

A collection of historical artifacts is arranged on a light-colored surface. In the top left, a portion of a wooden chessboard with a checkered pattern and several chess pieces is visible. Below the chessboard, there are several medals and orders. One prominent medal is a red ribbon with a circular emblem. Another is a blue ribbon with a circular emblem. A third is a white star-shaped medal with a central emblem. A pair of round-rimmed glasses with thin metal frames lies diagonally across the lower left. In the bottom left corner, a small, round, silver-colored compass with a white face and black markings is visible.

Advanced Computer Architecture

Floating Point Pipeline Hazard

Prof. Roger Luis Uy
De La Salle University
College of Computer Studies



Floating Point Pipeline

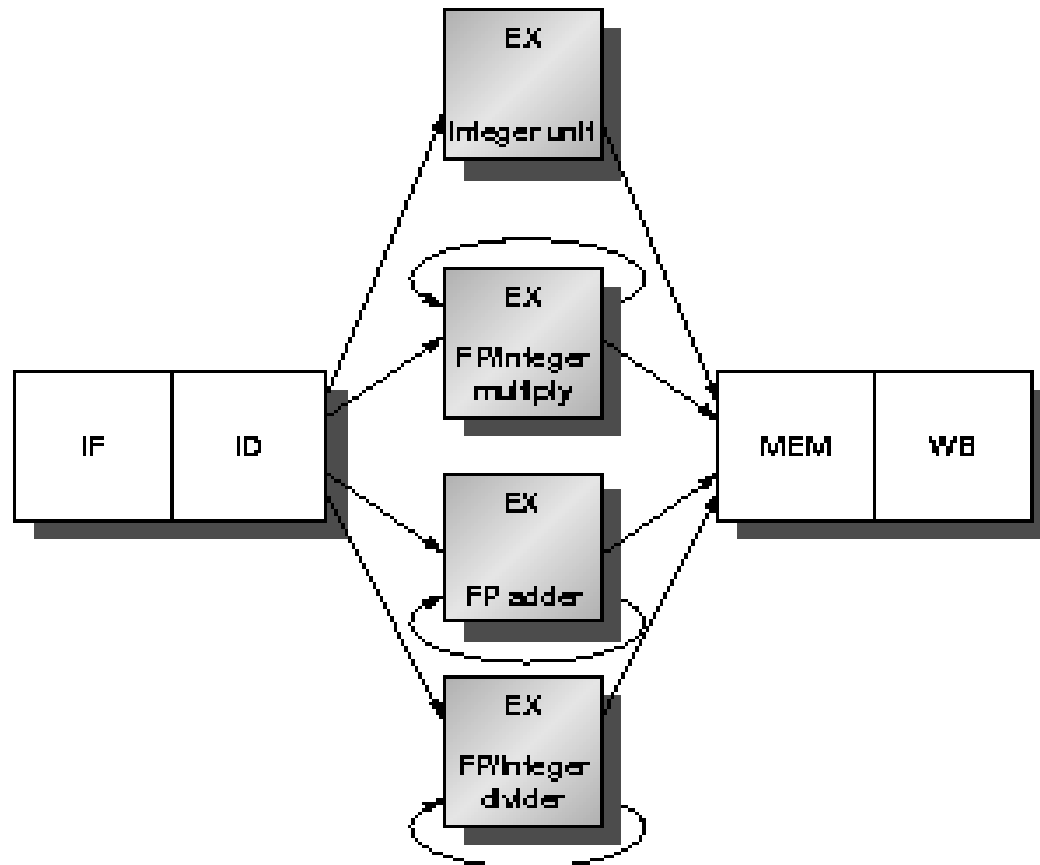
- ◆ MIPS Floating Point operations may not be completed in one clock cycle
- ◆ Solution?
 - Multiple floating point functional units
 - Pipeline the execution unit



Floating Point Pipeline

- ◆ For MIPS FP implementation, there will be 4 separate functions execution units
 - Integer unit: loads, store, integer ALU operations, branches
 - FP and integer multiplier
 - FP adder: FP add, subtract, conversion
 - FP and integer divider

MIPS FP Pipeline Structure





Floating Point Pipeline

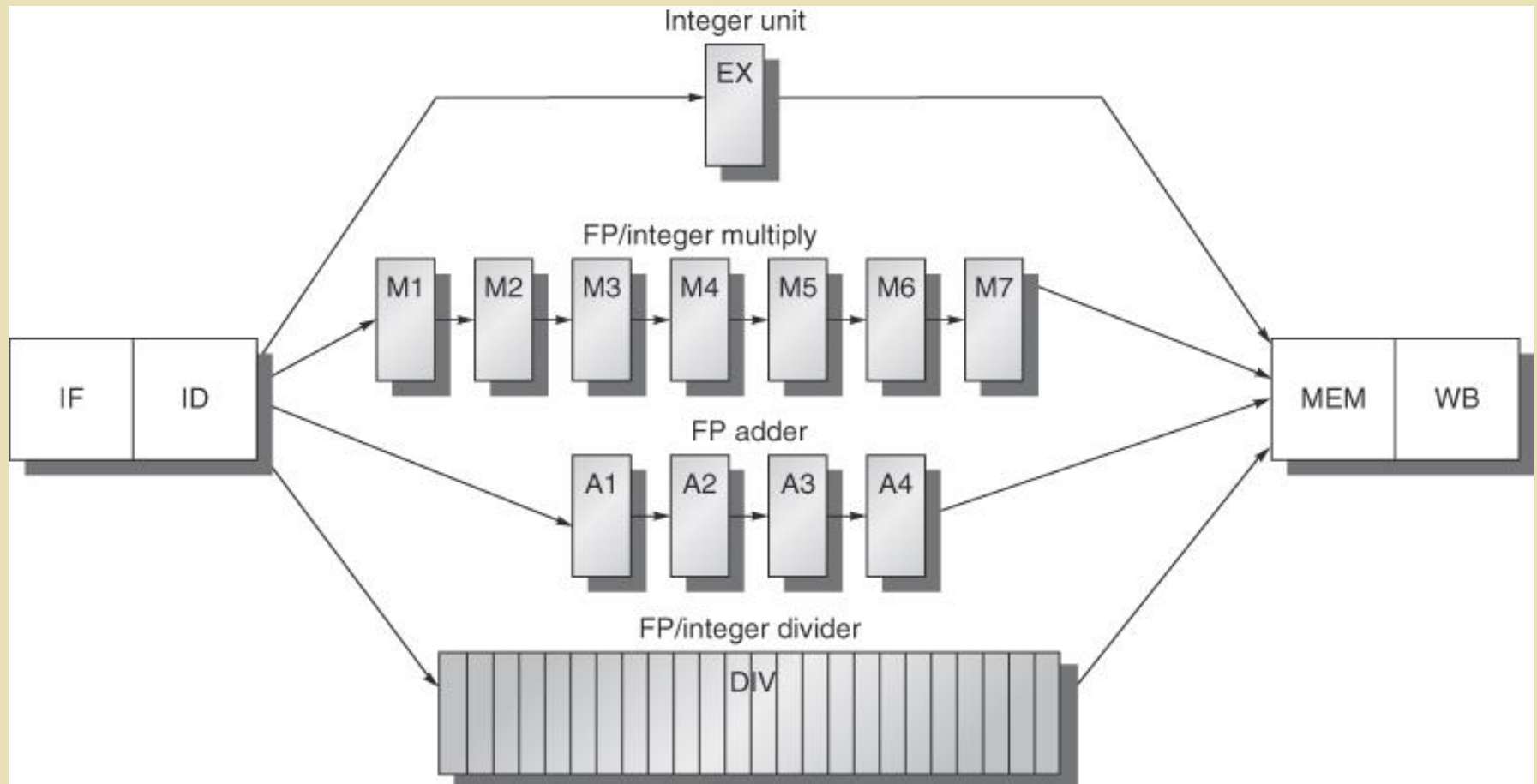
- ◆ Latency - the number of intervening cycles between an instruction that produces a result and an instruction that uses the result
- ◆ Note that the latency is usually the number of stages after EX that an instruction produces a result
- ◆ The initiation interval is the number of cycles that must elapse between issuing 2 operations of a given type



Floating Point Pipeline

Functional Unit	Latency	Initiation interval
Integer ALU	0	1
Data memory (int and FP loads)	1	1
FP add	3	1
FP multiply (integer multiply)	6	1
FP divide (integer divide)	24	25

*Assume that FP add has 4-pipelined execution unit (EU), FP multiply has 6-pipelined EU and FP divide has 24 un-pipeline execution unit



A pipeline that supports multiple outstanding FP operations. The FP multiplier and adder are fully pipelined and have a depth of seven and four stages, respectively. The FP divider is not pipelined, but requires 24 clock cycles to complete. The latency in instructions between the issue of an FP operation and the use of the result of that operation without incurring a RAW stall is determined by the number of cycles spent in the execution stages. For example, the fourth instruction after an FP add can use the result of the FP add. For integer ALU operations, the depth of the execution pipeline is always one and the next instruction can use the results.



Data Hazards in Longer Latency Pipeline

- ◆ 4 types of hazard will occur:
 - Read after Read (RAR) – not a hazard
 - Read after Write (RAW)
 - Write after Read (WAR)
 - Write after Write (WAW)

Illustration of Read after Read (RAR)

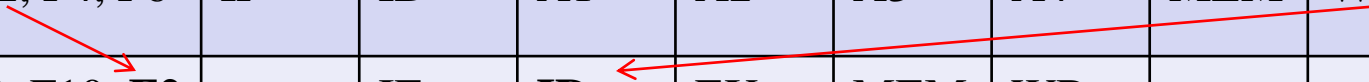
Instruction	1	2	3	4	5	6	8	9	10	11
ADD.D F0, F2, F4	IF	ID	A1	A2	A3	A4	MEM	WB		
SUB.D F10, F8, F4		IF	ID	A1	A2	A3	A4	MEM	WB	
SUB.D F14, F12, F4			IF	ID	A1	A2	A3	A4	MEM	WB

Assumption: Floating-point Add/Sub uses 4-stage pipeline execution unit (A1, A2, A3 & A4)

***RAR is not a hazard.** Since a read operation is non-destructive. Also, the order of the reads is not important

Illustration of Read after Write (RAW)

Instruction	1	2	3	4	5	6	7	8	9
ADD.D F2, F4, F6	IF	ID	A1	A2	A3	A4	MEM	WB	
SUB.D F8, F10, F2		IF	ID	EX	MEM	WB			



Assumption: Floating-point Add/Sub uses 4-stage pipeline execution unit (A1, A2, A3 & A4)

- RAW is a **hazard**. This is a common type of data hazard. Register F2 of SUB.D is initialized with an out-dated value since ADD.D instruction has not yet updated the value of register F2.
- Corresponds to a true data dependence
- This type of hazard can be solved using forwarding

Illustration of Write after Write (WAW)

Instruction	1	2	3	4	5	6	7	8	9
ADD.D F2, F4, F6	IF	ID	A1	A2	A3	A4	MEM	WB	
NOP		IF	ID	EX	MEM	WB			
L.D F2, 0(R2)			IF	ID	EX	MEM	WB		

Assumption: Floating-point Add/Sub uses 4-stage pipeline execution unit (A1, A2, A3 & A4)

- WAW is a **hazard**. Because floating-point instruction has a longer pipeline latency, this causes out-of-order execution. The L.D instruction was able to perform a Write-Back (WB) first on F2 before ADD.D instruction performs a WB on F2. This causes a WAW hazard.
- It is an output dependence and can be viewed as “artificial” dependence (can be resolve using register renaming)

Illustration of Write after Read (WAR)

ADD.D F0, F2, F4

SUB.D F2, F6, F8



- ADD.D uses the wrong “updated” F2 value since SUB.D updates the F2 value first
- Corresponds to antidependence
- Can’t happen in MIPS pipeline (reads in ID, writes in WB)
 - can happen if: out-of-order execution
 - can happen if: early writes (e.g., Auto-increment addressing mode)