Computer Architecture and operating systems

Theoretical background

Part II

Krystian Wojtkiewicz

Wrocław, Poland

https://i-cu.eu

Table of contents

Memory

The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 or 1. All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs.

A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage. For example, most computers don't have an instruction to move a bit but do have one to move a byte. A less common term is word, which is a given computer architecture's native unit of data. A word is made up of *one or more* bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time[2].

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.

a **kilobyte**, or KB, is 1,024 bytes

a **megabyte**, or MB, is $1{,}024^2$ bytes

a **gigabyte**, or GB, is $1{,}024^3$ bytes

a **terabyte**, or TB, is $1{,}024^4$ bytes

a **petabyte**, or PB, is $1{,}024^5$ bytes[1]

[1] Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

General remarks

- it is impossible to build an arbitrary unlimited memory storage with infinitely short access time
- access time grows with size
- with hierarchical layer architecture every next layer has bigger size with higher access time

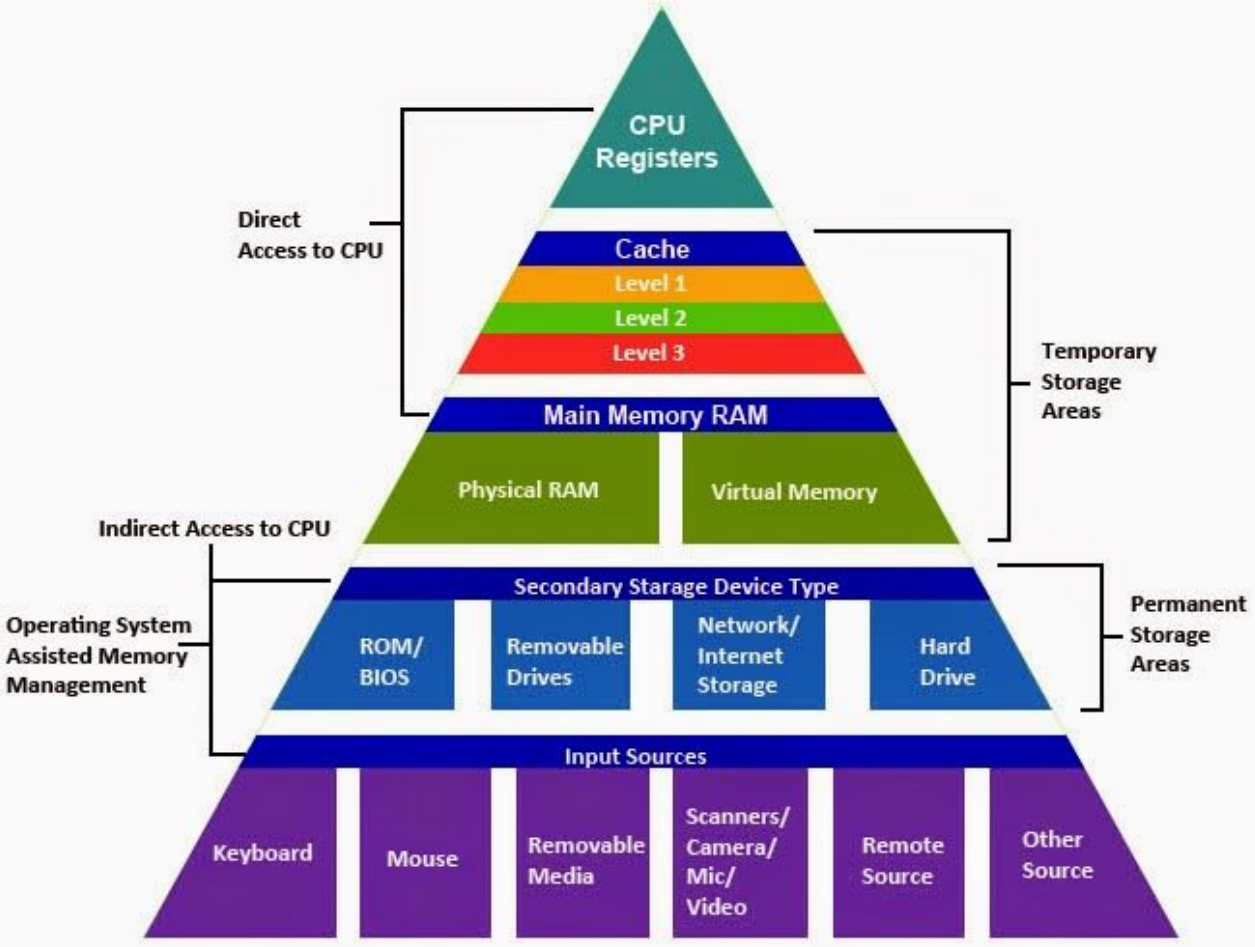


Figure: Memory Hierarchy [1]

General remarks

Memory can be generalised into five hierarchies based upon intended use and speed. If what the processor needs isn't in one level, it moves on to the next, to look for what it needs:

- volatile memory
  - registers,
  - cache,
  - main memory,
- permanent storage
  - secondary memory,
  - removable memory.

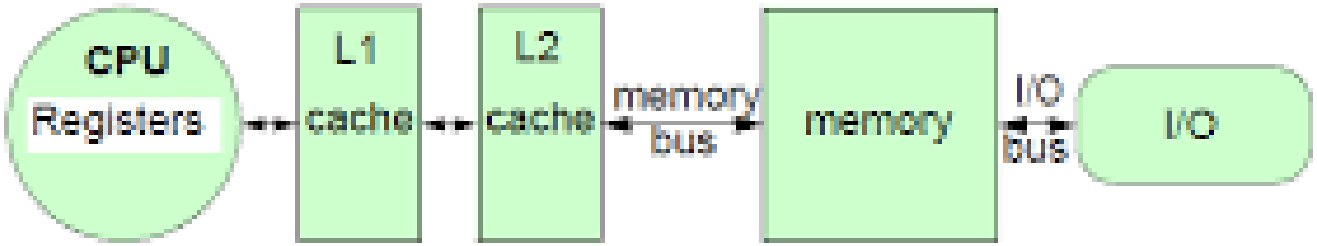
General remarks

While the principle of the memory hierarchy applies to all layers, the mechanisms controlling the movement of data between individual memory layers are different.

The most often used elements are moved up the hierarchy

- programmer writing the code
- hardware
- operating system
- program
- user

Typical memory hierarchy:



Level	1	2	3	4
Named as	Registers	Cache	memory	disk storage
Typical size	<1 KB	< 4 MB	<2 GB	>2GB
Access time (ns)	2 - 5	3 - 10	80 - 400	5'000'000
Bandwidth(MB/sec)	4000 - 32'000	800 - 5000	400 - 2000	4 - 32
Managed by	Compiler	Hardware	Operating system	Operating system / user

Figure: Memory Hierarchy access time

von Neumann architecture

General remarks

- instructions forming the program are stored in memory in the same way as data
- memory is built from number of enumerated cells
  - access to the memory occurs by entering the cell number by the processor
  - the cell number is referred as **address**
- in practise:
  - typically, the computer will load subsequent program instructions from subsequent memory cells
  - these cells will be selected by the increasing address, which should be stored and incremented in the processor
  - this address is stored in a special register called **Program Counter - PC**

Harvard

separate hierarchies for program and data[2]

Princeton

common hierarchy for program and data

[2] Some believe that this type of hierarchy does not comply with von Neumann principles since there are separate hierarchies for program and data

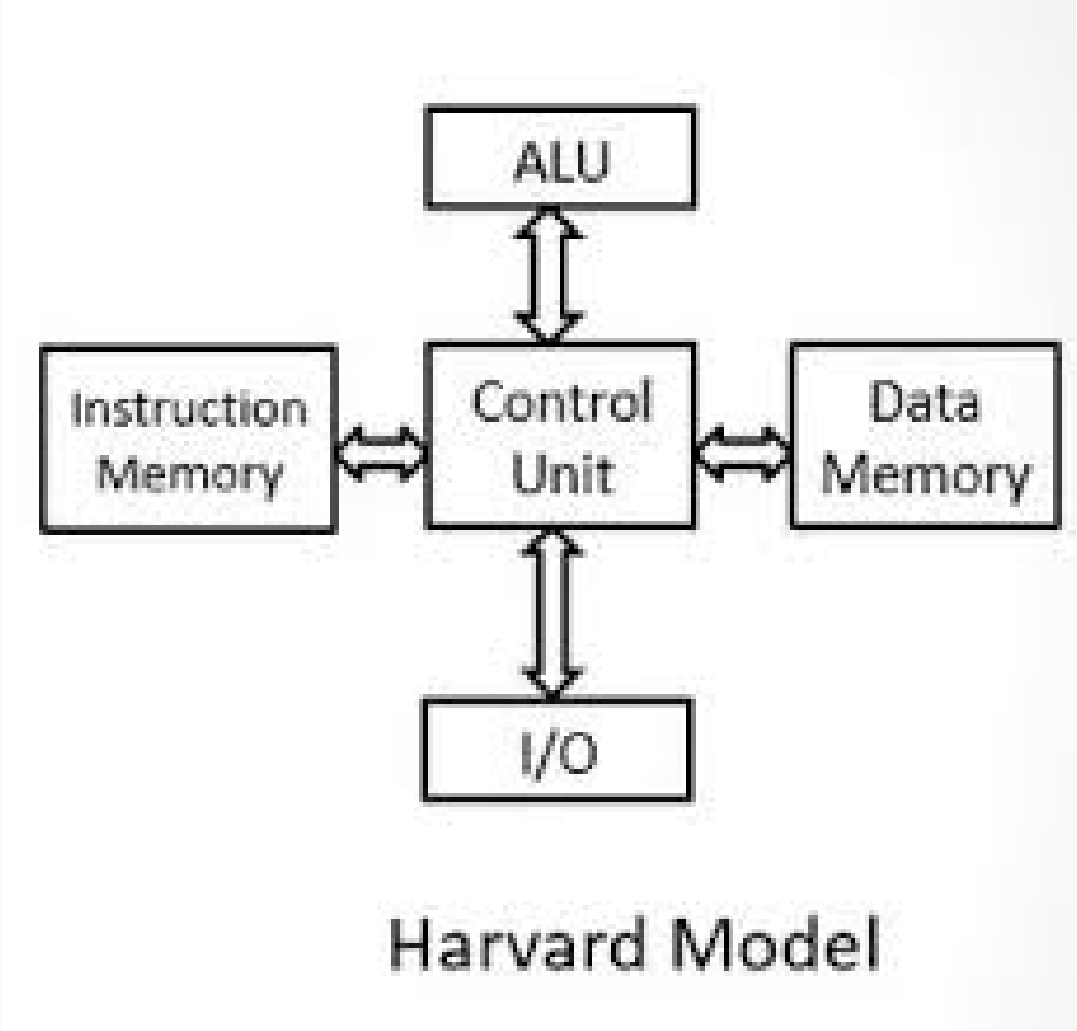


Figure: Harvard Architecture

Harvard architecture properties

- it is a von Neumann architecture implementation, however the program and data hierarchies are separate
- it has high performance due to ability to parallel instruction loading and operating on data hierarchy
- there is no possibility to write into program hierarchy
  - no programming
  - static (embedded) program only
  - only for dedicated solutions

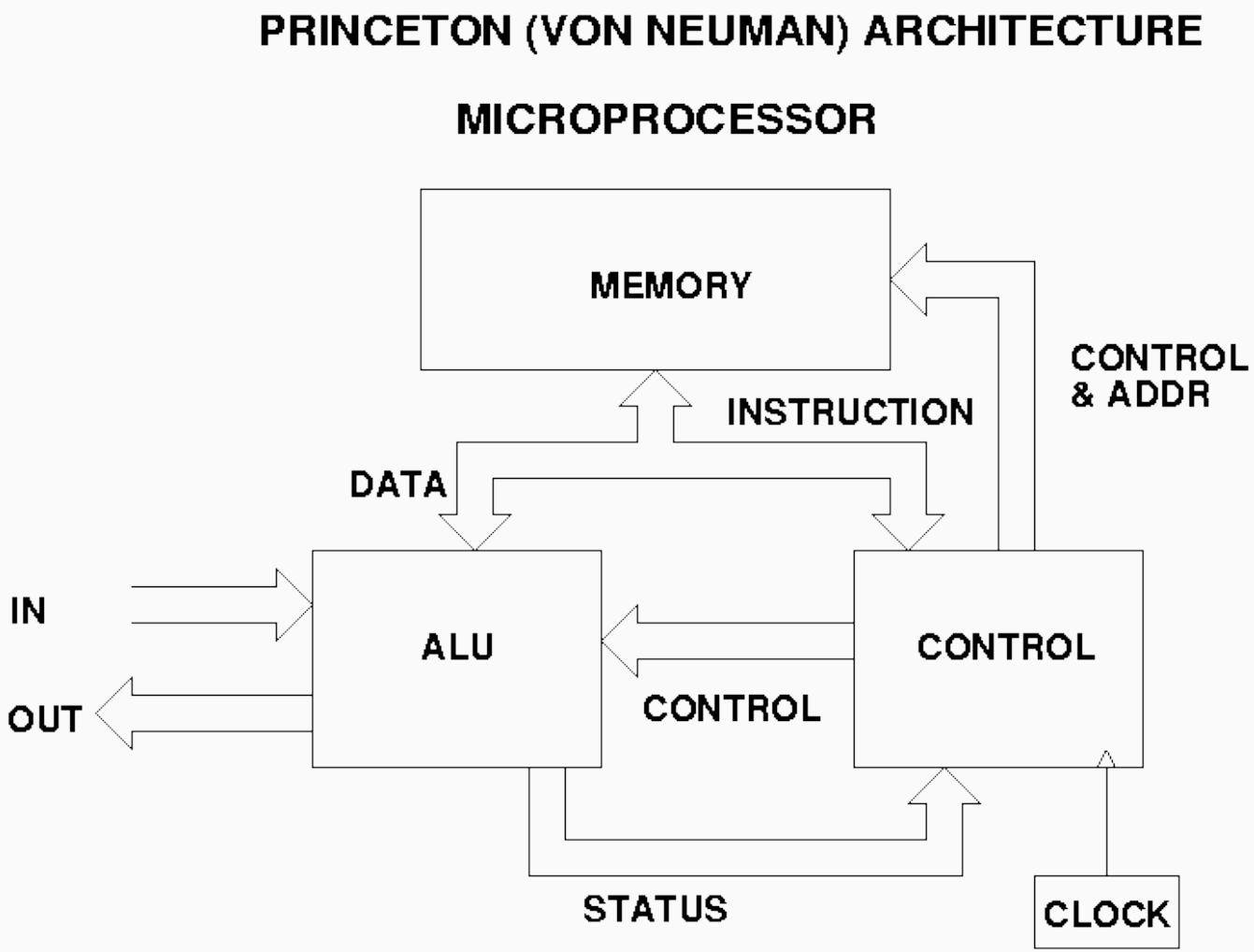


Figure: Princeton Architecture

Princeton architecture properties

- it is an exemplary von Neumann architecture implementation with common program and data hierarchy
- common hierarchy banishes parallel instruction loading and operating on data hierarchy

  von Neumann bottleneck
- there are no limitations on program changing
  - program stored by the data processor into memory hierarchy as data can be retrieved by the instruction processor as instruction
  - can be easily programmed - universal computers
  - program can modify itself (do we really want it?)

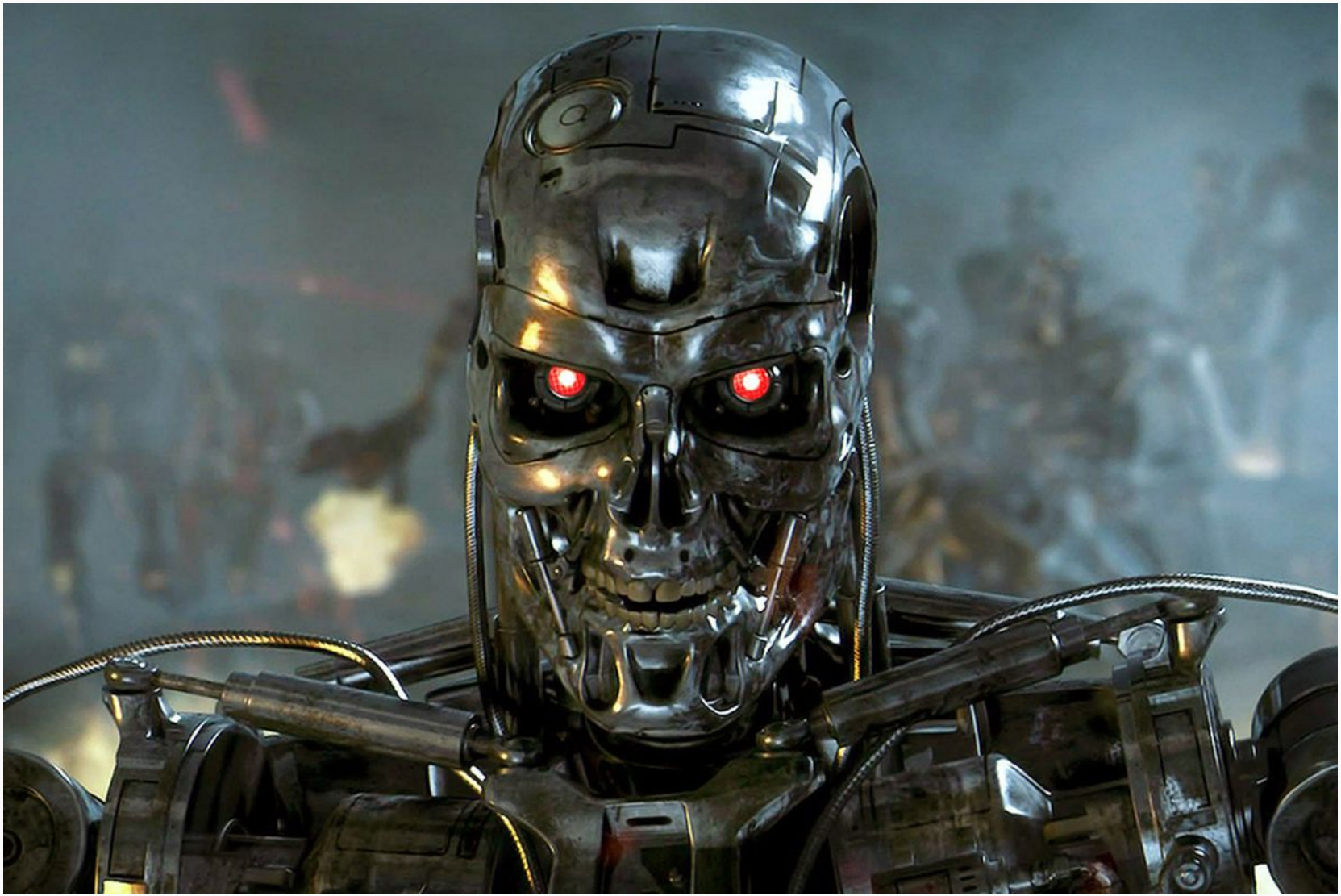


Figure: Terminator will return... soon

Harvard-Princeton architecture properties

- it is an von Neumann architecture implementation with common upper memory hierarchy and separate lower memory hierarchy parts
- at least one *cache* level is separate for data and instructions
- most of memory calls is made in higher memory hierarchy parts
  - higher performance due to Harvard architecture properties
- common lower parts of memory hierarchy allows for program modifications
  - it is programmable architecture
- program itself does not have full control on location of objects in the memory (it cannot modify itself)
- such a control can have operating system e.g. for running another program from a file storage

Data types

- Logical values (true/false)
- alphanumerical signs
- numbers
  - integer
    - signed
    - unsigned
  - non-integer
    - fixed-point
    - floating-point
- sounds and one-dimensional signals
- raster images

Basic rules

Binary rule

Computer can operate only on **bits** organized in groups *binary digits* forming binary numbers[3]

Complex data

Data that are not numbers (signs, signals) have to be represented by the numbers

[3] nowadays computers operate on binary words, which length equals 8×2^n (e.g. 8, 16, 32, 64)

References

References I

[1] Mark Lungo. *Memory Hierarchy / Useful Notes*. Dec. 2015. URL: https://tvtropes.org/pmwiki/pmwiki.php/UsefulNotes/MemoryHierarchy.

[2] A. Silberschatz, G. Gagne, and P.B. Galvin. *Operating System Concepts*. Wiley, 2018. ISBN: 9781119439257. URL: https://books.google.pl/books?id=VFV1DwAAQBAJ.