BILKENT UNIVERSITY

COMPUTER ENGINEERING

CS224-Computer Organization

PRELIMINARY DESIGN REPORT

Lab 5 Sec: 1

Berdan Akyürek/21600904

4.12.2019

b) List of hazards that can occur in this pipeline.

Hazard Name	**Hazard Type**	**Pipeline stage**
Compute-use	Data hazard	Execution
Load-use	Data hazard	Execution
Load-store	Data hazard	Memory
Branch	Control hazard	Decode

c) Hazard solutions

Hazard Name	**Solution**	**Explanation**
Compute-use	Forwarding	We need to connect first execution's ALU result to second instruction's ALU input.
Load-use	Forwarding	We need to connect first execution's MEM result to second instruction's ALU input.
Load-store	Forwarding	We need to connect first execution's MEM result to second instruction's MEM input.
Branch	Stalling	We need to wait 3 cycles to PC update.

d) Equations

StallF:
((rsD == rtE) OR (rtD == rtE)) AND MemtoRegE

StallD:
((rsD == rtE) OR (rtD == rtE)) AND MemtoRegE

ForwardAD:

(rsD != 0) AND (rsD == WriteRegM) AND RegWriteM

ForwardBD:

(rtD != 0) AND (rtD == WriteRegM) AND RegWriteM

FlushE:

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

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

ForwardBE:

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
```

e) Test Programs

No Hazard

```
addi $s0, $zero, 0
addi $s1, $zero, 1
addi $s2, $zero, 2
addi $s3, $zero, 3
addi $s4, $zero, 4
add $s5, $s0, $s1
nop
nop
nop
nop
add $s5, $s5, $s2
// expected result is 3

8'h00: instr = 32'h20100000;
8'h04: instr = 32'h20110001;
8'h08: instr = 32'h20120002;
8'h0c: instr = 32'h20130003;
8'h10: instr = 32'h20140004;
8'h14: instr = 32'h0211a820;
8'h18: instr = 32'h58000000;
8'h1c: instr = 32'h58000000;
8'h20: instr = 32'h58000000;
8'h24: instr = 32'h58000000;
8'h28: instr = 32'h02b2a820;
```

Compute-use

```
addi $s0, $s0, 0
nop
nop
nop
addi $s0, $s0, 8
addi $s1, $s0, 4
```

```
8'h00: instr = 32'h22100000;
8'h04: instr = 32'h58000000;
8'h08: instr = 32'h58000000;
8'h0c: instr = 32'h58000000;
8'h10: instr = 32'h22100008;
8'h14: instr = 32'h22110004;
```

Load-use

```
addi $s0, $zero, 8
addi $s1, $zero, 0
nop
nop
nop
sw $s0, 0($s0)
nop
nop
nop
lw $s1, 0($s0) #load
addi $s1, $s1, 18 # use
```

```
8'h00: instr = 32'h20100008;
8'h04: instr = 32'h20110000;
8'h08: instr = 32'h58000000;
8'h0c: instr = 32'h58000000;
8'h10: instr = 32'h58000000;
8'h14: instr = 32'hae100000;
8'h18: instr = 32'h58000000;
8'h1c: instr = 32'h58000000;
8'h20: instr = 32'h58000000;
8'h24: instr = 32'h8e110000;
8'h28: instr = 32'h22310012;
```

Load-store

```
addi $s0, $zero, 8
addi $s1, $zero, 0
addi $s2, $zero, 0
nop
sw $s0, 0($s0)
nop
nop
nop
lw $s1, 0($s0) # load
sw $s1, 0($s2) # store
```

```
8'h00: instr = 32'h20100008;
8'h04: instr = 32'h20110000;
8'h08: instr = 32'h20120000;
8'h0c: instr = 32'h58000000;
```

```
8'h10: instr = 32'hae100000;
8'h14: instr = 32'h58000000;
8'h18: instr = 32'h58000000;
8'h1c: instr = 32'h58000000;
8'h20: instr = 32'h8e110000;
8'h24: instr = 32'hae510000;
```

Branch

```
addi $s0, $zero, 8
addi $s1, $zero, 8
nop
nop
nop
beq $s0, $s1, L1
addi $s0, $s0, 128

L1:
addi $s0, $s0, 12

8'h00: instr = 32'h20100008;
8'h04: instr = 32'h20110008;
8'h08: instr = 32'h58000000;
8'h0c: instr = 32'h58000000;
8'h10: instr = 32'h58000000;
8'h14: instr = 32'h12110002;
8'h18: instr = 32'h22100080;
8'h1c: instr = 32'h2210000c;
```