CS224
5
1
Yağız Yaşar
21902951

LAB 5 PRELIMINARY REPORT

Part 1

Question B

There are 3 types of Data Hazards:

Load-use affects the Memory and Execute stages.

Load-store affects the Memory Stage.

Compute-use affects the Execute, Writeback and Decode stage.

There is 1 type of Control Hazard:

Branch affects the Memory stage.

Question C

1) Load-use Hazard

This type of hazard happens when the following instruction tries to reach the data, which is used in the instruction before. It happens in the Memory stage because until the Memory stage finishes, instructions from memory are not able to reach the data values. The instruction after this one won't reach the data from the earlier instruction in the Execute stage.

The solution for this hazard is stalling because the data would be reachable after the stall.

2) Load-store Hazard

This type of hazard happens when instructions, which are *sw* and *lw*, are being tried to use in the same register, *rt*. It happens because the data to be stored is tried to be stored after it is just loaded from the memory.

The solution for this hazard is stalling because it would allow data to be loaded from the memory, give more time to store the data.

3) Compute-use Hazard

This type of hazard happens when the data, which is computed in Execute stage and to be written in the Writeback stage, would not be written, and if the following instruction tries to reach this data when it is not written yet. It happens because the following instruction can read another data instead of the computed data.

The solutions for this hazard are stalling and forwarding. With forwarding the data, it would allow the following instruction to use the proper data value for the Execute stage, and with similar logic the stalling can be used because the following instruction would be stalled until the previous instruction's data is ready to be used.

4) Branch Hazard

It happens when a new branch is needed. It happens because the decision will be made when the following instruction is fetched, also it will be made in the Memory stage.

The solution for this hazard is stalling. Stalling will continue for 3 cycles, with the stalling the instructions would be flushed to prevent incorrect branches.

Question D

Stalling and Flushing Logic

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

StallF = StallD = FlushE = lwstall

Forwarding Logic

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

Question E

No Hazards

```
addi $t0, $zero, 0x0010
addi $t1, $zero, 0x00FF
addi $t2, $zero, 0x0008
addi $t3, $zero, 0x0012
add $t4, $t0, $t2
lw $a1, 0x0000($t1)
sw $a0, 0x0000($t1)
add $s0, $a0, $a1
beq $t4, $t3, 0x0FFF
```

add $t0, $s0, $t3

add $t0, $a0, $a1

add $t1, $t0, $t2

Load-use Hazard

addi $v0, $zero, 0x0010

lw $a0, 0x0000($v0)

add $a1, $a0, $v0

Load-store Hazard

addi $v0, $zero, 0x00FF

lw $a1, 0x0000($v0)

sw $a0, 0x0000($v0)

add $v0, $a0, $a1

Compute-use Hazard

addi $v0, $zero, 0x0008

addi $v1, $zero, 0x0012

add $t0, $v0, $v1

Branch Hazard

addi $v0, $zero, 0x0003

addi $v1, $zero, 0x0004

beq $v0, $v1, 0x0016

add $t0, $v0, $v1

or $t1, $t0, $v0

```
and $t1, $t1, $t0
```