Intro to CS: stored program concept

Panos Hilaris
Stored program concept

John von Neumann’s idea:

*Store programs, together with data, in the memory of the computer.*

Memory contains: (a) instructions, (b) data.

Conceptually, programs and data seem very different. In fact the first computers, distinguished between them and stored them in different places.

Another point of view: Treat programs as data.
Inside hardware

The two hardware parts, that are ‘inside’ the computer:
Processor or CPU (Central Processing Unit)
Main memory (RAM: Random Access Memory)

(Remember: the other part is input/output devices)
Components of CPU/memory

arithmetic and logic unit (ALU): circuitry for arithmetic and logic operations (here the ‘real’ work is done)

close control unit (CU): responsible for running programs (here the ‘book-keeping’ job is done)

• Fetch next instruction
• Decode instruction
• Execute instruction (activate relevant circuitry in ALU)

registers (in ALU and CU): hold data for use by arithmetic and logical operations

bus: wires connecting CPU and main memory
Review question

What is the difference between a computer and a calculator?
Review question

What is the difference between a computer and a calculator?

Answer: The computer is programmable, and this is achieved through the control unit. The control unit is responsible for running programs from memory.
Machine language

Most programmers write their programs in a high-level programming language, like Java, C, C++, Ada, Pascal, etc.

However, the CPU does not understand these languages. The CPU has its own machine language, which tells it what circuitry to activate to do some operation.

Compilers and interpreters translate from high-level language to machine language.
Example of high-level instruction

x := y + z

(a simple addition)

x, y, z are variables, i.e., positions (or addresses) in memory with a specific name, to which the programmer can refer.

For example, let’s say x is at the 100th position in memory, y at the 237th and z at the 17th.
Corresponding machine language instructions

(4 instructions in a typical computer)

1. Obtain first value (y) from memory
2. Obtain second value (z) from memory
3. Add two values in ALU
4. Store result in memory (at x)
Corresponding machine language instructions

LOAD RegA, 237
LOAD RegB, 17
ADD RegA, RegB, RegC
STORE RegC, 100
Corresponding machine language instructions

In fact, a bunch of zeros and ones!
Control unit registers

The control unit has two special registers:

- program counter (PC): contains the address (in binary) in main memory of the next instruction

- instruction register (IR): contains the instruction (in binary) that is currently being executed
Fetch-decode-execute cycle

1. Fetch the instruction from the address in memory pointed by PC (Program Counter) and put it in IR (Instruction Register), then increment the Program Counter by one, so that it points to the next instruction.

2. Decode the binary sequence in IR to get the meaning of the instruction (what circuitry must be activated in ALU).

3. Execute the instruction (activate the relevant circuitry in ALU).

and again and again . . .
Execution of a program

A program is executed by having the CU go over each of the instructions in the program sequentially.

The cycle is executed for each instruction until the end of the program is reached.

Sometimes a program can get out of this ‘straight-line’ order, if it has a JUMP instruction. A JUMP instruction can force the CU execute an instruction which is different from the next instruction. JUMP instructions are the building block of loops.
The power of the computer:

A typical computer can do this fetch-decode-execute cycle millions of times per second, and without errors.