

VE370 Project 1

Liu Yihao 515370910207

1 Introduction

Develop a MIPS assembly program that operates on a data segment consisting of an array of 32-bit unsigned integers. In the text (program) segment of memory, write a procedure called main that implements the main() function and 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 20 elements.

```
1  main() {
2      int size = ...; //determine the size of the array here
3      int PassCnt, FailCnt;
4      int testArray[size] = { 55, 83,
5          ... //compose your own array here
6      };
7      PassCnt = countArray(testArray, size, 1);
8      FailCnt = countArray(testArray, size, -1);
9  }
10
11  int countArray(int A[], int numElements, int cntType) {
12  /*******
13   * Count specific elements in the integer array A[] whose size is      *
14   * numElements and return the following:                                *
15   *                                                                      *
16   * When cntType = 1, count the elements greater than or equal to 60; *
17   * When cntType = -1, count the elements less than 60;                *
18   *****/
19      int i, cnt = 0;
20      for(i = numElements - 1; i > -1; i--) {
21          switch (cntType) {
22              case '1' : cnt += Pass(A[i]); break;
23              otherwise: cnt += Fail(A[i]);
24          }
25      }
26      return cnt;
27  }
28
29  int Pass(int x) {
30      if(x >= 60) return 1;
31      else return 0;
```

```

32 }
33
34 int Fail(int x) {
35     if (x < 60) return 1;
36     else return 0;
37 }

```

2 Simulation result

I use the data [55, 83, 21, 20, 40, 49, 42, 35, 92, 8, 65, 88, 25, 100, 43, 9, 98, 10, 81, 63, 83, 27, 42, 81, 94, 2, 40, 49, 75, 46, 67, 46, 89, 27, 39, 12, 19, 41, 86, 3, 14, 64, 22, 64, 8, 38, 32, 26, 64, 5], there are 50 in total, 18 pass, 22 fail.

The simulation result is shown in Figure 1.

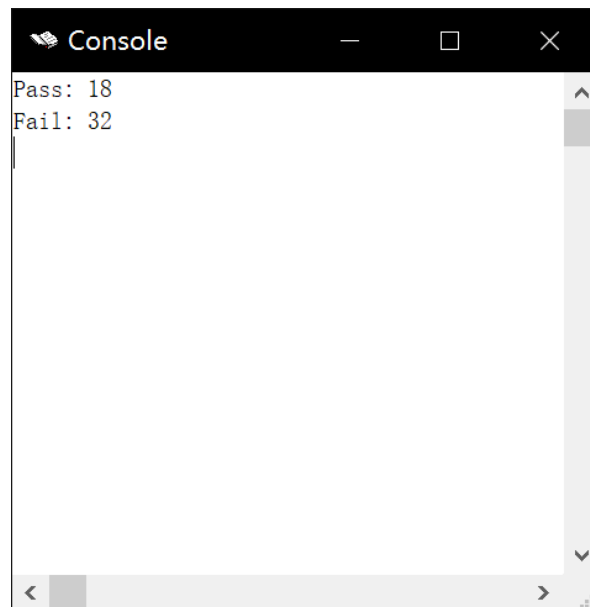


Figure 1: Simulation result

3 Appendix

```

1      .text
2      .globl __start
3  __start:
4      addi $sp, $sp, -212    # adjust stack for 50*4+12 item
5
6      addi $s0, $0, 50      # int size = 50
7      addu $s1, $0, $0      # int PassCnt = 0
8      addu $s2, $0, $0      # int FailCnt = 0
9      addu $s3, $0, $sp     # char buffer[10]

```

10	<code>addiu \$s4, \$s3, 12</code>	<code># int testArray[size]</code>
11		
12	<code>addi \$t0, \$0, 55</code>	<code># lt0 = 55</code>
13	<code>sw \$t0, 0(\$s4)</code>	<code># testArray[0] = lt0</code>
14	<code>addi \$t0, \$0, 83</code>	<code># lt0 = 83</code>
15	<code>sw \$t0, 4(\$s4)</code>	<code># testArray[1] = lt0</code>
16	<code>addi \$t0, \$0, 21</code>	<code># lt0 = 21</code>
17	<code>sw \$t0, 8(\$s4)</code>	<code># testArray[2] = lt0</code>
18	<code>addi \$t0, \$0, 20</code>	<code># lt0 = 20</code>
19	<code>sw \$t0, 12(\$s4)</code>	<code># testArray[3] = lt0</code>
20	<code>addi \$t0, \$0, 40</code>	<code># lt0 = 40</code>
21	<code>sw \$t0, 16(\$s4)</code>	<code># testArray[4] = lt0</code>
22	<code>addi \$t0, \$0, 49</code>	<code># lt0 = 49</code>
23	<code>sw \$t0, 20(\$s4)</code>	<code># testArray[5] = lt0</code>
24	<code>addi \$t0, \$0, 42</code>	<code># lt0 = 42</code>
25	<code>sw \$t0, 24(\$s4)</code>	<code># testArray[6] = lt0</code>
26	<code>addi \$t0, \$0, 35</code>	<code># lt0 = 35</code>
27	<code>sw \$t0, 28(\$s4)</code>	<code># testArray[7] = lt0</code>
28	<code>addi \$t0, \$0, 92</code>	<code># lt0 = 92</code>
29	<code>sw \$t0, 32(\$s4)</code>	<code># testArray[8] = lt0</code>
30	<code>addi \$t0, \$0, 8</code>	<code># lt0 = 8</code>
31	<code>sw \$t0, 36(\$s4)</code>	<code># testArray[9] = lt0</code>
32	<code>addi \$t0, \$0, 65</code>	<code># lt0 = 65</code>
33	<code>sw \$t0, 40(\$s4)</code>	<code># testArray[10] = lt0</code>
34	<code>addi \$t0, \$0, 88</code>	<code># lt0 = 88</code>
35	<code>sw \$t0, 44(\$s4)</code>	<code># testArray[11] = lt0</code>
36	<code>addi \$t0, \$0, 25</code>	<code># lt0 = 25</code>
37	<code>sw \$t0, 48(\$s4)</code>	<code># testArray[12] = lt0</code>
38	<code>addi \$t0, \$0, 100</code>	<code># lt0 = 100</code>
39	<code>sw \$t0, 52(\$s4)</code>	<code># testArray[13] = lt0</code>
40	<code>addi \$t0, \$0, 43</code>	<code># lt0 = 43</code>
41	<code>sw \$t0, 56(\$s4)</code>	<code># testArray[14] = lt0</code>
42	<code>addi \$t0, \$0, 9</code>	<code># lt0 = 9</code>
43	<code>sw \$t0, 60(\$s4)</code>	<code># testArray[15] = lt0</code>
44	<code>addi \$t0, \$0, 98</code>	<code># lt0 = 98</code>
45	<code>sw \$t0, 64(\$s4)</code>	<code># testArray[16] = lt0</code>
46	<code>addi \$t0, \$0, 10</code>	<code># lt0 = 10</code>
47	<code>sw \$t0, 68(\$s4)</code>	<code># testArray[17] = lt0</code>
48	<code>addi \$t0, \$0, 81</code>	<code># lt0 = 81</code>
49	<code>sw \$t0, 72(\$s4)</code>	<code># testArray[18] = lt0</code>
50	<code>addi \$t0, \$0, 63</code>	<code># lt0 = 63</code>
51	<code>sw \$t0, 76(\$s4)</code>	<code># testArray[19] = lt0</code>
52	<code>addi \$t0, \$0, 83</code>	<code># lt0 = 83</code>
53	<code>sw \$t0, 80(\$s4)</code>	<code># testArray[20] = lt0</code>
54	<code>addi \$t0, \$0, 27</code>	<code># lt0 = 27</code>
55	<code>sw \$t0, 84(\$s4)</code>	<code># testArray[21] = lt0</code>
56	<code>addi \$t0, \$0, 42</code>	<code># lt0 = 42</code>
57	<code>sw \$t0, 88(\$s4)</code>	<code># testArray[22] = lt0</code>
58	<code>addi \$t0, \$0, 81</code>	<code># lt0 = 81</code>

59	sw \$t0, 92(\$s4)	# testArray[23] = \$t0
60	addi \$t0, \$0, 94	# \$t0 = 94
61	sw \$t0, 96(\$s4)	# testArray[24] = \$t0
62	addi \$t0, \$0, 2	# \$t0 = 2
63	sw \$t0, 100(\$s4)	# testArray[25] = \$t0
64	addi \$t0, \$0, 40	# \$t0 = 40
65	sw \$t0, 104(\$s4)	# testArray[26] = \$t0
66	addi \$t0, \$0, 49	# \$t0 = 49
67	sw \$t0, 108(\$s4)	# testArray[27] = \$t0
68	addi \$t0, \$0, 75	# \$t0 = 75
69	sw \$t0, 112(\$s4)	# testArray[28] = \$t0
70	addi \$t0, \$0, 46	# \$t0 = 46
71	sw \$t0, 116(\$s4)	# testArray[29] = \$t0
72	addi \$t0, \$0, 67	# \$t0 = 67
73	sw \$t0, 120(\$s4)	# testArray[30] = \$t0
74	addi \$t0, \$0, 46	# \$t0 = 46
75	sw \$t0, 124(\$s4)	# testArray[31] = \$t0
76	addi \$t0, \$0, 89	# \$t0 = 89
77	sw \$t0, 128(\$s4)	# testArray[32] = \$t0
78	addi \$t0, \$0, 27	# \$t0 = 27
79	sw \$t0, 132(\$s4)	# testArray[33] = \$t0
80	addi \$t0, \$0, 39	# \$t0 = 39
81	sw \$t0, 136(\$s4)	# testArray[34] = \$t0
82	addi \$t0, \$0, 12	# \$t0 = 12
83	sw \$t0, 140(\$s4)	# testArray[35] = \$t0
84	addi \$t0, \$0, 19	# \$t0 = 19
85	sw \$t0, 144(\$s4)	# testArray[36] = \$t0
86	addi \$t0, \$0, 41	# \$t0 = 41
87	sw \$t0, 148(\$s4)	# testArray[37] = \$t0
88	addi \$t0, \$0, 86	# \$t0 = 86
89	sw \$t0, 152(\$s4)	# testArray[38] = \$t0
90	addi \$t0, \$0, 3	# \$t0 = 3
91	sw \$t0, 156(\$s4)	# testArray[39] = \$t0
92	addi \$t0, \$0, 14	# \$t0 = 14
93	sw \$t0, 160(\$s4)	# testArray[40] = \$t0
94	addi \$t0, \$0, 64	# \$t0 = 64
95	sw \$t0, 164(\$s4)	# testArray[41] = \$t0
96	addi \$t0, \$0, 22	# \$t0 = 22
97	sw \$t0, 168(\$s4)	# testArray[42] = \$t0
98	addi \$t0, \$0, 64	# \$t0 = 64
99	sw \$t0, 172(\$s4)	# testArray[43] = \$t0
100	addi \$t0, \$0, 8	# \$t0 = 8
101	sw \$t0, 176(\$s4)	# testArray[44] = \$t0
102	addi \$t0, \$0, 38	# \$t0 = 38
103	sw \$t0, 180(\$s4)	# testArray[45] = \$t0
104	addi \$t0, \$0, 32	# \$t0 = 32
105	sw \$t0, 184(\$s4)	# testArray[46] = \$t0
106	addi \$t0, \$0, 26	# \$t0 = 26
107	sw \$t0, 188(\$s4)	# testArray[47] = \$t0

```

108      addi $t0, $0, 64          # $t0 = 64
109      sw $t0, 192($s4)         # testArray[48] = $t0
110      addi $t0, $0, 5          # $t0 = 5
111      sw $t0, 196($s4)         # testArray[49] = $t0
112
113      addu $a0, $0, $s4         # $a0 = testArray
114      addu $a1, $0, $s0         # $a1 = size
115      addi $a2, $0, 1          # $a2 = 1
116      jal countArray           # $v0 = countArray(testArray, size, 1)
117      addi $t1, $0, 1          # wait for delay
118      addu $s5, $0, $v0        # save the result into $s4
119
120      addi $t0, $0, 80          # Init the string "Pass: "
121      sb $t0, 0($s3)
122      addi $t0, $0, 97
123      sb $t0, 1($s3)
124      addi $t0, $0, 115
125      sb $t0, 2($s3)
126      sb $t0, 3($s3)
127      addi $t0, $0, 58
128      sb $t0, 4($s3)
129      addi $t0, $0, 32
130      sb $t0, 5($s3)
131      addi $t0, $0, 0
132      sb $t0, 6($s3)
133      addiu $a0, $s3, 0         # $a0 = $s3 ("Pass: ")
134      addi $v0, $0, 4           # prepare to string output (system call 4)
135      syscall                  # string output
136
137      addu $a0, $0, $s5         # $a0 = $s4
138      addi $v0, $0, 1           # prepare to int output (system call 1)
139      syscall                  # int output
140
141      addi $t0, $0, 10          # Init the string "\n"
142      sb $t0, 0($s3)
143      addi $t0, $0, 0
144      sb $t0, 1($s3)
145      addiu $a0, $s3, 0         # $a0 = $s3 ("\n")
146      addi $v0, $0, 4           # prepare to string output (system call 4)
147      syscall                  # string output
148
149      addu $a0, $0, $s4         # $a0 = testArray
150      addu $a1, $0, $s0         # $a1 = size
151      addi $a2, $0, -1          # $a2 = -1
152      jal countArray           # $v0 = countArray(testArray, size, -1)
153      addi $t1, $0, 1          # wait for delay
154      addu $s5, $0, $v0        # save the result into $s4
155
156      addi $t0, $0, 70          # Init the string "Fail: "

```

```

157     sb $t0, 0($s3)
158     addi $t0, $0, 97
159     sb $t0, 1($s3)
160     addi $t0, $0, 105
161     sb $t0, 2($s3)
162     addi $t0, $0, 108
163     sb $t0, 3($s3)
164     addi $t0, $0, 58
165     sb $t0, 4($s3)
166     addi $t0, $0, 32
167     sb $t0, 5($s3)
168     addi $t0, $0, 0
169     sb $t0, 6($s3)
170     addiu $a0, $s3, 0      # $a0 = $s3 ("Fail: ")
171     addi $v0, $0, 4        # prepare to string output (system call 4)
172     syscall               # string output
173     addu $a0, $0, $s5      # $a0 = $v0
174     addi $v0, $0, 1        # prepare to int output (system call 1)
175     syscall               # int output
176
177     addi $t0, $0, 10      # Init the string "\n"
178     sb $t0, 0($s3)
179     addi $t0, $0, 0
180     sb $t0, 1($s3)
181     addiu $a0, $s3, 0      # $a0 = $s3 ("\n")
182     addi $v0, $0, 4        # prepare to string output (system call 4)
183     syscall               # string output
184
185     jal exit
186     addi $t0, $0, 0
187 countArray:
188     addi $sp, $sp, -24     # adjust stack for 6 items
189     sw $ra, 20($sp)        # save $ra on stack
190     sw $s4, 16($sp)        # save $s4 on stack
191     sw $s3, 12($sp)        # save $s3 on stack
192     sw $s2, 8($sp)         # save $s2 on stack
193     sw $s1, 4($sp)         # save $s1 on stack
194     sw $s0, 0($sp)         # save $s0 on stack
195
196     addu $s0, $0, $a0      # save $a0(int A[]) into $s0
197     addu $s1, $0, $a1      # save $a1(int numElements) into $s1
198     addu $s2, $0, $a2      # save $a2(int cntType) into $s2
199
200     addi $s3, $s1, -1      # $s3(i) = numElements - 1
201     addi $s4, $0, 0        # $s4(cnt) = 0
202 countArrayFor:
203     addi $t0, $0, 0        # wait for delay
204     slt $t0, $s3, $0       # $t0 = i < 0
205     bne $t0, $0, countArrayEndFor

```

```

206                                     # if (lt0 != 0) goto countArrayEndFor
207     sll $t0, $s3, 2                 # lt0 = i * 4
208     add $t0, $s0, $t0               # lt0 = A + lt0
209     lw $a0, 0($t0)                 # la0 = A[i]
210     addi $t1, $0, 1                 # lt1 = 1
211     addi $t1, $0, 1                 # wait for delay
212     bne $s2, $t1, countArrayElse
213                                     # if (cntType != 1) goto countArrayElse
214     jal Pass                        # lv0 = Pass(A[i])
215     addi $t1, $0, 1                 # wait for delay
216     j countArrayEndIf              # jump to endif
217     addi $t0, $0, 0                 # wait for delay
218 countArrayElse:
219     addi $t0, $0, 0                 # wait for delay
220     jal Fail                        # lv0 = Fail(A[i])
221     addi $t1, $0, 1                 # wait for delay
222 countArrayEndIf:
223     addi $t0, $0, 0                 # wait for delay
224     addu $s4, $s4, $v0              # cnt += lv0
225     addi $s3, $s3, -1              # i--
226     j countArrayFor                # jump to for begin
227     addi $t0, $0, 0                 # wait for delay
228 countArrayEndFor:
229     addi $t0, $0, 0                 # wait for delay
230     addu $v0, $0, $s4              # lv0 = cnt
231     lw $s0, 0($sp)                 # restore $s0 from stack
232     lw $s1, 4($sp)                 # restore $s1 from stack
233     lw $s2, 8($sp)                 # restore $s2 from stack
234     lw $s3, 12($sp)                # restore $s3 from stack
235     lw $s4, 16($sp)                # restore $s4 from stack
236     lw $ra, 20($sp)                # restore $ra from stack
237     addi $sp, $sp, 24              # recover the stack
238     addi $t1, $0, 0                 # wait for delay
239     jr $ra                         # return
240     addi $t0, $0, 0                 # wait for delay
241 Pass:
242     addi $t0, $0, 60                # lt0 = 60
243     slt $t1, $a0, $t0              # lt1 = x < 60
244     beq $t1, $0, PassIf            # if (lt1 == 1) goto PassIf
245     addi $v0, $0, 0                # lv0 = 0
246     jr $ra                         # return
247     addi $t0, $0, 0                 # wait for delay
248 PassIf:
249     addi $t0, $0, 0                 # wait for delay
250     addi $v0, $0, 1                # lv0 = 1
251     jr $ra                         # return
252     addi $t0, $0, 0                 # wait for delay
253 Fail:
254     addi $t0, $0, 60                # lt0 = 60

```

```

255      slt $t1, $a0, $t0      # $t1 = x < 60
256      bne $t1, $0, FailIf    # if ($t1 != 1) goto PassIf
257      addi $v0, $0, 0        # $v0 = 0
258      jr $ra                 # return
259      addi $t0, $0, 0        # wait for delay
260 FailIf:
261      addi $t0, $0, 0        # wait for delay
262      addi $v0, $0, 1        # $v0 = 1
263      jr $ra                 # return
264      addi $t0, $0, 0        # wait for delay
265 exit:
266      addi $v0, $0, 10       # prepare to exit (system call 10)
267      syscall                # exit

```