COMP.2030 LAB

recur: pushq %rbp

movq %rsp, %rbp

subq $16, %rsp

movq %rdi, -8(%rbp)

movq %rsi, -16(%rbp)

cmpq $0, -8(%rbp)

jne .L2

movl $-1, %eax

jmp .L3

.L2:

cmpq $10, -16(%rbp)

jne .L4

movl $0, %eax

jmp .L3

.L4:

movq -8(%rbp), %rax

movq (%rax), %rax

cmpq -16(%rbp), %rax

jle .L5

movq -8(%rbp), %rax

addq $8, %rax

movq (%rax), %rax

movq -16(%rbp), %rdx

movq %rdx, %rsi

movq %rax, %rdi

call recur

addl %eax, %eax

jmp .L3

.L5:

movl $1, %eax

.L3:

leave

ret

NAME \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. A recursive function, recur, is declared as

int recur(long \*x, long y).

When it is compiled into the x86 codes on the right,

complete the C code of the function recur() below.

int recur(long \*x, long y){

if (!x) return -1;

else

if (y == 10) return 0;

else {

if (x[0] > y)

return 2\*recur((long \*)x[1], y);

else

return 1;

}

}

int recur(long \*x, long y){

if (!x) return -1;

else

if (y == 10) return 0;

else {

if (x > y)

return 2\*recur((long \*)(x+1), y);

else

return 1;

}

}

1. The function swith\_prob() has the structure shown below, and it disassembled machine is shown on the right.





1 long switch\_prob(long x, long n) {

2 long result = x;

3 switch(n) {

4 case 60:

5 case 62:

6 result <<= 3;

7 break;

8 case 63:

9 result >>= 3;

10 break;

11 case 64:

12 result \*= 15;

13 /\* Fall through \*/

14 case 65:

15 result \*= result;

16 /\* Fall through \*/

17 default:

18 result += 75;

19 }

20 return result;

21 }

1. The following code transposes the elements of an MxM array, where M is a constant defined by #define. When compiled, gcc generates the assembly code for the inner loop of the function as shown on the right.

![Table

Description automatically generated]()

![Text, letter

Description automatically generated]()

1. Which register holds a pointer to array element A[i][j] ?

%rdx

1. Which register holds a pointer to array element A[j][i] ?

%rax

1. What is the value of M ?

M= 15