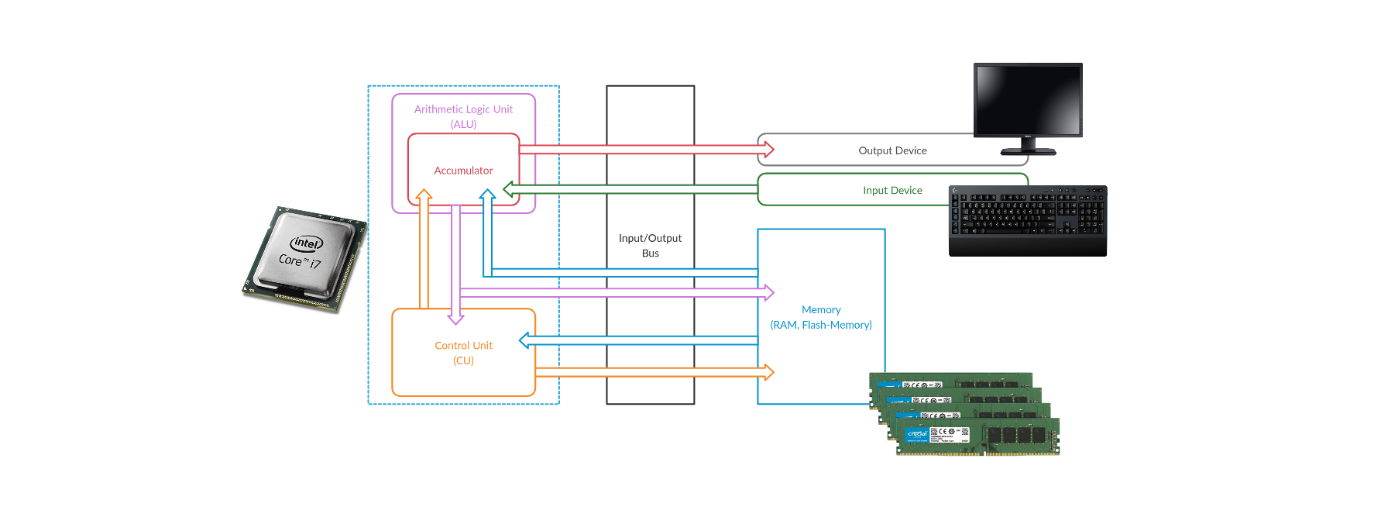
**CPU Architecture**

The architecture of the Von-Neumann was developed by the Hungarian mathematician John von Neumann, and it consists of four functional units:

* Memory
* Control Unit
* Arithmetical Logical Unit
* Input/Output Unit

In the Von-Neumann architecture, the most important units, the Arithmetical Logical Unit (ALU) and Control Unit (CU), are combined in the actual Central Processing Unit (CPU). The CPU is responsible for executing the instructions and for flow control. The instructions are executed one after the other, step by step. The commands and data are fetched from memory by the CU. The connection between processor, memory, and input/output unit is called a bus system, which is not mentioned in the original Von-Neumann architecture but plays an essential role in practice. In the Von-Neumann architecture, all instructions and data are transferred via the bus system.

**Von-Neumann Architecture**



**Memory**

The memory can be divided into two different categories:

* Primary Memory
* Secondary Memory

**Primary Memory**

The primary memory is the Cache and Random Access Memory (RAM). If we think about it logically, memory is nothing more than a place to store information. We can think of it as leaving something at one of our friends to pick it up again later. But for this, it is necessary to know the friend's address to pick up what we have left behind. It is the same as RAM. RAM describes a memory type whose memory allocations can be accessed directly and randomly by their memory addresses.

The cache is integrated into the processor and serves as a buffer, which in the best case, ensures that the processor is always fed with data and program code. Before the program code and data enter the processor for processing, the RAM serves as data storage. The size of the RAM determines the amount of data that can be stored for the processor. However, when the primary memory loses power, all stored contents are lost.

**Secondary Memory**

The secondary memory is the external data storage, such as HDD/SSD, Flash Drives and CD/DVD-ROMs of a computer, which is not directly accessed by the CPU, but via the I/O interfaces. In other words, it is a mass storage device. It is used to permanently store data that does not need to be processed at the moment. Compared to primary memory, it has a higher storage capacity, can store data permanently even without a power supply, and works much slower.

**Control Unit**

The Control Unit (CU) is responsible for the correct interworking of the processor's individual parts. An internal bus connection is used for the tasks of the CU. The tasks of the CU can be summarised as follows:

* Reading data from the RAM
* Saving data in RAM
* Provide, decode and execute an instruction
* Processing the inputs from peripheral devices
* Processing of outputs to peripheral devices
* Interrupt control
* Monitoring of the entire system

The CU contains the Instruction Register (IR), which contains all instructions that the processor decodes and executes accordingly. The instruction decoder translates the instructions and passes them to the execution unit, which then executes the instruction. The execution unit transfers the data to the ALU for calculation and receives the result back from there. The data used during execution is temporarily stored in registers.

**Central Processing Unit**

The Central Processing Unit (CPU) is the functional unit in a computer that provides the actual processing power. It is responsible for processing information and controlling the processing operations. To do this, the CPU fetches commands from memory one after the other and initiates data processing.

The processor is also often referred to as a Microprocessor when placed in a single electronic circuit, as in our PCs.

Each CPU has an architecture on which it was built. The best-known CPU architectures are:

* x86/i386 - (AMD & Intel)
* x86-64/amd64 - (Microsoft & Sun)
* ARM - (Acorn)

Each of these CPU architectures is built in a specific way, called Instruction Set Architecture (ISA), which the CPU uses to execute its processes. ISA, therefore, describes the behavior of a CPU concerning the instruction set used. The instruction sets are defined so that they are independent of a specific implementation. Above all, ISA gives us the possibility to understand the unified behavior of machine code in assembly language concerning registers, data types, etc.

There are four different types of ISA:

* CISC - Complex Instruction Set Computing
* RISC - Reduced Instruction Set Computing
* VLIW - Very Long Instruction Word
* EPIC - Explicitly Parallel Instruction Computing

**RISC**

RISC stands for Reduced Instruction Set Computer, a design of microprocessors architecture that aimed to simplify the complexity of the instruction set for assembly programming to one clock cycle. This leads to higher clock frequencies of the CPU but enables a faster execution because smaller instruction sets are used. By an instruction set, we mean the set of machine instructions that a given processor can execute. We can find RISC in most smartphones today, for example. Nevertheless, pretty much all CPUs have a portion of RISC in them. RISC architectures have a fixed length of instructions defined as 32-bit and 64-bit.

**CISC**

In contrast to RISC, the Complex Instruction Set Computer (CISC) is a processor architecture with an extensive and complex instruction set. Due to the historical development of computers and their memory, recurring sequences of instructions were combined into complicated instructions in second-generation computers. The addressing in CISC architectures does not require 32-bit or 64-bit in contrast to RISC but can be done with an 8-bit mode.

**Instruction Cycle**

The instruction set describes the totality of the machine instructions of a processor. The scope of the instruction set varies considerably depending on the processor type. Each CPU may have different instruction cycles and instruction sets, but they are all similar in structure, which we can summarize as follows:

| **Instruction** | **Description** |
| --- | --- |
| 1. FETCH | The next machine instruction address is read from the Instruction Address Register (IAR). It is then loaded from the Cache or RAM into the Instruction Register (IR). |
| 2. DECODE | The instruction decoder converts the instructions and starts the necessary circuits to execute the instruction. |
| 3. FETCH OPERANDS | If further data have to be loaded for execution, these are loaded from the cache or RAM into the working registers. |
| 4. EXECUTE | The instruction is executed. This can be, for example, operations in the ALU, a jump in the program, the writing back of results into the working registers, or the control of peripheral devices. Depending on the result of some instructions, the status register is set, which can be evaluated by subsequent instructions. |
| 5. UPDATE INSTRUCTION POINTER | If no jump instruction has been executed in the EXECUTE phase, the IAR is now increased by the length of the instruction so that it points to the next machine instruction. |

Top of Form

Bottom of Form

**Table of Contents**

**Introduction**

**Fundamentals**

**Exploit**

**Proof-Of-Concept**

**Skills Assessment**

**My Workstation**

OFFLINE