Computer Architecture

An Introduction

Virendra Singh

Associate Professor

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

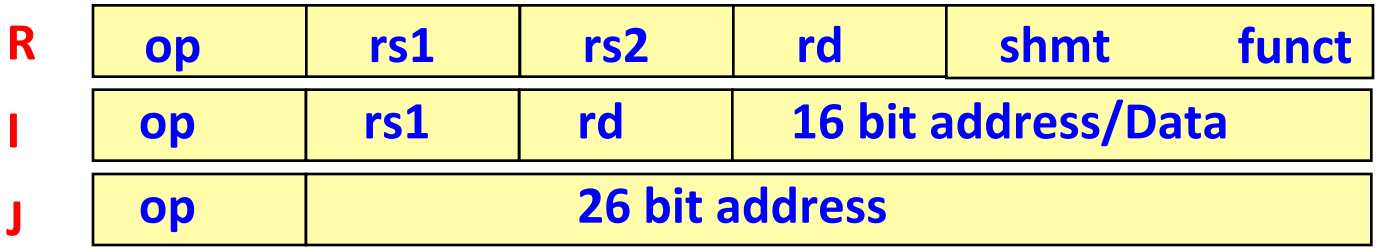
CS-683: Advanced Computer Architecture

Lecture 4 (07 Aug 2013)

CADSL

Overview of MIPS

- Simple instructions, all 32 bits wide
- Very structured, no unnecessary baggage
- Only three instruction formats



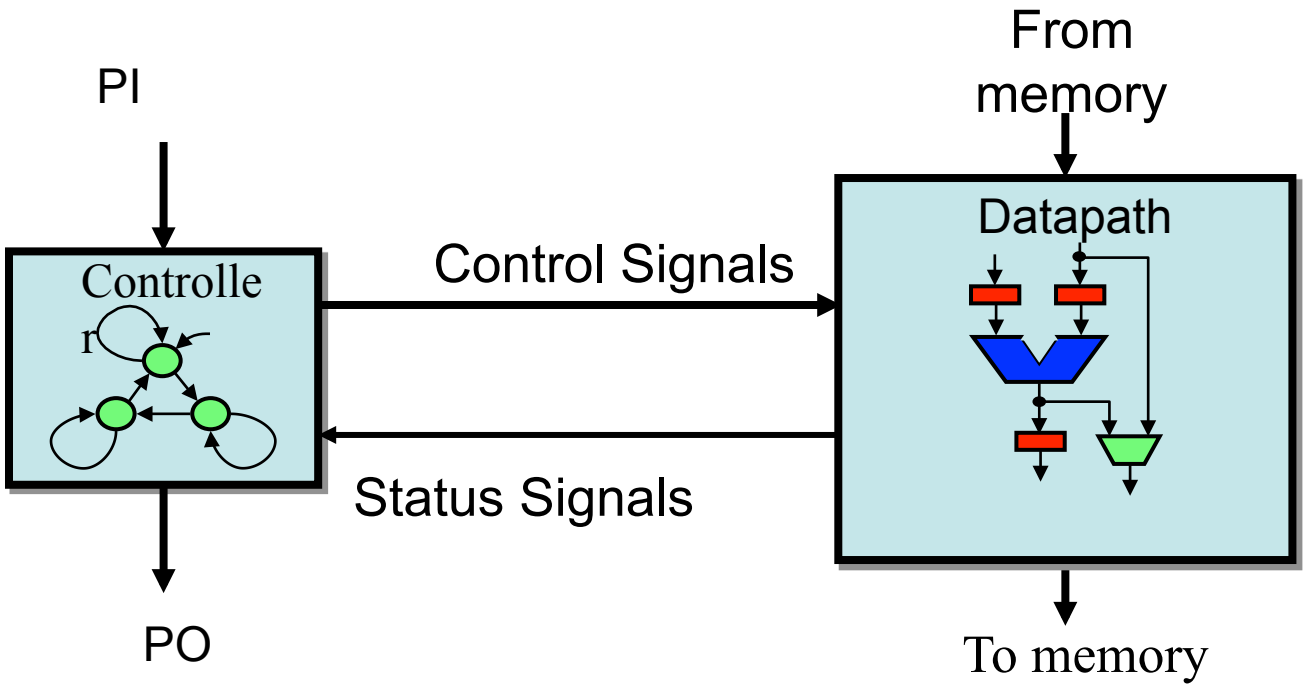
- Rely on compiler to achieve performance

Example processor MIPS subset

MIPS Instruction – Subset

- Arithmetic and Logical Instructions
  - add, sub, or, and, slt
- Memory reference Instructions
  - lw, sw
- Branch
  - beq, j

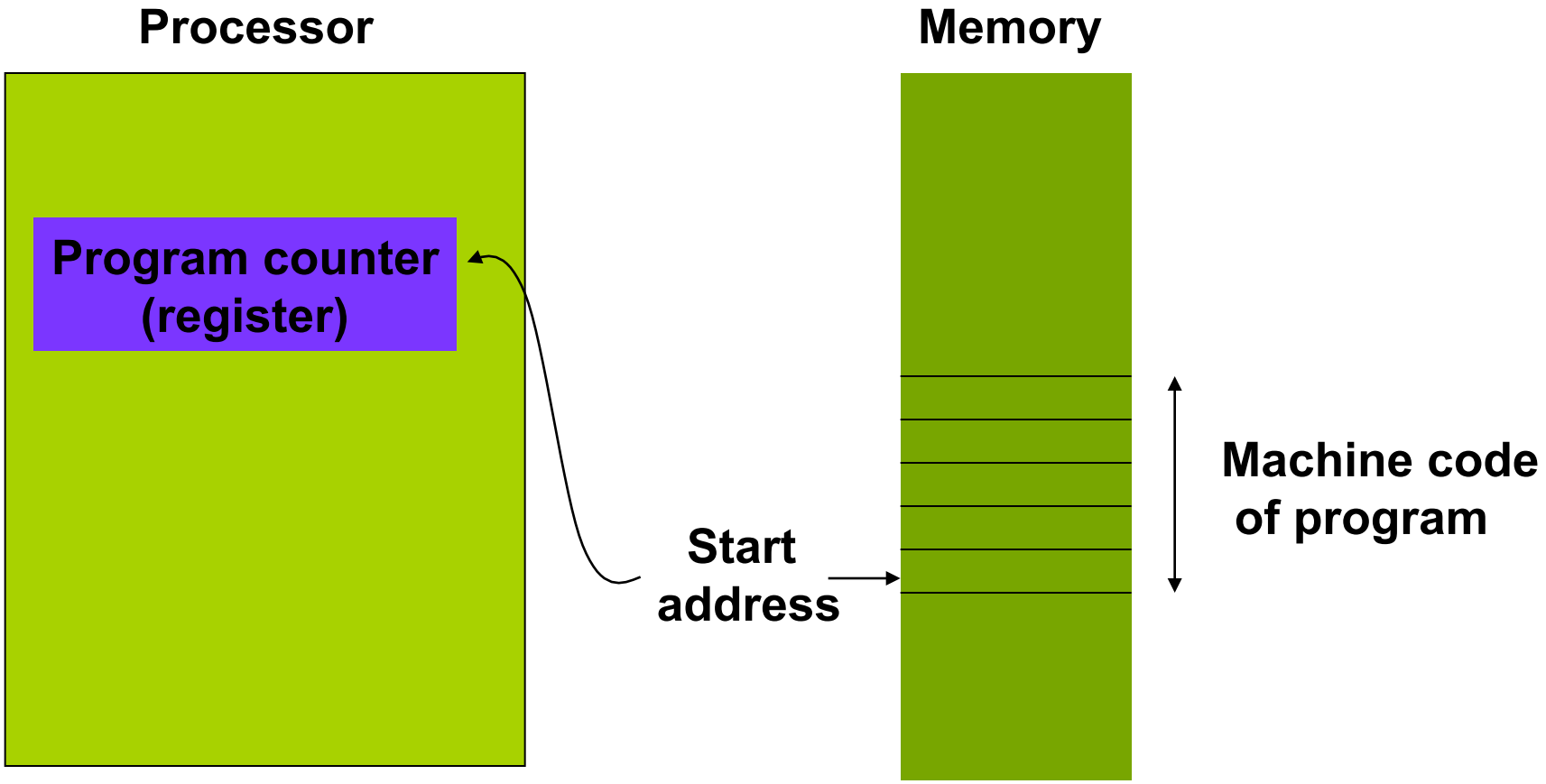
Processor Architecture



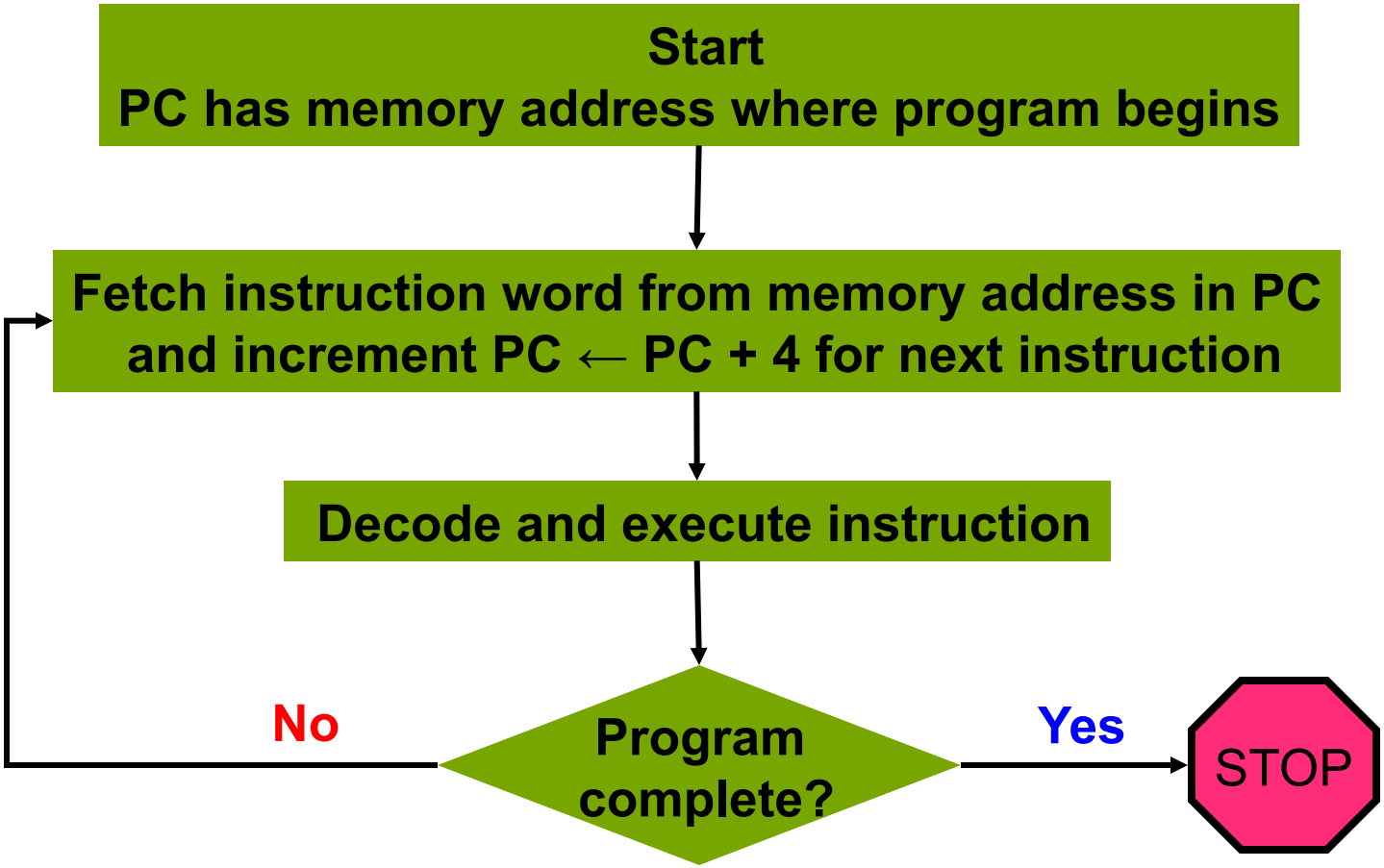
Where Does It All Begin?

- In a register called *program counter (PC)*.
- PC contains the memory address of the next instruction to be executed.
- In the beginning, PC contains the address of the memory location where the program begins.

Where is the Program?

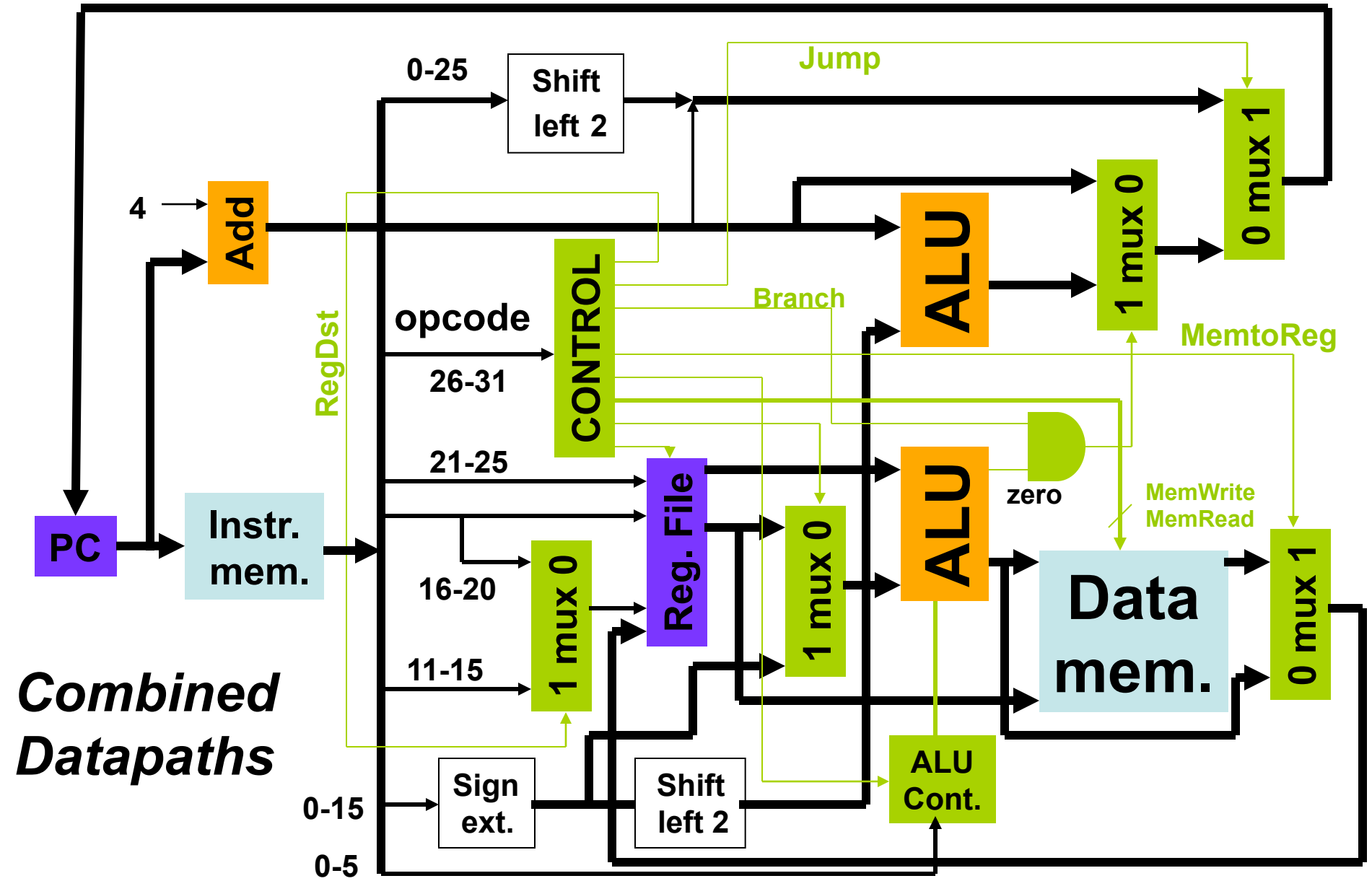


How Does It Run?



Datapath and Control

- Datapath: Memory, registers, adders, ALU, and communication buses. Each step (fetch, decode, execute) requires communication (data transfer) paths between memory, registers and ALU.
- Control: Datapath for each step is set up by control signals that set up dataflow directions on communication buses and select ALU and memory functions. Control signals are generated by a control unit consisting of one or more finite-state machines.


Combined Datapaths
0-25
Shift left 2
Jump
4
Add
ALU
1 mux 0
0 mux 1
RegDst
opcode
26-31
CONTROL
Branch
MemtoReg
21-25
Reg. File
ALU
zero
MemWrite
MemRead
PC
Instr. mem.
16-20
1 mux 0
1 mux 0
Data mem.
0 mux 1
11-15
Sign ext.
Shift left 2
ALU Cont.
0-15
0-5

How Long Does It Take?

- Assume control logic is fast and does not affect the critical timing. Major time delay components are ALU, memory read/write, and register read/write.
- Arithmetic-type (R-type)
  - Fetch (memory read) 2ns
  - Register read 1ns
  - ALU operation 2ns
  - Register write 1ns
  - Total 6ns

Time for lw and sw (I-Types)

- ALU (R-type) 6ns
- Load word (I-type)
  - Fetch (memory read) 2ns
  - Register read 1ns
  - ALU operation 2ns
  - Get data (mem. Read) 2ns
  - Register write 1ns
  - Total 8ns
- Store word (no register write) 7ns

Time for beq (I-Type)

- ALU (R-type) 6ns
- Load word (I-type) 8ns
- Store word (I-type) 7ns
- Branch on equal (I-type)
  - Fetch (memory read) 2ns
  - Register read 1ns
  - ALU operation 2ns
  - Total 5ns

Time for Jump (J-Type)

- ALU (R-type) 6ns
- Load word (I-type) 8ns
- Store word (I-type) 7ns
- Branch on equal (I-type) 5ns
- Jump (J-type)
  - Fetch (memory read) 2ns
  - Total 2ns

How Fast Can the Clock Be?

- If every instruction is executed in one clock cycle, then:
  - Clock period must be at least 8ns to perform the longest instruction, i.e., *lw*.
  - This is a single cycle machine.
  - It is slower because many instructions take less than 8ns but are still allowed that much time.
- Method of speeding up: Use multicycle datapath.

Multicycle Instruction Execution

Step	R-type (4 cycles)	Mem. Ref. (4 or 5 cycles)	Branch type (3 cycles)	J-type (3 cycles)
Instruction fetch	IR ← Memory[PC]; PC ← PC+4			
Instr. decode/ Reg. fetch	A ← Reg(IR[21-25]); B ← Reg(IR[16-20]) ALUOut ← PC + (sign extend IR[0-15]) << 2			
Execution, addr. Comp., branch & jump completion	**ALUOut ← A op B**	**ALUOut ← A+sign extend (IR[0-15])**	**If (A= =B) then PC←ALUOut**	**PC←PC[28-31] || (IR[0-25]<<2)**
Mem. Access or R-type completion	**Reg(IR[11-15]) ← ALUOut**	**MDR←M[ALUout] or M[ALUOut]←B**		
Memory read completion		**Reg(IR[16-20]) ← MDR**		

CPI of a Computer

$$\text{CPI} = \frac{\sum_k (\text{Instructions of type k}) \times \text{CPI}_k}{\sum_k (\text{instructions of type k})}$$

where

CPI_k = Cycles for instruction of type k

Note: CPI is dependent on the instruction mix of the program being run. Standard benchmark programs are used for specifying the performance of CPUs.

Example

- Consider a program containing:
  - loads 25%
  - stores 10%
  - branches 11%
  - jumps 2%
  - Arithmetic 52%
- CPI = 0.25×5 + 0.10×4 + 0.11×3 + 0.02×3 + 0.52×4

  = 4.12 for multicycle datapath
- CPI = 1.00 for single-cycle datapath

Multicycle vs. Single-Cycle

$$\text{Performance ratio} = \frac{\text{Single cycle time}}{\text{Multicycle time}}$$

$$= \frac{(\text{CPI} \times \text{cycle time}) \text{ for single-cycle}}{(\text{CPI} \times \text{cycle time}) \text{ for multicycle}}$$

$$= \frac{1.00 \times 8\text{ns}}{4.12 \times 2\text{ns}} = 0.97$$

Single cycle is faster in this case, but remember, performance ratio depends on the instruction mix.

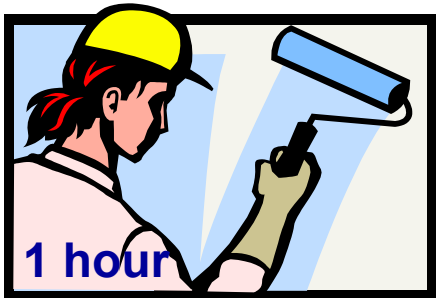
Traffic Flow

Single Lane Traffic

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.
  - Parallelism – Fetch multiple instructions every cycle.

Automobile Team Assembly



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

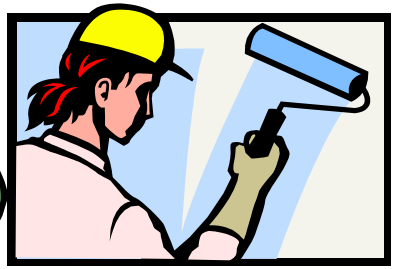
Automobile Assembly Line

Task 1
1 hour

Mecahnical

Task 2
1 hour

Electrical



Task 3
1 hour

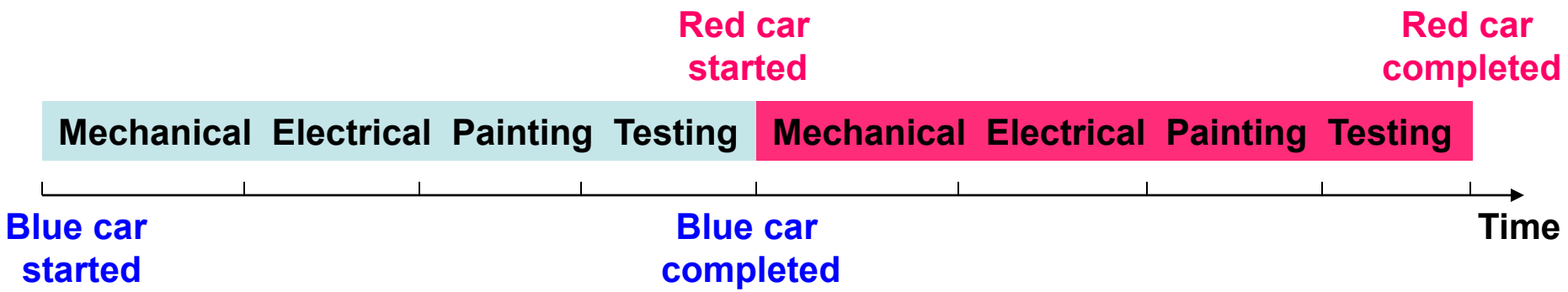
Painting

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

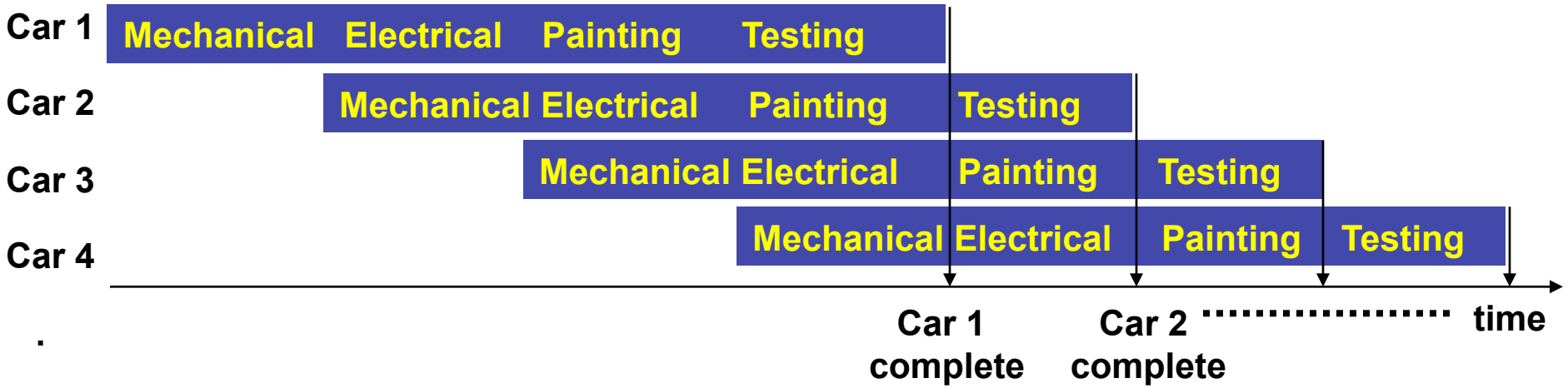


Time of assembling one car = n hours

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



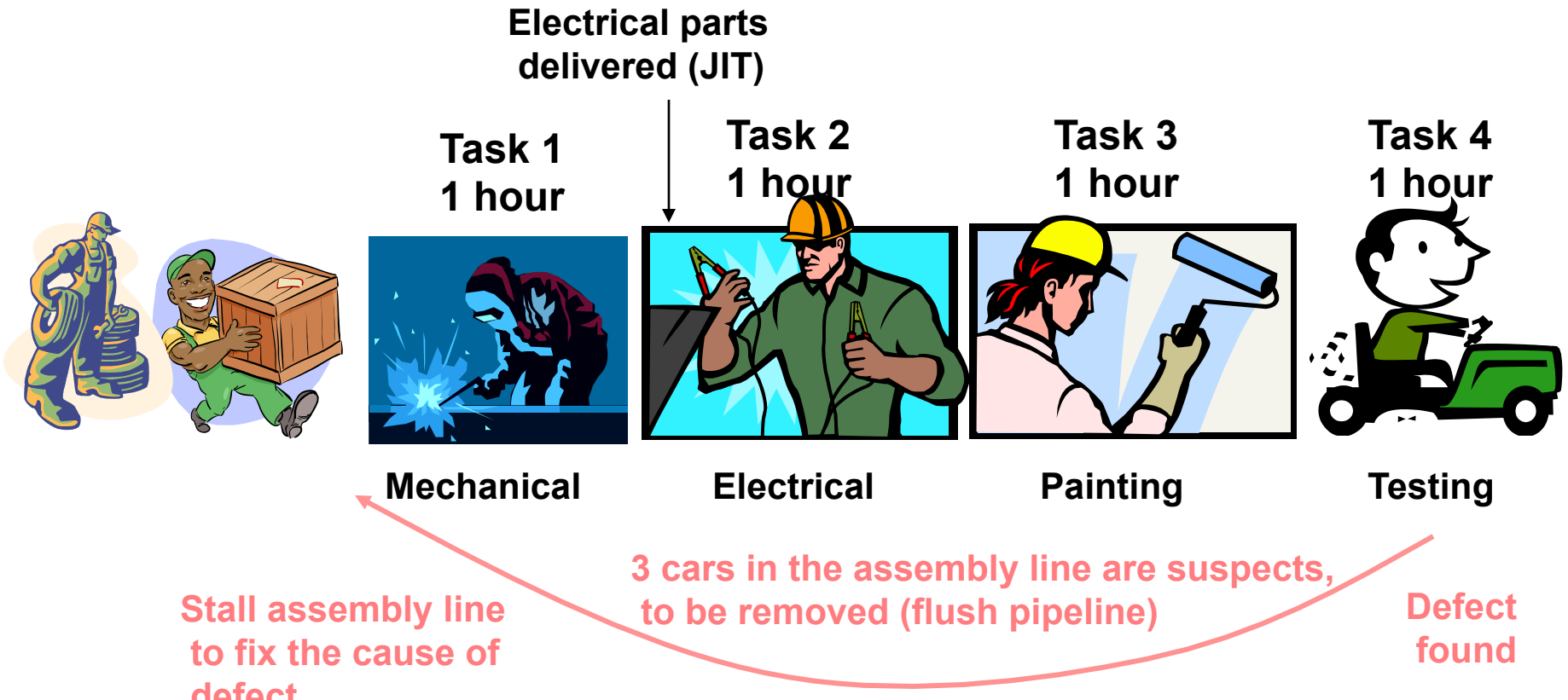
Time to complete first car = n time units (latency)

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



Pipelining in a Computer

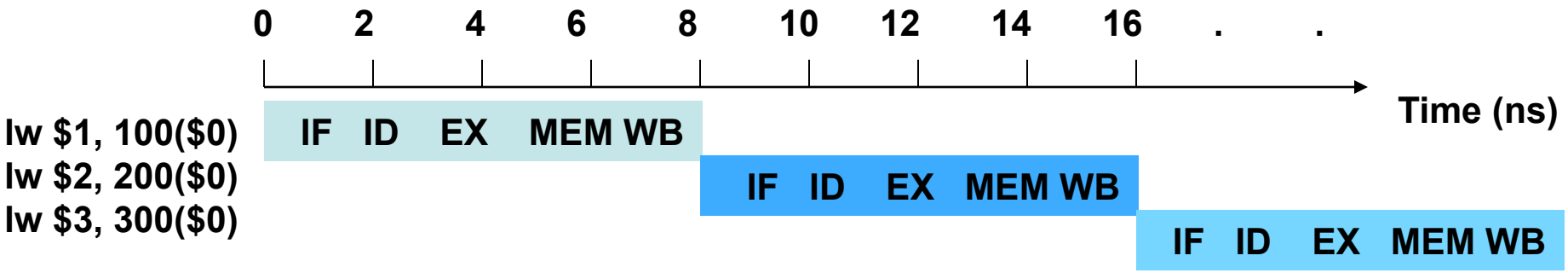
- Divide datapath into nearly equal tasks, to be performed serially and requiring non-overlapping resources.
- Insert registers at task boundaries in the datapath; registers pass the output data from one task as input data to the next task.
- Synchronize tasks with a clock having a cycle time that just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

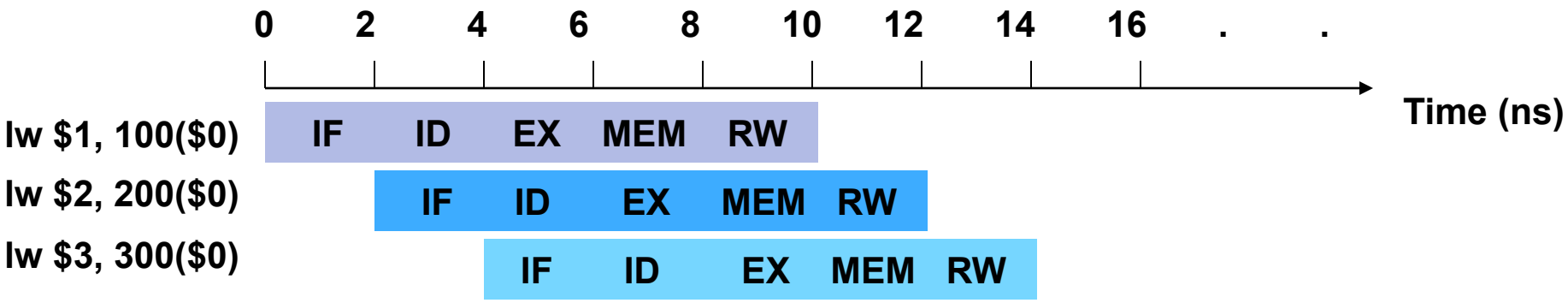
Total time for executing three lw instructions = 24 ns

Pipelined Datapath

Instruction class	Instr. fetch (IF)	Instr. Decode (also reg. file read) (ID)	Execu-tion (ALU Opera-tion) (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

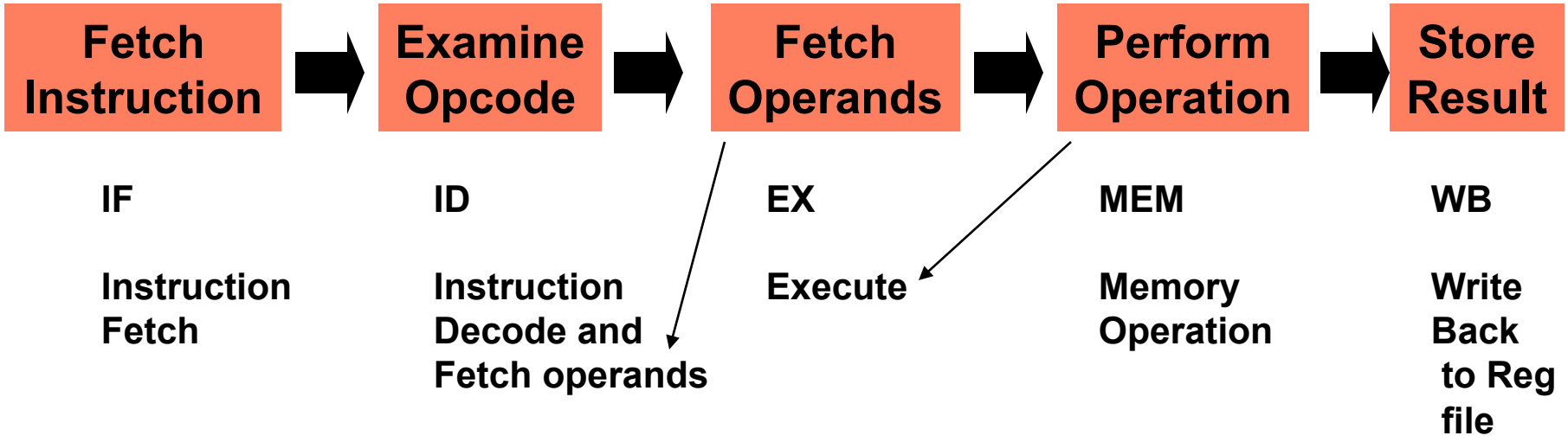
$$\text{Performance ratio} = \frac{\text{Single-cycle time}}{\text{Pipeline time}} = \frac{8{,}024}{2{,}014} = 3.98$$

10,003 *lw instructions:*

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

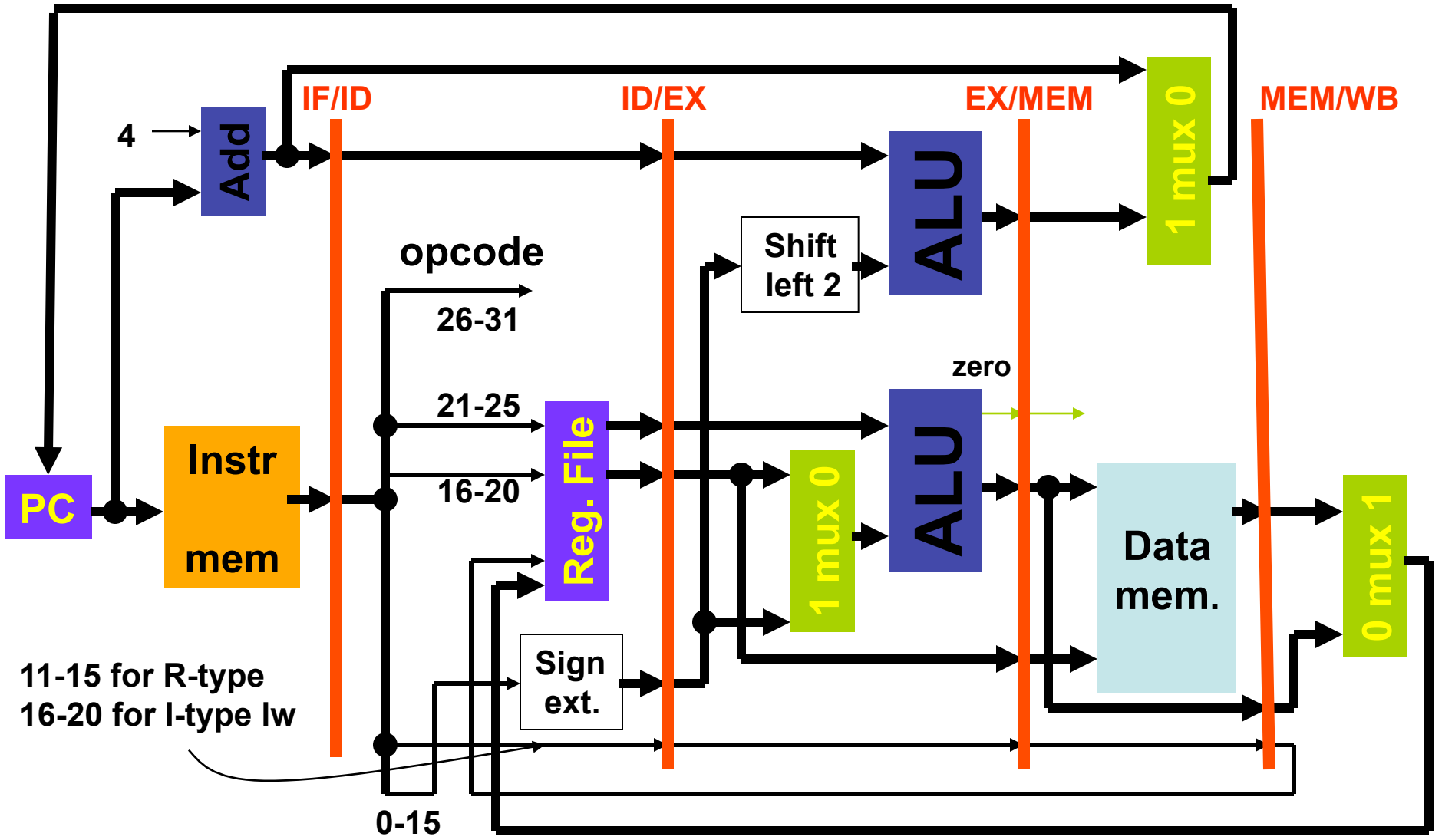
Pipeline performance approaches clock-cycle ratio for long programs.

Pipelining of RISC Instructions

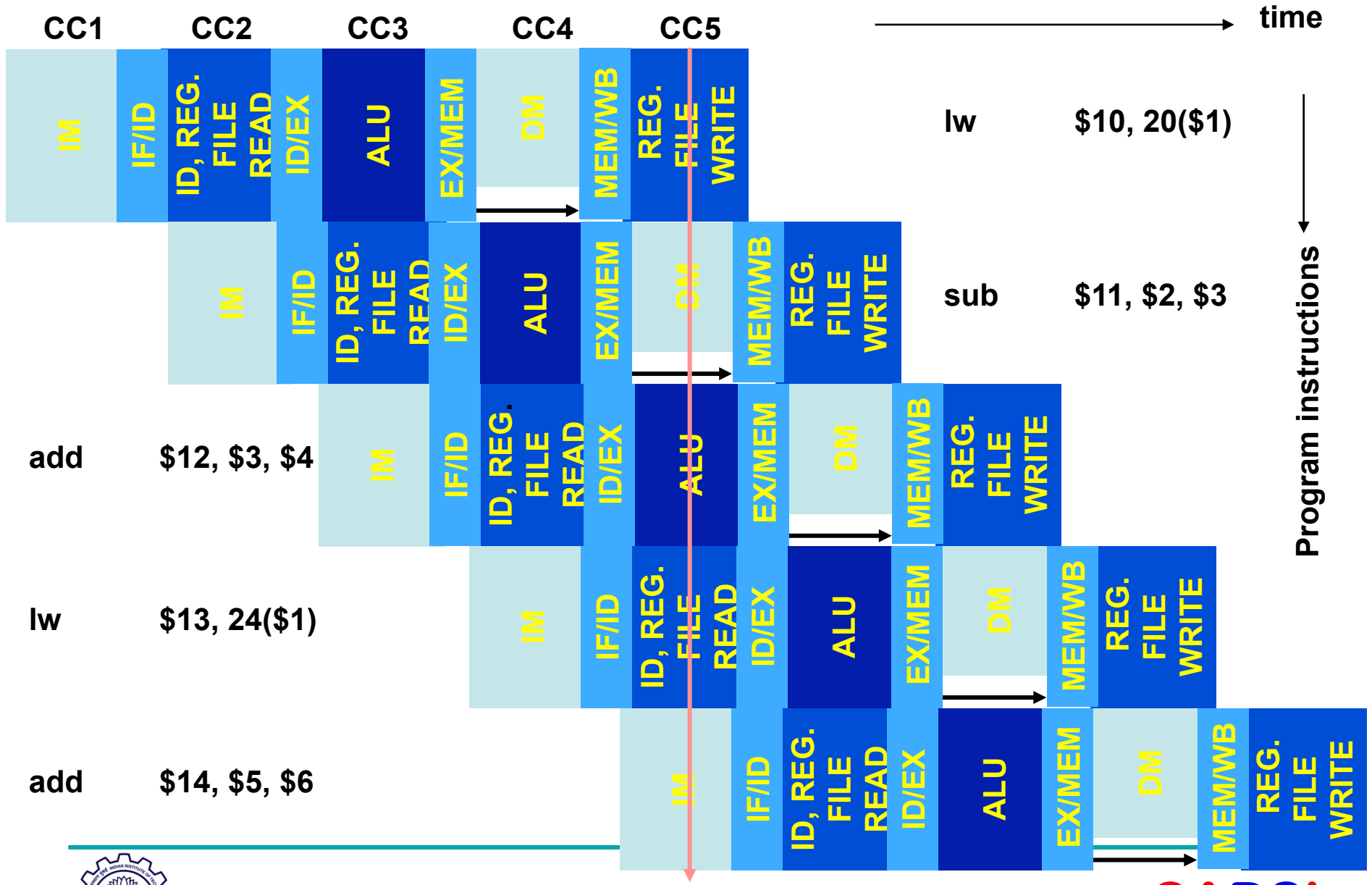


Although an instruction takes five clock cycles, one instruction is completed every cycle.

Pipelined Datapath



Program Execution



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 . . .

Science is always wrong. It never solves a problem without creating ten more.

George Bernard Shaw

Pipeline Hazards

- Definition: *Hazard in a pipeline is a situation in which the next instruction cannot complete execution one clock cycle after completion of the present instruction.*
- Three types of hazards:
  - Structural hazard (resource conflict)
  - Data hazard
  - Control hazard

Thank You