

## Computer History

- 1642: Pascal built add/subtract machine
- 1672: Leibniz built add/sub/mul/div machine
- 1822: Babbage built a **difference machine** with punch card output.  
Later he designs a general purpose **analytical engine** (1000 words of 50 decimal digits store), programmable in simple assembly language.
- 1936: Zuse (Germany) Automatic calculation machines using electromagnetic relays.

## Computer History (cont.)

- 1943: Colossus (UK, Alan Turing)
- 1946: ENIAC (Electronic Numerical Integrator and Computer)
- 1949: EDSAC (von Neumann architecture)
- 1960: PDP-1 (Digital)
- 1974: Intel 8080 (general purpose 8-bit microprocessor)
- 1981: IBM Personal Computer (PC)
- 1984: Apple Macintosh (Window system, mouse)

## Computer organisation

- Computer technology
- Architectures
  - Central processing unit (CPU)
  - Arithmetic logical unit (ALU)
  - Registers, memory
- References:
  - Chapter 1 and 3.1 in Bryant and O'Hallaron
  - Chapter 1 and 2 in Tanenbaum
  - Related links on web site (CPU info center)

## Computer technology

- First generation (1945 – 1955)  
Vacuum tubes and electromagnetic relays
- Second generation (1955 – 1965)  
Transistors
- Third generation (1965 – 1980)  
Integrated circuits
- Fourth generation (1980 – ?)  
Very large scale integration (VLSI)
- Moore's law: Number of transistors on a chip doubles every 18 months

## Moore's law and von Neumann architecture

- Page 127 Bryant  
Moore's law
- Figure 1-5 in Tanenbaum (page 18)  
The original *von Neumann* architecture
- Figure 1-2 in Tanenbaum (page 5)  
A multilevel machine abstraction
- Figure 1-4 in Bryant (page 7)  
Organisation of a simple computer

## Computer architecture

- **CPU (central processing unit):** Executes programs stored in memory
- **Bus:** Communication links between the main components of a computer (many parallel wires to transmit a word or more)
- **Main memory:** Contains programs and data. Volatile but fast: Random Access Memory (RAM)
- **Secondary storage:** Used to permanently store data and programs (hard disc, flash card, etc). Slower than RAM, but larger.
- **Input/Output (I/O) devices:** To exchange information with other computers or people (e.g. monitor, keyboard, mouse, printer, network, CD-ROM, etc)

## The central processing unit (CPU)

- **Control unit:** Responsible for fetching instructions from memory and initial decoding
- **Arithmetic Logic Unit (ALU):** Performs operations such as addition, multiplication, logical AND, logical OR, etc.
- **Registers:** Small, very fast memory located in the CPU, used to store control information and temporary results
  - Program counter (PC): Address in memory of next instruction to be fetched and executed
  - Instruction register (IR): Current instruction
  - Status register (SR): Status after operation (e.g. overflow, division-by-zero, etc.)
  - Other registers (e.g. accumulator, index register, data register, etc.)

## CPU architecture

- Figure 1 in the PeANUt specification  
Architecture of the PeANUt microprocessor
- Figure 1-5, 1-6, 1-7 in Bryant (page 10/11)  
Communication events in a typical computer

## The arithmetic logic unit (ALU)

- Where the *computation* is done
  - Input(s) given by the control unit
  - Performs operations (arithmetic or logical) on these inputs, as requested by the control unit
  - Produces a result, which the control unit stores back to a register (or memory)
- The **data path** – the heart of the CPU – essentially defines what the machine can do

## Executing machine instructions

- Instructions (a program) are stored in memory at consecutive addresses
- Usually instructions are executed in sequence, with the machine stepping through in the order they are stored in memory
- The control unit requests the next instruction from memory according to the value of the **program counter**
- The process of fetching and executing an instruction is known as the *fetch-decode-execute* cycle. It is central to the operation of all CPUs.

## Fetch-decode-execute cycle

1. Fetch next instruction from memory into **instruction register (IR)**
2. Change the **program counter (PC)** to point to the next instruction
3. Determine the type of instruction fetched
4. If the instruction refers to data in memory, determine the address
5. Fetch the data (if any) into registers
6. Execute the instruction
7. Handle exceptions
8. Store the results into registers (or memory)
9. Go to step 1.

## Example

### Registers:

ACCU:

PC:

IR:

MAR:

MDR:

### Main memory

Address Instruction

...

...

1A

load  $\#3_{10}$ 

1B

add 2E

1C

store 2E

...

...

 $2E_{16}$  $4_{10}$ Result at Memory  $2E_{16}$  is 7

## Instruction sets

- The collection of all instructions of the CPU available to the programmer is called **instruction set**
- Usually between 20 and 300 instructions
- Typical instructions include:
  - load, store (data from/to memory)
  - add, subtract, multiply, divide (for integer and floating-point numbers)
  - and, or, not, xor (logical operators)
  - jumps (e.g. go to) and branch (e.g. if A then go to...)
  - call, return (for subroutine or procedure calls)

## Complex Instruction Set Computer (CISC)

- Has a large number of instructions, including many highly specialised ones, e.g.
  - Supports many different address modes
  - Supporting binary and decimal arithmetic operations
  - Supporting generic loop operations as well as for **for**, **while** and **repeat** loops
  - Supporting **if** and **then**
- 70s to early 90s processors, incl. Intel x86, Motorola 68k
- Instructions require a lot of decoding (often done via microprogramming), and hence take a long time to execute (many clock cycles)

## Reduced Instruction Set Computer (RISC)

- A smaller instruction set, providing well chosen instructions to support the most common operations (incl. **if** and **call**)
- Basic load/store, no complex memory access instructions
- Fewer simpler instructions means less decoding and results in quicker execution (often one clock cycle!)
- Complex and/or uncommon operations have to be written as sequence of instructions (e.g. multiplication in early RISC processors). But quicker execution makes it worthwhile
- Idea developed in the 80s, and RISC dominated the high-end workstation market in the 90s
- Includes SPARC (SUN), MIPS, PowerPC (Macintosh), Intel i860

## Modern CPU architectures

- More transistors on a chip enables complex RISC (hybrids like Pentium)
- On-chip cache memory (Level 1 cache)
- Pipelining (overlapping of fetch-decode-execute cycle)
- Multiple ALUs and floating-point units (FPUs)  
(on-chip parallelism)
- Superscalar architectures