds j, because while j is number of row, %ecx \*4.

Problem 3.69

B. According to code in my 3.69A.c, result(which is going to be returned) equals the value of the tp.value, and each time after update the result, tp will point to its left one. The loop doesn’t stop until tp’s left equals null. So the function compute the most left one’s value.

Problem 4

Caller(g):

a is at %ebp+8, b is at %ebp+12

pushl %ebp //save old %ebp

movl %esp, %ebp // set %ebp as frame pointer

subl $24, %esp //allocate 24 bytes on stack

movl 12(%ebp), %eax //get b

movl 8(%ebp), %ecx //get a

movl %ecx, -4(%ebp)

movl %eax, -8(%ebp)

movl -4(%ebp), %eax //compute &a

movl %eax, (%esp) // store on stack

call f

movl -8(%ebp), %ecx //compute &b

movl %ecx, (%esp) //replace a with b, in order to pass only a argument to f, when call f

movl %eax, -12(%ebp) //put the value returned by f(b) in %ebp-12

call f

movl -12(%ebp), %ecx

addl %ecx, %eax // compute f(a)+f(b)

addl $24, %esp // %esp go back

popl %ebp

retl

Callee(f):  
 pushl %ebp

movl %esp, %ebp

pushl %eax

movl 8(%ebp), %eax //get x

leal (%eax, %eax, 2), %eax //3\*x

popl %ebp

retl