3rd Slide Set
Operating Systems

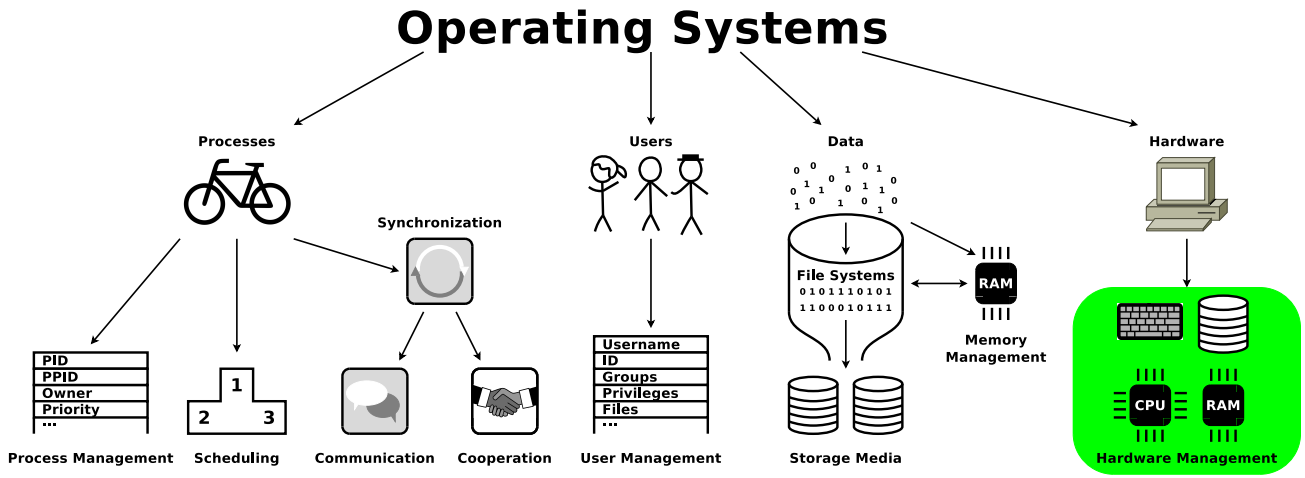
Prof. Dr. Christian Baun

Frankfurt University of Applied Sciences
(1971–2014: Fachhochschule Frankfurt am Main)
Faculty of Computer Science and Engineering
christianbaun@fb2.fra-uas.de

Learning Objectives of this Slide Set

- At the end of this slide set You know/understand...
  - the **computer architecture – Von Neumann architecture**
  - the **hardware components** of a computers
    - **CPU** and **Bus systems**
  - the difference of **character devices** and **block devices**
    - how **reading data from input/output devices** works
  - what **computer data storage** exists
    - the architecture of the **memory hierarchy**
    - what **primary/secondary/tertiary storage** is
    - the functioning of the **memory hierarchy**
    - how the different **write policies** (write-through and write-back) work

Exercise sheet 3 repeats the contents of this slide set which are relevant for these learning objectives



Why do we Discuss these Topics?

- Why do we discuss the functioning of the CPU, memory, storage and the bus systems in the operating systems course?

Edsger W. Dijkstra

„Computer Science is no more about computers than astronomy is about telescopes.“

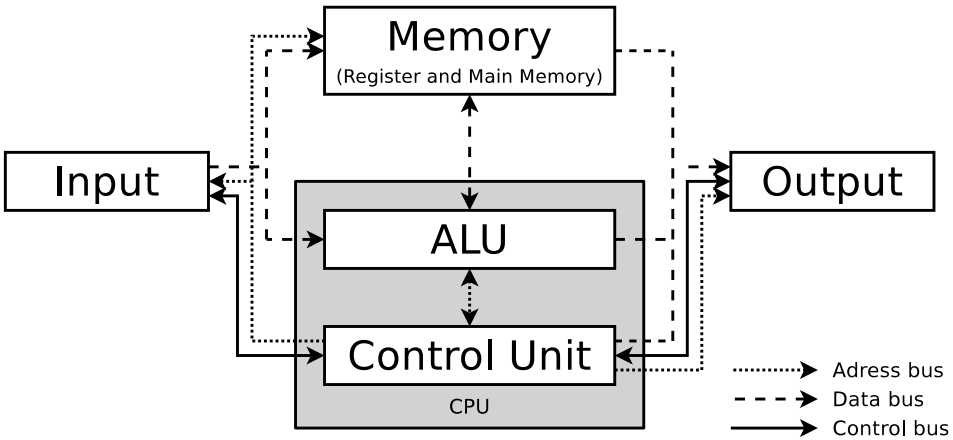
- Operating systems assist users and their processes in using the hardware
- Without an understanding of the functioning of the CPU, memory, storage and bus systems, it is impossible to understand the functioning of operating systems

Von Neumann Architecture

Image Source: US Department of Energy (Public Domain)

- Idea and structure of the general-purpose computer, which is not limited to a fixed program and has input and output devices
  - 1946: Developed by John von Neumann
  - Named after him is the **Von Neumann architecture**, or **Von Neumann computer**
- In the Von Neumann computer...
  - data and programs are **binary coded**
  - data and programs are stored in the **same memory**
- Essential concepts of the Von Neumann architecture were developed in 1936 by Konrad Zuse and implemented in 1937 in the Zuse Z1
- Von Neumann's merits:
  - He was a pioneer in the development and construction of computers in a scientific, mathematical way

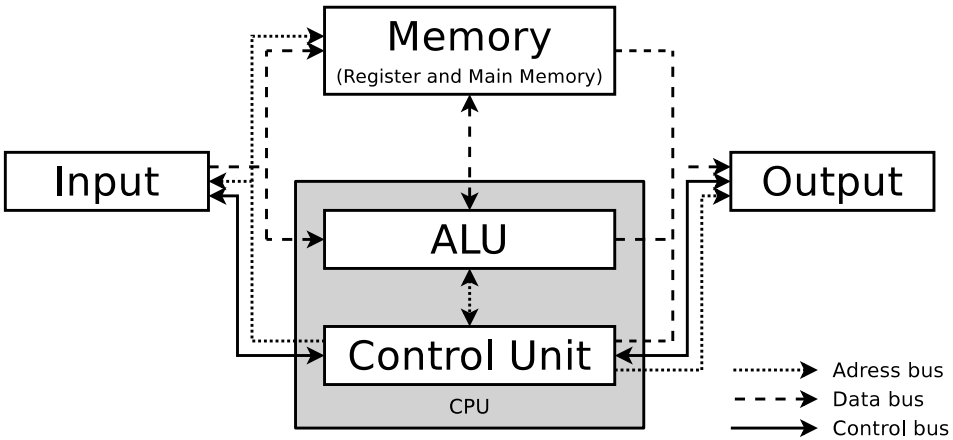
The Central Processing Unit (CPU)



- According to the Von Neumann Architecture, the memory is located outside the CPU
- In modern computer systems, parts of the memory (e.g. Registers and some Cache levels) is inside the CPU

- Most of the components of a computer are passive and controlled by the CPU
- Programs are sequences of machine instructions, which are stored in successive memory addresses
- During program execution, the CPU executes the machine instructions step by step
- A CPU consists of 2 components:
  - **Arithmetic Logic Unit** and **Control Unit**
- **Input/Output devices** ($\Longrightarrow$ slide 15) and **Memory** ($\Longrightarrow$ slide 21) are required too

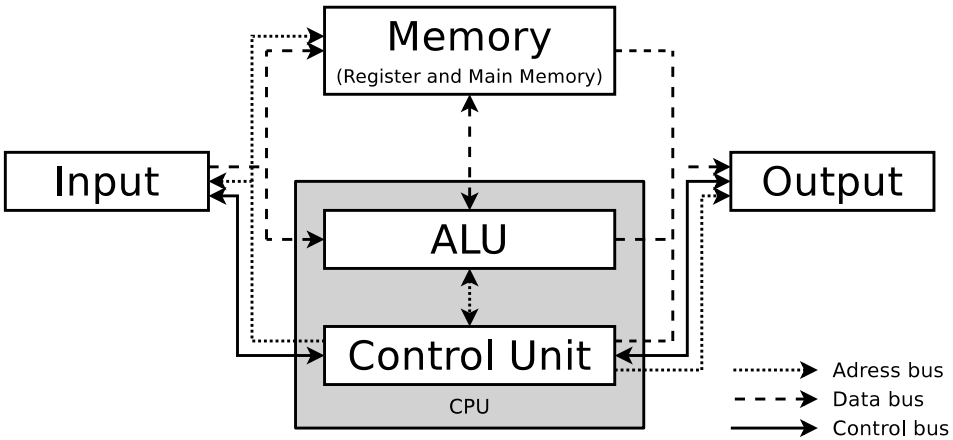
Components of the CPU



- According to the Von Neumann Architecture, the memory is located outside the CPU
- In modern computer systems, parts of the memory (e.g. Registers and some Cache levels) is inside the CPU

- **Control Unit**
  - Interprets instructions, coordinates the other CPU components, controls the input/output devices and the control bus
- **Arithmetic Logic Unit** (ALU)
  - Manipulates data and addresses
  - Carries out the logical (NOT, AND, OR, XOR,...) and mathematical (ADD, SUB,...) operations
- **Memory**
  - Registers for short-term storage of operands and addresses
    - Operate with the same speed as the rest of the CPU
  - Cache and main memory = memory for programs and data

Instruction Cycle (Fetch-Decode-Execute Cycle)

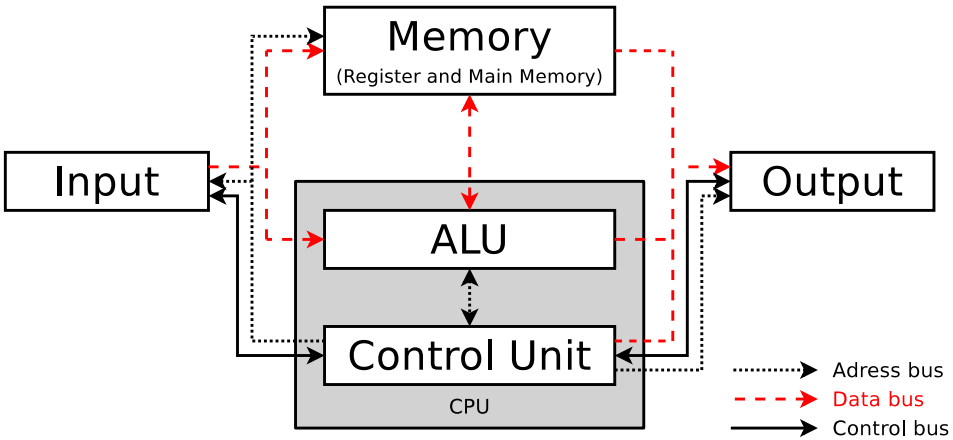


- Repeated by the CPU from bootup to the moment when the computer is shut down
  - Each phase may take several clock cycles to complete

1. **FETCH**: Fetch command to be processed from the memory into the Instruction Register
2. **DECODE**: Control Unit resolves the command into switching instructions for the ALU
3. **FETCH OPERANDS**: Fetch any available parameters (operands) for the command from the memory
4. **EXECUTE**: ALU carries out the command (switching instructions)
5. **UPDATE PROGRAM COUNTER**: The Program Counter register is set to the next command
6. **WRITE BACK**: Result generated is stored in a register, main memory, or sent to an output device

- The Fetch-Decode-Execute cycle and the Von Neumann architecture still implemented by modern CPUs
- Only difference: A single Bus to connect input and output devices directly with the CPU, is impossible today, and parts of the memory (Registers, L1/L2/L3-Cache) is located inside the CPU

Data Bus



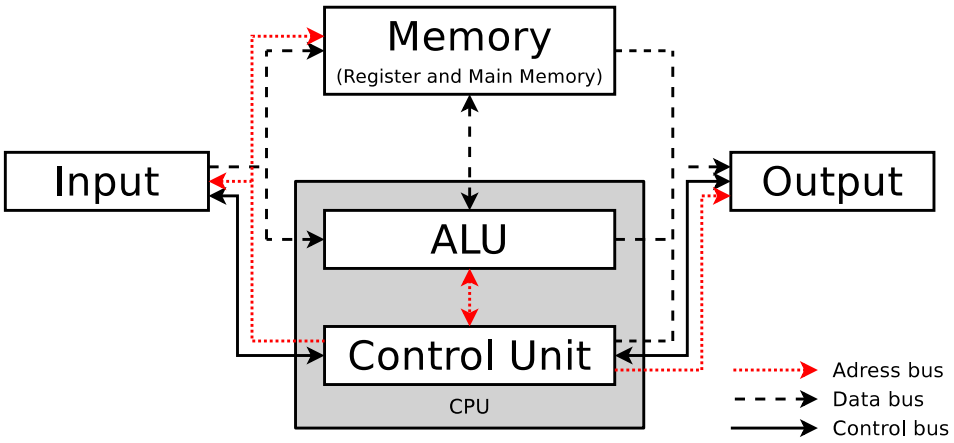
- Transmits data between CPU, main memory and I/O devices
- The number of lines specifies, how much data can be transmitted per clock cycle

- Usually, the number of lines is equal to the size of the registers of the Arithmetic Logic Unit
- Number of lines with modern CPUs: 64
  - Thus, the CPU can transfer 64 bits of data within a clock cycle to the main memory and away from it

Number of Data Bus lines of some CPUs

CPU	Data bus
4004, 4040	4 Bits
8008, 8080, 8085, 8088	8 Bits
8086 (XT), 80286 (AT), 80386SX	16 Bits
80386DX, 80486SX/DX/DX2/DX4	32 Bits
Pentium I/MMX/II/III/IV/D/M, Celeron, Core Solo/Duo, Core 2 Duo, Core 2 Extreme, Pentium Pro, Pentium Dual-Core, Core 2 Quad, Core i7, Itanium, AMD Phenom-II, Itanium 2, AMD64	64 Bits

Address Bus

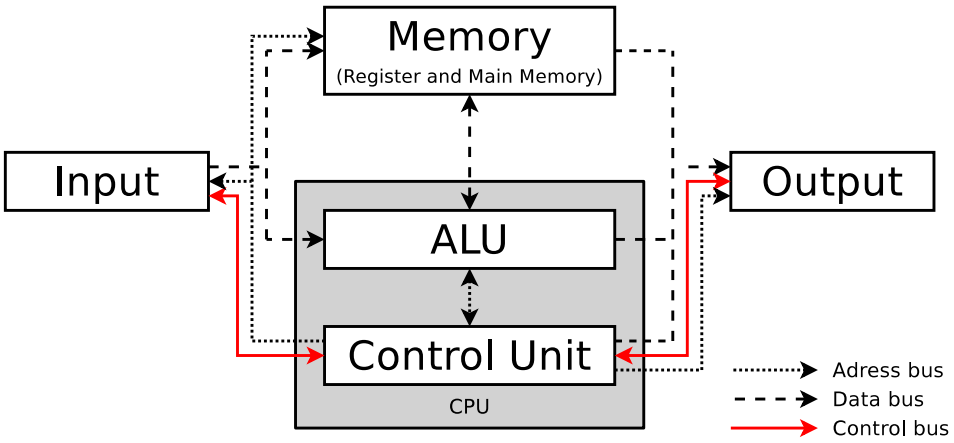


- Transmits memory addresses
- Memory addresses and I/O devices are accessed (addressed) via this bus
- The number of lines specifies the max. number of memory addresses

Number of Address Bus lines of some CPUs

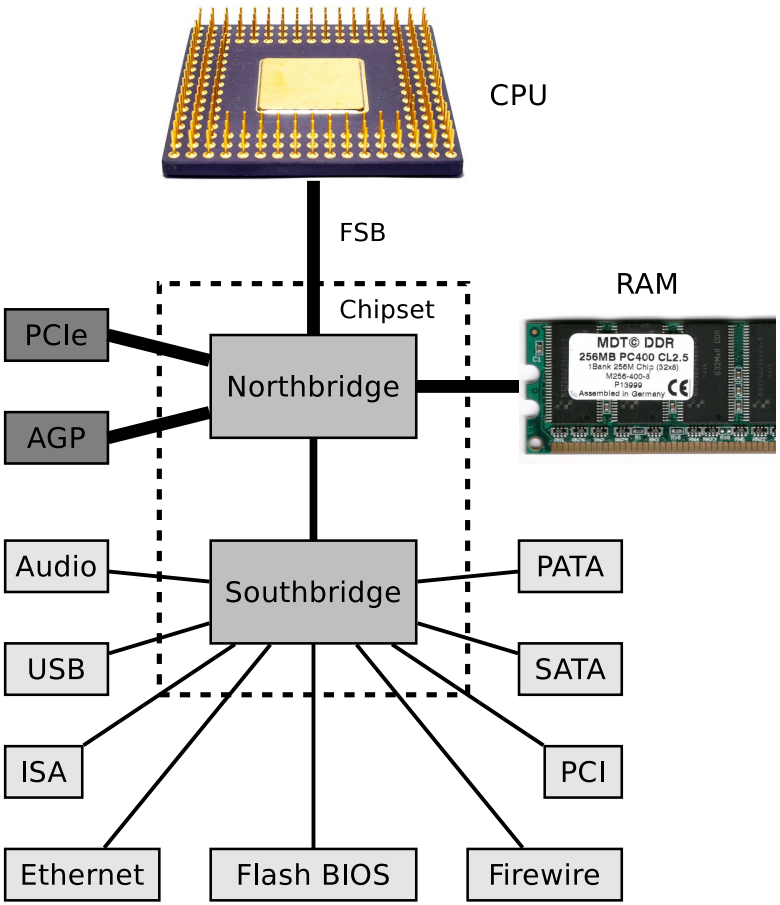
CPU	Address bus	max. addressable
4004, 4040	4 Bits	2^4 = 16 Bytes
8008, 8080	8 Bits	2^8 = 256 Bytes
8085	16 Bits	2^{16} = 65 kB
8088, 8086 (XT)	20 Bits	2^{20} = 1 MB
80286 (AT)	24 Bits	2^{24} = 16 MB
80386SX/DX, 80486SX/DX/DX2/DX4, Pentium I/MMX/II/III/IV/D/M, Celeron,	32 Bits	2^{32} = 4 GB
Core Solo/Duo, Core 2 Duo/Extreme/Quad, Pentium Pro, Pentium Dual-Core, Core i7	36 Bits	2^{36} = 64 GB
Itanium	44 Bits	2^{44} = 16 TB
AMD Phenom-II, Itanium 2, AMD64	48 Bits	2^{48} = 256 TB

Control Bus



- Transmits commands (e.g. read and write requests) from the CPU and returns status signals from the I/O devices
- Difference between address bus and control bus:
  - Components of the computer are addressed via the address bus and instructed via the control bus what to do
- Also contains lines which are used by I/O devices transmit interrupt requests to the CPU
- Typical number of lines: ≤ 10

Bus Systems in modern Computer Systems



- The **chipset** connects the CPU with the rest of the computer system
- The chipset consists of...
  - **Northbridge**
    - Located close to the CPU for rapid data transfer
    - Used for the connection of main memory and graphics card(s) with the CPU
  - **Southbridge**
    - Used for „slow“ connections
- The bus between CPU and chipset is called **Front Side Bus** (FSB)
  - It contains the address bus, data bus and control bus

Some Bus Systems

- **For performance and financial reasons, more and more parts of the chipset are relocated into the CPU**
  - In contrast to the Von Neumann architecture, I/O devices are not directly connected to the CPU
  - Computer systems today contain various serial and parallel bus systems, which are designed for the particular requirements
  - Point-to-point connections are used more and more often
  - Controllers for I/O devices operate between the devices and the CPU
- Some bus systems:

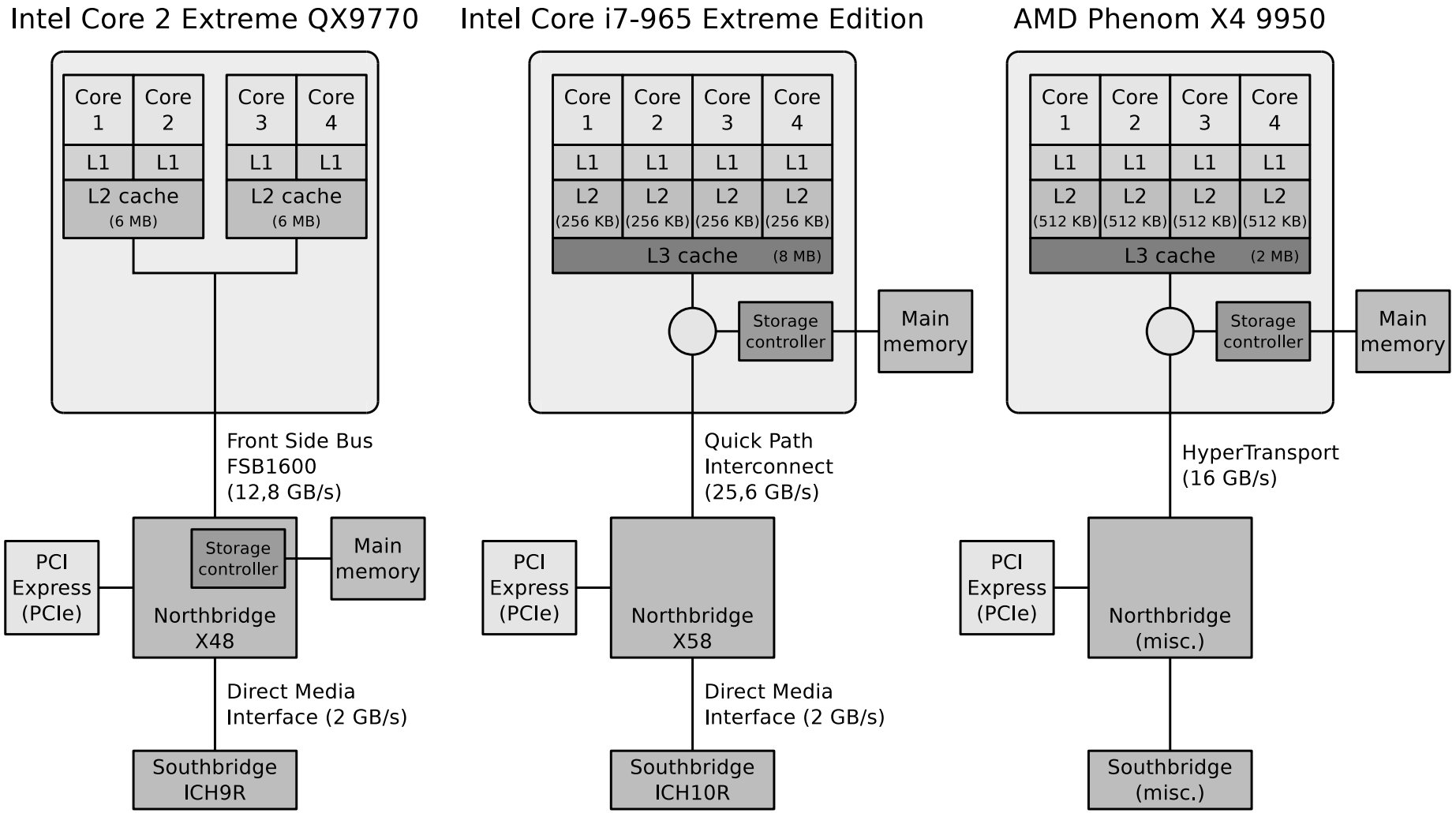
	Internal computer busses	External computer busses
Parallel busses	PATA (IDE), PCI, ISA, SCSI	PCMCIA, SCSI
Serial busses	SATA, PCI-Express	Ethernet, FireWire, USB, eSATA

More and more functions of the chipset are relocated to the CPU

The following two slides show how around 2008 the memory controller was shifted from the northbridge to the CPU and around 2009 also the PCIe interface

Relocation of the Memory Controller into the CPU

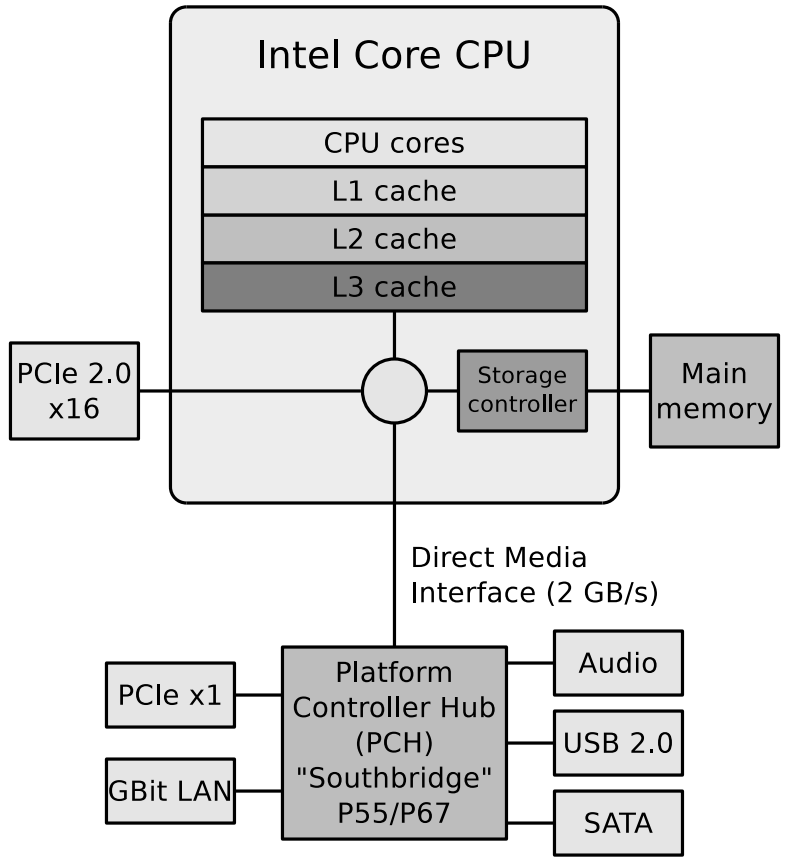
Source: c't 25/2008



Result: The Northbridge contains only the PCIe controller

Relocation of the Northbridge into the CPU

- In some modern computer systems, the Northbridge is relocated into the CPU
- Benefit: Reduced cost for the overall system



- The figure shows the placement of the functionalities in the chipset generations Intel P55 and P67 from 2009 and 2011
- Since this time, the Southbridge is also called **Platform Controller Hub** (PCH)

Character Devices and Block Devices

We already know...

- Input/Output devices are an essential component of the Von Neumann Architecture, but some relevant questions are still not answered
- What groups of Input/Output devices do exist?
- How can processes interact with Input/Output devices?

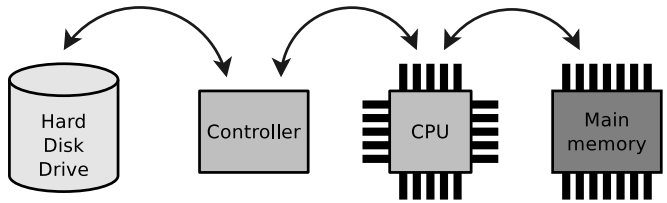
- Devices for computer systems are distinguished via their minimum transfer unit:
- **Character devices**
  - On arrival/request of each single character, communication with the CPU always takes place
  - Examples: Mouse, keyboard, printer, terminal and magnetic tape
- **Block devices**
  - Data transfer takes place only when an entire block (z.B. 1-4 kB) is present
  - Examples: HDD, SSD, CD/DVD drive and floppy drive

Reading Data

- Example: If a record of a HDD must be read, these steps are carried out:
  1. The CPU receives from a process the request to read a record from a HDD
  2. The CPU sends via the driver an I/O command to the controller
  3. The controller locates the record on the HDD
  4. The process receives the requested record
- 3 concepts exist of how processes can read data into a computer:
  - **Busy Waiting**
  - **Interrupt-driven**
  - **Direct Memory Access** (DMA)

Busy Waiting

- The driver sends the request to the device and waits in an **infinite loop** until the controller indicates that the data is available
  - If the data is available, it is written into the memory and the execution of the process continues
- Example: **Programmed Input/Output** (PIO)
  - The CPU accesses via read and write commands the memory areas of the devices and can copy this way data between the devices and the main memory
- **Benefit**:
  - No additional hardware required
- **Drawback**:
  - Causes CPU workload
  - Slows down simultaneous execution of multiple processes
    - Reason: The CPU must check periodically whether the data is available



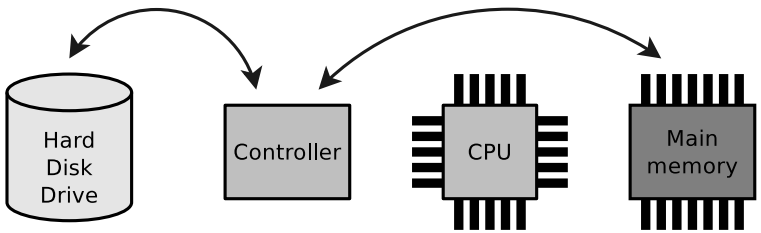
Examples: PATA HDDs in PIO mode, legacy serial ports, legacy parallel ports, PS/2 keyboard and mouse ports

Interrupt-driven

- Precondition: An **interrupt controller** and a **line** of the control bus exist for the transmission of the **interrupts**
- The driver initializes the I/O operation and waits for an interrupt from the controller $\Longrightarrow$ the driver *sleeps*
  - The CPU is not blocked while waiting for the interrupt and the operating system can assign the CPU to other processes
  - If an interrupt occurs, the driver is *woken* $\Longrightarrow$ gets the CPU assigned
    - Next, the CPU fetches the data from the controller and stores it inside the memory
    - Then, the interrupted process gets the CPU assigned and its execution continues
- **Benefits**:
  - The CPU is not blocked
  - Allows the simultaneous execution of multiple processes
- **Drawbacks**:
  - Additional hardware (interrupt controller) is required

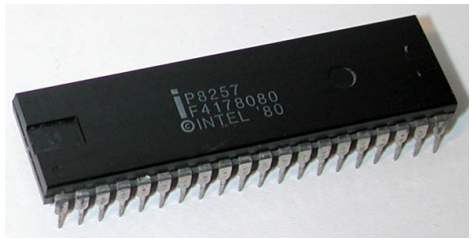
Direct Memory Access

- Precondition: **DMA controller**
  - Can transfer data directly between main memory and the I/O device
    - Examples: HDD/SSD, sound card, network adapter, TV/DVB tuner card
  - Triggers an interrupt after the data is transfered



- Example: **Ultra-DMA** (UDMA)
  - Successor protocol of the PIO mode
  - Specifies how data is transferred between the DMA controller and the main memory

- **Benefits**:
  - Reading data causes no CPU workload
  - Simultaneous execution of multiple processes is not slowed down
- **Drawbacks**:
  - Additional hardware (DMA controller) is required
    - Integrated in the chipset since the late 1980s



Bildquelle:
`http://www.cpu-world.com/Support/82/Intel-P8257.jpg`
(Usage for non-commercial and educational purposes allowed)

Computer Data Storage

We already know...

- Computer data storage is an essential component of the Von Neumann Architecture, but some relevant questions are still not answered
- What computer data storage technologies exist?
- What computer data storage components are attached to computer system?
- How are the different computer data storage organized and used by the operating system?

- Stores the data and the executables
- Different computer storage is connected via different bus systems
  $\Longrightarrow$ **memory hierarchy** (see slide 26)
- Reason for existence the memory hierarchy: price-performance ratio
  $\Longrightarrow$ The better the performance of a computer data storage is, the higher is the acquisition cost and the smaller is the capacity

Digital Data Storage

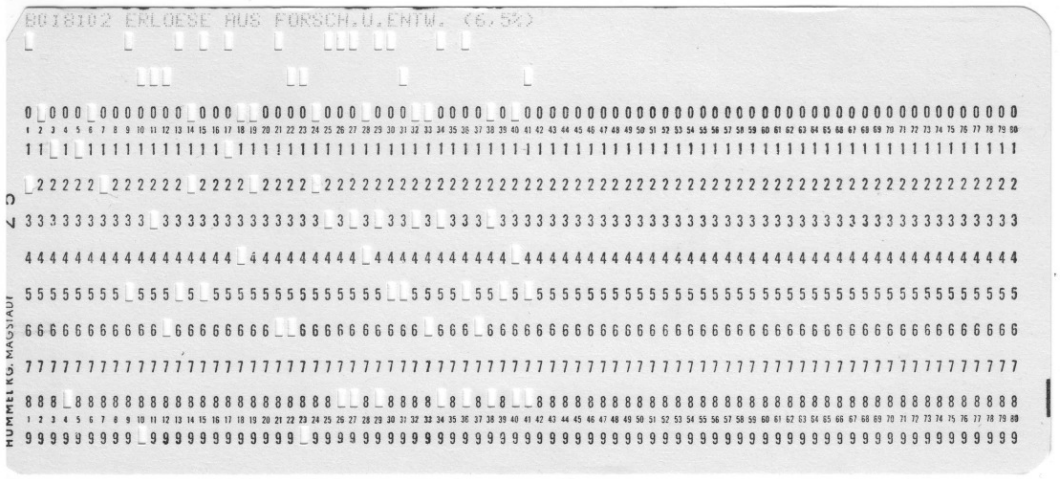
Storage	Write operation	Read operation	Access method	Movable parts	Persistent
Punched tape	mechanic		sequential	yes	yes
Punch card	mechanic		sequential	yes	yes
Magnetic tape	magnetic		sequential	yes	yes
Magnetic stripe card	magnetic		sequential	yes	yes
Drum memory	magnetic		random	yes	yes
Magnetic-core memory	magnetic		random	no	yes
Bubble memory	magnetic		random	no	yes
Cache and Registers (SRAM)	electric		random	no	no
Main memory (DRAM)	electric		random	no	no
Flash memory (USB drive, SSD, CF/SD card)	electric		random	no	yes
Compact cassette (Datasette)	magnetic		sequential	yes	yes
Floppy disk	magnetic		random	yes	yes
Hard disk drive	magnetic		random	yes	yes
CD-ROM/DVD-ROM	mechanic	optical	random	yes	yes
CD-R/CD-RW/DVD-R/DVD-RW	optical		random	yes	yes
MiniDisc	magneto-optical	optical	random	yes	yes
Magneto optical disc (MO-Disk)	magneto-optical	optical	random	yes	yes

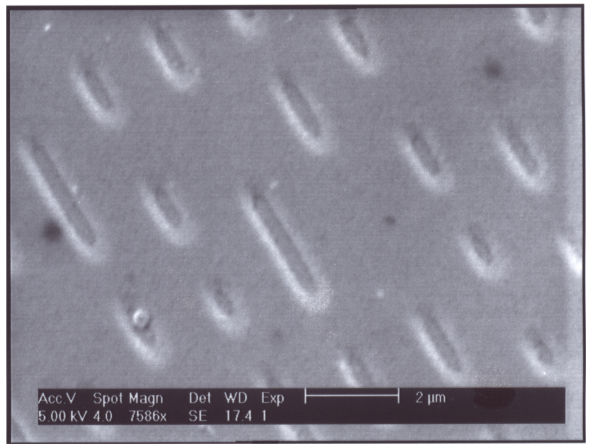
(gray background color means outdated/obsolete technology)

- **Random access** means that the medium does not need to be searched sequentially from the beginning – such as with magnetic tapes – to locate a specific record (file)
  - The heads of magnetic disks or a laser can jump to every point of the medium within a known maximum period

Mechanical Data Storage

Image source (punch card): own work



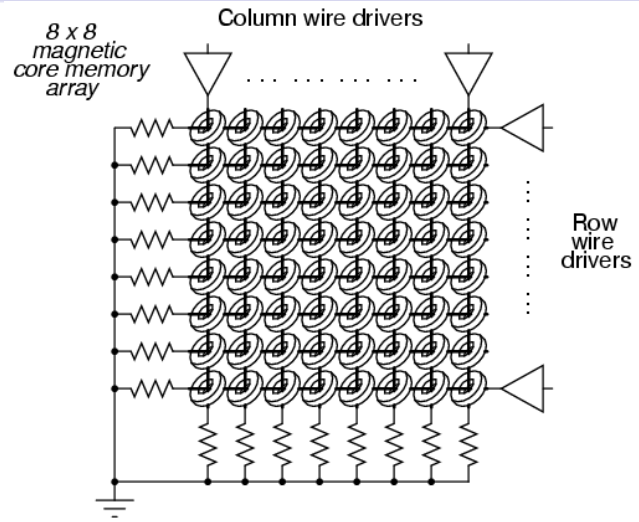


- Each punch card usually represents a single line of text with 80 characters or a corresponding number of binary data
- The punched tape in the image has 8 holes for data and narrower holes to feed the tape
  - 1 bytes per row can be stored
- Data is represented on CDs/DVDs by pits and lands, which are applied to a plastic material
  - The mass-production of CDs/DVDs is called *pressing* and is carried out by injection molding with a negative (*stamper*)

Image source (punched tape): TedColes. Wikimedia (CC0)
Image source (pressed CD with pits und lands): Stefan Kolb. Wikimedia (CC0)

Magnetic Data Storage

Image source: http://sub.allaboutcircuits.com/images/04212.png



- Data is stored on a magnetizable material
- Via read-and-write heads, the magnetization of the material is detected and modified
  - Exception: Magnetic-core memory
- Read-and-write heads may be movable (e.g. on HDDs) or fixed (e.g. on magnetic tapes)
- **Rotating data storage:**
  - Hard Disk Drive ($\Longrightarrow$ slide set 4), Floppy disk, Drum memory...
- **Non-rotating data storage:**
  - Magnetic-core memory, Magnetic tape, Magnetic stripe card, Datasette, Bubble Memory...

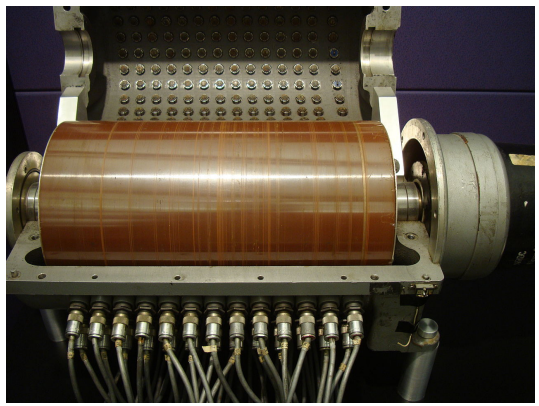
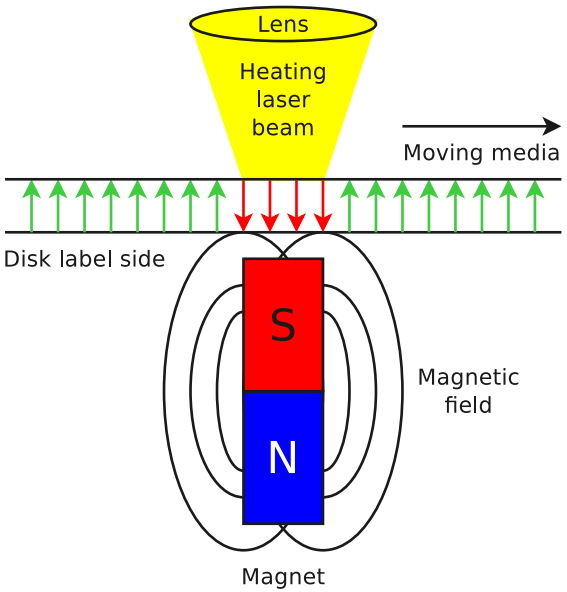


Image source (Drum memory): Gregg Tavares (CC-BY-2.0)
Image source (Floppy disks): George Chernilevsky (CC0)

Magneto-optical Data Storage

Image source (Medium): ocrho. Wikimedia (CC0)

- Rotating storage medium
- Write operations are carried out magnetically
- Before the magnetization can be modified, the medium must be heated until the *Curie point* is reached
  - Advantage: Insensitive to magnetic fields
  - The heating takes place via laser beam
- Read operations are carried out optically
  - Differently magnetized areas reflect light differently



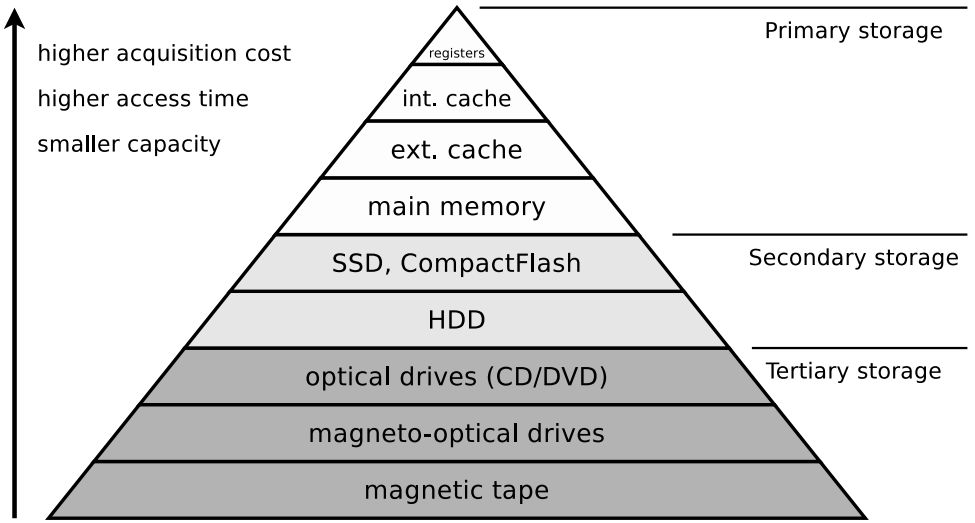
Magneto-optical data storage is obsolete today. It was popular between about 1990 and 2005, especially in Japan. The reason why I present it here is just to show how much effort engineers put into storing data on different types of storage media in the past decades

Electric Data Storage

- **Volatile memory** – Random-Access Memory (RAM)
  - Static Random-Access Memory (SRAM)
    - Information is stored as a change of state of *flip-flop* circuits
    - Information can be stored as long as the operating voltage is available
    - Faster and more expensive than DRAM
    - Used for **cache** and CPU-internal **registers**
  - Dynamic Random-Access Memory (DRAM)
    - Information is stored in capacitors
    - Requires periodic refreshing of the information
    - Stored data gets lost if the operating voltage is permanently missing or if the refresh was carried out too late because of leakage currents
    - Used for **main memory**
- **Non-volatile memory**
  - Read-Only Memory (ROM)
    - e.g. Electrically Erasable Programmable ROM (EEPROM)
  - Flash memory $\Longrightarrow$ slide set 4

Memory Hierarchy

- Primary storage and secondary storage are permanently connected to the computer
  - Advantage: Stored data can be accessed quickly

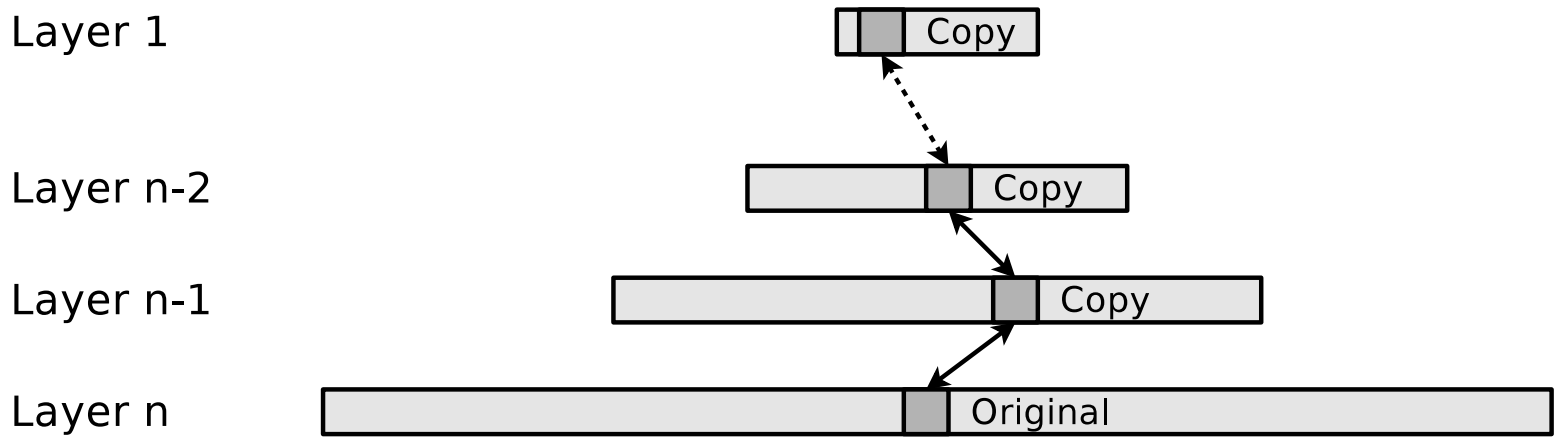


- **Primary storage**: The CPU has direct access to this storage
- **Secondary storage**: Storage, which is accessed via a controller
- **Tertiary storage**: Not permanently connected to the computer. Main purpose is archiving

- Tertiary storage can be:
  - **Near-line storage**: Is automatically and without human intervention connected to the system (e.g. tape library)
  - **Off-line storage**: Media are stored in cabinets or storage rooms and must be connected manually to the system
    - Removable HDDs are in a strict sense also off-line storage

Functioning of the Memory Hierarchy

- When a record is accessed for the first time, a copy is created and this copy travels along the memory hierarchy to the top layer



- If the record is modified, the modification must be passed down (written back) at some point in time
  - During write back, the copies of the record must be updated at all layers in order to avoid inconsistencies
  - Modifications cannot be passed directly to the lowest layer (to the original)!

Cache Write Policies

- **Write-through**: Modifications are immediately propagated to lower storage layers
  - **Advantage**: Consistency is ensured
  - **Drawback**: Lower performance

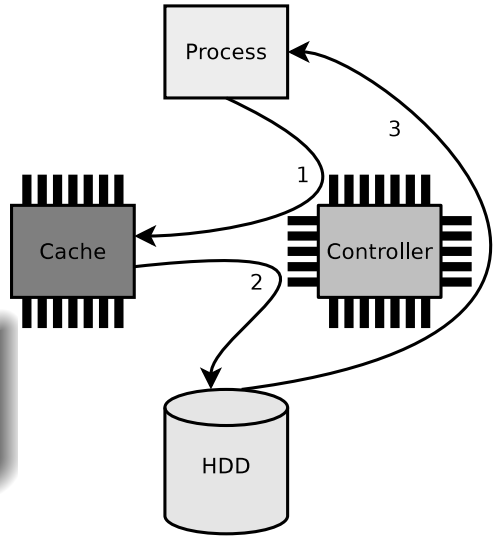


Figure: A process wants to carry out a write operation. It writes (1) the data into the cache and sends the write operation to the controller. The controller commands (2) the writing of the data into the storage. If the data was written successfully, the controller reports (3) the successful writing of the data to the process

- **Write-back**: Modifications are propagated when the corresponding page is removed from the cache
  - **Advantage**: Better performance
  - **Drawback**: Modifications get lost in case of a system failure

For every page in the cache a **dirty bit** is stored inside the cache, which indicates whether the page has been modified or not

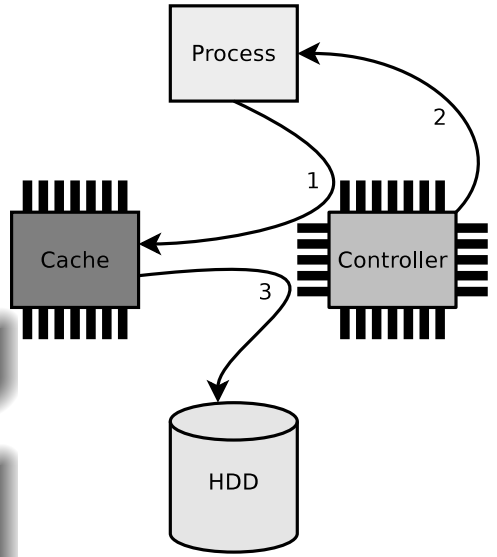


Figure: A process wants to carry out a write operation. It writes (1) the data into the cache and sends the write instruction to the controller. The controller reports (2) **immediately** the successful writing of the data to the process. The writing (3) of the data into the storage is carried out asynchronous to the write instruction in the process

Registers, Cache and Main Memory (1/4)

- Data inside **registers** can be accessed by the CPU immediately
- Registers operate with the same clock speed as the CPU itself

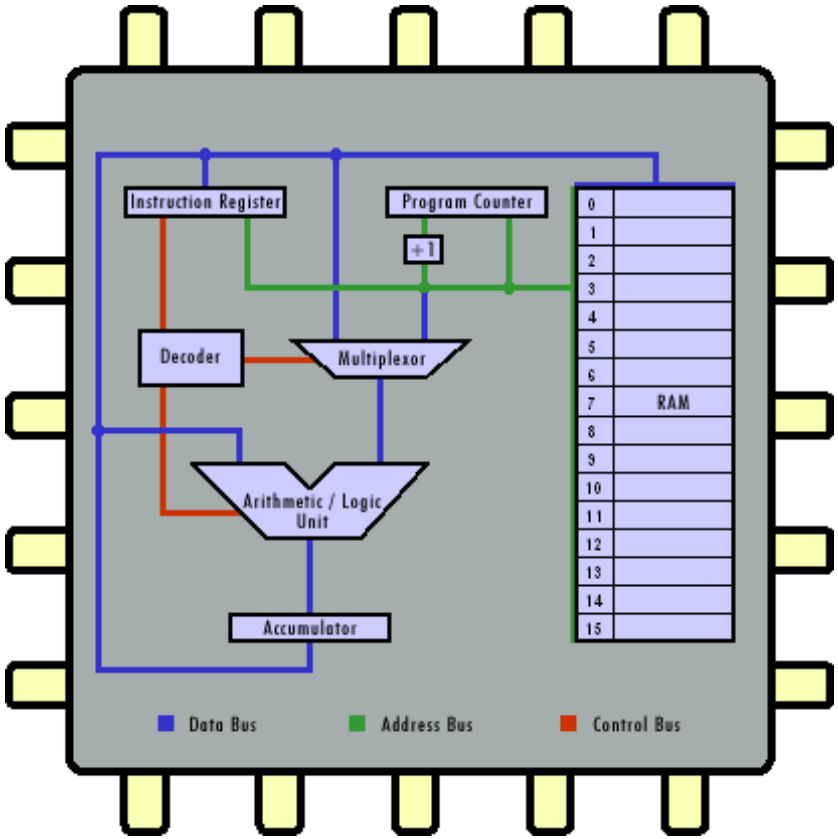
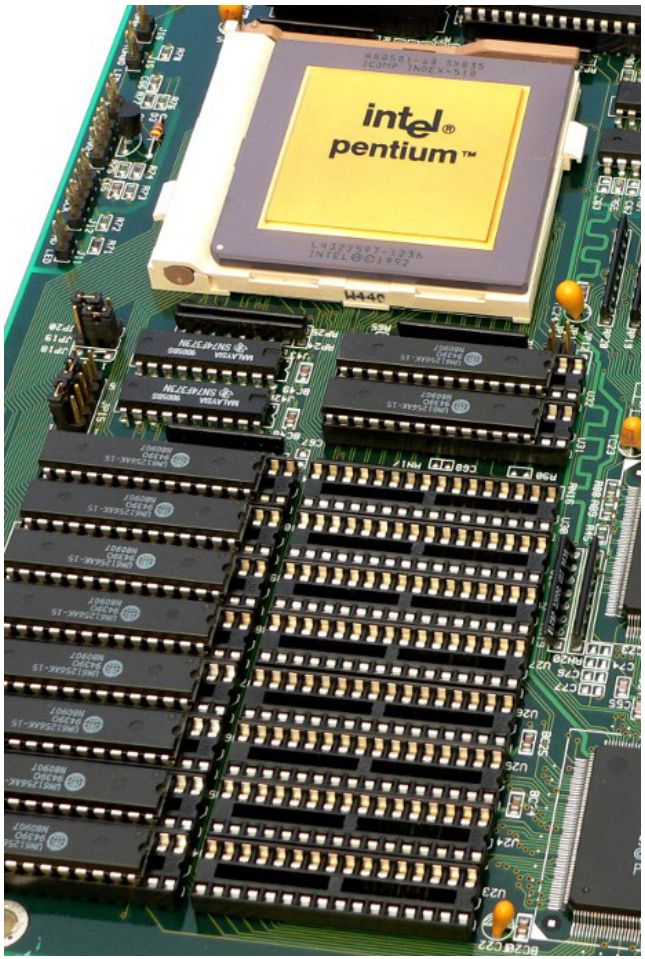


Image source: `http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/CPU/cpu_circuit.gif`

- **Data registers** (= *accumulators*) store operands for the ALU and their results,
  - e.g. EAX, ECX, EDX, EBX (32 bit)
    RAX, RBX, RCX, RDX (64 bit)
    $\Longrightarrow$ slide set 7
- **Address registers** for memory addresses of operands and instructions
  - e.g. **base register** (= *segment register*) and **index register** (for the offset)
    $\Longrightarrow$ slide set 5
- **Program counter** (= *instruction pointer*) contains the memory address of the next instruction
- **Instruction register** stores the instruction, which is currently executed
- **Stack pointer** stores the memory address at the current end of the stack $\Longrightarrow$ slide set 7

Registers, Cache and Main Memory (2/4)

- **Cache** (buffer memory) stores copies of parts of the main memory to accelerate access to these data



- **First Level Cache** (L1 cache)
  - Integrated into the CPU
- **Second Level Cache** (L2 cache)
  - Slower and bigger in size
  - originally external to the CPU

Image source:
Wikipedia (Konstantin Lanzet CC-BY-SA-3.0)
The image shows an Intel Mobile Pentium II „Tongae“ 233 MHz CPU with external 512 kB L2 cache. The L2 cache runs at half the clock frequency

Image source: `http://www.amoretro.de/2012/03/si5pi-aio-rev-1-1-socket-4-motherboard.html`
The image shows an Elitegroup SI5PI AIO with a Pentium 60.
The mainboard has 16 memory module sockets for L2 cache

Registers, Cache and Main Memory (3/4)

- Since 1999/2000 the CPU vendors increasingly integrating the L2 cache into the CPUs
  - For this reason, a **Third Level Cache** (L3 cache) as CPU-external cache was established
- In modern CPUs (e.g. Intel Core i-series and AMD Phenom II) the L3 cache is integrated into the CPU too
  - In multi-core CPUs with integrated L3 cache, the cores share the L3 cache, while each core has its own L1 cache and L2 cache

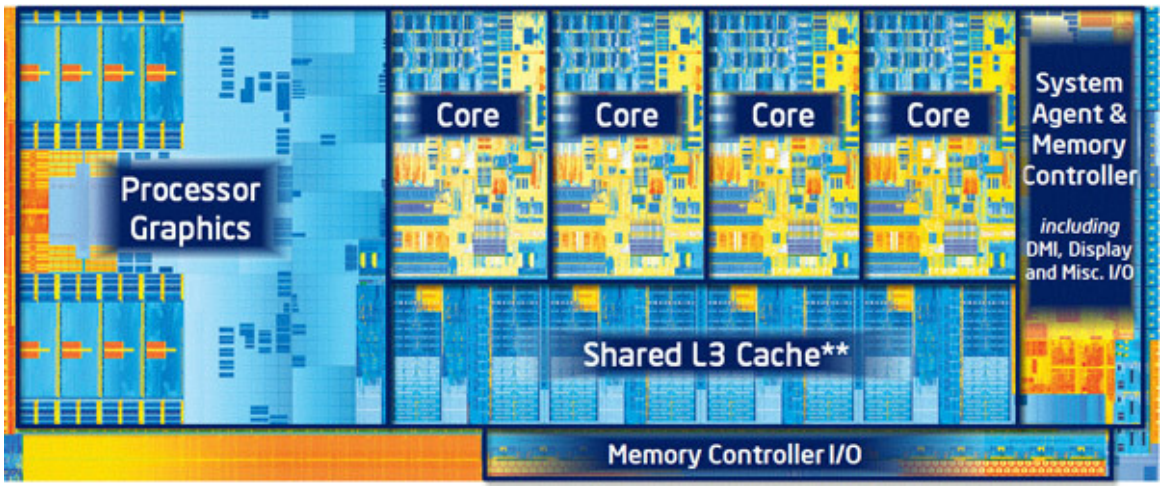


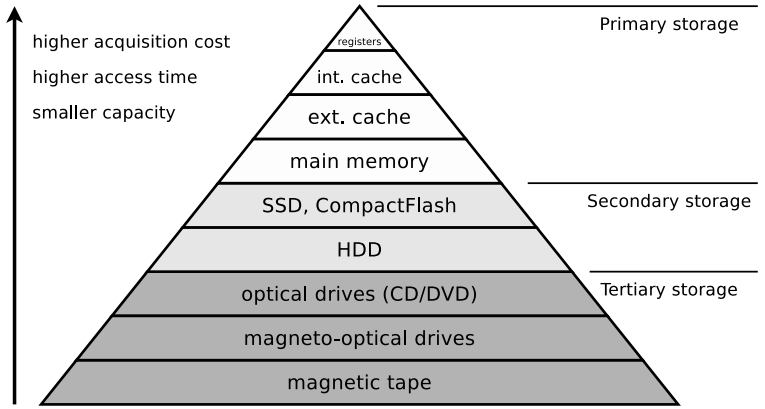
Image source: Intel
The image shows an Intel Core i7-3770K „Ivy Bridge“ CPU with 4 cores and integrated L3 cache

Some CPU architectures have a L4 cache

- Intel Itanium 2 (2003): 64 MB
- Some Intel Haswell CPUs (2013): 128 MB

Registers, Cache and Main Memory (4/4)

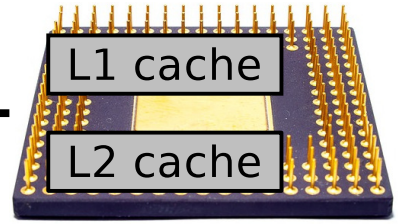
- Typical cache level capacities:
  - L1 cache: 4 kB to 256 kB
  - L2 cache: 256 kB to 4 MB
  - L3 cache: 1 MB to 16 MB



- **Main memory** = **Random Access Memory** (RAM)
  - Capacity: A few hundred MB up to several GB
  - All requests from the CPU, which can not be answered by the cache are forwarded to the main memory

Main memory (RAM)

L3 cache



CPU

RAM und ROM

In contrast to the volatile main memory **RAM** is **ROM** (*Read Only Memory*) a non-volatile read-only memory

Fundamentals of Computer Data Storage and Memory Management

Recap

- Until now we clarified regarding computer data storage:
  - It stores the data and the executables
  - A memory hierarchy exists
    - Reason: Price-performance ratio
    - The faster a storage is, the more expensive and smaller is it
  - If a record is accessed for the first time, a copy is created and this copy travels along the memory hierarchy to the top layer
  - Since the uppermost storage levels practically always lack free capacity, records must be replaced
  - Modifications must be passed down (written back)

- We need to discuss:
  - **Memory management** $\Longrightarrow$ slide set 5
  - **Addressing of computer data storage** $\Longrightarrow$ slide set 5