CS4100: 計算機結構

Computer Abstractions and Technology

國立清華大學資訊工程學系

一零零學年度第二學期

國立清華大學
National Tsing Hua University

Outline

- **Computer: A historical perspective**
- **Abstractions**
- **Technology**
  - **Performance**
    - **Definition**
    - **CPU performance**
  - **Power trends: multi-processing**
  - **Measuring and evaluating performance**
  - **Cost**

電腦是什麼時候發展出來的?

大約一千三百多年前...

為什麼我們不稱它為「電腦」?

電動算盤

「電腦」到底是什麼?

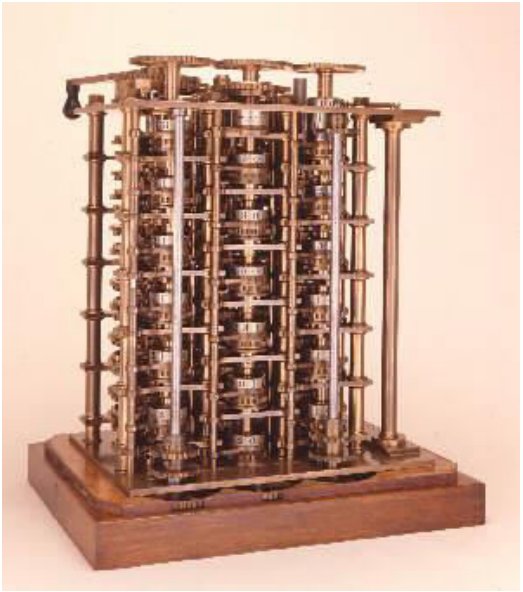
- **A device that computes, especially a programmable electronic machine that performs high-speed mathematical or logical operations or that assembles, stores, correlates, or otherwise processes information**
  -- *The American Heritage Dictionary of the English Language*, 4th Edition, 2000

其實歷史上已有許多計算裝置發展出來

- **Special-purpose versus general-purpose**
- **Non-programmable versus programmable**
- **Scientific versus office data processing**
- **Mechanical, electromechanical, electronic, …**

Tabulating machine
(H. Hollerith, 1889)

Harvard Mark I
(IBM, H. Aiken, 1944)



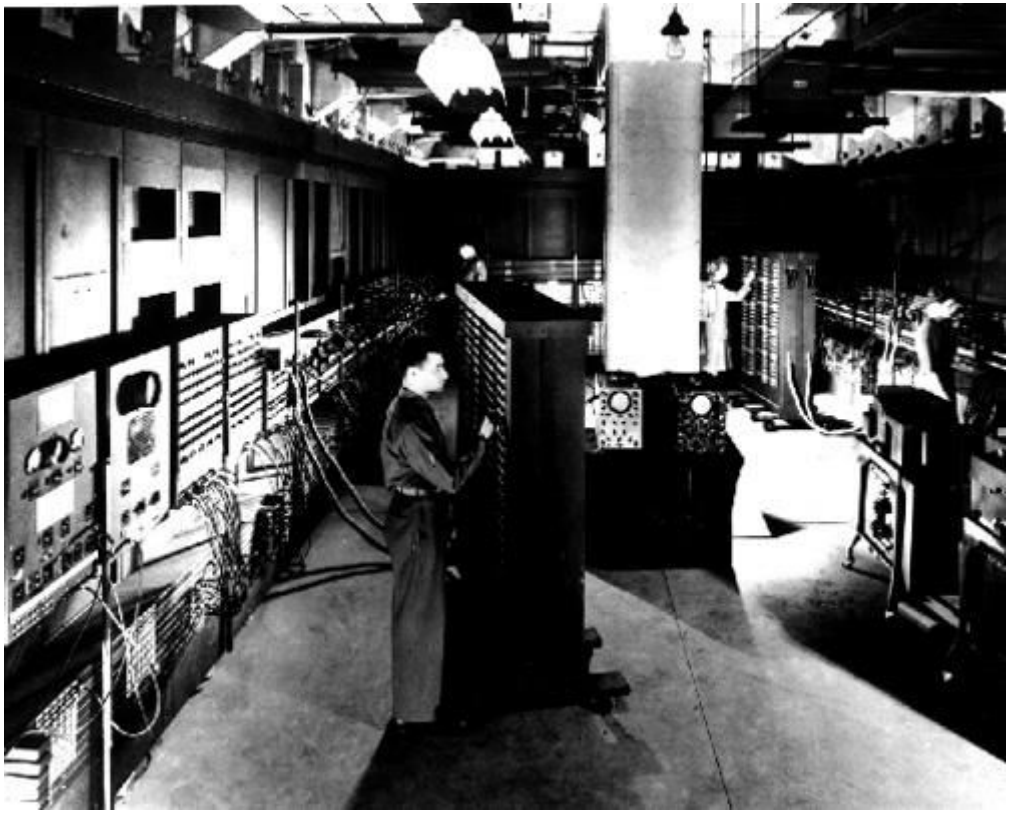
Difference Engine
(C. Babbage, 1822)

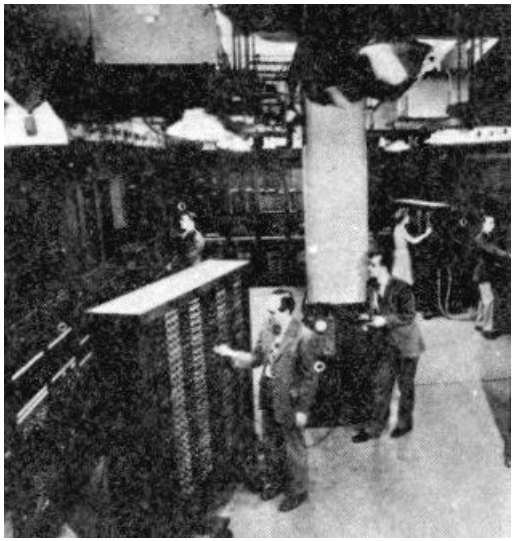
第一部
全電子式
可程式
一般用途
的電腦
是什麼時候發展出來的?

第一部「電」腦

- 一般認為：ENIAC (*Electronic Numerical Integrator and Calculator*)
- Work started in 1943 in Moore School of Electrical Engineering at the University of Pennsylvania, by John Mauchly and J. Presper Eckert
- Completed in 1946
- 約25公尺長、2.5公尺高
- 20 10-digit registers, each 2 feet
- 使用18,000個真空管 (electronic switches, 1906年發明)
- 每秒執行1900個加法
- Programming manually by plugging cables and setting switches

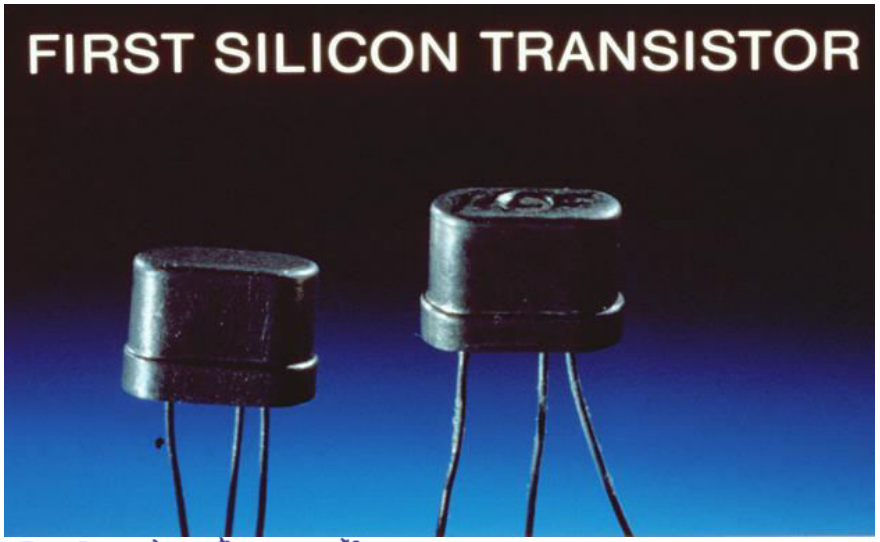
ENIAC

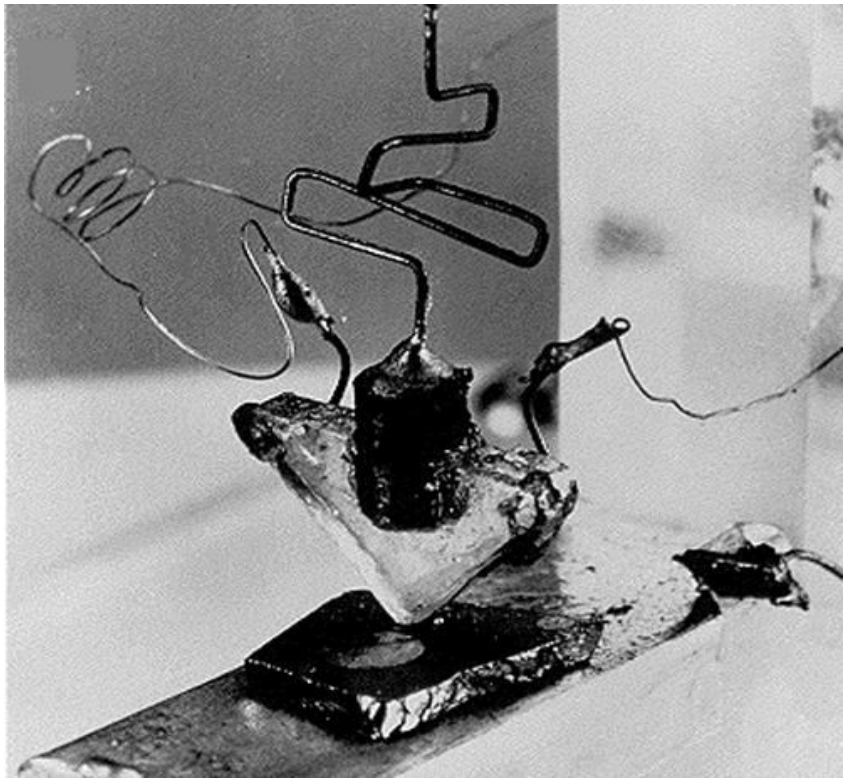




大約同一時期，人們發明了電晶體

- By W. Shockley, J. Bardeen, W. Brattain of Bell Lab. in 1947
  - Much more reliable than vacuum tubes
  - Electronic switches in "solids"





不久後電腦開始商品化

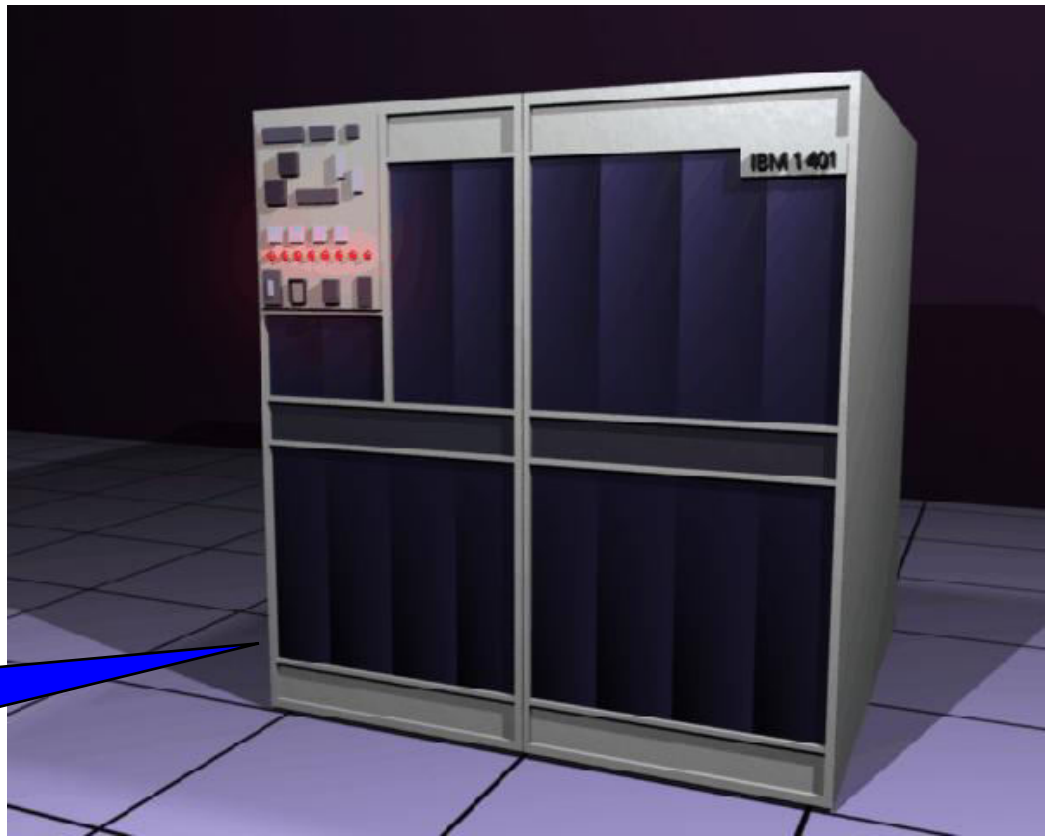
UNIVAC (Remington-Rand, 1951)

主要用途為商務、辦公室自動化
其次為科學計算

IBM 701 (IBM, 1952)

使用電晶體的電腦也跟著出現

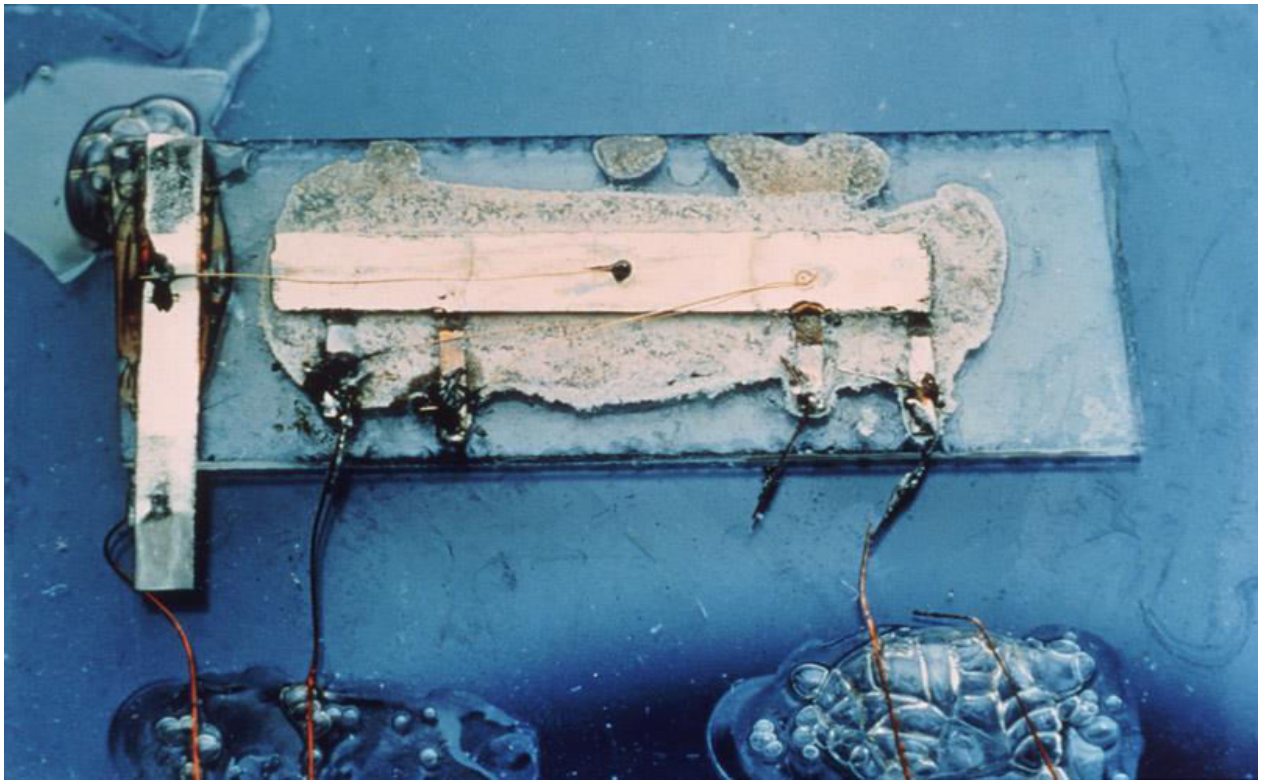
- **Ex.: IBM 1401 (IBM, 1959)**



This is how IBM is called "Big Blue"!

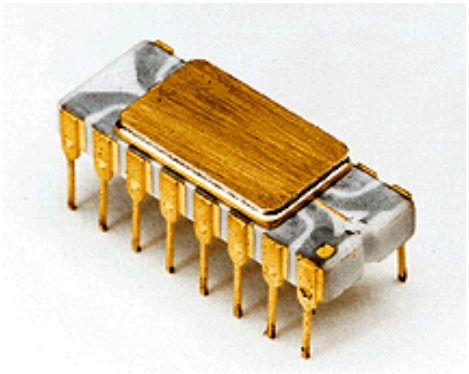
電腦元件的另一大突破是IC

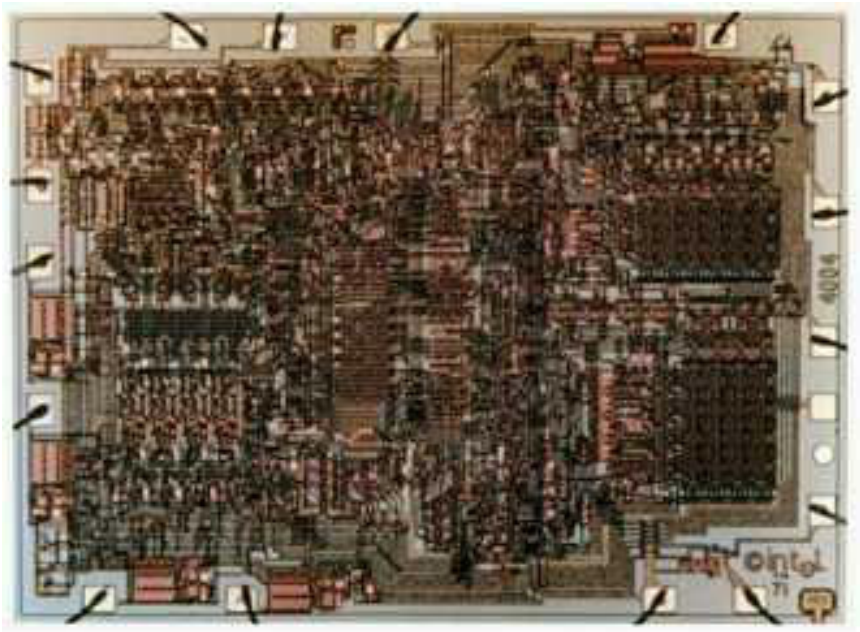
- 1958年德州儀器公司的Jack Kilby: integrated a transistor with resistors and capacitors on a single semiconductor chip, which is a monolithic IC



當更多的電晶體能放入IC後...

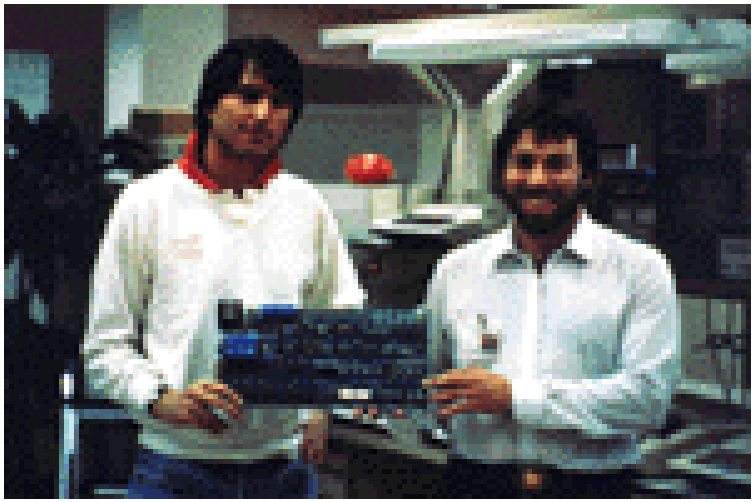
- 1971年第一個微處理器：Intel 4004
  - 108 KHz, 0.06 MIPS
  - 2300 transistors (10 microns)
  - Bus width: 4 bits
  - Memory addr.: 640 bytes
  - For Busicom calculator (original commission was 12 chips)





微處理器造就了...

- **1977年Apple II: Steve Jobs, Steve Wozniak Motorola 6502 CPU, 48Kb RAM**



以及PC

- **1981年IBM PC: Intel 8088, 4.77MHz, 16Kb RAM, two 160Kb floppy disks**

Microsoft Corporation, 1978

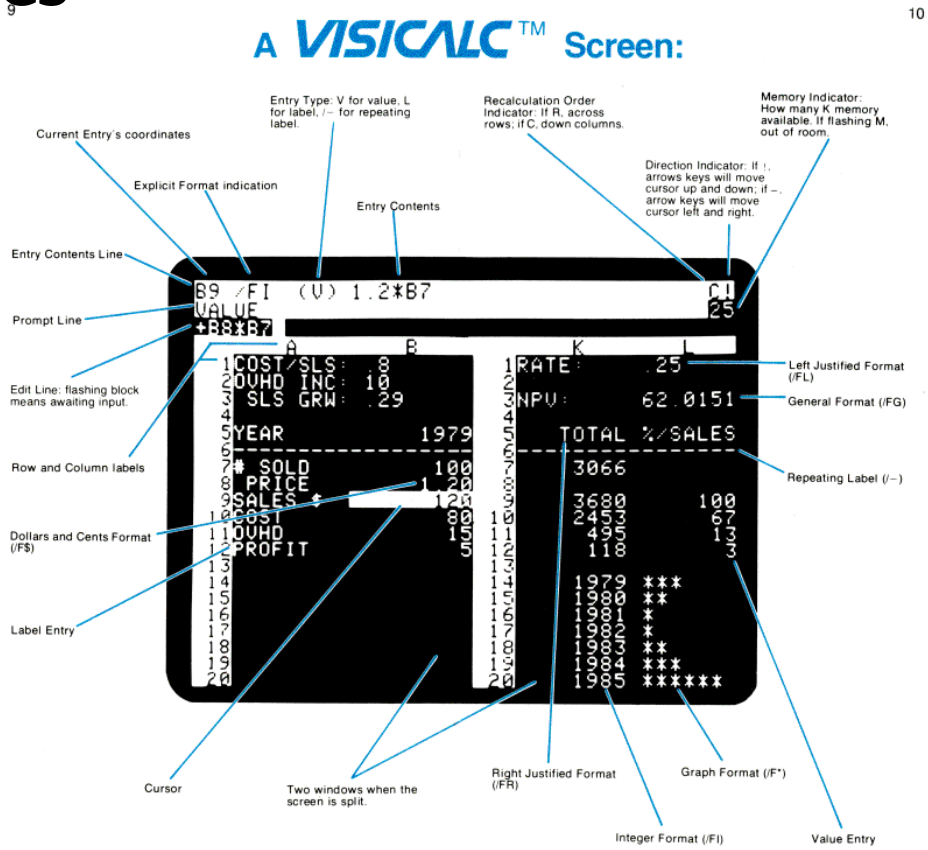
也造就了微軟

一些週邊設備也早已發展出來

- **1973: Researchers at Xerox PARC developed an experimental PC: Alto**
  - **Mouse, Ethernet, bit-mapped graphics, icons, menus, WYSIWG editing**
- **Hosted the invention of:**
  - **Local-area networking**
  - **Laser printing**
  - **All of modern client / server distributed computing**

讓PC成為真正有用的東西--應用程式

- **1979: 1st electronic spreadsheet (VisiCalc for Apple II) by Don Bricklin and Bob Franston**
  - **"The killer app for early PCs"**
  - **Followed by dBASE II, ...**



人們也先後發展出許多其他東西...

80年代，IC的集成進入VLSI

- New processor architecture was introduced: RISC (*Reduced Instruction Set Computer*)
  - IBM: John Cocke
  - UC Berkeley: David Patterson
  - Stanford: John Hennessy

- Commercial RISC processors around 1985
  - MIPS: MIPS
  - Sun: Sparc
  - IBM: Power RISC
  - HP: PA-RISC
  - DEC: Alpha
- They compete with CISC (complex instruction set computer) processors, mainly Intel x86 processors, for the next 20 years

後來的故事 ...

在計算機結構方面比較不精彩
不過似乎後PC的時代已經來臨
(Embedded Computer)

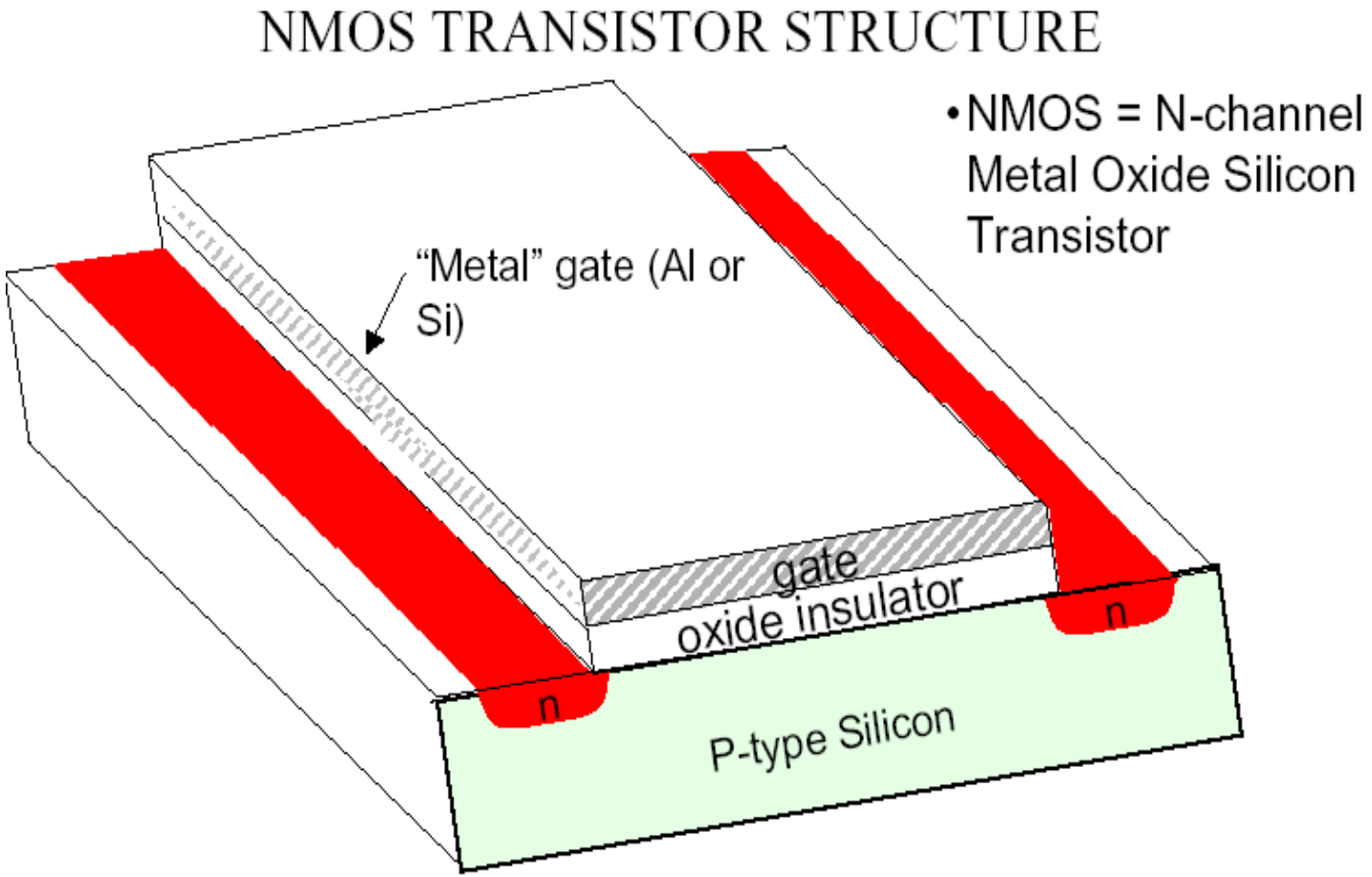




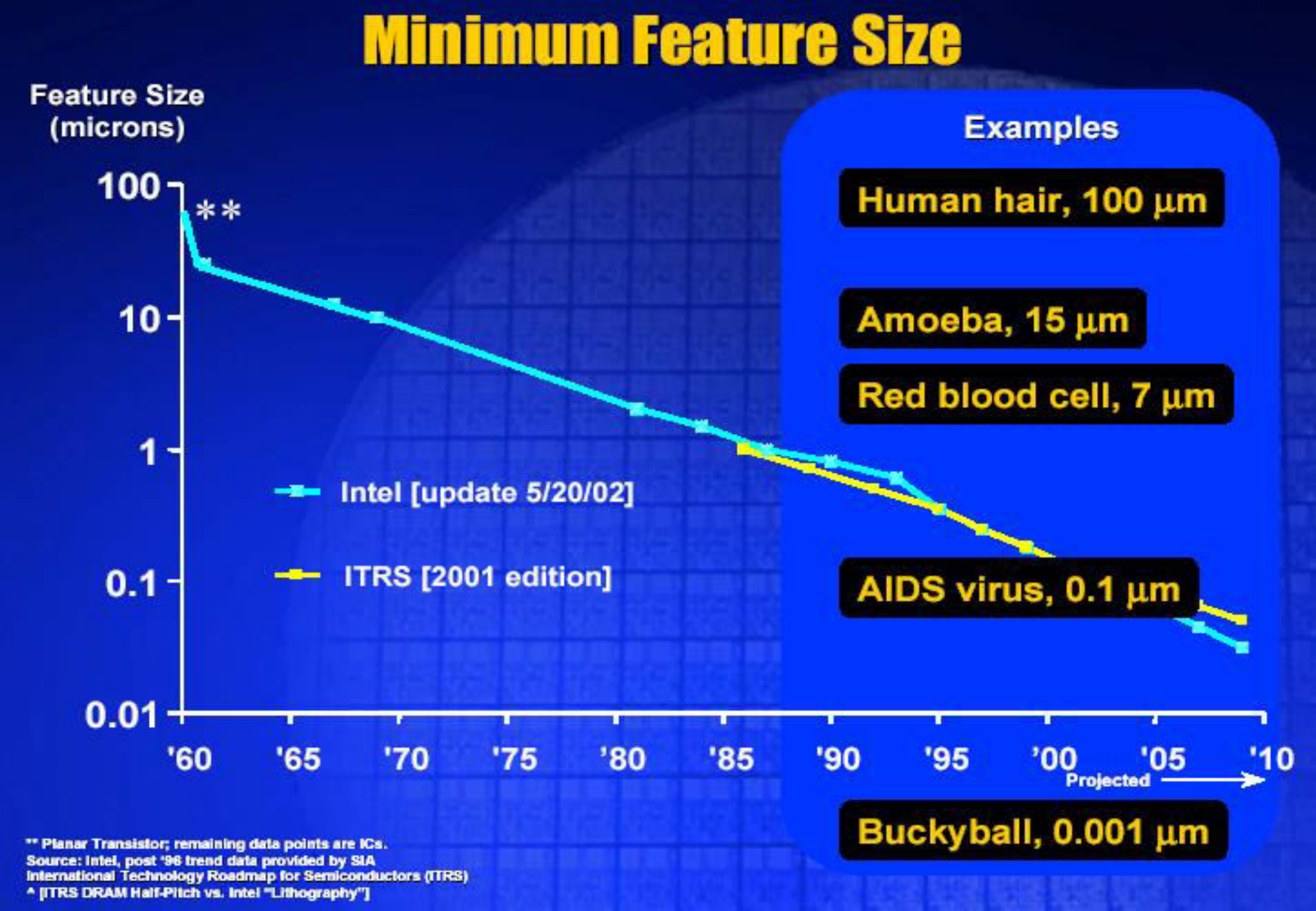
The Computer Revolution

- Progress in computer technology
  - Underpinned by Moore's Law
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive

Line Width/Feature Size



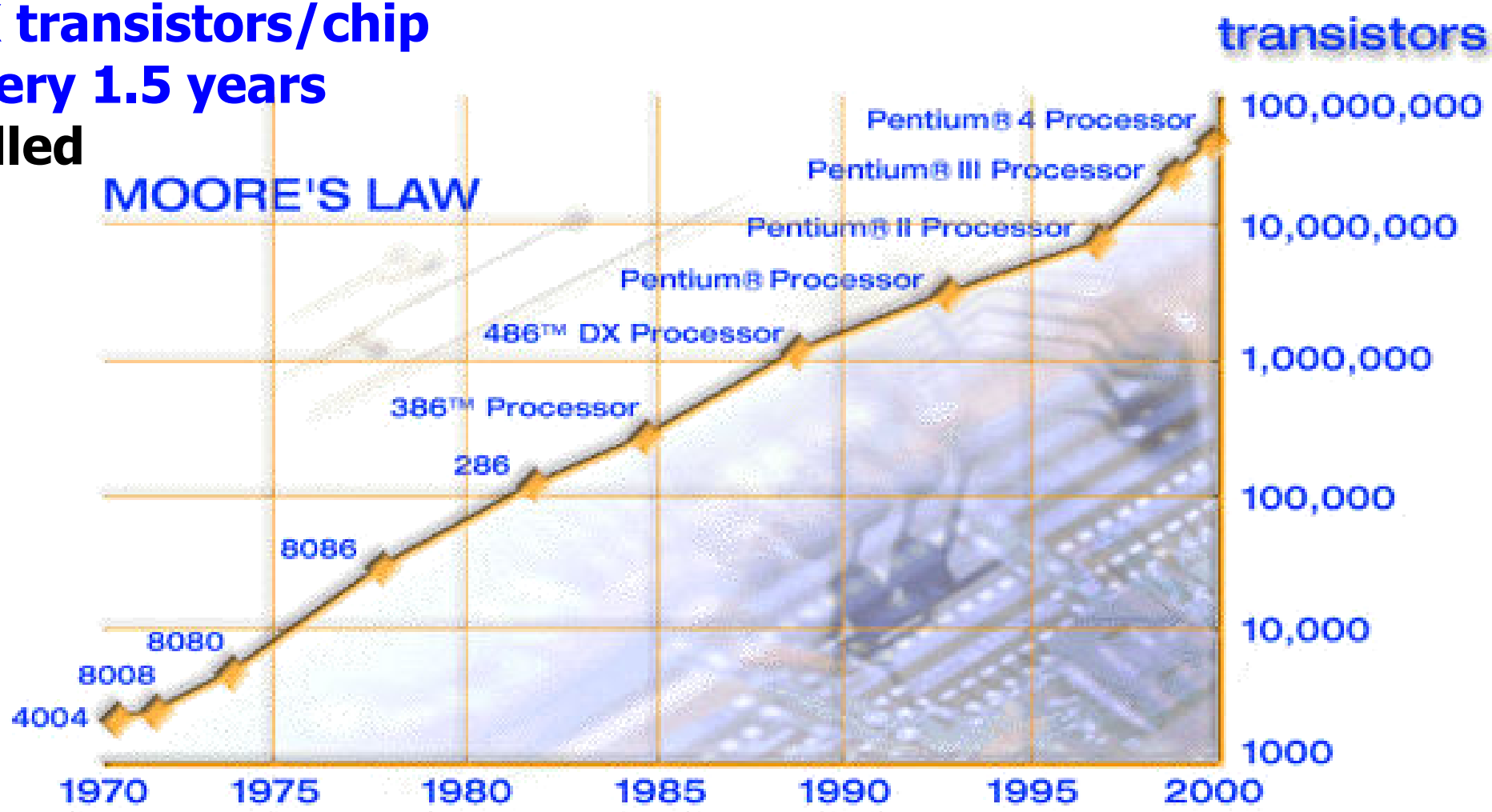
Minimum Feature Size



** Planar Transistor; remaining data points are ICs.
Source: Intel, post '96 trend data provided by SIA
International Technology Roadmap for Semiconductors (ITRS)
^ [ITRS DRAM Half-Pitch vs. Intel "Lithography"]

Technology Trends: Microprocessor Capacity

2X transistors/chip every 1.5 years called



Classes of Computers

- Desktop 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
- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

Computer Progress Supported/Driven by Market and Usage

- Applications drive machine "balance"
  - Numerical simulations: floating-point, memory BW
  - Transaction processing: I/O, INT performance
  - Media processing: low-precision 'pixel' arithmetic
- Applications drive machine performance
  - What if my computer runs all my software very fast?
  - Programs use increasing amount of memory:
    - Double per 1.5-2 year, or 0.5-1 addressing bit per year
  - High-level programming languages replace assembly languages => compilers important
    - Compiler and architecture work together
- Effects of compatibility and ease of use
- Effects of market demands and market share
  - Can investment in R&D, production be paid off?

Computer Usage: General Purpose (PC and Server)

- **Uses: commercial (int.), scientific (FP, graphics), home (int., audio, video, graphics)**
  - **Software compatibility is the most important factor**
  - **Short product life; higher price and profit margin**
  - **OS issue: OS serves another interface above arch.**
    - **Effects of OS developments on architecture**
    - **RISC-based Unix workstation vs x86-based PC: (1) units sold is only 1% of PC's, (2) emphasize more on performance than on price**
- **Future:**
  - **Use increased transistors for performance, human interface (multimedia), bandwidth, monitoring**

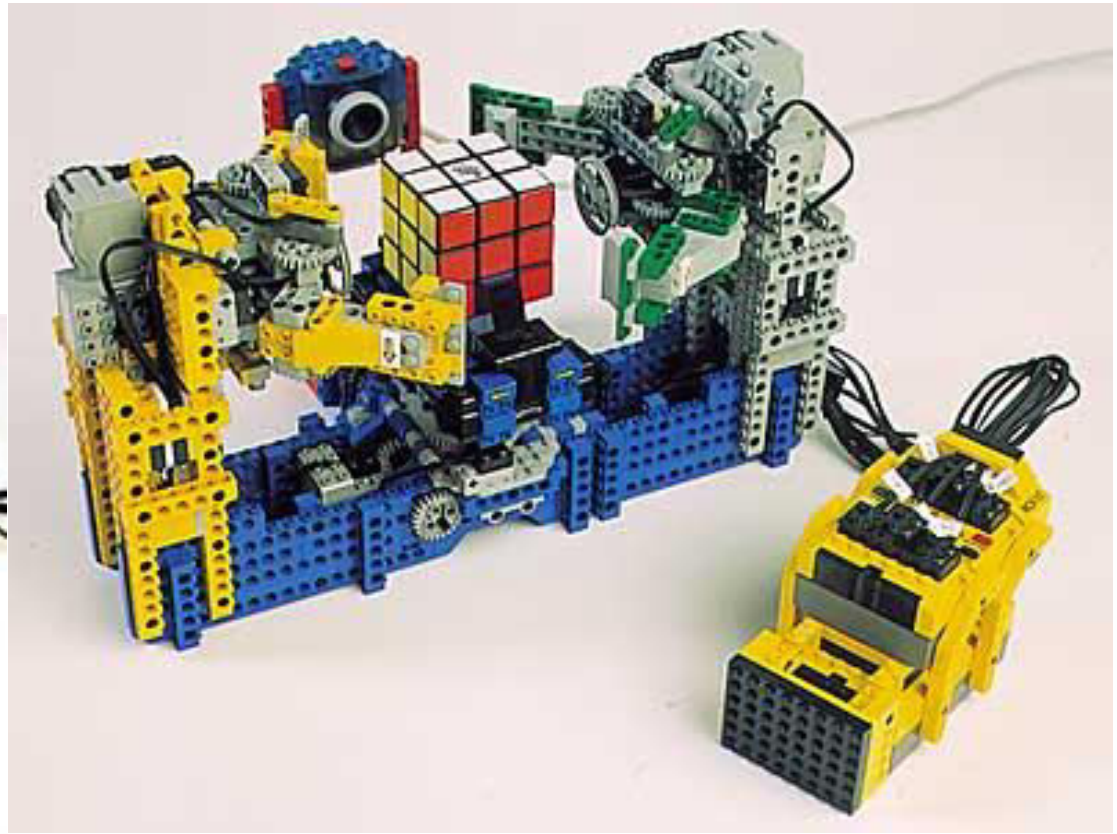
Computer Usage: Embedded

- **A computer inside another device used for running one predetermined application**
  , printer, disk); consumer
  ame, CD player, PDA); cell

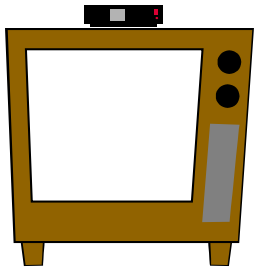
Lego Mindstorms

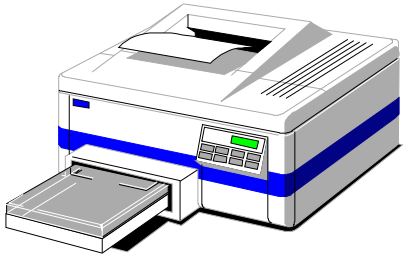
Robotic command explorer:
A "Programmable Brick",
Hitachi H8 CPU (8-bit), 32KB RAM,
LCD, batteries,
infrared transmitter/receiver,
4 control buttons, 6 connectors

它可以做什麼?

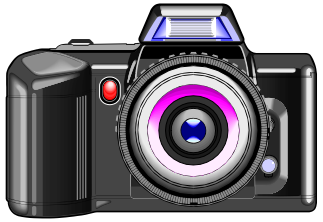


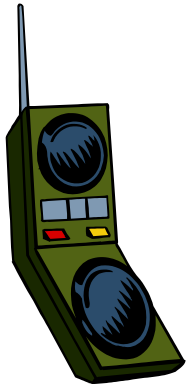
生活裡的應用比比皆是






Bluetooth
IrDA
Accelerometer
B/W LCD
(320x240dots)
Vibrator
Touch panel
Terminals
Microphone
Stem switch
Speaker
Fingerprint sensor
image source: IBM

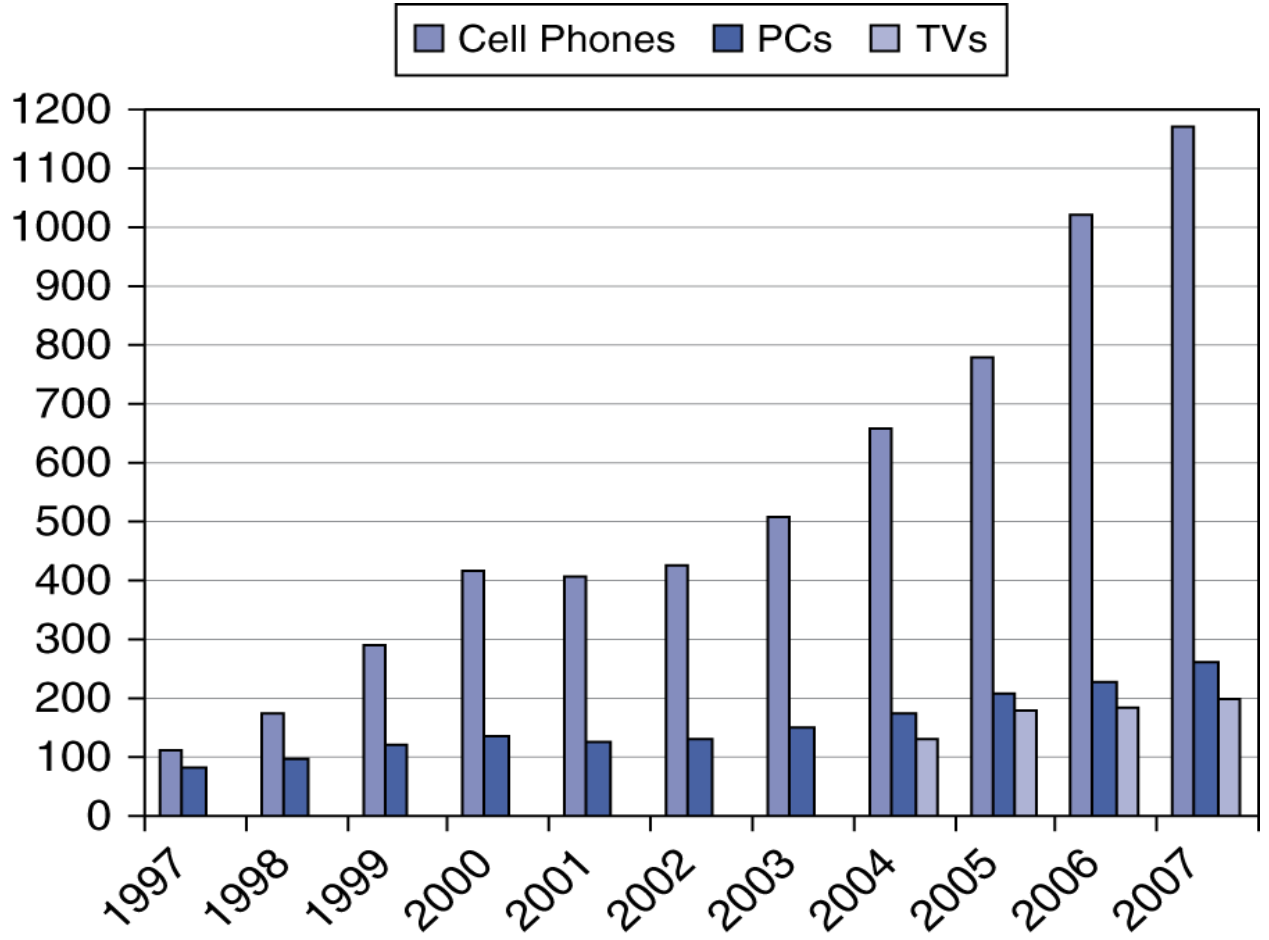




Embedded Computers

- Typically w/o FP or MMU, but integrating various peripheral functions, e.g., DSP
  - Large variety in ISA, performance, on-chip peripherals
  - Compatibility is non-issue, new ISA easy to enter, low power become important
- More architecture and survive longer:
  4- or 8-bit microprocessor still in use
  (8-bit for cost-sensitive, 32-bit for performance)
- Large volume sale (billions) at low price ($40-$5)
- Use of microprocessor:
  - 1995 #1: x86; #2: 6800; #3: Hitachi SuperH (Sega)
  - 2002 #1: ARM #2: x86; #3: Motorola 6800
- Trend: lower cost, more functionality
  - system-on-chip, μP core on ASIC

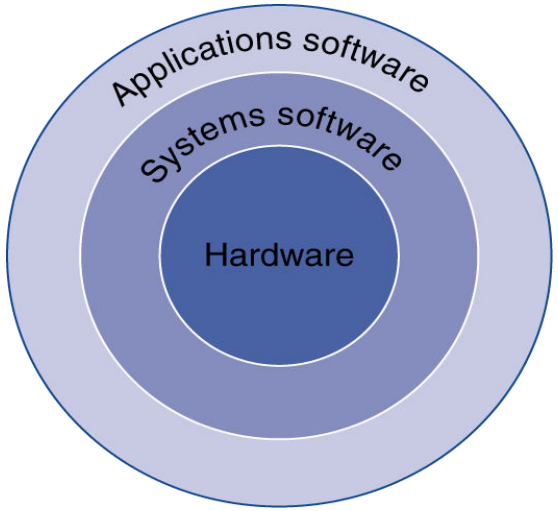
The Processor Market



Outline

- Computer: A historical perspective
- Abstractions
- Technology
  - Performance
    - Definition
    - CPU performance
  - Power trends: multi-processing
  - Measuring and evaluating performance
  - Cost

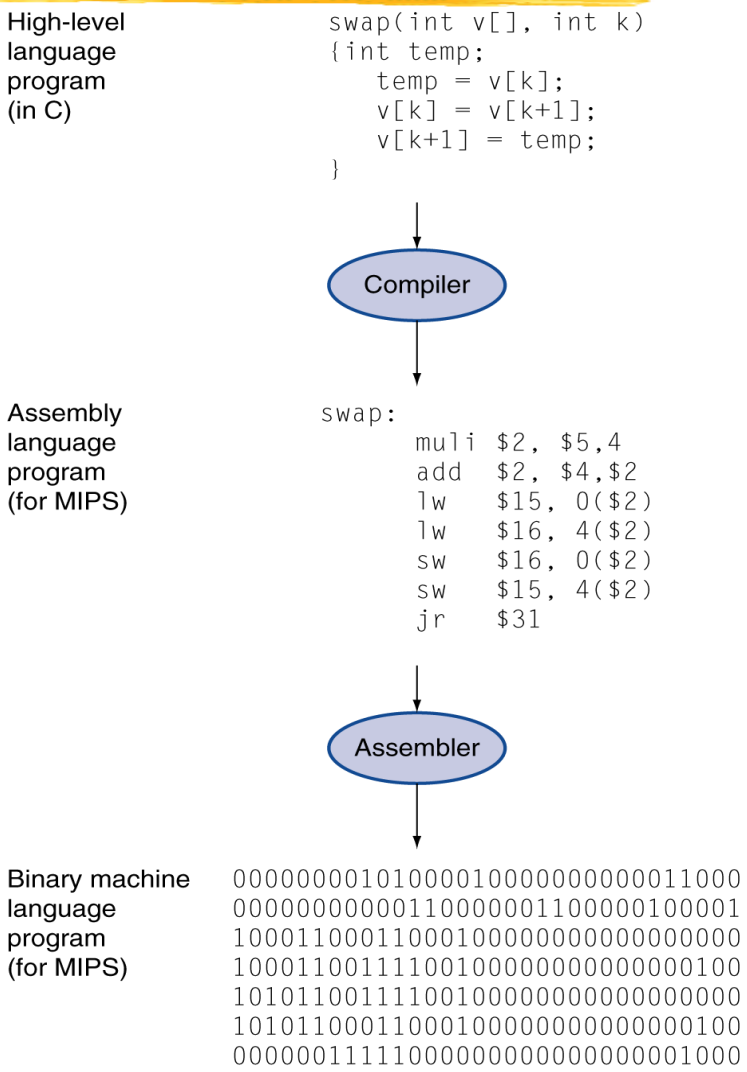
Below Your Program



- **Application software**
  - **Written in high-level language**
- **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**

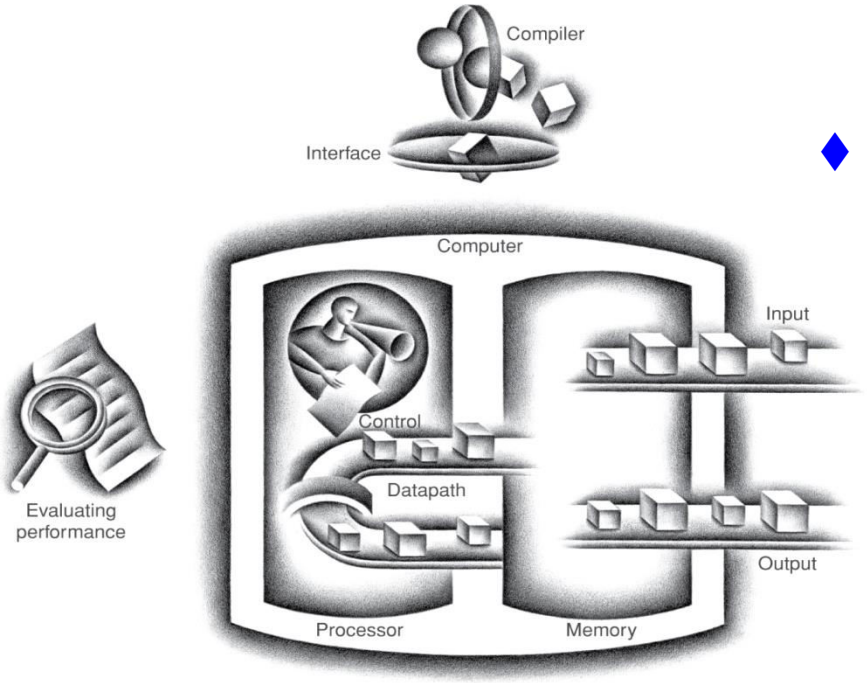
Levels of Program Code

- **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**



Components of a Computer

The BIG Picture

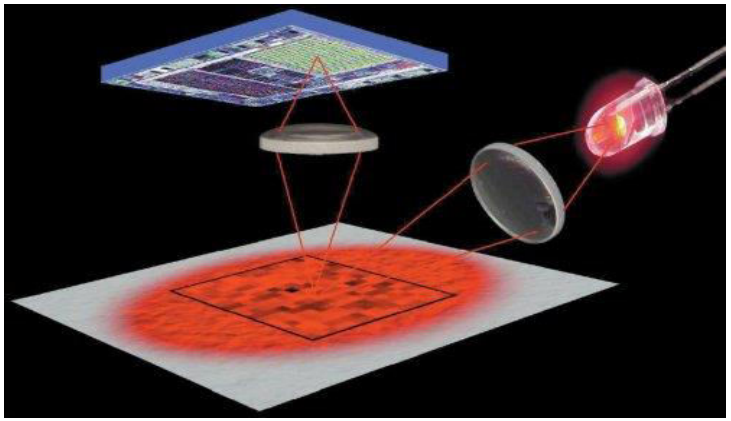


- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

Anatomy of a Computer

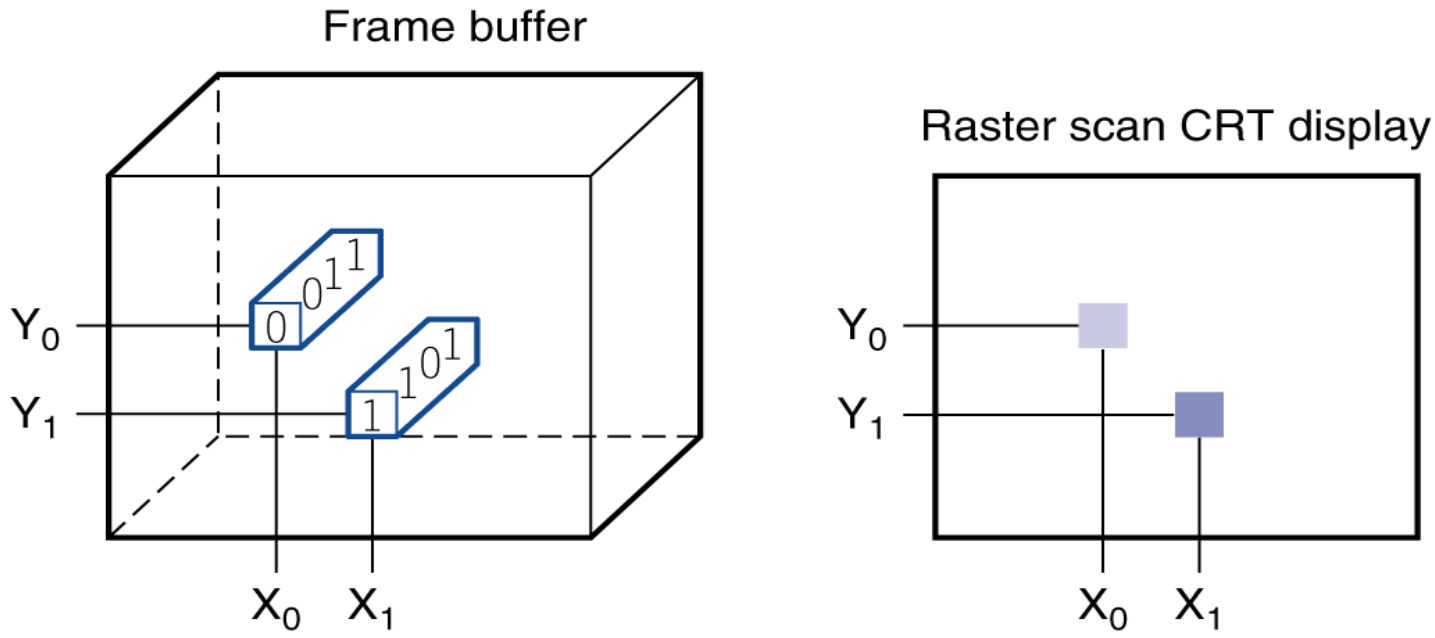
Anatomy of a Mouse

- **Optical mouse**
  - **LED illuminates desktop**
  - **Small low-res camera**
  - **Basic image processor**
    - **Looks for x, y movement**
  - **Buttons & wheel**
- **Supersedes roller-ball mechanical mouse**

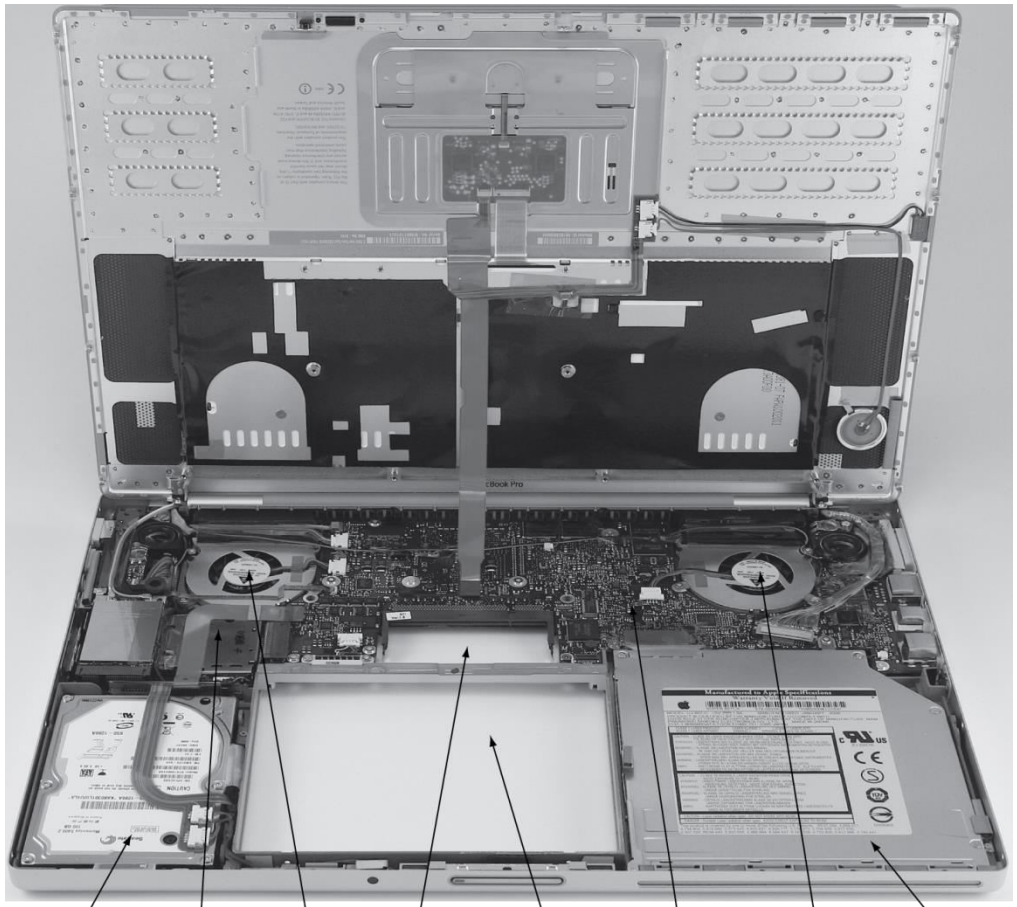


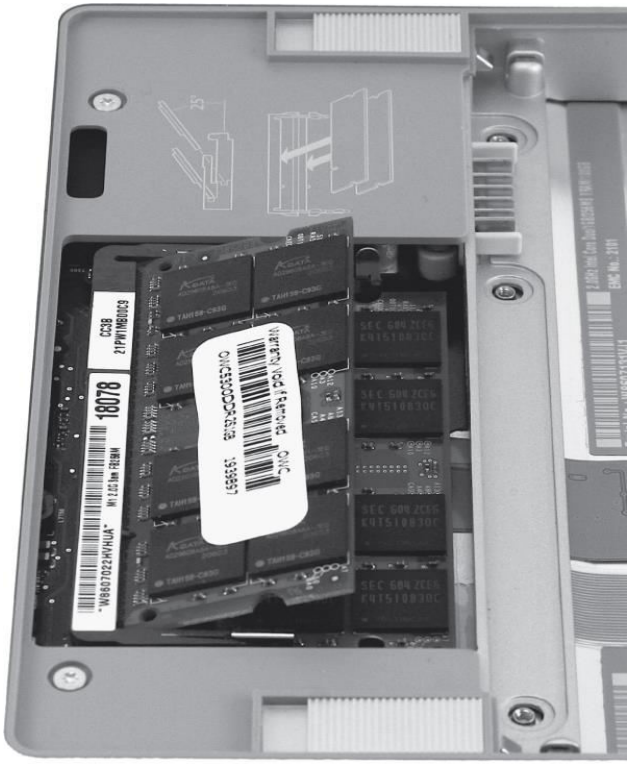
Through the Looking Glass

- **LCD screen: picture elements (pixels)**
  - **Mirrors content of frame buffer memory**
  - **Bit map: a matrix of pixels**
  - **Resolution in 2008: 640 x 480 to 2560 x 1600 pixels**



Opening the Box



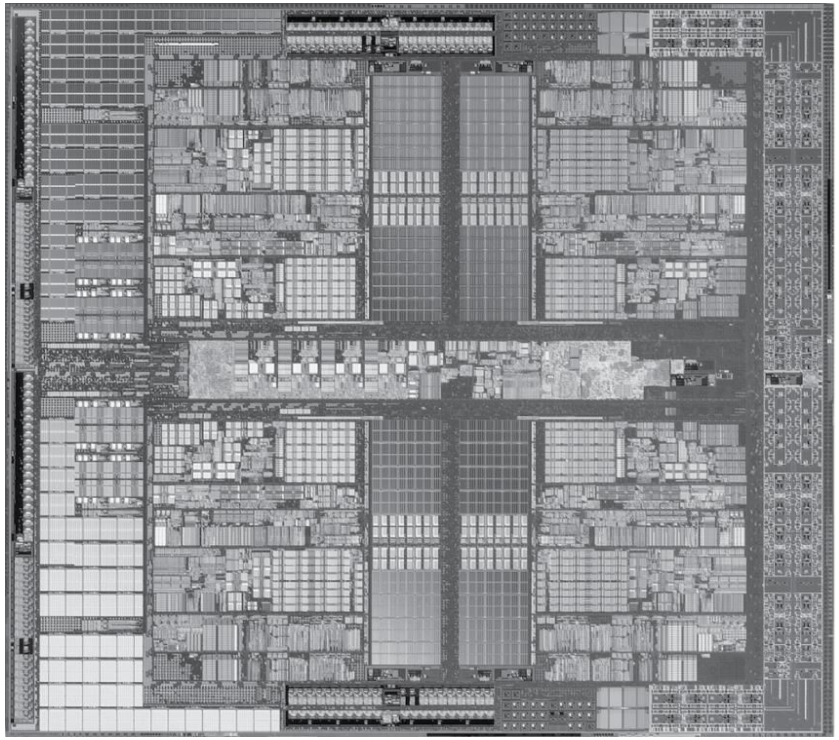


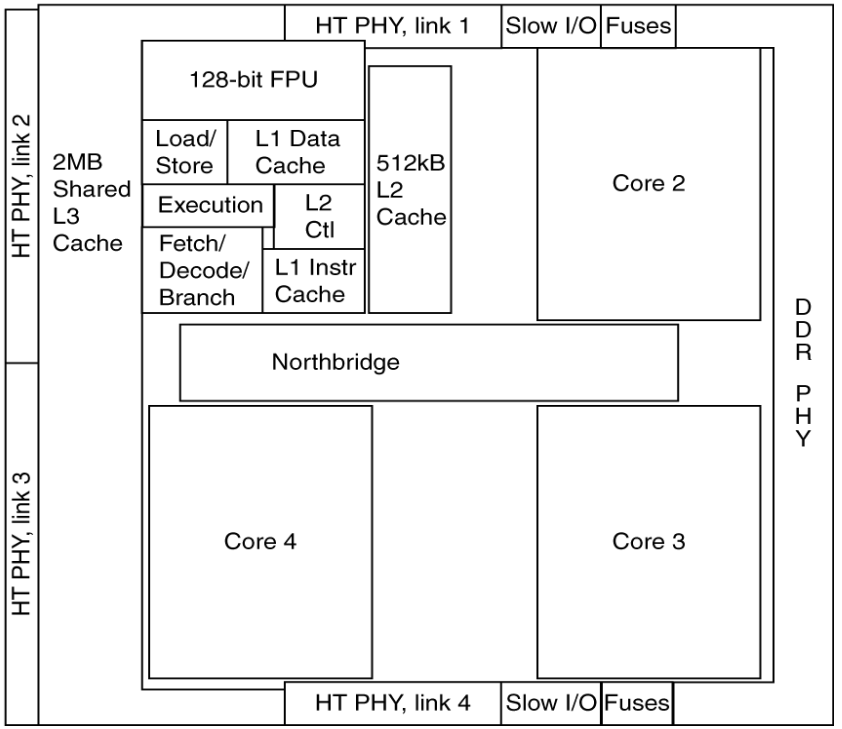
Inside the Processor (CPU)

- **Datapath: performs operations on data**
- **Control: sequences datapath, memory, ...**
- **Cache memory**
  - **Small fast SRAM memory for immediate access to data**

Inside the Processor

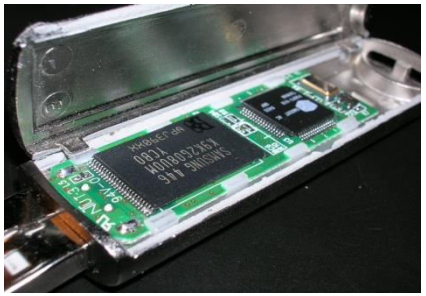
- **AMD Barcelona: 4 processor cores**





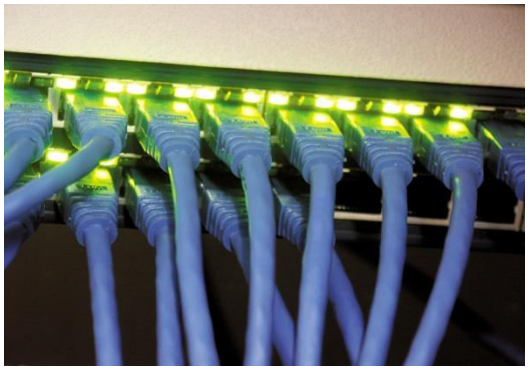
A Safe Place for Data

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



Networks

- Communication and resource sharing
- Local area network (LAN): Ethernet
  - Within a building
- Wide area network (WAN): the Internet
- Wireless network: WiFi, Bluetooth



Abstractions

The BIG Picture

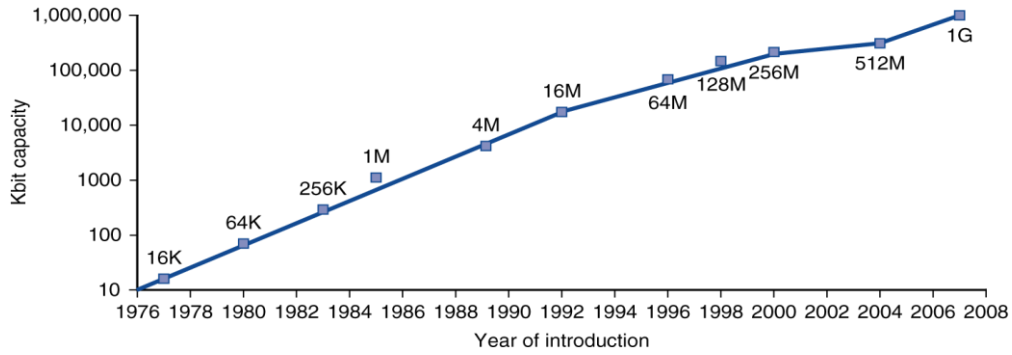
- Abstraction helps us deal with complexity
  - Hide lower-level detail
- Instruction set architecture (ISA)
  - The hardware/software interface
- Application binary interface
  - The ISA plus system software interface
- Implementation
  - The details underlying and interface

Outline

- Computer: A historical perspective
- Abstractions
- Technology
  - Performance
    - Definition
    - CPU performance
  - Power trends: multi-processing
  - Measuring and evaluating performance
  - Cost

Technology Trends

- **Electronics technology continues to evolve**
  - **Increased capacity and performance**
  - **Reduced cost**



DRAM capacity

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
2005	Ultra large scale IC	6,200,000,000

那一架飛機的效能比較好？

Concorde:

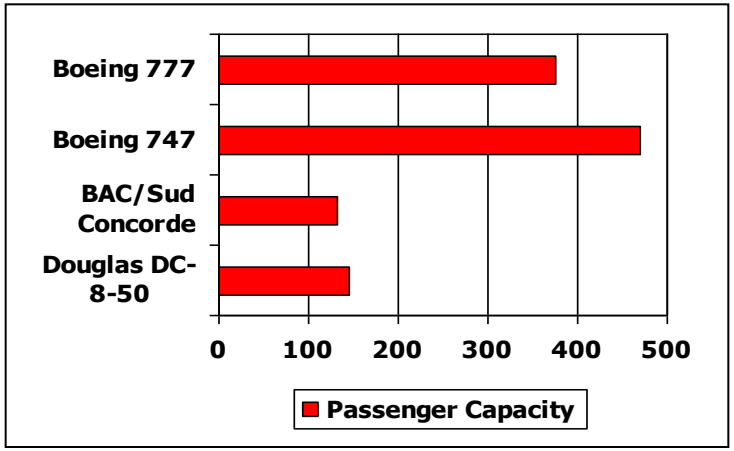
- Capacity: 132 persons
- Range: 4000 miles
- Cruising speed: 1350 mph

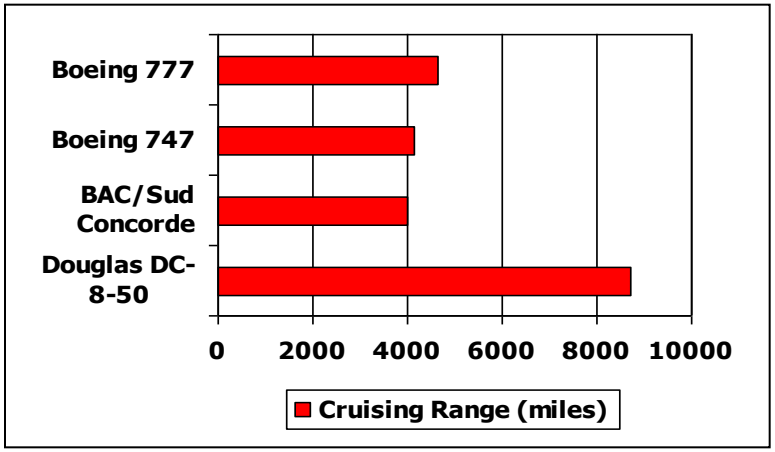
747-400:

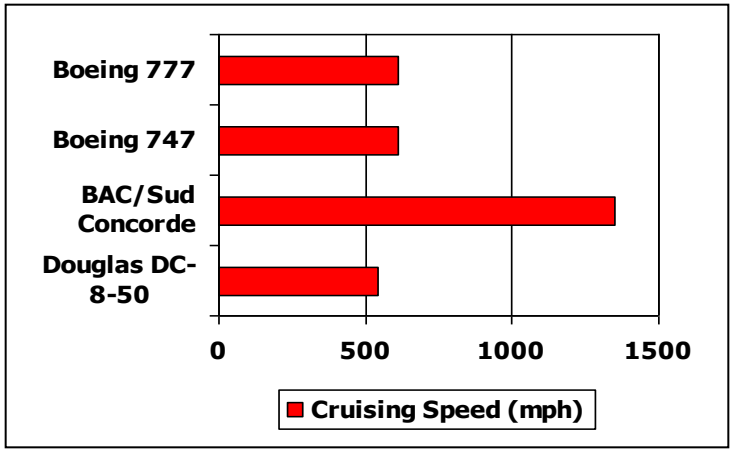
- Capacity: 470 persons
- Range: 4150 miles
- Cruising speed: 610 mph

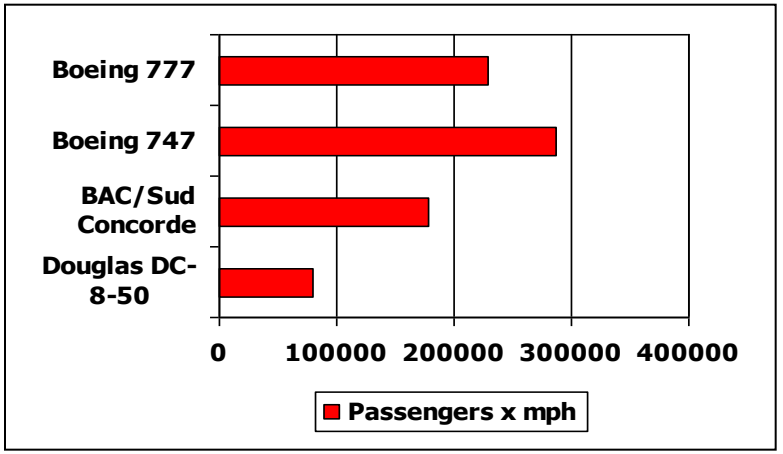
Defining Performance

- **Which airplane has the best performance?**









Response Time and Throughput

- **Response time**
  - **How long it takes to do a task**
- **Throughput**
  - **Total work done per unit time**
    - **e.g., tasks/transactions/… per hour**
- **How are response time and throughput affected by**
  - **Replacing the processor with a faster version?**
  - **Adding more processors?**
- **We'll focus on response time for now…**

Measuring Execution Time

- **Elapsed time**
  - **Total response time, including all aspects**
    - **Processing, I/O, OS overhead, idle time**
  - **Determines system performance**
- **CPU time**
  - **Time spent processing a given job**
    - **Discounts I/O time, other jobs' shares**
  - **Comprises user CPU time and system CPU time**
- **Different programs are affected differently by CPU and system performance**

Relative Performance

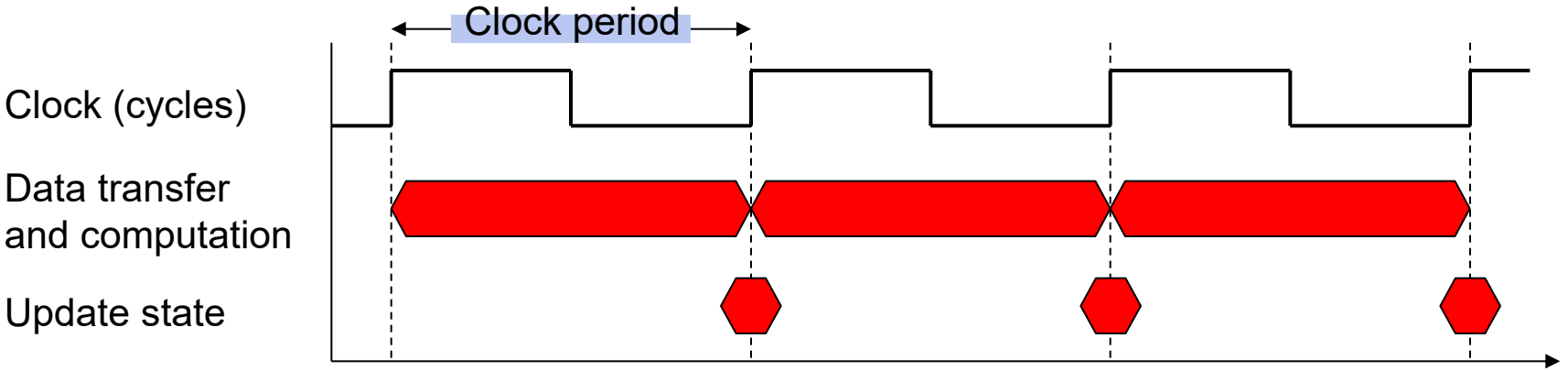
- **Define Performance = 1/Execution Time**
- **"X is n time faster than Y"**

$$\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$$

- **Example: time taken to run a program**
  - **10s on A, 15s on B**
  - **Execution Time$_B$ / Execution Time$_A$ = 15s / 10s = 1.5**
  - **So A is 1.5 times faster than B**

CPU Clocking

- **Operation of digital hardware governed by a constant-rate clock**



  - **Clock period: duration of a clock cycle**
    - **e.g., 250ps = 0.25ns = 250×10^{-12}s**
  - **Clock frequency (rate): cycles per second**
    - **e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz**

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- **Performance improved by**
  - **Reducing number of clock cycles**
  - **Increasing clock rate**
  - **Hardware designer must often trade off clock rate against cycle count**

CPU Time Example

- **Computer A: 2GHz clock, 10s CPU time**
- **Designing Computer B**
  - **Aim for 6s CPU time**
  - **Can do faster clock, but causes 1.2 × clock cycles**
- **How fast must Computer B clock be?**

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruct. Count} \times \text{Cycles per Instruct.}$$

$$\text{CPU Time} = \text{Instruct. Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruct. Count} \times \text{CPI}}{\text{Clock Rate}}$$

- **CPI : Clock Per Instruction**
- **Instruction Count for a program**
  - **Determined by program, ISA and compiler**
- **Average cycles per instruction**
  - **Determined by CPU hardware**
  - **If different instructions have different CPI**
    - **Average CPI affected by instruction mix**

CPI Example

- **Computer A: Cycle Time = 250ps, CPI = 2.0**
- **Computer B: Cycle Time = 500ps, CPI = 1.2**
- **Same ISA**
- **Which is faster, and by how much?**

$$\text{CPU Time}_A = \text{Instruct. Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster…

$$\text{CPU Time}_B = \text{Instruct. Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

…by this much

CPI in More Detail

- **If different instruction classes take different numbers of cycles**

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruct. Count}_i)$$

- **Weighted average CPI**

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruct. Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruct. Count}_i}{\text{Instruct. Count}}}_{\text{Relative frequency}} \right)$$

CPI Example

- **Alternative compiled code sequences using instructions in classes A, B, C**

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- **Sequence 1: IC = 5**
  - **Clock Cycles = 2×1 + 1×2 + 2×3 = 10**
  - **Avg. CPI = 10/5 = 2.0**
- **Sequence 2: IC = 6**
  - **Clock Cycles = 4×1 + 1×2 + 1×3 = 9**
  - **Avg. CPI = 9/6 = 1.5**

Performance Summary

The BIG Picture

$$\text{CPU Time} = \frac{\text{Instruct.}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruct.}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

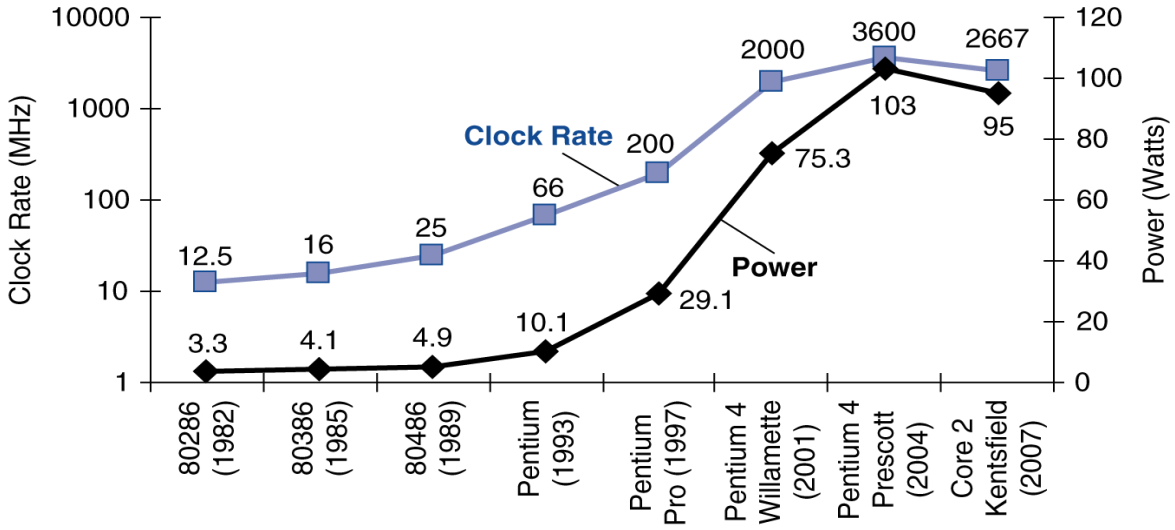
- **Performance depends on**

	Instruction Count	CPI	Clock Rate
Program			
Compiler			
Instruction Set			
Organization			
Technology			

Outline

- Computer: A historical perspective
- Abstractions
- Technology
  - Performance
    - Definition
    - CPU performance
  - Power trends: multi-processing
  - Measuring and evaluating performance
  - Cost

Power Trends



- **In CMOS IC technology**

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

×30 | 5V → 1V | ×1000

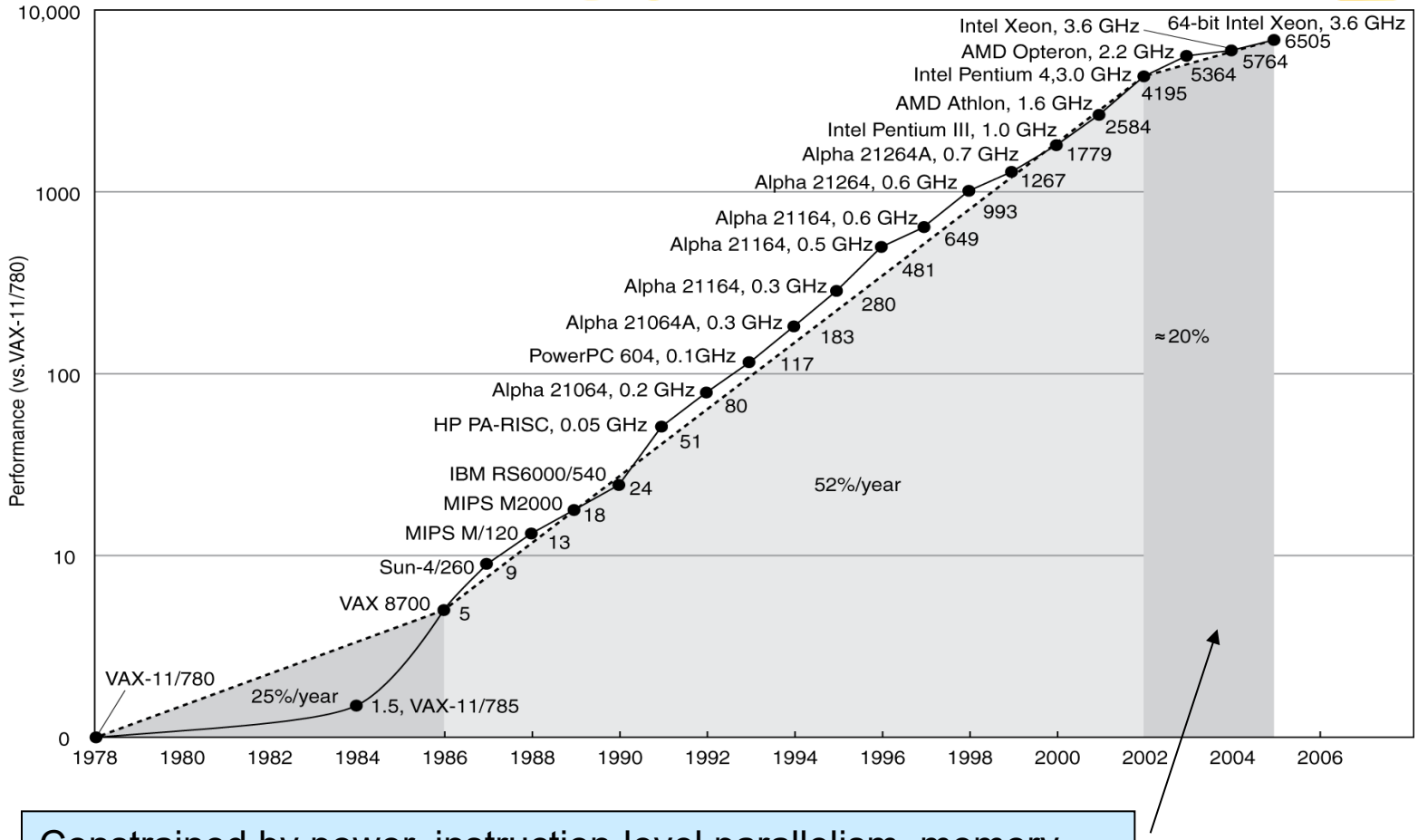
Reducing Power

- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

- The power wall
  - We can't reduce voltage further
  - We can't remove more heat
- How else can we improve performance?

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

§1.6 The Sea Change: The Switch to Multiprocessors

Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

Outline

- Computer: A historical perspective
- Abstractions
- Technology
  - Performance
    - Definition
    - CPU performance
  - Power trends: multi-processing
  - Measuring and evaluating performance
  - Cost

What Programs for Comparison?

- **What's wrong with this program as a workload?**

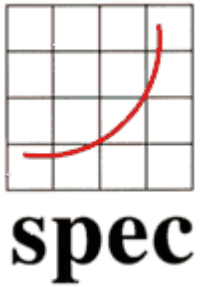
```
integer A[][], B[][], C[][];
for (I=0; I<100; I++)
  for (J=0; J<100; J++)
    for (K=0; K<100; K++)
      C[I][J] = C[I][J] + A[I][K]*B[K][J];
```

- **What measured? Not measured? What is it good for?**
- **Ideally run typical programs with typical input before purchase, or before even build machine**
  - **Called a "workload"; For example:**
  - **Engineer uses compiler, spreadsheet**
  - **Author uses word processor, drawing program, compression software**

Benchmarks

- Obviously, apparent speed of processor depends on code used to test it
- Need industry standards so that different processors can be fairly compared => *benchmark programs*
- Companies exist that create these benchmarks: "typical" code used to evaluate systems
- Tricks in benchmarking:
  - different system configurations
  - compiler and libraries optimized (perhaps manually) for benchmarks
  - test specification biased towards one machine
  - very small benchmarks used
- Need to be changed every 2 or 3 years since designers could target these standard benchmarks

Example Standardized Workload Benchmarks



- Standard Performance Evaluation Corporation (SPEC) : supported by a number of computer vendors to create standard set of benchmarks
- Began in 1989 focusing on benchmarking workstation and servers using CPU-intensive benchmarks
- The latest release: SPEC2006 benchmarks
  - CPU performance (CINT 2006, CFP 2006)
  - High-performance computing
  - Client-sever models
  - Mail systems
  - File systems
  - Web-servers …

SPEC CPU Benchmark

- **SPEC CPU2006**
  - **Elapsed time to execute a selection of programs**
    - **Negligible I/O, so focuses on CPU performance**
  - **Normalize relative to reference machine**
  - **Summarize as geometric mean of performance ratios**
    - **CINT2006 (integer)**

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

CINT2006 for Opteron X4 2356

Name	Description	IC×10^9	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.40	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.40	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.47	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.40	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.40	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.40	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.48	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.40	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.40	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.40	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.40	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.40	1,143	6,900	6.0
Geometric mean							11.7

High cache miss rates

SPEC Power Benchmark

- **Power consumption of server at different workload levels (10% increase each run, average them)**
  - **Performance: ssj_ops/sec**
  - **Power: Watts (Joules/sec)**

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i\right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i\right)$$

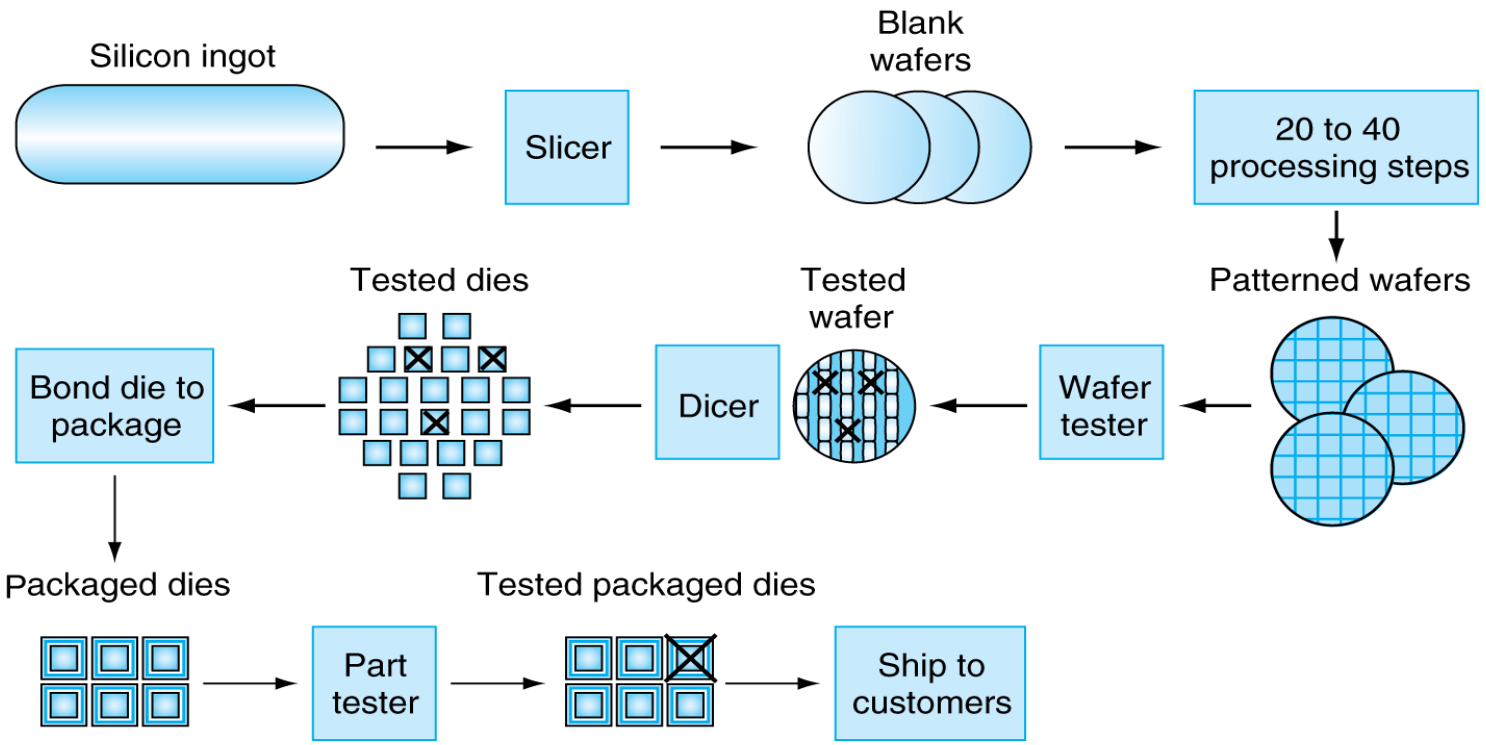
SPECpower_ssj2008 for X4

Target Load %	Performance (ssj_ops/sec)	Average Power (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
Σssj_ops/ Σpower		493

Outline

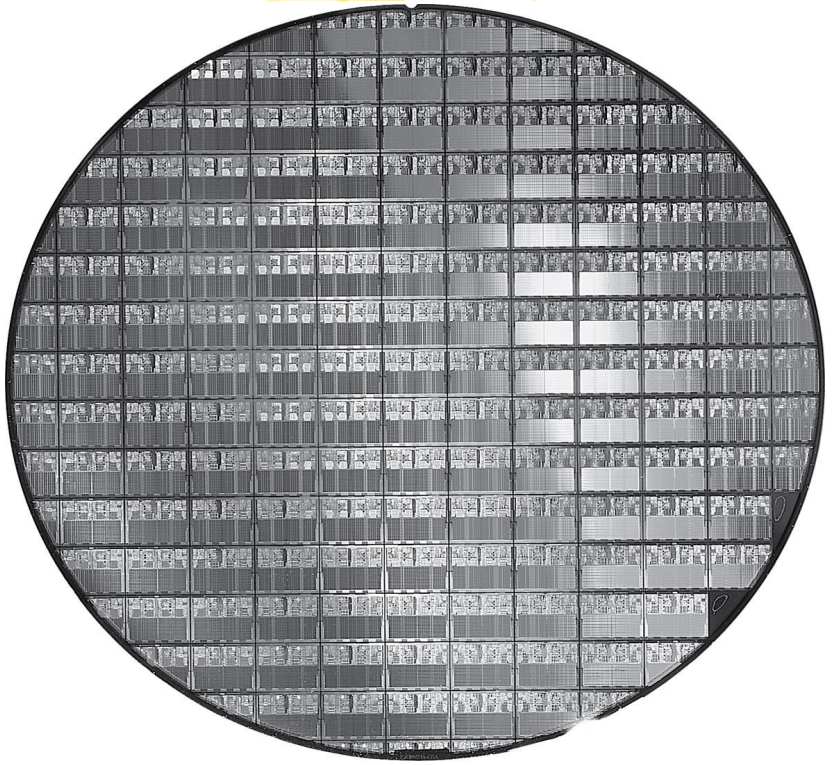
- Computer: A historical perspective
- Abstractions
- Technology
  - Performance
    - Definition
    - CPU performance
  - Power trends: multi-processing
  - Measuring and evaluating performance
  - Cost

Manufacturing ICs



- **Yield: proportion of working dies per wafer**

AMD Opteron X2 Wafer



- X2: 300mm wafer, 117 chips, 90nm technology
- X4: 45nm technology

Integrated Circuit Cost

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \text{Wafer area}/\text{Die area}$$

$$\text{Yield} = \frac{\text{\# of good dies}}{\text{\# of total dies}} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

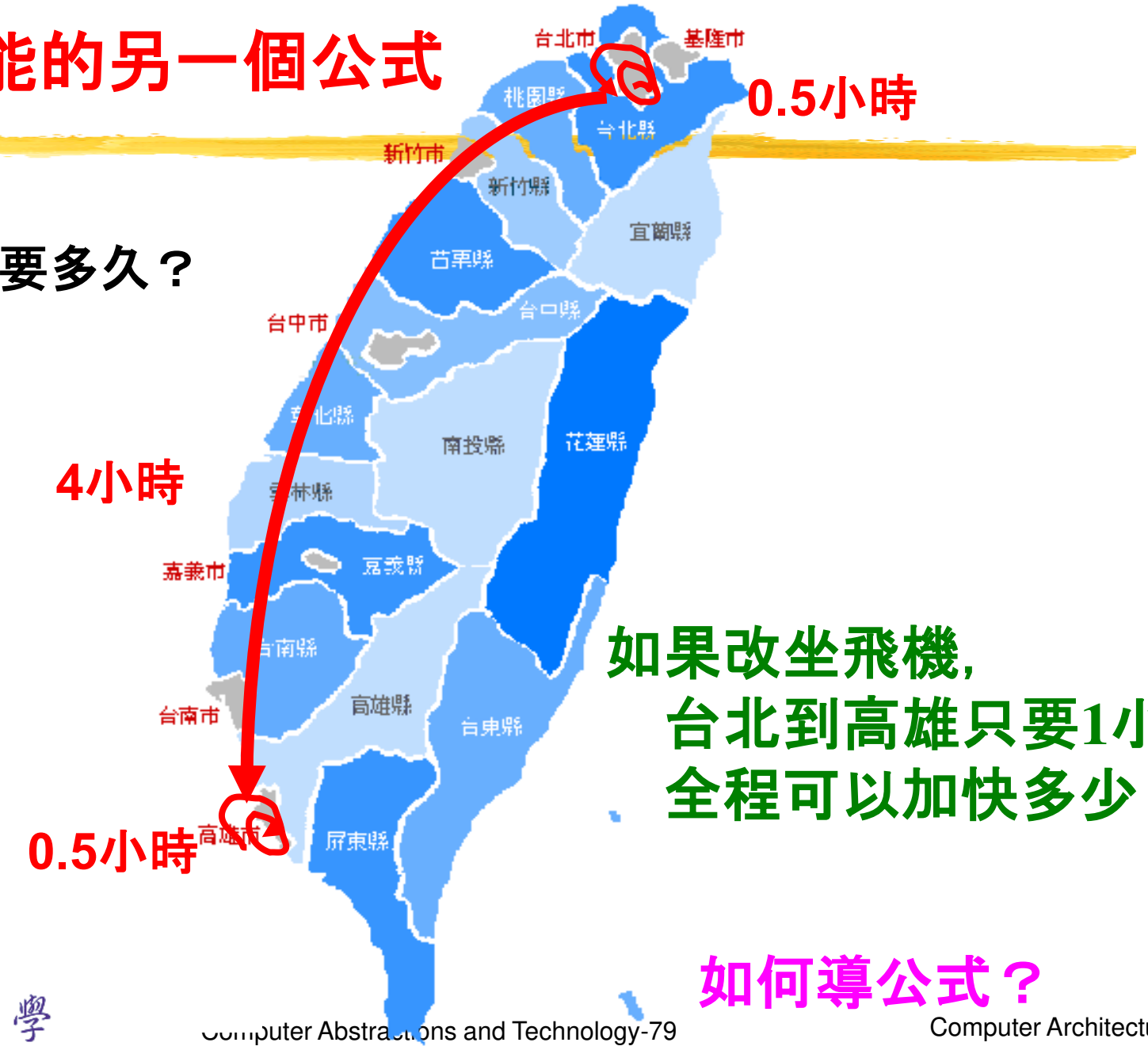
- **Nonlinear relation to area and defect rate**
  - **Wafer cost and area are fixed**
  - **Defect rate determined by manufacturing process**
  - **Die area determined by architecture and circuit design**

Cost of a Chip Includes ...

- **Die cost: affected by wafer cost, number of dies per wafer, and die yield (#good dies/#total dies)**
- **Testing cost**
- **Packaging cost: depends on pins, heat dissipation, ...**

有關效能的另一個公式

從台北到高雄要多久？



如果改坐飛機，
台北到高雄只要1小時
全程可以加快多少？

如何導公式？

由台北到高雄

- 不能enhance的部份為在市區的時間: 0.5 + 0.5 = 1小時
- 可以enhance的部份為在高速公路上的4小時
- 現在改用飛機, 可以enhance的部份縮短為1小時

- $$\text{speedup} = \frac{\text{走高速公路所需時間}}{\text{坐飛機所需時間}} = \frac{4 + 1}{1 + 1} = 2.5$$

Pitfall: Amdahl's Law

- **Improving an aspect of a computer and expecting a proportional improvement in overall performance**

$$T_{improved} = \frac{T_{affected}}{improvement\ factor} + T_{unaffected}$$

- **Example: multiply accounts for 80s/100s**
  - **How much improvement in multiply performance to get 5× overall?**

$$20 = \frac{80}{n} + 20$$

  - **Can't be done!**

- **Corollary: make the common case fast**

Fallacy: Low Power at Idle

- Look back at X4 power benchmark
  - At 100% load: 295W
  - At 50% load: 246W (83%)
  - At 10% load: 180W (61%)
- Google data center
  - Mostly operates at 10% – 50% load
  - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load

Pitfall: MIPS as a Performance Metric

- **MIPS: Millions of Instructions Per Second**
  - **Doesn't account for**
    - **Differences in ISAs between computers**
    - **Differences in complexity between instructions**

$$\text{MIPS} = \frac{\text{Instruct. count}}{\text{Execution time} \times 10^6}$$

$$= \frac{\text{Instruct. count}}{\frac{\text{Instruct. count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

  - **CPI varies between programs on a given CPU**

Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance