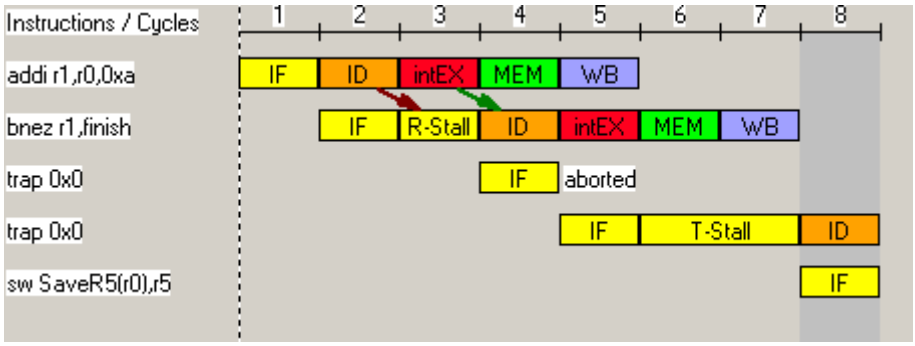
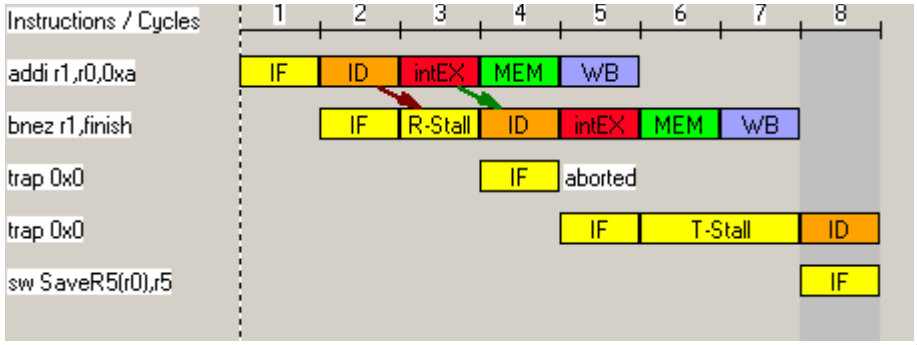
2) Code:2.s



Comments:
The program stores a value of 10 in r1 and then uses r1 to decide wether or not to do a jump. Thus a data stall occurs (R-Stall) at bnez.

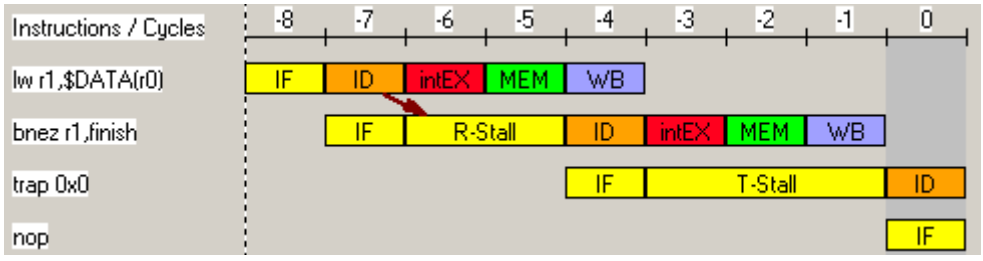
5) Code:5.s



Comments:
The program stores a value of 10 in r1 and then uses r1 to decide wether or not to do a jump. Thus a data stall occurs (R-Stall) at bnez. The value in r1 is required at ID stage of bnez and is obtained only after EX stage of addi. Thus data forwarding occurs from intEX stage of addi to ID stage of bnez

6)
Comments:
Not possible to do so. Write occurs in rising edge and read happens during falling edge. Both of these operations can occur within the same clock cycle. So because both operations WB and ID can occur in the same clock cycle data and hence forwarding doesn't occur from MEM (writing) to ID(reading)

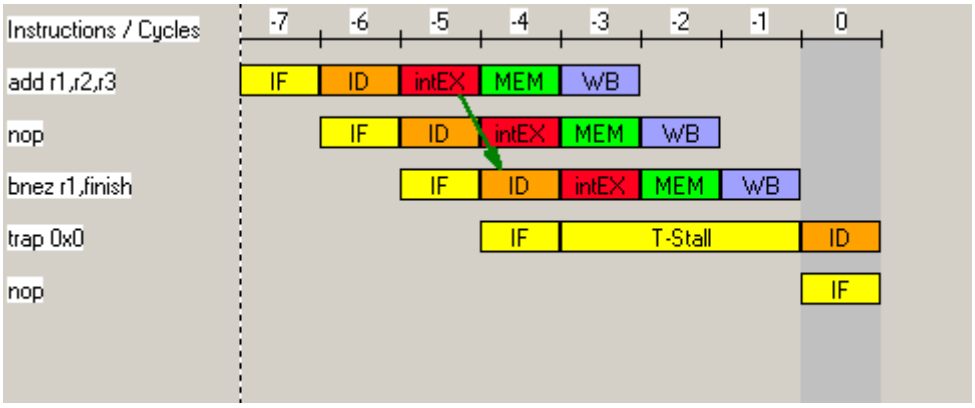
7) Code: 7.s



Comment:
Using mult and trap instructions it is possible to increase the stalls beyond 2 cycles but without them, 2 cycles is the maximum stall length caused by the previous instructions
Note: I am not considering cascading effect caused by previous instruction being stalled by the one

before it.

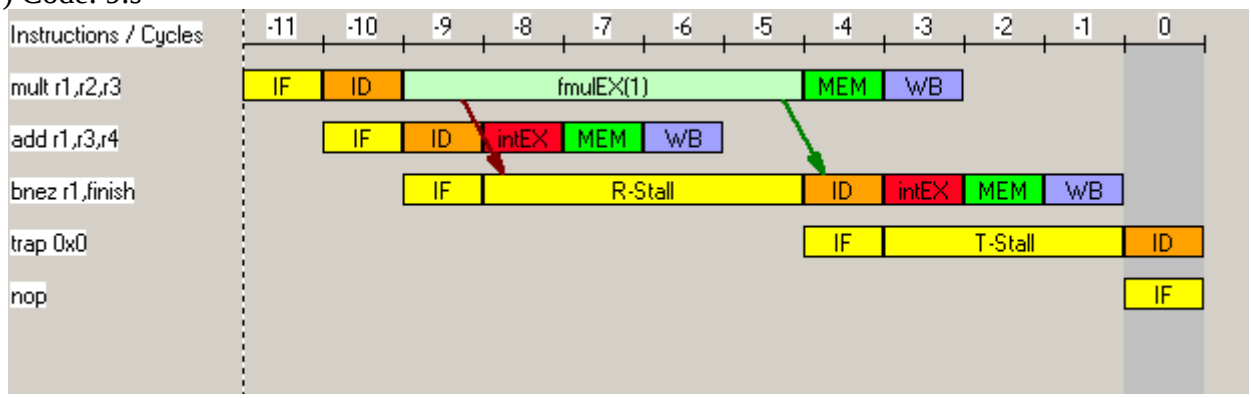
8) Code: 8.s



Comment:

bnez requires value of r1 during ID from addi which caclulates input in third cycle (EX). So inserting some more instructions between bnez and addi causes it to skip over a instruction

9) Code: 9.s



Comment:

Mult instruction takes more cycles than the subsequent add instruction. The value of r1 is forwarded from mult to bnez but it actually is supposed to check from the add instruction. This is supposed to cause WAW stall.