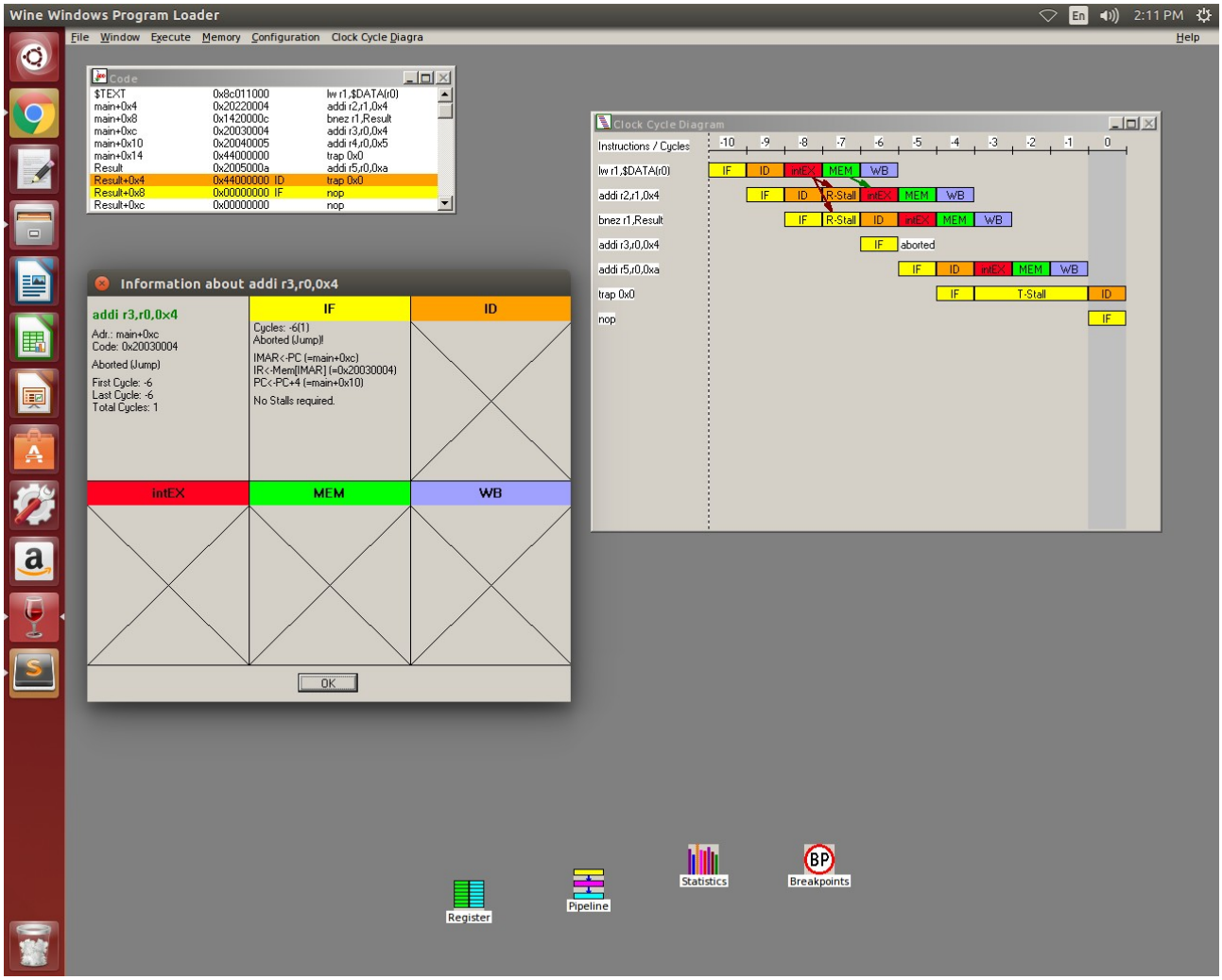
140050002
Question 0a

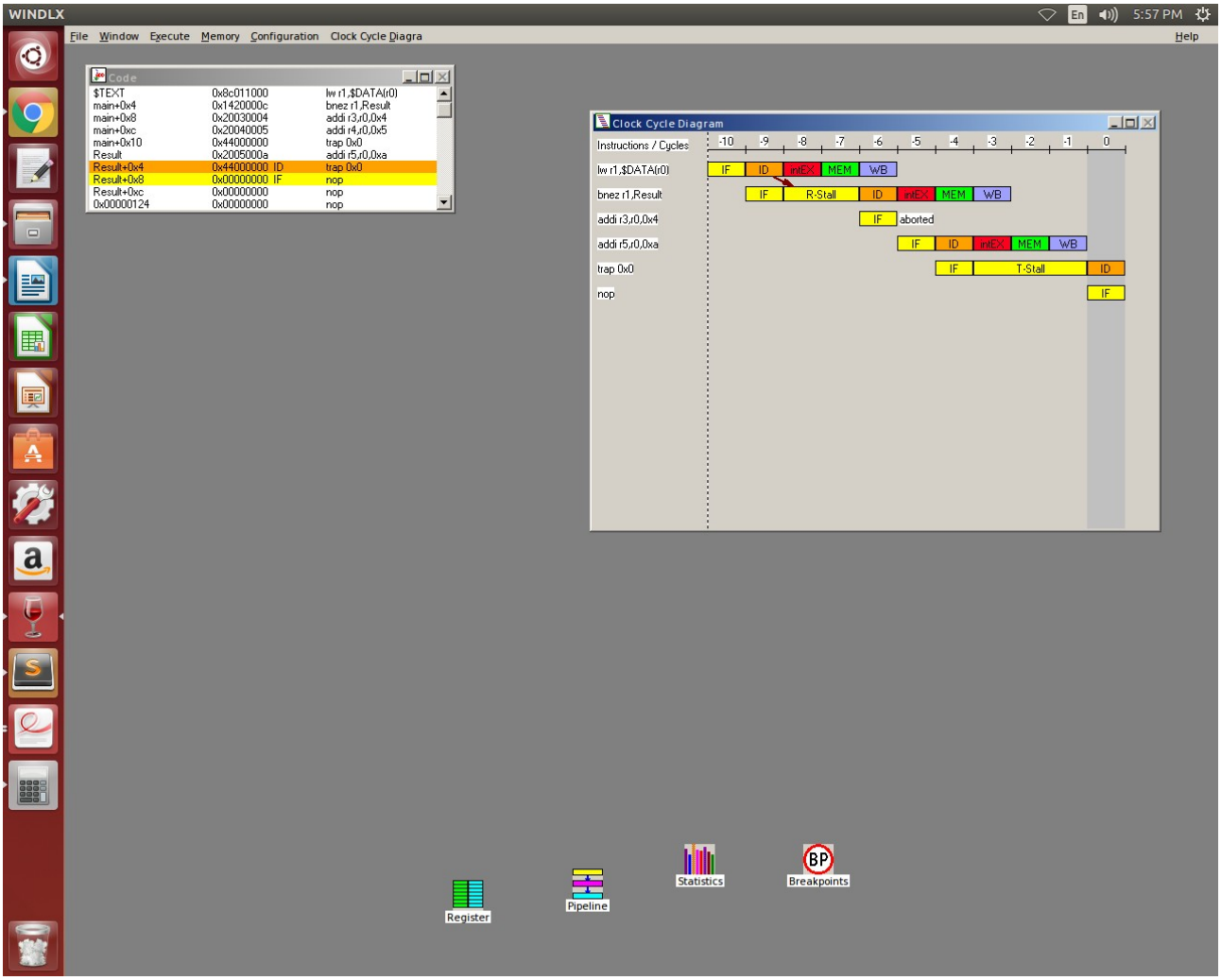


Here because of branch add instruction has been aborted.
Due to static prediction, instruction after bnez is executed but due to branching, previously started instruction must be aborted.

Question 0b

No such program is possible. Stall in ID in branch occurs only when values required for execution is not available. But branch doesn't do any execution in IntEx stage, stall is not possible.

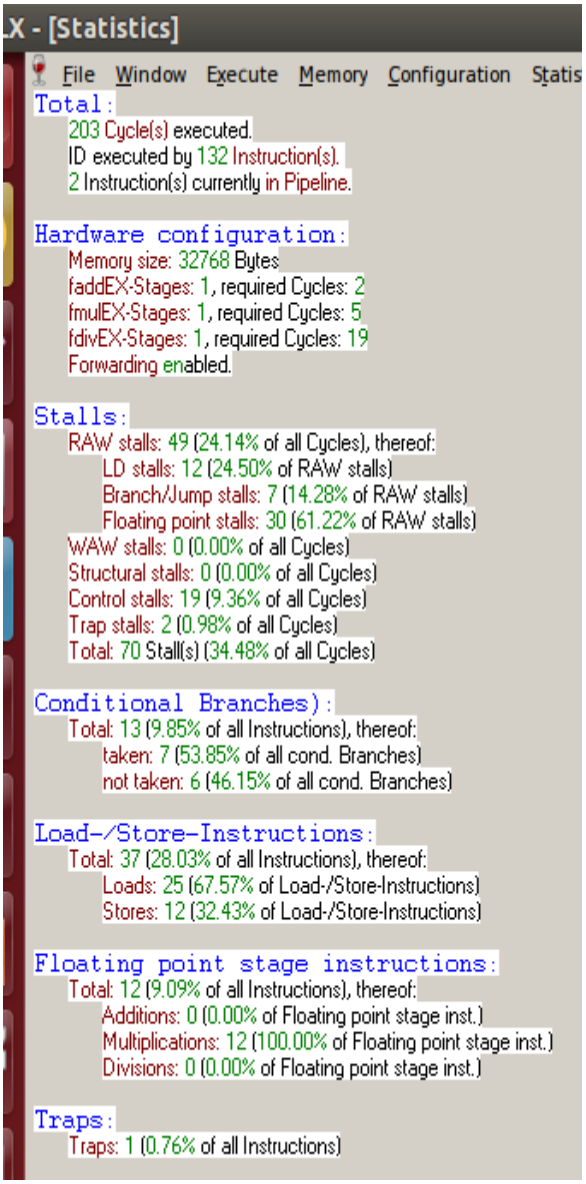
However stall in IF stage possible, because required registers are not available.



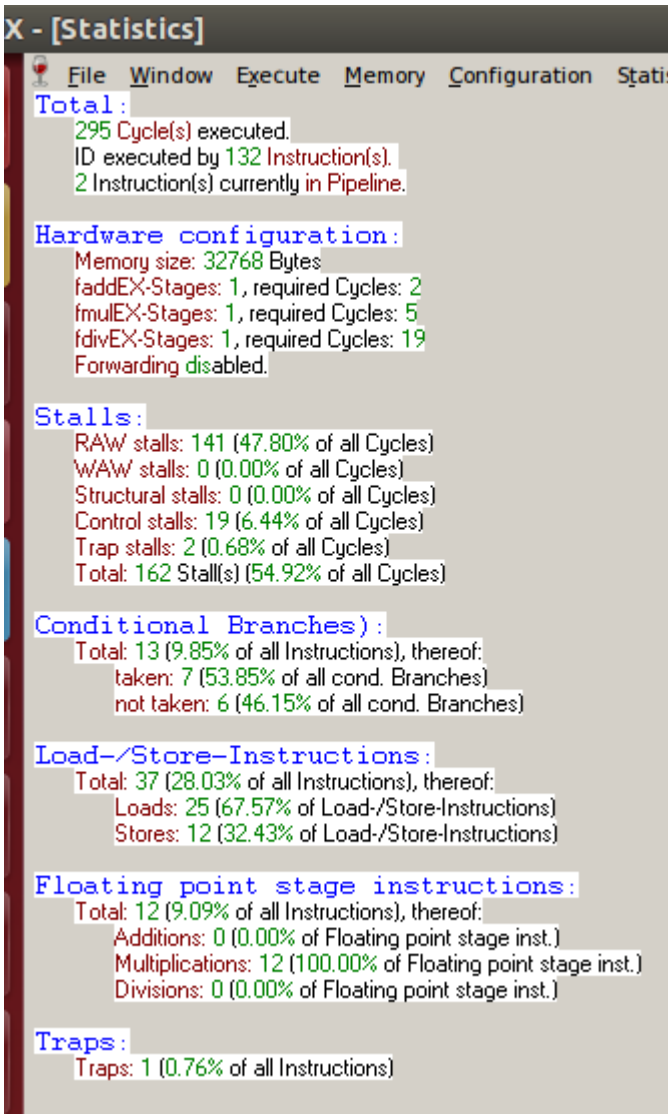
Question 1a

Number of clock cycle with forwarding : 203
Number of clock cycle without forwarding : 295
Speed up : 295 / 203 = 1.45

With forwarding :



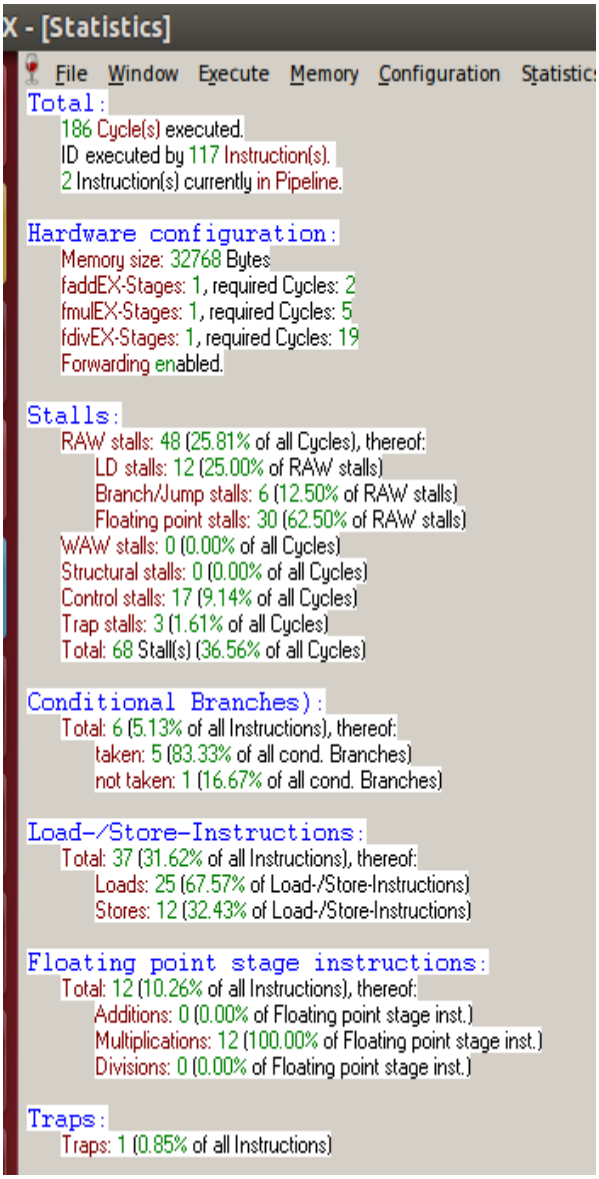
Without forwarding:



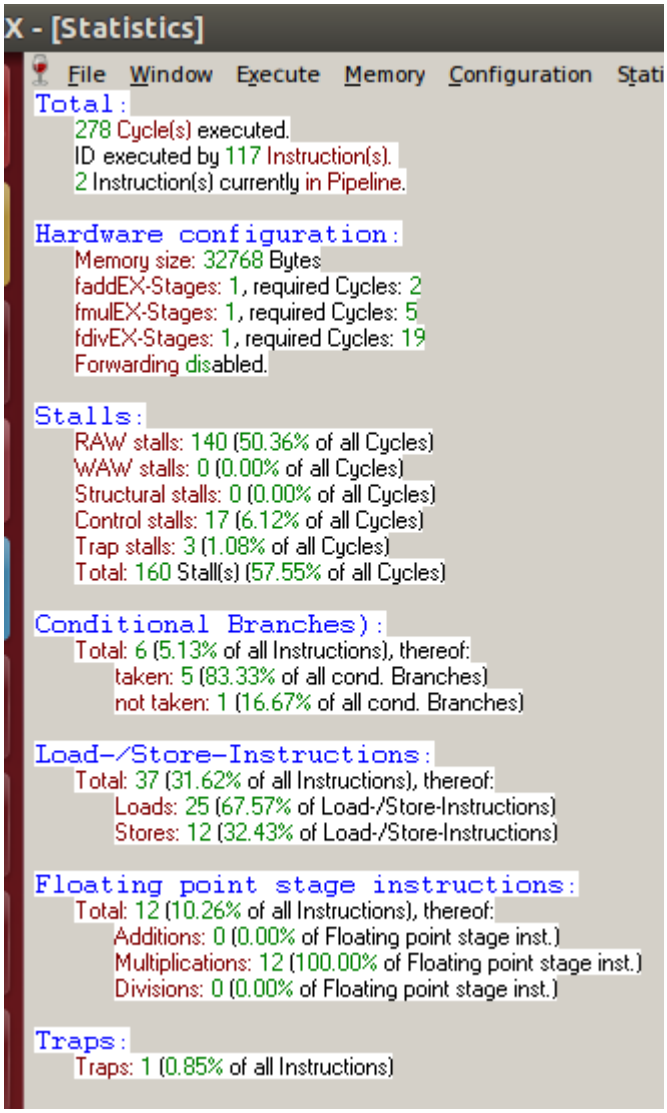
Question 1b

Number of clock cycle with forwarding : 186
Number of clock cycle without forwarding : 278
Speed up : 278 / 186 = 1.49

With forwarding :

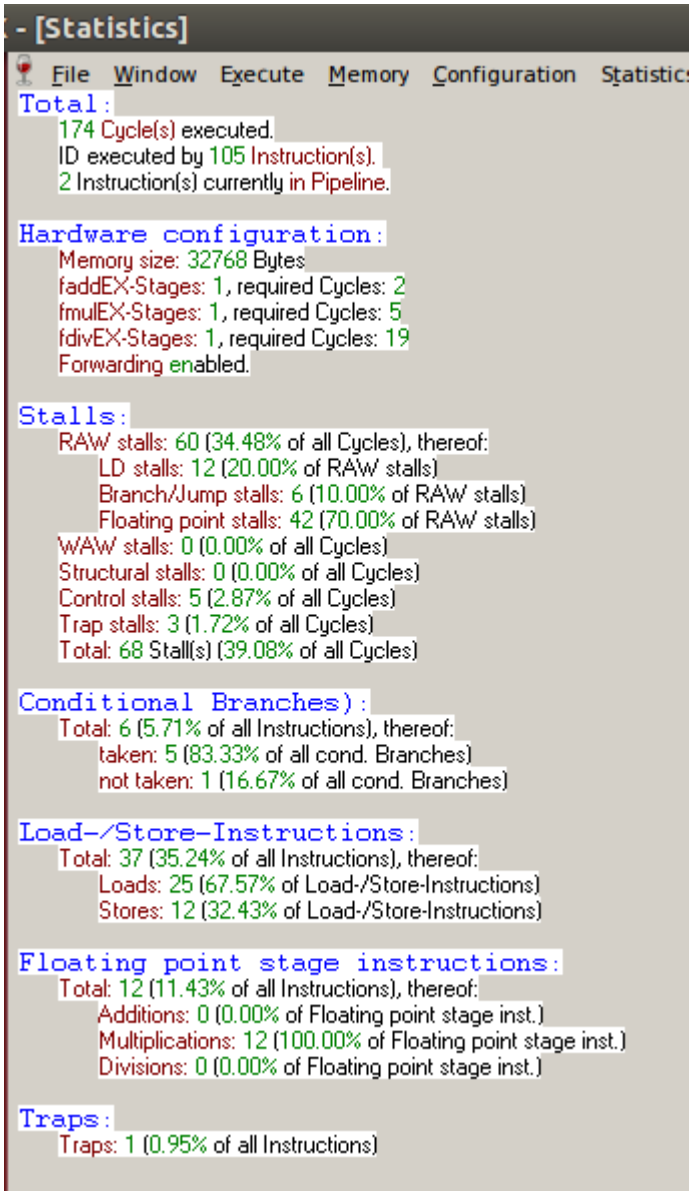


Without forwarding:



Question 2a

Number of clock cycle with forwarding : 174
Number of clock cycle with forwarding in 1(b) : 186
Speed up : 186 / 174 = 1.068



Question 3a

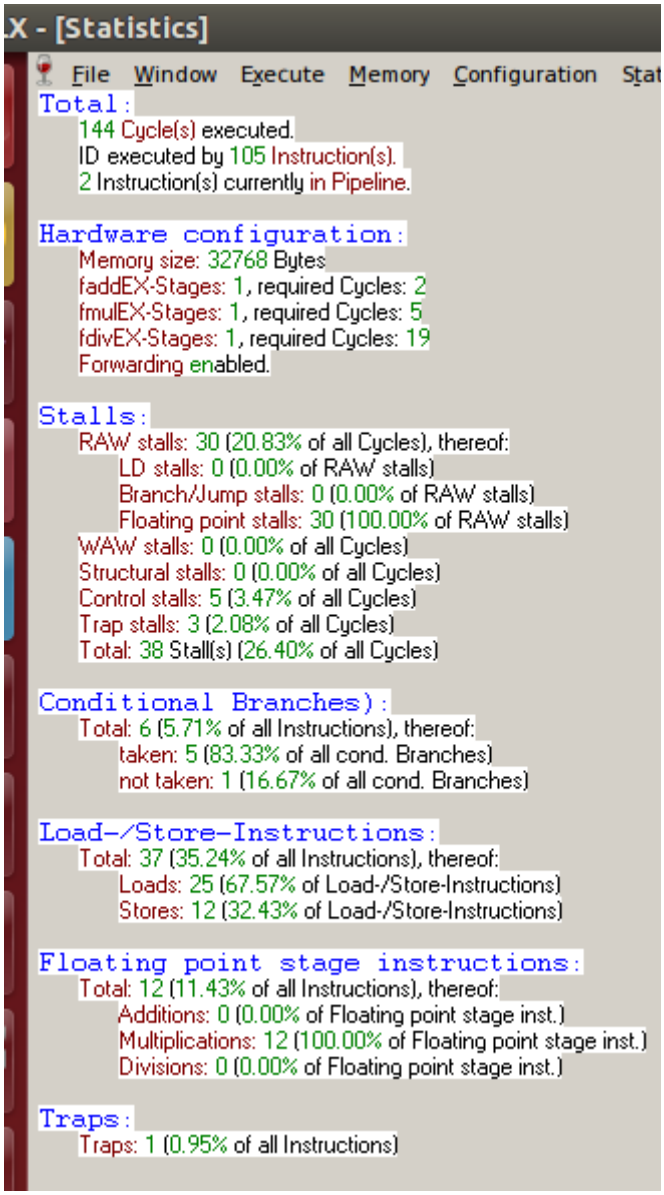
Justification in rearranging code :

In previous code, instructions using same registers were consecutive. Thus because of lw and multiplication; stalling was happening. In this changed code, insturctions using same registers are kept as away as possible to reduce this effect.

Number of clock cycle in scheduled code : 144
Number of clock cycle in unscheduled code : 174
Speed up : 174 / 144= 1.20

Question 3b

There are stalls because multiplication which can not be masked. It is because multiplication consumes large number of cycles and it is not possible to keep instructions further away without effecting each other.



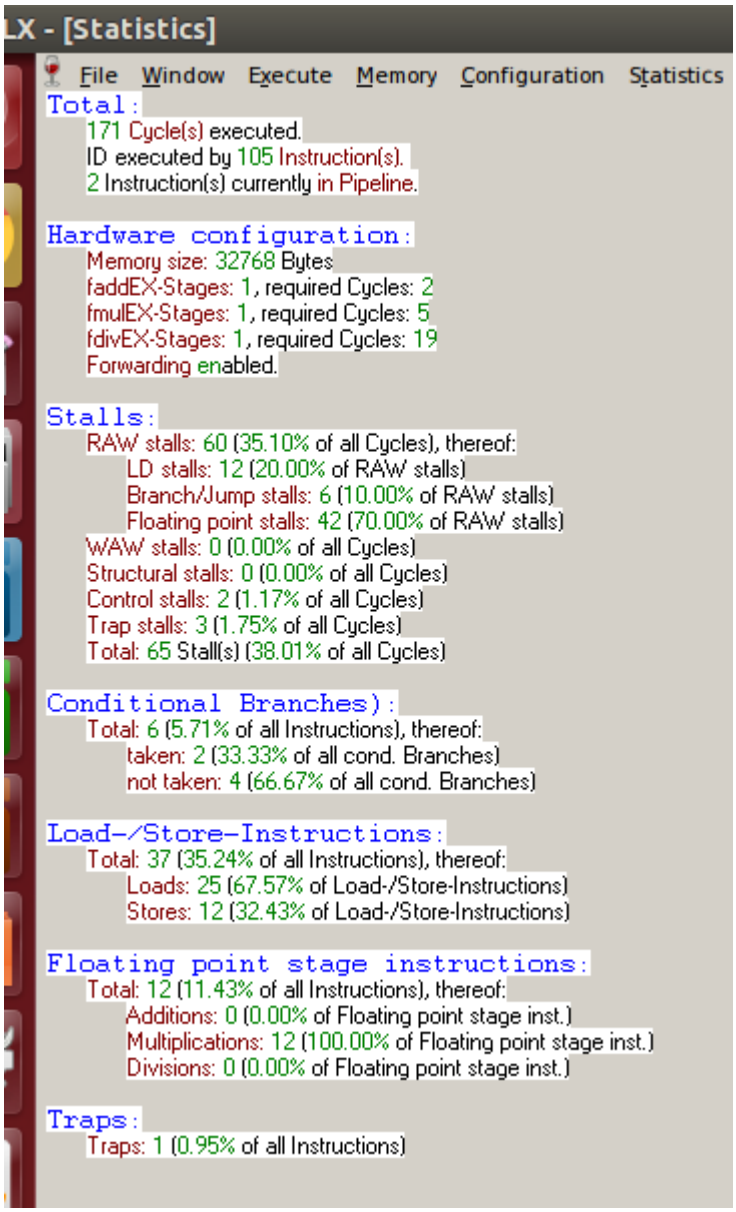
Question 4

Explanation :

Without unrolling, during execution of checking termination for loop, execution instruction after this is started and must be aborted for every iteration of loop other than last one. Thus there is waste of this clock cycle which is utilized in unrolling.

Note : if we assume N is even, we do not need to check termination condition in middle of loop, thus number of clock cycles reduces from 171 to 162; and speed up increses from 1.017 to 1.074 (what to change is mentioned in code)

Number of clock cycle : 171 (N/2 = 3 clock cycles decrease, one for each iteration saved)
Number of clock cycle without unrolling code : 174 (in question 2)
Speed up : 174 / 171= 1.017



Question 5a

Assuming N is even.

Number of clock cycle : 90
Number of clock cycle without unrolling code : 174 (in question 2)
Speed up : 174 / 90= 1.93

In previous part since we were using same registers in unrolling, we could not keep instructions using same register further away then in part 3. However here we can use other registers and thus we can keep such instructions as away as possible. Also two mult must be as far as possible. Naming convention used is each Ri is renamed as R1i eg. R4 to R14 except counters.

Question 5b

Even after renaming, multiplication must not be overlapping. For that stall occurs which can not be masked away.