

# **Habib University**



Dhanani School of Science and Engineering

CE/CS 321/330 Computer Architecture

## **Final Lab Project**

**5-Stage Pipelined Processor To Execute A  
Single Array Sorting Algorithm**

### **Group Members**

Hammad Sajid (hs07606)

Muhammad Azeem Haider (mh06858)

## Contents

<b>1</b>	<b>Sorting Algorithm on a Single Cycle Processor</b>	<b>3</b>
1.1	Selection Sort Assembly Code . . . . .	3
1.2	Selection Sort Python Code . . . . .	4

# 1 Sorting Algorithm on a Single Cycle Processor

## 1.1 Selection Sort Assembly Code

```
1 addi x11, x0, 6 #an arbitrary value to append in array
2 addi x29, x0, 6 #initializing size of the array to be 6
3 addi x30, x0, 0 #initializing offset to store values in
  array after one another
4 addi x31, x0, 0 #initializing i = 0 to loop through array to
  enter values.
5 addi x28, x0, 6 #temporary reg for checking length
6
7 #The code below is to initialize random values in the array
8 Array:
9
10 sw x11, 0x100(x30) #store values in array
11 addi x31, x31, 1 #performs i = i + 1
12 addi x30, x30, 4 #offset + 4 to jump to next memory
  address to store value
13 addi x11, x11, -1 #subtracting 1 to add next value in
  array (6->5->4....)
14 beq x28, x31, filled #if i = size of array, stop.
15 beq x0, x0, Array
16
17 filled:
18
19 #After the above code, the array is [6,5,4,3,2,1]
20
21 addi x30, x0, 0 #i = 0 (for i loop)
22 addi x31, x30, 0 #j = 0
23 addi x29, x0, 0 #for offset calculation
24 addi x11, x0, 6
25
26 #Code below is for 1st i loop
27
28 I_Loop:
29
30 beq x11, x30, Sorted #if i = size of array, array has
  been sorted
31 add x10, x29, x0 #assigning min_index = i
32 addi x31, x30, 1 #j = j + 1
33 addi x28, x29, 4 #jump to next address
34
35 #Code below is for nested j loop
36 J_Loop:
37
38 beq x31, x11, Swap
39 lw x15, 0x100(x28) #load Array[j]
```

```

40 lw x16, 0x100(x10) #load Array[min_index]
41 blt x15, x16, If #if Array[j] < Array[min_index]
42
43 #The code below it to iterate through the jth loop
44
45 return:
46
47 addi x31, x31, 1 #perform j = j + 1
48 addi x28, x28, 4 #jump to next address
49 beq x0, x0, J_Loop #jump to nested j loop
50
51 #The code below is to iterate through ith loop.
52
53 jump_back:
54
55 addi x30, x30, 1 #perform i = i + 1
56 addi x28, x28, 4 #jump to next address
57 beq x0, x0, I_Loop #jump to first i loop.
58
59 #Code below is for min_index = j line.
60
61 If:
62
63 addi x10, x28, 0 #assign min_index = j
64 beq x0, x0, return #jump back to j loop
65
66 #Code below is to perform swapping
67
68 Swap:
69
70 lw x13, 0x100(x10) #load Array[min_index]
71 lw x14, 0x100(x29) #load Array[i]
72 sw x13, 0x100(x29) #Array[min_index] = Array[i]
73 sw x14, 0x100(x10) #Array[i] = Array[min_index]
74 addi x29, x29, 4 #add 4 in x29 so that it doesnot
    include sorted value
75 beq x0, x0, jump_back
76
77 Sorted:

```

Listing 1: Selection Sort Assembly code

## 1.2 Selection Sort Python Code

```

1 def selectionSort(array, size):
2
3 for ind in range(size):
4     min_index = ind
5

```

```
6     for j in range(ind + 1, size):
7         if array[j] < array[min_index]:
8             min_index = j
9     (array[ind], array[min_index]) = (array[min_index],
        array[ind])
```

Listing 2: Selection Sort Python Code (Taken from GeeksforGeeks)