

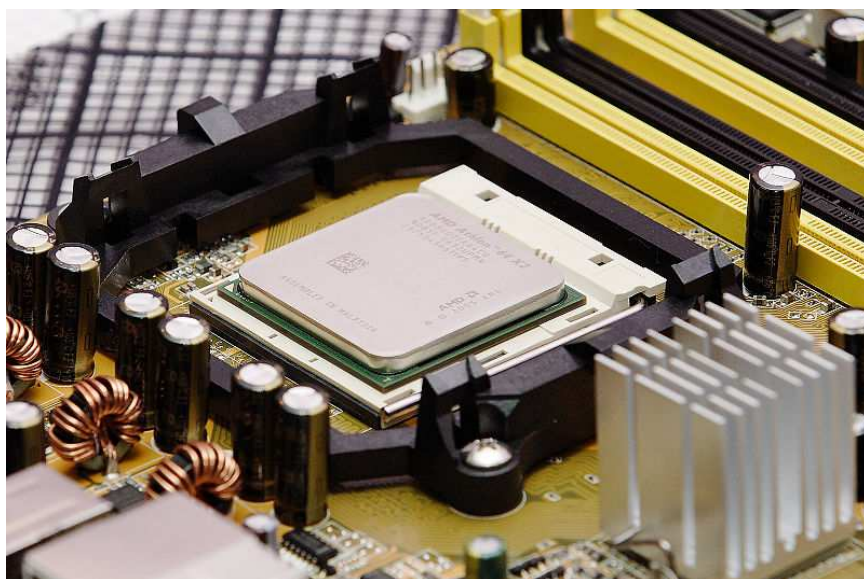
## The human computer Role-sheet: Control Unit

**You're the control unit.** The control unit (often called a control system or central controller) directs the various components of a computer. It reads and interprets (decodes) instructions in the program one by one. The control system decodes each instruction and turns it into a series of control signals that operate the other parts of the computer. It is like the nervous system of the computer.

A key component of a computer that the control unit interacts with is the program counter, a special memory (a register) that keeps track of which location in memory the next instruction is to be read from.

Another key component that the control unit interacts with is the ALU (arithmetic and logic unit). The ALU is the only part in the main computer that can do any maths. The control unit calls the ALU when it wants to e.g. add two numbers.

Finally, the control unit routinely interacts with the memory. Program instructions are stored in memory, at the location indicated by the program counter. When the control unit decodes an instruction, it will often have to fetch some data that also lies in memory, for instance, add 1 to the content of a given memory cell.



## Roles:

We need at least 4 people to role-play the control unit.

- 1 “messenger”: this person will travel from one part of the room to the other to read, write and process information from and to the other parts of the computer (i.e. the memory, the ALU, the registers and the program counter). All the persons role-playing in the control unit collaborate to decide what the messenger’s next action should be.
- 2 “decoders”: these 2 persons are responsible to decode and interpret the hexadecimal data that the messenger is reading from the other parts of the computer. This data includes numbers, texts but also instructions. The 2 “decoders” stand at the white board, receive information from the messenger, and collaborate with the other person in the control unit to decide what the messenger’s next action should be
- 1+ “schedulers”: these persons are responsible to decide what the next action of the messenger should be. The messenger and the decoders above also participate in this decision. Schedulers should follow the basic function of a control unit as indicated below.

## Basic function:

The control system's function is as follows:

1. Read the code for the next instruction from the cell indicated by the program counter:
  - Read the current location in program counter (a hexadecimal value)
  - Read the value in memory at this location (a hexadecimal value)
2. Decode the numerical code for the instruction into a set of commands or signals for each of the other systems.
  - Identify the 3-letter instruction
  - Identify the arguments of the instruction, which may be either a number or a memory address. Both are coded on 1 byte (i.e. 2 hexadecimal digits). Memory addresses can be recognized because they all start with 1111... (i.e. hexadecimal value F). For instance, hexadecimal value F1 is a memory address, while hexadecimal value 01 is a number (the binary number 0000 0001, i.e. decimal number 1)
3. Increment the program counter so it points to the next instruction.

- Read the current value of the program counter
  - Go to the ALU: position this value at input A, then position the number “1” at input B, then position the instruction “add” at input O
  - Wait until the value at output S is “true”
  - Read the result of the operation (counter+1) at output R
  - Write this value in the program counter
4. Read whatever data the instruction requires from cells in memory. The location of this required data is typically stored within the instruction code.
  5. Provide the necessary data to an ALU or register and instruct the ALU to perform the requested operation.
  6. Write the result from the ALU back to a memory location or to a register
  7. Jump back to step (1).

## **How to use the memory?**

- To read data in memory, the messenger should simply tell the address of the emplacement it wants to read (e.g. “F2”) to the people role-playing the memory.
- To write data in memory, the messenger should tell which value to write in which address, everything in hexadecimal value (e.g. “02” in “F2”)

## **How to use the program counter?**

Use it exactly like the memory (see above): tell it to read or write anything. Everything should be in hexadecimal data.

## **How to use the register?**

Use it exactly like the memory (see above): tell it to read or write anything. Everything should be in hexadecimal data.

## How to use the ALU?

The ALU in our human computer is composed of 3 inputs and 2 outputs (which we collectively refer to as “ports”). Each input/output port is role-played by one person (hence a total of 5 persons).

Input A (receives the first argument in the operation to be performed, e.g. A+B)
Input B (receives the second argument to be performed, e.g. A+B)
Input O (receives the identifier of the operation to be performed, e.g. addition)
Output S (outputs the status of the ALU's operation. If S=true, then the following output ® contains the result of the operation requested using inputs A,B,O. If S=false, then the output ® is not ready, and its value is undefined)
Output R (outputs the result of the ALU's operation requested using inputs A,B,O; IF the output S = true.

To use the ALU to process an operation on 2 numbers, do the following steps:

- Write (i.e. tell the right people to write) data in input A
- Write data in input B
- Write the operation in input O: there are only 2 operations you can ask in our human computer: “add” and “compare”.
- Wait until the output S = true
- Read the data in output R. If operation is “add”, the result is A+B. If operation is “compare”, the result is true if  $A \leq B$ .

## How to decode an instruction?

- Identify the 3-letter instruction
- Identify the arguments of the instruction, which may be either a number or a memory address. Both are coded on 1 byte (i.e. 2 hexadecimal digits). Memory addresses can be recognized because they all start with 1111... (i.e. hexadecimal value F). For instance, hexadecimal value F1 is a memory address, while hexadecimal value 01 is a number (the binary number 0000 0001, i.e. decimal number 1)

For instance, if an instruction is read in memory that says:

4D 4F 56 02 F0

you should be able to (quickly...) recognize that the first 3\*2 digits are the hexadecimal representation for a 3-letter instruction (M,O,V), that “02” is the hexadecimal representation for a number (because it doesn't start with “F”), and that this number is decimal value “2”; and finally that the last 2 digits are the address for a memory emplacement (because the first digit is “F”).

## A note about hexadecimal values

Every thing in the computer is stored as hexadecimal values. Every value that the control unit reads from memory or from the ALU is a hexadecimal value. Everything that the control unit should write in the memory or the ALU should be a hexadecimal value.

You don't have to decode anything (nearly).

If you give hexadecimal values A=02 and B=0B to the ALU, you don't need to realize these numbers are decimal values 2 and 11. The ALU will handle these numbers to, say, compute their sum – it is their problem, not yours. It will normally return “0D” (i.e. 13). Again, you don't need to realize this is “13” in order to store it in memory. Just instruct the memory to write this value “0D” in whatever address.

The only thing you really have to decode is the instruction keyword (3 letters), and to realize what type of data are the arguments of the instructions. For instance, in the instruction 4D 4F 56 02 F0, you have to be able to recognize (quickly) that

- 4D 4F 56 is the hexadecimal for “mov”
- “02” is a number, and “F0” is a memory address (because the second has the digit “F” and not the first)

However, in the instruction 4D 4F 56 F2 F0, you have to recognize that you need to move the content of a memory address F2 into another memory address F0. The steps needed to execute this instruction will be different (first read from F2, then write in F0)