# CONTROL HAZARDS

## Introduction:

Hi Guys! Today we see the  hazards topic do you know about  hazards and what are the types of hazards we will learn like when will hazard occur or if occur control how to solve the  hazards methods we will see the above topic in this presentation.

## Hazard:

Hazard are the problem within the instruction pipeline. When problem occur in the instruction we can not excute the next instruction the following the clock cycle.

## TYPES:

1.Data hazards

2.Structural hazards

3.Control hazards

Here we can see the control hazards. And when will occur and we see the solving methods for control hazards

## Control Hazard Definition:

Control hazards occurs when instruction like branch or jump instruction in the pipeline . Also Control called Branching hazards.

If the branch instruction successful we move onto the target .  But the branch instruction will not successful we execute the remaining instruction.

## Solving Methods:

There are several methods to solve the control hazards. Here we can see the some important methods to solve the control hazards.  They are,

1.Stall

2.Branch Delay Slots

## 1.Stall

In this method we need to stop fetch the instruction until the branch or Jump instruction complete.

### <u>For Example:</u>

**PC:200 I1 SUB R3,R2,R1**

**PC:204 I2 JMP 240**

**PC:208 I3 AND R4,R5,R6**

**PC:240 I4 ADD R7,R8,R9**

### <u>Explanation of the instructions:</u>

1. First, R1 and R2 between subtract and then value stores in R3.
2. Next Jump instruction come so we move into 240 memory address instructions.
3. Then add the R8 and R9 stores the value in R7.
4. Now we move onto Memory address 208.

The Instruction order is I1, I2, I4, I3.

| INSTRUCTION | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| SUB R3,R2,R1 | F | D | C | M | W | |
| JMP 240 | | F | D | C | M | W |
| AND R4,R5,R6 | | | F | D | C | M |
| ADD R7,R8,R9 | | | | F | D | C |

So We use stall method to solution for control hazards

| INSTRUCTION | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| SUB R3,R2,R1 | F | D | C | M | W | |
| JMP 240 | | F | D | C | M | W |
| Delay | - | - | - | - | - | - |
| ADD R7,R8,R9 | | | | F | D | C |

We use Stall method we need ti delay Jump instruction or Branch instruction. So After, Jump or Branch instructon will be done they fetch the correct instruction.

Here we see the above example, in the example first instruction do not have Jump instruction or Branch instruction.So the next instruction will be fetch when the first instruction will decode simultaneously. In the Second instruction we can see the Jump instruction that is JMP 240 instruction until complete the other instruction do not fetch the instruction.

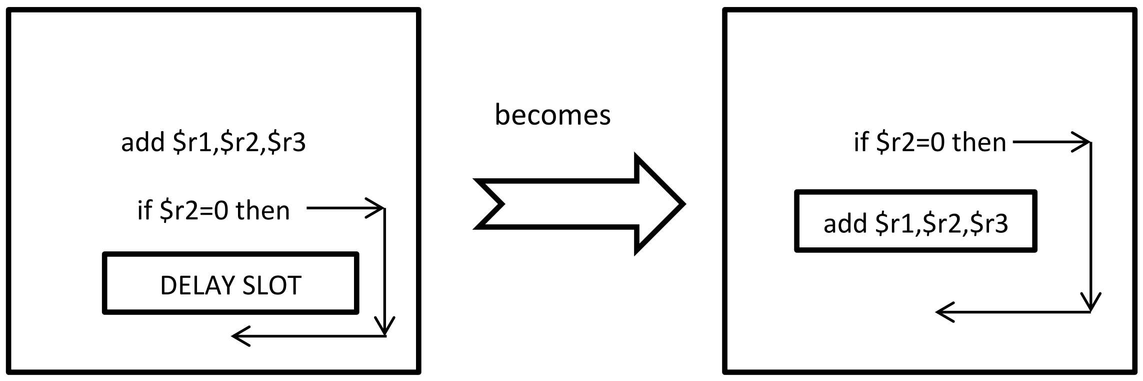
## 2.Branch Delay Slots

The branch delay slot help to keep the pipeline instructions and they improving the performance of the processor. By allow an instruction to be executed immediately following a branch instruction, regardless of whether the branch is taken or not, the processor can continue processing instructions without waiting for the branch condition to be resolved.

This is particularly helpful for in pipelined processors, where multiple instructions are executed simultaneously in the pipeline. The Branch delay slot with useful the processor can minimize pipeline stalls stages and maintain performance.

Here we see the examples:-

a)



You see the this instruction writes something into r1 and this branch is reading r2 there no really data dependence between these two instructions if you reorder these two instructions you still end up with the program that has exactly the same result.

If instruction do not have data dependency the instruction will move to the delay slot. If in the instruction data dependency occur the instruction will not move to the delay slot.

In this example ,

add \$r1, \$r2, \$r3

If \$r2 = 0 then

Delay slot

Following this condition we will see the delay slot whatever the instruction executed regardless of the outcome of the condition.

If the condition is met \$r2 equals 0, then the instruction in the delay slot will be executed alongwith the branch instruction. In this case, the add instruction will be executed, adding the values in \$r2 and \$r3 and storing the result in \$r1.

If the condition is not met \$r2 is not equal to 0, the add instruction will still be executed, as the delay slot instruction is not dependent on the branch outcome.

So we reorder the instruction becomes,

If \$r2 = 0 then

add \$r1, \$r2, \$r3

in the instruction will execute we decrease the instruction lines.