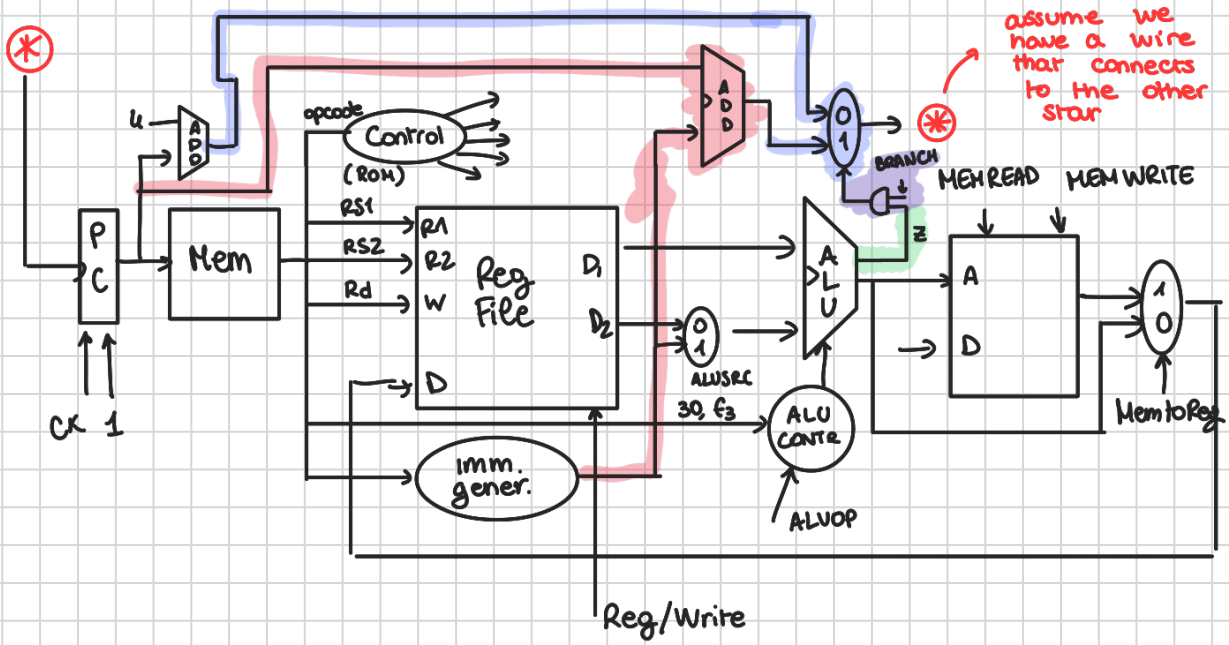
Computer Architecture

DATA: 23/03/2022



BEQ r1, r2, offset → PC ← PC + offset

THIS INSTRUCTION NEEDS TO:

- compare the content of 2 registers
- add or not a certain offset to PC

We can use the ALU. We set it to perform a substraction. If it's zero, then the content of 2 registers are equal.

Usually, ALU's have secondary outputs called "z". If the result of the sub is zero, then the output of z will be 1

Since we need to add a certain offset to the PC, we will need another adder. It will have in input the current PC and the immediate.

We add a multiplexer in order to select whether doing PC+offset if registers are equal otherwise PC+4

We need another control for "beq" because what if we wanted to do the "sub" instruction between 2 regs.? What if it gives 0? It would jump instead of storing the result of sub. That's why a BRANCH control (which will be put at 1 for beq) It will be put in AND with z.

HOW WILL BE THE CONTROLS SET FOR BEQ?

ALUSRC - 0 (we need to do substraction in ALU)

ALUOP → it has to do substraction (we didn't study yet)

MEMREAD - 0 (we don't have to read memory, only used with lw)

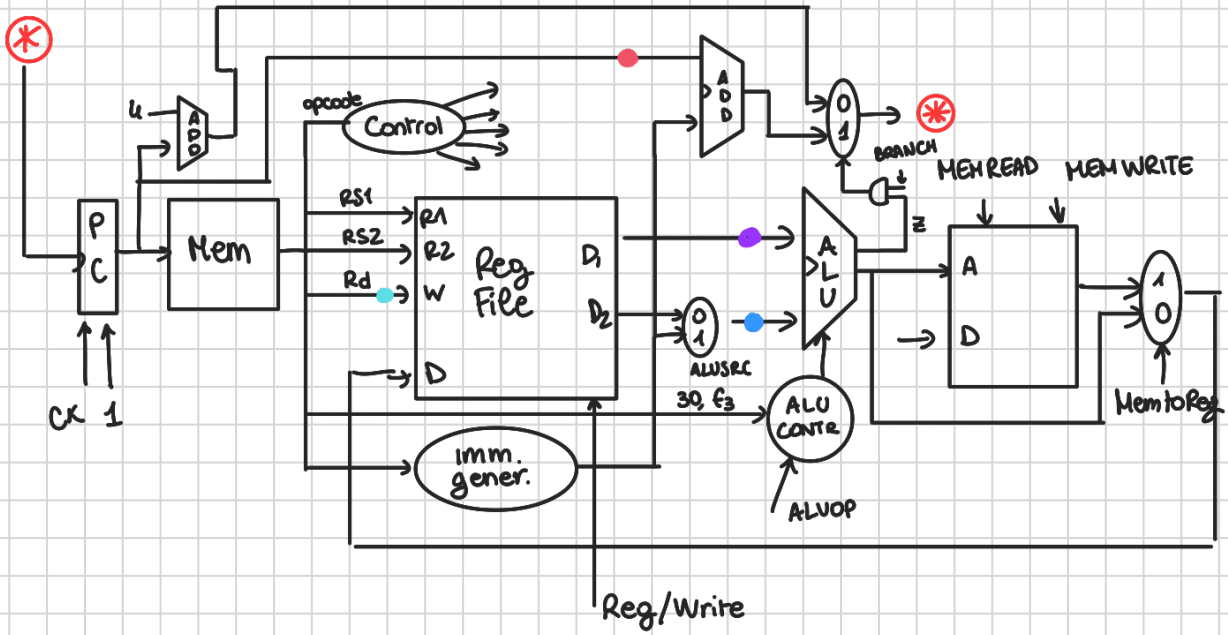
MEMWRITE - 0 (used for sw only)

MEMTOREG - DON'T CARE

REGWRITE - 0

BRANCH - 1

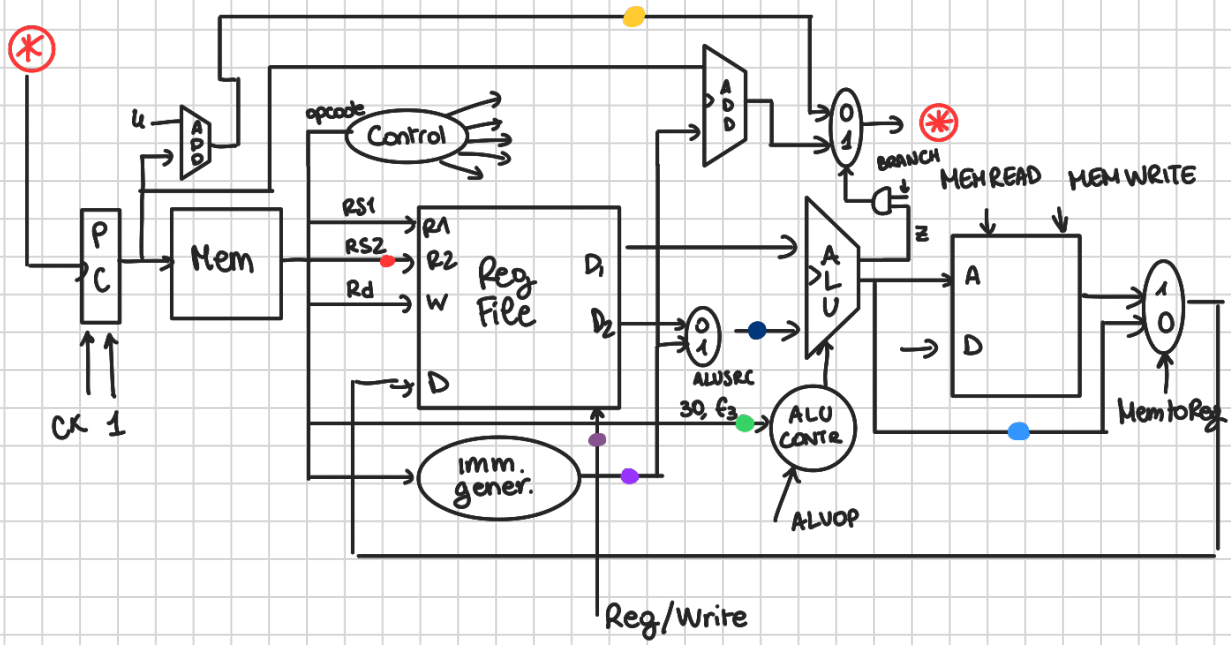
LET'S ASSUME WE'RE EXECUTING **SW x6, 8(x12)**



- here, we will find the immediate 8
- here, we have the content of x12
- here, we have the PC
- here, we have part of the immediate
  (it's technically wrong but REGWRITE = 0 so all cool)

NOTE: DONT CARE isn't a value that you can read when an architecture runs. "NOTHING" also doesn't exist

LET'S ASSUME WE'RE EXECUTING ADD 5, 6, 7



- here, we have the content of Register 7
- here, we have the result of R6 + R7
- here, we have garbage
- here, we have PC+4 (next instruction)
- here, we will have f3 (000) and 30th bit (1)
- here, we have content of R7.
- here, we have 1 (REGWRITE)

NOTE: better to say garbage. "Don't care" would be error

This is a slow architecture because it does one instruction in one clock cycle

The slowest instruction is lw, and then sw.

The clock cycle should be very long and the frequency very slow.