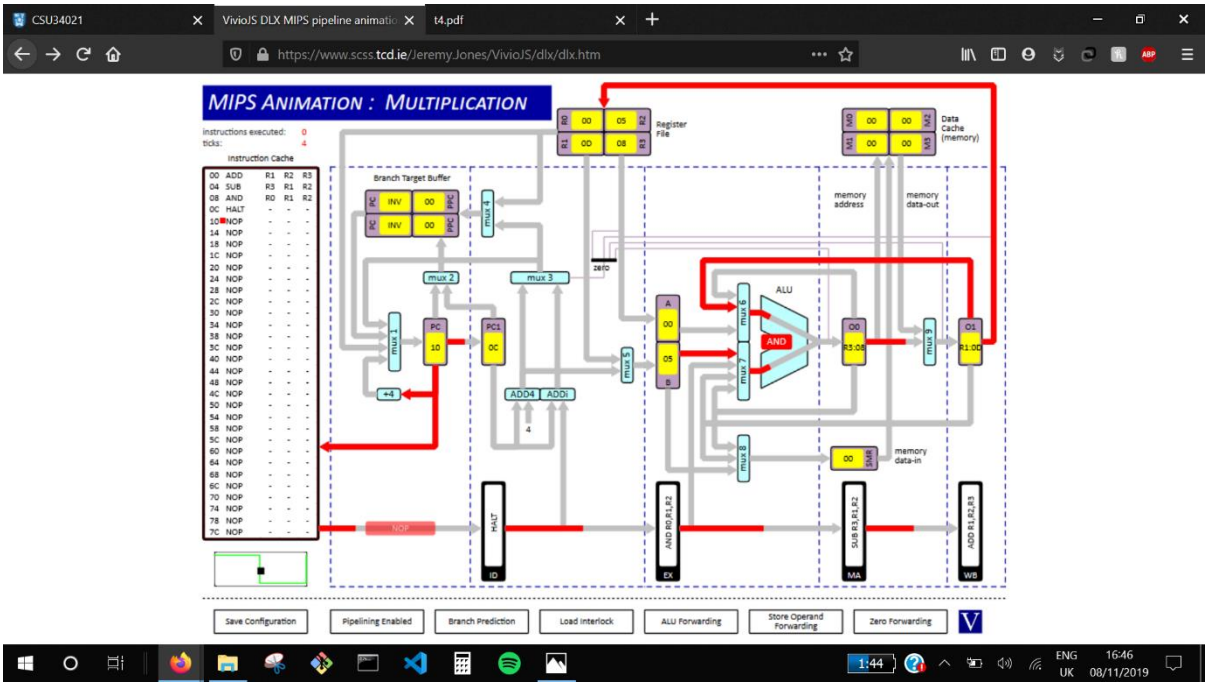
Name: Muhammad Rvail Naveed

Student No: 17321983

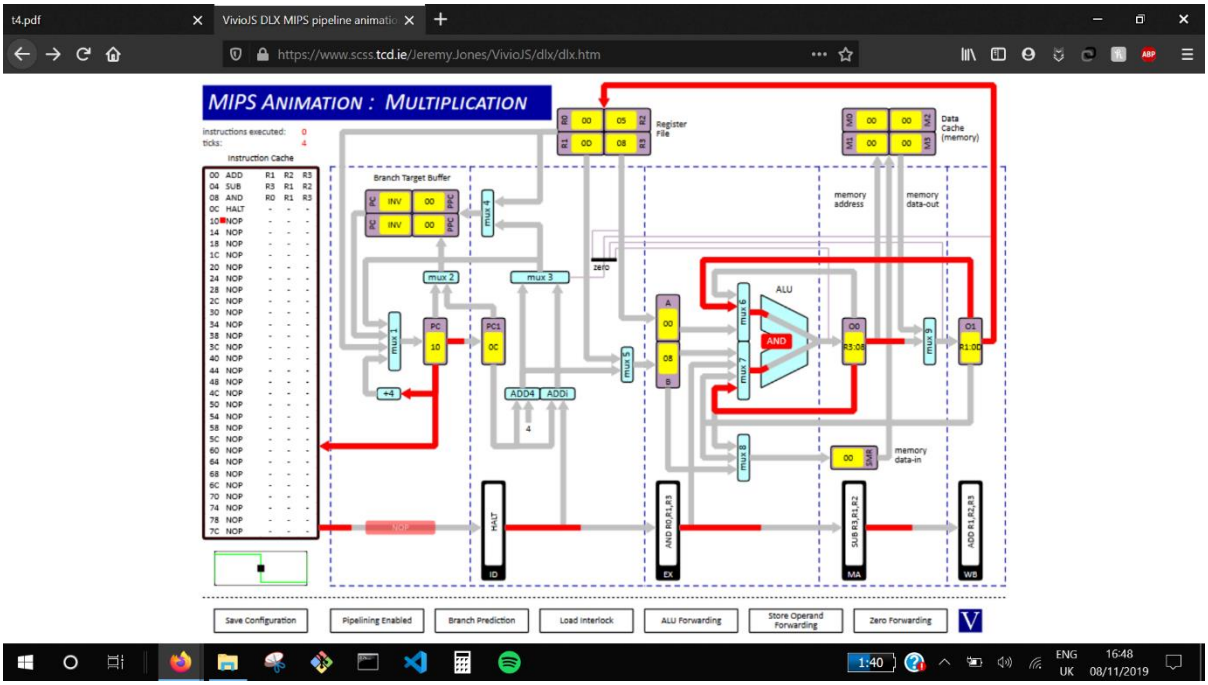
CSU34021 Tutorial 4

Question 1

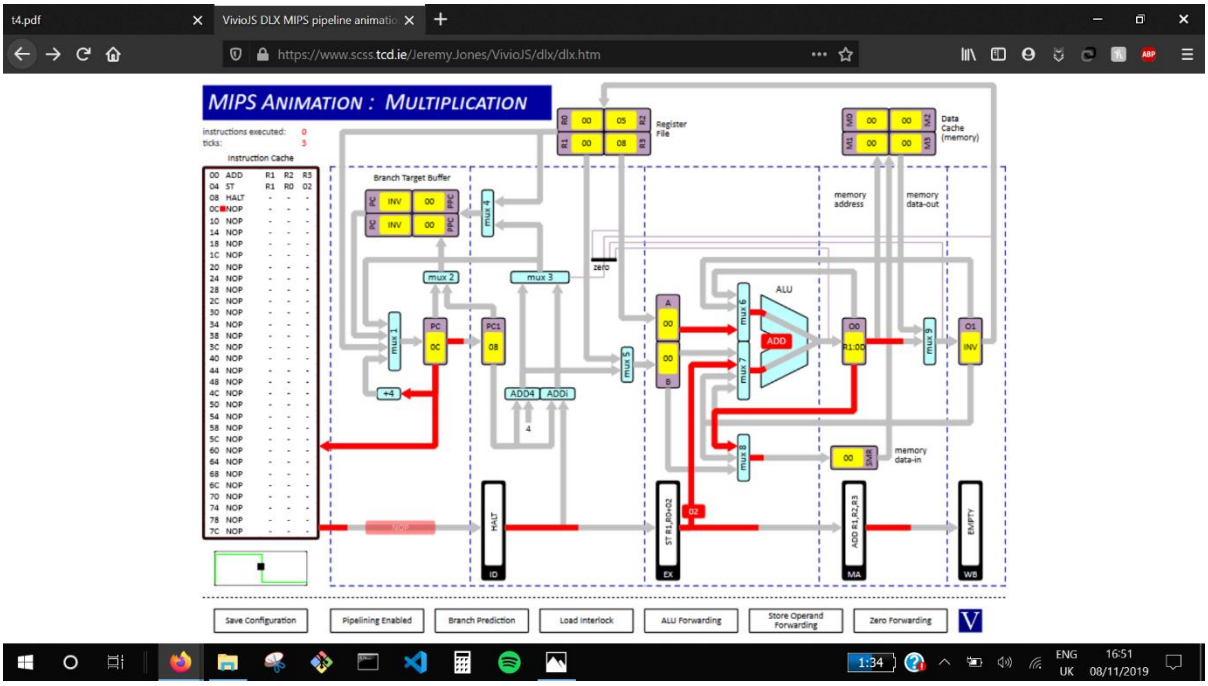
1. ADD R1, R2, R3
   SUB R3, R1, R2
   AND R0, R1, R2



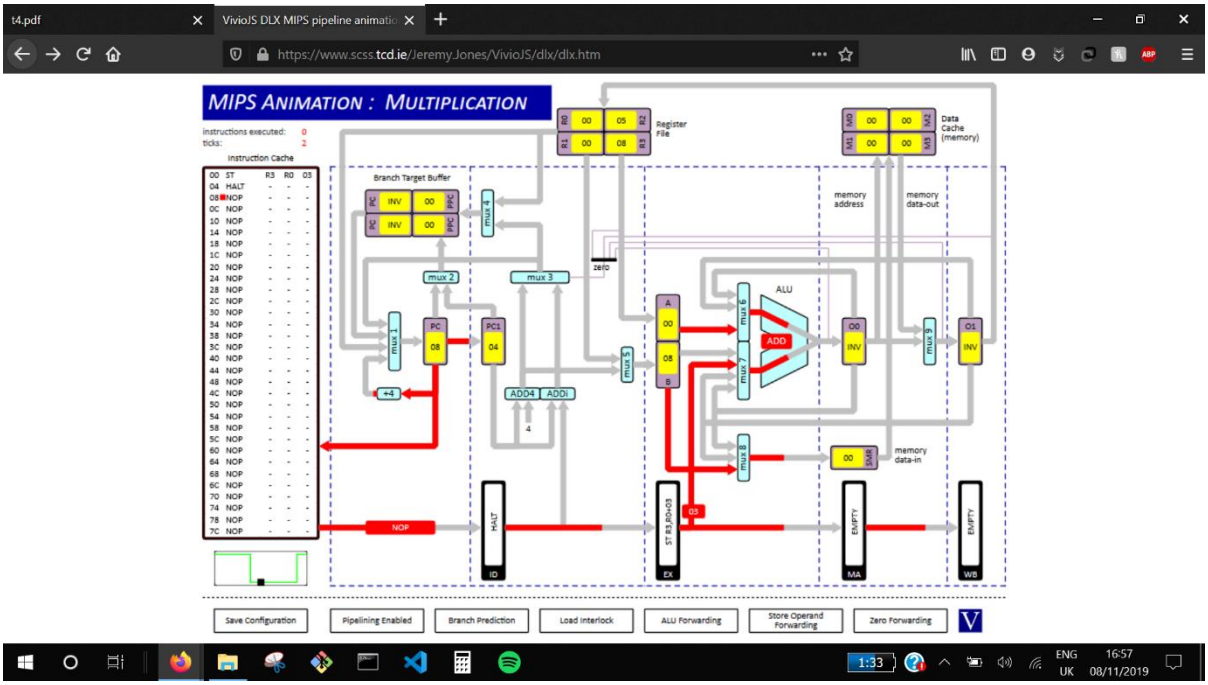
2. ADD R1, R2, R3
   SUB R3, R1, R2
   ADD R0, R1, R3



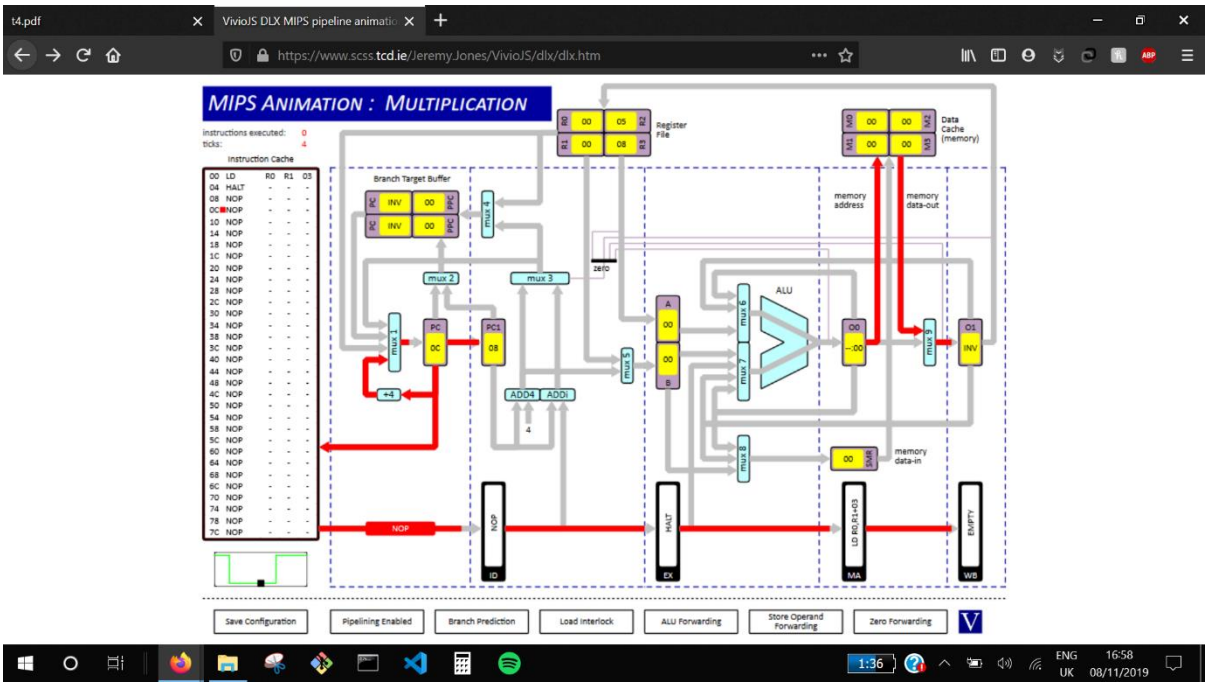
3. ADD R1, R2, R3
   ST R1, R0, 02



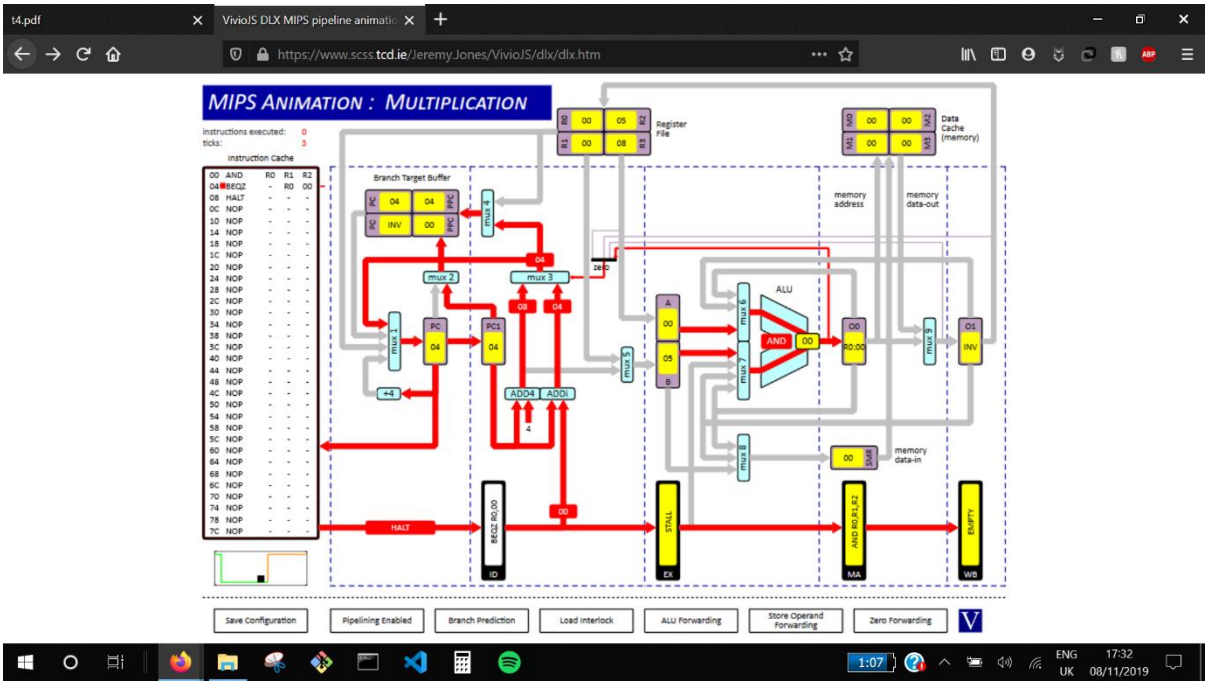
4. ST R3, R0, 03



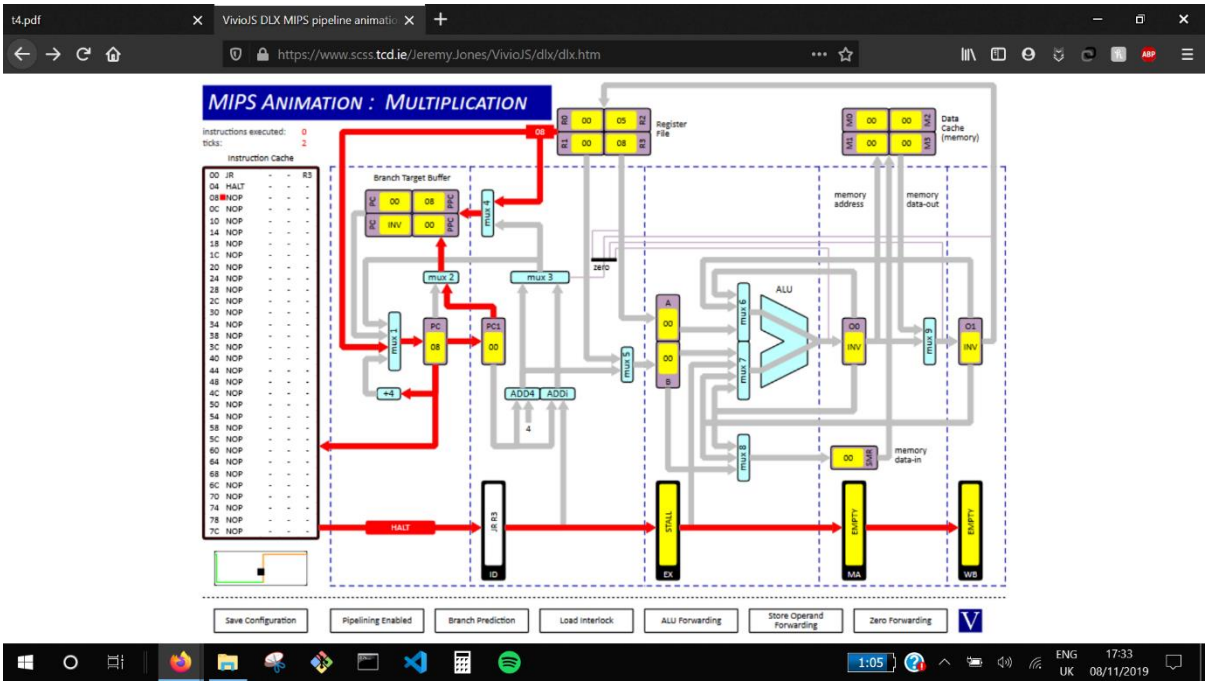
5. LD R0, R1, 03



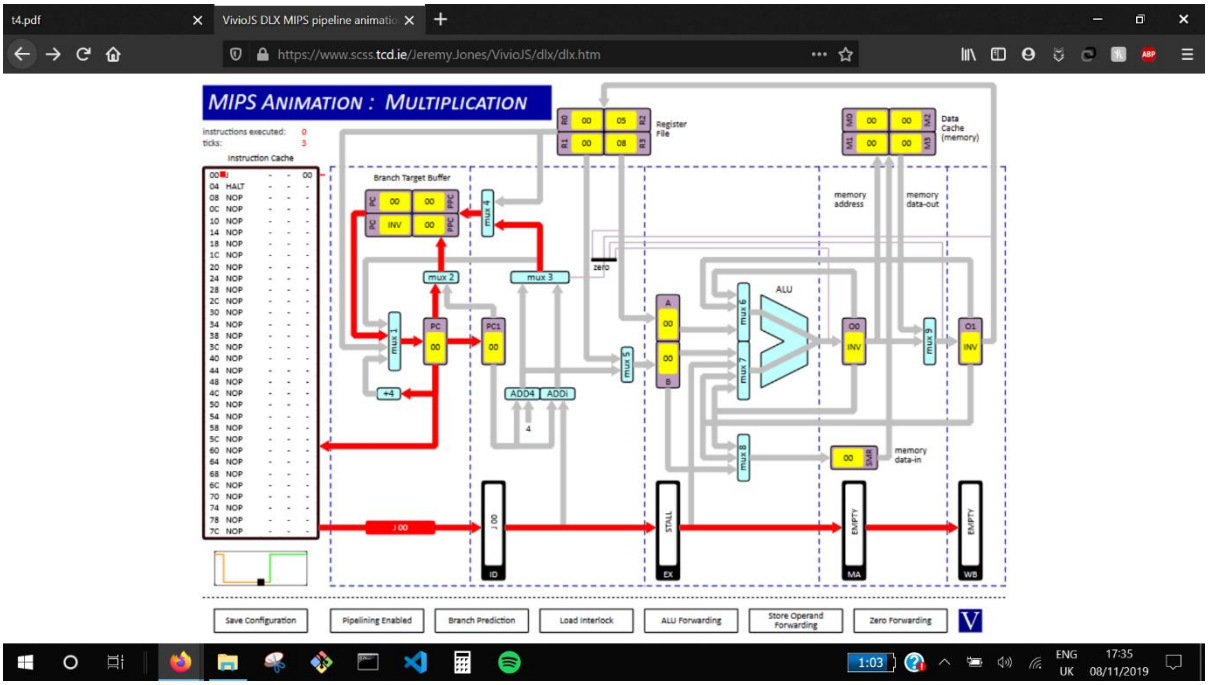
6. AND R0, R1, R2
   BEQZ R0, 00



7. JR R3



8. J 00



Question 2

i. R1 = 0x15, Clock cycles = 10
ii. R1 = 0x15, Clock cycles = 18
iii. R1 = 0x06, Clock cycles = 10

With ALU forwarding, results from instructions that have been previously executed can be stored in the immediate registers (O1 & O2) depending on where they are in the pipeline; Memory Access or Write Back. When ALU forwarding is disabled with the enabling of CPU data dependency interlocks, the CPU does not have to implement an extra 2 stalls between instructions. Since there are 4 instructions succeeding the first and between instructions there should be 2 stalls. In total making 8 stalls, the extra 8 being seen in (ii).

In the last case with ALU forwarding enabled and CPU data dependency interlocks disabled no stalls can be implemented like in (ii). This is because there are no registers to store the results from previous instructions. This occurrence is referred to as a **data hazard** and is seen as a problem in the instruction pipeline. Because of this the instructions executed incorrectly resulting in incorrect calculations.

Question 3

i. 38 instructions, 50 cycles

1 instruction takes 4 cycles to complete as this is a 4-stage pipeline. There is a pipeline stall when running LD and so another clock cycle is lost.

Every time a SRLi instruction is called after LD, a stall is placed between them to give the LD instruction time to complete. Due to data dependency of SRLi on LD and LD's 2 cycle completion requirement, there is a stall inserted. This instruction is performed 4 times, resulting in 4 extra clock cycles until finished.

Lastly when J is being decoded, the succeeding ST is fetched. A stall is placed before the jump to allow the ST to be removed and BEQZ, the correct instruction to be fetched. This happens 4 times, resulting in 4 clock cycles.

Adding up all the clock cycles from conditions outlined above: 4+4+4=12. 38-50 is 12 indicating the source of the extra clock cycles.

ii. 38 instructions, 53 cycles

There are 3 more clock cycles than part (i). This is because branch prediction is turned off and therefore the accuracy of loading is reduced significantly. This leads the last Jump instruction to be incorrect 4 times resulting in 4 stalls.

iii. 38 instructions, 46 cycles

8 more cycles than instructions. With instructions SRLi and SLLi swapped, a stall is avoided between SRLi and LD. SLLi does not require any operands used in LD so there is no need for it to wait one clock cycle for results to be stored in an immediate register. Since this LD occurs 4 times in (i), there is a reduction of 4 stalls, so this results in a reduction of difference between cycles and instructions. (Goes from 12 to 8).