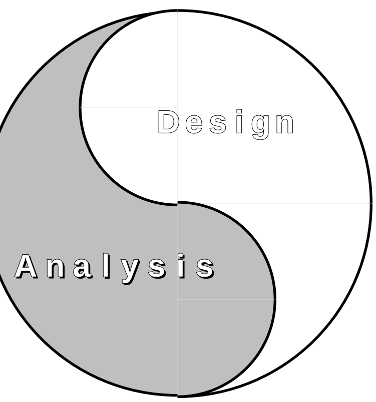
Lecture 3: Architectural Performance Laws and Rules of Thumb

Prof. Randy H. Katz

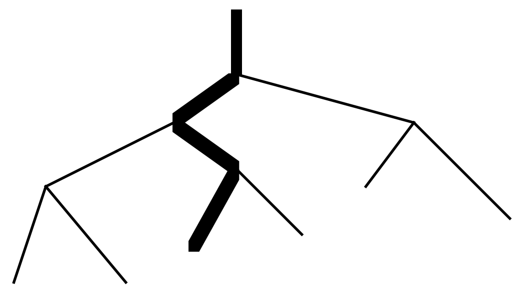
Computer Science 252

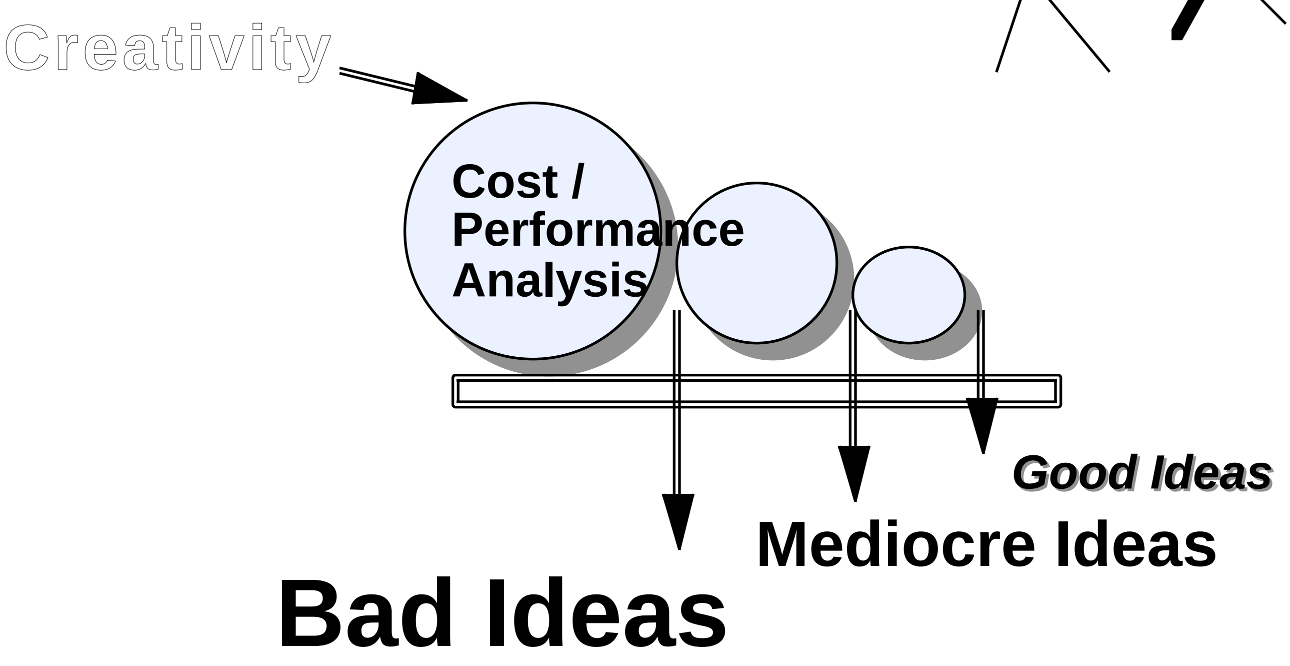
Spring 1996

Measurement and Evaluation



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





Measurement Tools

- **Benchmarks, Traces, Mixes**
- **Cost, delay, area, power estimation**
- **Simulation (many levels)**
  - **ISA, RT, Gate, Circuit**
- **Queuing Theory**
- **Rules of Thumb**
- **Fundamental Laws**

The Bottom Line: Performance (and Cost)

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

- **Time to run the task (ExTime)**
  - **Execution time, response time, latency**
- **Tasks per day, hour, week, sec, ns … (Performance**
  - **Throughput, bandwidth**

The Bottom Line: Performance (and Cost)

"X is n times faster than Y" means

```
ExTime(Y)        Performance(X)
---------   =   ---------------
ExTime(X)        Performance(Y)
```

- **Speed of Concorde vs. Boeing 747**

- **Throughput of Boeing 747 vs. Concorde**

Performance Terminology

"X is n% faster than Y" means:

$$\frac{\text{ExTime}(Y)}{\text{ExTime}(X)} = \frac{\text{Performance}(X)}{\text{Performance}(Y)} = 1 + \frac{n}{100}$$

$$n = \frac{100(\text{Performance}(X) - \text{Performance}(Y))}{\text{Performance}(Y)}$$

Example: Y takes 15 seconds to complete a task, X takes 10 seconds. What % faster is X?

Example

$$\frac{\text{ExTime(Y)}}{\text{ExTime(X)}} = \frac{15}{10} = \frac{1.5}{1.0} = \frac{\text{Performance (X)}}{\text{Performance (Y)}}$$

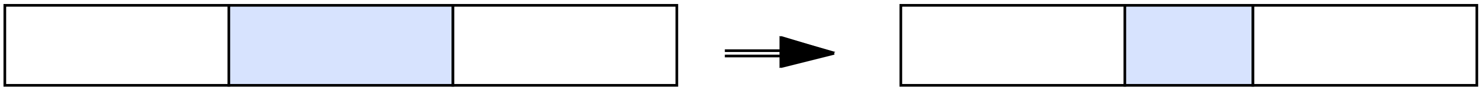
$$n = \frac{100\ (1.5\ -\ 1.0)}{1.0}$$

$$n = 50\%$$

Amdahl's Law

Speedup due to enhancement E:

```
                ExTime w/o E          Performance w/  E
Speedup(E) = -------------    =    -------------------
                ExTime w/  E          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(E)  =
Speedup(E) =
```

Amdahl's Law

$$\text{xTime}_{new} = \text{ExTime}_{old} \times \left[(1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}} \right]$$

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

Amdahl's Law

- **Floating point instructions improved to run 2X; but only 10% of actual instructions are FP**

$$\mathbf{ExTime_{new}} =$$

$$\mathbf{Speedup_{overall}} =$$

Amdahl's Law

- **Floating point instructions improved to run 2X; but only 10% of actual instructions are FP**

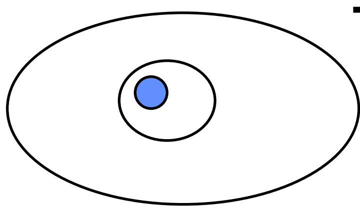
$$\text{ExTime}_{new} = \text{ExTime}_{old} \times (0.9 + .1/2) = 0.95 \times \text{ExTime}_{old}$$

$$\text{Speedup}_{overall} = \frac{1}{0.95} = 1.053$$

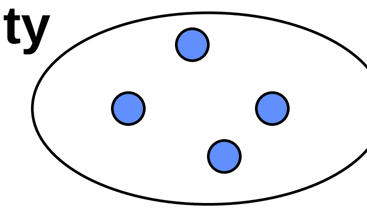
Corollary: Make The Common Case Fast

- **All instructions require an instruction fetch, only a fraction require a data fetch/store.**
  - **Optimize instruction access over data access**
- **Programs exhibit *locality***

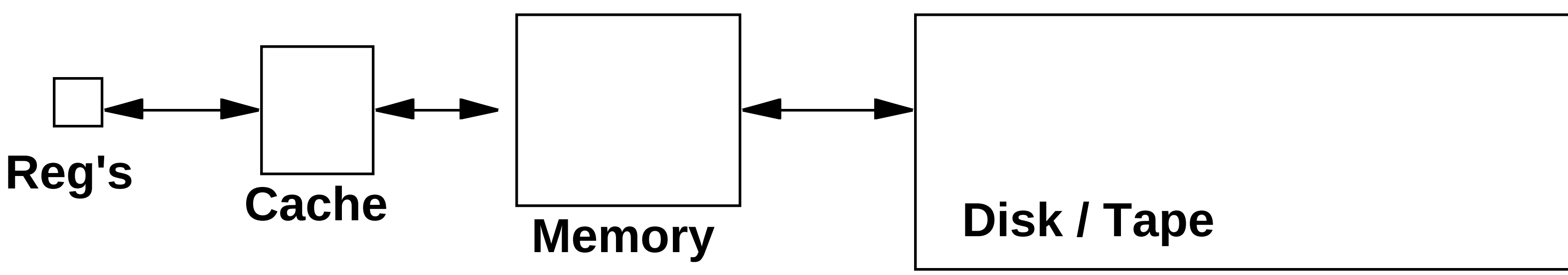
Spatial Locality



Temporal Locality



- **Access to small memories is faster**
  - **Provide a *storage hierarchy* such that the most frequent accesses are to the smallest (closest) memories.**

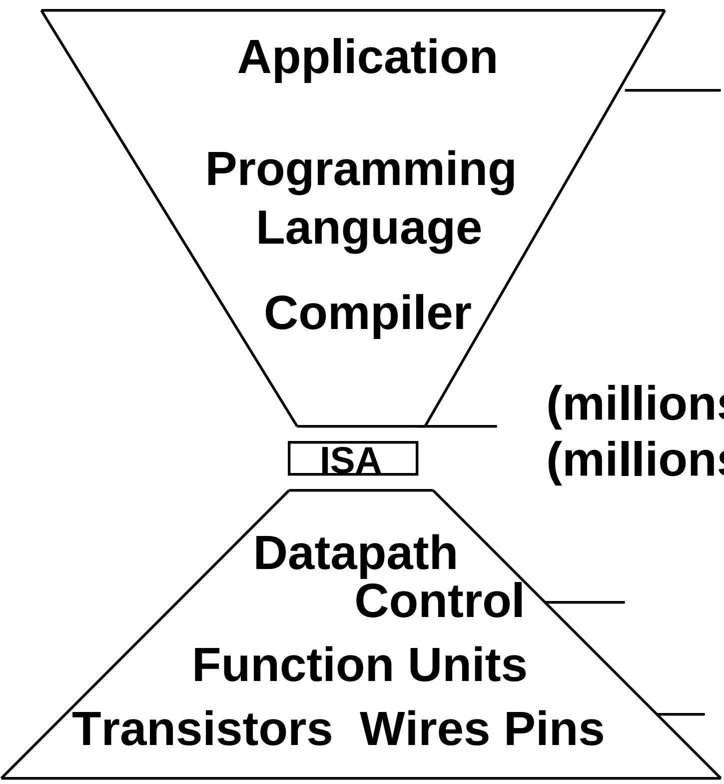


Occam's Toothbrush

- **The simple case is usually the most frequent and the easiest to optimize!**

- **Do simple, fast things in hardware and be sure the rest can be handled correctly in software**

Metrics of Performance



Answers per month
Operations per second

(millions) of Instructions per second: MIPS
(millions) of (FP) operations per second: MFLOP

Megabytes per second

Cycles per second (clock rate)

Aspects of CPU Performance

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

	Instr. Cnt	CPI	Clock Rate
Program			
Compiler			
Instr. Set			
Organization			
Technology			

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

Normalized:

add,sub,compare,mult

divide, sqrt

exp, sin, . . .

Cycles Per Instruction

Average Cycles per Instruction"

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

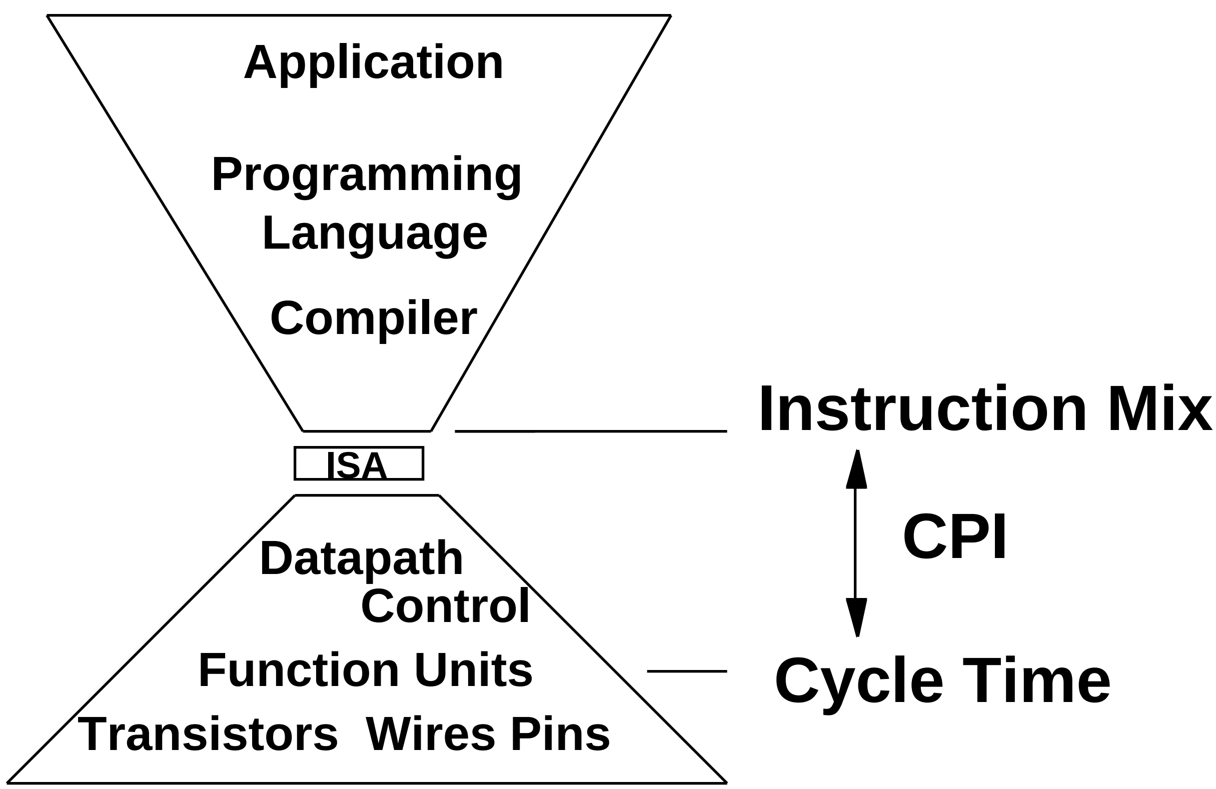
$$\text{CPU time} = \text{CycleTime} * \sum_{i=1}^{n} CPI_i * I_i$$

"Instruction Frequency"

$$CPI = \sum_{i=1}^{n} CPI_i * F_i \quad \text{where } F_i = \frac{I_i}{\text{Instruction Count}}$$

Invest Resources where time is Spent!

Organizational Trade-offs



Example: Calculating CPI

Base Machine (Reg / Reg)

Op	Freq	Cycles	CPI(i)	(% Time)
ALU	50%	1	.5	(33%)
Load	20%	2	.4	(27%)
Store	10%	2	.2	(13%)
Branch	20%	2	.4	(27%)
			1.5	

Typical Mix

Example

Add register / memory operations:

- One source operand in memory
- One source operand in register
- Cycle count of 2

Branch cycle count to increase to 3.

What fraction of the loads must be eliminated for thi to pay off?

Base Machine (Reg / Reg)

Op	Freq	Cycles
ALU	50%	1
Load	20%	2
Store	10%	2
Branch	20%	2

Typical Mix