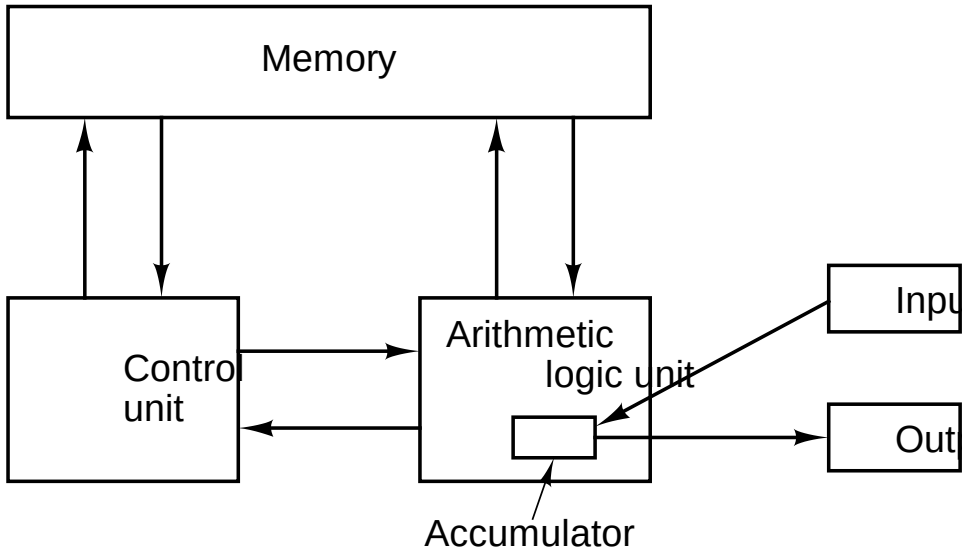
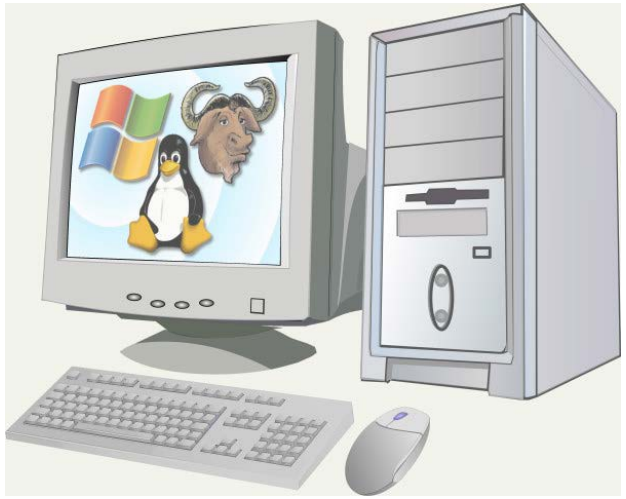
TI II: Computer Architecture Introduction

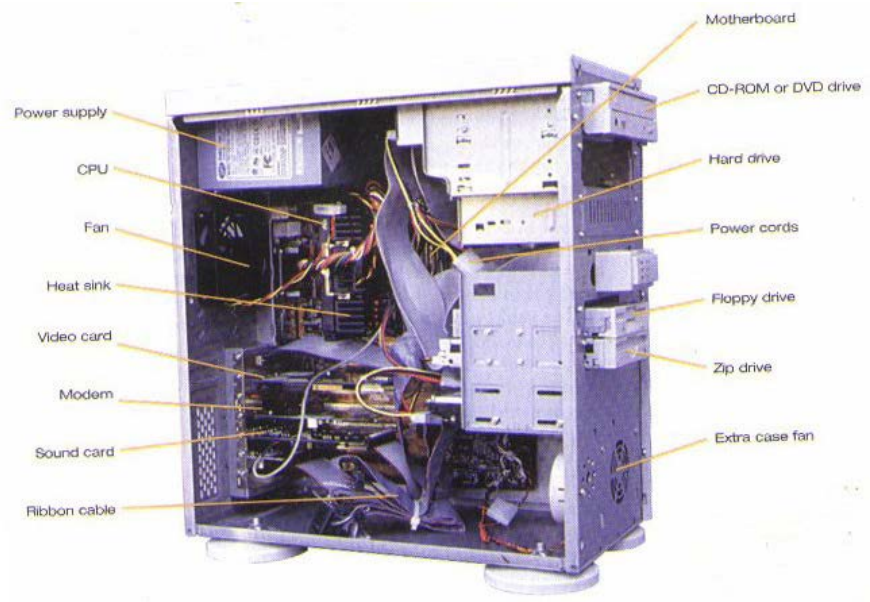


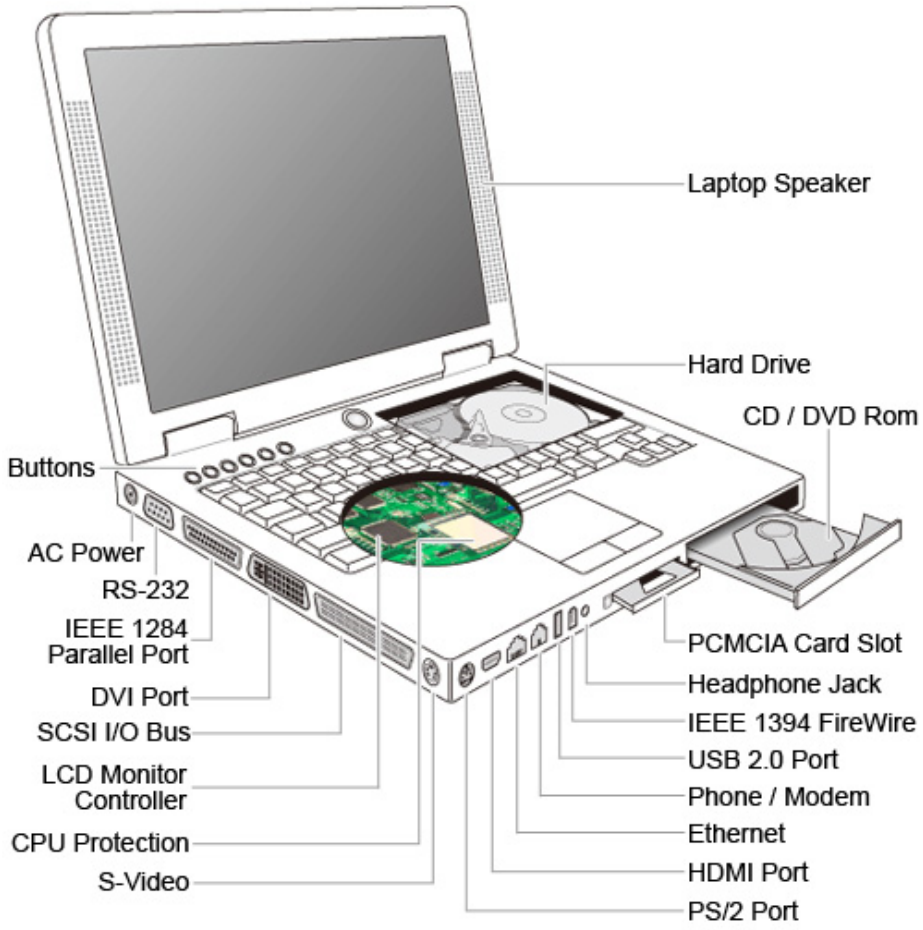
Single Processor Systems
Historical Background
Classification / Taxonomy
Architectural Overview
Examples
The Layered Computer Model

SINGLE PROCESSOR SYSTEMS

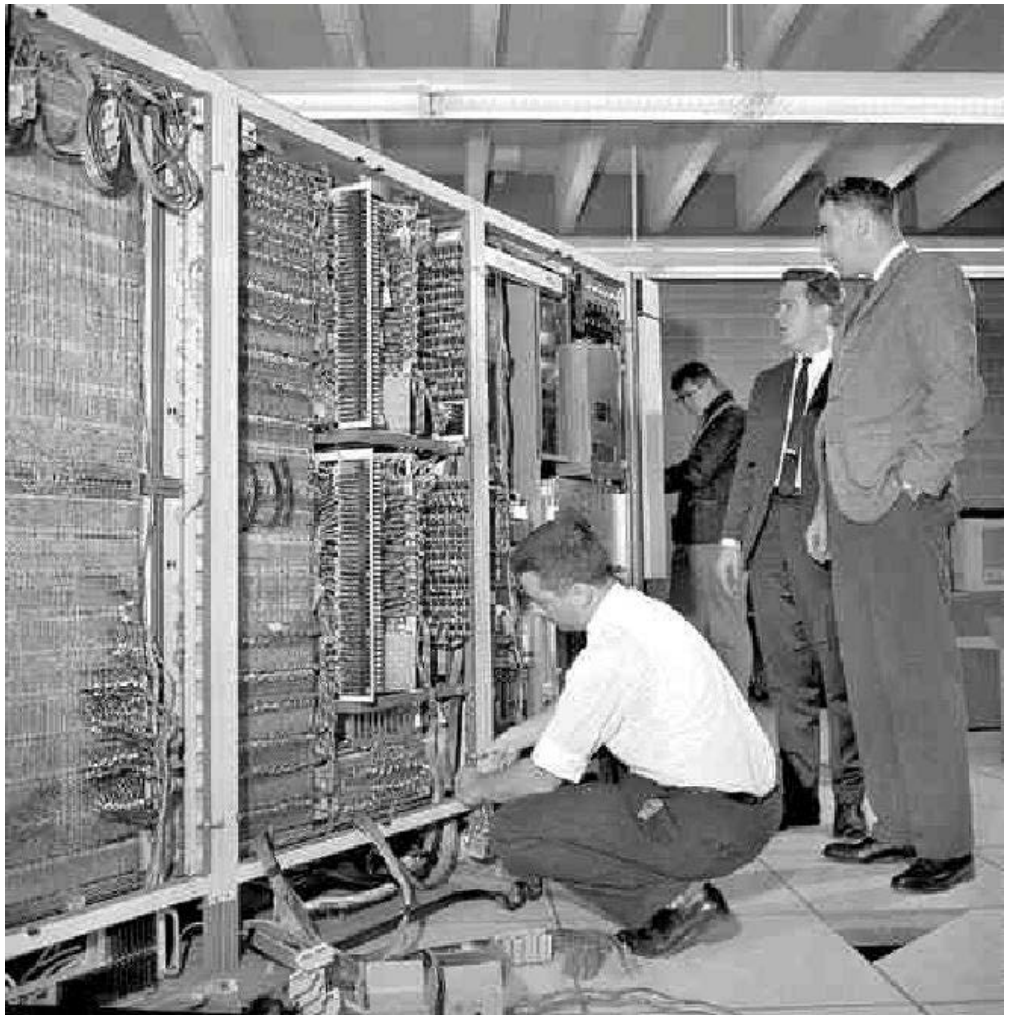
Computer System





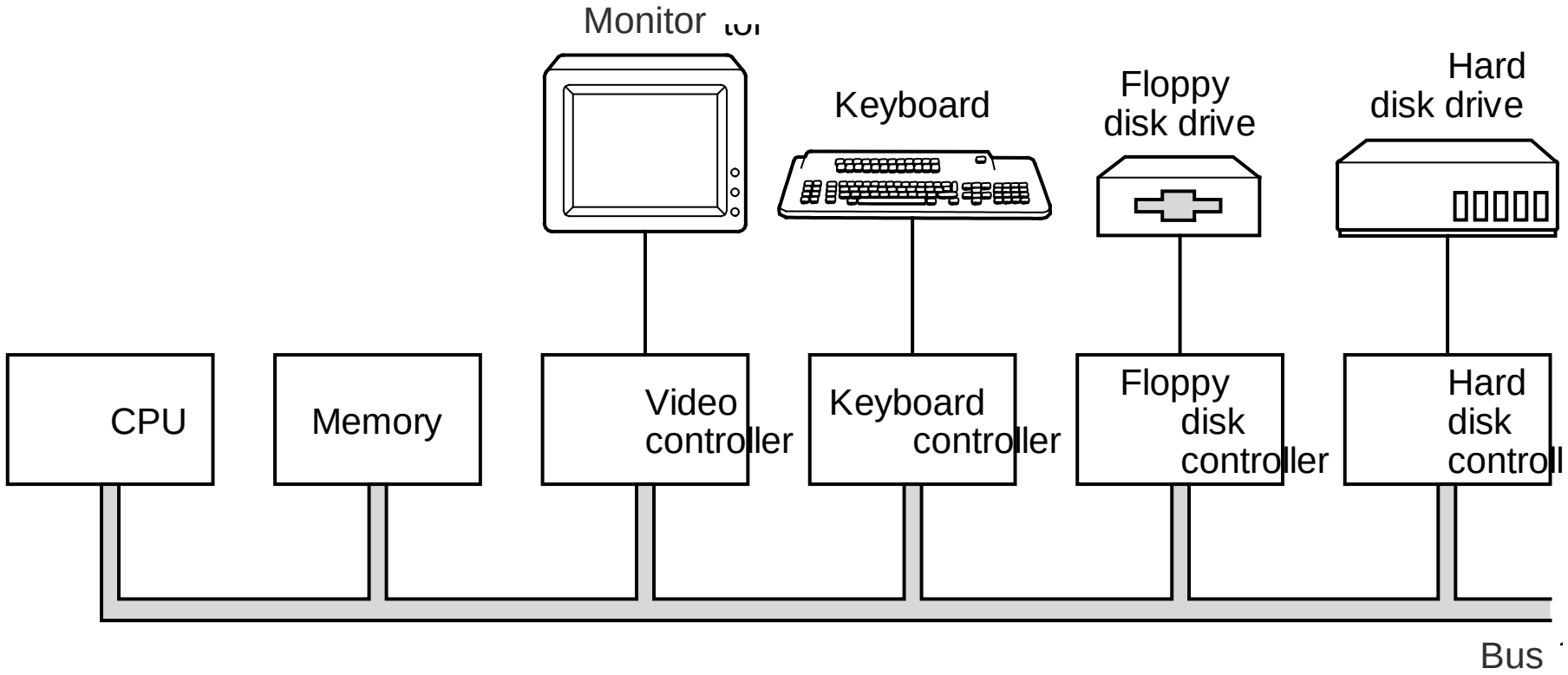


Computer System

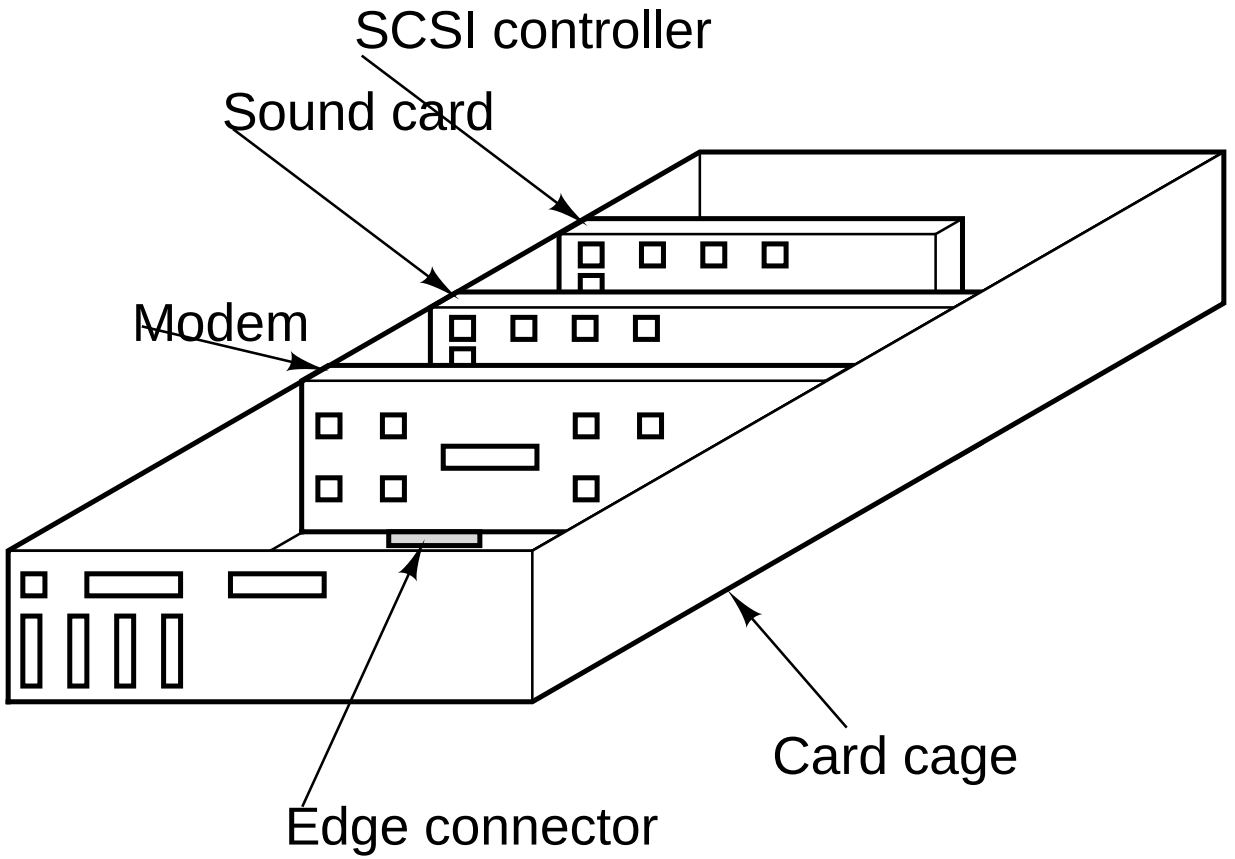


Computer System

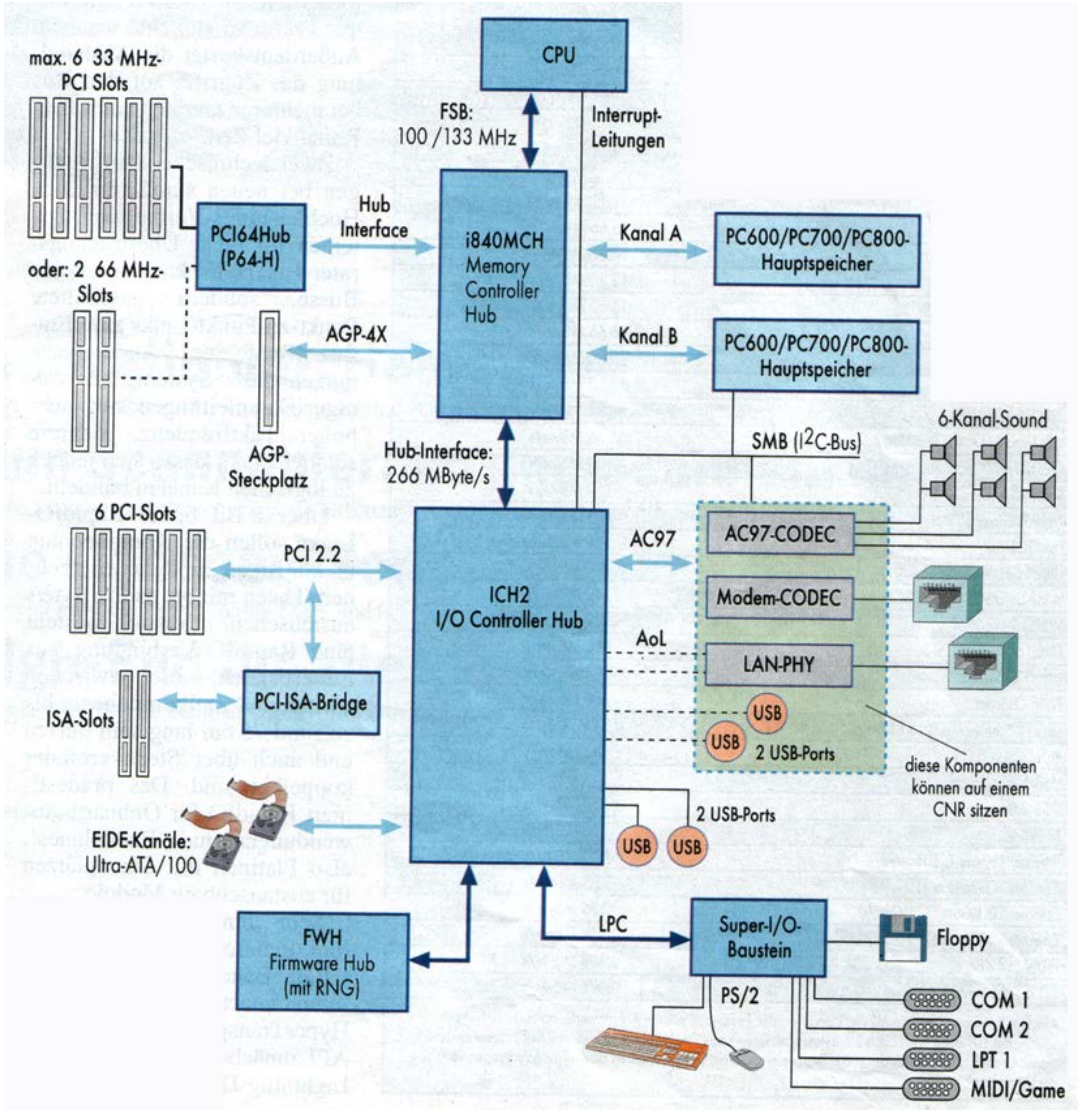
Classical architecture of microcomputers



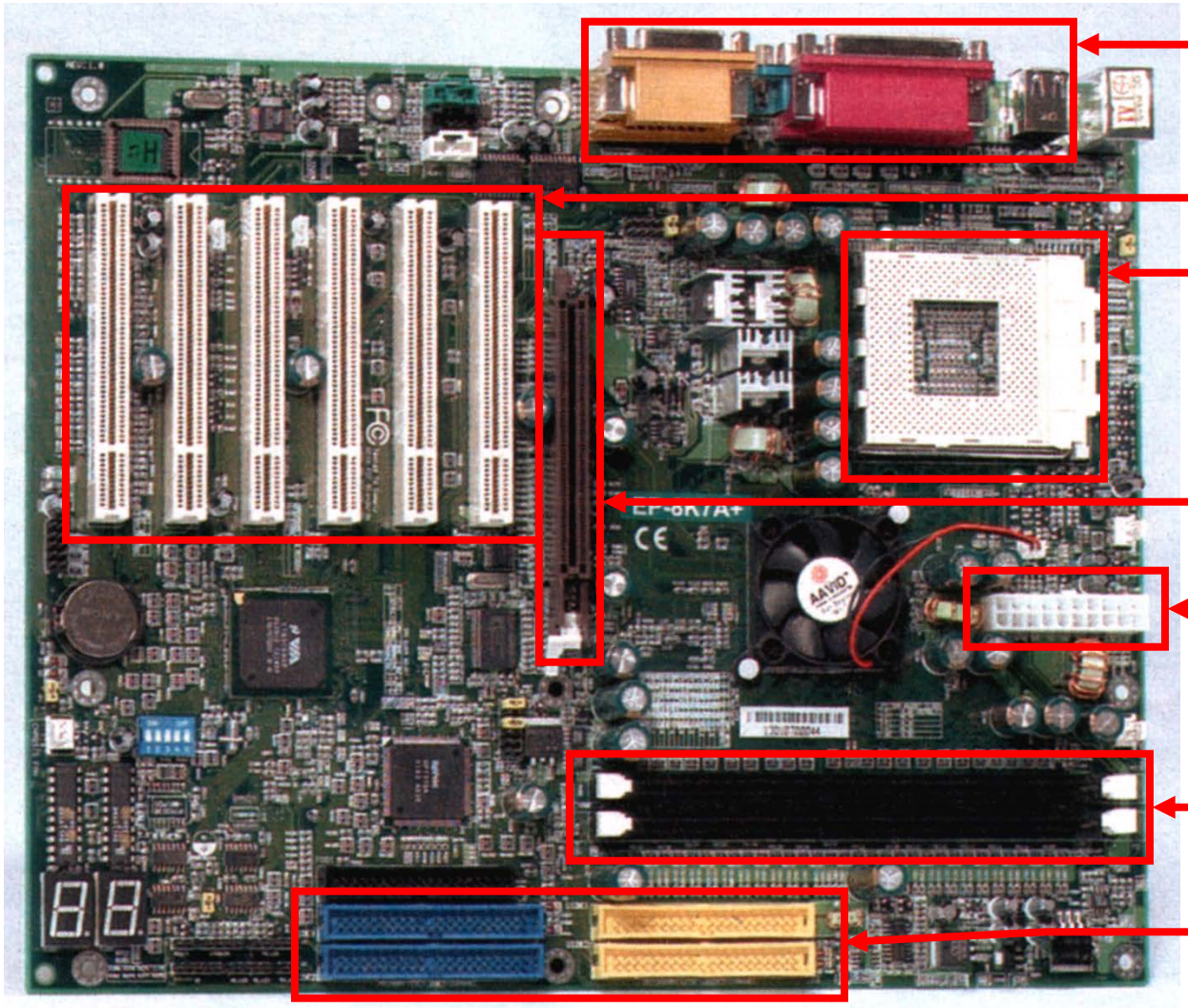
Classical idea of a modular computer system



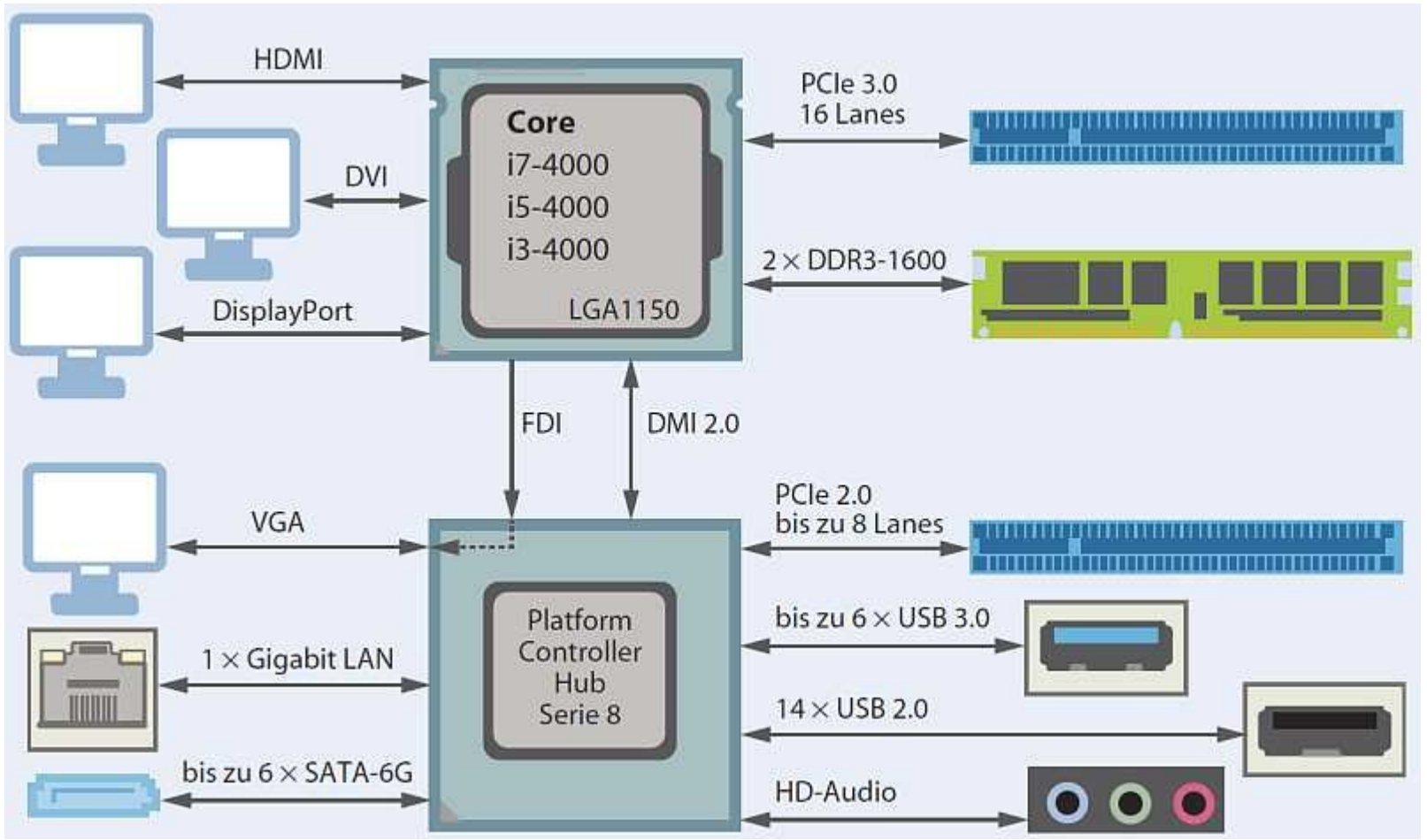
A complete single processor system



A complete single processor system

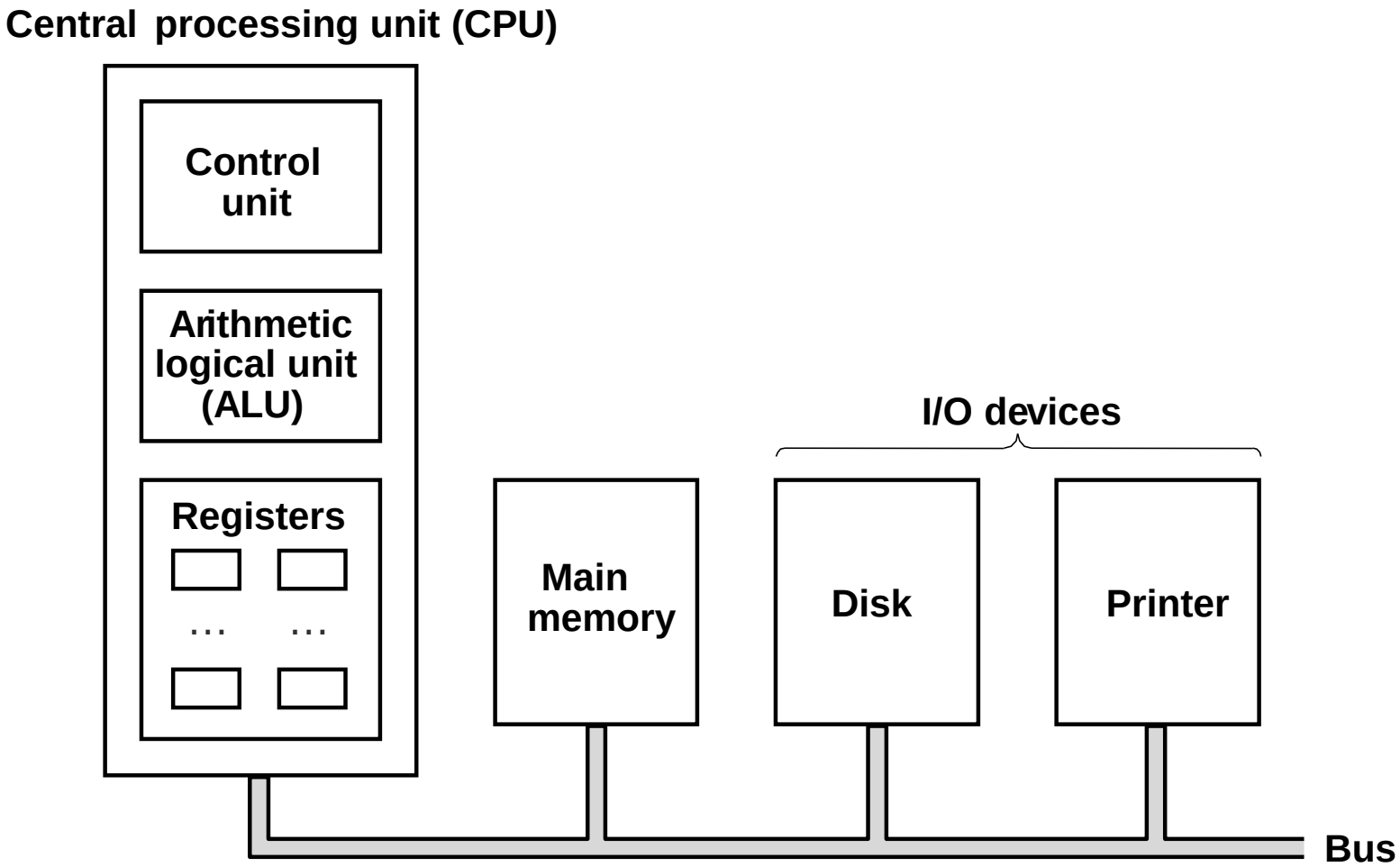


PC Architecture 2014



www.heise.de

Simple single processor system



Do we still have this today? Think of, e.g., mobile phone with main CPU, radio modem, graphics accelerator, GPS, WLAN, ...

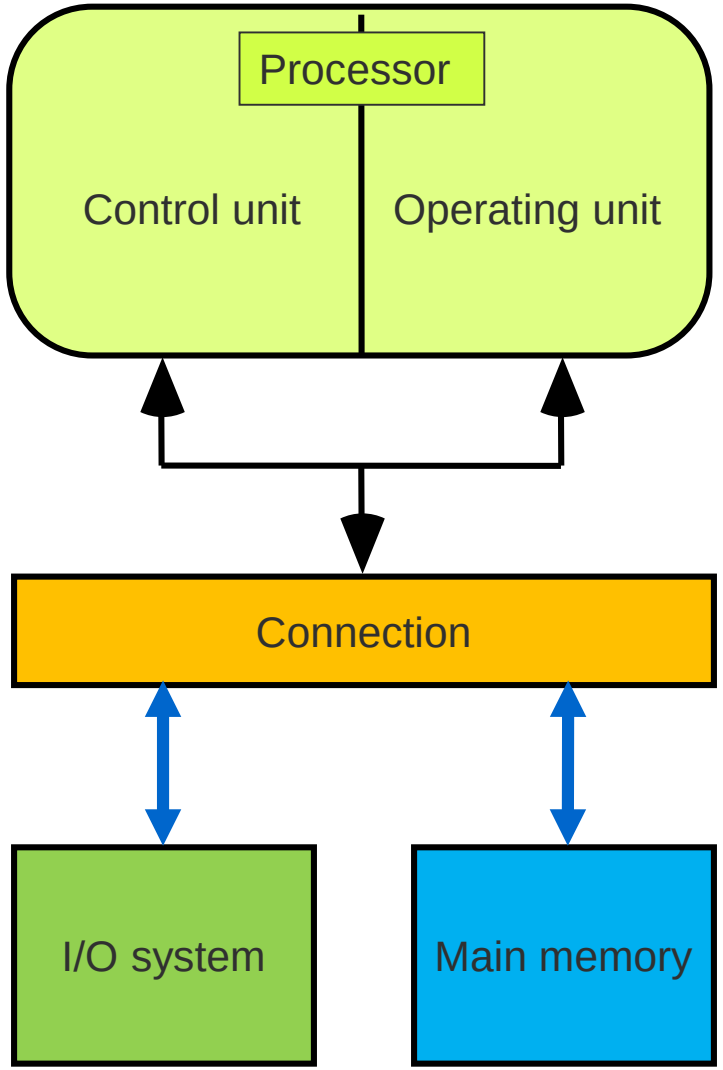
THE VON NEUMANN ARCHITECTURE

The von Neumann architecture

The von Neumann architecture forms the basis of many hardware architectures presented in this course.

The architecture comprises the following main components

- Central processing unit
  - Control unit
  - ALU / Operating unit
- Memory
- Input/Output units
- Interconnection



The von Neumann architecture

Central control of the computer

A computer consists of **several functional units** (central processing unit, memory, input/output unit, connection)

The computer is **not tailored to a single problem** but a general purpose machine. In order to solve a problem a program is stored in the memory (“program controlled universal computer”) – yes, today this sounds so simple…

IMPORTANT

- Instructions (the program) and data (input and output values) are stored in the **same** memory.
- The memory consists of memory cells with a fixed length, each cell can be addressed individually.

The von Neumann architecture

Processor, central unit (CPU: "central processing unit")

- Controls the flow and execution of all instructions

Control unit

- Interprets the CPU instructions
- Generates all control commands for other components

Arithmetic Logical Unit (ALU)

- Executes all instructions (I/O and memory instructions with the help of these units)

Input/Output system

- Interface to the outside world
- Input and output of program and data

Memory

- Storage of data and program as sequence of bits

Interconnection

The von Neumann Architecture

PRINCIPLE OF OPERATION OF A COMPUTER

Principle of Operation of a Computer

At any time the CPU executes only a **single instruction**. This instruction can only manipulate a **single operand**.

- Traditionally, this is called **SISD** (Single Instruction Single Data).

Code and data are stored in the **same memory** without any distinction. There are no memory protection mechanisms – programs can destroy each other, programs can access arbitrary data.

Memory is unstructured and is addressed linearly. Interpretation of memory content depends on the context of the current program only.

Two phase principle of instruction processing:

- During the interpretation phase the content of a memory cell is fetched based on a **program counter**. This content is then interpreted as an instruction (note: this is a pure interpretation!).
- During the execution phase the content of a memory cell is fetched based on the address contained in the instruction. This content is then interpreted as data.

The instruction flow follows a strict sequential order.

Principle of Operation of a Computer Instruction Execution

```
public class Interpreter
{
        static int PC;                  // Program counter holds the address of the next instruction
        static int AC;                  // Register for doing arithmetic, accumulator
        static int instruction;         // Current instruction
        static int instructionType;     // Type of the current instruction, i.e. what to do
        static int dataLocation;        // Address of the data for the instruction
        static int data;                // Holds the operand
        static boolean runBit = true;   // Bit used to halt the computer

        public static void interpreter(int memory[], int startingAddress)
        {
                PC = startingAddress;                                   // Initialize the program
                while( runBit ) {
                        instruction = memory[PC];                       // Fetch next instruction
                        PC = PC + 1;                                    // Increment PC
                        instructionType = getInstructionType(instruction);  // Determine instruction
                        dataLocation = findData(instruction, instructionType); // Locate data
                        if( dataLocation >= 0 )                         // No operand if -1
                                data = memory[dataLocation];            // Fetch data
                        execute(instructionType, data);                 // Execute instruction
                }
        }
}
```

Principle of Operation of a Computer Instruction Execution

Example: interpreter(memory, 256);

memory[256] = 80	memory[261] = 7
memory[257] = 0	memory[262] = 20
memory[258] = 5	memory[263] = 2
memory[259] = 80	memory[264] = 0
memory[260] = 1	memory[265] = 1

Byte sequence: 80 0 5 80 1 7 20 2 0 1

Bit sequence: 01010000 00000000 00000101
01010000 00000001 00000111
00010100 00000010 00000000 00000001

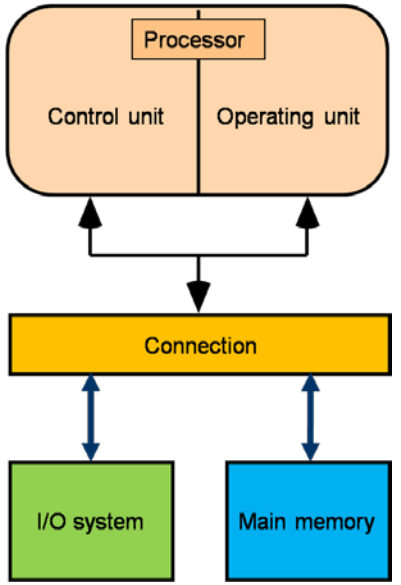
Assembler representation (MMIX Style)

```
LDB   $0,5
LDB   $1,7
ADD   $2,$0,$1
```

High-level programming language representation

Z = X + Y

Pros and Cons of the von Neumann architecture



Advantages

- Principle of minimal hardware requirements
- Principle of minimal memory requirements

Disadvantages

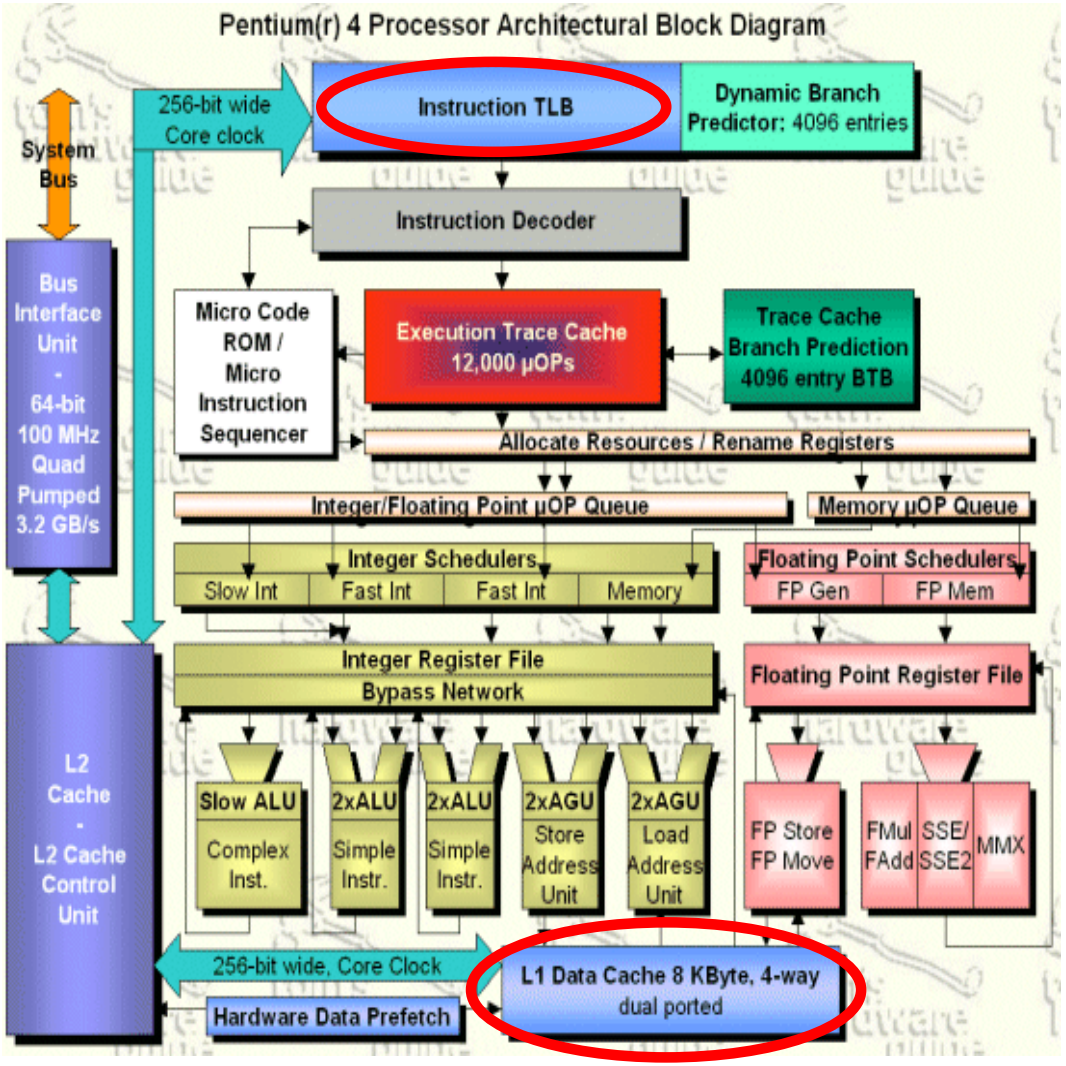
- The main interconnection (memory ↔ CPU) is the central bottleneck: the "**von Neumann bottleneck**"
- Programs have to take into account the sequential data flow across the von Neumann bottleneck → Influences on higher programming languages ("intellectual bottleneck")
- Low structuring of data (a long sequence of bits…)
- The instruction determines the **operand type**
- No memory protection

Look around:
Not everything is von Neumann!

Basic “von Neumann-architecture”: data and program are stored in the same memory

Typical for the PC architecture…

- well, depends on your viewpoint…



Harvard Architecture

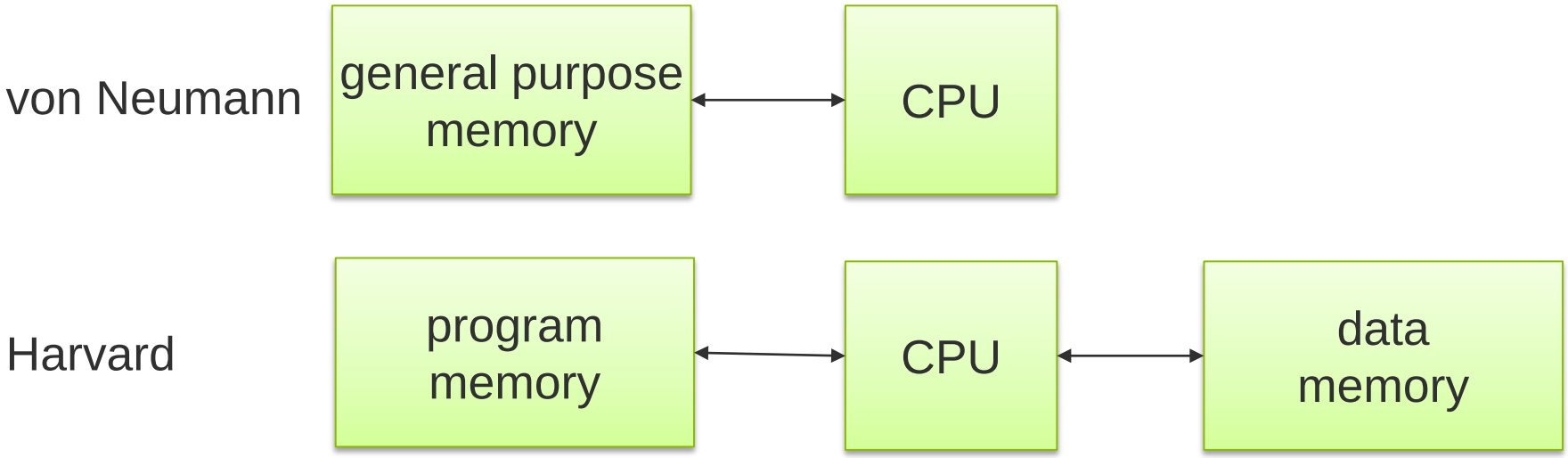
Classical definition of the Harvard architecture

- Separation of data and program memory

Most processors with microarchitecture

- von Neumann from the outside
- Harvard from the inside
- Reason
  - Different time scales and locality when caching data and instructions

High-level comparison

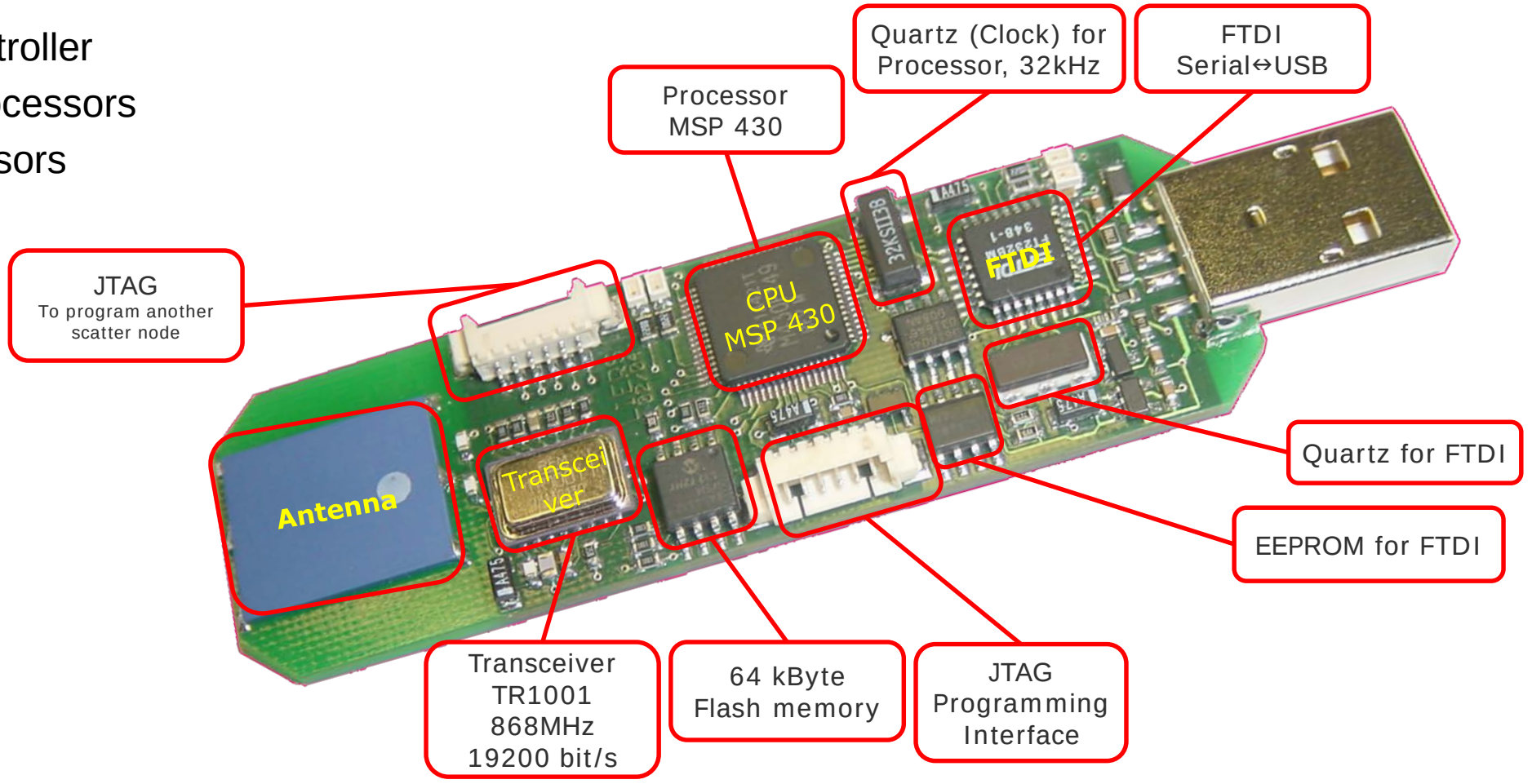


MICRO COMPUTERS

Special purpose micro processors

micro processors for special applications exist next to universal microprocessors (standard-micro-processs):

- Micro controller
- Signal processors
- Coprocessors



Definition of a Micro computer system

Micro processor system:
- Digital system, using a micro processor as central control and/or arithmetic unit

Micro computer:
- includes a micro processor that communicates with memory, controllers and interfaces for external devices using the system bus.

Special cases of microcomputers:
- Single-chip microcomputer
  - All components of the microcomputer are located on one chip.
- Single-circuit microcomputer (dt. Ein-Platinen-Mikrocomputer)
  - All components of the microcomputer are on one circuit board.

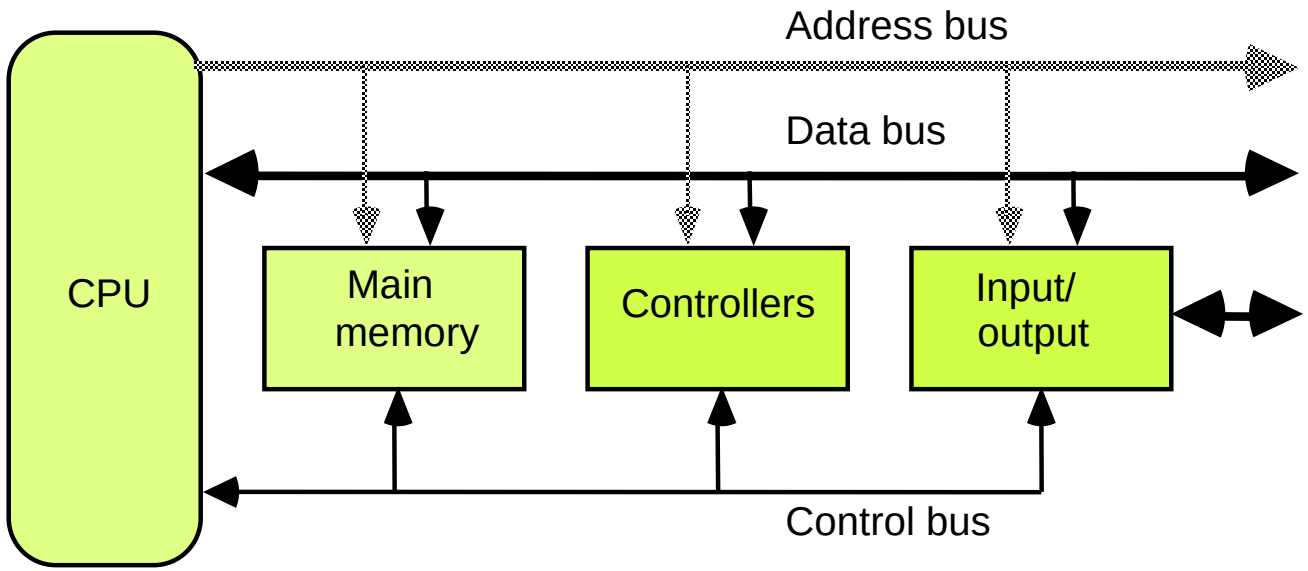
Microcomputer system:
- Microcomputer with connected **external devices**
- Can be small – think of the Internet of Things!

Basic concept of a micro computer

The IBM PC is a modified von Neumann architecture and was introduced by IBM fall 1981.

The interconnection structure was realized by a **bus**.

- The bus connects the CPU with the main memory, several controllers and the input/output system.



Interconnection

The bus comprises the

- data bus
- address bus
- control bus

The bus is controlled by a **bus controller** and uses a buffer for data and addresses to communicate with the CPU.

The CPU fetches each instruction sequentially from the main memory.

The controllers are accessed with an I/O mapped model

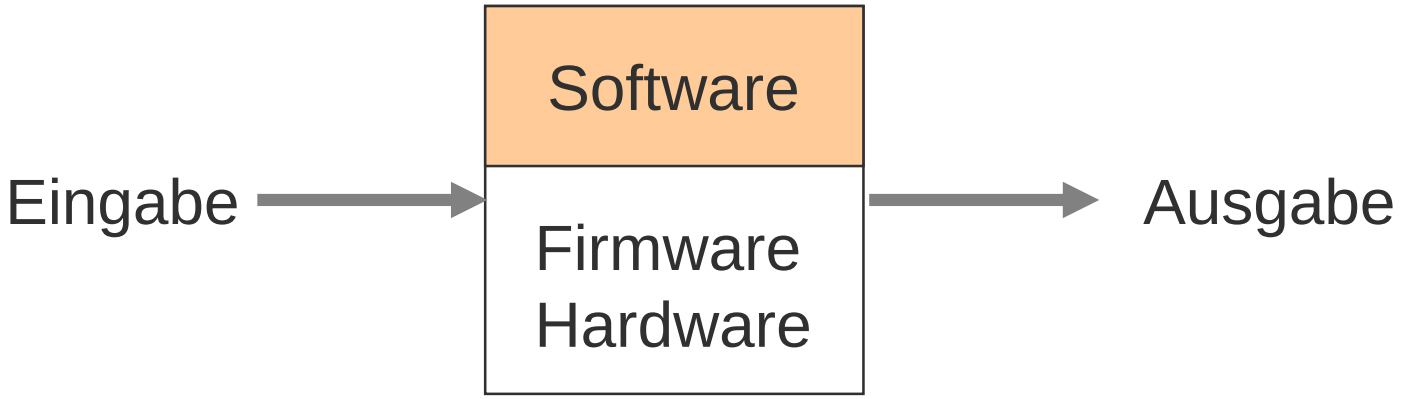
- The internal registers of a controller are accessed with special I/O instructions using port addresses.

Komponenten eines Digitalrechners

Hardware: Alle mechanischen und elektronischen Bauelemente

Software: Alle Programme, die auf dem Rechner ablaufen

Firmware: Mikroprogramme in ROMs, Mittelstellung zwischen Hardware und Software



➡Gegenstand der Forschung: Welche Funktionen in Software und welche in Hardware realisieren?

➡Hardware und Software sind logisch äquivalent!

History of Computers

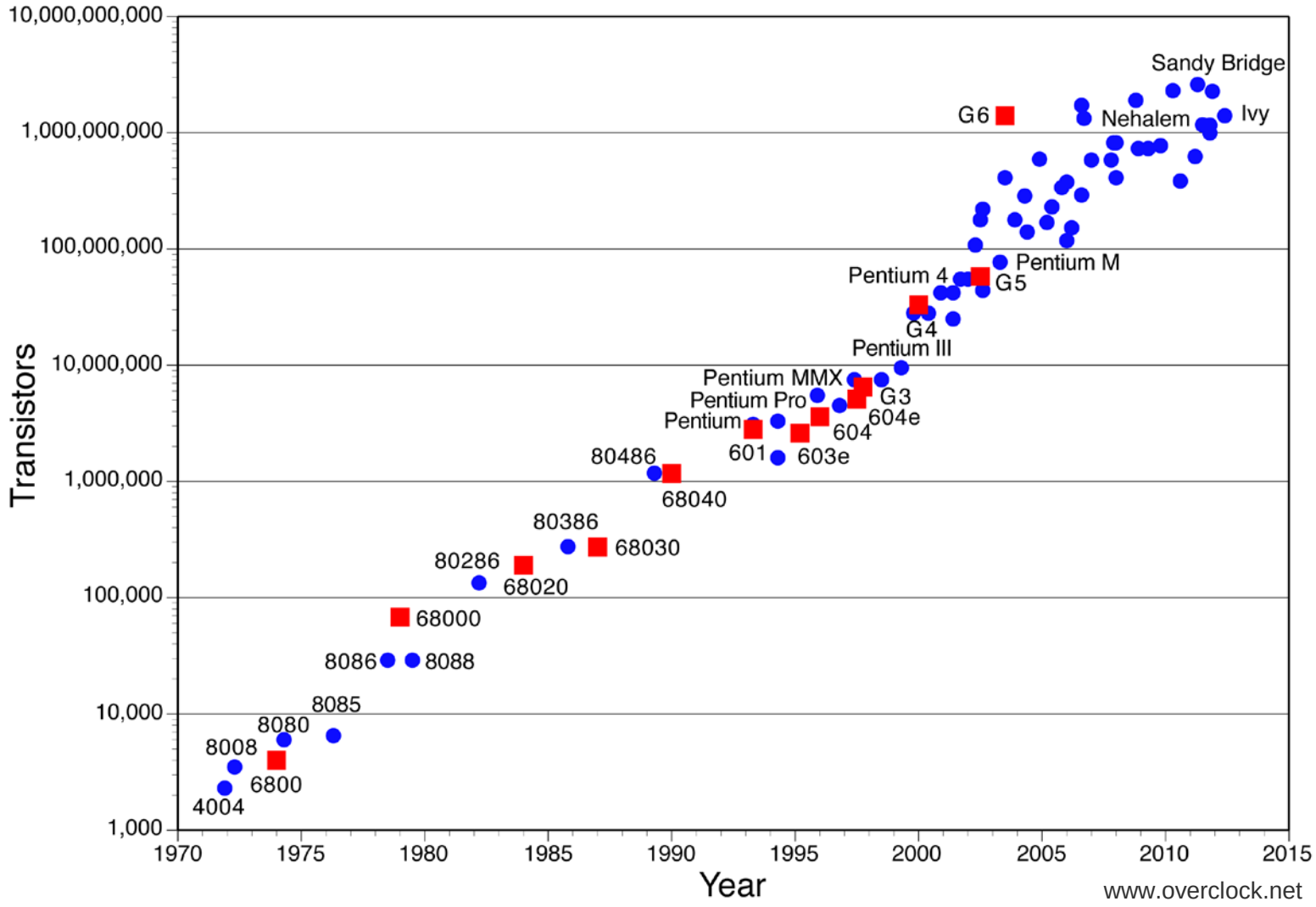
See for example:

- http://www.computerhistory.org/
- http://en.wikipedia.org/wiki/History_of_computing_hardware

„Computer history museum“ by Dzou - en.wikipedia / CC BY-SA 3.0

PERFORMANCE OF PROCESSORS

Anwendung vom Mooreschen Gesetz auf CPU-Chips



„Alle 18 Monate verdoppelt sich die verfügbare Rechenleistung“

Performance increase, price decrease

Year	Name	Size (cu. ft.)	Power (watts)	Performance (adds/sec)	Memory (KB)	Price	Price-performance vs. UNIVAC	Adjusted price (2003 $)	Adjusted price-performance vs. UNIVAC
1951	UNIVAC I	1,000	125,000	2,000	48	$1,000,000	1	$6,107,600	1
1964	IBM S/360 model 50	60	10,000	500,000	64	$1,000,000	263	$4,792,300	318
1965	PDP-8	8	500	330,000	4	$16,000	10,855	$75,390	13,135
1976	Cray-1	58	60,000	166,000,000	32,000	$4,000,000	21,842	$10,756,800	51,604
1981	IBM PC	1	150	240,000	256	$3,000	42,105	$5,461	154,673
1991	HP 9000/ model 750	2	500	50,000,000	16,384	$7,400	3,556,188	$9,401	16,122,356
1996	Intel PPro PC (200 MHz)	2	500	400,000,000	16,384	$4,400	47,846,890	$4,945	239,078,908
2003	Intel Pentium 4 PC (3.0 GHz)	2	500	6,000,000,000	262,144	$1,600	1,875,000,000	$1,600	11,452,000,000

Source: Patterson, Hennessy, Computer Organization and Design

Example Processor: Itanium 2 (Madison)

410 million transistors

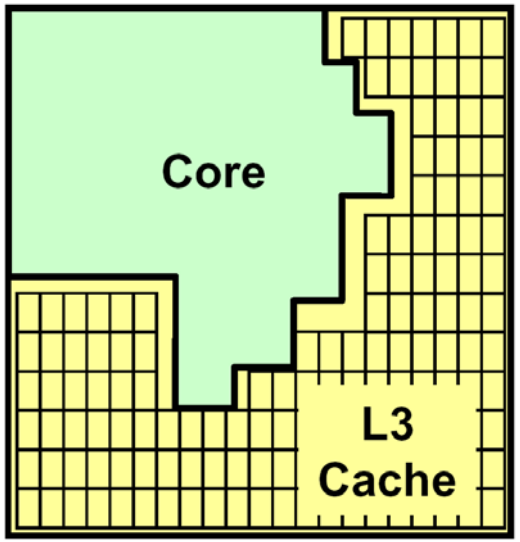
- mainly due to cache!

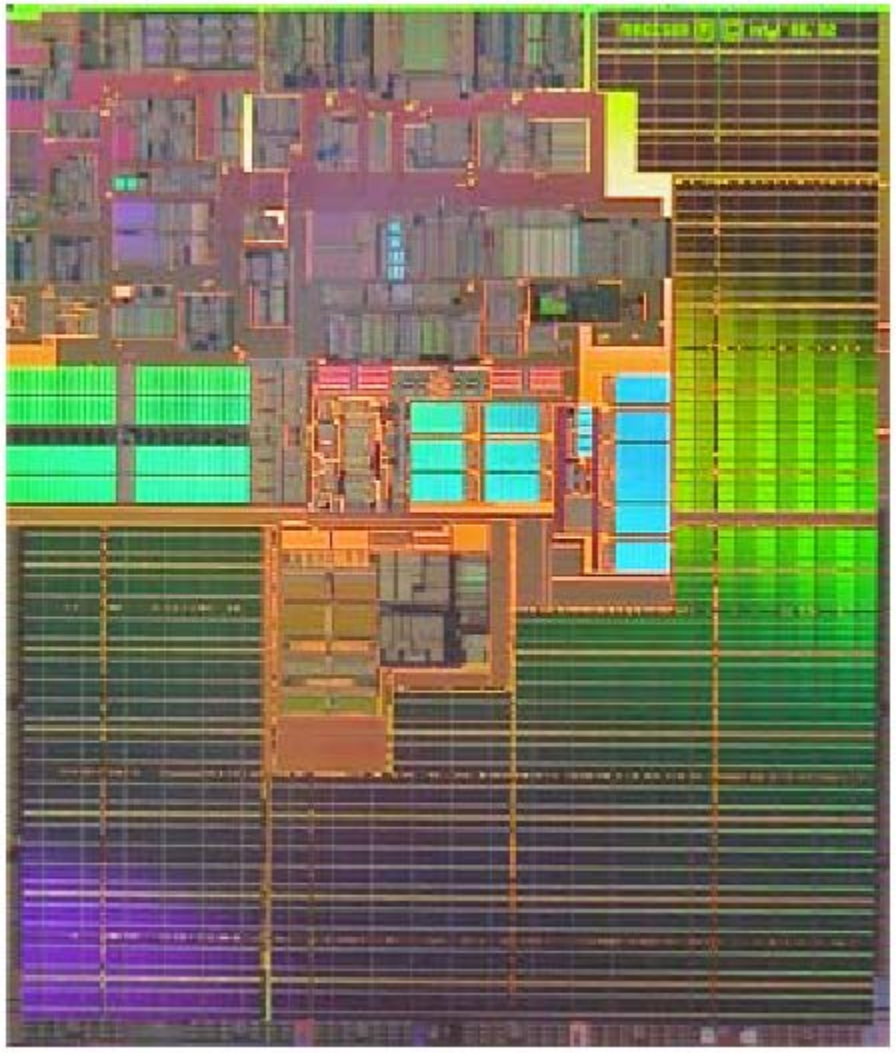
374 mm² die size

6 Mbyte on-die L3 cache

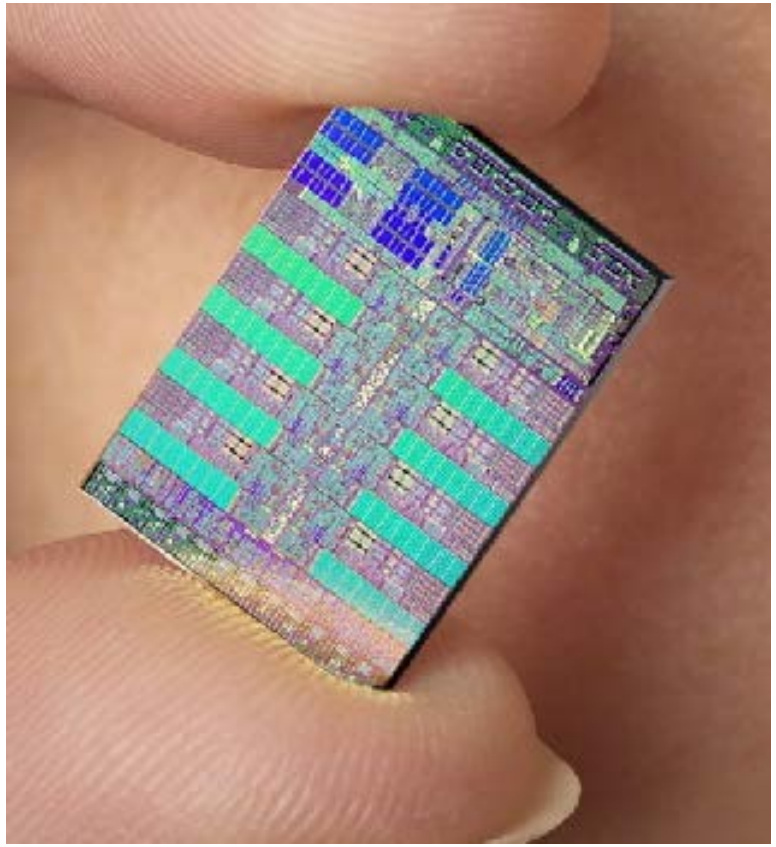
1.5 GHz @ 1.3 V

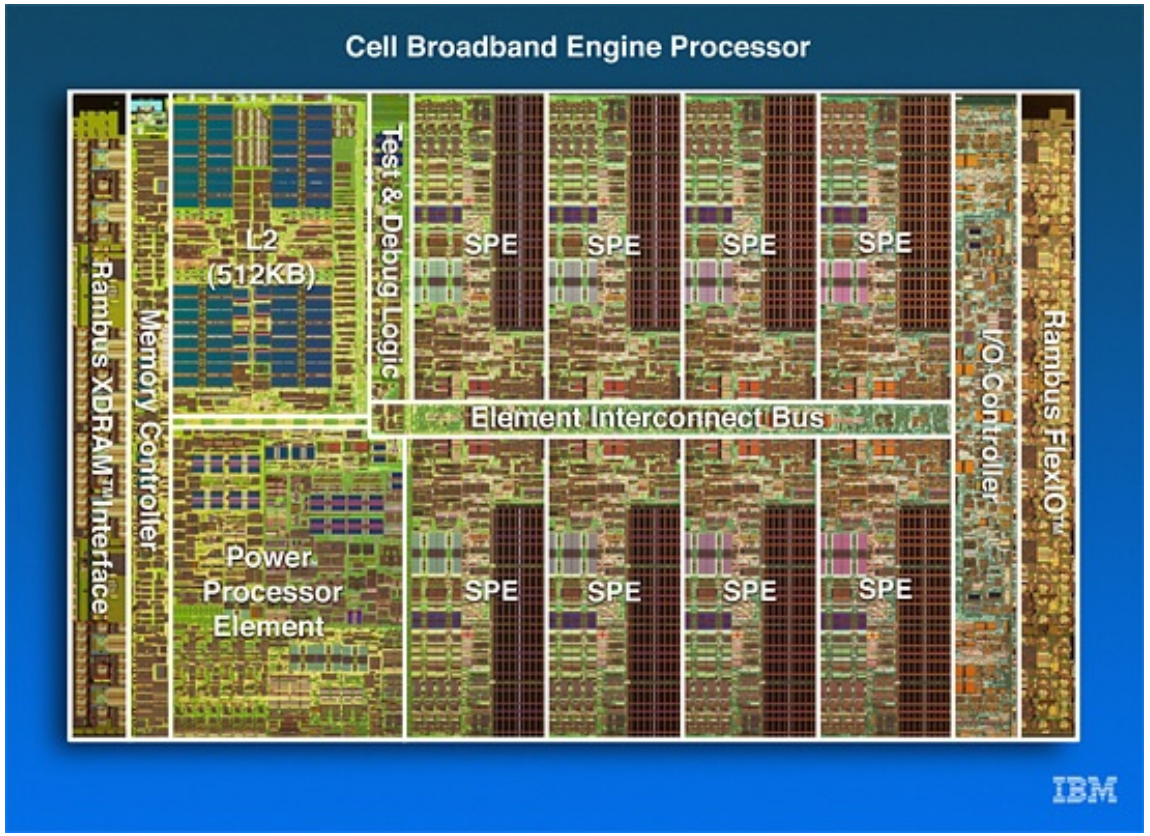
6.4 Gbyte/s bus interface



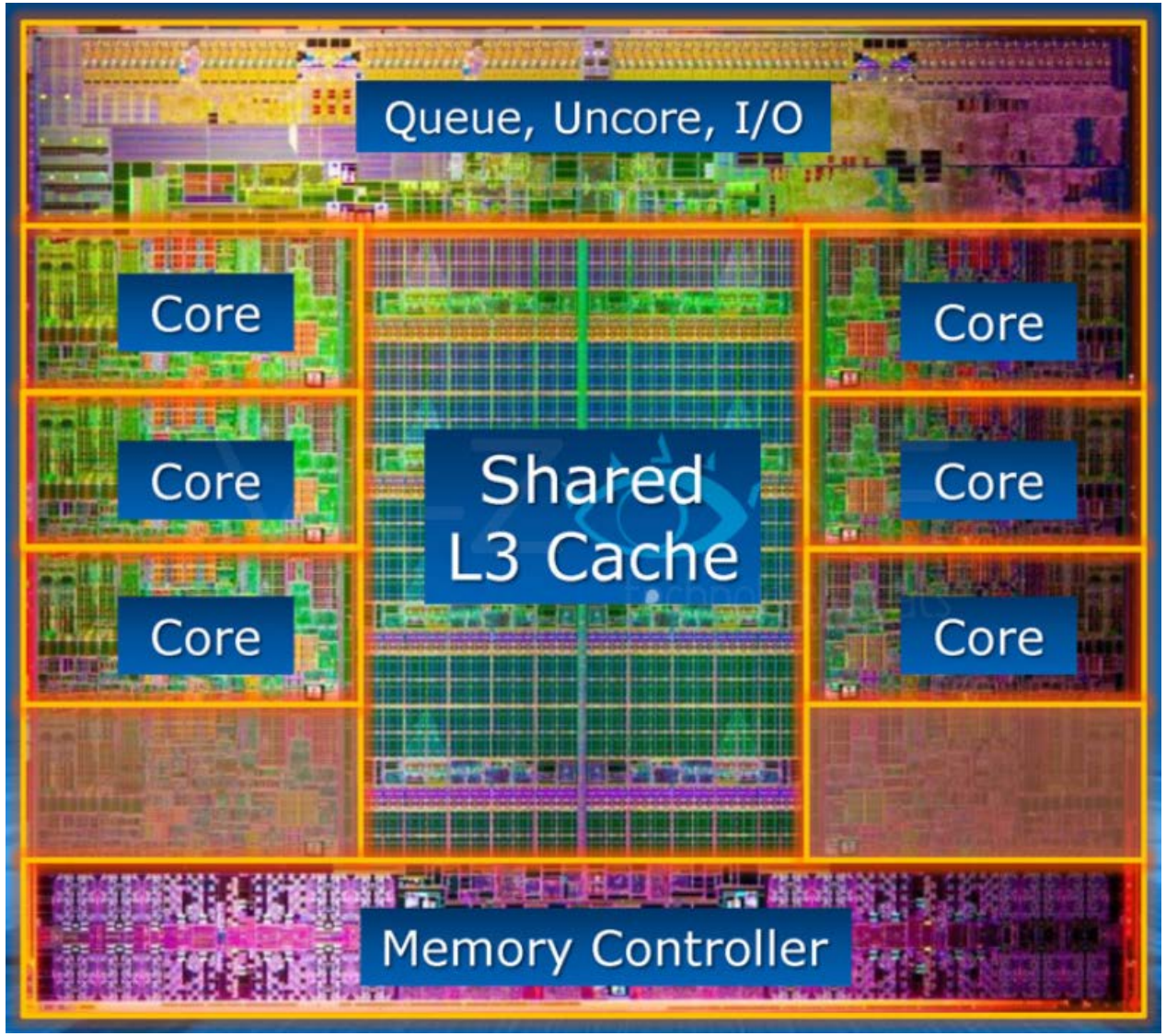


IBM Cell Processor





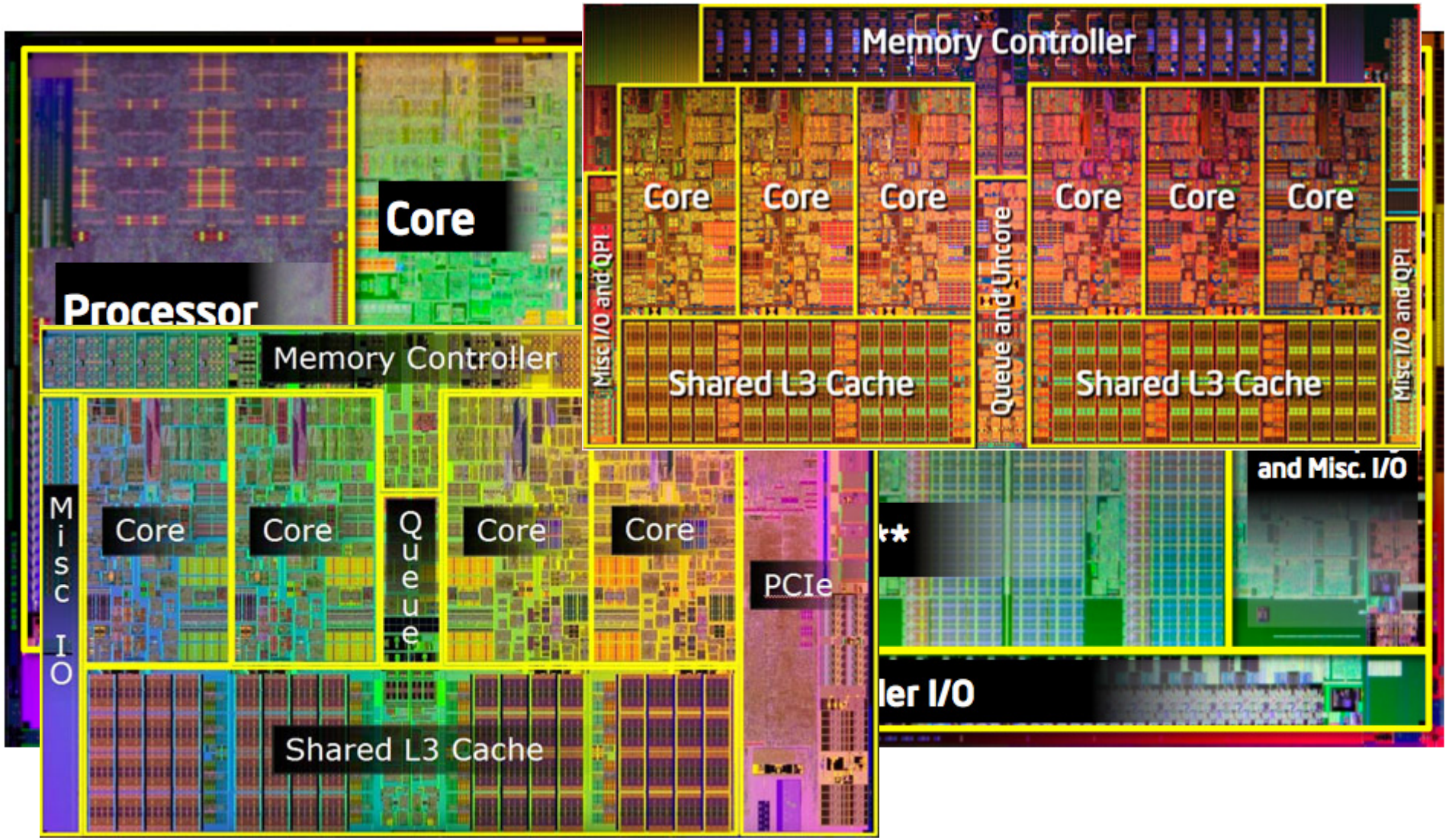
Example: Intel Core i7-3960X



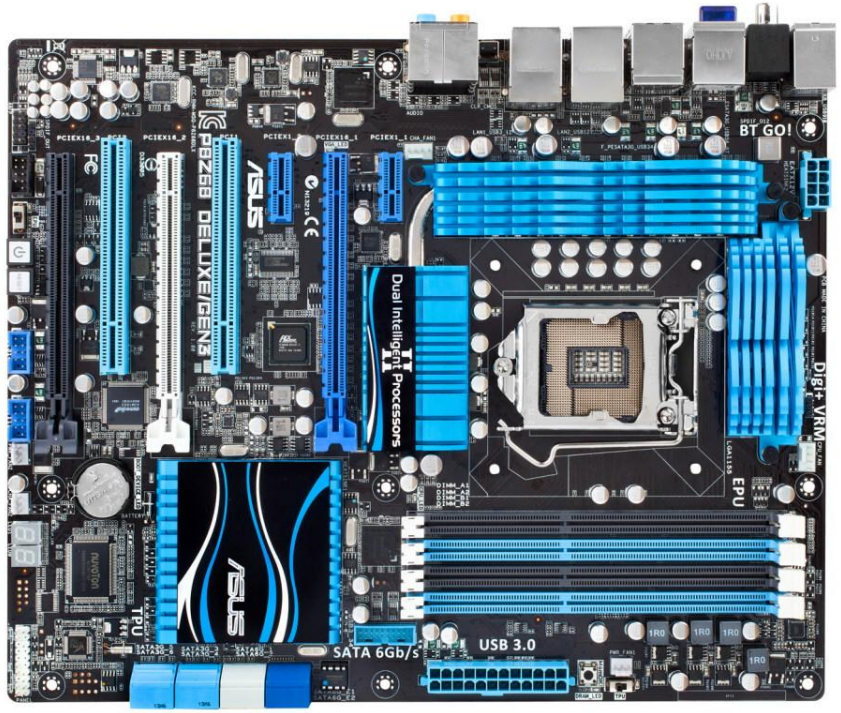
21 x 21 mm²

2.27 billion transistors

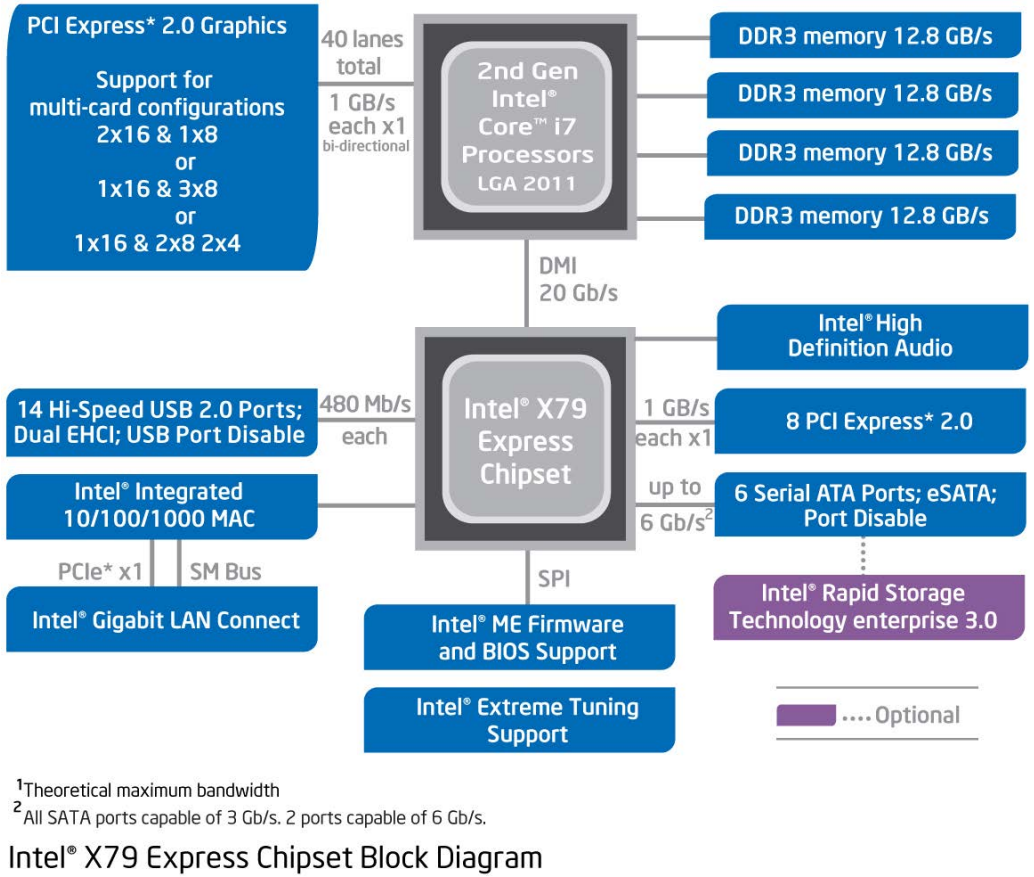
Several versions



Computers today



Ivy Bridge motherboard



[1]Theoretical maximum bandwidth
[2]All SATA ports capable of 3 Gb/s. 2 ports capable of 6 Gb/s.

Intel® X79 Express Chipset Block Diagram

Sandy Bridge example architecture

PC today – Example

Intel® Core™ i7-5960X (8 Core, 20MB Cache Overclocked up to 4Ghz)

32 inch UltraSharp™ LCD digital

32768 MB (4 x 8 GB), 2133 MHz DDR4-Quad-Channel-Memory

Solid-State-Disk (SSD) with 512 GB + SATA-II-HD 4 TB, 7,200 1/min and 32 MB Cache

2x Dual GPU 2x12GB NVIDIA Geforce GTX Titan Z

8x Blu-ray-RW-Drive and 24x DVD+/-RW-Drive

Sound Blaster® XFi Titanium PCIe

…

WHAT IS COMPUTER ARCHITECTURE?

What is Computer Architecture?

Different opinions exist

- Hardware structure, components, interfaces
- Basic operation principle, applications
- Only external view
- Internal and external view
- …

Computer architecture is NOT (only) standard PC architecture!

- The vast majority of computers are embedded systems, specialized solutions
- One size does NOT fit it all

➡ Should be: Computer Architecture**s**

Amdahl, Blaauw and Brooks 1967

“Computer architecture is defined as the attributes and behavior of a computer as seen by a machine-language programmer. This definition includes the instruction set, instruction formats, operation codes, addressing modes, and all registers and memory locations that may be directly manipulated by a machine language programmer.

Implementation is defined as the actual hardware structure, logic design, and data path organization of a particular embodiment of the architecture.“

Another view: processor architecture

The processor architecture (Instruction Set Architecture) comprises the description of attributes and functions of a system as seen from a machine language programmer's point of view.

The specification of the processor architecture comprises:

- instruction set
- instruction formats
- addressing modes
- interrupt handling
- logical address space
- Register/memory model (as far as a programmer can access it)

The processor architecture does not describe details of the implementation or hardware – all internal operations and components are explicitly excluded.

From the architects of the DEC Alpha microprocessor 1992

„Thus, the architecture is a document that describes the behavior of all possible implementations; an implementation is typically a single computer chip.

The architecture and software written to the architecture are intended to last several decades, while individual implementations will have much shorter lifetimes.

The architecture must therefore carefully describe the behavior that a machine-language programmer sees, but must not describe the means by which a particular implementation achieves that behavior.“

Processor micro architecture

An **implementation** (micro architecture) describes the hardware structure, all data paths, the internal logic etc. of a certain realization of the processor architecture, thus a real microprocessor.

The **micro architecture** defines:

- Number and stages of pipelines
- Usage of super scalar technology
- Number of internal functional units (ALUs)
- Organization of cache memory

The definition of a **processor architecture** (ISA, instruction set architecture) enables the use of programs independent of a certain internal implementation of a microprocessor.

All microprocessors following the same processor architecture specification are called “**binary compatible**” (i.e., the same binaries run on them).

CLASSIFICATION OF COMPUTERS

Classification of Computers: Levels of and techniques for parallelism

How to classify those many different systems, different architectures?

- By hardware? Changes too fast…
- By software? Can run on many systems…
- By size? Price? ???
- Here: by parallelism supported

A parallel program can be seen as a partially ordered set of instructions. The order is given by the dependencies among the instructions. Independent instructions can be executed in parallel.

Levels of parallelism: within a program/a set of programs

Techniques for parallelism: implemented in hardware

Classification of Computers: 5 levels of parallelism

Program level

Process level (or task level)

- Tasks (heavy weighted processes, coarse-grained tasks)

Block level

- Threads, Light-Weighted Processes

Instruction level

- Basic instructions (of the chosen language, of the instruction set)

Sub-operation level

- Basic instructions of a language/an instruction set can be divided into sub-operations/micro operations by a compiler/the processor (e.g., Pentium 4 uses micro instructions internally)

Level of parallelism and granularity

The **granularity** depends on the relation of computation effort over communication or synchronization effort

Typically, program, process, and thread level are considered as **large grained** parallelism

Thus, instruction and sub-operation level may support
fine grained parallelism

Sometimes, the block or thread level is considered as medium grained

Techniques of parallelism vs. levels of parallelism

Techniques	Program level	Process level	Block level	Instruction level	Sub-operation level
Loosely coupled computers					
Hyper- and Meta-computer	X	X			
Workstation-Cluster	X	X			
Coupled multi processor systems					
Message coupled	X	X			
Memory coupled (SMP)	X	X	X		
Memory coupled (DSM)	X	X	X		
Large grained data flow principle		X	X		
Processor internal structures					
Instruction pipelining				X	
Super scalar				X	
VLIW				X	
Overlapping I/O and CPU				X	
Fine grained data flow principle				X	
SIMD and microarchitectures					
Vector and field computers					X
Micro architectures					X

PERFORMANCE ASSESSMENT OF COMPUTER SYSTEMS

Performance assessment of computer systems

Needed for:

- Selection of a new computer system
- Tuning of an existing system
- Design of new computer systems

Methods for performance assessment

- Evaluation of hardware parameters
- Run-time measurements of existing programs
- Monitoring during operation of real computer systems
- Model theoretic approaches

Methods for performance assessment

Performance parameters
- Hypothetical maximum performance
- MIPS or MOPS (Millions of Instructions/Operations per Second)
- MFLOPS (Millions of Floating Point Operations per Second, today Tera FLOPS, TFLOPS ~10^{12})
- MLIPS (Millions of Logical Inferences per Second)

Mixes
- Theoretical mix of operations

Benchmark programs
- Number crunching, office applications, …

Monitors / measurement during operation
- Hardware monitor
- Software monitor

Model theoretic approaches
- Analytical methods
- Software simulation

Mixes

Calculation of an assumed average run-time based on the durations of n operations according to the formula:

$$T = \sum_{i=1}^{n} p_i t_i$$

T	average duration
n	number of distinct operations
t_i	duration of operation i
p_i	relative weight of operation i (i.e., relative number of appearances)

The following must hold:

$$\sum_{i=1}^{n} p_i = 1 \quad \text{and} \quad p_i \geq 0$$

Benchmark programs

Sieve of Erathostenes, Ackermann function

Whetstone (1970, typ. Fortran, lots of floating point arithmetic)

Dhrystone (begin of 80s, 53% assignments, 32% control statements, 15% function/procedure calls)

Savage-Benchmark (mathematical standard functions)

Basic Linear Algebra Subprograms (BLAS), core of the LINPACK/LAPACK (Linear Algebra Package) software package

Lawrence Livermore Loops (for vectorizing compilers)

SPEC-Benchmark Suite

SPEC benchmarks

SPEC Standard Performance Evaluation Corporation

- since 1989, many vendors, general purpose applications, focuses on calculation speed and throughput (www.spec.org)

Many benchmark suites exist:

- SPEC CPU95, CPU2000, CPU2006 (Integer, Floating point)
- SPEC HPC2002, MPI2007 (High Performance Computing)
- SPEC JVM98 (Java Client/Server)
- SPEC Web99, Web2005 (Web Servers)
- GPC (graphic performance characterization) group

“[…] SPEC is once again seeking to encourage those outside of SPEC to assist us in locating applications that could be used in the next CPU-intensive benchmark suite, currently planned to be SPEC CPU2004.” (http://www.spec.org/cpu2004/ , March 2004)

CINT2006: Integer Component of SPEC CPU2006

Benchmark	PL	Application Area	Brief Description
400.perlbench	C	Programming Language	The workload includes SpamAssassin, MHonArc (an email indexer), and specdiff.
401.bzip2	C	Compression	Modified to do most work in memory.
403.gcc	C	C Compiler	Generates code for Opteron.
429.mcf	C	Combinatorial Optimization	Vehicle scheduling. Uses a network simplex algorithm to schedule public transport.
445.gobmk	C	AI: Go	Plays the game of Go.
456.hmmer	C	Search Gene Sequence	Protein sequence analysis using profile hidden Markov models (profile HMMs)
458.sjeng	C	AI: Chess	A highly-ranked chess program.
462.libquantum	C	Physics	Simulates a quantum computer.
464.h264ref	C	Video Compression	H.264/AVC, encodes a video stream.
471.omnetpp	C++	Discrete Event Simulation	Uses the OMNet++ discrete event simulator to model a large Ethernet campus network.
473.astar	C++	Path-finding Algorithms	Pathfinding library for 2D maps, including the well known A* algorithm.
483.xalancbmk	C++	XML Processing	A modified version of Xalan-C++, which transforms XML docs. to other document types.

CFP2006: Floating Point Component of SPEC CPU2006

Benchmark	PL	Application Area	Brief Description
410.bwaves	Fortran	Fluid Dynamics	Computes 3D transonic transient laminar viscous flow.
416.gamess	Fortran	Quantum Chemistry.	Gamess implements a wide range of quantum chemical computations.
433.milc	C	Physics	A gauge field generating program for lattice gauge theory.
434.zeusmp	Fortran	Physics / CFD	ZEUS-MP is for the simulation of astrophysical phenomena.
435.gromacs	C, Fortran	Biochemistry / Molecular Dynamics	Molecular dynamics, i.e. simulate Newtonian equations of motion for hundreds to millions of particles.
436.cactusADM	C,Fortr.	Physics	Solves the Einstein evolution equations using a numerical method
437.leslie3d	Fortran	Fluid Dynamics	Computational Fluid Dynamics (CFD)
444.namd	C++	Biology / Molecular Dynamics	Simulates large biomolecular systems.
447.dealII	C++	Finite Elem. Analysis	A library targeted at adaptive finite elements and error estimation.
450.soplex	C++	Linear Programming, Optimization	Solves a linear program using a simplex algorithm and sparse linear algebra. Test cases include railroad planning and military airlift models.
453.povray	C++	Image Ray-tracing	Image rendering. The testcase is a 1280x1024 anti-aliased image of a landscape with some abstract objects with textures.
454.calculix	C, Fortran	Structural Mechanics	Finite element code for linear and nonlinear 3D structural applications. Uses the SPOOLES solver library.
459.GemsFDTD	Fortran	Computational Electromagnetics	Solves the Maxwell equations in 3D using the finite-difference time-domain (FDTD) method.
465.tonto	Fortran	Quantum Chemistry	An open source quantum chemistry package, using an object-oriented design in Fortran 95
470.lbm	C	Fluid Dynamics	Implements the "Lattice-Boltzmann Method" to simulate fluids in 3D
481.wrf	C,Fortr.	Weather	Weather modeling from scales of meters to thousands of kilometers.
482.sphinx3	C	Speech recognition	A widely-known speech recognition system from Carnegie Mellon University

www.top500.org

Nov. 2014

RANK	SITE
1	National Super Computer Center in Guangzhou China
2	DOE/SC/Oak Ridge National Laboratory United States
3	DOE/NNSA/LLNL United States
4	RIKEN Advanced Institute for Computational Science (AICS) Japan
5	DOE/SC/Argonne National Laboratory United States
6	Swiss National Supercomputing Centre (CSCS) Switzerland
7	Texas Advanced Computing Center/Univ. of Texas United States
8	Forschungszentrum Juelich (FZJ) Germany
9	DOE/NNSA/LLNL United States
10	Government United States

Juni 2015

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
7	King Abdullah University of Science and Technology Saudi Arabia	**Shaheen II** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	196,608	5,537.0	7,235.2	2,834
8	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462,462	5,168.1	8,520.1	4,510
9	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458,752	5,008.9	5,872.0	2,301
10	DOE/NNSA/LLNL United States	**Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393,216	4,293.3	5,033.2	1,972

Model theoretic approaches

Analytical methods

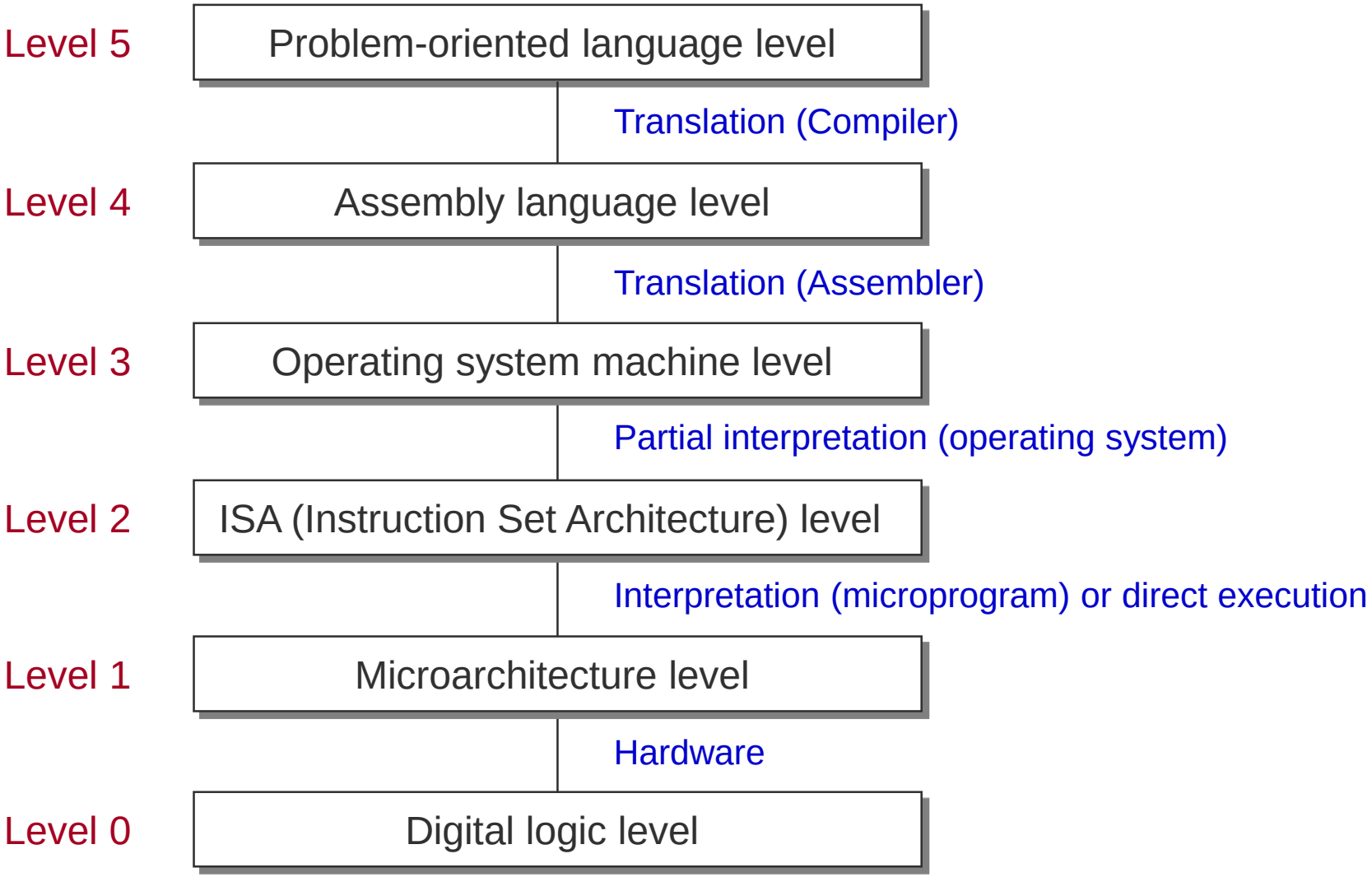
- deterministic queuing models (parameter: fixed values)
- stochastic queuing models (parameter: average values)
- operational queuing models (parameter: measured values)

Software simulations

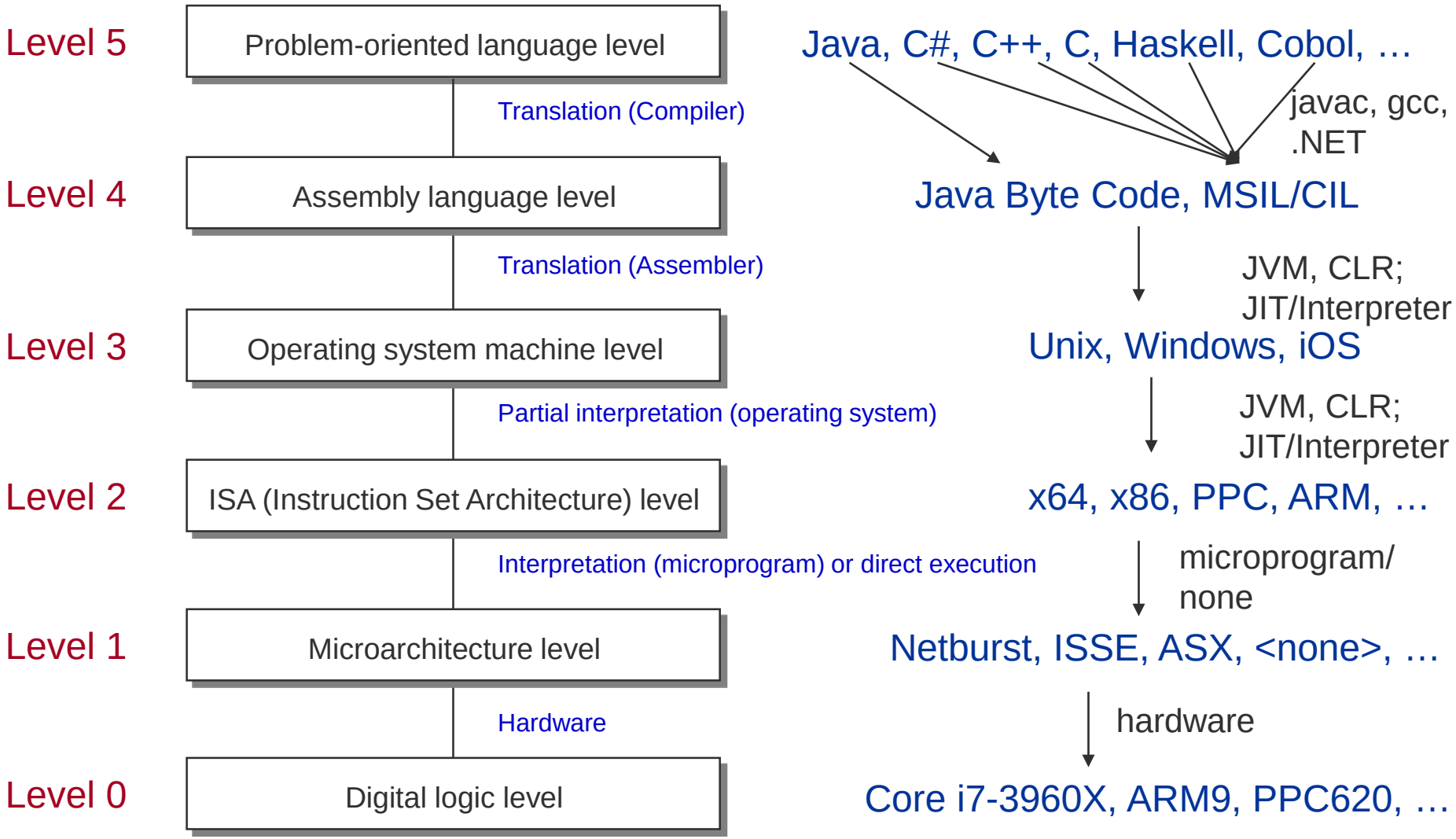
- deterministic simulation
- stochastic simulation
- simulation based on traces

THE LAYERED COMPUTER MODEL

A Six-level Computer



A Six-level Computer – Examples



Summary

Classical computer architecture
- von Neumann
- Harvard

Universal & special purpose computers

µComputer

Milestones of computer development

A definition of computer architecture

Classification of computers

Performance assessment

Layered computer model