ES215: Assignment-2

Name: Daniel Giftson E

Roll No: 20110051

1) **MIPS32 Assembly Code:**

```
main:
    la $t0, arr          #load address of the array
    lw $t1, 0($t0)       #lowest is arr[0]
    lw $t2, 0($t0)       #highest is arr[0]
    addi $t3, $0, 0      #lowestindex = 0
    addi $t4, $0, 0      #highestindex = 0
    addi $t5, $0, 0      #sum = 0
    addi $t6, $0, 0      #i = 0
    addi $t7, $0, 101 #value of $t7 = 101
    addi $t8, $0, 1    #used for comaparing successive values of the array

while:
    beq $t6, $t7, avg_value    #if value at $t6 = $t7, it goes to avg_value function
    lw $s2, 0($t0)             #$s2 saves the value of arr[i]
    slt $s1, $t2, $s2          #checking the first if condition
    beq $s1, $t8, highest_val #if value at $s1 = $t8, it goes to highest_val function
    slt $s1, $s2, $t1          #checking the else if condition
    beq $s1, $t8, lowest_val  #if value at $s1 = $t8, it goes to lowest_val function
    add $t5, $t5, $s2          #updating sum
    addi $t6, $t6, 1           #incrementing i
    addi $t0, $t0, 4           #goes to the next address
    j while

highest_val:
     add $t2, $s2, $0
```

```
    add $t4, $t6, $0
    add $t5, $t5, $s2  #updating sum
    addi $t6, $t6, 1    #incrementing i
    addi $t0, $t0, 4   #goes to the next address
    j while

lowest_val:
    add $t1, $s2, $0
    add $t3, $t6, $0
    add $t5, $t5, $s2  #updating sum
    addi $t6, $t6, 1   #incrementing i
    addi $t0, $t0, 4   #goes to the next address
    j while

avg_value:
    div $t5, $t5, $t7  #Initially $t5 has the sum of all elements in the array which is divided by
                         101 and the result is stored back in $t5
    jr $ra              #End of the code
```

Grace Question:

1) **b) Observations:**

- I have taken the Integer Matrix Multiplication C++ program from Assignment -1 for generating .i, .s, .o, .out files (preprocessed file, compiled code, assembled code, and binary codes respectively).

- Size:
    1. mat_x86.i - 668 KB
    2. mat_x86.s - 6 KB
    3. mat_x86.o - 3.94 KB
    4. mat_x86.out - 14.4 KB

5. mat_mips.i - 662 KB
6. mat_mips.s - 11.9 KB
7. mat_mips.o - -Didn’t generate-
8. mat_mips.out - 12.8 KB

- Lines of Code:
  1. mat_x86.i - 28752
  2. mat_x86.s - 320
  3. mat_x86.o - 19
  4. mat_x86.out - 21
  5. mat_mips.i - 28204
  6. mat_mips.s - 675
  7. mat_mips.o - -Didn’t generate-
  8. mat_mips.out - 45