UM-SJTU JOINT INSTITURE

Intro to Computer Organization
(VE370)

Project 1

prof. Zheng

XINMIAO YU
518021910792

DATE: 28 Sept. 2020

1 Introduction

In this project, an array of 32-bits signed integer has a predefined **size 32**. These 32 numbers are assigned randomly. The C program is shown below.

```c
int main() {
    int size = 32; //determine the size of the array here
    int hotDay, coldDay, comfortDay;
    int tempArray[32] = {36, 9, -8, 40, 25, 20, 18, 19, 15, 16, 17, 16, 15,
    ↪  14,
    13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, -3, 30, -19, 33};
    hotDay = countArray (tempArray, size, 1);
    coldDay = countArray (tempArray, size, -1);
    comfortDay = countArray (tempArray, size, 0);
}
int countArray(int A[], int numElements, int cntType) {
    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 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;
}
```

A MIPS program with same functions as the C program is developed. In addition, no pseudo-instruction used.

2 Procedures

The program is developed following the logic of C program.

- As use `la` to load in the address of array is not allowed, the address used to store the numbers is predefined, as $0x10001000$. Then `main` function, where `int tempArray[32]=...` is needed, the base address ($0x10001000$) is load in $a0 through the command `lui` and `ori`.

- In `main` function, other variables needed is defined as

```
$a1 = size                  // as numElements = 32
$s0 = hotDay                // as 0
$s1 = coldDay               // as 0
$s2 = comfortDay      // as 0
```

  function `countArray` is called three times. Before each call, make $a2 as 1/-1/0 and the **stack pointer** is adjusted for 6 items. The function arguments ($a0, $a1), return address ($ra) and saved register ($s0, $s1, $s2) are all saved into the stack before call `countArray` and restored after the function call. According to the return value in $v0, the number of hotDay/coldDay/comfortDay are all determined.

- In function `countArray`, define

```
$s0 = 0                  // as numElements - 1 = 31
$s1 = cnt                // as 0
```

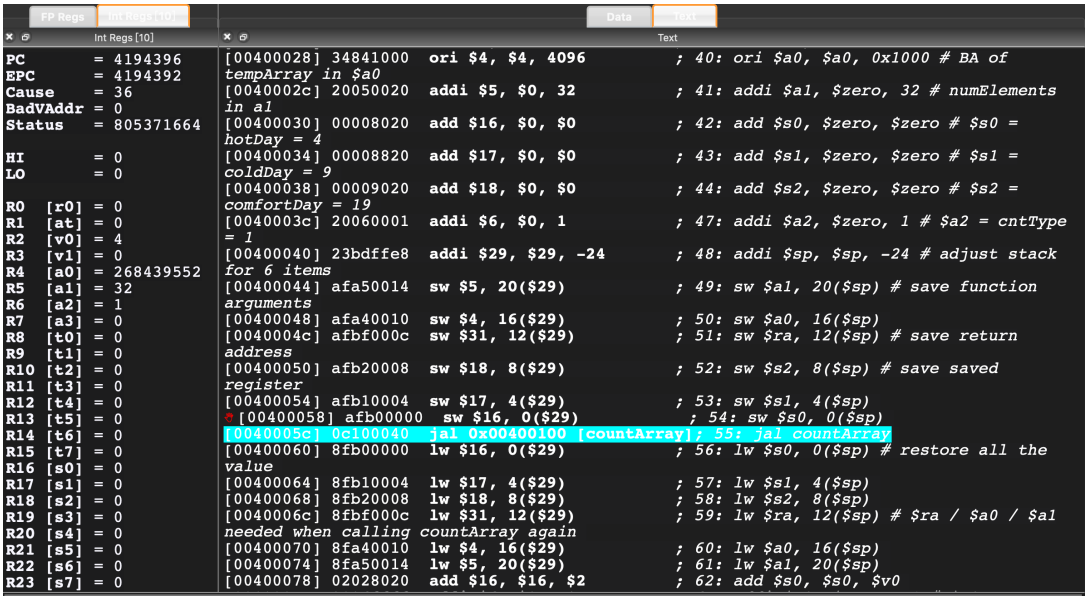
  Same as before, before function call adjust stack pointer for saving $a0, $a1, $a2, $ra, $s0, $s1. Set $a0 = A[i], where the address of A[i] is obtained through $a0 + $s0 *4. Restore all the value after each function call and change $s1 according to the value in $v0.

- For leaf function `hot/cold/comfort`, return the right value, stored in $v0.

3 Simulation Result

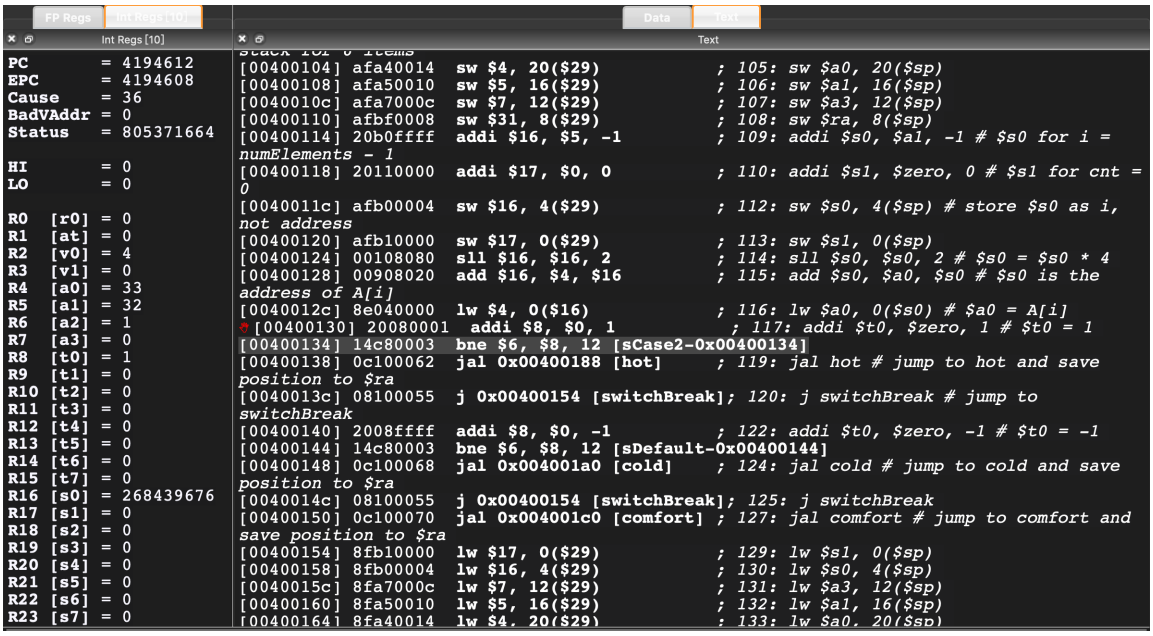
The simulation result is obtained through *QtSpim*. All the register show decimal value for simpler explanation.

1. **Before first call** `countArray`



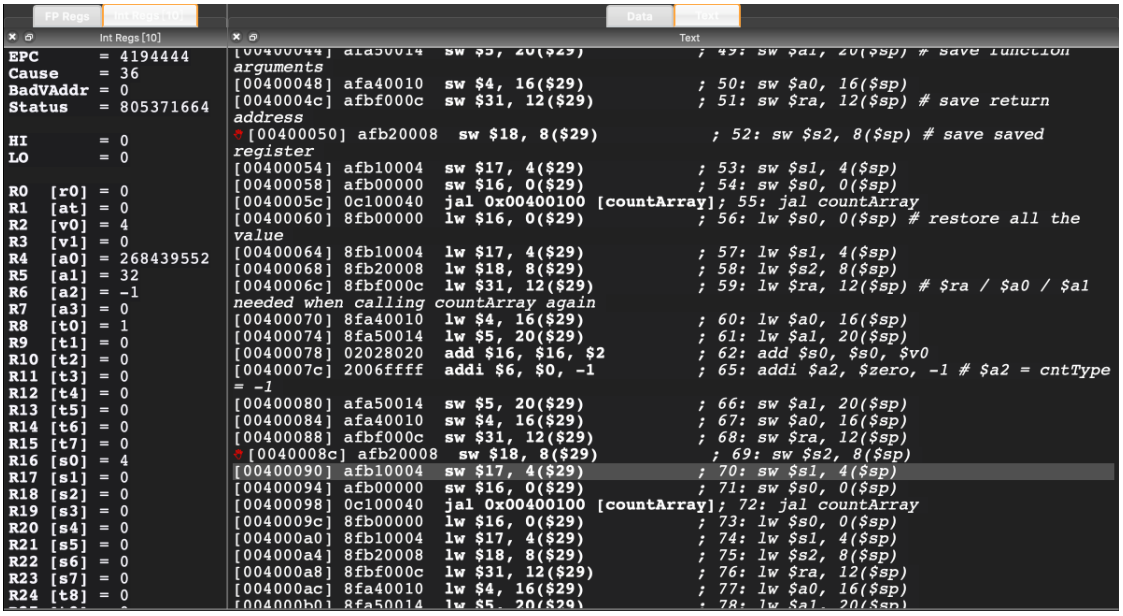
we could see that $a0 = $(268439552)_{10}$ = 0x10001000, $a1 = 32, $a2 = 1, as expected.

2. **Before first call** `hot/cold/comfort`



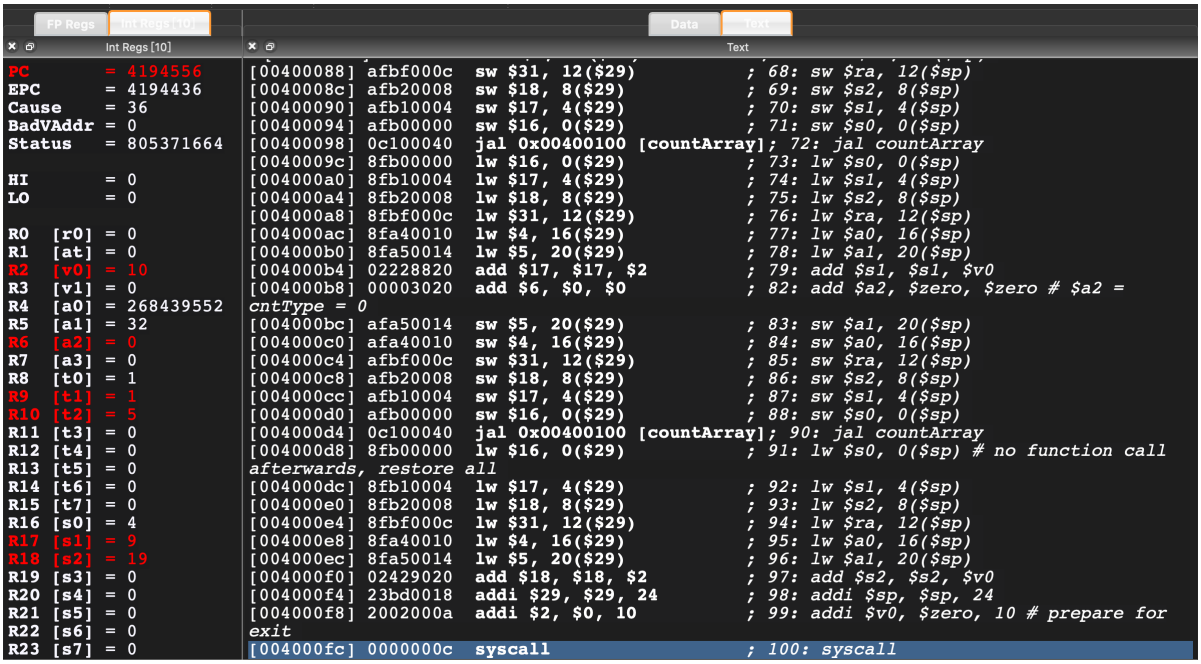
the value of A[i], which is A[31] = 33, is load in $a0, as function argument to pass to the leaf function. Now $s0 has the address of A[31] = $(268439676)_{10}$ = 0x1000107c = 0x10001000 + $(32 * 4)_{10}$. Then the loop will continue until all the value in the array are passed.

3. **After first call** `countArray`



The value of hotDay is stored in $s0, which is 4. $a2 is set as -1 for count coldDay. The value of $a0 and $a1 should not change as restored from stack. Then the procedure is similar, only differ in the leaf function called.

4. **Finish**



When the program finishes, $s0 = 4 (the number of hotDays), $s1 = 9 (the number of coldDays), $s2 = 19 (the number of comfortDays), as expected.

4 Conclusion

In conclusion, the program written through MIPS successfully finished the expected tasks and output the correct answer. Potential error might caused by

- use pseudo-instruction, caused syntax error.
- forget to adjust the stack pointer to save/restore values before/after each function call.
- the existed delay. Add some meaningless command will help.

A Program

```
1       .data 0x10001000
2   tempArrary:
3       .word    36
4       .word    9
5       .word    -8
6       .word    40
7       .word    25
8       .word    20
9       .word    18
10      .word    19
11      .word    15
12      .word    16
13      .word    17
14      .word    16
15      .word    15
16      .word    14
17      .word    13
18      .word    12
19      .word    11
20      .word    10
21      .word    9
22      .word    8
23      .word    7
24      .word    6
25      .word    5
26      .word    4
27      .word    3
28      .word    2
29      .word    1
30      .word    0
31      .word    -3
32      .word    30
33      .word    -19
34      .word    33
35  str1:
36      .asciiz "hotDay = "
37  str2:
38      .asciiz "coldDay = "
39  str3:
40      .asciiz "comfortDay = "
41      .text
42      .align    2
43      .globl    main
44  main:
45      lui     $a0, 0x1000
46      ori     $a0, $a0, 0x1000    # BA of tempArray in $a0
47      addi    $a1, $zero, 32      # numElements in a1
```

```
48      add     $s0, $zero, $zero   # £s0 = hotDay = 4
49      add     $s1, $zero, $zero   # £s1 = coldDay = 9
50      add     $s2, $zero, $zero   # £s2 = comfortDay = 19
51
52      ## First call: hotDay = countArray (tempArray, size, 1);
53      addi    $a2, $zero, 1       # £a2 = cntType = 1
54      addi    $sp, $sp, -24       # adjust stack for 6 items
55      sw      $a1, 20($sp)        # save function arguments
56      sw      $a0, 16($sp)
57      sw      $ra, 12($sp)        # save return address
58      sw      $s2, 8($sp)         # save saved register
59      sw      $s1, 4($sp)
60      sw      $s0, 0($sp)
61      jal     countArray
62      lw      $s0, 0($sp)          # restore all the value
63      lw      $s1, 4($sp)
64      lw      $s2, 8($sp)
65      lw      $ra, 12($sp)         # £ra / £a0 / £a1 needed when calling
        ↪  countArray again
66      lw      $a0, 16($sp)
67      lw      $a1, 20($sp)
68      add     $s0, $s0, $v0
69      addi    $v0, $zero, 1       # to output the number
70      add     $a0, $s0, $zero
71      syscall
72      lw      $a0, 16($sp)
73
74      ## Second call: coldDay = countArray (tempArray, size, -1);
75      addi    $a2, $zero, -1      # £a2 = cntType = -1
76      sw      $a1, 20($sp)
77      sw      $a0, 16($sp)
78      sw      $ra, 12($sp)
79      sw      $s2, 8($sp)
80      sw      $s1, 4($sp)
81      sw      $s0, 0($sp)
82      jal     countArray
83      lw      $s0, 0($sp)
84      lw      $s1, 4($sp)
85      lw      $s2, 8($sp)
86      lw      $ra, 12($sp)
87      lw      $a0, 16($sp)
88      lw      $a1, 20($sp)
89      add     $s1, $s1, $v0
90      addi    $v0, $zero, 1       # to output the number
91      add     $a0, $s1, $zero
92      syscall
93      lw      $a0, 16($sp)
94
95      ## comfortDay = countArray (tempArray, size, 0);
```

```
96      add     $a2, $zero, $zero   # $a2 = cntType = 0
97      sw      $a1, 20($sp)
98      sw      $a0, 16($sp)
99      sw      $ra, 12($sp)
100     sw      $s2, 8($sp)
101     sw      $s1, 4($sp)
102     sw      $s0, 0($sp)
103     ## only $s1 has chanegd, store this only
104     jal     countArray
105     lw      $s0, 0($sp)          # no function call afterwards, restore all
106     lw      $s1, 4($sp)
107     lw      $s2, 8($sp)
108     lw      $ra, 12($sp)
109     lw      $a0, 16($sp)
110     lw      $a1, 20($sp)
111     add     $s2, $s2, $v0
112     addi    $v0, $zero, 1       # to output the number
113     add     $a0, $s2, $zero
114     syscall
115     lw      $a0, 16($sp)
116     addi    $sp, $sp, 24
117     addi    $v0, $zero, 10      # for exit
118     syscall
119
120 ### Function countArray ###
121 countArray:
122     addi    $sp, $sp, -24       # adjust the stack for 6 items
123     sw      $a0, 20($sp)        # save function arguments
124     sw      $a1, 16($sp)
125     sw      $a3, 12($sp)
126     sw      $ra, 8($sp)         # save return address
127     addi    $s0, $a1, -1        # $s0 for i = numElements - 1
128     addi    $s1, $zero, 0       # $s1 for cnt = 0
129 cntLoop:
130     sw      $s0, 4($sp)         # store $s0 as i, not address
131     sw      $s1, 0($sp)
132     sll     $s0, $s0, 2         # $s0 = $s0 * 4
133     add     $s0, $a0, $s0       # $s0 is the address of A[i]
134     lw      $a0, 0($s0)         # $a0 = A[i]
135     addi    $t0, $zero, 1       # $t0 = 1
136     bne        $a2, $t0, sCase2    # if $a2 != 1, then not hot
137     jal        hot                    # jump to hot and save position to $ra
138     j       switchBreak            # jump to switchBreak
139 sCase2:
140     addi    $t0, $zero, -1      # $t0 = -1
141     bne     $a2, $t0, sDefault  # if $a2 != -1, then not cold
142     jal        cold                # jump to cold and save position to $ra
143     j       switchBreak
144 sDefault:
```

```
145     jal         comfort                 # jump to comfort and save position to
        ↪ $ra
146 switchBreak:
147     lw      $s1, 0($sp)
148     lw      $s0, 4($sp)
149     lw      $a3, 12($sp)
150     lw      $a1, 16($sp)
151     lw      $a0, 20($sp)
152     add     $s1, $s1, $v0       # cnt += $v0
153     addi    $s0, $s0, -1        # i--
154     slti    $t0, $s0, 0         # if $s0 < 0, $t0 = 0
155     beq        $t0, $zero, cntLoop    # if $t0 != 0 then continue the loop
156     lw      $ra, 8($sp)         # else, exit the loop, restore $ra
157     add     $v0, $s1, $zero     # $v0 = cnt
158     addi    $sp, $sp, 24        # destroy spaces on stack
159     jr      $ra
160
161
162 ### Function hot ###
163 hot:
164     slti    $v0, $a0, 30
165     beq     $v0, $zero, hotTrue
166     add     $v0, $zero, $zero   # A[i] < 30, $v0 = 0
167     jr      $ra
168 hotTrue:
169     addi    $v0, $zero, 1       # A[i] >= 30, $v0 = 1
170     jr      $ra
171
172 ### Function cold ###
173 cold:
174     slti    $t0, $a0, 5         # A[i] < 5, $t0 = 1
175     beq     $t0, $zero, coldFalse
176 coldTrue:
177     addi    $v0, $zero, 1       # $v0 = 1
178     jr      $ra
179 coldFalse:
180     addi    $t1, $zero, 5
181     beq        $a0, $t1, coldTrue    # if $a0 == $t1 == 5 then coldTrue
182     add        $v0, $zero, $zero    # $v0 = $zero + $zero = 0
183     jr      $ra
184
185 ### Function comfort ###
186 comfort:
187     slti    $t0, $a0, 30
188     addi    $t1, $zero, 1
189     beq        $t0, $t1, comfortTrue    # if $t0 == 1, $a0 < 30, then
        ↪ comfortTrue
190 comfortFalse:
191     add     $v0, $zero, $zero
```

```
192     jr      $ra
193 comfortTrue:
194     slti    $t0, $a0, 5
195     beq        $t0, $t1, comfortFalse    # if £t0 == 1, £a0 < 5 then
        ↪  comfortFalse
196     addi    $t2, $zero, 5
197     beq        $a0, $t2, comfortFalse    # if £a0 == 5 comfortFalse
198     addi    $v0, $zero, 1
199     jr      $ra
200
```