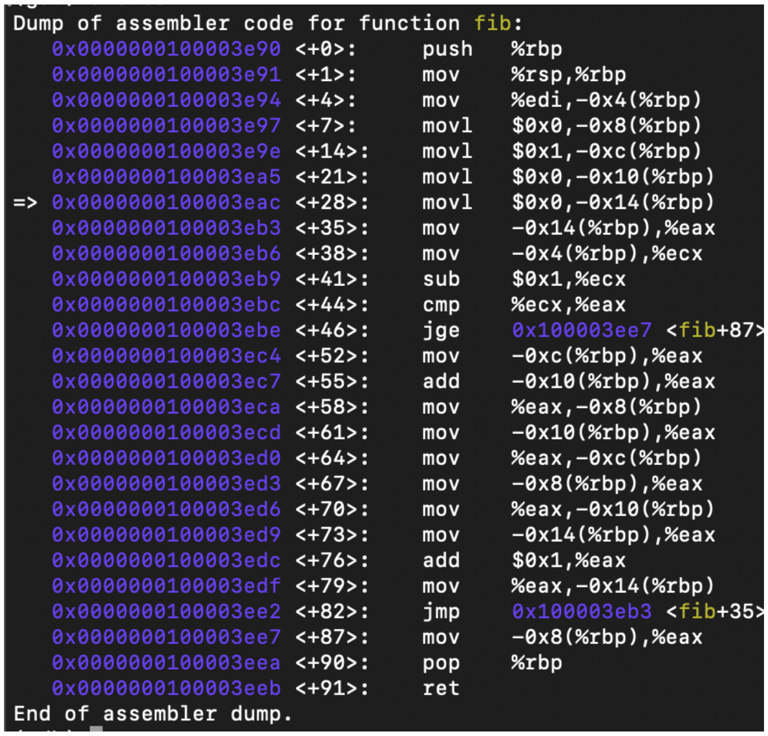
Name: Juliann Zhou NetID: kyz224



```
Dump of assembler code for function fib:
   0x0000000100003e90 <+0>:     push   %rbp
   0x0000000100003e91 <+1>:     mov    %rsp,%rbp
   0x0000000100003e94 <+4>:     mov    %edi,-0x4(%rbp)
   0x0000000100003e97 <+7>:     movl   $0x0,-0x8(%rbp)
   0x0000000100003e9e <+14>:    movl   $0x1,-0xc(%rbp)
   0x0000000100003ea5 <+21>:    movl   $0x0,-0x10(%rbp)
=> 0x0000000100003eac <+28>:    movl   $0x0,-0x14(%rbp)
   0x0000000100003eb3 <+35>:    mov    -0x14(%rbp),%eax
   0x0000000100003eb6 <+38>:    mov    -0x4(%rbp),%ecx
   0x0000000100003eb9 <+41>:    sub    $0x1,%ecx
   0x0000000100003ebc <+44>:    cmp    %ecx,%eax
   0x0000000100003ebe <+46>:    jge    0x100003ee7 <fib+87>
   0x0000000100003ec4 <+52>:    mov    -0xc(%rbp),%eax
   0x0000000100003ec7 <+55>:    add    -0x10(%rbp),%eax
   0x0000000100003eca <+58>:    mov    %eax,-0x8(%rbp)
   0x0000000100003ecd <+61>:    mov    -0x10(%rbp),%eax
   0x0000000100003ed0 <+64>:    mov    %eax,-0xc(%rbp)
   0x0000000100003ed3 <+67>:    mov    -0x8(%rbp),%eax
   0x0000000100003ed6 <+70>:    mov    %eax,-0x10(%rbp)
   0x0000000100003ed9 <+73>:    mov    -0x14(%rbp),%eax
   0x0000000100003edc <+76>:    add    $0x1,%eax
   0x0000000100003edf <+79>:    mov    %eax,-0x14(%rbp)
   0x0000000100003ee2 <+82>:    jmp    0x100003eb3 <fib+35>
   0x0000000100003ee7 <+87>:    mov    -0x8(%rbp),%eax
   0x0000000100003eea <+90>:    pop    %rbp
   0x0000000100003eeb <+91>:    ret
End of assembler dump.
```

First three lines are commands to set up the stack frame.

At line 4, we are putting the value 0 into the memory location -0x8(%rbp), which stores the variable a.

At line 5, we are putting the value 1 into the memory location -0xc(%rbp), which stores the variable b.

At line 6, we put the value 0 into the memory location -0x10(%rbp), which stores the variable c.

The following 6 lines are the for loop statement:
At line 7, we put the value 0 into memory location -0x14(%rbp), which stores the loop variable i.
At line 8, we move the value in -0x14(%rbp) (variable a) to register %eax.
At line 9, we move the value in -0x4(%rbp) to register %ecx, which is the variable n that we get from the function parameter.
At line 10, we subtract 1 from the value in %ecx, which corresponds to n-1.
At line 11, we compare the values in %ecx and %eax.
At line 12, we jump if the value in %eax is greater than %ecx, which is the conditional to break the for loop.

The following lines are inside the for loop where we perform the addition and exchanging of variables:
At line 13, we move the value stored in -0xc(%rbp) for variable b to register %eax.
At line 14, we add the value stored in -0x10(%rbp) for variable c to %eax.
At line 15, we move the value in register %eax (the sum of b and c) to -0x10(%rbp), which holds the variable a.
At line 16, we move the value in -0x10(%rbp) which stores variable c to register %eax.
At line 17, we move the value in register %eax to memory location -0xc(%rbp), which stores the variable b, therefore setting b to the value of c.
At line 18, we move the value in -0x8(%rbp), which stores the variable a to %eax.
At line 19, we move the value in register %eax to -0x10(%rbp), therefore setting variable c to the value of a.

The following line is to increment i in the for loop:

At line 20, we move the variable in memory location -0x14(%rbp) to register %eax.
At line 21, we add 1 to the value stored in register.
At line 22, we move the variable in register %eax to -0x14(%rbp), which stores the variable i.
At line 23, we perform an unconditional jump to the beginning of the for loop.

The following line is to get the result:
At line 24, we move the value in -0x8(%rbp), which stores variable a to register %eax.
At line 25, we retrieve the value stored in %rbp
At line 26, we restore main's copy from the stack.
At line 27, we return the value in the stack.