Bilkent University - Computer Science - CS224

Design Report LAB5

Section 3

Sertaç Derya 22003208

9.05.2023

Part 1.

b)

- RAW hazard, it can result from lw or any other instruction that can't update a register before using that register again. Types:
  Compute use (add, addi, sub, or, etc.)
  Affected stage: E, M, W

  Load use (lw)
  Affected stage: F, D, E
- Control hazards can happen from branch instructions. The affected stages are Fetch, Decode, Execute and Memory.

c)

- For compute use hazards, best solution is forwarding. We can look at signals to see if the register is being updated when the data is in execute stage then forward the data from the ResultW or ALUOutM. Then we can decide if the ALU is going to get which signal by comparing if the new value is in the W or M stage.
- For load use hazards, we need to stall the processor to get the new value from the memory, so forwarding by itself won't be enough. Then we will forward the data according to the instruction. Stalling will only be done for instructions that come right after lw, for others forwarding will be enough.
- Branch control hazard can be solved by clearing the Fetch-Decode register. If the branch instruction uses a register that was used by another instruction and wasn't updated yet; the PC and Fetch-Decode registers are stalled and the Decode-Execute register is flushed. After the register value is computed, it is written to the register file and the instructions continue with a normal branch hazard.

d)

- if (RsE != 0 and (RegWriteM and (WriteRegM == RsE))) => ForwardAE = 10
  else if (RsE != 0 and (RegWriteW and (WriteRegW == RsE))) => ForwardAE = 01
  to solve the RAW data hazard without lw.

  To solve RAW data hazard with lw:
  Stall = MemtoRegE and ((RsD == RsE) or (RtD == RtE))
  Stall = FlushE = StallD = StallF

  To solve branch hazard:
  BranchStall = BranchD and RegWriteE and (RsD == WriteRegE or RtD == WriteRegE)
  or BranchD and MemtoRegM and (RsD == WriteRegM or RtD == WriteRegM)

  BranchStall = StallF = StallD = FlushE
  To solve branch data hazard:
  ForwardAD = RsD != 0 and RsD == WriteRegM and RegWriteM
  ForwardBD = RtD != 0 and RtD == WriteRegM and RegWriteM

e) My instruction will be **rol.** It won't create any new hazards because it is a normal R-type instruction but it will still be affected by the old hazards. The solution to these hazards will be the same as compute-use hazards.

```
addi $t0, $zero, 0xA0A0
rol $t1, $t0, 2 # $t1 should be 0xFFFE8283
addi $t2, $t1, 2

addi $t0, $zero, 5
sw $t0, 0($t0)
lw $t1, 0($t0)
rol $t2, $t1, 2
```