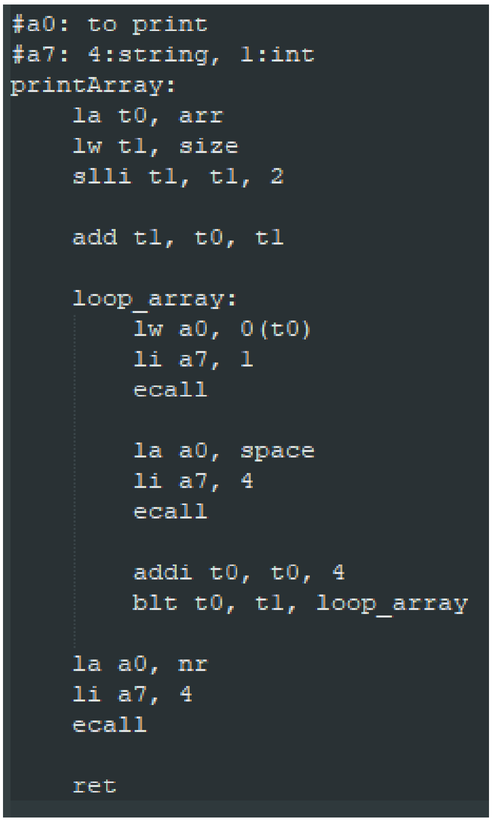
Lab01 – Report

Bubble sort

1. Instruction counts: 179

Array size=5, Array=5, 3, 6, 7, 31

printArray: 4 + loop_array * 5 + 4 = 4 + 8 * 5 + 4 = 48 instructions



```
#a0: to print
#a7: 4:string, 1:int
printArray:
    la t0, arr
    lw t1, size
    slli t1, t1, 2

    add t1, t0, t1

    loop_array:
        lw a0, 0(t0)
        li a7, 1
        ecall

        la a0, space
        li a7, 4
        ecall

        addi t0, t0, 4
        blt t0, t1, loop_array

    la a0, nr
    li a7, 4
    ecall

    ret
```

loop1(0) = loop1 + loop2(-1) + loop2e = 6 instructions

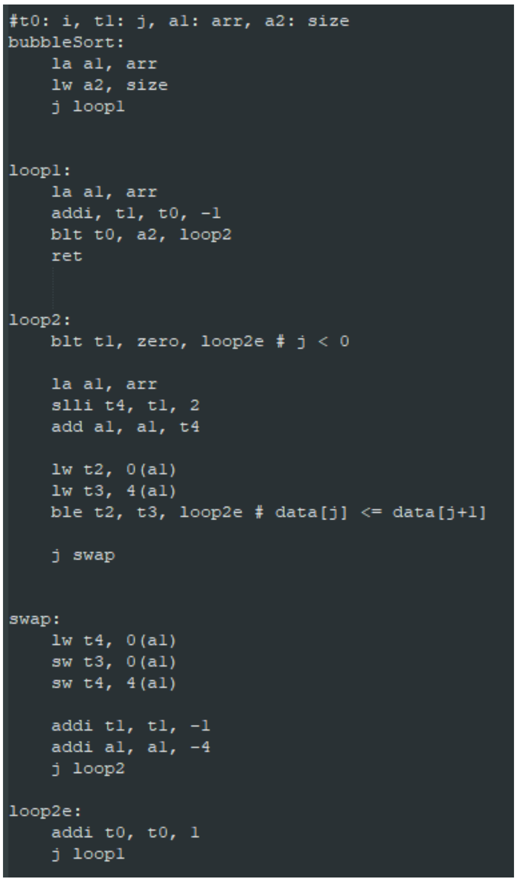
loop1(1) = loop1 + loop2(0) + swap + loop2(-1) + loop2e = 3 + 8 + 6 + 1 + 2 = 20 instructions

After loop1(1), the array is changed to 3, 5, 6, 7, 31, so don't need to swap anymore, and loop2 will always jump to loop2e after 7 instructions.

loop1(i, i = 2 ~ 4) = loop1 + loop2(i-1) + loop2e = 3 + 7 + 2 = 12 instructions

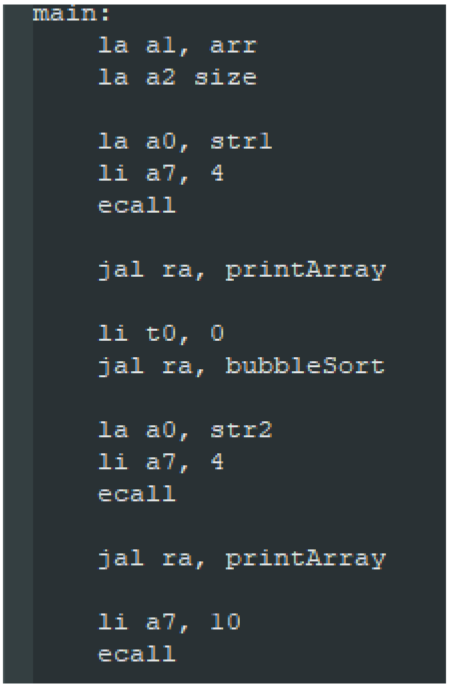
loop1(5) = 4 instructions

bubbleSort: 3 + loop1(i=0~5) = 3 + 6 + 20 + 12*3 + 4 = 69 instructions



```
#t0: i, t1: j, a1: arr, a2: size
bubbleSort:
    la a1, arr
    lw a2, size
    j loop1


loop1:
    la a1, arr
    addi, t1, t0, -1
    blt t0, a2, loop2
    ret


loop2:
    blt t1, zero, loop2e # j < 0

    la a1, arr
    slli t4, t1, 2
    add a1, a1, t4

    lw t2, 0(a1)
    lw t3, 4(a1)
    ble t2, t3, loop2e # data[j] <= data[j+1]

    j swap


swap:
    lw t4, 0(a1)
    sw t3, 0(a1)
    sw t4, 4(a1)

    addi t1, t1, -1
    addi a1, a1, -4
    j loop2

loop2e:
    addi t0, t0, 1
    j loop1
```

Total = 14 + printArray * 2 + bubbleSort = 14 + 48*2 + 69 = 179 instructions



```
main:
    la a1, arr
    la a2 size

    la a0, str1
    li a7, 4
    ecall

    jal ra, printArray

    li t0, 0
    jal ra, bubbleSort

    la a0, str2
    li a7, 4
    ecall

    jal ra, printArray

    li a7, 10
    ecall
```

2. Stack variable counts: 0

Don’t need to push variable into stack, so it’s 0 variables.

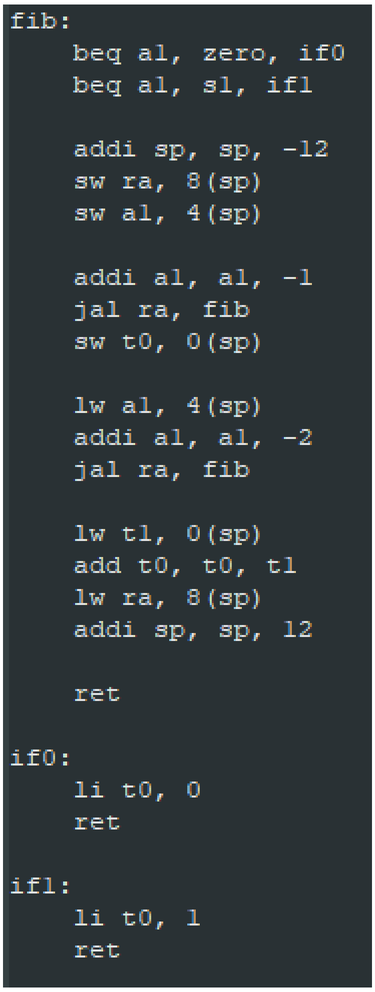
Fibonacci

1. Instruction counts: 390

fib(0) = 3 instructions

fib(1) = 4 instructions

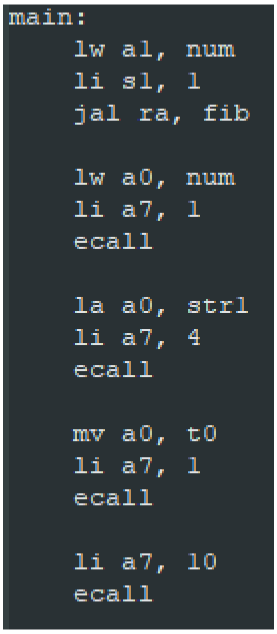
fib(i) = fib(i - 1) + fib(i - 2) + 15 instructions



```
fib:
    beq a1, zero, if0
    beq a1, s1, if1

    addi sp, sp, -12
    sw ra, 8(sp)
    sw a1, 4(sp)

    addi a1, a1, -1
    jal ra, fib
    sw t0, 0(sp)

    lw a1, 4(sp)
    addi a1, a1, -2
    jal ra, fib

    lw t1, 0(sp)
    add t0, t0, t1
    lw ra, 8(sp)
    addi sp, sp, 12

    ret

if0:
    li t0, 0
    ret

if1:
    li t0, 1
    ret
```

fib(2) = 22, fib(3) = 41, fib(4) = 78, fib(5) = 134, fib(6) = 227, fib(7) = 376 instructions

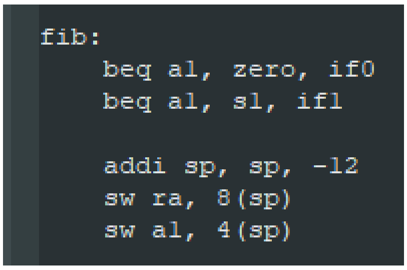
total = 14 + fib(7) = 390 instructions



```
main:
    lw a1, num
    li s1, 1
    jal ra, fib

    lw a0, num
    li a7, 1
    ecall

    la a0, str1
    li a7, 4
    ecall

    mv a0, t0
    li a7, 1
    ecall

    li a7, 10
    ecall
```

2. Stack variable counts: 24

Every time calling a fib(i > 1) function, it will push three variables, which is the ra, the input value, and one of the return value from a subsequent fib() function, into the stack.



```
fib:
    beq a1, zero, if0
    beq a1, s1, if1

    addi sp, sp, -12
    sw ra, 8(sp)
    sw a1, 4(sp)
```

fib(0) = 0 variables

fib(1) = 0 variables

fib(i) = 3 + fib(i – 1) + fib(i - 2) variables

fib(2) = 3, fib(3) = 3, fib(4) = 6, fib(5) = 9, fib(6) = 15, fib(7) = 24 variables

When calling fib(7), it will at most 24 variables in the stack at the same time.

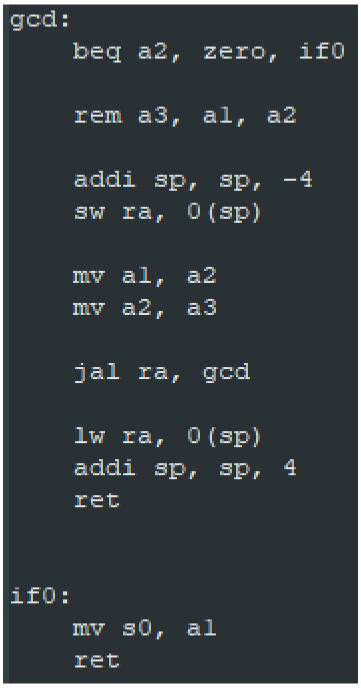
GCD

1. Instruction counts: 46

gcd(4, 0) = 3 instructions

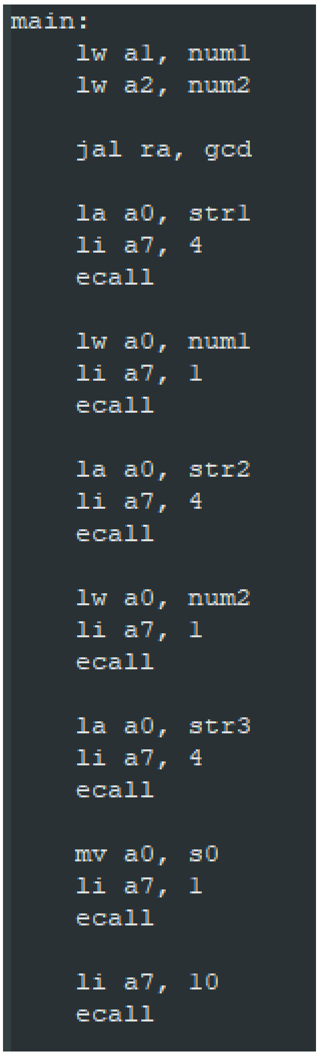
gcd(8, 4) = 10 + gcd(4, 0) = 13 instructions

gcd(4, 8) = 10 + gcd(8, 4) = 23 instructions



```
gcd:
    beq a2, zero, if0

    rem a3, a1, a2

    addi sp, sp, -4
    sw ra, 0(sp)

    mv a1, a2
    mv a2, a3

    jal ra, gcd

    lw ra, 0(sp)
    addi sp, sp, 4
    ret


if0:
    mv s0, a1
    ret
```

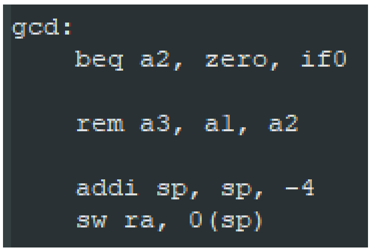
total = 23 + gcd(4, 8) = 46 instructions



```
main:
    lw a1, num1
    lw a2, num2

    jal ra, gcd

    la a0, str1
    li a7, 4
    ecall

    lw a0, num1
    li a7, 1
    ecall

    la a0, str2
    li a7, 4
    ecall

    lw a0, num2
    li a7, 1
    ecall

    la a0, str3
    li a7, 4
    ecall

    mv a0, s0
    li a7, 1
    ecall

    li a7, 10
    ecall
```

2. Stack variable counts: 2

Every time calling a gcd(m, n ≠ 0), it will push one variable into the stack, which is the ra.



```
gcd:
    beq a2, zero, if0

    rem a3, a1, a2

    addi sp, sp, -4
    sw ra, 0(sp)
```

gcd(4, 8) = 1 variable
gcd(8, 4) = 1 variable
gcd(4, 0) = 0 variable
So there are at most two variables in the stack.

Experience

At the beginning when I started working on this lab, I didn't really understand what is RISC-V, what are the meaning of the instructions, and how C code runs. After searching for information on the Internet, asking classmates, and trying on my own, I have much more understanding about the code I have written. Completing this lab from scratch teaches me a lot and gives me a great sense of achievement.