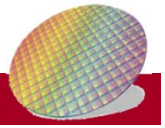
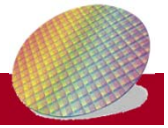
Chapter 1

Computer Abstractions and Technology



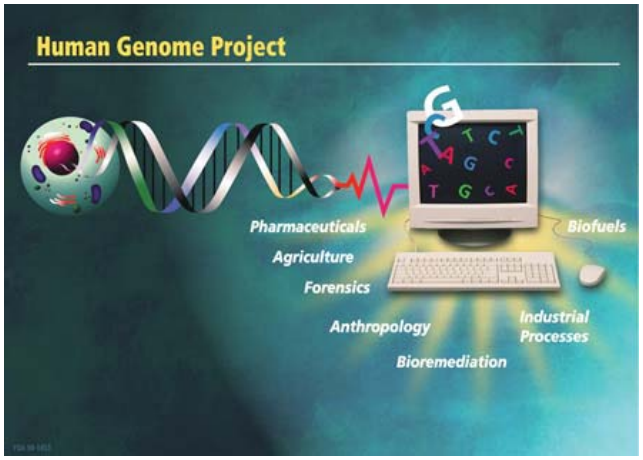
Course Administration

- Course Website:
- See NCKU Moodle Website
- Homework 1: Due: 3/7 11:55 PM
- Reading: Chapter 1



The Computer Revolution

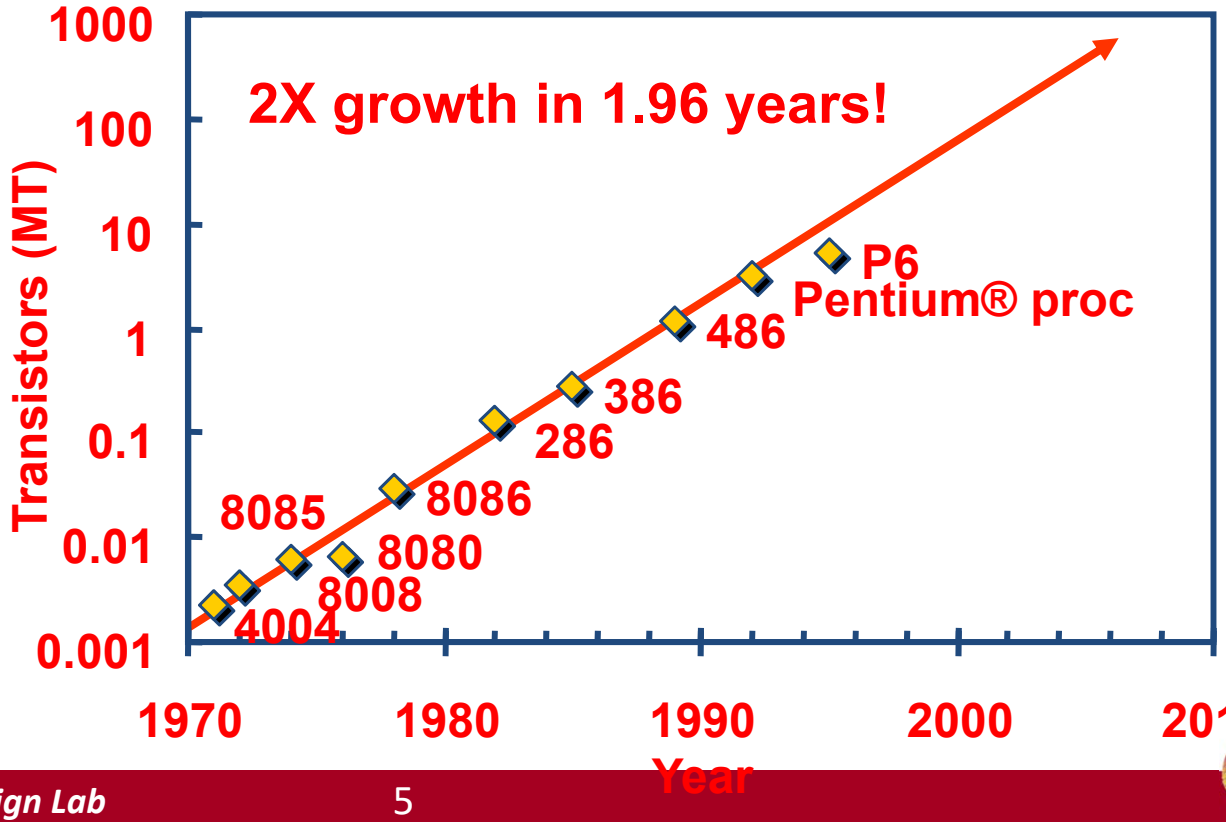
- Computers are almost everywhere:
  - PC, Mobile Phone, embedded system, etc
- Makes novel applications feasible
  - Internet
  - Search Engines
  - Vehicle Navigation
  - Human genome project



Moore's Law

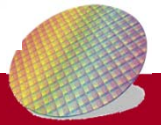
- In 1965, Gordon Moore predicted that the number of transistors that can be integrated on a die would double every 18 to 24 months (i.e., grow exponentially with time).

Gordon Moore



Classes of Computers

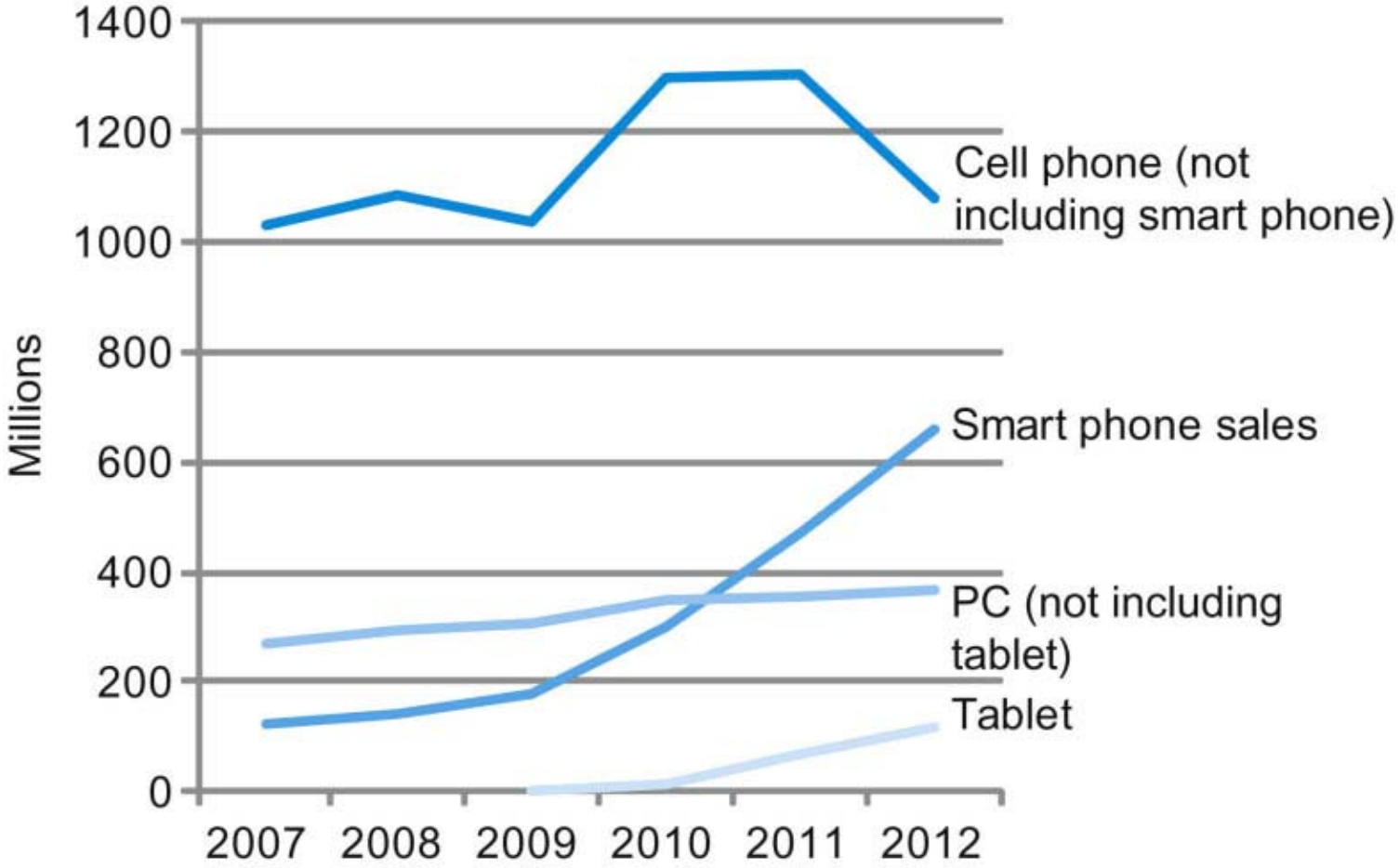
- Personal computers
  - General purpose, variety of software
- Server computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
  - Extreme High performance: Supercomputer
- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints



PostPC Era

- Personal mobile device (PMD)
  - Battery operated with wireless connectivity to the Internet
  - Likely have touch-sensitive screen
  - Normally do not have keyboard and mouse
  - Different appearance
- Cloud Computing
  - Companies can rent portions of servers so that they can provide software services

Device Growth

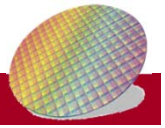


The fast growth of tablet and smart reflect the PostPC era.

2^X vs. 10^Y bytes ambiguity

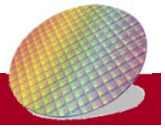
- 1KB used to have two meanings : 2^{10} Bytes or 10^2 bytes
  - Create confusion
  - Add a binary notation to remove confusions
  - Ie. Kilo vs. kibi
- Last column shows how much larger the binary term is than its corresponding decimal term.

Decimal term	Abbreviation	Value	Binary term	Abbreviation	Value	% Larger
kilobyte	KB	10^3	kibibyte	KiB	2^{10}	2%
megabyte	MB	10^6	mebibyte	MiB	2^{20}	5%
gigabyte	GB	10^9	gibibyte	GiB	2^{30}	7%
terabyte	TB	10^{12}	tebibyte	TiB	2^{40}	10%
petabyte	PB	10^{15}	pebibyte	PiB	2^{50}	13%
exabyte	EB	10^{18}	exbibyte	EiB	2^{60}	15%
zettabyte	ZB	10^{21}	zebibyte	ZiB	2^{70}	18%
yottabyte	YB	10^{24}	yobibyte	YiB	2^{80}	21%



What You Will Learn

- How programs are translated into the machine language
  - And how the hardware executes them
- The hardware/software interface
  - Instructions
- What determines program performance
  - How to compare the performance
- How hardware designers improve performance
- What are the reasons for the consequences of the recent switch from sequential processing to parallel processing?



Eight Great Ideas

Design for ***Moore's Law***

Use ***abstraction*** to simplify design

Make the ***common case fast***

Performance *via* ***parallelism***

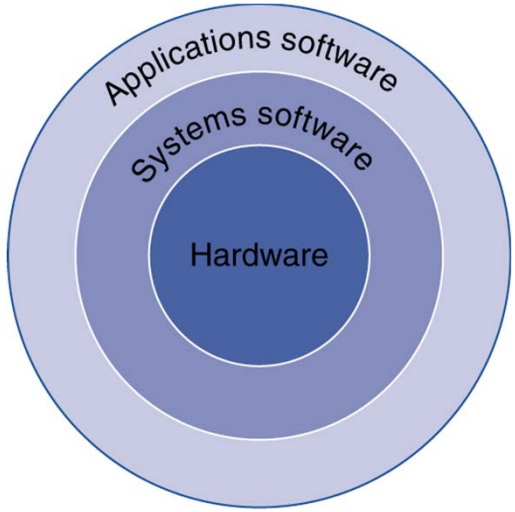
Performance *via* ***pipelining***

Performance *via* ***prediction***

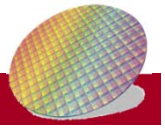
Hierarchy of memories

Dependability *via* redundancy

Below Your Program



- 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

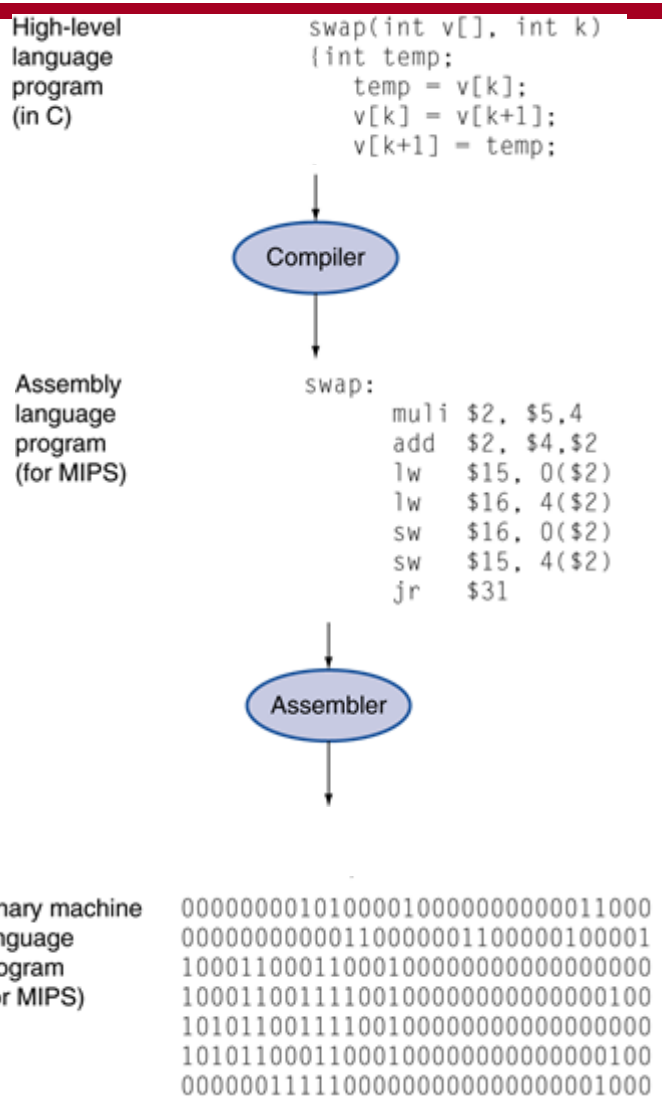
- Level of abstraction closer to problem domain
- Provides for productivity and portability

Assembly language

- Textual representation of instructions

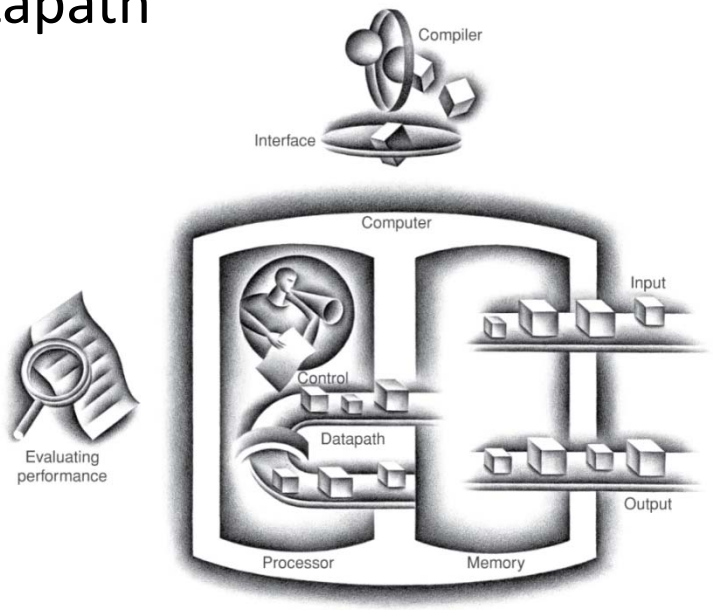
Hardware representation (Machine code)

- Binary digits (bits)
- Encoded instructions and data

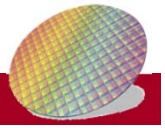


Components of a Computer

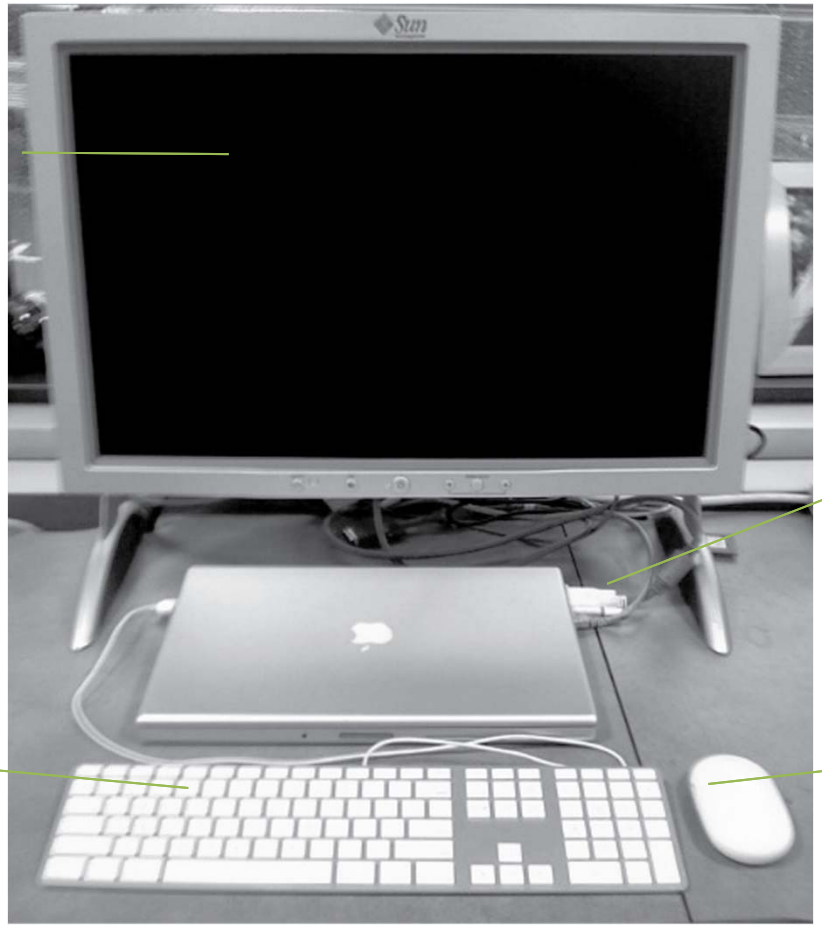
- Five components
  - Input
  - Output
  - Memory
  - Control
  - Datapath



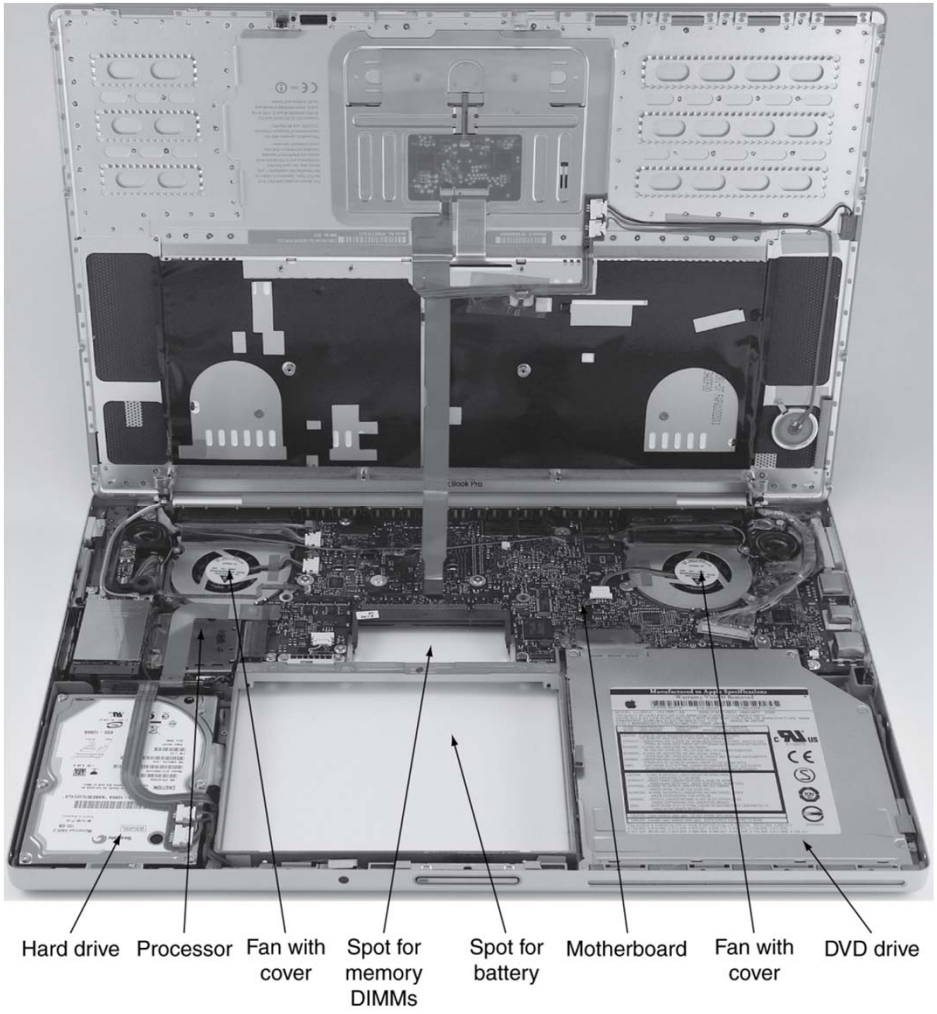
- Almost Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

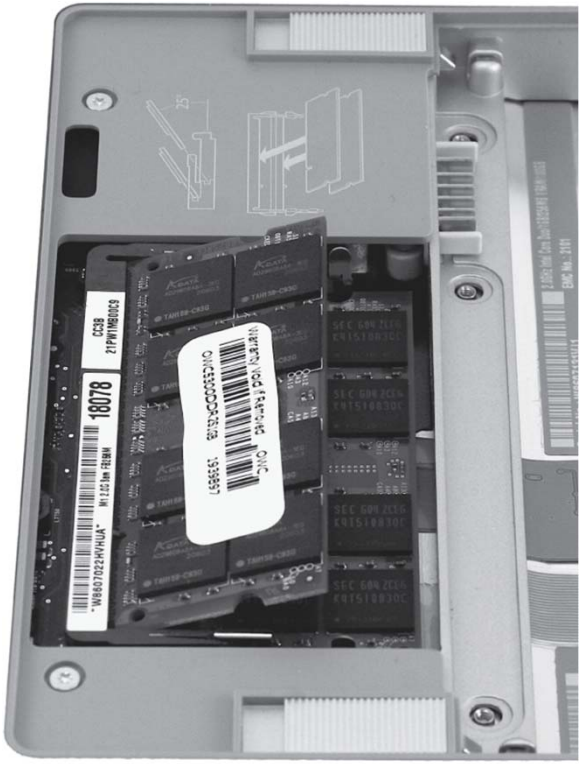


Anatomy of a Computer

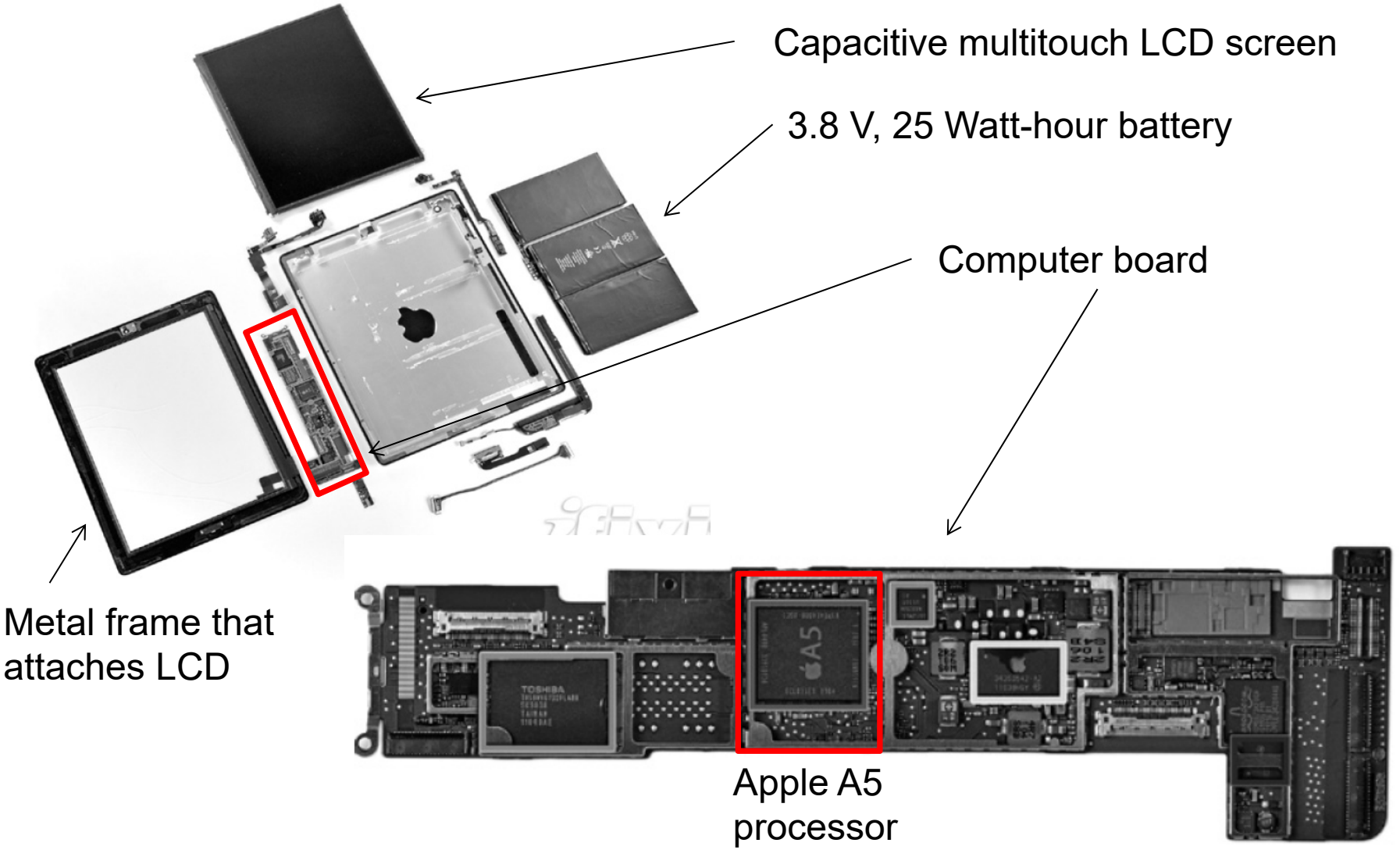


Opening the Box - Macbook



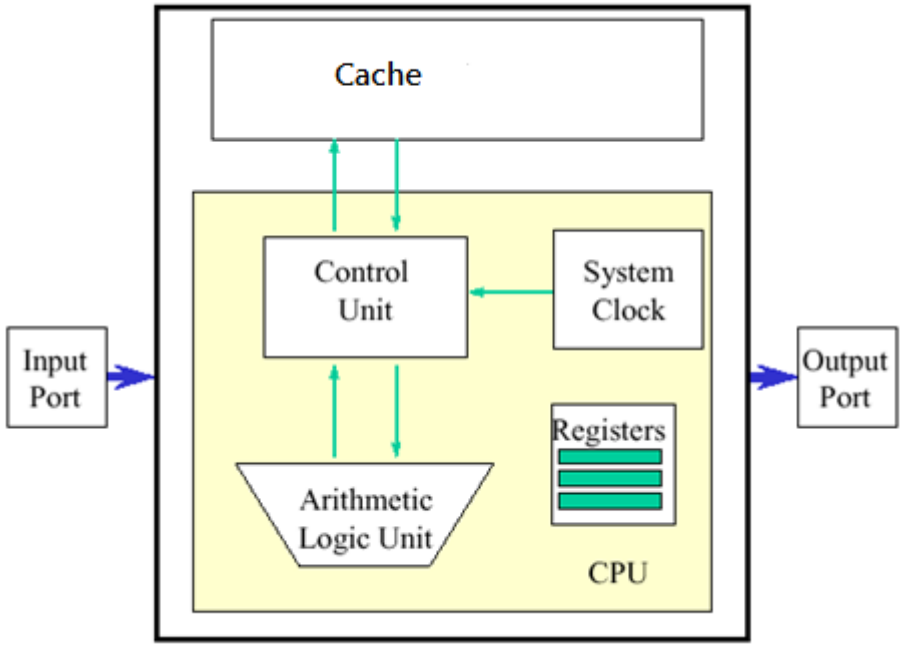


Opening the Box – iPad 2



Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: tells datapath, memory, and I/O devices what to do according to the wishes of the instructions.
- Cache memory
  - Small fast SRAM memory for immediate access to data

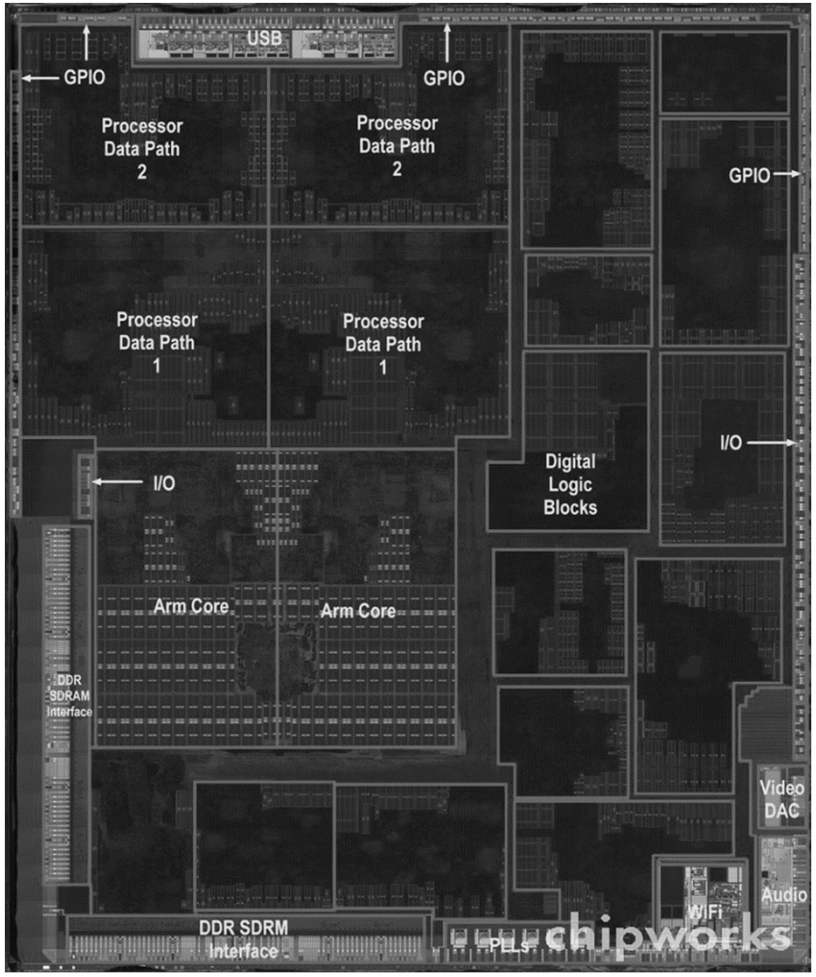


More in Chapter 3, 4 and 5

- Nontechnical definition of cache: a safe place for hiding thing

Inside the Processor- A5

- Apple A5



A Safe Place for Data

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

DRAM

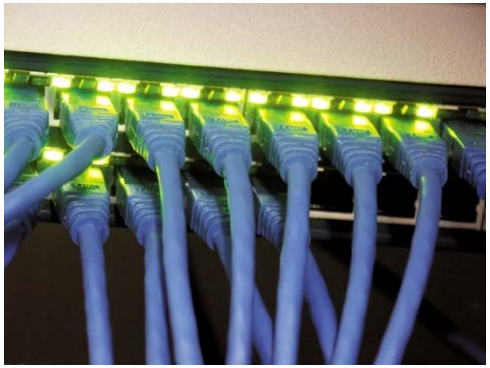
Hard Disk

Flash

CDROM

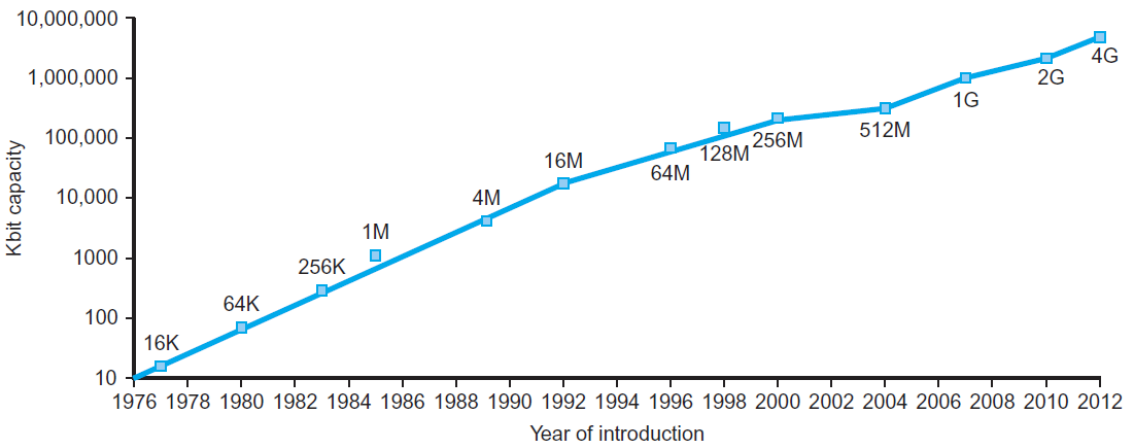
Networks

- Communication and resource sharing
- Local area network (LAN): Ethernet
  - Within a building
- Wide area network (WAN: the Internet)
- Wireless network: WiFi, Bluetooth, etc
- Mobile network: 3G(UMTS, CDMA2000), 3.5G (EVDO, HSDPA), 4G (LTE), 5G…



Technology Trends

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

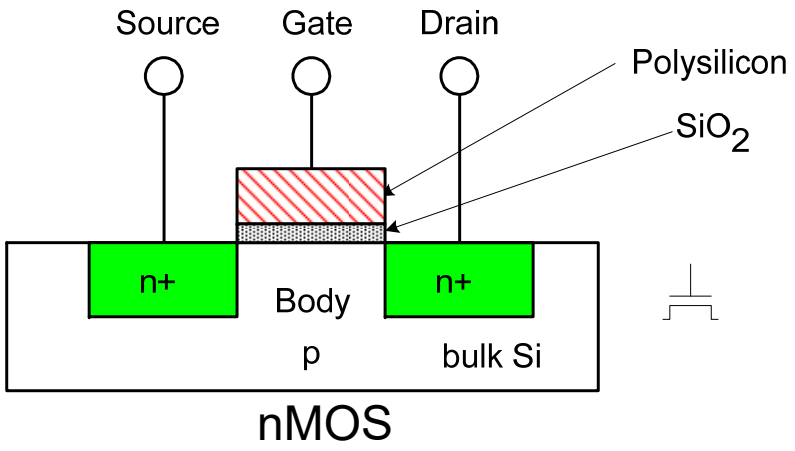


DRAM capacity

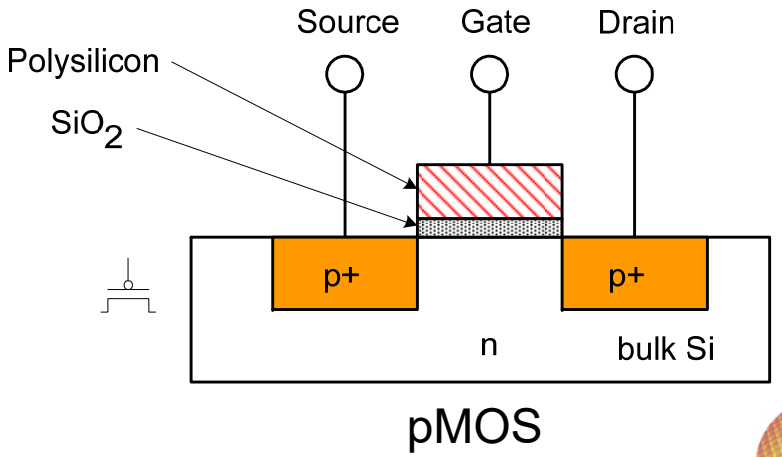
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2013	Ultra large scale IC	250,000,000,000

nMOS and pMOS transistor

- nMOS transistor
  - when input is 1 (V_{gate}=1)=> conduct current
- pMOS transistor
  - when input is 1 (V_{gate}=0), => Conduct current
- Three parts in a transistor: conductor, insulator, and switch
- Feature size => normally refer to the length of the channel

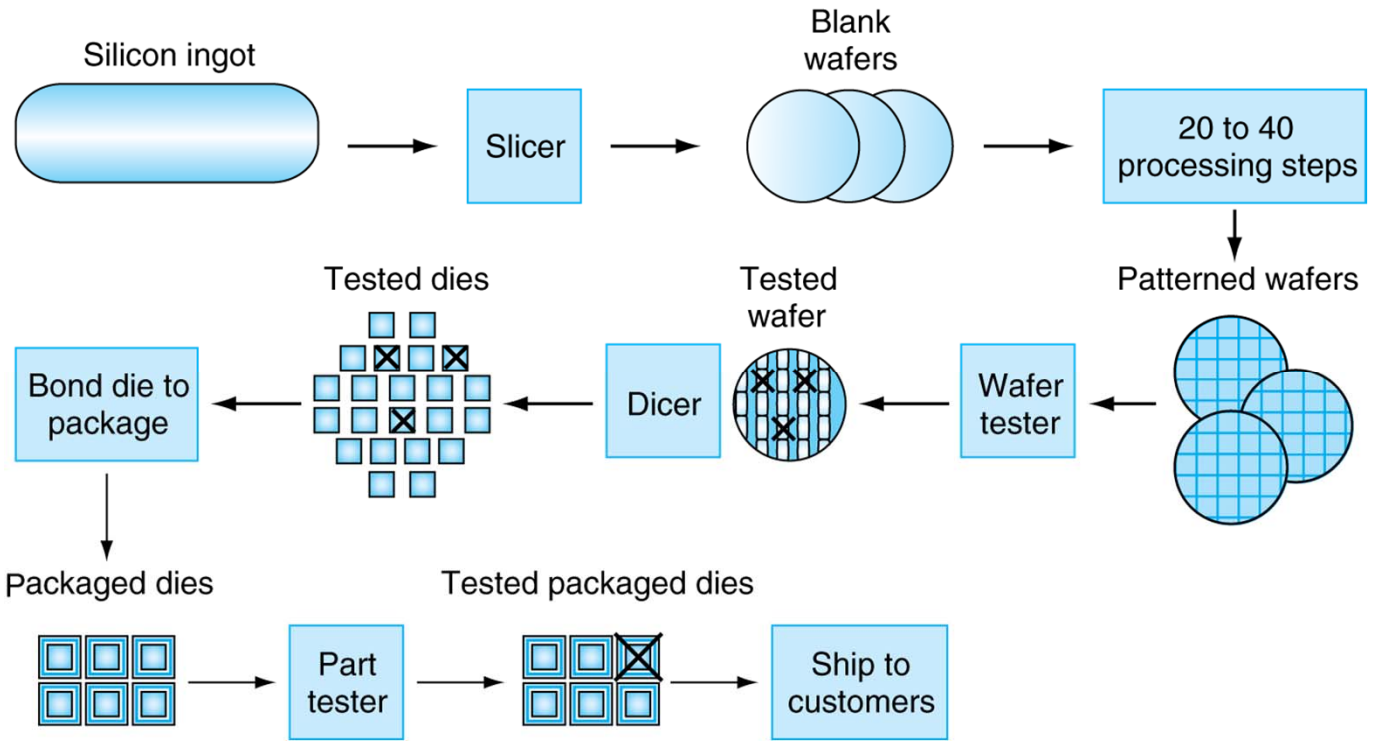


nMOS

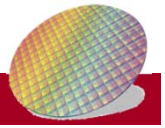


pMOS

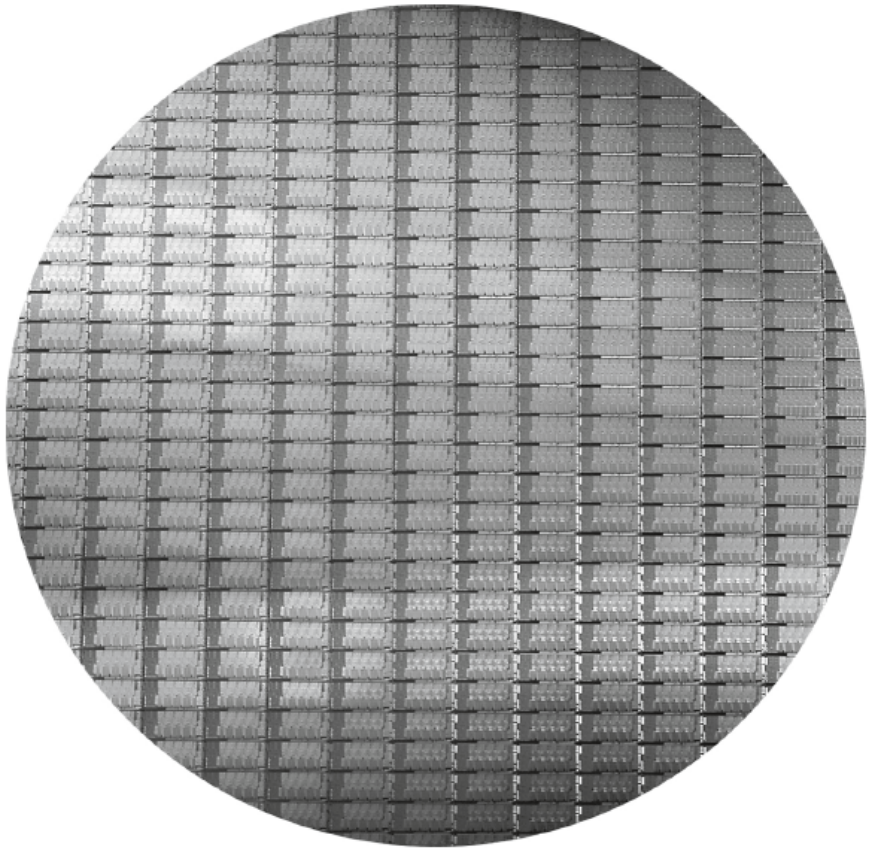
Manufacturing ICs



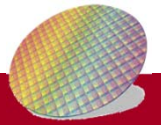
- Yield: proportion of working dies per wafer

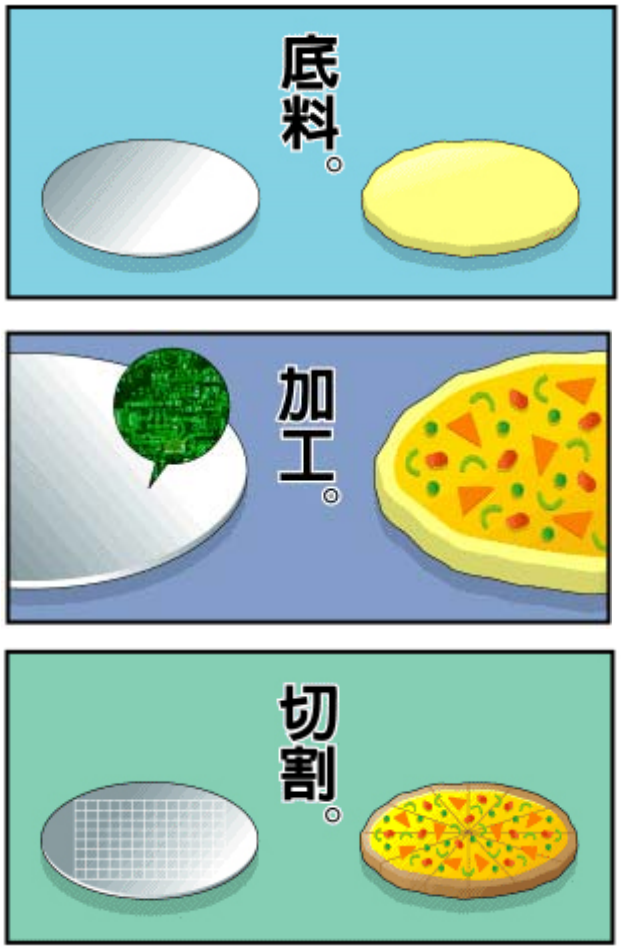


Intel Core i7 Wafer



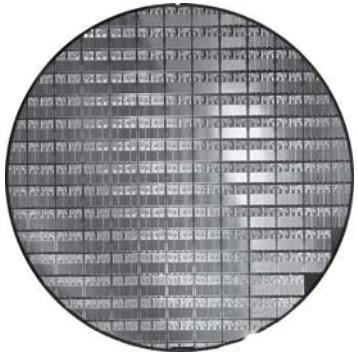
- 300mm wafer, 280 chips, 32nm technology
- Each chip is 20.7 x 10.5 mm





Analogy to IC Manufacturing

Integrated Circuit Cost



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

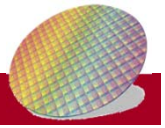
$$\text{Dies per wafer} \approx \text{Wafer area}/\text{Die area}$$

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

Just an approximation

$$Cost\ per\ die \propto area^3\ or\ area^4$$

- Wafer cost and area are fixed
- Defect rate determined by manufacturing process
- Die area determined by architecture and circuit design



Hidden

Yield Example

- Example: Assume a wafer diameter is 15cm, dies per wafer is 90, defect per unit area is 0.018 defects/cm^2, cost per wafer is 10
  - Find the yield
  - Find the cost per die
- Solutions

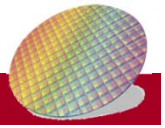
  Wafer area = $\pi \times (d/2)^2 = \pi \times 7.5^2 = 176.7\ cm^2$

  Die area = $176.7/90 = 1.96\ cm^2$

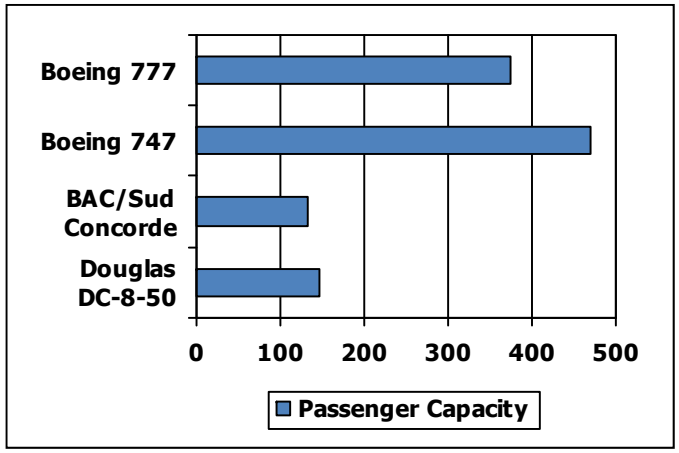
  Yield = ?

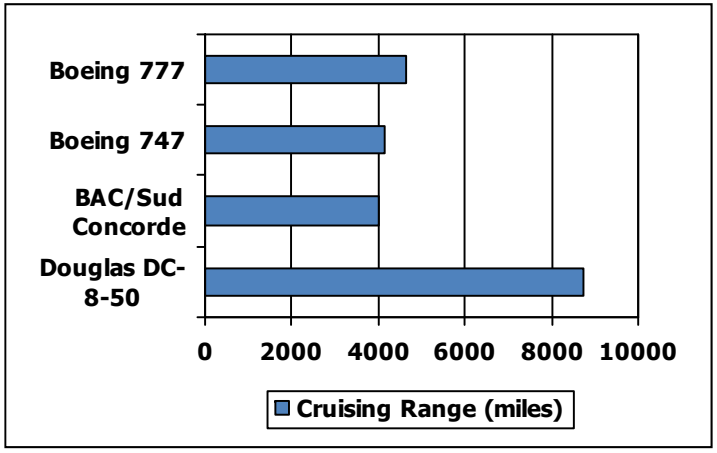
  Cost per die = cost per wafer/(dies per wafer $\times$ yield)

  Cost per die = ?



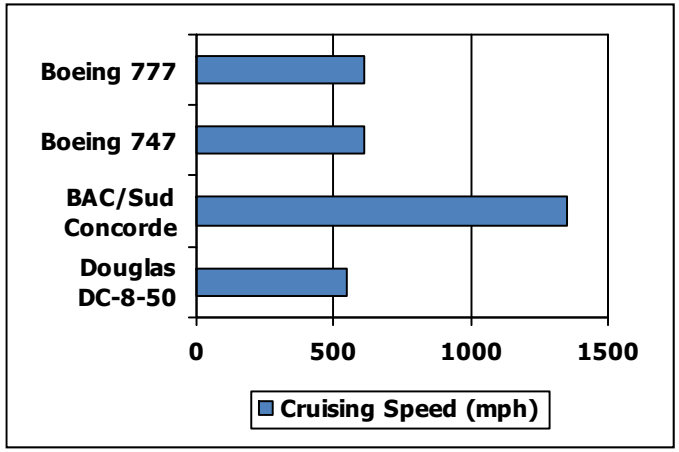
Which airplane has the best performance?

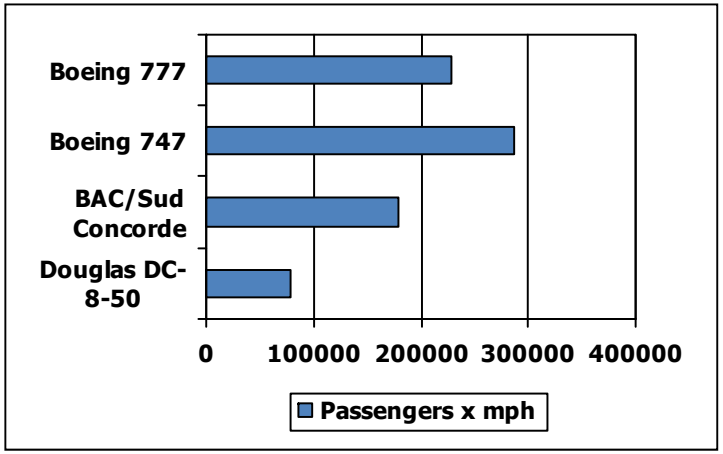




747

Concorde

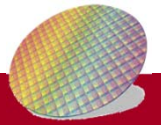




Response Time and Throughput

- Response time
  - The time between the start and completion of a task
    - e.g., 5 min to finish a task/transaction
- Throughput
  - Total amount of work done in a given time
    - e.g., tasks/transactions/… per hour
- We'll focus on response time for now…
- Performance of a computer X

$$Performance_X = \frac{1}{Execution\ Time_X}$$



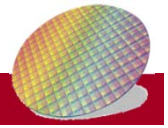
Hidden

Relative Performance

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

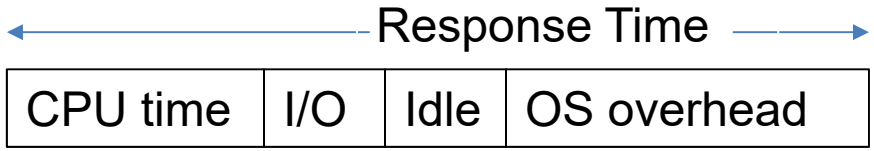
$$\frac{Performance_X}{Performance_Y} = n = \frac{Execution\ Time_Y}{Execution\ Time_X}$$

- Example: A takes 10s and B takes 15s to run a program, how much faster is A than B?



Response Time

- Response time (or called elapsed time)
  - Total response time,
  - including all aspects
    - Processing,
    - I/O,
    - OS overhead
    - idle time
  - e.g. A task takes 5 min to finish, including all processing time, I/O, and etc.
  - It is the time that users will experience

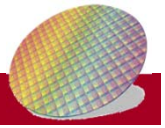


CPU Time

- CPU execution time (CPU time)
  - Time spent processing a given job (CPU may process many jobs at the same time)
  - Do not consider I/O time, other jobs' shares
  - Can be divided into user CPU time and system CPU time
    - User CPU time: CPU time spent in the program
    - System CPU time: CPU time spent in OS for this program

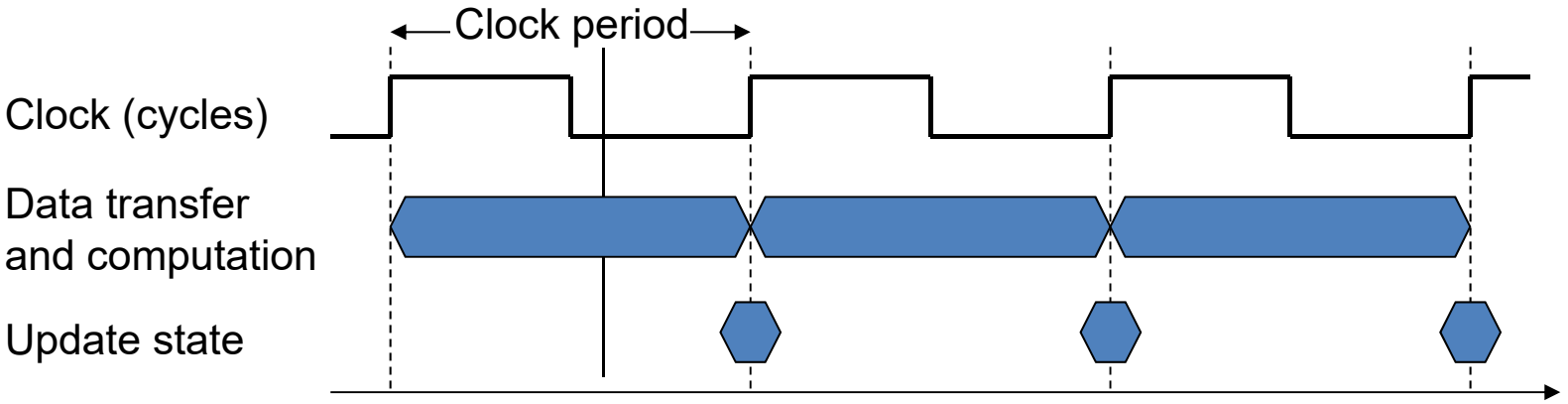
Quick questions: Define the following term

Response Time:
Throughput:
CPU time:



Review: 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

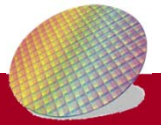
CPU Time

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

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

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count
- Example: a task takes 100 CPU cycles and each cycle is 1ns, what is the CPU Time?

CPU Time = 100 x 1 = 100 ns

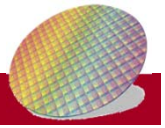


CPU Time Example

- Computer A: 2GHz clock, 10s CPU time. Design Computer B. It has 1.2 × clock cycles. However, CPU time is 6s because it uses faster clock. What is the clock rate of Computer B?

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$
$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

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



Instruction Count and CPI

- CPI: Cycle per instruction
  - Number of cycles each instruction takes to execute

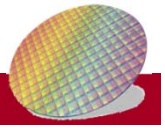
$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}}$$

- Instruction Count for a program determined by program, ISA and compiler
- Redefine CPU time using CPI

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

$$= \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

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



CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA (same number of instructions)
- Which is faster, and by how much?

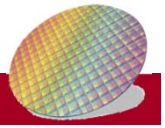
$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster…

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

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

1.2 times faster



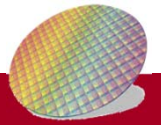
CPI in More Detail

- If different instruction classes take different numbers of cycles

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

  - Weighted average CPI

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

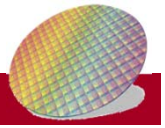


CPI Example

- Compiled code sequences using instructions in classes A, B, C. CPI_A=1, CPI_B=2, CPI_C=3, find average CPI for instruction sequence 1 and 2

Instruction Class	A	B	C
Inst. Count in sequence 1	2	1	2
Inst. Count 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

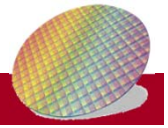


Performance Summary

The BIG Picture

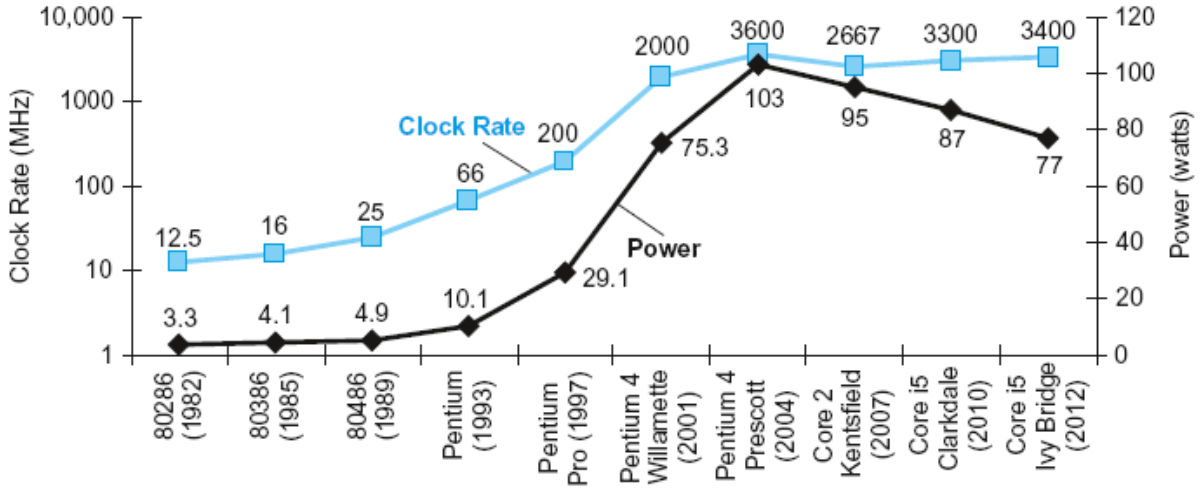
$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c



Power Trends

- CPU clock rate and power has increased rapidly for decades



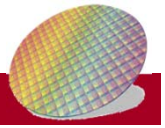
- In CMOS IC technology, energy of single transition

$$\text{Energy} = \frac{1}{2}\text{CV}^2$$

- Power of single transition

$$\text{Power} = \frac{1}{2}\text{CV}^2\text{f}$$

Frequency increase 1000x
Voltage decrease from 5V-> 1V,
Power increase 30x

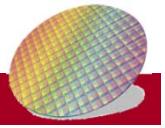


Hidden

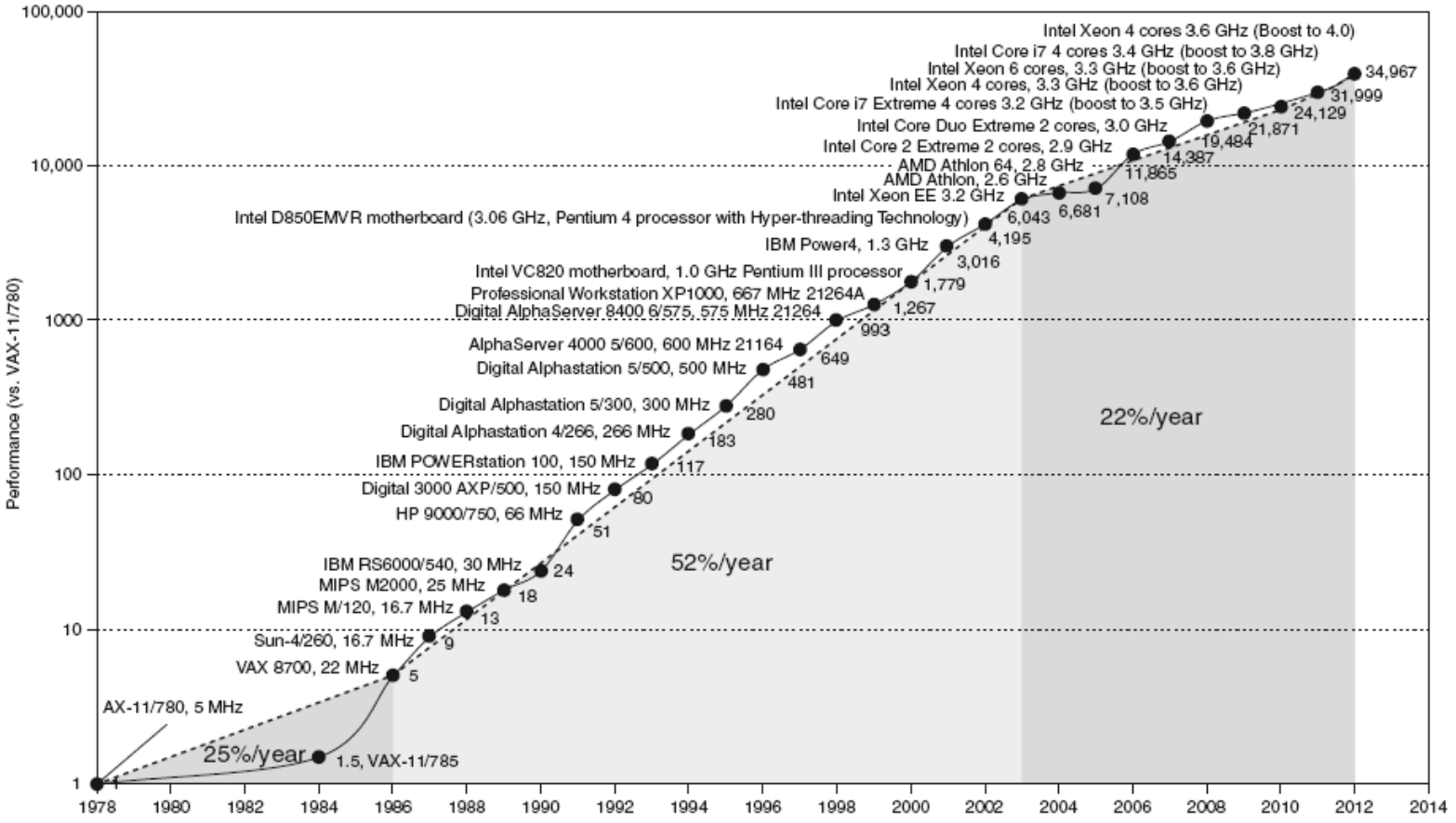
Reducing Power

- Suppose a new CPU has 85% of capacitive load of old CPU, and its voltage and frequency is reduced by 15%
  - What is $\frac{\text{Power}_{new}}{\text{Power}_{old}}$?

- The power wall
  - Hard to reduce voltage further
  - Hard to remove more heat
- How else can we improve performance?

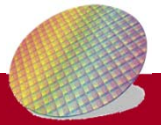


Uniprocessor Performance



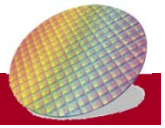
Constrained by power, instruction-level parallelism, memory latency

The Sea Change: The Switch to Multiprocessors



Multiprocessors

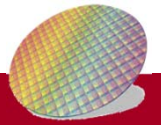
- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming
  - Compared to Instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Why parallel programming is hard to do?
    - Programming for both correctness and performance
    - Load balancing
    - Optimizing communication and synchronization



SPEC CPU Benchmark

- Benchmark: Programs used to measure performance
  - Supposedly typical of actual workload
- Standard Performance Evaluation Corp (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2006
  - 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) and CFP2006 (floating-point)

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



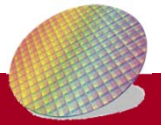
CINT2006 for Opteron X4 2356

Ref time: Time required by a reference computer

Name	Description	IC×10^9	CPI	Tc (ns)	Exec time	Ref time	SPEC ratio
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

15.3 times faster

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



SPECRatio of a benchmark suite

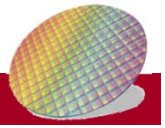
Benchmarks	Ultra 5 (sec)	Opteron (Sec)	SPECRatio
Wupwise	1600	51.5	31.06
Swim	3100	125.0	24.73
Mgrid	1800	98.0	18.37
Applu	2100	94.0	22.34
Mesa	1400	64.6	21.69
Galgel	2900	86.4	33.57
Art	2600	92.4	28.13
Equake	1300	72.6	17.92
Facerec	1900	73.6	25.80
Ammp	2200	136.0	16.14
Lucas	2000	88.8	22.52
Fma3d	2100	120.0	17.48
sixtrack	1100	123.0	8.95
apsi	2600	150.0	17.36
G. M.			20.886

If a benchmark suite contains many benchmarks, the overall SPECratio is the geometric mean of each SPECratio

$$SPECRatio_{overall} = \sqrt[n]{\prod_{i=1}^{n} SPECRatio_i}$$

$$= \sqrt[14]{31.06 \times 24.73 \ldots \times 17.36}$$

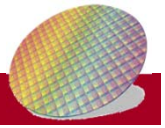
$$= 20.86$$



SPEC Power Benchmark

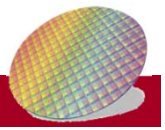
- Power consumption of server at different workload levels
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

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



SPECpower_ssj2008 for X4

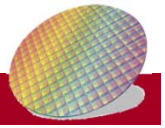
Target Load %	Performance (ssj_ops/sec)	Average Power (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
∑ssj_ops/ ∑power		493



Pitfall

- Improving an aspect of a computer and expecting a proportional improvement in overall performance
  - Example: multiplication accounts for 80s of 100s. Improve multiplication by 5x and expect overall performance to improve 5x, which is 20s
  - => Wrong
- Amdahl's Law: Possible improvement is limited by the amount that the improved feature is used

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



Amdahl's Law

- Amdahl's Law:Possible improved is limited by the amount that the improved feature is used

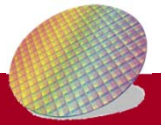
$$T_{improved} = \frac{T_{affected}}{improvement\ factor} + T_{unaffected}$$

- Example: multiplication accounts for 80s of 100s. How much improvement in multiplication performance to get 4x overall?

$$25 = \frac{80}{n} + 20$$

n = 16

multiplication needs to be 16x faster



Pitfall: use MIPS as a Performance Metric

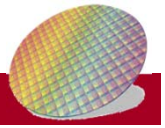
- MIPS: Millions of Instructions Per Second

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

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

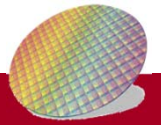
- Question: A and B has different instruction set. A runs 5 MIPS, and B run 4 MIPS, which one is faster?

We don't know because we don't know how much time each instruction takes



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
  - Note: CPI varies between programs on a given CPU, between various instruction sets
- See example next slide



MIPS Example

Measurement	Computer A	Computer B
Instr. Count	10 billion	8 billion
Clock rate	4 GHz	4GHz
CPI	1.0	1.1

$$\text{MIPS} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

- Which computer has higher MIPS?

$\text{MIPS}_A = 4/(1.0 \times 10^6)$

$\text{MIPS}_B = 4/(1.1 \times 10^6)$

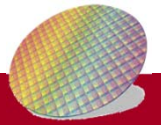
A has larger MIPS

- Which is faster?

$\text{CPU time}_A = 10 \times 10^9 \times 1.0/4 \times 10^9 = 2.5$

$\text{CPU time}_B = 8 \times 10^9 \times 1.1/4 \times 10^9 = 2.2$

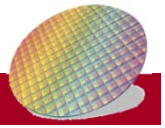
B is faster



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

Read Chapter 1



Backup slides