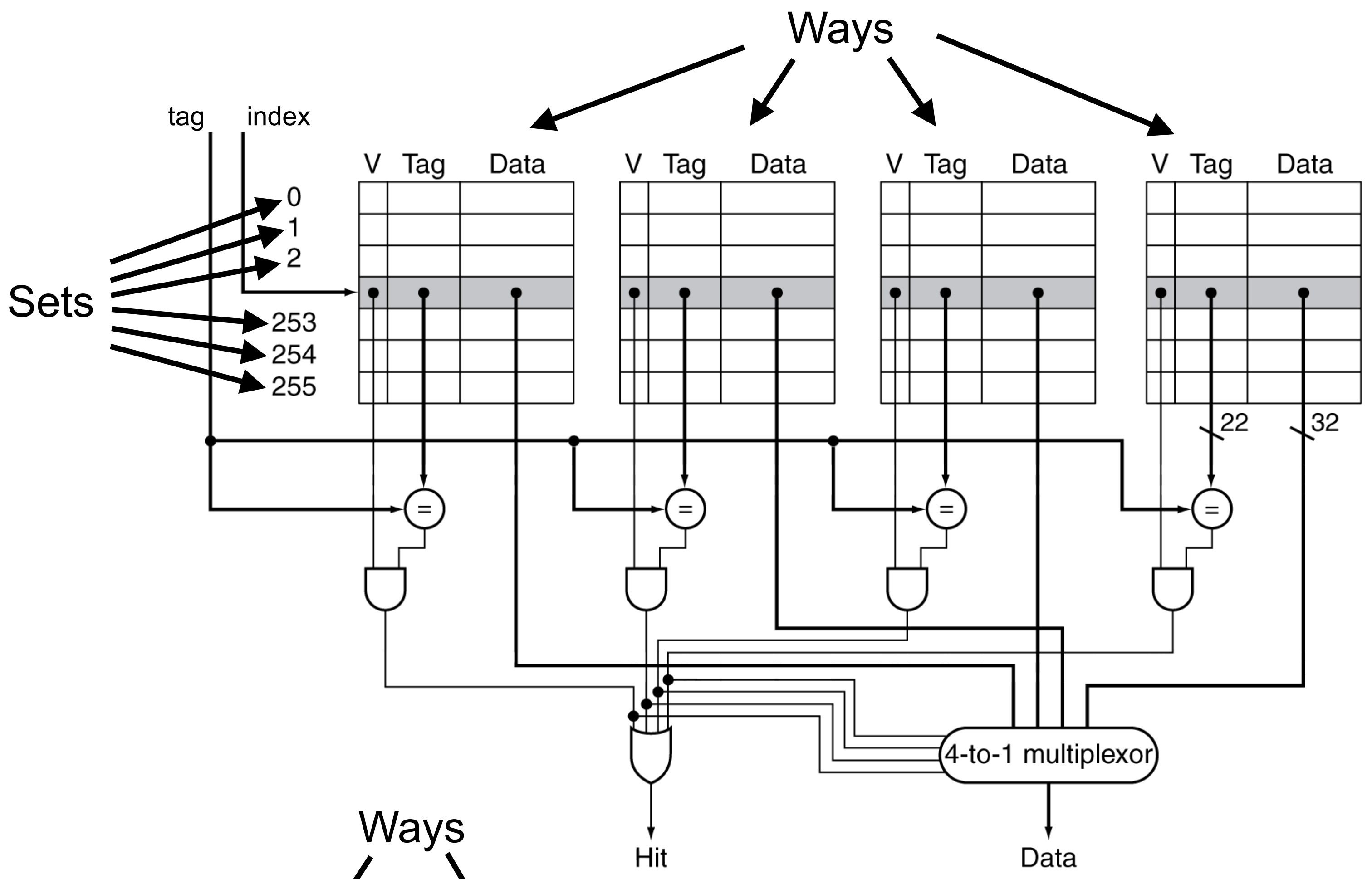
ECE 2300
Digital Logic & Computer Organization

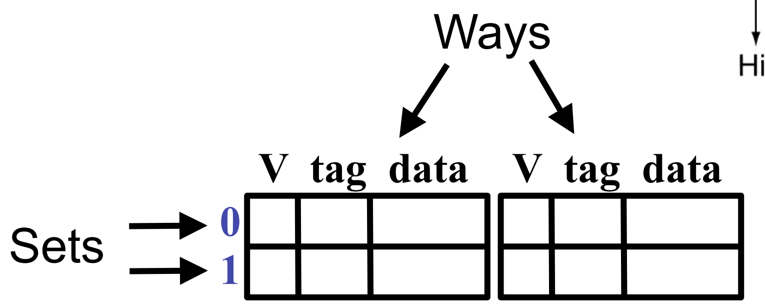
Fall 2016

Measuring Performance
Performance Tradeoffs

Cornell University

Set Associative Cache: Sets and Ways





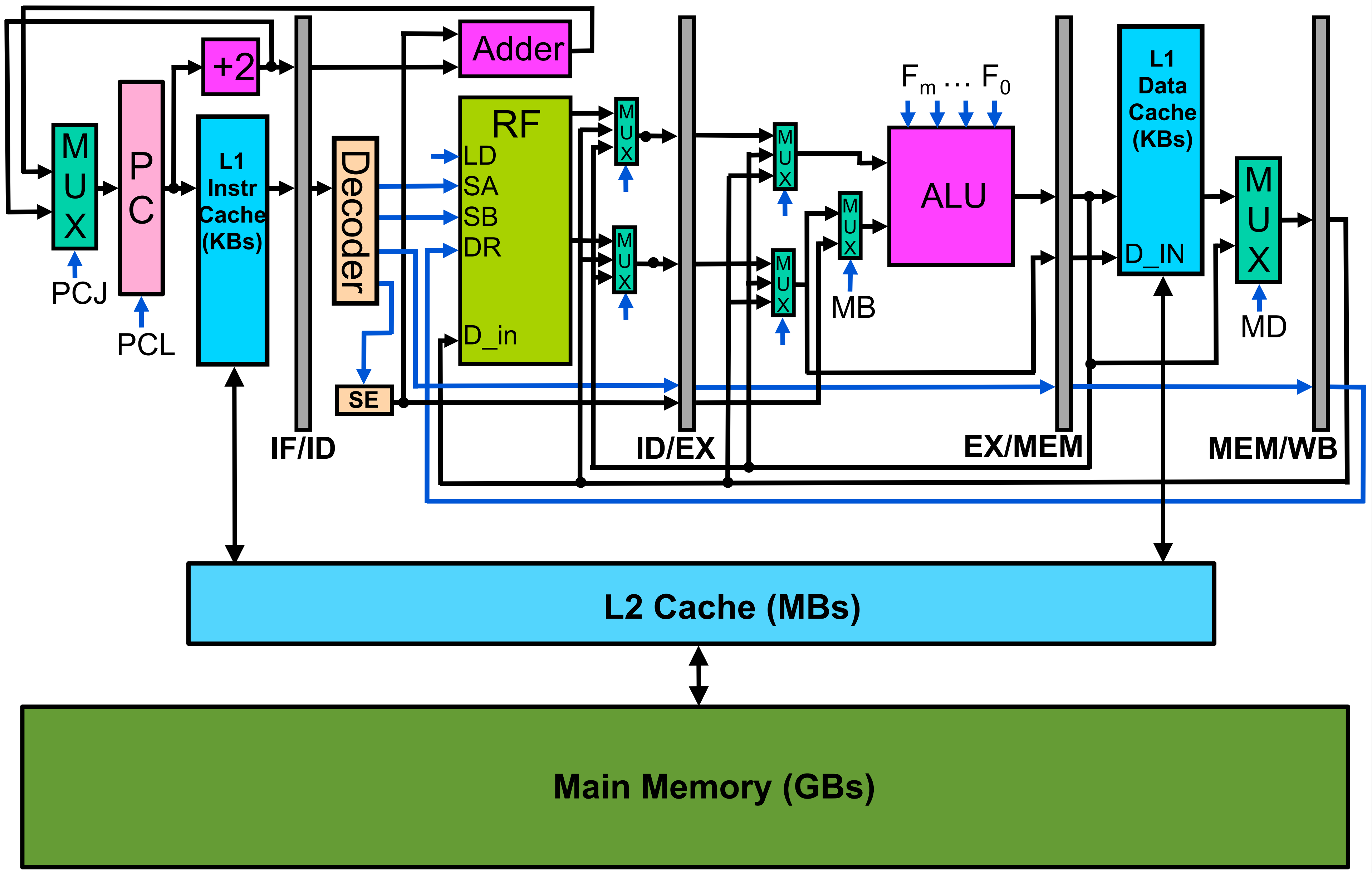
Set Associative Cache: Sets and Ways

Sets

Block address	Cache index	Hit/miss	Cache contents after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

Ways Ways

Pipeline + Memory Hierarchy



How Do We Measure Performance?

- Execution time: The time between the start and completion of a task
- Throughput (or bandwidth): Total amount of work done in a given time
- Improving performance means
  - Reducing execution time, or
  - Increasing throughput

CPU Execution Time

- **Amount of time the CPU takes to run a program**

- **Derivation**

$$\text{CPU execution time} = \text{clock cycles} \times \text{clock cycle time}$$

$$\text{clock cycles} = \text{number of instructions} \times \text{cycles per instruction}$$

$$\Rightarrow \text{CPU execution time} = \underbrace{\text{I}}_{\text{number of instructions in the program}} \times \underbrace{\text{CPI}}_{\text{average number of cycles per instruction}} \times \underbrace{\text{CT}}_{\text{clock cycle time (1/frequency)}}$$

number of instructions in the program

clock cycle time (1/frequency)

average number of cycles per instruction

Instruction Count (I)

- Total number of instructions in the program

- Factors impacting I
  - Instruction set
  - Mix of instructions chosen by the compiler

Cycle Time (CT)

- Clock period (1/frequency)

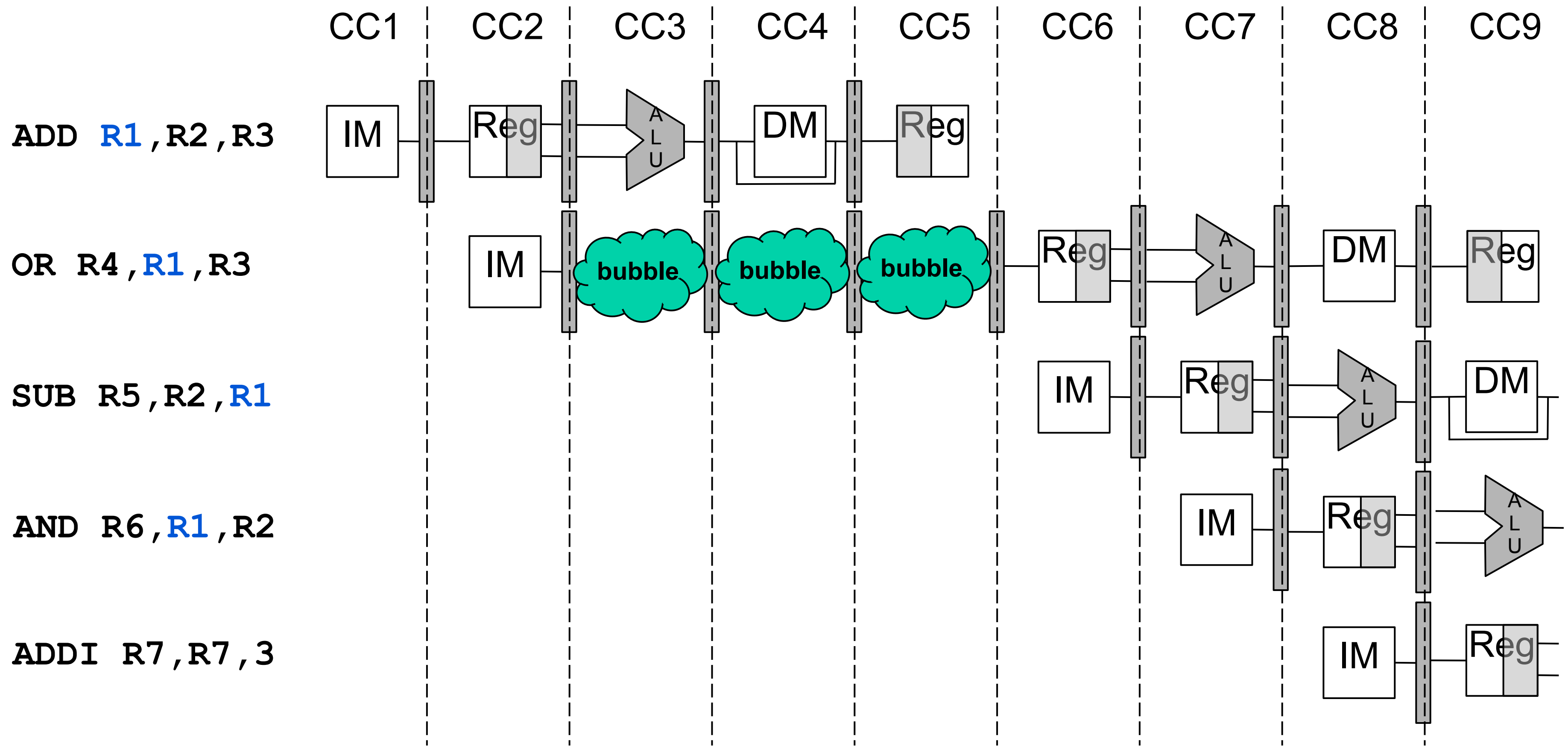
- Factors impacting CT
  - Instruction set
  - Structure of the processor and memory hierarchy

Cycles Per Instruction (CPI)

- Average number of cycles required to execute each instruction

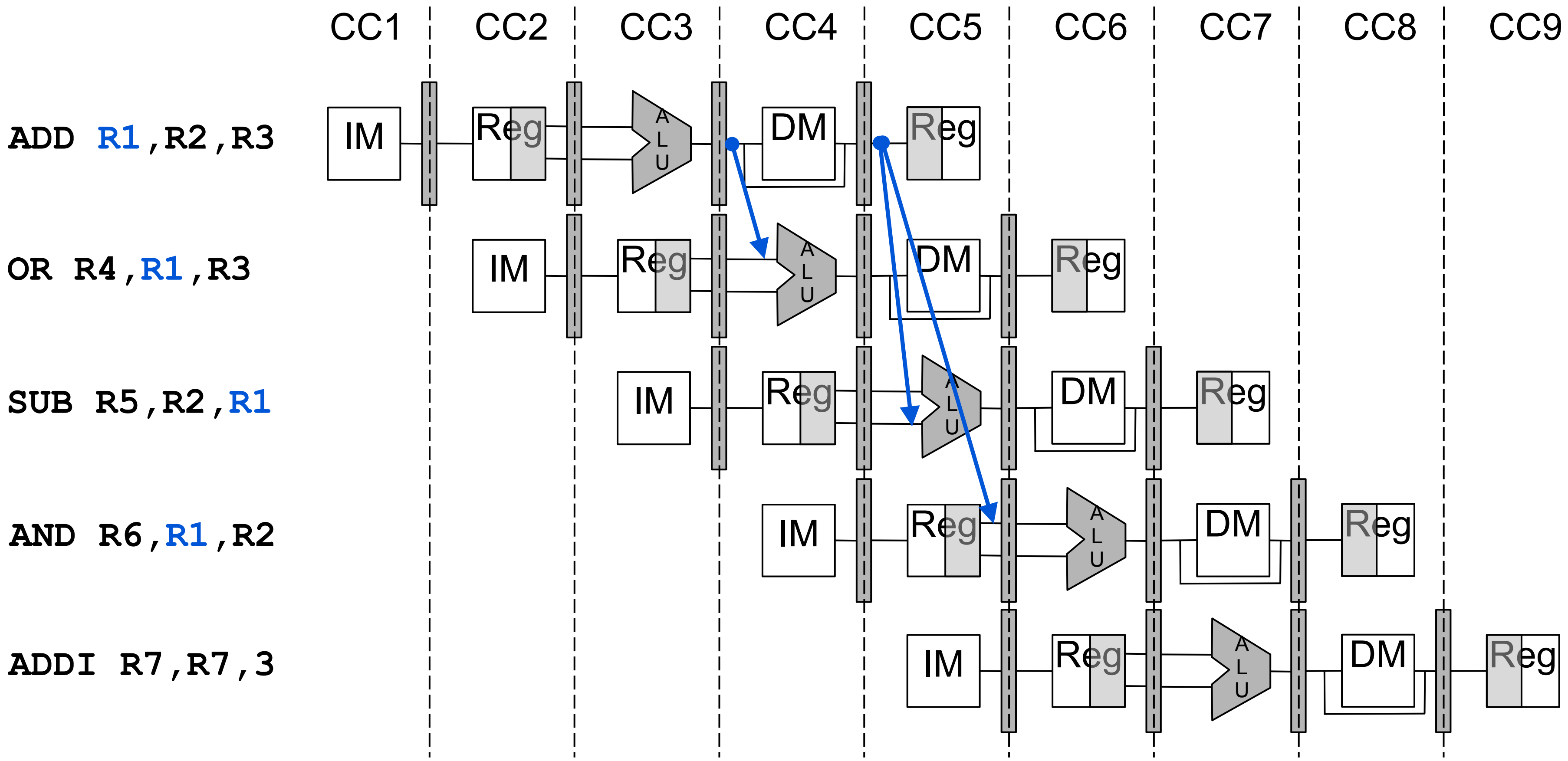
- Factors impacting CPI
  - Instruction set
  - Mix of instructions chosen by the compiler
  - Ordering of the instructions by the compiler
  - Structure of the processor and memory hierarchy

Processor Organization Impact on CPI (Example 1)



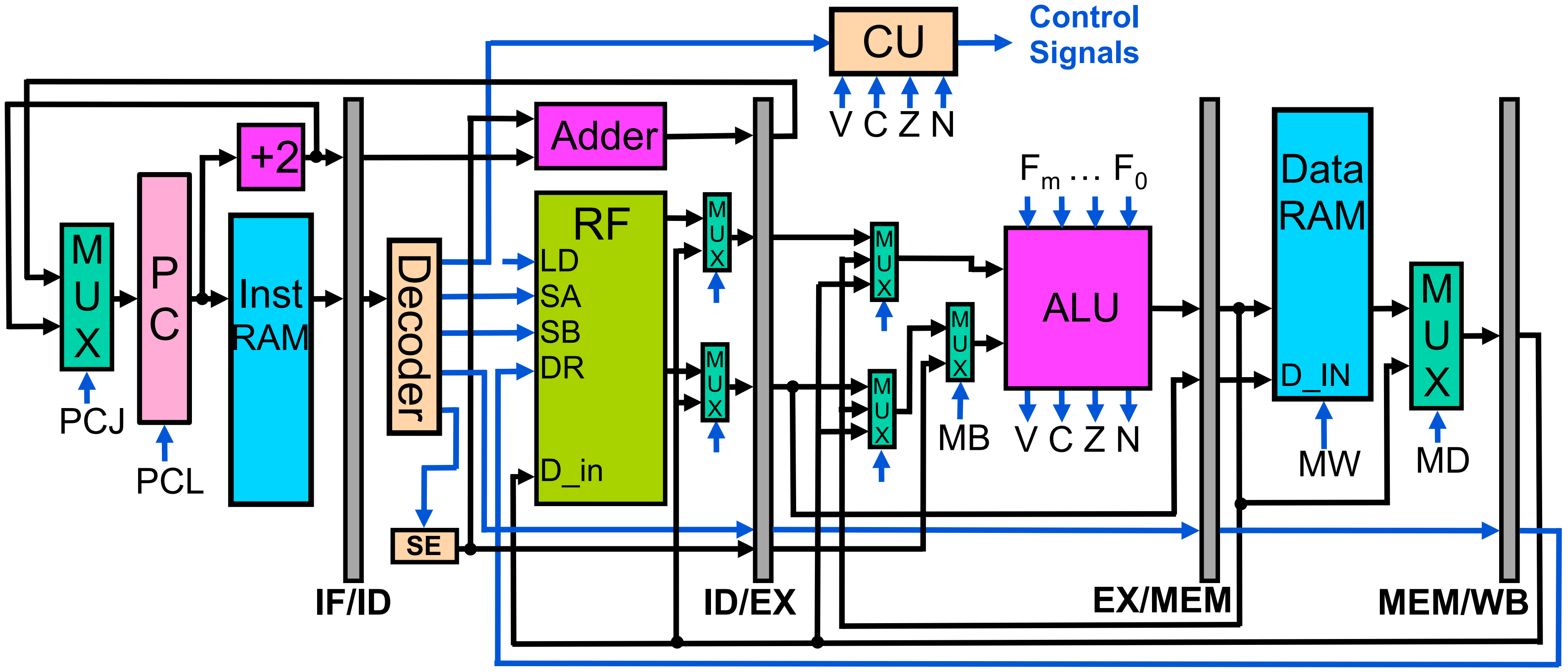
Without forwarding: OR instruction remains in the IF stage for three additional cycles ⇒ stall cycles

Processor Organization Impact on CPI (Example 1)



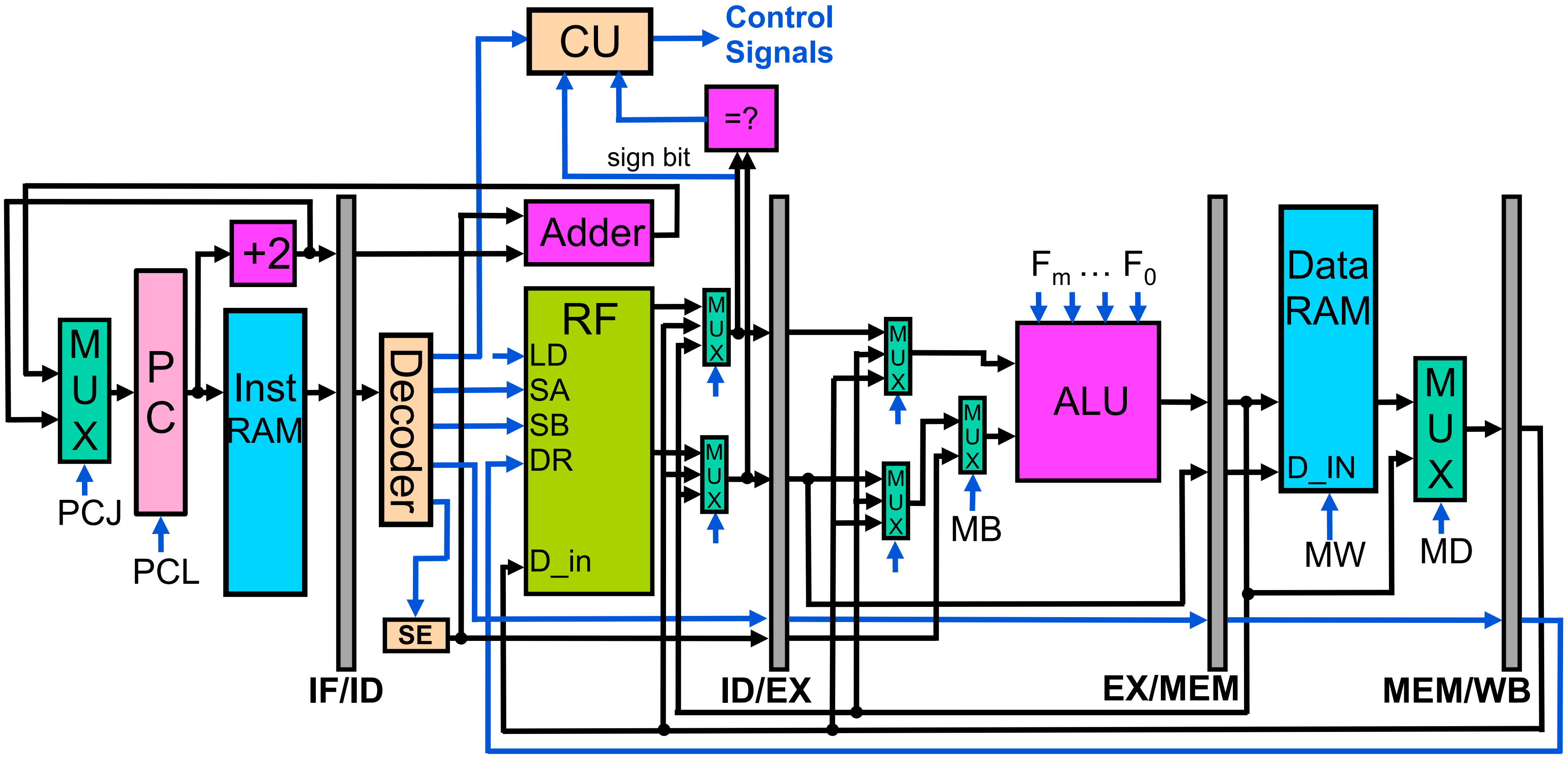
With forwarding: No stall cycles
Lower CPI, reduced execution time

Processor Organization Impact on CPI (Example 2)



Branch resolved in EX
Two instructions thrown away if the branch is taken

Processor Organization Impact on CPI (Example 2)

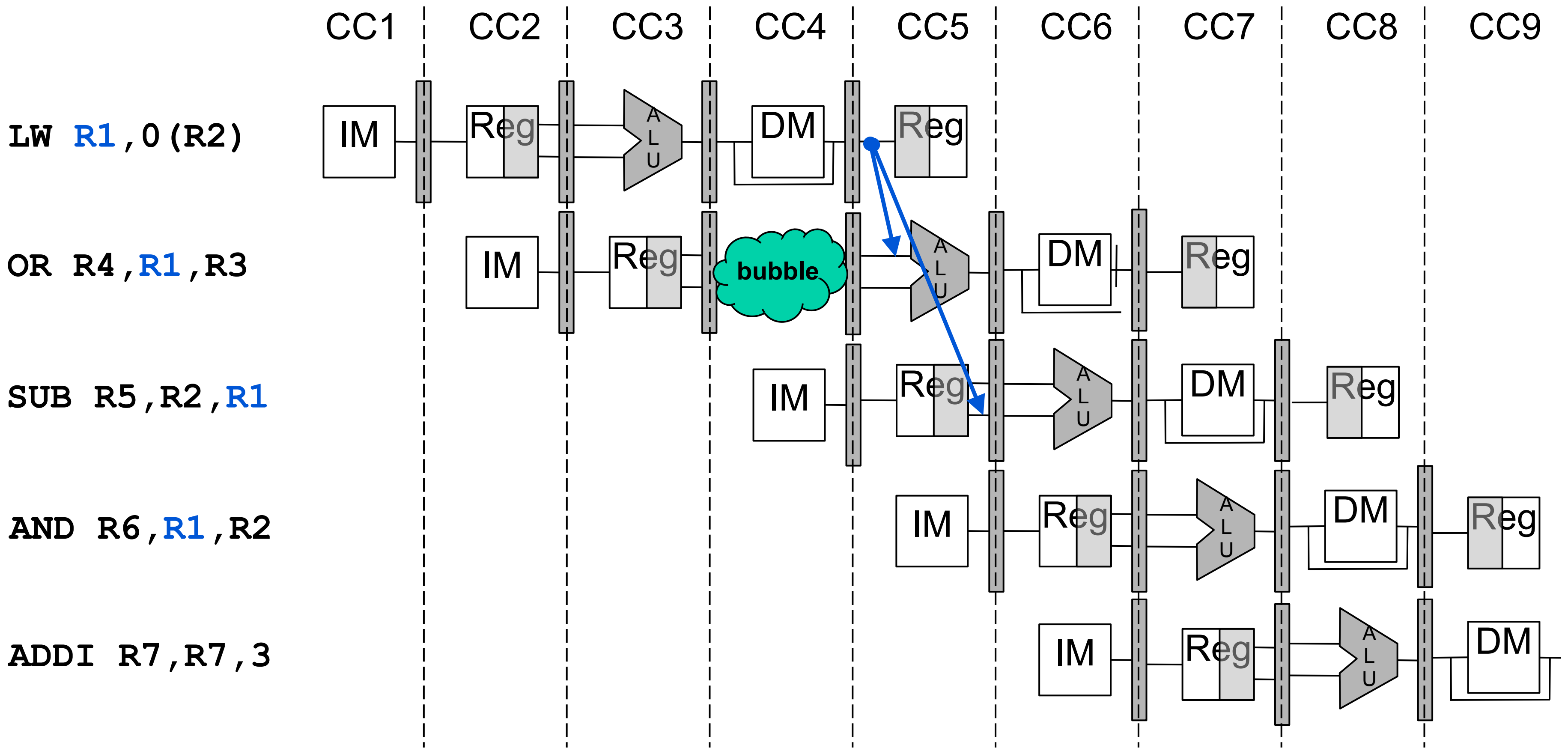


Branch resolved in ID

One instruction thrown away if the branch is taken

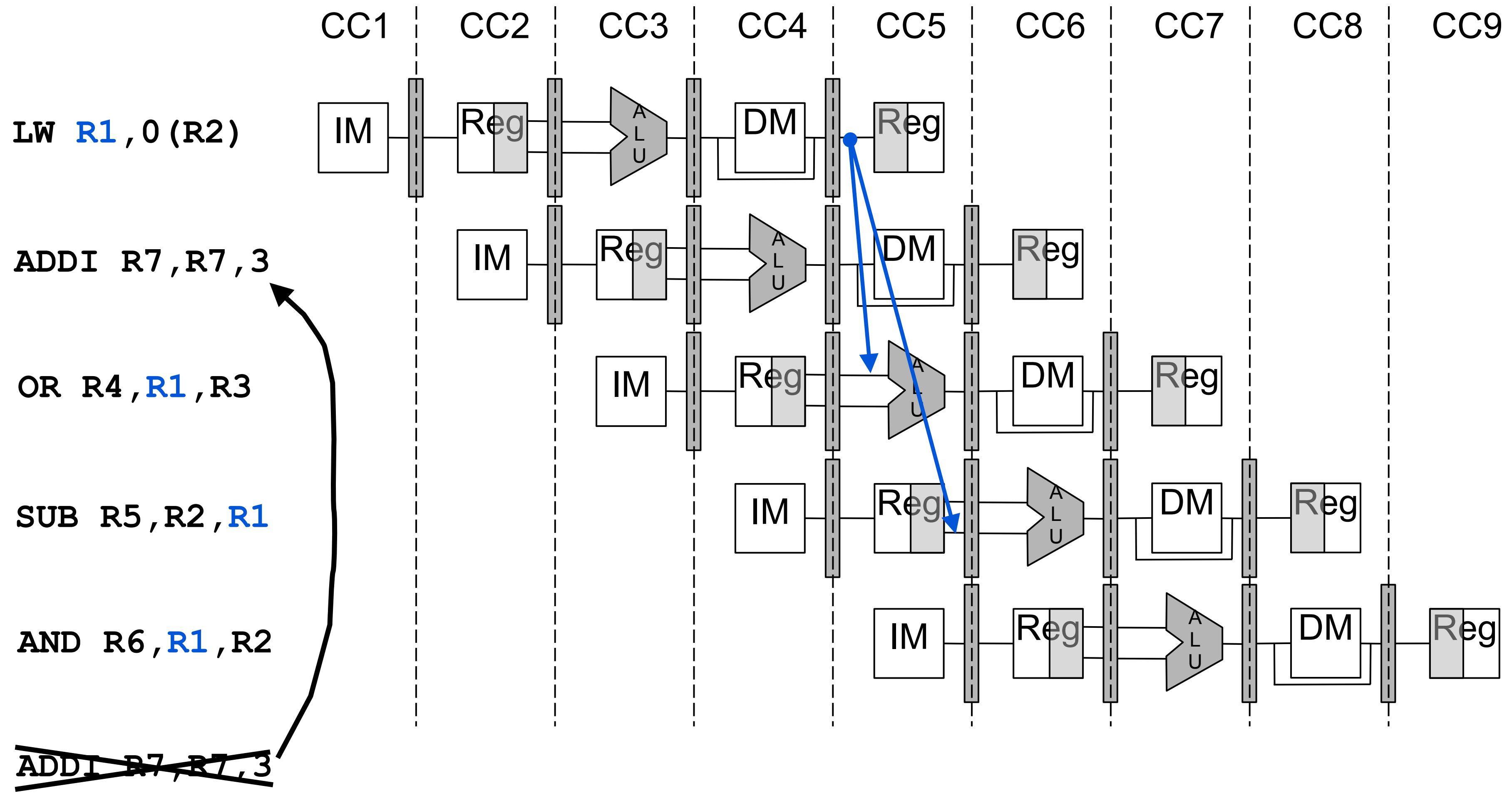
Lower CPI, faster execution time

Compiler Impact on CPI (Example 1)



Data-dependent OR instruction placed immediately after LW ⇒ stall cycle

Compiler Impact on CPI (Example 1)



Independent ADDI instruction moved in between LW and data-dependent OR instruction

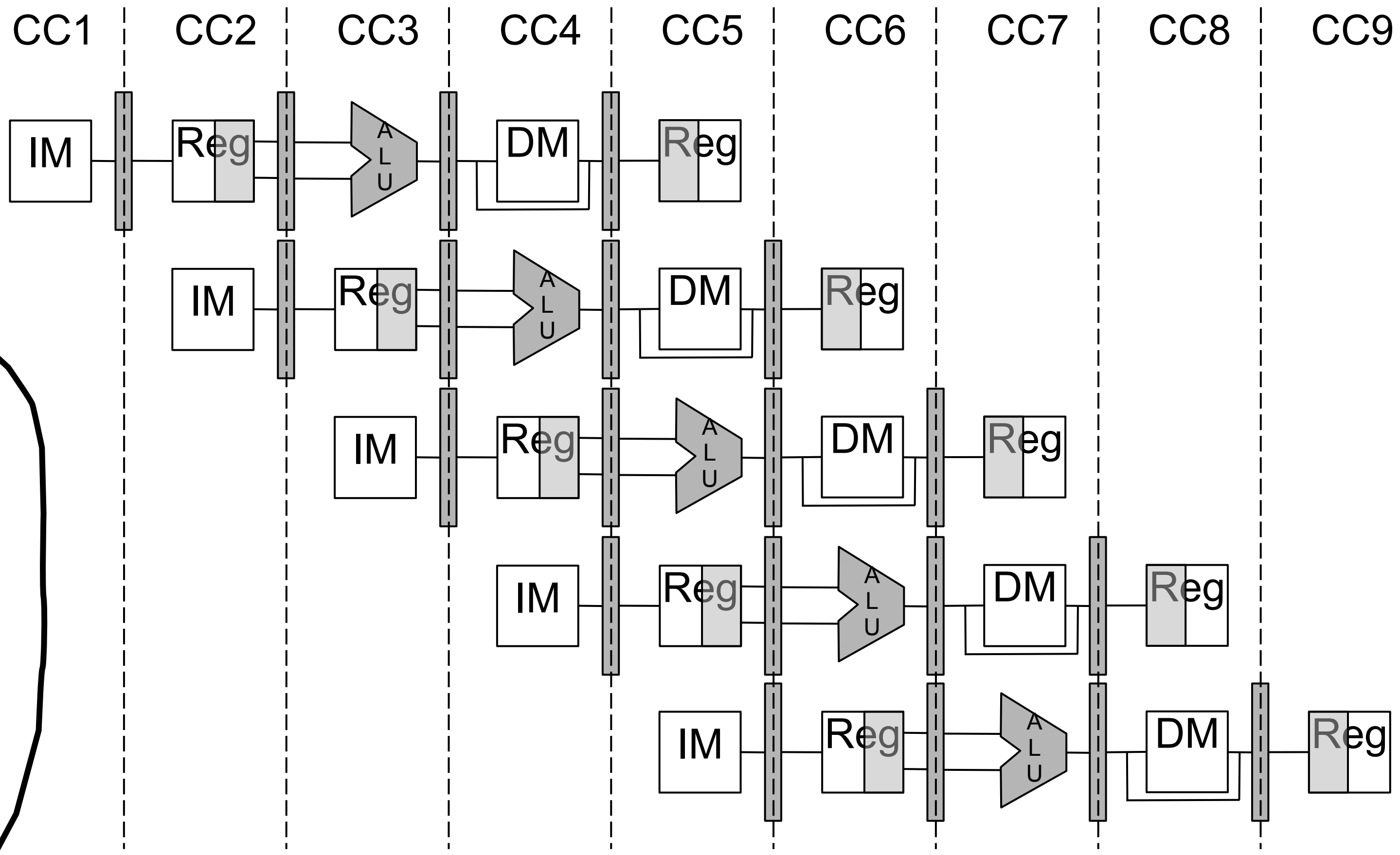
No stalls, lower CPI, faster execution time

Compiler Impact on CPI (Example 2)

```
BEQ R2,R3,X
ADDI R7,R7,3
OR R4,R1,R3
SUB R5,R2,R1
AND R6,R1,R2
```

~~ADDI R7,R7,3~~

```
    .
    .
    .
X: SRL R4,R5
```



Filling the branch delay slot with a useful instruction

Impact of the Memory Hierarchy

- In an ideal memory hierarchy, instruction fetches and data memory accesses incur no extra delay
  - $CPI_{baseline}$ is the CPI with this ideal memory hierarchy
- With no caches (only main memory)
  - Every instruction is read from main memory
  - Every load and store instruction accesses main memory
- Assume
  - 100 cycles to access main memory
  - 25% of all instructions are loads, 10% are stores
- $CPI_{memhier} = 100 + 0.35 \times 100 = 135$ (!)

Impact of the Memory Hierarchy

- With L1 caches
  - L1 instruction cache miss rate = 2%
  - L1 data cache miss rate = 4%
  - Miss penalty = 100 cycles (access main memory)
  - 25% of all instructions are loads, 10% are stores

- $CPI_{memhier} = 0.02 \times 100 + 0.35 \times 0.04 \times 100 = 3.4$

Impact of the Memory Hierarchy

- With L1 and L2 caches
  - L1 instruction cache miss rate = 2%
  - L1 data cache miss rate = 4%
  - L2 access time = 15 cycles
  - L2 miss rate = 25%
  - L2 miss penalty = 100 cycles (to access main memory)
  - 25% of all instructions are loads, 10% are stores

- $\mathrm{CPI}_{\mathrm{memhier}} = 0.02 \times (15 + 0.25 \times 100) + 0.35 \times 0.04 \times (15 + 0.25 \times 100) = 1.36$

Relative Performance

- Used to compare the performance of machines
- Used to report the performance benefit [loss] of adding [subtracting] a feature

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X}$$

Relative Performance Example

- Relative $\text{CPI}_{\text{memhier}}$ of memory hierarchy alternatives

$$\frac{\text{Performance}_{\text{L1}}}{\text{Performance}_{\text{no caches}}} = \frac{\text{CPI}_{\text{memhier}_{\text{no caches}}}}{\text{CPI}_{\text{memhier}_{\text{L1}}}} = \frac{135}{3.4} = 39.7$$

$$\frac{\text{Performance}_{\text{L1+L2}}}{\text{Performance}_{\text{L1}}} = \frac{\text{CPI}_{\text{memhier}_{\text{L1}}}}{\text{CPI}_{\text{memhier}_{\text{L1+L2}}}} = \frac{3.4}{1.36} = 2.5$$

Relative Performance Example

- Relative $\text{CPI}_{\text{total}}$ of memory hierarchy alternatives
  - Assume $\text{CPI}_{\text{baseline}} = 1.5$

$$\frac{\text{Performance}_{\text{L1}}}{\text{Performance}_{\text{no caches}}} = \frac{\text{CPI}_{\text{total}_{\text{no caches}}}}{\text{CPI}_{\text{total}_{\text{L1}}}} = \frac{1.5+135}{1.5+3.4} = 27.9$$

$$\frac{\text{Performance}_{\text{L1+L2}}}{\text{Performance}_{\text{L1}}} = \frac{\text{CPI}_{\text{total}_{\text{L1}}}}{\text{CPI}_{\text{total}_{\text{L1+L2}}}} = \frac{1.5+3.4}{1.5+1.36} = 1.7$$

Amdahl's Law

- Performance improvement possible with a given enhancement is limited by the amount that the enhancement is used

$$\text{Execution Time}_{\text{enhanced}} = \frac{\text{Execution Time affected}}{\text{Amount of improvement}} + \text{Execution Time unaffected}$$

- Example
  - Feature improves multiply operations by factor of 10
  - Total execution time of a program is 100 sec
  - Multiply operations consume 5 sec of the total

$$\text{Execution Time}_{\text{enhanced}} = \frac{5}{10} + 95 = 95.5\,\text{sec}$$

Performance Tradeoffs

$$\text{CPU execution time} = \text{I} \times \text{CPI} \times \text{CT}$$

- A decision regarding the ISA or processor organization often improves one aspect of CPU execution time at the expense of another
  - I versus CPI
  - I versus CT
  - CPI versus CT

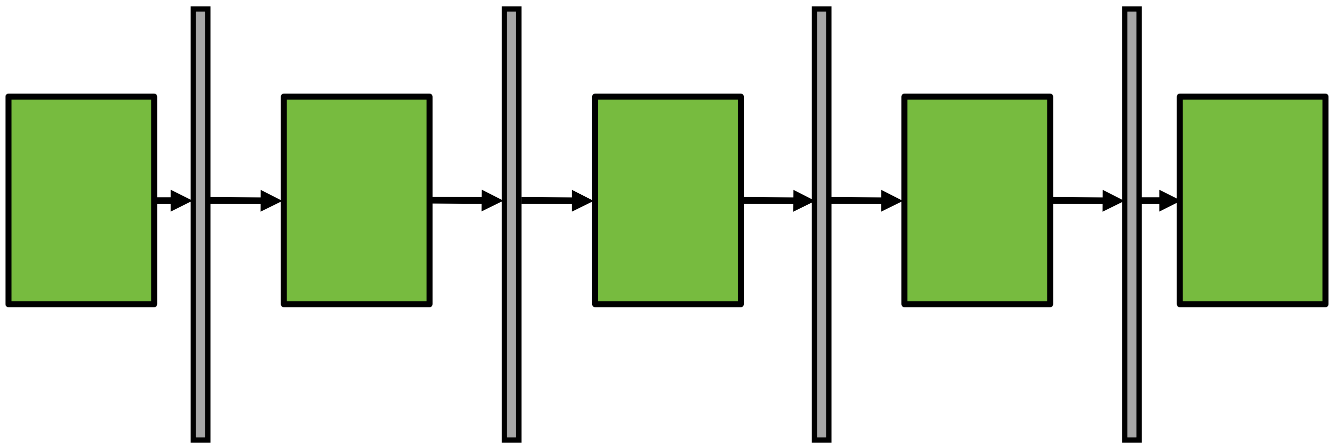
Performance Tradeoff Example 1

```
ADD 8(R1),0(R1),4(R1)
```

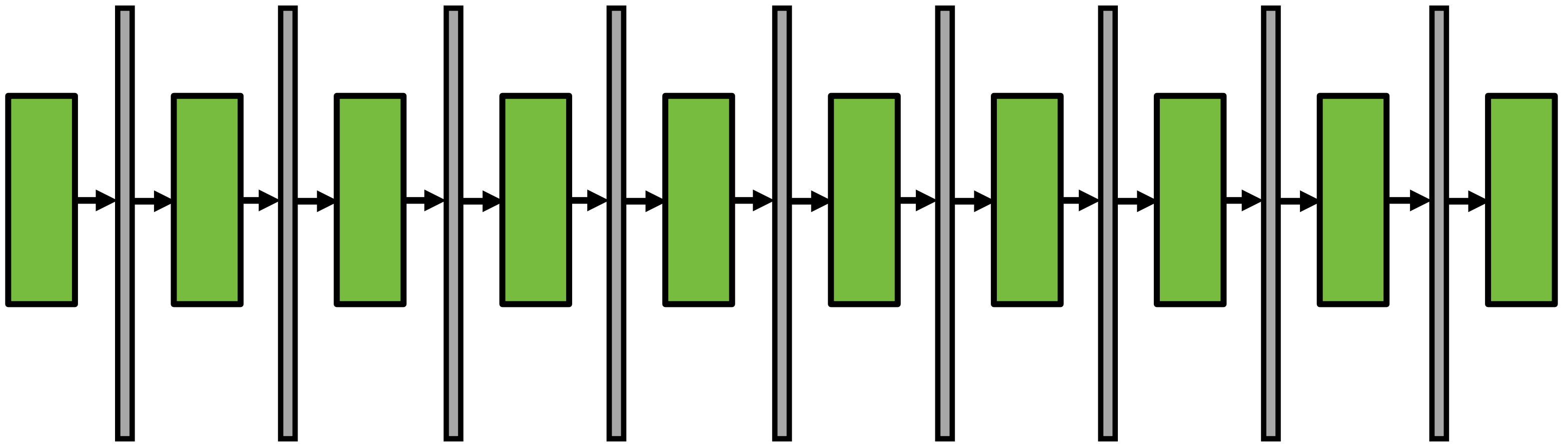
vs.

```
LW    R2,0(R1)
LW    R3,4(R1)
ADD  R2,R2,R3
SW   R2,8(R1)
```

Performance Tradeoff Example 2



vs.



Performance Tradeoff Example 3

- Increase the size of the L1 caches?
  - Reduces miss rate $\Rightarrow$ lower CPI
  - Increases hit time $\Rightarrow$ higher CT or higher CPI

- Example
  - L1 cache miss rate drops from 4% to 2%
  - L1 cache hit time increases by 1 cycle
  - Miss penalty = 40 cycles
  - $CPI_{old} = 1.5 + 0.04 \times 40 + 0.35 \times 0.04 \times 40 = 3.66$
  - $CPI_{new} = 1.5 + (1 + 0.02 \times 40) + 0.35 \times (1 + 0.02 \times 40) = 3.93$

Before Next Class

- H&H 8.4

Next Time

Virtual Memory