BILKENT UNIVERSITY

COMPUTER SCIENCE

CS224 – COMPUTER ORGANIZATION

DESIGN REPORT

LAB 5

SECTION 6

DENİZ SEMİH ÖZAL

21802414

09/04/2021

PART 1

Question B

- Data Hazards:
    1. Compute-use

       Since register has not been updated yet in the preceding instruction, following instruction has fetched wrong data. The execute and writeback stages can be affected.

    2. Load-use
       A load-use hazard require time delaying in execution stage since the result of preceding instruction made not be available for following instruction. In another saying, reading happens before writing, so execution and memory stages can be affected.

    3. Load-store
       A load-store hazard can happen when preceding instruction is still not completed, so wrong data can be written into the memory. Execution and memory stages can be affected.

- Control Hazards:
    1. Branch
       Branch not determined after the Memory stage so instructions after the branch are fetched before the branch occurs. If branch happens, these instructions must be flushed. When branch prediction is made, following instructions until the Memory Stage have been computed unnecessarily.

Question C

- Data Hazards:
  1. Compute-use
  Data-forward method can be use in Execute and Writeback stages. Another method can be stalling the pipeline which is a hardware solution for it. This problem occurs when preceding instruction does not reach Writeback stage and following instruction reach wrong data since true data is not written yet in the WriteBack stage. The solution for that is data forwarding because once the data is not written, the result of ALU result gives the same result so ALU

result can be used for data forwarding. Similarly data forwarding is applicable for Writeback stage

2. Load-use
This hazard happens when preceding instruction cannot load the data from the memory to the destination register until the next instruction comes. Therefore following instruction try to access the data that is not loading from the memory yet in the previous instruction. Again stalling the pipeline can be a solution for this as a harware solution.

3. Load-store
This hazard occurs when load and store instructions come one after another so that store instruction may not work. The solution for this again stalling the pipeline for may be the decode stage.

- Control Hazards:
  1. Branch
  The branch decision is made in the memory stage and this decision process may cause a delay. Therefore this hazar occurs when a branch instruction is loaded and at the same time the branch instruction is taken. The solution for this might be adding a branch prediction component for early stages or early branch resolution. Thus unnecessiraly instructions will be flushed. Another solution can be stalling the pipeline until the memory stage and postpone the following instructions.

Question D

Forwarding Logic for ForwardAE:

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

Stalling Logic:

lwstall = ((*rsD*==*rtE*) OR (*rtD*==*rtE*)) AND *MemtoRegE*
StallF = StallD = FlushE = lwstall

Question E

No hazard

addi $t0, $0, 0x0028

subi $t1, $0, 0x0045

add $t2, $0, $0

addi $t3, $0, 0x0048

addi $t4, $0, 0x0010

addi $t5, $0, 0x0024

subi $t6, $t0, 0x0004

addi $t7, $0, 0x0030

addi $s0, $0, 0x0046

sw $t1, 0x0000($t2)

addi $s1, $0, 0x0020

addi $s2, $0, 0x0050

slt $t3, $t4, $t5

and $t0, $t6, $t7

subi $s3, $0, 0x0122

addi $s4, $0, 0x0078

lw $s0, 0x0000($t2)

Compute Use Hazard

addi $t0, $0, 0x0012

subi $t1, $t0, 0x0048

addi $t2, $t1, 0x0003

Load-Use & Load-Store Hazard

addi $t0, $0, 0x0042

addi $t1, $0, 0x0032

addi $t2, $0, 0x0056

addi $t3, $0, 0x0022

addi $t4, $0, 0x0030

addi $t5, $0, 0x0060

sw $t0, 0x0000($t3)

lw $t1, 0x0000($t0)

Branch Hazard

addi $t0, $0, 0x0040

addi $t1, $0, 0x0060

beq $t0, $t1, 0x000C

addi $t2, $0, 0x000D

addi $t3, $0, 0x000f

add $t4, $t2, $t3