Ve370 Project1

Gengchen Yang

518370910088

Objectives

In this lab, I developed a MIPS assembly program that counts the cold, hot and comfort days in a period of time, with respect to temperature.

Code

The source code is shown below with comments and explanation on the back of each role.

```
.data
    Array: .word
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30
,31,32,33,34,35,36,37

.text
.globl __start
__start: j main
        addi $t6, $0, 0
        addi $t6, $0, 0 #no use, for debug only
main:   addi $t0, $0, 37 #int size = 32;
        add $s0, $0, $0 #stores the value of hotDay
        add $s1, $0, $0 #stores the value of coldDay
        add $s2, $0, $0 #stores the value of comfortDay
        lui $t1, 0x1000
        ori $t1, $t1, 0x0000 #set tt0 to be the address of Array
        add $a0, $0, $t1 #pass &Array to later used functions
        add $a1, $0, $t0 #t0 is numElements, and pass it to a1
        addi $a2, $0, 1 #a2 is the mode (hotDay, coldDay, comfortDay)
        jal countArray
        addi $t6, $0, 0
        addi $t6, $0, 0 #no use, for debug only
        add $s0, $s0, $v0 #store the value passed back into hotDay
        addi $a2, $0, -1 #change the mode for coldDay
        jal countArray
        addi $t6, $0, 0
        addi $t6, $0, 0 #no use, for debug only
        add $s1, $s1, $v0 #store the value passed back to coldDay
        addi $a2, $0, 0 #change the mode for comfortDay
        jal countArray
        addi $t6, $0, 0
        addi $t6, $0, 0 #no use, for debug only
        add $s2, $s2, $v0 #store the value passed back into comfortDay
        addi $v0, $0, 1 #change v0 for print integers
        add $a0, $s0, $0 #set hotDay to be print
        syscall #print
        add $a0, $s1, $0 #set coldDay to be print
        syscall
        add $a0, $s2, $0 #set comfortDay to be print
        syscall
```

```
        #main finished
countArray:
        addi $sp, $sp, -4
        sw $ra, 0($sp) #initialization
        addi $s3, $0, 0 #initialize cnt
        sll $s4, $a1, 2 #i * 4 so that s4+a0 will give us A[i]
        addi $s5, $0, -1 #temp = -1, just a constant for comparation
        addi $s6, $0, 1 # temp2 = 1, just a constant for comparation
    Loop:
        addi $s4, $s4, -4 #numElements -1
        addi $t0, $0, -4 #constant, for comparation
        beq $s4, $t0 Exit #while i >= 0
        beq $a2, $s6 case1 #if a2 == 1, go to hotDay
        beq $a2, $s5 case2 #if a2 == -1, go to coldDay
        j case3 #else go to comfortDay
        addi $t6, $0, 0
        addi $t6, $0, 0 #no use, for debug only
    case1: #hotDay case
        add $t0, $a0, $s4 #load A[i]
        lw $a3, 0($t0) #pass x to the function
        jal hot
        addi $t6, $0, 0
        addi $t6, $0, 0 #no use, for debug only
        beq $s6, $v0 Add #if hot returns true, cnt++
        j Loop #else continue the loop
    case2: #coldDay case
        add $t0, $a0, $s4 #load A[i]
        lw $a3, 0($t0) #pass x to the function
        jal cold
        addi $t6, $0, 0
        addi $t6, $0, 0 #no use, for debug only
        beq $s6, $v0 Add #if cold returns true, cnt++
        j Loop #else do nothing and continue the loop
        addi $t6, $0, 0
        addi $t6, $0, 0 #no use, for debug only
    case3: #similar to previous cases
        add $t0, $a0, $s4
        lw $a3, 0($t0) #pass x to the function
        jal comfort
        addi $t6, $0, 0
        addi $t6, $0, 0
        beq $s6, $v0 Add
        j Loop
        addi $t6, $0, 0
        addi $t6, $0, 0
    Add: #subfunction for cnt++
        addi $s3, $s3, 1
        j Loop
        addi $t6, $0, 0
        addi $t6, $0, 0
    Exit: #after the loop is finished, set v0 to be cnt, and go back to main
        add $v0, $0, $s3 #v0 = cnt
        lw $ra, 0($sp) #load the position of main we were when we entered this
function
        addi $sp, $sp, 4 #give the memory we asked for back
        jr $ra
        addi $t6, $0, 0
        addi $t6, $0, 0 #no use, for debug only
```

```
hot: #judge whether x belongs to hotDay
        slti $t0, $a3, 30 #if x < 30
        beq $t0, $0 return1 #if !(x < 30), return 1
        addi $v0, $0, 0 #else return 0
        jr $ra
        addi $t6, $0, 0
        addi $t6, $0, 0
    return1:
        addi $v0, $0, 1 #return 1
        jr $ra
        addi $t6, $0, 0
        addi $t6, $0, 0
cold: #judge whether x belongs to coldDay
        slti $v0, $a3, 6 #if x < 6, return 1, else return 0 (<6 is equal to <=5)
        jr $ra
        addi $t6, $0, 0
        addi $t6, $0, 0
comfort: #judge whether x belongs to comfortDay
        slti $t0, $a3, 30 #first judge if x <30
        addi $t1, $0, 1
        beq $t0, $t1 Next #if x < 30, go to further judgement
        addi $v0, $0, 0 #else return 0
        jr $ra
        addi $t6, $0, 0
        addi $t6, $0, 0
    Next:
        slti $t0, $a3, 6 #then judge if x > 5
        addi $t1, $0, 1
        beq $t0, $t1 return0 #if x < 6, then it must not > 5, hence return 0
        addi $v0, $0, 1 #else return 1
        jr $ra
        addi $t6, $0, 0
        addi $t6, $0, 0
    return0:
        addi $v0, $0, 0 #return 0
        jr $ra
        addi $t6, $0, 0
        addi $t6, $0, 0
```

Notes: In large projects we should also store the values of a0-a3 into memory, but in this project, it happened that the a* registers are enough, hence I think we can save the trouble to sw and then lw, like the following code, for convenience and ease to read.

```
    addi $sp, $sp, -8
    sw $a0, 0($sp)
    sw $a1, 4($sp)
    add $a0, $0, $t1 #suppose t1 is the value we need
    ...
    lw $a0, 0($sp)
    lw $a1, 4($sp)
    ...
```

Process and expectation

The usage of registers are shown below:

t0~t9: For temp use, instead of storing.

s0: hotDay

s1: coldDay

s2: comfortDay

s3: cnt

s4: i (A[i], starts from numElements - 1)

s5: constant -1

s6: constant 1 (note: I should use t-registers to do the job, but it turned out that there are enough registers and hence using s-registers saves much lines)

a0: address of A[0]

a1: numElements

a2: mode (i.e. hotDay, coldDay, comfortDay)

a3: A[i] that needs to be passed into hot, cold, comfort

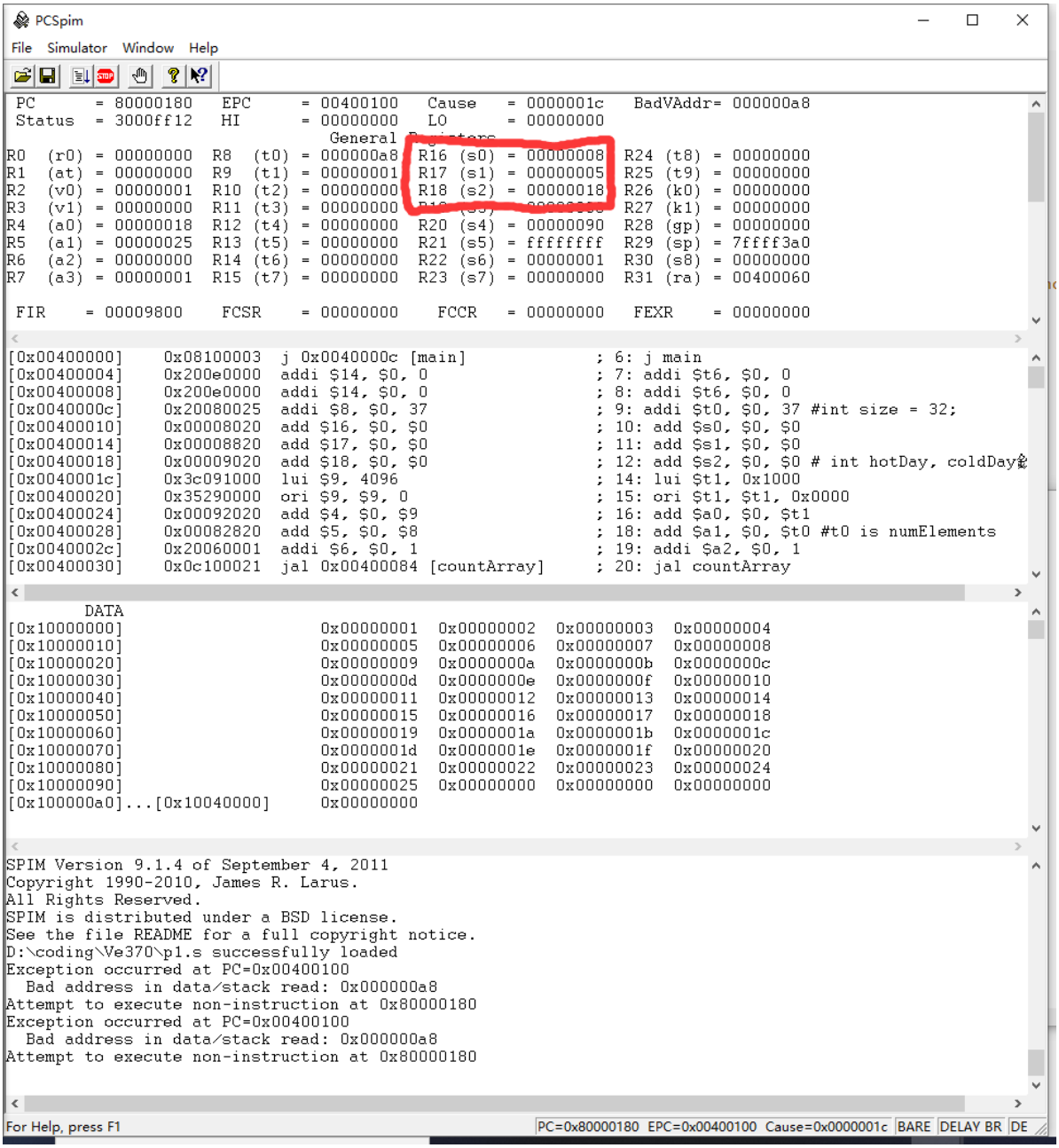
v0: returned values

Expectation:

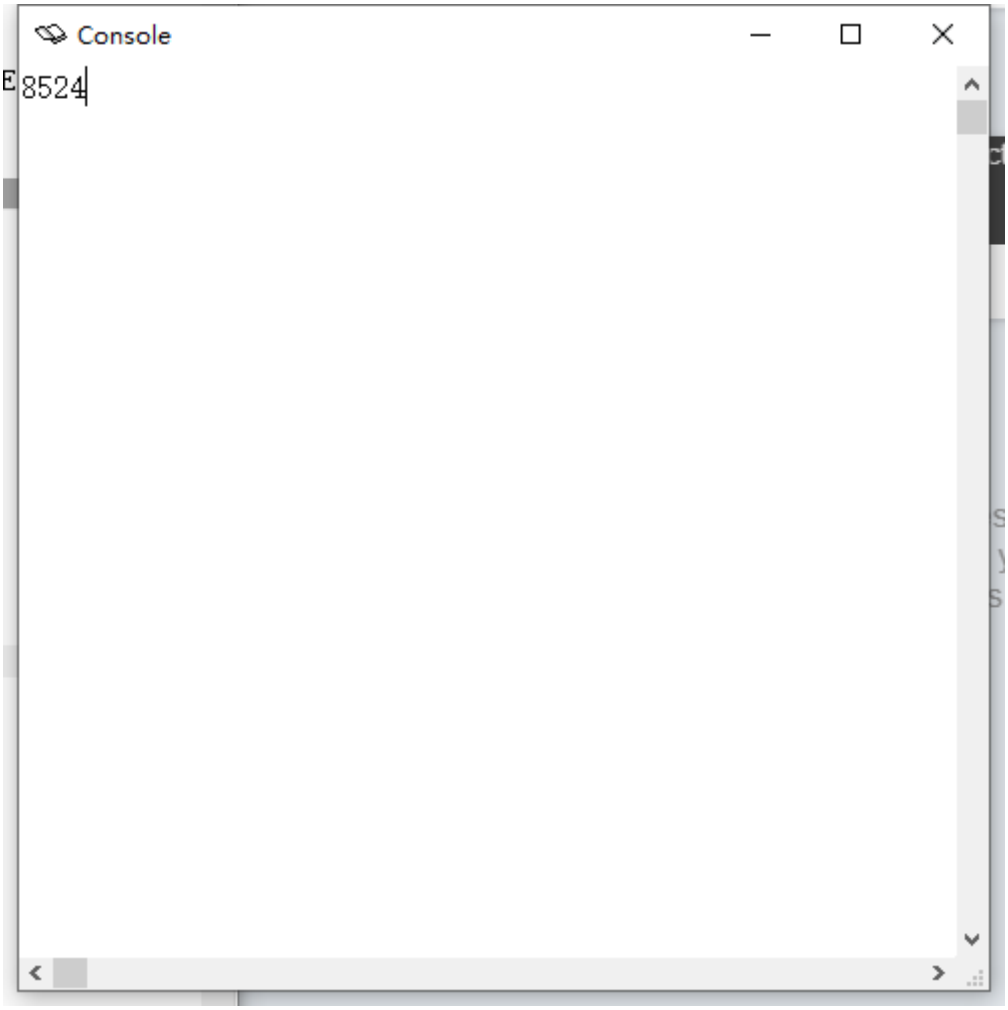
Since the array I set is 1-37, increasing 1 at a time, we should have 5 (1-5) coldDay (<=5), 24 (6-29) comfortDay (>5 && <30) and 8 (30-37) hotDays (>=30).

Results

I stored the values of hotDay in s0, coldDay in s1, and comfortDay in s2, the results are shown below



I also printed the results out, hence we also get



Where hotDay = 8, coldDay = 5, comfortDay = 24, just as we expected.