**Assembly Language (D and E)**

**Fall 2019**

**Assignment-4**

**Submission: Submit soft copies on slate. Submission deadline is November 30, 2019 5 PM. Clearly mention your roll number and section on submission folder**

**Question no 1 50**

Write an assembly code to split the screen in 2 halves (upper half and lower half) and rotate the upper half UP (scroll the upper half up and copy the 1st line to the bottom of upper half) and lower half DOWN (scroll the lower half down and copy the last line to the top of bottom half). Write two subroutines for this. First subroutine will rotate upper half up, and the second subroutine will rotate lower half down. Modify the timer interrupt so that when the timer interrupt comes for the first time, it calls the first subroutine. After the delay of 2 seconds, it calls the second subroutine. After 2 seconds, the first subroutine is called again and this process goes on infinitely.

Note: There are total 25 rows displayed on the screen. Upper Half consists of upper 12 rows and Lower Half consists of lower 12 rows. The middle row will be same throughout.

**Question no 2 50**

Assume that for some reason the top row of the keyboard containing the letters (QWERTYUIOP) stopped working. One solution is to program the row above(the number keys 1234567890) so that they behave like the first row character row e.g. when 1 is pressed INT 16h returns the scan code and the ASCII for character 'q'; when 2 is pressed INT 16h should return the scan code and ASCII for character 'w' and so on. Write the assembly language code for the mentioned solution. Write a test code for your solution as well.

**Question no 3 (Solve this question on paper and submit picture of that paper) 50**

Consider the following MIPS assembly language loop. Assume that we run this code on the 5 stages pipelined data path. Adder requires 2 cycles and Multiplier requires 3 cycles. Forwarding has been implemented.

1. Loop: ld R4, 0(R1)

2. mul R4, R4, R0

3. ld R5, 0(R2)

4. add R4, R4, R5

5. sw R4, 0(R2)

6. sub R3, R3, 1

7. addi R1, R1, 4

8. addi R2, R2, 4

9. bne R3, 0, Loop

a) Find all possible hazards in the above code. Write numbers of both instructions in front of the following if a specific hazard exists (e.g. 1 & 2, 3 & 4, .....)

**WAR:**

**WAW:**

**RAW:**

b) Add stalls in the above code. Find the number of clock cycles needed to execute this code (all iterations), accounting for all possible stalls. Assume that R3 is initially set to 10.