```
1  # MIPS Implementation of selection sort
2  # Brandon Ingli
3  # 4 March 2019
4
5  .data
6  prompt1:  .asciiz "Enter number of elements: "
7  prompt2:  .asciiz "Enter elements one per line:\n"
8  newline:  .asciiz "\n"
9
10       .align 2 #Make sure our words line up appropriately
11
12 list: .space 400 #Space for up to 100 words
13
14 #Begin Code
15 .text
16
17 # n -> $s0
18 # Base(list) -> $s1
19 # i -> $s3
20 # j -> $s4
21 # min_pos -> $s5
22 # temp -> $s6
23
24 main:
25       la $s1, list #Load address of list into $s1
26
27       # Prompt for number of elements
28       la $a0, prompt1 #load address of prompt 1 into first argument
29       li $v0, 4 #load print string syscall code
30       syscall #make the syscall
31
32       #Read number of elements, store in $s0
33       li $v0, 5 #read int syscall code
34       syscall #read int
35       move $s0, $v0 #move read int into $s0
36
37       # Prompt for elements
38       la $a0, prompt2 #load address of prompt 1 into first argument
39       li $v0, 4 #load print string syscall code
40       syscall #make the syscall
41
42       #Loop 1: Read ints into array
43       #i = 0
44       li $s3, 0
45 for1:
46       bge $s3, $s0, for1_exit #"for i < n;" branch if i >= n
```

```
47
48        #Calculate address of list[i]
49        sll $t0, $s3, 2 #t0 = i * 4 for offset
50        addu $t0, $t0, $s1 #to is now &list[i]
51
52        #read integer
53        li $v0, 5 #read int syscall code
54        syscall
55
56        sw $v0, 0($t0) #store that integer into list[i]
57
58        addi $s3, $s3, 1 #i++
59        j for1 #loop back
60
61  for1_exit:
62        #Loop 2: Outer loop of sort
63        li $s3, 0 #i=0
64  for2:
65        addi $t0, $s0, -1 #$t0 = n-1
66        bge $s3, $t0, for2_exit #"for i < n - 1;" branch if i >= n-1
67
68        move $s5, $s3 #min_pos = i
69
70        # Loop 3: Inner loop of sort
71        addi $s4, $s3, 1 #j = i + 1
72  for3:
73        bge $s4, $s0, for3_exit #"for j < n;" branch if j >= n
74
75        #Load list[j]
76        sll $t0, $s4, 2 #t0 = j * 4 for offset
77        addu $t0, $t0, $s1 #t0 = &list[j]
78        lw $t0, 0($t0) #t0 = list[j]
79
80        #load list[min_pos]
81        sll $t1, $s5, 2 #t1 = min_pos * 4 for offset
82        addu $t1, $t1, $s1 #t1 = &list[min_pos]
83        lw $t1, 0($t1) #t1 = list[min_pos]
84
85        #if list[j] < list[min_pos]
86        bge $t0, $t1, if1_exit #"if list[j] < list[min_pos];" branch when list[
87        move $s5, $s4 #min_pos = j
88        # No need to jump since there's no else
89  if1_exit:
90        addi $s4, $s4, 1 #j++
91        j for3 #loop again on inner loop
92  for3_exit:
93        #swap
94        #temp = list[i]
```

```
95        sll $t0, $s3, 2 #t0 = i *= 4 for offset
96        addu $t0, $t0, $s1 #t0 = &list[i]
97        lw $s6, 0($t0) #temp = list[i]
98
99        #list[i] = list[min_pos]
100       sll $t1, $s5, 2 #t1 = min_pos * 4 for offset
101       addu $t1, $t1, $s1 #t1 = &list[min_pos]
102       lw $t2, 0($t1) #t2 = list[min_pos]
103       sw $t2, 0($t0) #list[i] = list[min_pos]
104
105       sw $s6, 0($t1) #list[min_pos] = temp
106
107
108       addi $s3, $s3, 1 #i++
109
110       j for2 #loop again on outer loop
111 for2_exit:
112       #print "\n"
113       la $a0, newline
114       li $v0, 4
115       syscall
116
117       #Loop 4: Print out contents
118       li $s3, 0 #i = 0
119 for4:
120       bge $s3, $s0, for4_exit #"for i < n;" branch when i >= n
121
122       #Calculate &list[i]
123       sll $t0, $s3, 2 #t0 = i * 4 for offset
124       addu $t0, $t0, $s1 #t0 = &list[i]
125
126       #Print list[i]
127       lw $a0, 0($t0) #load list[i] into first argument
128       li $v0, 1 #set print int syscall code
129       syscall
130
131       #print "\n"
132       la $a0, newline
133       li $v0, 4
134       syscall
135
136       addi $s3, $s3, 1 #i++
137
138       j for4 #loop again
139
140 for4_exit:
141       jr $ra #exit
```