Computer Architecture

Ch1 – Introduction

Nguyễn Quốc Đính, FIT – IUH

HCMC, Aug 2015

Information

- Me - Nguyễn Quốc Đính
- Contact me via nqdinh@hui.edu.vn
- Site: https://github.com/kientrucmaytinh/kientrucmaytinh-dhth10
  - Download lecture slides
  - News
- Office hours: H2.1, 1pm – 4pm Monday.

Reading

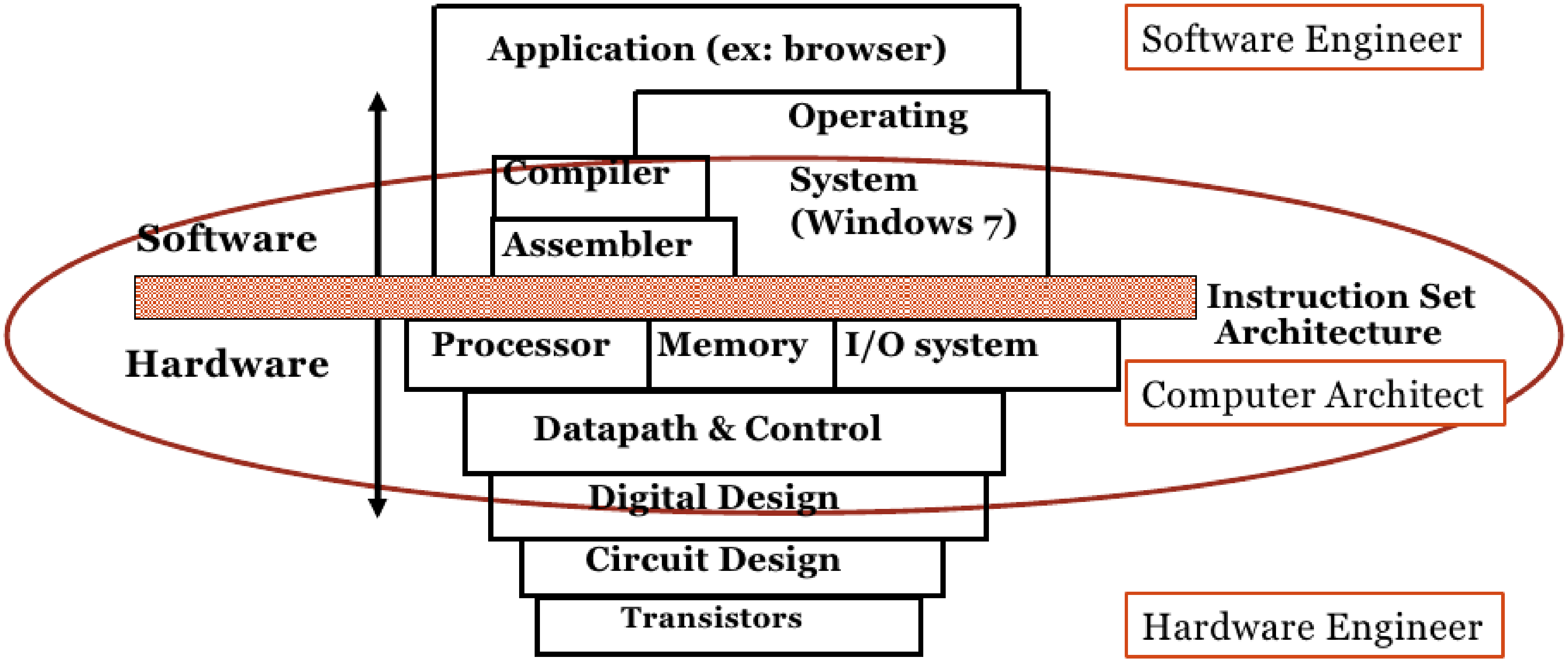
- Textbook (must have):
  - Andrew S. Tanenbaum, *Structured Computer Organization* (5th Edition), Prentice Hall, 2006
- More
  - John L. Hennessy, and David A. Patterson, *Computer Architecture : A Quantitative Approach* (5th Edition), The Morgan Kaufmann Series in Computer Architecture and Design, 2011

Tentative topics

1. Introduction (Chapter 1)
2. Overview of computer systems organization (Chapter 2)
3. The digital logic level (Chapter 3)
4. The instruction set architecture level (Chapter 5)
5. Assembly language: linking and loading macros (Chapter 7)
6. Advanced computer architectures (Chapter 8)

Course Goals

Computer Structure



Coordination many levels of abstraction

Copied from UC Berkeley CS61C

Levels of Abstraction

Application

High Level Languages

Assembly language

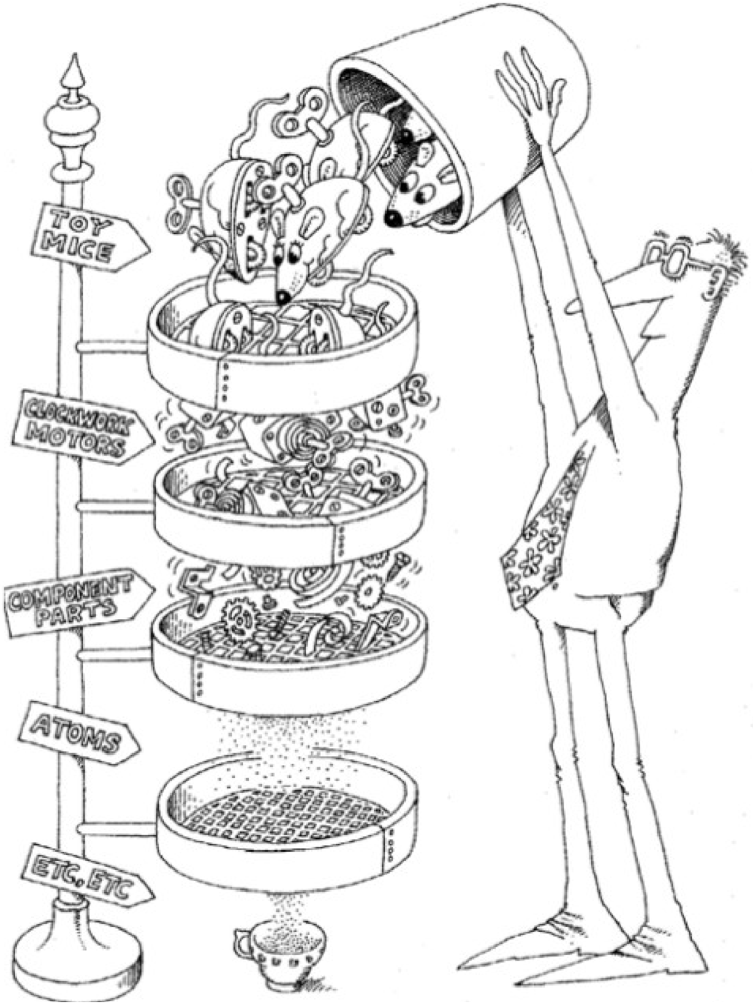
Architecture

Gate-level logic

Electronics

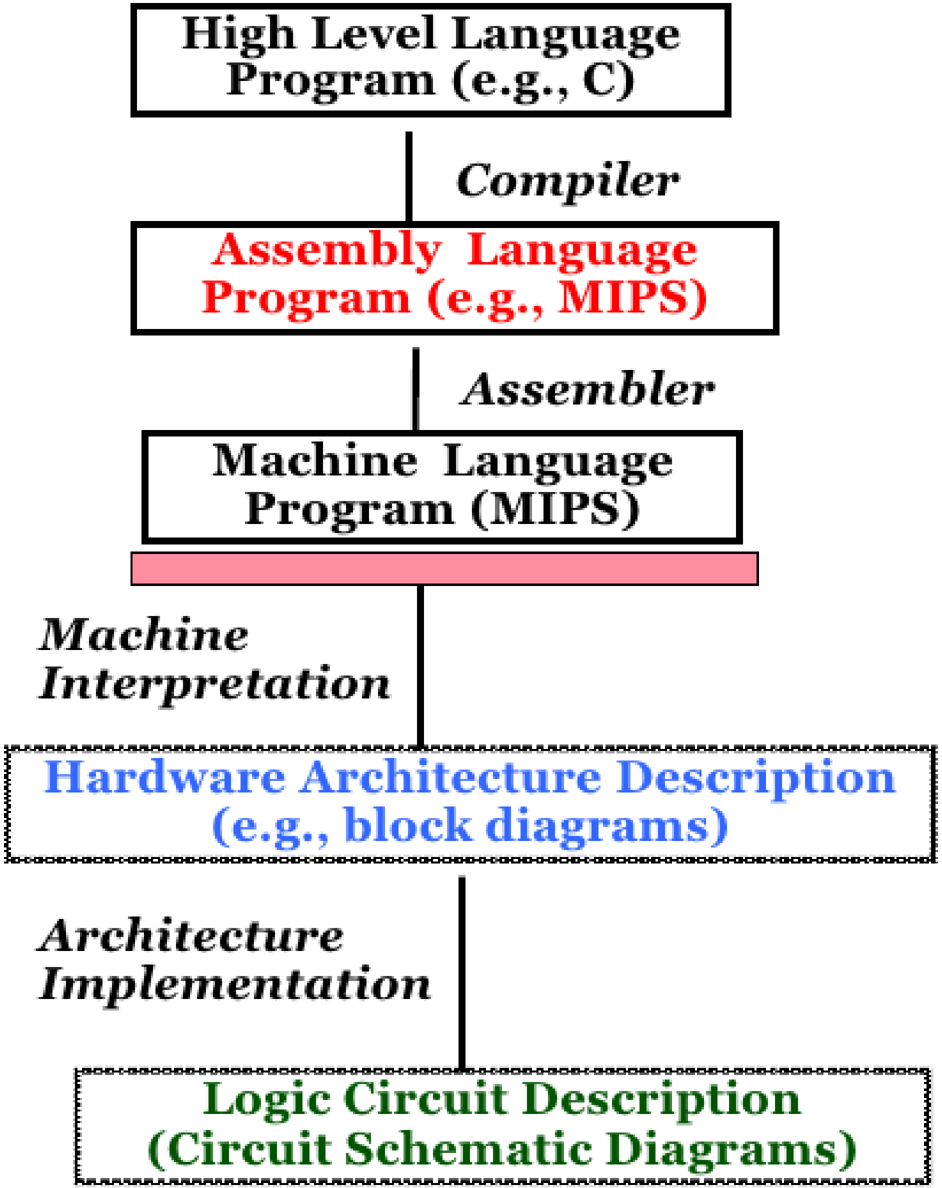
Semiconductor

Real world of physics



Grady Booch et al, Object oriented analysis and design with applications, Addison Wesley, 2007

Level of Representation



High Level Language Program (e.g., C)

Compiler

Assembly Language Program (e.g., MIPS)

Assembler

Machine Language Program (MIPS)

Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)

Architecture Implementation

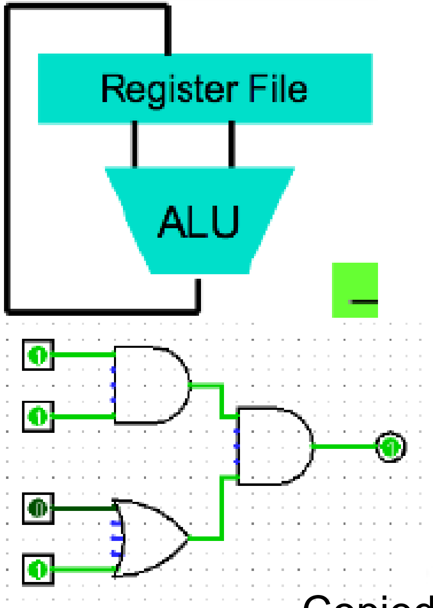
Logic Circuit Description (Circuit Schematic Diagrams)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

Anything can be represented as a *number*, i.e., data or instructions

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```



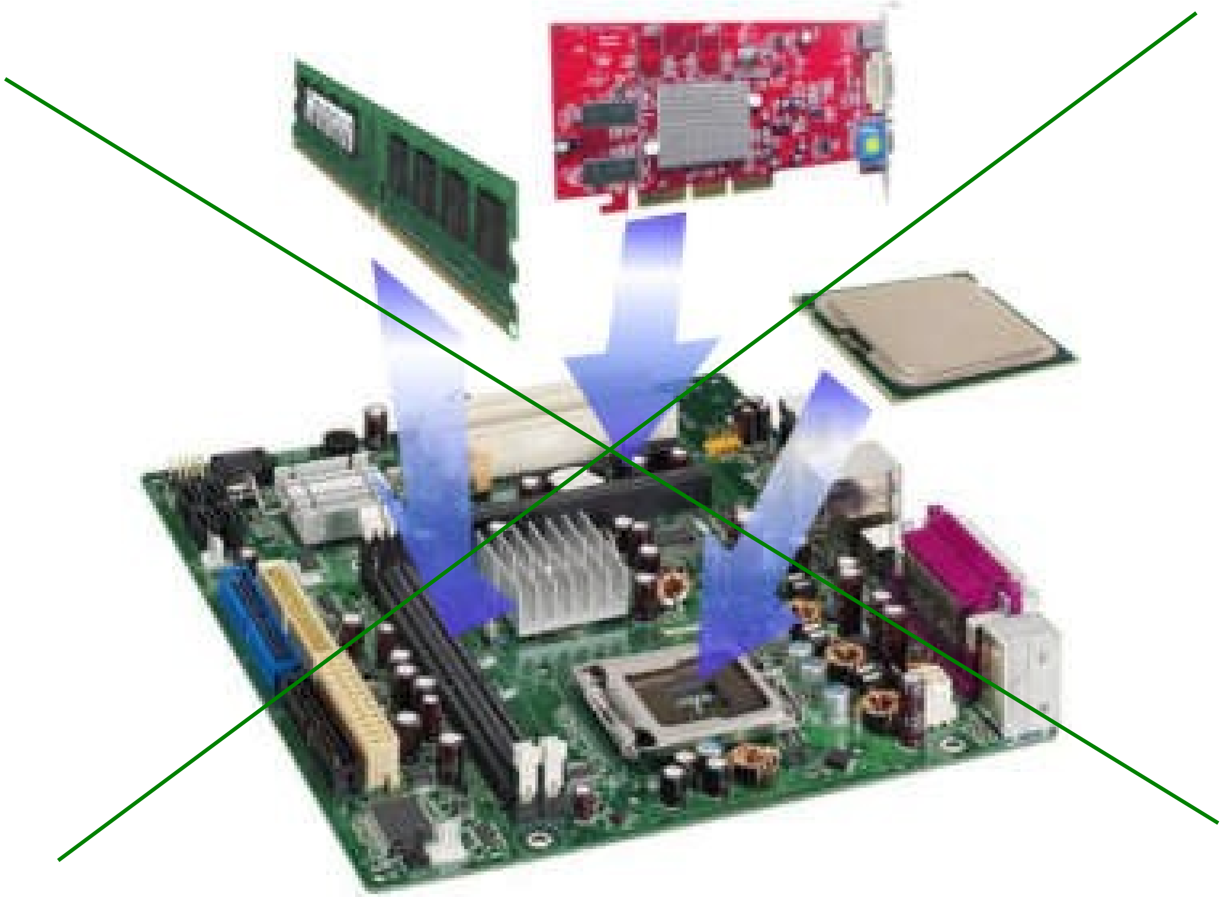
Copied from UC Berkeley CS61C

Course Goals

- Learn to think about the machine from the
  - ISA level
  - Assembly language level
  - Logic level
- Understand relationships and interactions between levels
- Be able to understand key computer organization concepts.

Course Goals (cont'd)

- How to assemble a PC? Boring



Picture from intelligenceunited.com

Machine and Assembly Language

Op Code Data reg. #5 Data reg. #4

MC68000 Assembly Language	Machine Language
MOVE.W D4, D5	0011 101 000 000 100
ADDI.W #9, D2	00000001 01 000 010 0000 0000 0000 1001

Op Code

Who Uses ASM Language?

- Machine designers
  - Implement and tradeoffs of instruction functionality
- Compiler writers
  - Generate machine language from a HLL
- The writer of time or space critical code
  - Performance goals may force program specific optimizations of the assembly language (e.g. DSP, encryption)
- Embedded processor programmers
  - I/O devices

Architecture's View

- Functional building blocks and their interconnections
  - ISA (interface)
  - CPU and memory
  - Buses

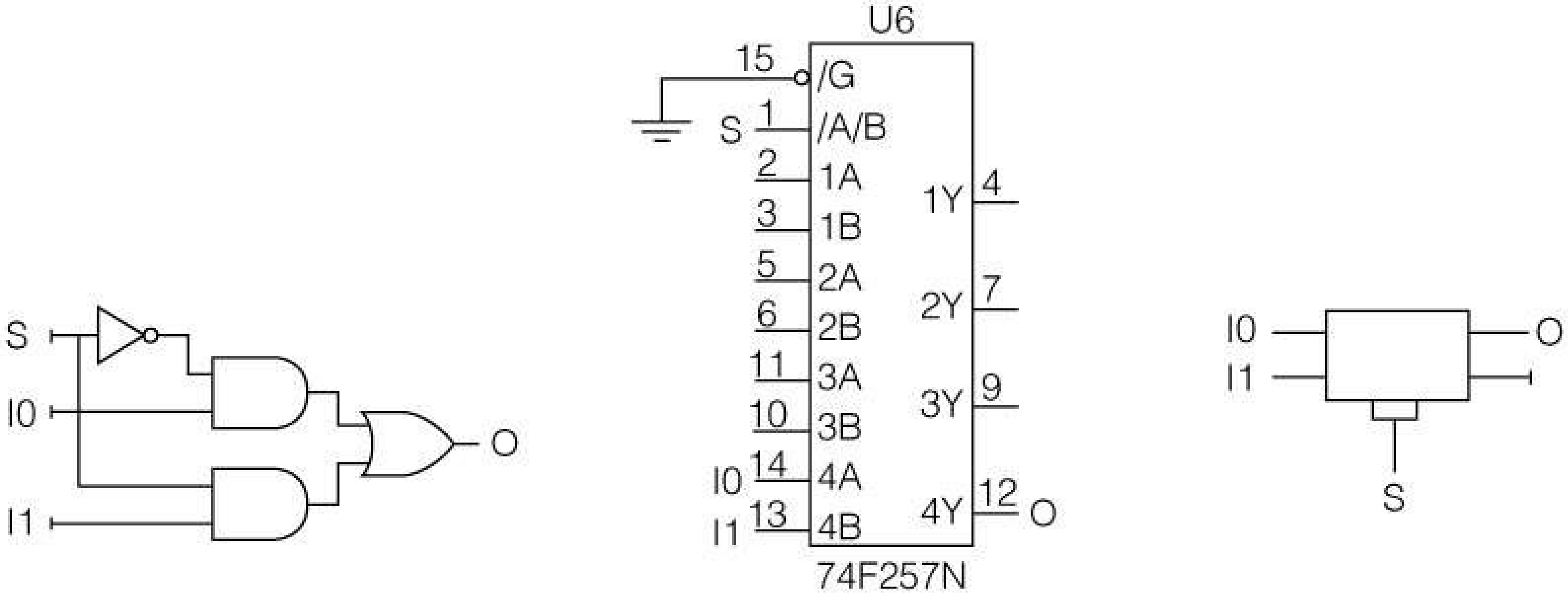
- Tools
  - Benchmark programs
  - Simulators, emulators
  - Register transfer notation

Logic

- Computer aided design tools
  - Logic design and simulation packages
  - Printed circuit layout tools
  - IC (integrated circuit) design and layout tools
- Logic analyzers and oscilloscopes
- Hardware development system

Implementation Domain

- Well defined building blocks are used repeatedly



(a) Abstract view of Boolean logic

(b) TTL implementation domain

(c) Optical switch implementation

Levels of Abstraction (again)

Application

High Level Languages

Assembly language

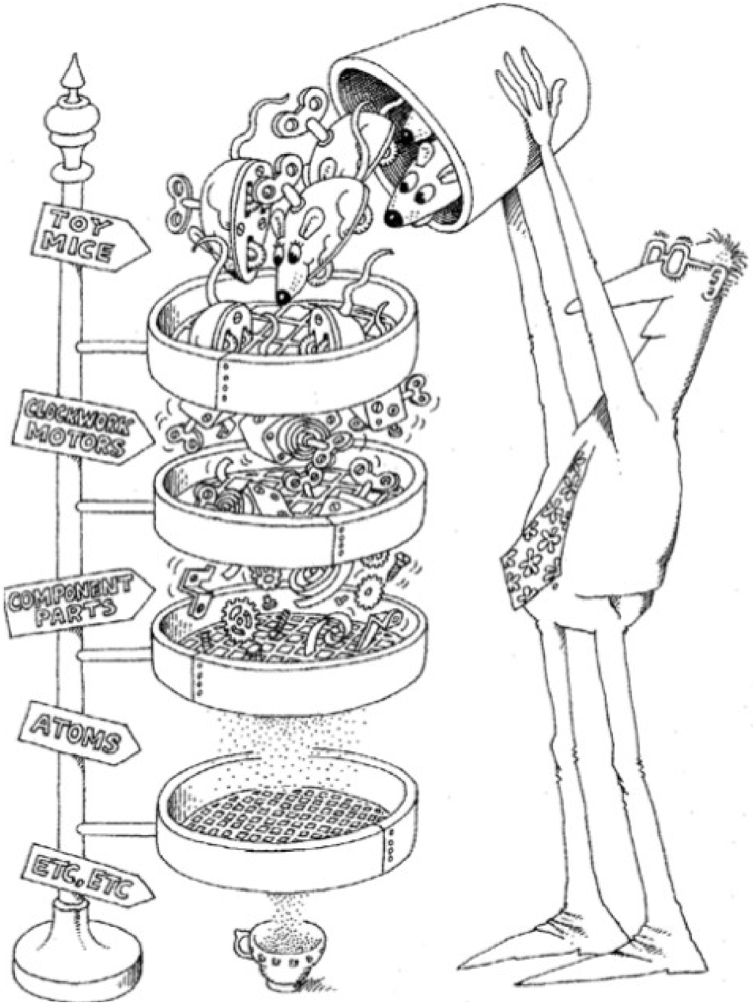
Architecture

Gate-level logic

Electronics

Semiconductor

Real world of physics



Grady Booch et al, Object oriented analysis and design with applications, Addison Wesley, 2007

A brief overview of computer history

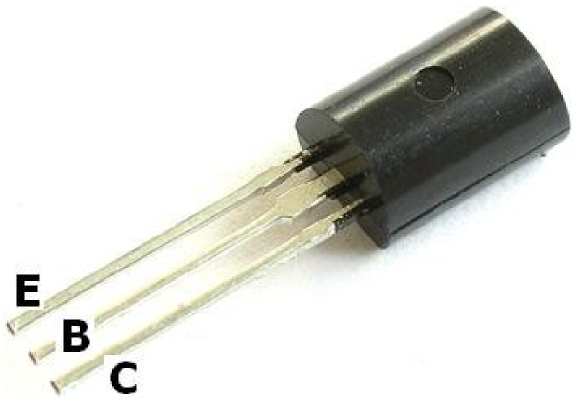
Computer Generations

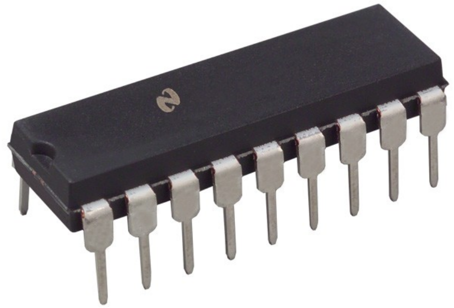
Expensive, large, unreliable

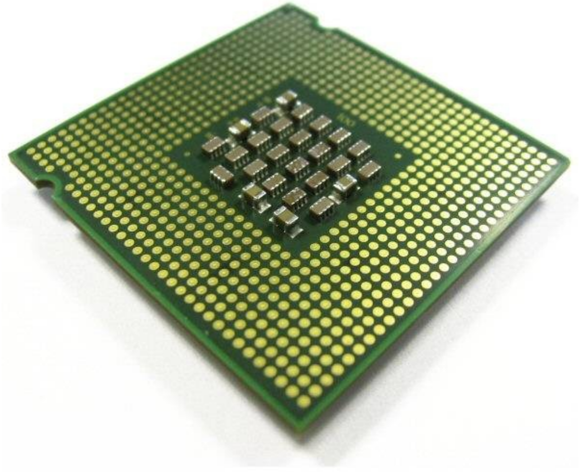
- Zeroth: Mechanical Computers (1642 – 1945)
- First: Vacuum Tubes (1945 – 1955)
- Second: Transistors (1955 – 1965)
- Third: Integrated Circuits (1965 – 1980)
- Fourth: Very Large Scale Integration (1980 – ?)

Cheaper, smaller, reliable

Computer Generations

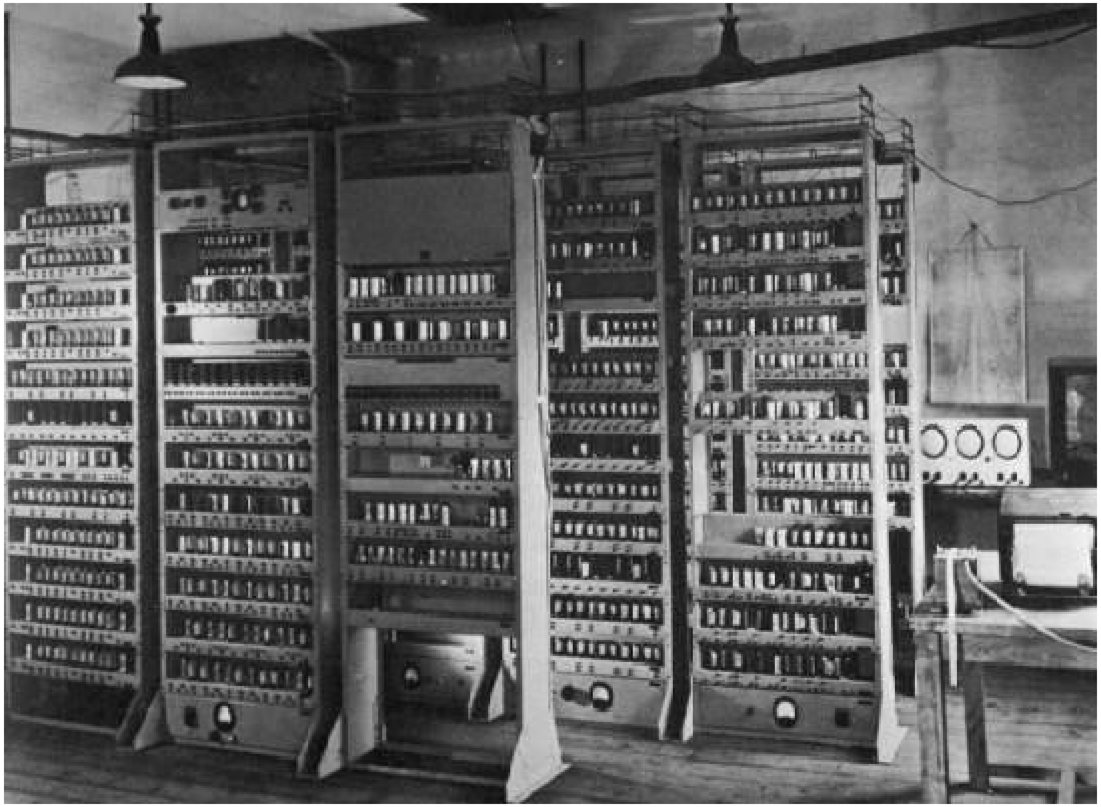






Vacuum Tube (45-55)

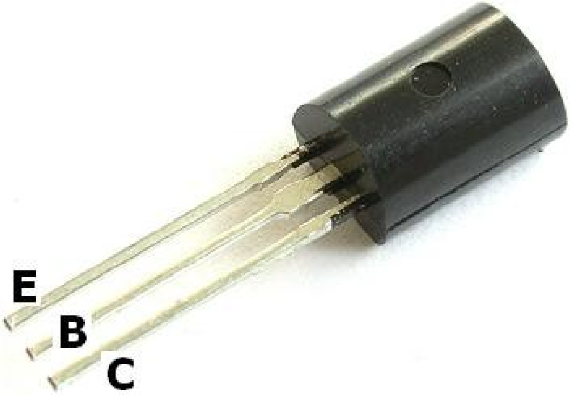
- ENIAC (WWII)
  - Eckhert and Mauchly (UPenn)
  - 18,000 vacuum tubes, 1500 electronic relays
  - hardwired programs
- Stored program concept
  - Von Neumann
- No OS, no programming language



EDSAC, University of Cambridge, UK, 1949

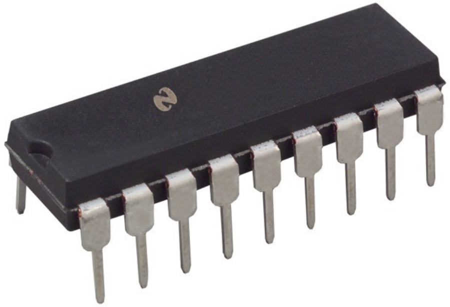
Transistor (55-65)

- 1/10 size of vacuum tube
- Lower operating temperature → longer life
- Programming with ASM, Fortran, Cobol



IC (65-80)

- Kilby of Texas Instruments
  - Idea to build entire circuits on semiconducting material
- 10 – 10,000 components per chip
- Computers became smaller
- Programming with Algol, Pascal, C.
  - Unix born

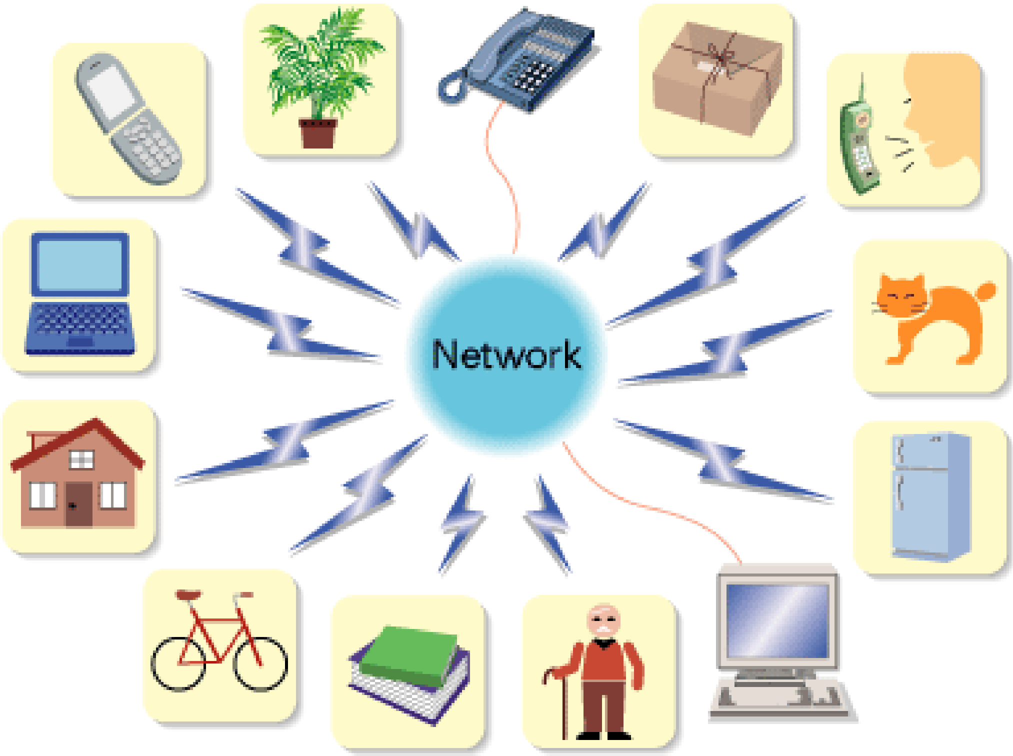


VLSI and Ultra VLSI

- Million to trillion transistors in one chip
- SoC

Next generation?

- Invisible computers
  - Ubiquitous computing
  - Perversive computing
- Computer everywhere



Picture from quantumcinema.blogspot.com

Classes of Computing

- Desktop
  - Req: Low cost, high performance
- Server
  - Req: Stable and safety
- Super computer
  - Req: High performance
- Embedded Processor
  - Req: variety (performance, cost, consuming energy, resistance to failure …)

Next ...

- General purpose computer
- Read Chapter II <<computer system organization>>

Copyright note

- Some slides are adapted from Dr. Michele Co, University of Virginia.
- Contents taken from Andrew S. Tanenbaum. Structured Computer Organization, 5th.