BILKENT UNIVERSITY

COMPUTER SCIENCE

CS224-005 COMPUTER ORGANIZATION

DESIGN REPORT

LAB#5

SECTION#005

MUHAMMED IKBAL DOGAN

ID#21702990

9 APRIL 2021

Question B)

NAME: Compute-Use

TYPE: Data Hazard

Stages that are Affected: Decode stage is affected because data which is fetched from register file is not updated since the previous instruction could not finish the write back stage. Therefore, execute and write back stage would perform with the wrong value.

NAME: Load-Use

TYPE: Data Hazard

Stages that are Affected: Because load needs to write back the significant register, there is 2 cycle latency. Therefore, execute and memory stages would be affected.

NAME: Load-Store

TYPE: Data Hazard

Stages that are Affected: Memory stage would be affected because the value which is going to stored in the memory would not be the updated value.

NAME: Branch

TYPE: Control Hazard

Stages that are Affected: Branch decision would be made in memory stage, therefore when the branch prediction is wrong, following three instructions would be fetched.So, fetch, decode and execute stages would be flushed.

Question C)

NAME: Compute-Use Hazard

What is Compute-Use Hazard?

It occurs If there are subsequent instructions and the following instruction reads from the same register which previous instruction write on the same register.

When Compute-Use Hazard occurs?

For example if add instruction uses $t0 as destination register rd and the following register uses $t0 as a source register, compute-use hazard occurs because it does not use $t0's last value.

How Compute-Use Hazard occurs?

When the computing instruction at execute stage, the following instruction is at read stage. So following instruction reads the previous value because to read new value, compute instruction should write back it to register file. However, compute instruction's execute stage and the following instructions read stage happens at the same cycle. In addition, second instruction after the compute instruction also get affected because when the compute instruction at memory stage , second instruction is at decode stage therefore, it also reads the previous value.

Solution of Compute-Use Hazard

Following two instructions can be stalled for two times to wait until computng instruction write back the data. Or more sensible solution is the forwarding the computed data to following instruction's execute stage.

NAME: Load-Use Hazard

What is Load-Use Hazard?

It is the hazard that the following two instruction comes after load instruction cannot read the right value because lw has to write back the new data to register file.

When Load-Use occurs?

When the following instruction tries to reach the data which is going to be loaded by previous instruction load-use hazard occurs.

How Load-Use occurs?

Instruction that loads the data cannot write the loaded data to register until it finishes the write back stage.Therefore following instructions having trouble when they try to execute that register because it would not have been uploaded that time.

Solution of Load-Use Hazard

Forwarding cannot solve the problem therefore stalling the following instructions untill the data is available is one solution or both stalling and forwarding could be used for reducing the stalls.

NAME: Load-Store Hazard

What is Load-Store Hazard?

It is the hazard that the subsequent instructions load and store on the same register.

When Load-Store Hazard occurs?

If the lw and sw comes consecutively and they use the same rt register, Load –store hazard occurs

How Load-Store Hazard occurs?

Storing instruction would store the wrong value at Memory stage because that register has not been uploaded yet. Therefore storing instruction stores the previous data.

Solution of Load-Store Hazard

Stalling the pipeline until the load instruction write backs the memory therefore Decode stage of the following instruction could read the right value.In addition stalling and forwarding could be combinated therefore at writeback stage of the load instruction, data could be forwarded to excecute stage of the following storing instruction.

NAME: Branch Hazard

What is Branch Hazard?

Branch hazard is the control hazard which causes delay in because of the late decision of where to branch.

When Branch Hazard occurs?

When branch instruction is loaded, branch hazard occurs.

How Branch Hazard occurs?

Branch decision is given at the Memory stage of the pipeline therefore following three instrucions are not certain until the end of the Memory stage of the branch instruction.

Solution of Branch Hazard

Stalling the pipeline for three cycles could be solution. Or, adding an additional equality comparator could be another solution therefore misprediction penalty could be reduced and if there is a misprediction, stages before the decision must be flashed

Question D)

Logic Equations of Hazard Unit for Forwarding

if ((rsE != 0) AND (rsE == WriteRegM) AND RegwriteM) then

ForwardAE = 10

else if ((rsE != 0) AND (rsE == WriteRegW) AND RegwriteW) then

ForwardAE = 01

else ForwardAE = 00

if ((rtE != 0) AND (rtE == WriteRegM) AND RegwriteM) then

ForwardAE = 10

else if ((rtE != 0) AND (rtE == WriteRegW) AND RegwriteW) then

ForwardAE = 01 else ForwardAE = 00

Logic Equations of Hazard Unit for Stalling and Flushing

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

StallF = StallD = FlushE = lwstall

Question D)

No Hazard

addi $t0, $0, 1 8'h04: instr = 32'h20080001;

addi $t1, $0, 2 8'h08: instr = 32'h20090002;

addi $t2, $0, 3 8'h0c: instr = 32'h200a0003;

addi $t3, $0, 4 8'h10: instr = 32'h200b0004;

sw $t3, 4($0) 8'h14: instr = 32'hac0b0004;

add $s0, $t1,$t2 8'h18: instr = 32'h012a8020;

and $s1, $t2,$t1 8'h1c: instr = 32'h01498824;

or $s2, $t3,$t0 8'h20: instr = 32'h01689025;

lw $t2, 4($0) 8'h24: instr = 32'h8c0a0004;

Compute-Use Hazard

addi $t0,$zero, 10 8'h00: instr = 32'h2008000a;

addi $t1, $t0, 20 8'h04: instr = 32'h21090014;

add $t3,$t1,$t0 8'h08: instr = 32'h01285820;

Load-Use and Load-Store Hazard

addi $t0,$zero, 10 8'h00: instr = 32'h2008000a;

addi $t1, $t0, 20 8'h04: instr = 32'h21090014;

sw $t1,0($zero) 8'h08: instr = 32'hac090000;

lw $s1, 0($zero) 8'h0c: instr = 32'h8c110000;

add $s2, $s1,$zero 8'h10: instr = 32'h02209020;

Branch Hazard

addi $t0,zero, 10 8'h00: instr = 32'h2008000a;

addi $t1, $zero, 10 8'h04: instr = 32'h2009000a;

beq $t0,$t1,6 8'h08: instr = 32'h11090003;

```
addi $t2,$zero,0        8'h0c: instr = 32'h200a0000;
sub $t3, $t1,$t0        8'h10: instr = 32'h01285822;
add $s2, $t1,$zero      8'h14: instr = 32'h01209020;
sw $t0, 0($zero)        8'h18: instr = 32'hac080000;
```