EECS 252 Graduate Computer Architecture

Lec 1 - Introduction

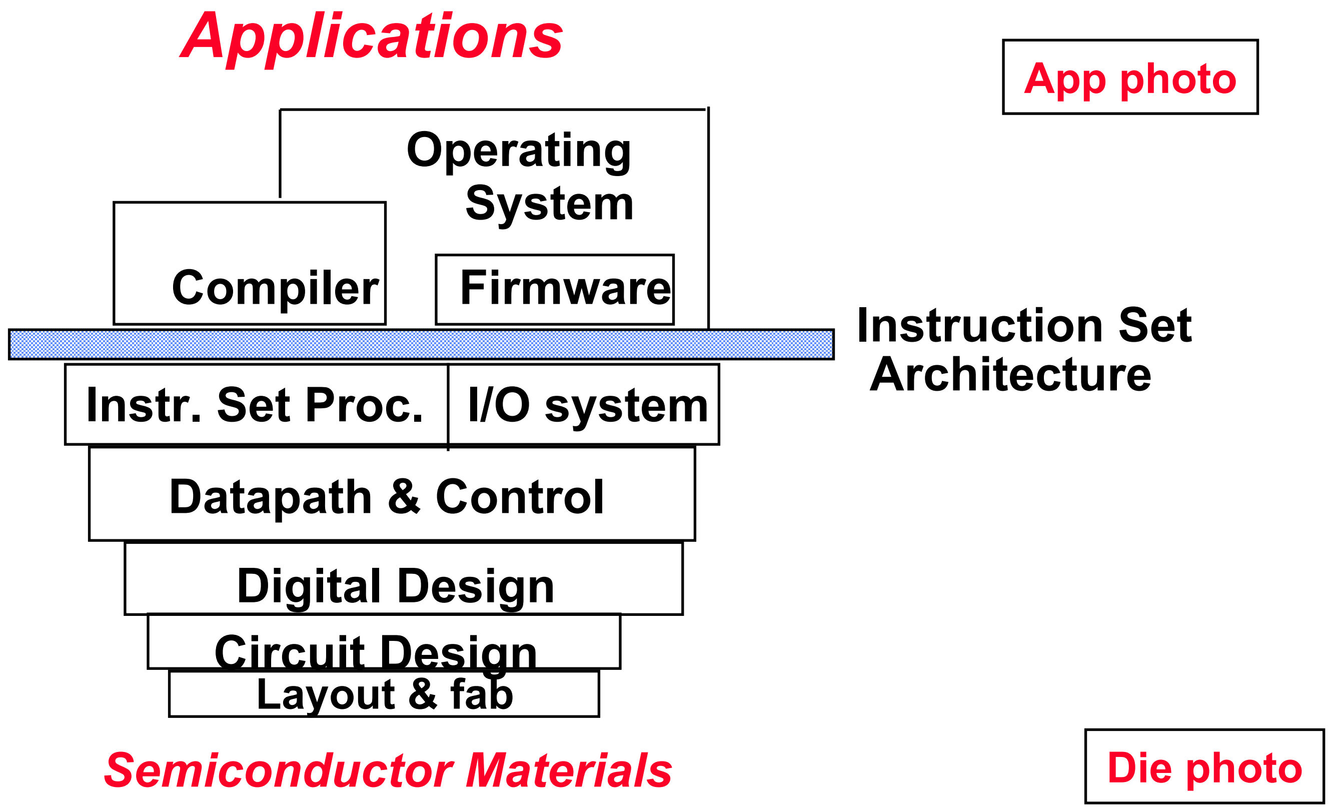
David Culler
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~culler
http://www-inst.eecs.berkeley.edu/~cs252

Outline

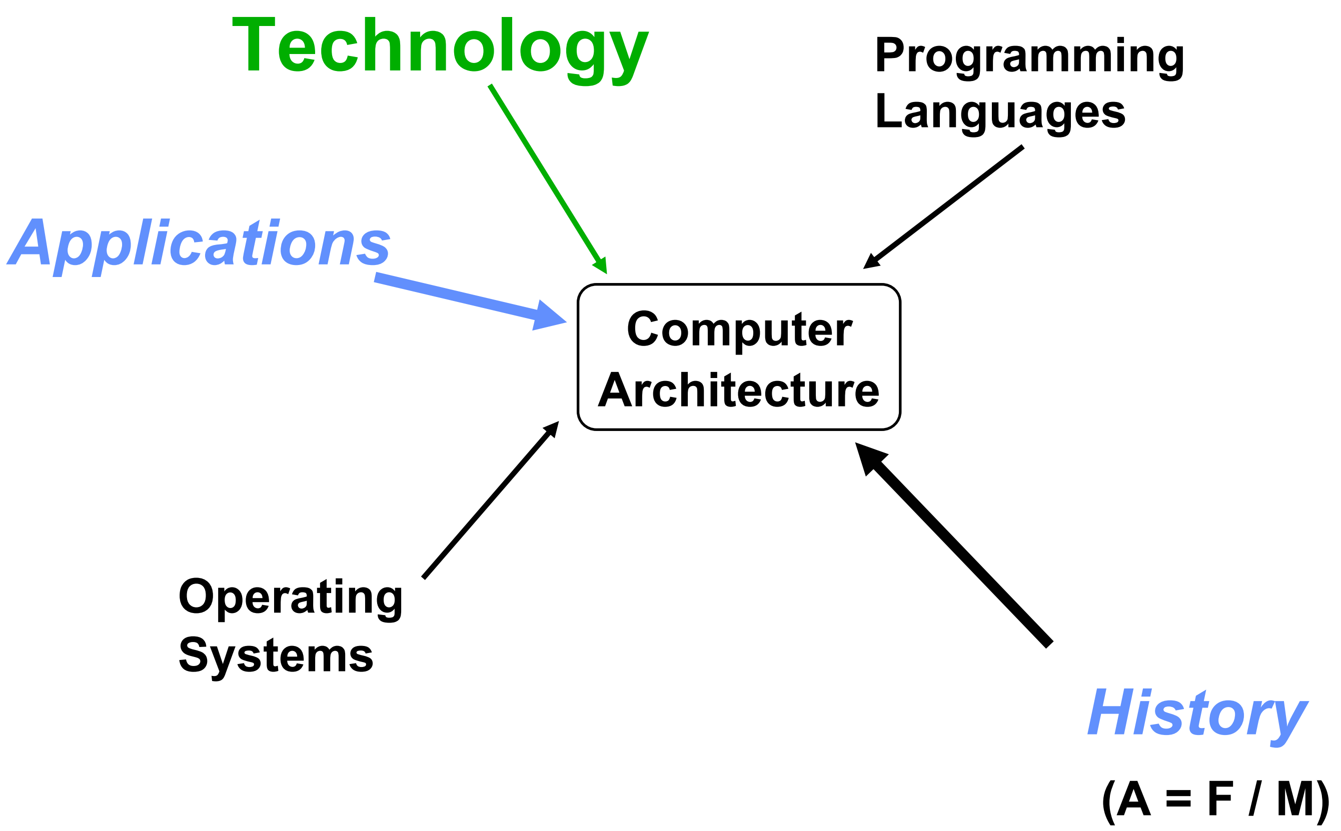
- What is Computer Architecture?
- Computer Instruction Sets – the fundamental abstraction
  - review and set up
- Dramatic Technology Advance
- Beneath the illusion – nothing is as it appears
- Computer Architecture Renaissance
- How would you like your CS252?

What is "Computer Architecture"?

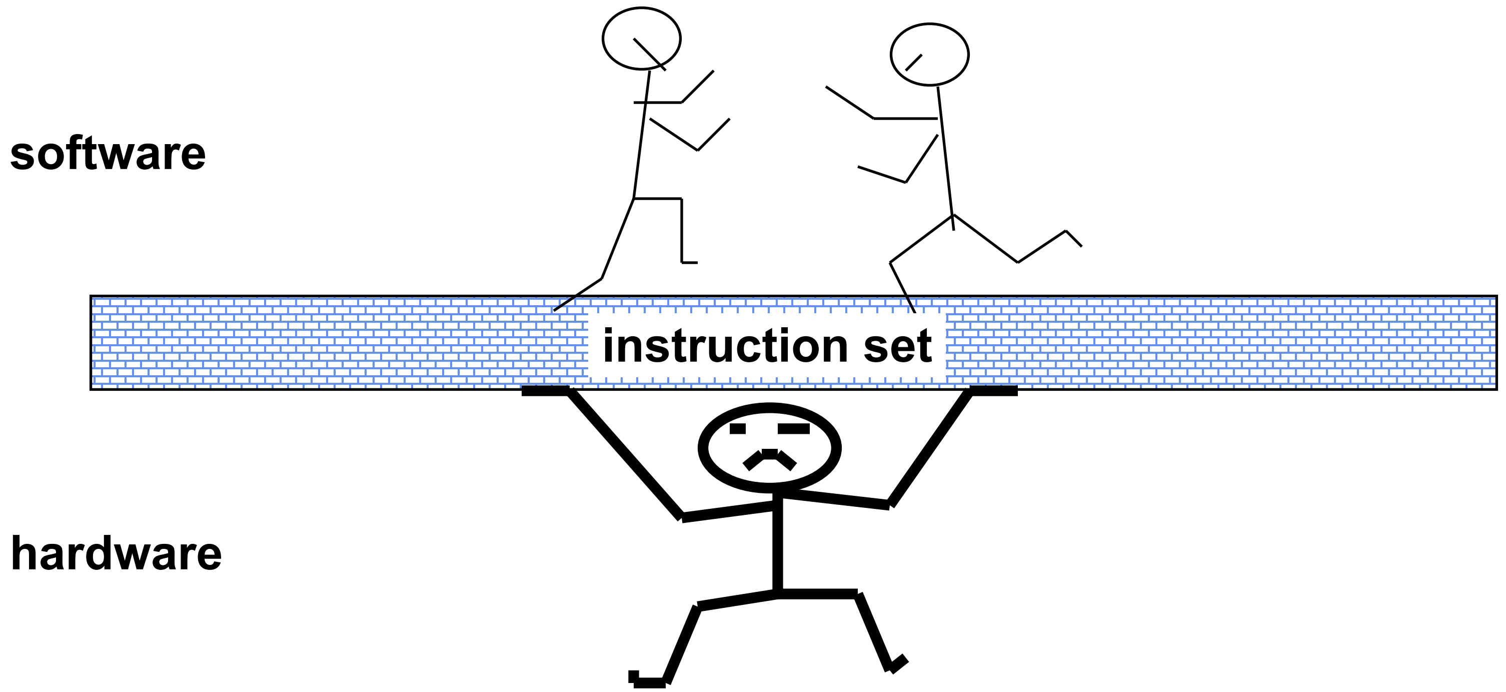


- Coordination of many *levels of abstraction*
- Under a rapidly changing set of forces
- Design, Measurement, *and* Evaluation

Forces on Computer Architecture



The Instruction Set: a Critical Interface

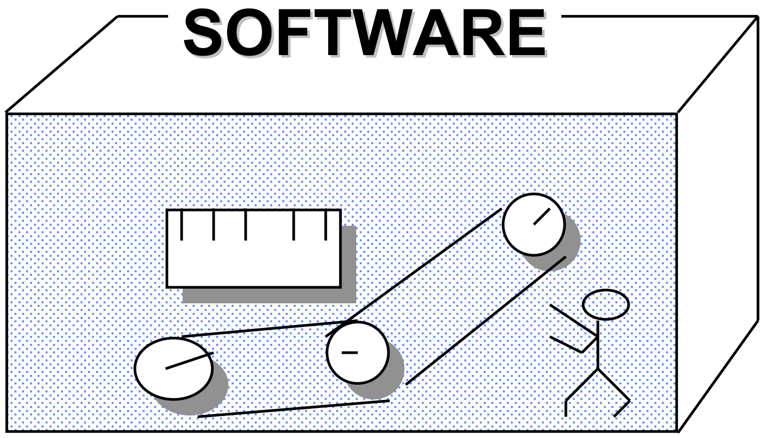


- Properties of a good abstraction
  - Lasts through many generations (portability)
  - Used in many different ways (generality)
  - Provides convenient functionality to higher levels
  - Permits an efficient implementation at lower levels

Instruction Set Architecture

... the attributes of a [computing] system as seen by the programmer, *i.e.* the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls the logic design, and the physical implementation.

– Amdahl, Blaaw, and Brooks, 1964



-- Organization of Programmable Storage

-- Data Types & Data Structures: Encodings & Representations

-- Instruction Formats

-- Instruction (or Operation Code) Set

-- Modes of Addressing and Accessing Data Items and Instructions

-- Exceptional Conditions

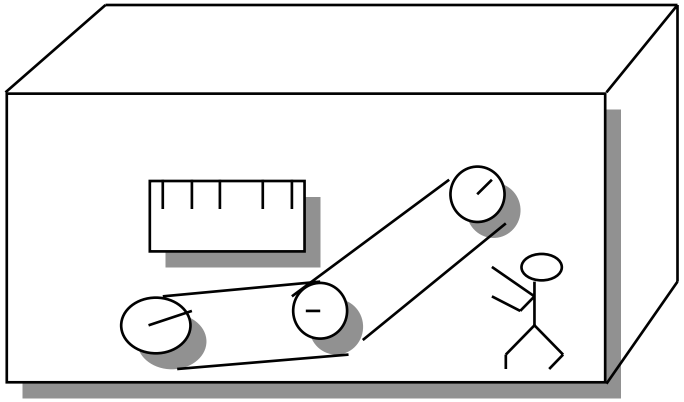
Computer Organization

- Capabilities & Performance Characteristics of Principal Functional Units
  - (e.g., Registers, ALU, Shifters, Logic Units, ...)
- Ways in which these components are interconnected
- Information flows between components
- Logic and means by which such information flow is controlled.
- Choreography of FUs to realize the ISA
- Register Transfer Level (RTL) Description

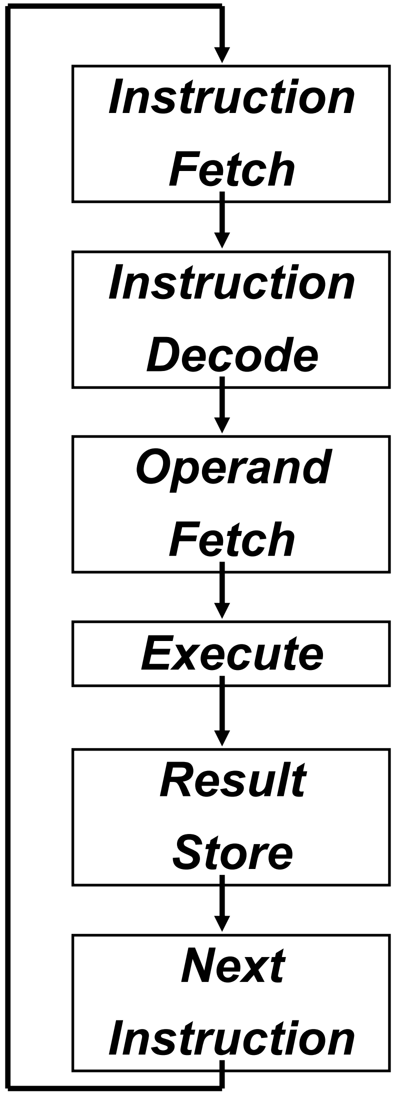
Logic Designer's View

ISA Level

FUs & Interconnect



Fundamental Execution Cycle



Obtain instruction from program storage

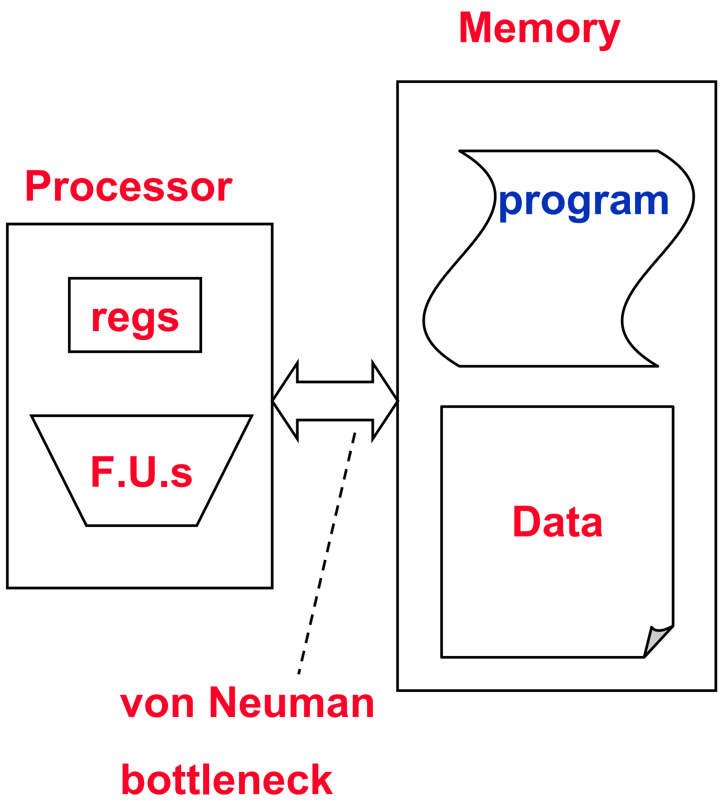
Determine required actions and instruction size

Locate and obtain operand data

Compute result value or status

Deposit results in storage for later use

Determine successor instruction

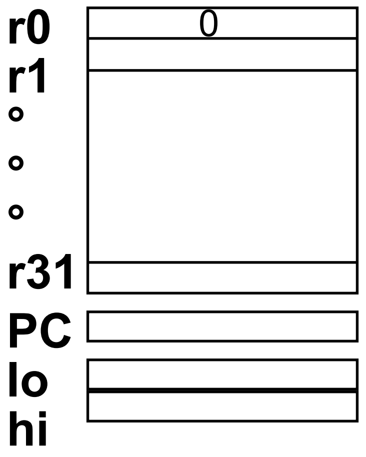


Elements of an ISA

- Set of machine-recognized data types
  - bytes, words, integers, floating point, strings, . . .
- Operations performed on those data types
  - Add, sub, mul, div, xor, move, ….
- Programmable storage
  - regs, PC, memory
- Methods of identifying and obtaining data referenced by instructions (addressing modes)
  - Literal, reg., absolute, relative, reg + offset, …
- Format (encoding) of the instructions
  - Op code, operand fields, …

$$\boxed{\text{Current Logical State of the Machine}} \vdash \boxed{\text{Next Logical State of the Machine}}$$

Example: MIPS R3000



Programmable storage

2^32 x <u>bytes</u>

31 x 32-bit GPRs (R0=0)

32 x 32-bit FP regs (paired DP)

HI, LO, PC

Data types ?

Format ?

Addressing Modes?

Arithmetic logical

Add, AddU, Sub, SubU, And, Or, Xor, Nor, SLT, SLTU,

AddI, AddIU, SLTI, SLTIU, AndI, OrI, XorI, *LUI*

SLL, SRL, SRA, SLLV, SRLV, SRAV

Memory Access

LB, LBU, LH, LHU, LW, LWL,LWR

SB, SH, SW, SWL, SWR

Control

J, JAL, JR, JALR

BEq, BNE, BLEZ,BGTZ,BLTZ,BGEZ,BLTZAL,BGEZAL

32-bit instructions on word boundary

Evolution of Instruction Sets

Single Accumulator (EDSAC 1950)

|

Accumulator + Index Registers

(Manchester Mark I, IBM 700 series 1953)

|

Separation of Programming Model
from Implementation

High-level Language Based (Stack)
(B5000 1963)

Concept of a Family
(IBM 360 1964)

General Purpose Register Machines

Complex Instruction Sets

(Vax, Intel 432 1977-80)

Load/Store Architecture
(CDC 6600, Cray 1 1963-76)

|

RISC

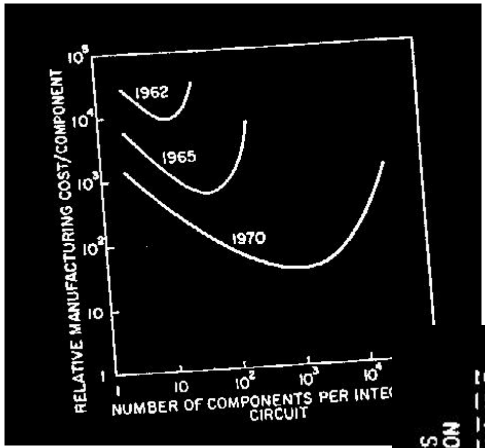
(MIPS,Sparc,HP-PA,IBM RS6000, 1987)

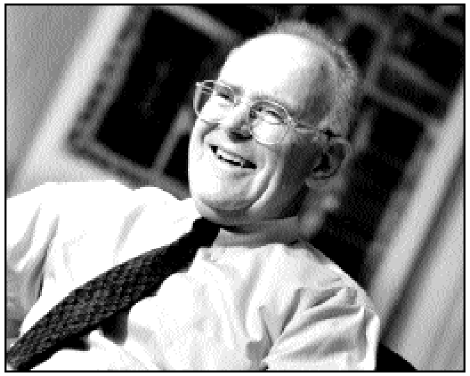
iX86?

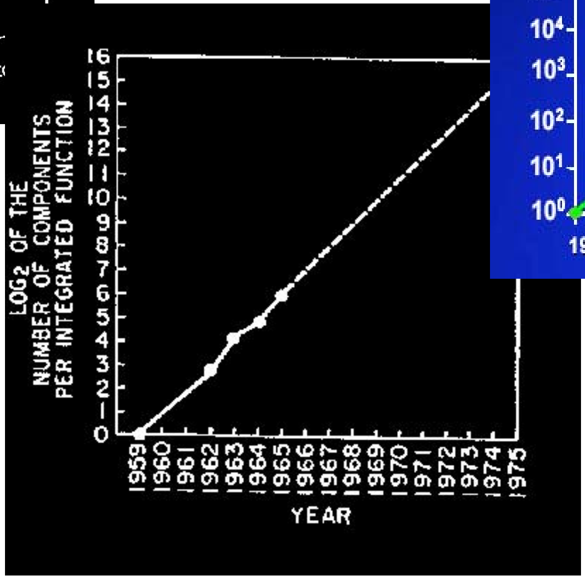
Dramatic Technology Advance

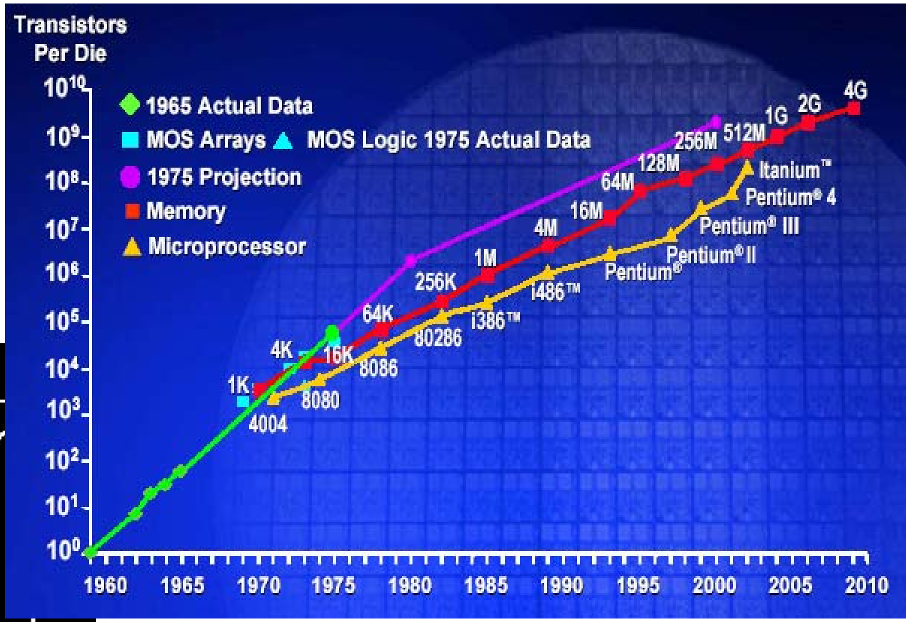
- Prehistory: Generations
  - 1st Tubes
  - 2nd Transistors
  - 3rd Integrated Circuits
  - 4th VLSI….
- Discrete advances in each generation
  - Faster, smaller, more reliable, easier to utilize
- Modern computing: Moore's Law
  - Continuous advance, fairly homogeneous technology

Moore’s Law



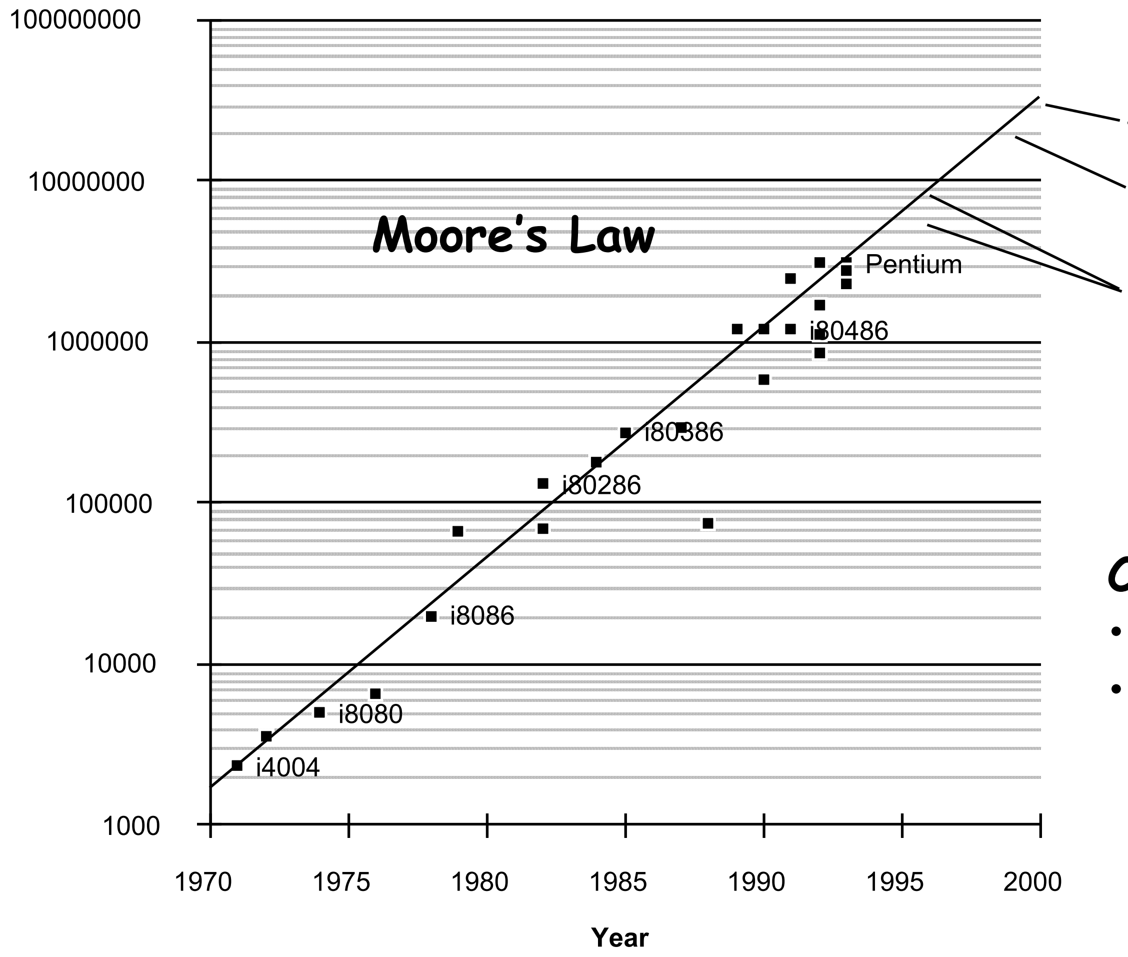






- “Cramming More Components onto Integrated Circuits”
  - Gordon Moore, Electronics, 1965
- # on transistors on cost-effective integrated circuit double every 18 months

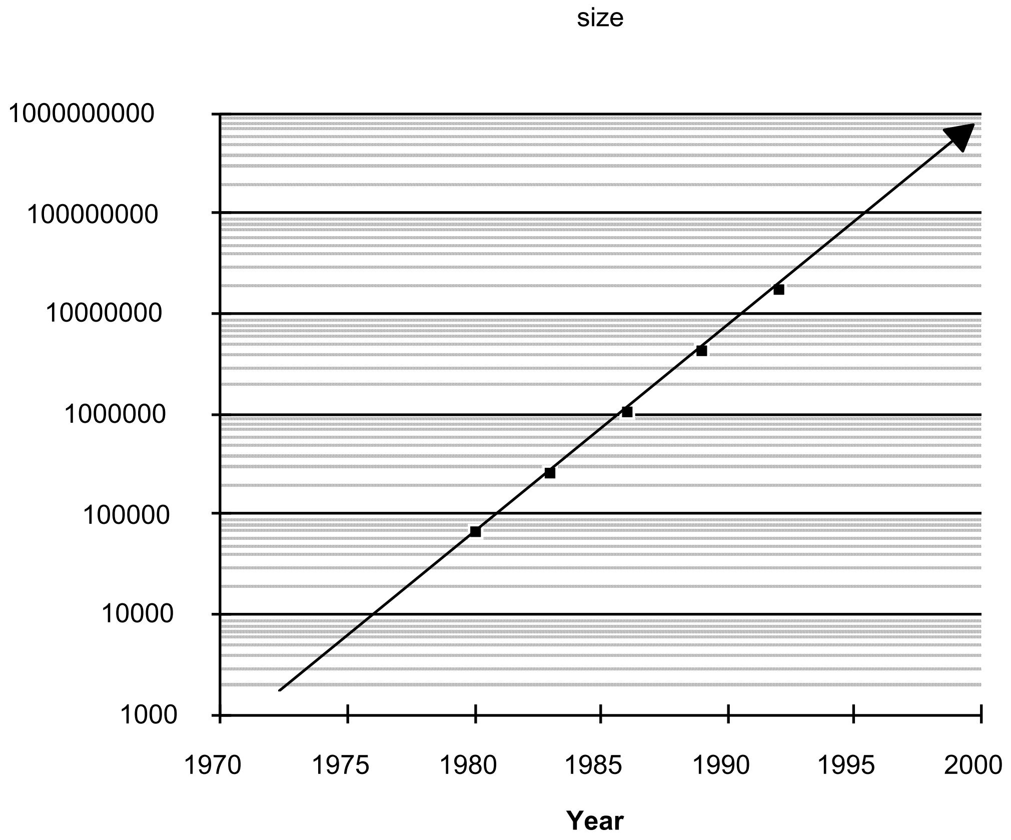
Technology Trends: Microprocessor Capacity



Itanium II: 241 million
Pentium 4: 55 million
Alpha 21264: 15 million
Pentium Pro: 5.5 million
PowerPC 620: 6.9 million
Alpha 21164: 9.3 million
Sparc Ultra: 5.2 million

CMOS improvements:
- **Die size: 2X every 3 yrs**
- **Line width: halve / 7 yrs**

Memory Capacity (Single Chip DRAM)



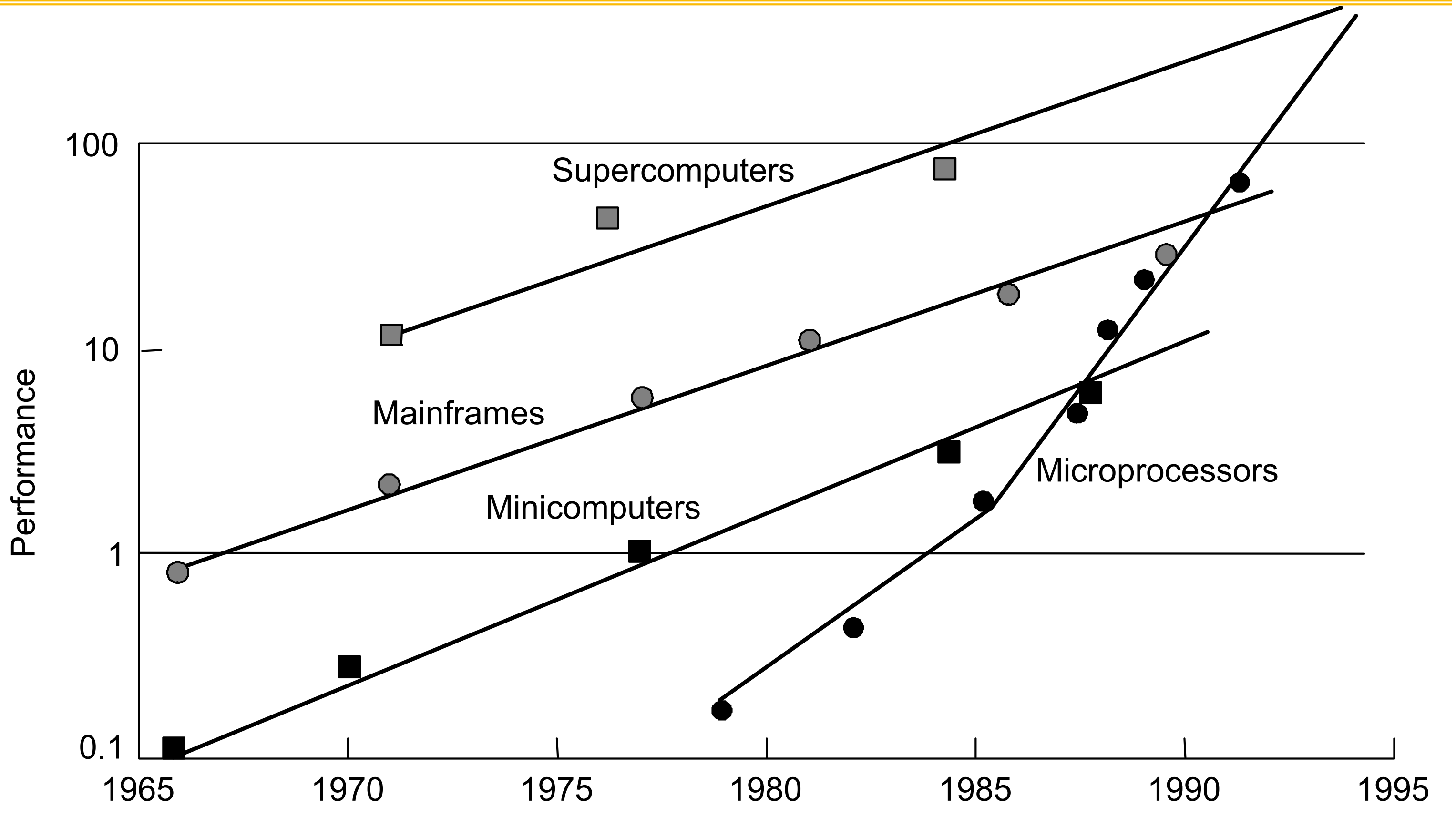
year	size(Mb)	cyc time
1980	0.0625	250 ns
1983	0.25	220 ns
1986	1	190 ns
1989	4	165 ns
1992	16	145 ns
1996	64	120 ns
2000	256	100 ns
2003	1024	60 ns

Technology Trends

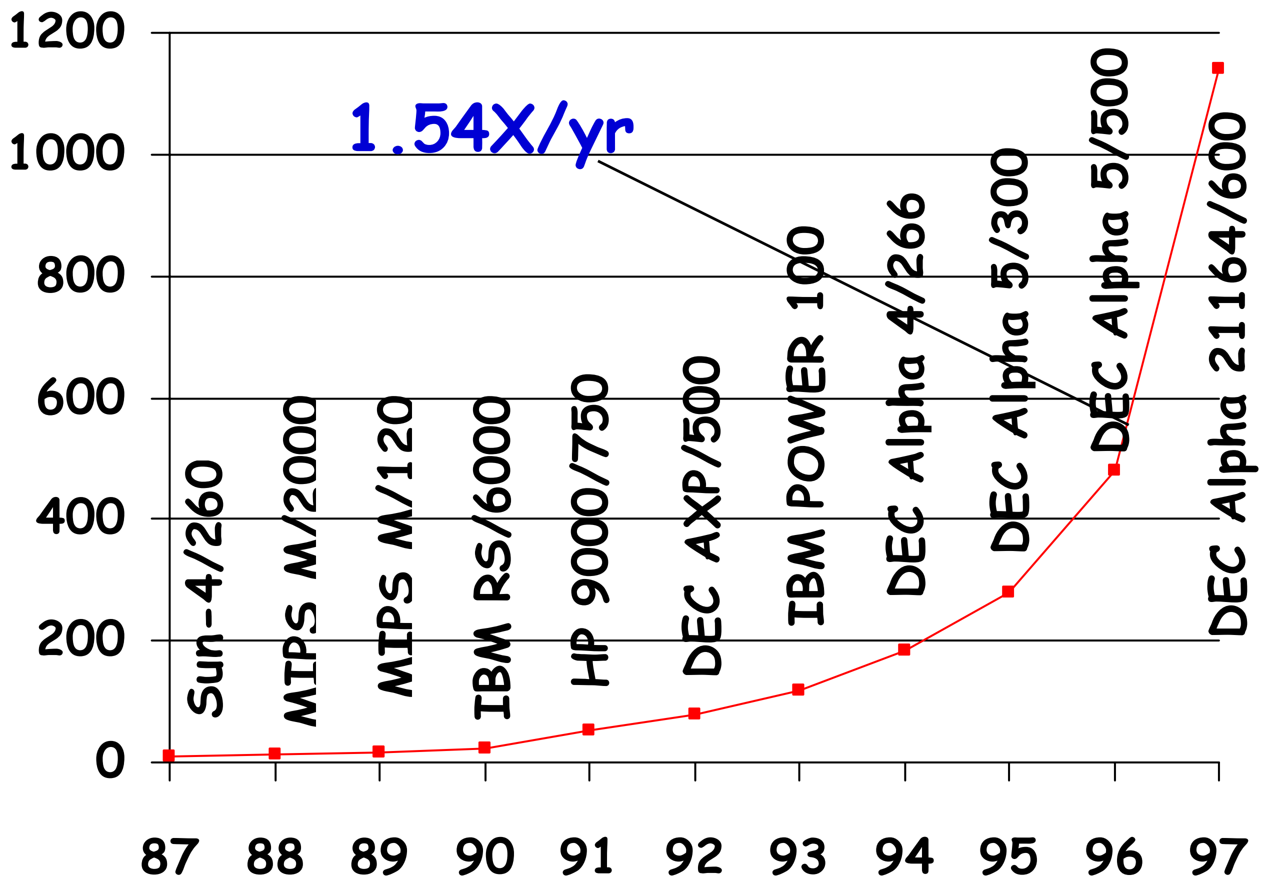
- Clock Rate: ~30% per year
- Transistor Density: ~35%
- Chip Area: ~15%
- Transistors per chip: ~55%
- Total Performance Capability: ~100%
- by the time you graduate...
  - 3x clock rate (~10 GHz)
  - 10x transistor count (10 Billion transistors)
  - 30x raw capability

- plus 16x dram density,
- 32x disk density (60% per year)
- Network bandwidth, …

Performance Trends



Processor Performance (1.35X before, 1.55X now)



Definition: Performance

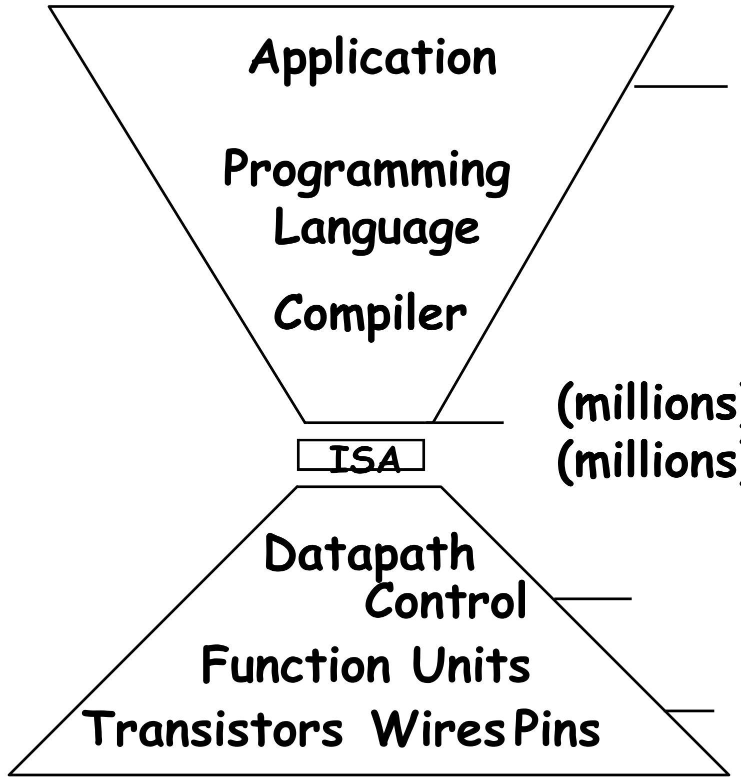
- Performance is in units of things per sec
  - bigger is better
- If we are primarily concerned with response time

$$\text{performance}(x) = \frac{1}{\text{execution\_time}(x)}$$

" X is n times faster than Y" means

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

Metrics of Performance



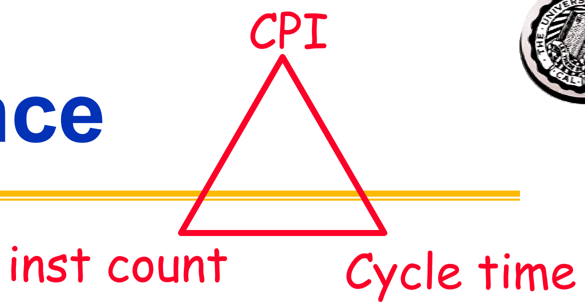
Answers per day/month

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

Megabytes per second

Cycles per second (clock rate)

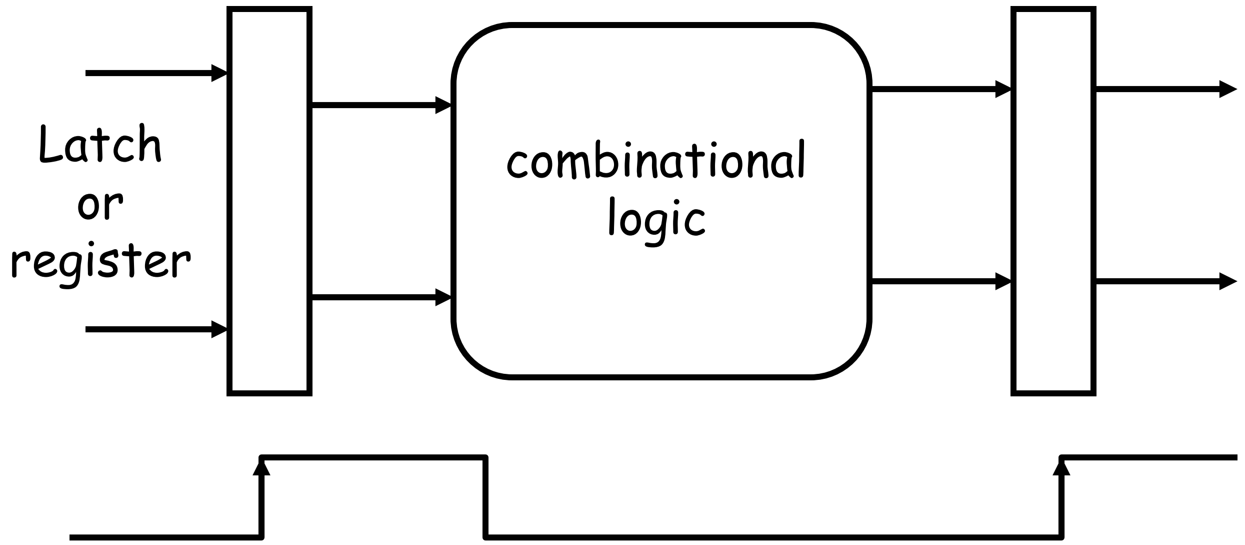
Components of Performance



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

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

What's a Clock Cycle?



- **Old days: 10 levels of gates**
- **Today: determined by numerous time-of-flight issues + gate delays**
  - **clock propagation, wire lengths, drivers**

Integrated Approach

What really matters is the functioning of the complete system, I.e. hardware, runtime system, compiler, and operating system

In networking, this is called the "End to End argument"

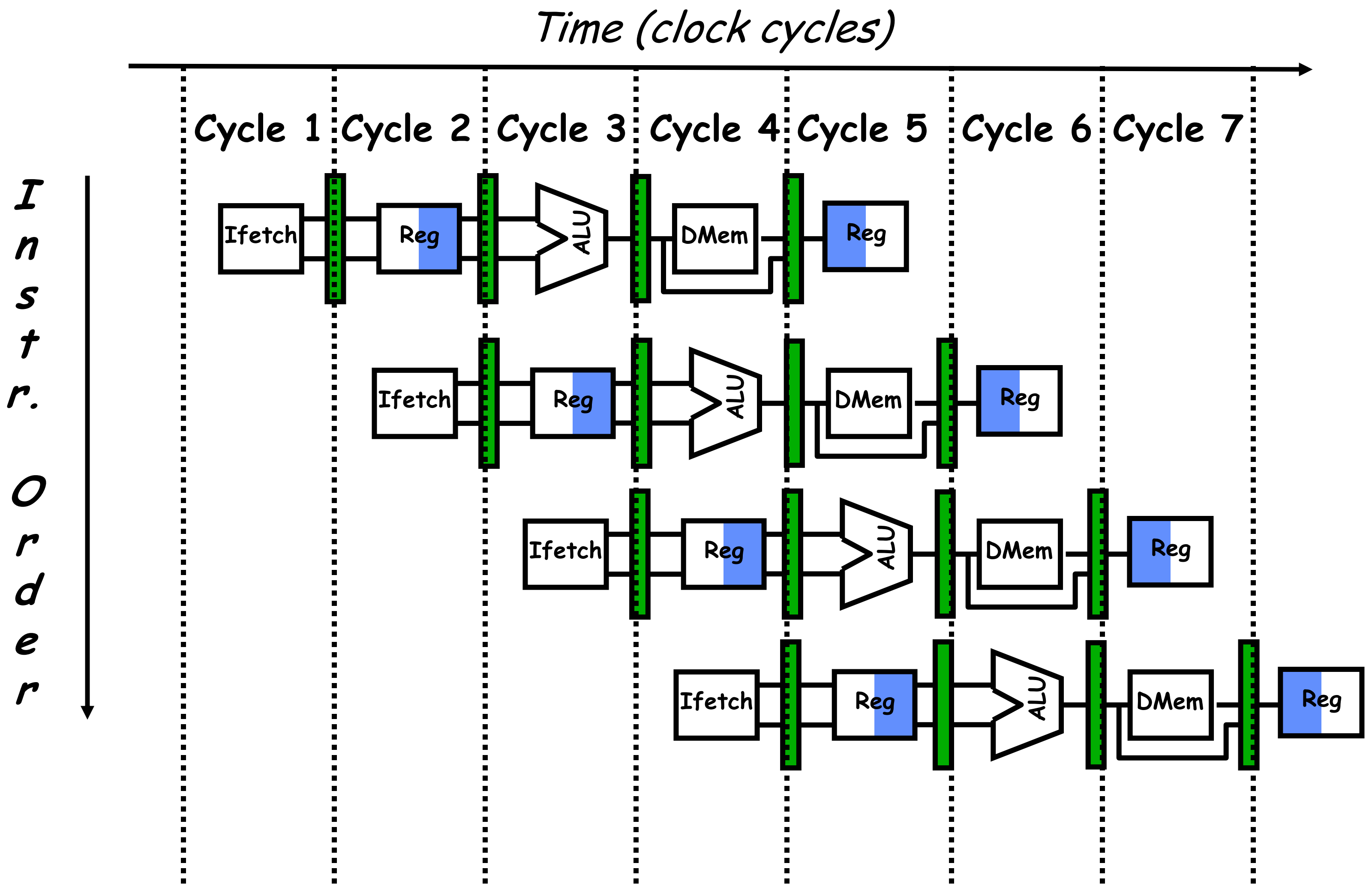
- Computer architecture is not just about transistors, individual instructions, or particular implementations
- Original RISC projects replaced complex instructions with a compiler + simple instructions

How do you turn more stuff into more performance?

- Do more things at once
- Do the things that you do faster

- Beneath the ISA illusion….

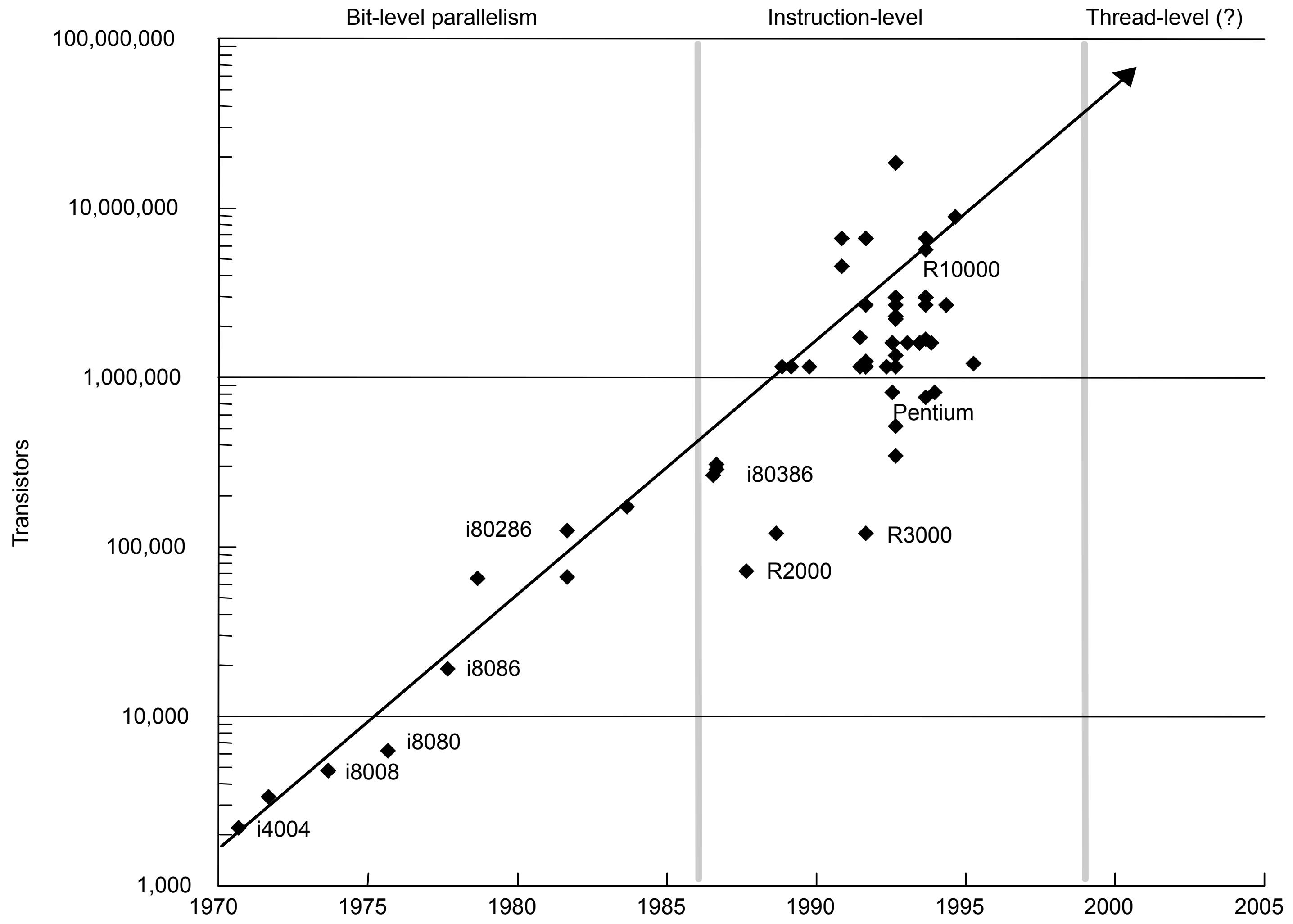
Pipelined Instruction Execution



Limits to pipelining

- Maintain the von Neumann "illusion" of one instruction at a time execution
- Hazards prevent next instruction from executing during its designated clock cycle
  - Structural hazards: attempt to use the same hardware to do two different things at once
  - Data hazards: Instruction depends on result of prior instruction still in the pipeline
  - Control hazards: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps).

A take on Moore's Law



Progression of ILP

- 1st generation RISC - pipelined
  - Full 32-bit processor fit on a chip => issue almost 1 IPC
    - » Need to access memory 1+x times per cycle
  - Floating-Point unit on another chip
  - Cache controller a third, off-chip cache
  - 1 board per processor ➔ multiprocessor systems
- 2nd generation: superscalar
  - Processor and floating point unit on chip (and some cache)
  - Issuing only one instruction per cycle uses at most half
  - Fetch multiple instructions, issue couple
    - » Grows from 2 to 4 to 8 …
  - How to manage dependencies among all these instructions?
  - Where does the parallelism come from?
- VLIW
  - Expose some of the ILP to compiler, allow it to schedule instructions to reduce dependences

Modern ILP

- Dynamically scheduled, out-of-order execution
- Current microprocessor fetch 10s of instructions per cycle
- Pipelines are 10s of cycles deep

=> many 10s of instructions in execution at once

- Grab a bunch of instructionsdetermine all their dependences, eliminate dep's wherever possible, throw them all into the execution unit, let each one move forward as its dependences are resolved
- Appears as if executed sequentially
- On a trap or interrupt, capture the state of the machine between instructions perfectly
- Huge complexity

Have we reached the end of ILP?

- Multiple processor easily fit on a chip
- Every major microprocessor vendor has gone to multithreading
  - Thread: loci of control, execution context
  - Fetch instructions from multiple threads at once, throw them all into the execution unit
  - Intel: hyperthreading, Sun:
  - Concept has existed in high performance computing for 20 years (or is it 40? CDC6600)
- Vector processing
  - Each instruction processes many distinct data
  - Ex: MMX
- Raise the level of architecture – many processors per chip

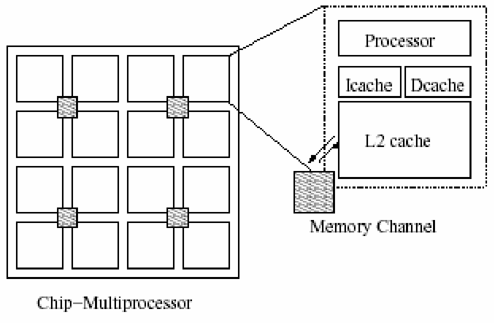
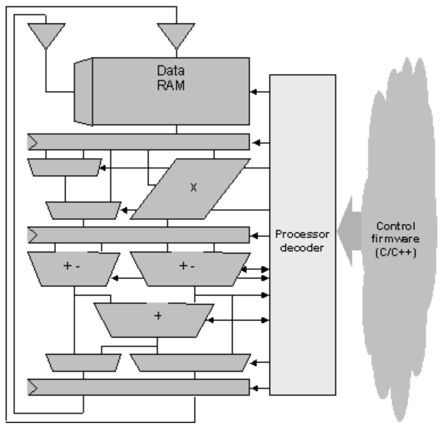


Figure 1. Chip-multiprocessor model.



Tensilica Configurable Proc

When all else fails - guess

- Programs make decisions as they go
  - Conditionals, loops, calls
  - Translate into branches and jumps (1 of 5 instructions)
- How do you determine what instructions for fetch when the ones before it haven't executed?
  - Branch prediction
  - Lot's of clever machine structures to predict future based on history
  - Machinery to back out of mis-predictions
- Execute all the possible branches
  - Likely to hit additional branches, perform stores

$\Rightarrow$speculative threads

$\Rightarrow$What can hardware do to make programming (with performance) easier?

CS252: Adminstrivia

Instructor: Prof David Culler

Office: 627 Soda Hall, culler@cs

Office Hours: Wed 3:30 - 5:00 or by appt.

(Contact Willa Walker)

T. A: TBA

Class: Tu/Th, 11:00 - 12:30pm 310 Soda Hall

Text: Computer Architecture: A Quantitative Approach, Third Edition (2002)

Web page: http://www.cs/~culler/courses/cs252-F03/

Lectures available online <9:00 AM day of lecture

Newsgroup: ucb.class.cs252

Typical Class format (after week 2)

- *Bring questions to class*
- 1-Minute Review
- 20-Minute Lecture
- 5- Minute Administrative Matters
- 25-Minute Lecture/Discussion
- 5-Minute Break (water, stretch)
- *25-Minute Discussion based on your questions*

- I will come to class early & stay after to answer questions
- Office hours

Grading

- **15% Homeworks (work in pairs) and reading writeups**
- **35% Examinations (2 Midterms)**
- **35% Research Project (work in pairs)**
  - Transition from undergrad to grad student
  - Berkeley wants you to succeed, but you need to show initiative
  - pick topic (more on this later)
  - meet 3 times with faculty/TA to see progress
  - give oral presentation or poster session
  - written report like conference paper
  - 3 weeks work full time for 2 people
  - Opportunity to do “research in the small” to help make transition from good student to research colleague
- **15% Class Participation (incl. Q’s)**

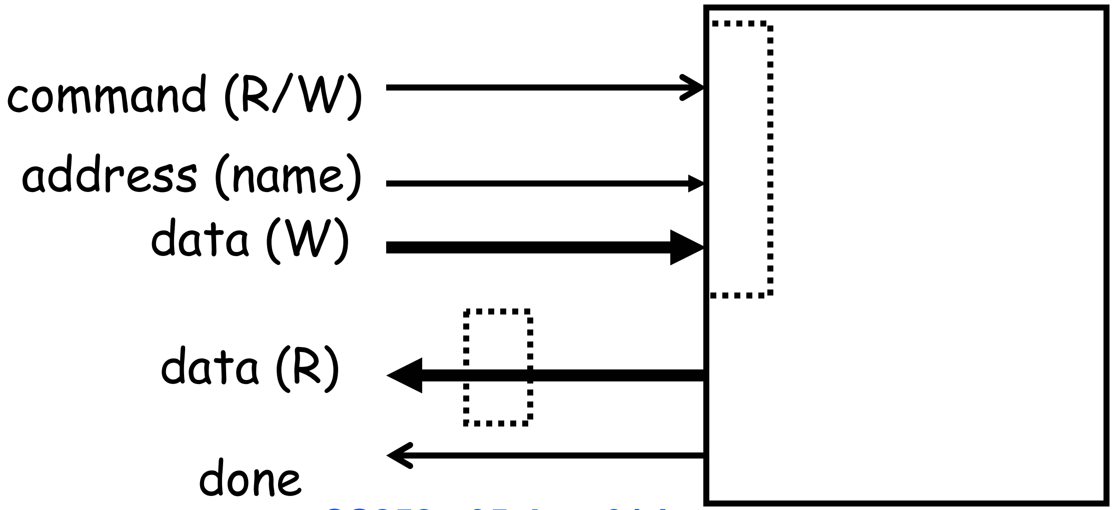
Quizes

- Preparation causes you to systematize your understanding
- Reduce the pressure of taking exam
  - 2 Graded quizes: Tentative: 2/23 and 4/13
  - goal: test knowledge vs. speed writing
    - » 3 hrs to take 1.5-hr test (5:30-8:30 PM, TBA location)
  - Both mid-terms can bring summary sheet
    - » Transfer ideas from book to paper
  - Last chance Q&A: during class time day before exam

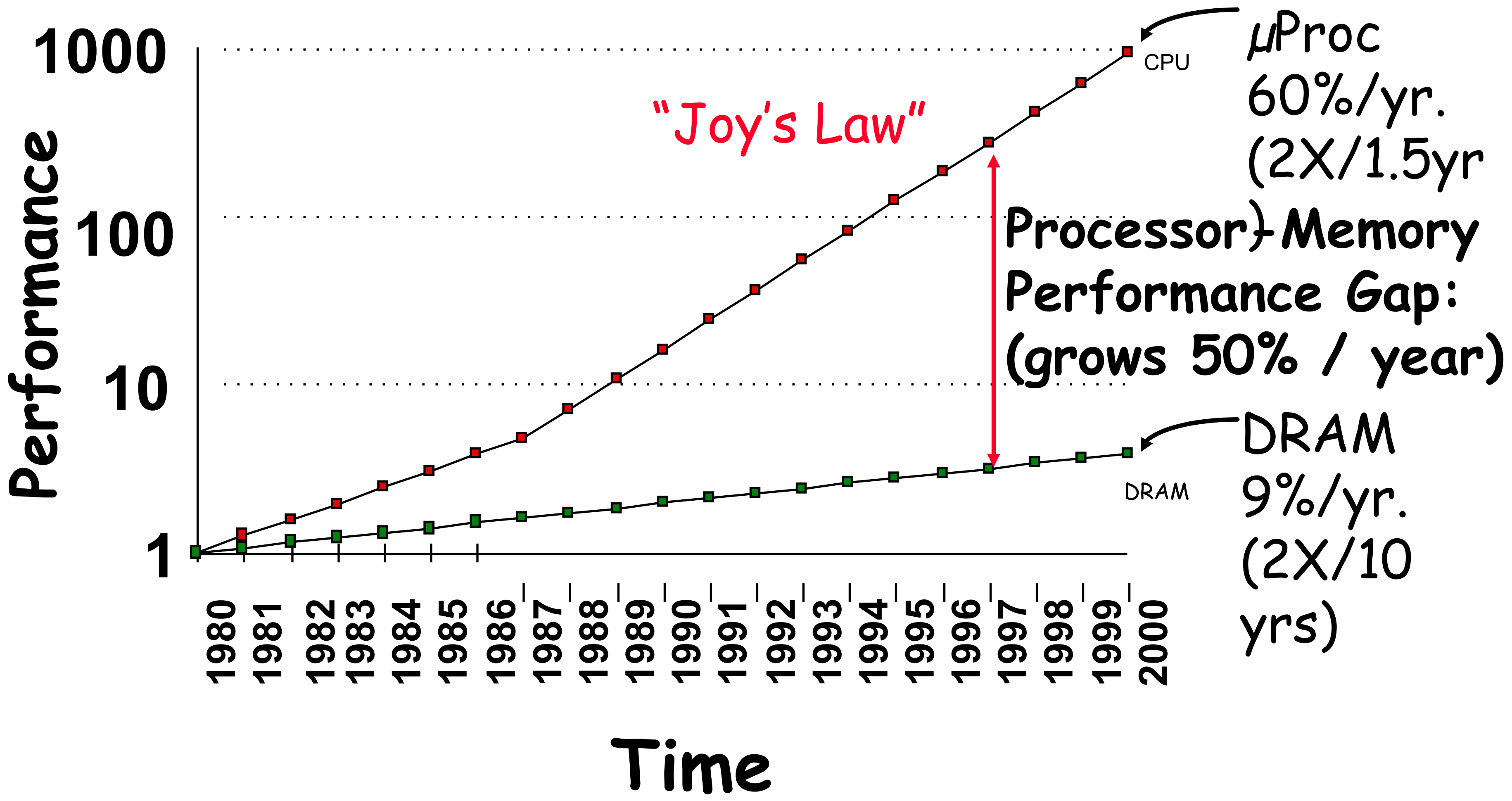
- Students/Staff meet over free pizza/drinks at La Vals: Wed Feb 23 (8:30 PM) and Wed Apr 13 (8:30 PM)

The Memory Abstraction

- Association of <name, value> pairs
  - typically named as byte addresses
  - often values aligned on multiples of size
- Sequence of Reads and Writes
- Write binds a value to an address
- Read of addr returns most recently written value bound to that address



Processor-DRAM Memory Gap (latency)



Levels of the Memory Hierarchy

Capacity
Access Time
Cost

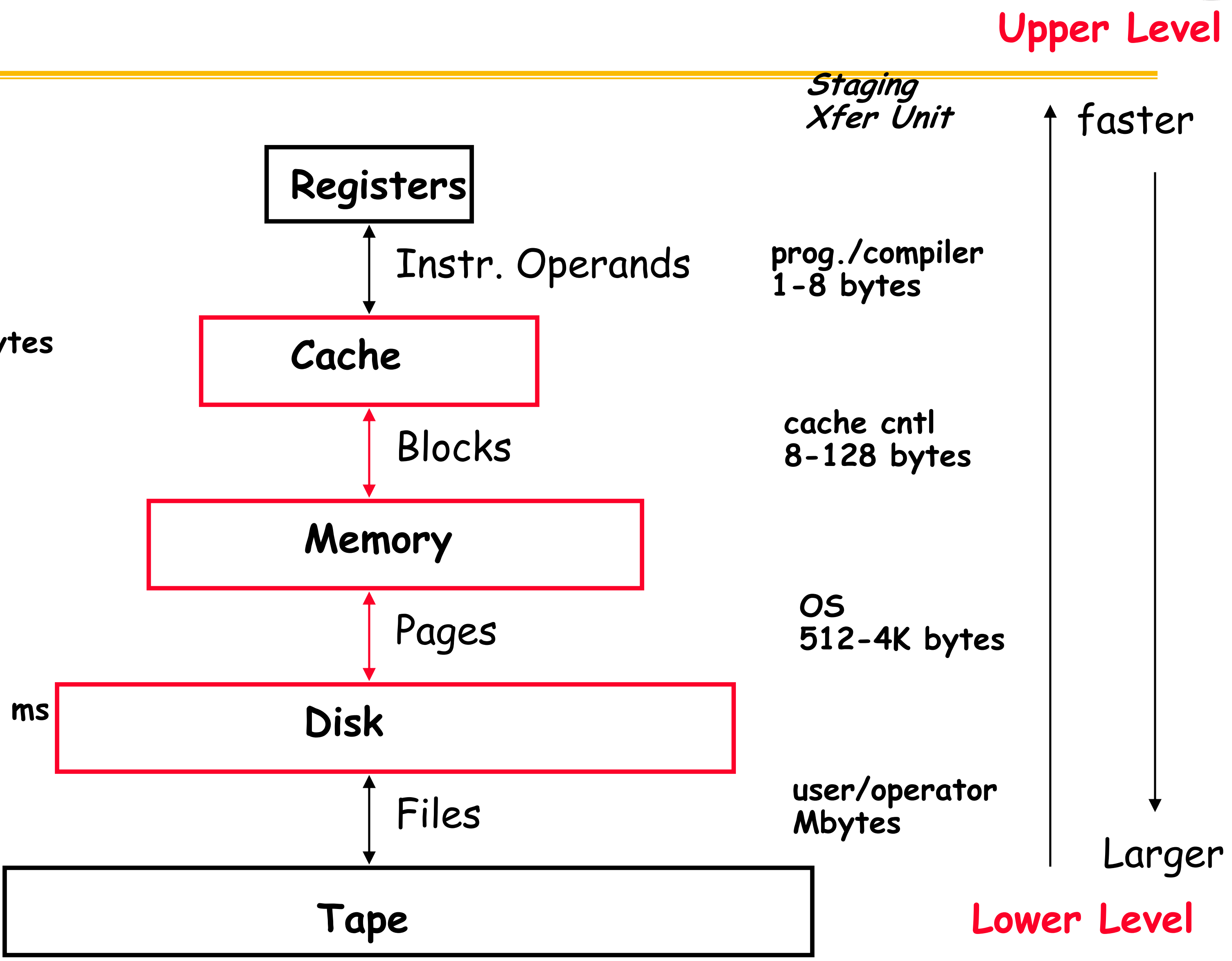
CPU Registers
100s Bytes
<< 1s ns

Cache
10s-100s K Bytes
~1 ns
$1s/ MByte

Main Memory
M Bytes
100ns- 300ns
$< 1/ MByte

Disk
10s G Bytes, 10 ms
(10,000,000 ns)
$0.001/ MByte

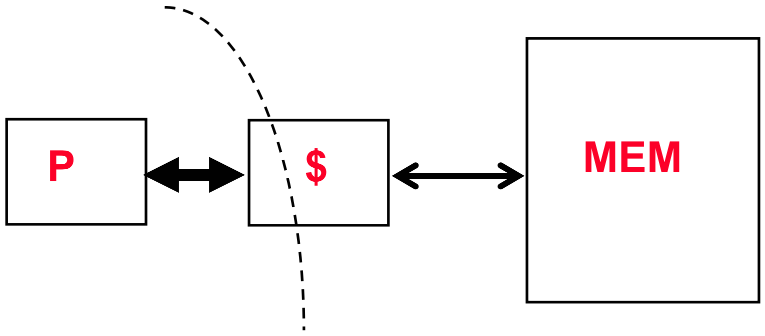
Tape
infinite
sec-min
$0.0014/ MByte



circa 1995 numbers

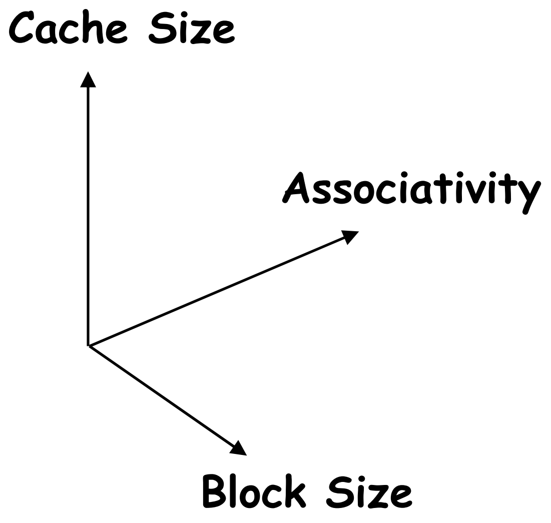
The Principle of Locality

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straightline code, array access)
- Last 30 years, HW relied on locality for speed

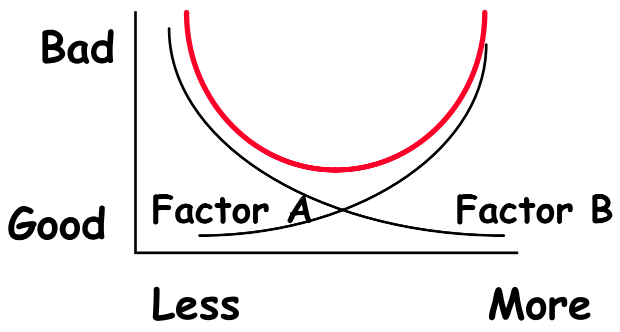


The Cache Design Space

- Several interacting dimensions
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back



- The optimal choice is a compromise
  - depends on access characteristics
    - » workload
    - » use (I-cache, D-cache, TLB)
  - depends on technology / cost
- Simplicity often wins

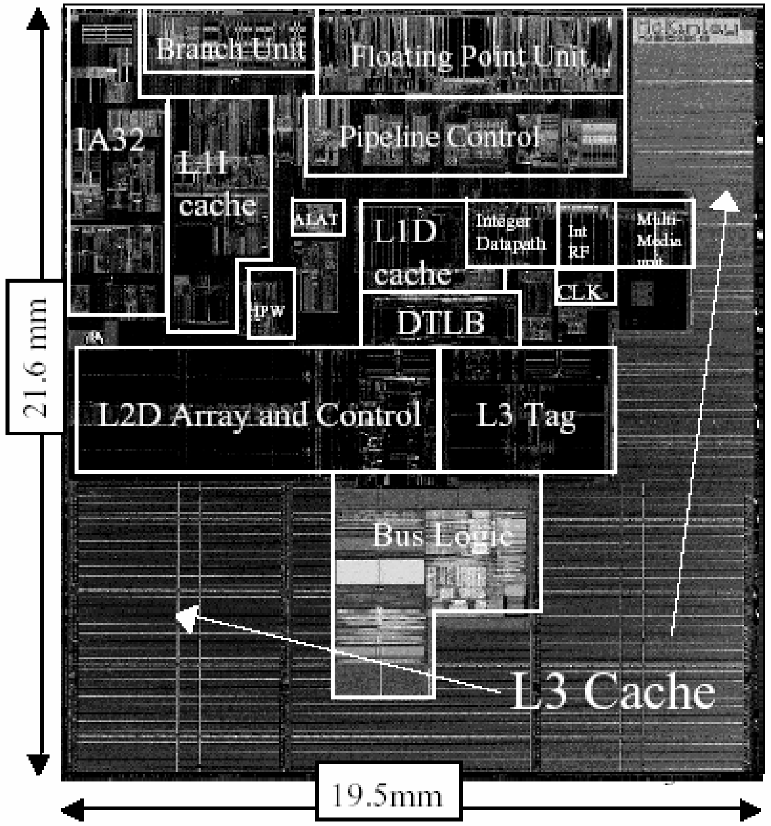


Is it all about memory system design?

- **Modern microprocessors are almost all cache**

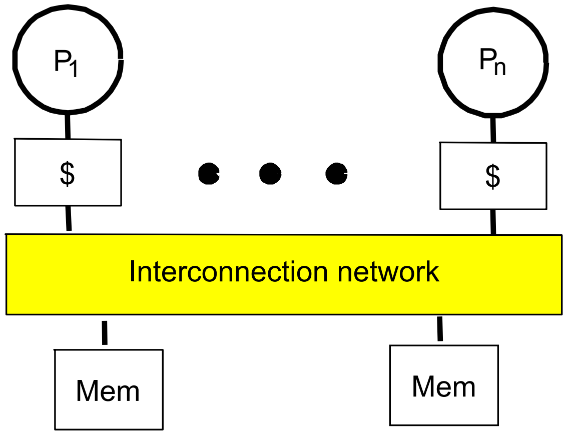
McKinley Floorplan

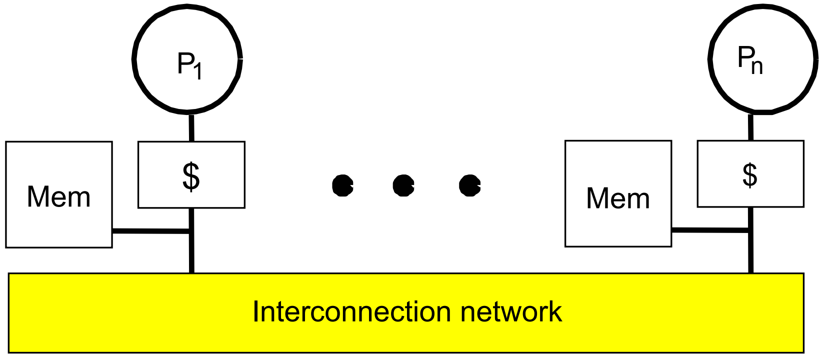
- 0.18 µm, Al process
- 200MHz system clock
- 1GHz core clock
- Core clocking:
  - 260 mm^2
  - 1 primary driver
  - 5 repeaters
  - 33 delay SLCBs
  - 18k gated buffers
  - 157k clocked latches



Memory Abstraction and Parallelism

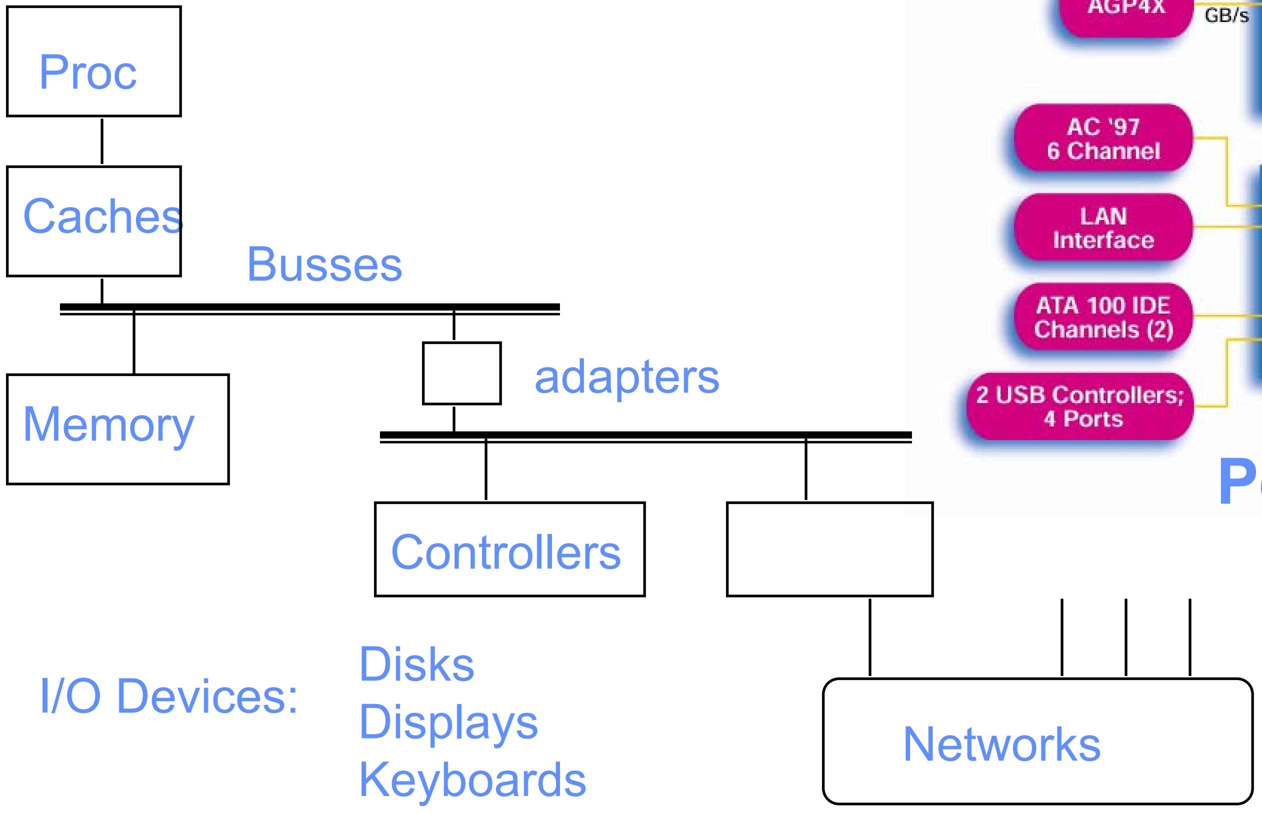
- Maintaining the illusion of sequential access to memory
- What happens when multiple processors access the same memory at once?
  - Do they see a consistent picture?

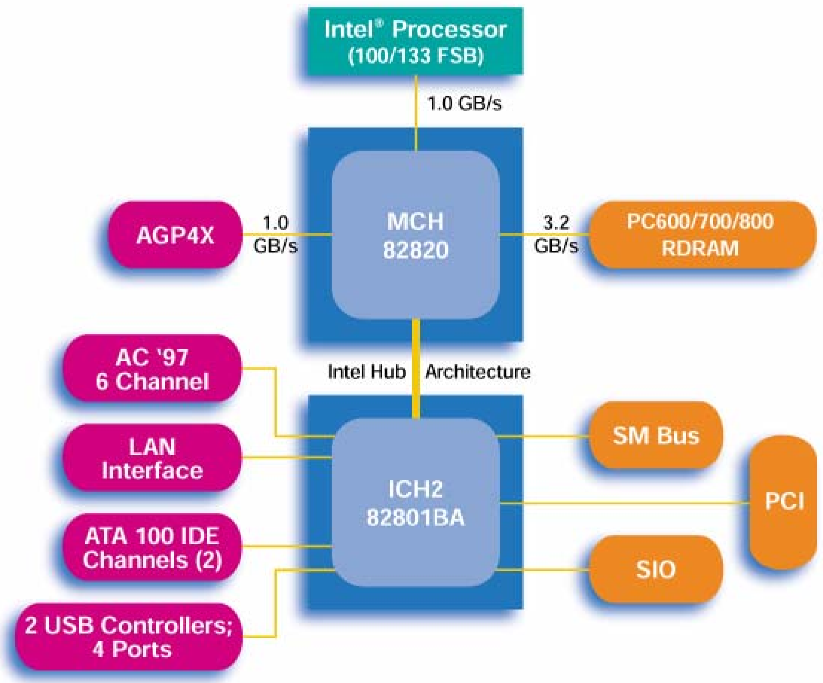




- Processing and processors embedded in the memory?

System Organization: It's all about communication



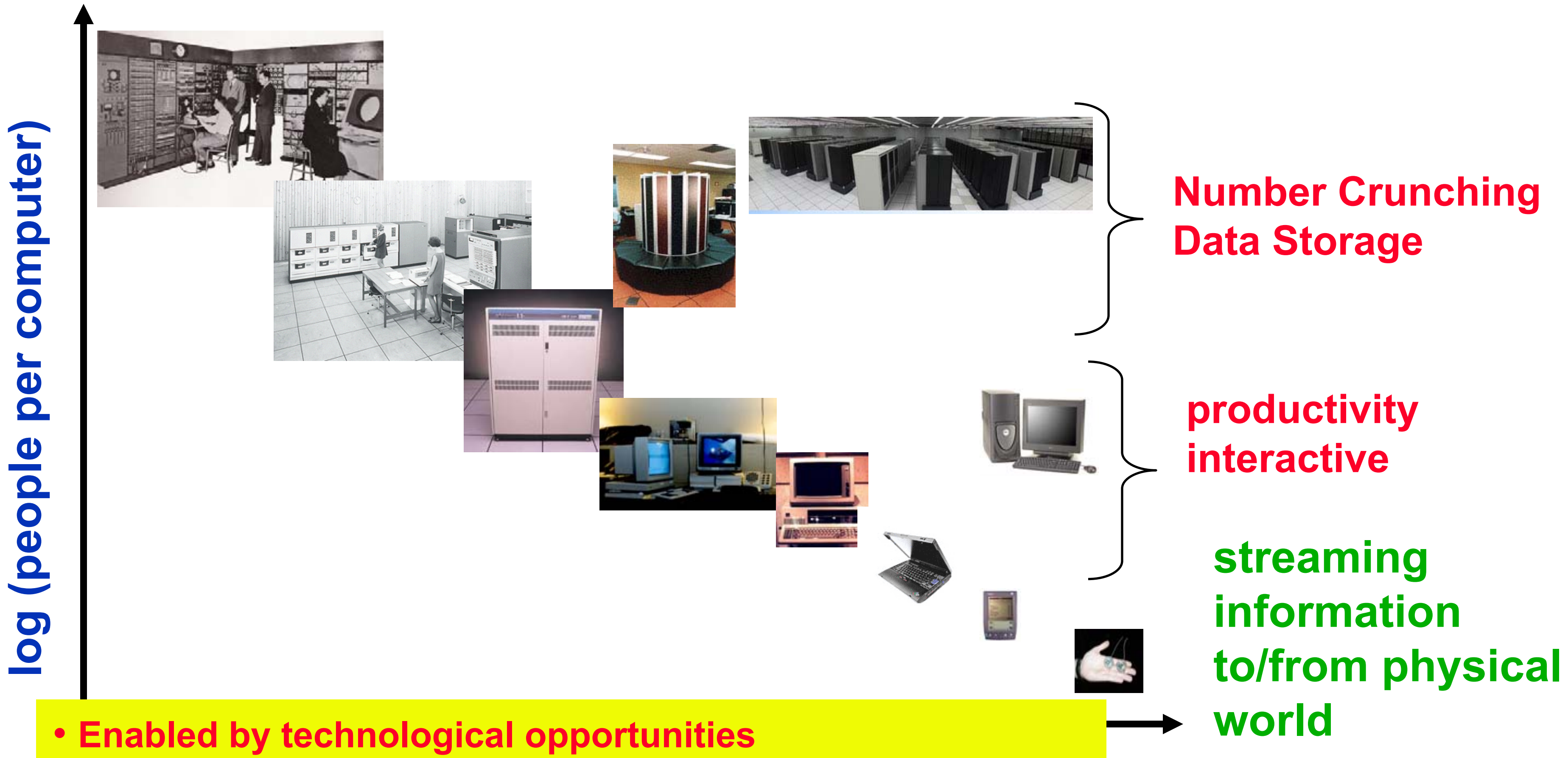


Pentium III Chipset

Breaking the HW/Software Boundary

- Moore's law (more and more trans) is all about volume and regularity
- What if you could pour nano-acres of unspecific digital logic "stuff" onto silicon
  - Do anything with it. Very regular, large volume
- Field Programmable Gate Arrays
  - Chip is covered with logic blocks w/ FFs, RAM blocks, and interconnect
  - All three are "programmable" by setting configuration bits
  - These are huge?
- Can each program have its own instruction set?
- Do we compile the program entirely into hardware?

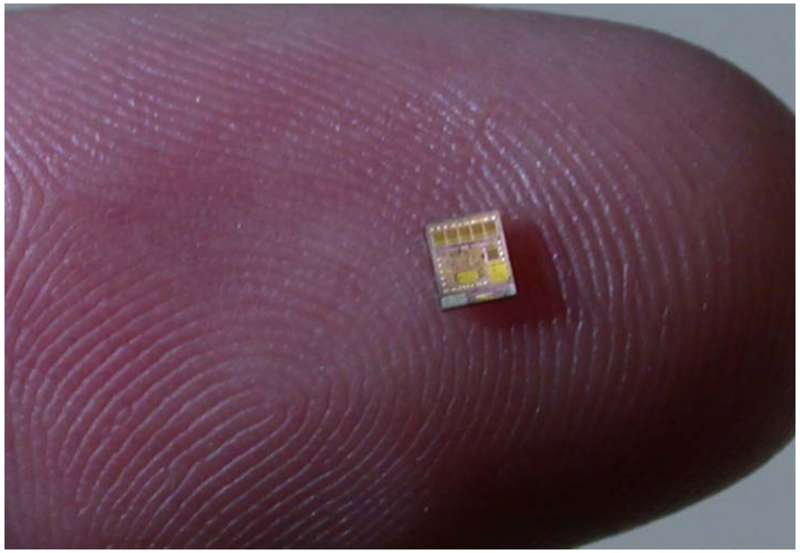
“Bell’s Law” – new class per decade



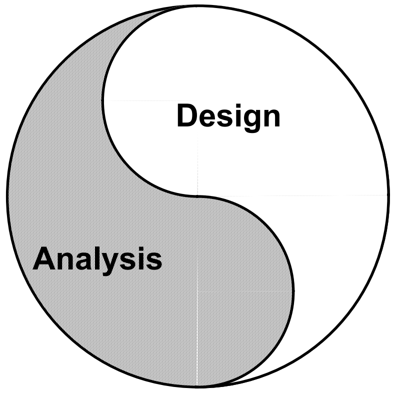
- Enabled by technological opportunities
- Smaller, more numerous and more intimately connected
- Brings in a new kind of application
- Used in many ways not previously imagined

It's not just about bigger and faster!

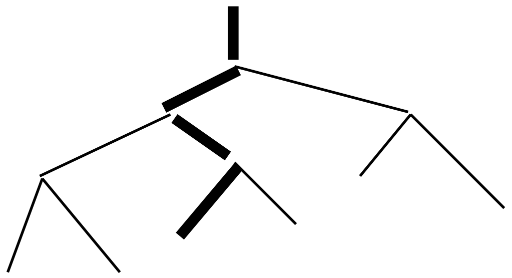
- **Complete computing systems can be tiny and cheap**
- **System on a chip**
- **Resource efficiency**
  - **Real-estate, power, pins, …**

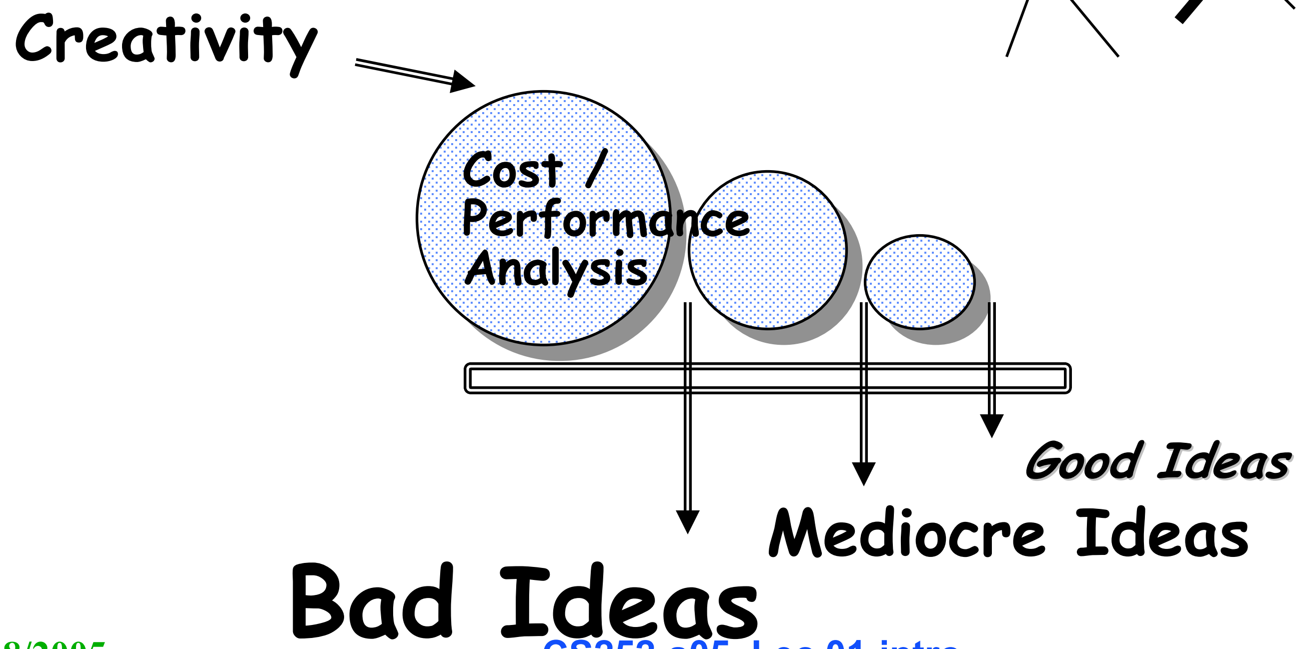


The Process of Design



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





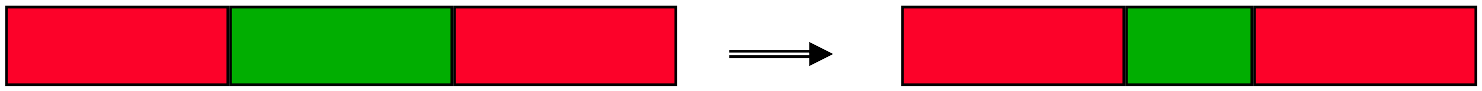
Amdahl's Law

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

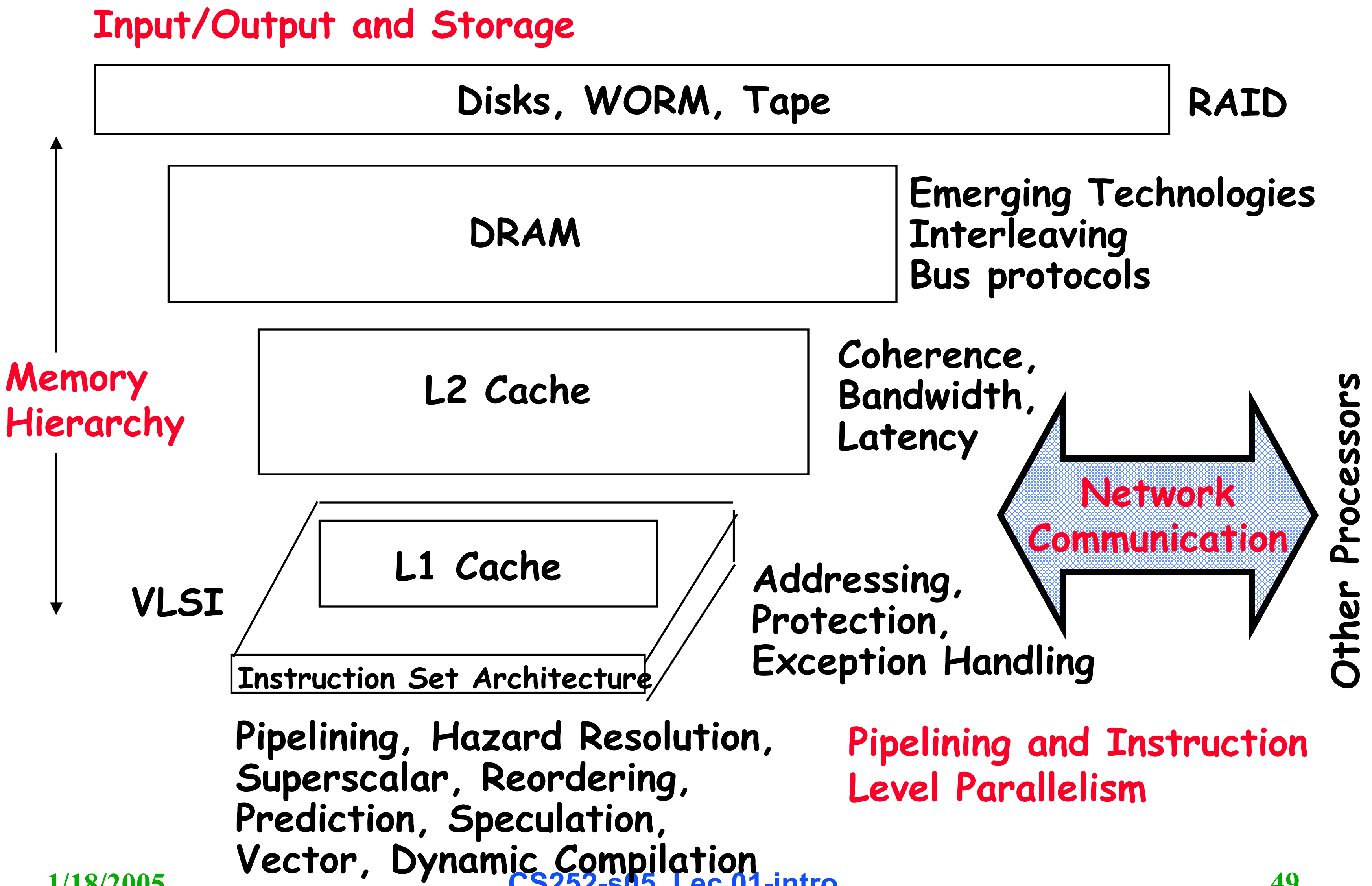
$$\text{Speedup}_{\text{overall}} = \frac{\text{ExTime}_{\text{old}}}{\text{ExTime}_{\text{new}}} = \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Best you could ever hope to do:

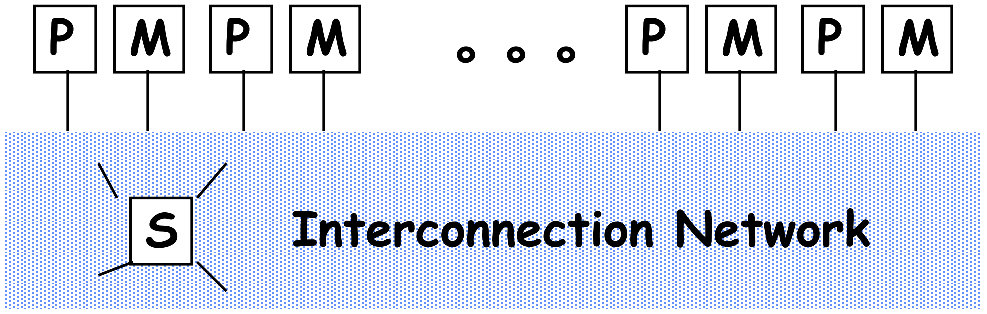
$$\text{Speedup}_{\text{maximum}} = \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}}\right)}$$



Computer Architecture Topics



Computer Architecture Topics



Processor-Memory-Switch

Multiprocessors
Networks and Interconnections

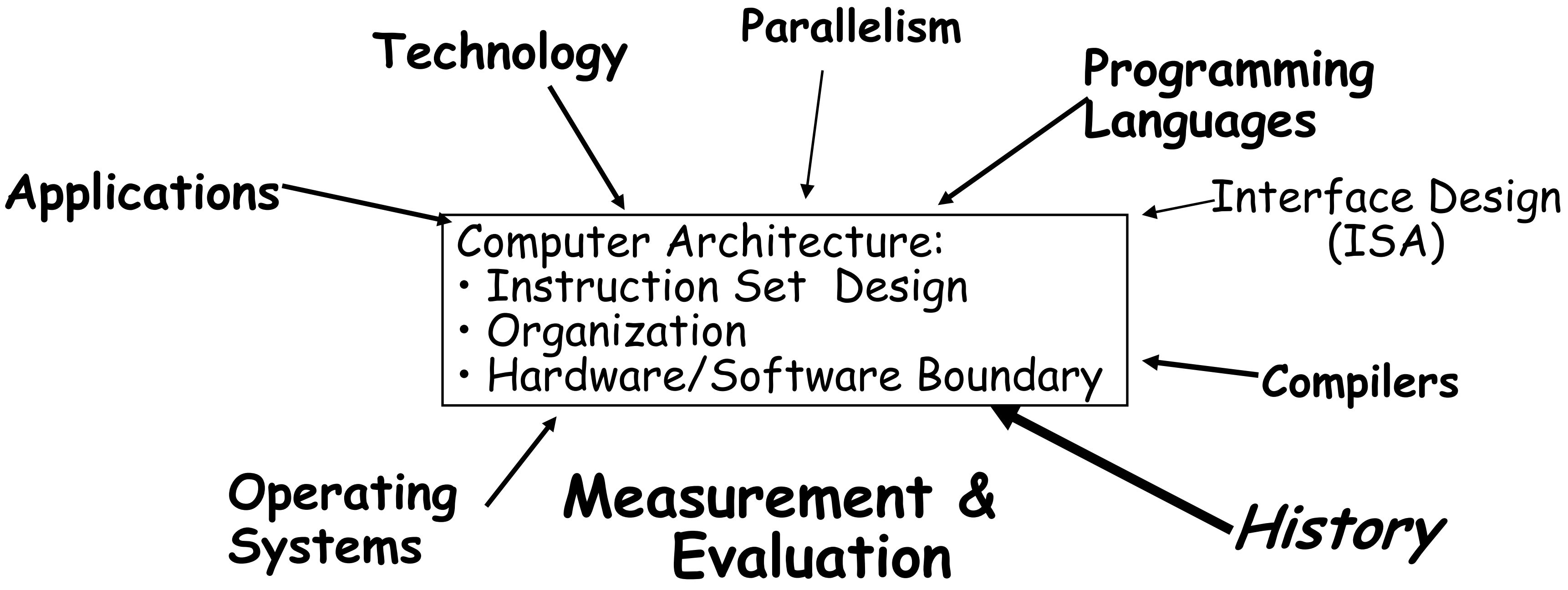
Shared Memory, Message Passing, Data Parallelism

Network Interfaces

Topologies, Routing, Bandwidth, Latency, Reliability

CS 252 Course Focus

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



Topic Coverage

Textbook: Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 3rd Ed., 2002.

Research Papers – on-line

- 1.5 weeks Review: Fundamentals of Computer Architecture (Ch. 1), Instruction Set Architecture (Ch. 2), Pipelining (App A), Caches
- 2.5 weeks: Pipelining, Interrupts, and Instructional Level Parallelism (Ch. 3, 4), Vector Proc. (Appendix G)
- 1 week: Memory Hierarchy (Chapter 5)
- 2 weeks: Multiprocessors,Memory Models, Multithreading,
- 1.5 weeks: Networks and Interconnection Technology (Ch. 7)
- 1 weeks: Input/Output and Storage (Ch. 6)
- 1.5 weeks: Embedded processors, network proc, low-power
- 3 week: Advanced topics

Your CS252

- Computer architecture is at a crossroads
  - Institutionalization and renaissance
  - Ease of use, reliability, new domains vs. performance
- Mix of lecture vs discussion
  - Depends on how well reading is done before class
- Goal is to learn how to do good systems research
  - Learn a lot from looking at good work in the past
  - New project model: reproduce old study in current context
    - » Will ask you do survey and select a couple
    - » Looking in detail at older study will surely generate new ideas too
  - At commit point, you may chose to pursue your own new idea instead.

Research Paper Reading

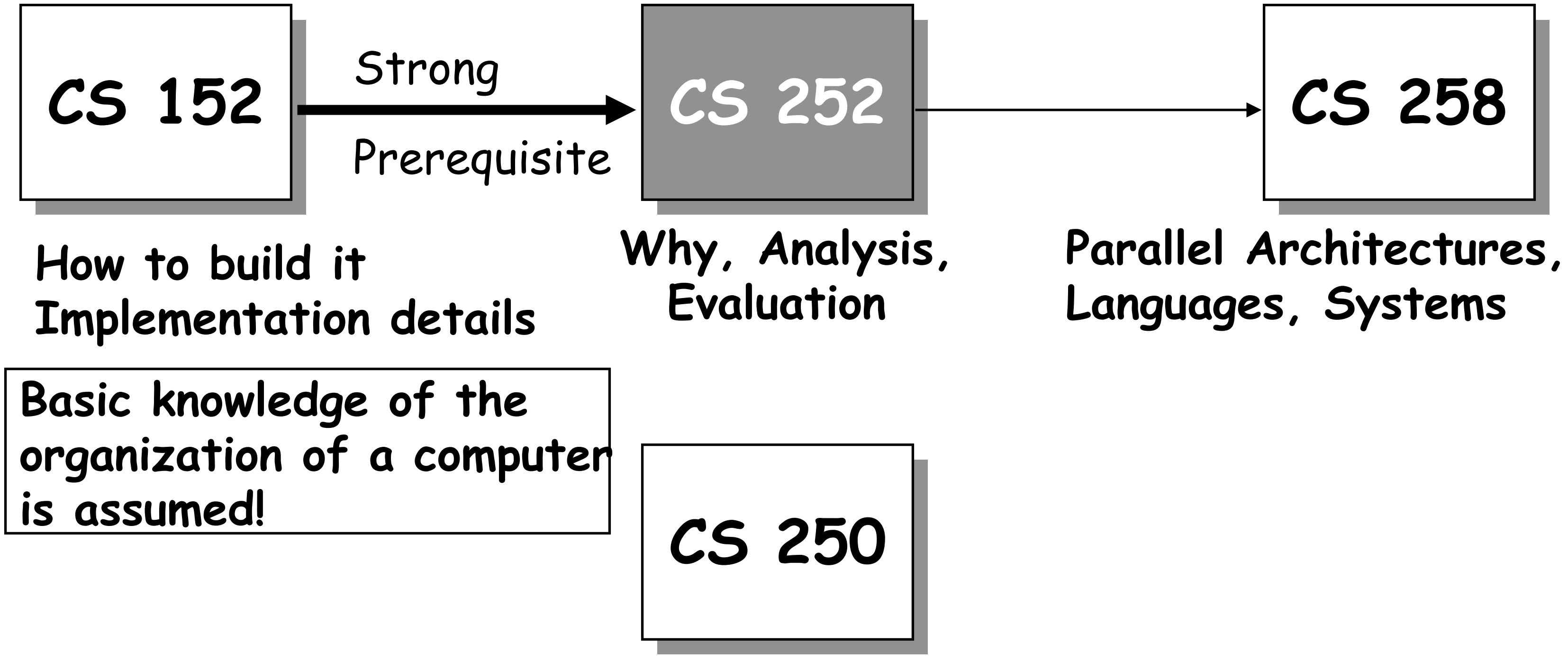
- As graduate students, you are now researchers.
- Most information of importance to you will be in research papers.
- Ability to rapidly scan and understand research papers is key to your success.

- So: you will read lots of papers in this course!
  - Quick 1 paragraph summaries and question will be due in class
  - Important supplement to book.
  - Will discuss papers in class
- Papers will be scanned and on web page.

Coping with CS 252

- Students with too varied background?
  - In past, CS grad students took written prelim exams on undergraduate material in hardware, software, and theory
  - 1st 5 weeks reviewed background, helped 252, 262, 270
  - Prelims were dropped => some unprepared for CS 252?
- Review: Chapters 1-3, CS 152 home page, maybe "Computer Organization and Design (COD)2/e"
  - Chapters 1 to 8 of COD if never took prerequisite
  - If took a class, be sure COD Chapters 2, 6, 7 are familiar
  - Copies in Bechtel Library on 2-hour reserve
- Not planning to do prelim exams
  - Undergrads must have 152
  - Grads without 152 equivalent will have to work hard
    - » Will schedule Friday remedial discussion section

Related Courses



How to build it
Implementation details

Why, Analysis, Evaluation

Parallel Architectures, Languages, Systems

Basic knowledge of the organization of a computer is assumed!

Integrated Circuit Technology
from a computer-organization viewpoint