Ce face CPU-ul cand ii dam o instructiune din codul masina?

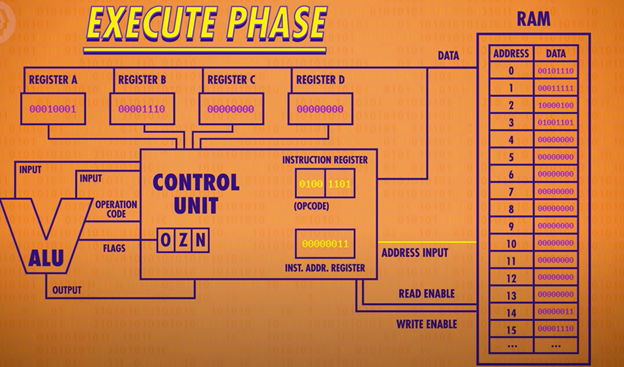
Scop:

* Sa intelegem ce se intampla mai jos de cat assembly
* Cum functioneaza un CPU
* Cum putem imbunatatii procesul?

La acest tutoriat am facut “**Microarchitecture**” si inseamna cum procesura executa instructii, cum se ocupa cu datele si cum efectueaza diferite operatii.

Recomand acest video pentru a intelege ce se intampla in CPU:

[The Central Processing Unit (CPU): Crash Course Computer Science #7](https://youtu.be/FZGugFqdr60?si=WpMz00X1HxrhpacG)



Cateva definitii:

Instruction Address Register -> are ca valoare pozitia unde se afla la executie.

Instruction Register -> Ce instructiune se face la acea linie de cod.

Instruction Table -> un “dictionar” unde sunt scrise instructiuni care se pot face cu elemente (mov, add, etc.) El in spate este doar o multitudine de logic units, daca una este activata, atunci se duce pe ea.

**Fetch Phase** -> Momentul cand l-om datele din memorie sau registrii si ii punem in control unit    
   
(Mai specific: luam data de pe pozitia Instrcution Address Register si o punem in Instruction Register.)

**Decode Phase** -> Momentul cand control unit afla ce trebuie sa faca cu datele de acolo    
   
(Mai specific: Se uita in instruction table sa vada ce trb sa se intample (primii 4 bites (doar exemplu)) urmatorii 4 bites sunt adresa de la care trb sa ia datele.) 

**Execute Phase** ->Momentul cand Face ce ii este zis la Decode.    
   
(Exemplu Load A. Citeste din ram la adresa la care trb sa se uite si da write in registrul A. (mov) Dupa incrementam Instruction Address Register.)

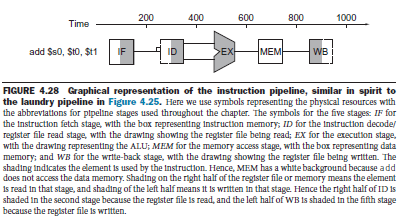
**Clock speed**->Se masoara in Hertz per second = Fetch Phase + Decode Phase + Execute Phase

**Alogirthm logic unit** -> este un circuit electronic digital complex care poate efectua operații aritmetice și logice. Va recomand acest video pt ALU : [How Computers Calculate - the ALU: Crash Course Computer Science #5](https://youtu.be/1I5ZMmrOfnA?si=oxb7eLOhQReIpM9B)

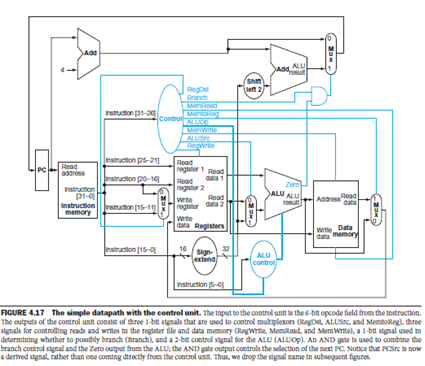
2.4 gigahertz =

2 400 000 000 hertz

<https://en.wikipedia.org/wiki/X86_instruction_listings> -> unde gaseste acest instruction table.

In carti o sa gasiti mai multe faze, in unele o faza este data numai la Algortihm Logic unit, alta la preluarea datelor din memorie si alta faza la punerea in memorie. Asta e facut pentru a impartii cat mai mult acest task, in mai mult bucati pentru ca mai tarziu sa le optimizam.

Daca sunteti curiosi cum arata circuitul din CPU un pic mai in detaliu:



“**Why a Single-Cycle Implementation Is Not Used Today**

Although the single-cycle design will work correctly, it would not be used in modern designs because it is inefficient. To see why this is so, notice that the clock cycle must have the same length for every instruction in this single-cycle design. Of course, the longest possible path in the processor determines the clock cycle. This path is almost certainly a load instruction, which uses five functional units in series: the instruction memory, the register file, the ALU, the data memory, and the register file.

Although the CPI is 1 (see Chapter 1), the overall performance of a single-cycle implementation is likely to be poor, since the clock cycle is too long. The penalty for using the single-cycle design with a fixed clock cycle is significant but might be considered acceptable for this small instruction set. Historically, early computers with very simple instruction sets did use this implementation technique. However, if we tried to implement the floating-point unit or an instruction set with more complex instructions, this single-cycle design wouldn’t work well at all.

Because we must assume that the clock cycle is equal to the worst-case delay for all instructions, it’s useless to try implementation techniques that reduce the delay of the common case but do not improve the worst-case cycle time. A single-cycle implementation thus violates the great idea from Chapter 1 of making the common case fast.

“   
Computer\_Organization\_and\_Design\_5th\_Edition