Computer Science Basics Architecture

Fall Term 2025/2026

Emmanuel Benoist | BFH-TI

Architecture of computers

- Last Week
- What is a computer?
- Memory
- Random Access Memory
- Central Processing Unit - CPU
- Machine instructions
- Operating System

Last Week

Features of a computer

What is a computer ?

- A computer can make calculations.
- Contains Data and a Program
- Data and Program are in memory (von Neuman Architecure).
- Computers are made of transistors.

How does it work?

- Programs must be started and managed by an Operating System (OS).
- Files must be stored on permanent devices (tapes, floppy, disks, flash memory, …).

What is a computer?

Question

- **What is a computer made of?**
- **Mainly transistors**

Memory

Switches, Transitors and Memory

- **If you switch light**
  - By entering the room you switch the light on
  - By leaving the room you do not switch it off
  - When you come back: light is still on
  - It is Memory: it can be `On` or `Off`
- **The switch remembers the last command**
  - Until you change it, it remains
  - Change = “overwrite” the value
- **Light switch are mechanical**
  - Babage computer was also mechanical

Memory

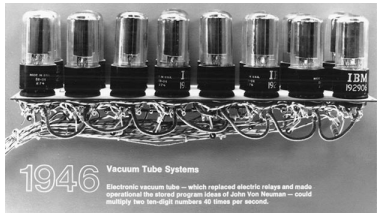
- **Definition**
  - container for alterable patterns that retain an entered pattern until someone or something alters the pattern
- **In computers**
  - The patterns are electrical: there is a voltage or there is no voltage
- **In history**
  - Memeory was physical (made out of cams for babbage)
  - Could have been hydraulic (high / low position),
  - …

Memory out of electricity

- **Problems with light switches**
  - They require fingers to set them
  - Computer memory should be operated by the same force it controls
  - So memory can be passed to other memory storage locations
  - Such a switch is called a `relay`
- **How do relays work**
  - You can flip a relay by feeding it a pulse of electricity
  - Computers have been made out of relays
  - It was a long time ago
  - They were not powerfull : relay is about the size of an ice cube

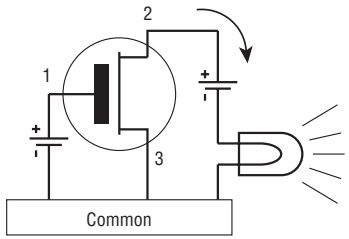
Memory out of electronics

- **Memory is made out of transistor switches**
  - Early in history, they were made out of vacuum tube switches



- **Transistors**
  - Use properties of silicon to act as switches

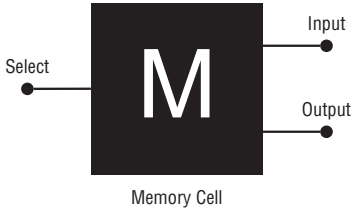
Transistor



Transistor Switch

- **Field effect transistor**
  - When Electric voltage is applied to pin `1`
  - Current flows between pin `2` and `3`
  - When voltage is removed from pin `1`
  - current ceases to flow between pins `2` and `3`

Memory Cell



- **In real life**
  - other components needed (diodes, capacitors, …),
- **Voltage**
  - If you put a tiny voltage on select, a voltage will appear and remain on its ouput pin
  - If you select and change the input, it changes the output voltage

Store more than One bit

- **One Chip contained one transistor**
  - Together with other components: one bit of memory
- **Ingenieurs crisscrossed the chip into four transtors (+ components)**
  - Four bits in one chip
- **Shrinking process begins**
  - 8 bits, 16 bits, 256 bits, 1kbits (1024 bits)
  - …
  - 4 GB (Bytes not bits)

Random Access Memory

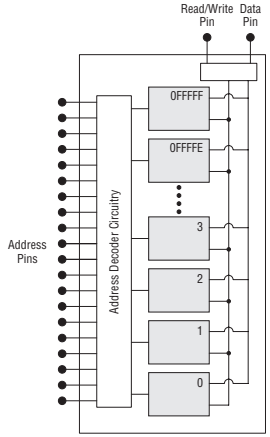
Random Access

- **Randomness of memory?**
  - Access any memory cell independently
  - Give an address : get the value
  - No need to read all memory
- **Opposite**
  - Serial access memory
  - Tapes
  - Hard disk

Random Access Memory

- **In memory, each bit is stored in its own memory cell**
  - Analog to the one seen previously
- **Each cell has one unique number: Its Address**
  - Addresses start by 0, 1, 2, 3,
  - …
  - … $n-1$ (where n is the number of memory cells)
- **The chip has pins called address pins**
  - Used to input the address of the desired cell
- **It has two other pins**
  - One read / write pin
  - One data pin

RAM



Access 1 bit of RAM

- **Address pins**
  - You put voltage on some pins (5V for instance) and not on some other
  - It represents a number in binary (where 0V is 0 and 5V represents 1)
  - The cell with the given address is activated
- **Depending on status of Data pin**
  - Voltage / no voltage
  - The system is read or write
- **You can read or write from the read/write pin**
  - Read the voltage (0 or 5V) (in read mode)
  - Set the voltage to overwrite the value (in write mode)

Memory is more than 1bit

- **Other units**
  - A bit is a single binary digit 0 or 1
  - A byte is 8 bits side by side
  - A word is 2 bytes side by side
  - A double word is 2 words side by side
  - A quadword is 2 double words side by side
- Most computers process information in quadwords (64 bits)

Access 8 bits

- **We have 8 memory chips**
  - The address bits are related
  - The same address is sent to 8 chips
- **Each chip delivers one bit**
  - The bit can be read or write
  - One bit at a time
- **8 chips: 8 bits**

RAM

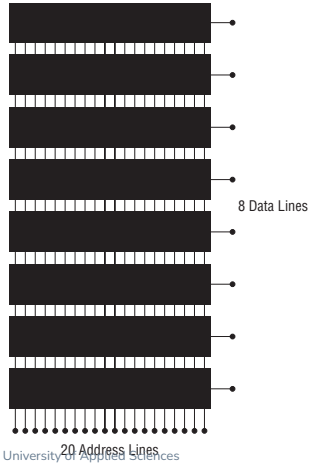


Figure 3-3: A 1-megabyte memory system

RAM is somehow more complex

- **So simple architecture no longer exists**
  - There are many different ways of distributing chips
- **Chips storing 1,2,3,4,8 bits per address are relatively common**
  -
- **Desing of memory is a subdiscipline in electrical engineering**
  -
- **Today memory is provided in Dual Inline Memory Modules (DIMMs)**

Accessing memory

- **Number of bits accessed**
  - In 8 bits computers: memory returns a byte
  - In 16 bits computers : memory returns a word
  - In 32 bits computers: memory returns a double word
  - In 64 bits computers: memory returns a quad word
- **Convention**
  - Regardless on the size of the return value
  - Every byte has its own address
- **Example**
  - If you access the memory at address 123 on a 32 bits architecture,
  - you receive the bytes having the addresses 123, 124, 125, and 126,
  - If you access memory at address 126
  - You receive bytes with the addresses 126, 127, 128, 129
- **Any byte in memory can be addressed!**

Central Processing Unit - CPU

CPU

- **Central Processing Unit**
  - "does the work"
  - Arithmetic computation, reading writing in memory,
- **Peripherals also work**
  - Network or USB ports
  - Video display board (using Graphical Processing Unit - GPU and memory)
- **Intel is the market leader**
  - AMD is a big competitor
  - CPU's are standardized
  - Commands to CPU are the same for all manufacturers

Talking to Memory

- **CPU has to communicate with memory**
- **Address pins**
  - CPU has pins for memory addresses
  - They are connected to pins of memory chips
- **Data lines**
  - Are connected to Data output of the memory
  - Are used to read (or write) the memory
- **The CPU writes an address on the bus, the Memory answers the value on data lines**
  - (Reality is a little bit more complicated, but the principle remains true)

The CPU and Memory

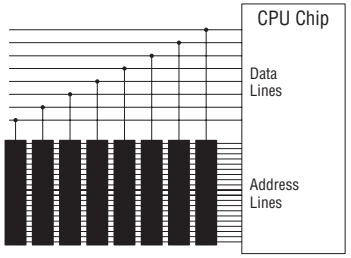
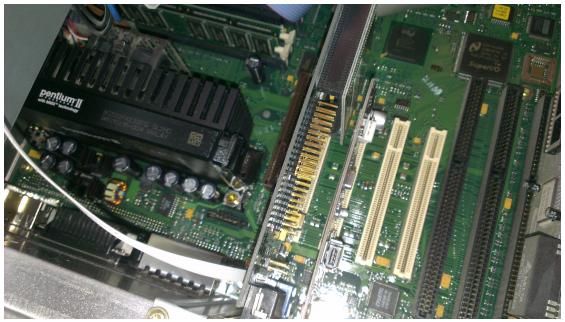


Figure 3-4: The CPU and memory

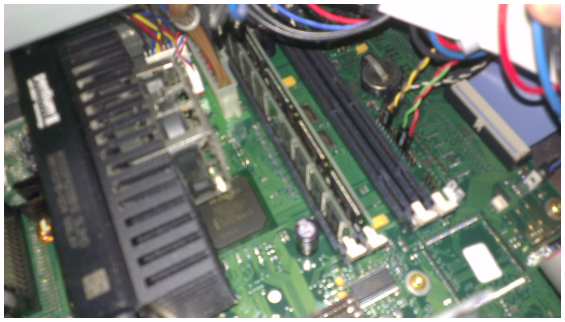
Data Bus

- **Used to communicate with *Peripherals***
  - Video display boards
  - Disk drives
  - USB ports
  - Network ports
  - *source and/or destination for information*
- **Peripherals "talk" to the CPU**
  - Done using electrical connections linking all address pins and data pins
  - It links all the peripherals and the CPU together
  - It is called *data bus*
- **Communication using data bus**
  - An address is placed on the bus, followed by some data
  - Special signals go out on the bus to indicate if target is memory or one peripheral
  - Address of peripheral = *I/O address* (unlike *memory address*)

A computer



CPU and Memory



Registers

- **Every CPU contains data storage: *Registers***
  - Very few storage
  - Very fast compared to memory access
- **Workbench**
  - Where the CPU places elements to work on
- **Example: addition**
  - You place one opperand in one register
  - The other operand in another register
  - The sum replaces one of the register
  - Sum can be placed in another register
  - for being used in another operation
  - or stored in memory for a later use

Registers

- **Registers do not have addresses**
  - Like memory (address 00558D2H)
- **Registers have names**
  - EAX, EDI, R9
  - Names are the results of a long history
  - From 8 bits CPU to 64 bits CPU's
- **Some registers have special powers**
  - Registers can be used for storing information
  - some registers are used for a special function
- **Registers are presented in detail next week**

Machine instructions

Data Bus

- **Information enters through a network port**
  - Assemble the signal into bytes of data
  - Information is placed on the data bus
- **CPU reads the information**
  - Processes it in other ways
  - place it back on the data bus
- **Display board retrieves bytes from the data bus**
  - writes into the video memory
  - one can see the information on screen

What is a Program

- **A program is just data**
  - Stored in memory
  - Not different to other data
  - Machine instructions are just bytes
- **CPU reads from memory**
  - CPU asks the memory for a given address (it knows it is an instruction)
  - An instruction is placed on the bus
  - CPU reads it: 0B6H 073H for instance
  - Means load the value 73H into register DH
- **Each instruction tells the CPU to perform one generally small and limited task**
  - Writing the sequence of instruction is Assembly language programming
- **The CPU has a special counter (register) pointing to the next instruction**
- **The loading is automatic**

Machine instruction

- **Machine instruction is binary code**
  - It is not numbers (numbers are used to represent them)
  - The binary instructions triger the transistors: switches in the CPU
- **Instruction 01000000B (40H)**
  - Directs CPU to add 1 to the value stored in register AX
  - The sum is placed back in AX
- **The 8 bits are loaded in the CPU**
  - They push 8 switches
  - which push dozens of switches
  - and finally push thousands of switches
  - In the end, the value contained in register AX is incremented

Changing course

- **CPU normally picks up the next comming instruction**
  - the address of the current instruction is automatically incremented
- **Some instructions change the order of the instructions**
- **The CPU can jump to another instruction**
  - Anywhere in the program
- **Jump the next instruction**
  - after a test
- **Go back to the start of a program**
  - or a sub-program

Operating System

Operating System

- **Responsible for basic functions**
  - Storage of data on disk
  - Pick input from keyboard
  - Display information on screen
- **Nowadays, much more complicated**
  - Memory management
  - Multi-user
  - Multi-task
  - …
- **Just another program**
  - Written also in memory
  - Contains CPU instructions

Very old OS: CP/M

- **For desktop microcomputers in 1979**
  - "State of the art" in 1979
- **OS is loaded at startup**
- **If you type the name of a program**
  - CP/M loads the program from a disk file into memory
  - Hands over all power over the machine to the program
  - The program overwrites the OS in memory (because memory was VERY expensive)
  - Only one program could be executed at a time
  - Program runs
- **When program finishes**
  - CP/M was reloaded from the floppy disk into memory
  - wait for another command to be typed

DOS and BIOS

- **PC DOS replaced CP/M on IBM PC**
  - DOS did not go away
  - remained in place while application was loaded
- **IBM took some code on a memory chip**
  - Program for handling Keyboard, display and disk drives
  - Took away from disk drive
  - Program written on a Read Only Memory (ROM) soldered to the motherboard
- **Program was called BIOS**
  - Basic Input/Output System
- **It is nicknamed "firmware" because it is not volatile**
- **All modern computers have BIOS**

Multitasking

- **DOS was not realy an OS**
  - Just file manager
  - and program launcher
- **Windows 95 was multitask**
  - Operated in 32-bits protected mode
  - But needed to support DOS
- **More than one task**
  - Memory was cheaper
  - Possibility to have several programs at the same time in memory
- **Example**
  - One program starts
  - After few milliseconds, windows "preempt"
  - Windows starts the next program
  - And so on, each program receive a small slice of CPU time

Multitasking

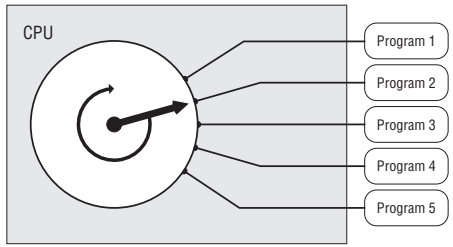
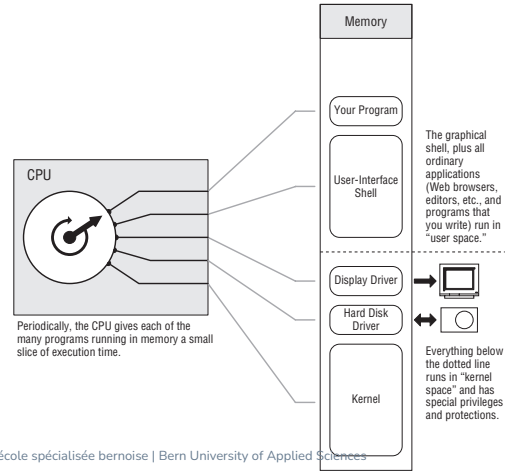


Figure 3-5: The idea of multitasking

Linux

- **Linus Torvals wrote Linux in 1991**
  - Unix like operating system
  - Multitask
  - More powerfull structure than Win 95
- **Core of Linux : *The Kernel***
  - Took full advantage of IA-32 protected mode
  - Kernel: entirely separated from user interface
  - System memory entirely tagged: *kernel space* or *user space*
  - Programs in user space do not have access to kernel space
  - Communication throw *System calls*
- **Direct access to hardware limited to kernel**
  - Memory, video, peripherals
- **This design is still in use in successors of Linux and Win NT**

Protected Mode



Conclusion

- **A program is just data**
  - Instructions are data
  - They are bits, and are represented as numbers (in hex for instance)
- **Memory and CPU are linked throw a Bus**
  - It links also all periferals
  - CPU writes on it and periferals read what is for them
- **Multitasking requires an OS that can protect programs and memory**
  - To protect features
  - To avoid security risks

Bibliography

- **This course corresponds to chapter 3 of the book:**
- Assembly Language Step by Step (3rd Edition)