CMPE 411
Computer Architecture

Lecture 1

Introduction and Overview

August 31, 2017

www.csee.umbc.edu/~younis/CMPE411/CMPE411.htm

Lecture's Overview

- Course resources, syllabus and work load
- Grade structure and policy
- Teaching style and philosophy
- An introduction to computer architecture
- The importance of studying computer architecture
- Organization and anatomy of computers
- The impact of microelectronics technology on computers
- The evolution of the computer industry and generations

Course Resources

Instructor: **Dr. Mohamed Younis**

Office: ITE 318 E-mail: younis@cs.umbc.edu

Office hours: Tuesday and Thursday 10:30 - 11:30 AM

Research interest: (Research Lab: esnet.cs.umbc.edu)

Wireless Ad-hoc and Sensor Networks, Secure communication, Location privacy, Vehicular Networks, Real-time systems, Fault tolerant computing.

TA: **Mr. Chayutra (Charlie) Pailom**

Office: ITE 349 E-mail: cpailom1@umbc.edu

Office hours: Monday and Wednesday 3:00 - 4:00 PM

Textbook:

Computer Organization and Design, The hardware/software interface, *5th Edition*

David A. Patterson and John L. Hennessy

Morgan Kaufmann Publishers, ISBN 978-0-12-374493-7

Web page: **www.csee.umbc.edu/~younis/CMPE411/CMPE411.htm**

Instructor will stay after class to answer questions

Scope

Course Goals

- To learn the organizational paradigms that determine the capabilities and performance of computer systems
- To understand the interactions between the computer's architecture and its software so that
  - future software designers (compiler writers, operating system designers, database programmers, …) can achieve the best cost-performance trade-offs
  - future architects understand the effects of their design choices on software applications

Course Prerequisites

- CMPE 310: Systems Design and Programming

* Slide is (partially) courtesy of Mary Jane Irwin

Course Syllabus

1. Instruction Set Architecture
- Instruction formats & semantics
- Addressing modes

2. Performance Evaluation
- Measures of performance
- Benchmarks and metrics

3. Machine Arithmetic
- ALU design
- Integer multiplication & division
- Floating-point arithmetic

4. Processor Design
- Datapath design
- Instruction exec. & sequencing
- Hardwired & microcode control

5. Performance boosting features
- Pipelining
- Instruction level parallelism

6. Memory Hierarchy
- Cache design & evaluation
- Virtual addressing
- Performance evaluation

7. Input/Output
- Types of I/O devices
- Device access and interface
- Device control
- I/O performance

8. Multiprocessor (time permitting)
- Interconnection networks
- Programming issues

Course Workload

❑ *Assignments*

- 5 assignments will be given and normalized to %20 of the final grade
- An average assignments requires about 2-3 hours to perform
- Assignments are due in class on the due date (not later)

❑ *Exams*

- A midterm exam is scheduled on October 31th during scheduled class time
- The final exam is scheduled on December 14th during UMBC specified hours
- Final Exam is comprehensive covering all what is covered in class

❑ *Project*

- A design project will be given; contributing 25% to the final grades
- The project involves architecture simulation and performance analysis
- The project is to be implemented using the programming language of your choice, given that it your submit a makefile that automate the compilation of the source files
- The project must be finished and submitted on time to earn a grade

Grade structure and policy

	Grade distribution
Final Exam	30%
Mid-term Exam	25%
Project	25%
Homework	20%

Course grade	Range
A	90% - 100%
B	80% - 89.9%
C	70% - 79.9%
D	60% - 69.9%

- Assignments are due in class (*Late assignments are not accepted*)
- UMBC rules apply to cheating/copying
    - You may discuss the homework and the project
    - You must do your own work and not copy from anyone else
    - You better off skipping an assignment or get a partial credit
- Copying/cheating will result in a minimum punishment of a zero grade for the assignment or project

Teaching Style and Philosophy

❑ *Instructor's role*

- Facilitate and guide the students to the fundamental concepts
- Make it simple and elaborate with examples
- Relate as much as possible to available products
- Prepare class notes to be as rich and comprehensive as possible

❑ *Student's role*

- Focus on understanding and digesting the concept
- Do not worry about the grade more than concepts, soon will be a professional
- Slow down the instructor if you do not understand and raise questions
- Be prepared to answer an oral quiz, when you get involved in a side talk

❑ *TA's role*

- Help students with questions related to their assignments
- Resolve computer and tool issues related to the project
- Grade assignments and projects

Exams will question the level of understanding of fundamental concepts

Introduction & Motivation

- Computer systems are responsible of 5-10% of the gross national product of the US
- Has the transportation industry kept pace with the computer industry, a coast to coast trip would take 5 seconds and cost 50 cents
- WWW, ATM, DNA mapping, … are among the applications that were economically infeasible and became practical
- Cashless society, anywhere computing, automated intelligent highways, mobile health care… are next computer science fiction on their way to become a reality
- Computer architecture has been at the core of such technological development and is still on a forward move

What is "Computer Architecture"?

- Instruction set architecture deals with the functional behavior of a computer system as viewed by a programmer (like the size of a data type – 32 bits to an integer).
- Computer organization deals with structural relationships that are not visible to the programmer (like clock frequency or the size of the physical memory).
- The Von Neumann model is the most famous computer organization

Computer Architecture

Instruction Set Architecture

- Interfaces
- Compiler/System View
- "Building Architect"

Machine Organization

- Hardware Components
- Logic Designer's View
- "Construction Engineer"

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

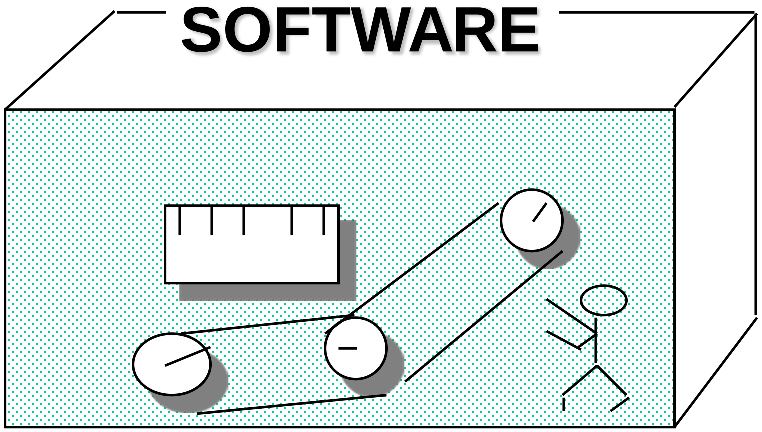
-- Data Types & Data Structures:
Encoding & Representation

-- Instruction Set

-- Instruction Formats

-- Modes of Addressing and Accessing
Data Items and Instructions

-- Exceptional Conditions



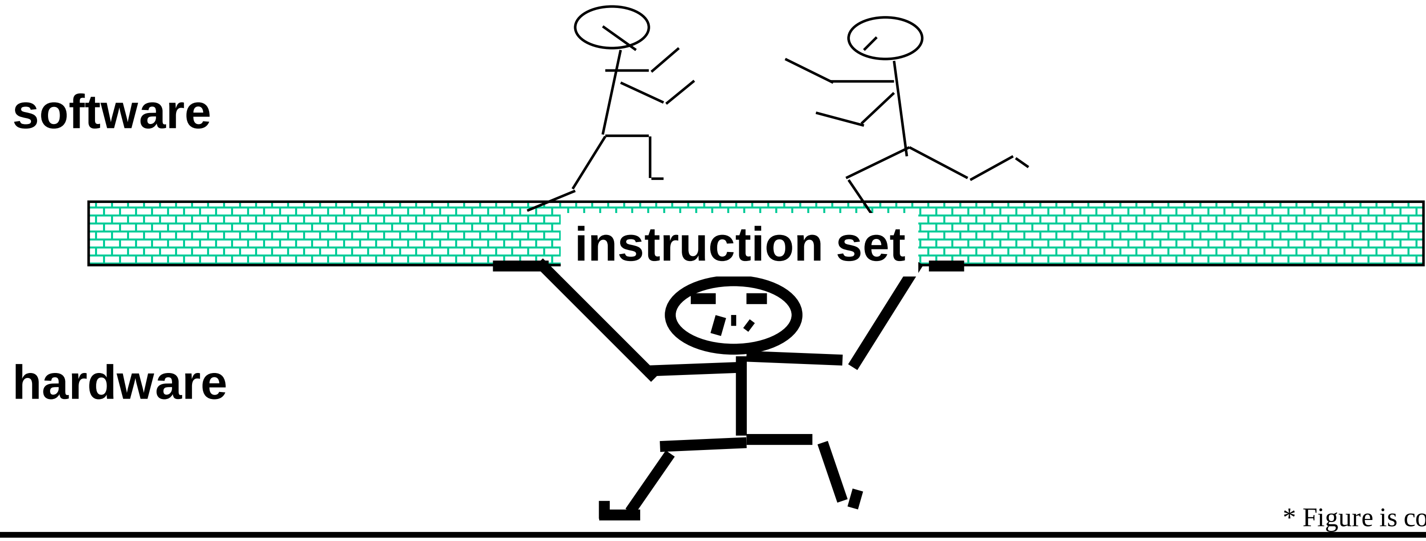
The instruction set architecture distinguishes the semantics of the architecture from its detailed hardware implementation

The Instruction Set: a Critical Interface

Examples:

• DEC Alpha	(v1, v3)	1992-1997
• HP PA-RISC	(v1.1, v2.0)	1986-1996
• Sun Sparc	(v8, v9)	1987-1995
• SGI MIPS	(MIPS I, II, III, IV, V)	1986-1996
• Intel	(8086,80286,80386, 80486,Pentium, MMX, ...)	1978-2000

The instruction set can be viewed as an abstraction of the H/W that hides the details and the complexity of the H/W



* Figure is courtesy of Dave Patterson

MIPS R3000 Instr. Set Arch. (Summary)

Instruction Categories

- Load/Store
- Computational
- Jump and Branch
- Floating Point
  - coprocessor
- Memory Management
- Special

Registers

Registers
R0 - R31
PC
HI
LO

3 Instruction Formats: all 32 bits wide

OP	rs	rt	rd	sa	funct
OP	rs	rt	immediate		
OP	jump target				

* Slide is courtesy of Dave Patterson

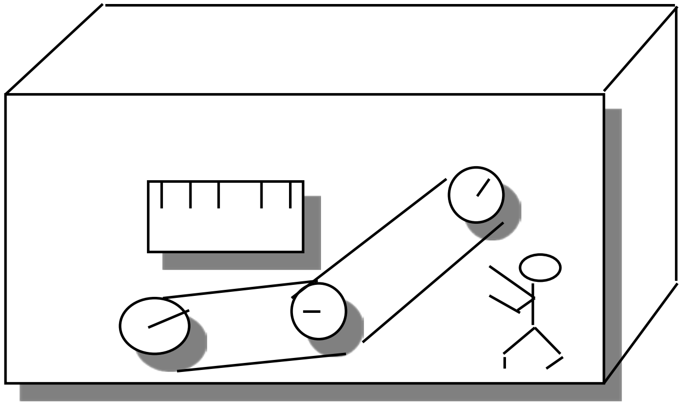
Machine Organization

- 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 functional units to realize the instruction set architecture
- Register Transfer Level Description

Logic Designer's View

ISA Level

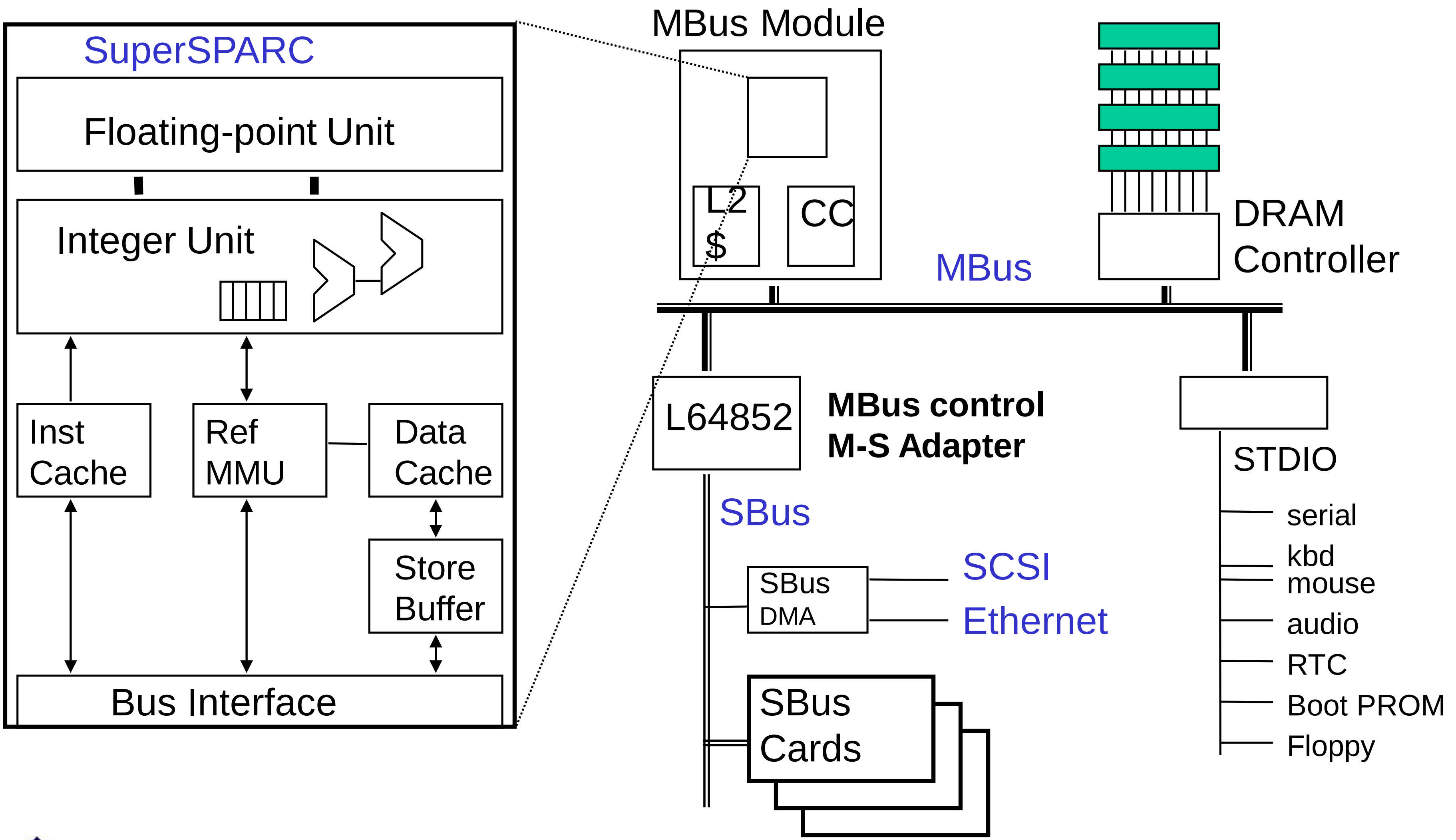
Functional Units & Interconnect



* Slide is courtesy of Dave Patterson

Example Organization

- TI SuperSPARC™ TMS390Z50 in Sun SPARCstation20



* Slide is courtesy of Dave Patterson

Levels of Behavior Representation

High Level Language Program

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

Compiler

Assembly Language Program

```
lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 4($2)
```

Assembler

Machine Language Program

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Machine Interpretation

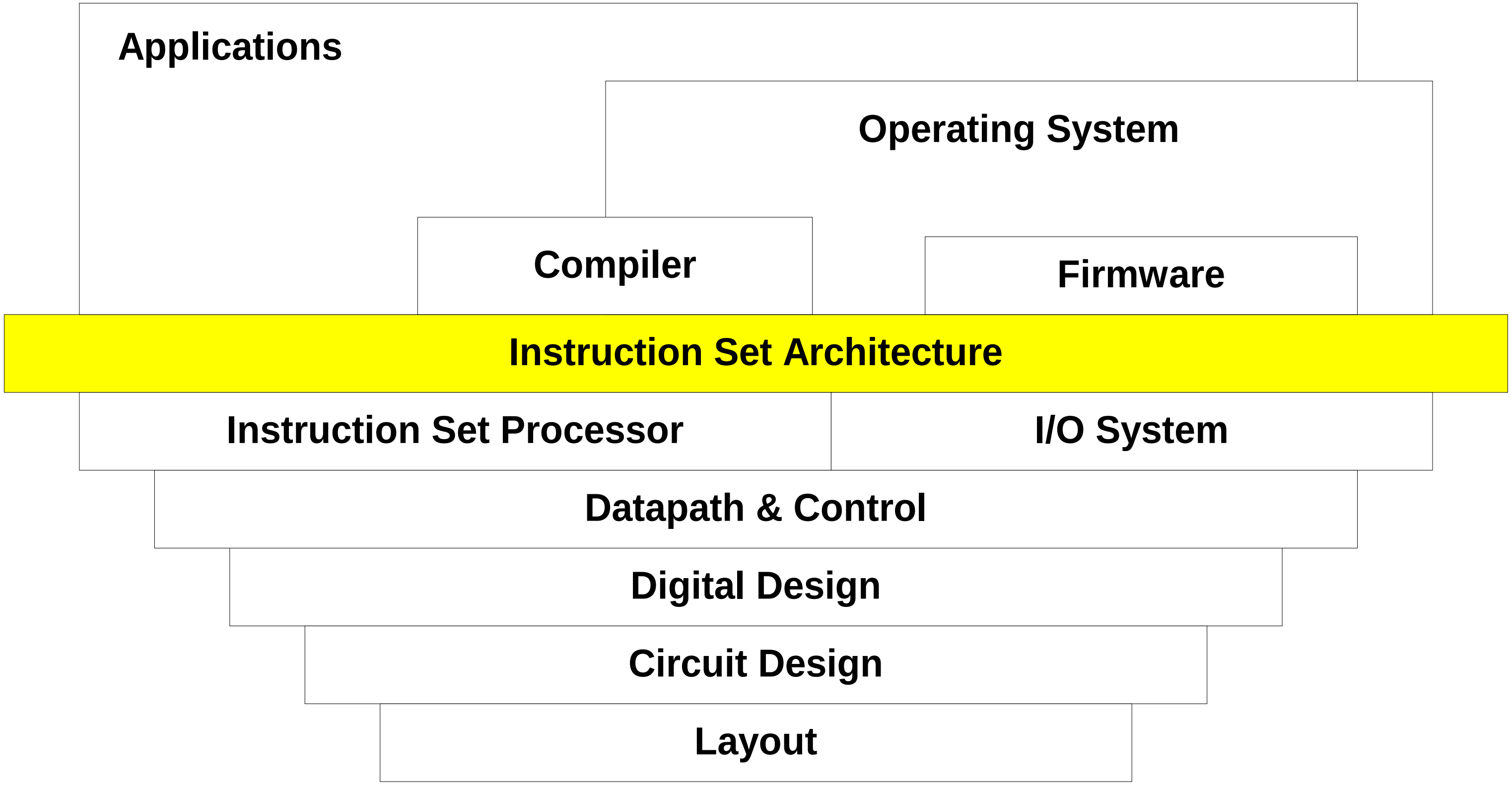
Control Signal Specification

ALUOP[0:3] <= InstReg[9:11] & MASK

°
°

* Slide is courtesy of Dave Patterson

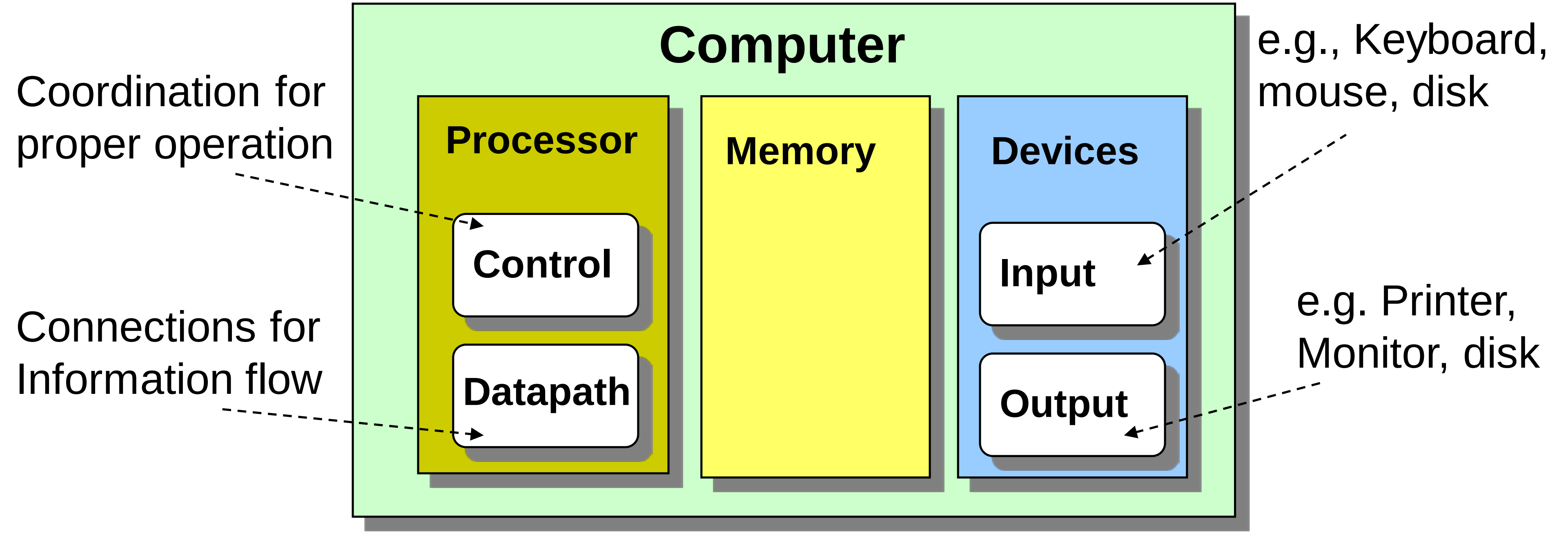
Levels of Abstraction



- S/W and H/W consists of hierarchical layers of abstraction, each hides details of lower layers from the above layer
- The instruction set arch. abstracts the H/W and S/W interface and allows many implementation of varying cost and performance to run the same S/W

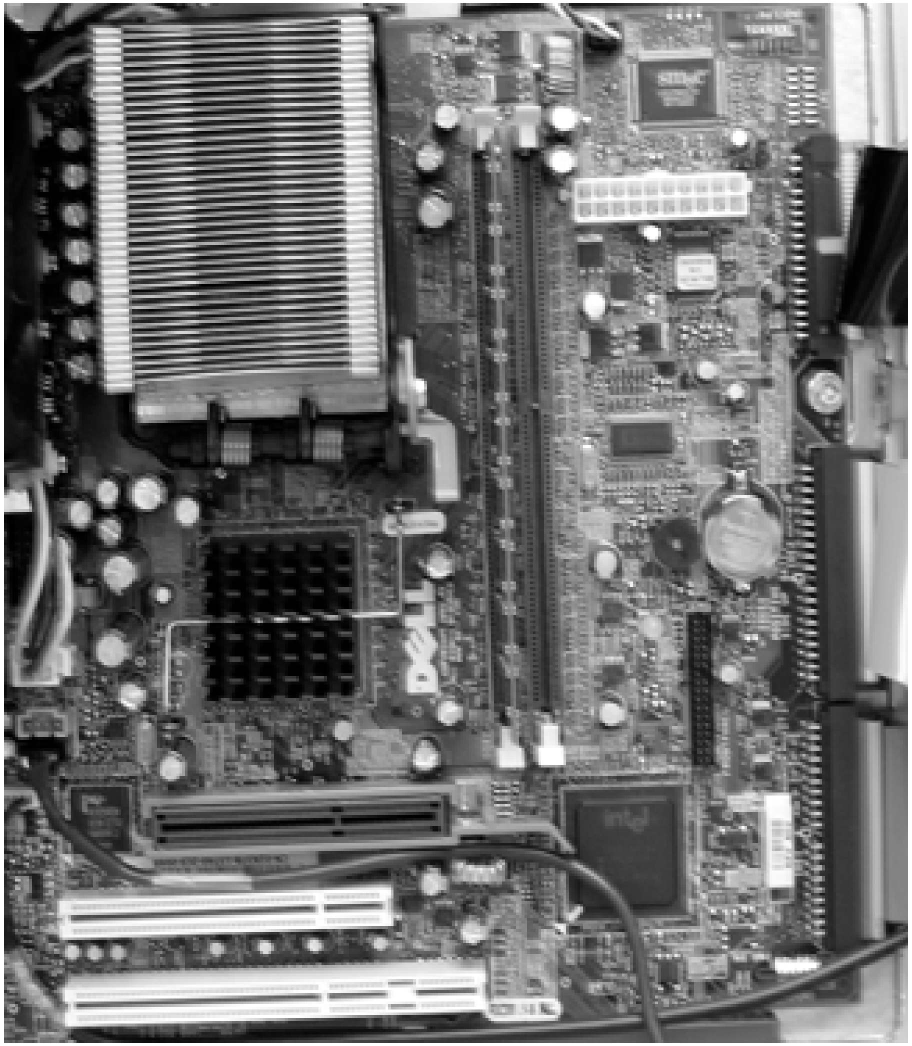
* Figure is courtesy of Dave Patterson

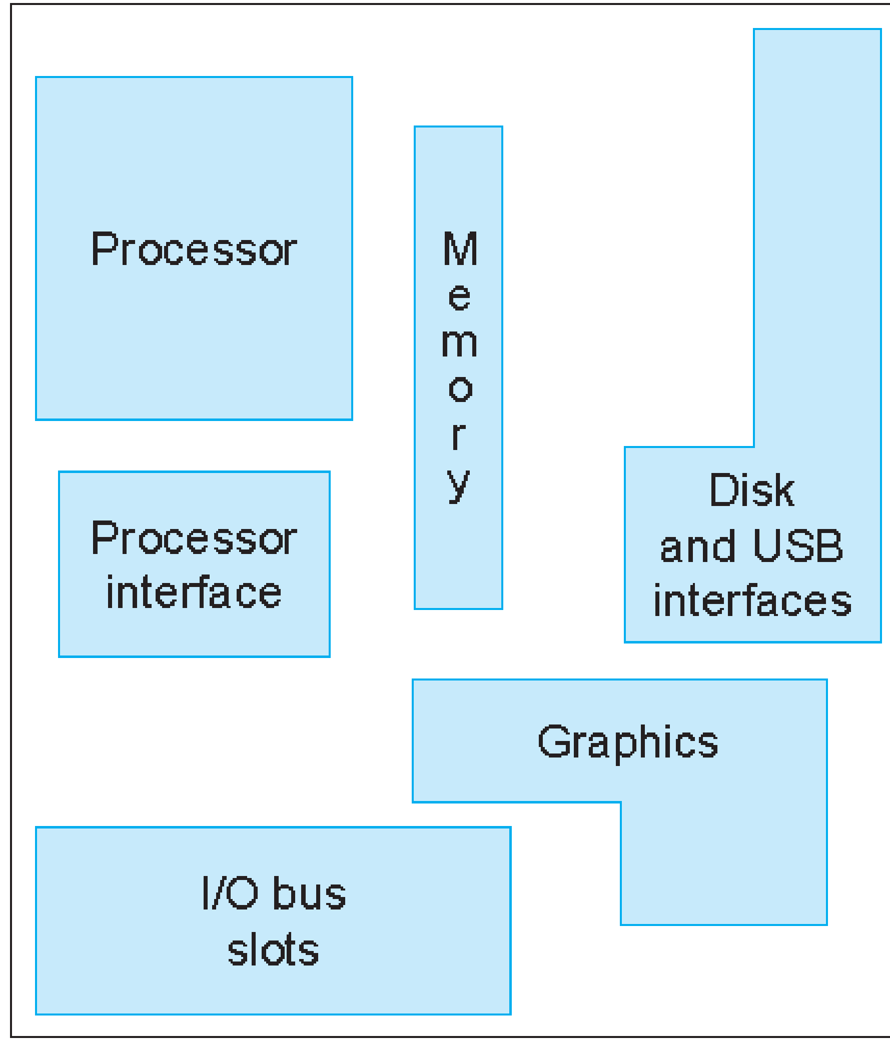
General Computer Organization



- Every piece of every computer, past and present, can be placed into input, output, memory, datapath and control
- The design approach is constrained by the cost and size and capabilities required from every component
- An example design target can be 25% of the cost for Processor, 25% of the cost for minimum memory size, leaving the remaining budget for I/O devices, power supplies, and chassis

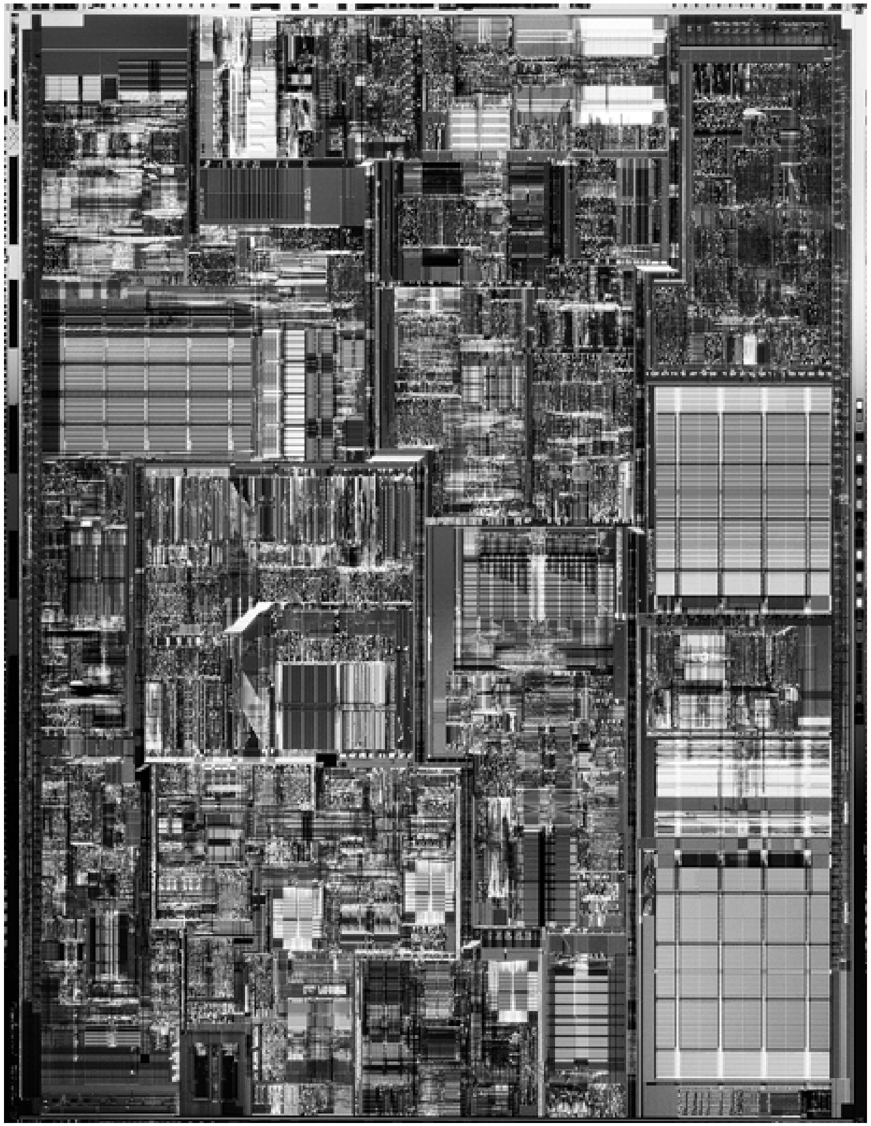
PC Motherboard: A Close Look

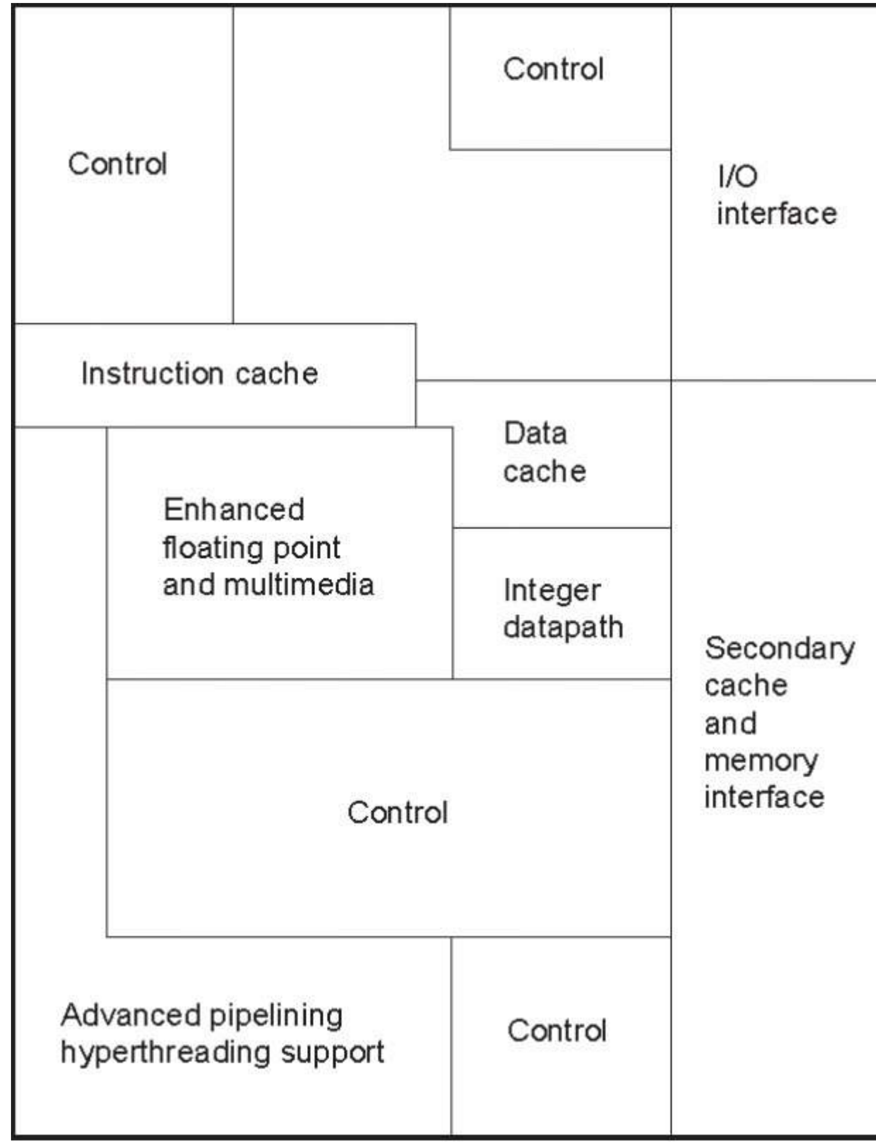




* Slide is (partially) courtesy of Mary Jane Irwin

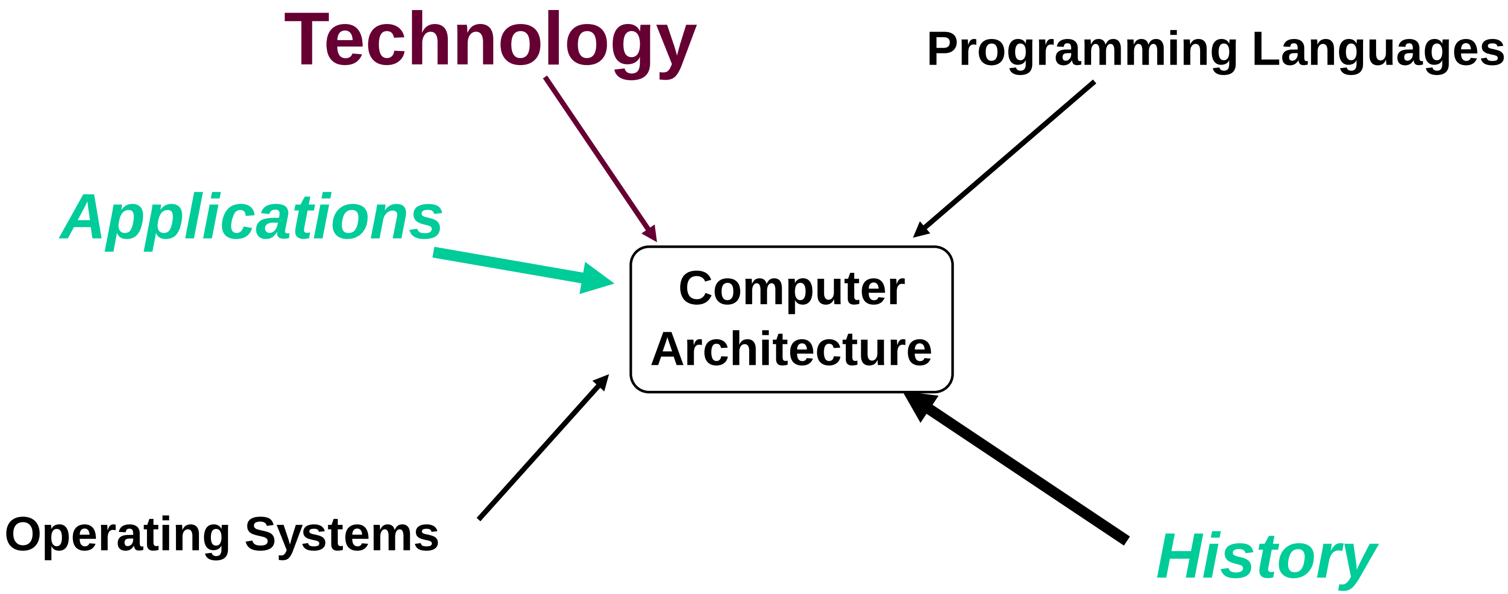
Inside the Pentium 4 Processor Chip





Forces on Computer Architecture

- Programming languages might encourage architecture features to improve performance and code size, e.g. Fortran and Java
- Operating systems rely on the hardware to support essential features such as semaphores and memory management
- Technology always raises the bar for what could be done and changes design's focus
- Applications usually derive capabilities and constrains, e.g. embedded computing
- History always provides the starting point and filter out mistakes



* Figure is courtesy of Dave Patterson

Technology => dramatic change

- **Processor**
  - ➔ logic capacity: about 30% increase per year
  - ➔ clock rate: about 20% increase per year

Higher logic density gave room for instruction pipeline & cache

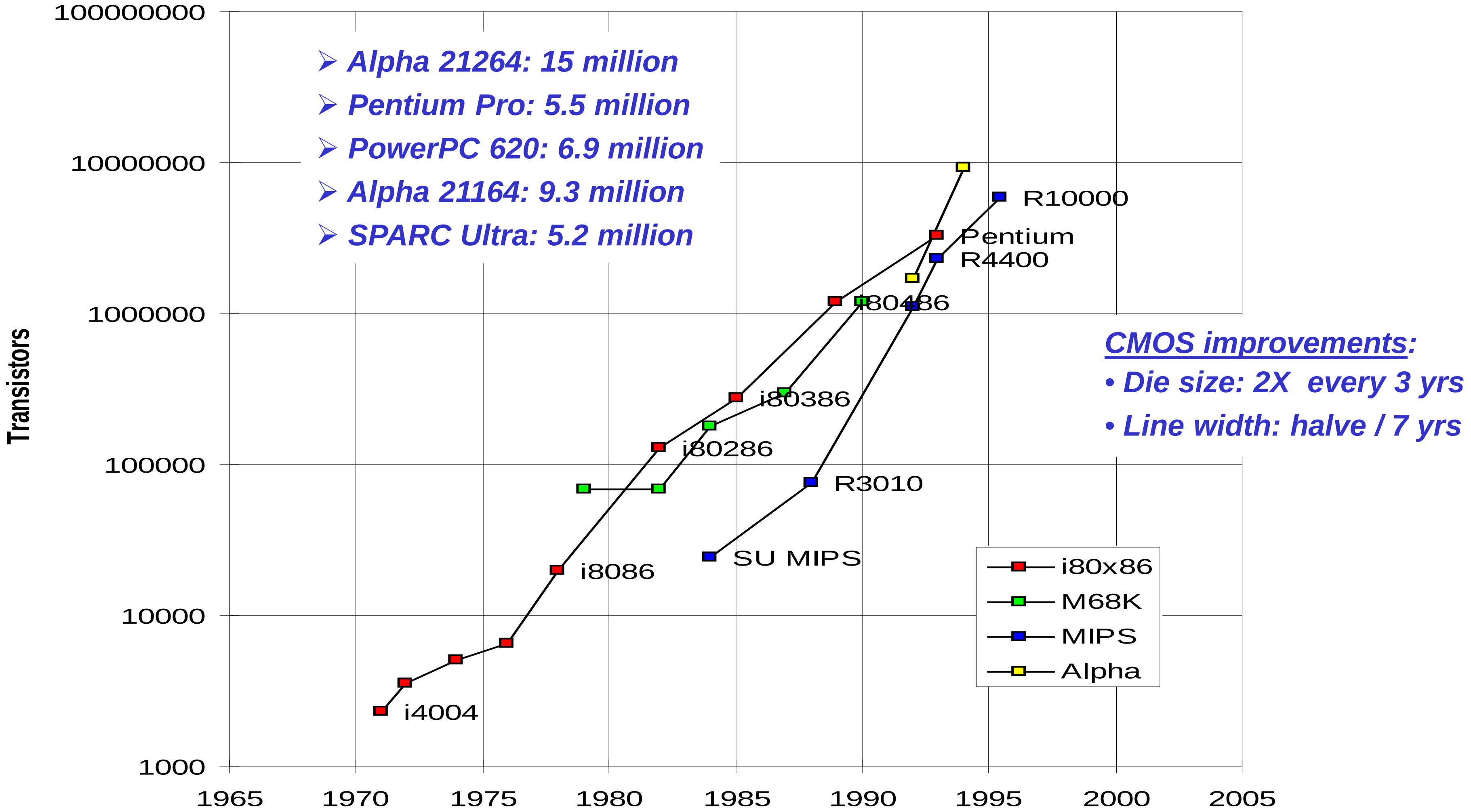
- **Memory**
  - ➔ DRAM capacity: about 60% increase per year (4x every 3 years)
  - ➔ Memory speed: about 10% increase per year
  - ➔ Cost per bit: about 25% improvement per year

Performance optimization no longer implies smaller programs

- **Disk**
  - ➔ Capacity: about 60% increase per year

Computers became lighter and more power efficient

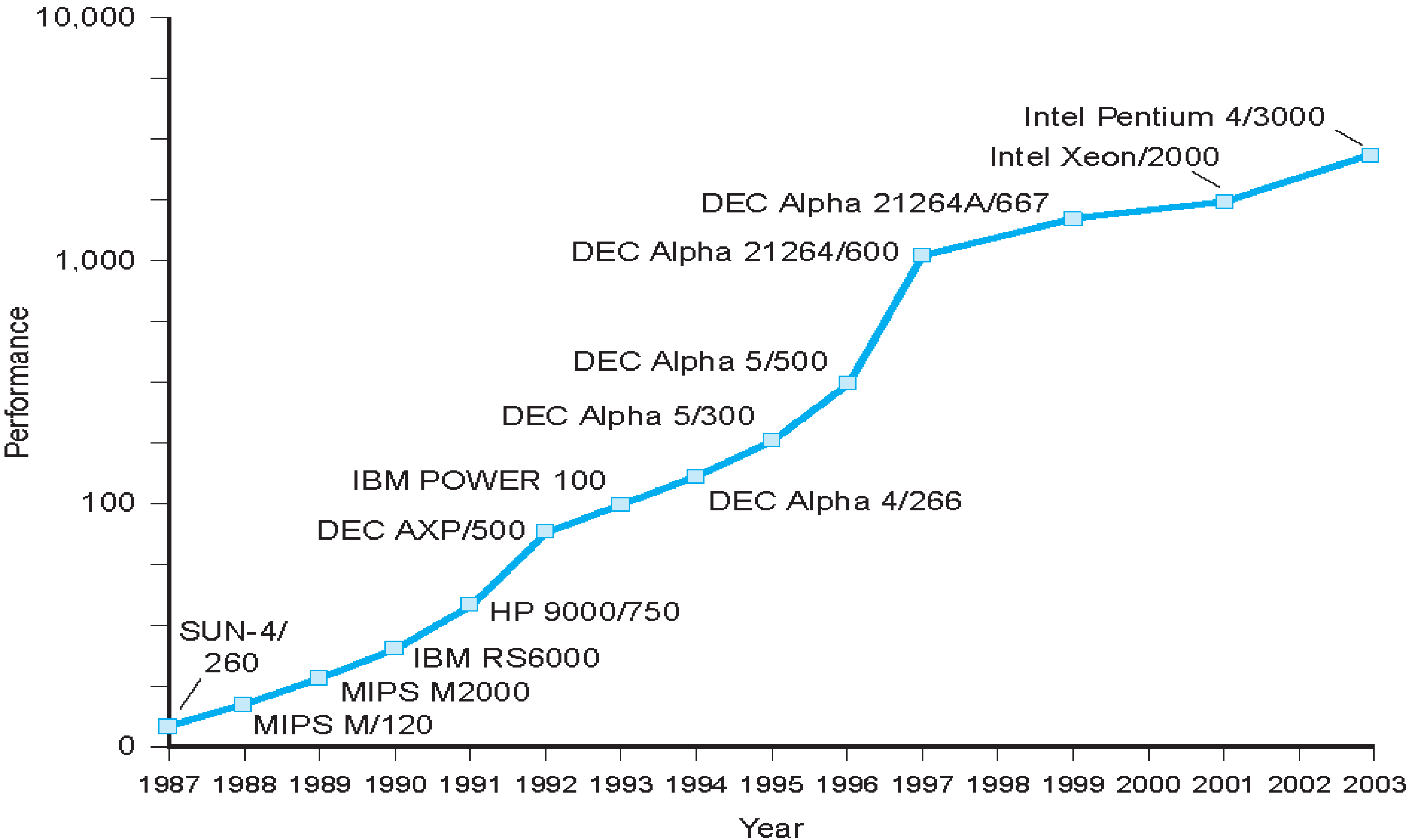
Technology Impact on Processors



- In ~1985 the single-chip processor and the single-board computer emerged
- In the 2004+ timeframe, multi-core processors with increased parallelism

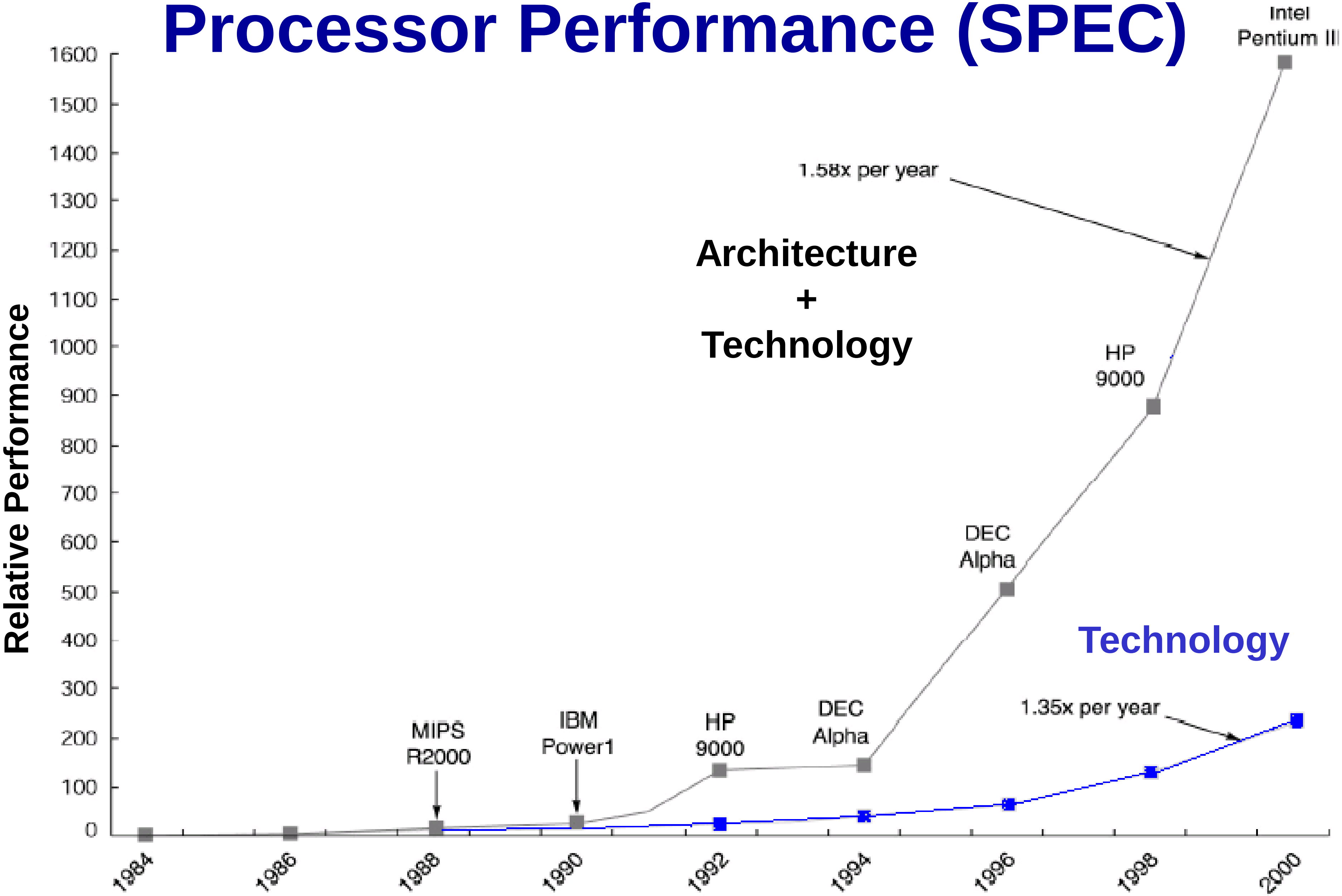
* Figure is courtesy of Dave Patterson

Processor Performance Increase (SPEC)



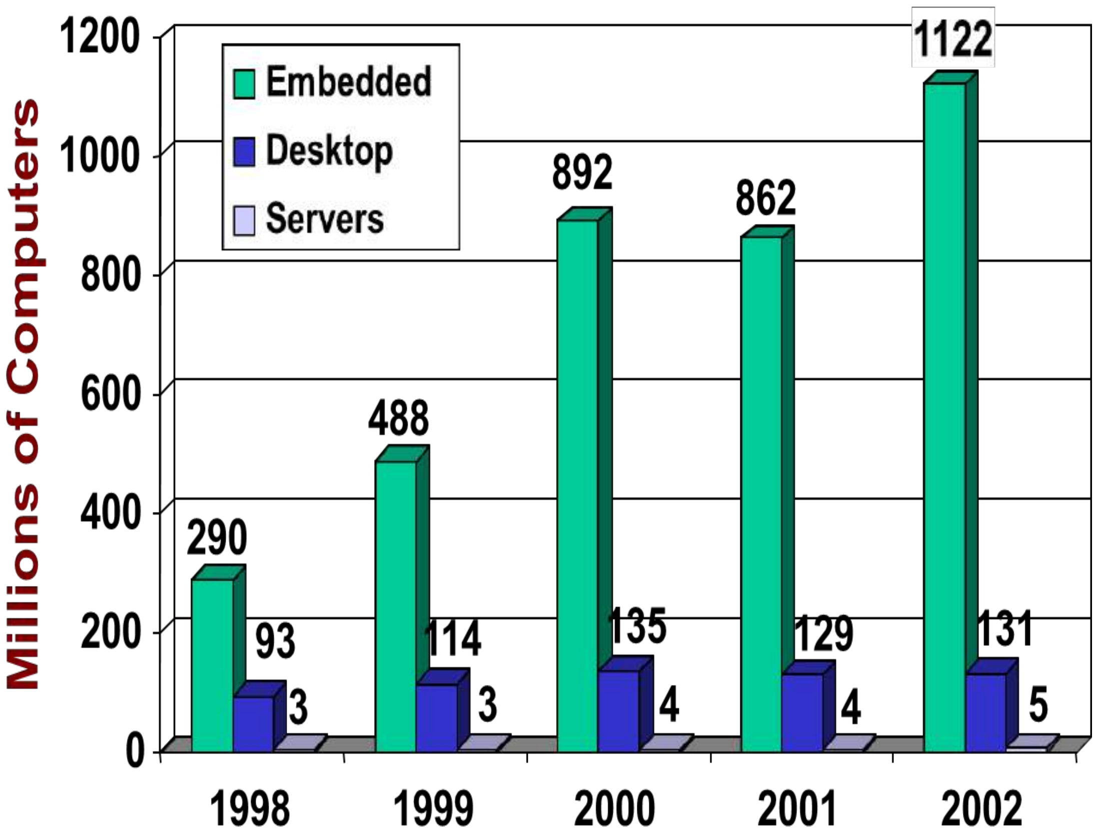
Performance now improves 50% per year (2x every 1.5 years)

Processor Performance (SPEC)



Relying on technology alone would have kept us 8 years behind

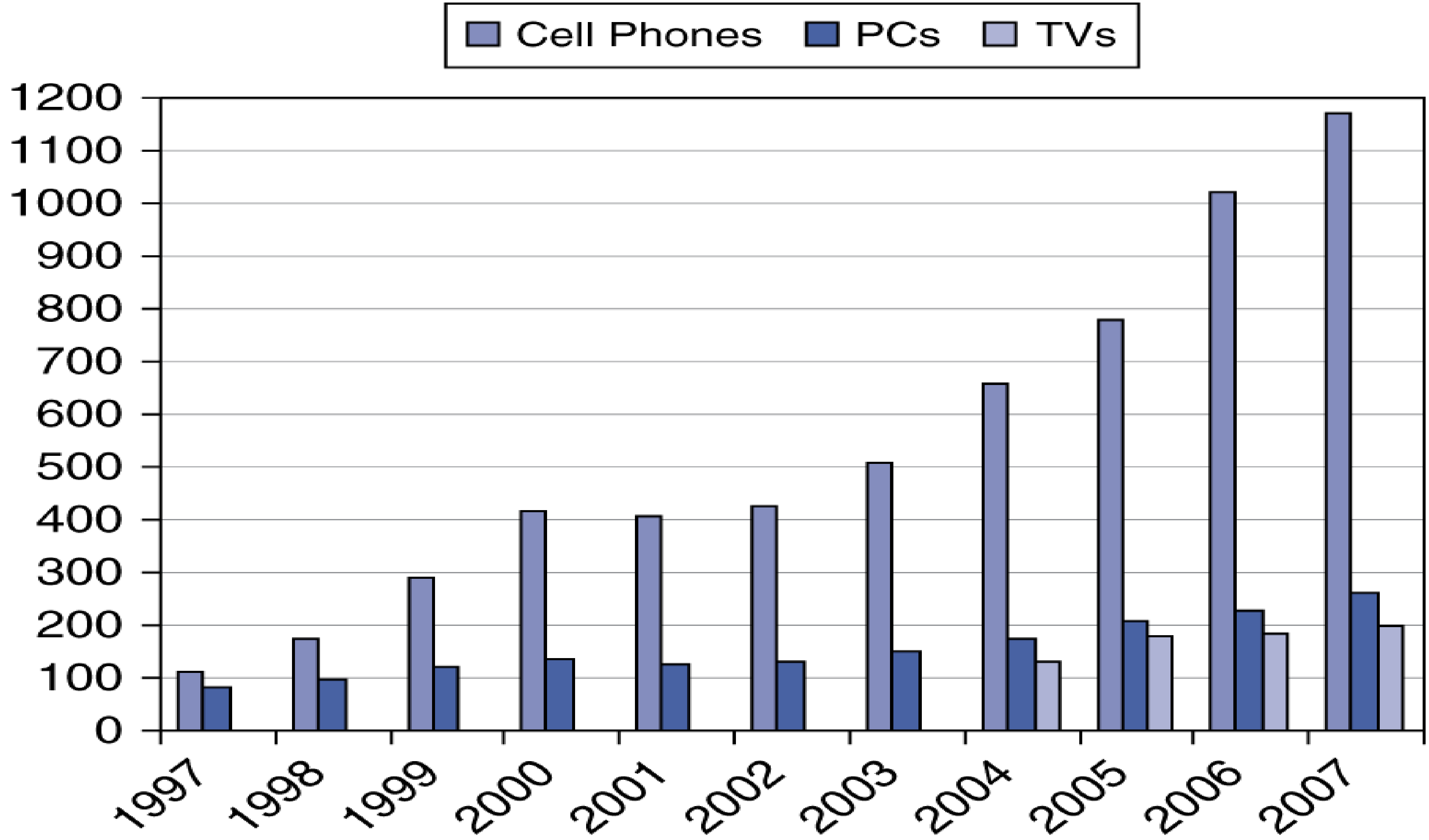
Computers in the Market



- Desktop computers
  - General purpose, variety of software
  - Subject to performance and cost tradeoff
- Server computers
  - Network based
  - High capacity, performance, reliability
  - Range from low-end to very powerful machines
- Embedded computers
  - Hidden as components of systems
  - Stringent power, cost, and performance constraints
  - Cell phones, TV, cars, etc.

Slide is courtesy of Morgan Kaufmann Publishers

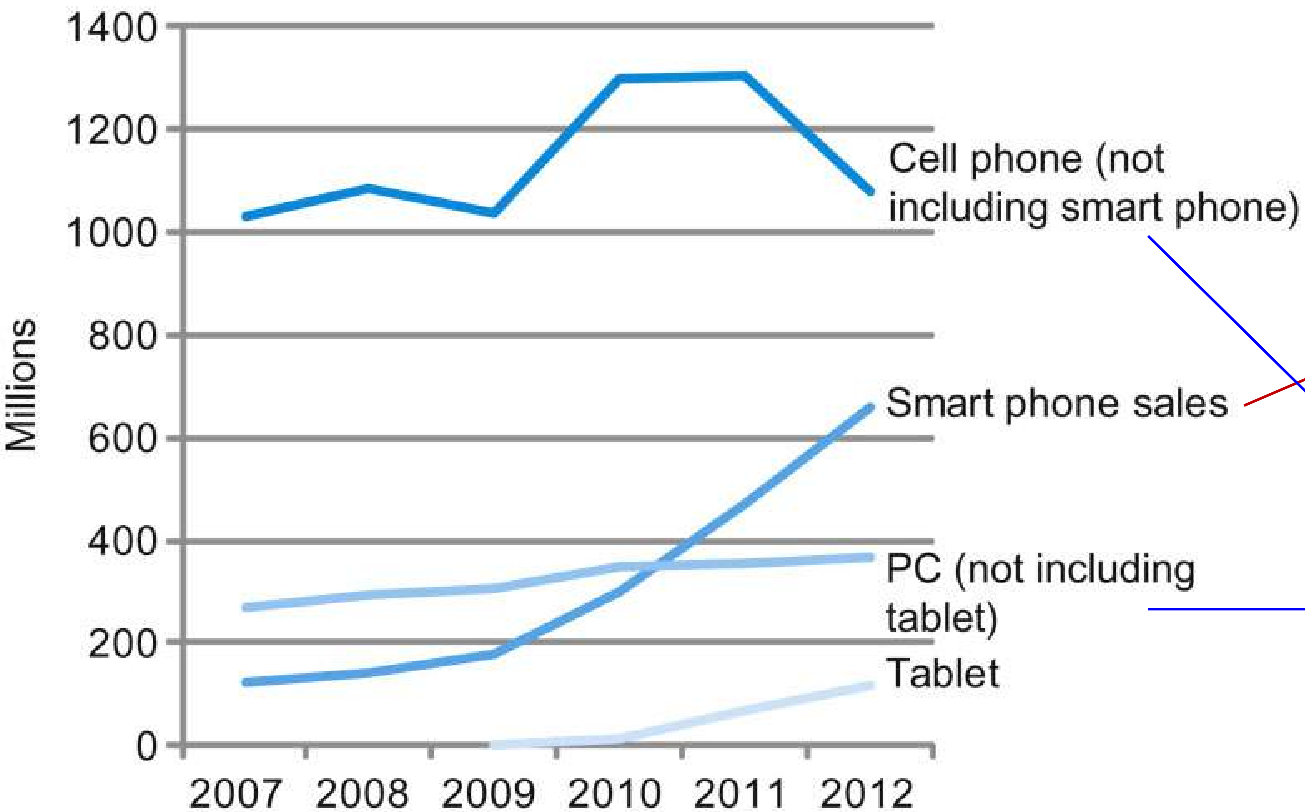
Where is the Market going?



Any where computing and computers every where are not that far away?

* Slide is (partially) courtesy of Mary Jane Irwin

Where is the Market going?



Smart phones represent the recent growth in the cell phone industry, and they passed PCs in 2011

PCs and traditional cell phone categories are relatively flat or declining

- Tablets and smart phones reflect the PostPC era, versus personal computers and traditional cell phones.
- Tablets have fastest growth, nearly doubling between 2011 and 2012.

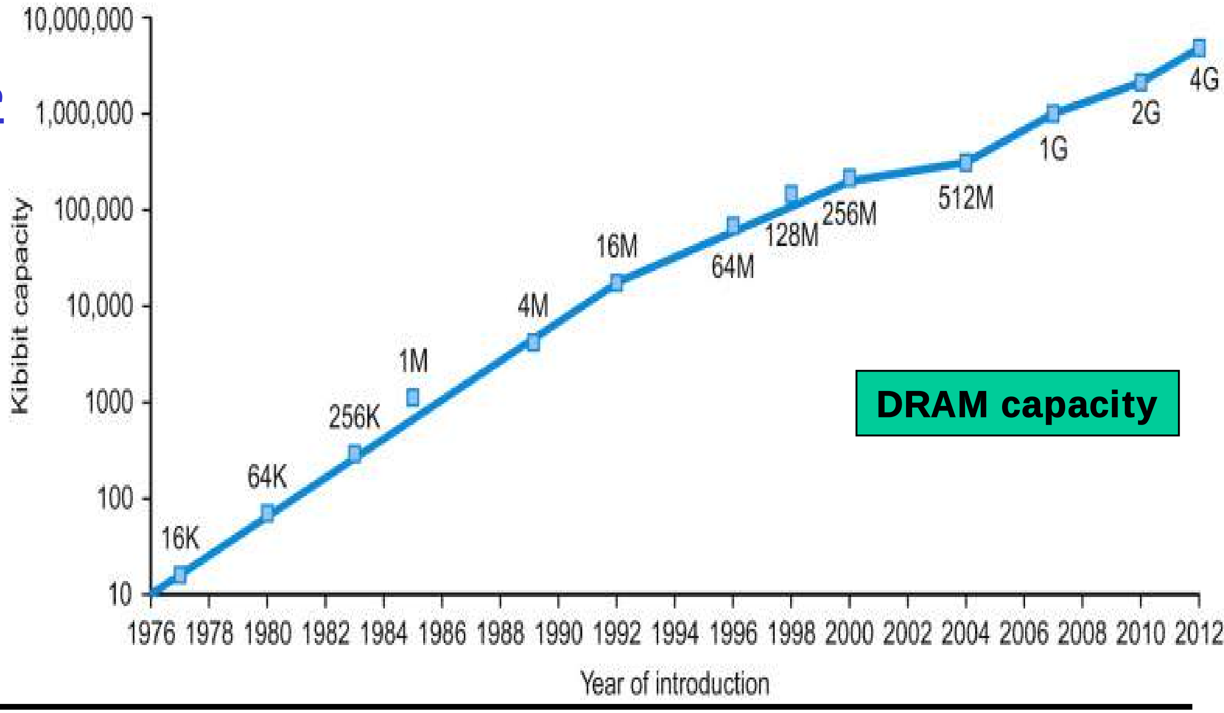
Any where computing and computers every where are a reality

Slide is courtesy of Morgan Kaufmann Publishers

Technology Impact on DRAM

- DRAM capacity has been consistently quadrupled every 3 years, a 60% increase per year, resulting over 16,000 times in 20 years (recently slowed down doubling every 2 years or 4 times every 4 years)
- Processor organization is becoming a main focus of performance optimization
- Technology advances got H/W designer to focus not only on performance but also on functional integration and power consumption (e.g. system on a chip)
- Programming is more concerned with cache and no longer constrained by the RAM size

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



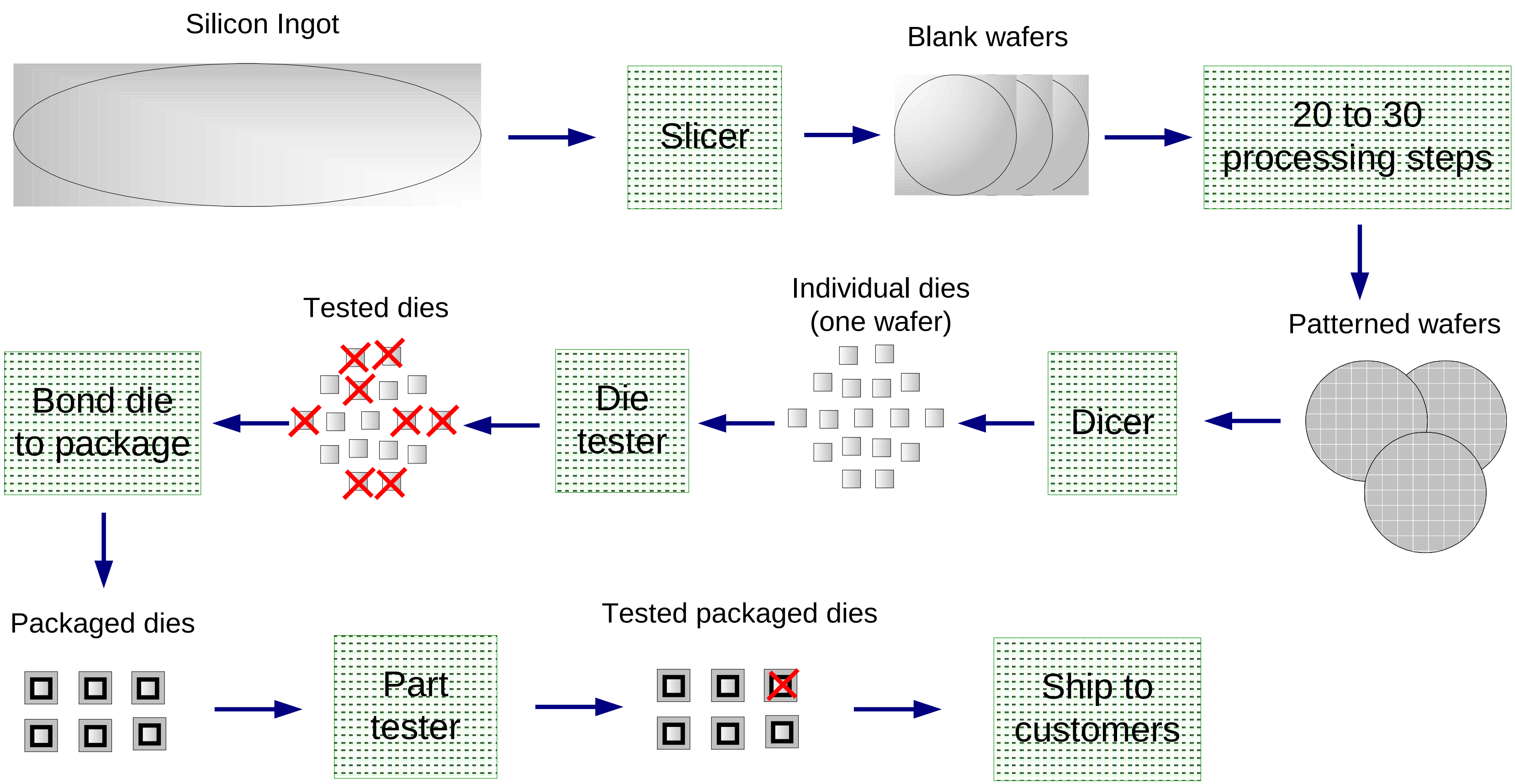
Integrated Circuits: Fueling Innovation

- The manufacture of a chip begins with silicon, a substance found in sand
- Silicon does not conduct electricity well and thus called semiconductor
- A special chemical process can transform tiny areas of silicon to either:
  1. Excellent conductors of electricity (like copper)
  2. Excellent insulator from electricity (like glass)
  3. Areas that can conduct or insulate under a special condition (a switch)
- A transistor is simply an on/off switch controlled by electricity
- Integrated circuits combines dozens of hundreds of transistors in a chip

Advances of the IC technology affect H/W and S/W design philosophy

Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2013	Ultra large scale IC	250,000,000,000

Microelectronics Process

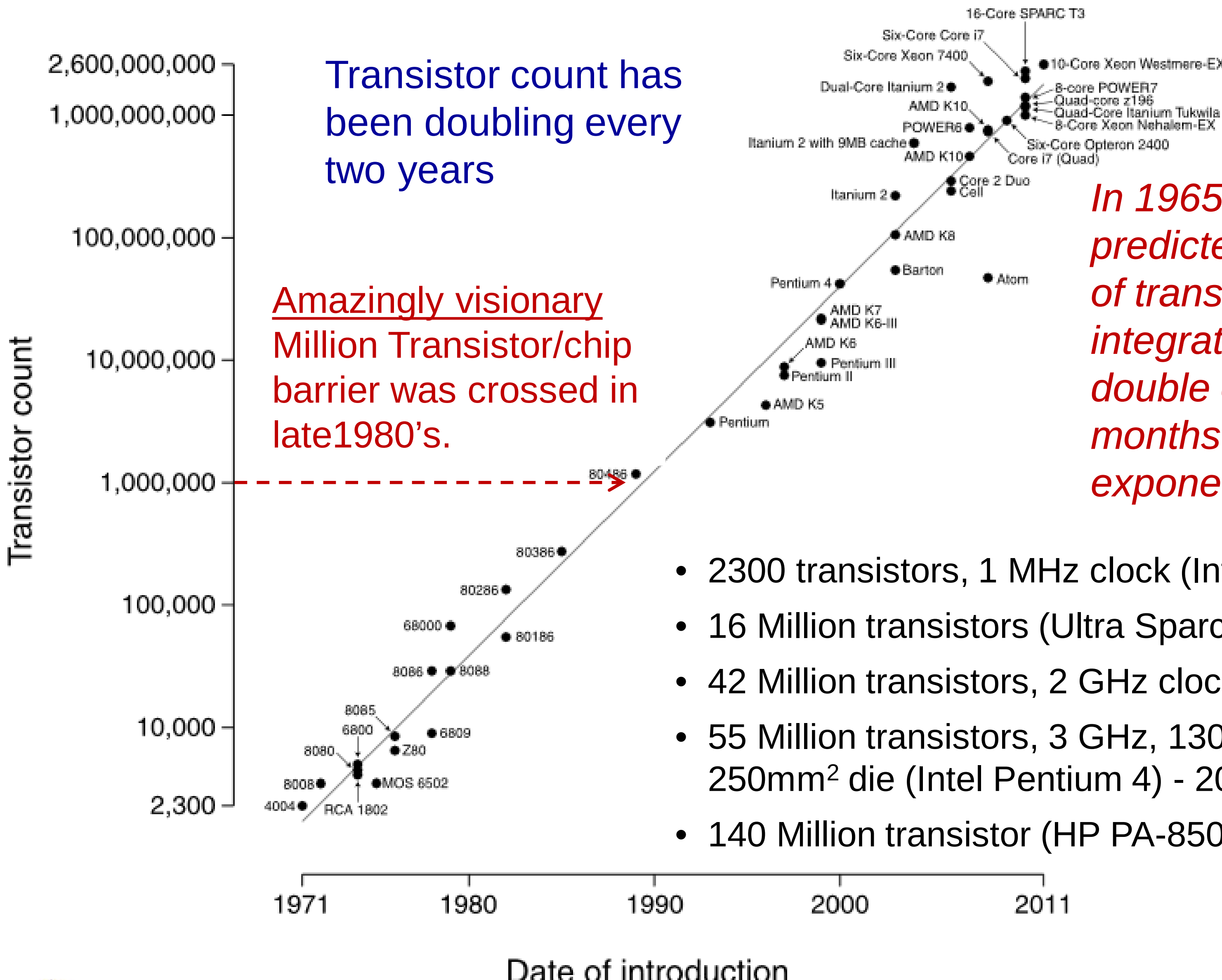


- Silicon ingot are 6-12 inches in diameter and about 12-24 inches long
- The manufacturing process of integrated circuits is critical to the cost of a chip
- Impurities in the wafer can lead to defective devices and reduces the yield

Moore's Law

Transistor count has been doubling every two years

<u>Amazingly visionary</u>
Million Transistor/chip barrier was crossed in late1980's.



In 1965, Gordon Moore predicted that the number of transistors that can be integrated on a die would double every 18 to 24 months (i.e., grow exponentially with time).

- 2300 transistors, 1 MHz clock (Intel 4004) - 1971
- 16 Million transistors (Ultra Sparc III)
- 42 Million transistors, 2 GHz clock (Intel Xeon) – 2001
- 55 Million transistors, 3 GHz, 130nm technology, 250mm^2 die (Intel Pentium 4) - 2004
- 140 Million transistor (HP PA-8500)

Curve is from Wikipedia, text is courtesy of Mary Jane Irwin

Computer Generations

- Computers were classified into 4 generations based on revolutions in the technology used in the development
- By convention, electronic computers are considered as the first generation rather than the electromechanical machines that preceded them
- Today computer generations are not commonly referred to due to the long standing of the VLSI technology and the lack of revolutionary technology in sight

Generations	Dates	Technology	Principal new product
1	1950-1959	Vacuum tube	Commercial electronic computer
2	1960-1968	Transistor	Cheaper computers
3	1969-1977	Integrated circuits	Minicomputer
4	1978- ?	LSI and VLSI	Personal computers and workstations

Historical Perspective

Year	Name	Size ($Ft.^3$)	Power (Watt)	Perform. (adds/sec)	Mem. (KB)	Price	Price/ Perform. vs. UNIVAC	Adjusted price 1996	Adjusted price/perform vs. UNIVAC
1951	UNIVAC 1	1000	124K	1.9K	48	$1M	1	$5M	1
1964	IBM S/360 model 50	60	10K	500K	64	$1M	263	$4.1M	318
1965	PDP-8	8	500	330K	4	$16K	10,855	$66K	13,135
1976	Cray-1	58	60K	166M	32,768	$4M	21,842	$8.5M	15,604
1981	IBM PC	1	150	240K	256	$3K	42,105	$4K	154,673
1991	HP 9000/ model 750	2	500	50M	16,384	$7.4K	3,556,188	$8K	16,122,356
1996	Intel PPro PC 200 Mhz	2	500	400M	16,384	$4.4K	47,846,890	$4.4K	239,078,908

After adjusting for inflation, price/performance has improved by about 240 million in 45 years (about 54% per year)

Conclusion

- ***So what's in it for you?***
  - ➔ In-depth understanding of the inner-workings of modern computers, their evolution, and trade-offs present at the hardware/software boundary.
  - ➔ Experience with the *design process* in the context of a reasonable size hardware design
- ***Why should a programmer care?***
  - ➔ In the 60's and 70's performance was constrained by the size of memory, not an issue today
  - ➔ Performance optimization needs knowledge of memory hierarchy, instruction pipeline, parallel processing, etc.
  - ➔ Systems' programming is highly coupled with the computer organization, e.g. embedded systems

Computer architecture is at the core of computer science & Eng.