Computer Architecture Lab (CS 341)

Assignment 7: MIPS Pipeline Simulator

Lab Assignment 4

Bhaskar Gupta - 180050022
Niraj Mahajan - 180050069
Shivam Goel - 180050098
Tathagat Verma - 180050111

Question 1

Instruction sequence:

```
dadd   r8, r9, r10
lw     r10, 400(r8)
dsub   r9, r10, r8
dadd   r8, r9, r10
halt
```

Total execution time:
With forwarding : 10 cycles
Without forwarding : 15 cycles

Question 2

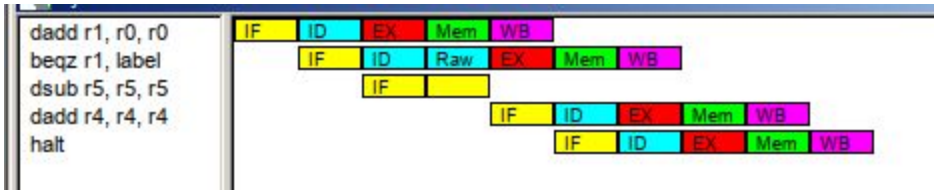
Instruction sequence:

```
main:
        dadd   r1, r0, r0
        beqz   r1, label
        dsub   r5, r5, r5
        dadd   r2, r0, r0
label:
        dadd   r4, r4, r4
        halt
```

Conditions: Enabled forwarding

(a) Branch target address is computed in **ID stage**
(b) Branch condition is evaluated in **ID stage**
(c) **1 stall** due to taken branch without a delay slot
(d) **0 stalls** due to taken branch with a delay slot

Pipeline cycle diagram(without delay slot)



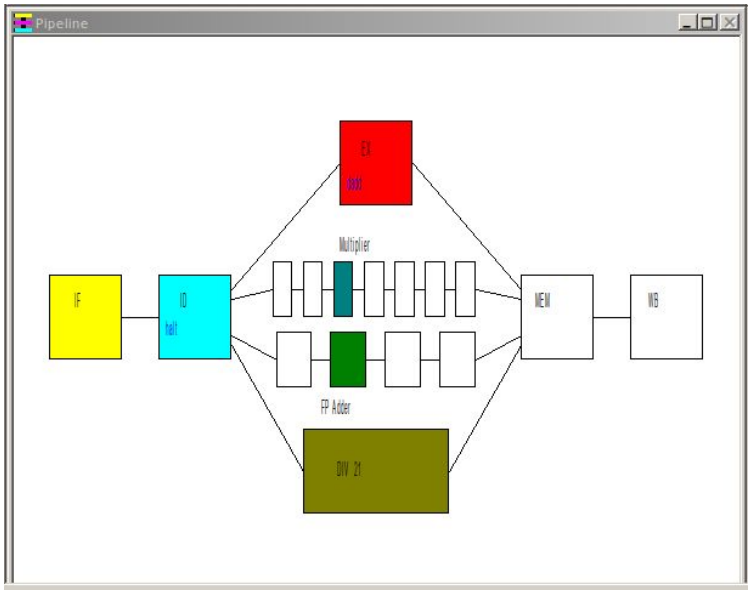
Referring the above diagram, we conclude that the branch condition evaluation happens in the ID stage as the program counter was updated in ID stage (as the IF of the next instruction occurred until ID of the branch was completed) and the value of r1 was used in the ID stage itself, not before that, due to which the Raw stall is in the ID stage and hence the branch condition was evaluated in the ID stage.
The branch target address is also computed in the ID stage, as the “label” is read in OF which is part of ID. Now since the entire process (i.e until updating PC) is completed in ID, the branch target address computation was also done in ID (as it started and ended within ID).

Question 3

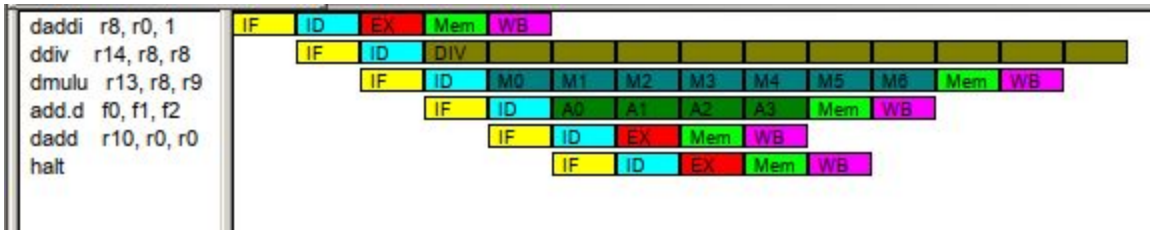
Instruction sequence:
1. daddi r8, r0, 1
2. ddiv r14, r8, r8
3. dmulu r13, r8, r9
4. add.d f0, f1, f2
5. dadd r10, r0, r0
6. halt

Conditions: Enabled forwarding
At cycle 7, all functional units are simultaneously busy

Pipeline diagram



Pipeline cycle diagram, at cycle number 7, all the functional units are simultaneously busy



Question 4

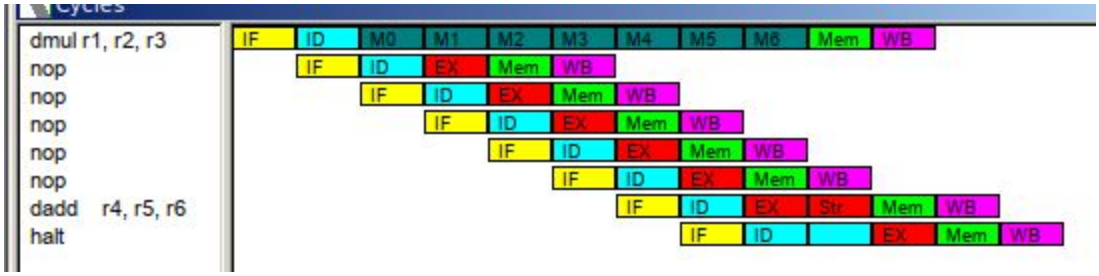
Yes, two or more instructions can finish execution during the same cycle
The instruction which had started earlier gets promoted to the MEM stage

Conditions: Enabled forwarding

Instruction sequence showing the above:

1. dmulu r1,r2,r3
2. nop
3. nop
4. nop
5. nop
6. nop
7. dadd r4,r5,r6

Instruction 1 & 7 complete execution in the same cycle. Instr 1 gets promoted to MEM stage and Instr 7 gets stalled
Pipeline cycle diagram:



Question 5

Yes, structural hazards can occur on the processor being simulated. So, in the following sequence of instructions FP/integer divider cause the structural hazard.

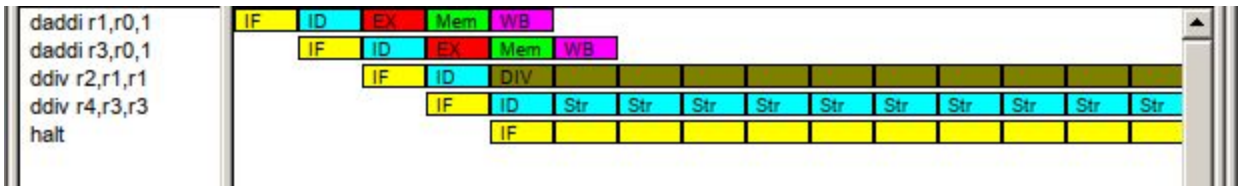
Conditions : Enabled forwarding

Instruction sequence showing the above:

1. daddi r1,r0,1
2. daddi r3,r0,1
3. ddiv r2,r1,r1
4. ddiv r4,r3,r3

Instruction 3 uses the FP/integer divider component and in the next cycle instruction 4 also attempts to use the same component but has to stall because instruction 3 is still using that component. This leads to structural hazard.

Pipeline cycle diagram showing the structural hazard:



Question 6

Yes, WAW hazard can occur
Conditions : Enabled forwarding

Code sequence:

1. dmulu r8, r0, r0
2. dadd r8, r0, r0
3. halt

Pipeline timing diagram:

Question 7

Yes, WAR hazard can occur
Conditions : Enabled forwarding

Code sequence:

1. ddiv r8, r9, r10
2. dmulu r11, r8, r12
3. dadd r12, r0, r0
4. halt

Pipeline timing diagram: