計算機組織 Lab1: RISC-V Programming Report

109550025 謝翔丞

1. Bubble_sort

1-1. How many instructions are actually executed?

這題因為助教很慷慨地給予我們縮減 array，所以我將長度變為 5，取原先的前五個值也就是:5, 3, 6, 7, 31。也因此從原本預估一千多道指令縮減為 206 道。

計算指令的方法非常簡單，我根據寫的組合語言，一行一行的假裝自己是電腦跟著走，然後仿照助教給予的 count 檔案將該步數寫在對應的 code 後方(如下圖所示)，從讀入 array 接著輪流印出，然後進入複雜的 sorting 再存入 array 最後印出排序好的結果，一步一步照著我寫扣的邏輯跟著指令走並且註記在該行程式碼後方。

```
    ecall                                                                   206


bubble:
    #first loop
    #if i > num of num then ready to print
    bge t1,t2,ready_end         68          98      110     122     134     146
    #j = i-1
    addi t3,t1,-1               69          99      111     123     135

bubble_calcu:
    # second loop == j for loop

    # if j < 0 then out
    blt t3,zero,add_i           70      88      100     112     124     136
    #num = 4
    addi s2,zero,4              71      89      101     113     125     137
    #let s5 be 4*j to point the array
    mul s2,s2,t3                72      90      102     114     126     138
    add s2,t0,s2                73      91      103     115     127     139

    lw s3,0(s2)                 74      92      104     116     128     140
    lw s4,4(s2)                 75      93      105     117     129     141

    #if s3 <= s4 then out to add_i
    ble s3,s4,add_i             76      94      106     118     130     142
    j swap                      77      95      107     119     131     143
```

```
.data
n: .word 5
str1: .string "Array:"
str2: .string "Sorted:"
space: .string " "
array: .word 5, 3, 6 , 7, 31
changeline: .string "\n"

.text
main:
    la a0,str1                  1
    li a7,4                     2
    ecall                       3
    addi t4,zero,1              4

    la t0,array                 5
    lw t2,n                     6
    jal ra,print_array          7


    la t0,array                 62
    lw t2,n                     63
    #for i , i is t1
    addi t1,zero,0              64
    #for j , j is t3,j = i-1
    addi t3,t1,-1               65
    addi s5,zero,1              66
```

```
    #i++
    addi t1,t1,1                96      108     120     132     144
    j bubble                    97      109     121     133     145

ready_end:
    la a0,str2                                                      147
    li a7,4                                                         148
    ecall                                                           149

    jal ra,print_array                                              150

    li a7,10
    ecall
```

```
swap:
    #num = 4
    addi s2,zero,4              78
    #Let s5 be 4*j to point the array
    mul s2,s2,t3                79
    add s2,t0,s2                80

    lw s3,0(s2)                 81
    lw s4,4(s2)                 82

    sw s3,4(s2)                 83
    sw s4,0(s2)                 84

ret                             85
```

```
print_array:

    la a0,space                 8       18      28      38      48          151     161     171     181     191
    li a7,4                     9       19      29      39      49          152     162     172     182     192
    ecall                       10      20      30      40      50          153     163     173     183     193

    li a7,1                     11      21      31      41      51          154     164     174     184     194
    lw t1,0(t0)                 12      22      32      42      52          155     165     175     185     195
    mv a0,t1                    13      23      33      43      53          156     166     176     186     196

    ecall                       14      24      34      44      54          157     167     177     187     197

    addi t0,t0,4                15      25      35      45      55          158     168     178     188     198
    addi t2,t2,-1               16      26      36      46      56          159     169     179     189     199

    bne t2,zero,print_array     17      27      37      47      57          160     170     180     190     200

    la a0,changeline                                    58                                                  201
    li a7,4                                             59                                                  202
    ecall                                               60                                                  203

    ret                                                 61                                                  204
```

1-2. What is the maximum number of variable be pushed into the stack at the same time when yourcode execute?

這小題慷慨的助教說可以不用寫。

2. Gcd

2-1. How many instructions are actually executed?

這題經過我的計算我使用了 73 道指令。

計算指令的方法也非常簡單，我根據寫的組合語言，一行一行的又一次假裝自己是電腦跟著走，然後仿照助教給予的 count 檔案將該步數寫在對應的 code 後方(如下圖所示)，從分別讀入兩個數字接著輪流印出，然後進入比較程序再分別更改其值，最後印出結果，一步一步照著我寫扣的邏輯跟著指令走並且註記在該行程式碼後方。

```
.data
num1: .word 4
num2: .word 8
str1: .string "GCD value of "
str2: .string " and "
str3: .string " is "

.text
main:

    lw      a0, num1                    1
    lw      a1, num2                    2
    jal     ra, gcd                     3


    mv      a2, a0                      46
    lw      a0, num1                    47
    lw      a1, num2                    48
    jal     ra, print                   49

    li      a7,10                       72
    ecall                               73
```

```
gcd:
    addi    sp, sp, -24                 4   14  24

    sw      ra, 16(sp)                  5   15  25
    sw      a1, 8(sp)                   6   16  26
    sw      a0, 0(sp)                   7   17  27

    # to = i , do i for loop
    addi    t0, a1, -1                  8   18  28
    # if i still >= 0 , them keep doing calcu
    bge     t0, zero, ngcd              9   19  29

    addi        sp, sp, 24                      30
    ret                                         31

ngcd:
    # t1 = a0 % a1
    rem     t1, a0, a1                  10  20

    mv      a0,a1                       11  21
    mv      a1,t1                       12  22
    jal     ra, gcd                     13  23

    mv      t2,a0                       39  32
    lw      a0, 0(sp)                   40  33
    lw      a1, 8(sp)                   41  34
    lw      ra, 16(sp)                  42  35

    addi        sp, sp, 24              43  36
```

```
print:
        mv      t0, a0                  50
        mv      t1, a1                  51
        mv      t2, a2                  52

        la      a0, str1                53
        li      a7, 4                   54
        ecall                           55

        mv      a0, t0                  56
        li      a7, 1                   57
        ecall                           58

        la      a0, str2                59
        li      a7, 4                   60
        ecall                           61

        mv      a0, t1                  62
        li      a7, 1                   63
        ecall                           64

        la      a0, str3                65
        li      a7, 4                   66
        ecall                           67

        mv      a0, t2                  68
        li      a7, 1                   69
        ecall                           70
```

2-2. What is the maximum number of variable be pushed into the stack at the same time when yourcode execute?

這題經過我的判斷和計算，需要用到3層 * 3個 (a0,a1,ra)總共9個 variable在stack裡面。

3. Fibonacci

3-1. How many instructions are actually executed?

這題經過我縝密的計算，我使用了 133 個指令。

我根據寫的組合語言，一行一行的再一次假裝自己是電腦跟著走，然後仿照助教給予的 count 檔案將該步數寫在對應的 code 後方（如下圖所示），從 n=7 開始進入 fibo 函示，然後隨著 code、開 sp 空間、減少 n 等等，以及算到後面開始 return 回來，這些都沒有一絲遺漏，最後印出結果，一步一步照著我寫扣的邏輯跟著指令走並且註記在該行程式碼後方。

```
.data
N:    .word 7
str1:    .string "th number in the Fibonacci sequence is "
.text
main:
       lw    a0, N                  1
       li    s5, 1                  2
       jal   ra, fibo               3

       mv    a1, a0                 117
       lw    a0, N                  118
       jal   ra, print              119

       li a7,10                     132
       ecall                        133

fibo:
       #sp  8  8  8
       #sp n-1 n ra

       #if n <=1 then return
       ble   a0, s5, return         4     10    16    22    28    34    40    50    60    70    80    90    100   110

       addi  sp, sp, -24            5     11    17    23    29    35    41

       sw    ra, 16(sp)             6     12    18    24    30    36    42
       sw    a0, 8(sp)              7     13    19    25    31    37    43

       addi  a0, a0, -1             8     14    20    26    32    38    44
       jal   ra, fibo               9     15    21    27    33    39    45

       #save f(n-1)
       sw    a0, 0(sp)                                                  46

       #from 8(which is n)get n-2
       lw    a0, 8(sp)              107   97    87    77    67    57    47
       addi  a0, a0, -2             108   98    88    78    68    58    48
       jal   ra, fibo               109   99    89    79    69    59    49
       #get f(n-1) and do f(n)=f(n-1)+f(n-2)
       lw    t0, 0(sp)              112   102   92    82    72    62    52
       add   a0, a0, t0             113   103   93    83    73    63    53

       lw    ra, 16(sp)             114   104   94    84    74    64    54
       addi  sp, sp, 24             115   105   95    85    75    65    55
       ret                          116   106   96    86    76    66    56

return:
       ret                                  111   101   91    81    71    61    51

print:
       mv    t0, a0                 120
       mv    t1, a1                 121

       mv    a0, t0                 122
       li    a7, 1                  123
       ecall                        124

       la    a0, str1               125
       li    a7, 4                  126
       ecall                        127

       mv    a0, t1                 128
       li    a7, 1                  129
       ecall                        130
       ret                          131
```

3-2. What is the maximum number of variable be pushed into the stack at the same time when yourcode execute?

這題經過我的判斷和計算，需要用到 7層 * 3個(a0,ra) 總共21個variable在stack裡面，但是當n=1時只需要存一個argument 所以推測應該是21-1=20。

Experience:

這次的lab其實對我個人而言是滿大的挑戰，畢竟之前沒有接觸過組合語言，要從無到生出一份可以執行的code本身就是一件不容易的事情，而這次又有兩份的遞迴要寫更是難上加難，因為遞迴本身就不是學程式語言的基礎入門課程。

先從我寫的第一份code講起，我在讀完範例code之後第一個先從fibonacci下手。一開始我沒注意到.c檔是用遞迴的方式，因此我很直覺地用兩個暫存器直接痛快地爆加到7，然後解出來答案也對，所以就爽快地進入下一題。沒想到過幾天發現要照.c檔的方式寫遞迴，導致我原本寫好的count版本與report都要大改，真是有驚無險。

第二份則是gcd，有了第一份的經驗之後就比較容易上手，這次個人比較深刻的印象是我一開始一直找不到取餘數的指令，所以自己多寫了一個用來把兩個數扣到底的函式。雖然有效的找出餘數，但是因為我沒有使用sp，導致我ret回去的位址都十分詭異，因此雖然有印出正確答案，但是他會不斷印出來、無法停止，後來是利用stack pointer然後存取ra才成功救回我的code。並且找到rem來取代落落長又沒效率的自創函式，減少不少的指令數。

最後寫出來的是bubble_sort，我花了很多天在處理這道題目，主要是在思考怎麼存一整串array進去，後來才想到放成一整個word然後根據第幾個把他*4去找出對應的位置。

另外比較好笑的是，我在2.2.4版本上面跑bubble_sort結果跑了快一分種還沒結束我就以為我寫錯讓他死掉了，所以一直中斷並尋找我認為可能出現錯誤的地方。最後是我丟著讓他跑、先去做別的事，才發現他有結束的一天。

總結，這次除了寫code以外，最辛苦的地方是計算instruction數量，因為我的作法是當作自己是電腦、一行一行去走，有時候快算完才發現在很初期的地方有遺漏，整個就要重來 十分崩潰，也幸好bubble_sort那題可以縮短array長度，德政!否則我大概處理算錯指令數量　的時間會比debug久。

這次lab下來，感覺對組語有基本的認識，也很感謝助教幾乎隨時都在線上幫我們解決問題，疑惑都能在短時間內得到很完整的答覆，辛苦了!

109550025 謝翔丞