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

1 Sorting Algorithm on a Single Cycle Processor

1.1 Selection Sort Assembly Code

```
addi x11, x0, 6 #an arbitrary value to append in array
addi x29, x0, 6 #initializing size of the array to be 6
addi x30, x0, 0 #initializing offset to store values in
    array after one another
addi x31, x0, 0 #initializing i = 0 to loop through array to
     enter values.
addi x28, x0, 6 #temporary reg for checking length

#The code below is to intialize random values in the array
Array:

    sw x11, 0x100(x30)  #store values in array
    addi x31, x31, 1 #performs i = i + 1
    addi x30, x30, 4 #offset + 4 to jump to next memory
        address to store value
    addi x11, x11, -1 #subtracting 1 to add next value in
        array (6->5->4....)
    beq x28, x31, filled #if i = size of array, stop.
    beq x0, x0, Array

filled:

#After the above code, the array is [6,5,4,3,2,1]

addi x30, x0, 0 #i = 0 (for i loop)
addi x31, x30, 0 #j = 0
addi x29, x0, 0 #for offset calculation
addi x11, x0, 6

#Code below is for 1st i loop

I_Loop:

    beq x11, x30, Sorted #if i = size of array, array has
        been sorted
    add x10, x29, x0  #assigning min_index = i
    addi x31, x30, 1 #j = j + 1
    addi x28, x29, 4 #jump to next address

#Code below is for nested j loop
J_Loop:

    beq x31, x11, Swap
    lw x15, 0x100(x28) #load Array[j]
```

```
    lw x16, 0x100(x10) #load Array[min_index]
    blt x15, x16, If  #if Array[j] < Array[min_index]

    #The code below it to iterate through the jth loop

    return:

    addi x31, x31, 1 #perform j = j + 1
    addi x28, x28, 4 #jump to next address
    beq x0, x0, J_Loop #jump to nested j loop

    #The code below is to iterate through ith loop.

    jump_back:

    addi x30, x30, 1 #perform i = i + 1
    addi x28, x28, 4 #jump to next address
    beq x0, x0, I_Loop #jump to first i loop.

#Code below is for min_index = j line.

If:

    addi x10, x28, 0 #assign min_index = j
    beq x0, x0, return  #jump back to j loop

#Code below is to perform swapping

Swap:

    lw x13, 0x100(x10) #load Array[min_index]
    lw x14, 0x100(x29) #load Array[i]
    sw x13, 0x100(x29) #Array[min_index] = Array[i]
    sw x14, 0x100(x10) #Array[i] = Array[min_index]
    addi x29, x29, 4  #add 4 in x29 so that it doesnot
        include sorted value
    beq x0, x0, jump_back

Sorted:
```

Listing 1: Selection Sort Assembly code

1.2 Selection Sort Python Code

```
def selectionSort(array, size):

for ind in range(size):
    min_index = ind

```

```
    for j in range(ind + 1, size):
        # select the minimum element in every iteration
        if array[j] < array[min_index]:
            min_index = j
        # swapping the elements to sort the array
    (array[ind], array[min_index]) = (array[min_index],
        array[ind])
```

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