

Computer Architecture (Unit 2)

DATE : 23-02-2022

Course Structure

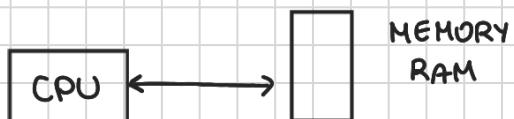
- Theory
- Assembly

EXAMS

- Midterm
 - Final
- } Average

HOW DOES THE CPU WORK

- CPU : does all the computation
- Memory : stores programs and data



Memory has sequences of bytes inside

SRAM (static)

- expensive
- made of flip flops
- fast
- used if you have few bits to store

DRAM (Dynamic)

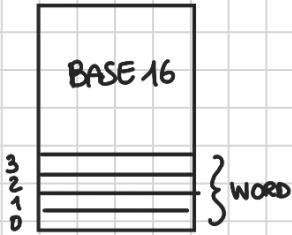
- cheaper
- slow
- mostly used if we need a lot of storage

- KILO-BYTE 10^3
- MEGA-BYTE 10^6
- GIGA-BYTE 10^9 → example : 8 GB = 8 billion bytes
- TERA-BYTE 10^{12}

NOTE : 1024 Kibibytes

- CPU needs to read and write data from the memory so it's necessary for each byte to have a name (address) to identify it

MEMORY



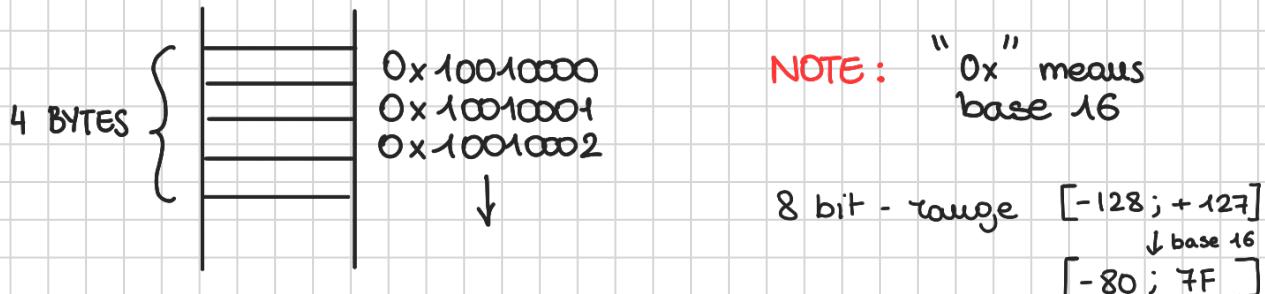
- We typically use base 16 because it's easier to convert it into binary
- CPU reads and writes 4 bytes (WORD) at a time to get more information

- We normally use a x32 architecture since we are reading 4 bytes

$$1 \text{ byte} = 8 \text{ bit} \Rightarrow 4 \text{ bytes} \cdot 8 \text{ bit} = 32 \text{ bit}$$

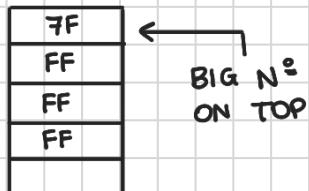
(x64 bit architecture reads 8 bytes at a time, $8 \cdot 8 = 64$)

MEMORY



LARGEST 2-comp n° REPRESENTABLE IN 32 BITS (4 byte) $\rightarrow 0x\text{FFFFFFF}$ ($2^{32-1} - 1$)

BIG ENDIAN COMPUTER



SMALL ENDIAN COMPUTER



- C language is close to assembly

We need to encode our data in order to put it in the memory:

- ASCII (ENCODING STANDARD)

HOW TO STORE A STRING IN MEMORY?

- Each byte will contain ONE character
- In order to store "Ale" we need to convert each character into ASCII

65
108
101
32

} A → 65 in ASCII
 e → 108 = =
 e → 101 = =
 space → 32 = =

HOW TO KNOW THE STRING ENDED?

- we put zero in the last byte
- we put the length of the string anywhere in the string

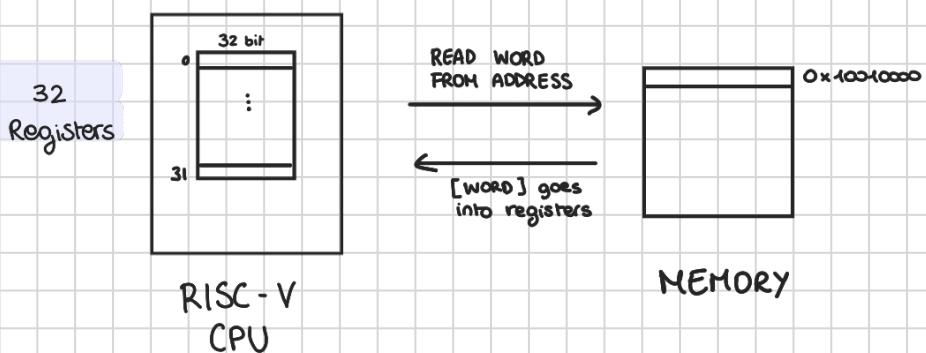
HOW TO KNOW IF THE CONTENT OF A BYTE IS A NUMBER OR AN ASCII ENCODING?

You can't. The problem will specify it

We have small memory (REGISTER) inside the CPU.

RISC-V CPU → 32 registers

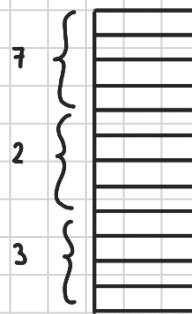
- CPU can read words in a particular address and store it inside the register.



HOW TO STORE ARRAYS

- We store arrays elements one after the other

$$a = [7, 2, 3]$$



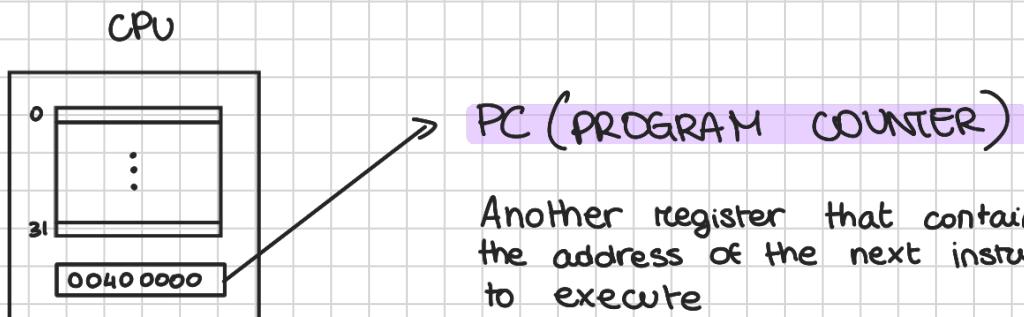
- We can put the length of the array wherever you want in order to know when it ends

HOW TO STORE PROGRAMS

ADD (instruction) → adds the content of two registers and puts it into another =

NOTE: You cant add the content of two addresses in a memory. You have to put it in the CPU first.

We have instructions for many purposes which are executed in the CPU.



- ① We read the address and take the word contained in the memory
- ② We put it inside a register in the CPU
- ③ We perform the instruction inside the CPU and put the result in another specified register. Meanwhile, the address in the PC will be increased by 4
(NOTE: CPU reads 4 bytes at a time)
- ④ Read next address / instruction in the PC

FETCH

&

EXECUTE

NOTE: RISC-V starts at 0x00400000 while DATA at 0x10010000