RISC Design:

Pipeline Hazards

Virendra Singh

Associate Professor

Computer **A**rchitecture and **D**ependable **S**ystems **L**ab

Department of Electrical Engineering

Indian Institute of Technology Bombay

http://www.ee.iitb.ac.in/~viren/

E-mail: viren@ee.iitb.ac.in

CP-226: Computer Architecture

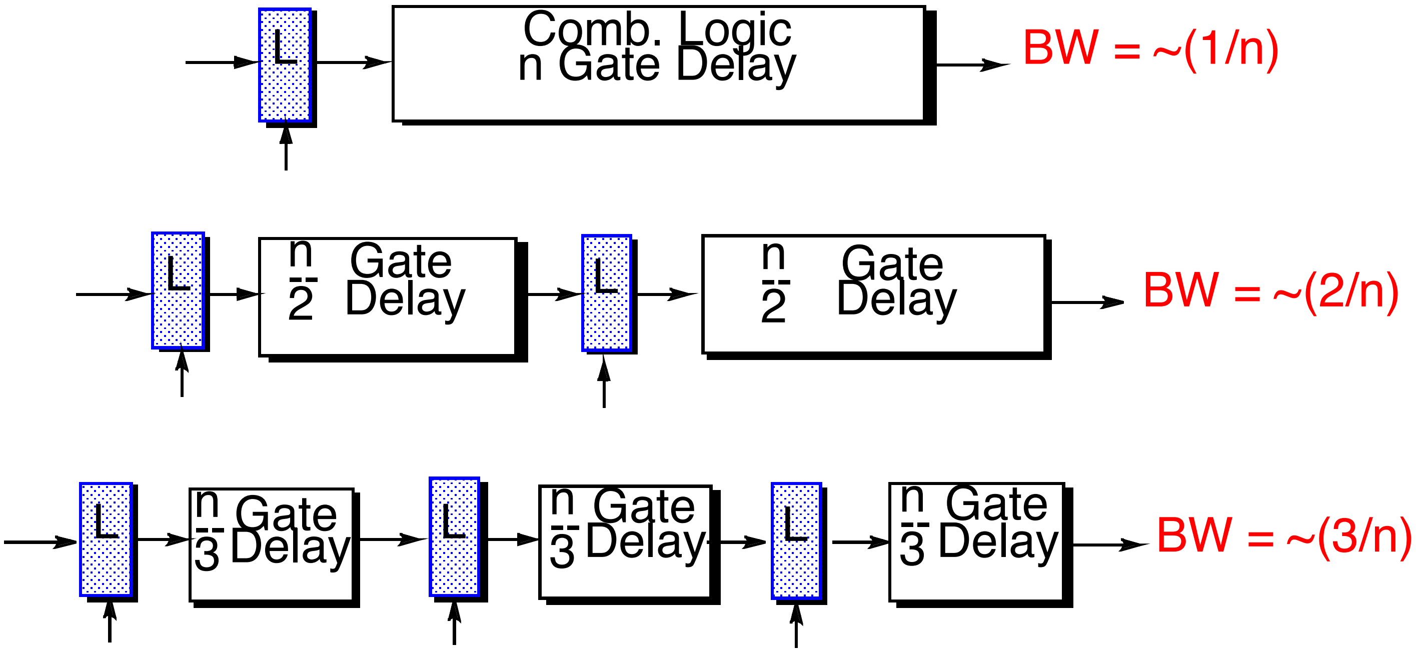
Lecture 11 (22 Feb 2013)

CADSL

Pipelining in a Computer

- Divide datapath into nearly equal tasks, to be performed serially and requiring non-overlapping resources.
- Insert registers at task boundaries in the datapath; registers pass the output data from one task as input data to the next task.
- Synchronize tasks with a clock having a cycle time that just exceeds the time required by the longest task.
- Break each instruction down into a fixed number of tasks so that instructions can be executed in a staggered fashion.

Ideal Pipelining

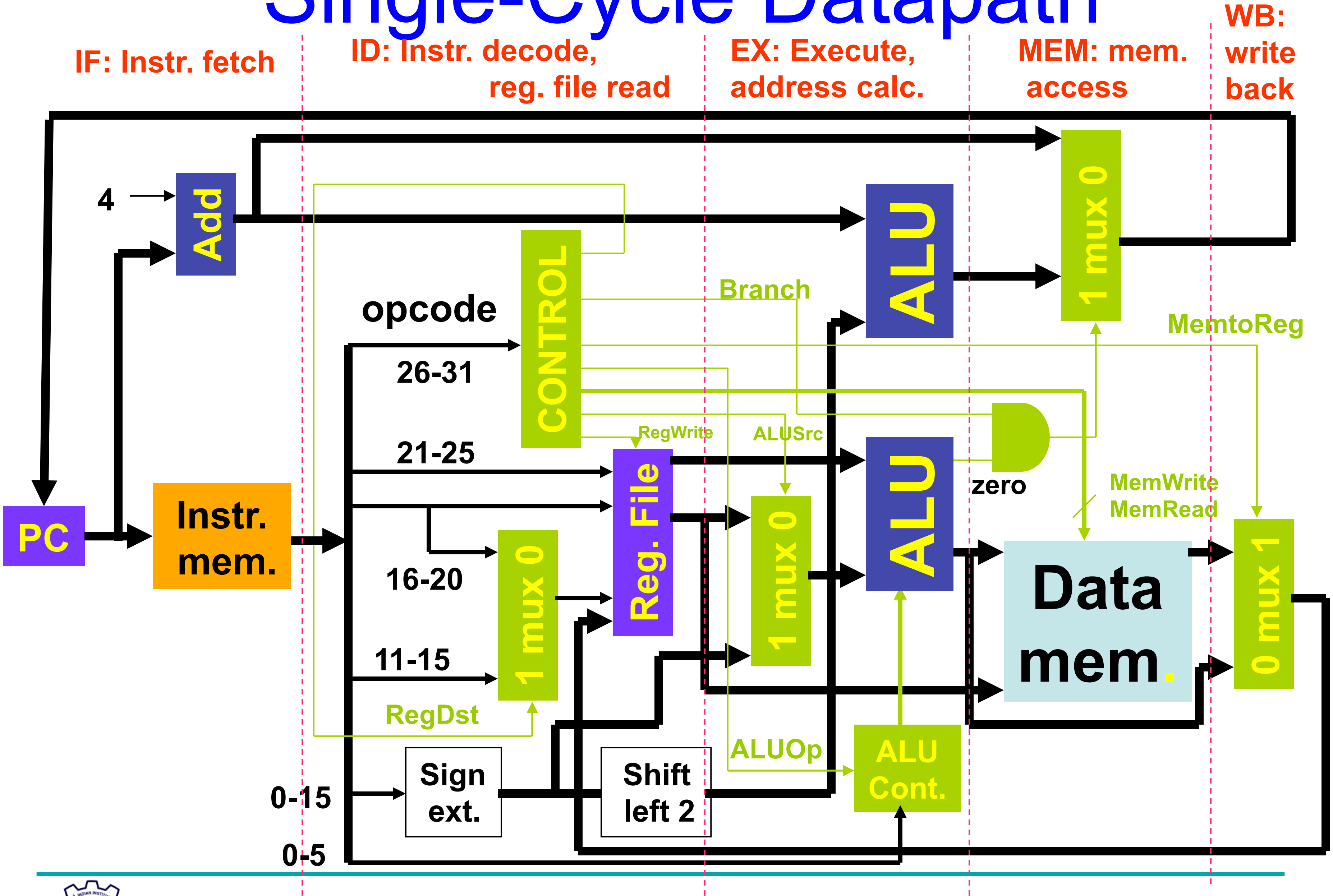


- Bandwidth increases linearly with pipeline depth
- Latency increases by latch delays

Pipelining Idealisms

- Uniform subcomputations
  - Can pipeline into stages with equal delay
  - Balance pipeline stages
- Identical computations
  - Can fill pipeline with identical work
  - Unify instruction types
- Independent computations
  - No relationships between work units
- Are these practical?
  - No, but can get close enough to get significant speedup

Single-Cycle Datapath

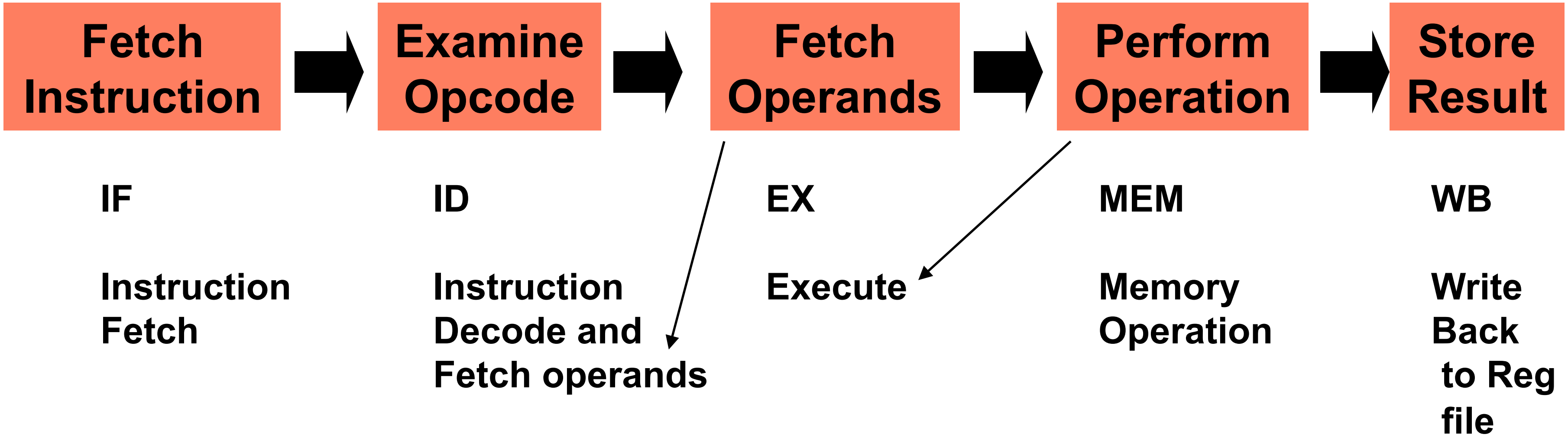


Pipelined Datapath

Instruction class	Instr. fetch (IF)	Instr. Decode (also reg. file read) (ID)	Execution (ALU Operation) (EX)	Data access (MEM)	Write Back (Reg. file write) (WB)	Total time
lw	2ns	~~1ns~~ 2ns	2ns	2ns	~~1ns~~ 2ns	10ns
sw	2ns	~~1ns~~ 2ns	2ns	2ns	~~1ns~~ 2ns	10ns
R-format: add, sub, and, or, slt	2ns	~~1ns~~ 2ns	2ns	2ns	~~1ns~~ 2ns	10ns
B-format: beq	2ns	~~1ns~~ 2ns	2ns	2ns	~~1ns~~ 2ns	10ns

No operation on data; idle time inserted to equalize instruction lengths.

Pipelining of RISC Instructions



Fetch Instruction

Examine Opcode

Fetch Operands

Perform Operation

Store Result

IF

Instruction Fetch

ID

Instruction Decode and Fetch operands

EX

Execute

MEM

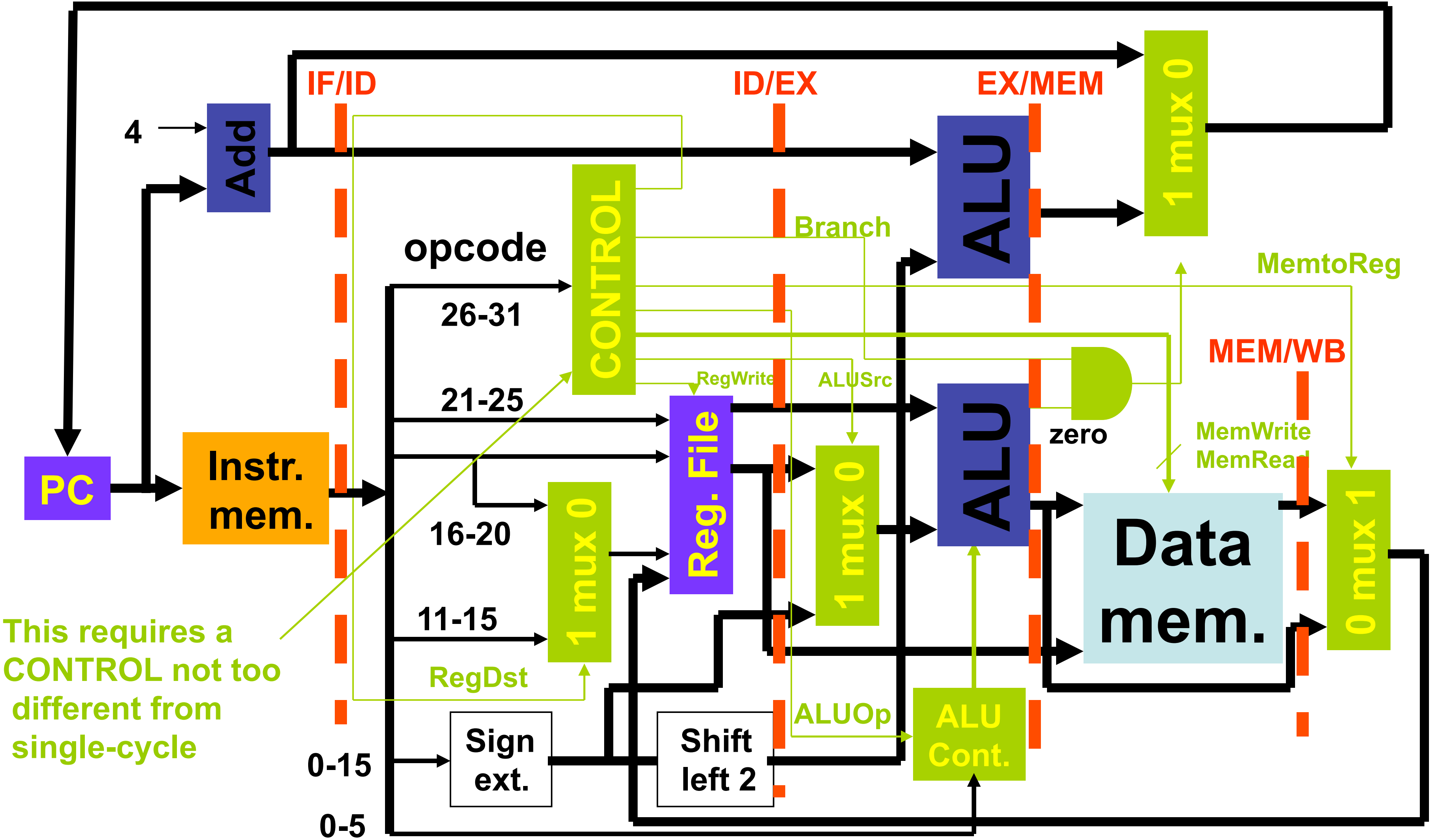
Memory Operation

WB

Write Back to Reg file

Although an instruction takes five clock cycles, one instruction is completed every cycle.

Pipeline Registers



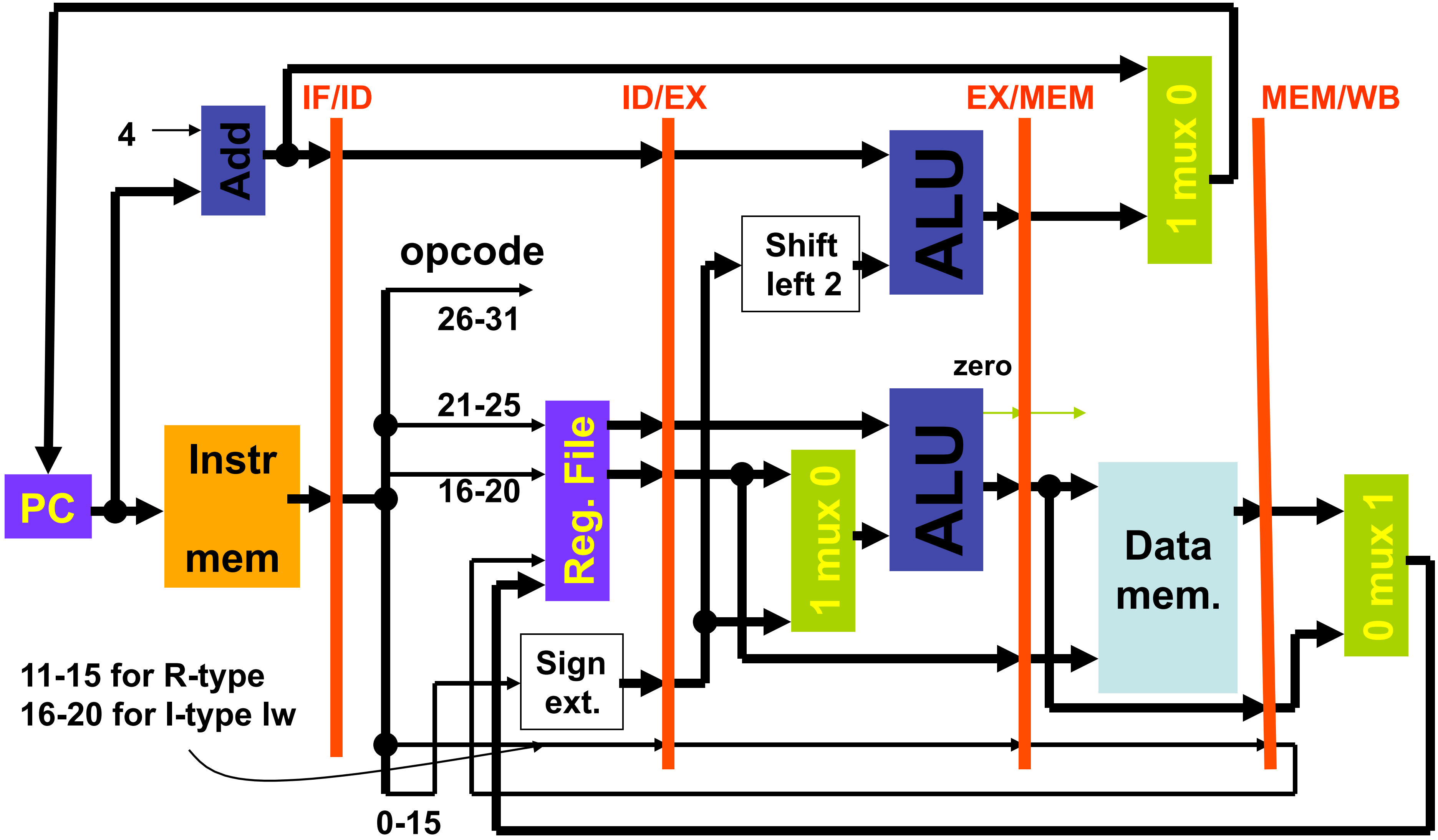
This requires a CONTROL not too different from single-cycle

Pipeline Register Functions

- Four pipeline registers are added:

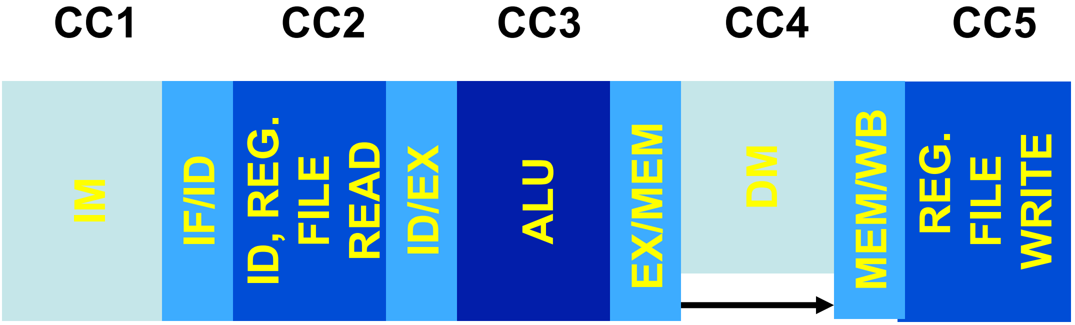
Register name	Data held
IF/ID	PC+4, Instruction word (IW)
ID/EX	PC+4, R1, R2, IW(0-15) sign ext., IW(11-15)
EX/MEM	PC+4, zero, ALUResult, R2, IW(11-15) or IW(16-20)
MEM/WB	M[ALUResult], ALUResult, IW(11-15) or IW(16-20)

Pipelined Datapath



11-15 for R-type
16-20 for I-type lw

Five-Cycle Pipeline



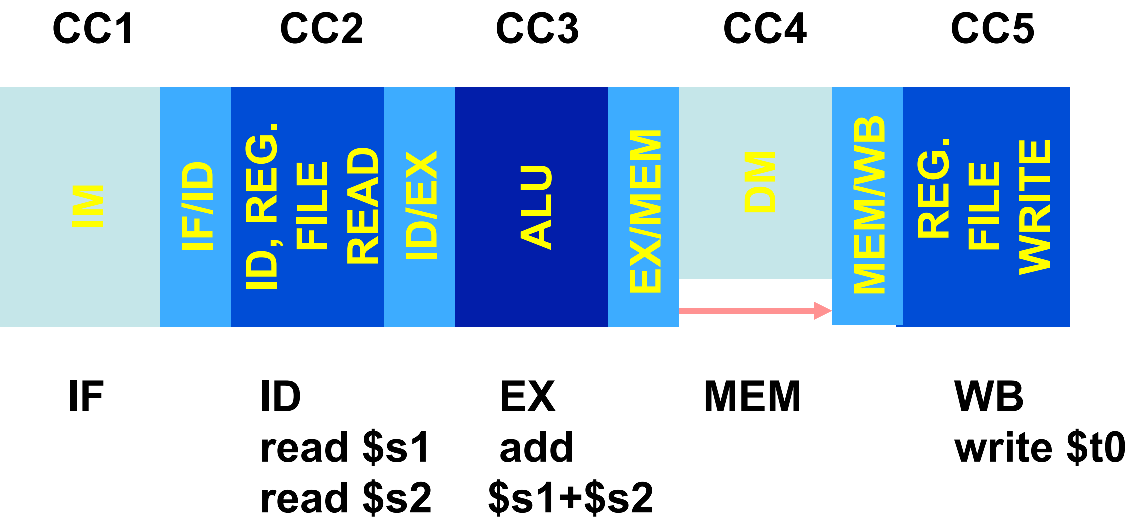
Add Instruction

- add $t0, $s1, $s2

  Machine instruction word

  000000 10001 10010 01000 00000 100000

  opcode $s1 $s2 $t0 function



CC1 CC2 CC3 CC4 CC5

IF

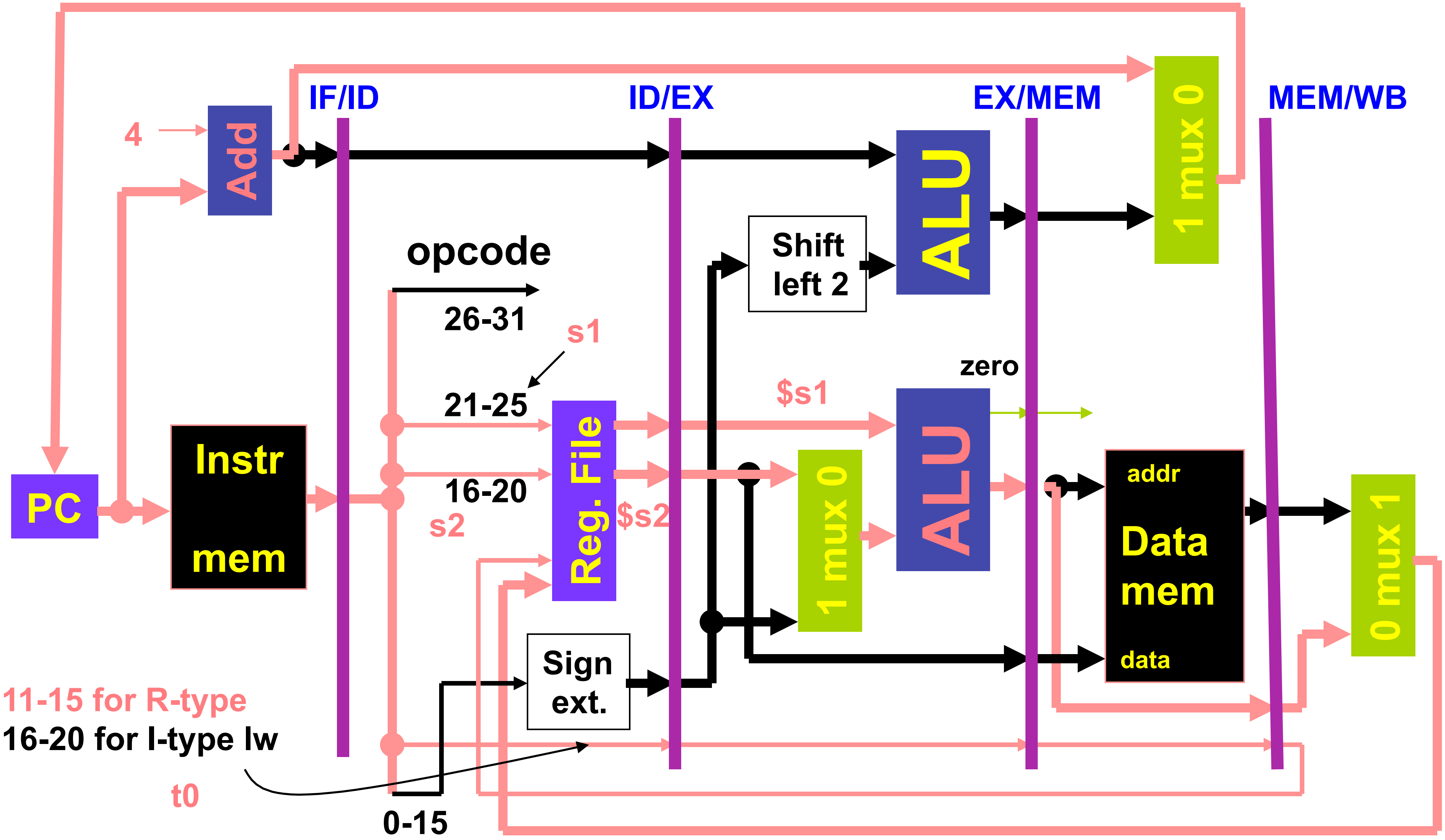
ID
read $s1
read $s2

EX
add
$s1+$s2

MEM

WB
write $t0

Pipelined Datapath Executing add

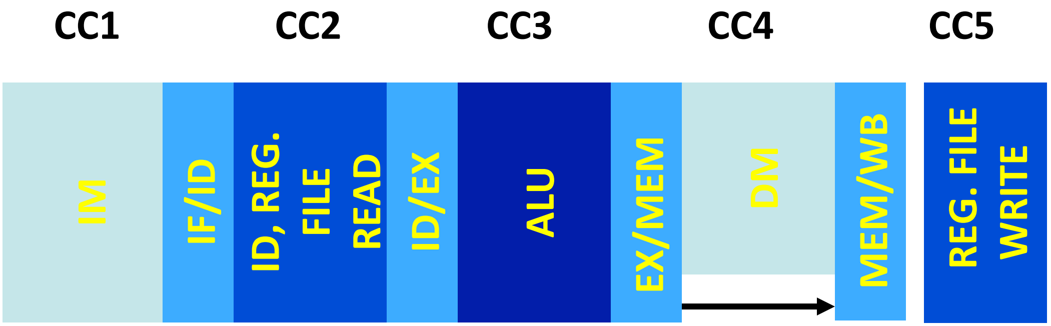


Load Instruction

- lw $t0, 1200 ($t1)

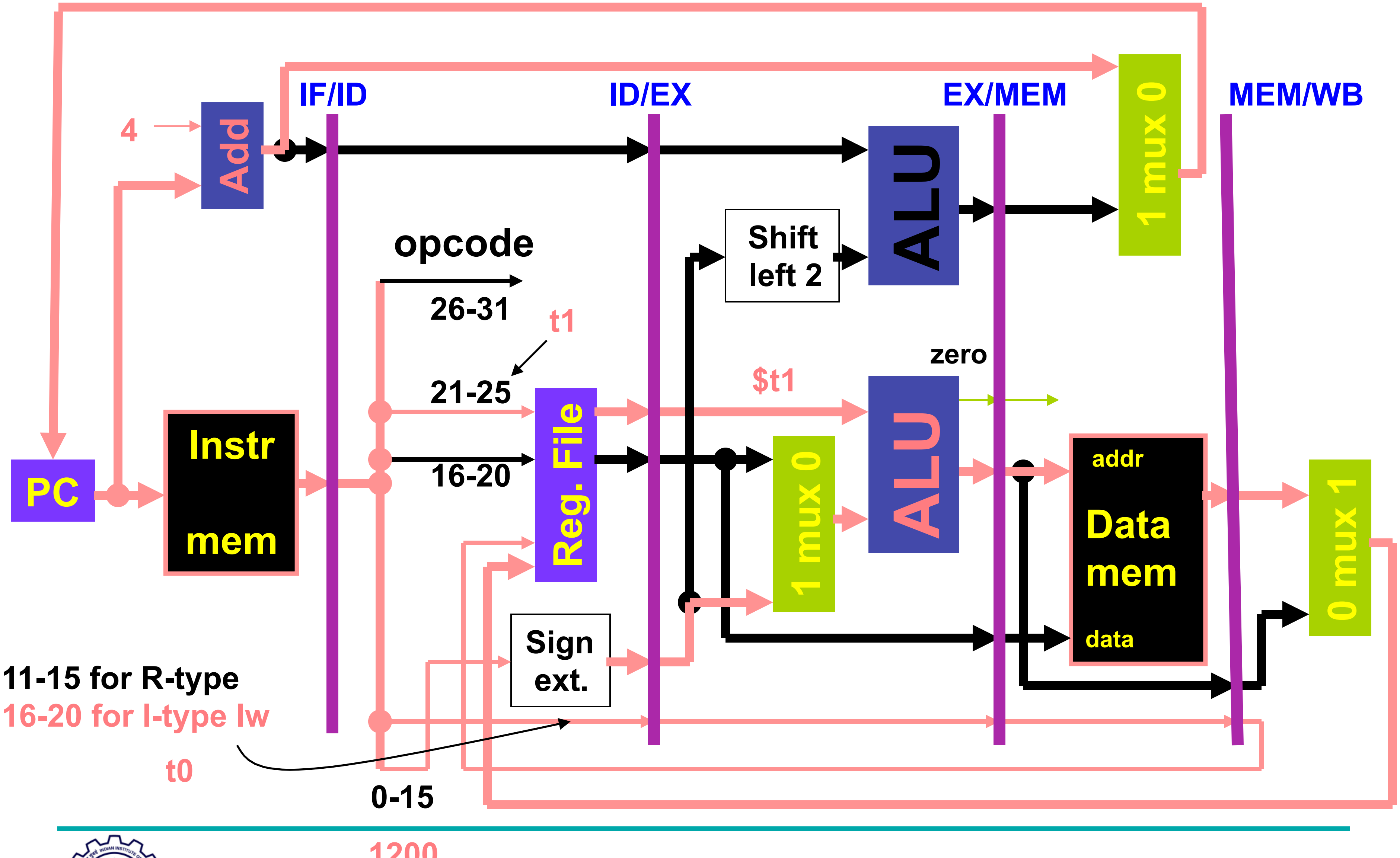
100011 01001 01000 0000 0100 1000 0000

opcode $t1 $t0 1200



IF	ID	EX	MEM	WB
	read $t1	add	read	write $t0
	sign ext	$t1+1200	M[addr]	
	1200			

Pipelined Datapath Executing lw

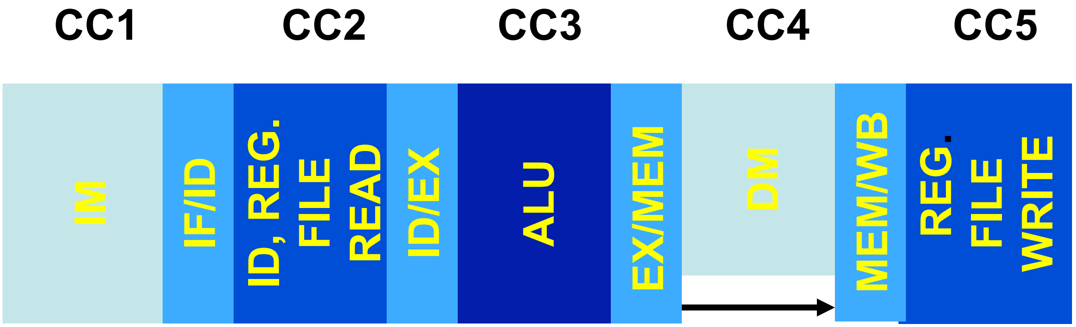


Store Instruction

- sw $t0, 1200 ($t1)

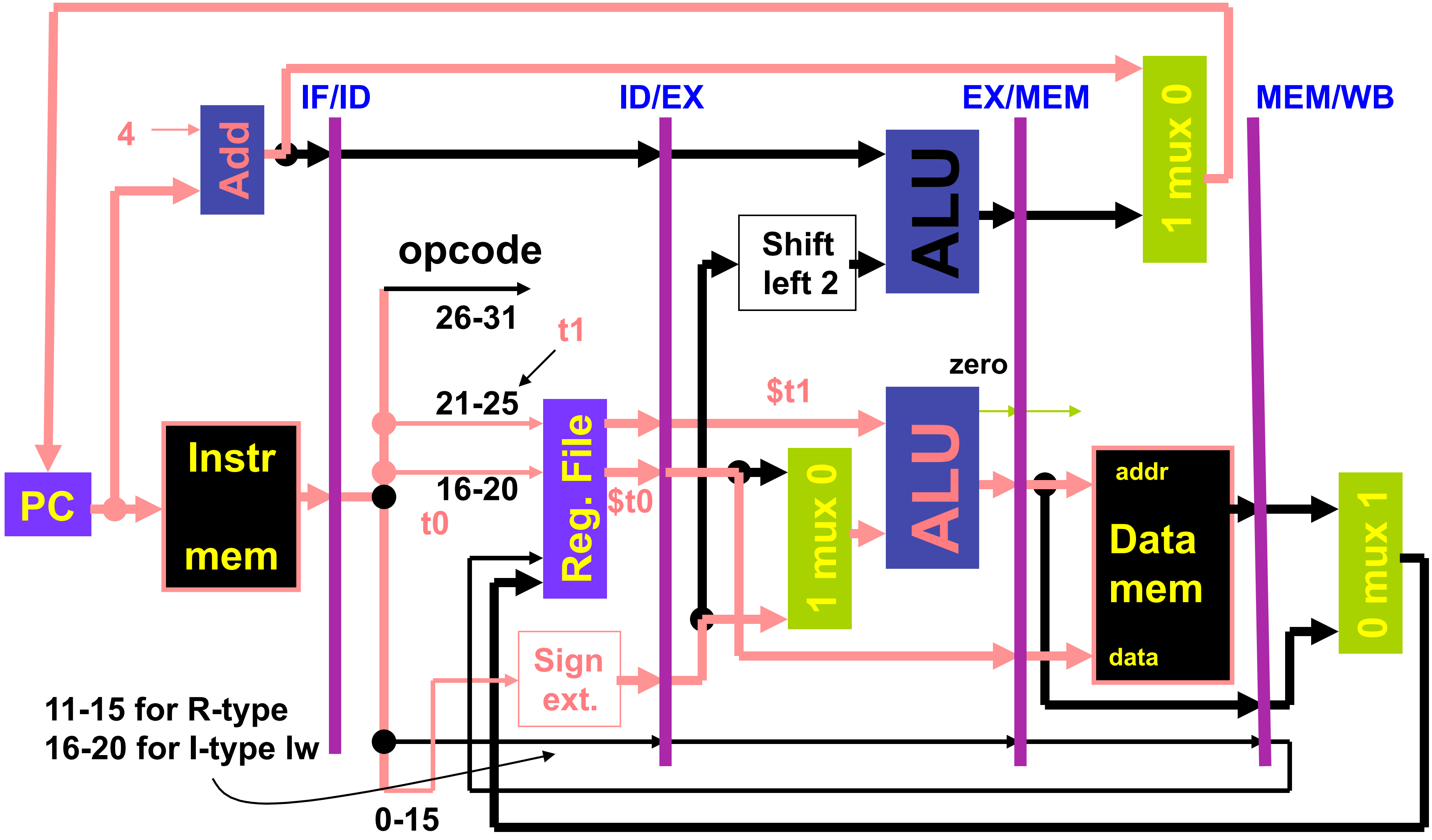
101011 01001 01000 0000 0100 1000 0000

opcode $t1 $t0 1200



IF	ID	EX	MEM	WB
	read $t1 sign ext 1200	add $t1+1200 (addr)	write M[addr] ← $t0	

Pipelined Datapath Executing sw

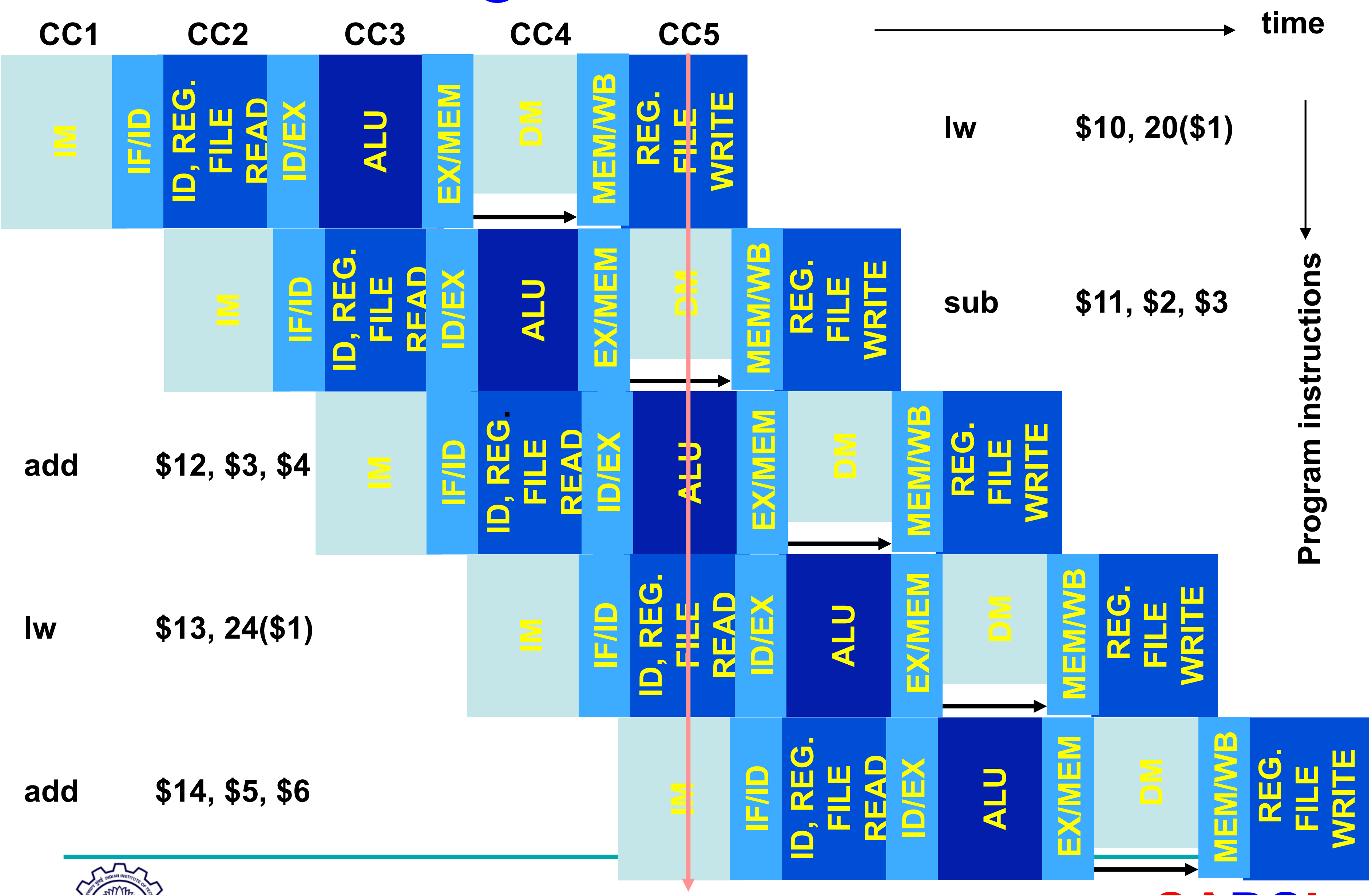


Executing a Program

Consider a five-instruction segment:

```
lw    $10, 20($1)
sub   $11, $2, $3
add   $12, $3, $4
lw    $13, 24($1)
add   $14, $5, $6
```

Program Execution



lw $10, 20($1)

sub $11, $2, $3

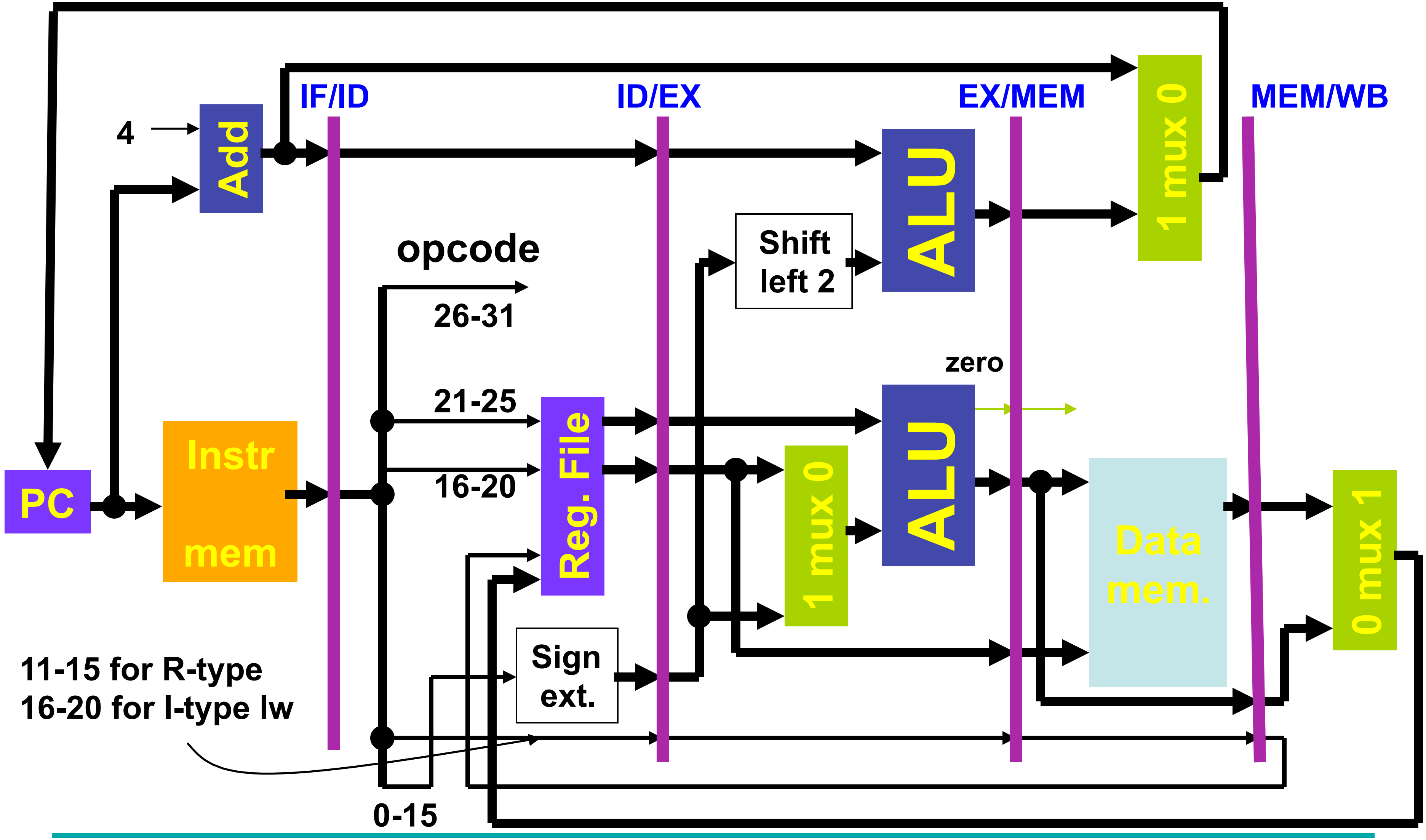
add $12, $3, $4

lw $13, 24($1)

add $14, $5, $6

CC5

IF: add $14, $5, $6 ID: lw $13, 24($1) EX: add $12, $3, $4 MEM: sub $11, $2, $3 WB: lw $10, 20($1)



Advantages of Pipeline

- After the fifth cycle (CC5), one instruction is completed each cycle; CPI ≈ 1, neglecting the initial pipeline latency of 5 cycles.
  - *Pipeline latency is defined as the number of stages in the pipeline, or*
  - *The number of clock cycles after which the first instruction is completed.*
- The clock cycle time is about four times shorter than that of single-cycle datapath and about the same as that of multicycle datapath.
- For multicycle datapath, CPI = 3. ….
- So, pipelined execution is faster, but . . .

Thank You