Bilkent University

CS 224

Preliminary Design Report

Lab 05

Section 01

Mert Gençtürk

22003506

28.11.2022

RTL Expression for *sracc:*

IM[PC]

RF[rd] = RF[rd] + (RF[rs] << RF[rt])

PC <- PC + 4

Data Hazards and Solutions:

Compute-use : Value is needed before writeback stage. Decode stage of the pipelined instruction will be affected. Operations are performed on data that is not updated. Forwarding data to decode or execution stage from previous instructions execution or writeback stage will solve the hazard. Stalling can also be used.

Load-use : Register value has not been got from memory until mem stage but next instructions need it before. Flushing the execute stage and stalling fetch & decode is needed.

Load-store : Register value has been got from memory but it will be immediately used in next memory store instruction. Solution is stalling the pipeline

Control Hazards and Solutions:

branch : Executions started to being executed before branch address is decided. Prevent unwanted instruction to be completed flushing is needed. Since branch is decided in decode stage, only one instruction is going to be flushed.

d)

Forwarding to Execution Stage:

ForwardAE:

```
if ((rsE != 0) && (rsE == WriteRegM) && RegWriteM)
ForwardAE = 10
else
 if ((rsE != 0) && (rsE == WriteRegW) && RegWriteW)
ForwardAE = 01
 else ForwardAE = 00
```

ForwardBE:

```
if ((rtE != 0) && (rtE == WriteRegM) && RegWriteM)
ForwardBE = 10
else
 if ((rtE != 0) && (rtE == WriteRegW) && RegWriteW)
ForwardBE = 01
 else ForwardBE = 00
```

Stalling Instruction:

```
lwstall = ((rsD==rtE) || (rtD==rtE)) && MemtoRegE
StallF = lwstall
StallD = lwstall
FlushE = lwstall
```

Forwarding to Decode Stage:

```
ForwardAD = (rsD !=0) && (rsD == WriteRegM) && RegWriteM
ForwardBD = (rtD !=0) && (rtD == WriteRegM) && RegWriteM
```

Stalling and Flushing for branch:

```
branchstall = (BranchD && RegWriteE && (WriteRegE == rsD || WriteRegE == rtD))
 || (BranchD && MemtoRegM && (WriteRegM == rsD || WriteRegM == rtD))
StallF = (lwstall || branchstall)
StallD = (lwstall || branchstall)
FlushE = (lwstall || branchstall)
```

e)

sraac can cause compute-use hazard. Similar to part b, it can be solved with data forwarding to decode or execution stage of next instruction.

- compute-use:

```
li $t0, 5
li $t1, 10
li $t2, 15
sraac $t2, $t1,$t0
```

- load-use hazard:

```
li $t0, 10
li $t1, 20
lw $t2, 0($t1)
sraac $t0 ,$t2,$t1
```

- load-store hazard

```
li $t0,10
li $t1, 5
li $t3, 30
sw $t0, 0($t0)
lw $t2, 0($t0)
sw $t2, 0($t1)
sraac $t0 ,$t2,$t1
```

- branch hazard :

```
li $t0,10
li $t1, 5
beq $zero, $zero, 2
add $t0, $t0, $t0
sub $t2, $t1, $t0
add $t3, $t2, $t0
sraac $t0 ,$t2,$t1
```

- no hazard :

```
addi $t0, $zero, 9
addi $t1, $zero, 8
addi $t2, $zero 7
addi $t3, $zero 6
```

```
add $t2, $t0, $t1
and $t3, $t1,$t0
sw $t1, 0($t1)
add $t1, $t2, $t0
lw $t3, 0($t0)
sraac $t0 ,$t2,$t1
```