計算機組織與結構

114-1課程資訊

- 上課時間：週二01:10-04:00PM
- 上課地點：教108
- 授課教師：陳麒元 教授
- 辦公室：教穡417
- 電子信箱：chency@niu.edu.tw
- 個人網站：http://sites.google.com/site/chiyuanchen/
- 課程助教：劉育瑋 ryan891118@gmail.com
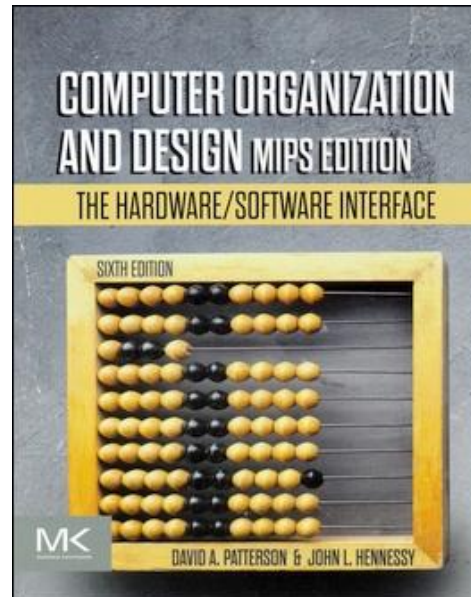- 實驗室：教穡419 網路安全實驗室

計算機組織與結構

114-1課程資訊

- 指定教材:
  - David A. Patterson & John L. Hennessy, Computer Organization and Design 6/e (MIPS Asia Edition), Elsevier B.V., 2021, ISBN: 9780128201091
  - 鍾崇斌, 楊惠親(譯),計算機組織與設計: 硬體/軟體的介面 6/e Patterson,東華書局, 2023, ISBN: 9786267130681

- 成績計算方式:
  - 平時成績與作業: 40%; (含至少1-2次作業)
  - 期中考: 30%;
  - 期末考: 30%

計算機組織與結構

114-1課程資訊

- 1　　　計算機抽象化與科技 I
- 2　　　計算機抽象化與科技 II
- 3　　　指令：計算機的語言 I
- 4　　　指令：計算機的語言 II
- 5　　　指令：計算機的語言 III
- 6　　　計算機的算術 I
- 7　　　計算機的算術 II
- 8　　　計算機的算術 III
- **9**　　　期中考
- 10　　處理器 I
- 11　　處理器 II
- 12　　處理器 III
- 13　　利用記憶體階層 I
- 14　　利用記憶體階層 II
- 15　　利用記憶體階層 III
- 16　　從客戶端到雲端的平行處理器 I
- **17**　　**從客戶端到雲端的平行處理器 II**
- **18**　　期末考

- 成績計算方式:
  - 平時成績與作業: 40%;
    (含至少1-2次作業)
  - 期中考: 30%;
  - 期末考: 30%

Table of Contents

Chapter 1

Computer Abstractions and Technology

The Computer Revolution

- Progress in computer technology
    - Underpinned by domain-specific accelerators
- Makes novel applications feasible
    - Computers in automobiles
    - Cell phones
    - Human genome project
    - World Wide Web
   - Search Engines
- Computers are pervasive

Classes of Computers

- Personal computers(個人電腦)
  - General purpose, variety of software
  - Subject to cost/performance tradeoff

- Server computers(伺服器電腦)
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized

Classes of Computers

- Supercomputers(超級電腦)
    - Type of server
    - High-end scientific and engineering calculations
    - Highest capability but represent a small fraction of the overall computer market

- Embedded computers(嵌入式電腦)
    - Hidden as components of systems
    - Stringent power/performance/cost constraints

The PostPC Era
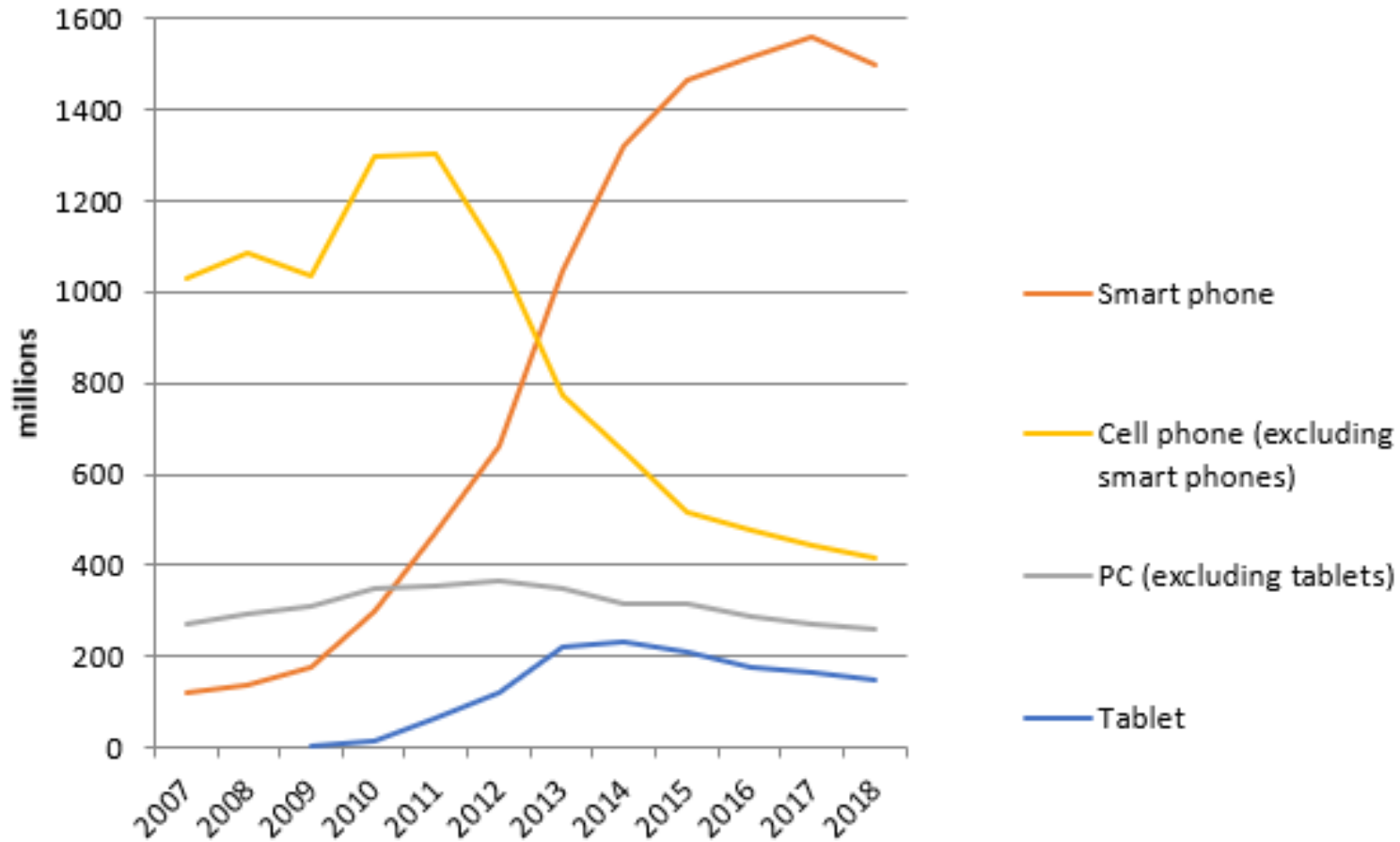
The PostPC Era

- Personal Mobile Device (PMD,個人行動裝置)
    - Battery operated
    - Connects to the Internet
    - Hundreds of dollars
    - Smart phones, tablets, electronic glasses
- Cloud computing (雲端運算)
    - Warehouse Scale Computers (WSC,倉儲規模電腦)
    - Software as a Service (SaaS,軟體即服務)
    - Portion of software run on a PMD and a portion run in the Cloud
    - Amazon and Google

What You Will Learn

- How programs are translated into the machine language
  - And how the hardware executes them
- The hardware/software interface
- What determines program performance
  - And how it can be improved
- How hardware designers improve performance
- What is parallel processing

Understanding Performance

Hardware or software component	How this component affects performance	Where is this topic covered?
Algorithm	Determines both the number of source-level statements and the number of I/O operations executed	Other books !
Programming language, compiler, and architecture	Determines the number of computer instructions for each source-level statement	Chapters 2 and 3
Processor and memorysystem	Determines how fast instructions can be executed	Chapters 4, 5, and 6
I/O system (hardware and operating system)	Determines how fast I/O operations may be executed	Chapters 4, 5, and 6

Seven Great Ideas

- Use **abstraction (*抽象化*)** to simplify design

- Make the **common case (*常見情形*) fast**

- Performance *via* **parallelism (*平行性*)**

- Performance *via* **pipelining (*管道化*)**

- Performance *via* **prediction (*預測*)**

- **Hierarchy (*階層式*)** of memories
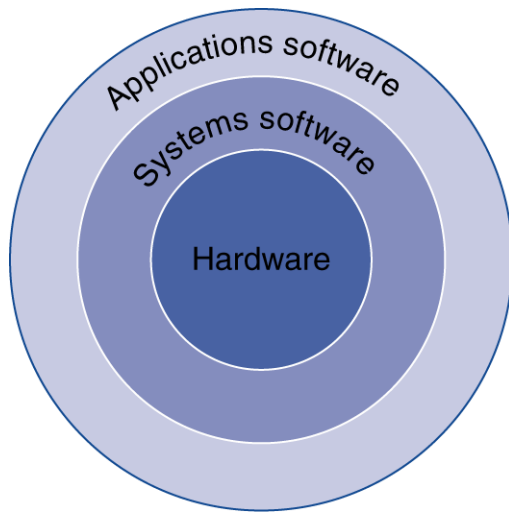
- **Dependability (*可靠性*)** *via* redundancy (冗餘)

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

Applications software

Systems software

Hardware

Levels of Program Code

- High-level language (高階語言)
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language (組合語言)
  - 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)

```
00000000101000010000000000011000
00000000000110000000110000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Components of a Computer

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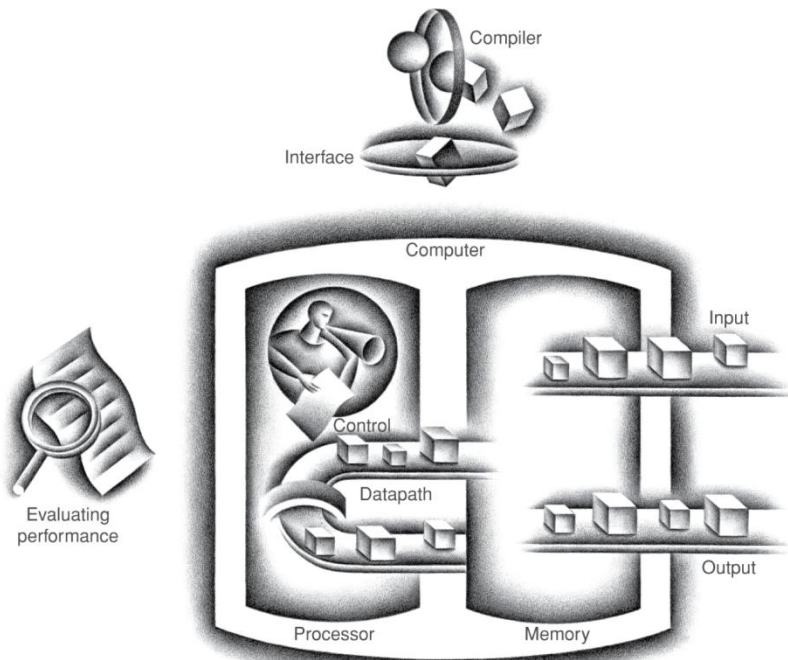
- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
   - For communicating with other computers

Touchscreen

- PostPC device

- Supersedes keyboard and mouse

- Resistive and Capacitive types
    - Most tablets, smart phones use capacitive
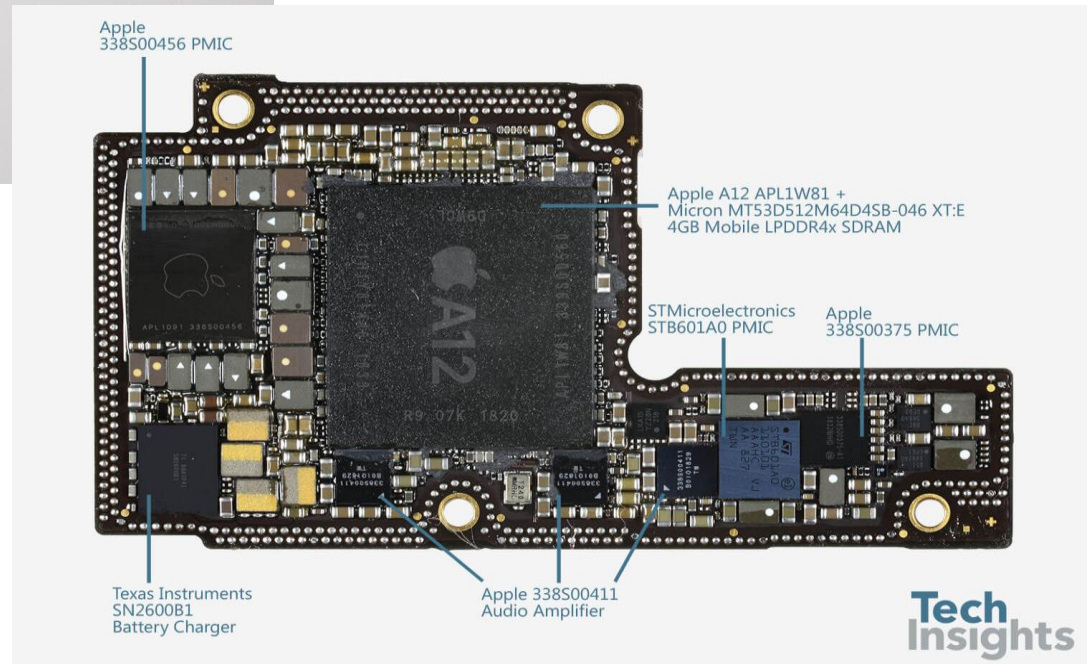   - Capacitive allows multiple touches simultaneously
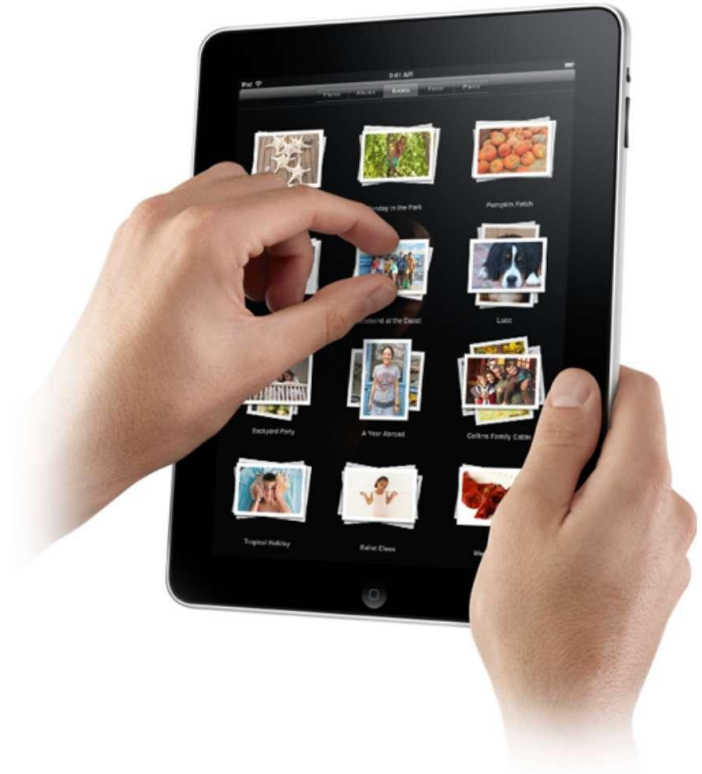
Through the Looking Glass

- LCD screen: picture elements (pixels)
  - Mirrors content of frame buffer memory

Frame buffer

Raster scan CRT display
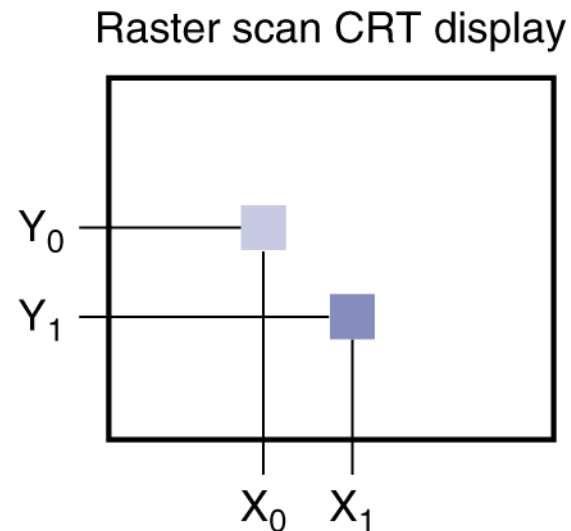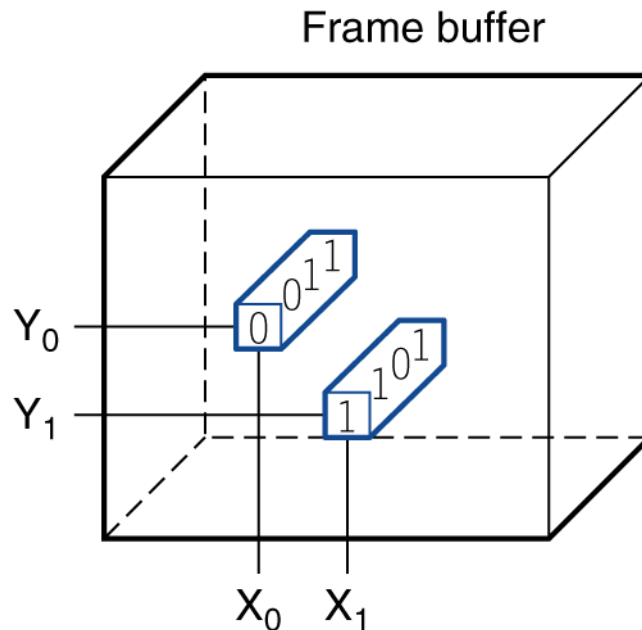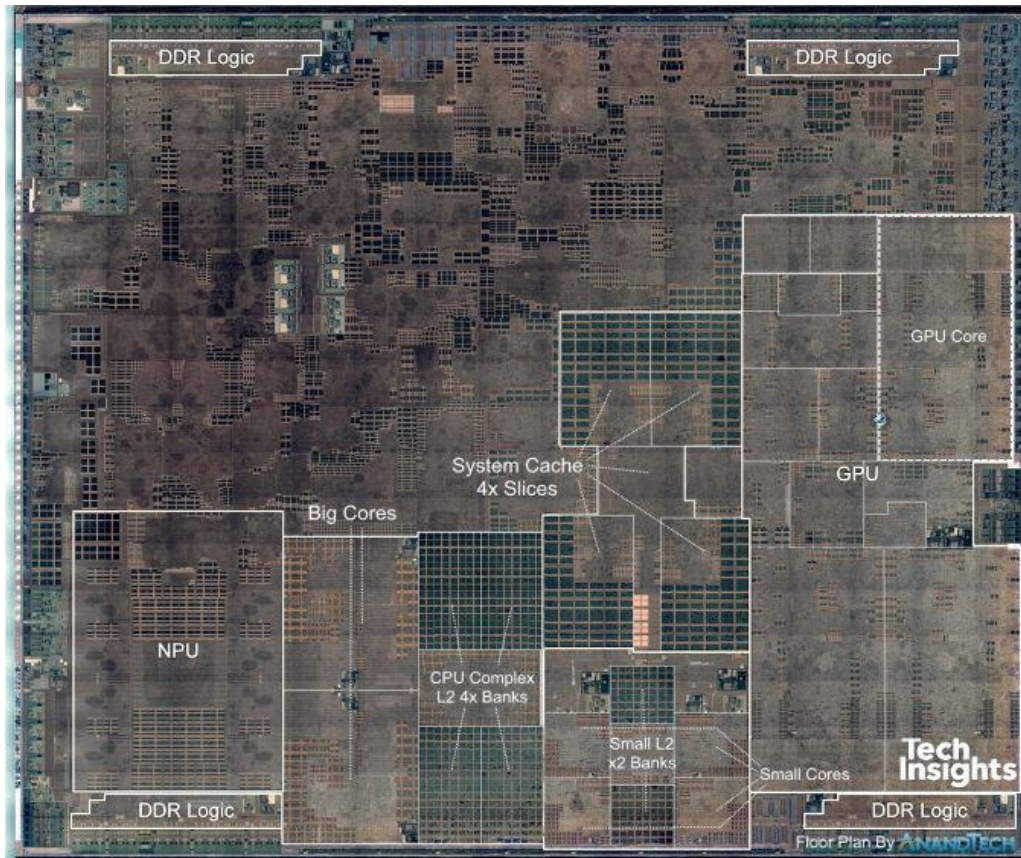
Opening the Box
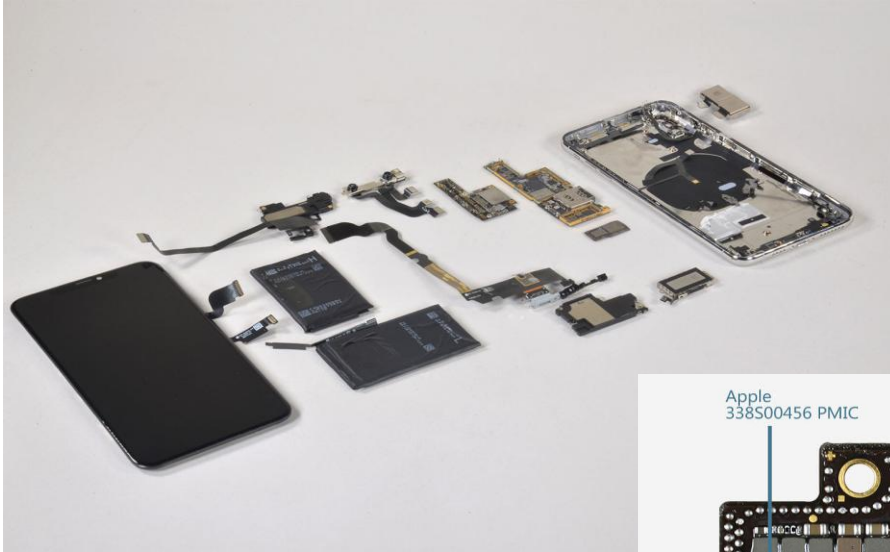
Inside the Processor (CPU)

- Datapath(資料路徑): performs operations on data

- Control Unit(控制單元): sequences datapath, memory, ...

- Cache memory(快取記憶體)
  - Small fast SRAM memory for immediate access to data

Inside the Processor

- A12 processor

Abstractions

- 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

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)
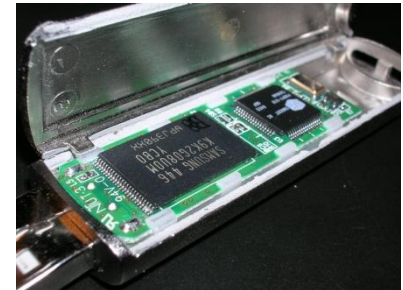
Networks

- Communication, resource sharing, nonlocal access

- Local area network (LAN): Ethernet
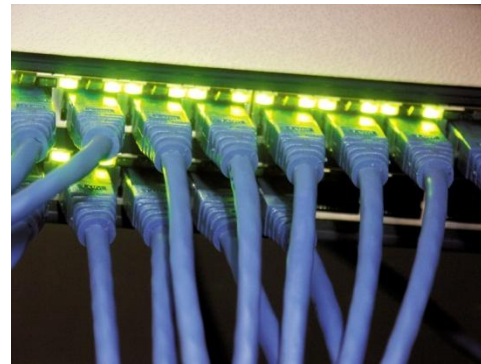
- Wide area network (WAN): the Internet

- Wireless network: WiFi, Bluetooth

Technology Trends

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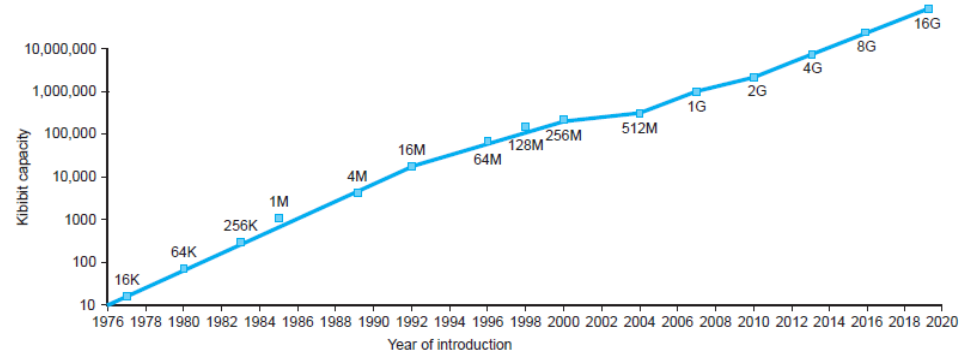
DRAM capacity

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
2020	Ultra large scale IC	500,000,000,000

Semiconductor Technology

- Silicon: semiconductor

- Add materials to transform properties:
    - Conductors
    - Insulators
    - Switch

Manufacturing ICs
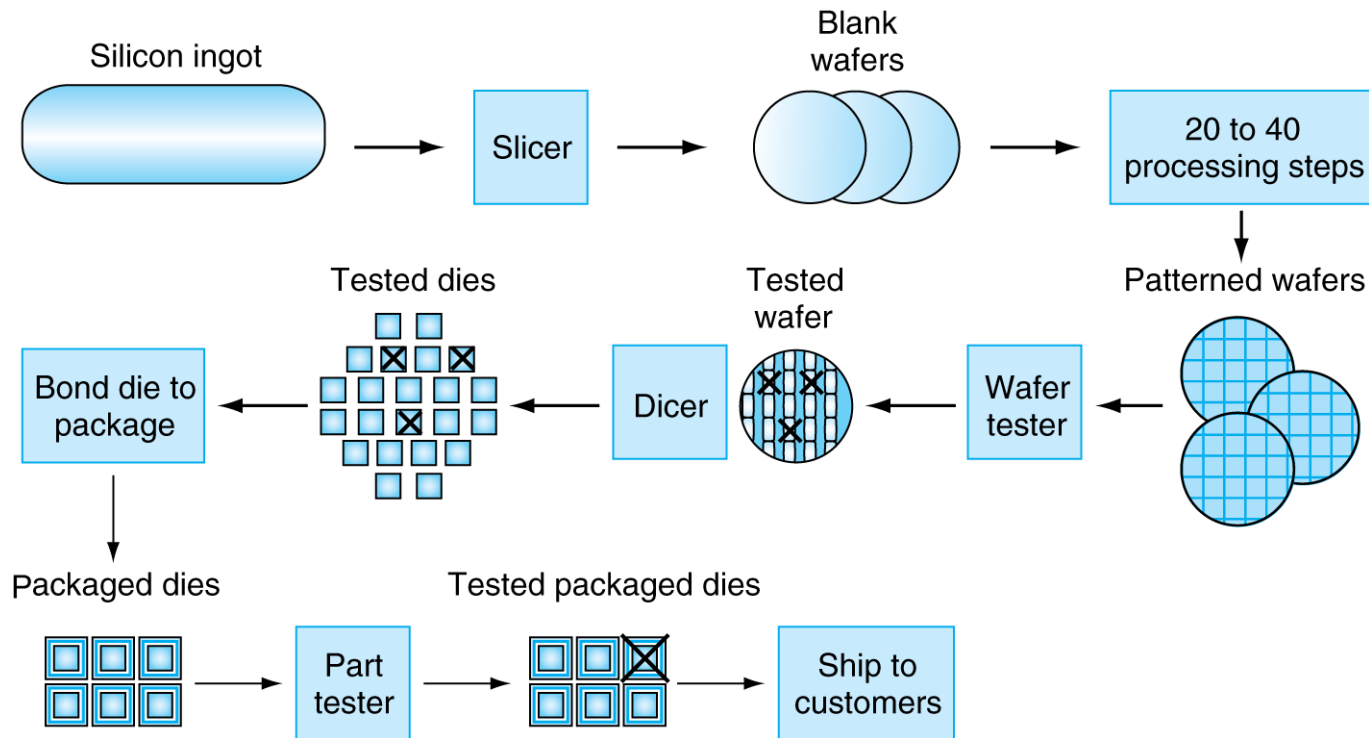
- Yield(良率): proportion of working dies per wafer(晶圓)
- https://www.youtube.com/watch?v=ife-z2T9QLM
- https://www.youtube.com/watch?v=AHe_yVZW6no

Intel® Core 10th Gen
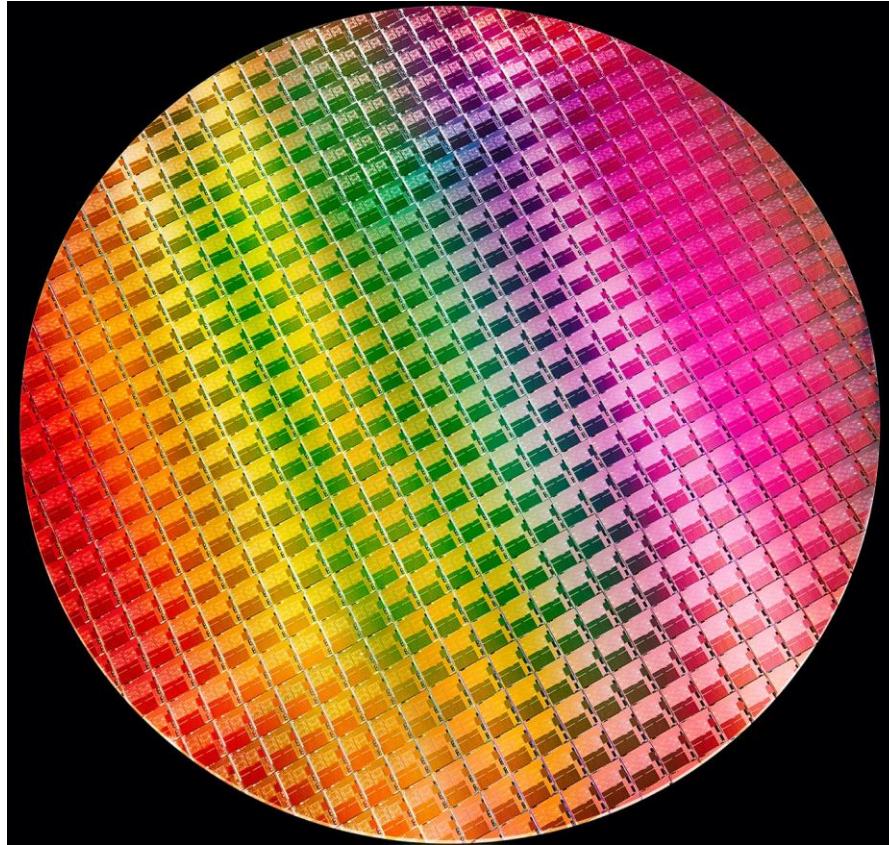
- 300mm wafer, 506 chips, 10nm technology
- Each chip is 11.4 x 10.7 mm

Integrated Circuit Cost

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

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

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

- 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

Defining Performance

- Which airplane has the best performance?

Passenger capacity — Airbus A380-800, Boeing 777-200LR, BAC/Sud Concorde, Boeing 737

Cruising range (miles) — Airbus A380-800, Boeing 777-200LR, BAC/Sud Concorde, Boeing 737

Cruising speed (m.p.h.) — Airbus A380-800, Boeing 777-200LR, BAC/Sud Concorde, Boeing 737

Passenger throughput (passengers × m.p.h.) — Airbus A380-800, Boeing 777-200LR, BAC/Sud Concorde, Boeing 737
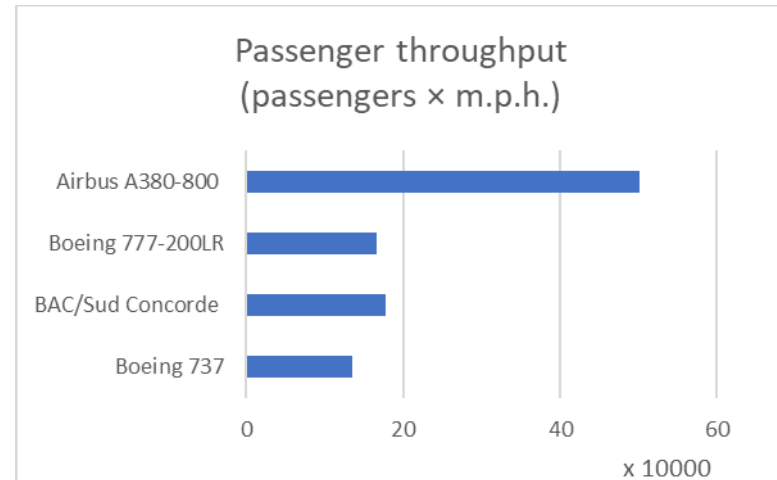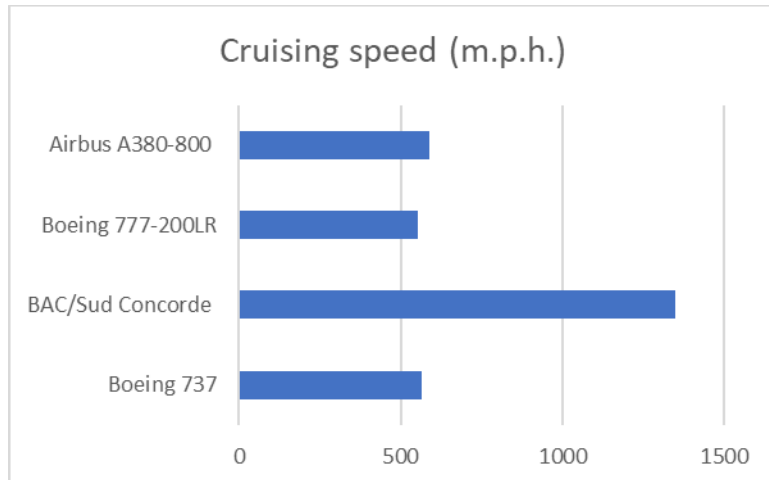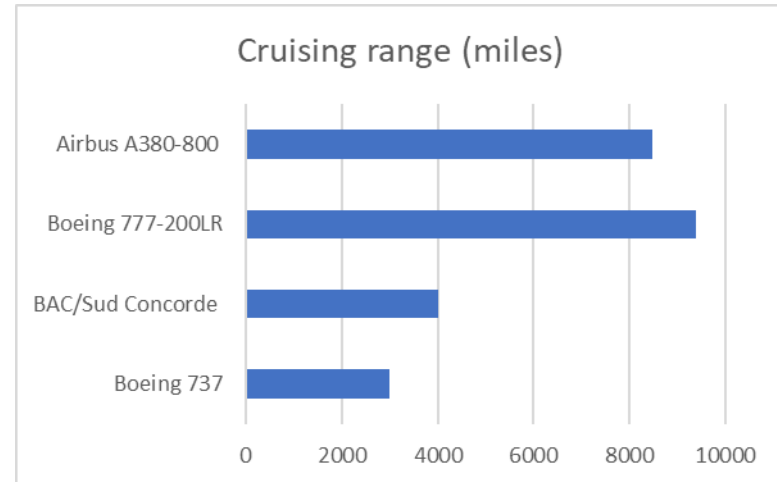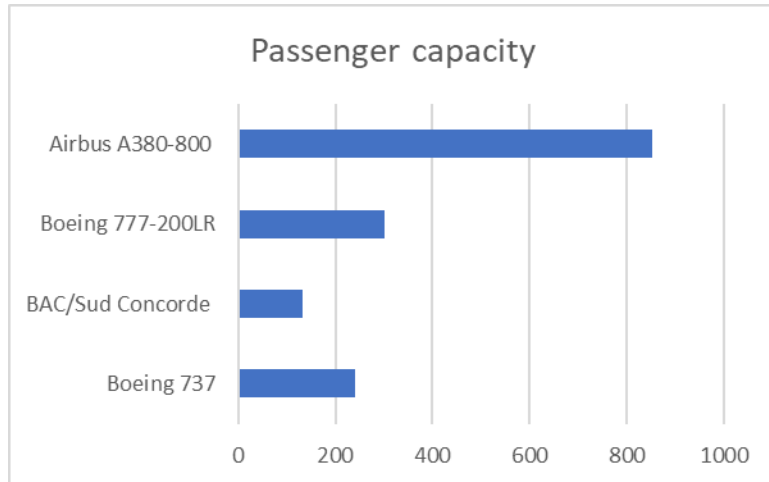
Response Time and Throughput

- Response time(反應時間)
  - How long it takes to do a task
- Throughput(處理量)
  - 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…

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

Measuring Execution Time

- Elapsed time(經過時間)
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time(執行時間)
  - 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

CPU Clocking

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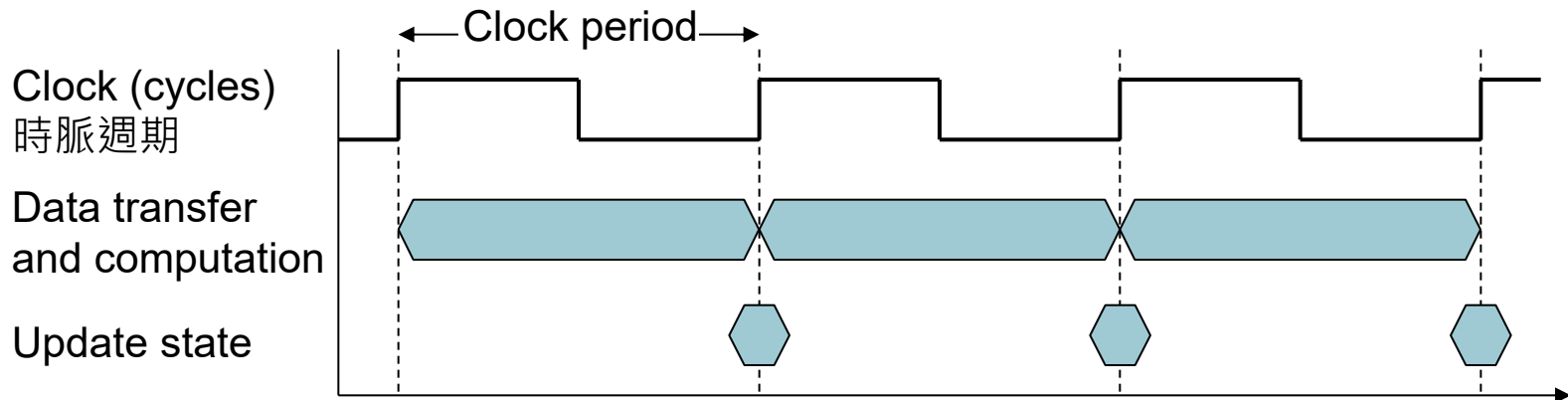
- Clock period: duration of a clock cycle
    - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second
    - e.g., 4.0GHz = 4000MHz = 4.0×10^9Hz

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?

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

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

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

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

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

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

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

- 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?

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

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps} \quad \leftarrow \boxed{\text{A is faster...}}$$

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

$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2 \quad \leftarrow \boxed{\text{...by this much}}$$

CPI in More Detail

- If different instruction classes take different numbers of cycles

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

  - Weighted average CPI

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

Relative frequency

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

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

- 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
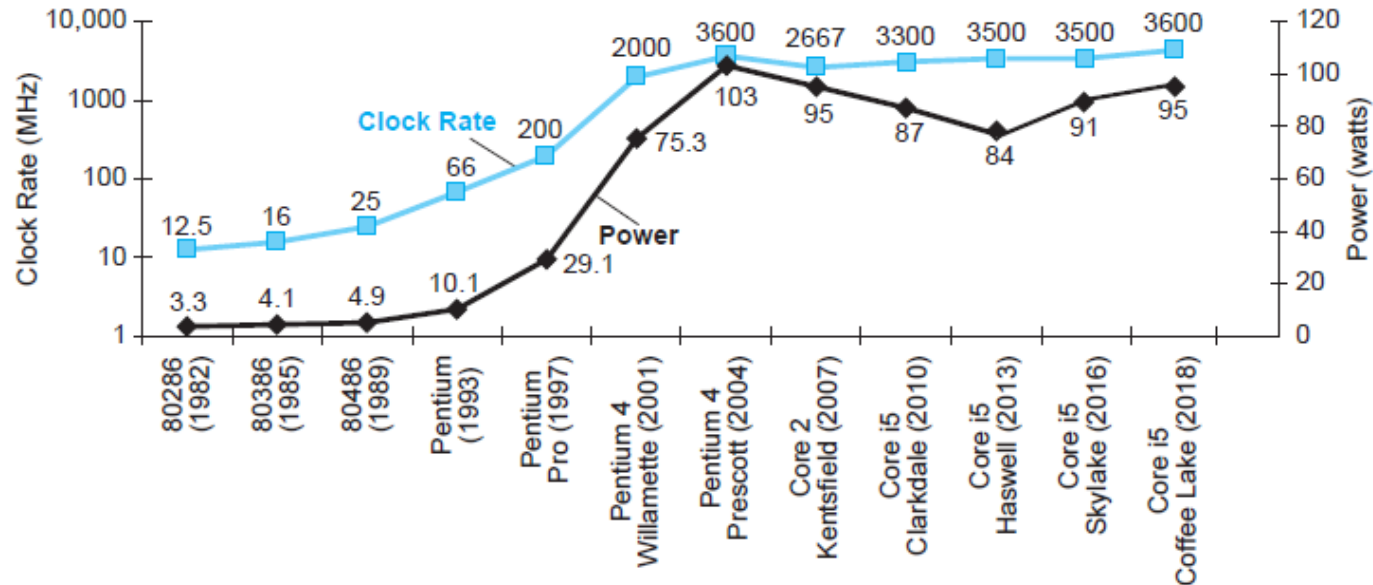
- In CMOS IC technology

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

×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
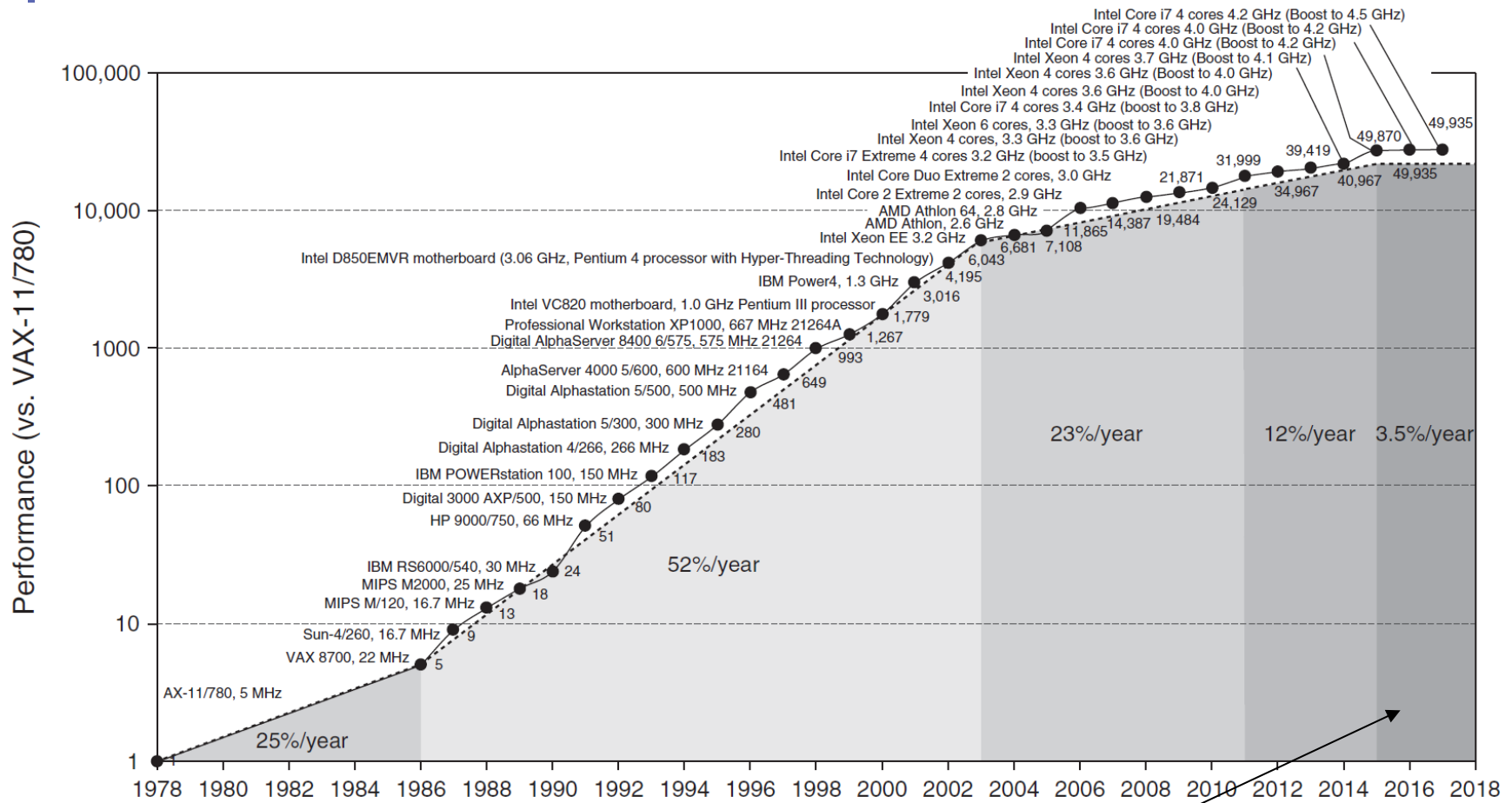
Constrained by power, instruction-level parallelism, memory latency

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

SPEC CPU 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
  - 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) and CFP2006 (floating-point)

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

SPECspeed 2017 Integer benchmarks on a 1.8 GHz Intel Xeon E5-2650L

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^–9)	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Perl interpreter	perlbench	2684	0.42	0.556	627	1774	2.83
GNU C compiler	gcc	2322	0.67	0.556	863	3976	4.61
Route planning	mcf	1786	1.22	0.556	1215	4721	3.89
Discrete Event simulation - computer network	omnetpp	1107	0.82	0.556	507	1630	3.21
XML to HTML conversion via XSLT	xalancbmk	1314	0.75	0.556	549	1417	2.58
Video compression	x264	4488	0.32	0.556	813	1763	2.17
Artificial Intelligence: alpha-beta tree search (Chess)	deepsjeng	2216	0.57	0.556	698	1432	2.05
Artificial Intelligence: Monte Carlo tree search (Go)	leela	2236	0.79	0.556	987	1703	1.73
Artificial Intelligence: recursive solution generator (Sudoku)	exchange2	6683	0.46	0.556	1718	2939	1.71
General data compression	xz	8533	1.32	0.556	6290	6182	0.98
Geometric mean							2.36

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 Xeon E5-2650L

Target Load %	Performance (ssj_ops)	Average Power (watts)
100%	4,864,136	347
90%	4,389,196	312
80%	3,905,724	278
70%	3,418,737	241
60%	2,925,811	212
50%	2,439,017	183
40%	1,951,394	160
30%	1,461,411	141
20%	974,045	128
10%	485,973	115
0%	0	48
Overall Sum	26,815,444	2,165
$\sum$ssj_ops / $\sum$power =		12,385

Pitfall: Amdahl's Law

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

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

- 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 i7 power benchmark
  - At 100% load: 258W
  - At 50% load: 170W (66%)
  - At 10% load: 121W (47%)
- 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

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

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

  - ### 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