COMPUTER ARCHITECTURE

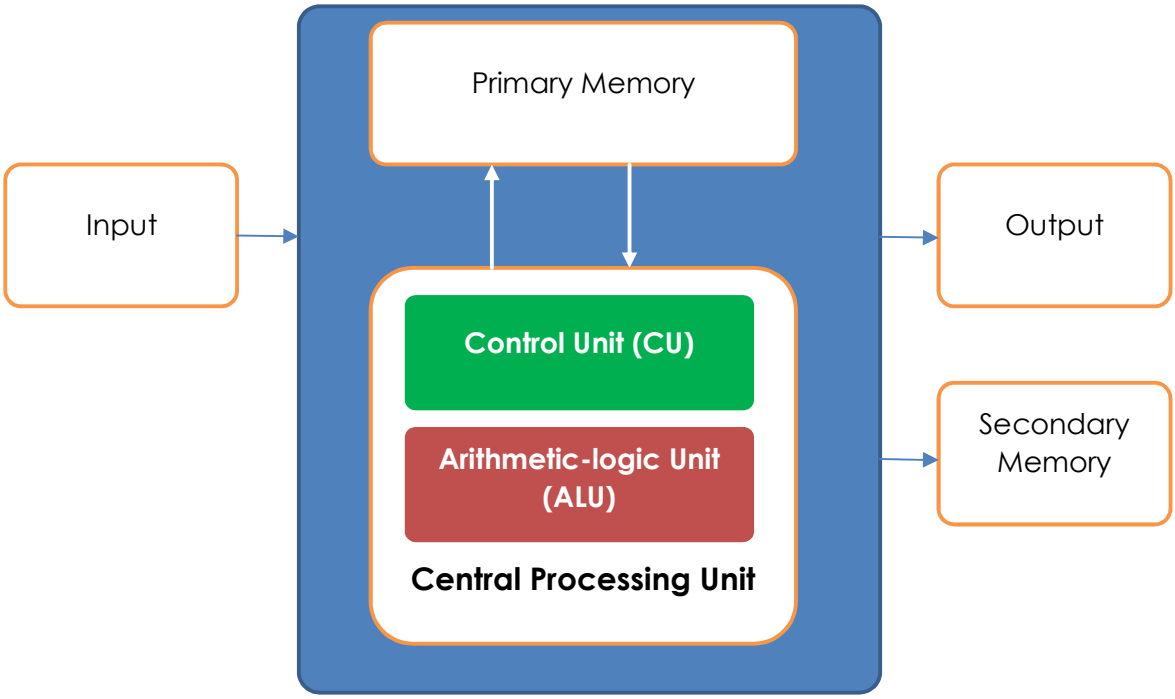
Stored-program Concept

In 1945 computer scientists realised that data and instructions, which manipulate the data, are technically the same thing – just sequences of zeros and ones. Therefore they can be stored in the same place – single computer memory.

This design is known as the ***von Neumann architecture***, although there are doubts, whether von Neumann had any contribution to that concept.

The von Neumann architecture comprises 5 major components:

1. **Input**
2. **Output**
3. **Control Unit + Arithmetic-logic Unit**
4. **Primary Memory**
5. **Secondary Memory**



Features of the von Neumann Architecture

- **Primary memory holds both – data and instructions** (unlike the Harvard architecture, which has separate memory units for them)
- **ALU performs arithmetic and logic operations**
- **Control unit manages actions** done by the remaining units

- Input and output devices communicate with the interacting user/another computer system
- **Secondary memory is an optional unit** – it overcomes the limitations of primary memory units like capacity and volatility.

Primary Memory

Memory is a collection of ***cells***, each with a unique physical address. The term **CELL** is preferred as the size of the addressable unit varies. Current computer systems use mostly 1 Byte as the least addressable cell.

The number of addressable units depends on the number of bits dedicated to address representation. E.g. 32 bits are capable of differentiating 2^{32} = 4 294 967 296 cells. The number of bits for addressing depends on the processor.

The cell addresses are numbered in the C style – from 0.

The content of the cells may represent both, data or instructions – that depends on the interpretation. This feature can be misused exploiting problems in the memory management.

Arithmetic-logic Unit

It is a combination of circuits designed to perform essential

- arithmetic operations - +, -, *, /, mod, bitwise operations
- logic operations - AND, OR, NOT

ALU processes ***words*** – groups of data the processor can store in ***registers*** and use in instructions. E.g. a 64-bit processor can add 2 numbers 64-bits long each in one instruction. Larger values must be processed in several steps.

Registers are memory units, which can contain each one word ready for immediate processing. Access to registers is much faster than to the primary memory.

Input and Output

Input unit is a device through which data and programs from the outside world are entered into the computer. The first input units interpreted holes punched on paper tape or cards. Modern-day input devices include the terminal keyboard, the mouse, and scanning devices used at supermarkets as well as various sensors.

Output unit is a device through which results stored in the computer memory are made available to the outside world. The most common output devices are printers and displays of various kinds.

Control Unit

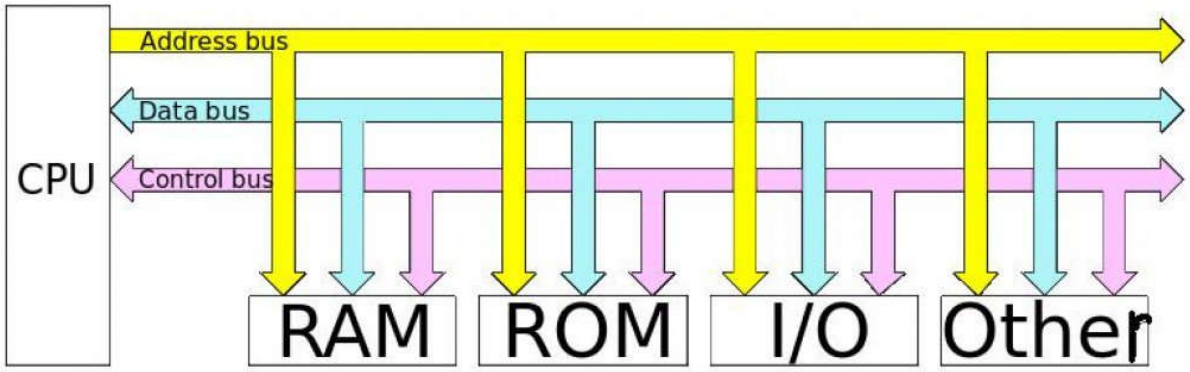
The control unit is the organizing force in the computer, for it is in charge of the ***fetch-execute cycle***. There are two registers in the control unit:

1. The ***instruction register (IR)*** contains the *instruction that is being executed*,
2. The ***program counter (PC)*** or ***instruction pointer(IP)*** contains the *address of the next instruction* to be executed.

Because the ALU and the control unit work so closely together, they are often thought of as one unit called the ***Central Processing Unit***, or ***CPU***.

The flow of data between the units is done over the group of connections called ***bus***.

Structure of Buses in the Computer System



Address Bus – transfer of addresses (from which, to which data is sent)

Control Bus – the instruction = what to do (read, write, get ready)

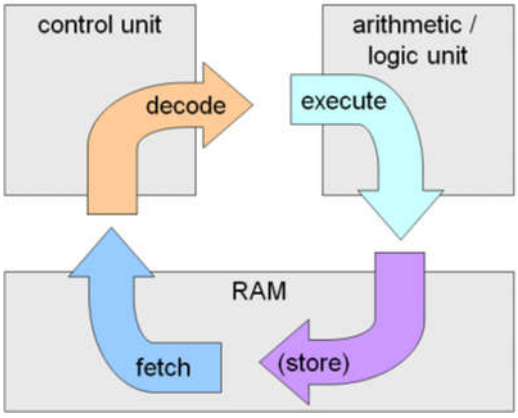
Data Bus – the data being transferred

Fetch-Execute Cycle

One of the definitions of a computer states:

> A computer is a device that can store, retrieve, and process data.

That clarifies the limitations of what a computer system can do. It retrieves an instruction from the memory, loads related data from the memory, processes that, and stores the result. Then it retrieves the next instruction … etc. This process is called ***fetch-execute cycle.***



The steps in the fetch-execute cycle are:

1. ***Fetch an instruction***
2. ***Decode the instruction***
3. ***Get related data***
4. ***Execute the instruction***

Fetch instruction

One of the CU registers – the **program counter** – contains the address of the following instruction ready for processing. CU copies the bits representing this instruction into the instruction register.

After the instruction retrieval the program counter must be modified to point to the next instruction. Usually it is done by incrementing its value by 1, if the instruction occupies one cell. However, the execution of the instruction may modify its value, so the next instruction can be taken from another location of the memory.

Decode the instruction

The logic circuits in the CU are activated according to the value of bits in the instruction. In this phase is also determined the number of operands and their source – registers, memory, ports etc.

Each processor has just a predefined set of instructions, which are understood – it is defined by the ***instruction set.*** There are essentially 2 attitudes:

1. ***CISC – Complex Instruction Set*** – a lot of specialized instructions; some of them are used rarely (e.g. x86 processors);
2. ***RISC – Reduced Instruction Set*** – it implements a few frequently used instructions; other operations must be done as their combinations (e.g. ARM processors).

Get related data

In this phase are copied data into related registers defined in the instruction, if necessary.

Execute the instruction

Once an instruction has been decoded and any operands (data) fetched, the control unit is ready to execute the instruction. Execution involves sending signals to the arithmetic/logic unit to carry out the processing. In the case of adding a number to a register, the operand is sent to the ALU and added to the contents of the register.

Von Neumann bottleneck

The fetch-execute cycle induces one problem brought by the von Neumann architecture. It is the ***von Neumann bottleneck*** – **the *limited throughput* (*data transfer rate*) between the CPU and memory compared to the amount of memory.**

Because program memory and data memory cannot be accessed at the same time, throughput is much smaller than the rate at which the CPU can work. This seriously limits the effective processing speed when the CPU is required to perform minimal processing on large amounts of data. The CPU is continually forced to wait for needed data to be transferred to or from memory.

In modern computers it is usually solved by ***cache*** between the CPU and memory.

Architectures different from the von Neumann

There are various architectures, which differ significantly from the von Neumann architecture:

- ***Analogue computers*** – it uses analogue physical phenomena (they change continuously), which model the solved problem. An example is Deltar computer, which modeled flooding in the Netherland; electric current represented the water flow in the Dutch channels. Other examples can be found here: http://www.cracked.com/blog/4-amazing-computers-made-before-microchips-existed/.
- ***Harvard Architecture*** – it has 2 separate memory units – one stores data, the other stores instructions. It is more secure, and it addresses the problems with the von Neumann bottleneck. On the other hand the hardware is more complicated.
- ***Synchronous processing*** – there are multiple processors, which apply the same code (program) to different groups of data.

Questions

1. How many different memory locations can a 16-bit processor access?
2. What is an instruction register? What is its function?
3. What is a program counter? What is its function?
4. What are the CPU registers? What purpose do they serve?
5. Recount the units of the von Neumann architecture and link them to real components
    a. of a laptop
    b. of a smartphone
6. Punched cards and paper tape were early input/output mediums. Discuss their advantages and disadvantages.
7. What is a von Neumann bottleneck? How could the cache reduce the effects of the said bottleneck?
8. "**The content of the cells may represent both, data or instructions** – that depends on the interpretation. This feature can be misused exploiting problems in the memory management."
   How could be the described feature misused?