// Type your code here, or load an example.

int ack(int m,int n)

{

    if (m==0) return n+1;

    else if (n==0) return ack(m-1,1);

         else return ack(m-1,ack(m,n-1));

}

RISC-V (32-bits) gcc 15.2

ack(int, int):

        addi    sp,sp,-32//分配栈空间

        sw      ra,28(sp)//保存返回地址

        sw      s0,24(sp)//保存调用者栈帧（活动记录基址），建立调用链。

        sw      s1,20(sp)//保存s1寄存器

        addi    s0,sp,32//设置被调函数新栈帧

        sw      a0,-20(s0)//转储第一参数，m

        sw      a1,-24(s0)//转储第二参数，n

        lw      a5,-20(s0)//读参数m到a5寄存器

        bne     a5,zero,.L2//若m不等于0转L2

        lw      a5,-24(s0)//读n到a5

        addi    a5,a5,1 // 计算n+1到a5

        j       .L3 //无条件跳转到L3

.L2:

        lw      a5,-24(s0)//读n到a5

        bne     a5,zero,.L4//若n不等于0转L4

        lw      a5,-20(s0)//读m到a5

        addi    a5,a5,-1//计算m-1到a5

        li      a1,1//设置第二参数到a1（常数1）

        mv      a0,a5//设置第一参数到a0（m-1）

        call    ack(int, int)//递归调用ack(m-1,1)

       mv      a5,a0//保存函数返回值a0到a5

        j       .L3//无条件跳转到L3

.L4:

        lw      a5,-20(s0)//读m到a5

        addi    s1,a5,-1//计算m-1到s1

        lw      a5,-24(s0)//读n到a5

        addi    a5,a5,-1//计算n-1到a5

        mv      a1,a5//设置第二参数到a1

        lw      a0,-20(s0)//读m到第一参数a0

        call    ack(int, int)//递归调用ack(m,n-1)

       mv      a5,a0//保存函数返回值a0到a5

        mv      a1,a5//设置第二参数a1

        mv      a0,s1//设置第一参数a0(m-1)

        call    ack(int, int)//递归调用ack(m-1,ack…)

       mv      a5,a0//保存函数返回值a0到a5

        nop //空操作

.L3:

        mv      a0,a5//设置返回值a0（a5）

        lw      ra,28(sp)//恢复返回地址ra

        lw      s0,24(sp)//恢复调用者栈帧

        lw      s1,20(sp)//恢复s1寄存器

        addi    sp,sp,32//回收栈空间

        jr      ra//返回调用者

X86-64 gcc 15.2

ack(int, int):

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     DWORD PTR [rbp-4], edi

        mov     DWORD PTR [rbp-8], esi

        cmp     DWORD PTR [rbp-4], 0

        jne     .L2

        mov     eax, DWORD PTR [rbp-8]

        add     eax, 1

        jmp     .L3

.L2:

        cmp     DWORD PTR [rbp-8], 0

        jne     .L4

        mov     eax, DWORD PTR [rbp-4]

        sub     eax, 1

        mov     esi, 1

        mov     edi, eax

        call    ack(int, int)

        jmp     .L3

.L4:

        mov     eax, DWORD PTR [rbp-8]

        lea     edx, [rax-1]

        mov     eax, DWORD PTR [rbp-4]

        mov     esi, edx

        mov     edi, eax

        call    ack(int, int)

        mov     edx, eax

        mov     eax, DWORD PTR [rbp-4]

        sub     eax, 1

        mov     esi, edx

        mov     edi, eax

        call    ack(int, int)

        nop

.L3:

        leave

        ret