CMP 301 30/1/24

PIPELINING

What makes a pipeline to stall

* Instruction stalling
* Instruction dependency
* Data dependency
* Types of data dependency

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * F | D | * OF | * EX | * WB |

TOTAL OPERATIONS INVOLVED

Pipelining design concepts: when we have instructions and we want to overlap them.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| F1 | D1 | E1 | W1 | F2 | D2 | E2 | W2 | F3 | D3 | E3 | W3 |
|  |  |  |  | Sequential processing |  |  |  |  |  |  |  |
| F1 | D1 | E1 | W1 |  |  |  |  |  |  |  |  |
|  | F2 | D2 | E2 | W2 |  |  |  |  |  |  |  |
|  |  | F3 | D3 | E3 | W3 |  |  |  |  |  |  |

Space against time chart is called the Gantt’s chart

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| finish |  |  |  |  |  | I1 | I2 |  |  | I3 |  |  | I4 |
| WB |  |  |  |  | I1 | I2 |  |  | I3 |  |  | I4 | I5 |
| E |  |  |  | I1 | I2 |  |  | I3 |  |  | I4 | I5 |  |
| OF |  |  | I1 | I2 |  |  | I3 |  |  | I4 | I5 |  | I6 |
| D |  | I1 | I2 | I3 | I3 | I3 | I4 | I4 | I4 | I5 | I6 | I6 |  |
| F | I1 | I2 | I3 | I4 | I4 | I4 | I5 | I5 | I5 | I6 | I7 | I7 |  |

A PIPELINE is said to have been stalled if one-unit stage requires more time to perform its function thus forcing other stages to become idle. The time wasted is called a ‘**Bubble’.**

Pipeline stall due to instruction dependency: correct operation of a pipeline requires that operation performed by a stage must not depend on the operations performed by other stages. Meaning that no instruction should depend on another so that the operation needs more instructions won’t waste much time less time wasted.

If instructions are dependent, it means that future instructions depend on the results of the previous instructions before they can execute.

Example: consider the execution of 10 instructions in a pipeline.

3. pipeline stall caused by data dependency: occurs when the source operand depends on the previous operands execution before it can start executing.

e.g. ADD R1, R2, R3; R3 <- R1 + R2

SL R3; R3 <- SL(R3)

SUB R5, R6, R4; R4 <- R5 – R6

Here we can’t tell the value of the R3 without knowing the values of R1 and R2.

Difference between the instruction and data is that instruction deals with the whole operation while data deals only with the operand.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IS |  |  |  |  | I1 |  |  |  |  |
| IE |  |  |  | I1 | I2 |  |  |  |  |
| OF |  |  | I1 | I2 |  |  |  |  |  |
| ID |  | I1 | I2 | I3 |  |  |  |  |  |
| IF | I1 | I2 | I3 | I4 |  |  |  |  |  |
| . | . | K | K+1 | K+2 | K+3 | K+4 | K+5 | K+6 | . |

Example 3. Consider the execution of the following sequence of the five stage pipelining.

I1 -> Load -1, R1; R1 <- -1;

I2 -> Load 5, R2; R2 <- 5;

I3 -> Sub R2, 1, R2; R2 <- R2 -1;

I4 -> Add R1, R2, R3; R3 <- R1 + R2;

I5 -> Add R4, R5, R6; R6 <- R4 + R5;

I6 -> SL R3; R3 <- SL(R3);

I7 ->Add R6, R4, R7; R7 <- R4 + R6;

RAW (read after write). WAW (write after write) RAR (read after read) WAR (write after read)

AREA OF CONCENTRATION

Slide 6 take it out

Take away slide 3

leave slide 3b

Take out slide 7(floating point)

No compulsory questions

Answer 4 out of 6

We have lecture 1, 2, 3b, 4, 5, 8, 9

write after read

I3 and I2 RAW, WAW

I4 and I1 RAW

I4 and I2 RAW

I4 and I3 RAW

I6 and I4 RAW, WAW

I7 and I5 RAW