LAB_1_report　　0716074 蔡育呈

一、作業目標：

學會以組合語言完成以前學習 c 語言寫過的 GCD、Fibonacci、bubble sort。

二、前置檔案及預備知識：

1. 下載好 Ripes
2. Ripes：圖像化運算電路運作的模擬器、RISC-V 編輯器
3. TA 給了兩個檔案：Factorial.c、Factorial.s 提供了我對照組合語言和 C 語言的範例
4. 準備開始寫囉！

三、作業程式碼與註釋 (my code with notation)

1. gcd.s

```
.data
argument1: .word 512 # Number to find the factorial value of
argument2: .word 480
str1: .string "GCD value of "
str2: .string " and "
str3: .string " is "

.text
main:
        lw      a1, argument1   # Load argument from static data
        lw      a0, argument2
        jal     ra, gcd         # Jump-and-link to the 'fact' label
        # result->a1
        # Print the result to console
        mv      a2, a1
        lw      a1, argument1
        jal     ra, printResult

        # Exit program
        li      a0, 10
        ecall

gcd:    # gcd(a0,a1)
        addi    sp, sp, -24
        sw      ra, 16(sp)
        # store data
        sw      a1, 8(sp)
        sw      a0, 0(sp)
        # if a0>0 -> call ngcd()
```

```
        bgt     a0, zero, ngcd

        addi    t0, a1, 0
        addi    sp, sp, 24
        sw      t0, 8(sp)
        ret
ngcd:
        # gcd(a0, a1)=> gcd(mod(a1,a0), a0)
        addi    t0, a0, 0
        remu    a0, a1, a0    # a0 = mod(a1, a0)
        mv      a1, t0

        jal     ra, gcd
        # 把gcd 做完 store 完的a0, a1 load 出來用
        # 為了下一次ngcd 使用

        # 每次call gcd 都會有一組ra,a0,a1
        lw      a0, 0(sp)
        lw      a1, 8(sp)
        lw      ra, 16(sp)
        addi    sp, sp, 24
        sw      a1, 8(sp)

        ret
# expects:
# a0: Value which factorial number was computed from
# a1: Factorial result
printResult:
        mv      t0, a0
        mv      t1, a1
        mv      t2, a2

        la      a1, str1
        li      a0, 4
        ecall

        mv      a1, t1
        li      a0, 1
        ecall

        la      a1, str2
        li      a0, 4
```

```
        ecall

        mv      a1, t0
        li      a0, 1
        ecall


        la      a1, str3
        li      a0, 4
        ecall

        mv      a1, t2
        li      a0, 1
        ecall

        ret
```

2. fibonacci.s

```
# This example shows an implementation of the mathematical
# fibonacci function
.data
argument: .word 10 #
str: .string "th number in the Fibonacci sequence is "

.text
main:
        lw      a0, argument   # Load argument from static data
        addi    a1, zero, 0
        jal     ra, fibo       # Jump-and-link to the 'fibo' label

        # Print the result to console
        lw      a0, argument
        jal     ra, printResult

        # Exit program
        li      a0, 10
        ecall
fibo:
        addi    sp, sp, -24
        sw      ra, 16(sp)
        sw      a0, 0(sp)
        sw      a1, 8(sp)
```

```
#if
        addi    t0, a0, -2
        bge     t0, zero, nfibo
#else if
        addi    t0, a0, -1
        beq     t0, zero, fibo_1
#else
        addi    a1, zero, 0
        sw      a1, 8(sp)
        addi    sp, sp, 24
        ret
fibo_1:
        addi    a1, zero, 1
        sw      a1, 8(sp)
        addi    sp, sp, 24
        ret
nfibo:
        addi    a0, a0, -1
        # fibo(n-1)
        jal     ra, fibo
        sw      a1, 8(sp)

        lw      a0, 0(sp)
        lw      ra, 16(sp)

        addi    sp, sp, -24
        addi    a0, a0, -2

        # fibo(n-2)
        jal     ra, fibo
        addi    sp, sp, 24
        addi    t0, a1, 0

        lw      a0, 0(sp)
        lw      a1, 8(sp)
        lw      ra, 16(sp)

        # fibo(n) = fibo(n-1) + fibo(n-2)
        add     a1, a1, t0
        sw      a1, 8(sp)
        addi    sp, sp, 24
        ret
```

```
# expects:
# a0: Value which fibonacci number was computed from
# a1: Fibonacci result
printResult:
        mv      t0, a0
        mv      t1, a1

        mv      a1, t0
        li      a0, 1
        ecall

        la      a1, str
        li      a0, 4
        ecall

        mv      a1, t1
        li      a0, 1
        ecall

        ret
```

3. bubblesort.s

```
    .data
    N:      .word   10

    arr:     .word 5,3,6,7,31,23,43,12,45,1
    str1:   .string "Array: "
    str2:   .string "Sorted: "
    str3:   .string "\n"
    space:  .string " "

    .text
    main:
            lw      t0, N               # read N to temp reg(t0->N)
        # print str1
            la      a1, str1
            li      a0, 4
            ecall
        # print str3
            la      a1, str3
            li      a0, 4
            ecall
```

```
    # read input arr to a0
       la      a0, arr

    # jump to function
       jal     ra, printArray

    # print str3
       la      a1, str3
       li      a0, 4
       ecall
    # print str2
       la      a1, str2
       li      a0, 4
       ecall

    # read input arr to a0
    # jump to function
       la      a0, arr
       jal     ra, bubblesort
    # jump to function
       jal     ra, printArray

       la      a1, str3
       li      a0, 4
       ecall

       li      a0, 10
       ecall
bubblesort:
       addi    sp, sp, -8
       # make room on stack for 2 registers (each: 4bytes)
       sw      ra, 4(sp)   # save return address on stack
       sw      a0, 0(sp)   # save a0 on stack (a0->arr)
       li      a1, 0       # init: i = 0 (a1 -> i)
       j       sort_for1   # jump to for-loop
sort_for1:  # set forloop init, step, and exit.
       bge     a1, t0, exit1  # go to exit1 if i >= n
       addi    a2, a1, -1     # j = i - 1
       j       sort_for2      # jump to for-loop
sort_for2:  # set second for-loop (a2->j)
       blt     a2, zero, exit2 # go to exit2 if j < 0
```

```
    # shift to next array element:
       slli    t1, a2, 2       # t1 = j * 4
       add     t1, a0, t1      # t1 = arr + t1
    # t1 is our new arr[]
    # always compare to arr[j] and arr[j+1]
       lw      t2, 0(t1)       # t2 = arr[j+0]
       lw      t3, 4(t1)       # t3 = arr[j+1]
       ble     t2, t3, exit2   # go to exit2 if t2 < t3

       mv      t1, a2          # swap parameter is j
       jal     ra, swap        # call swap

       addi    a2, a2, -1      # step
       j       sort_for2       # go to for2tst
exit2:
       addi    a1, a1, 1 # i += 1
       j       sort_for1 # go to for1tst
exit1:
       lw      a0, 0(sp)  # load a0 from stack
       lw      ra, 4(sp)  # load return address from stack
       addi    sp, sp, 8  # load stack pointer
       ret
swap:
       # t1 is j
       slli    t2, t1, 2   # t2 = j*4
       add     t2, t2, a0  # t2 = t2 + arr
       # t2 is our new arr
       lw      a4, 0(t2)   # load arr[j]
       lw      a3, 4(t2)   # load arr[j+1]

       sw      a3, 0(t2)   # store arr[j]
       sw      a4, 4(t2)   # store arr[j+1]
       ret

printArray:
       addi    sp, sp, -8
       sw      ra, 4(sp)
       sw      a0, 0(sp)
       li      a1, 0       # init: i=0
       j       print_for
```

```
print_for:
        bge     a1, t0, exit1   # exit if i>=N
        slli    t1, a1, 2       # t1 = i*4
        add     t1, a0, t1      # t1 = t1 + arr
        lw      t2, 0(t1)       # load arr[i]
        mv      t0, a0          # t0 = arr
        mv      t1, a1          # t1 = i*4

        mv      a1, t2          # a1 = arr[i]
        li      a0, 1           # type: INT
        ecall

        la      a1, space
        li      a0, 4           # type: STRING
        ecall

        mv      a0, t0          # a0 = arr
        mv      a1, t1          # a1 = 4*i
        lw      t0, N           # load N to t0
        addi    a1, a1, 1       # i++
        j       print_for
```

四、作業問題討論與解決方法

1. 一開始以為用上次 lab 使用的 compiler explorer 把 C 語言翻譯成 Risc-V 即可，但發現同樣是 RISC-V，但是使用的指令有很大不同，為甚麼會這樣？
   - 本次作業使用 Ripes 和 compiler explorer 的指令集不同，故無法直接複製貼上，但是記憶體存取及運算方式差不了多少，先理解 compiler explorer 上面的 code，對於 RISC-V 的學習也會有幫助。
2. Recursion 要如何實現？(分成兩部分說明)
   a. gcd
      - 建立一個 gcd function
      - gcd 裡面做判斷是否要遞迴，如果要：call ngcd()，如果不用，找出答案 return。(這邊需要用 store 更新做完 ngcd()的結果 a0,a1,ra)
      - ngcd 裡面做 gcd 的運算 gcd(a,b)=gcd(b mod a,a)，然後 call gcd()，最後把 a0,a1,ra 的值 loafid 出來(目的是讓最後一層一層 return 時最初的 a0,a1 值不改變)

   【注意】
      - 每次 call gcd()時會 sp=sp-24，這裡的空間何時拿來 store data 何時拿來 load data 需要注意。因為遞迴到初始條件回傳之後，會一層一層推回初始條件

   b. fibonacci
      - 建立一個 fibo() function
      - fibo()裡面做判斷誰麼時候要做遞迴。如果要：call n_fibo()，如果不要，return 1。

- n_fibo 裡面做 fibonacci 數列的運算(先 call fibo(n-1)，再 call fibo(n-2)，最後做運算)

3. 如何 output 出來(像是 print 和 cout 的功能)？
   - 印出的內容被存在 str1：
     la a1 str1
   - 印出的內容的 type：
     li a0 4　(1：INT；4：STRING；10：EXIT)
   - 使用者互動函式：ecall

4. 如何使用 array 中的值？
   - Int array 中一個 element 有 4 個 bytes，故當我們要取 arr[i]值的時候，只需要將 arr = arr + 4*i 即可
   - 像是 bubblesort 中每次要比較下兩個元素，每次進去 function 需要 arr = arr + 4*i，接者再 load 出(0)arr 和(4)arr 即可

5. 如何實作 for-loop？
   - bubblesort 一開始先設迴圈初始值和結束條件，沒有結束的話就 jump 到 forloop_1
   - forloop_1 沒有結束的話 call forloop_2
   - 結束條件：exit_1、exit_2

五、心得：

本次編寫 RISC-V 花了我不少時間，尤其在 debug 一步一步的觀察 stack 和 register 的數值變化時，我的眼睛和腦袋都快要炸掉了，但大部分的錯誤都是在於 register 和 memory 間的 load 和 store，希望經由本次作業，之後在編寫組合語言時，能更有系統的 coding，這次學到了很多，感謝用心的老師和 TA 們！