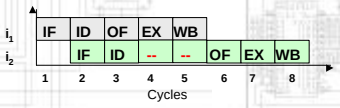
Instruction Pipe Hazards

Three primary data harzards [RAW]

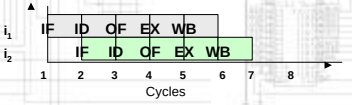
- RAW Read-after-write
  - eg i_1 $R_2 \leftarrow R_3 \div R_4$
  - i_2 $R_5 \leftarrow R_2 + R_1$
  - Delay i_2 – OF until i_1 – WB



Instruction Pipe Hazards

Three primary data harzards [WAR]

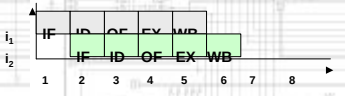
- WAR Write-after-read
  - [concurrent execution – see later]
  - eg i_1 $R_2 \leftarrow R_3 \times R_4$
  - i_2 $R_4 \leftarrow R_5 + R_6$
  - Delay i_2 – WB until i_1 - OF



Instruction Pipe Hazards

Three primary data harzards [WAW]

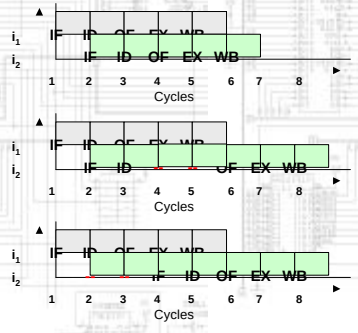
- WAW Write-after-write
  - [concurrent execution – see later]
  - eg i_1 $R_2 \leftarrow R_3 \div R_4$
  - i_2 $R_2 \leftarrow R_2 + R_6$
  - Delay i_2 – WB until i_1 – WB



- Solution
  - The incedence of data dependency hazards may be lowered by having compiler or assembler move:
    - NOPs between them
    - Independent instructions between them

Instruction Pipe Hazards

Solution



Instruction Pipe Hazards

Rearranging the order of instruction execution

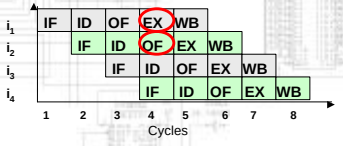
- Problem

i_1 $R_2 \leftarrow R_3 + R_4$

i_2 $R_5 \leftarrow R_2 + R_1$

i_3 $R_6 \leftarrow R_6 + R_7$

i_4 $R_8 \leftarrow R_8 + R_7$



Instruction Pipe Hazards

Rearranging the order of instruction execution

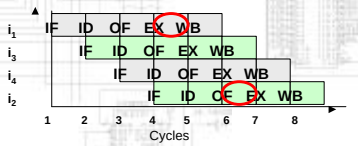
- Solution -> Move i_3, i_4 between i_1 and i_2

i_1 $R_2 \leftarrow R_3 + R_4$

i_3 $R_6 \leftarrow R_6 + R_7$

i_4 $R_8 \leftarrow R_8 + R_7$

i_2 $R_5 \leftarrow R_2 + R_1$



Instruction Pipe Hazards

Insert NOP [no operation]

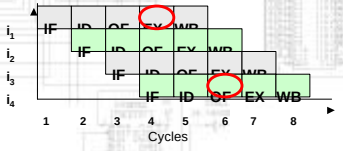
- Insert NOPs

i_1 $R_2 \leftarrow R_3 + R_4$

i_2 NOP

i_3 NOP

i_4 $R_5 \leftarrow R_2 + R_1$



Dynamic dependency Checking

- The existence of control branches means that we must also check dynamically
- Two approaches have evalved:
  - Tomasulo's Method
  - Scoreboard Method

Tomasulo's Method – IBM 360

- Here all results are placed on a common data bus with a TAG to show from where it comes.
- Hence all waiting functional units and registers may obtain a copy simultaneously.