CMPE 200
Computer Architecture & Design

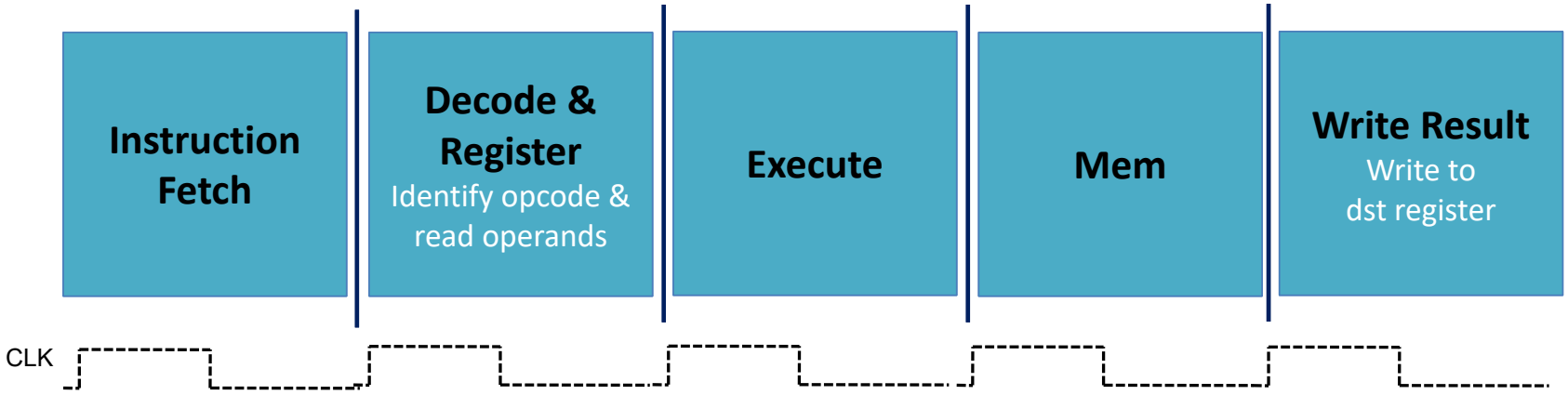
Lecture 3.
Processor Microarchitecture and Design (3)

Haonan Wang

SAN JOSÉ STATE
UNIVERSITY

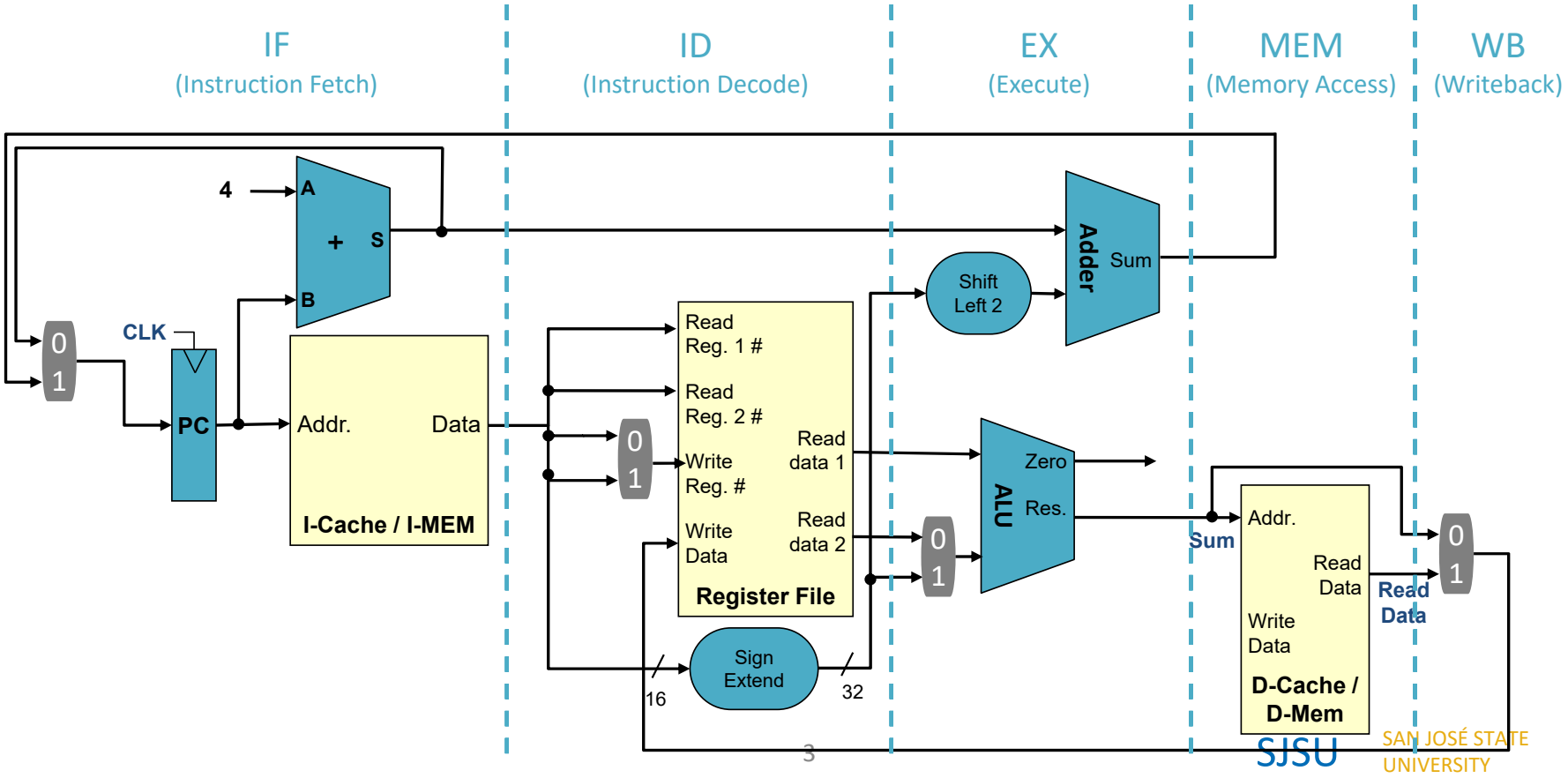
MIPS Pipeline

- **Entire datapath can be broken into five stages**
- **Cycle period is changed to the time taken for a stage**
  - Shorter clock cycle
  - Higher CPI (e.g., CPI = 5 in MIPS)



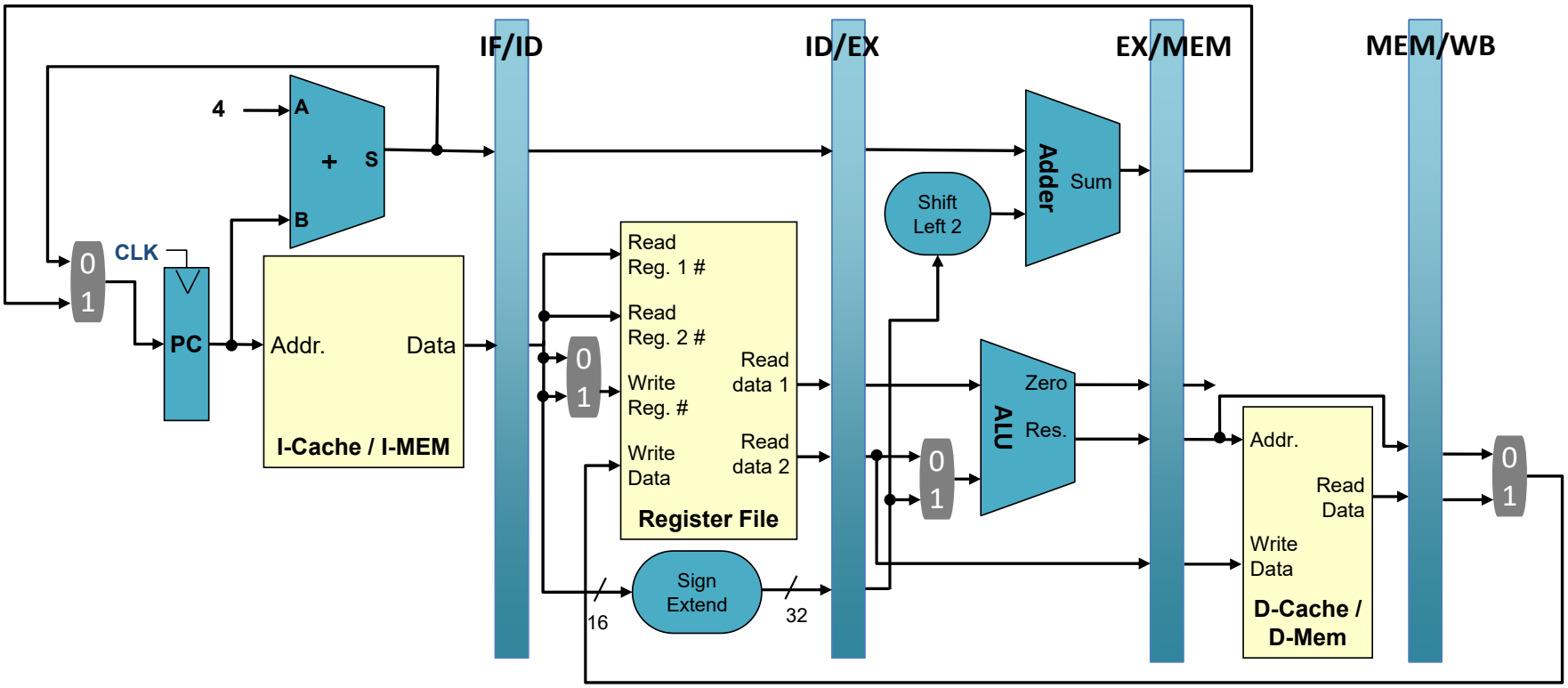
Basic 5 Stage Pipeline

- **Same structure as single cycle but now broken into 5 stages**

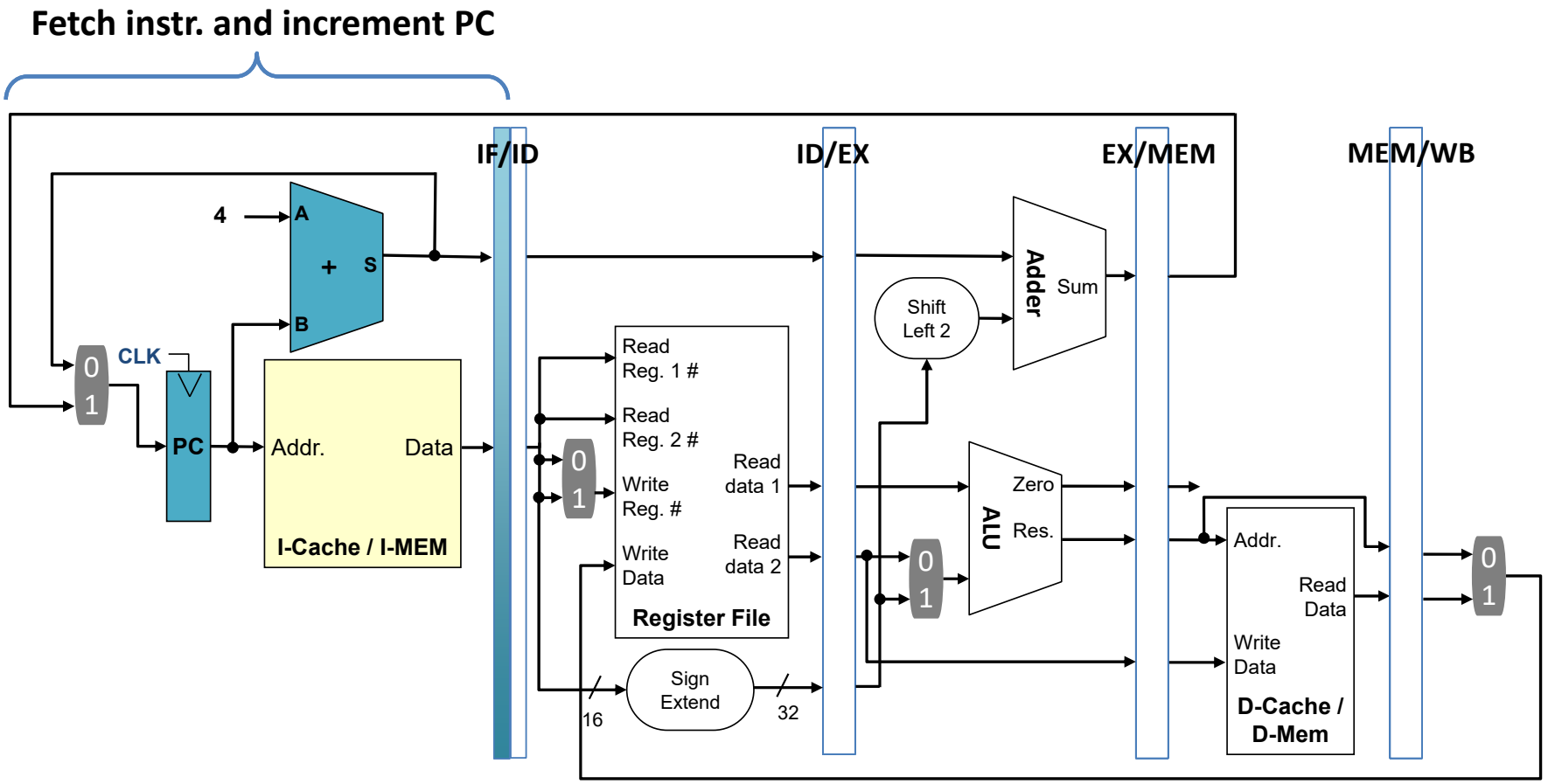


Pipeline Stage Registers

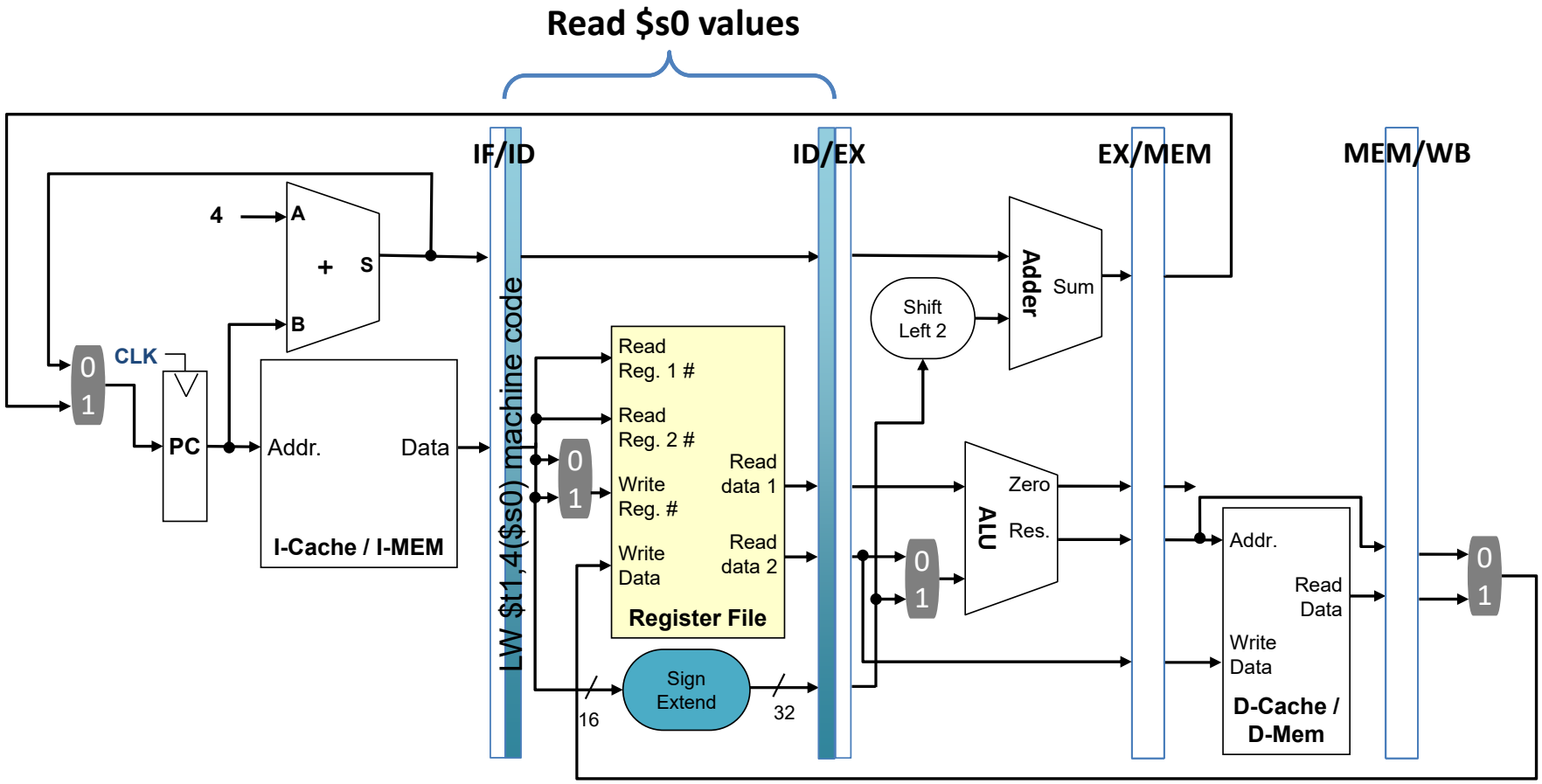
- **Intermediate results need to be stored to allow stage reuse**



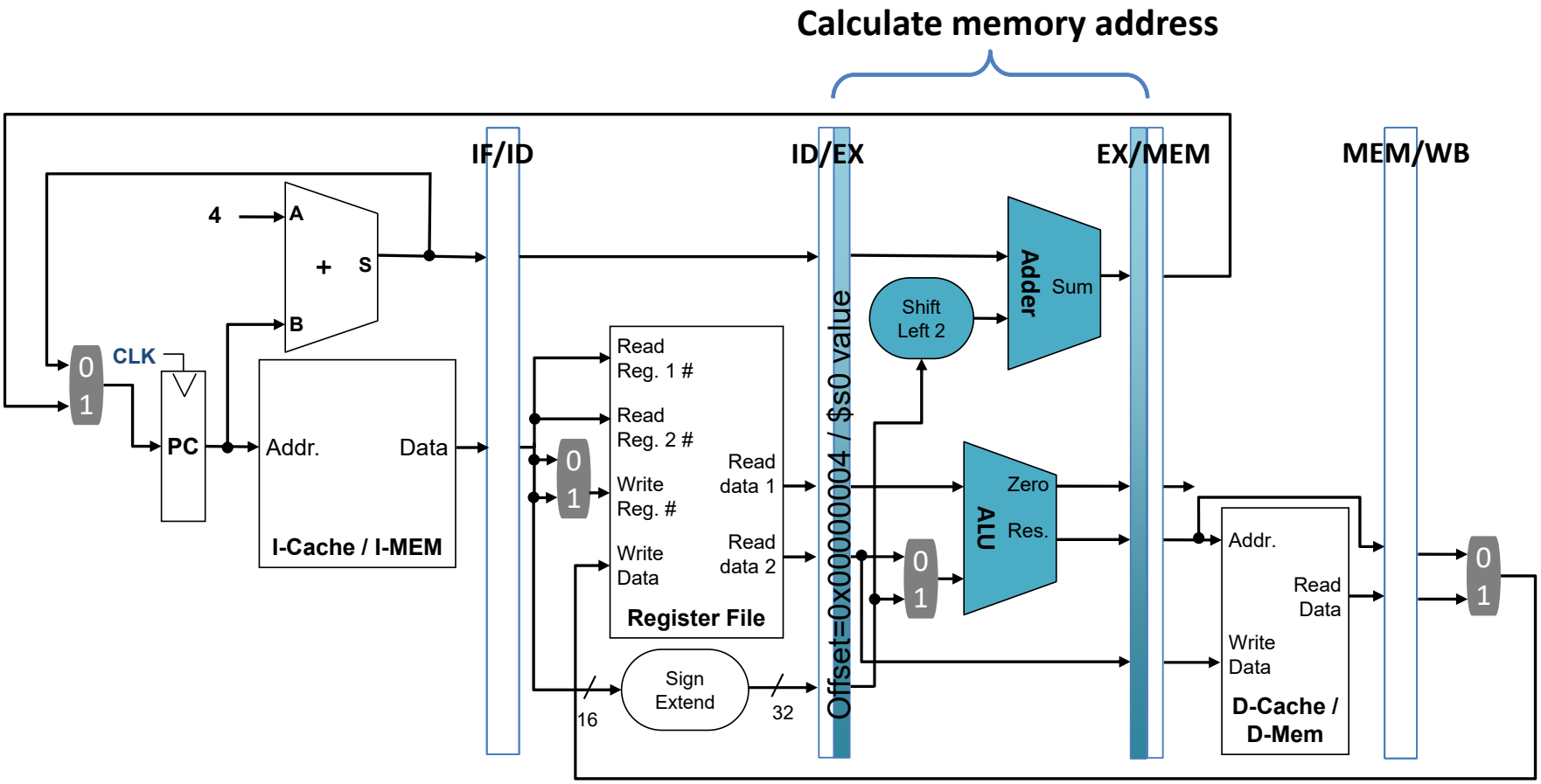
Pipeline Example: LW $t1, 4($s0)



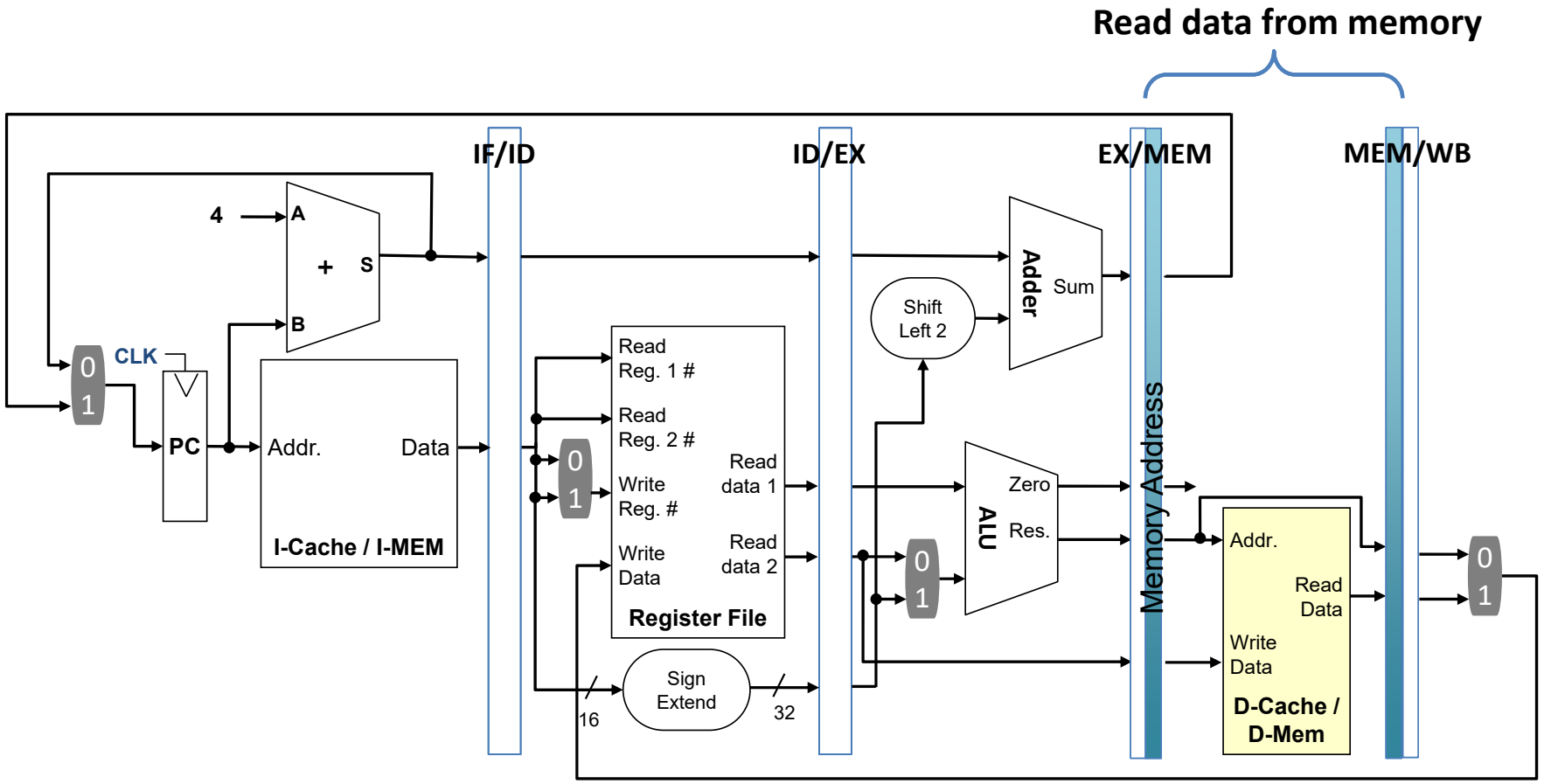
Pipeline Example: LW $t1, 4($s0)



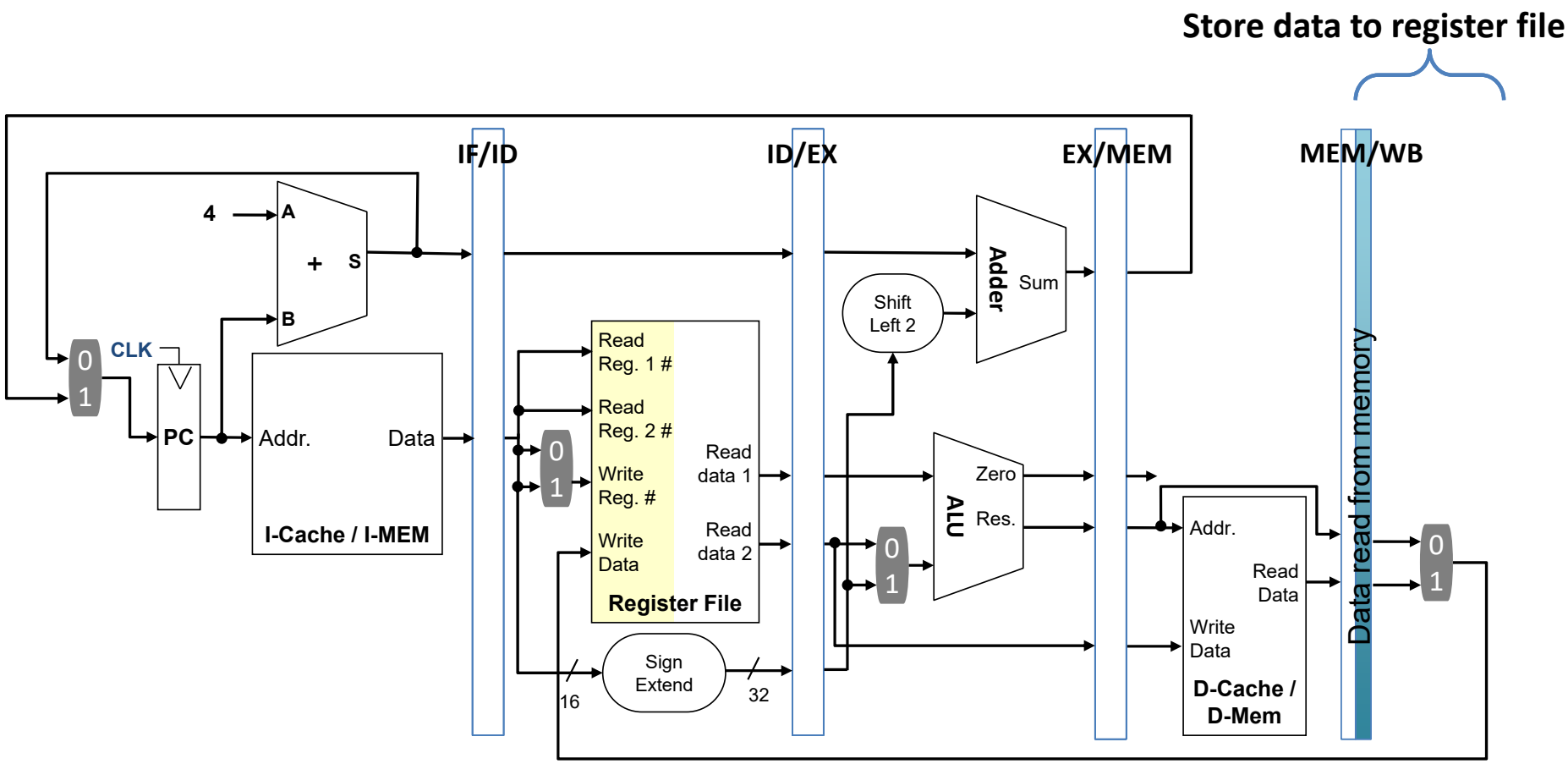
Pipeline Example: LW $t1, 4($s0)



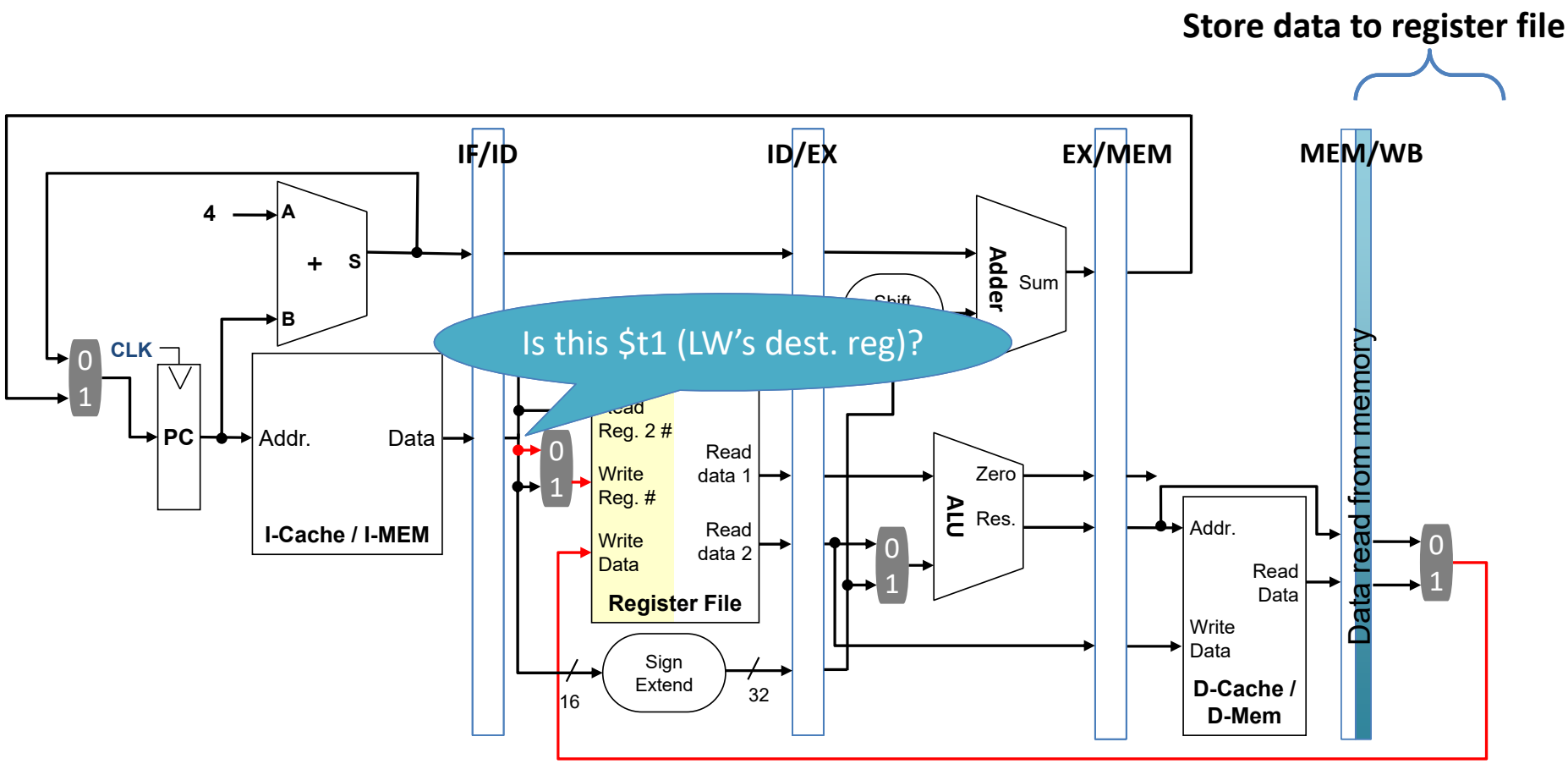
Pipeline Example: LW $t1, 4($s0)



Pipeline Example: LW $t1, 4($s0)

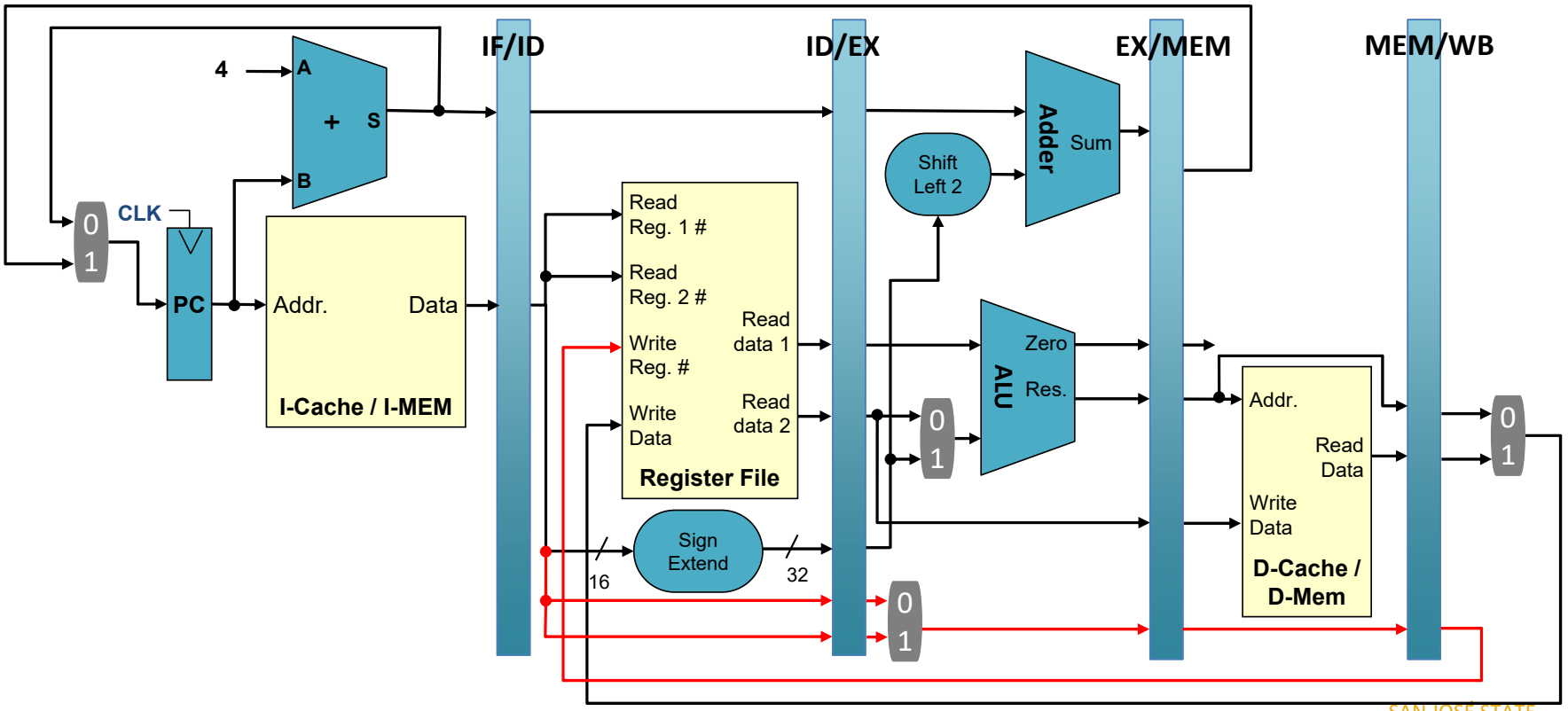


Pipeline Example: LW $t1, 4($s0)

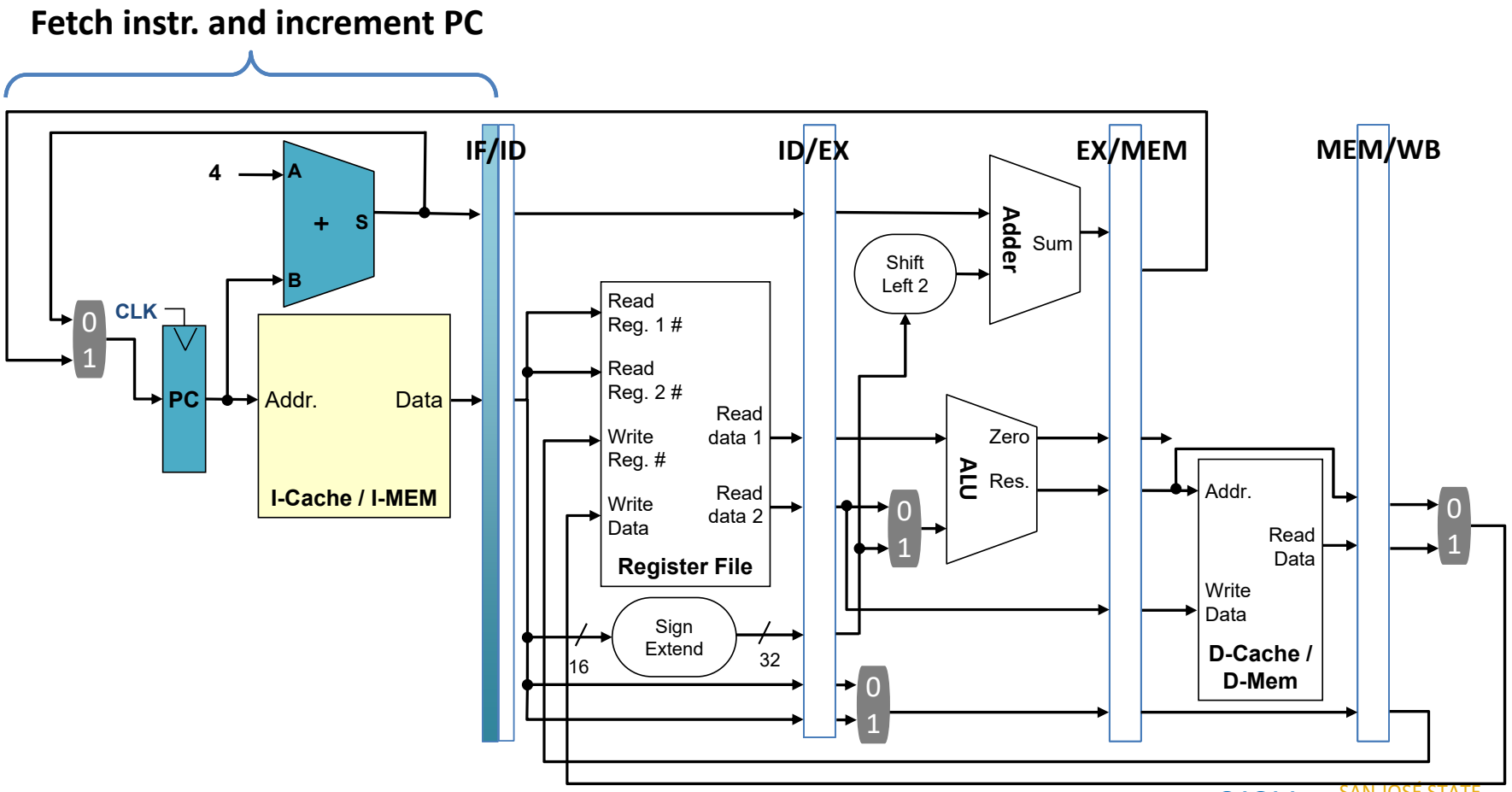


Revised Datapath

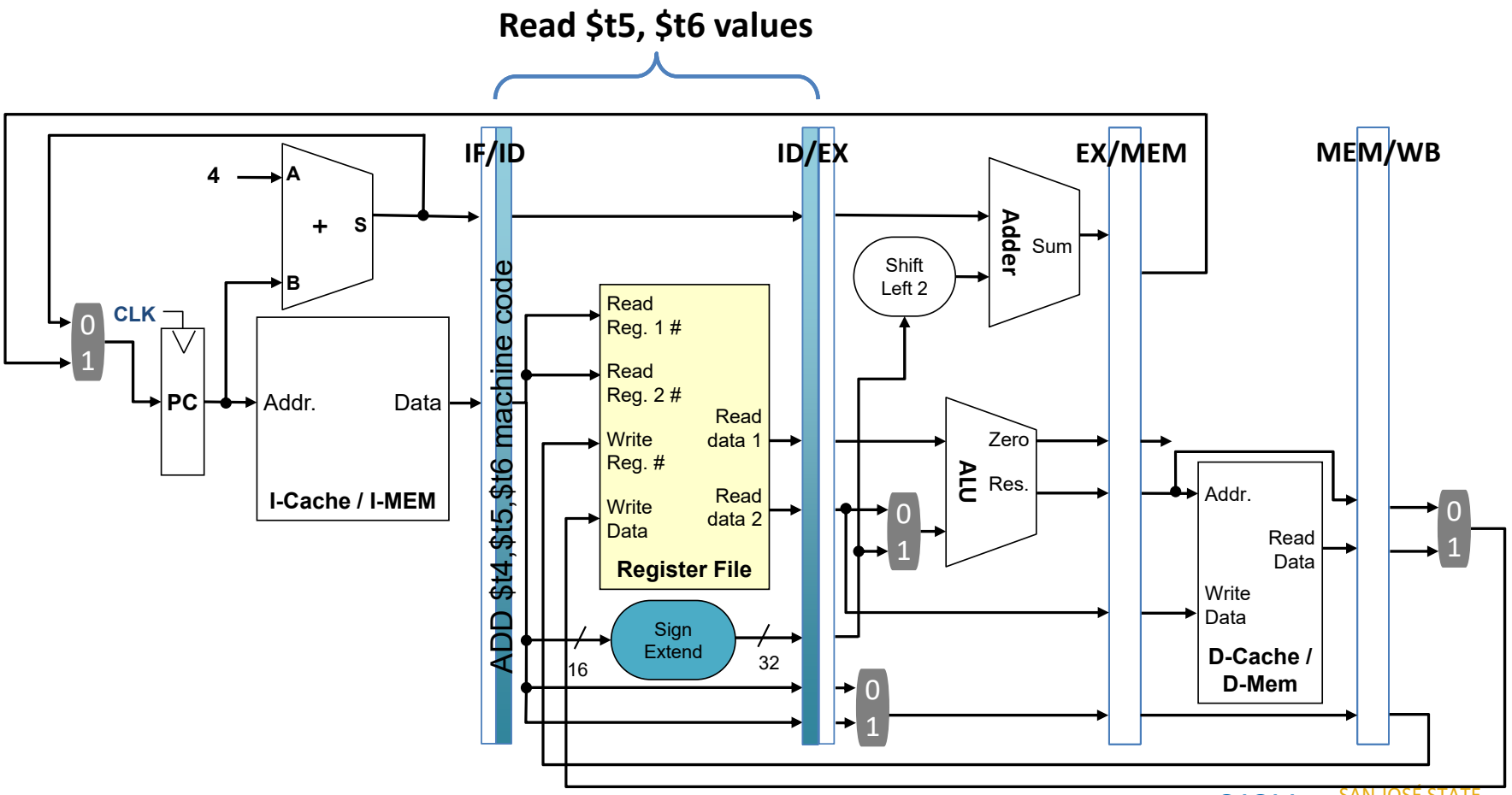
- Dest. register number should be stored in the pipeline registers
- The dest. register number is passed together with data during writeback



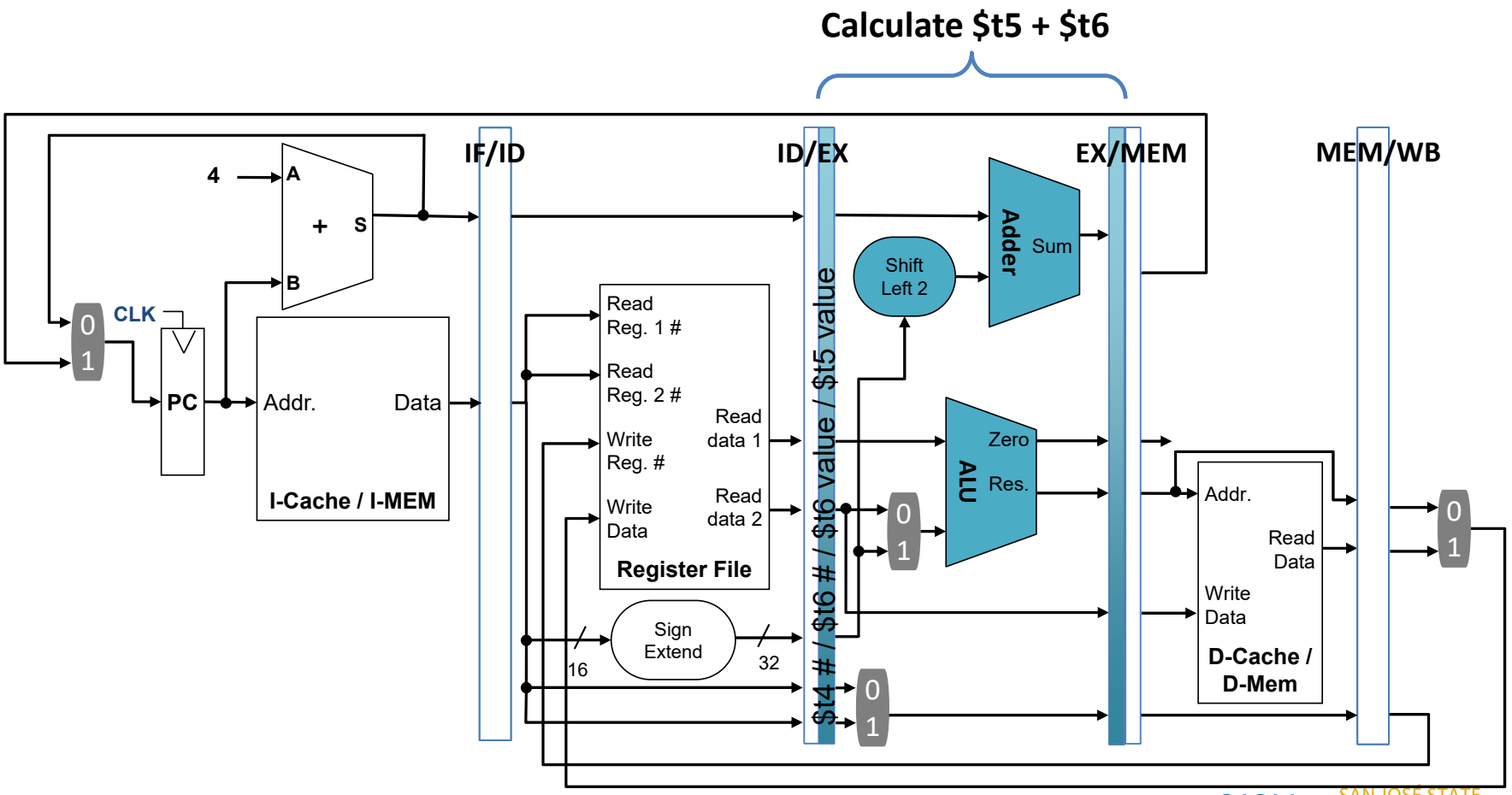
Pipeline Example: ADD $t4,$t5,$t6



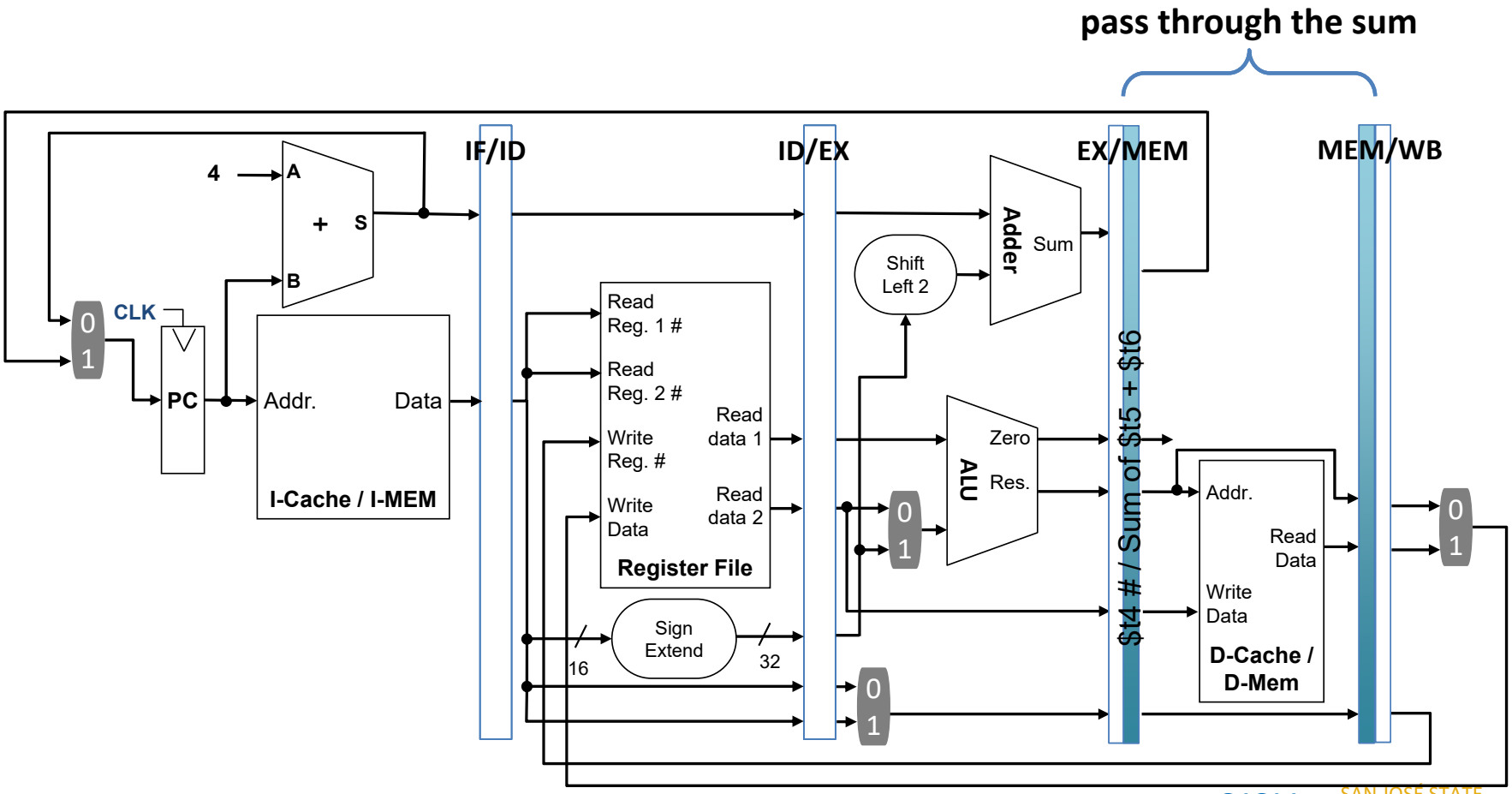
Pipeline Example: ADD $t4,$t5,$t6



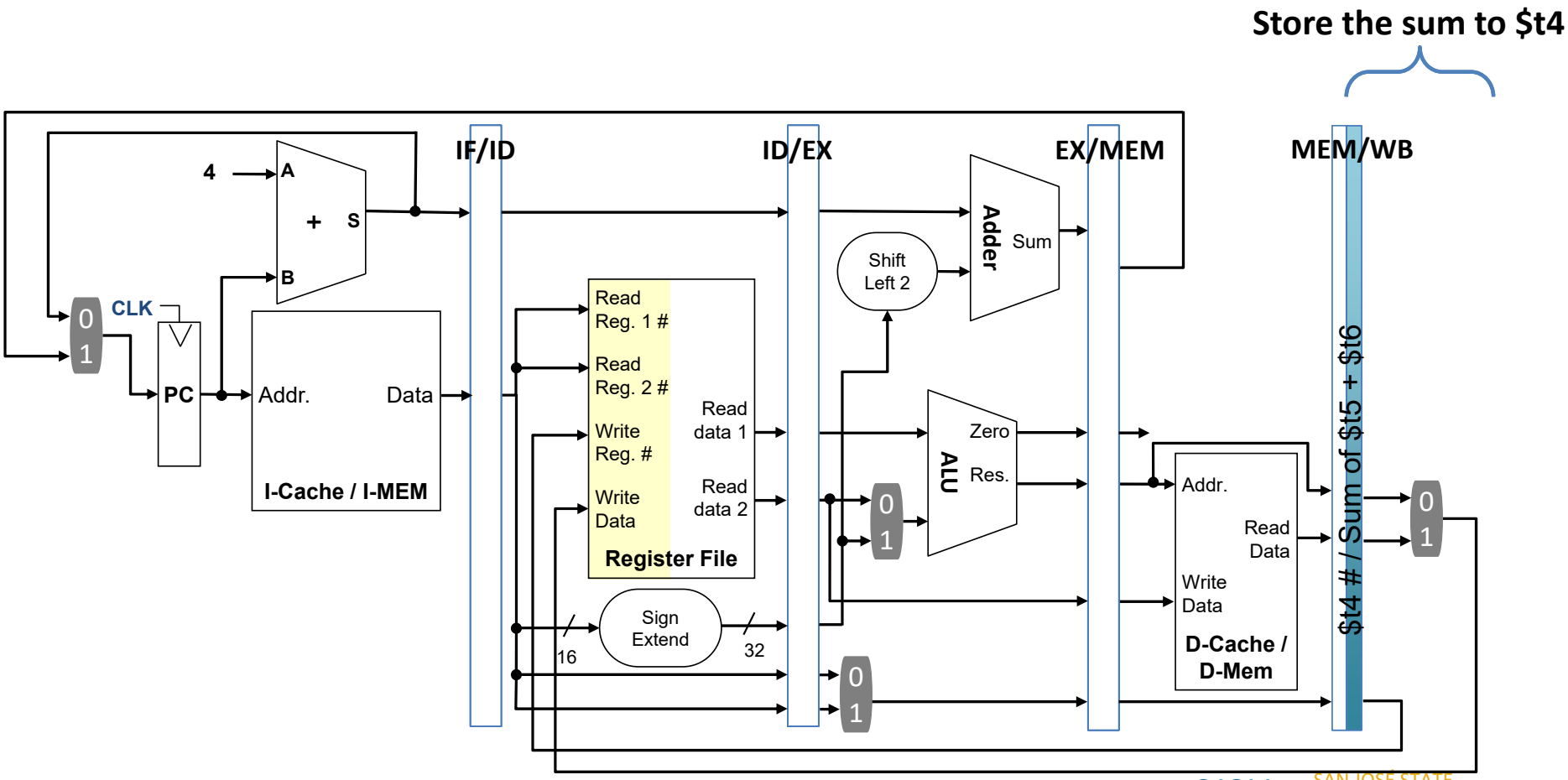
Pipeline Example: ADD $t4,$t5,$t6



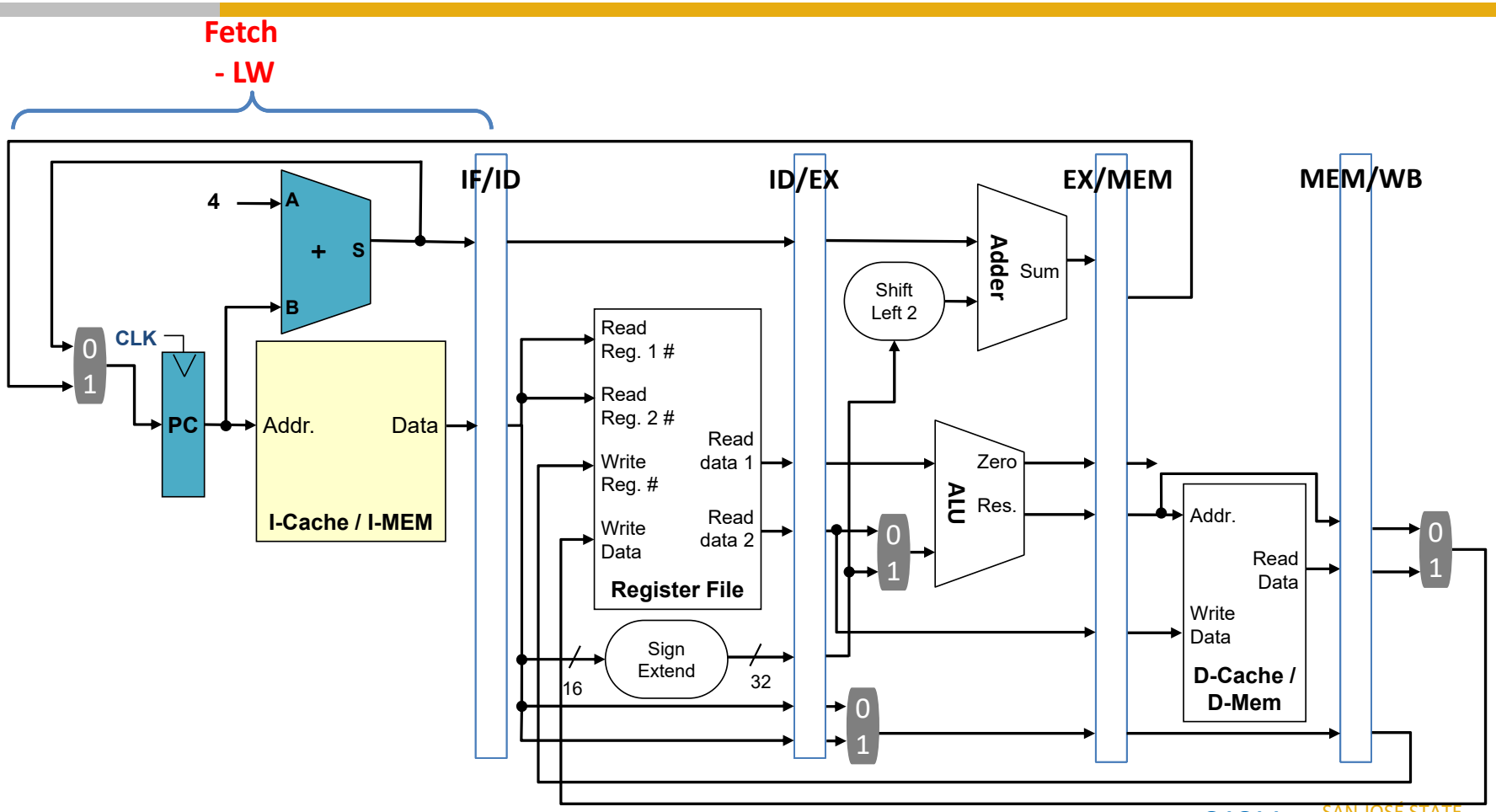
Pipeline Example: ADD $t4,$t5,$t6



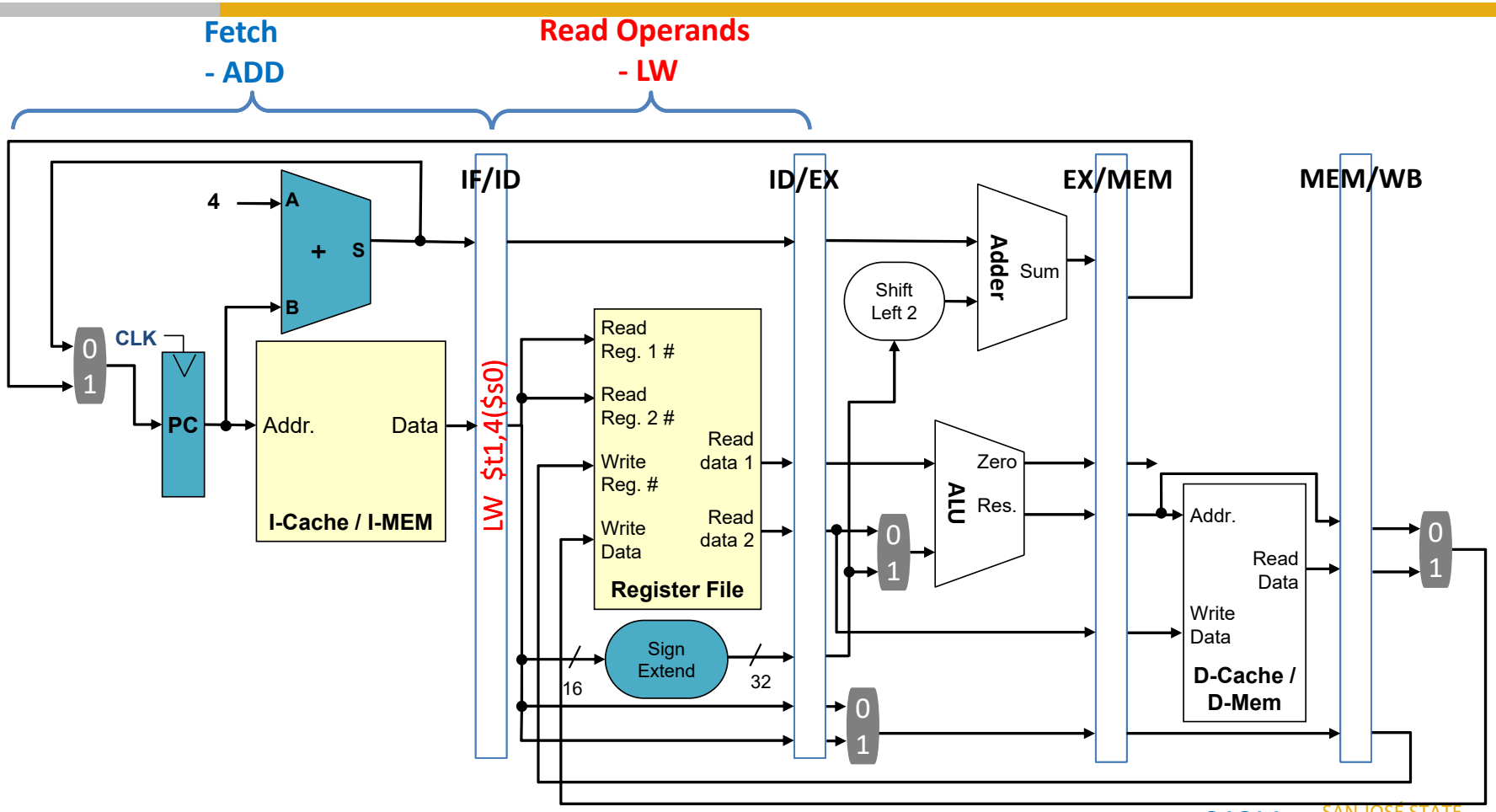
Pipeline Example: ADD $t4,$t5,$t6



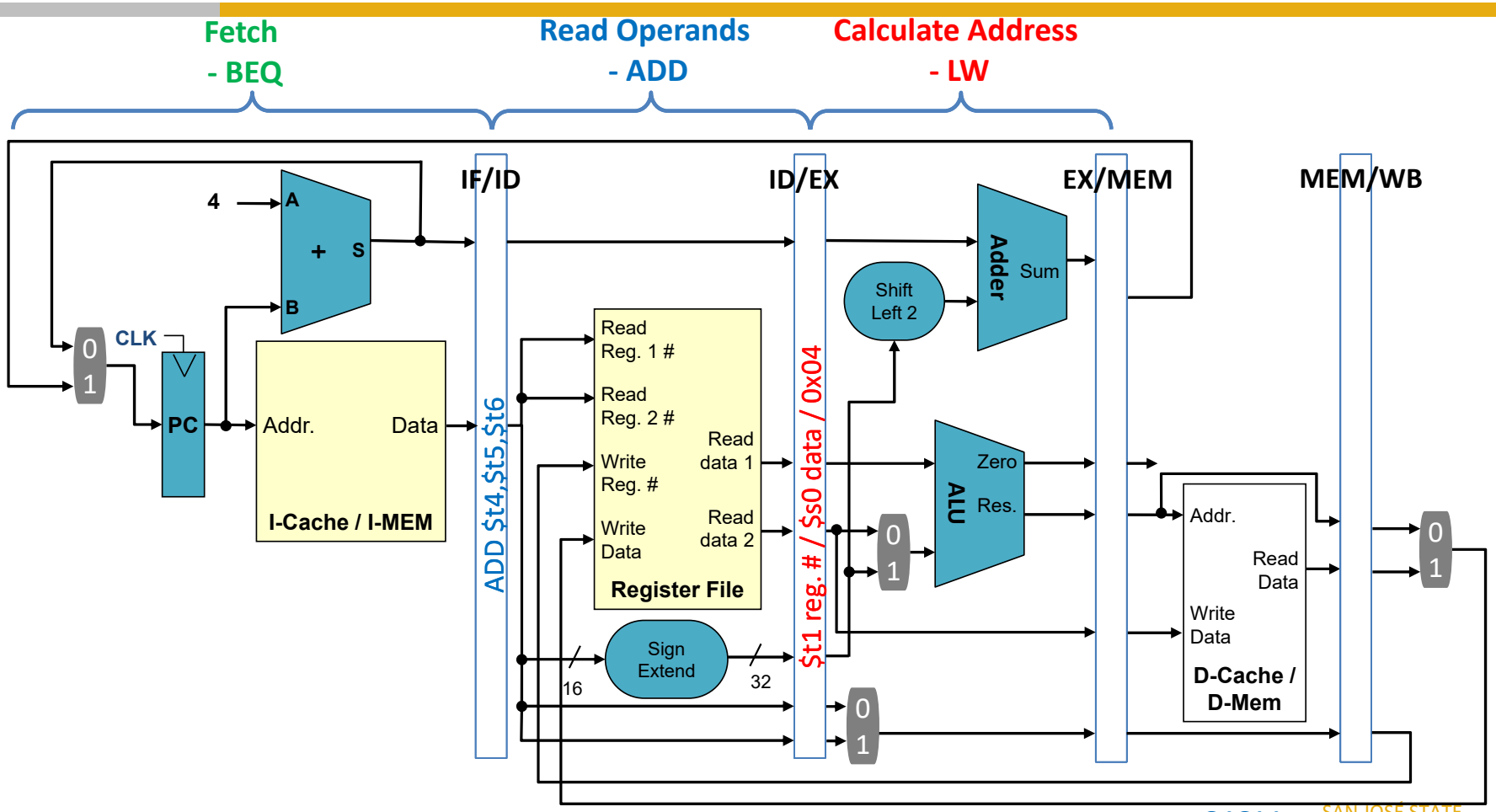
Multiple Instructions in 5-Stage Pipeline



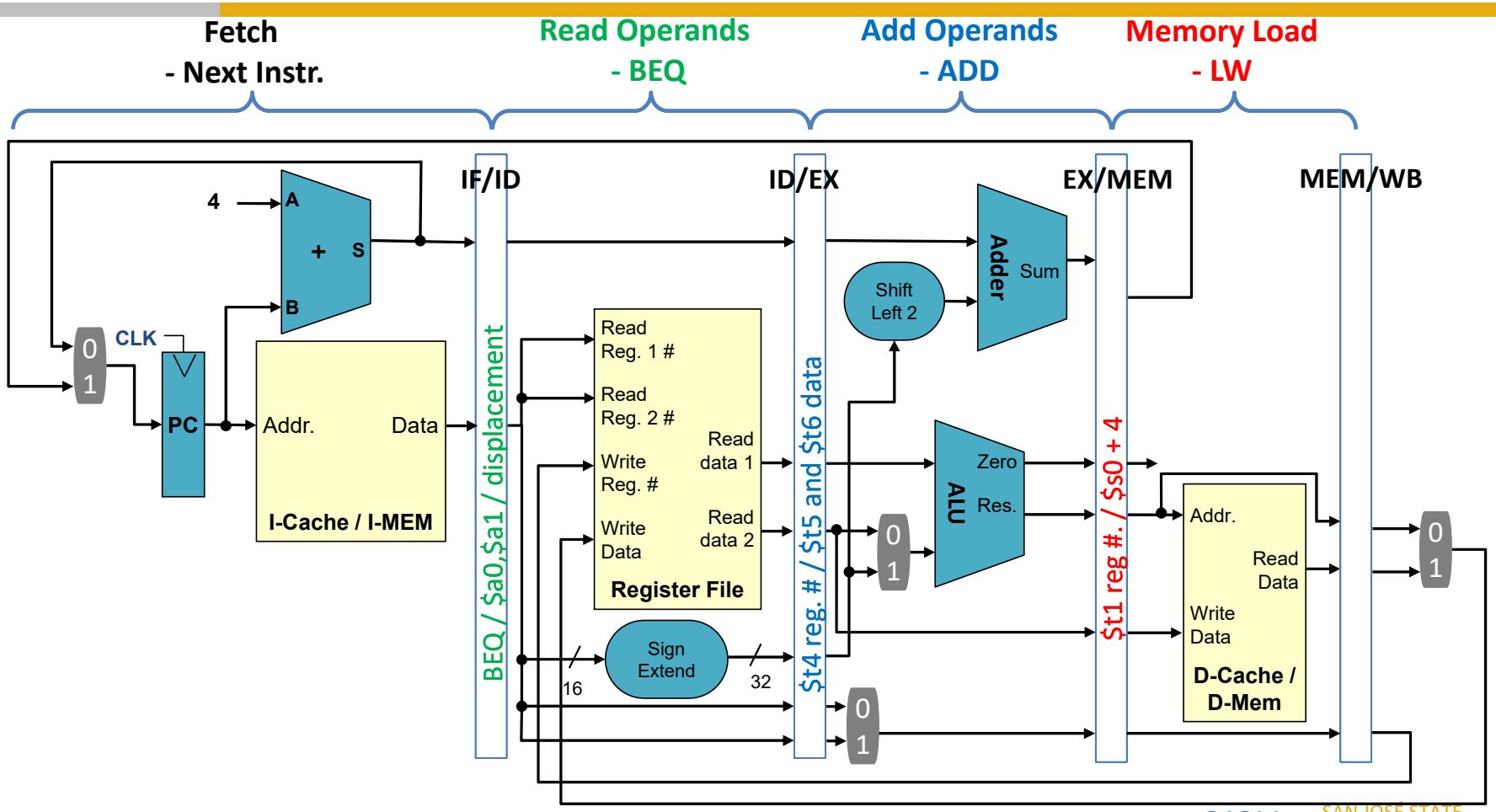
Multiple Instructions in 5-Stage Pipeline



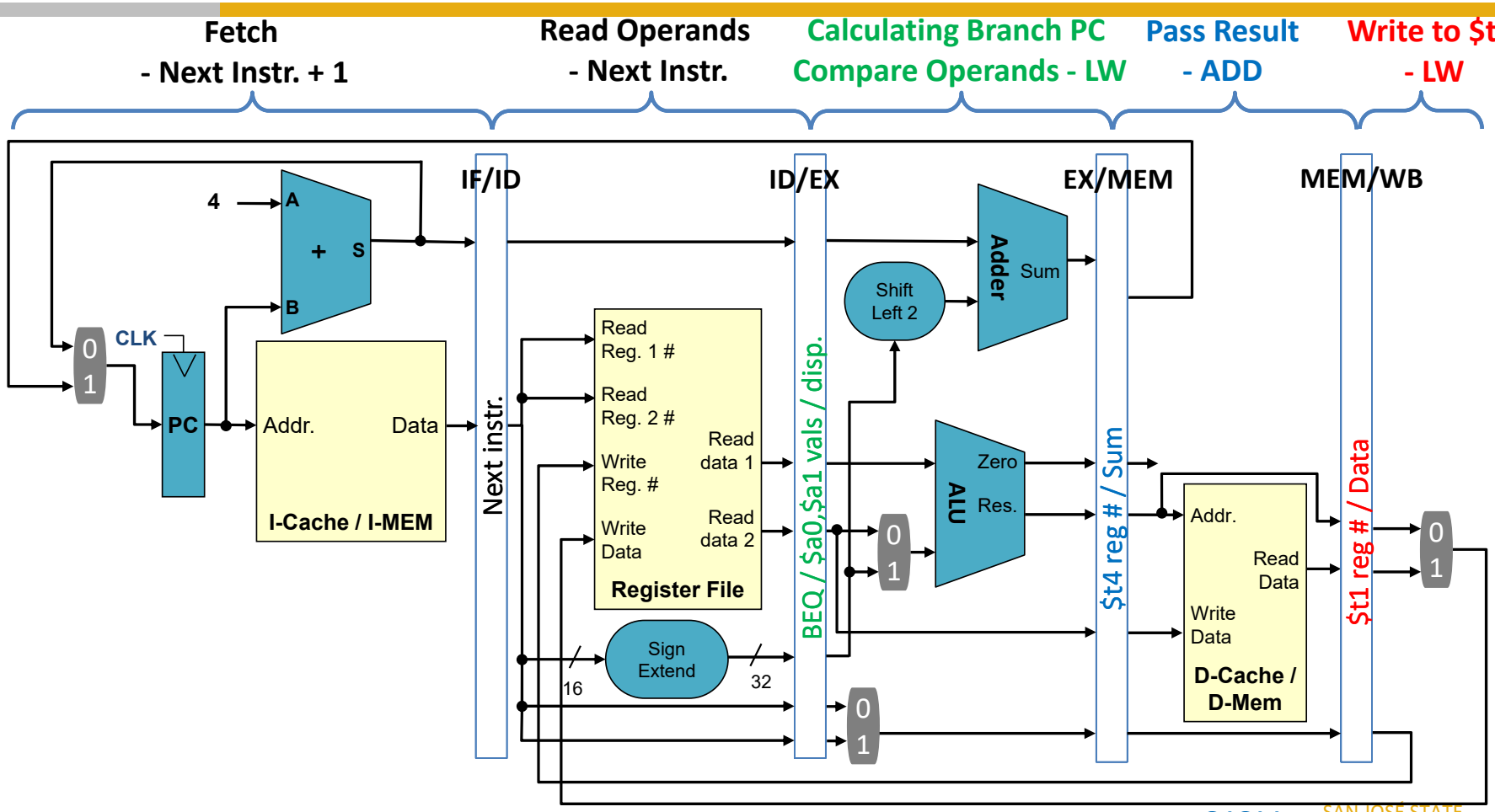
Multiple Instructions in 5-Stage Pipeline



Multiple Instructions in 5-Stage Pipeline

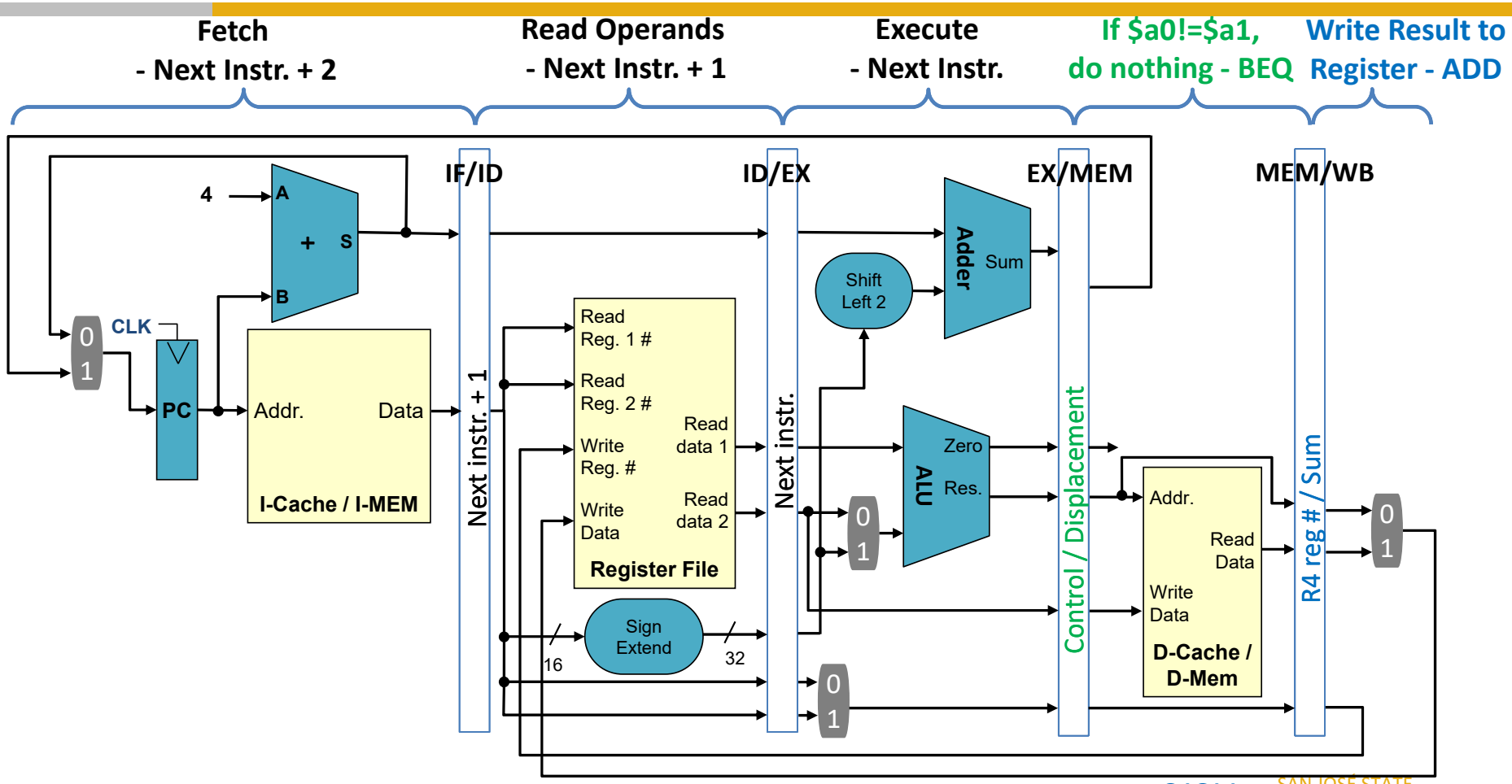


Multiple Instructions in 5-Stage Pipeline

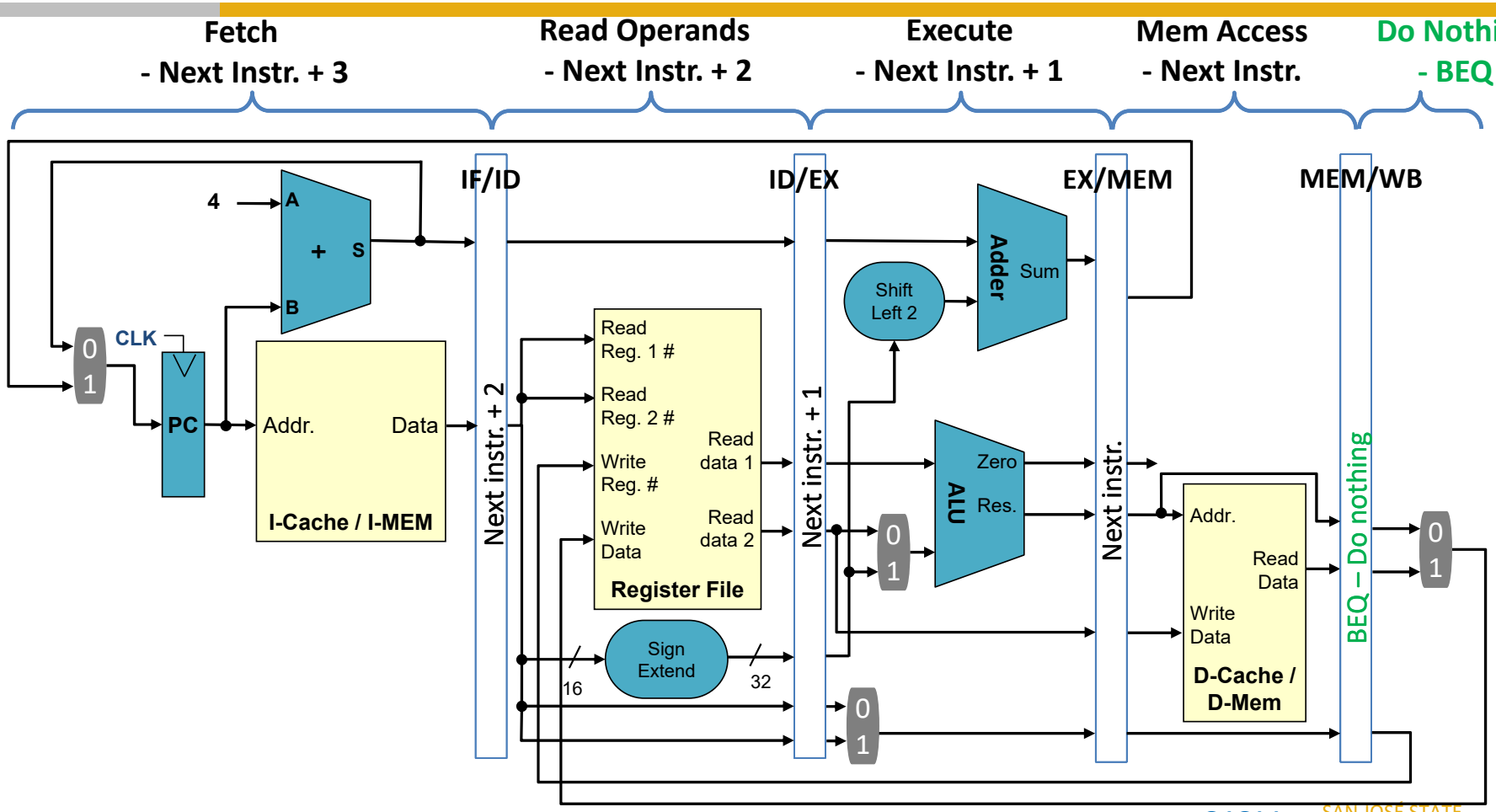


SJSU SAN JOSÉ STATE UNIVERSITY

Multiple Instructions in 5-Stage Pipeline



Multiple Instructions in 5-Stage Pipeline



Pipelined Timing

- Suppose we execute **N instructions** using a **K stage** datapath
- Total execution cycle: ***K+N-1 cycles***
  - K cycle for 1st inst + (N-1) cycles for remaining insts
  - Assume we keep the pipeline full

7 Instrs. =

11 clocks (5 + 7 – 1)

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11
LW	IF	ID	EXE	MEM	WB						
ADD		IF	ID	EXE	MEM	WB					
SUB			IF	ID	EXE	MEM	WB				
SW				IF	ID	EXE	MEM	WB			
AND					IF	ID	EXE	MEM	WB		
OR						IF	ID	EXE	MEM	WB	
XOR							IF	ID	EXE	MEM	WB

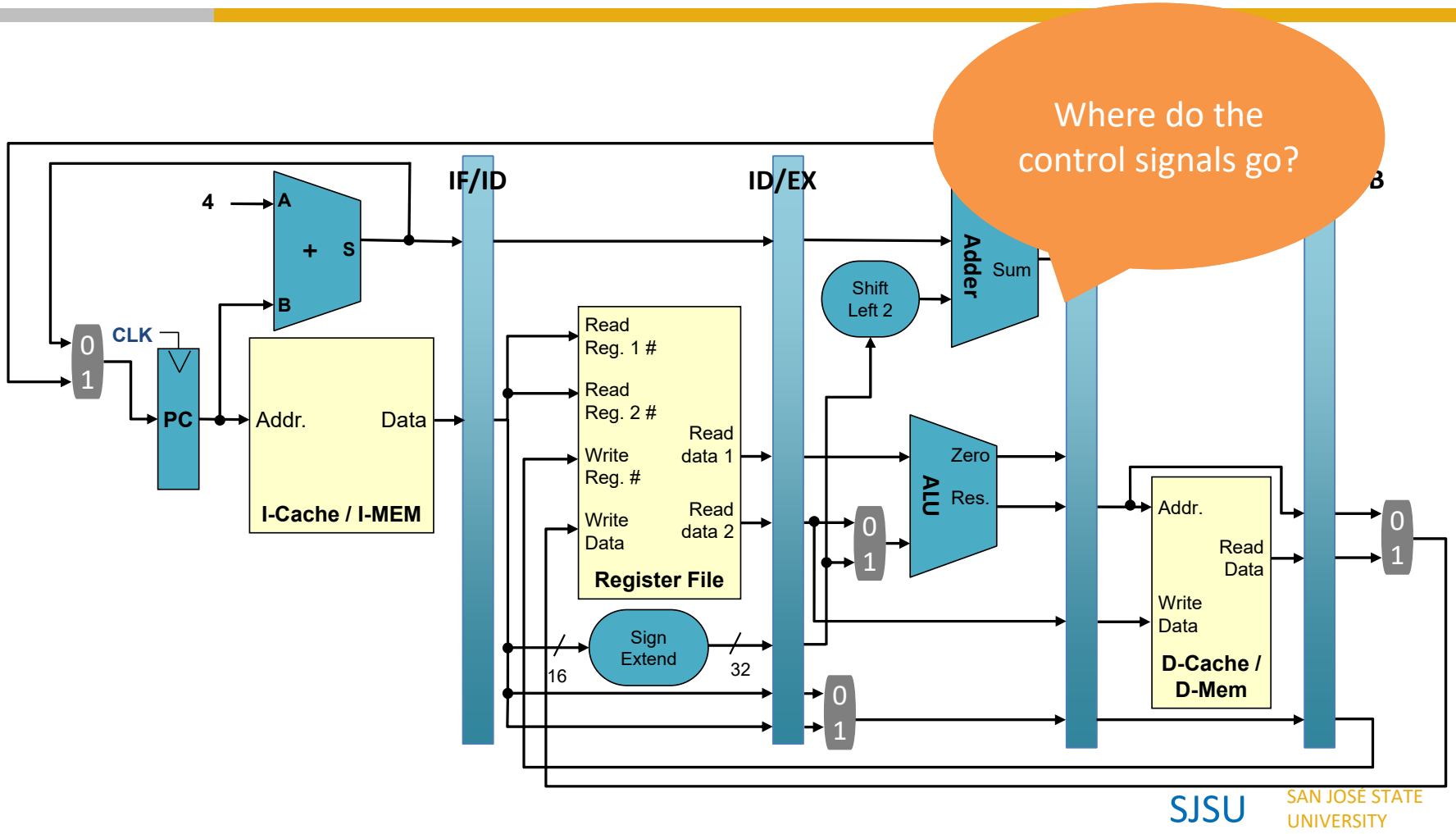
Throughput

- **Throughput (T) = # of instructions executed / time**
- **IPC = Instruction Per Cycle = 1 / CPI**
  - For a large number of instructions, the IPC of a pipelined processor is **1 instruction per clock cycle**
  - **Only when we keep the pipeline full of instructions**

Assume we execute **N instructions** using a **K stage** datapath

Pipeline IPC	N / (N+K-1)
Let n → ∞ (n = infinity)	1

5-Stage Pipelined CPU



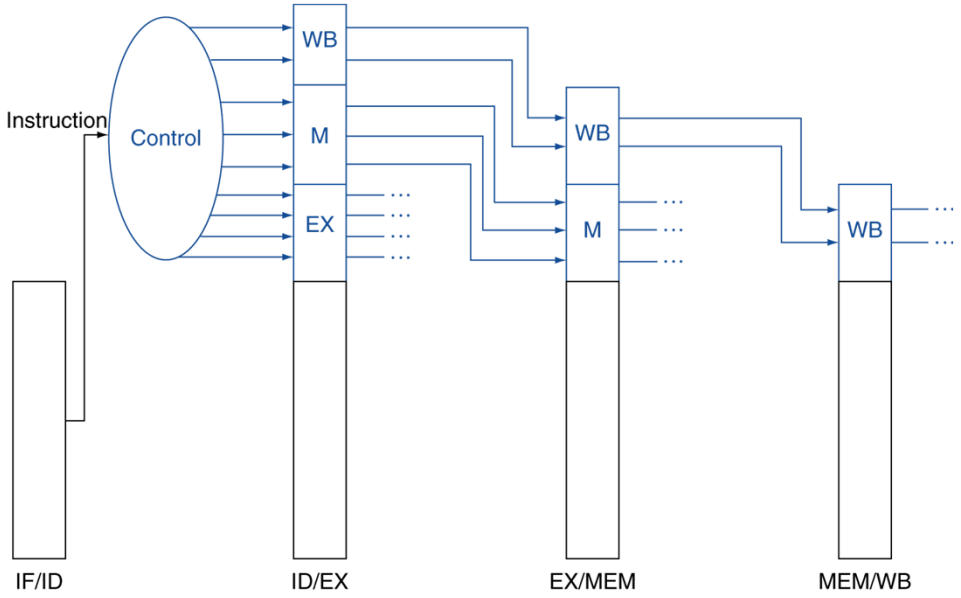
Pipelined Control

- **Using the same control signals as in Single-cycle CPU**
- **Grouped in three pipeline stages**

Instruction	EXE stage			MEM stage		WB stage	
	$alu\_op_{1:0}$	reg_dst	alu_src	branch	we_dm	dm2reg	we_reg
R-type	10	1	0	0	0	0	1
lw	00	0	1	0	0	1	1
sw	00	X	1	0	1	X	0
beq	01	X	0	1	0	X	0

Pipelined Control

- **Control signals are generated in ID stage and used by the following three stages**
- **Pipeline registers are extended to also include control information**



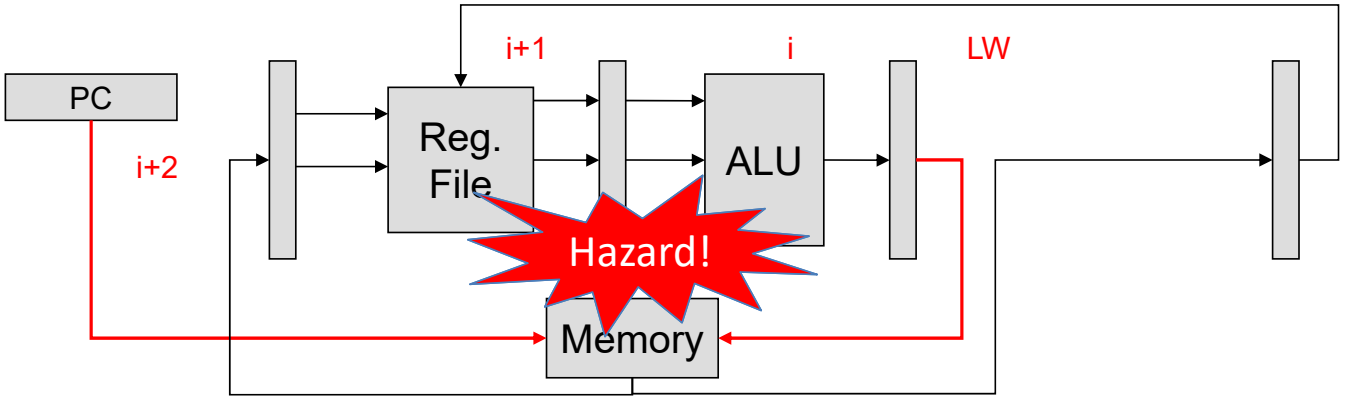
Hazards

- **Any sequence of instructions that prevent full pipeline utilization**
  - Often causes the pipeline to **stall** for one or more instructions
- **Structural Hazards**
  - HW organization cannot support certain combinations of instructions being overlapped
- **Data Hazards**
  - Data dependencies
- **Control Hazards**
  - Branches & changes to PC in the pipeline
  - If branch is determined to be taken later in the pipeline, flush (delete) the instructions in the pipeline that shouldn't be executed

Structural Hazards

- **Combinations of instructions that cannot be overlapped in the given order due to HW constraints**
  - Often due to lack of HW resources

- **Example: A single memory rather than separate I & D Memories**
  - Structural hazard whenever an instruction needs to perform a data access (i.e. 'lw' or 'sw')



Data Hazards

- **Read-After-Write (RAW)**
  - A following instruction reads a result from a previous instruction
- Write-After-Read (WAR)
- Write-After-Write (WAW)

- **RAW Example**
  - LW $t1,4($s0)
  - ADD $t5,$t1,$t4

Initial Conditions:

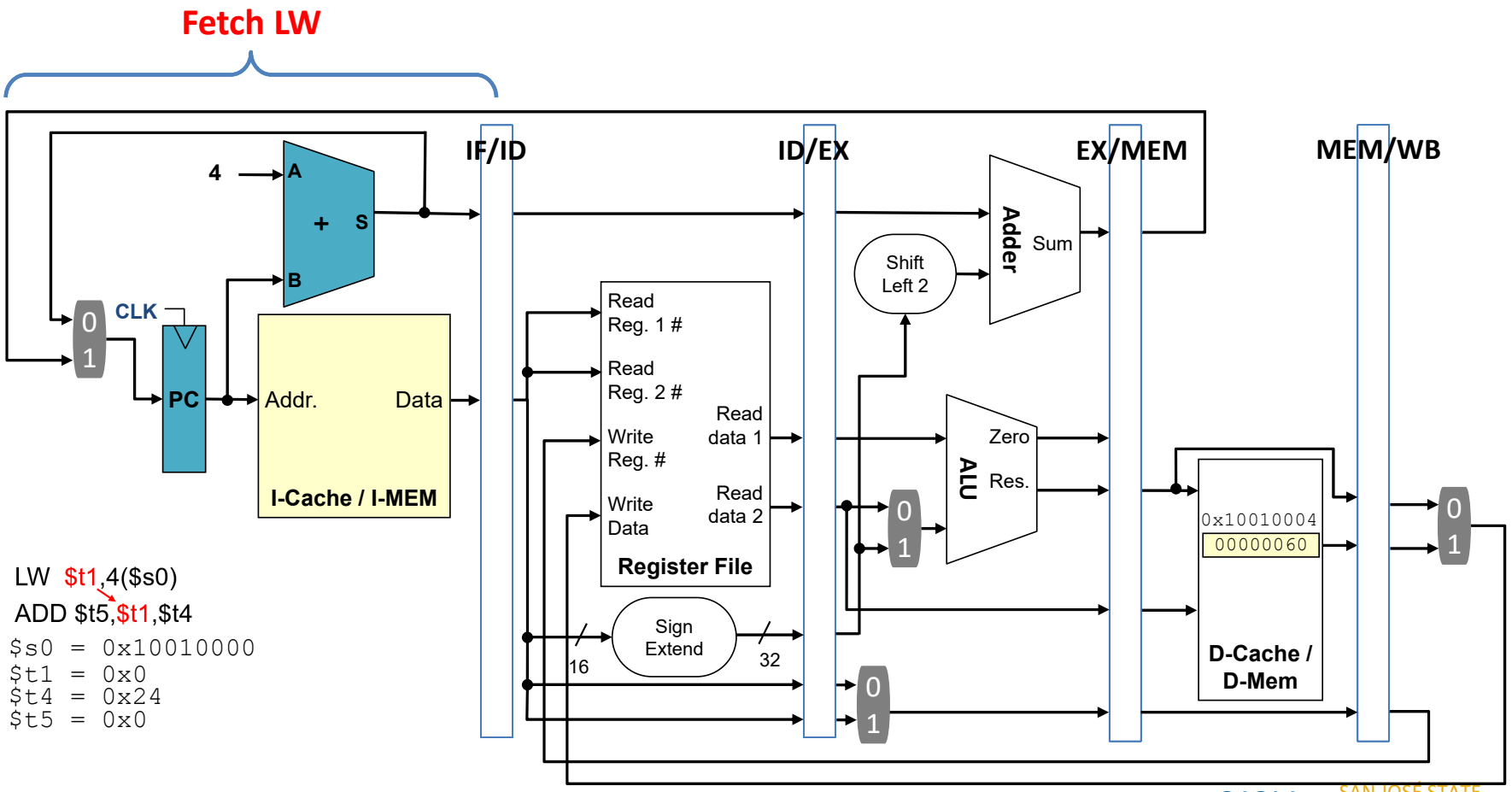
```
$s0 = 0x10010000
$t1 = 0x0
$t4 = 0x24
$t5 = 0x0
```

00000060	0x10010004
12345678	0x10010000

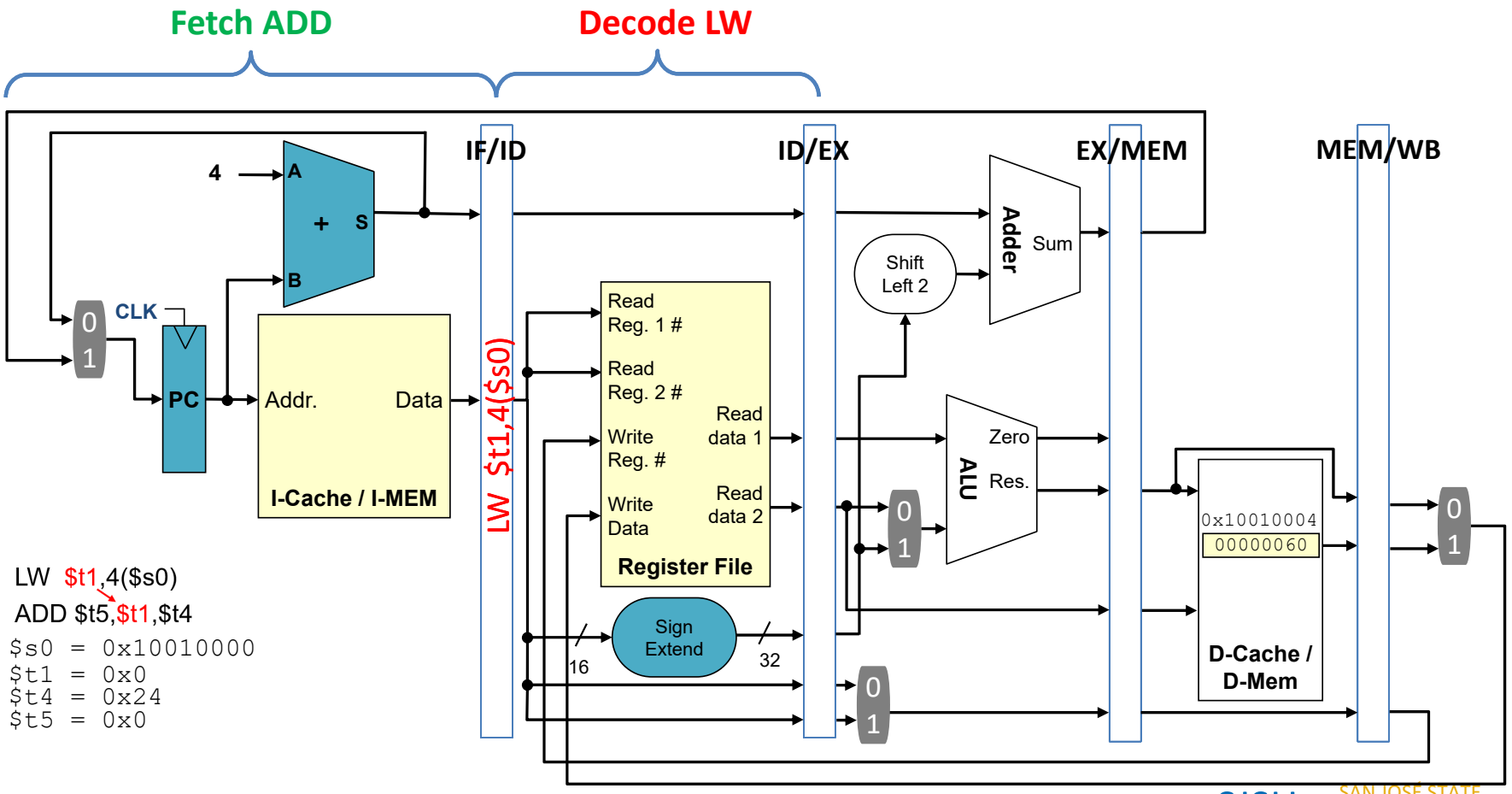
After execution values should be:

```
$s0 = 0x10010000
$t1 = 0x60
$t4 = 0x24
$t5 = 0x84
```

Data Hazards Example



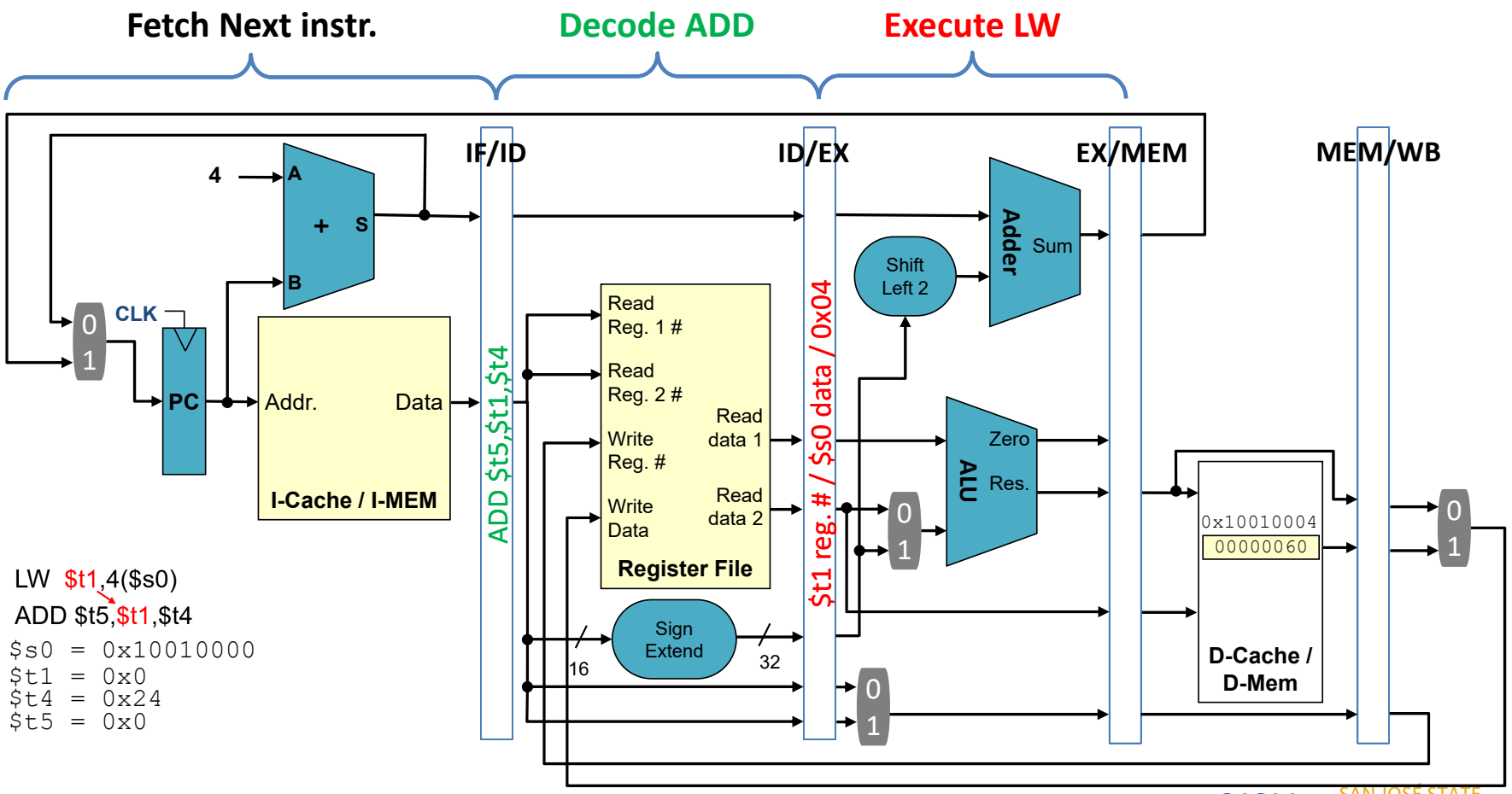
Data Hazards Example



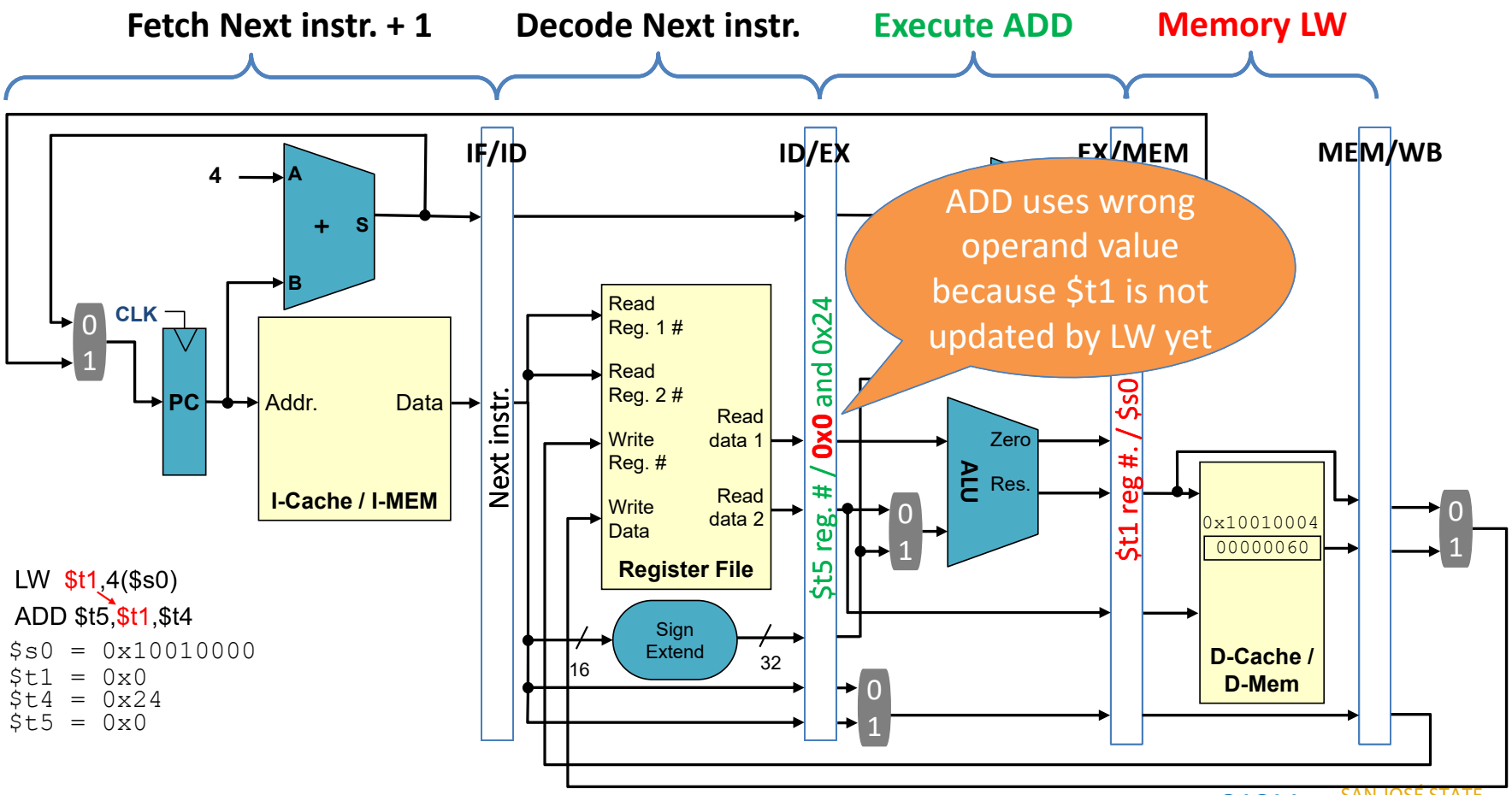
LW $t1,4($s0)
ADD $t5,$t1,$t4

```
$s0 = 0x10010000
$t1 = 0x0
$t4 = 0x24
$t5 = 0x0
```

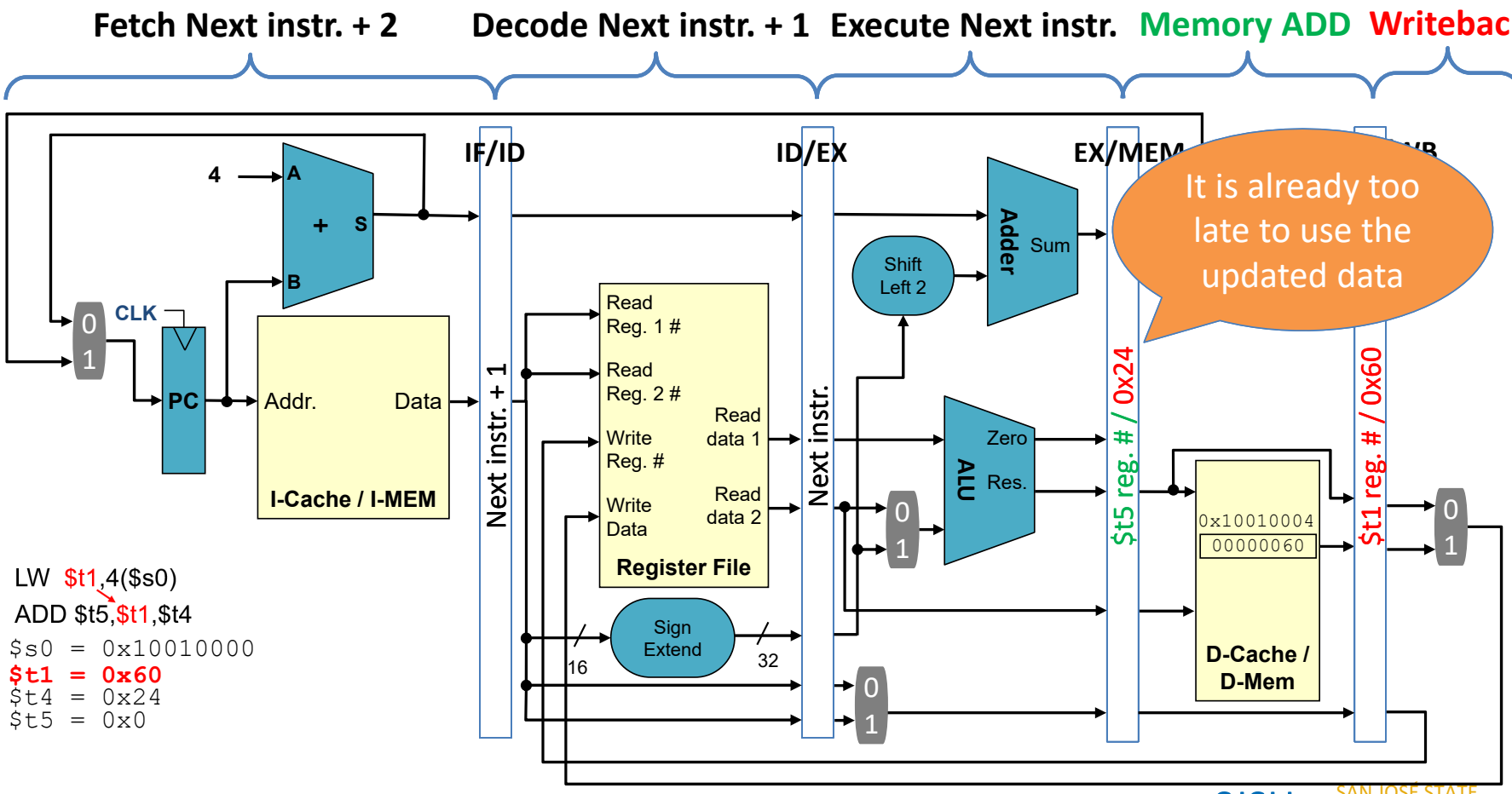
Data Hazards Example



Data Hazards Example



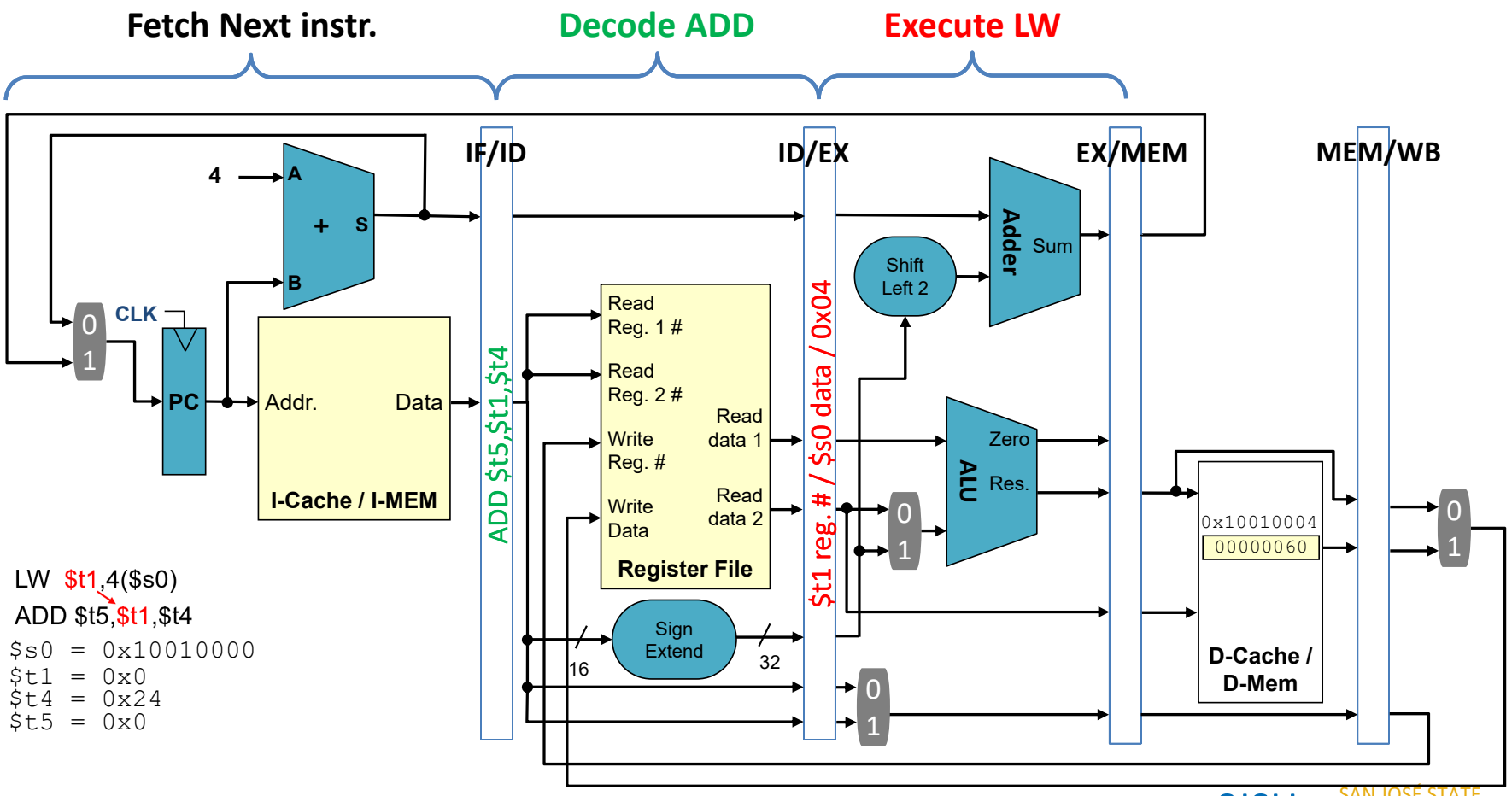
Data Hazards Example



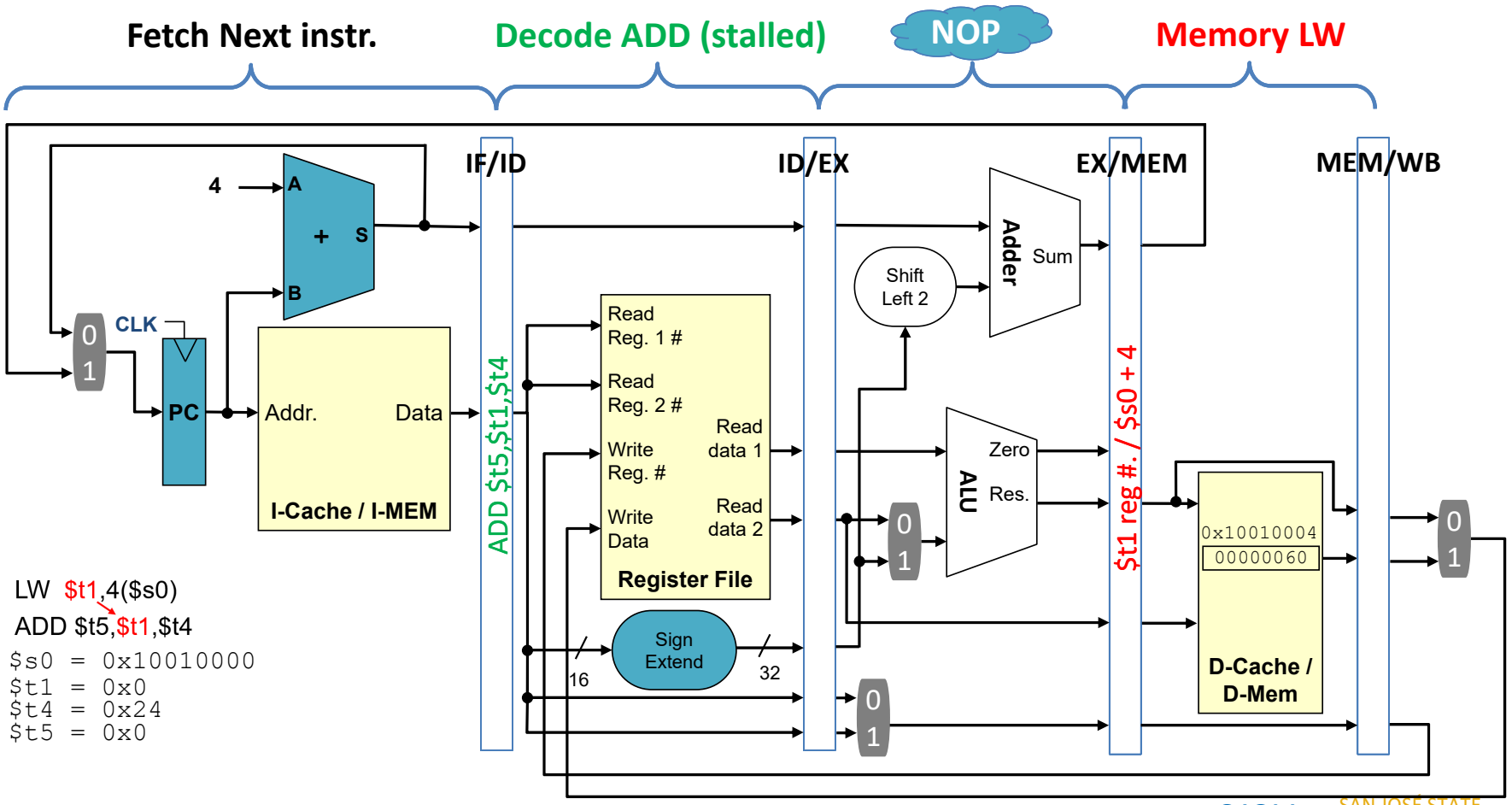
Cause of Data Hazards

- **Stalling the pipeline to prevent error:**
  - All instructions in front of the stalled instruction can continue
  - All instructions behind the stalled instruction must also stall
- **Stalling inserts "bubbles" / nops (no-operations) into the pipeline**
  - A "nop" is an actual instruction in the MIPS ISA that does NOTHING

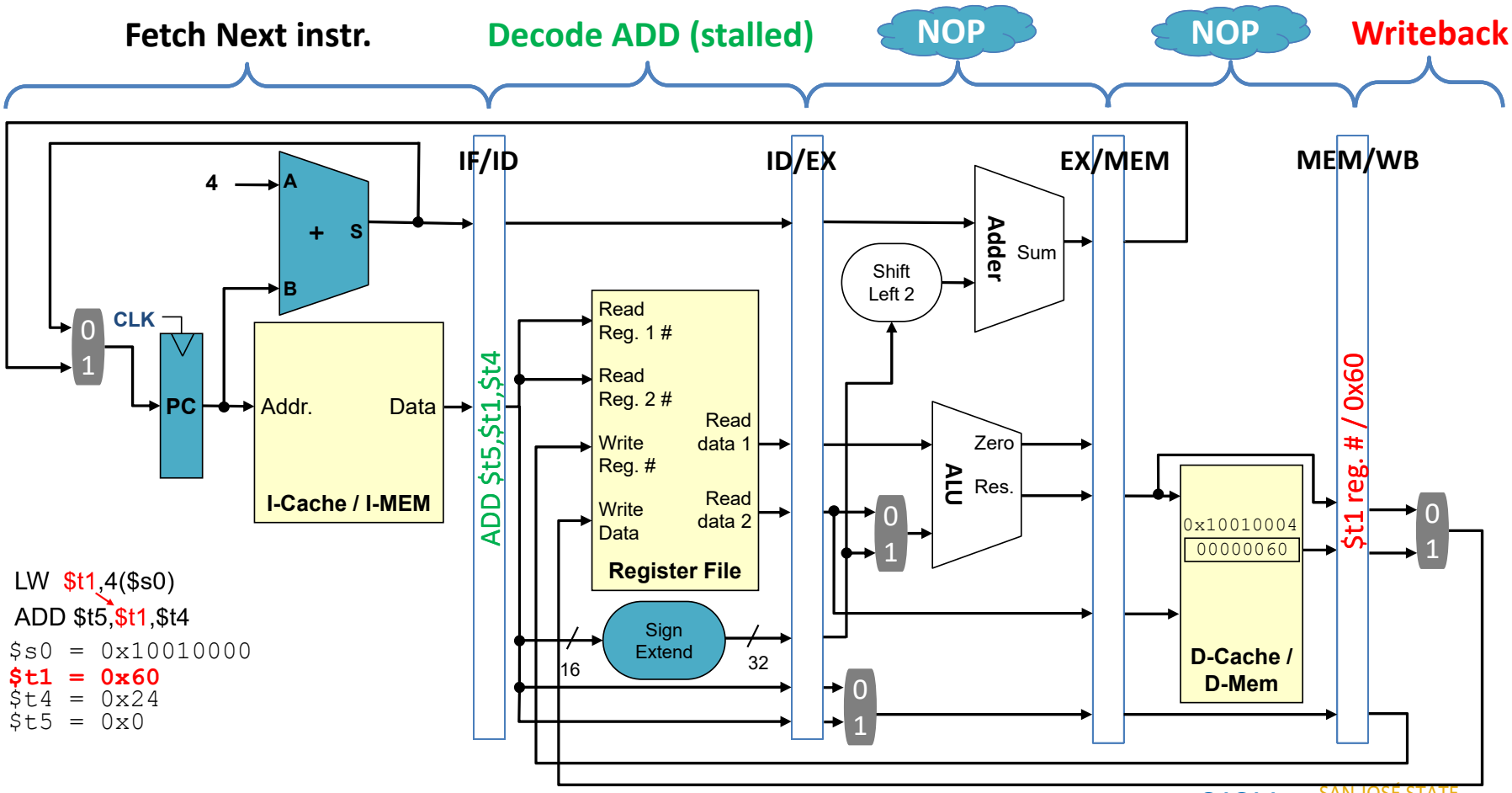
Pipeline Stall for Data Hazards



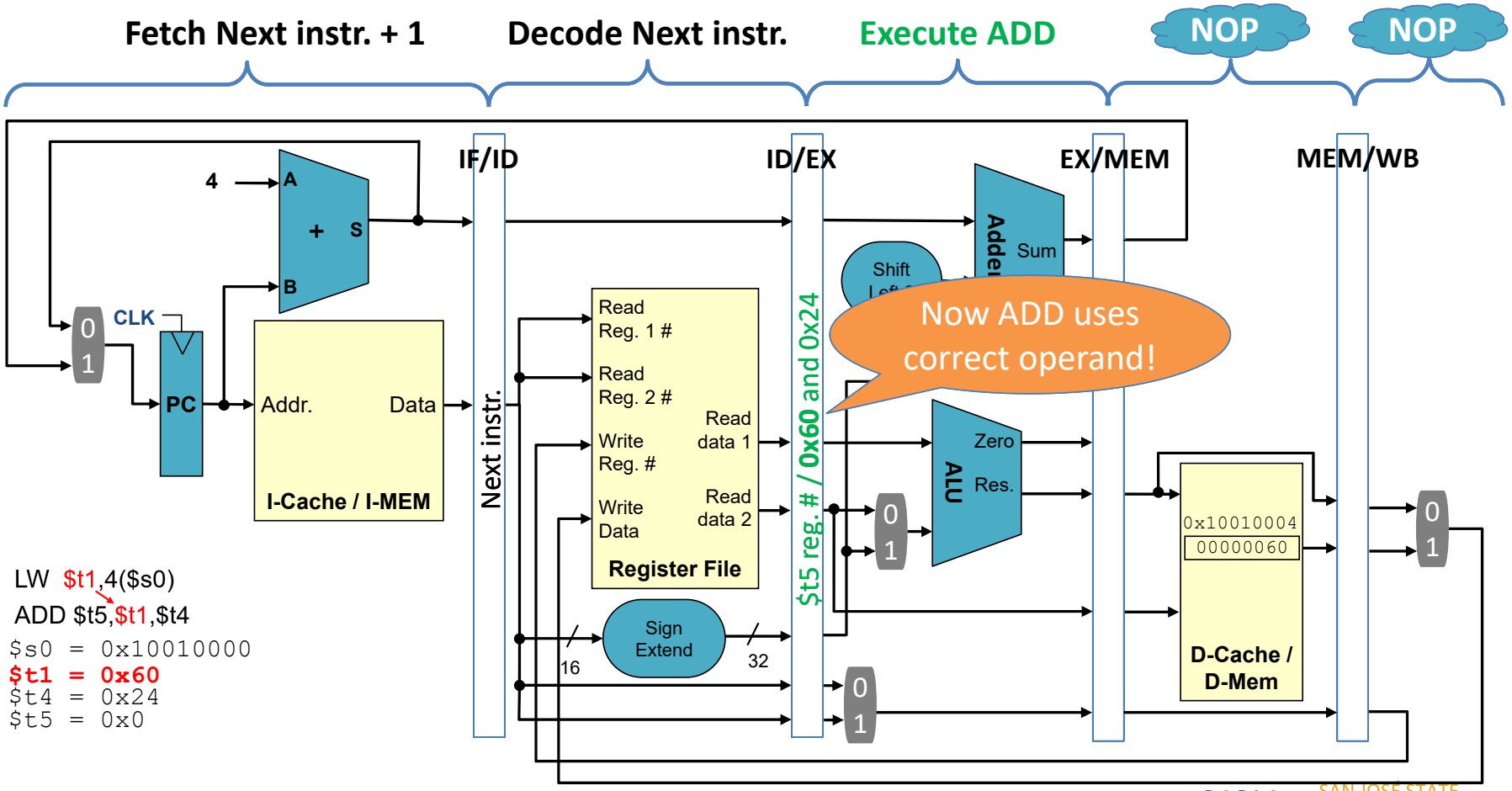
Pipeline Stall for Data Hazards



Pipeline Stall for Data Hazards



Pipeline Stall for Data Hazards



Data Hazards Example

- **Using stalls to handle dependencies**

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11
LW	IF										
ADD											
Next instr.											
Next+1											
Next+2											
Next+3											
Next+4											

Data Hazards Example

- **Using stalls to handle dependencies**

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11
LW	IF	ID									
ADD		IF									
Next instr.											
Next+1											
Next+2											
Next+3											
Next+4											

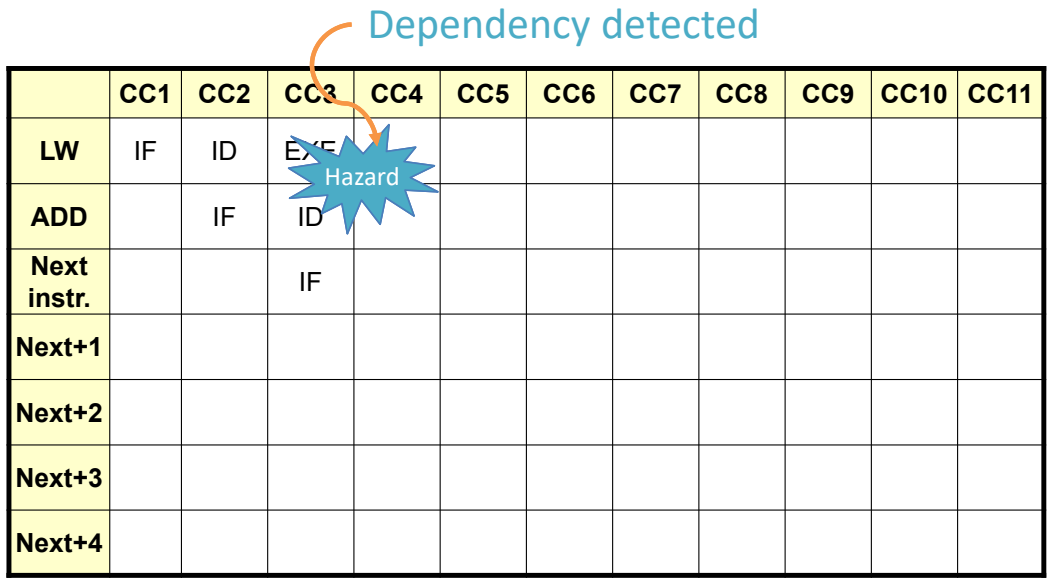
Data Hazards Example

- **Using stalls to handle dependencies**

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11
LW	IF	ID	EXE								
ADD		IF	ID								
Next instr.			IF								
Next+1											
Next+2											
Next+3											
Next+4											

Data Hazards Example

- **Using stalls to handle dependencies**



	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11
LW	IF	ID	EX								
ADD		IF	ID								
Next instr.			IF								
Next+1											
Next+2											
Next+3											
Next+4											

Data Hazards Example

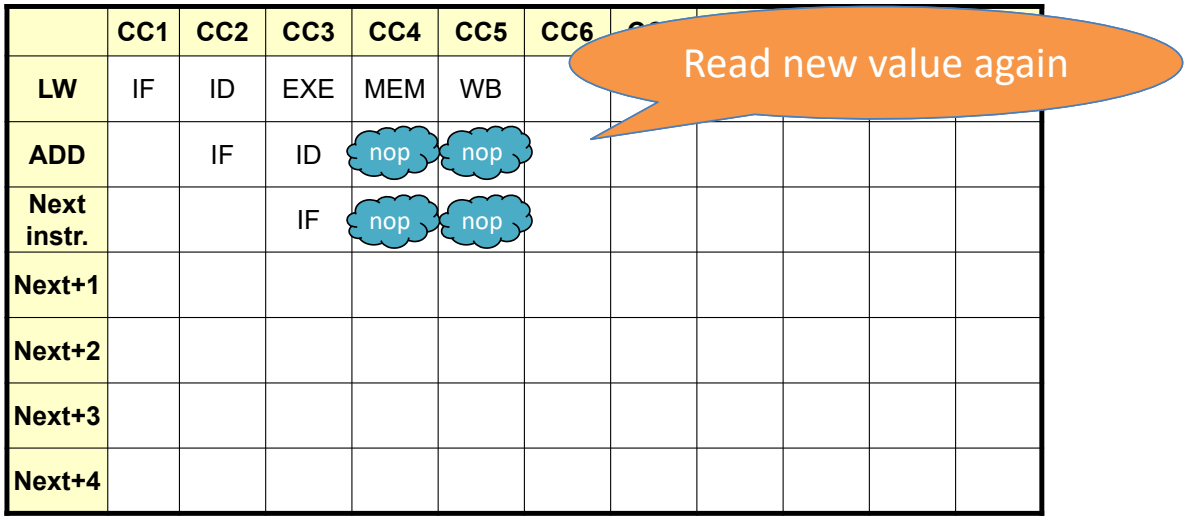
- **Using stalls to handle dependencies**

	CC1	CC2	CC3	CC4						CC10	CC11
LW	IF	ID	EXE	MEM							
ADD		IF	ID	nop							
Next instr.			IF	nop							
Next+1											
Next+2											
Next+3											
Next+4											

Stays in ID stage

Data Hazards Example

- **Using stalls to handle dependencies**



Data Hazards Example

- **Using stalls to handle dependencies**

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11
LW	IF	ID	EXE	MEM	WB						
ADD		IF	ID	nop	nop	EXE					
Next instr.			IF	nop	nop	ID					
Next+1						IF					
Next+2											
Next+3											
Next+4											

Data Hazards Example

- **Using stalls to handle dependencies**

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13
LW	IF	ID	EXE	MEM	WB								
ADD		IF	ID	nop	nop	EXE	MEM	WB					
Next instr.			IF	nop	nop	ID	EXE	MEM	WB				
Next+1						IF	ID	EXE	MEM	WB			
Next+2							IF	ID	EXE	MEM	WB		
Next+3								IF	ID	EXE	MEM	WB	
Next+4									IF	ID	EXE	MEM	WB

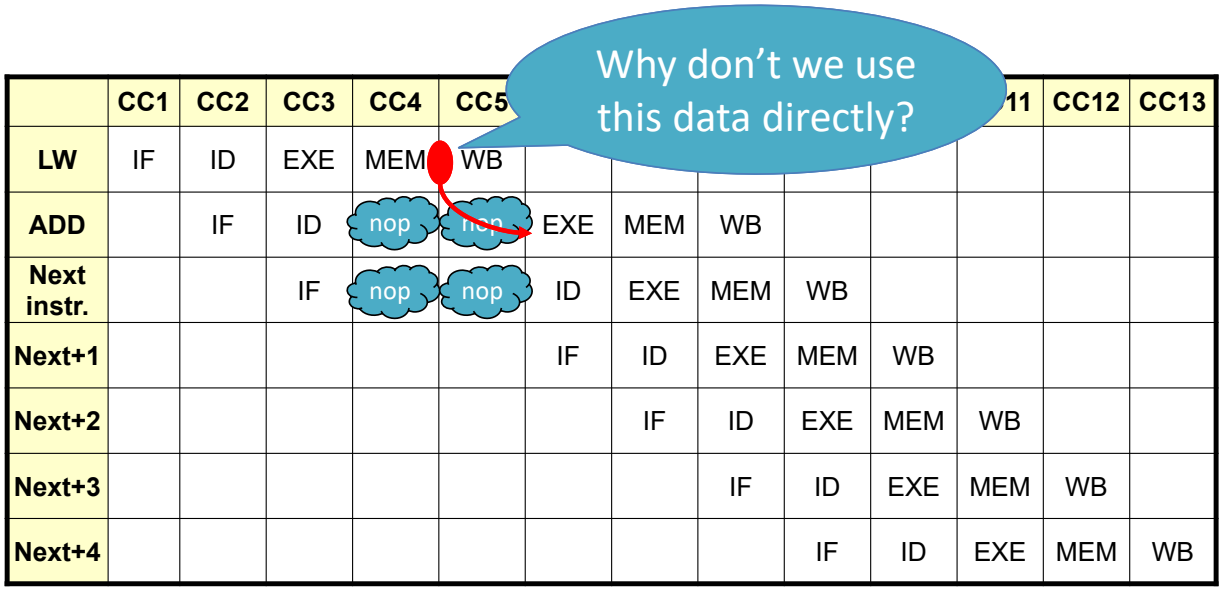
Throughput = 7 insts/13 cycles = 0.53

vs.

Throughput without hazard = 7 insts/11 cycles = 0.63

Pipeline Stall for Data Hazards

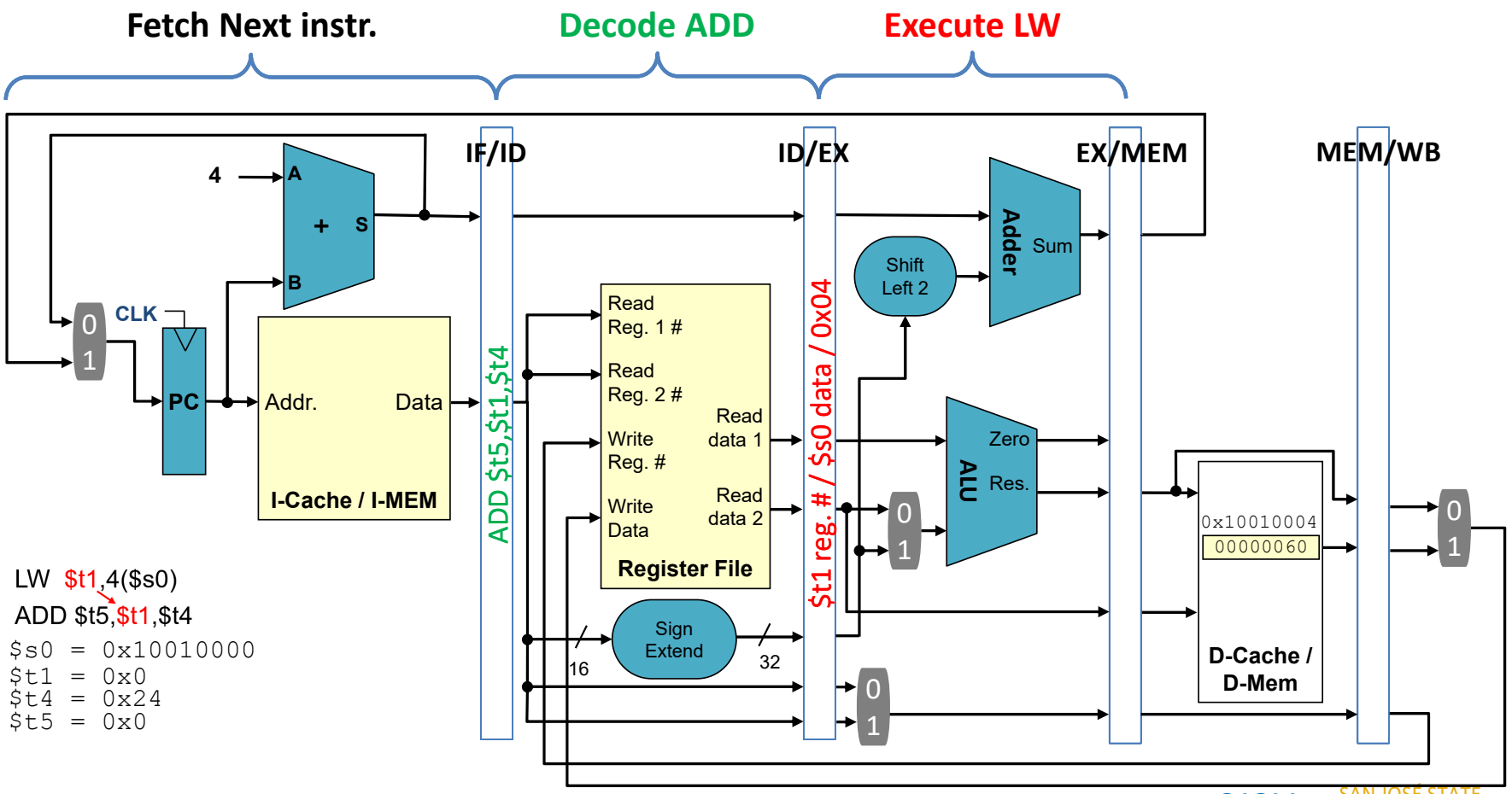
- **ADD can read the updated register value from the register file in the same cycle as LW writes the loaded data to register**
- **Do we really need to wait until LW updates the register file?**



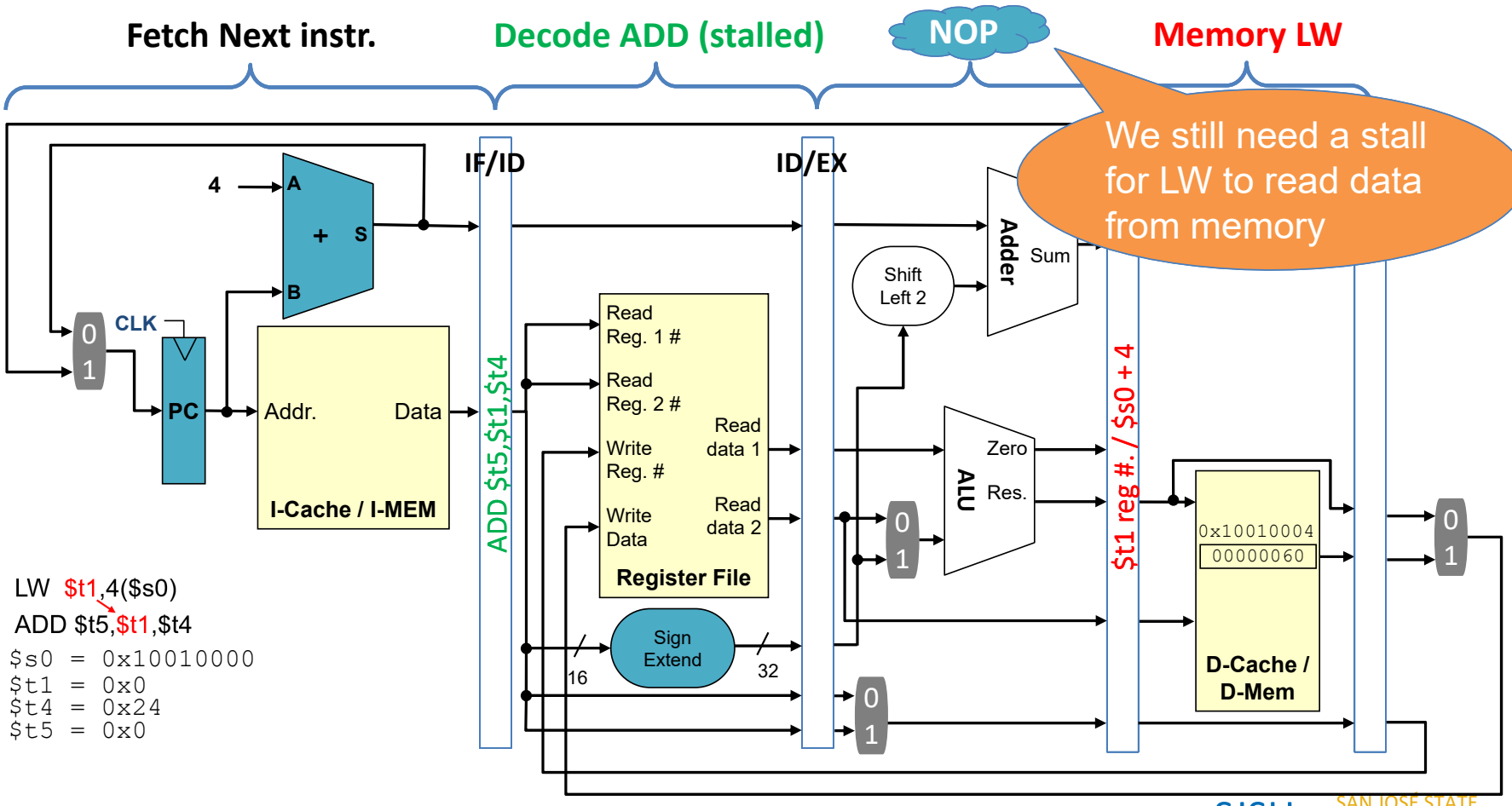
Data Hazards Solutions

- **Compiler**
  - Reorder Code
- **Hardware**
  - Use forwarding / bypassing
- **Data forwarding**
  - Take results still in the pipeline (not yet written back to a register) and pass them to dependent instructions
  - Forwarding Path
    - Load instruction: from WB to EXE
    - R-type instructions: from MEM to EXE

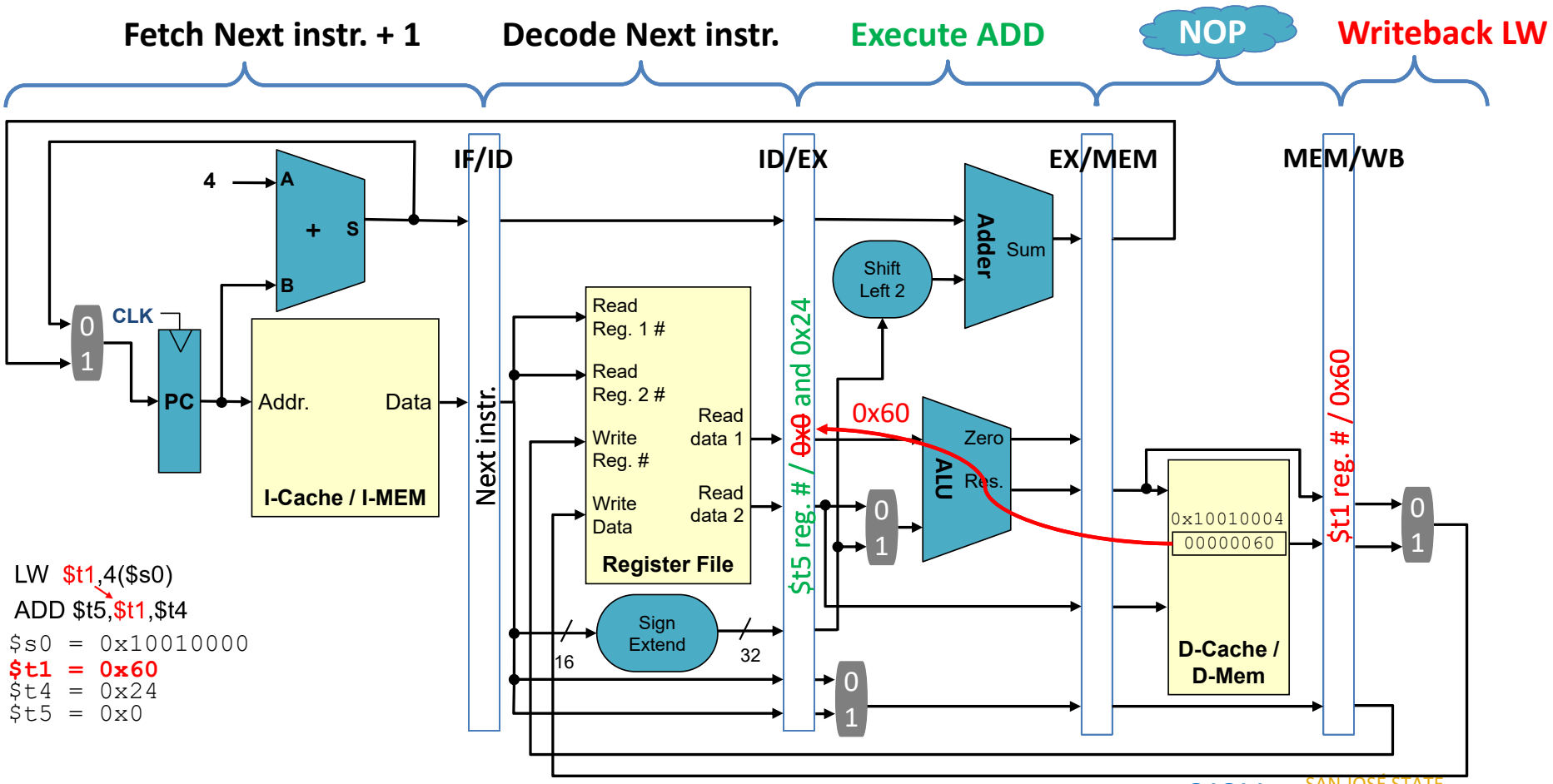
Data Forwarding



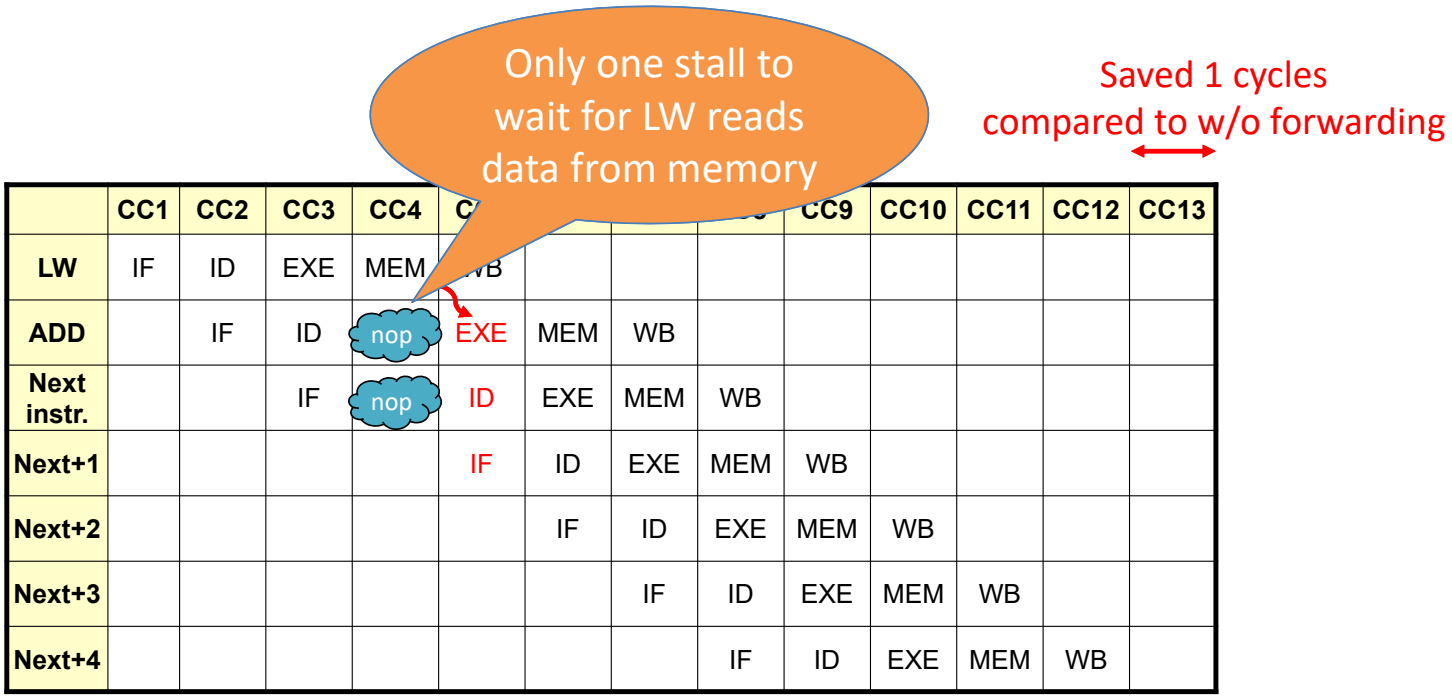
Data Forwarding



Data Forwarding



Data Forwarding



	CC1	CC2	CC3	CC4	C				CC9	CC10	CC11	CC12	CC13
LW	IF	ID	EXE	MEM	WB								
ADD		IF	ID	nop	EXE	MEM	WB						
Next instr.			IF	nop	ID	EXE	MEM	WB					
Next+1					IF	ID	EXE	MEM	WB				
Next+2						IF	ID	EXE	MEM	WB			
Next+3							IF	ID	EXE	MEM	WB		
Next+4								IF	ID	EXE	MEM	WB	

Conclusion Time

What is the main cause for structural hazards?

- **Lacking structure**

What is the main cause for data hazards?

- **Lacking data**

SAN JOSÉ STATE UNIVERSITY *powering* SILICON VALLEY