VE370 Project 1

Mingxuan Lu 518021911166

1 Objectives

Develop a MIPS assembly program that operates on a data segment consisting of an array of 32-bit signed integers. In the text (program) segment of memory, write a procedure called main that implements the main() function and as well as procedures for other subroutines described below. Assemble, simulate, and carefully comment the file. Screen print your simulation results and explain the results by annotating the screen prints. You should compose an array whose size is determined by you in the main function and is not less than 30 elements.

2 Procedures

2.1 Convert into C

First, the pseudocode is converted into more accurate C language.

```c
int hot(int x){
    if(x >= 30) return 1;
    else return 0;
}

int cold(int x){
    if (x <= 5) return 1;
    else return 0;
}

int comfort(int x){
    if (x > 5 && x < 30) return 1;
    else return 0;
}

int countArray(int A[], int numElements, int cntType){
    /*************************************************************************
    * Count specific elements in the integer array A[] whose size is *
    * numElements and return the following: *
    * *
    * When cntType = 1, count the elements greater than or equal to 30; *
    * When cntType = -1, count the elements less than or equal to 5; *
    * When cntType = 0, count the elements greater than 5 and less than 30. *
    *************************************************************************/
    int i, cnt = 0;
    for(i = numElements-1; i >= 0; i--){
        switch (cntType) {
            case 1 : cnt += hot(A[i]); break;
            case -1 : cnt += cold(A[i]); break;
            default: cnt += comfort(A[i]);
        }
    }
    return cnt;
}

int main(int argc, char const *argv[]){
    int size = ...; //determine the size of the array here
    int hotDay, coldDay, comfortDay = 0;
    int tempArray[size] = {30, 25, ...};//compose your own array here
    hotDay = countArray(tempArray, size, 1);
    coldDay = countArray(tempArray, size, -1);
    comfortDay = countArray(tempArray, size, 0);
}
```

2.2 main function

In the main function, I store the array on stack. I set the size of array as 30 and write a python program to generate and write the data in assembly language into .s file.

```
import random
size = 30
Arr = []

for i in range(size):
    Arr.append(random.randint(-20, 55))

ifile = open('storeArray.s', 'w')
for i in range(size):
    ifile.write('   addi $t0, $0, %2d    # $t0 = %d\n' % (Arr[i], Arr[i]))
    ifile.write('   sw $t0, %3d($s4)    # testArray[%d] = $t0\n' % (i * 4, i))
```

Then I called the function countArray to get the value of hotDay, comfortDay, and substract them from total size to get the value of coldDay. I store the result of comfortDay, hotDay and coldDay respectively into register **s3, s5, s6**.

2.3 countArray function

I save the value of register **s1~s4** and **ra**. Then I wrote a loop function called "Loop" to get run the counting process. After the loop ends, I load the value of register **ra** to jump back to the main function.

2.4 hotDay and comfortDay funciton

In these two functions, I runs some condition statements to get the final results. In comfortDay function, I use "and" instruction to identify if "A[i]" is in the range that $5 < A[i] < 30$.

3 Result

The array I use have the following data: [-25, 47, 46, -7, -7, 9, 18, 53, 16, 15, 18, 17, 7, 15, 37, 24, 53, 53, 16, 32, 10, 11, 26, 35, 15, 24, 2, 1, 4, 34]. The simulation result is shown in the figure below (Figure 1).

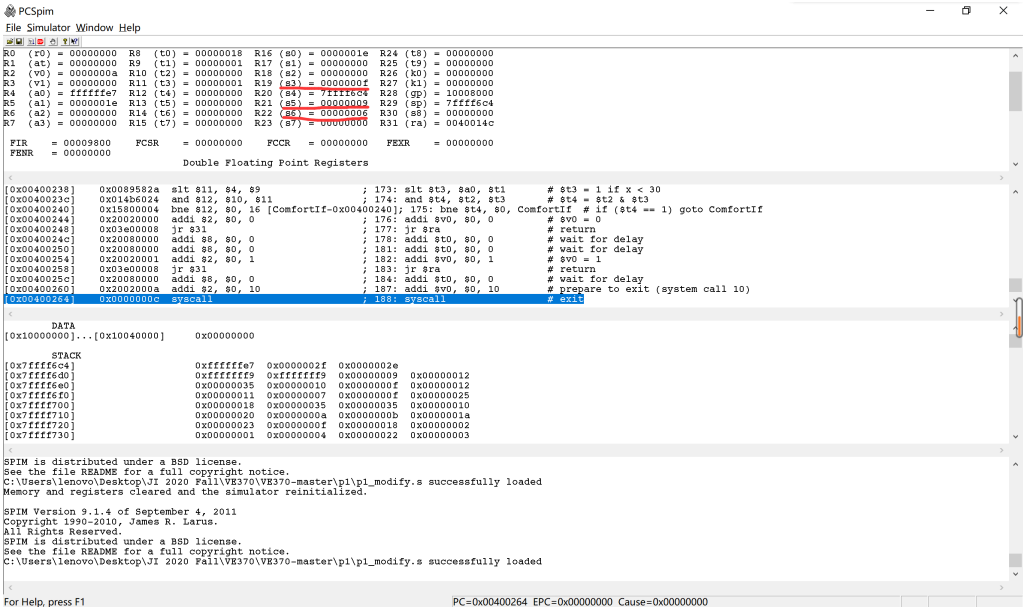

Figure 1. Simulation Result

From Figure 1, there are total 15 comfortdays, 6 colddays and 9 hotdays.

4 Conclusion

Overall, there is one issue I want to bring up: the problem with j, jar and jr. At first, it really bothers me a lot due to strange mistakes using j-instructions. Later I know that two non-meaning instruction must be put in front of and behind one j-instruction to prevent errors mentioned above.

5 Appendix

```
######### Created by Mingxuan Lu ##########
        .globl __start
__start:
        addi $sp, $sp, -120     # adjust stack for 50*4 item

        addi $s0, $0, 30        # int size = 30
        add $s1, $0, $0         # intialize registers
        add $s2, $0, $0
        add $s3, $0, $0
        add $s4, $0, $sp        # int testArray[size]
        add $s5, $0, $0
        add $s6, $0, $0

        #[-25, 47, 46, -7, -7, 9, 18, 53, 16, 15, 18, 17, 7, 15, 37, 24, 53, 53, 16, 32, 10, 11, 26, 35, 15, 24, 2,
1, 4, 34]
        addi $t0, $0, -25       # $t0 = -25
        sw $t0,   0($s4)        # testArray[0] = $t0
        addi $t0, $0, 47        # $t0 = 47
        sw $t0,   4($s4)        # testArray[1] = $t0
        addi $t0, $0, 46        # $t0 = 46
        sw $t0,   8($s4)        # testArray[2] = $t0
        addi $t0, $0, -7        # $t0 = -7
        sw $t0,  12($s4)        # testArray[3] = $t0
        addi $t0, $0, -7        # $t0 = -7
        sw $t0,  16($s4)        # testArray[4] = $t0
        addi $t0, $0,  9        # $t0 = 9
        sw $t0,  20($s4)        # testArray[5] = $t0
        addi $t0, $0, 18        # $t0 = 18
        sw $t0,  24($s4)        # testArray[6] = $t0
        addi $t0, $0, 53        # $t0 = 53
        sw $t0,  28($s4)        # testArray[7] = $t0
        addi $t0, $0, 16        # $t0 = 16
        sw $t0,  32($s4)        # testArray[8] = $t0
        addi $t0, $0, 15        # $t0 = 15
        sw $t0,  36($s4)        # testArray[9] = $t0
        addi $t0, $0, 18        # $t0 = 18
        sw $t0,  40($s4)        # testArray[10] = $t0
        addi $t0, $0, 17        # $t0 = 17
        sw $t0,  44($s4)        # testArray[11] = $t0
        addi $t0, $0, 7         # $t0 = 7
        sw $t0,  48($s4)        # testArray[12] = $t0
        addi $t0, $0, 15        # $t0 = 15
        sw $t0,  52($s4)        # testArray[13] = $t0
        addi $t0, $0, 37        # $t0 = 37
        sw $t0,  56($s4)        # testArray[14] = $t0
        addi $t0, $0, 24        # $t0 = 24
        sw $t0,  60($s4)        # testArray[15] = $t0
        addi $t0, $0, 53        # $t0 = 53
        sw $t0,  64($s4)        # testArray[16] = $t0
        addi $t0, $0, 53        # $t0 = 53
        sw $t0,  68($s4)        # testArray[17] = $t0
        addi $t0, $0, 16        # $t0 = 16
        sw $t0,  72($s4)        # testArray[18] = $t0
        addi $t0, $0, 32        # $t0 = 32
        sw $t0,  76($s4)        # testArray[19] = $t0
        addi $t0, $0, 10        # $t0 = 10
        sw $t0,  80($s4)        # testArray[20] = $t0
        addi $t0, $0, 11        # $t0 = 11
        sw $t0,  84($s4)        # testArray[21] = $t0
        addi $t0, $0, 26        # $t0 = 26
        sw $t0,  88($s4)        # testArray[22] = $t0
        addi $t0, $0, 35        # $t0 = 35
        sw $t0,  92($s4)        # testArray[23] = $t0
        addi $t0, $0, 15        # $t0 = 15
        sw $t0,  96($s4)        # testArray[24] = $t0
        addi $t0, $0, 24        # $t0 = 24
        sw $t0, 100($s4)        # testArray[25] = $t0
        addi $t0, $0,  2        # $t0 = 2
        sw $t0, 104($s4)        # testArray[26] = $t0
        addi $t0, $0, 1         # $t0 = 1
        sw $t0, 108($s4)        # testArray[27] = $t0
        addi $t0, $0, 4         # $t0 = 4
        sw $t0, 112($s4)        # testArray[28] = $t0
        addi $t0, $0, 34        # $t0 = 34
        sw $t0, 116($s4)        # testArray[29] = $t0

        add $a0, $0, $s4        # $a0 = testArray
        add $a1, $0, $s0        # $a1 = size
        addi $a2, $0, 1         # $a2 = 1
        jal countArray          # $v0 = countArray(testArray, size, 1)
        addi $t1, $0, 1         # wait for delay
        add $s5, $0, $v0        # save the result of hotday into $s5
```

```
        add $a0, $0, $s4        # $a0 = testArray
        add $a1, $0, $s0        # $a1 = size
        addi $a2, $0, 0         # $a2 = 0
        jal countArray          # $v0 = countArray(testArray, size, 0)
        addi $t1, $0, 1         # wait for delay
        add $s3, $0, $v0        # save the result of comfortday into $s3

        add $t0, $s3, $s5       # substract to get the number of colddays
        sub $s6, $s0, $t0

        jal exit                # goto exit
        addi $t0, $0, 0         # wait for delay

countArray:
        addi $sp, $sp, -24      # adjust stack for 6 items
        sw $ra, 20($sp)         # save $ra on stack
        sw $s4, 16($sp)         # save $s4 on stack
        sw $s3, 12($sp)         # save $s3 on stack
        sw $s2, 8($sp)          # save $s2 on stack
        sw $s1, 4($sp)          # save $s1 on stack
        sw $s0, 0($sp)          # save $s0 on stack

        add $s0, $0, $a0        # save $a0(int A[]) into $s0
        add $s1, $0, $a1        # save $a1(int numElements) into $s1
        add $s2, $0, $a2        # save $a2(int cntType) into $s2

        addi $s3, $s1, -1       # $s3(i) = numElements - 1
        addi $s4, $0, 0         # $s4(cnt) = 0

Loop:
        addi $t0, $0, 0         # wait for delay
        slt $t0, $s3, $0        # $t0 = i < 0
        bne $t0, $0, LoopEnd
                                # if ($t0 != 0) goto countArrayEndFor
        sll $t0, $s3 ,2         # $t0 = i * 4
        add $t0, $s0, $t0       # $t0 = A + $t0
        lw $a0, 0($t0)          # $a0 = A[i]
        addi $t1, $0, 1         # $t1 = 1
        addi $t0, $0, 0         # wait for delay
        beq $s2, $t1, Hotday
                                # if (cntType = 1) go to Hot
        jal Comfort             # $v0 = Comfort(A[i])
        addi $t1, $0, 1         # wait for delay
        j nextLoop              # jump to nextLoop
        addi $t0, $0, 0         # wait for delay

Hotday:
        addi $t0, $0, 0         # wait for delay
        jal Hot                 # $v0 = Hot(A[i])
        addi $t0, $0, 0         # wait for delay

nextLoop:
        addi $t0, $0, 0         # wait for delay
        add $s4, $s4, $v0       # cnt += $v0
        addi $s3, $s3, -1       # i--
        j Loop          # jump to for begin
        addi $t0, $0, 0         # wait for delay

LoopEnd:
        addi $t0, $0, 0         # wait for delay
        add $v0, $0, $s4        # $v0 = cnt
        lw $s0, 0($sp)          # restore $s0 from stack
        lw $s1, 4($sp)          # restore $s1 from stack
        lw $s2, 8($sp)          # restore $s2 from stack
        lw $s3, 12($sp)         # restore $s3 from stack
        lw $s4, 16($sp)         # restore $s4 from stack
        lw $ra, 20($sp)         # restore $ra from stack
        addi $sp, $sp, 24       # recover the stack
        addi $t0, $0, 0         # wait for delay
        jr $ra                  # return
        addi $t0, $0, 0         # wait for delay

Hot:
        addi $t0, $0, 30
        slt $t1, $a0, $t0       # $t1 = x < 30
        bne $t1, $0, HotIf      # if ($t1 == 1) goto HotIf
        add $v0, $0, 1          # $v0 = 1
        jr $ra                  # return
        addi $t0, $0, 0         # wait for delay

HotIf:
        addi $t0, $0, 0         # wait for delay
        addi $v0, $0, 0         # $v0 = 0
        jr $ra                  # return
        addi $t0, $0, 0         # wait for delay

Comfort:
```

```
        addi $t0, $0, 5         # $t0 = 5
        addi $t1, $0, 30        # $t1 = 30
        slt $t2, $t0, $a0       # $t2 = 1 if 5 < x
        slt $t3, $a0, $t1       # $t3 = 1 if x < 30
        and $t4, $t2, $t3       # $t4 = $t2 & $t3
        bne $t4, $0, ComfortIf  # if ($t4 == 1) goto ComfortIf
        addi $v0, $0, 0         # $v0 = 0
        jr $ra                  # return
        addi $t0, $0, 0         # wait for delay

ComfortIf:
        addi $t0, $0, 0         # wait for delay
        addi $v0, $0, 1         # $v0 = 1
        jr $ra                  # return
        addi $t0, $0, 0         # wait for delay

exit:
        addi $v0, $0, 10        # prepare to exit (system call 10)
        syscall                 # exit
```