Another example of the hazard where register file is successively written and read:

addi $t1,$zero,50

addi $t2,$zero,142

add $t9,$zero,$zero // in place of nop

add $t9,$zero,$zero // in place of nop

sw $t1,150($t1)

add $t9,$zero,$zero // in place of nop

add $t9,$zero,$zero // in place of nop

add $t9,$zero,$zero // in place of nop

add $t1,$t1,$t1

lw $t2,100($t1)

$t1 and $t2 are initialized with certain values. The value of $t1 is stored at memory location 200 ($t2+150). Then $t1 is modified and subsequently used for load word. The last 2 instructions pose a data hazard. It is solved by taking the ALU result passed through the EX pipeline and forwarding it back to inputs of ALU. The input MUXs of ALU are appropriately controlled by a forwarding unit.



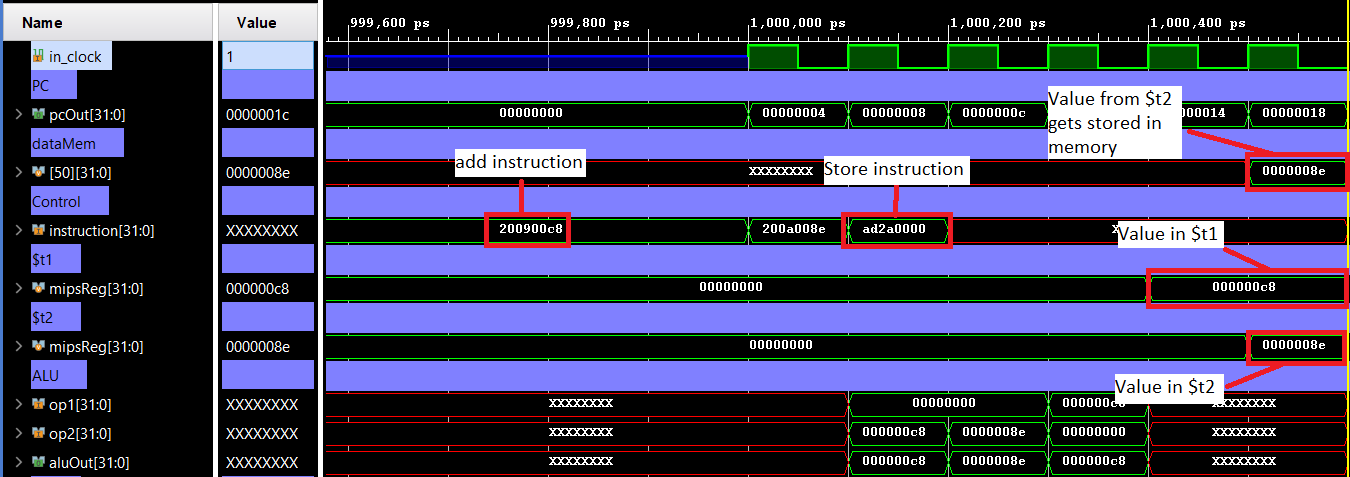
2. Hazards where an instruction attempts to read from a register, one instruction after it has been written into:

addi $t1,$zero,200

addi $t2,$zero,142

sw $t2,($t1)

The first and third instructions pose a hazard. (The second and third also pose a hazard but that has been resolved in the previous examples). To solve this hazard, we have forwarded the value of the ALU result from the last stage (writeback) to the input of the ALU. The input MUXs of ALU are appropriately controlled by forwarding unit. (Image below)



Consider another example:

addi $t2,$zero,150

addi $t1,$zero,50

addi $t3,$zero,173

sw $t3,150($t1)

add $t2,$t2,$t1

add $t9,$zero,$zero

lw $t1,($t2)

This program contains multiple hazards. Focusing on the fifth and seventh instructions, the hazard is solved by forwarding the value of ALU result from the writeback stage to the input of the ALU. The input MUXs of ALU are appropriately controlled by forwarding unit.



3. Hazards where an instruction attempts to read from a register, two instructions after it is written into:

addi $t1,$zero,100

addi $t2,$zero,200

add $t9,$zero,$zero

sw $t3,100($t1)

The hazard posed by first and fourth instruction is solved by forwarding the ALU result from the writeback stage into the MUXs connected at the outputs of the Register File. These MUXs give their output to the next pipeline. The data is thus directly taken from the writeback stage, since an additional clock cycle will be required by the Register File to write new values into its registers.