# Computer Organization Lab1: RISC-V Programming

0711239 李勝維

## Bubble\_sort:

(A) How many instructions are actually executed?

.data

argument: .word 10

data: .word 5, 3, 6, 7, 31, 23, 43, 12, 45, 1

str1: .string "Array: "

str2: .string "Sorted: "

str3: .string " "

str4: .string "\n"

Instructions count = 0

.text

main:

    addi t1, zero, 5

    # stores array to x22

    la   x22, data

    # N = 10

    lw   s0, argument

    # print "Array: \n"

    la   a1, str1

    li   a0, 4

    ecall

    la   a1, str4

    li   a0, 4

    ecall

    # print actual array

    jal  ra, printArray

Instructions count += 10

Instructions count = 10

printArray:

    li   x19, 0     # i = x19 N = s0

Instructions count += 1

Instructions count = 10+1 = 11

    loop:

    bge  x19, s0, exit

    slli x10, x19, 2

    add  x10, x10, x22

    lw   t0, 0(x10) # print Array[i]

    mv   a1, t0

    li   a0, 1

    ecall

    la   a1, str3

    li   a0, 4

    ecall

    addi x19, x19, 1

    j    loop

    exit:

    la   a1, str4

    li   a0, 4

    ecall

    ret

由於N = 10，第二行(bge)至第十三行（j loop）總共會執行10次，其中第11次時執行到第三行（bge）時會跳到exit並執行4行指令，因此：

Instructions count += 12\*10 + 1 + 4

Instructions count = 11+125 = 136

（之後可知只要N=10，進printArray就是執行126個instructions）

回到main:

# sort

    jal  ra, bubblesort

Instruction count += 1

Instruction count = 137

跳到bubblesort:

bubblesort:

    addi sp, sp, -48

    sw   x23, 0(sp)

    sw   x24, 8(sp)

    sw   t1, 16(sp)

    sw   t2, 24(sp)

    sw   x20, 32(sp)

    sw   ra, 40(sp)

    li   x23, 0   #i = x23

Instruction count += 8

Instruction count = 136+8 = 145

for1loop:

    bge  x23, s0, for1exit

    addi x24, x23, -1   #j = x24

    for2loop:

    blt  x24, zero, for2exit

    slli x10, x24, 2

    add  x10, x10, x22

    lw   t1, 0(x10)  #v[j]

    lw   t2, 4(x10)  #v[j+1]

    bge  t2, t1, for2exit

    mv   x20, x24

    jal  ra, swap

    addi x24, x24, -1

    j    for2loop

    for2exit:

    addi x23, x23, 1

    j    for1loop

    for1exit:

    lw   x23, 0(sp)

    lw   x24, 8(sp)

    lw   t1, 16(sp)

    lw   t2, 24(sp)

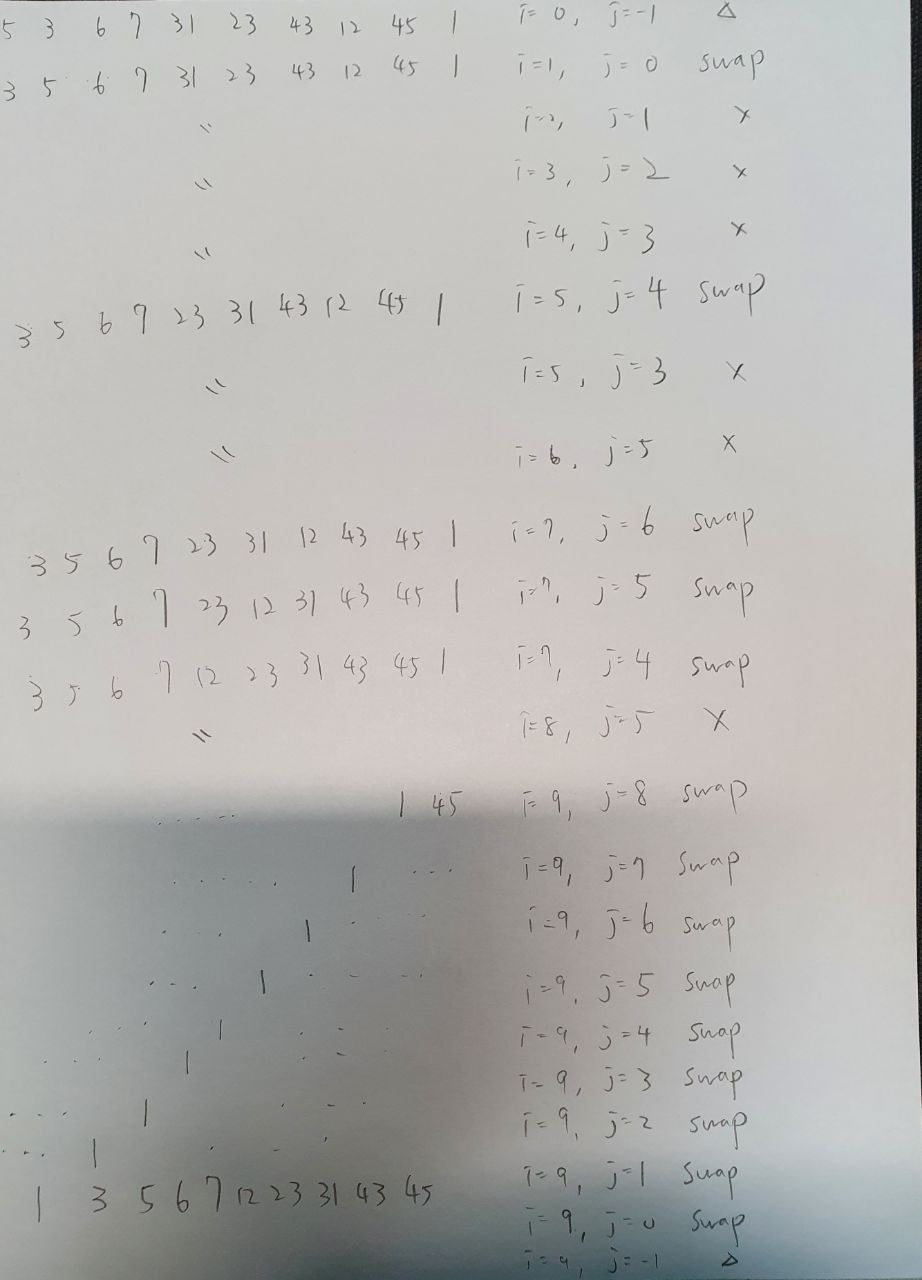
    lw   x20, 32(sp)

    lw   ra, 40(sp)

    addi sp, sp, 48

    ret

整個outer for loop (for1loop) 會執行 10次，而每個inner for loop(for2loop)最多可能會執行i次，然而有可能會因為滿足data[j] > data[j+1] (bge t2, t1, for2exit)而提早結束，因此無法用乘法直接乘，要一步一步手算，計算過程如下：



最右邊的符號，三角形代表了執行到(blt x24, zero, for2exit)就跳出inner loop了，叉叉代表了執行到(bge t2, t1, for2exit)時跳出inner loop，”swap”代表執行到swap (jal ra swap)，總共有2次三角形，6次叉叉，14次swap：

Instruction count += 4 \* 10 （for1loop 不包含 for2loop的部分）

+ 2 \* 1 （一次三角形在for2loop中只有執行一行）

+ 6 \* 6 （一次叉叉在for2loop中執行了6行）

+ 14 \* (swap + 9) （每次執行到swap包含了9行+swap本 身）

因此我們先來計算每次swap執行了幾個instruction：

swap:

    slli x10, x20, 2    #k = x20

    add  x10, x10, x22

    lw   t3, 0(x10)

    lw   t4, 4(x10)

    sw   t3, 4(x10)

    sw   t4, 0(x10)

    ret

7 行，因此上頁的算式：

Instruction count += 4\*10 + 2\*1 + 6\*6 + 14\*16

Instruction count = 145 + 302 = 447

回到main：

    #print "Sorted: \n"

    la   a1, str2

    li   a0, 4

    ecall

    la   a1, str4

    li   a0, 4

    ecall

    # print actual array

    jal  ra, printArray

    # exit

    li   a0, 10

    ecall

Instruction count += 9 + printArray，其中printArray前面算過了是126個instruction，因此：最終的Instruction count = 447 + 9 + 126 = 582

(B) What is the maximum number of variable be pushed into the stack at the same time when your code execute?

只有在bubble sort時在loop外push 6 個變數進stack，  
因此maximum number = 6

## GCD:

(A)How many instructions are actually executed?

.data

N1: .word 4

N2: .word 8

str1: .string "GCD value of "

str2: .string " and "

str3: .string " is "

Instruction count = 0

.text

main:

        lw       s0, N1

        lw       s1, N2

        jal      ra, gcd

Instruction count += 3

Instruction count = 3

跳到 gcd:

gcd:

        addi     sp, sp, -32    # 1 12 23

        sw       ra, 24(sp)     # 2 13 24

        sw       s2, 16(sp)     # 3 14 25

        sw       s1, 8(sp)      # 4 15 26

        sw       s0, 0(sp)      # 5 16 27

        mv       t0, s1         # 6 17 28

        blt      zero, t0, ngcd # 7 18 29

        mv       a0, s0         #  30 39

        addi     sp, sp, 32     #  31 40

        jalr     x0, x1, 0      #  32 41

ngcd:

        rem      s2, s0, s1     # 8 19

        mv       s0, s1         # 9 20

        mv       s1, s2         # 10 21

        jal      ra, gcd        # 11 22

        lw       s0, 0(sp)      #  33

        lw       s1, 8(sp)      #  34

        lw       s2, 16(sp)     #  35

        lw       ra, 24(sp)     #  36

        addi     sp, sp, 32     #  37

        ret                     #  38

後面的註解為instruction的順序，總共執行了39個instructions

Instruction count += 31

Instruction count = 44

回到main:

        jal      ra, printResult

Instruction count += 1

Instruction count = 45

跳到printResult：

printResult:

        mv       t0, a0

        mv       t1, a1

        la       a1, str1

        li       a0, 4

        ecall

        mv       a1, s0

        li       a0, 1

        ecall

        la       a1, str2

        li       a0, 4

        ecall

        mv       a1, s1

        li       a0, 1

        ecall

        la       a1, str3

        li       a0, 4

        ecall

        mv       a1, t0

        li       a0, 1

        ecall

        ret

Instruction count += 21

Instruction count = 66

回到main:

        li       a0, 10

        ecall

Instruction count += 2

最終的Instruction count = 68

(B) What is the maximum number of variable be pushed into the stack at the same time when your code execute?

只有以下部分code有使用到stack：

gcd:

        addi     sp, sp, -32    # 1 12 23

        sw       ra, 24(sp)     # 2 13 24

        sw       s2, 16(sp)     # 3 14 25

        sw       s1, 8(sp)      # 4 15 26

        sw       s0, 0(sp)      # 5 16 27

        mv       t0, s1         # 6 17 28

        blt      zero, t0, ngcd # 7 18 29

        mv       a0, s0         # 30 39

        addi     sp, sp, 32     # 31 40

        jalr     x0, x1, 0      # 32 41

ngcd:

        rem      s2, s0, s1     # 8 19

        mv       s0, s1         # 9 20

        mv       s1, s2         # 10 21

        jal      ra, gcd        # 11 22

        lw       s0, 0(sp)      # 33

        lw       s1, 8(sp)      # 34

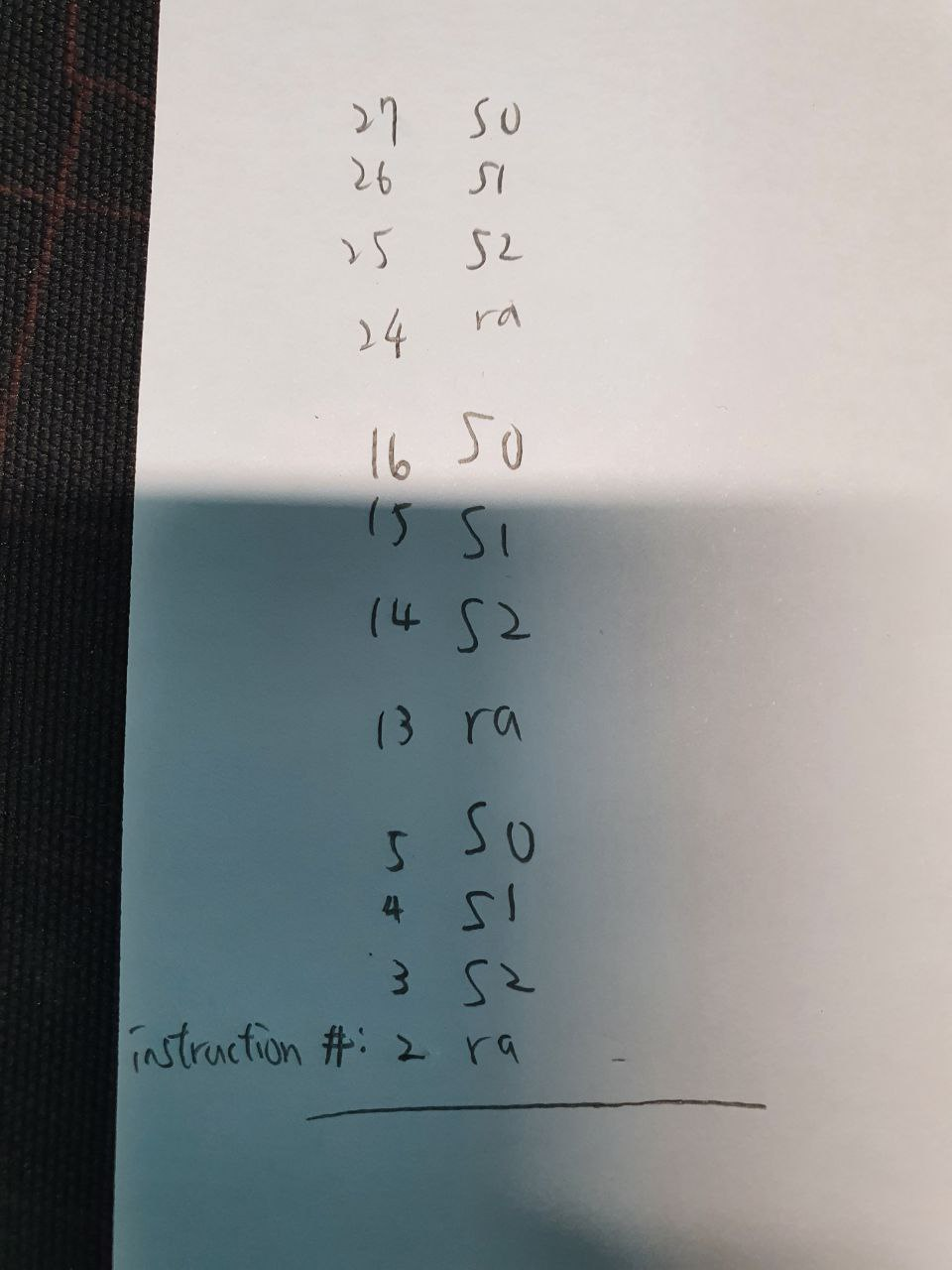
        lw       s2, 16(sp)     # 35

        lw       ra, 24(sp)     # 36

        addi     sp, sp, 32     # 37

        ret                     # 38

根據其手算的結果如下（圖代表了stack）：



左邊的數字代表了的第幾個指令會把這個變數push進去，我的答案為：maximum number = 12

## Fibonacci:

(A)How many instructions are actually executed?

.data

N: .word 7

str1: .string "th number in the Fibonacci sequence is "

str2: .string "\n"

Instruction count = 0

.text

main:

    lw s0, N

    mv s1, s0

    addi s2, zero, 1

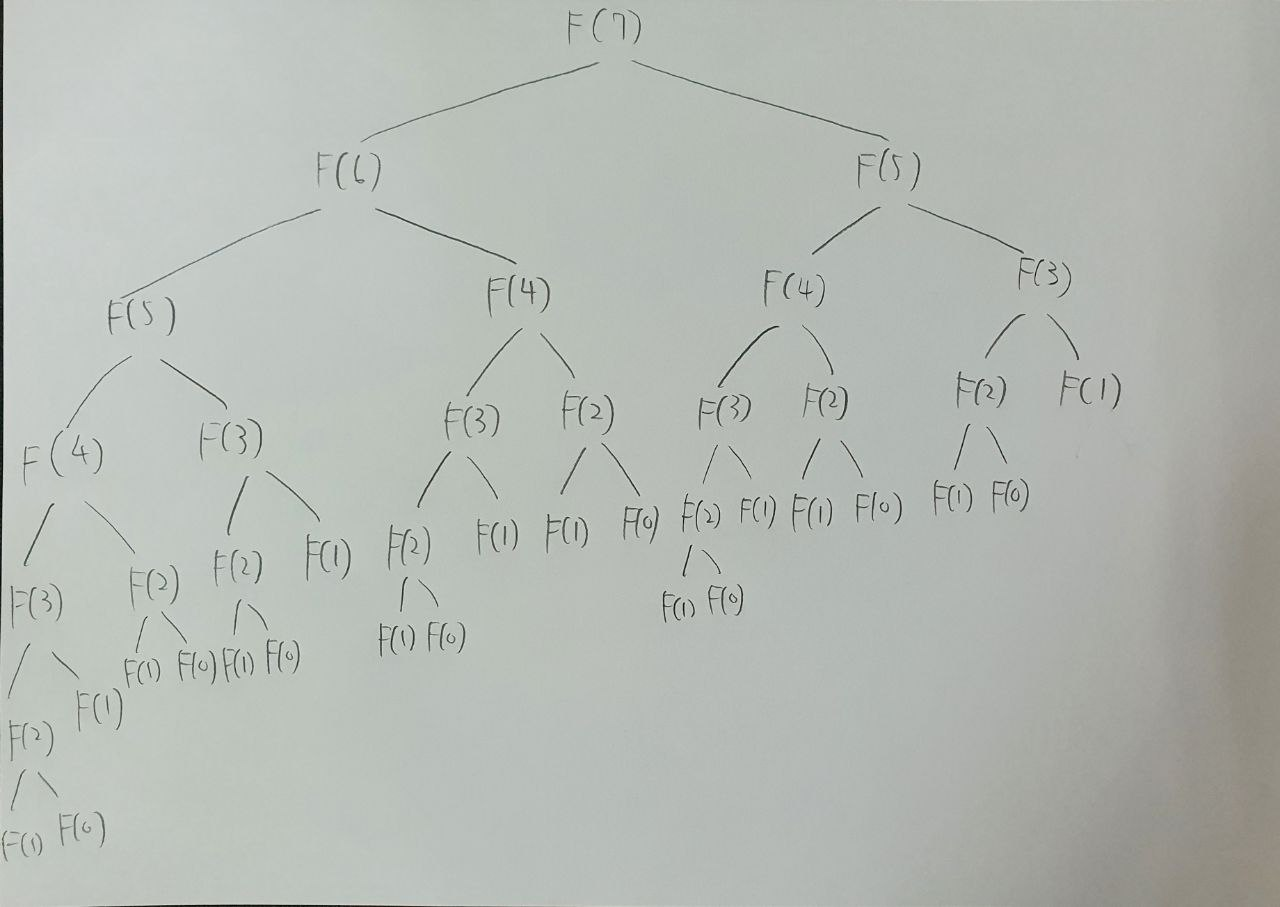
    jal ra, fibonacci

Instruction count += 4

Instruction count = 4

跳到fibonacci:

我們先整理出費氏數列的計算量，此函式的遞迴樹如下：



因此可知在此程式，Fibonacci(7)（後簡稱F(7)），中，總共會執行F(6)一次、F(5)2次、F(4)3次、F(3)5次、F(2)8次、F(1)13次、F(0)8次

再來看程式碼：

fibonacci:

    addi sp, sp, -32

    sw   ra, 24(sp)

    sw   s2, 16(sp)

    sw   s1, 8(sp)

    sw   s0, 0(sp)

    blt  s2, s1, nfibonacci

    beq  s1, s2, ro

    addi sp, sp, 32

    jalr x0, x1, 0

ro:

    addi a2, a2, 1

    addi sp, sp, 32

    ret

nfibonacci:

    addi s1, s1, -1

    jal  ra, fibonacci

    lw   s0, 0(sp)

    lw   s1, 8(sp)

    lw   s2, 16(sp)

    lw   ra, 24(sp)

    addi sp, sp, 32

    addi s1, s1, -2

    ble  zero, s1, fibonacci

    ret

有幾種case：

1. Fibonacci(2) ~ Fibonacci(7): 會首先由(addi sp, sp, -32)執行至(blt s2, s1, nfibonacci)後再執行nfibonacci，共有16個Instruction（不用考慮由nfibonacci跳回fibonacci是因為我把這次的跳躍歸類成fibonacci(n-1)的呼叫），而此種case總共會執行(1 + 1 + 2 + 3 + 5 + 8) = 20次  
   Instruction count += 16\*20  
   Instruction count = 324
2. Fibonacci(1): 會首先由(addi sp, sp, -32)執行至(beq s1, s2, ro)後再執行ro，共執行10個instruction，而此種case總共會執行13次  
   Instruction count += 13\*10  
   Instruction count = 454
3. Fibonacci(0): 直接執行完fibonacci，共9行：  
   Instruction count += 8\*9  
   Instruction count = 526

再回到main:

    mv a1, s0

    li a0, 1

    ecall

    la a1, str1

    li a0, 4

    ecall

    mv a1, a2

    li a0, 1

    ecall

    la a1, str2

    li a0, 4

    ecall

    li a0, 10

    ecall

Instruction count += 14  
最終的Instruction count = 540

(B) What is the maximum number of variable be pushed into the stack at the same time when your code execute?

每次進fibonacci，會push 4個變數進stack，但是如果有進nfibonacci，會再pop掉4個變數，但是nfibonacci會在pop stack前呼叫fibonacci，而每次的N都會少1，因此最多同時會有4 \* 7個變數在stack中（因N=7），即fibonacci -> nfibonacci ->fibonacci->…的循環會有7組，因此maximum number = 28

## Experience:

在這次的lab1中，我學到了很多關於stack的知識，原本修OS時不太明白的sp也熟練了很多，尤其是像是費氏數列這種遞迴關係式，在運作的過程中會經歷的步驟有了更多更完整的理解，而撰寫組合語言也更加的瞭解各種CPU register的功能，對我來說是一個非常好的經驗！

其中在翻譯Ripes不支援的pseudo instructions時遇到了一些困難，後來跟同學討論才成功解決，也是很寶貴的經驗。