1、Fibonacci.s

How many instructions are actually executed? You have to explain clearly how you calculate your instructions.

There are 136 instructions which are actually executed.

```
.data
argument: .word   7
str1:     .string "th number in the Fibonacci sequence is "
.text
main:
        lw  a0, argument   # Load argument from static data
        jal ra, Fibonacci       # Jump-and-link to the 'fact' label

        # Print the result to console
        mv  a1, a0
        lw  a0, argument
        jal ra, printResult

        # Exit program
        li a7, 10
        ecall

Fibonacci:
        addi sp, sp, -16
        sw   ra, 8(sp)
        sw   a0, 0(sp)
        addi t0, a0, -1
        bge  t0, zero, nFibonacci1

        addi a0, zero, 0
        addi sp, sp, 16
        jr x1

nFibonacci1:
        addi t1, a0, -2
        bge  t1,zero,nFibonacci2

        addi a0,zero,1
        addi sp, sp, 16
        jr x1

nFibonacci2:
       addi a0, a0, -1
       jal ra,Fibonacci
       addi t1,t2, 0
       addi t2,a0, 0
       lw  a0, 0(sp)
       lw  ra, 8(sp)
       addi sp, sp, 16
       add  a0,t1,t2
       ret

# --- printResult ---
# a0: Value which factorial number was computed from
# a1: Factorial result
printResult:
        mv t1,a1
        li a7, 1
        ecall
       # mv t0, a0
       # mv t1, a1
        la a0, str1
        li a7, 4
        ecall
        mv a0, t1
        li a7, 1
        ecall
        ret
```

首先，將 argument 存入 a0，進入 Fibonacci 的 label，(接著在 Fibonacci 中

進行 sp 的移動和 ra、a0 的存入，接著將 a0-1 存入 t0，經過和 0 比較後進入 nFibonacci1，t0 再減 1，並且在和 0 比較後進入 nFibonacci2，將 a0 減一後再次進入 Fibonacci)，括號內的內容會進行六次，之後會在 Fibonacci 中進行 sp 的移動和 ra、a0 的存入，接著將 a0-1 存入 t0，經過和 0 比較後進入 nFibonacci1，t0 減一後，將 a0 存為 1，sp 移動，接著 return 回 Fibonnacci，a0 存為 0，移動 sp，進入 nFibonacci2，(t1 存上一次結果的 a0，t2 存新的 a0，接著，將 a0 和 ra 讀出，移動 sp，t1 和 t2 相加)，括號內進行五次，return 回 main，進入 printResult 中，將存在 a1 的 argument 印出，str1 的位址讀入 a0，並印出，接著讀入暫存於 t1 的結果並印出，最後執行離開 program。

What is the maximum number of variable be pushed into the stack at the same time when your code execute?
The maximum number of variable be pushed into the stack at the same time is 14.

2、gcd.s

How many instructions are actually executed? You have to explain clearly how you calculate your instructions.
There are 79 instructions which are actually executed.

```
.data
argument1: .word   4
argument2: .word   8
str1:      .string "GCD value of "
str2:      .string " and "
str3:      .string " is "

.text
main:
        lw  a0, argument1
        lw  a1, argument2
        jal ra, gcd

        # Print the result to console
        mv  a2, a0
        lw  a0, argument1
        lw  a1, argument2
        jal ra, printResult

        # Exit program
        li a7, 10
        ecall
```

```
gcd:
        addi sp, sp, -24
        sw   ra, 16(sp)
        sw   a1, 8(sp)
        sw   a0, 0(sp)
        addi t0, a1, -1
        bge  t0, zero, ngcd

        add a0, zero, a0
        addi sp, sp, 24
        jr x1

ngcd:
        addi  t1,a1, 0
        rem  a1,a0, t1
        addi  a0,t1, 0
        jal  ra, gcd

  #     lw   a0, 0(sp)
  #     lw   a1, 8(sp)
        lw   ra, 16(sp)
        addi sp, sp, 24
        ret
```

```
printResult:
        mv t0, a0
        mv t1, a1
        mv t2, a2
        la a0, str1
        li a7, 4
        ecall
        mv a0, t0
        li a7, 1
        ecall
        la a0, str2
        li a7, 4
        ecall
        mv a0, t1
      # lw a0, argument2
        li a7, 1
        ecall
        la a0, str3
        li a7, 4
        ecall
        mv a0, t2
        li a7, 1
        ecall
        ret
```

首先將 N1 和 N2 作為 argument1 和 argument2 存入，接著進入 gcd，移動 sp，並且存入 ra、a1、a0，將 a1 存入 t0 並減一，經過和 0 的比較後進入 ngcd，將 a1 的數值暫存於 t1，將 a0 和 t1 進行餘數運算，將 t1 的數值存入 a0，接著進入 gcd，移動 sp，並且存入 ra、a1、a0，將 a1 存入 t0 並減一，經過和 0 的比較後，將 a0 加上 0，移動 sp 後，進入 ngcd，將 ra 取出，移動 sp 後，回到 main，將 a0 的值移動到 a2，取出 argument1、argument2 後，進入 printResult，將 a0 暫存於 t0，a1 暫存於 t1，a2 暫存於 t2，印出 str1，將 t0 移動至 a0 後印出 argument1，印出 str2，印出 argument2，再印出 str3 和存於 t2 的結果，接著 return 回 main，執行離開 program。

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

The maximum number of variable be pushed into the stack at the same time is 9.

3.bubble_sort.s

How many instructions are actually executed? You have to explain clearly how you calculate your instructions.

There are 836 instructions which are actually executed.(以題目最原始N1=10 的狀況計算)

```
.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 "

.text
main:
        lw  s0, argument
        la  s1, data

        # Print the result to console
        la  a0, str1
        li  a7, 4
        ecall
        la  a0, str4
        li  a7, 4
        ecall

        jal ra, printArray

        jal ra, bubblesort

        la  a0, str2
        li  a7, 4
        ecall
        la  a0, str4
        li  a7, 4
        ecall

        jal ra, printArray
        # Exit program
        li a7, 10
        ecall
```

```
bubblesort:
        addi sp, sp, -48
        sw   ra, 40(sp)
        sw   t2, 32(sp)
        sw   t1, 24(sp)
        sw   s4, 16(sp)
        sw   s3, 8(sp)
        sw   s2, 0(sp)
        li   s2, 0
        forloop1:
            bge s2,s0, forloopexit1
            addi s3,s2, -1

        forloop2:
            blt s3,zero, forloopexit2
            slli a0,s3,2
            add  a0,a0, s1
            lw  t1,0(a0)
            lw  t2,4(a0)
            bge t2,t1, forloopexit2
            mv  s4,s3
            jal ra, swap
            addi s3,s3, -1
            j forloop2
```

```
65
66          forloopexit2:
67              addi s2,s2, 1
68              j forloop1
69
70          forloopexit1:
71              lw  s2,0(sp)
72              lw  s3, 8(sp)
73              lw  s4, 16(sp)
74              lw  t1, 24(sp)
75              lw  t2, 32(sp)
76              lw  ra, 40(sp)
77              addi sp,sp, 48
78              ret

80  swap:
81          slli a0,s4, 2
82          add a0,a0, s1
83          lw  t3,0(a0)
84          lw  t4,4(a0)
85          sw  t4,0(a0)
86          sw  t3,4(a0)
87          ret

92  printArray:
93      li s5, 0
94      loop:
95        bge s5,s0, printexit
96        slli a0,s5, 2
97        add  a0,a0, s1
98        lw  t0, 0(a0)
99        mv  a0,t0
00        li  a7, 1
01        ecall
02        la  a0,str3
03        li  a7,4
04        ecall
05        addi s5,s5,1
06        j loop
07      printexit:
08        la  a0, str4
09        li  a7, 4
10        ecall
11        ret
```

首先，將 N1 以及 data 的位址，分別存入 s0 和 s1，接著印出 sttr1 和換行(\n)，進入 printArray，指定 s5 為 0 後，進入內部的 loop，(將 s5 和 s0 進行比較，a0 存入 s5*4，將 a0 加上 s1 後存入 a0，t0 存入 a0 的值，接著印出 t0，印出 str3，將 s5 加 1)，括號進行十次，進入 printexit，印出換行後，回到 main，進入 bubbulsort，移動 sp 後，分別放入 ra、t2、t1、s4、s3、s2 的變數，進入內部的 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，將 s3 的值移動至 s4，並且進入 swap，將 a0 存入 s4*4，a0 再存為 a0 加上 s1，將 t3、t4 的值取出，並互換再存入，接著將 s3 減 1，並且再進入 forloop2，s3 和 0 進行比較後，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，

s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，進入 forloopexit2，將 s2 加 1，進入 forloop1， s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，將 s3 的值移動至 s4，並且進入 swap，將 a0 存入 s4*4，a0 再存為 a0 加上 s1，將 t3、t4 的值取出，並互換再存入，接著將 s3 減 1，並且再進入 forloop2，s3 和 0 進行比較後，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，將 s3 的值移動至 s4，並且進入 swap，將 a0 存入 s4*4，a0 再存為 a0 加上 s1，將 t3、t4 的值取出，並互換再存入，接著將 s3 減 1，並且再進入 forloop2，s3 和 0 進行比較後，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，將 s3 的值移動至 s4，並且進入 swap，將 a0 存入 s4*4，a0 再存為 a0 加上 s1，將 t3、t4 的值取出，並互換再

存入，接著將 s3 減 1，並且再進入 forloop2，s3 和 0 進行比較後，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，將 s3 的值移動至 s4，並且進入 swap，將 a0 存入 s4*4，a0 再存為 a0 加上 s1，將 t3、t4 的值取出，並互換再存入，接著將 s3 減 1，並且再進入 forloop2，s3 和 0 進行比較後，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，s3 存為 s2 減 1，(進入內部的 forloop2，s3 和 0 進行比較，a0 存為 s3*4，a0 加上 s1，將 t1、t2 自 a0 和 a0 移動 4bytes 的地方取出並各自存入，將 t1 和 t2 進行比較，將 s3 的值移動至 s4，並且進入 swap，將 a0 存入 s4*4，a0 再存為 a0 加上 s1，將 t3、t4 的值取出，並互換再存入，接著將 s3 減 1，並且再進入 forloop2)括號內重複九次，s3 和 0 進行比較後，進入 forloopexit2，將 s2 加 1，進入 forloop1，s2 和 s0 進行比較，進入 forloopexit1，將所有變數取出，移動 sp，印出 str2 和換行，進入 printArray，指定 s5 為 0 後，進入內部的 loop，(將 s5 和 s0 進行比較，a0 存入 s5*4，將 a0 加上 s1 後存入 a0，t0 存入 a0 的值，接著印出 t0，印出 str3，將 s5 加 1)，括號進行十次，進入 printexit，印出換行後，回到 main，最後執行離開 program。

4、Experience

對於組語超級不熟悉，一開始就花很多時間看懂 fatorial.c 和 factorial.s 之間的轉換，光是要把判斷式拆開來寫的這個邏輯就釐清了很久，gcd 分別儲存兩個餘數的方法也摸索很久，尤其是 bubble_sort 的迴圈，因為不知道該怎麼表達 i 和 j，著實花了很多時間和精力，不過經過實作，在過程中也不斷複習老師上課的內容，受益良多，也幸好有聰明又熱心的同學幫助我。