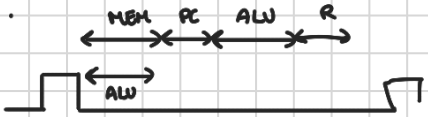
Computer Architecture

DATA : 16/03/2022

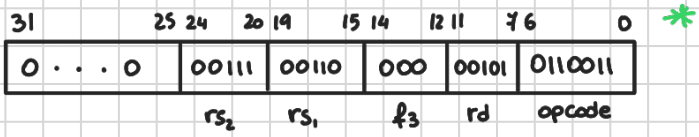
PC (PROGRAM COUNTER) - contains the address of the instruction that will be executed. It stores 32 bits / 4 byte.

CLOCK - This is a single-clock architecture.
We execute one instruction in one CK.
We use LOAD to start the execution.
(1 - ON , 0 - OFF)

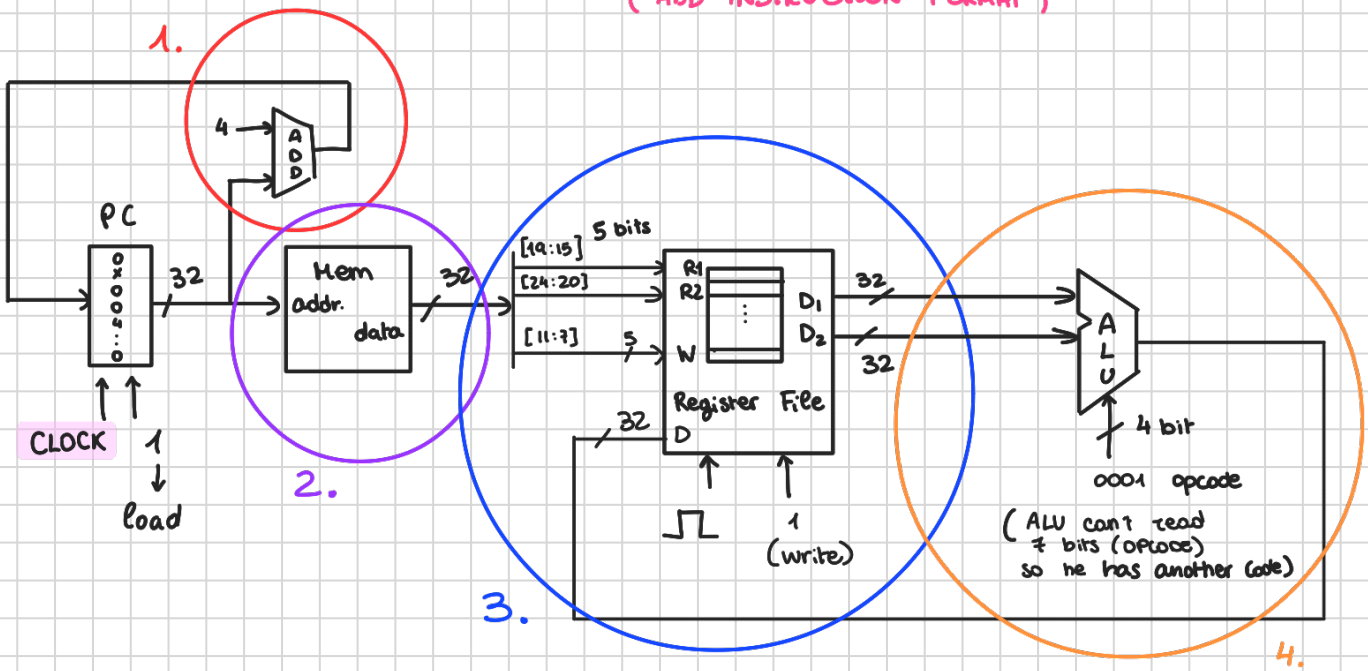


LET'S SEE HOW TO EXECUTE AN ADD INSTRUCTION

add x5, x6, x7
(address 0x00400000)



(ADD INSTRUCTION FORMAT)



1. So the PC contains the address of the instruction and in order to do the next instruction, we will use an adder to increment our address of 4 bytes (+4).

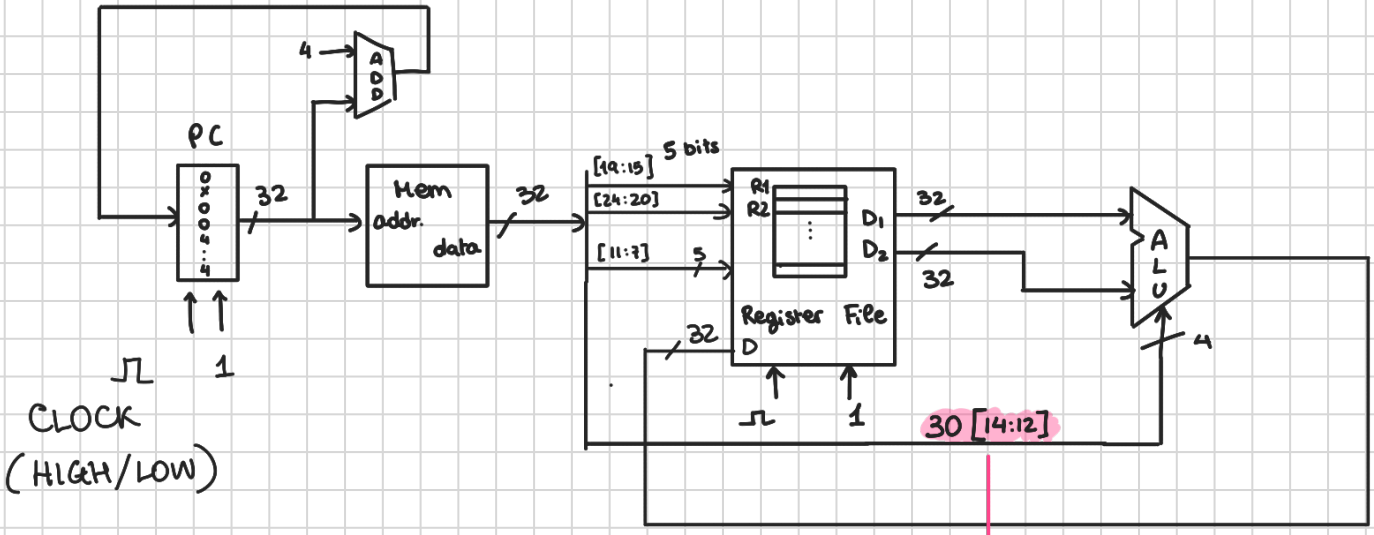
2. Here, we take the address and find the corresponding instruction. We will see the format * of the instruction

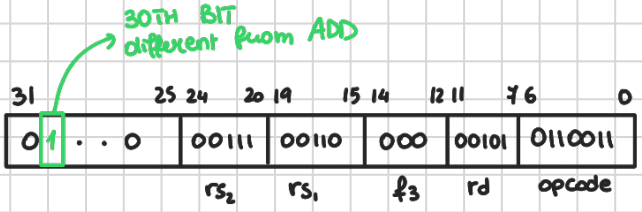
3. Here, we load the addresses of our registers to the CPU.
[19:15] is the part of the format where we have Rs1.
[24:20] " " " " " Rs2.
[11:7] " " " " " Rd.
And then, we read their contents using other registers of the CPU and send to the ALU.

4. Now we perform the actual operation (addition). The ALU will know what operation to perform thanks to the code sent through the control lines (ALU has 4 control lines, so 4 bits)
The output will go to the destination register D.

SUB INSTRUCTION

sub x5, x6, x7 (0x00400004)



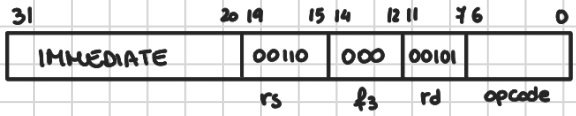


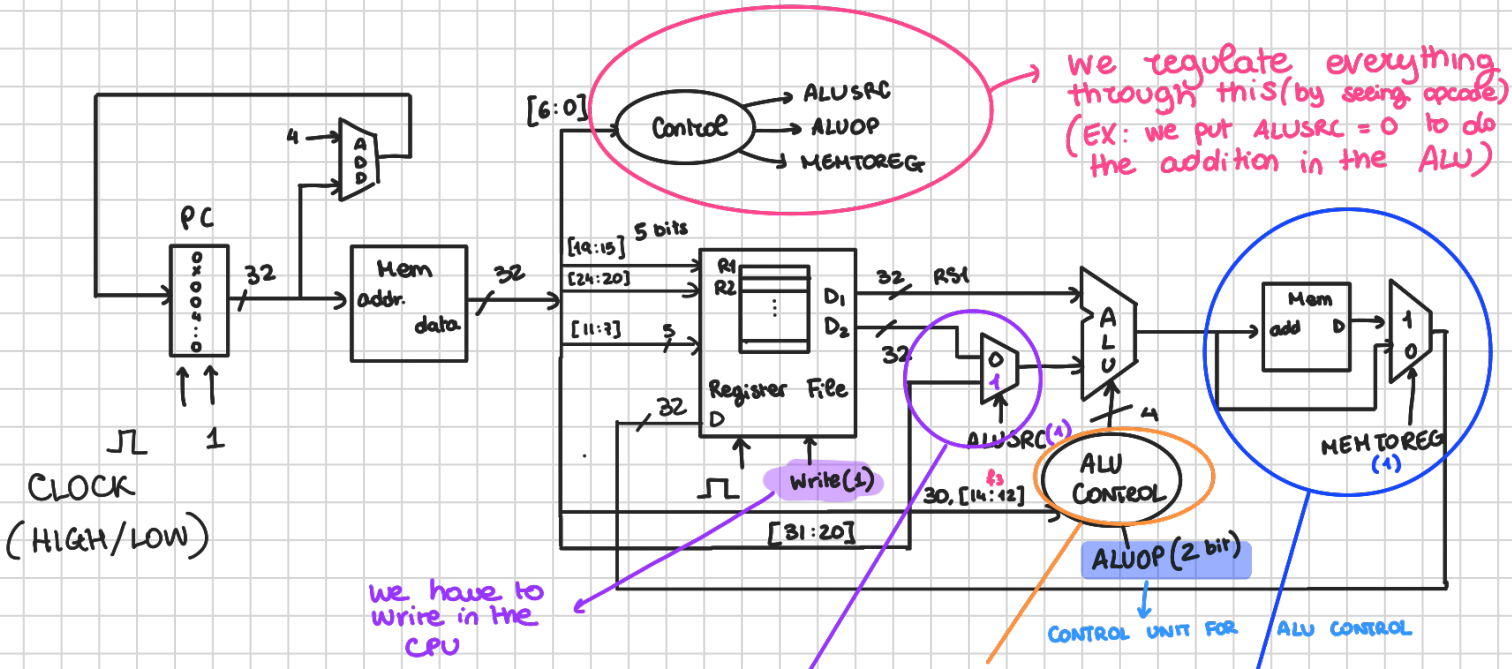
(SUB INSTRUCTION FORMAT)

* for the "sub", the opcode is the same as add. The only thing that differs is the 30th bit in the F7. So here we send the 30th bit + f3 ([14:12])

LOAD WORD

lw rd, offset(rs1)



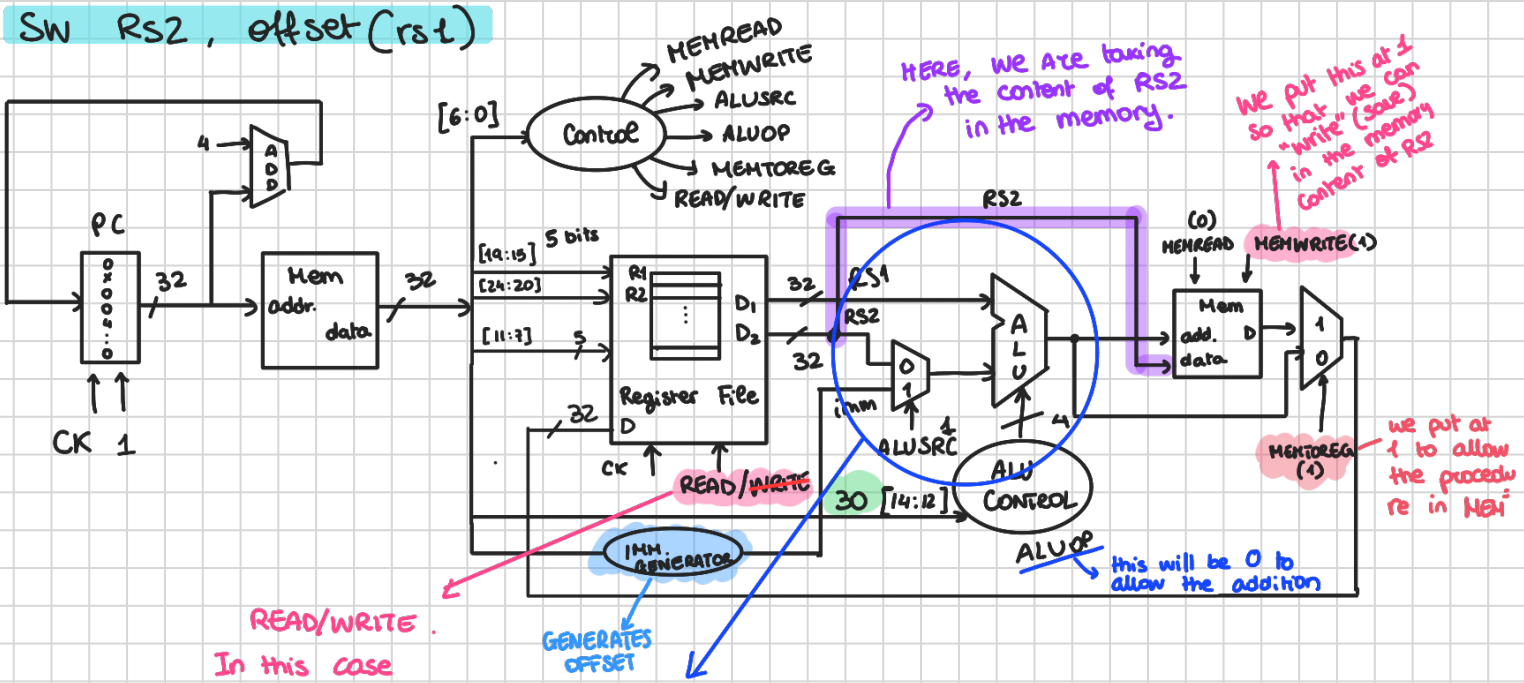


OFFSET. This way we add it to rs1 ([31:20]) We put the ALUSRC to 1 so we can select the second input with the offset

opcode isn't same so we use a general ALU CONTROL

Here, we get the address, take the content and put it into our destination register. To do so, we have to put MEMTOREG = 1 so we can select with the MUX (multiplexer)

SW RS2, offset(rs1)



READ/WRITE. In this case (SW instruction) we will put this on READ because we don't have to write on the memory.

We will not only take the content of RS2 in the MEM but ALSO do the sum of RS1 with the offset so the ALUSRC = 1 in order to allow this.

VALUES THEY SHOULD BE PUT AT FOR EACH INSTR.

ADD

- ALUSRC = 0
- ALUOP = 0
- MEMTO REG = 0
- READ/WRITE = WRITE
- MEMREAD = 0
- MEMWRITE = 0

SUB

- ALUSRC = 0
- ALUOP = 1
- MEMTO REG = 0
- READ/WRITE = WRITE
- MEMREAD = 0
- MEMWRITE = 0

LW

- ALUSRC = 1
- ALUOP = 0
- MEMTO REG = 1
- READ/WRITE = WRITE
- MEMREAD = 1
- MEMWRITE = 0

SW

- ALUSRC = 1
- ALUOP = 0
- MEMTO REG = 1
- READ/WRITE = READ
- MEMREAD = 0
- MEMWRITE = 1