2021–2022 SPRING SEMESTER

CS224 – LAB 5 DESIGN REPORT

IMPLEMENTING THE MIPS PROCESSOR WITH PIPELINED MICROARCHITECTURE

NAME: ABDULLAH RIAZ
ID : 22001296
COURSE:CS224- COMPUTER ORGANIZATION
SECTION: 06
DATE: 13/04/2022

b)

Table 1

Hazard Name	**Hazard Type**	**Pipeline Stages Affected**
Compute-Use	Data Hazard	Decode Stage
Load-Use	Data Hazard	Execution Stage
Load-Store	Data Hazard	Memory Stage
Branch	Control Hazard	3 unnecessary instructions are fetched in case of wrong branch prediction

c)

Compute-Use

- Forwarding is the solution for this data hazard. It occurs when data has not been written to the register in the writeback stage of the initial instruction and a subsequent instruction is reading the data of that register. The data read would be incorrect and a way to solve this is to forward the data to the next instruction before the writeback stage of the initial instruction.

Load-Use

- This hazard can be solved using stalling. It occurs when instructions reading from memory have not completed the memory stage and have not loaded to the destination register and the subsequent instruction try to read that register. Since the memory stage is at the end and the subsequent instruction requires it in the execution stage, it cannot be forwarded. In this case the subsequent instruction can be stalled.

Load-Store

- This hazard can also be solved using stalling.It occurs when instructions loading to register have not completed and subsequent storing instructions try to use that register. By stalling the subsequent instruction till the loading instruction is complete, this hazard can be solved.

Branch

- This control hazard can be solved by stalling the next 3 instructions.It occurs when a branch instruction needs to make a branch decision and it is not made by time the next instruction is fetched.

d) Equations:

StallF = **StallD** = **FlushE** = (**lwstall** OR **branchstall**)

lwstall = ((rsD== rtE) OR (rtD== rtE)) AND MemtoRegE

branchstall = BranchD AND RegWriteE AND (WriteRegE == rsD OR WriteRegE == rtD)
OR BranchD AND MemtoRegM AND (WriteRegM == rsD OR WriteRegM == rtD)

ForwardAD = (rsD != 0) AND (rsD == WriteRegM) AND RegWriteM

ForwardBD = (rtD != 0) AND (rtD == WriteRegM) AND RegWriteM

e) sracc can cause compute-use hazards mentioned in Table 1 and it can be solved by forwarding. Using this small mips assembly code , we can test if our pipelined processor can handle the compute-use hazard as this code would cause such a hazard.

```
addi $s1, $s1, 5
add $s0, $s1, $zero
subi $t1, $s0, 1
slt $t1, $t1, $s1
```

Code for test program with no hazards:

```
addi $a0, $zero, 5
addi $a1, $zero, 4
addi $a2, $zero, 1
addi $t0,  $t0, 10
and $t1, $a1, $a0
or $t3, $a0, $a1
```