

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 comparison
    addi $s6, $0, 1 # temp2 = 1, just a constant for comparison
Loop:
    addi $s4, $s4, -4 #numElements -1
    addi $t0, $0, -4 #constant, for comparison
    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

```
PCSpim
File Simulator Window Help

PC = 80000180 EPC = 00400100 Cause = 0000001c BadVAddr= 000000a8
Status = 3000ff12 HI = 00000000 LO = 00000000

General Registers
R0 (r0) = 00000000 R8 (t0) = 000000a8 R16 (s0) = 00000008 R24 (t8) = 00000000
R1 (at) = 00000000 R9 (t1) = 00000001 R17 (s1) = 00000005 R25 (t9) = 00000000
R2 (v0) = 00000001 R10 (t2) = 00000000 R18 (s2) = 00000018 R26 (k0) = 00000000
R3 (v1) = 00000000 R11 (t3) = 00000000 R19 (s3) = 00000000 R27 (k1) = 00000000
R4 (a0) = 00000018 R12 (t4) = 00000000 R20 (s4) = 00000090 R28 (gp) = 00000000
R5 (a1) = 00000025 R13 (t5) = 00000000 R21 (s5) = ffffffff R29 (sp) = 7ffff3a0
R6 (a2) = 00000000 R14 (t6) = 00000000 R22 (s6) = 00000001 R30 (s8) = 00000000
R7 (a3) = 00000001 R15 (t7) = 00000000 R23 (s7) = 00000000 R31 (ra) = 00400060

FIR = 00009800 FCSR = 00000000 FCCR = 00000000 FEXR = 00000000

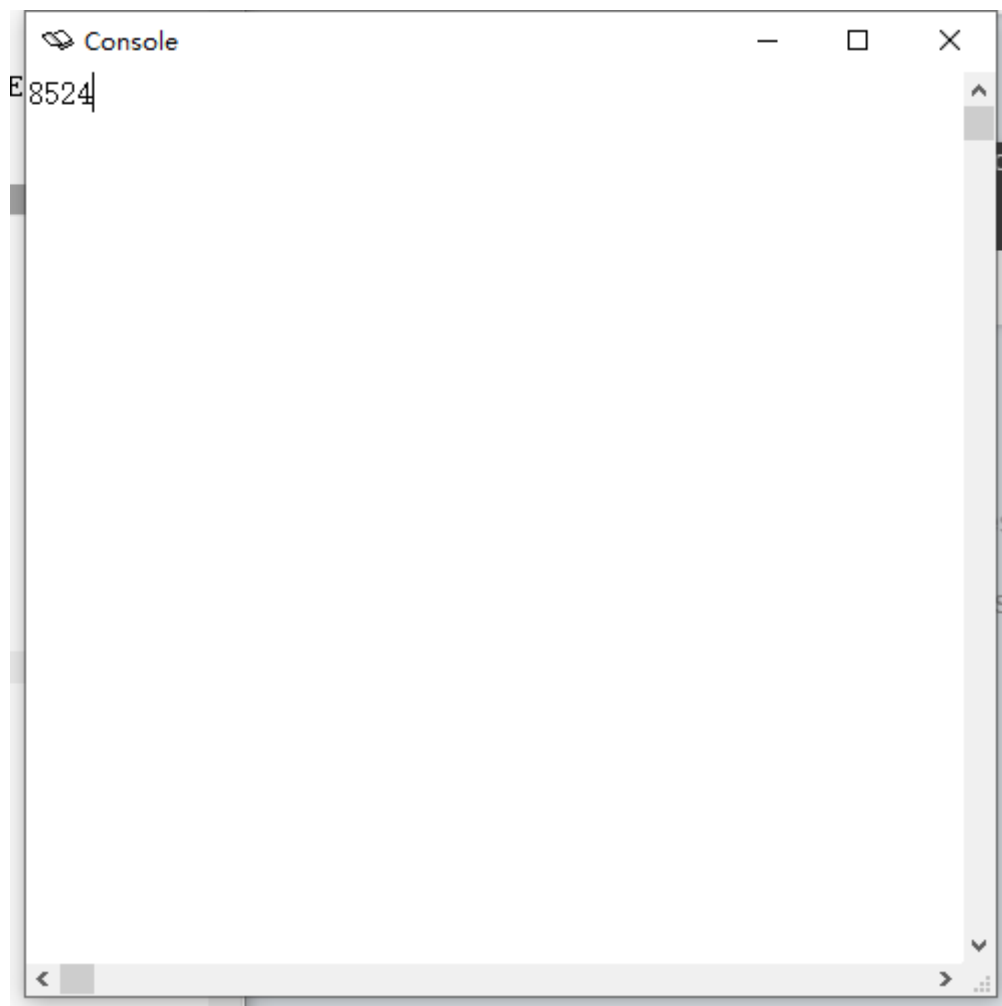
<
[0x00400000] 0x08100003 j 0x0040000c [main] ; 6: j main
[0x00400004] 0x200e0000 addi $t4, $0, 0 ; 7: addi $t6, $0, 0
[0x00400008] 0x200e0000 addi $t4, $0, 0 ; 8: addi $t6, $0, 0
[0x0040000c] 0x20080025 addi $8, $0, 37 ; 9: addi $t0, $0, 37 #int size = 32;
[0x00400010] 0x00008020 add $t6, $0, $0 ; 10: add $s0, $0, $0
[0x00400014] 0x00008820 add $t7, $0, $0 ; 11: add $s1, $0, $0
[0x00400018] 0x00009020 add $t8, $0, $0 ; 12: add $s2, $0, $0 # int hotDay, coldDay
[0x0040001c] 0x3c091000 lui $9, 4096 ; 14: lui $t1, 0x1000
[0x00400020] 0x35290000 ori $9, $9, 0 ; 15: ori $t1, $t1, 0x0000
[0x00400024] 0x00092020 add $4, $0, $9 ; 16: add $a0, $0, $t1
[0x00400028] 0x00082820 add $5, $0, $8 ; 18: add $a1, $0, $t0 #t0 is numElements
[0x0040002c] 0x20060001 addi $6, $0, 1 ; 19: addi $a2, $0, 1
[0x00400030] 0x0c100021 jal 0x00400084 [countArray] ; 20: jal countArray

<
DATA
[0x10000000] 0x00000001 0x00000002 0x00000003 0x00000004
[0x10000010] 0x00000005 0x00000006 0x00000007 0x00000008
[0x10000020] 0x00000009 0x0000000a 0x0000000b 0x0000000c
[0x10000030] 0x0000000d 0x0000000e 0x0000000f 0x00000010
[0x10000040] 0x00000011 0x00000012 0x00000013 0x00000014
[0x10000050] 0x00000015 0x00000016 0x00000017 0x00000018
[0x10000060] 0x00000019 0x0000001a 0x0000001b 0x0000001c
[0x10000070] 0x0000001d 0x0000001e 0x0000001f 0x00000020
[0x10000080] 0x00000021 0x00000022 0x00000023 0x00000024
[0x10000090] 0x00000025 0x00000026 0x00000027 0x00000028
[0x100000a0]...[0x10040000] 0x00000000

SPIM Version 9.1.4 of September 4, 2011
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
D:\coding\Ve370\p1.s successfully loaded
Exception occurred at PC=0x00400100
Bad address in data/stack read: 0x000000a8
Attempt to execute non-instruction at 0x80000180
Exception occurred at PC=0x00400100
Bad address in data/stack read: 0x000000a8
Attempt to execute non-instruction at 0x80000180

For Help, press F1 PC=0x80000180 EPC=0x00400100 Cause=0x0000001c BARE DELAY BR DE
```

I also printed the results out, hence we also get



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