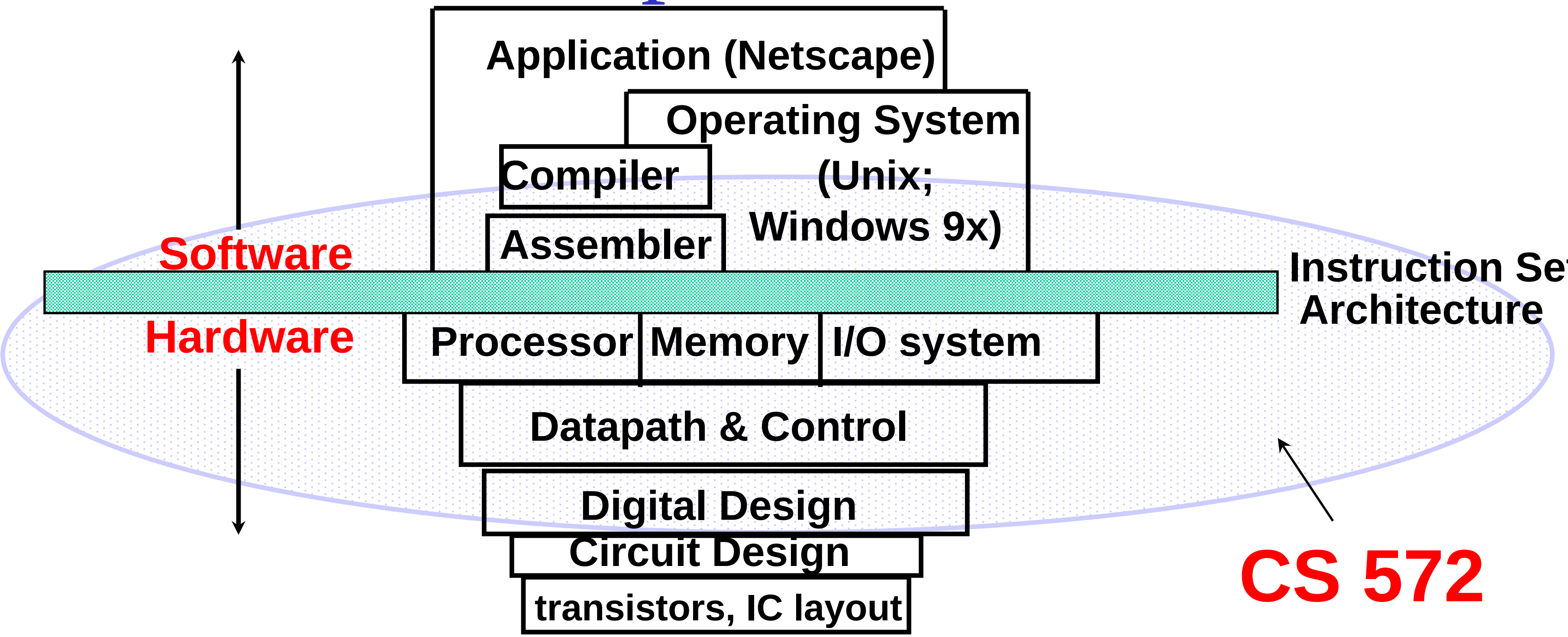
What is Computer Architecture ?



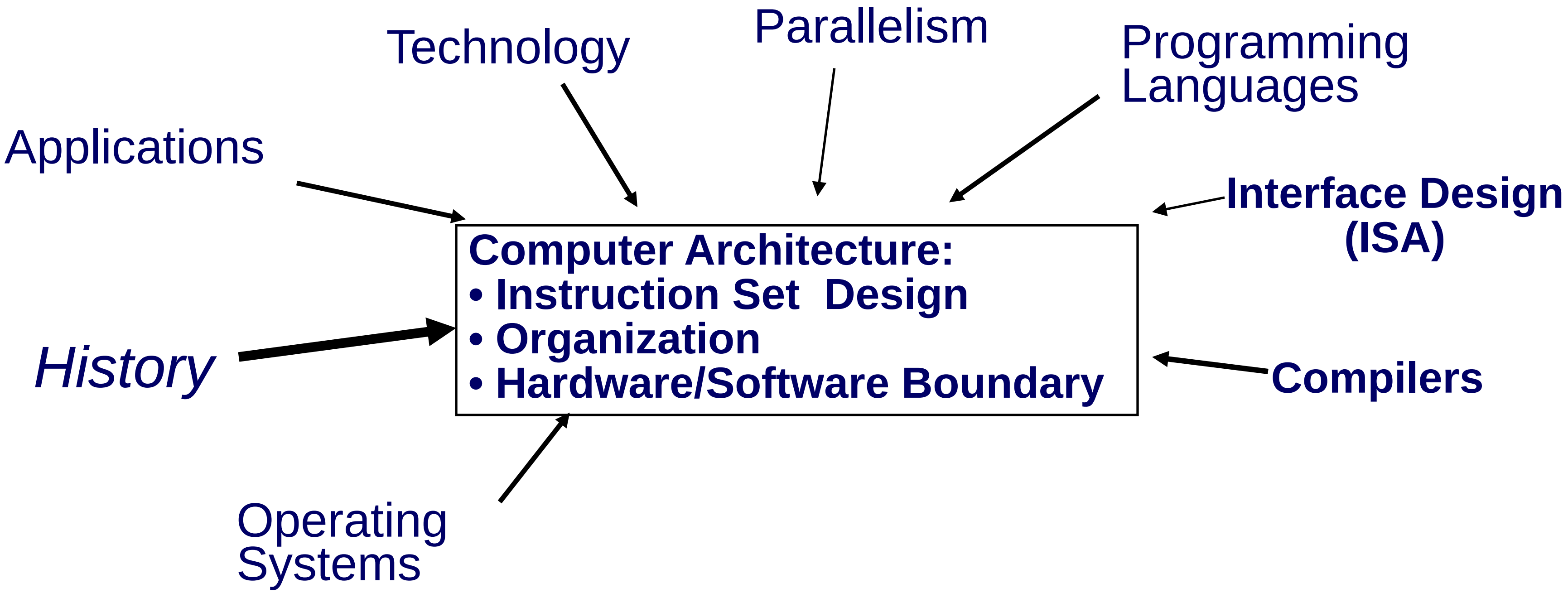
- Key Idea: *levels of abstraction*
  – hide unnecessary implementation details
  – helps us cope with enormous complexity of real systems

Computer Architecture's Changing Definition

- 1950s to 1960s Computer Architecture Course:
  Computer Arithmetic
- 1970s to mid 1980s Computer Architecture Course:
  Instruction Set Design, especially ISA appropriate for compilers
- 1990s Computer Architecture Course:
  Design of CPU, memory system, I/O system, Multiprocessors, Networks
- 2010s: Computer Architecture Course:
  Self adapting systems? Self organizing structures? DNA Systems/Quantum Computing?

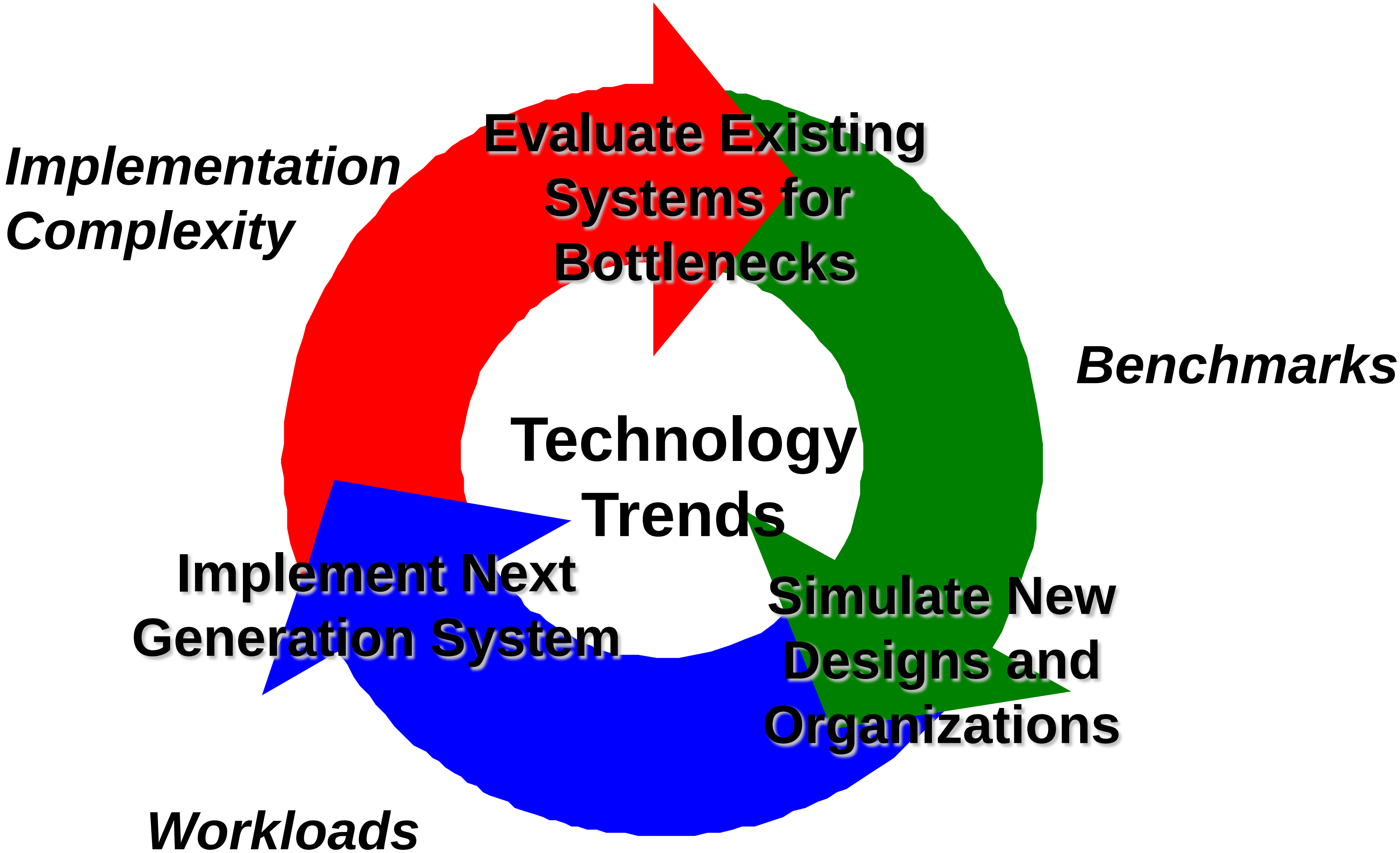
CS 572 Course Focus

Understanding the design techniques, machine structures, technology factors, evaluation methods that will determine the form of computers in 21st Century



Measurement & Evaluation

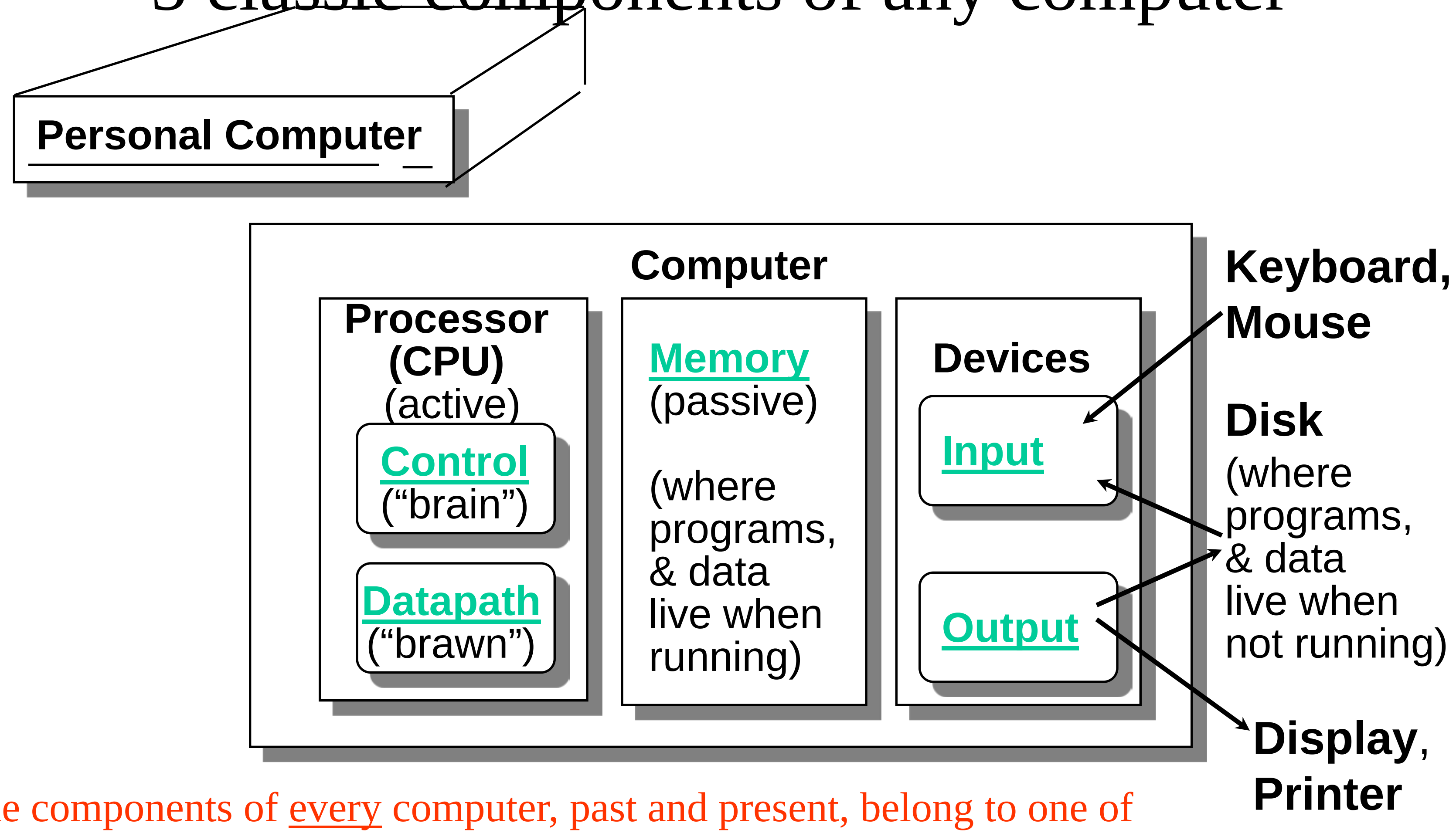
Computer Engineering Methodology



Architecture design is an iterative process: Searching the space of possible designs at all levels of computer systems

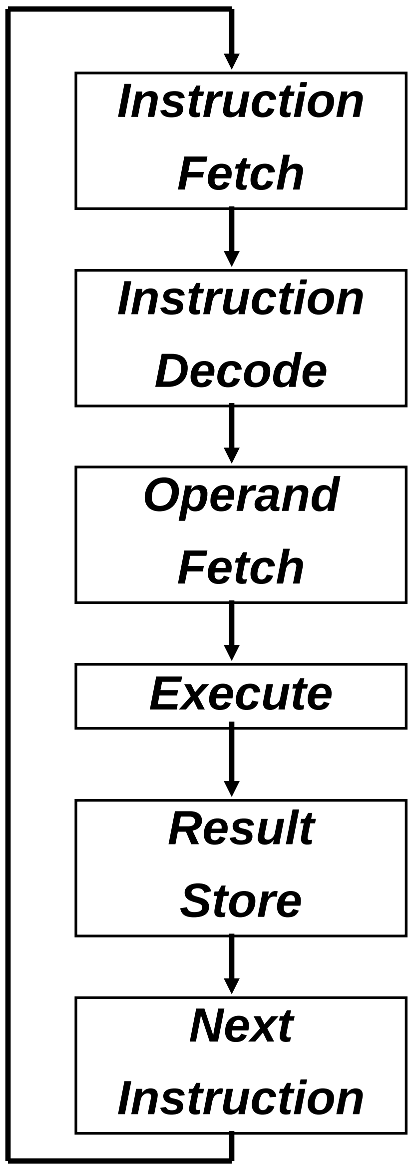
Machine Organization

5 classic components of any computer



The components of <u>every</u> computer, past and present, belong to one of these five categories

Execution Cycle



Obtain instruction from program storage

Determine required actions and instruction size

Locate and obtain operand data

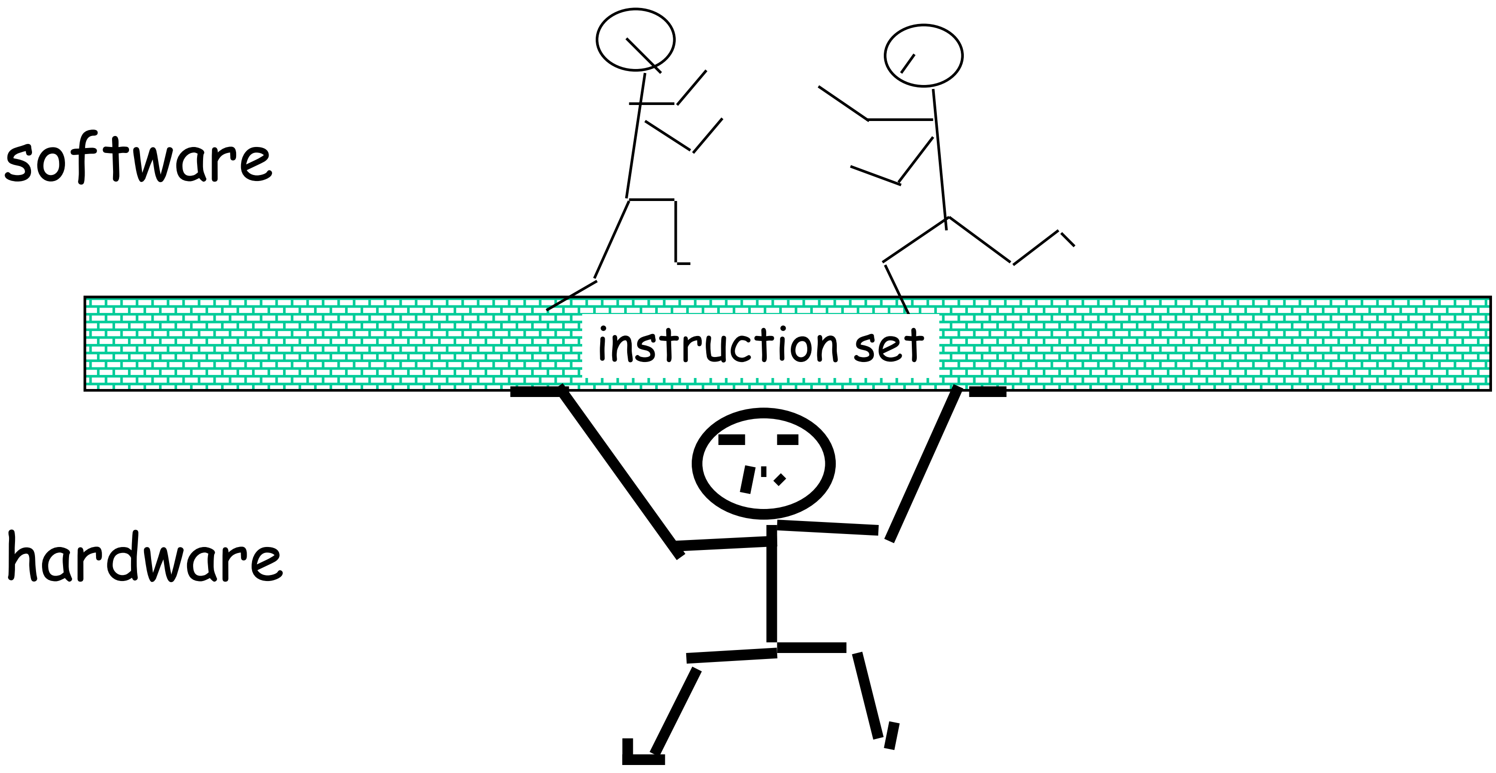
Compute result value or status

Deposit results in storage for later use

Determine successor instruction

The Instruction Set: a Critical Interface

The actual programmer visible instruction set



Outline

- Performance Metrics: How do we conclude that System-A is better than System-B?
- Measuring CPU time
- Amdahl's Law: Relates total speedup of a system to the speedup of some portion of that system.

- Topics: (Sections 1.1, 1.2, 1.3, 1.8, 1.9)

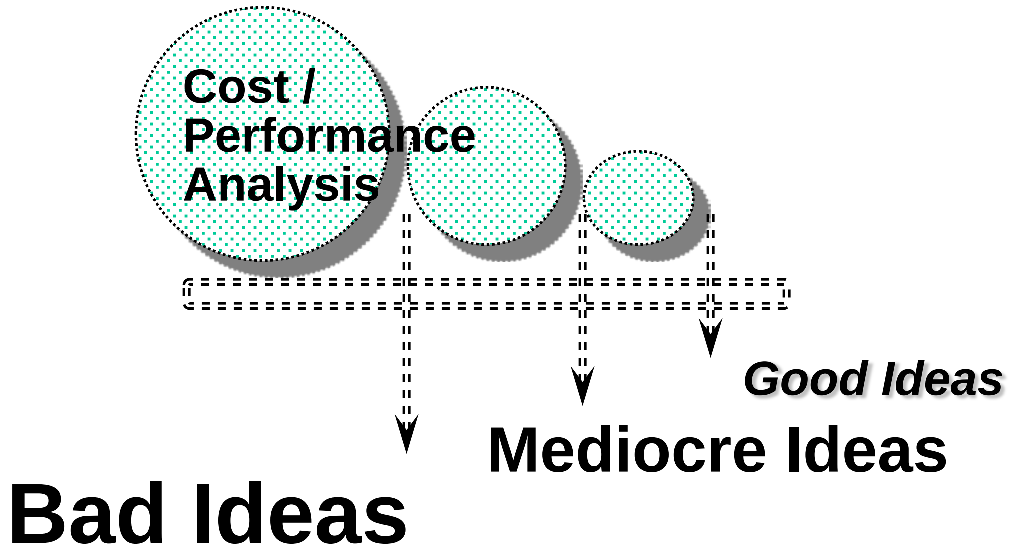
Importance of Measurement

Architecture design is an iterative process:

- Search the possible design space
- Make selections
- Evaluate the selections made

⇨

Good measurement tools are required to accurately evaluate the selection.



Two notions of performance

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

- Which has higher performance?
  1. Time to deliver 1 passenger?
  2. Time to deliver 400 passengers?

Example of Response Time v. Throughput

- Flying Time: Concorde vs. Boeing 747?
  - Concord is 6.5 hours / 3 hours = 2.2 times as fast (response time,)
- Throughput: Boeing vs. Concorde?
  - Boeing 747: 286,700 p-mph / 178,200 p-mph = 1.6 times as fast (throughput,)

•Time to do the task (Interest to users)

– execution time, response time, latency, etc.

•Tasks per day, hour, week, sec, (Interest to system administrators)

– throughput, bandwidth, etc.

Who do we care as computer architect?

Performance Definitions

- We are primarily concerned with <u>response time</u>
- To *maximize* performance, we must *minimize* response time for some task:
  - $\text{Performance}_x > \text{Performance}_y$

    $\Rightarrow \text{response\_time}_x < \text{response\_time}_y$
- "X is n times as fast as Y" means
- $\text{Performance}_x = n \times \text{Performance}_y$

What is Execution Time?

- Definition 1:
  - Total time to complete a task, including disk accesses, memory accesses, I/O activities, operating system overhead, ...
  - "wall-clock time", "response time", or "elapsed time"
- Definition 2: measure time processor is working on your program only (since multiple processes running at same time)
  - "CPU execution time" or "CPU time "
  - Often divided into system CPU time (in OS) and user CPU time (in user program)

How to Measure Time?

- User $\Rightarrow$ actual elapsed time to complete particular task is only true basis for comparison
  - sum of I/O time, User + System CPU, time spent on other tasks, boot time, etc.
  - alternatives may mislead!
- CPU designer $\Rightarrow$ want measure relating to how fast processor hardware can perform basic functions (CPU execution time)

Measuring CPU time

- Most computers are constructed using a clock that runs at a constant rate and determines when events take place in the hardware
  - These discrete time intervals called clock cycles
  - Length of clock period: clock cycle time (e.g., 2 nanoseconds or 2 ns) and clock rate (e.g., 500 megahertz, or 500 MHz), which is the inverse of the clock period;

$$\frac{1}{2 \times 10^{-9}\,\text{Sec.}} = 500\text{MHz}$$

  - Execution time = # Clock cycles X clock cycle time

Example of Measuring CPU time

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

- If a computer has a clock rate of 50 MHz, how long does it take to execute a program with 1,000 instructions, if the CPI for the program is 3.5?
- Using the equation

  CPU time = Instruction count x CPI / clock rate

  gives

  $\text{CPU time} = 1000 \times 3.5 / (50 \times 10^{6})$

Example of Measuring CPU time

- If a computer's clock rate increases from 200 MHz to 250 MHz and the other factors remain the same, how many times faster will the computer be?

$$\frac{\text{CPU time old}}{\text{CPU time new}} = \frac{\text{clock rate new}}{\text{clock rate old}} = \frac{250\ \text{MHz}}{200\ \text{MHZ}} = 1.25$$

- What simplifying assumptions did we make?

Performance Example

- Two computers M1 and M2 with the same instruction set.
- For a given program, we have

	Clock rate (MHz)	CPI
M1	50	2.8
M2	75	3.2

- How many times faster is M2 than M1 for this program?

$$\frac{ExTime_{M1}}{ExTime_{M2}} = \frac{IC_{M1} \times CPI_{M1} / Clock\ Rate_{M1}}{IC_{M2} \times CPI_{M2} / Clock\ Rate_{M2}} = \frac{2.8/50}{3.2/75} = 1.31$$

Example

Question:

A program runs on a 400 MHz computer in 10 secs. We like the program to run in 6 secs by designing a faster CPU. Assume that increasing clock rate would mean the program needs 20% more clock cycles. What clock rate should the designer target?

Answer:

The number of clock cycles for the program on the present computer =

10 X 400 X 10^6 = 4000 X 10^6

With 20% increase, the new computer should take 1.2 X 4000 X 10^6 =

4800 X 10^6 cycles

Required execution time = 6 seconds

Then the required clock rate = 4800/6 X 10^6 cycles/sec = 800 MHz

CPI: Cycles Per Instruction

- Clock Cycles for program

= Instructions *executed* for a program
(called "Instruction Count")

x Average Clock cycles Per Instruction
(abbreviated "CPI")

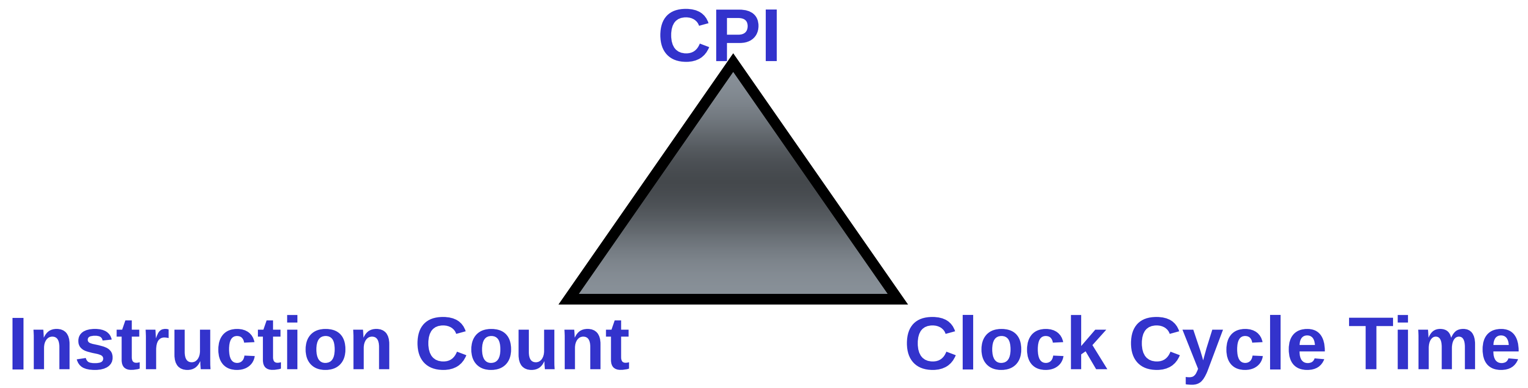
- CPI also gives insight into *style* of ISA:
  - RISC (e.g., MIPS, DEC Alpha, PowerPC) higher instruction count, lower CPI
  - CISC (e.g., Intel) lower instruction count, higher CPI

"Iron Triangle" of CPU Performance

– CPU execution time for program
  = Clock Cycles for program x Clock Cycle Time

– Substituting for clock cycles:

CPU execution time for program
= (Instruction Count x CPI)
x Clock Cycle Time
= Instruction Count x CPI x Clock Cycle Time



How Calculate the 3 Components?

- Clock Cycle Time: in specification of computer (Clock Rate in advertisements)

- Instruction Count:
  - count number of instructions executed in loop of small program
  - Use a simulator to count instructions
  - Use a hardware counter in special CPU "register" (e.g., Pentium II)

How Calculate the 3 Components?

- Average CPI:
  - Calculate:

    (1) Program Execution Time / Clock cycle time
    = Total Clock Cycles (for program)

    (2) $\frac{\text{Total Clock Cycles}}{\text{Instruction Count}}$
  - To determine average CPI must execute program!

Final thoughts: Performance Equation

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

Seconds / Program: Goal is to optimize execution time, not individual equation terms.

Instructions / Program: Machines are optimized with respect to program workloads.

Cycles / Instruction: The CPI of the program. Reflects the program's instruction mix.

Seconds / Cycle: Clock period. Optimize jointly with machine CPI.

Final thoughts: Performance Equation

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

	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
Inst. Set.	X	X	
Organization		X	X
Technology			X

Example

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

A program executed in machine A with a 1ns clock gives a CPI of 2.0. The same program with machine B having same ISA and a 2ns clock gives a CPI of 1.2. Which machine is faster and by how much?

Answer: Let I be the instruction count.

CPU clock cycles for A = I x 2.0

Execution time on A = 2 I ns

CPU clock cycles for B = I x 1.2

Execution time on B = I x 1.2 x 2 ns = 2.4 I ns

=> CPU A is faster by 1.2 times.

CPI

"Average cycles per instruction"

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

$$\text{CPI} = \frac{\text{total CPU clock cycles}}{\text{Instruction count}}$$

$$\text{total CPU clock cycles} = \sum_{\text{i}=1}^{\text{n}} IC_i \times CPI_i$$

IC_i is the instruction count of the ith instruction, CPI_i is the cycle per instruction of the ith instruction.

Example (RISC processor)

Base Machine (Reg / Reg)

Op	Freq	Cycles	CPI(i)	% Time
ALU	50%	1	.5	23%
Load	20%	5	1.0	45%
Store	10%	3	.3	14%
Branch	20%	2	.4	18%
			2.2	

Typical Mix

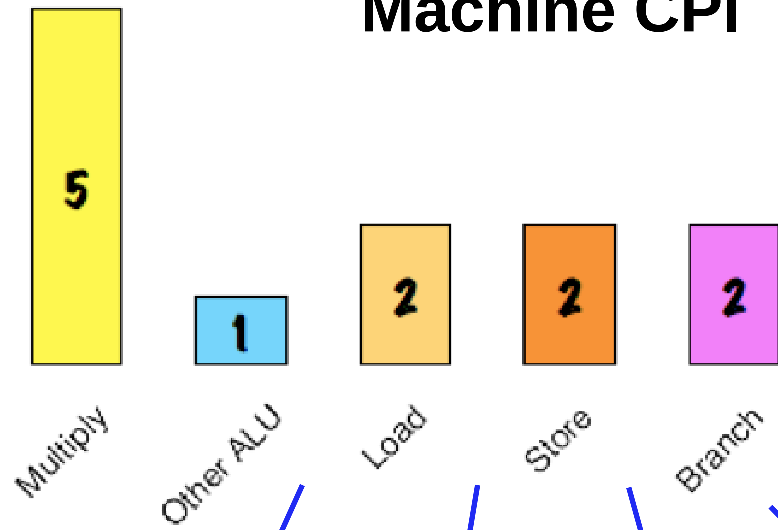
How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?

How does this compare with using branch prediction to shave a cycle off the branch time?

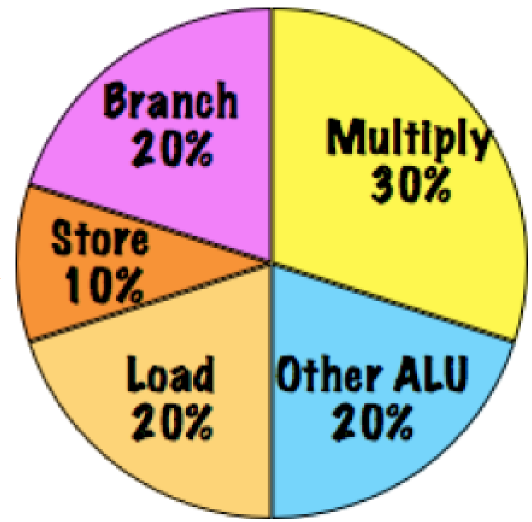
What if two ALU instructions could be executed at once?

CPI as an analytical tool to guide design

Machine CPI

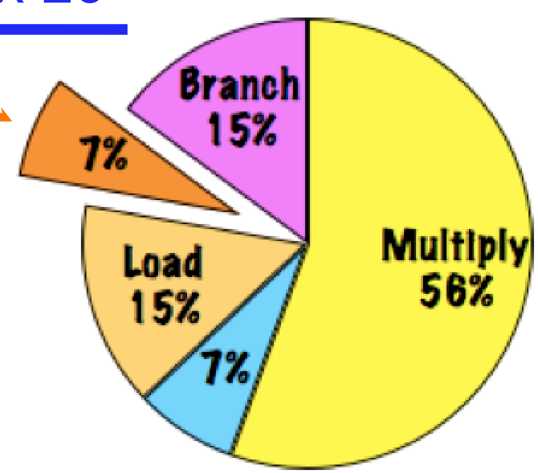


Program Instruction Mix



$$\frac{5 \times 30 + 1 \times 20 + 2 \times 20 + 2 \times 10 + 2 \times 20}{100}$$

= 2.7 cycles/instruction



Where program spends its time

Performance Evaluation Techniques

- Measurements – Only for the given machine
- Simulation – Answers what if questions

  accurate execution-driven simulators widely used for computer architecture research

EX: Simplescalar

What Programs Measure for Comparison?

- User reality: CPI varies with program, workload mix, OS, compiler, etc.
- Ideally would run typical programs with typical input before purchase
- Called a "workload"; For example:
  - Engineer uses compiler, spreadsheet
  - Author uses word processor, drawing program, compression software
- In some situations its hard to do
  - Don't have access to machine to "benchmark" before purchase
  - Don't know workload in future

Basis of Evaluation

Pros		Cons
• representative	Actual Target Workload	• very specific • non-portable • difficult to run, or measure • hard to identify cause
• portable • widely used • improvements useful in reality	Full Application Benchmarks	•less representative
• easy to run, early in design cycle	Small “Kernel” Benchmarks	• easy to “fool”
• identify peak capability and potential bottlenecks	Microbenchmarks	• “peak” may be a long way from application performance

Performance Summary

- Two performance metrics execution time and throughput.
- Measuring CPU time: CPI

CPU time = Instruction count x CPI x clock cycle time

CPU time = Instruction count x CPI / clock rate