Bilkent University
Computer Science
CS224

Design Report
Lab5
Section: 2

Arman Engin Sucu
ID: 21801777

07.04.2021

b) HAZARDS

Data Hazards

1. **Compute-Use Hazard**
    - **The pipeline stages that are affected:** The decode stage will be affected, since the destination register(rd) of the instruction is not written yet. Wrong value or values(belongs to old instruction) will be used in execute and write back stage.

2. **Load-Use Hazard**
    - **The pipeline stages that are affected:** Execute and memory write stages are affected. Since the register is written, the datas fetched by subsequent instructions will be wrong. The instructions that will read from the memory have to wait until the end of the memory write stage. Therefore, execute and memory stages affected 2 clock cycles.

3. **Load-Store Hazard**
    - **The pipeline stages that are affected:** Memory stage of sw instructions are affected. Since the data is not written the register yet, the store instruction can not retrieve the right data.

Control Hazards

1. **Branch Hazard**
    - **The pipeline stages that are affected:** A delay happens. Since without determining branch instruction will be executed or not next instruction will be fetched, unnecesary 3 instructions will be fetched.

c) SOLUTIONS FOR HAZARDS

1. Compute-Use Hazard

- **When It Occurs:** When an R-type instruction(for example sub) executed and it is destination register has not been written yet. Subsequent instruction's rt's and rs' values will be wrong.
- **How to Fix It:** Stalling can be used. However, the effective solution is; instead of waiting fort he write back stage, data can be forwarded to the execute stage of the instruction.

2. Load-Use Hazard

- **When It Occurs:** When after the lw instruction, subsewuent instructions try to fetch the data from register before the data is loaded by lw instruction.
- **How to Fix It:** Until the end of the memory stage the data that will be loaded can not be reached, therefore, to forward the data to execute stage from the memory stage of the lw, stalling has to be used, Then forwarding should be used.

3. Load-Store Hazard

- **When It Occurs:** When there is a sw instruction right after the lw instruction.
- **How to Fix It:** The value for rt register has to be fetched by sw from lw. To enable this after lw completed memory stage, it should forward the value to the sw instructions memory stage.

4. Branch Hazard

- **When It Occurs:** When branch makes the misprediction, and though it takes the branch, PC+4 is fetched.
- **How to Fix It:** 3 stalls can be used to understand whether branch makes the jump or not, since in the end of memory stage it can be understood. Moreover, flushes should be used to clear unnecessary instructgions. Additional hardware can be added(“comparator” and “and gate” to the decoder stage).

d)

Logic Of The 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

- Same logic is applied for rtE as well

Logic Of The Hazard Unit For Stalling

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

StallF = StallD = StallE = lwstall

Logic For BranchStalling

branchstall = (BranchD AND RegWriteE AND

(WriteRegE = = rsD OR WriteRegE = = rtD))

OR

BranchD AND MemToRegM AND

(WriteRegM = = rsD OR WriteRegM = = rtD))

e)

Test Program With No Hazards

```
addi  $t0, $zero, 3
addi  $t1, $zero, 4
addi  $t2, $zero, 5
add  $t3, $t3, $t0
and  $t3, $t0, $t1
or  $t3, $t0, $t1
sub  $t3, $t2, $t1
slt  $t3, $t2, $t1
sw  $t0, 0($t1)
lw  $t3, 1($t0)
beq  $t2, $t1, 8
addi  $t0, $zero, 10
addi  $t1, $zero, 11
addi  $t2, $zero, 12
```

Test Program For Compute Use Hazard

```
addi $t0, $zero, 10
Addi $t1, $t0, 6
And $t2, $t0, $t1
```

Test Program For Load Use Hazard

```
addi $t0, $zero, 1
addi $t1, $zero, 8
lw  $t2, 5($t0)
addi $t0, $t2, 5
```

Test Program For Branch Hazard

```
addi $t0, $zero, 0
addi $t1, $zero, 3
addi $t2, $zero, 4
beq $t0,  $zero, 18
and $t3, $t0, $t1
addi $t2, $t2, -3
```