

ECE/CSE 511: Computer Architecture

Lecture 10: Superscalar Processor

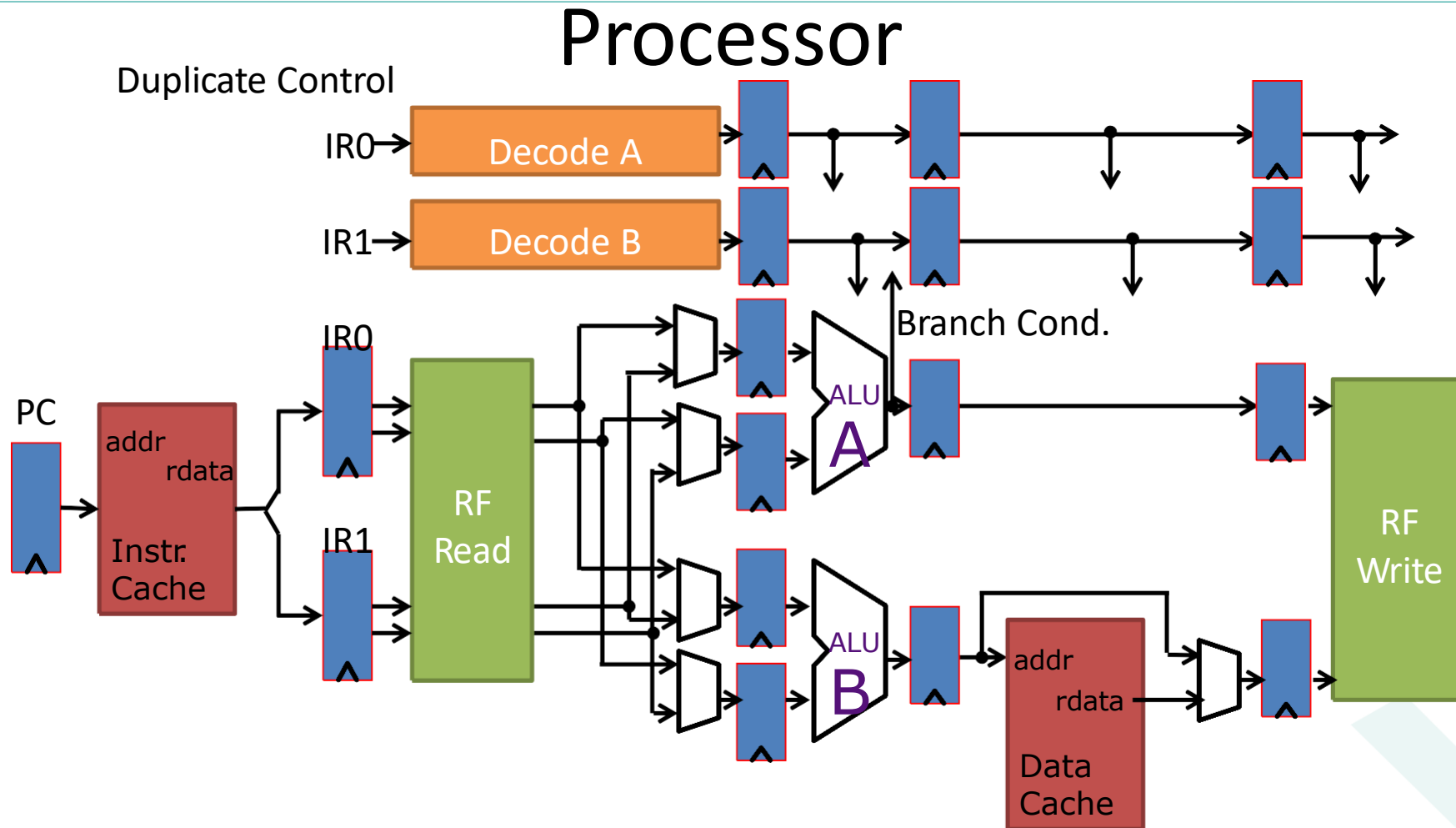


INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI



AMS Lab @IIITD

Baseline 2-Way In-Order Superscalar



Pipe A: Integer Ops., Branches
Pipe B: Integer Ops., Memory

Issue Logic Pipeline Diagrams



OpA	F	D	A0	A1	W		
OpB	F	D	B0	B1	W		
OpC		F	D	A0	A1	W	
OpD		F	D	B0	B1	W	
OpE			F	D	A0	A1	W
OpF			F	D	B0	B1	W

CPI = 0.5 (IPC = 2)

Double Issue Pipeline
Can have two instructions in
same stage at same time

ADDIU	F	D	A0	A1	W						
LW	F	D	B0	B1	W						
LW		F	D	B0	B1	W					
ADDIU		F	D	A0	A1	W					
LW			F	D	B0	B1	W				
LW			F	D	D	B0	B1	W			




Instruction Issue Logic swaps from
natural position

Structural
Hazard

With Alignment Constraints



Cyc	Addr	Instr
?	0x000	OpA
?	0x004	OpB
?	0x008	OpC
?	0x00C	J 0x100
...		
?	0x100	OpD
?	0x104	J 0x204
...		
?	0x204	OpE
?	0x208	J 0x30C
...		
?	0x30C	OpF
?	0x310	OpG
?	0x314	OpH

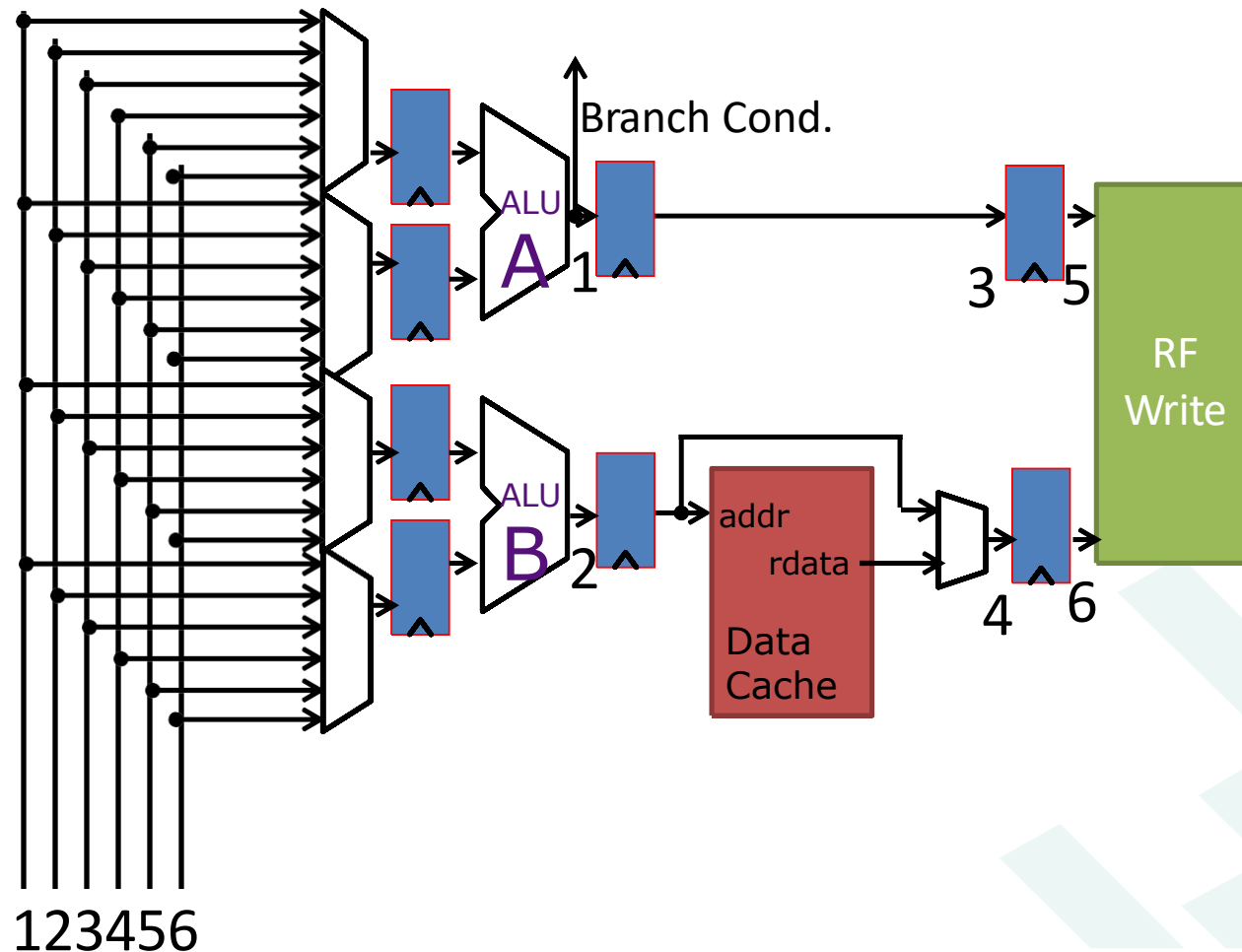
0x000	0	0	1	1
...				
0x100	2	2		
...				
0x200	3 	3	4	4 
...				
0x300			5 	5
0x310	6	6		

With Alignment Constraints



Cyc	Addr	Instr	F	D	A0	A1	W						
1	0x000	OpA	F	D	A0	A1	W						
1	0x004	OpB	F	D	B0	B1	W						
2	0x008	OpC		F	D	B0	B1	W					
2	0x00C	J 0x100		F	D	A0	A1	W					
3	0x100	OpD			F	D	B0	B1	W				
3	0x104	J 0x204			F	D	A0	A1	W				
4	0x200	?				F	-	-	-	-			
4	0x204	OpE				F	D	A0	A1	W			
5	0x208	J 0x30C					F	D	A0	A1	W		
5	0x20C	?					F	-	-	-	-		
6	0x308	?						F	-	-	-	-	
6	0x30C	OpF						F	D	A0	A1	W	
7	0x310	OpG							F	D	A0	A1	W
7	0x314	OpH							F	D	B0	B1	W

Bypassing in Superscalar Pipelines



Breaking Decode and Issue Stage



- Bypass Network can become very complex
- Can motivate breaking Decode and Issue Stage

D = Decode, Possibly resolve structural Hazards

I = Register file read, Bypassing, Issue/Steer
Instructions to proper unit

OpA	F	D	I	A0	A1	W	
OpB	F	D	I	B0	B1	W	
OpC		F	D	I	A0	A1	W
OpD		F	D	I	B0	B1	W

Exceptions

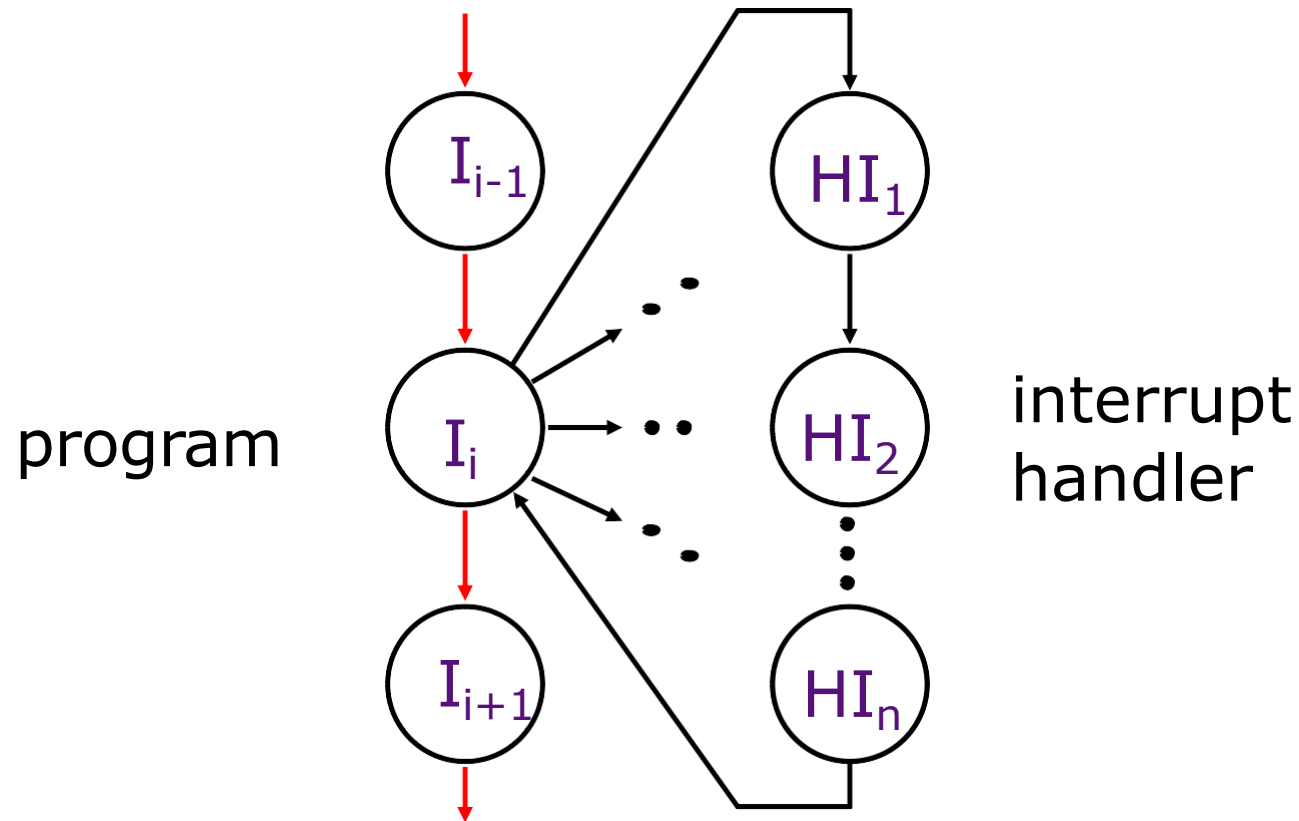


INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI



Interrupts:

altering the normal flow of control



An external or internal event that needs to be processed by another (system) program. The event is usually unexpected or rare from program's point of view.

Interrupt: an *event* that requests the attention of the processor

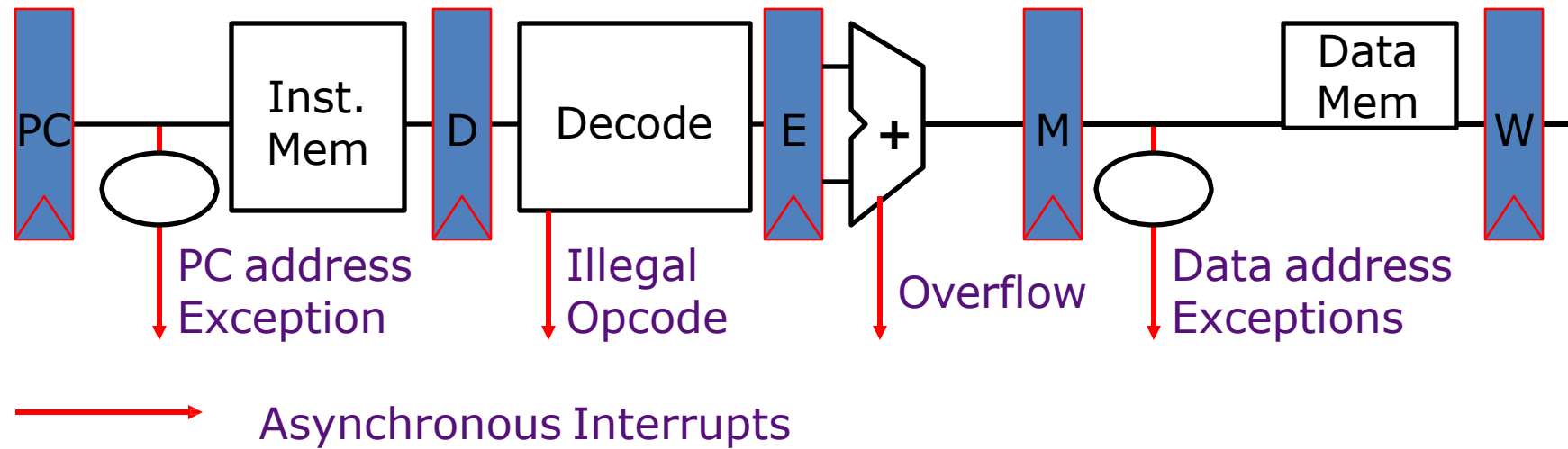
- Asynchronous: an *external event*
 - input/output device service request
 - timer expiration
 - power disruptions, hardware failure
- Synchronous: an *internal exception* (*a.k.a. exceptions/trap*)
 - undefined opcode, privileged instruction
 - arithmetic overflow, FPU exception
 - misaligned memory access
 - *virtual memory exceptions*: page faults, TLB misses, protection violations
 - *software exceptions*: system calls, e.g., jumps into kernel

- An I/O device requests attention by asserting one of the *prioritized interrupt request lines*
- When the processor decides to process the interrupt
 - It stops the current program at instruction I_i , completing all the instructions up to I_{i-1} (a *precise interrupt*)
 - It saves the PC of instruction I_i in a special register (EPC)
 - It disables interrupts and transfers control to a designated interrupt handler running in the kernel mode

- Saves EPC before re-enabling interrupts to allow nested interrupts \Rightarrow
 - need an instruction to move EPC into GPRs
 - need a way to mask further interrupts at least until EPC can be saved
- Needs to read a *status register* that indicates the cause of the interrupt
- Uses a special indirect jump instruction RFE (*return-from-exception*) to resume user code, this:
 - enables interrupts
 - restores the processor to the user mode
 - restores hardware status and control state

- A synchronous interrupt (exception) is caused by a *particular instruction*
- In general, the instruction cannot be completed and needs to be *restarted* after the exception has been handled
 - requires undoing the effect of one or more partially executed instructions
- In the case of a system call trap, the instruction is considered to have been completed
 - syscall is a special jump instruction involving a change to privileged kernel mode
 - Handler resumes at instruction after system call

Exception Handling 5-Stage Pipeline

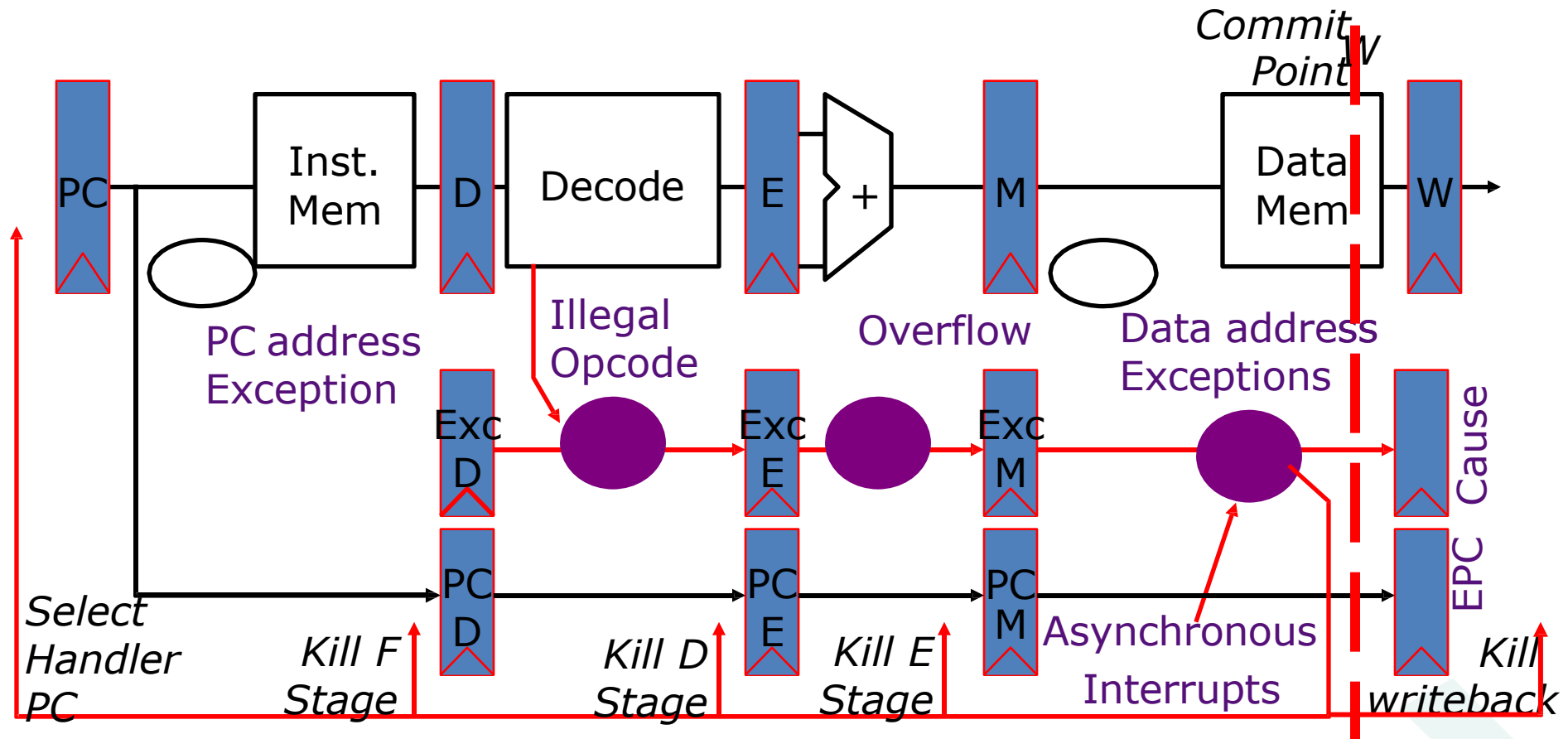


- How to handle multiple simultaneous exceptions in different pipeline stages?
- How and where to handle external asynchronous interrupts?

Class Interaction # 10



Exception Handling 5-Stage Pipeline

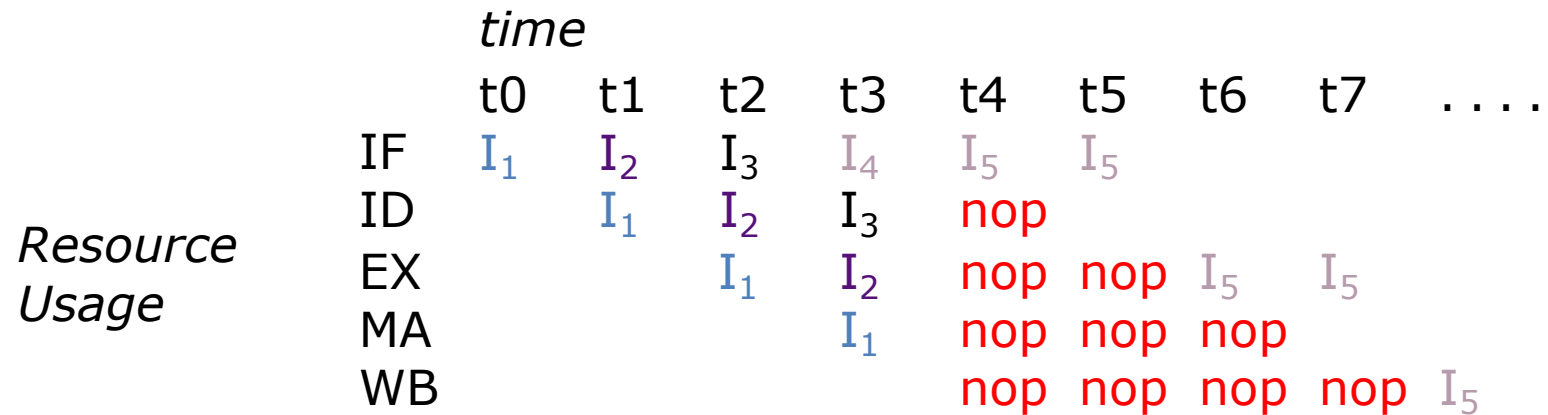
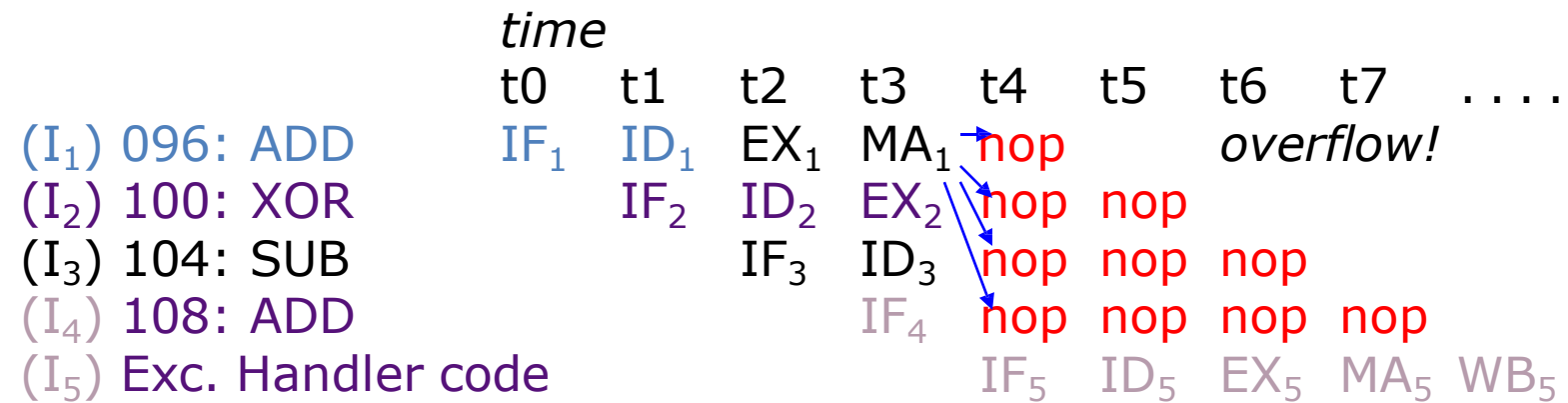


- Hold exception flags in pipeline until commit point (M stage)
- Exceptions in earlier pipe stages override later exceptions *for a given instruction*
- Inject external interrupts at commit point (override others)
- If exception at commit: update Cause and EPC registers, kill all stages, inject handler PC into fetch stage



- **Prediction mechanism**
 - Exceptions are rare, so simply predicting no exceptions is very accurate!
- **Check prediction mechanism**
 - Exceptions detected at end of instruction execution pipeline, special hardware for various exception types
- **Recovery mechanism**
 - Only write architectural state at commit point, so can throw away partially executed instructions after exception
 - Launch exception handler after flushing pipeline
- **Bypassing allows use of uncommitted instruction results by following instructions**

Exception Pipeline Diagram



ECE/CSE 511: Computer Architecture

Out-of-Order Processors



INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI



Out-Of-Order (OOO) Introduction



Name	Frontend	Issue	Writeback	Commit	
I4	IO	IO	IO	IO	Fixed Length Pipelines Scoreboard
I2O2	IO	IO	OOO	OOO	Scoreboard
I2OI	IO	IO	OOO	IO	Scoreboard, Reorder Buffer, and Store Buffer
IO3	IO	OOO	OOO	OOO	Scoreboard and Issue Queue
IO2I	IO	OOO	OOO	IO	Scoreboard, Issue Queue, Reorder Buffer, and Store Buffer