

## Lecture 1

### **Course Introduction**

# Course Introduction

---

- Today's topics:
  - ◆ Why computer organization is important
  - ◆ Course information
  - ◆ Background and Computer Abstraction

# Why to Learn Computer Organization?

- Embarrassing if you are a student in CS and can't make sense of the following terms: DRAM, pipelining, cache hierarchies, I/O, virtual memory, ...
- Embarrassing if you are a student in CS and can't decide which processor to buy: 3 GHz P4 or 2.5 GHz Athlon (this course helps us reason about performance/power), ...
- First step for chip designers, compiler/ OS writers
- Knowledge of the hardware will help you write better programs

# Must a Programmer Care about Hardware?

- Must know how to reason about program performance and energy
- CPU Performance: if we understand how CPU process data, we can enhance the computation efficiency
- Memory management: if we understand how/where data is placed, we can help ensure that relevant data is nearby
- Thread management: if we understand how threads interact, we can write smarter multi-threaded programs
- I/O management

# What you have learned?

---

- Binary numbers
- Read and write basic C/Java programs
- Understand the steps in compiling and executing a program
- Digital Circuit, Logic design:
  - ◆ Logical equations, schematic diagrams
  - ◆ Combinational vs. sequential logic
  - ◆ Finite state machines (FSMs)

# What you will learn?

## ■ Major content

- ◆ Basic parts of a computer (processor, memory, disk, etc)
- ◆ Principles of computer architecture: CPU datapath and control unit design
- ◆ Assembly language programming in MIPS
- ◆ Memory hierarchies and design
- ◆ I/O organization and design

## ■ Course goals

- ◆ To learn the organizational structures that determine the capabilities and performance of computer systems
- ◆ To understand the interactions between the computer's architecture and its software
- ◆ To understand cost performance trade-offs

# Key Topics

---

- Introduction (Chapter 1)
  - ◆ Basic terms
  - ◆ Moore's Law, power wall
  - ◆ Core ideas in computer architecture
- Processors (Chapter 2-4)
  - ◆ Assembly language (Chapter 2)
  - ◆ Computer arithmetic (Chapter 3)
  - ◆ Pipelining (Chapter 4)
- Memory (Chapter 5)
- Parallel Processors (Chapter 6)

# The content is useful and important

- Computer organization principles are everywhere
  - ◆ Embedded computer vs. general-purpose computers:
    - Cellphone, Digital Camera, MP3 music player, Industrial process control
- Complex system design
  - ◆ How to partition a problem
  - ◆ Functional Spec → Control & Datapath → Physical implementation
  - ◆ Modern CAD tools
- Both EEs and CSEs need this information in almost all jobs

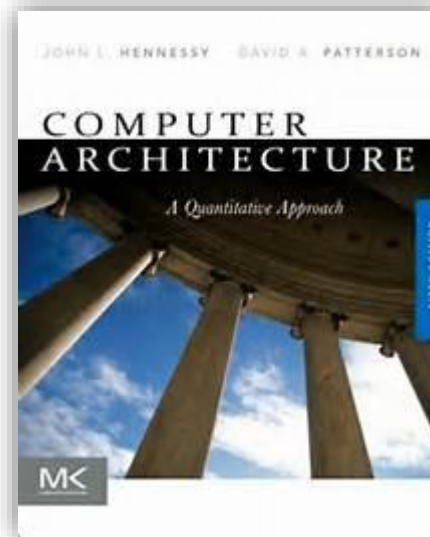


# Course Information

- Course Materials:
  - ◆ Sakai system: <http://sakai.sustc.edu.cn/>
- Lectures:
  - ◆ Monday 14:00-15:50 (1<sup>st</sup> Liyuan Buil. 102)
- Labs
- Instructors:
  - ◆ Prof. Jin Zhang ([zhang.j4@sustc.edu.cn](mailto:zhang.j4@sustc.edu.cn))
  - ◆ Office: i-Park A7, 904
  - ◆ Office hour: Thursday 16:00-18:00
- TA:
  - ◆ Wei Wang ([wangw6@sustc.edu.cn](mailto:wangw6@sustc.edu.cn))
  - ◆ Office: i-Park A7, 913
  - ◆ Office hour: Thursday 16:00-18:00

# Course Information

- Textbook: Computer Organization and Design – the HW/SW Interface, Patterson and Hennessy, 5<sup>th</sup> edition
- Reference book: Computer Architecture - a quantitative approach, Hennessy and Patterson, 5<sup>th</sup> edition



# Patterson and Hennessy



- Turing award 2017

For pioneering a systematic, quantitative approach to the design and evaluation of **computer architectures** with enduring impact on the **microprocessor industry**.



David A. Patterson  
Professor of UC Berkeley  
Distinguished Engineer at Google



John L. Hennessy  
President of Stanford University  
Chairman of Alphabet

# Recommended Reading

- Code: The Hidden Language of Computer Hardware and Software, Charles Petzold, 2000. 《编码的奥秘》, 《编码——隐匿在计算机软硬件背后》
- Computer Systems: A Programmer's Perspective, R. E. Bryant, D. R. O'Hallaron, 3<sup>rd</sup> Edition, Pearson, 2015. 《深入理解计算机系统》

# Course Information

- Assessment:

- ◆ Theoretical part:

- Final exam (~35%)
    - Midterm exam (~35%)
    - Assignments (~5%)

- ◆ Lab part (~25%)

- Please Submit on time

- ◆ all the assignments and reports should be submitted in the Sakai system, late submission will not be accepted in the system.

# Rules about Plagiarism

- No Plagiarism is allowed
  - ◆ If plagiarism on homework or project is found for the first time, **the plagiaristic part is graded as 0** and warning is given to the students
  - ◆ If plagiarism is found for the second time, **the course is graded as 0**
  - ◆ For lab/project report, any sentence that is copied from other paper or article **should cite the original source** as the reference, otherwise, the report is considered as plagiarism
- Submit the commitment letter on Sakai system before homework #1

# Tips for Attending the Lecture

- Having 150 students in one room is horrible
- To get the best use of lecture
  - ◆ interactive
  - ◆ ask whenever you have question, interrupt whenever you want
  - ◆ Ask immediately after the class if you are shy
  - ◆ Give me suggestions and feedback frequently
- Get the main idea in class, read the details after class

# 10 Interventions that Changed World

- ❑ 1. Internet
- ❑ 2. Computer
- ❑ 3. Light Bulb
- ❑ 4. Automobile
- ❑ 5. Steam Engine
- ❑ 6. Telephone
- ❑ 7. Refrigeration
- ❑ 8. Printing Press
- ❑ 9. Wheel
- ❑ 10. The Plow



---

**Source: <http://www.geniusstuff.com>**



# Evolution of the Desk (1980-2015)



# Evolution of the Desk (1980-2015)



**Source:** [http://www.360doc.com/content/14/0919/20/5052258\\_410779481.shtml](http://www.360doc.com/content/14/0919/20/5052258_410779481.shtml)

# Various Forms of Computers



# Classes of Computers

---

- Personal computers
  - ◆ General purpose, variety of software
  - ◆ Subject to cost/performance tradeoff
- Server computers
  - ◆ Network based
  - ◆ High capacity, performance, reliability
  - ◆ Range from small servers to building sized

# Classes of Computers

- Supercomputers
  - ◆ High-end scientific and engineering calculations
  - ◆ Highest capability but represent a small fraction of the overall computer market
- Embedded computers
  - ◆ Hidden as components of systems
  - ◆ Stringent power/performance/cost constraints

# Evolvemement of Computers

**Petaflop**

**Teraflop**

**Gigaflop**





# The PostPC Era

- Cloud computing
  - ◆ Warehouse Scale Computers (WSC)
  - ◆ Software as a Service (SaaS)
  - ◆ Portion of software run on a Personal Mobile Device and a portion run in the Cloud
  - ◆ Amazon and Google
- Data centers
  - ◆ Millions of computers connected by off-the-shelf networking devices

## Google Data Centers



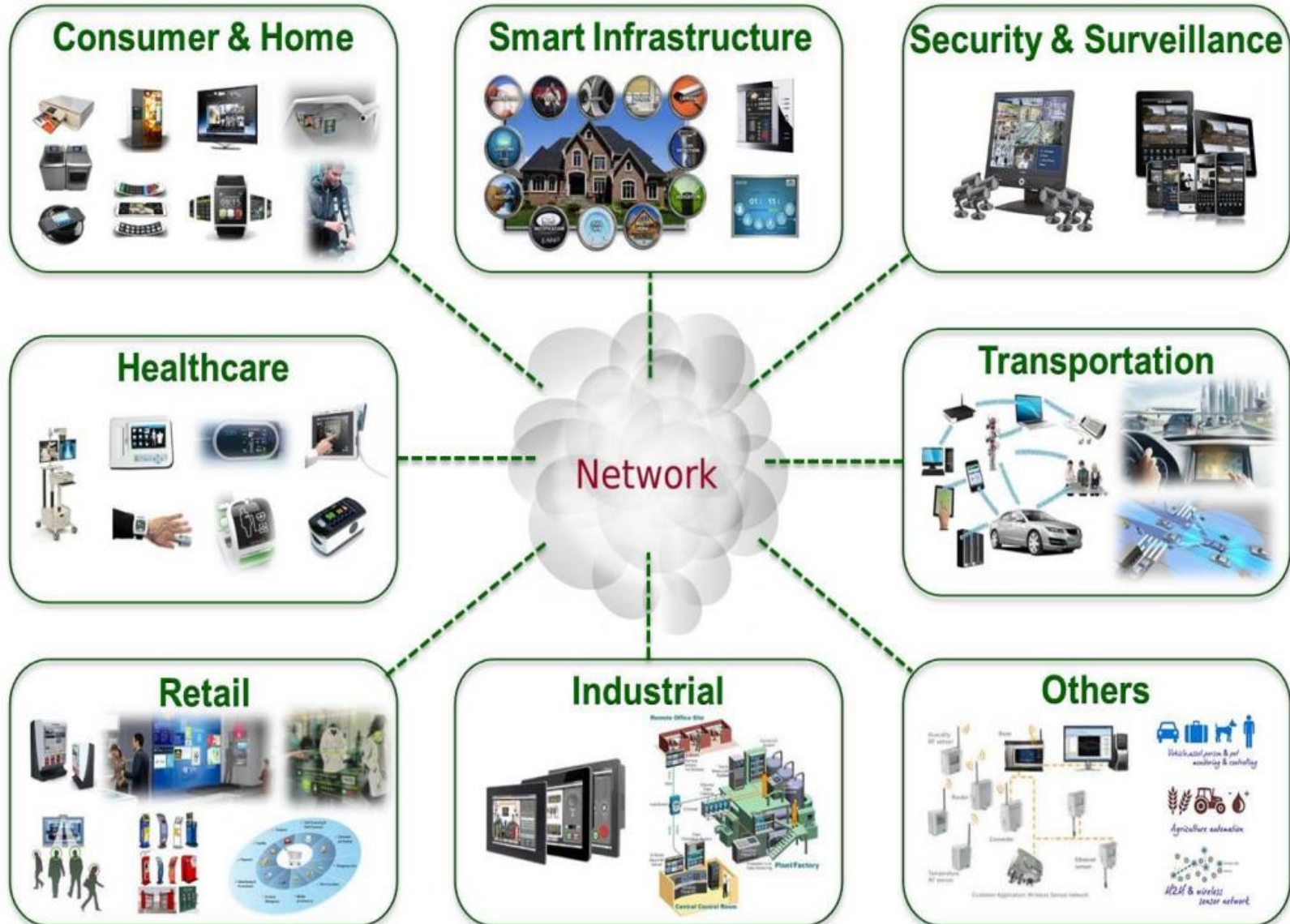
# The PostPC Era

- Personal Mobile Device (PMD)
  - ◆ Battery operated
  - ◆ Connects to the Internet
  - ◆ Hundreds of dollars
  - ◆ Smart phones, tablets, electronic glasses





# Internet of Things

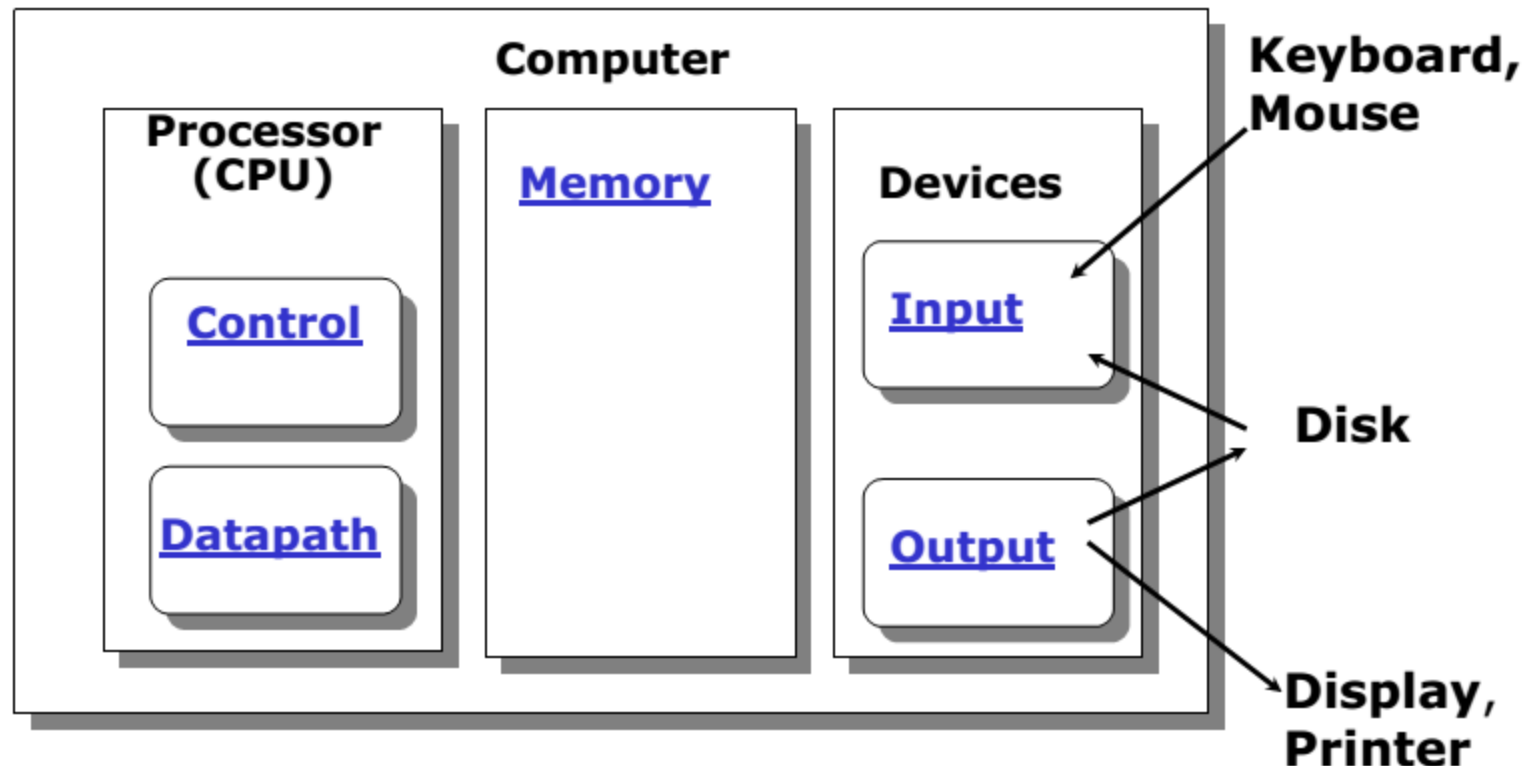


# New Computer Architecture for AI era

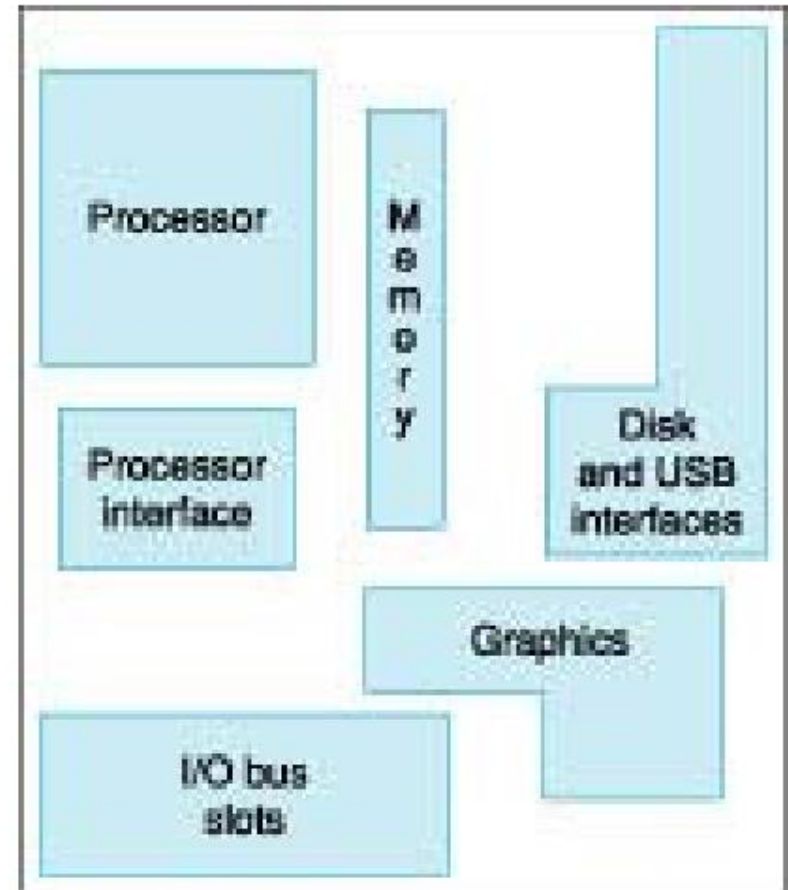
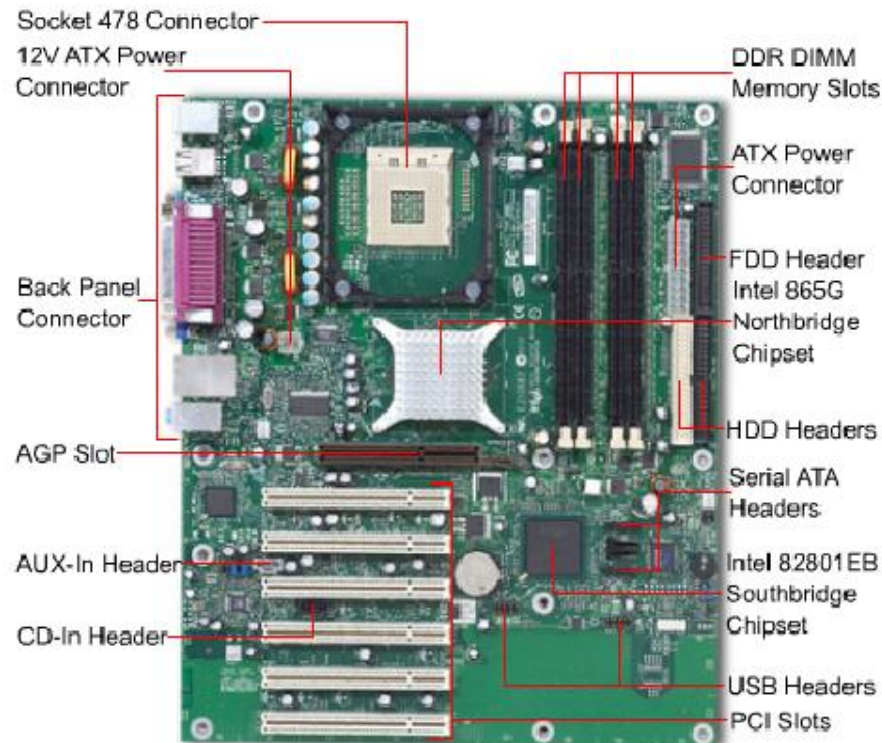
- AI and big data requires new computer architecture
  - ◆ More Suitable for deep learning
  - ◆ High requirement on parallel
  - ◆ Low energy
- From CPU to GPU, TPU...
- AI chips on smartphone
- Dozens of companies dive in this area:
  - ◆ Google, NVIDIA, 华为、地平线、寒武纪、深鉴
  - ◆ Nobody knows who will win

# Components of a Computer

- Same components for all kinds of computer:
  - ◆ Input Device, Output Device, Memory, Processor (Control, Datapath)



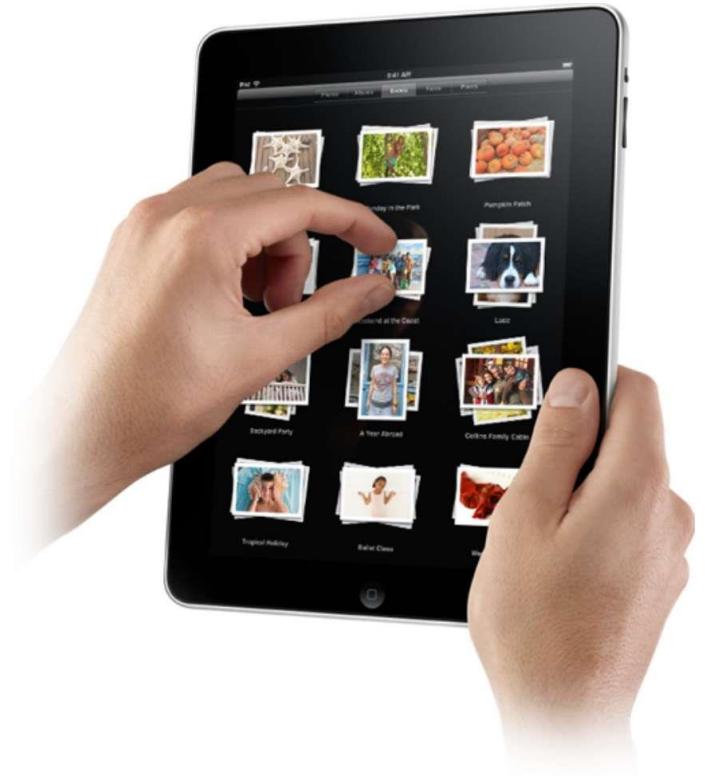
# PC motherboard



Courtesy: [www.tigerdirect.com](http://www.tigerdirect.com)

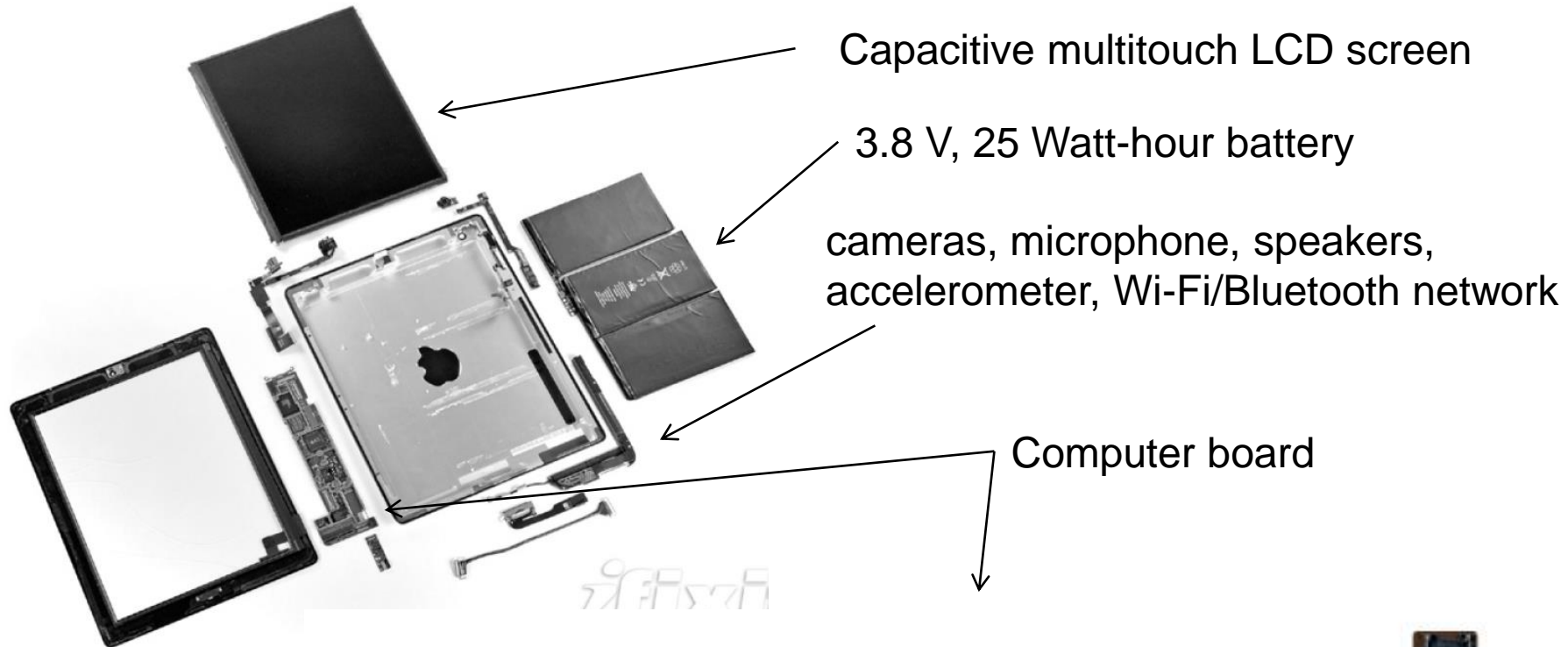
# Let's break up an ipad

- LCD (liquid crystal display)
  - ◆ A matrix of pixels:  
 $1024 * 768 * 24$  bits ( 8 bit for one color)
- Touchscreen
  - ◆ Replace keyboard and mouse in PostPC device
  - ◆ Capacitive screen, which allows multiple touches simultaneously





# Opening the Box



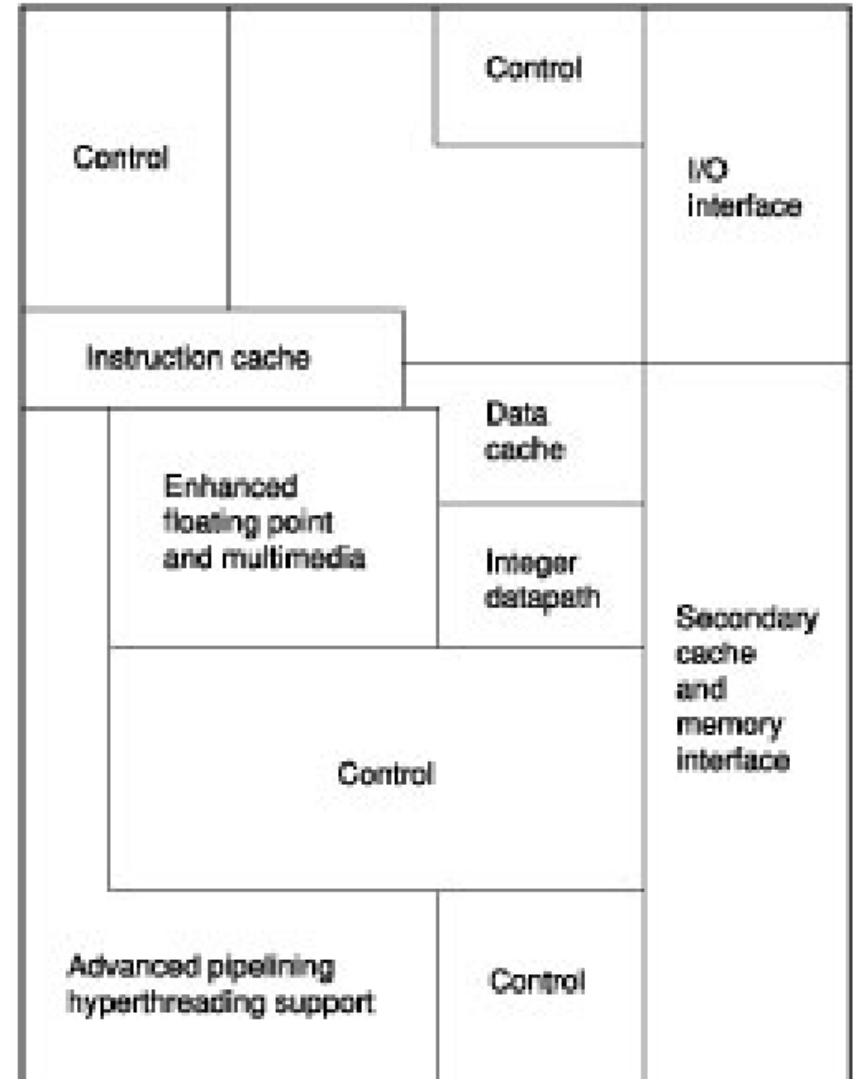
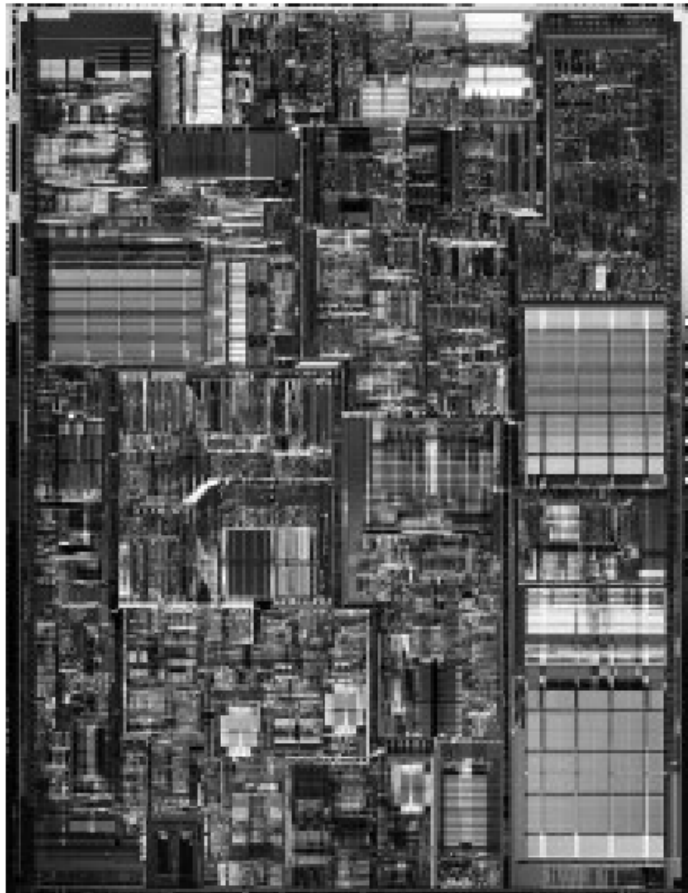
Chips: integrated circuit  
PCB board: traditional circuit

# Components of the ipad

---

- Input device: touch screen, camera, microphone, network
- Output device: LCD, speaker, network
- Memory: flash memory, main memory
- Central processor unit (CPU):
  - ◆ A5 processor

# Inside the Processor





# Inside the Processor

- Apple A5



# Inside the Processor (CPU)

---

- Datapath: performs operations on data
- Control: control the sequence of datapath, memory, I/O
- Cache memory
  - ◆ Small fast SRAM memory for immediate access to data

# Technologies for Processors and Memory

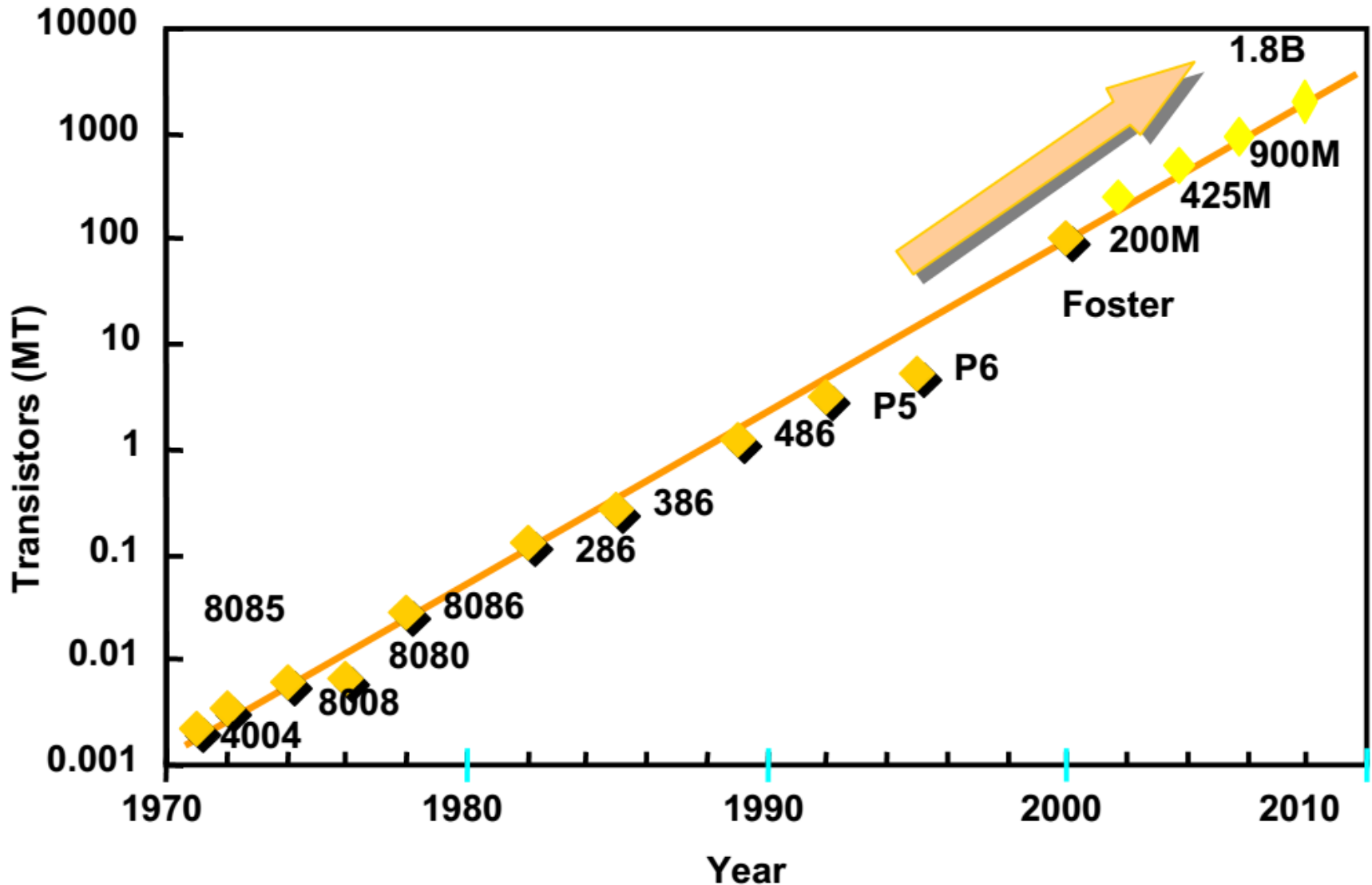
---

- Processors
- Memory
- I/O

# Micro Processor Advances-Moore's Law

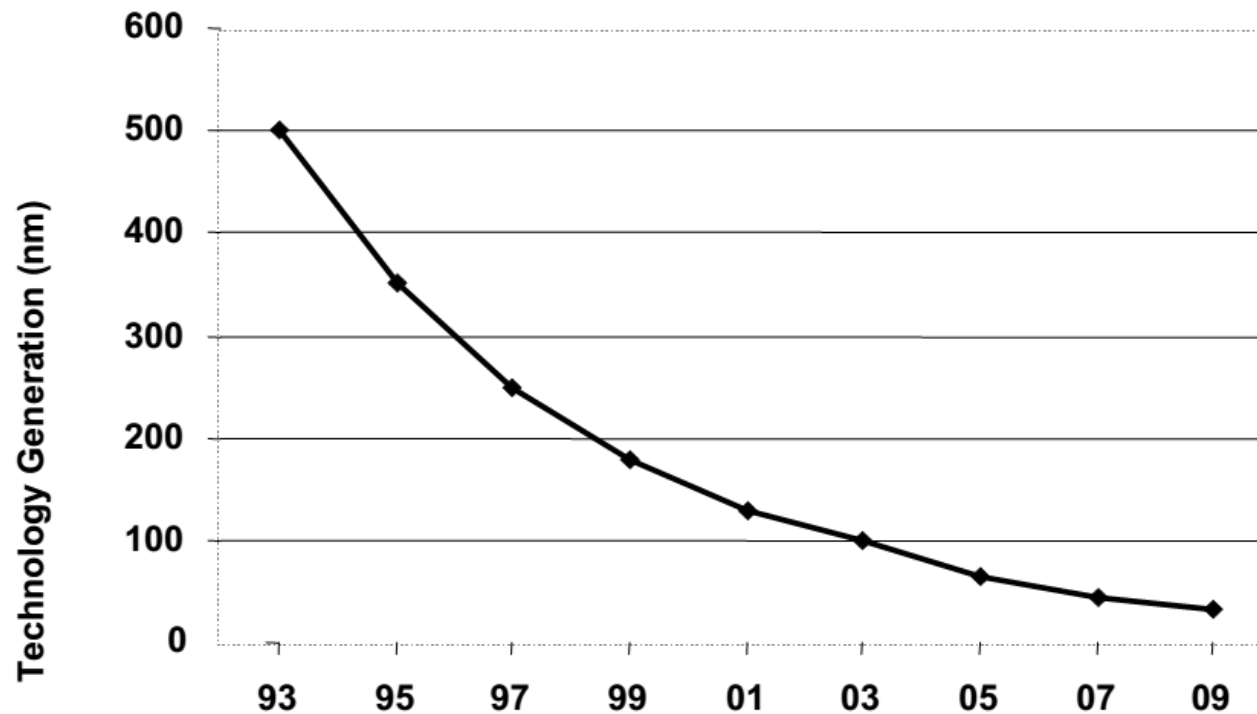
- In 1965, Gordon Moore made a prediction
  - ◆ **The number of transistors that can be integrated on a die would double every 18 to 24 months**
- Amazingly visionary – million transistor/chip barrier was crossed in the 1980's
  - ◆ 2300 transistors, 1 MHz clock (Intel 4004) – 1971
  - ◆ 16 Million transistors (Ultra Sparc III)
  - ◆ 42 Million transistors, 2 GHz clock (Intel Xeon) – 2001
  - ◆ 55 Million transistors, 3 GHz, 130nm technology, 250mm<sup>2</sup> die (Intel Pentium 4) – 2004
  - ◆ 140 Million transistor (HP PA-8500)
  - ◆ 1.8 Billion transistors (Itanium II)

# Moore's Law



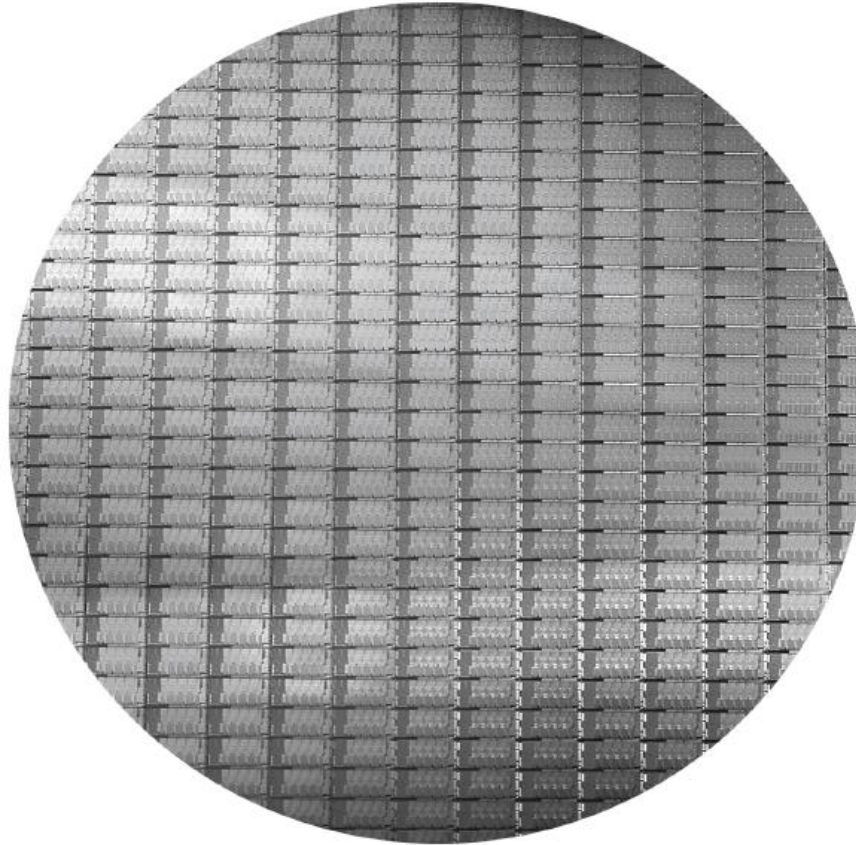
# How is that possible?

- Scale the transistor channel length



Feature size scaling to reduce die size

# Intel Core i7 Wafer



- 300mm wafer, 280 chips, 32nm technology
- Each chip is 20.7 x 10.5 mm

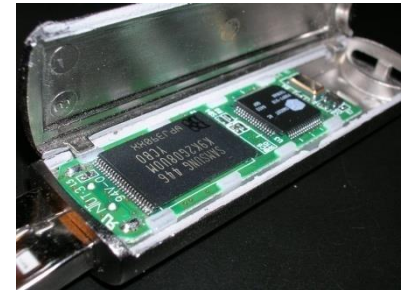
# Processor Technology Trends

- Shrinking of transistor sizes: 250nm (1997) → 130nm (2002) → 70nm (2008) → 35nm (2014)
- Transistor density increases by 35% per year and die size increases by 10-20% per year... functionality improvements!
- Transistor speed improves linearly with size (complex equation involving voltages, resistances, capacitances)
- Wire delays do not scale down at the same rate as transistor delays



# Storage

- Volatile main memory
  - ◆ Loses instructions and data when power off
- Non-volatile secondary memory
  - ◆ Magnetic disk
  - ◆ Flash memory
  - ◆ Optical disk (CDROM, DVD)

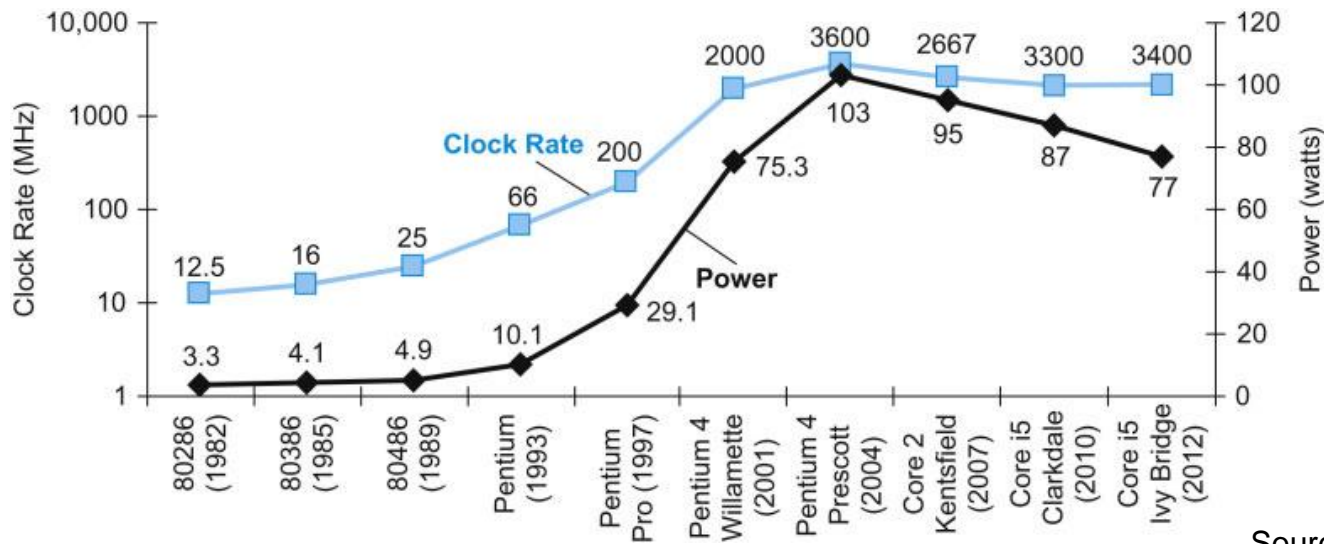


# Memory and I/O Technology Trends

- DRAM density increases by 40-60% per year, latency has reduced by 33% in 10 years (the memory wall!), bandwidth improves twice as fast as latency decreases
- Disk density improves by 100% every year, latency improvement similar to DRAM
- Networks: primary focus on bandwidth; 10Mb → 100Mb in 10 years; 100Mb → 1Gb in 5 years

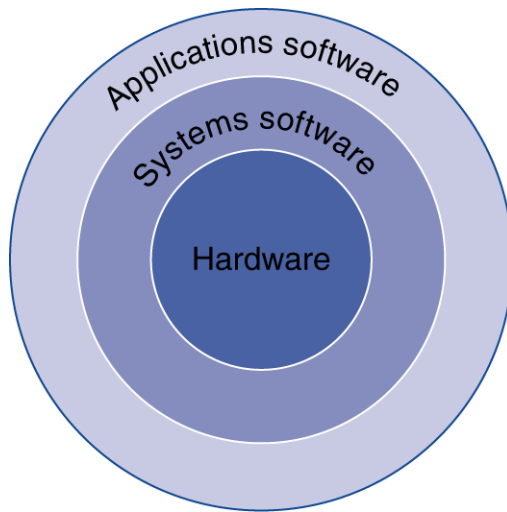
# Power Consumption Trends

- Dyn power  $\propto$  activity x capacitance x voltage<sup>2</sup> x frequency
- Voltage and frequency are somewhat constant now, while capacitance per transistor is decreasing and number of transistors (activity) is increasing
- Leakage power is also rising (function of #trans and voltage)



Source: H&P Textbook

# Between Your Program and Hardware



- Application software
  - ◆ Written in high-level language (HLL)
- System software
  - ◆ Compiler: translates HLL code to machine code
  - ◆ Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - ◆ Processor, memory, I/O controllers

# Levels of Program Code

- High-level language
  - ◆ Level of abstraction closer to problem domain
  - ◆ Provides for productivity and portability
- Assembly language
  - ◆ Textual representation of instructions
- Hardware representation
  - ◆ Binary digits (bits)
  - ◆ Encoded instructions and data

High-level  
language  
program  
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

Compiler

Assembly  
language  
program  
(for MIPS)

```
swap:
  muli $2, $5, 4
  add  $2, $4, $2
  lw   $15, 0($2)
  lw   $16, 4($2)
  sw   $16, 0($2)
  sw   $15, 4($2)
  jr   $31
```

Assembler

Binary machine  
language  
program  
(for MIPS)

```
000000001010000100000000000011000
000000000000110000001100000100001
100011000110001000000000000000000
100011001111001000000000000000100
101011001111001000000000000000000
101011000110001000000000000000100
00000011111000000000000000001000
```

# Eight Great Ideas

- Design for *Moore's Law*
- Use *abstraction* to simplify design
- Make the *common case fast*
- Performance *via parallelism*
- Performance *via pipelining*
- Performance *via prediction*
- *Hierarchy* of memories
- *Dependability* via redundancy

