

## Chapter-4

(How instructions are being executed inside processor?)

$$\text{CPU time} = \underbrace{\text{Instruction count}}_{\substack{\text{Determined by the} \\ \text{language or platform} \\ \text{you are using.}}} \times \underbrace{\text{CPI} \times \text{Clock Cycle Time}}_{\text{Determined by CPU hardware.}}$$

→ How many Risc-V instructions we need to complete the task/code?

\* Two types of Risc-V implementations -

✓ (i) A simplified version (Slow, too much time wasted)

✓ (ii) A more realistic pipelined version (Optimized)

\* We will only look into subset of instructions divided into 3 categories -

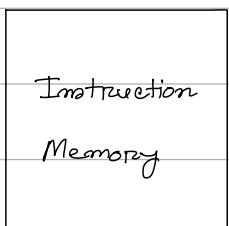
(i) Memory Reference :  $ld, sd$  → (main memory associated)  
     $l(i), s(g)$

✓ (ii) Arithmetic / Logical : add, sub, and, orz (P)

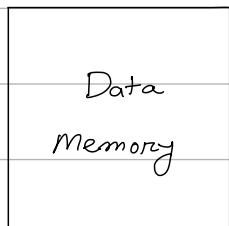
(iii) Control Transfer : beq ✓

    → (Branching)

### Instruction Execution :



(holds the instructions only)



(Refers to the main-memory)

e.g. while using load/store instruction,

this memory is accessed.

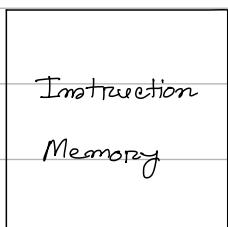
# Instruction Execution

- PC → instruction memory, fetch instruction
- Register numbers → register file, read registers
- Depending on instruction class
  - Use ALU to calculate
    - Arithmetic result
    - Memory address for load/store
    - Branch comparison
  - Access data memory for load/store
  - $PC \leftarrow \text{target address or } PC + 4$

#8000

PC contains the address of the instruction in the program being executed.

Now, go to that address in the



and fetch that instruction

#8000 → add x<sub>20</sub>, x<sub>21</sub>, x<sub>0</sub>

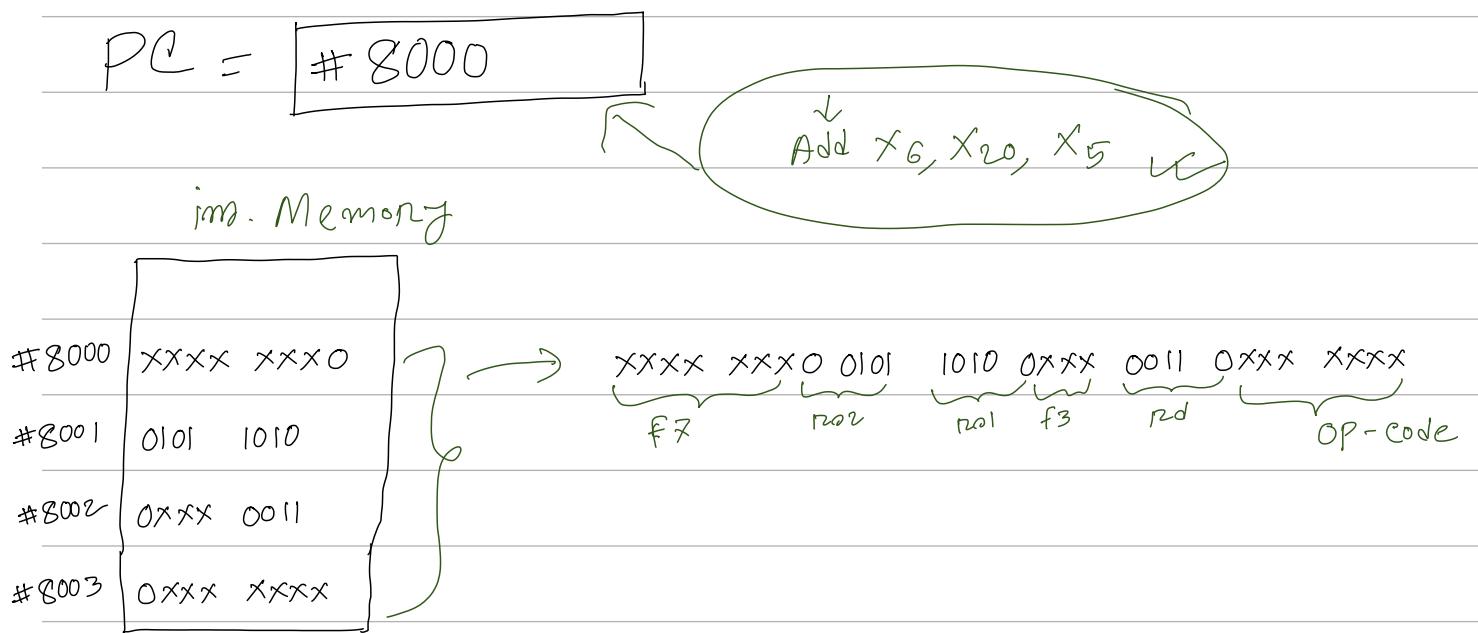
the fetched instruction will now be decoded.

↳ the registers associated with that decoded instruction

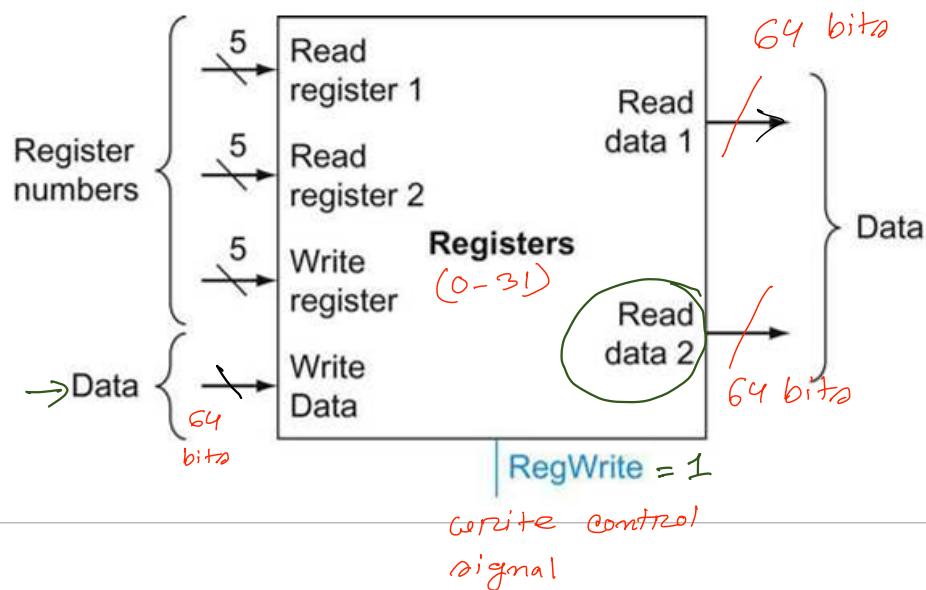
will be activated to read data.

## Sample Simulation of the ins. exe.

Steps



# Register File



(i) Read  $\Rightarrow$  Provide the register number that you want to read through the **Read register bus**.

$\Rightarrow$  Data output will come from **Read data bus**.

(ii) Write  $\Rightarrow$  Provide the register number that you want to write through the **Write register bus**.

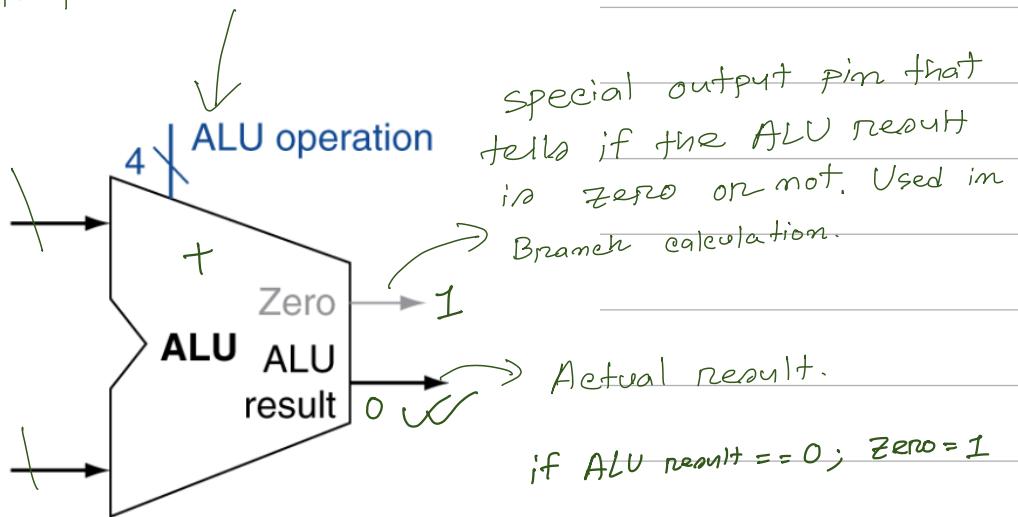
$\Rightarrow$  Provide the data to be written via the **Write data bus**.

$\Rightarrow$  **Write control signal** must be asserted.

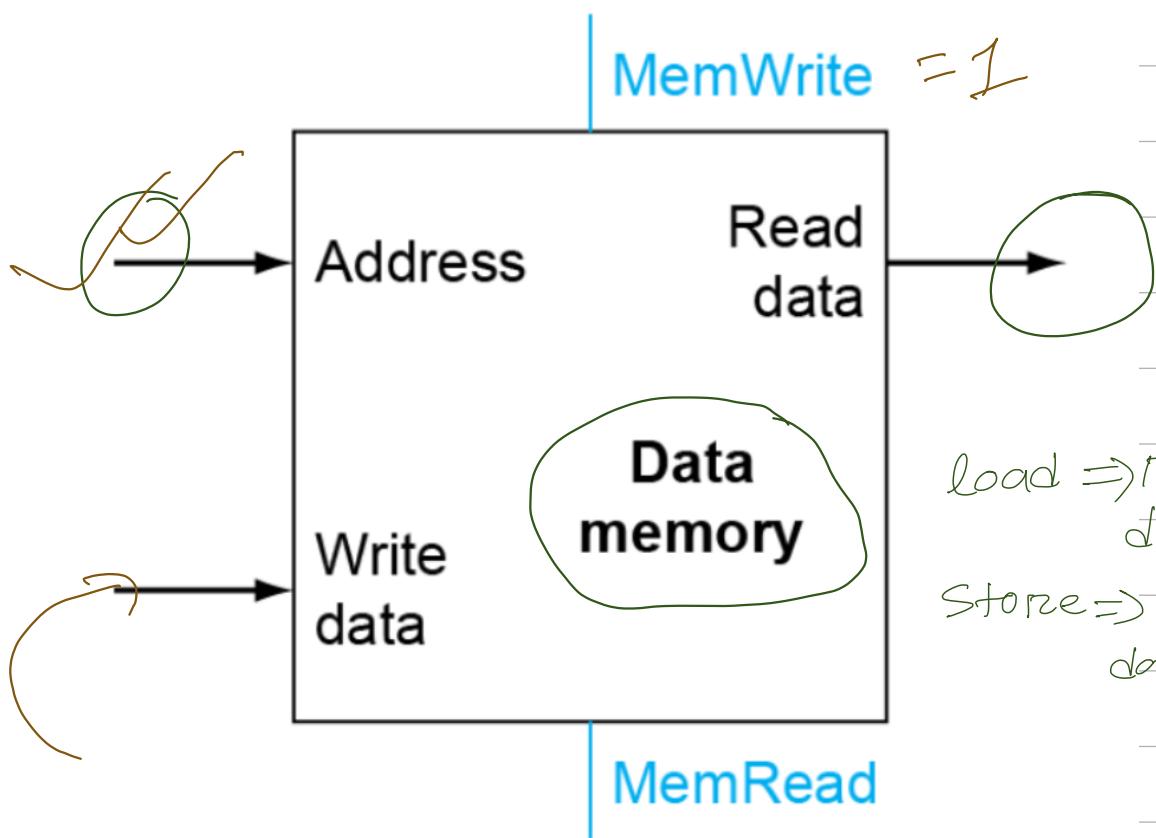
## ALU (Arithmetic Logic Unit)

tells ALU which operation to perform on the inputs.

size of both inputs must be same

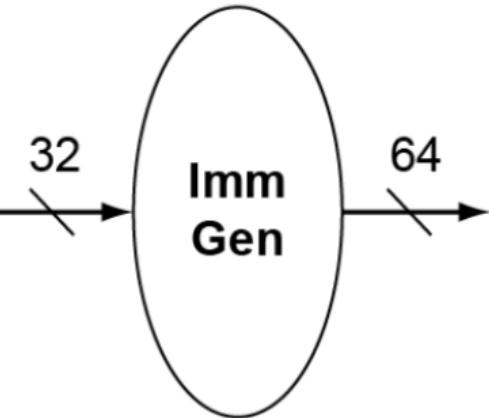


## Data Memory



## Immediate Generation unit:

↳ Expands 12 bit immediate to 64 bit value.



⇒  $(d \times 2^1), 10 (x_{22})$

Size of immediate in I type instruction  
is 12 bits.

⇒ Size of the complete instruction is 32 bits.

⇒ which is entered  
as input in the  
immediate generation  
unit.

⇒ This unit then  
extracts the 12 bit  
immediate and expands  
it into 64 bit.