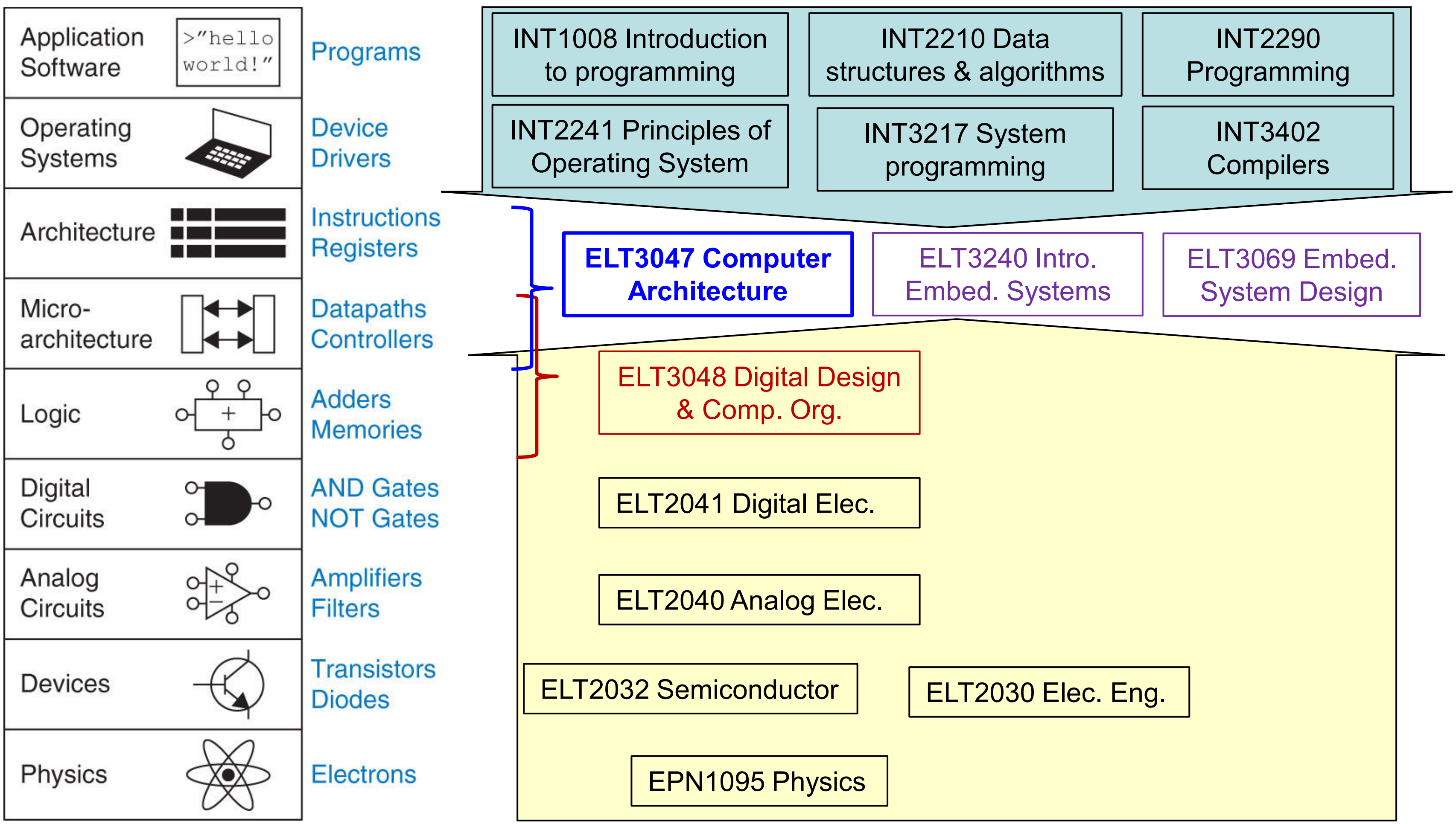
ELT3047 Computer Architecture

Lecture 1: Introduction

Hoang Gia Hung
Faculty of Electronics and Telecommunications
University of Engineering and Technology, VNU Hanoi

Course overview (1)

- VNU-UET's ECE subject chart



Course overview (2)



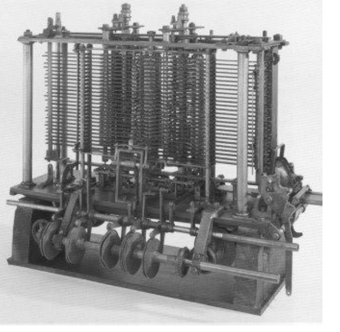
- **<u>Course contents</u>**: you'll learn what's under the hood of a **modern** computer
  - How programs are translated into the machine language
    - ✓ And how the hardware executes them
  - The hardware/software interface
    - ✓ How does software instruct the hardware to perform needed functions?
  - What determines program performance
    - ✓ And can a programmer improve the performance?
  - How hardware designers improve performance

Course overview (3)

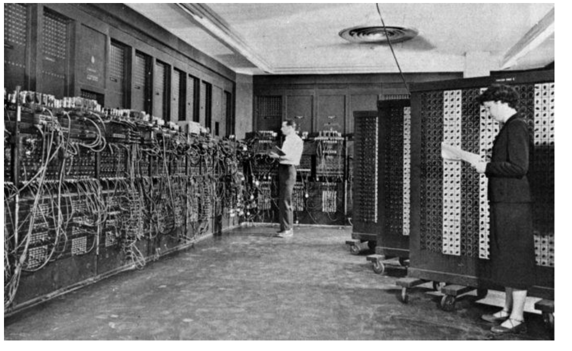
- Lecturer
  - Hoàng Gia Hưng, Dept. Elect. Comp. Eng. (R702, E3 buiding, 144 Xuan Thuy)
  - Appointment-based consultation
- Pre-requisites: INT1008/2290, ELT2041.
- Text book: David Patterson and John Hennessy, "Computer Organization & Design: The Hardware/Software Interface" (5th Ed.), Morgan Kaufmann Publishers, 2014.
- Grading:
  - Quizzes/essays: 15%
  - Midterm: 25%
  - Final: 60%
- Some ground rules:
  - Respect
  - Proactive
  - Punctual

The computer evolution

- Mechanical computers
  - Schickhard (1623), Pascal (1642)
  - Babbage (1823 – Difference Engine, 1834 – Analytical Engine)



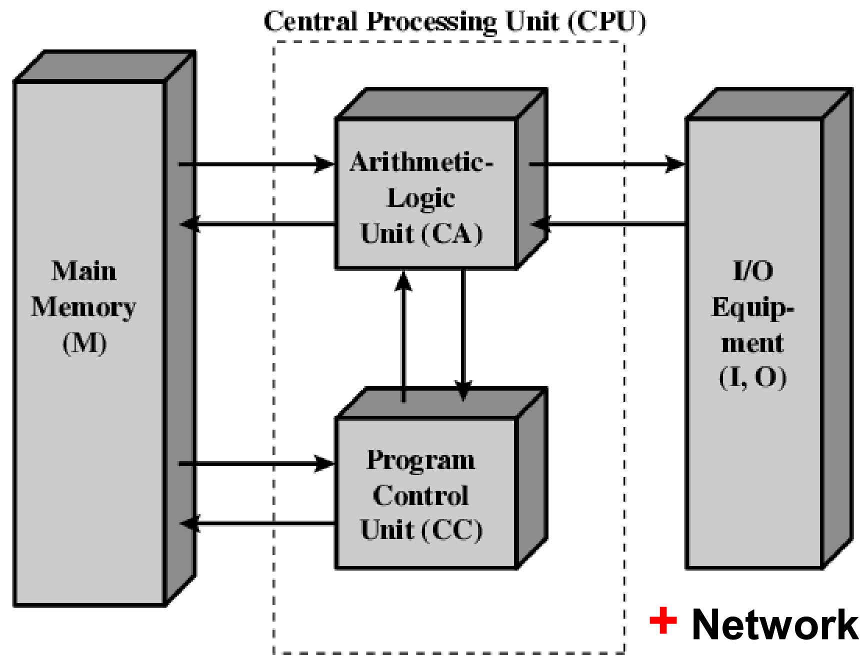
- The Electronic Era (1946-1974)
  - ENIAC (1943-1946)
  - EDVAC (1944-1952) & **Von Neumann computer** (1945)
  - Mark I-IV (1944-1952) **Harvard architecture**



- Modern computers
  - The PC Era (1975-2009)
  - The Post-PC Era (2010-present)

The Five Classic Components of a (Von Neumann) Computer

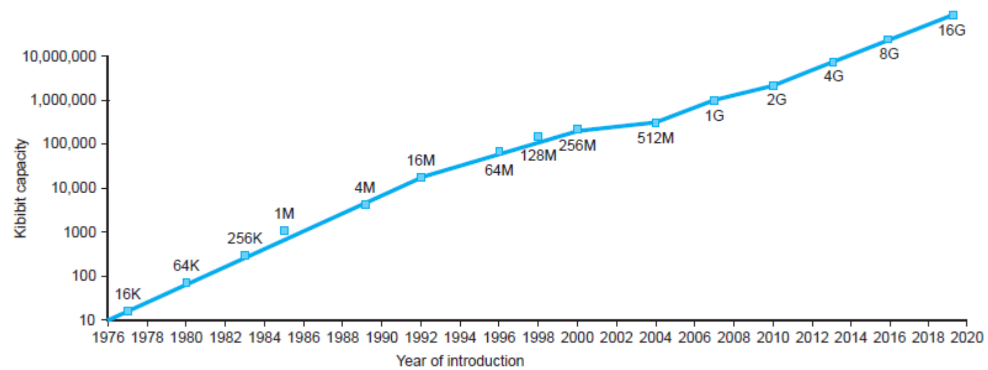
- A central arithmetical unit capable of perform the elementary operations of arithmetic (***Datapath***)
- A central control unit capable of logical control of the device, i.e. properly sequencing of its operations (***Control***)



- A main memory, which stores both data and instructions (***Memory***)
  - Stored-program concept
- Input units to transfer information from the outside recording medium (R) into its specific parts C (CA+CC) and M (***Input***).
- Output units to transfer to transfer information from C (CA+CC) and M to R (***Output***).

Technology Trends

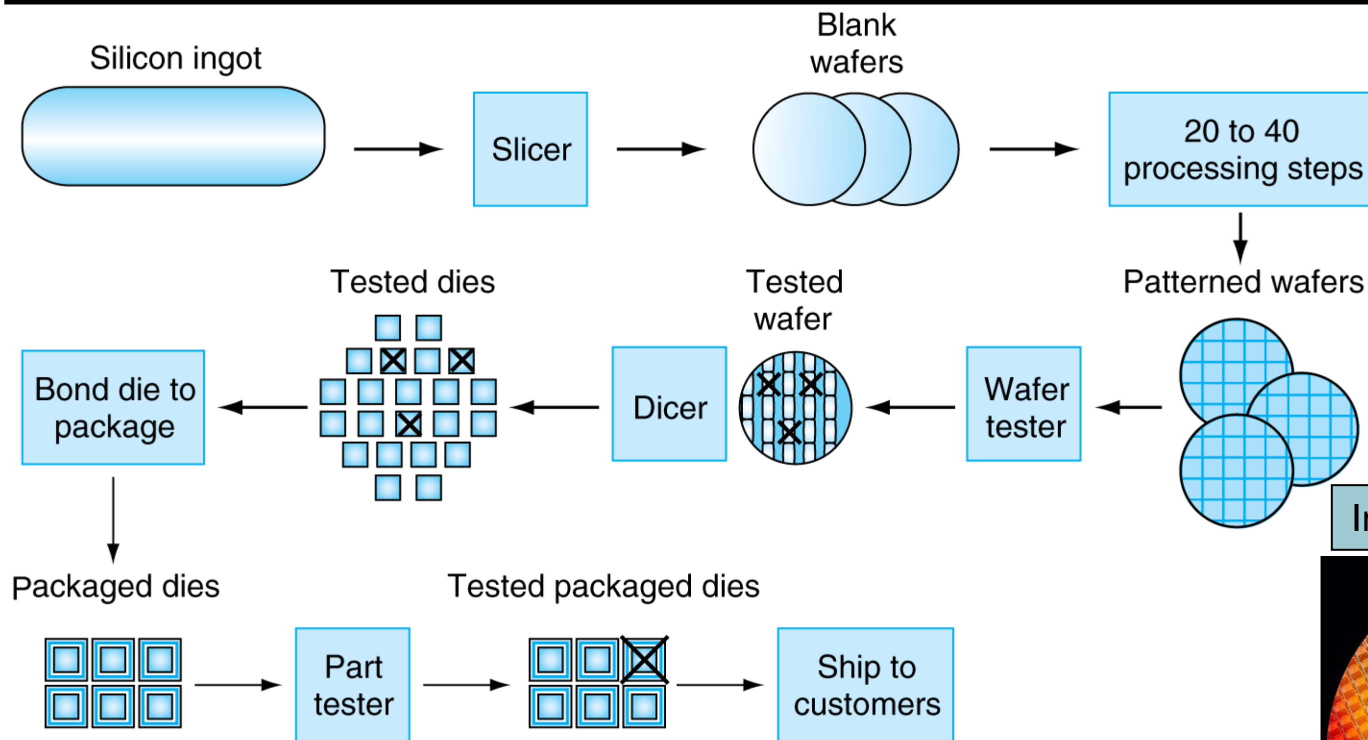
- Electronics technology continues to evolve
  - ✓ Increased capacity and performance
  - ✓ Reduced cost



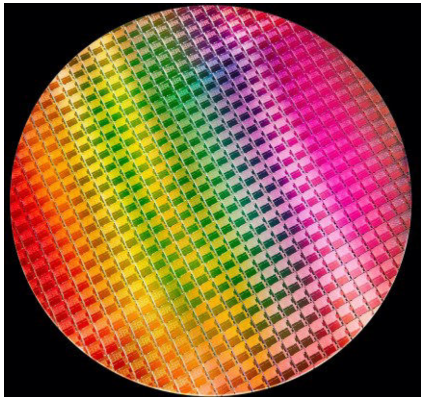
DRAM capacity

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

Semiconductor Technology



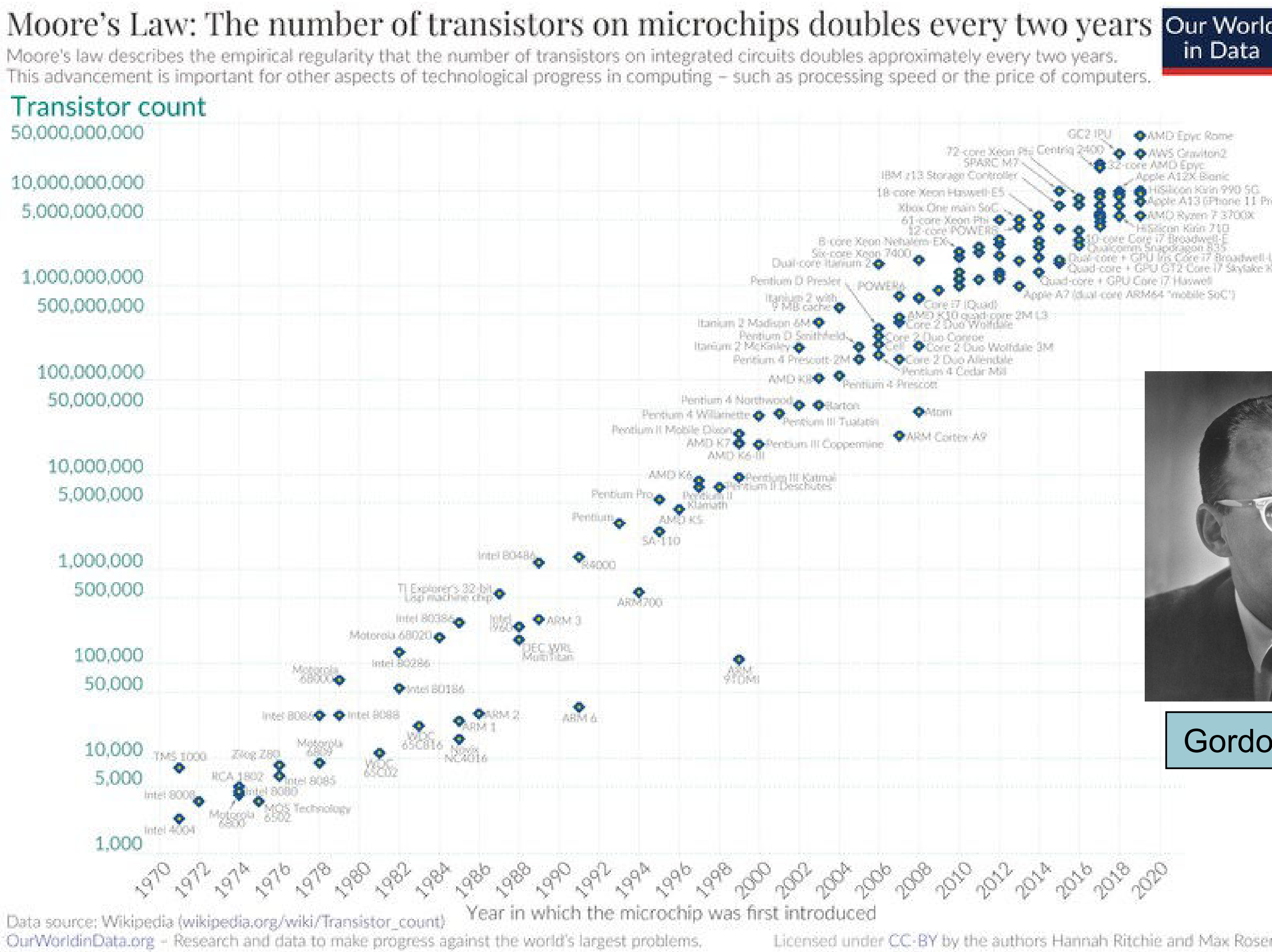
Intel® Core 10th Gen



10nm technology
300mm wafer, 506 chips,
Each chip is 11.4 x 10.7 mm

- IC manufacturing process
  - ✓ Yield: proportion of working dies per wafer

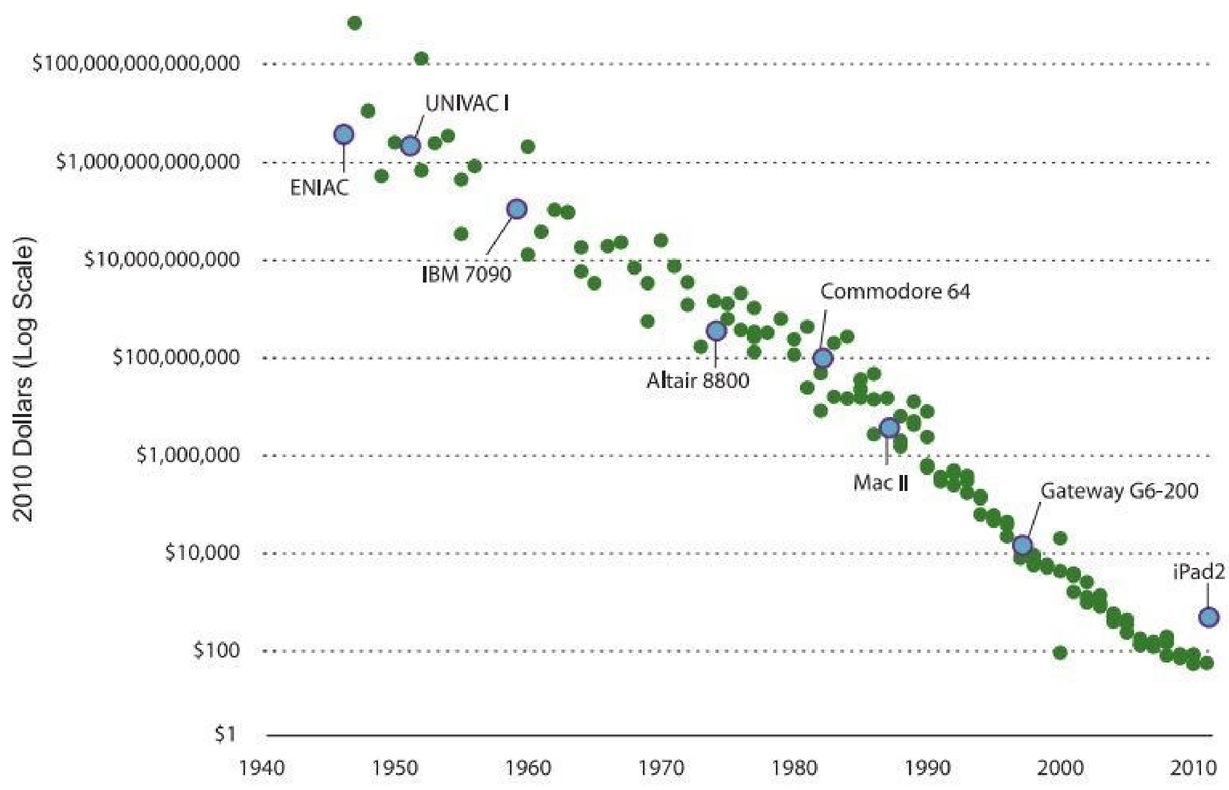
Moore’s law



Gordon Moore

Computing Cost Trends

Cost of Computing Power Equal to an iPad 2

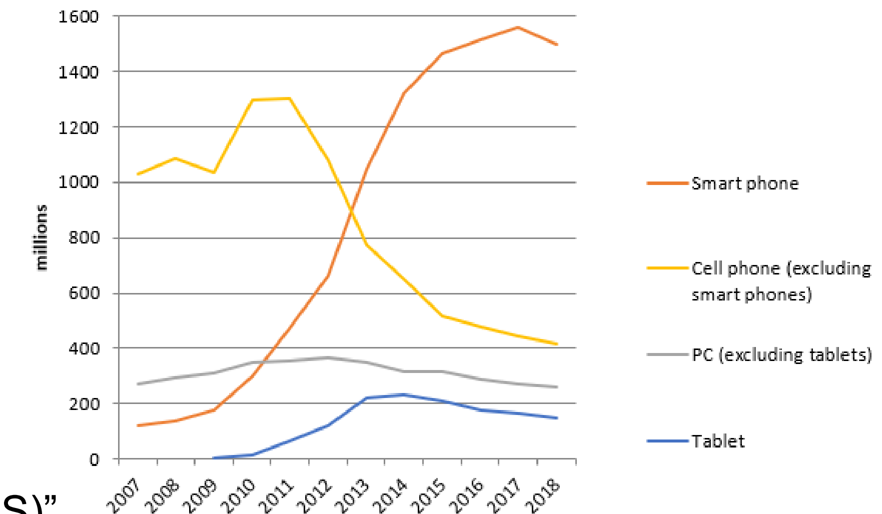


Note: The iPad2 has computing power equal to 1600 million instructions per second (MIPS). Each data point represents the cost of 1600 MIPS of computing power based on the power and price of a specific computing device released that year.

Source: Moravec n.d..

Classes of today computers

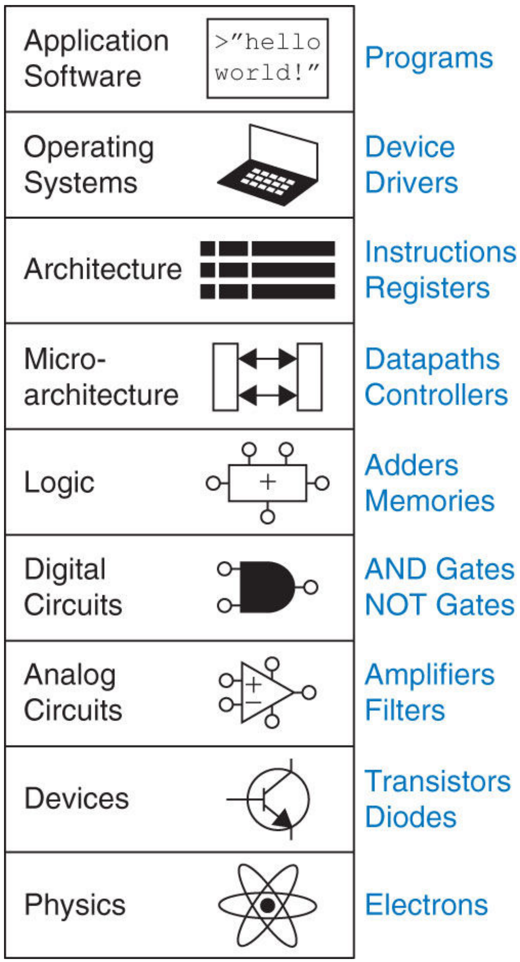
- Personal Mobile Device (PMD)
  - e.g. start phones, tablet computers
  - Emphasis on energy efficiency and real-time
- Desktop Computing
  - Emphasis on price-performance
- Servers
  - Emphasis on availability, scalability, throughput
- Clusters / Warehouse Scale Computers (WSC)
  - Used for “Software as a Service (SaaS)”
  - Emphasis on availability and price-performance
- Internet of Things (IoT)/Embedded Computers
  - Emphasis: price



The Task of a Computer Architect

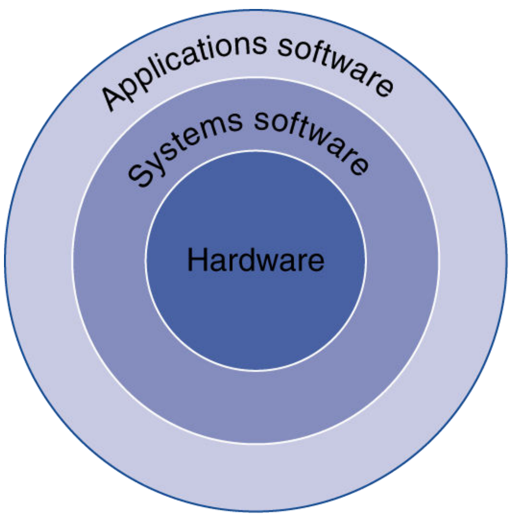
- Computer architects must design a computer to meet functional requirements as well as price, power, and performance goals.
  - inspired by the target market (desktop/server/embedded)
- How? The designer will have to determine:
  - **Instruction set architecture** (ISA): programmer's view of the computer (what the computer does).
  - **Organization:** physical view of the computer (how the ISA is implemented)
  - **Hardware:** implementation of the ISA on specific hardware, including the detailed logic design and the packaging technology.
- Dramatic changes on the computer market makes computer architect's job an extremely complex one.
  - requires familiarity with a very wide range of technologies, from compilers and operating systems to logic design and packaging.

The Art of Managing Complexity



- Abstraction
  - ✓ Hide lower-level implementation detail
- Discipline
  - ✓ Intentionally restrict design choices
  - ✓ Example: digital discipline → discrete voltages instead of continuous
- The Three – **y**'s
  - ✓ Hierarch**y**: A system divided into modules and submodules
  - ✓ Modularit**y**: Having well-defined functions and interfaces
  - ✓ Regularit**y**: Encouraging uniformity, so modules can be easily reused

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
  - ✓ Electronic components organized in accordance with a certain design
  - ✓ Examples of principal components are: 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
  - Human-readable format of instructions
- Machine language
  - Computer-readable format
  - 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)

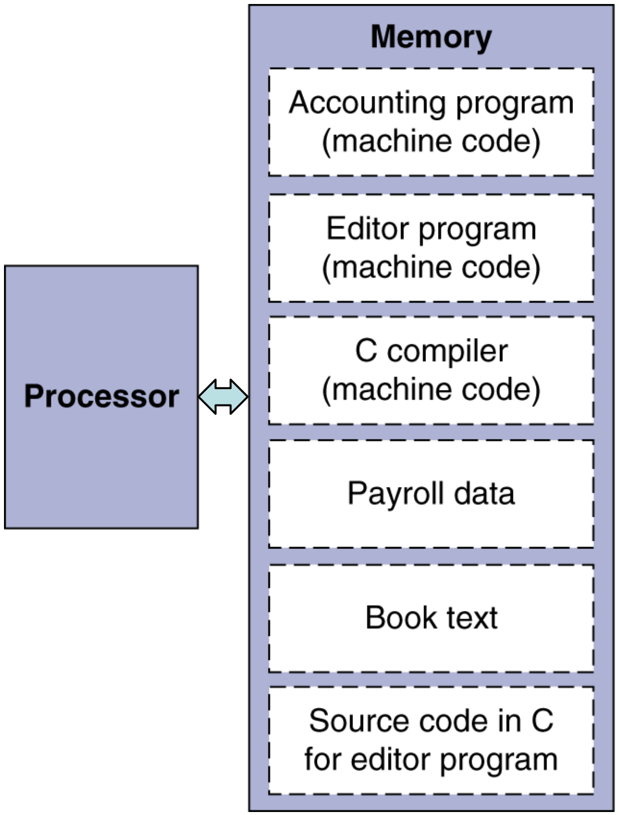
```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Assembler

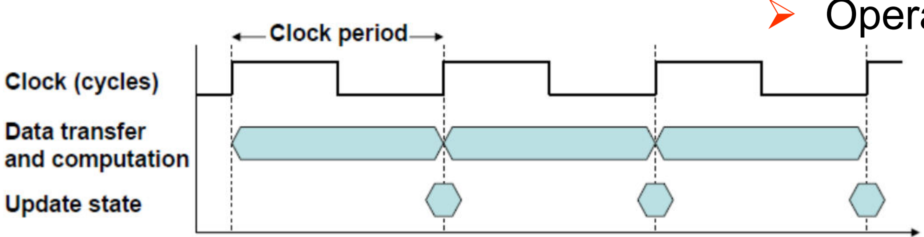
Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

How a stored program is run in computer?



- A program written in HLL is a series of instructions, which will be turn into binary numbers, just like data, and stored in memory.
  - c.f. Harvard architecture
- To run or **execute** the stored program, the processor **fetches** the instructions from memory sequentially.
  - The fetched instructions are then decoded and executed by the digital hardware.
  - Large, complex program execution is a series of memory reads and instruction executions.
  - Operation of HW is governed by a clock



A brief review of binary numbers

- The following slides are for classes who have **not** taken the “Digital Design & Computer Organization” course.
- Those have taken the “Digital Design & Computer Organization” course can jump straight to the section “Computer Performance Measurement & Reporting”

Binary representations of integers

- Natural numbers: unsigned binary

MSB **LSB**

```
0000 0000 0000 0000 0000 0000 0000 0000two = 0ten
0000 0000 0000 0000 0000 0000 0000 0001two = 1ten
0000 0000 0000 0000 0000 0000 0000 0010two = 2ten
. . .                                        . . .
1111 1111 1111 1111 1111 1111 1111 1101two = 4,294,967,293ten
1111 1111 1111 1111 1111 1111 1111 1110two = 4,294,967,294ten
1111 1111 1111 1111 1111 1111 1111 1111two = 4,294,967,295ten
```

- Negative numbers
  - ***Sign-magnitude***: one bit is used for the sign, the remaining represent the magnitude of the number → several disadvantages.
  - ***Two's complement***: the positive half (from 0 to 2^{31} -1) use the same bit pattern as unsigned binary. The negative half begins with $1000\ldots0000_{two}$ representing – 2^{31} and ends with $1111\ldots1111_{two}$ = -1.

```
1000 0000 0000 0000 0000 0000 0000 0000two = -2,147,483,648ten
1000 0000 0000 0000 0000 0000 0000 0001two = -2,147,483,647ten
1000 0000 0000 0000 0000 0000 0000 0010two = -2,147,483,646ten
. . .                                        . . .
1111 1111 1111 1111 1111 1111 1111 1101two = -3ten
1111 1111 1111 1111 1111 1111 1111 1110two = -2ten
1111 1111 1111 1111 1111 1111 1111 1111two = -1ten
```

Binary number conversions

- Given an n-bit two's compliment number

$$x = -x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_1 2^1 + x_0 2^0$$

  - Leading bit is the sign bit (0 → +ve, 1 → -ve)

- Example: 1111 1111 1111 1111 1111 1111 1111 1100_2 = $?_{10}$

  1111 1111 1111 1111 1111 1111 1111 1100_2
  $= -1\times2^{31} + 1\times2^{30} + \ldots + 1\times2^2 + 0\times2^1 + 0\times2^0$
  $= -2{,}147{,}483{,}648 + 2{,}147{,}483{,}644 = -4_{10}$

- Example: $-5_{10} = ?_2$ (two's complement)

  1. Convert magnitude to binary: 5_{10} = 0101
  2. Invert bits: 1010
  3. Add 1 to lsb: + 1

  1011_2

Some useful shortcuts

- Sign extension
  - How does computer convert a two's complement number stored in 8 bit format to its 16 bit equivalent?

$\boxed{11111111}$ 10110011 = −77

- Negation
  - Is there a quick way to negate a two's complement binary number?

starting value	00100100 = +36
step1: reverse the bits (1's complement)	11011011
step 2: add 1 to the value from step 1	+ 1
sum = 2's complement representation	11011100 = −36

Another way to obtain the 2's complement:

Start at the least significant 1

Leave all the 0s to its right unchanged

Complement all the bits to its left

Binary Value

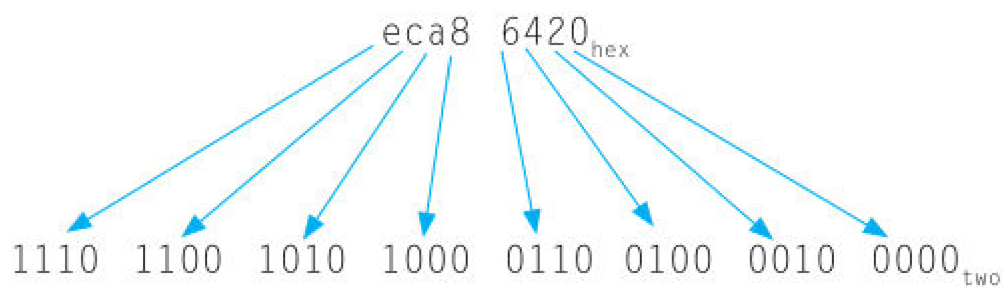
= 00100 $\boxed{1}$ 00 least significant 1

2's Complement

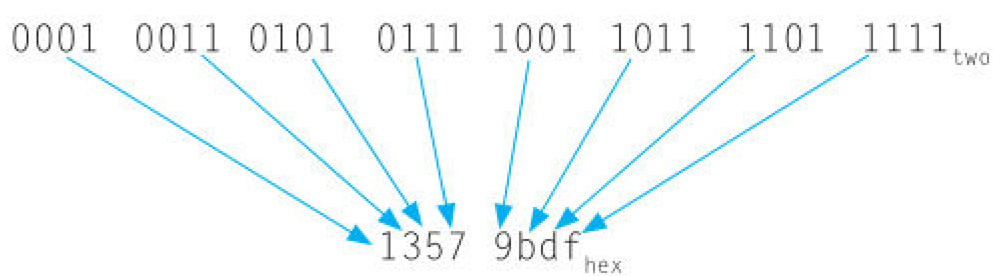
= 11011 $\boxed{1}$ 00

Hexadecimal representation

- Binary numbers are written in long, tedious strings of 0s and 1s
  - Hexadecimal: a higher base that can be easily converted to binary
- Easy binary-hexadecimal conversion



And then the other direction:



Binary representation of fractions

- Binary point is ***implied***
- Fixed point: the number of integer and fraction bits must be agreed upon (fixed) beforehand.
  - Example: What's the binary representation of 6.75_{10} using 4 integer bits and 4 fraction bits?

    01101100, implying:

    0110.1100

    $2^2 + 2^1 + 2^{-1} + 2^{-2} = 6.75$
- Floating point: binary point floats to the right of the MSB
  - Similar to decimal scientific notation: $-2340 = -2.34 \times 10^3$ (normalized, i.e. exactly one non-zero digit appears before the point), or -0.234×10^4 (not normalized)
  - Normalized binary representation: $\pm 1.xxxxxxx_2 \times 2^{yyyy} \rightarrow$ ***significand*** = $\pm 1.xxxxxxx_2$, and ***fraction*** = $xxxxxxx_2$. Notice that the exponent is also binary, i.e. ***exponent*** = $yyyy_2$, but the notation was dropped in the above expression for simplification.

IEEE 754 Floating-Point Format

	single: 8 bits double: 11 bits	single: 23 bits double: 52 bits
S	Exponent	Fraction

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

- S: sign bit (0 → non-negative, 1 → negative)
- Normalize significand: 1.0 ≤ |significand| < 2.0
  - Always has a leading pre-binary-point 1 bit, so no need to represent it explicitly (hidden bit)
  - Significand is Fraction with the "1." restored
- Exponent = actual exponent + Bias (excess representation)
  - Ensures exponent is unsigned
  - Single: Bias = 127; Double: Bias = 1023
- Example: What number is represented by the single-precision float 11000000101000…00?
  - S = 1, Fraction = $01000\ldots00_2$, Exponent = 10000001_2 = 129
  - $x = (-1)^1 \times (1 + .01_2) \times 2^{(129-127)} = (-1) \times 1.25 \times 2^2 = -5.0$

IEEE 754 Ranges

single: 23 bits double: 52 bits	single: 8 bits double: 11 bits	single: 127 double: 1023

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

- Exponents 000…00 and 111…11 are reserved
- **Single-precision range**:
  - Smallest value: Exponent = 00000001 (i.e. actual exponent = 1 – 127 = –126), Fraction: 000…00 → significand = 1.0 → the smallest values are $\pm 1.0 \times 2^{-126} \approx \pm 1.2 \times 10^{-38}$.
  - Largest value: Exponent = 11111110 (i.e. actual exponent = 254 – 127 = +127), Fraction: 111…11 → significand ≈ 2.0 → the largest values are $\pm 2.0 \times 2^{+127} \approx \pm 3.4 \times 10^{+38}$.
- **Double-precision range**:
  - Smallest value: Exponent = 00000000001 (i.e. actual exponent = 1 – 1023 = –1022), Fraction: 000…00 → significand = 1.0 → the smallest values are $\pm 1.0 \times 2^{-1022} \approx \pm 2.2 \times 10^{-308}$.
  - Largest value: Exponent = 11111111110 (i.e. actual exponent = 2046 – 1023 = +1023), Fraction: 111…11 → significand ≈ 2.0 → the largest values are $\pm 2.0 \times 2^{+1023} \approx \pm 1.8 \times 10^{+308}$.

Practice

- Convert –0.8125 to binary in single and double precision.
- **Solution**:
  - Fraction bits can be obtained using multiplication by 2
    - $0.8125 \times 2 = 1.625$
    - $0.625 \times 2 = 1.25$
    - $0.25 \times 2 = 0.5$
    - $0.5 \times 2 = 1.0$
    - Stop when fractional part is 0

    $\boxed{0.8125 = (0.1101)_2}$
  - Fraction $= (0.1101)_2 = (1.101)_2 \times 2^{-1}$ (Normalized)
  - Exponent $= -1 + \text{Bias} = 126$ (single precision) and 1022 (double)

Single Precision

1	0	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Double Precision

1	0	1	1	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Summary

- Fundamental concepts in computer architechure
  - Computer definition
  - Computer evolution
  - Technology and cost trends
  - Classes of modern computers
- Computer architecture
  - Abstract layers
  - ISA and computer organization
  - Stored program concept
- Binary representations of numbers
  - Unsigned and signed integers.
  - IEEE 754 floating point format
- Next week: performance measurement and reporting.