Name - ARPAN MANDAL

Roll - 00191050106 1

YEAR :- 2nd Subject - Computer Architecture

Sem :- 2nd Exam Roll - CSE 2140 21

Group-A

1. pipelining is efficient because

c) it overlaps the execution of several instructions.

2. A pipelined processor doesn't have separate instruction and data memories. If a load instruction, i_1, is initiated (occupies IF stage) in cycle-1. Subsequent instructions i_2, i_3, i_4, i_5 are initiated in cycle 2, 3, 4, 5 respectively.

There is a hazard involving i_1 and i_4 in

a) cycle-4 because both access memory

3. Instruction DSUB

b) may not get the correct value of R_1

4. Instruction AND

a) gets correct value of R_1

5. Instruction OR

a) gets the correct value of R_1

6) The data hazard involving LD and DSUB
a) can be resolved by forwarding

7) b) cycle - 5
~~a) cycle - 4~~

8) a) 1

9) b) 2

10) a) 1

11) b) S.D

12) d) none of the above

13) B) multicomputers

14) c) both communication and synchronization

15) a) multiprocessor

16) b) any processor can access the local memory of ~~another~~ any other processor

17) a) distributed cache directories

18) b) remote nodes that have a copy of each memory block

19) b) switching network

20) a) Contain bus request lines

21) b) deactivates its request line when the arbiter activates its grant line

22) c)

23) c)

24) a)

25) c)

26) c)

27) d)

28) c)

29) c)

30) b)

31) c)

32) d)

33) c)

34) b)

35) a)

36. pipeline process has five stages: fetch - IF,
decode - ID
register read - RR
execute - EX
write Back - WB

~~for normal~~
ADD R_1, R_4, R_7 ; $R_1 = R_4 + R_7$

BEQ $R_2, \#0, R_1$; jump to the address specified in R_2 if $R_1 = 0$

SUB R_8, R_{10}, R_{11}

MUL R_{12}, R_{13}, R_{14}

for normal execution of instructions →

	1	2	3	4	5	6	7	8	9	10
ADD	IF	ID	RR	EX	WB					
BEQ		IF	ID	RR	EX	WB				
SUB			IF	ID	RR	EX	WB			
MUL				IF	RR ID	RR	EX	WB		

(i) So If we perform this instructions normally then data hazard will occure, Here BEQ need the modified value of R_1 to execute the branch instruction, So for correct execution we need a Stall before RR cycle of BEQ instruction. So modified ~~data~~ diagram →

	1	2	3	4	5	6	7	8	9	10
ADD	IF	ID	RR	EX	WB					
BEQ		IF	ID	Stall	RR	EX	WB			
SUB			IF	Stall	ID	RR	EX	WB		
MUL				Stall	IF	ID	RR	EX	WB	

(ii) Here after finishing of first instruction we get the value of R_1, so after that BEQ can execute

Now If branch instruction is not taken and considering the 'Stalling Solution', for control Hazard - we can say there need 12 cycles to execute the ~~words~~ 4 instructions. But for other kind of Solution ~~cycles~~ No of cycles are not 12.

~~ADD~~	1	2	3	4	5	6	7	8	9	10	11	12
ADD	IF	ID	RR	EX	RW							
BEQ		IF	ID	Stall	RR	EX	RW					
SUB			Stall	Stall	Stall	Stall	IF	ID	RR	EX	RW	
MUL				Stall	Stall	Stall		IF	ID	RR	EX	RW

(*) This is only true for the "Stalling Solution" of Control hazard