LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

P6 – Scientific Programming

Marcus Mohr
Jens Oeser

Geophysics Section
Department of Earth and Environmental Sciences
Ludwig-Maximilians-Universität München

SoSe 2021

Part #2

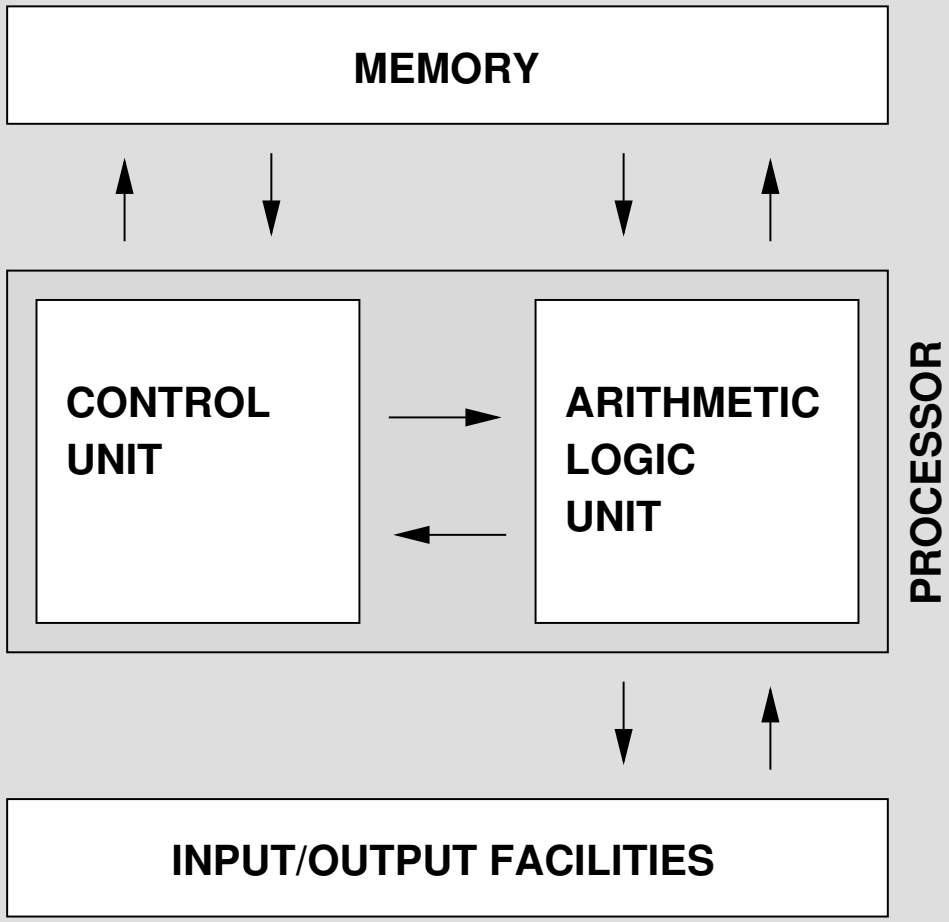
Primer on Computer Architecture

von Neumann architecture, characteristics of processors, CISC/RISC, registers, pipelining, ...

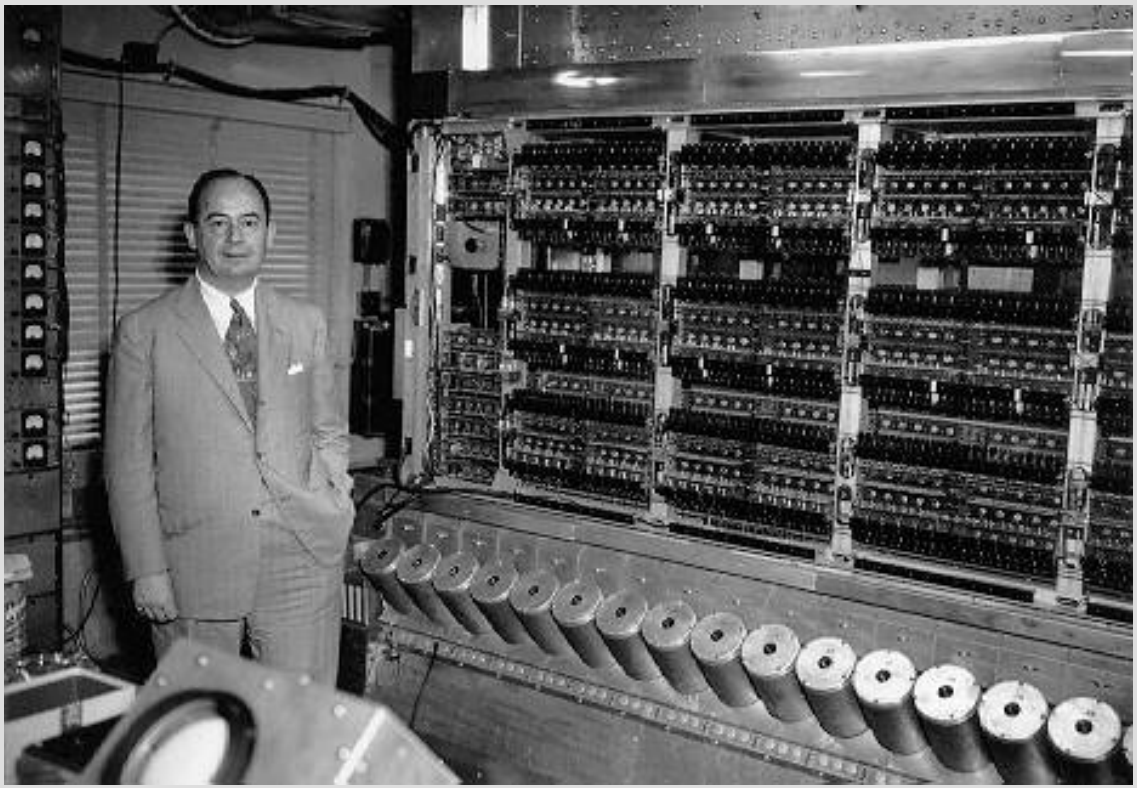
Organisation of Computer Systems

- Fundamental Questions:
  - How should a computer be set up and organised?
  - How should computations be controlled and steered?
- Burks, Goldstine and von Neumann: Preliminary Discussion of the Logical design of an Electronic Computing Instrument, U.S. Army Ordonance Dept. Report 1946, 1946.
- Resulting Concept
  - von Neumann Architecture
  - still the basis for most current system designs

von Neumann Architecture I



von Neumann Architecture II



The first computer system implemented based on this design concept:

Institute for Advanced Studies (IAS) Computer, 1952

Architectural Details I

- system consists of four basic units
  - control & execution unit
  - main memory (with random read-write access)
  - input-output facility
- connected by simple busses
- general-purpose computer
  - system must not be changed physically, e.g. re-wired, for different applications
  - data and program both reside in main memory (stored program approach)

Architectural Details II

- (Main) memory is divided into cells
  - of equal size
  - consecutively numbered (addresses)
- A program consists of a sequence of instructions to the processor
- Program execution can be altered dynamically
- binary design, two values per digit: 0/1

Architectural Details III

- von Neumann architecture still dominates
  (basis of all popular system/chip designs)
- minor modifications
  - control & execution unit form one device: CPU
  - new interfaces to memory and IO
    (e.g. Direct Memory Access (DMA), Hypertransport)
- enhancements of concept
  - multiple execution units (ALUs/FPUs) per CPU
  - multiple CPUs per system
  - on chip parallelism
  - multi-core CPUs

What's Characteristic for a Processor?

What's Characteristic for a Processor?

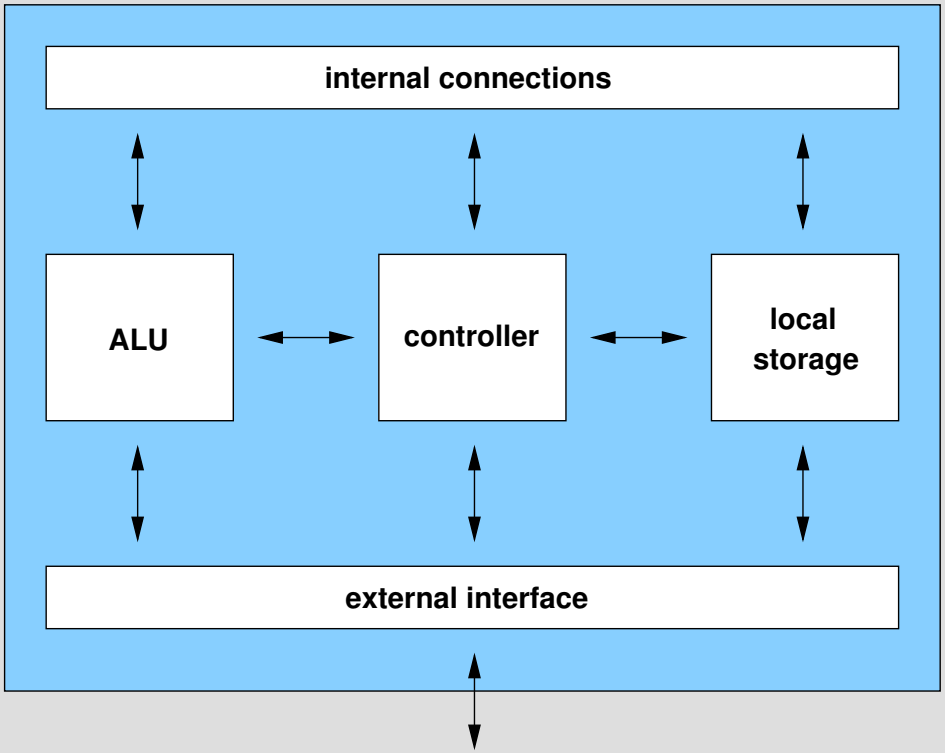
- clock rate / frequency
- power consumption
- interfaces to memory & IO
- type and size of caches
- data representation / endianess
- instructions set (CISC, RISC, EPIC, ...)
- types & numbers of registers
- number of cores
- ...

Types of Processors

- Fixed logic:
  performs a single (hard-wired) operation; example: compute $\sin(x)$, a single graphics operation, ...
- Selectable logic:
  able to perform several hard-wired operations; operation can be selected when processor is invoked; example: compute $\sin(x)$ or $\cos(x)$
- Parameterised logic:
  offers additional flexibility by parameterisation of function; example: hash function $h(x)$ using parameters (p, q)
- Programmable logic:
  most flexible variant; for each invocation of the processor the type and order of operations can be chosen individually; example: desktop CPU

Idealised Processor Design

- controller
- execution Unit
  - often a Arithmetic Logical Unit (ALU)
  - and a separate Floating Point Unit (FPU)
- local memory (Caches / Registers)
- internal connections
- external interface



Registers

- registers are memory slots on the processor itself
- they provide very fast access
  (example: Digital Alpha PWS 500au A21164, register 2 ns, main memory 112 ns)
- have short addresses
- are used to store data (for operations), program instructions, states or counters
- register are classified by their purpose

Example: IA-64 Architecture (Itanium 2)

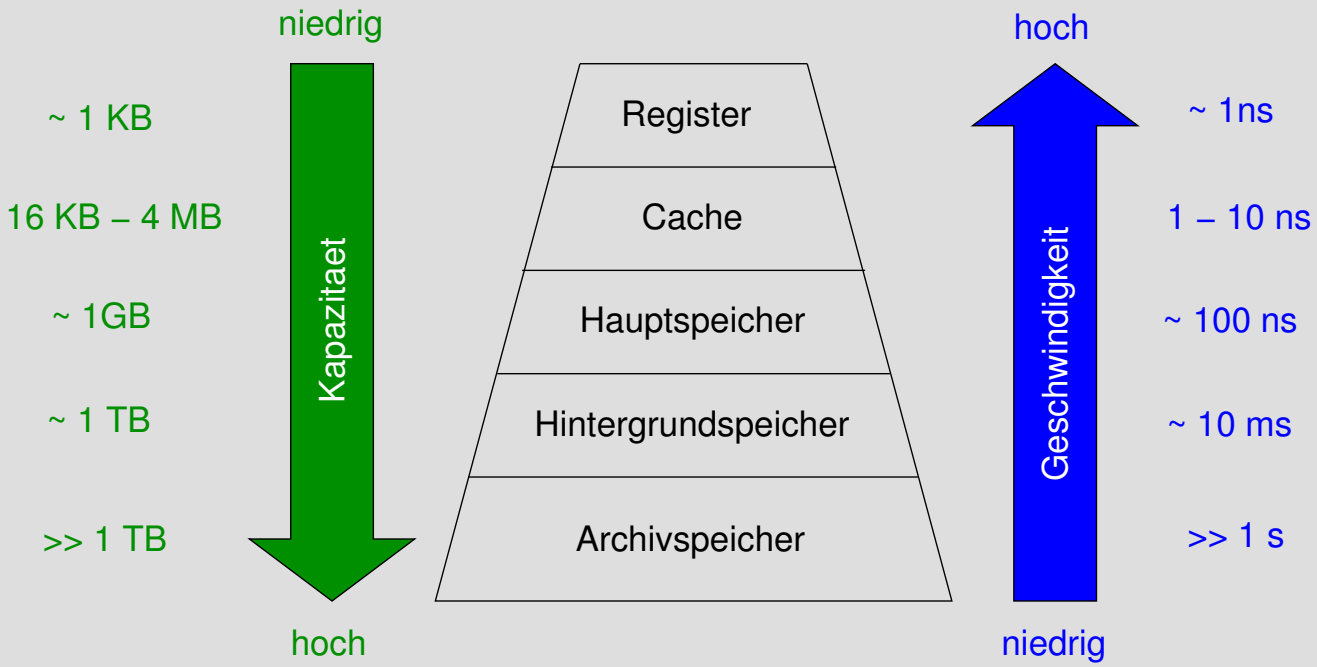
distinguishes 10 different types:

type	number	usage
general purpose	128	multi-purpose 64-bit wide
floating point	128	FP registers, 82-bit wide ($\pm$, 64-Bit mantissa, 17 bit exponent)
branch	8	for indirect jumps, e.g. return address for sub-routine calls
predicate	64	truth values for comparisons and tests

as well as Instruction Pointer, Current Frame Marker, Application Registers, Performance Monitor Data Registers, User Mask and Processor Identification Registers

Memory Wall

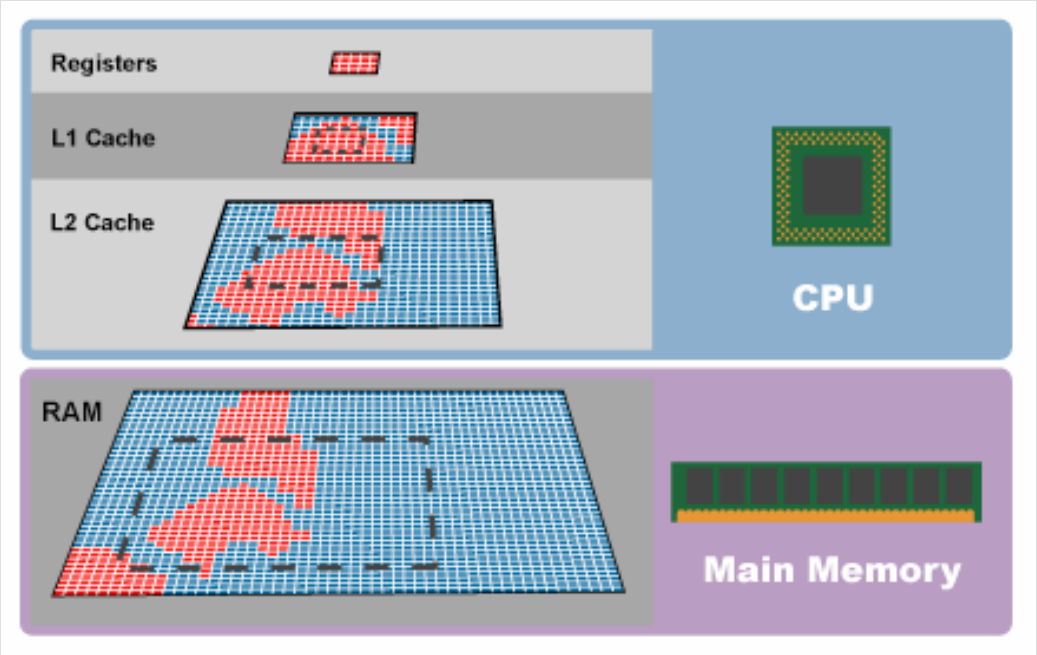
- processing speed of the processor typically exceeds speed of memory interconnect → processor does not get data, has to wait
- roblem known as “memory wall” or algorithm is “memory-bound”



(Abb. nach M. Gerndt, LRR, TUM)

Caches

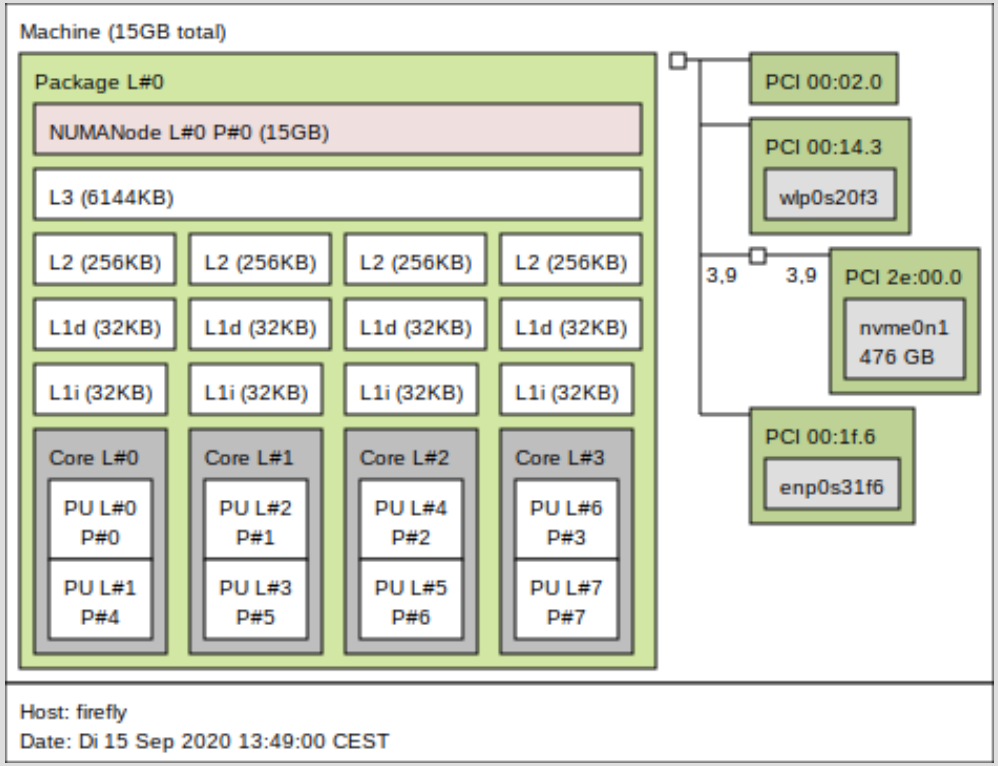
- to alleviate the problem computers employ a complex memory hierarchy
- composed of successively smaller, but faster memory storage units
- today: typically 3 levels of caches (L1, L2, L3)



from a talk by U. Rüde & J. Treibig

Example 1: Lenovo Thinkpad

- Topology of my current Thinkpad
- CPU: Intel Core i5-10210U
- 4 physical cores
- 8 "virtual" cores with hyperthreading
- Level 1 caches are split into
  - L1i: for instructions
  - L1d: for data



Example 2: AMD Epyc Milan

- Schematic of the new (2021) 64-core flagship CPU of the AMD Epyc Milan family
- Example of System-on-a-Chip (SoC) technology

  combines functionality traditionally placed on other parts of the motherboards into the chip

Fetch-Execute-Cycle

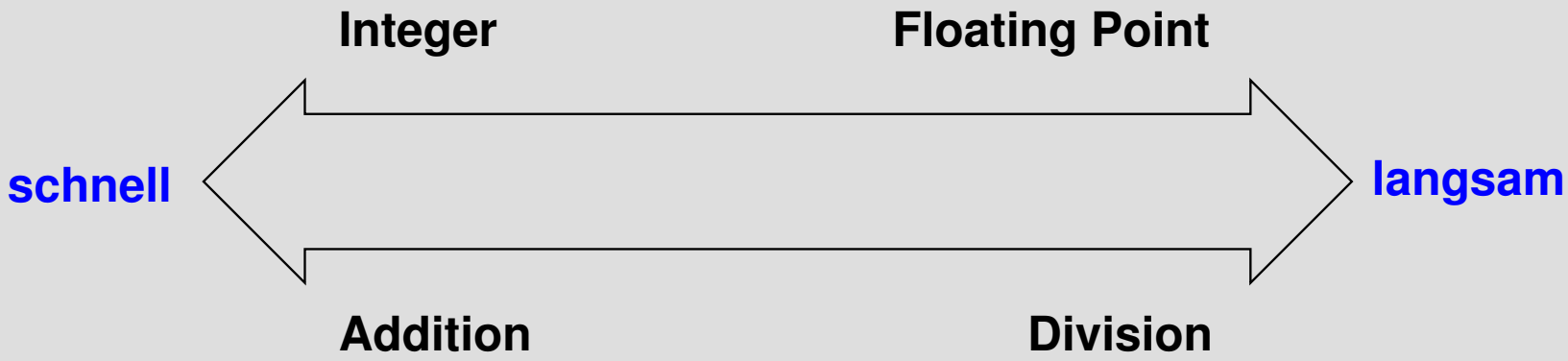
The control unit forms the CPU's headquarter. It steers program execution via the von Neumann cycle:

loop
FETCH {fetch next program instruction from memory}
DECODE {decode instruction}
FETCH OPERANDS {load data for operation}
EXECUTE {execute operation}
UPDATE INSTRUCTION POINTER {increase instruction counter}
end loop

Clock and Instruction Rate

The rate at which a processor's digital logic operates is given by its clock rate (e.g. 2.4 GHz).

However, the Fetch-Execute-Cycle has no fixed frequency. Different instructions require different spans of time to complete.



Instruction Set Architecture

One essential characteristic of a processor is its Instruction Set Architecture:

- Instructions Provided
- Native Data Formats and their Storage
- Access to Operands (Register, Stack, ...)
- Classification: CISC vs. RISC
- Privileges and Security Mechanisms

"Language and Grammar of the Processor"

CISC vs. RISC

CISC = Complex Instruction Set Computer

- large set of several hundred instructions
- some instructions are very powerful
- this allows for small executable program codes
- makes implementation difficult (and maybe slow)

RISC = Reduced Instruction Set Computer

- small set of instructions (e.g. only 32)
- simple, efficient implementation

CISC vs. RISC (cont.)

The distinction between CISC (e.g. Intel/AMD x86ers) and RISC (e.g. DEC Alpha, Sun SPARC, MIPS) becomes less and less important:

- RISC processors get more complex
- CISC processors borrow RISC concepts
- Pentium Pro and Co. have a CISC to RISC conversion pre-stage
- new concepts are gaining importance, such as e.g. Explicitly Parallel Instruction Computing (EPIC) in the Itanium family

Types of Instructions

The commands of an instruction set can be divided into several groups:

- Arithmetic Instructions (Integer)
- Logical Instructions (Booleans)
- Access to and Transfer of Data
- Control Flow (Branching)
- Floating Point Instructions
- Commands for Processor Control

Example: MIPS Floating Point Instructions

Arithmetic	
FP add	floating point addition
FP subtract	floating point subtraction
FP multiply	floating point multiplication
FP divide	floating point division
FP add double	double precision addition
FP subtract double	double precision subtraction
FP multiply double	double precision multiplication
FP divide double	double precision division
Data Transfer	
load word co-processor	load value into FP register
store word co-processor	store FP register to memory
Conditional Branch	
branch FP true	branch if FP condition is true
branch FP false	branch if FP condition is false
FP compare single	compare two FP registers
FP compare double	compare two double precision values

Bits, Bytes and Words

Bit:

Smallest unit of information; a Binary Digit can only have value 0 or 1

Byte:

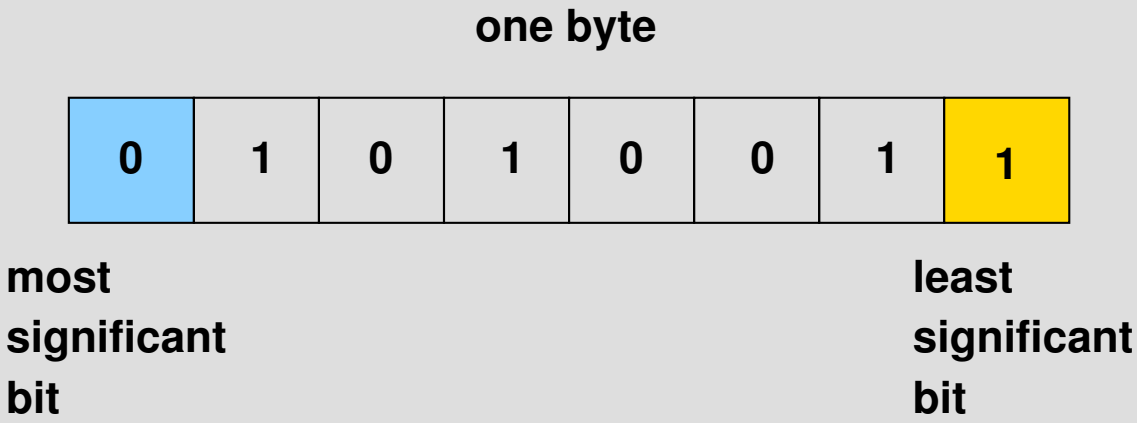
A consecutive set of bits, today always 8 bits

Word:

Main memory of a computer is divided into chunks of equal size = words; corresponds to the width of the memory bus; each read-/write-access always relates to a complete word (e.g. 64-bit system: 1 word = 8 bytes)

Bit Layout

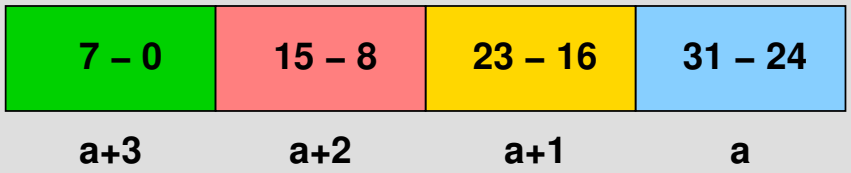
The bits in a byte (can be thought to be) stored as with decimal numbers going from left/largest power to right/smallest power



$$= 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 83$$

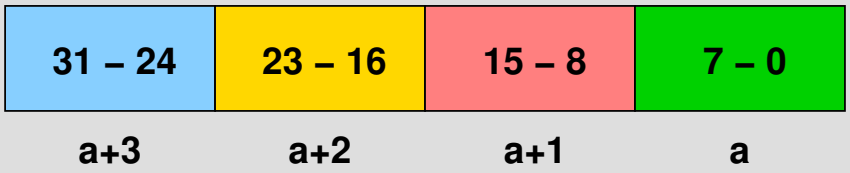
Endianess

How to store multi-byte entities?



Big Endian

e.g. Motorola 68000,
SPARC, PowerPC



Little Endian

e.g. Intel x86
IA-64 (with Linux)

Gulliver's Travels

Big-Enders versus Little-Enders

References

- Douglas Comer: 'Essentials of Computer Architecture', Pearson Prentice Hall
- David Patterson & John Hennessy: 'Computer architecture - A Quantitative Approach, Morgan Kaufmann
- Georg Hager & Gerhard Wellein: 'Introduction to High Performance Computing for Scientists and Engineers', CRC Press
- Wikipedia