(5) # Data Hazards

→ It occurs when the pipeline changes the order of read/write accesses to operands so that the order differs from the order seen by sequentially executing instructions on the unpipelined machine.

Consider the pipelined execution of these instructions:

		1	2	3	4	5	6	7	8	9
ADD	R1, R2, R3	IF	ID	EX	MEM	WB				
SUB	R4, R5, R1		IF	ID_{sub}	EX	MEM	WB			
AND	R6, R1, R7			IF	ID_{and}	EX	MEM	WB		
OR	R8, R1, R9				IF	ID_{or}	EX	MEM	WB	
XOR	R10, R1, R11					IF	ID_{XOR}	EX	MEM	WB

All the instructions after the ADD use the result of the ADD instruction (in R1).
The ADD instruction writes the value of R1 in the WB stage, and the SUB instructions read the value during ID stage (ID_{sub}).
This problem is called a data hazard.

Sunday 24

Handling Hazards

- Structural hazards
  - Stalling : pipeline interlock
  - Code Scheduling

- Data hazards
  - Stalling : pipeline interlock
  - Forwarding
  - Load delay
    - * Stalling : pipeline interlock
    - * Code Scheduling : Fill the load delay slot.

- Control hazards
  - Early branch resolution
  - Stalling : Flushing the pipeline
  - Delayed branch
  - Predict non taken (or taken)
  - static branch prediction
  - Dynamic branch prediction

Static Branch Prediction

- Compiler determines whether branch is likely to be taken or likely to be not taken.
- Decision is based on analysis or profile information.
  - 90% of backward-going branches are taken.
  - 50% of forward-going branches are not taken.
  - BTFN: "backwards taken, forwards not-taken"
  - Used in ARC 600 and ARM 11
- Decision is encoded in the branch instructions themselves.
  - Uses 1 bit: 0 ⇒ not likely to branch,
    1 ⇒ likely to branch.
- Prediction may be wrong!
  - Must kill instructions in the pipeline when a bad decision is made.
  - Speculatively issued instructions must not change processor state.

Dynamic Branch Prediction

- Monitor branch behaviour and learn
- Predict present behaviour based on past behaviour
- Identify individual branches by their PC
- Predict
  - Outcome : taken or not taken
  - Target : address of instruction to branch to
- check actual outcome and fix future predictions
- Squash incorrectly executed instructions
- Simplest Scheme : 1-bit branch prediction (taken/not-taken) indexed by PC.
  - Incures at least 2 mis-predictions per loop.