1. Translate the following C code to RISC-V assembly code.

Answer

```
LOOPI:
    addi x7, x0, 0       // Init i = 0
    bge x7, x5, ENDI     // While i < a
    addi x30, x10, 0     // x30 = &D
    addi x29, x0, 0      // Init j = 0
LOOPJ:
    bge x29, x6, ENDJ    // While j < b
    add x31, x7, x29     // x31 = i+j
    sd x31, 0(x30)       // D[4*j] = x31
    addi x30, x30, 32    // x30 = &D[4*(j+1)]
    addi x29, x29, 1     // j++
    jal x0, LOOPJ
ENDJ:
    addi x7, x7, 1       // i++;
    jal x0, LOOPI
ENDI:
```

2. Implement the following C code in RISC-V assembly. Hint: Remember that the stack pointer must remain aligned on a multiple of 8（原来的16 应改成 8）.

------- -------

Answer

```
// IMPORTANT! Stack pointer must reamin a multiple of 16!!!!
fib:
beq x10, x0, done // If n==0, return 0
addi x5, x0, 1
beq x10, x5, done // If n==1, return 1
addi x2, x2, -16 // Allocate 2 words of stack
space
sd x1, 0(x2) // Save the return address
sd x10, 8(x2) // Save the current n
addi x10, x10, -1 // x10 = n-1
jal x1, fib // fib(n-1)
ld x5, 8(x2) // Load old n from the stack
sd x10, 8(x2) // Push fib(n-1) onto the stack
addi x10, x5, -2 // x10 = n-2
jal x1, fib // Call fib(n-2)
ld x5, 8(x2) // x5 = fib(n-1)
add x10, x10, x5 // x10 = fib(n-1)+fib(n-2)
// Clean up:
```

```
ld x1, 0(x2) // Load saved return address
addi x2, x2, 16 // Pop two words from the stack
done:
jalr x0, x1
```