ECE411: Computer Organization and Design

Lecture 3: Performance, Energy, and Power Metrics

Rakesh Kumar

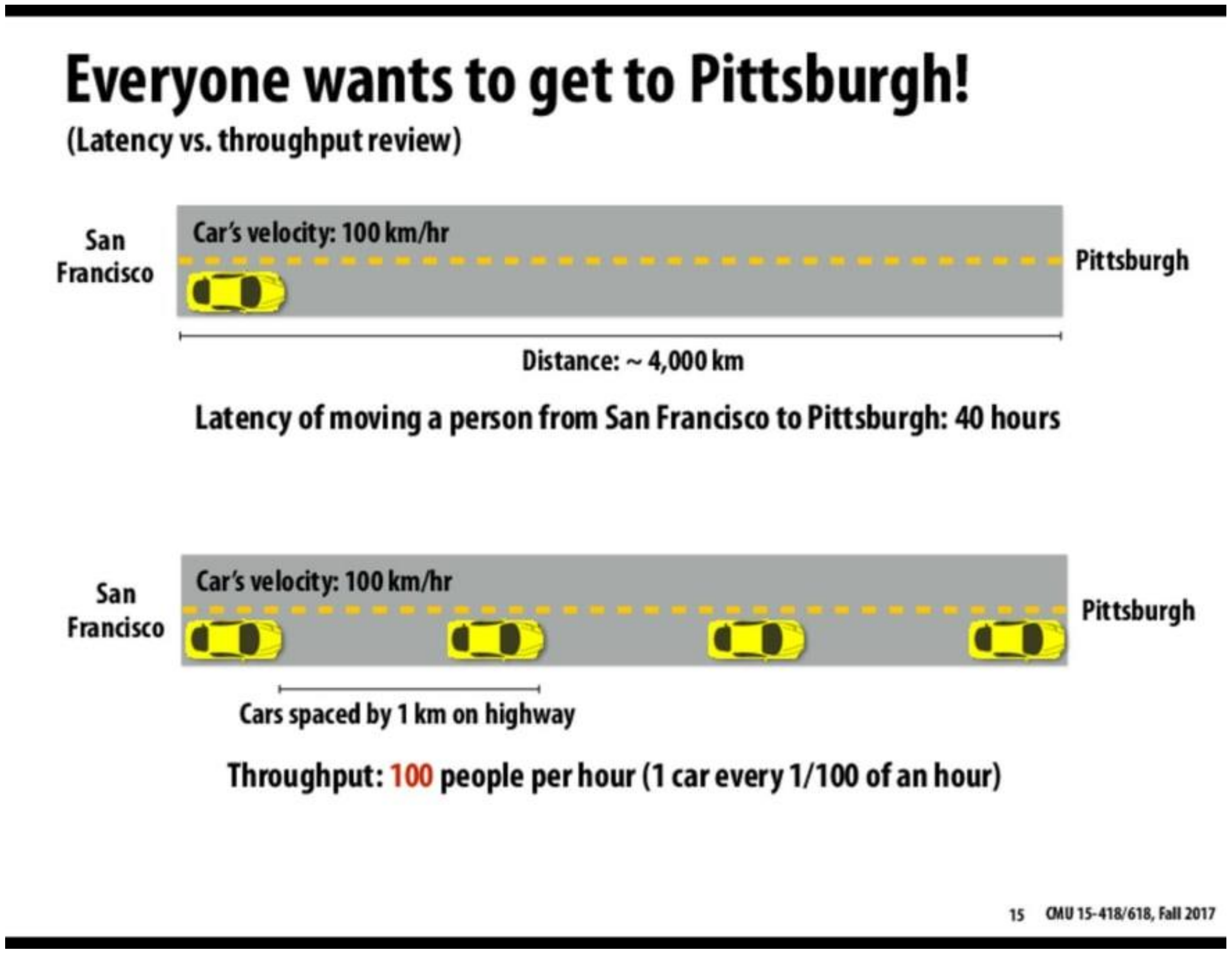
ILLINOIS

Computer Performance: time!, time!, time!

- latency (response time)
  - how long does it take to execute a task?
  - how long must i wait for the database query?
  - high-percentile response time (SLO, SLA, etc. at datacenters)

- throughput
  - how many jobs can the machine complete in a minute?
  - what is the average execution rate?
  - how much work is getting done?

Computer Performance: time!, time!, time!



Everyone wants to get to Pittsburgh!

(Latency vs. throughput review)

San Francisco — Car's velocity: 100 km/hr — Pittsburgh

Distance: ~ 4,000 km

Latency of moving a person from San Francisco to Pittsburgh: 40 hours

San Francisco — Car's velocity: 100 km/hr — Pittsburgh

Cars spaced by 1 km on highway

Throughput: 100 people per hour (1 car every 1/100 of an hour)

15 CMU 15-418/618, Fall 2017

Source: [Latency vs. Throughput (DEV Community)](#)

How Do We Quantify CPU Performance?

- CPU execution time = seconds / program
  - time the CPU spends working on a task
  - does not include time waiting for I/O or running other programs

$$\frac{\text{Instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{Instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

$$\updownarrow \qquad\qquad \updownarrow \qquad\qquad \updownarrow$$

- programmer
- algorithms
- ISA
- compilers

- microarchitecture
- system architecture

- microarchitecture, pipeline depth
- circuit design
- Semiconductor technology

CISC vs. RISC

- Design difference between CISC vs. RISC

$$\frac{\text{Instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{Instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

- Share the same goal of improving program performance, but
  - RISC: focuses on reducing the cycles per instruction
  - CISC: focuses on reducing the number of instructions per program

Performance Metric

- Metric #1: time to complete a task (Texe)
  - execution time, response time, latency
    - X is N time faster than Y means Texe(Y)/Texe(X) = N
  - example:
    - machine A runs a program in 20 seconds
    - machine B runs the same program in 25 seconds
    - A is ________ times faster than B?

Performance Metric

- Metric #2: # of tasks per day, hour, seconds,
  - throughput or bandwidth
  - not the same as latency
    - Multi-core processors improve throughput but not latency

- Examples of other metrics
  - millions instructions per second (MIPS)
  - millions floating-point operations per second (MFLOPS)
  - Trillions (10^{12}) of operations per second (TOPS)

How to Improve Performance

- to improve performance (everything else being equal) you can either

  __________ the # of required cycles for a program, or
  __________ the clock cycle time or, said another way,
  __________ the clock rate.

An example: if a computer has a clock cycle time of 5 ns, the clock rate is: 1/(5 x 10^{-9} sec) = 200 MHz

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

Performance Related Metrics

- an execution of a given program will require
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds
- we have a vocabulary of metrics that relate these quantities:
  - cycle time (seconds per cycle)
  - clock rate (cycles per second)
  - **CPI (cycles per instruction)**
- instruction set metrics

  Q: if two machines have the same ISA and run the same program

  which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will likely be identical during a comparison?

Performance Examples

1. Suppose we have two implementations of the same ISA, what machine is faster for a given program, and by how much?
   - machine A w/ a clock cycle time of 10ns and a CPI of 2.5
   - machine B w/ a clock cycle time of 20ns and a CPI of 1

Performance Examples

2. Consider two code sequences for a given machine w/ 3 different classes of instructions: class A, B, and C requiring 1, 2, and 3 cycles per instruction, which sequence will be faster?

- first code sequence w/ 5 instructions (2 of A, 1 of B, and 2 of C)
- second code sequence w/ 6 instructions (4 of A, 1 of B, and 1 of C)

Performance Examples

3. Consider two compilers and a machine operating at 100MHz w/ 3 classes of instructions: class A, B, and C requiring 1, 2, and 3 cycles per instruction, which compiler generates faster running code?

- first compiler's code uses 5M Class A instructions, 1M Class B instructions, and 1M Class C instructions
- second compiler's code uses 10M Class A instructions, 1M Class B instructions, and 1M Class C instructions

Effective CPI

- computing the overall effective CPI is done by looking at the different types of instructions and their individual cycle counts and averaging
    - $$\text{Overall effective CPI} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{IC}_i)$$
    - where IC_i is the count (percentage) of the number of instructions of class i executed
    - CPI_i is the (average) number of clock cycles per instruction for that instruction class
    - n is the number of instruction classes

Effective CPI: Exercise

op	freq	CPI_i	freq x CPI_i
ALU	50%	1	
load	20%	5	
store	10%	3	
branch	20%	2	
		overall effective CPI	$\sum =$

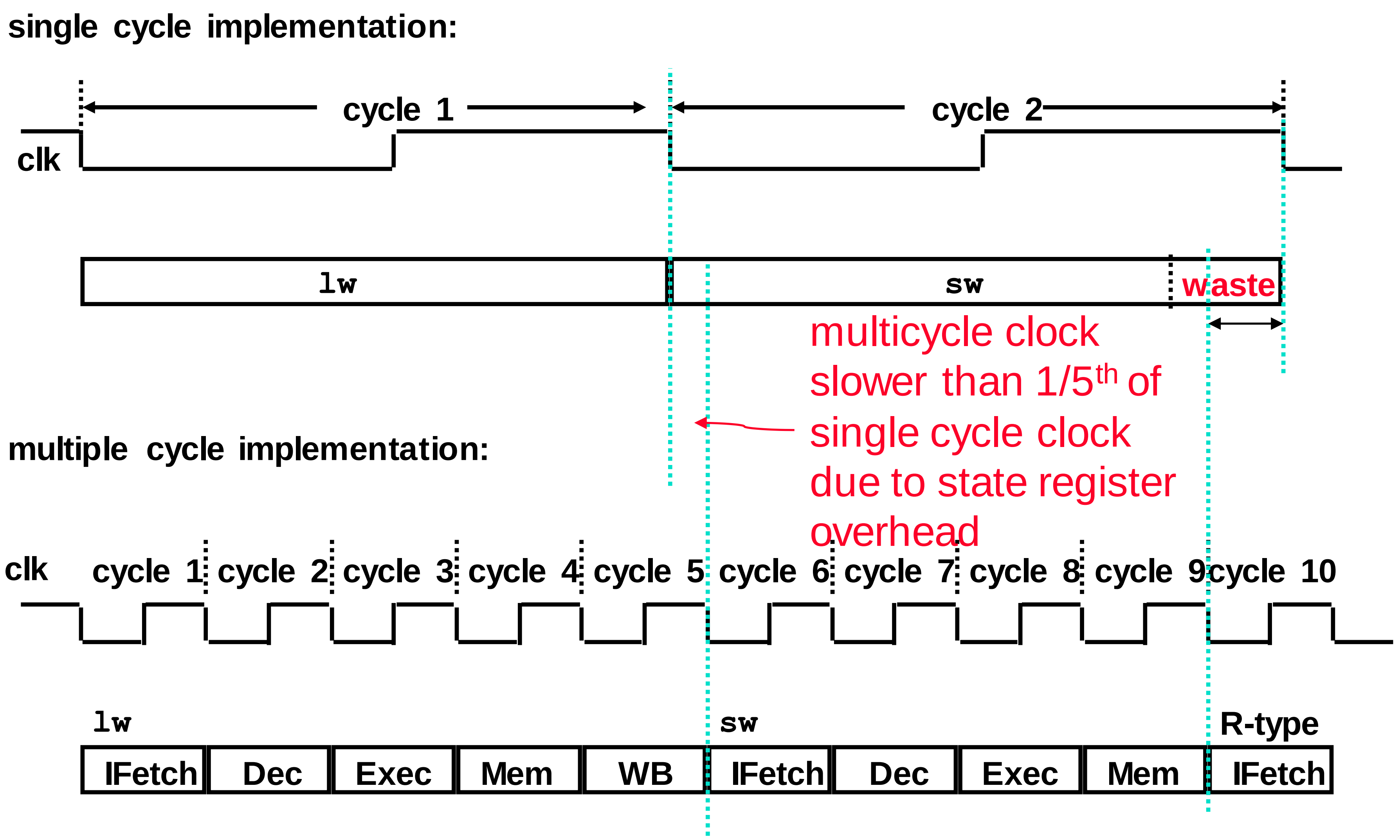
- how much faster would the machine be if a better data cache reduced the average load time to 2 cycles?
- how much faster, as we use branch prediction to shave a cycle off the branch time?
- what if two ALU instructions could be executed at once?

- Processor A:
  - ✓ CPI_CORE= 1, CPIMEM = 1, proc. clock = 500 MHz (2ns)

What is the speedup if we double processor clock frequency?

What about an infinite clock frequency?

Review: Single Cycle vs. Multiple Cycle Timing



Review: Will multicycle design be faster?

- let's assume $t_{setup} + t_{cq}$ time for registers = 0.1 ns
  - single cycle design:
    - clock cycle time = 4.7 + 0.1 = 4.8 ns
    - time/inst = 1 cycle/inst × 4.8 ns/cycle = 4.8 ns/inst
  - multicycle design:
    - clock cycle time = 1.0 + 0.1 = 1.1
    - time/inst = CPI × 1.1 ns/cycle (depends on the types or mixture of instructions!)

	I Fetch	Decode, R-Read	ALU	PC update	D Memory	R-Write	Total (ns)
Add	1	1	.9	-	-	.8	3.7
Load	1	1	.9	-	1	.8	4.7
Store	1	1	.9	-	1	-	3.9
beq	1	1	.9	.1	-	-	3.0

Will multicycle design be faster?

	Cycles needed	Instruction frequency
R-type	4	60%
Load	5	20%
Store	4	10%
beq	3	10%

What is CPI assuming this instruction mix???

Let's assume setup + hold time = 0.1 ns

Single cycle design:

Clock cycle time = 4.7 + 0.1 = 4.8 ns

time/inst = 1 cycle/inst * 4.8 ns/cycle = 4.8 ns/inst

Multicycle design:

Clock cycle time = 1.0 + 0.1 = 1.1

time/inst = CPI * 1.1 ns/cycle = ???

- Calculation assumptions:
  - Most instructions take 10 nanoseconds (ns)
  - But multiply instruction takes 40ns
  - Multiplies are 10% of all instructions

How much faster is a Multi-cycle Datapath over a single-cycle datapath (setup=hold=0)?

- Assumptions
  - 30% loads, 5ns
  - 10% stores, 5ns
  - 50% adds, 4ns
  - 10% multiplies, 20ns

How much faster is a Multi-cycle Datapath over a single-cycle datapath (setup=hold=0)?

Benchmarks

Which program to choose?

- Real programs
  - porting problem, complexity, not easy to understand the cause of results
- Kernels
  - computationally intense piece of real programs
- Toy benchmarks
  - QuickSort
- Synthetic benchmarks
- Benchmark suites
  - SPEC for scientific, engineering, and general purpose
  - TPC benchmarks for commercial systems
  - EEMBC (Embedded Microprocessor Benchmark Consortium) for embedded systems

SPEC Benchmarks

Integer benchmarks		FP benchmarks	
gzip	compression	wupwise	Quantum chromodynamics
vpr	FPGA place & route	swim	shallow water model
gcc	GNU C compiler	mgrid	multigrid solver in 3D fields
mcf	combinatorial optimization	applu	parabolic/elliptic pde
crafty	chess program	mesa	3D graphics library
parser	word processing program	galgel	computational fluid dynamics
eon	computer visualization	art	image recognition (NN)
perlbmk	perl application	equake	seismic wave propagation simulation
gap	group theory interpreter	facerec	facial image recognition
vortex	object oriented database	ammp	computational chemistry
bzip2	compression	lucas	primality testing
twolf	circuit place & route	fma3d	crash simulation fem
		sixtrack	nuclear physics accel
		apsi	pollutant distribution

Reporting Performance

How do we summarize performance for benchmark set w/ a single number?

let Ti be the execution time of program i:

- Arithmetic mean
  - Average execution time
  - Gives more weight to longer-running programs
- (weighted) arithmetic mean of execution times:

$$\sum_i T_i / N \qquad \sum_i T_i \times W_i$$

  - programs w/ the longest execution time will dominate the result
- Weighted arithmetic mean
  - More important programs can be emphasized
  - But what do we use as weights?
  - Different weight will make different machines look better

Takeaways

CPU Performance

$$\frac{\text{Instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{Instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

Amdahl’s Law

How should we report performance: Arithmetic Mean vs. Geometric Mean

Power/Energy Metrics

Pipelining

Pipelining: Natural Phenomenon

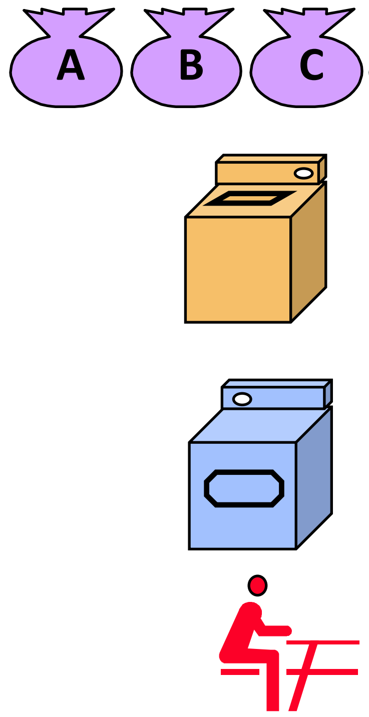
Laundry Example:

Ann, Brian, Ming, Ashok
each has one load of clothes
to wash, dry, and fold

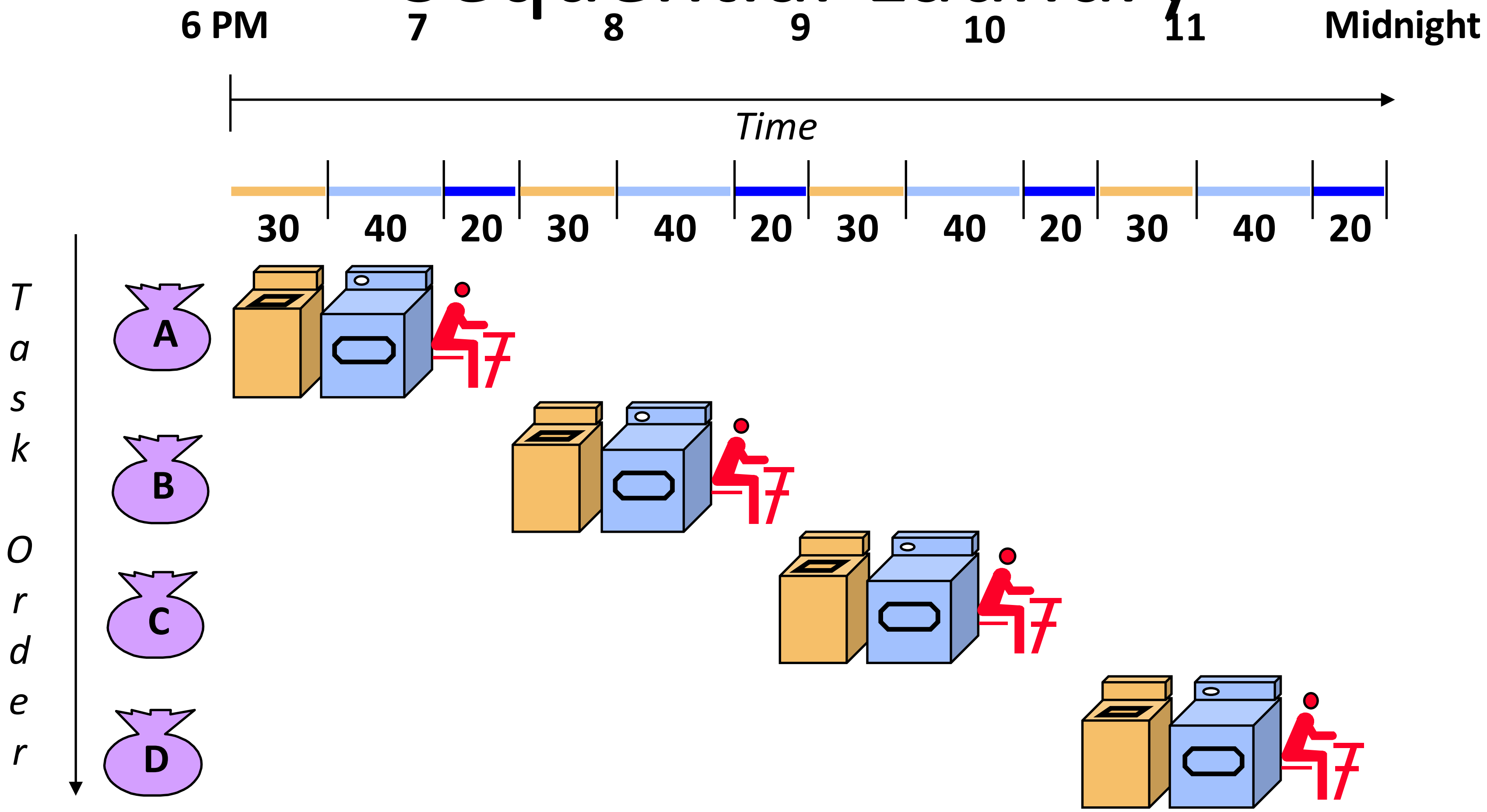
Washer takes 30 minutes

Dryer takes 40 minutes

“Folder” takes 20 minutes

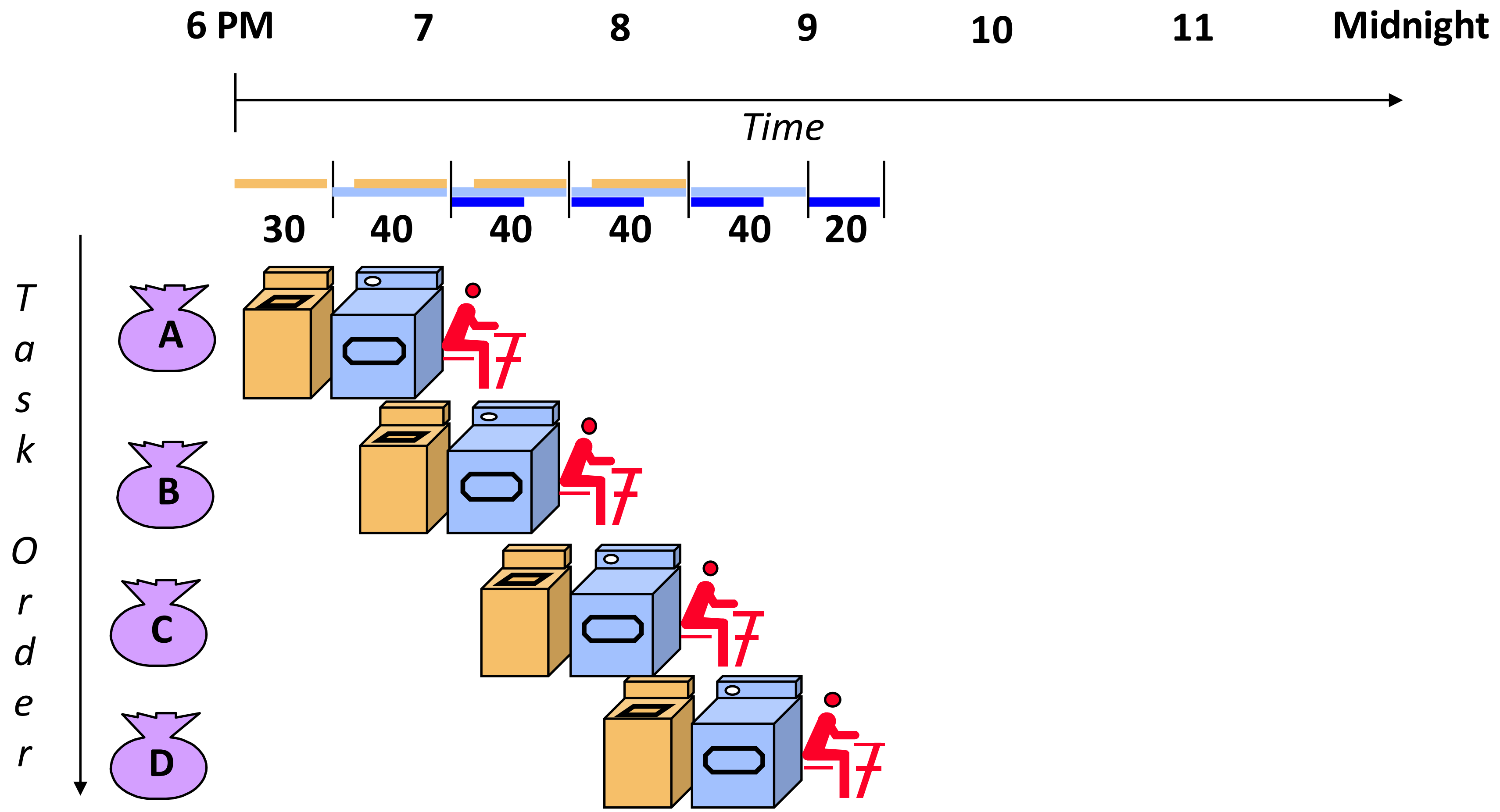


Sequential Laundry

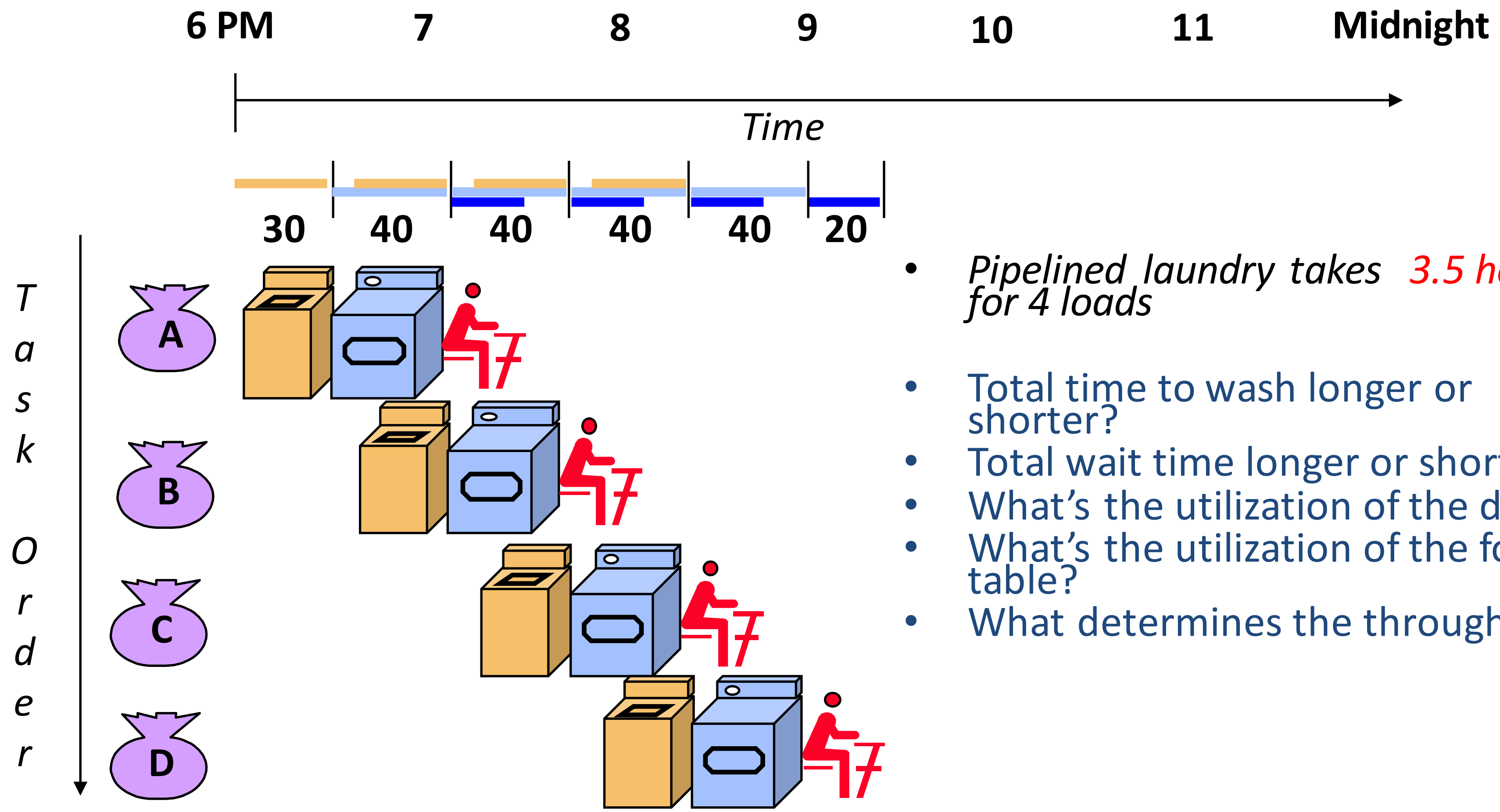


- Sequential laundry takes 6 hours for 4 loads

Pipelined Laundry -- Start work ASAP



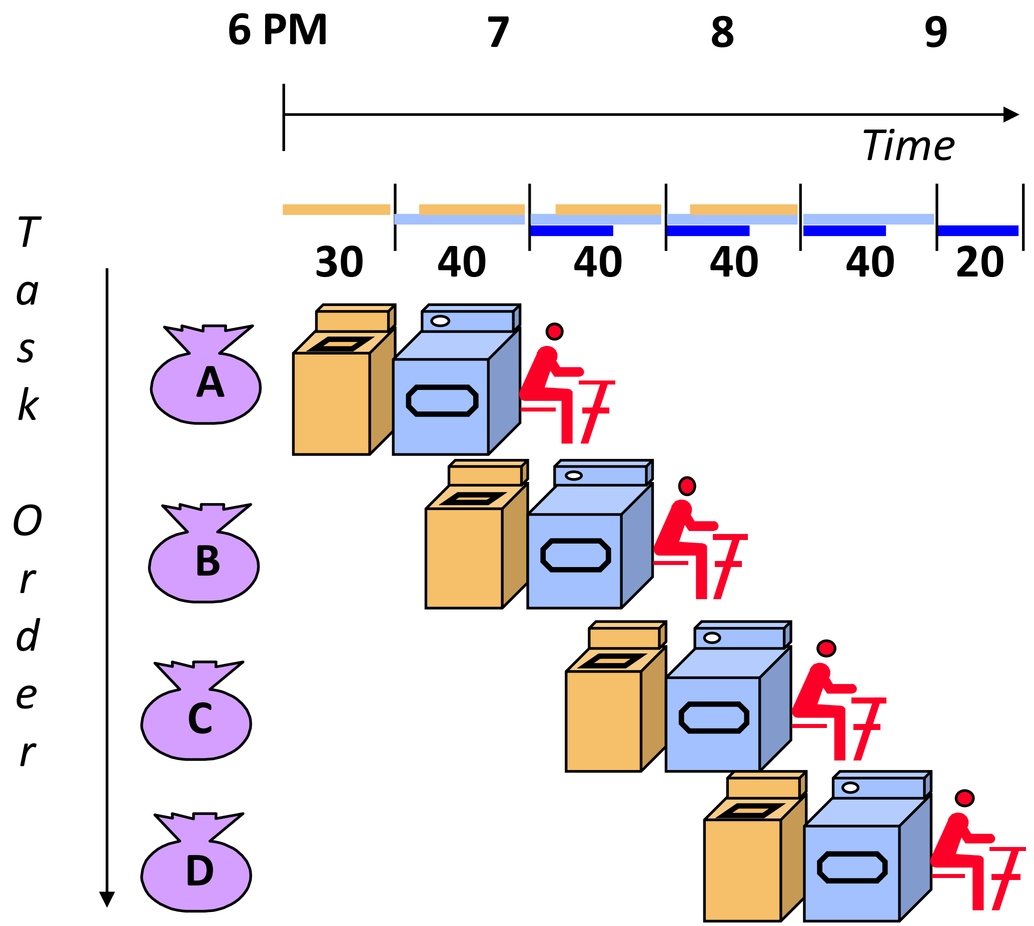
Pipelined Laundry -- Start work ASAP



- *Pipelined laundry takes* *3.5 hours* *for 4 loads*

- Total time to wash longer or shorter?
- Total wait time longer or shorter?
- What's the utilization of the dryer?
- What's the utilization of the folding table?
- What determines the throughput?

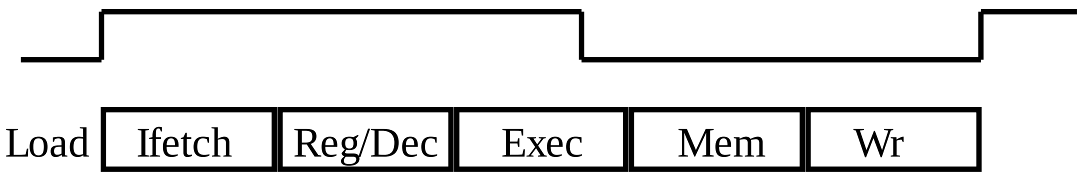
Pipelining Lessons



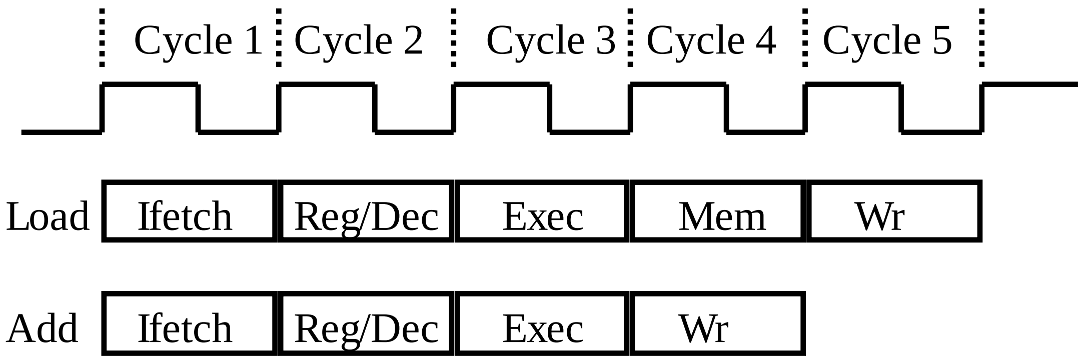
- Pipelining doesn't help *latency* of single task, it helps *throughput* of entire workload
- Pipeline rate limited by *slowest pipeline stage*
- Multiple tasks operating simultaneously
- Potential speedup = Number of pipe stages
- Unbalanced lengths of pipe stages reduces speedup
- Time to "fill" pipeline and time to "drain" it reduces speedup

Review -- Instruction Latencies

•**Single-Cycle CPU**

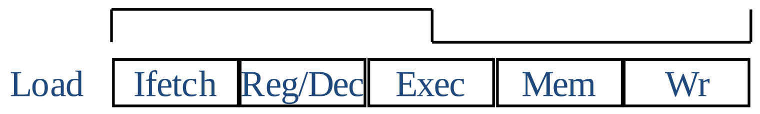


•**Multiple Cycle CPU**

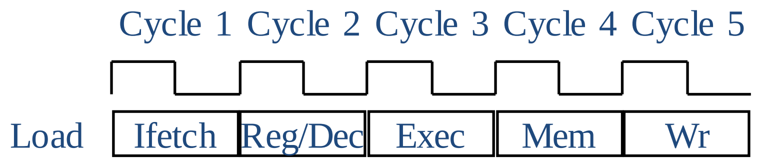


Instruction Latencies and Throughput

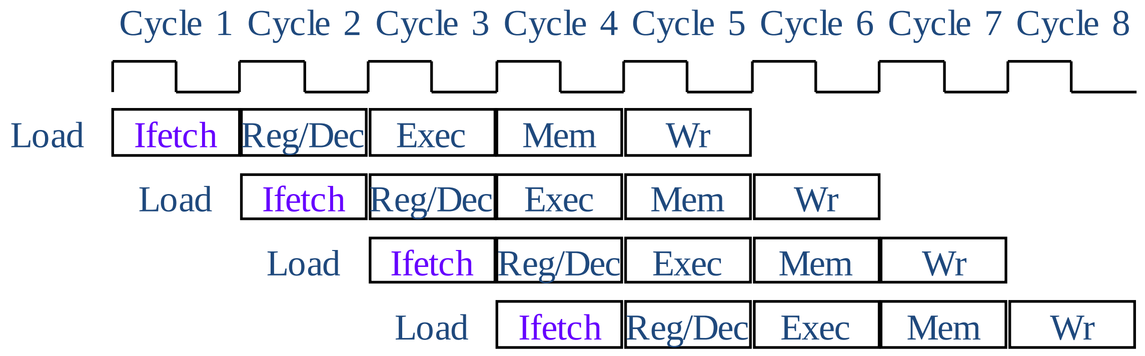
- **Single-Cycle CPU**



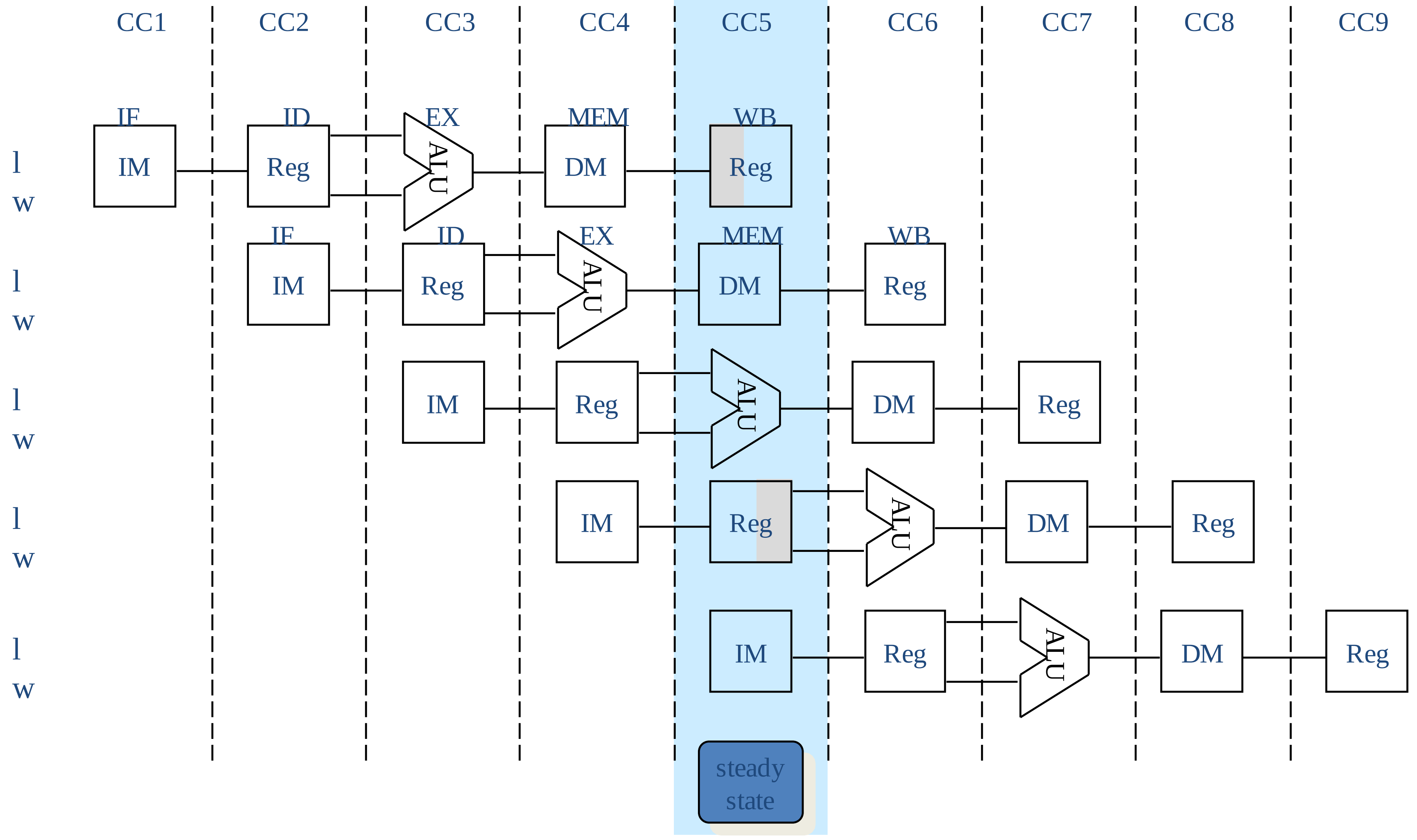
- **Multiple Cycle CPU**



- **Pipelined CPU**



Execution in a Pipelined Datapath



Speed up calculation

- Assuming the stages are perfectly balanced

- *Time between instructions (pipelined) =*
  Time between instructions (nonpipelined) /
  Number of pipe stages

Pipelining Example 1

- A processor that takes 5ns to execute an instruction is pipelined into 5 equal stages. The latch between each stage has a delay of 0.25 ns.
  - What is the minimum clock cycle time of the pipelined processor?
  - What is the maximum speedup that can be achieved by this pipelined processor? (compared to the original processor)
  - Can we have much deeper pipelining?