109550031 李旻融

Report

fibonacci

Q1: How many instructions are actually executed?

Ans: 共 107 個指令，在 main 中的指令各跑一次(6)，Fabonacci 和 nFabonacci 會一直互相遞迴，Fabonacci 前五個指令會跑七次，最後一次才會跑最後兩個指令進而進行 return(5*7+2=37)，nFabonacci 的一開始會先執行前兩個指令，等 Fabonacci return 回來之後每一層都會將指令執行完畢，所以九個指令皆會執行六次(9*6=54)，Print 則是有十個指令(10)、不重複，總共是 6+37+54+10=107 個指令。

```
.data
argument: .word   7
str1:     .string "th number in the Fibonacci sequence is "

.text
main:
        lw  a0, argument                 1
        jal ra, Fabonacci                2

        mv  a1, a0                      94
        jal ra, Print                   95

        li a7, 10                      106
        ecall                          107

Fabonacci:
        addi sp, sp, -16                 3  10  17  24  31  38  45
        sw   a0, 0(sp)                   4  11  18  25  32  39  46
        sw   ra, 8(sp)                   5  12  19  26  33  40  47
        addi t0, a0, -2                  6  13  20  27  34  41  48
        bge  t0, zero, nFabonacci        7  14  21  28  35  42  49
        addi sp, sp, 16                                         50
        jr ra                                                   51

nFabonacci:
        addi a0, a0, -1                  8  15  22  29  36  43
        jal  ra, Fabonacci               9  16  23  30  37  44
        addi t2, t1, 0                  87  80  73  66  59  52
        addi t1, a0, 0                  88  81  74  67  60  53
        lw   a0, 0(sp)                  89  82  75  68  61  54
        lw   ra, 8(sp)                  90  83  76  69  62  55
        addi sp, sp, 16                 91  84  77  70  63  56
        add a0, t1, t2                  92  85  78  71  64  57
        ret                             93  86  79  72  65  58

Print:
        lw  a0, argument                96
        li a7, 1                        97
        ecall                           98
        la a0, str1                     99
        li a7, 4                       100
        ecall                          101
        mv a0, a1                      102
        li a7, 1                       103
        ecall                          104
        ret                            105
```

Q2: What is the maximum number of variable be pushed into the stack at the same time when your code execute?

Ans: Stack Variable=14，由上圖可發現每執行一次 Fabonacci 會用到兩個 Stack Variable，共有七層，所以是 2*7=14。

gcd

Q1: How many instructions are actually executed?

Ans: 共 51 個指令，在 main 中的指令各跑一次(7)，gcd 和 ngcd 會互相遞迴，gcd 前三個指令共會跑三次，最後一次才會跑最後兩個指令進而進行 return(3*3+2=11)，ngcd 的一開始會先執行前四個指令，等 gcd return 回來之後每一層都會將指令執行完畢，所以七個指令皆會執行兩次(7*2=14)，Print 則是有十九個指令(19)且都不重複，所以總共是 7+11+14+19=51 個指令。

```
    lw a0, argument1                1
    lw a1, argument2                2
    jal ra, gcd                     3

    mv a1, a0                       29
    jal ra, Print                   30

    li a7, 10                       50
    ecall                           51
gcd:
    addi sp, sp, -8                 4   11   18
    sw ra, 0(sp)                    5   12   19
    bne a1, zero, ngcd              6   13   20
    addi sp, sp, 8                              21
    jr ra                                       22
ngcd:
    addi t0, a1, 0                  7   14
    rem a1, a0, t0                  8   15
    addi a0, t0, 0                  9   16
    jal ra, gcd                     10  17
    lw ra, 0(sp)                    26  23
    addi sp, sp, 8                  27  24
    ret                             28  25
Print:
    la a0, str1                     31
    li a7, 4                        32
    ecall                           33
    lw a0, argument1                34
    li a7, 1                        35
    ecall                           36
    la a0, str2                     37
    li a7, 4                        38
    ecall                           39
    lw a0, argument2                40
    li a7, 1                        41
    ecall                           42
    la a0, str3                     43
    li a7, 4                        44
    ecall                           45
    mv a0, a1                       46
    li a7, 1                        47
    ecall                           48
    ret                             49
```

Q2: What is the maximum number of variable be pushed into the stack at the same time when your code execute?

Ans: Stack Variable=3，由上圖可發現每執行一次 Fabonacci 會用到一個 Stack Variable，共有三層，所以是 1*3=3。

bubble_sort

Q1: How many instructions are actually executed?

Ans: 共 206 個指令，因為 array 中有十個數字的話太多指令了，所以我將它縮減成五個來計算指令數(抓原先十個中的前五個)。在 main 中的指令各跑一次(16)，每呼叫一次 PrintArray(3)就會 call 到 PrintLoop，然後一次會跑五次的 PrintLoop，最後一次 return 回去 main(3+8*5+1=44)，前後各跑一次 PrintArray(44*2=88)，每次跑進 bubblesort 所執行的指令數都不相同，有一次直接跳到 swap(3*1=3)，進去 swap 之後會再執行六個指令(6*1=6)，五次跳到 exit(5*5=25)，跳進 exit 後會執行五個指令，第五次會 return 回去 main(5*5+1=26)，六次跳回自己，也就是 bubblesort(7*6=42)，所以總共是 16+88+3+6+25+26+42=206 個指令。

```
.data
data: .word   5,3,6,7,31
N: .word 5
str1: .string "Array:\n"
str2: .string "\nSorted:\n"
str3: .string " "
.text
main:
        lw a2, N                    1
        la a0, str1                 2
        li a7, 4                    3
        ecall                       4
        la  a1, data                5
        jal ra, PrintArray          6

        mv a3, zero                51
        sub a1, a1, t1             52
        jal ra, bubblesort         53

        la a0, str2               156
        li a7, 4                  157
        ecall                     158
        la  a1, data              159
        jal ra, PrintArray        160

        li a7, 10                 205
        ecall                     206

bubblesort:
        lw t1, 0(a1)               54   63   73   83   90  100  107  114  124  131  138  145
        lw t2, 4(a1)               55   64   74   84   91  101  108  115  125  132  139  146
        bgt t1, t2, swap           56   65   75   85   92  102  109  116  126  133  140  147

        addi a4, a4, -1                 66   76   86   93  103  110  117  127  134  141  148
        blt a4, zero, exit              67   77   87   94  104  111  118  128  135  142  149

        addi a1, a1, -4                      88        105  112       129  136  143
        jal zero, bubblesort                 89        106  113       130  137  144

swap:
        addi t0, t1, 0            57
        addi t1, t2, 0            58
        addi t2, t0, 0            59

        sw t1, 0(a1)              60
        sw t2, 4(a1)              61
        jal zero, bubblesort      62

exit:
        addi a3, a3, 1                  68   78   95  119  150
        addi a4, a3, -1                 69   79   96  120  151
        slli t0, a4, 2                  70   80   97  121  152
        add a1, a1, t0                  71   81   98  122  153

        blt a3, a2, bubblesort          72   82   99  123  154
        jr ra                                          155

PrintArray:
        slli t1, a2, 2            7                       161
        add t0, a1, t1            8                       162
        jal zero, PrintLoop       9                       163

PrintLoop:
        lw a0, 0(a1)              10   18   26   34   42  164  172  180  188  196
        li a7, 1                  11   19   27   35   43  165  173  181  189  197
        ecall                     12   20   28   36   44  166  174  182  190  198
        la a0, str3               13   21   29   37   45  167  175  183  191  199
        li a7, 4                  14   22   30   38   46  168  176  184  192  200
        ecall                     15   23   31   39   47  169  177  185  193  201

        addi a1, a1, 4            16   24   32   40   48  170  178  186  194  202
        bne a1, t0, PrintLoop     17   25   33   41   49  171  179  187  195  203
        ret                                     50                        204
```

Q2: What is the maximum number of variable be pushed into the stack at the same time when your code execute?

Ans: 這題沒有使用到遞迴，所以沒有用到 stack variable。

Experience

還記得四天前我打開助教提供的 factorial.s 檔案，打算來研究一下組合語言，畢竟以前完全沒有接觸過組語，看到的當下整個人都懵了，每一條指令幾乎都不認識，除了少數有在課堂上聽老師介紹過，那時候真的完全不知道該怎麼完成這份作業，只能上網搜尋每條指令分別是什麼意思，邊查邊和同學討論，漸漸知道該如何使用那些指令。

Fabonacci 是我第一個開始寫的，因為我認為它的結構和助教提供的那份程式碼結構較相近，最一開始我只能算出 1+2+…+n，後來開始一個地方一個地方慢慢修正，找到方法去存前兩個，終於印出正確答案，之後我又發現輸入 0 的時候會輸出 1，不符合 C code，於是我又修改了一些地方，輸入 0 和 1 的時候都會輸出自己本身，於是乎這個作業就完成了。

那天下午一寫完 Fabonacci，我就很興奮的想寫下一個，因為我覺得慢慢產生出成就感了，沒想到在 gcd 卻又遇到問題，我一直存不到最終值(最後要 return 的 m)，每次他都會印出 argument1，後來發現是因為我每次都把 sp 的值 lw 回去 a0，這樣他 return 到最後，a0 就會是初始值，所以我就把要 lw a0 的地方放到 fact 的函式裡頭，這個問題也成功被解決了。

一打開 bubblesort 的 C code，看到有三個函式，甚至還有 array，完全不知道怎麼辦，想了好一陣子都不知道怎麼下手，一值卡在要如何記錄 array 的地方，查了很多資料都說要用 sp 來存放，有一堆很長的程式碼，看了很久都沒看懂，也想過要上網直接轉，但也都是很長的程式碼，所以決定要自己寫。後來發現可以直接寫一串數字，然後存位置，每次都用+4 來往下一個走，解決這個問題後順利很多，遇到的另一個問題是不知道該在哪 i++和 j--，後來想說寫另一個函式(exit)，修改了很多次終於成功了，最後一個問題是我一開始只能 swap 一次，會發現式這個問題是因為，我讓程式執行並且跟著他跑一遍，結果發現是 return 的地方錯了，讓他 swap 完就跑回去 main，導致結果出錯。

剛才看到它順利印出結果的那一刻，真的超感動，原本還想說不知道有幾個晚上不能睡覺了，對於這次作業，我認為我對 Risc-V 真的有更深的了解，雖然一開始真的心力交瘁，但我學到了好多，尤其是他的指令該如何運用，還有在迴圈彼此間會如何跳來跳去的。