RISC Design

Pipelining

Virendra Singh
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

EE-309: Microprocessors

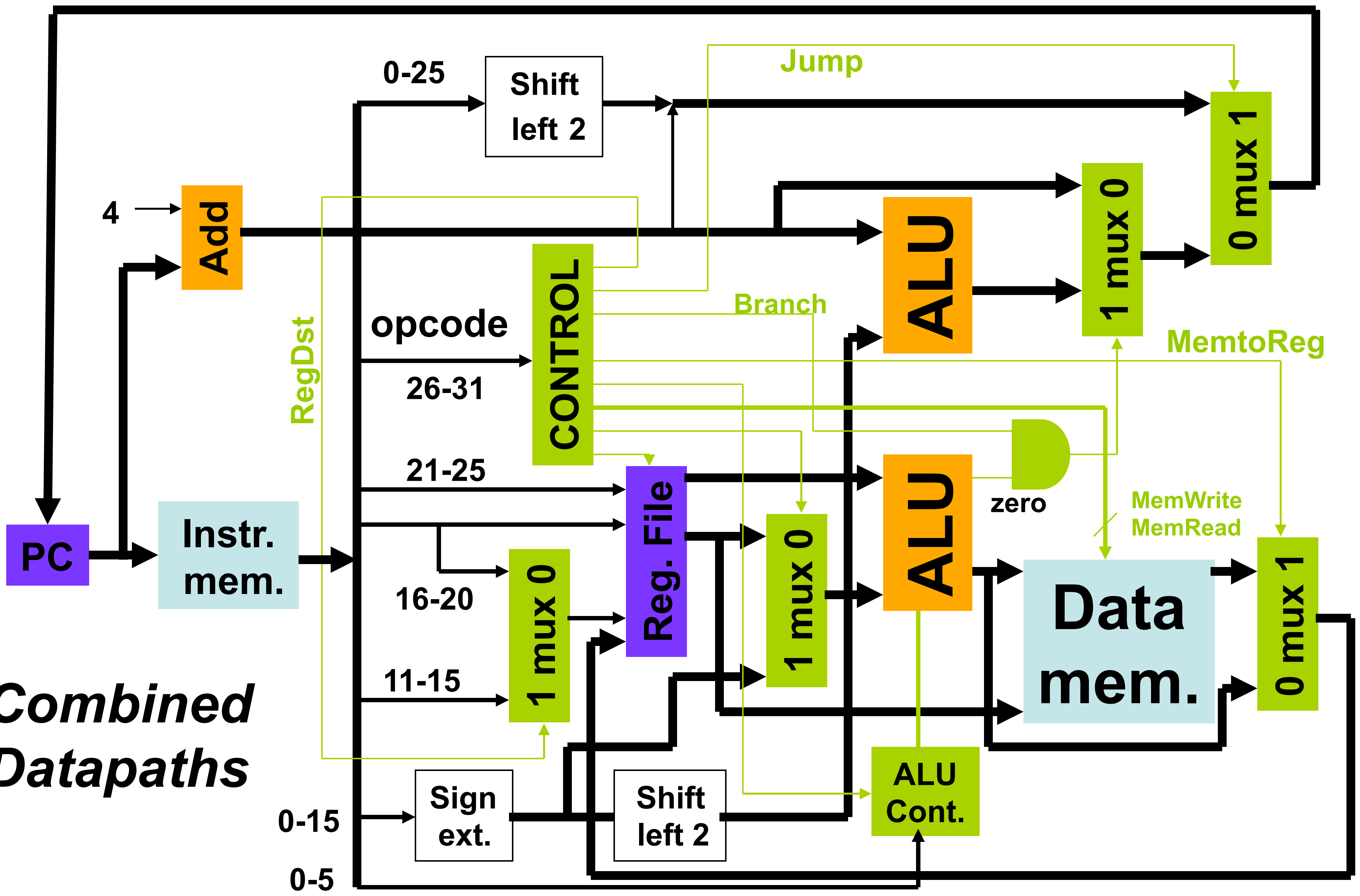
Lecture 35 (15 Oct 2015)

CADSL

Example Instruction Set: DLX Subset

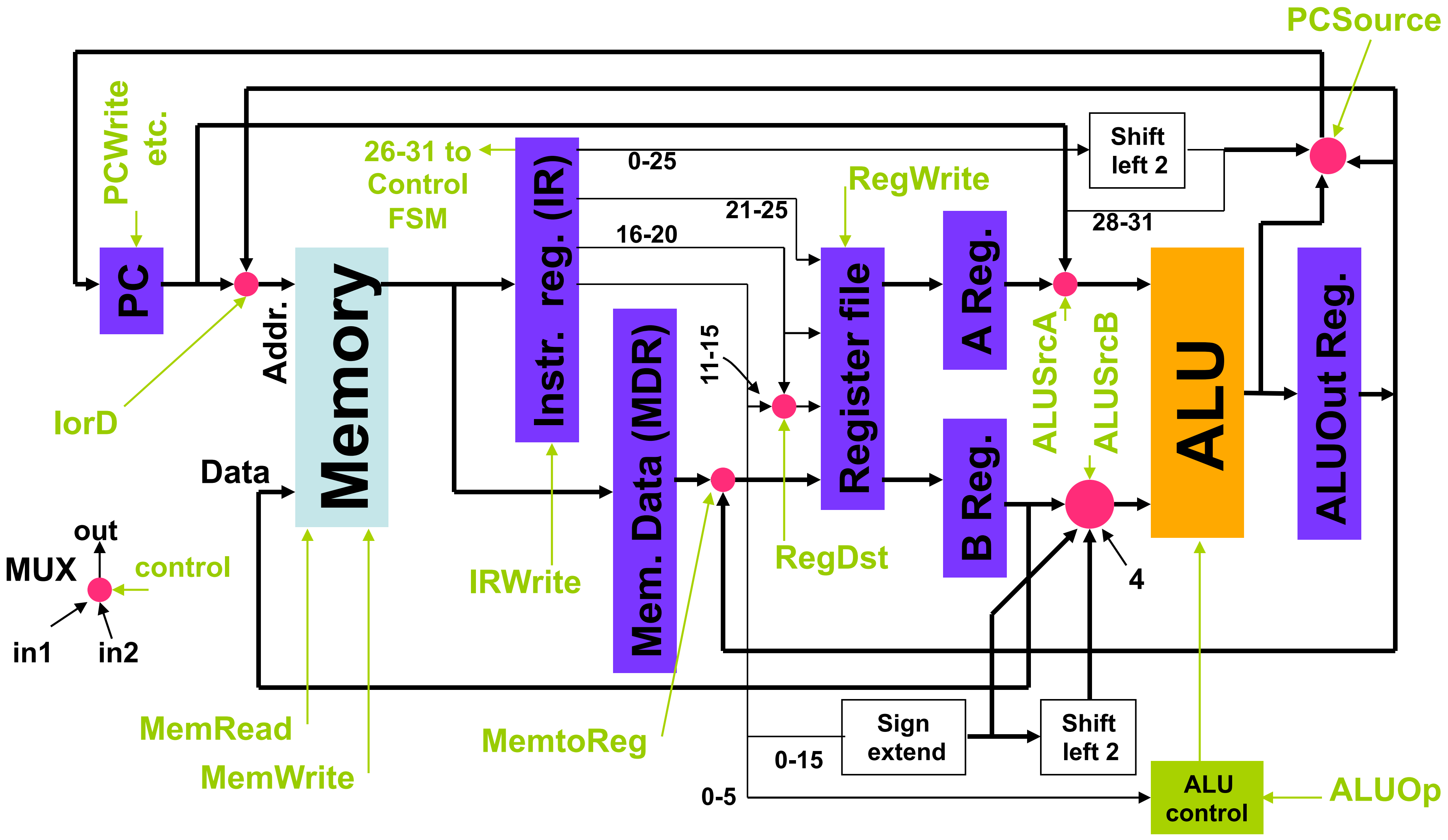
DLX Instruction – Subset

- Arithmetic and Logical Instructions
  - add, sub, xor, and, sll
- Memory Reference Instructions
  - lw, sw
- Branch
  - beq, j



Combined Datapaths

Multi-cycle Datapath



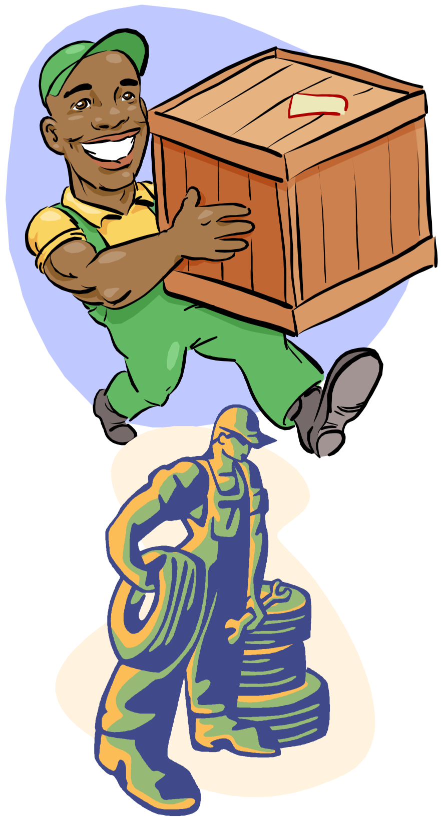
Beyond Single Cycle/ Multi-cycle Implementation

Traffic Flow

ILP: Instruction Level Parallelism

- Single-cycle and multi-cycle datapaths execute one instruction at a time.
- How can we get better performance?
- Answer: Execute multiple instruction at a time:
    - Pipelining – Enhance a multi-cycle datapath to fetch one instruction every cycle.

Automobile Team Assembly



1 car assembled every four hours
6 cars per day
180 cars per month
2,040 cars per year

Automobile Assembly Line

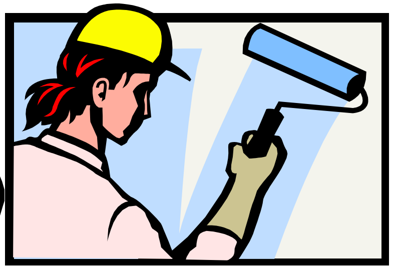
Task 1
1 hour

Mecahnical

Task 2
1 hour

Electrical

Task 3
1 hour



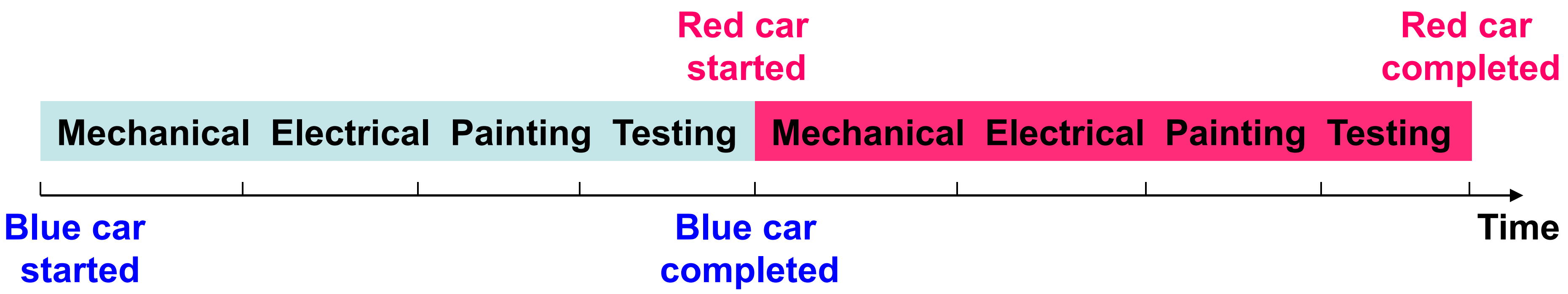
Painting

Task 4
1 hour

Testing

First car assembled in 4 hours (pipeline latency)
thereafter 1 car per hour
21 cars on first day, thereafter 24 cars per day
717 cars per month
8,637 cars per year

Throughput: Team Assembly

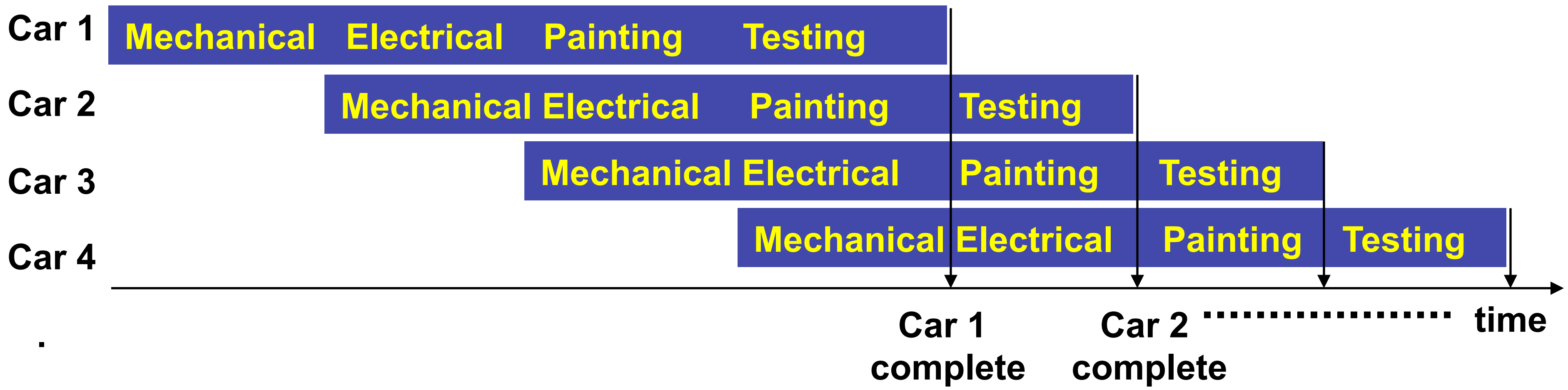

Red car started
Red car completed
Mechanical Electrical Painting Testing
Mechanical Electrical Painting Testing
Blue car started
Blue car completed
Time

Time of assembling one car = n hours

where n is the number of nearly equal subtasks, each requiring 1 unit of time

Throughput = $1/n$ cars per unit time

Throughput: Assembly Line



.

.

Time to complete first car = n time units (latency)

Cars completed in time T = $T - n + 1$

Throughput = $1 - (n - 1)/T$ car per unit time

$$\frac{\text{Throughput (assembly line)}}{\text{Throughput (team assembly)}} = \frac{1 - (n-1)/T}{1/n} = n - \frac{n(n-1)}{T} \rightarrow n \text{ as } T \rightarrow \infty$$

Some Features of Assembly Line

Electrical parts delivered (JIT)

Task 1	Task 2	Task 3	Task 4
1 hour	1 hour	1 hour	1 hour
Mechanical	Electrical	Painting	Testing

Defect found

3 cars in the assembly line are suspects, to be removed (flush pipeline)

Stall assembly line to fix the cause of defect

Pipelining in a Computer

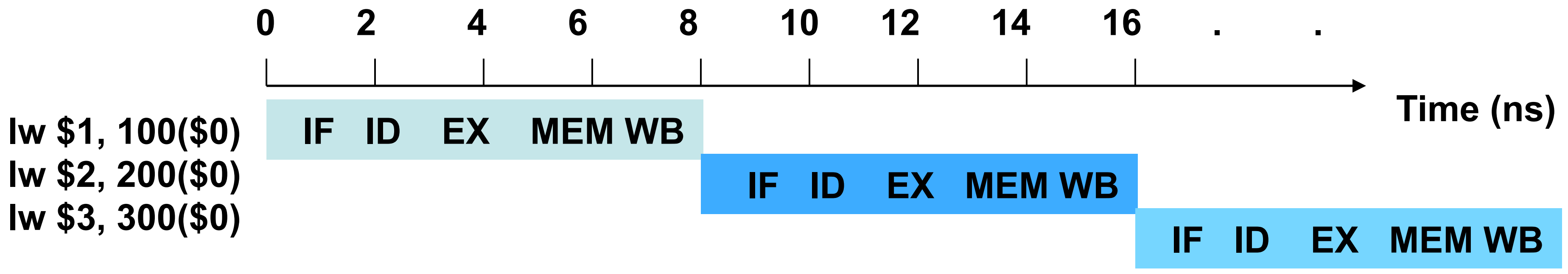
- Divide datapath into nearly equal tasks, to be performed serially and requiring non-overlapping resources.
- Insert registers at task boundaries in the datapath; registers pass the output data from one task as input data to the next task.
- Synchronize tasks with a clock having a cycle time that just 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.

Single-Cycle 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	1ns	8ns
sw	2ns	1ns	2ns	2ns	🟦	8ns
R-format add, sub, and, xor, sll	2ns	1ns	2ns	🟦	1ns	8ns
B-format, beq	2ns	1ns	2ns	🟦	🟦	8ns

No operation on data; idle time equalizes instruction length to a fixed clock period.

Execution Time: Single-Cycle



Clock cycle time = 8 ns

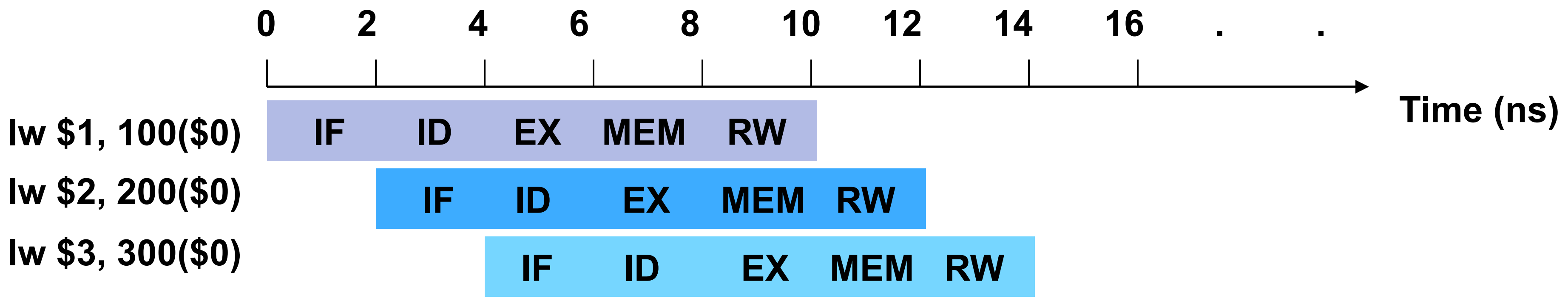
Total time for executing three lw instructions = 24 ns

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.

Execution Time: Pipeline



Clock cycle time = 2 ns, *four times faster than single-cycle clock*

Total time for executing three lw instructions = 14 ns

$$\text{Performance ratio} = \frac{\text{Single-cycle time}}{\text{Pipeline time}} = \frac{24}{14} = 1.7$$

Pipeline Performance

Clock cycle time = 2 ns

1,003 *lw instructions:*

Total time for executing 1,003 lw instructions = 2,014 ns

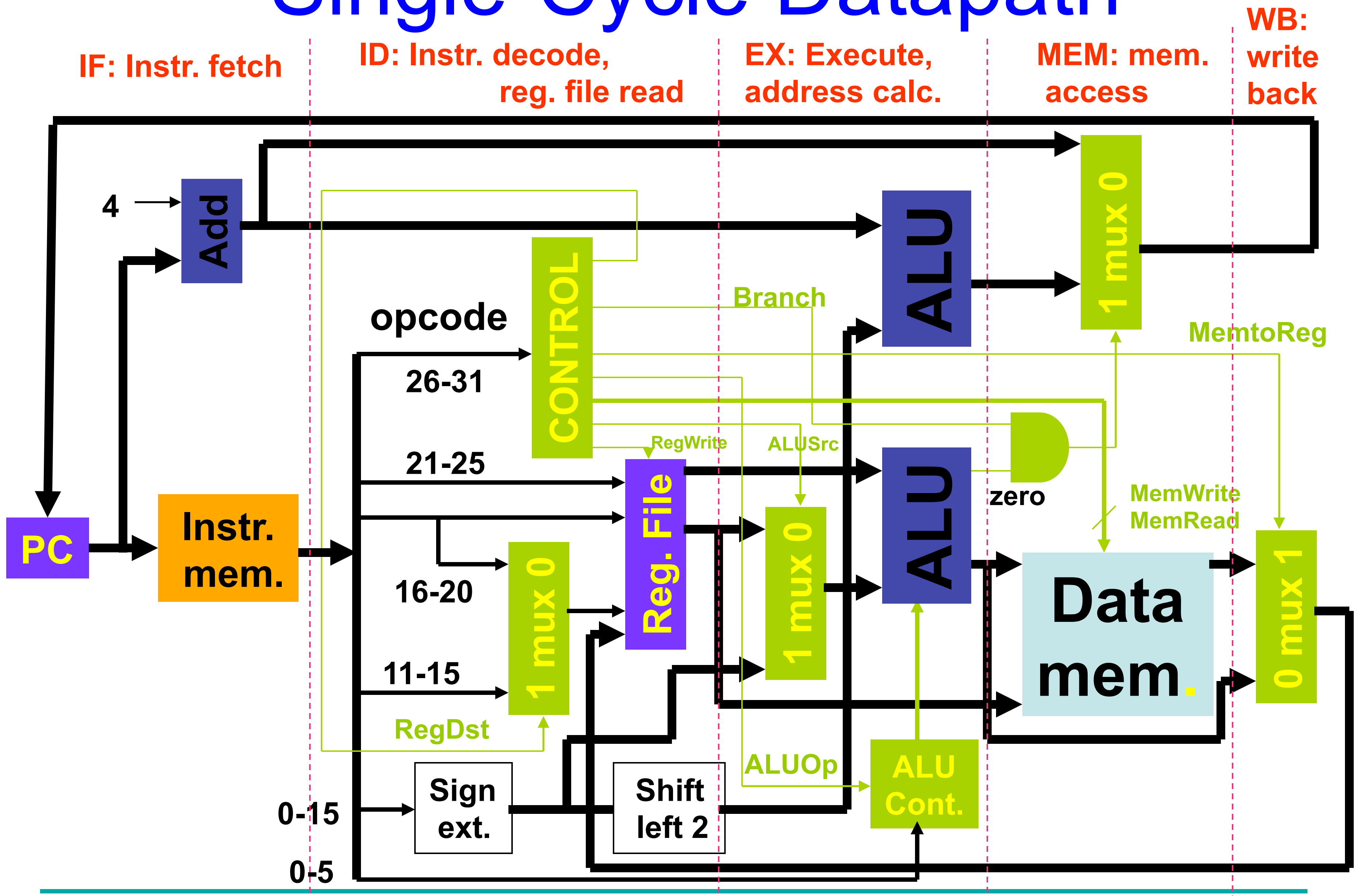
$$\text{Performance ratio} = \frac{\text{Single-cycle time}}{\text{Pipeline time}} = \frac{8{,}024}{2{,}014} = 3.98$$

10,003 *lw instructions:*

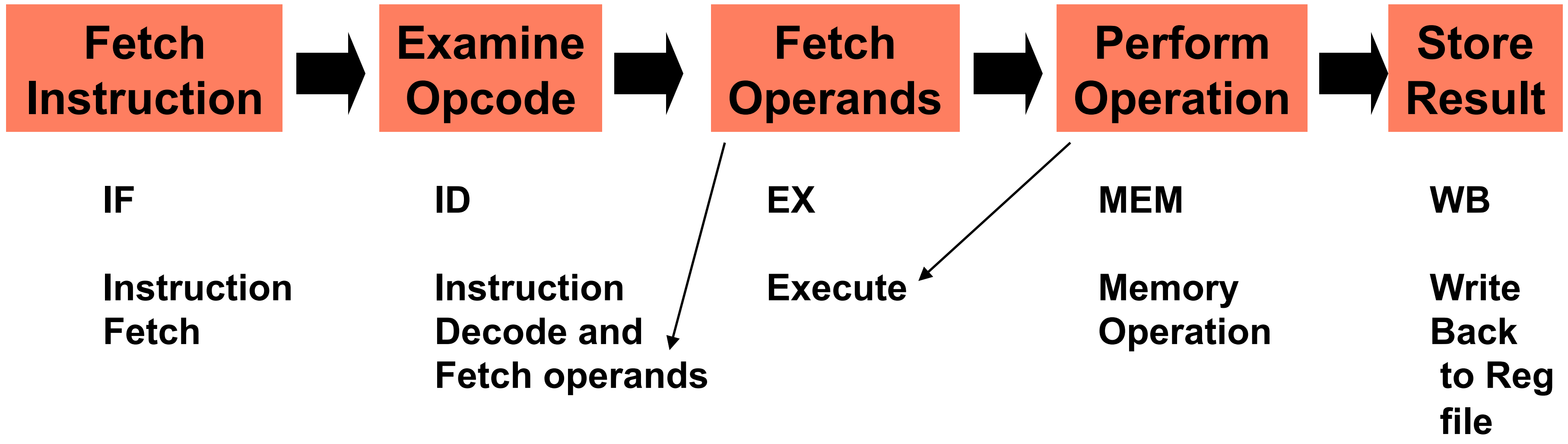
Performance ratio = 80,024 / 20,014 = 3.998 → Clock cycle ratio (4)

Pipeline performance approaches clock-cycle ratio for long programs.

Single-Cycle Datapath

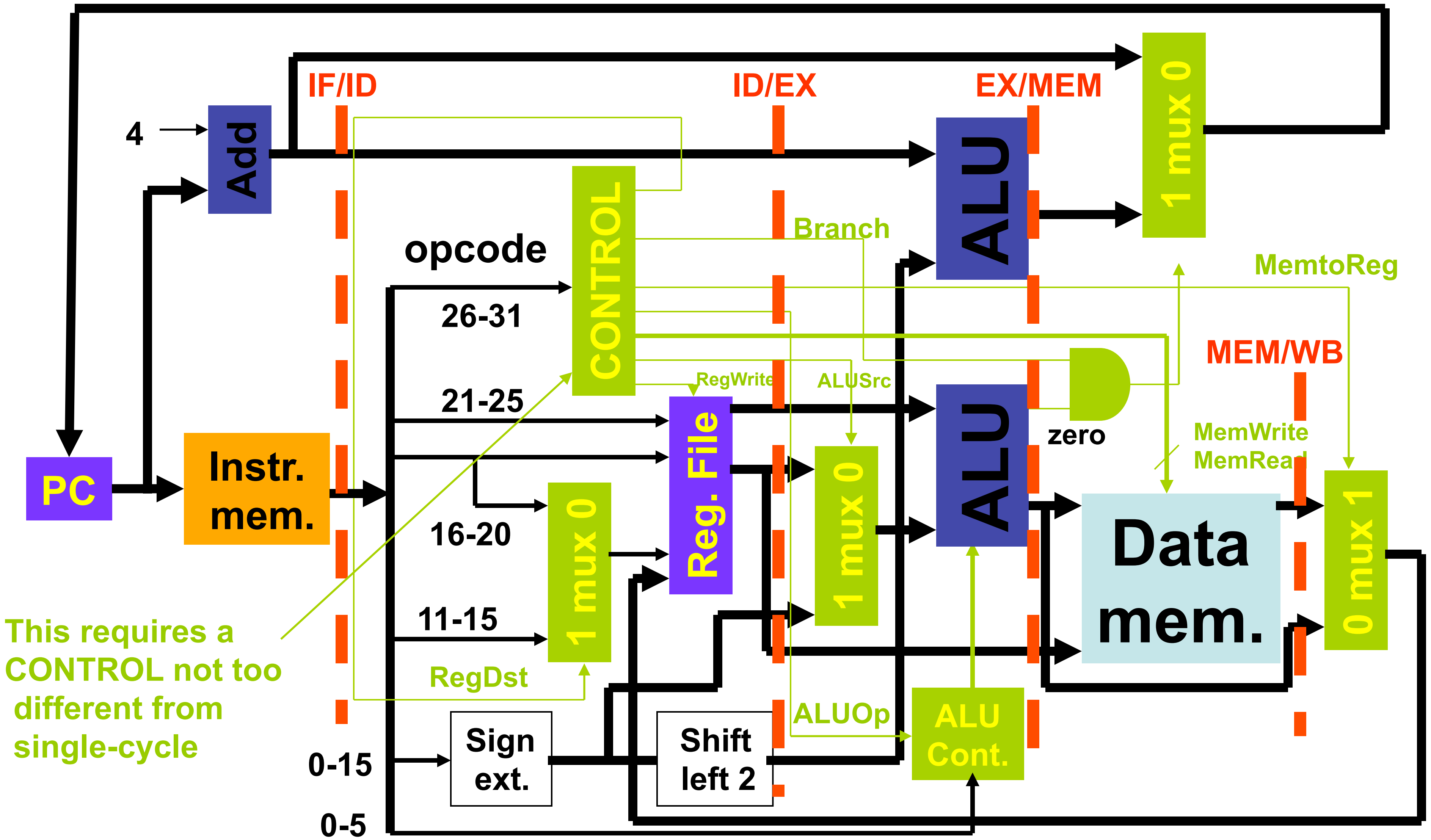


Pipelining of RISC Instructions



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

Pipeline Registers



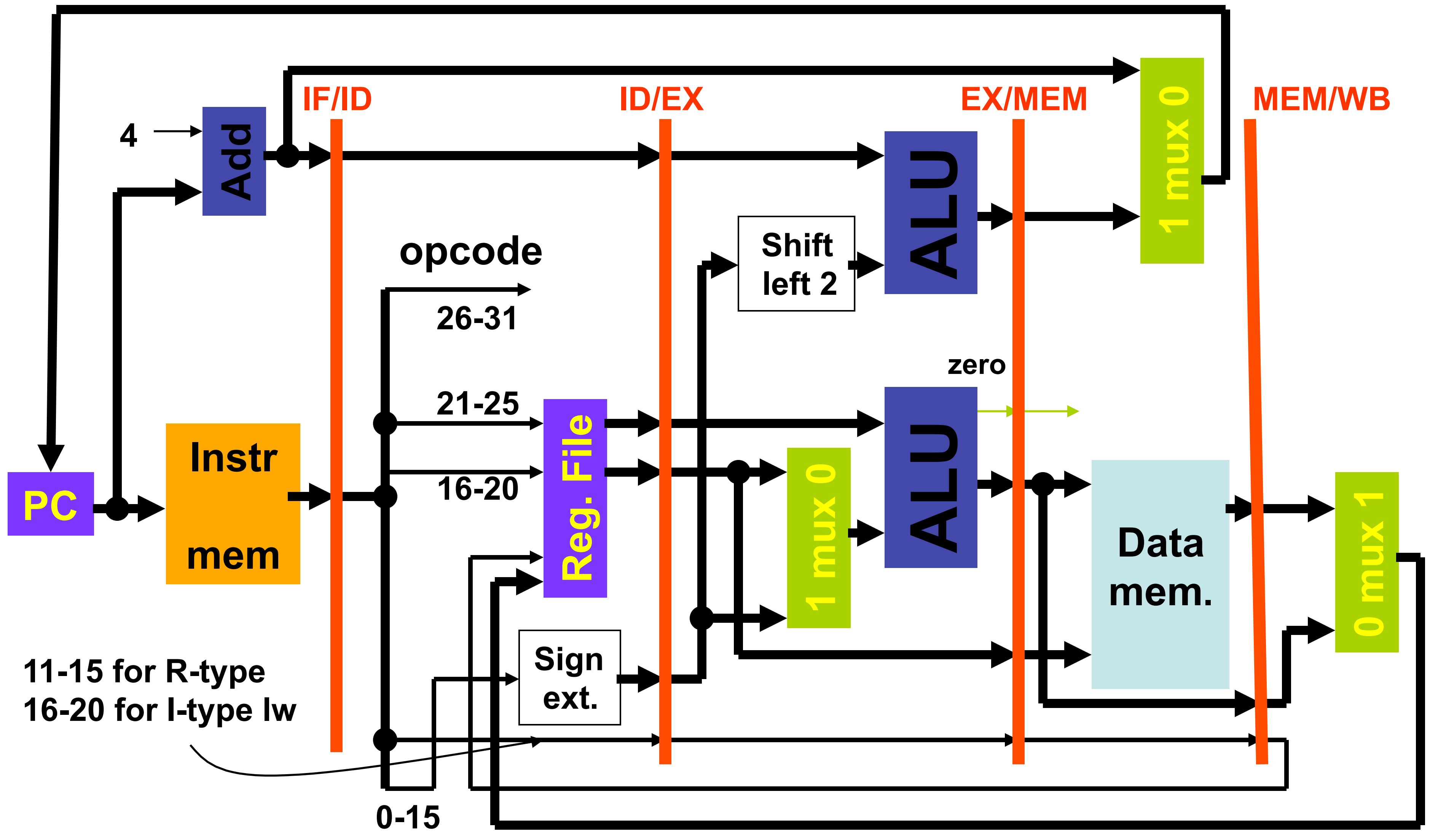
This requires a CONTROL not too different from single-cycle

Pipeline Register Functions

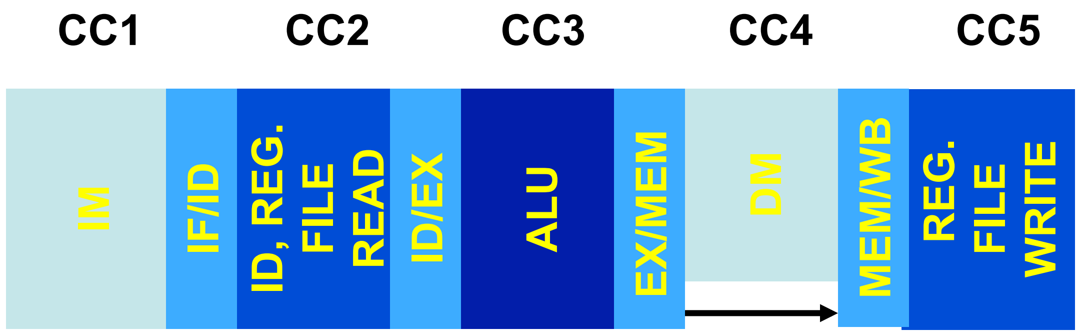
- Four pipeline registers are added:

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



Five-Cycle Pipeline



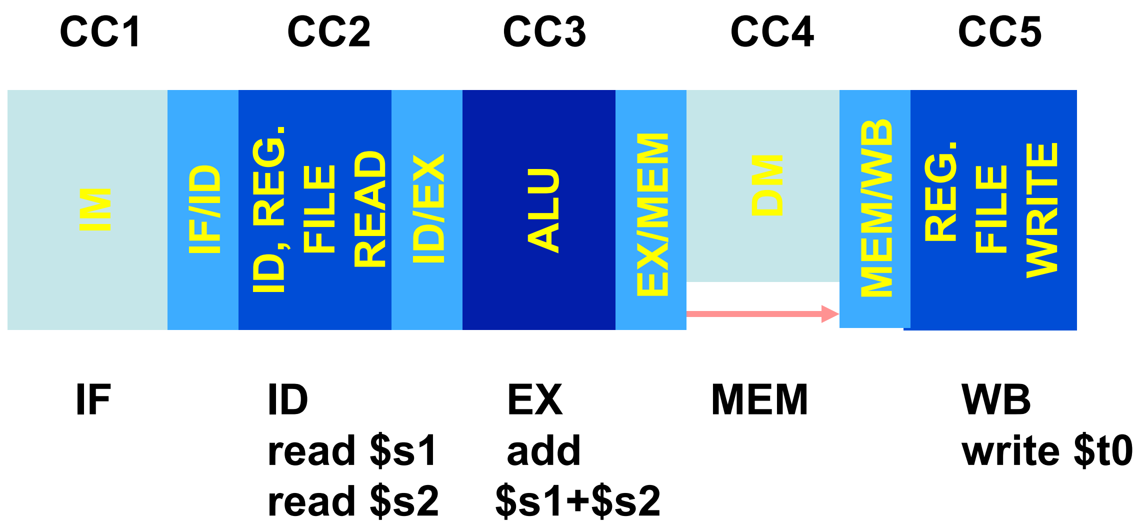
Add Instruction

- add R8, R17, R18

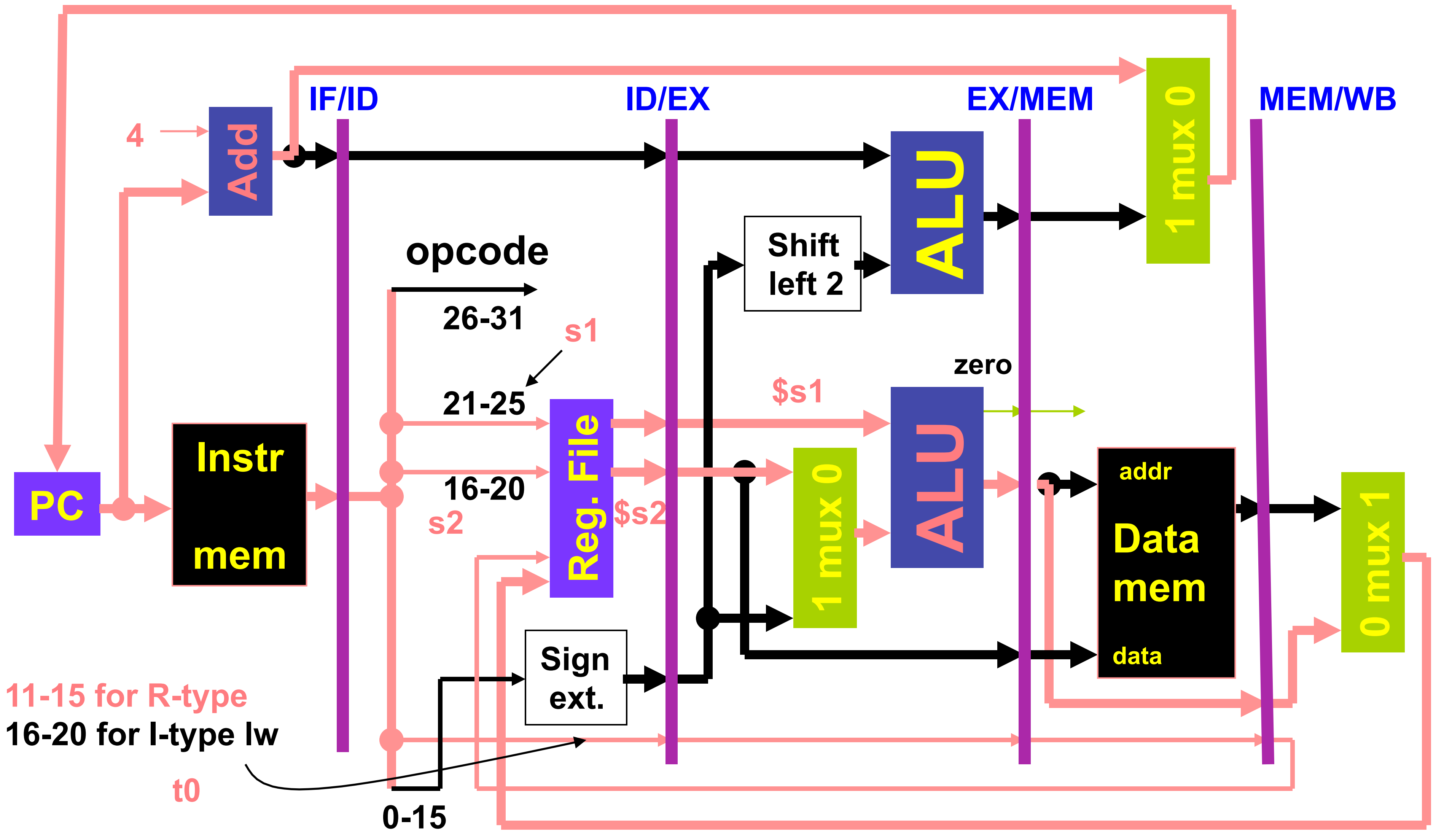
  Machine instruction word

  000000 10001 10010 01000 00000 100000

  opcode $s1 $s2 $t0 function



Pipelined Datapath Executing add

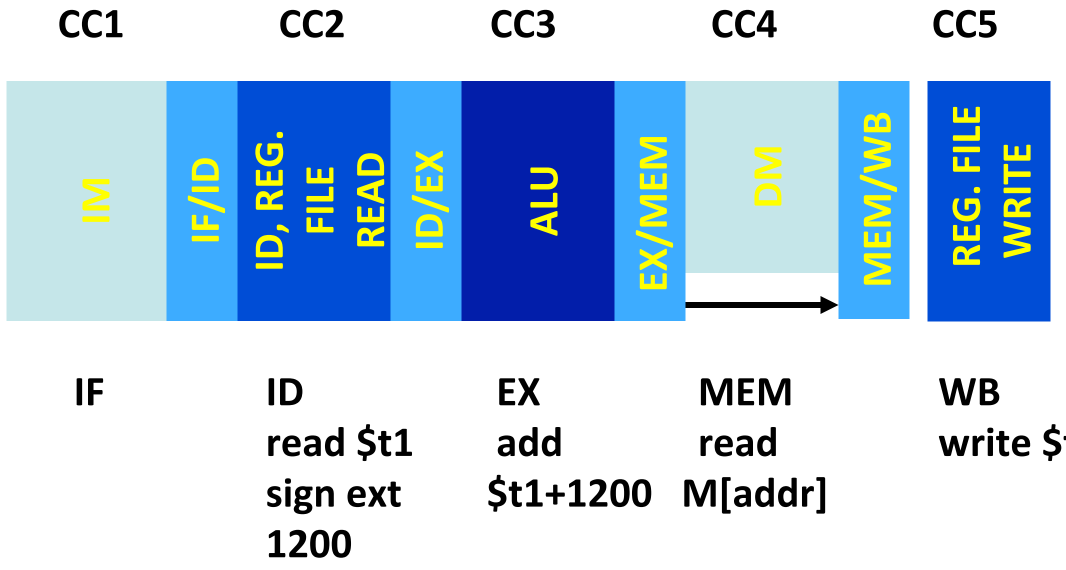


Load Instruction

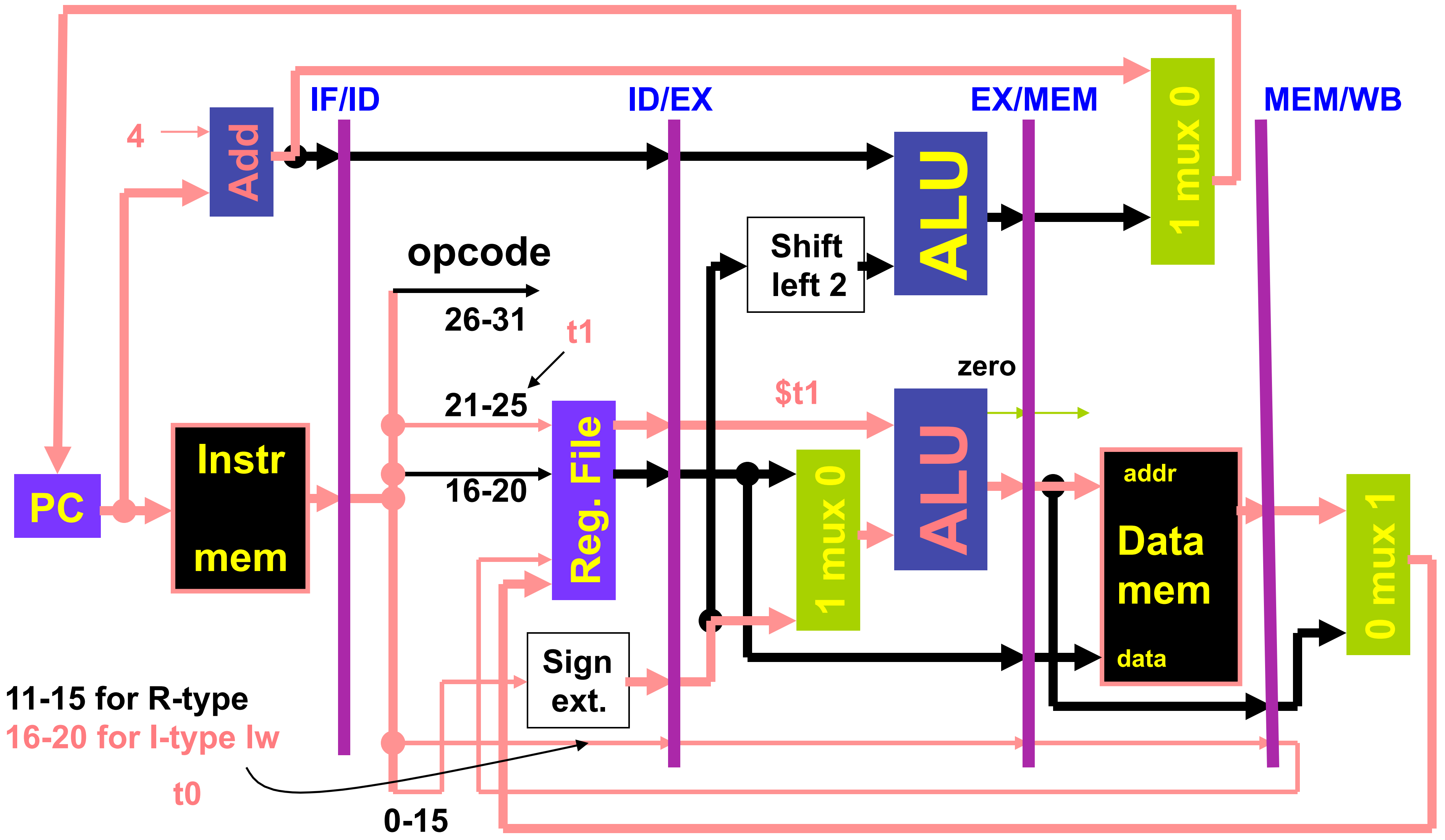
- lw R8, 1200 (R9)

100011 01001 01000 0000 0100 1000 0000

opcode R9 R8 1200



Pipelined Datapath Executing lw



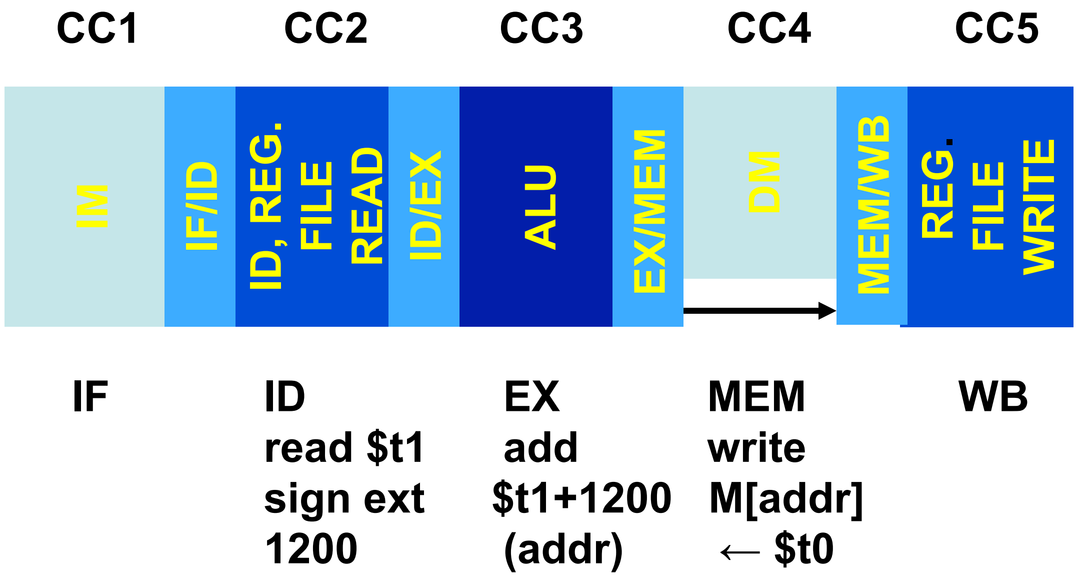
1200

Store Instruction

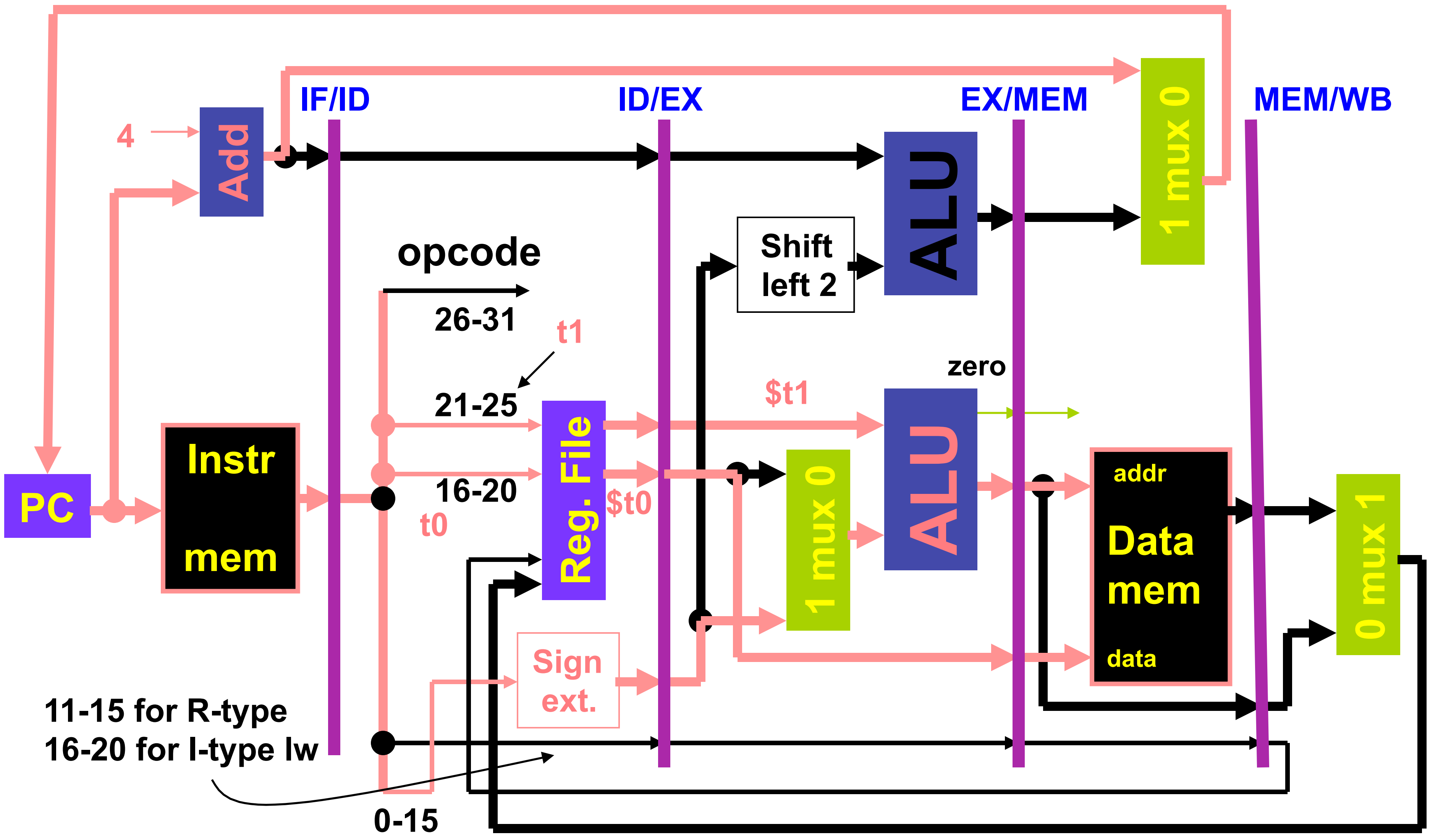
- sw R8, 1200 (R9)

101011 01001 01000 0000 0100 1000 0000

opcode R9 R8 1200



Pipelined Datapath Executing sw

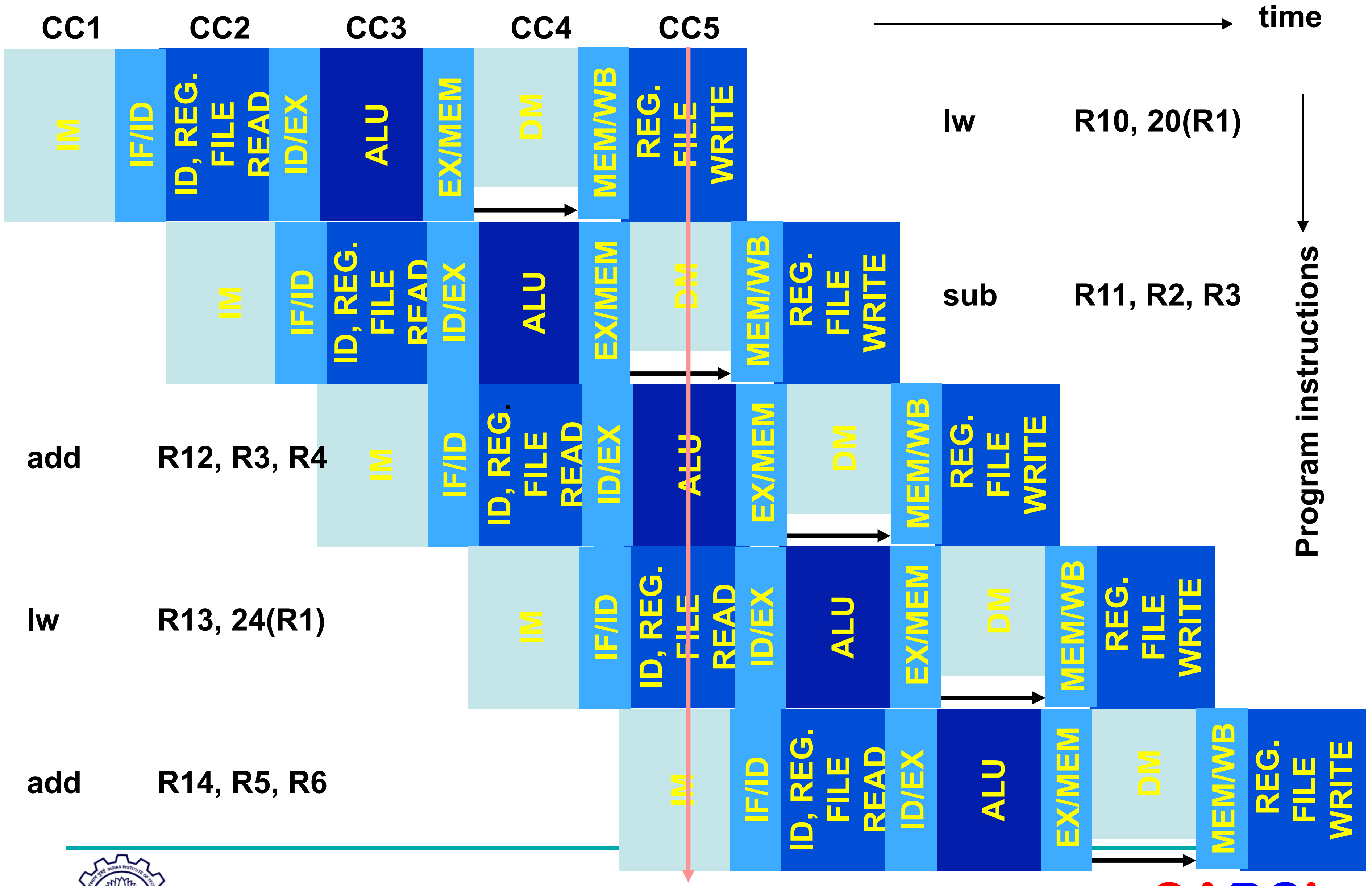


Executing a Program

Consider a five-instruction segment:

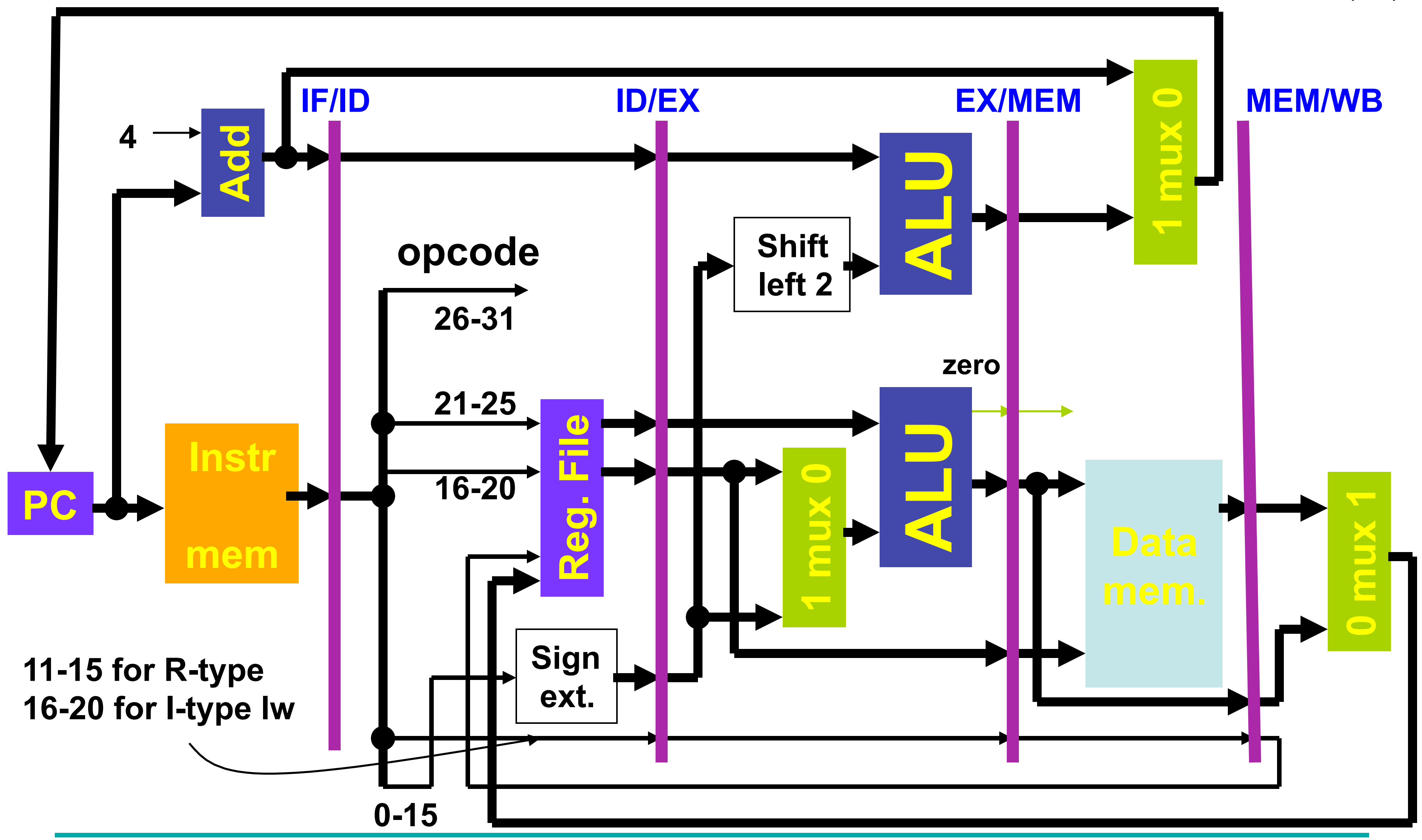
```
lw    R10, 20(R1)
sub   R11, R2, R3
add   R12, R3, R4
lw    R13, 24(R1)
add   R14, R5, R6
```

Program Execution



CC5

IF: add R14, R5, R6 ID: lw R13, 24(R1) EX: add R12, R3, R4 MEM: sub R11, R2, R3 WB: lw R10, 20(R1)



Single Lane Traffic

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