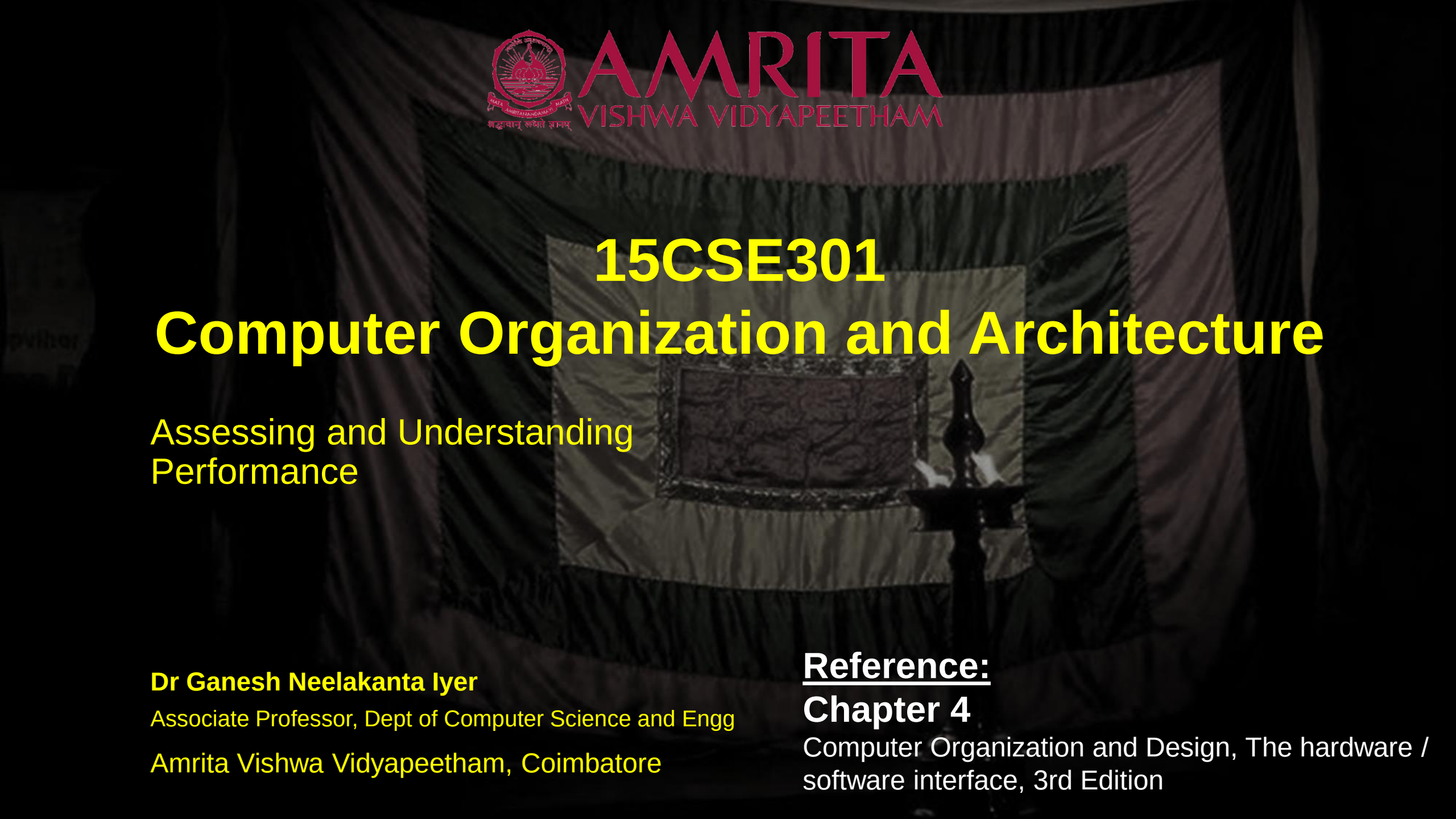


AMRITA VISHWA VIDYAPEETHAM

15CSE301
Computer Organization and Architecture

Assessing and Understanding Performance

Dr Ganesh Neelakanta Iyer
Associate Professor, Dept of Computer Science and Engg
Amrita Vishwa Vidyapeetham, Coimbatore

Reference:
Chapter 4
Computer Organization and Design, The hardware / software interface, 3rd Edition

RECAP

- You have learnt MIPS architecture
  - Instruction set
  - Addressing Modes
  - Conversion of instructions to different formats
  - Single cycle and multi cycle datapath implementation

Performance Perspectives

- Purchasing perspective
  - Given a collection of machines, which has the
    - Best performance ?
    - Least cost ?
    - Best performance / cost ?
- Design perspective
  - Faced with design options, which has the
    - Best performance improvement ?
    - Least cost ?
    - Best performance / cost ?
- Both require
  - basis for comparison
  - metric for evaluation

Our goal: understand cost & performance implications of architectural choices

Two Notions of "Performance"

Plane	DC to Paris	Speed	Passengers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

- Which has higher performance?
- Execution time (response time, latency, …)
  - Time to do a task
- Throughput (bandwidth, …)
  - Tasks per unit of time
- Response time and throughput often are in opposition

Definitions

- Performance is typically in units-per-second
  - bigger is better
- If we are primarily concerned with response time
  - performance = $\frac{1}{\text{execution\_time}}$

- " X is n times faster than Y" means

$$\frac{ExecutionTime_y}{ExecutionTime_x} = \frac{Performance_x}{Performance_y} = n$$

Example

- Time of Concorde vs. Boeing 747?
  - Concord is 1350 mph / 610 mph = 2.2 times faster
    = 6.5 hours / 3 hours

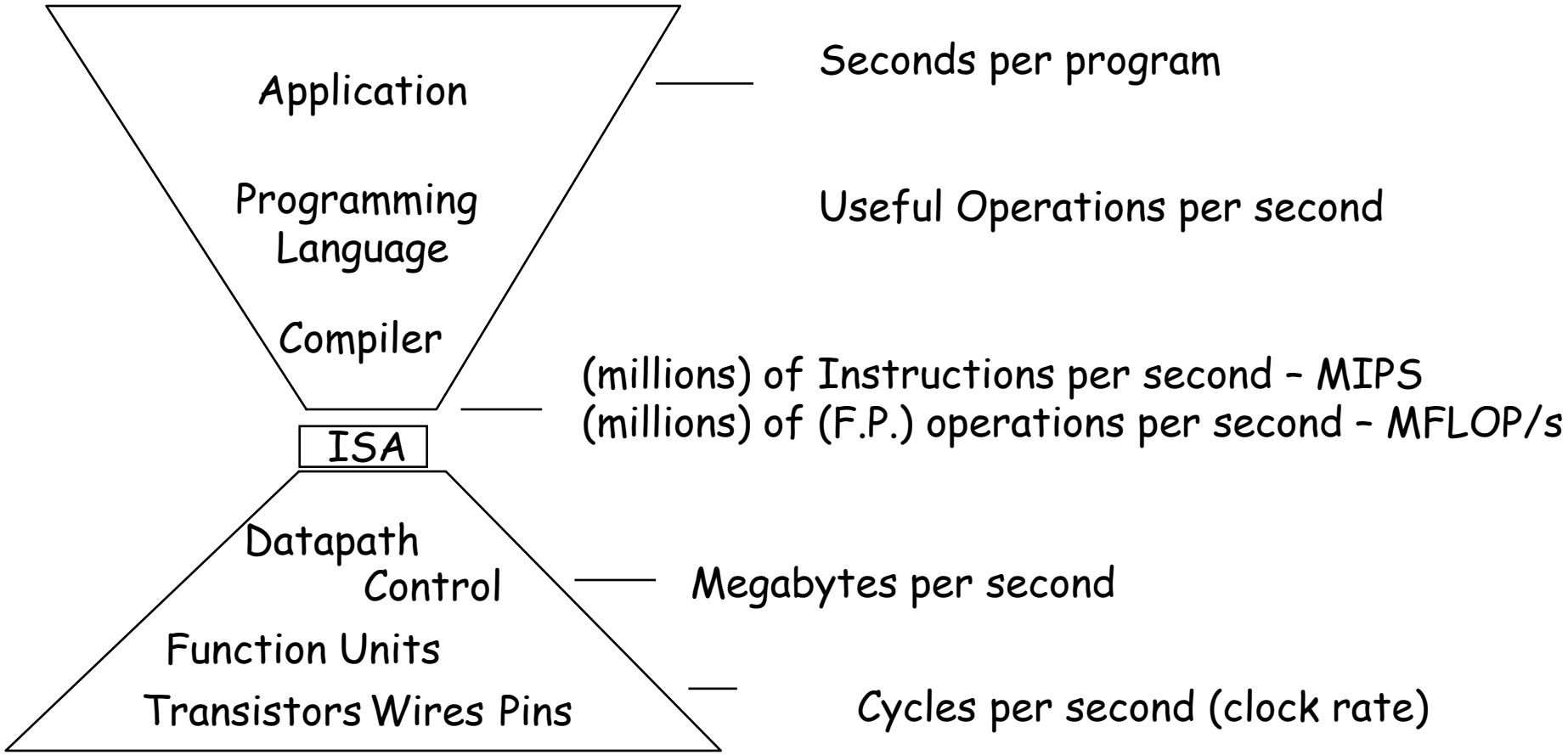
- Throughput of Concorde vs. Boeing 747 ?
  - Concord is 178,200 pmph / 286,700 pmph = 0.62 "times faster"
  - Boeing is 286,700 pmph / 178,200 pmph = 1.60 "times faster"

- Boeing is 1.6 times ("60%") faster in terms of throughput
- Concord is 2.2 times ("120%") faster in terms of flying time

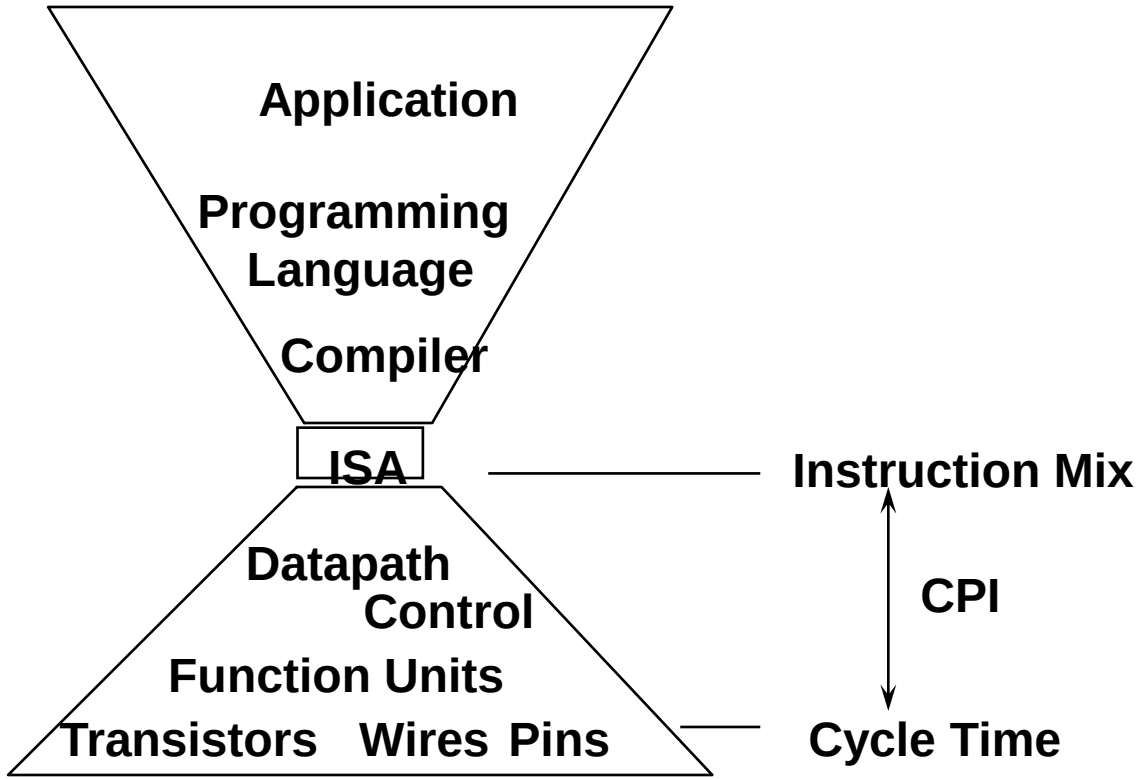
We will focus primarily on execution time for a single job

Lots of instructions in a program => Instruction throughput important!

Metrics of Performance

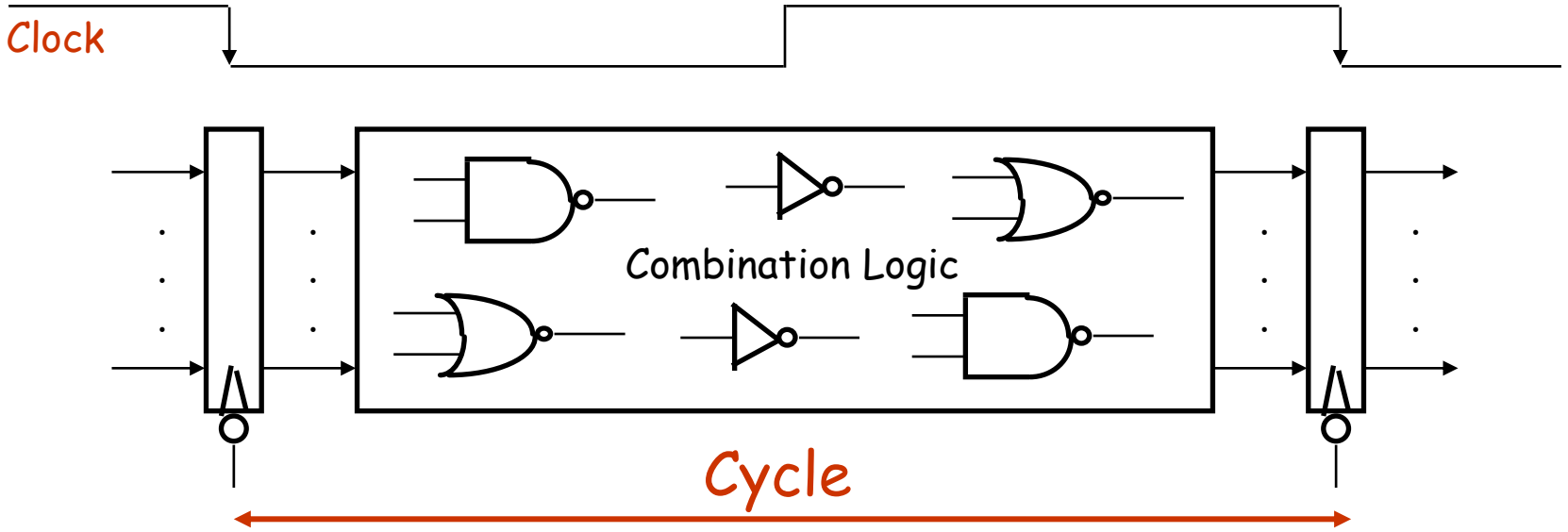


Organizational Trade-offs



CPI is a useful design measure relating the Instruction Set Architecture with the Implementation of that architecture, and the program measured

Processor Cycles



Most contemporary computers have fixed, repeating clock cycles

Three Components of CPU Performance

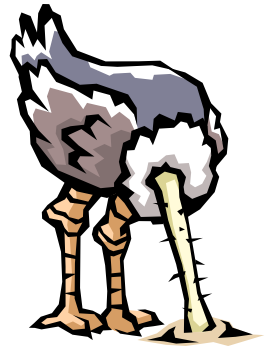
$$\text{CPU time}_{X,P} = \text{Instructions executed}_{P} * \text{CPI}_{X,P} * \text{Clock cycle time}_{X}$$

Cycles Per Instruction

Instructions Executed

- Instructions executed:
  - We are not interested in the static instruction count, or how many lines of code are in a program.
  - Instead we care about the dynamic instruction count, or how many instructions are actually executed when the program runs.
- There are three lines of code below, but the number of instructions executed would be XXXX?.

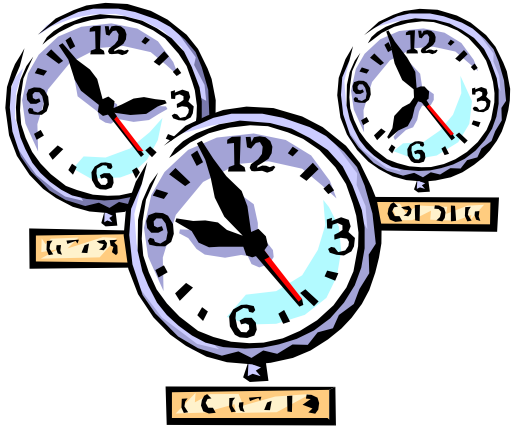
```
            li   $a0, 1000
Ostrich:    sub  $a0, $a0, 1
            bne  $a0, $0, Ostrich
```



CPI

- The average number of clock cycles per instruction, or CPI, is a function of the machine and program.
  - The CPI depends on the actual instructions appearing in the program—a floating-point intensive application might have a higher CPI than an integer-based program.
  - It also depends on the CPU implementation. For example, a Pentium can execute the same instructions as an older 80486, but faster.
- It is common to each instruction took one cycle, making CPI = 1.
  - The CPI can be >1 due to memory stalls and slow instructions.
  - The CPI can be <1 on machines that execute more than 1 instruction per cycle (superscalar).

Clock cycle time



- One “cycle” is the minimum time it takes the CPU to do any work.
  - The clock cycle time or clock period is just the length of a cycle.
  - The clock rate, or frequency, is the reciprocal of the cycle time.
- Generally, a higher frequency is better.
- Some examples illustrate some typical frequencies.
  - A 500MHz processor has a cycle time of 2ns.
  - A 2GHz (2000MHz) CPU has a cycle time of just 0.5ns (500ps).

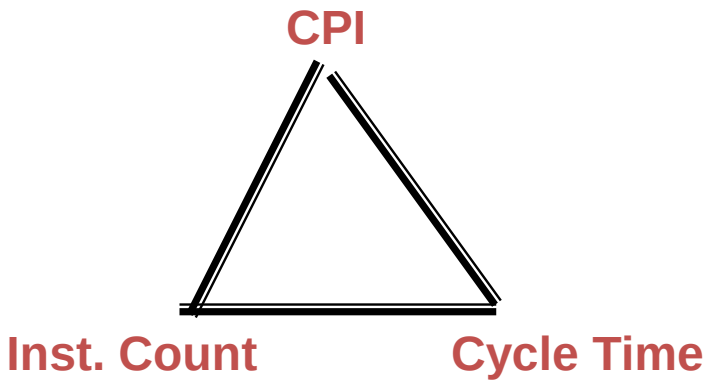
CPU Performance

$$\text{CPU time}_{X,P} = \text{Instructions executed}_P * \text{CPI}_{X,P} * \text{Clock cycle time}_X$$

$$CPUtime = \frac{Seconds}{Program} = \frac{Cycles}{Program} \cdot \frac{Seconds}{Cycle}$$

$$= \frac{Instructions}{Program} \cdot \frac{Cycles}{Instruction} \cdot \frac{Seconds}{Cycle}$$

	IC	CPI	Clock Cycle
Program	√		
Compiler	√	(√)	
Instruction Set	√	√	
Organization		√	√
Technology			√



Example 1

- CPU clock rate is 1 MHz
- Program takes 45 million cycles to execute
- What's the CPU time?

Example 1

- CPU clock rate is 1 MHz
- Program takes 45 million cycles to execute
- What's the CPU time?

- 45,000,000 * (1 / 1,000,000) = 45 seconds

Example 2

- CPU clock rate is 500 MHz
- Program takes 45 million cycles to execute
- What's the CPU time?

Example 2

- CPU clock rate is 500 MHz
- Program takes 45 million cycles to execute
- What's the CPU time?

- 45,000,000 * (1 / 500,000,000) = 0.09 seconds

Example 3

- Let assume that a benchmark has 100 instructions:
  - 25 instructions are loads/stores (each take 2 cycles)
  - 50 instructions are adds (each takes 1 cycle)
  - 25 instructions are square root (each takes 50 cycles)
  - What is the CPI for this benchmark?

Example 3

- Let assume that a benchmark has 100 instructions:
  - 25 instructions are loads/stores (each take 2 cycles)
  - 50 instructions are adds (each takes 1 cycle)
  - 25 instructions are square root (each takes 50 cycles)
  - What is the CPI for this benchmark?

- CPI = ((0.25 * 2) + (0.50 * 1) + (0.25 * 50)) = 13.5

Computing CPI

- The CPI is the average number of cycles per instruction
- If for each instruction type, we know its frequency and number of cycles need to execute it, we can compute the overall CPI as follows

$$CPI = \sum CPI \times F$$

- For example

Op	F	CPI	CPI x F	% Time
ALU	50%	1	.5	23%
Load	20%	5	1.0	45%
Store	10%	3	.3	14%
Branch	20%	2	.4	18%
Total	100%		2.2	100%

Cycles Per Instruction (Throughput)

"Cycles per Instruction"

CPI = (CPU Time * Clock Rate) / Instruction Count
= Cycles / Instruction Count

$$\text{CPU time} = \text{Cycle Time} \times \sum_{j=1}^{n} \text{CPI}_j \times \text{I}_j$$

"Instruction Frequency"

$$\text{CPI} = \sum_{j=1}^{n} \text{CPI}_j \times \text{F}_j \qquad \text{where } \text{F}_j = \frac{\text{I}_j}{\text{Instruction Count}}$$

Example 4

- Suppose we have two implementations of the same ISA
- For some program,
  - Machine A has a clock cycle time of 10 ns. and a CPI of 2.0
  - Machine B has a clock cycle time of 20 ns. and a CPI of 1.2
  - Which machine is faster for this program, and by how much?
- Assume that # of instructions in the program is 1,000,000,000

Example 4

- Suppose we have two implementations of the same ISA
- For some program,
  - Machine A has a clock cycle time of 10 ns. and a CPI of 2.0
  - Machine B has a clock cycle time of 20 ns. and a CPI of 1.2
  - Which machine is faster for this program, and by how much?
- Assume that # of instructions in the program is 1,000,000,000

$$\text{CPU Time}_A = 10^9 * 2.0 * 10 * 10^{-9} = 20 \text{ seconds}$$

$$\text{CPU Time}_B = 10^9 * 1.2 * 20 * 10^{-9} = 24 \text{ seconds}$$

Machine A is faster

$$\frac{24}{20} = 1.2 \text{ times}$$

Principal Design Metrics: CPI and Cycle Time

$$Performance = \frac{1}{ExecutionTime}$$

$$Performance = \frac{1}{CPI \times CycleTime}$$

$$Performance = \frac{1}{\frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}} = \frac{Instructions}{Seconds}$$

Example

Typical Mix

Op	Freq	Cycles	CPI
ALU	50%	1	.5
Load	20%	5	1.0
Store	10%	3	.3
Branch	20%	2	.4
			2.2

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?
  - Load → 20% x 2 cycles = .4
  - Total CPI 2.2 → 1.6
  - Relative performance is 2.2 / 1.6 = 1.38
- How does this compare with reducing the branch instruction to 1 cycle?
  - Branch → 20% x 1 cycle = .2
  - Total CPI 2.2 → 2.0
  - Relative performance is 2.2 / 2.0 = 1.1

Example 5

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively)

The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C

The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C

- Which sequence will be faster? How much?
- What is the CPI for each sequence?

Example 5

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively)

The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C

The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C

- Which sequence will be faster? How much?
- What is the CPI for each sequence?

# of cycles for first code = (2 * 1) + (1 * 2) + (2 * 3) = 10 cycles

# of cycles for second code = (4 * 1) + (1 * 2) + (1 * 3) = 9 cycles

10 / 9 = 1.11 times

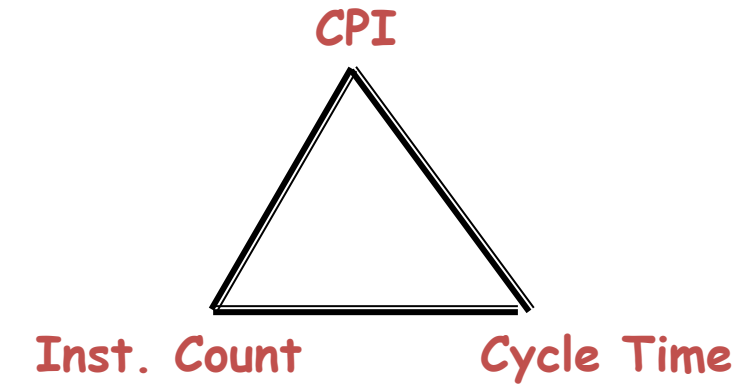
CPI for first code = 10 / 5 = 2

CPI for second code = 9 / 6 = 1.5

Summary: Evaluating Instruction Sets and Implementation

- Design-time metrics:
  - Can it be implemented, in how long, at what cost?
  - Can it be programmed? Ease of compilation?
- Static Metrics:
  - How many bytes does the program occupy in memory?
- Dynamic Metrics:
  - How many instructions are executed?
  - How many bytes does the processor fetch to execute the program?
  - How many clocks are required per instruction?
  - How "lean" a clock is practical?
- Best Metric:
  Time to execute the program!

NOTE: Depends on instructions set, processor organization, and compilation techniques.



Amdahl's Law

- Amdahl's Law states that optimizations are limited in their effectiveness.

$$\text{Execution time after improvement} = \frac{\text{Time affected by improvement}}{\text{Amount of improvement}} + \text{Time unaffected by improvement}$$

- For example, doubling the speed of floating-point operations sounds like a great idea. But if only 10% of the program execution time T involves floating-point code, then the overall performance improves by just 5%.

$$\text{Execution time after improvement} = \frac{0.10\ T}{2} + 0.90\ T = 0.95\ T$$

- What is the maximum speedup from improving floating point?
  - Second Law of Performance:

Make the fast case common

Example 6

- Assume that a program runs in 100 seconds on a machine, with multiply operations responsible for 80 seconds. How much do I have to improve the speed of multiplication if I want my program to run 2 times faster?

Example 6

- Assume that a program runs in 100 seconds on a machine, with multiply operations responsible for 80 seconds. How much do I have to improve the speed of multiplication if I want my program to run 2 times faster?

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

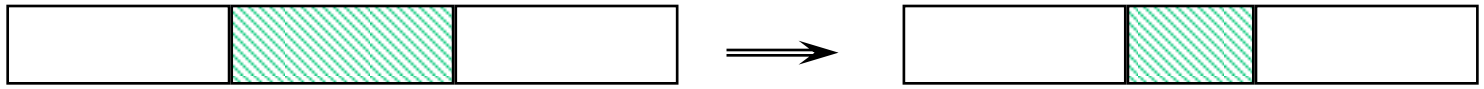
$$50 \text{ seconds} = \frac{80 \text{ seconds}}{n} + (100 - 80 \text{ seconds})$$

$$n = \frac{80 \text{ seconds}}{30 \text{ seconds}} = 2.67$$

Amdahl's “Law”: Make the Common Case Fast

Speedup due to enhancement E:

$$Speedup(E) = \frac{\text{ExTime w/o E}}{\text{ExTime w/ E}} = \frac{\text{Performance w/ E}}{\text{Performance w/o E}}$$



- Suppose that enhancement E accelerates a fraction F of the task
- by a factor S and the remainder of the task is unaffected then,

ExTime(with E) = ((1-F) + F/S) X ExTime(without E)

Speedup(with E) = ExTime(without E) ÷
((1-F) + F/S) X ExTime(without E)

Performance improvement is limited by how much the improved feature is used → Invest resources where time is spent.

$$Speedup_{overall} = \frac{ExecTime_{old}}{ExecTime_{new}} = \frac{1}{\frac{Fraction_{enhanced}}{Speedup} + (1 - Fraction_{enhanced})}$$

Example 7

- Floating point instructions are improved to run twice as fast, but only 10% of the time was spent on these instructions originally. How much faster is the new machine?

Example 7

- Floating point instructions are improved to run twice as fast, but only 10% of the time was spent on these instructions originally. How much faster is the new machine?

$$\text{Speedup} = \frac{\text{ExTime}_{old}}{\text{ExTime}_{new}} = \frac{1}{(1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}}}$$

$$\text{Speedup} = \frac{1}{(1 - 0.1) + 0.1/2} = 1.053$$

- The new machine is 1.053 times as fast, or 5.3% faster

Example 8

- Computers M1 and M2 are two implementations of the same instruction set
- M1 has a clock rate of 50 MHz and M2 has a clock rate of 100 MHz
- M1 has a CPI of 2.8 and M2 has a CPI of 3.2 for a given program
- How many times faster is M2 than M1 for this program?

Example 8

- Computers M1 and M2 are two implementations of the same instruction set
- M1 has a clock rate of 50 MHz and M2 has a clock rate of 100 MHz
- M1 has a CPI of 2.8 and M2 has a CPI of 3.2 for a given program
- How many times faster is M2 than M1 for this program?

$$\frac{\text{ExTime}_{M1}}{\text{ExTime}_{M2}} = \frac{\text{IC}_{M1} \times \text{CPI}_{M1} / \text{Clock Rate}_{M1}}{\text{IC}_{M2} \times \text{CPI}_{M2} / \text{Clock Rate}_{M2}} = \frac{2.8/50}{3.2/100} = 1.75$$

Example 9

- Computers M1 and M2 are two implementations of the same instruction set
- M1 has a clock rate of 50 MHz and M2 has a clock rate of 100 MHz
- M1 has a CPI of 2.8 and M2 has a CPI of 3.2 for a given program
- How many times faster is M2 than M1 for this program?

$$\frac{\text{ExTime}_{M1}}{\text{ExTime}_{M2}} = \frac{\text{IC}_{M1} \times \text{CPI}_{M1} / \text{Clock Rate}_{M1}}{\text{IC}_{M2} \times \text{CPI}_{M2} / \text{Clock Rate}_{M2}} = \frac{2.8/50}{3.2/100} = 1.75$$

- What would the clock rate of M1 have to be for them to have the same execution time?

Example 9

- Computers M1 and M2 are two implementations of the same instruction set
- M1 has a clock rate of 50 MHz and M2 has a clock rate of 100 MHz
- M1 has a CPI of 2.8 and M2 has a CPI of 3.2 for a given program
- How many times faster is M2 than M1 for this program?

$$\frac{\text{ExTime}_{M1}}{\text{ExTime}_{M2}} = \frac{\text{IC}_{M1} \times \text{CPI}_{M1} / \text{Clock Rate}_{M1}}{\text{IC}_{M2} \times \text{CPI}_{M2} / \text{Clock Rate}_{M2}} = \frac{2.8/50}{3.2/100} = 1.75$$

- What would the clock rate of M1 have to be for them to have the same execution time?
- 2.8 / Clock RateM1 = 3.2 / 100 Clock RateM1 = 87.5 MHz

Marketing Metrics

- MIPS = Instruction Count / Time * 10^6
  = Clock Rate / CPI * 10^6
  - machines with different instruction sets ?
  - programs with different instruction mixes ?
  - dynamic frequency of instructions
  - uncorrelated with performance
- MFLOP/s= FP Operations / Time * 10^6
  - machine dependent
  - often not where time is spent

Summary

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

- Time is the measure of computer performance!
- Good products created when have:
  - Good benchmarks
  - Good ways to summarize performance
- If not good benchmarks and summary, then choice between improving product for real programs vs. improving product to get more sales → sales almost always wins
- Remember Amdahl's Law: Speedup is limited by unimproved part of program

Thank you!

Dr Ganesh Neelakanta Iyer

ni_amrita@progress.com
ganesh.vigneswara@gmail.com

त्यागेनैके अमृतत्वमानशुः

MATA AMRITANANDAMAYI MATH

श्रद्धावान् लभते ज्ञानम्