Computer architecture

Introduction

http://d3s.mff.cuni.cz/teaching/computer_architecture/

Lubomír Bulej

bulej@d3s.mff.cuni.cz

CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

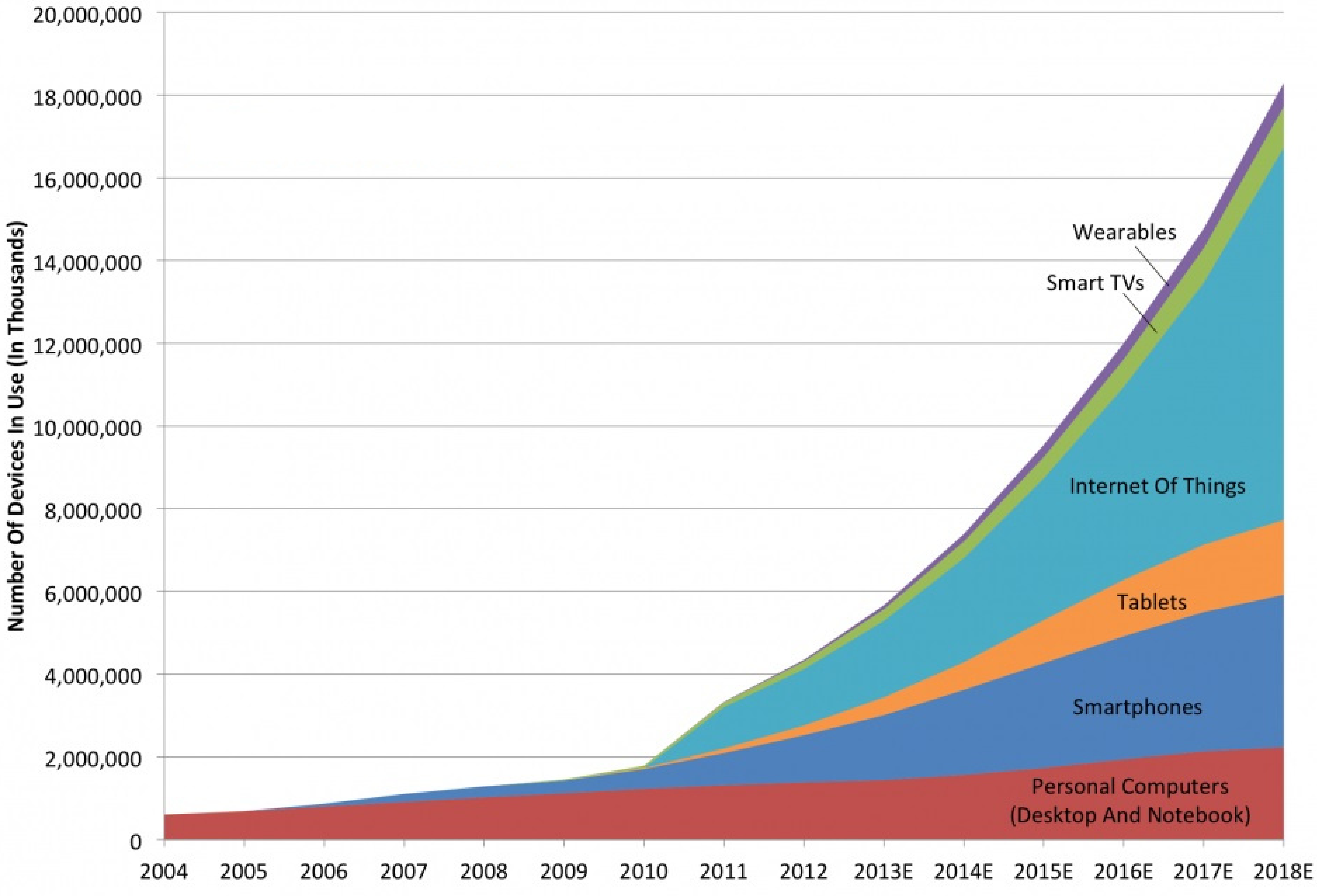
What is interesting on computers?

- **Very dynamic field**
  - First electronic computers around 1940
  - 60 years later: pervasive
  - New technologies replaced before they become old
- **Tremendous impact on everyday life**
  - Internet, embedded computers, human genome, computational chemistry, ...
  - New possibilities with every new order of magnitude in cost reduction, performance increase, size reduction

What is a computer?

- A broad term
  - Many common technologies
  - Different architecture to match different requirements
- Main classes
  - **Personal computers**
    - Optimal price/performance ratio (drives development)
  - **Servers, mainframes, supercomputers**
    - Higher throughput, reliability, computing power
    - Scientific calculations, serving high number of users
  - **Embedded computers**
    - The most repidly growing market (not only mobile devices)
    - Limited resources (memory, performance, energy, cost), special requirements (sturdiness)

Global Internet Device Installed Base Forecast



Source: Gartner, IDC, Strategy Analytics, Machina Research, company filings, BII estimates

Mainframe (1964)

- IBM System 360
  - Integrated circuits
  - Revolutionary elements
    - Modular constructions
    - Unified data and instructions
    - Unified interface for peripheral devices
    - Memory protection
  - Architectural elements kept even in today‘s mainframes

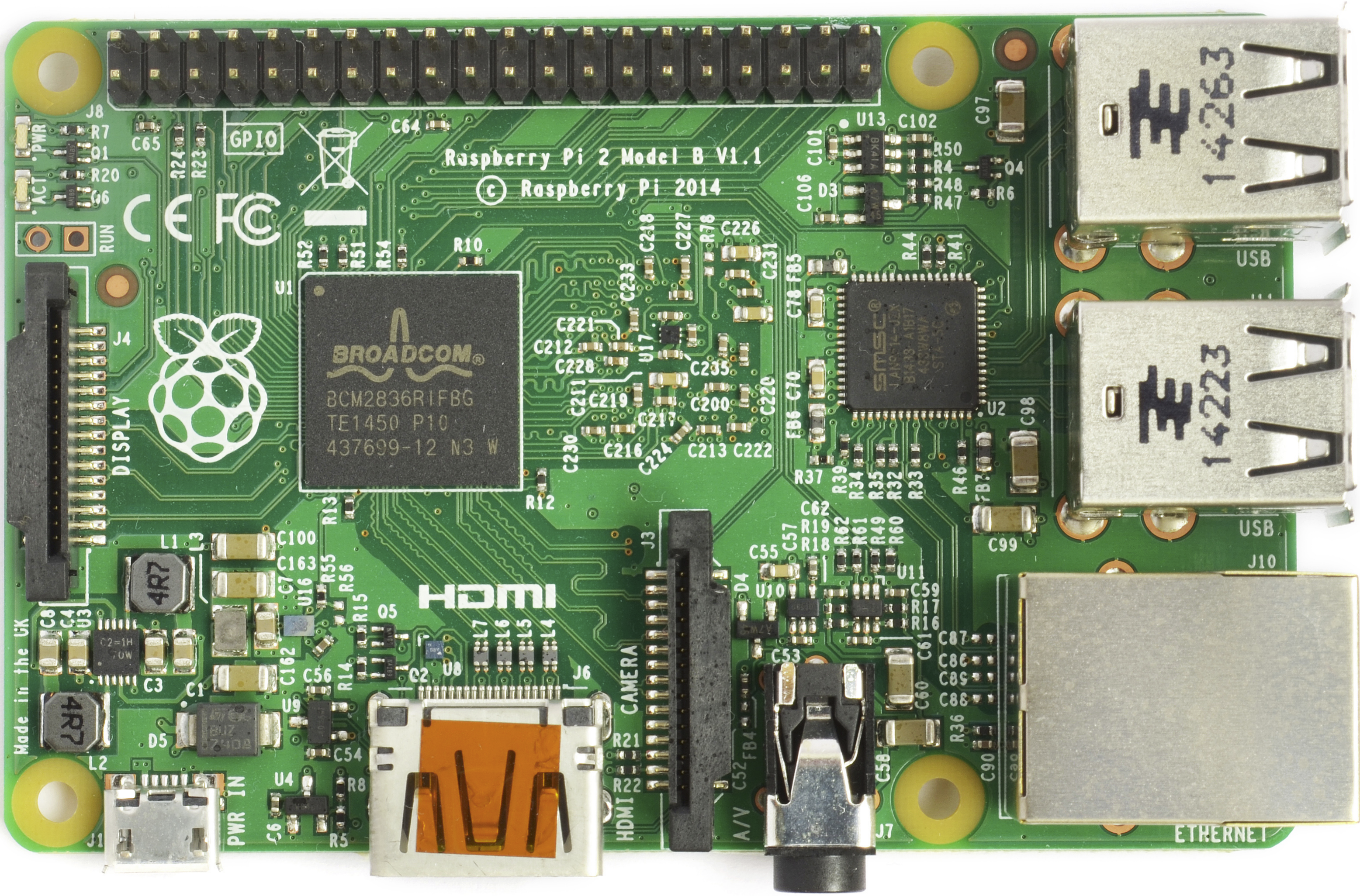
[1]

Mainframe (2005)

- **IBM System Z9-109 model S54**
  - 60 configurable LPARS
  - *Special-purpose processors*
  - 512 GB of memory
  - 1 740 kg, 2,49 m², 18.3 kW input power
  - Availability/reliability, throughput, security

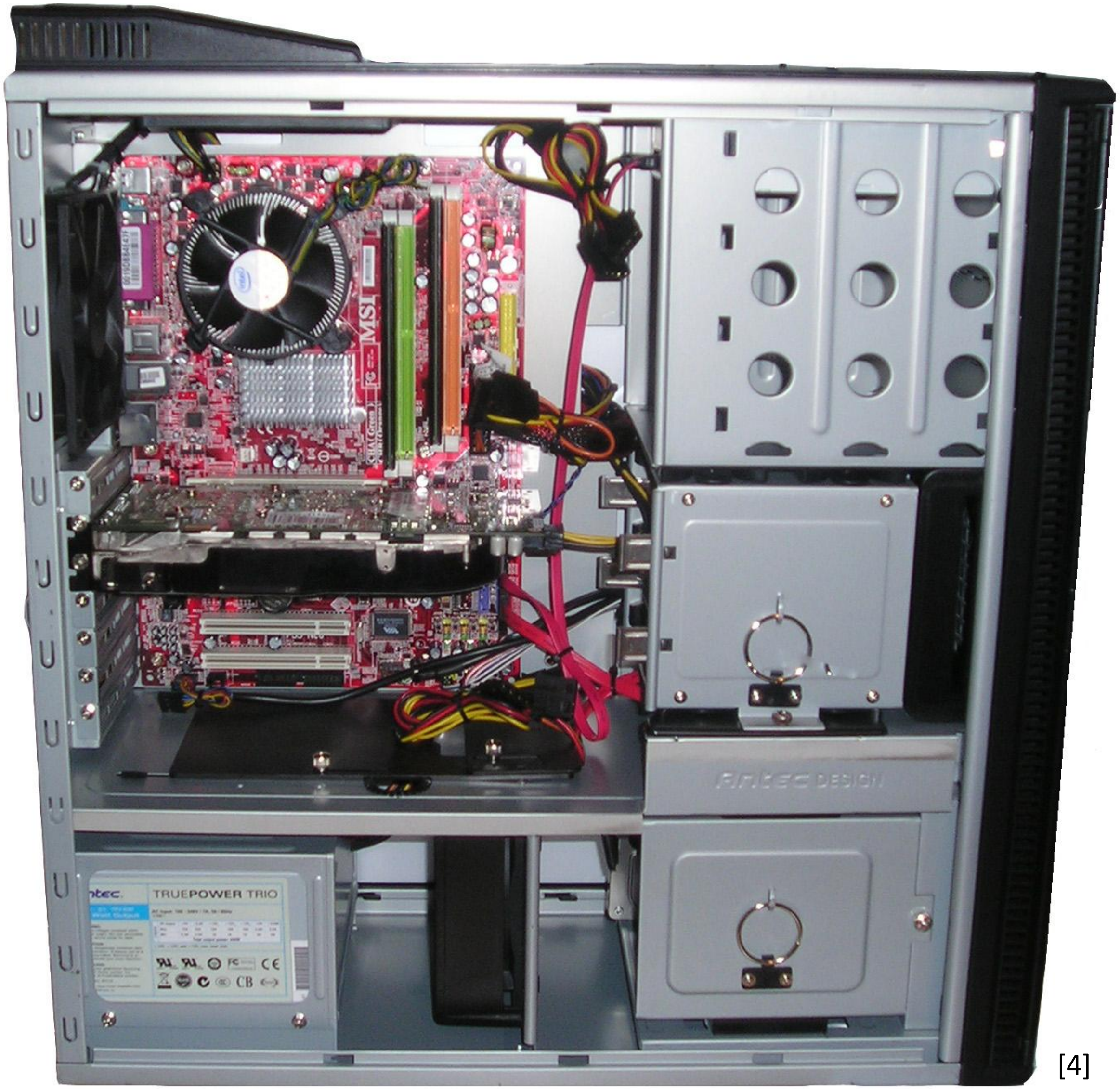
[2]

Less common personal computer



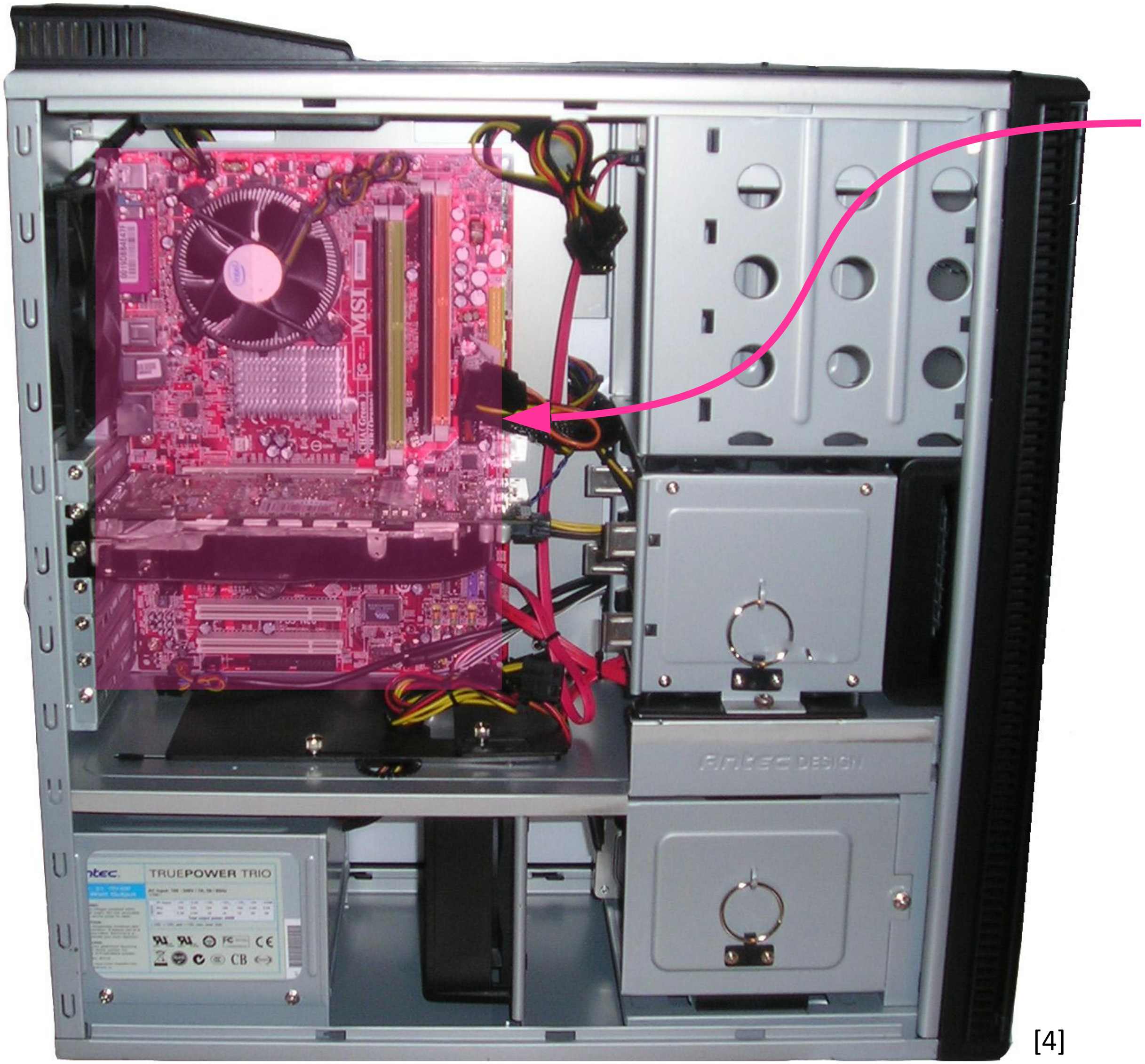
[3]

Typical personal computer



[4]

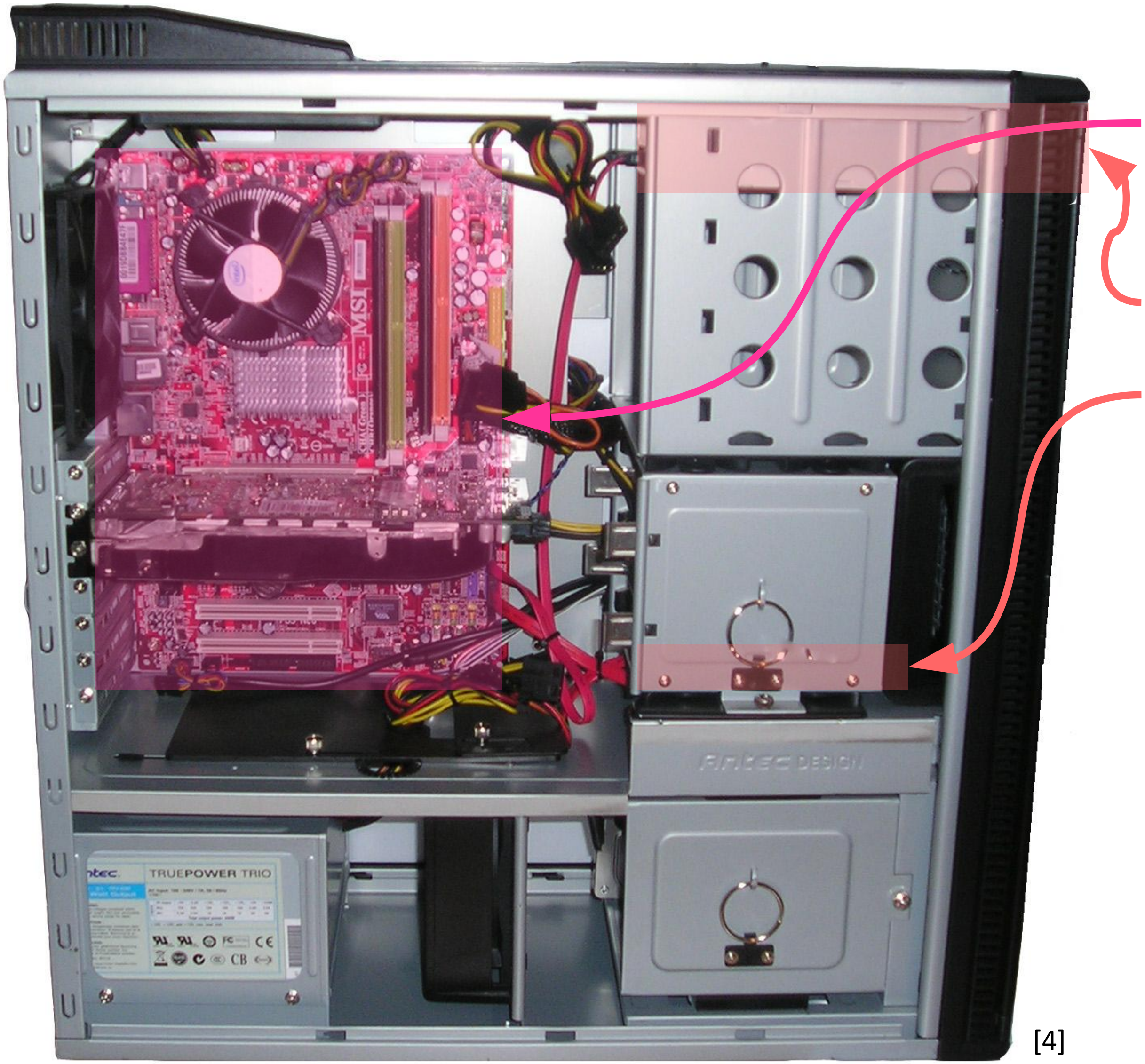
What’s in the box?



[4]

Motherboard
Processor
Memory (RAM, ROM)
Chipset
Basic I/O devices

What’s in the box?



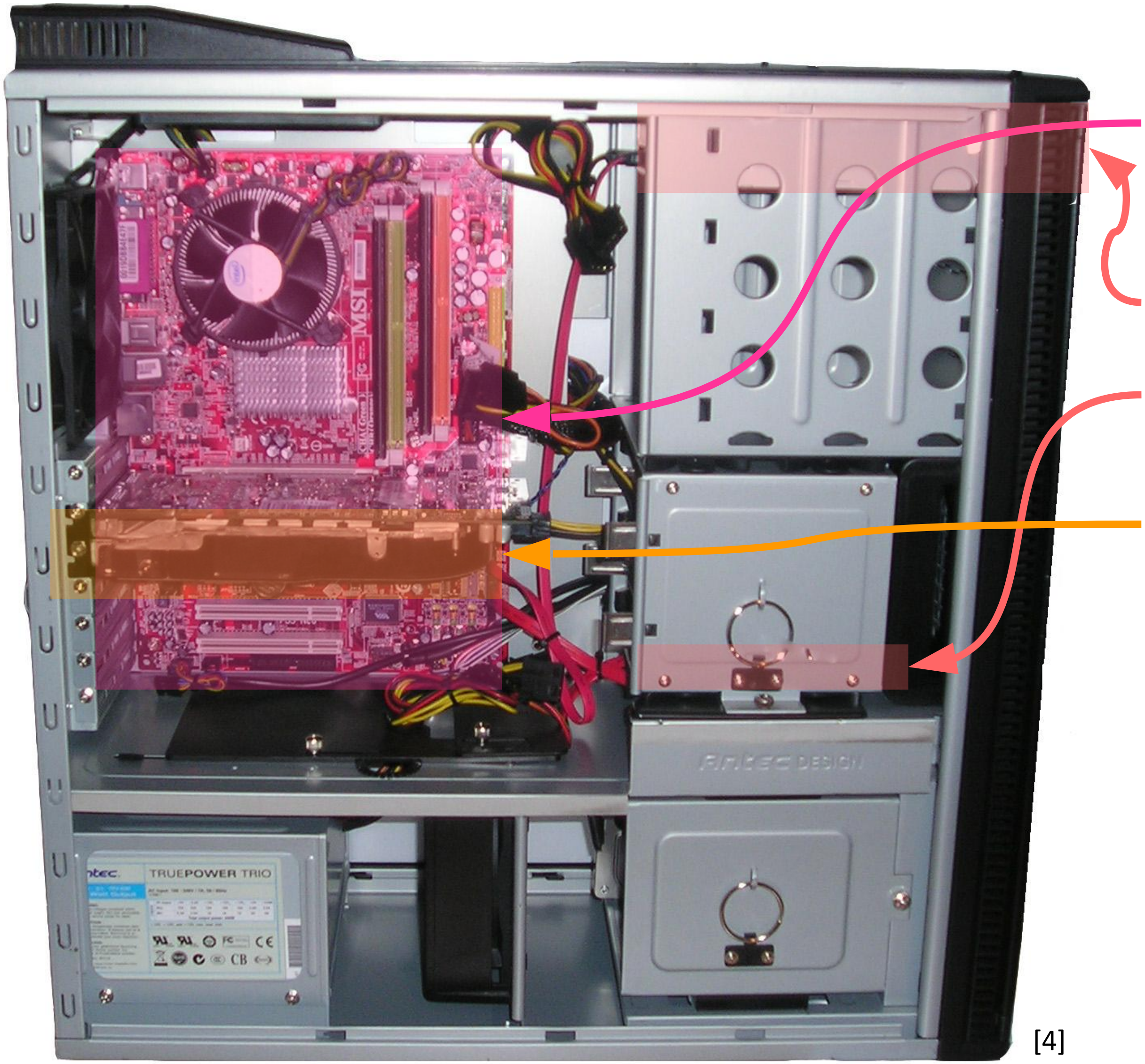
Motherboard
Processor
Memory (RAM, ROM)
Chipset
Basic I/O devices

Optical drive

Hard drive

[4]

What’s in the box?



Motherboard
Processor
Memory (RAM, ROM)
Chipset
Basic I/O devices

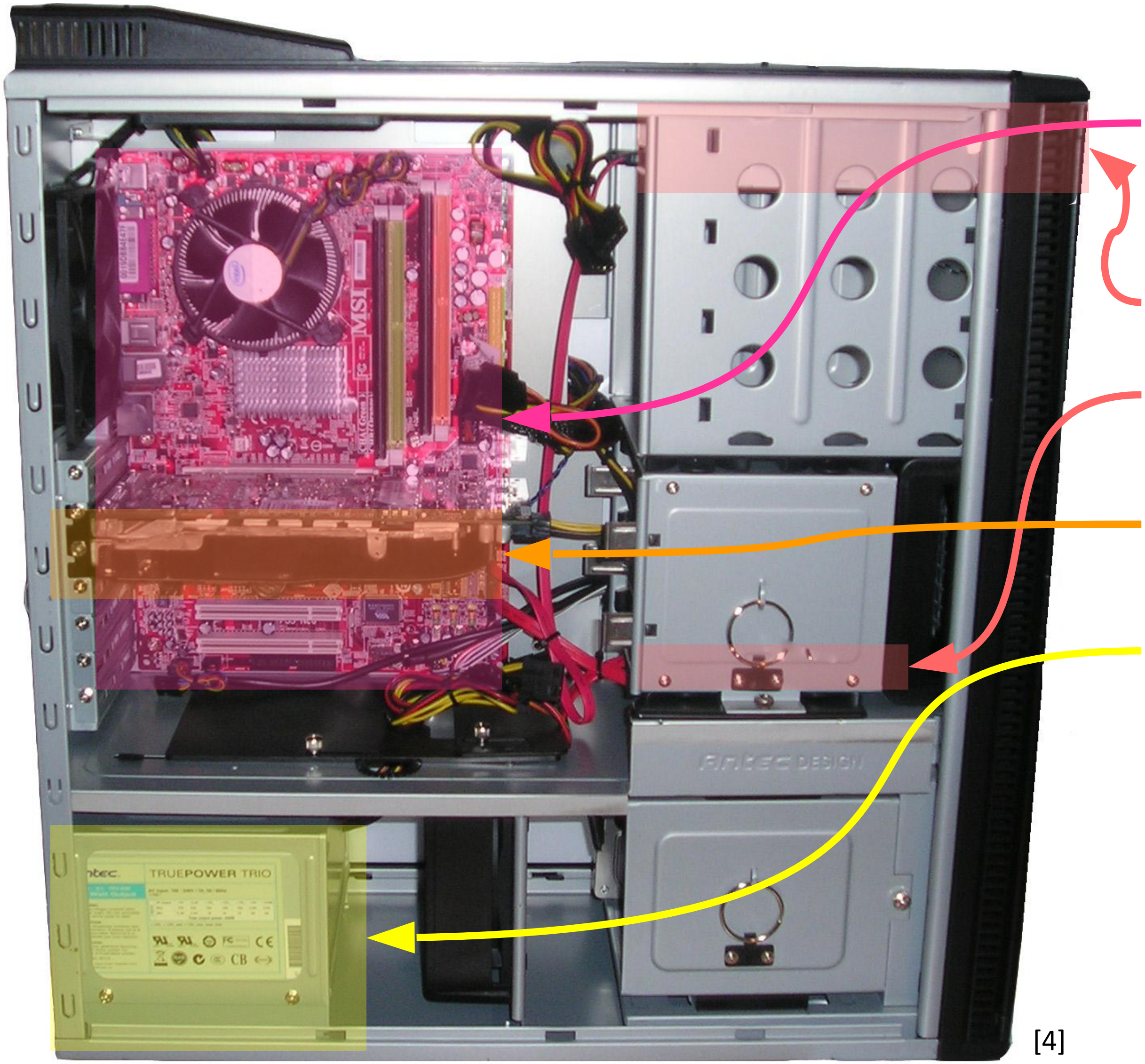
Optical drive

Hard drive

Expansion cards
Video card

[4]

What’s in the box?



Motherboard
Processor
Memory (RAM, ROM)
Chipset
Basic I/O devices

Optical drive

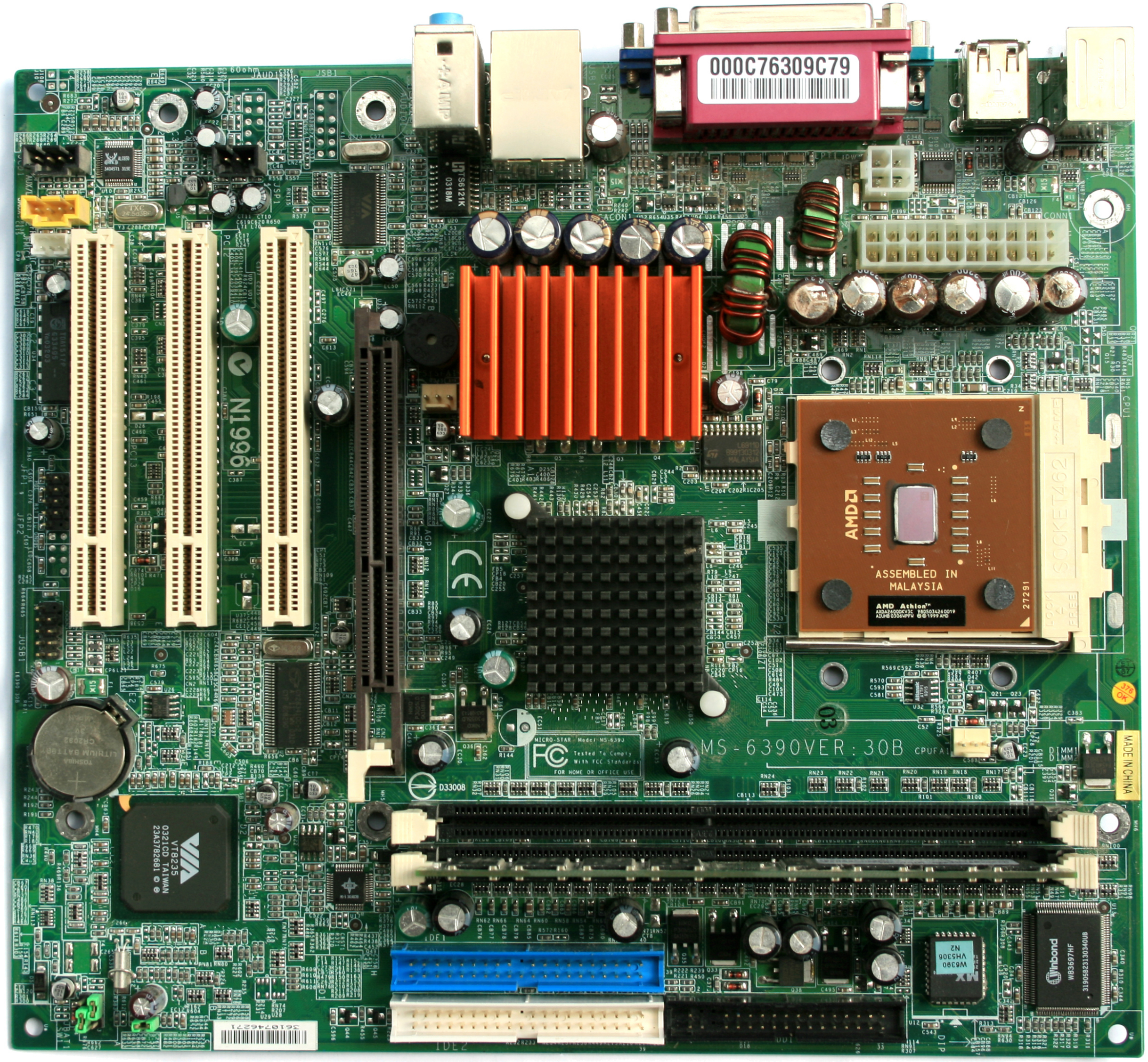
Hard drive

Expansion cards
Video card

Power supply

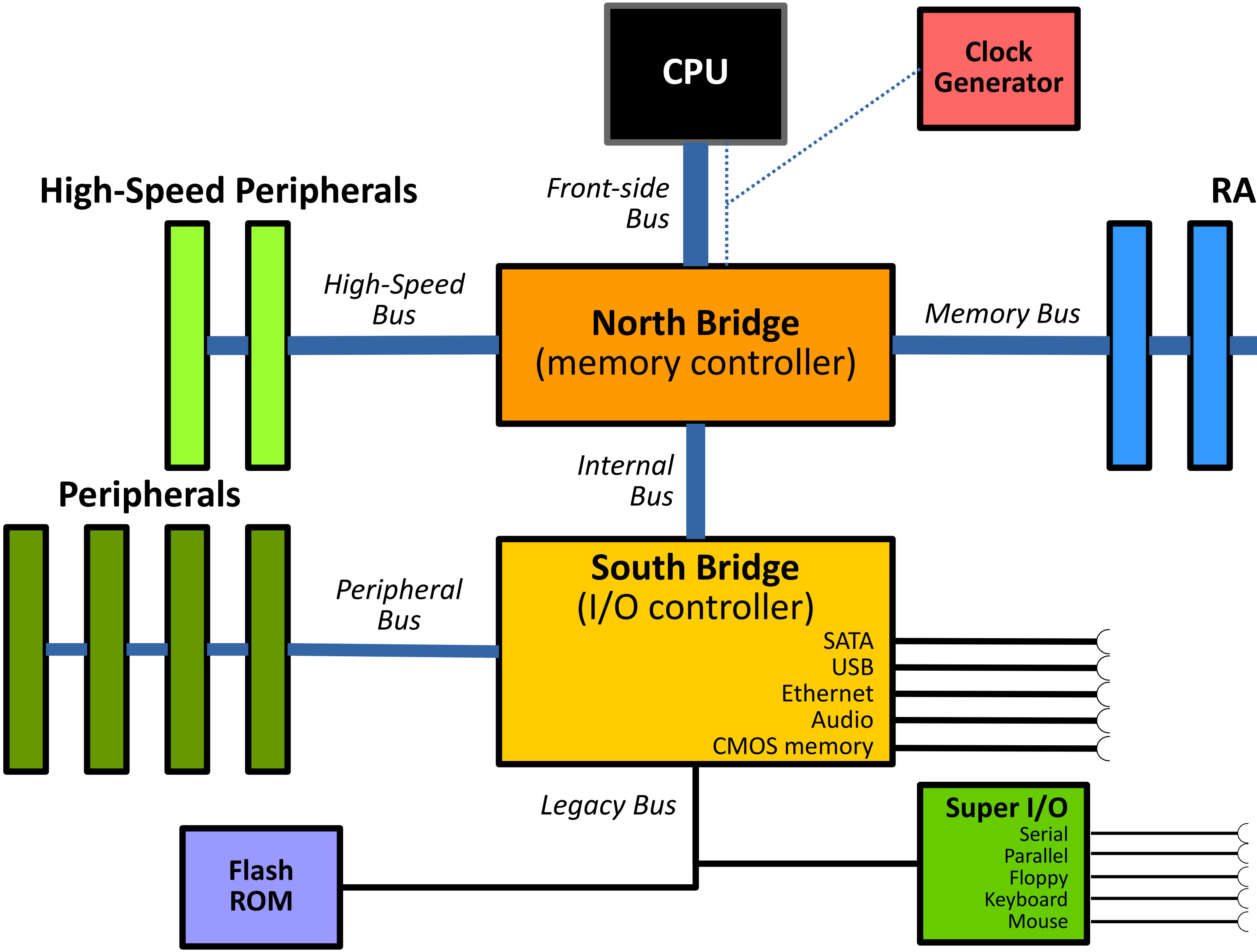
[4]

Motherboard



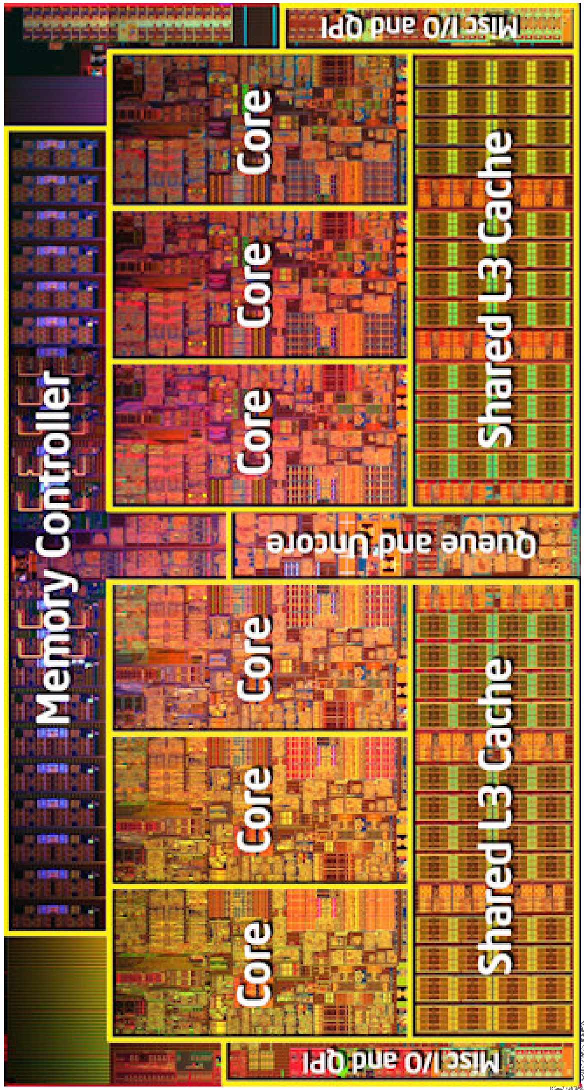
[5]

Motherboard (2)



Processor

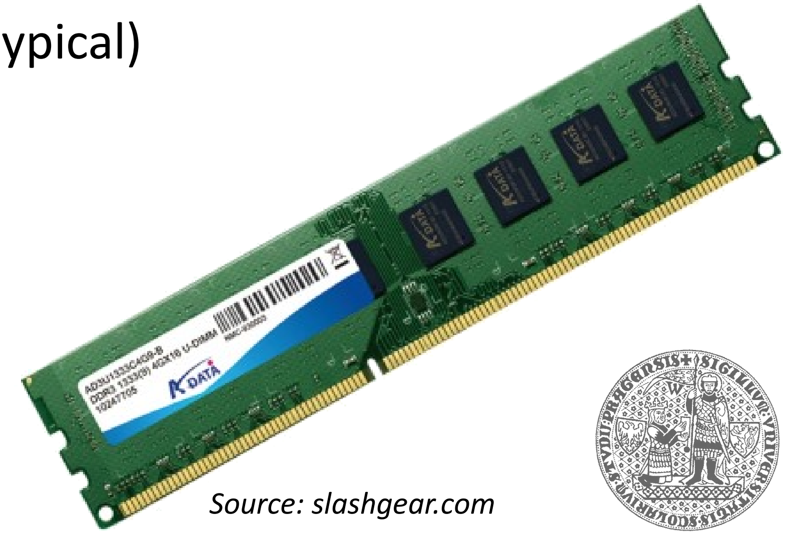
- **Key elements**
  - Data path
    (operates on data)
  - Control
    (controls data path)
  - Memory elements
    (registers and cache)
- **Intel Core i7-980X**
  - 6 cores, 12 MB L3 cache, clock frequency 3.33 GHz
  - 32 nm technology, 248 mm^2, 1.2 billion transistors



Source: intel.com

Operating memory

- Volatile
  - Running programs and data
  - Directly addressed by the processor
  - *Dynamic Random-Access Memory* (DRAM)
    - Constant access time (tens of nanoseconds)
    - Bits stored as charge in capacitors
      - Needs periodic refresh (16 Hz typical)
    - Capacity in gigabytes



Source: *slashgear.com*

Operating memory (2)

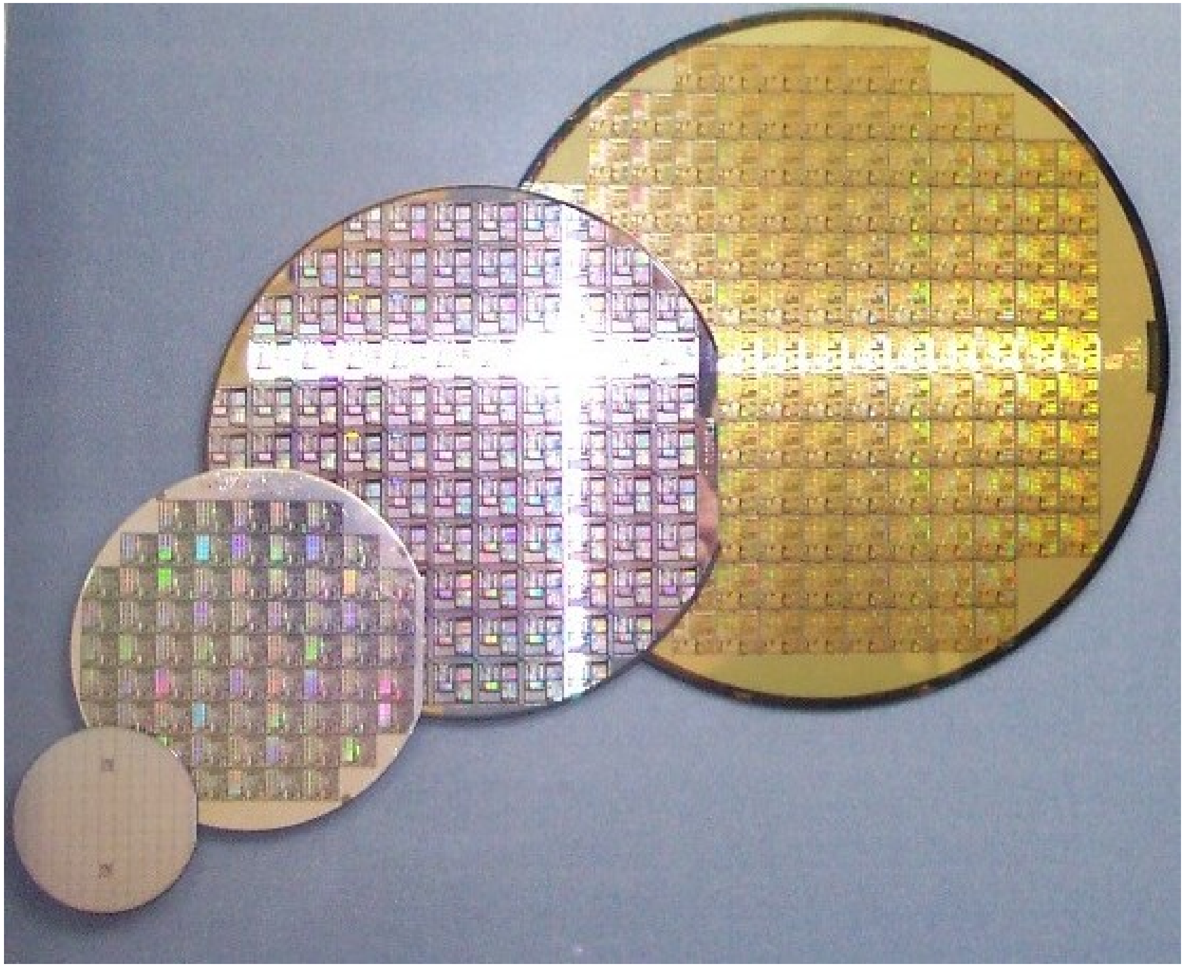
- **Volatile**
  - *Static Random-Access Memory* (SRAM)
    - Implemented using two-state flip flops (requires 4 to 6 transistors per bit)
      - No need of periodic refresh
      - Significantly faster (units of nanoseconds), significantly lower density, significantly higher cost
    - Processor caches and register
    - Other kinds of processor-internal memory

Processor and memory technology

- **Transistor**
  - Basic building block
    - Discrete (a controllable switch) instead of analog (amplifier) application
- **Integrated circuit**
  - Multiple transistors on a single chip
    - Additional parts (capacitors, resistors, etc.)
  - Better technology → smaller dimensions → higher level of integration → higher processor speed and higher memory capacity

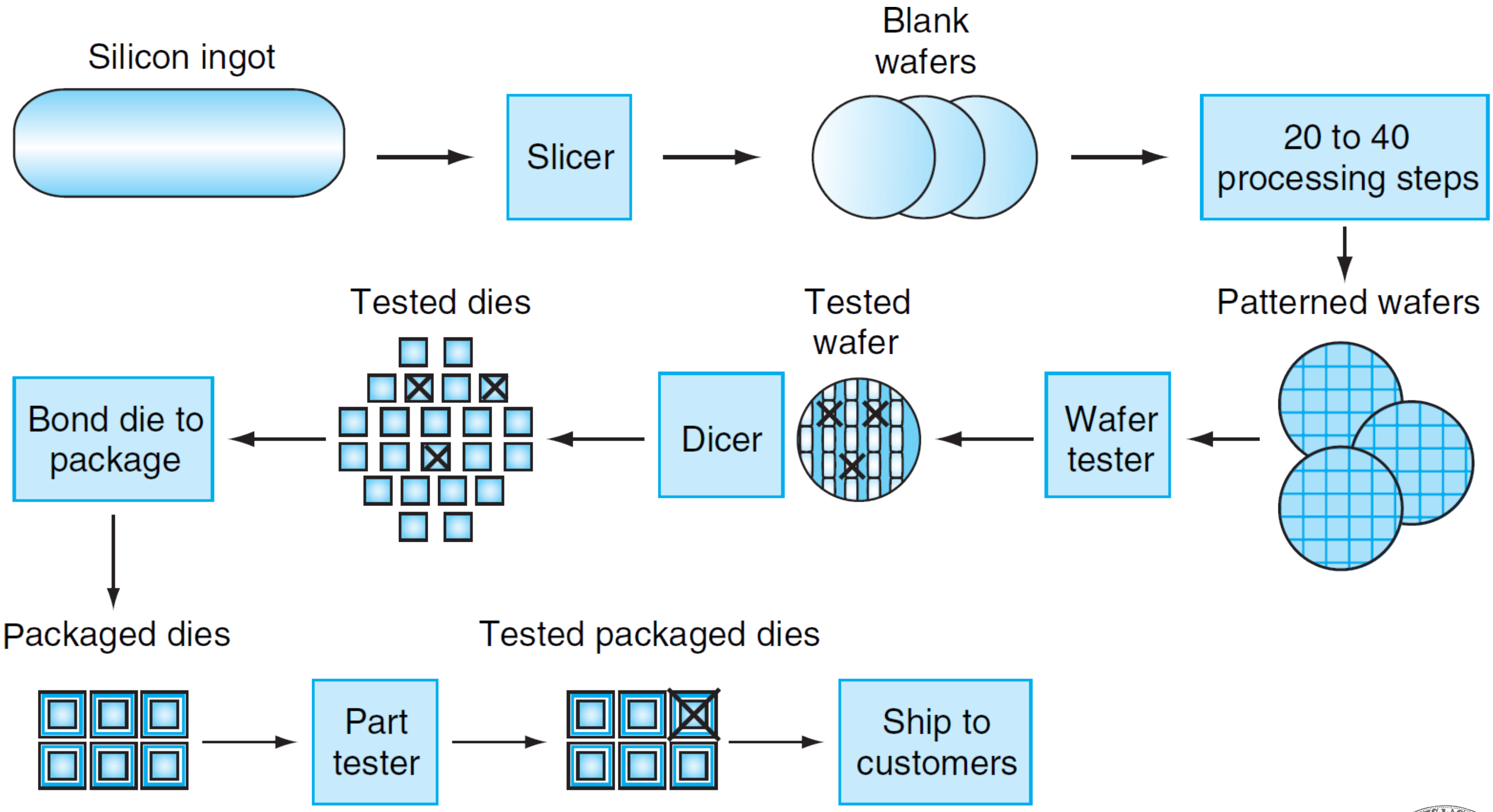
Processor and memory technology (2)

[6]



[7]

Processor and memory technology (3)

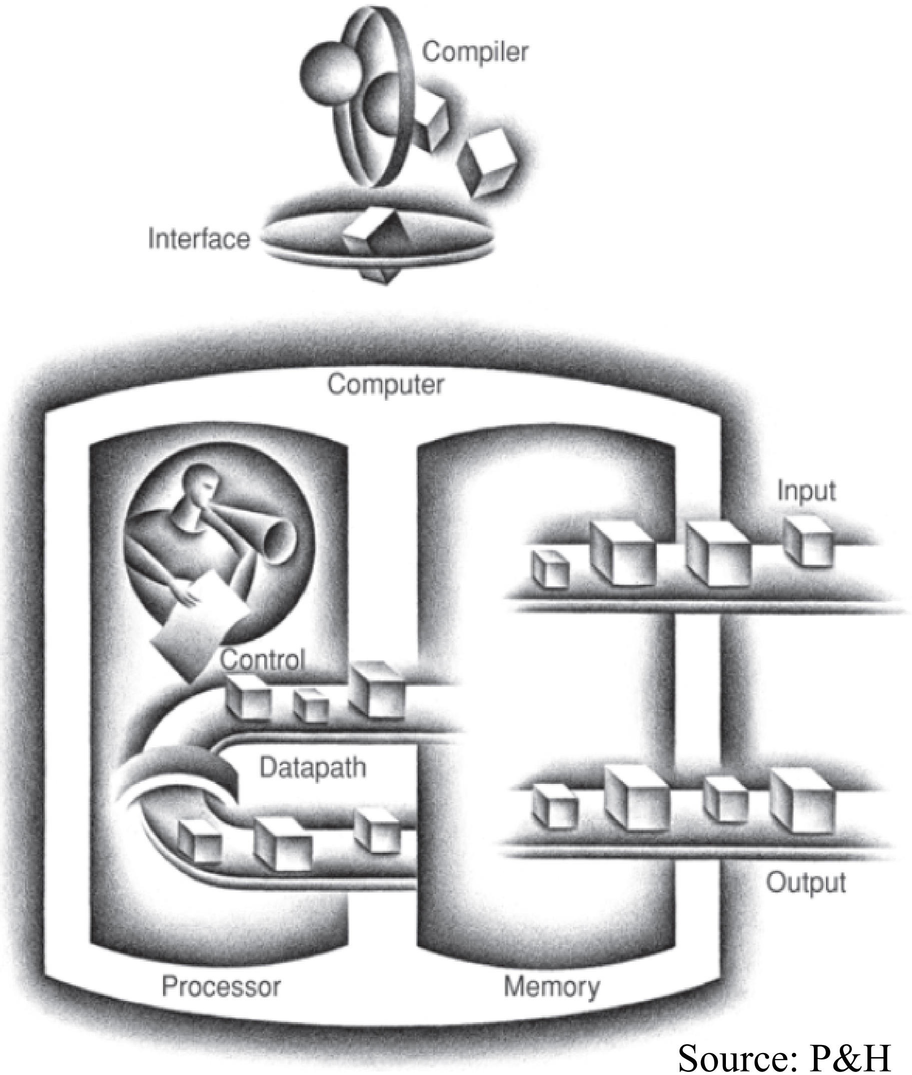


Source: P&H

Secondary storage

- **Persistent**
  - Data retained without power
  - Data files and executables
  - Not directly addressable by CPU ☠ (I/O devices, controlled by a program – operating system)
  - Hard drive
    - Magnetic rotational medium
    - Sector-based addressing (chunks of 512 B or 4 KB), access times in tens of milliseconds (not constant)
  - *Solid-State Drive* (SSD), flash memory
    - Solid (non-moving), transistor-based persistent storage (*floating-gate MOSFET*)
    - Asymmetric read/write operations (read individual bits, write large blocks), constant access time in tens to hundreds of microseconds

Basic computer organization



Source: P&H

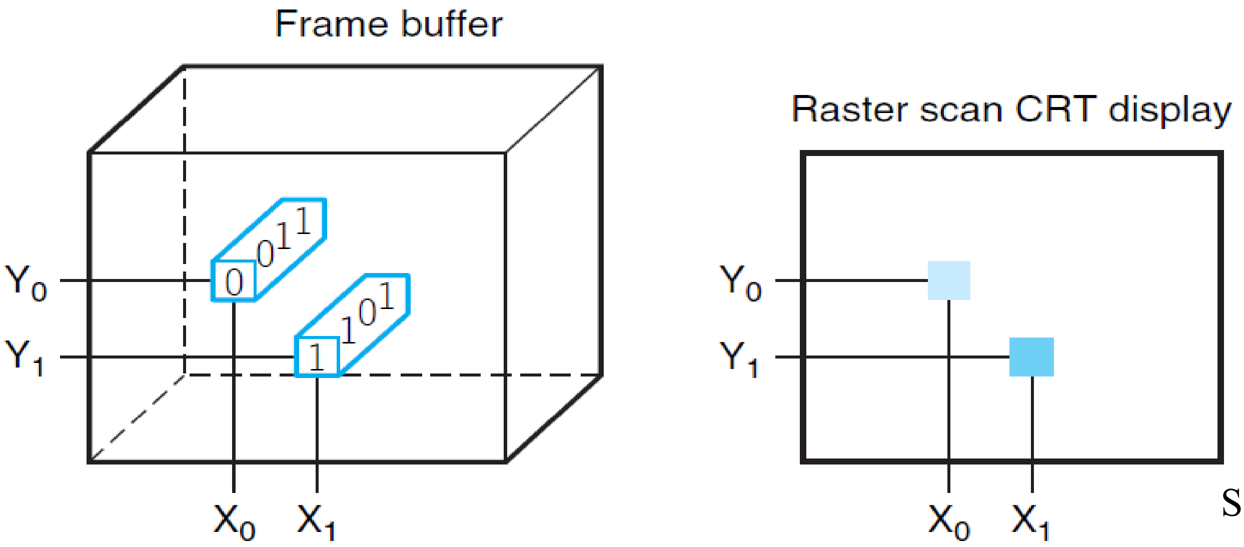
- **Computer**
  - input
  - output
  - memory
  - processor
    - data path
    - control
- **Technology independent**
  - First both today‘s and past computers

Inputs and outputs

- **Input devices**
  - Keyboard, mouse, tablet, fingerprint reader, joystick, camera, ...
- **Output devices**
  - CRT display, LCD panel, graphic card, printer
- **Input/output devies**
  - Network interface card, hard drive, sound card, camera, force-feedback steering wheel, ...

Graphical screen output

- **Framebuffer (memory on the graphic card)**
  - Every place in memory (or a group of places) corresponds to a pixel on the screen
  - Contents of the place determines color
  - Size of the place determines color resolution



Source: P&H

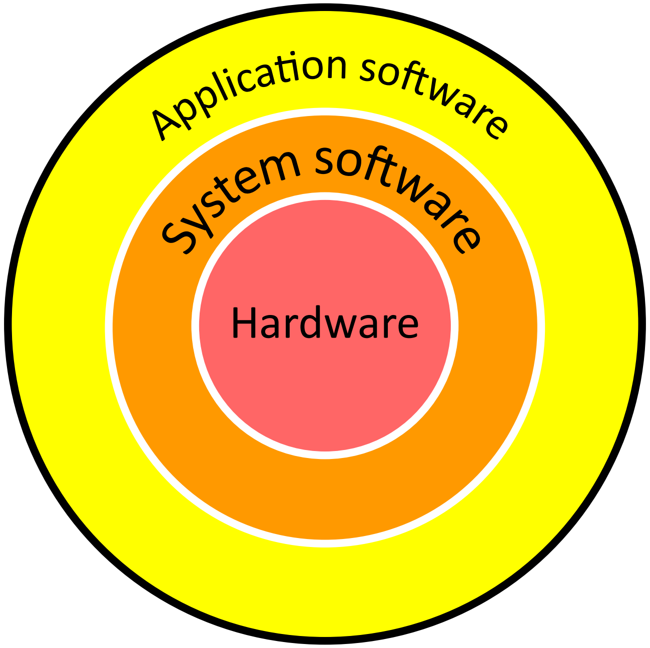
Below your program

From power-on to running applications

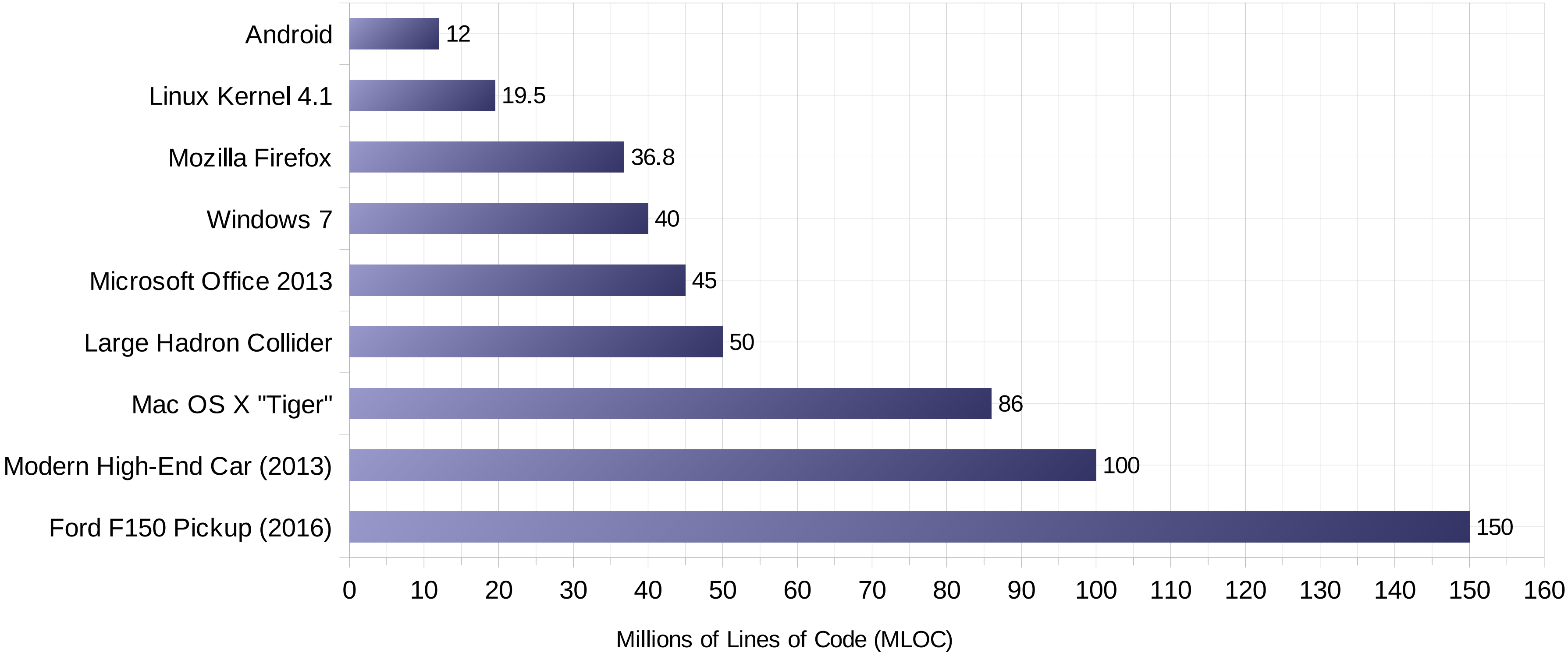
- **Firmware**
  - BIOS (Basic Input/Output System)
- **Operating system loader**
  - Boot sector
  - Boot loader
- **Operating system**
- **User interface/desktop environment**
- **Application**

100s of 1000s of lines of code

- **Application software**
  - Text editor, spread sheet, ...
  - User interface libraries
- **System software**
  - Operating system
    - Input/output operations
    - Memory and storage management
    - Resource sharing
  - Firmware
- **Hardware**
  - Processor, memory, I/O devices



100s of 1000s of lines of code



Source: https://informationisbeautiful.net/visualizations/million-lines-of-code (data as of 2016)

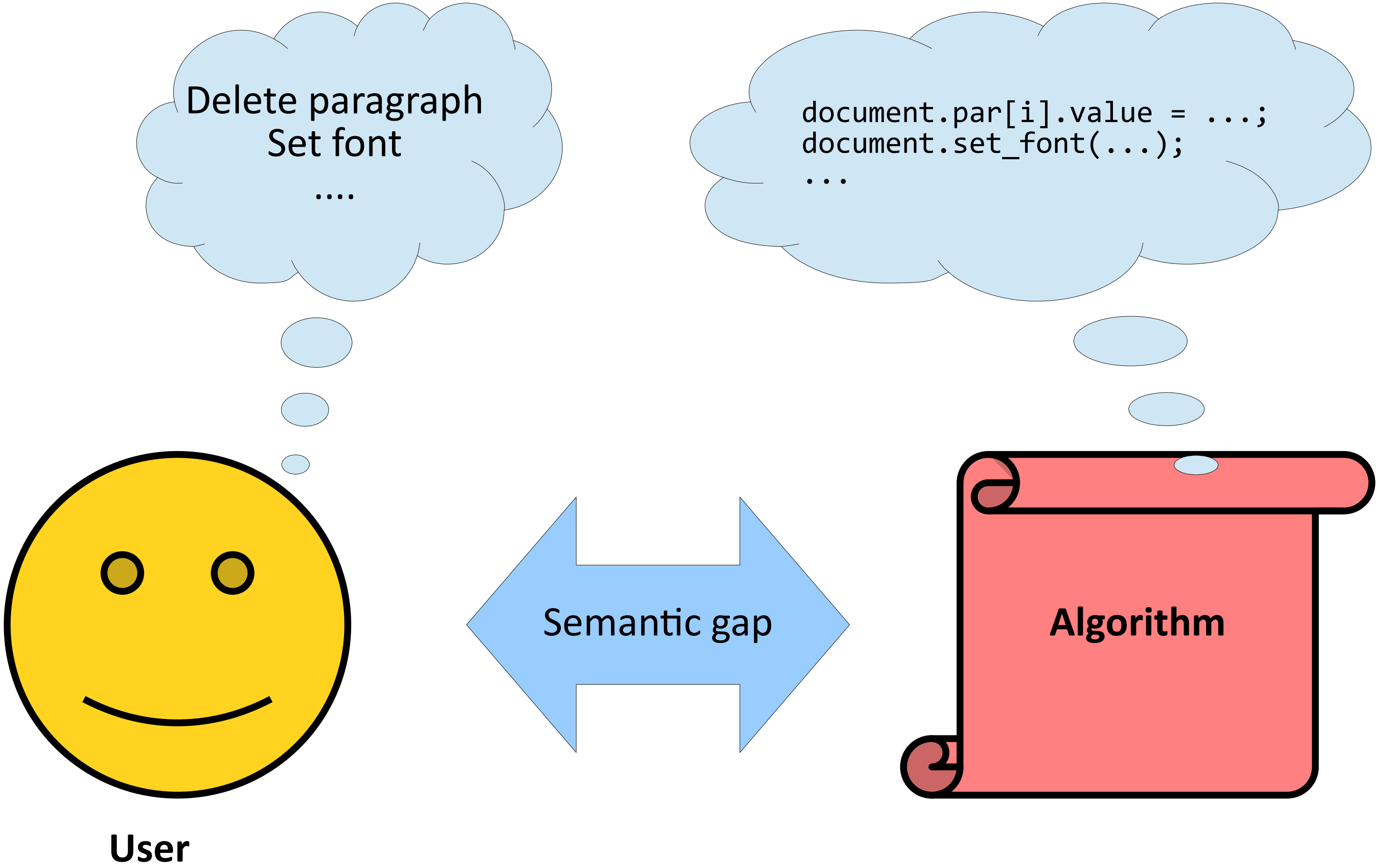
Basic concept of computer architecture

Abstraction

Abstraction

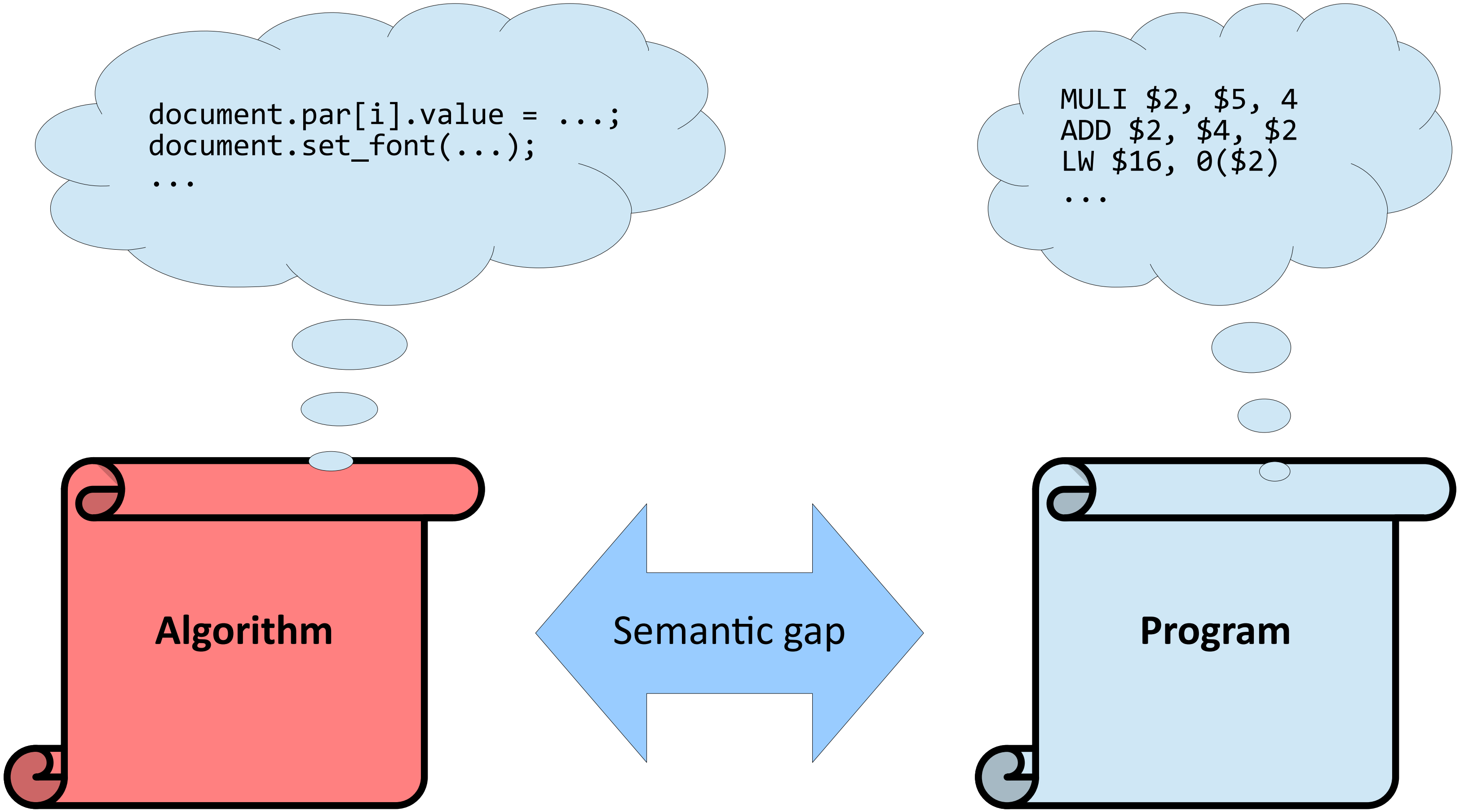
- **Required to bridge semantic gaps**
  - From a concrete (technical) language to an abstract (general) language
  - Expressing the same using more general terms while encapsulating internal details and preserving accuracy
    - More concise and compact expression
  - *„An abstraction is one thing that represents several real things equally well.“* (Edsger Dijkstra)

From a user to an algorithm



```
document.par[i].value = ...;
document.set_font(...);
...
```

From an algorithm to a program



```
document.par[i].value = ...;
document.set_font(...);
...
```

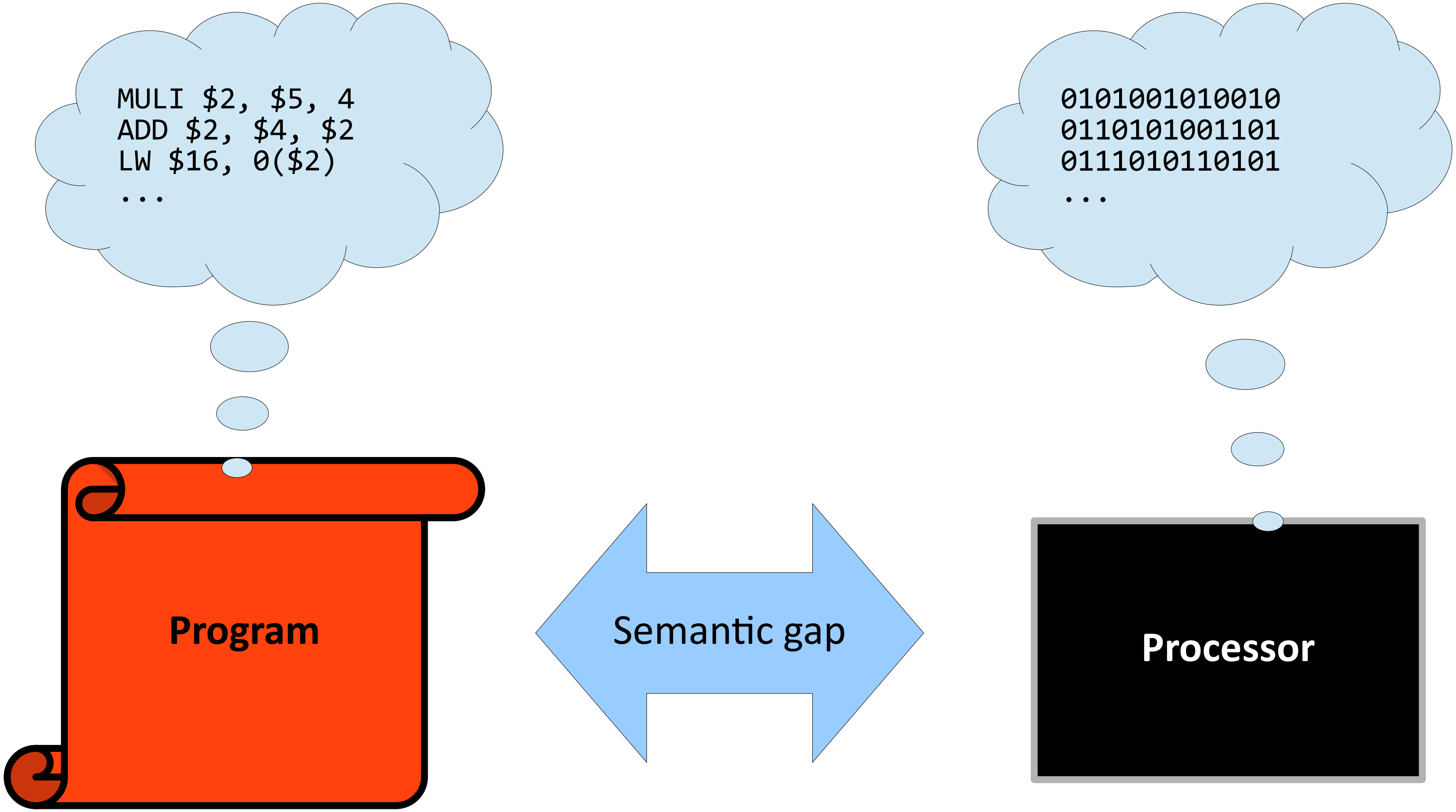
```
MULI $2, $5, 4
ADD $2, $4, $2
LW $16, 0($2)
...
```

Algorithm

Semantic gap

Program

From a program to machine code



```
MULI $2, $5, 4
ADD $2, $4, $2
LW $16, 0($2)
...
```

```
0101001010010
0110101001101
0111010110101
...
```

Example: Swap k-th and (k+1)-th element

- **High-level programming language**

```
void swap(unsigned int array[], unsigned int k) {
   unsigned int old = array[k];
   array[k] = array[k + 1];
   array[k + 1] = old;
}
```

Example: Swap k-th and (k+1)-th element

- Assembler representation for MIPS

```
swap:
    sll  $a1, $a1, 2
    addu $a1, $a1, $a0
    lw   $v0, 0($a1)
    lw   $v1, 4($a1)
    sw   $v1, 0($a1)
    sw   $v0, 4($a1)
    jr   $ra
```

Example: Swap k-th and (k+1)-th element

- **Assembler representation for SuperH**

```
swap:
   shll2 r5
   mov   r4,r1
   add   r5,r1
   mov.l @r1,r2
   add   #4,r5
   add   r5,r4
   mov.l @r4,r3
   mov.l r3,@r1
   rts
   mov.l r2,@r4
```

Example: Swap k-th and (k+1)-th element

- **Assembler representation for x86-64**

```
swap:
   movslq %esi, %rsi
   leaq   (%rdi, %rsi, 4), %rdx
   leaq   4(%rdi, %rsi, 4), %rax
   movl   (%rdx), %ecx
   movl   (%rax), %esi
   movl   %esi, (%rdx)
   movl   %ecx, (%rax)
   retq
```

Example: Swap k-th and (k+1)-th element

- Machine code for MIPS

```
00000000000001010010100010000000
00000000101001000010100000100001
10001100101000100000000000000000
10001100101000110000000000000100
10101100101000100000000000000100
10101100101000110000000000000000
00000011111000000000000000001000
```

Example: Swap k-th and (k+1)-th element

- Machine code for SuperH

```
0000100001000101
0100001101100001
0101110000110001
0001001001100010
0000010001110101
0101110000110100
0100001001100011
0011001000100001
0000101100000000
0010001000100100
```

Example: Swap k-th and (k+1)-th element

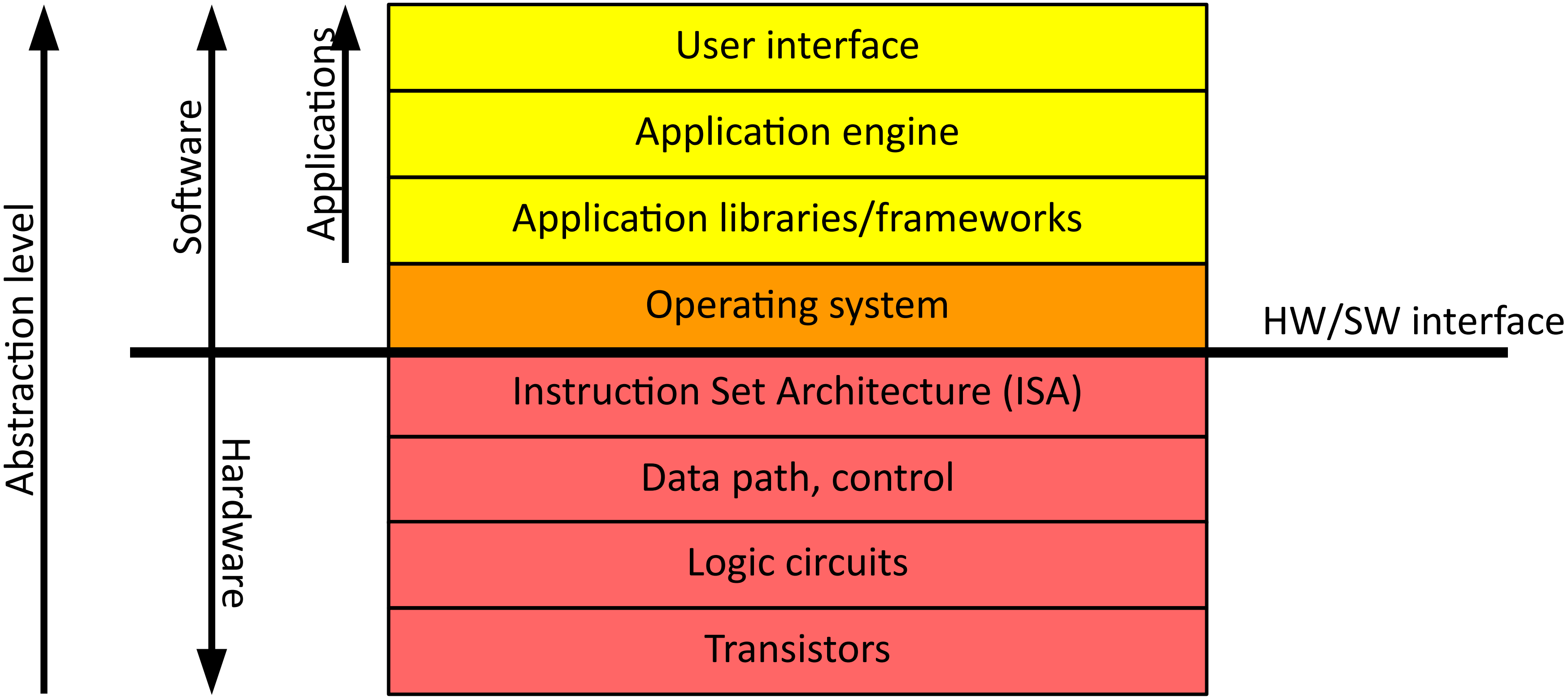
- **Machine code for x86-64**

```
010010000110011111110110
01001000100011010001010010110111
0100100010001101010001001011011100000100
1000101100001010
1000101101110000
1000100101110010
1000100100001000
11000111
```

Implementation

- **The opposite of abstraction**
  - Concretization
  - From computer architecture to concrete computer
  - High-level language
    - Block diagrams, functional description of circuits
  - Low-level language
    - Circuit diagrams connecting electronic components, masks for producing semiconductor elements in an integrated circuit
  - „Machine code“
    - Physical realization of a computer

Abstraction layers in a computer



User interface

Application engine

Application libraries/frameworks

Operating system

HW/SW interface

Instruction Set Architecture (ISA)

Data path, control

Logic circuits

Transistors

Abstraction level

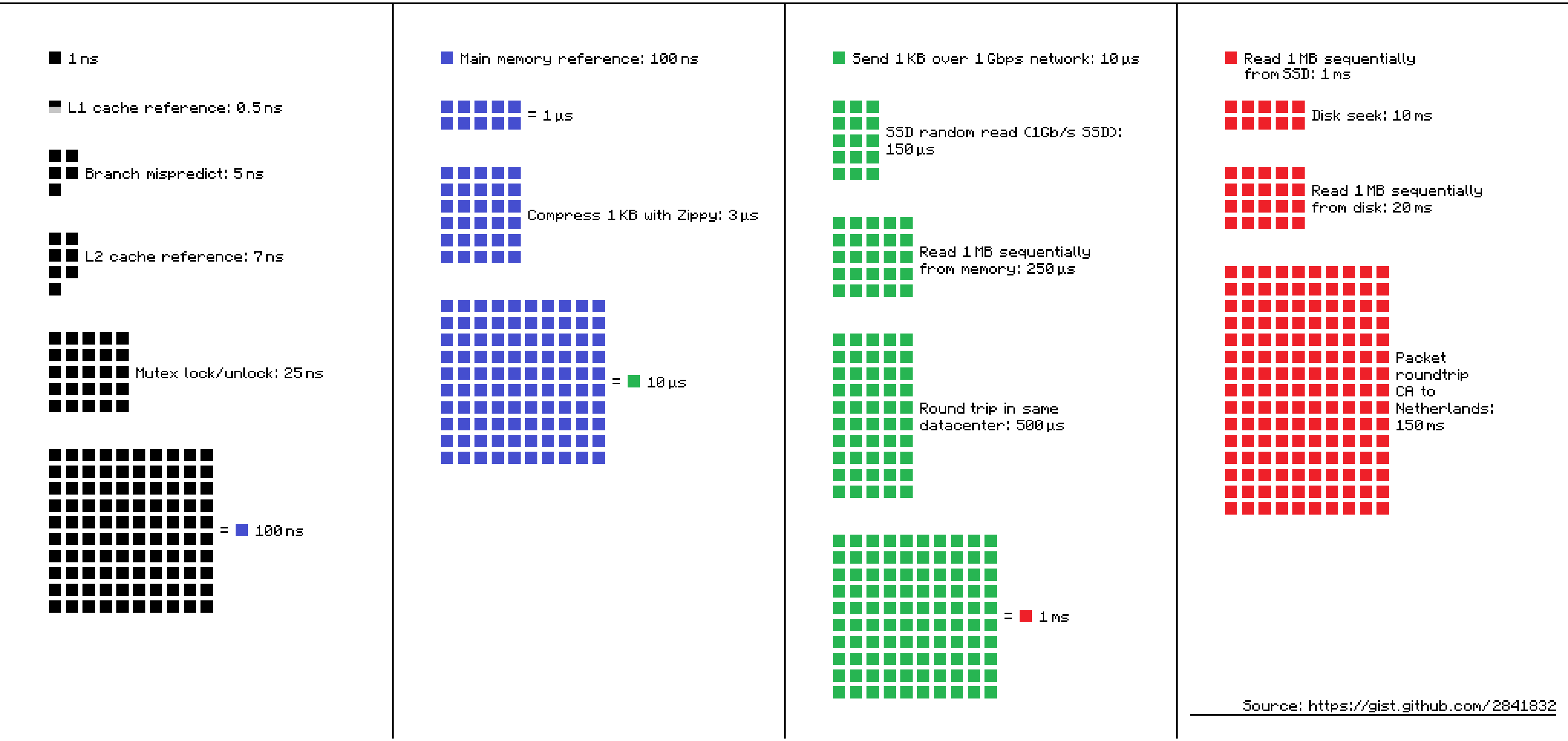
Software

Hardware

Applications

Beware: abstraction is (only) a tool!

Latency Numbers Every Programmer Should Know



Source: https://gist.github.com/2841832