

CS430- Computer Architecture
Lecture 2
September 2001

William J. Taffe

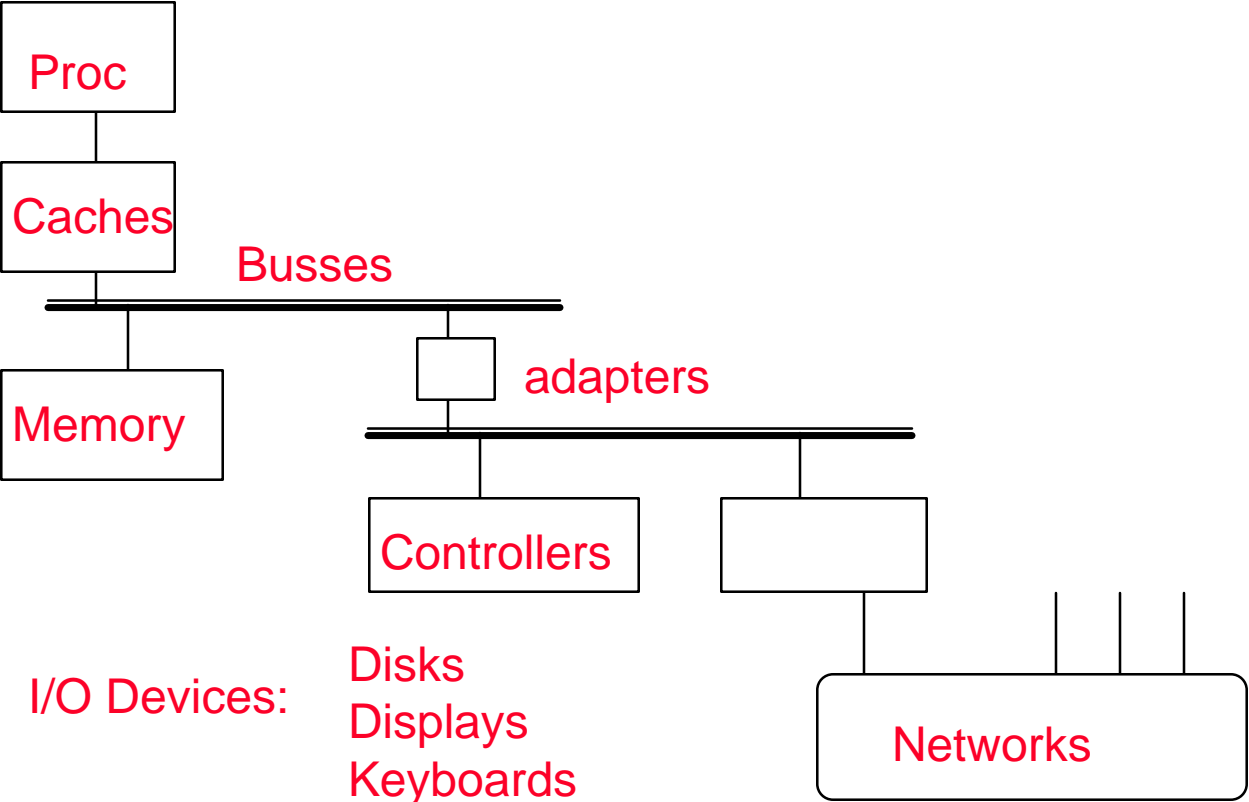
using slides of

Dave Patterson

Review: Computer Organization

- **All computers consist of five components**
 - **Processor: (1) datapath and (2) control**
 - **(3) Memory**
 - **(4) Input devices and (5) Output devices**
- **Not all “memory” is created equally**
 - **Cache: fast (expensive) memory are placed closer to the processor**
 - **Main memory: less expensive memory--we can have more**
- **Input and output (I/O) devices have the messiest organization**
 - **Wide range of speed: graphics vs. keyboard**
 - **Wide range of requirements: speed, standard, cost ...**
 - **Least amount of research (so far)**

Summary: Computer System Components



◦ All have interfaces & organizations

Performance

- **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 is to understand cost & performance implications of architectural choices**

Two notions of “performance”

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

Which has higher performance?

- ° **Time to do the task (Execution Time)**
 - execution time, response time, **latency**
- ° **Tasks per day, hour, week, sec, ns. .. (Performance)**
 - **throughput**, bandwidth

Response time and throughput often are in opposition

Definitions

- Performance is in units of things-per-second
 - bigger is better
- If we are primarily concerned with response time
 - $\text{performance}(x) = \frac{1}{\text{execution_time}(x)}$
 - smaller is better

" X is n times faster than Y" means

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

$$n = \frac{\text{Execution time (Y)}}{\text{Execution time (X)}}$$

Example

- Time of Concorde vs. Boeing 747?
 - Concord is $1350 \text{ mph} / 610 \text{ mph} = 2.2$ **times faster**
 $= 6.5 \text{ hours} / 3 \text{ hours}$
- Throughput of Concorde vs. Boeing 747 ?
 - Concord is $178,200 \text{ pmph} / 286,700 \text{ pmph} = 0.62$ “times faster”
 - Boeing is $286,700 \text{ pmph} / 178,200 \text{ pmph} = 1.6$ “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

Benchmarks – Basis of Evaluation

Pros

- representative

- portable
- widely used
- improvements useful in reality

easy to run, early in design cycle

identify peak capability and potential bottlenecks

Cons

- very specific
- non-portable
- difficult to run, or measure
- hard to identify cause

less representative

easy to “fool”

“peak” may be a long way from application performance

Actual Target Workload

Full Application Benchmarks

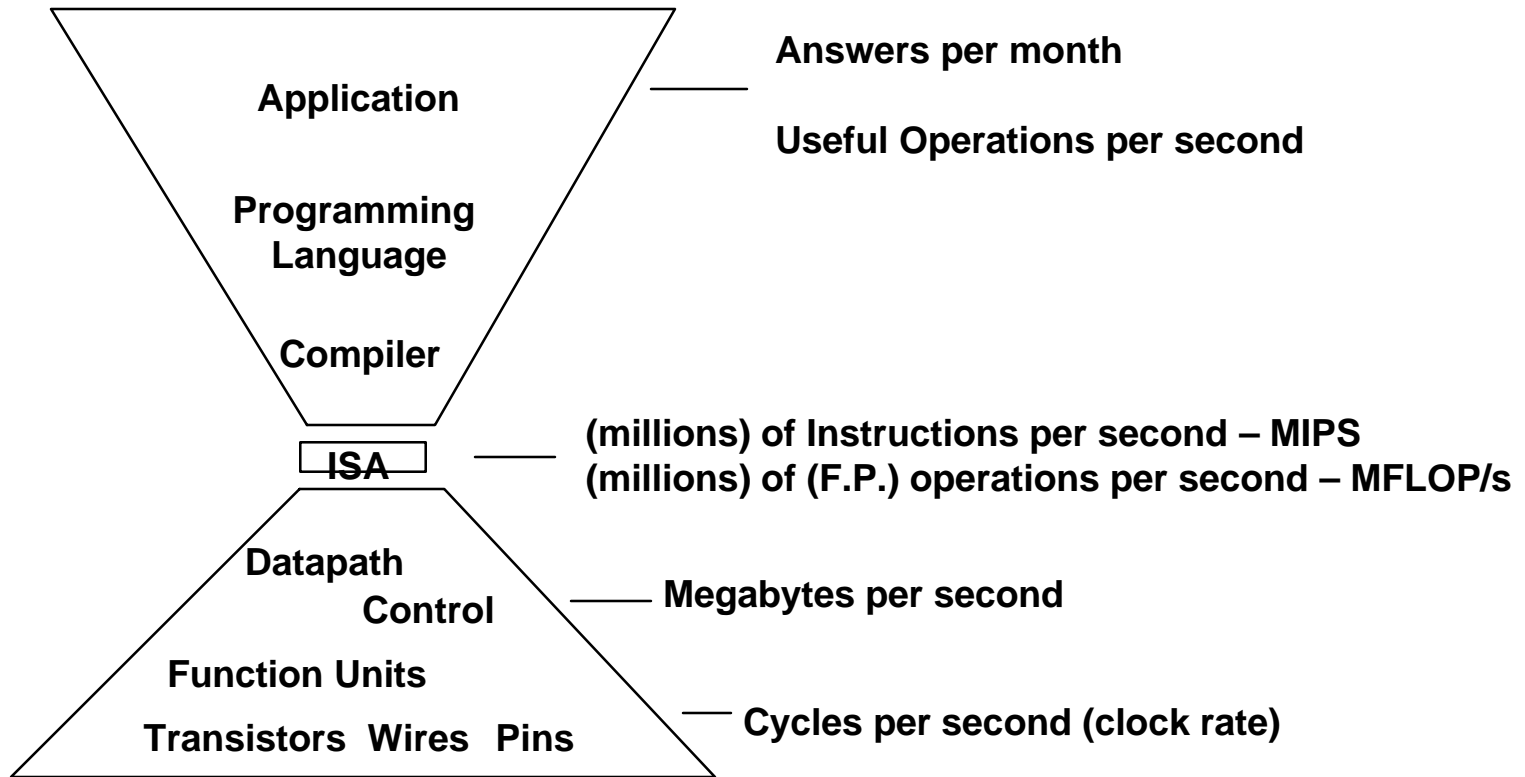
Small “Kernel” Benchmarks

Microbenchmarks

SPEC95

- **Eighteen application benchmarks (with inputs) reflecting a technical computing workload**
- **Eight integer**
 - **go, m88ksim, gcc, compress, li, jpeg, perl, vortex**
- **Ten floating-point intensive**
 - **tomcatv, swim, su2cor, hydro2d, mgrid, applu, turb3d, apsi, fppp, wave5**
- **Must run with standard compiler flags**
 - **eliminate special undocumented incantations that may not even generate working code for real programs**

Metrics of performance



Each metric has a place and a purpose, and each can be misused

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. count	CPI	clock rate
Program			
Compiler			
Instr. Set Arch.			
Organization			
Technology			

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 count	CPI	clock rate
Program	X		
Compiler	X	X	
Instr. Set	X	X	
Organization		X	X
Technology			X

CPI “Average cycles per instruction”

$$\begin{aligned}\text{CPI} &= (\text{CPU Time} * \text{Clock Rate}) / \text{Instruction Count} \\ &= \text{Clock Cycles} / \text{Instruction Count}\end{aligned}$$

$$\text{CPU time} = \text{ClockCycleTime} * \sum_{i=1}^n \text{CPI}_i * I_i$$

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

"instruction frequency"

Invest Resources where time is Spent!

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%
			<hr/>	
			2.2	

Typical Mix

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

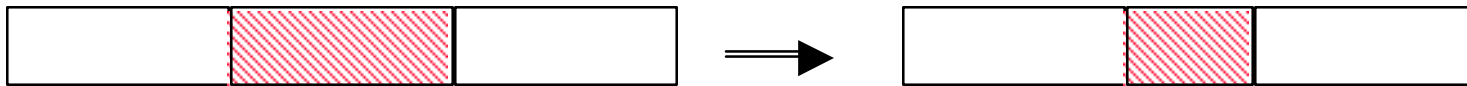
How does this compare with using branch prediction to shave a cycle off the branch time?

What if two ALU instructions could be executed at once?

Amdahl's Law

Speedup due to enhancement E:

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



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

$$\text{ExTime}(w/ E) = ((1-F) + F/S) \times \text{ExTime}(w/o E)$$

$$\text{Overall Speedup}(w/ E) = \frac{1}{(1-F) + F/S}$$

Summary: Evaluating Instruction Sets?

Design-time metrics:

- Can it be implemented, in how long, at what cost?
- Can it be programmed? Ease of compilation?

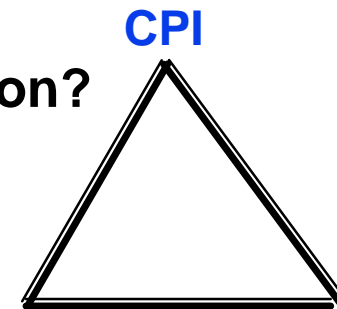
Static Metrics:

- 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!



Inst. Count

Cycle Time

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