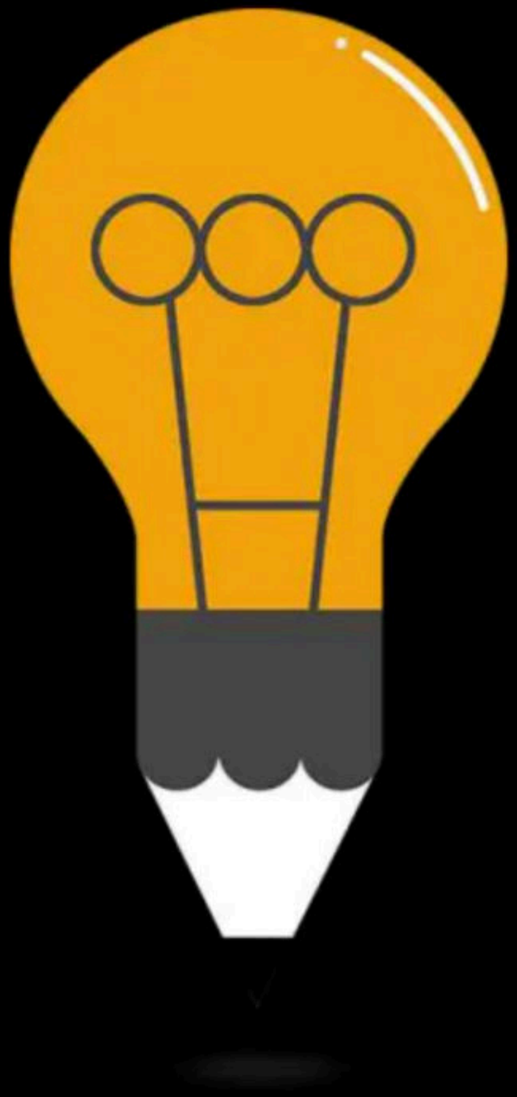
# Registers and Memory Access

Complete Course on Computer Organization & Architecture for GATE 2024 & 2025

**Vishvadeep Gothi** • Lesson 2 • Feb 14, 2023

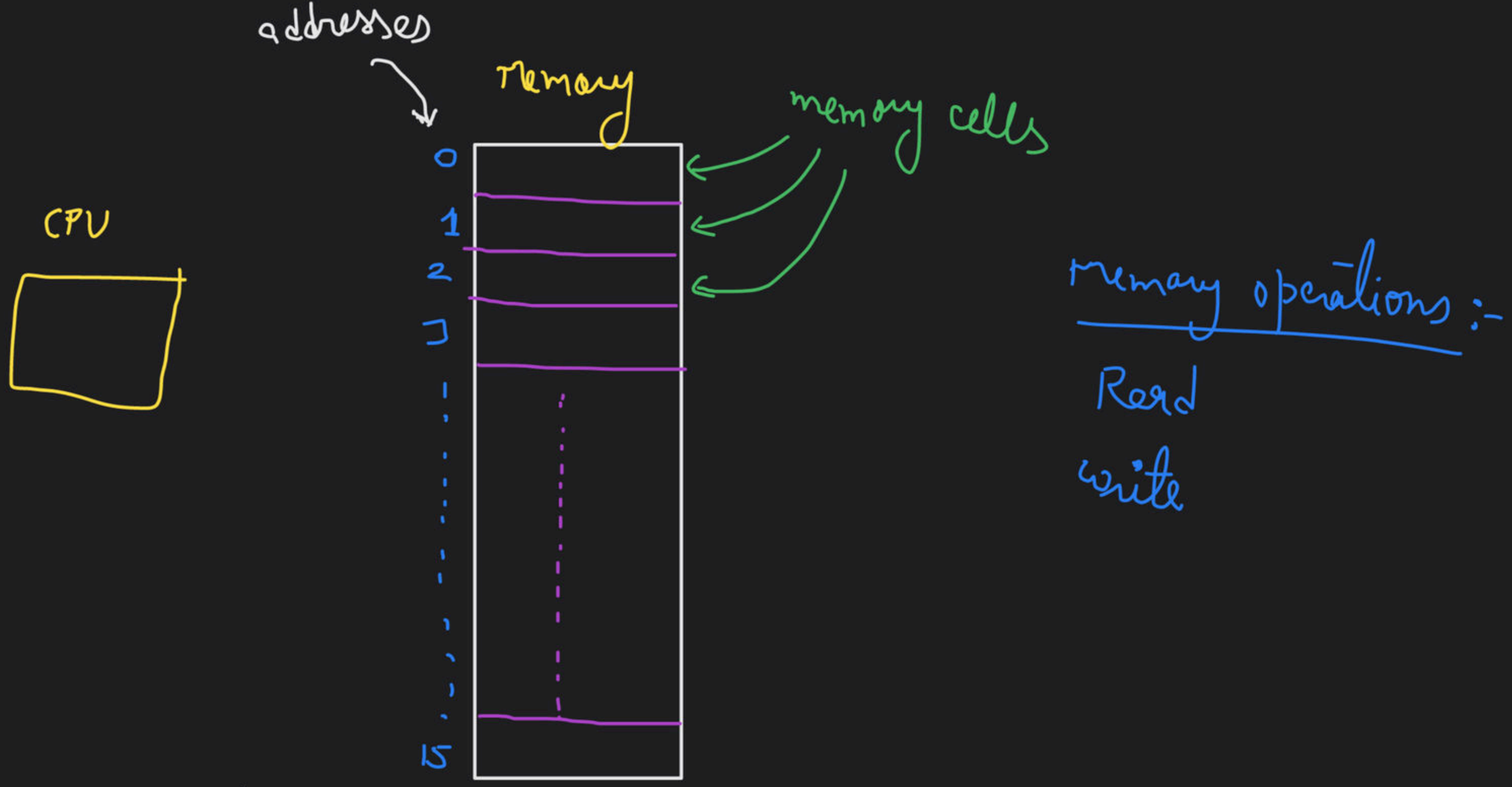


# CPU Registers

By: Vishvadeep Gothi

# Motivation

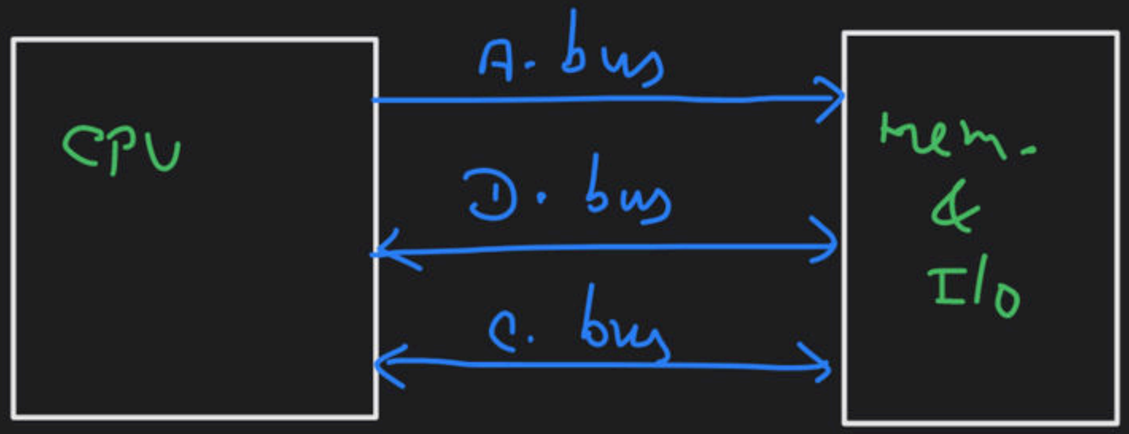
Hard-work can beat the talent




memory operations :-

Read

write




## Read :-

1. CPU sends address to memory through address bus
2. CPU sends enabled read control signal to memory
3. Memory performs read on given address and sends content present on that address through data bus.

## write operation :-

1. CPU sends add. to memory through address bus
2. — " — data to — " — " — data — " —
3. CPU sends enabled write control signal to memory.
4. Memory writes given content on given address.

# CPU Register

Small memories inside CPU

# CPU Registers

- CPU Register
  - General Purpose Registers (GPRs) → R0 , R1, R2, R3 , R4, .. ....
  - Special Purpose Registers

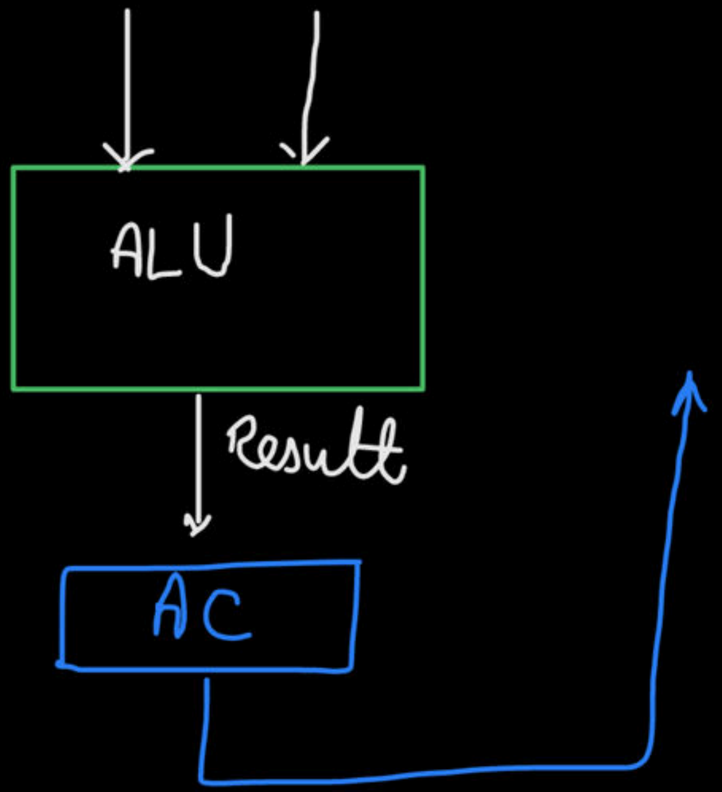
# CPU Registers

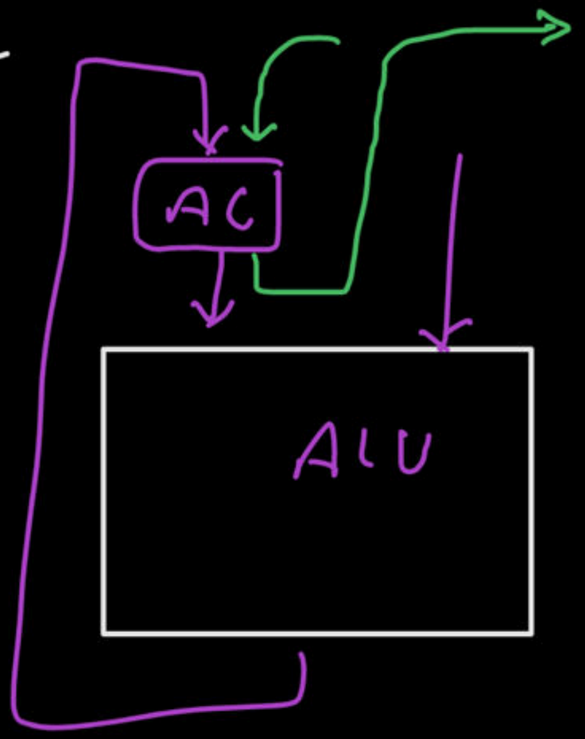
- CPU Register
  - General Purpose Registers (GPRs)
  - Special Purpose Registers
    1. Accumulator (AC)
    2. Program Counter (PC)
    3. Instruction Register (IR)
    4. Stack Pointer (SP)
    5. Flag Register / Program Status Word (PSW) / status Reg.
    6. Address Register (AR) / Memory Address Register (MAR)
    7. Data Register (DR) / Memory Data Register (MDR) / MBR

memory buffer Reg.

# Accumulator

Used to store result of ALU and sometimes ~~on~~ one of the operand for ALU too.



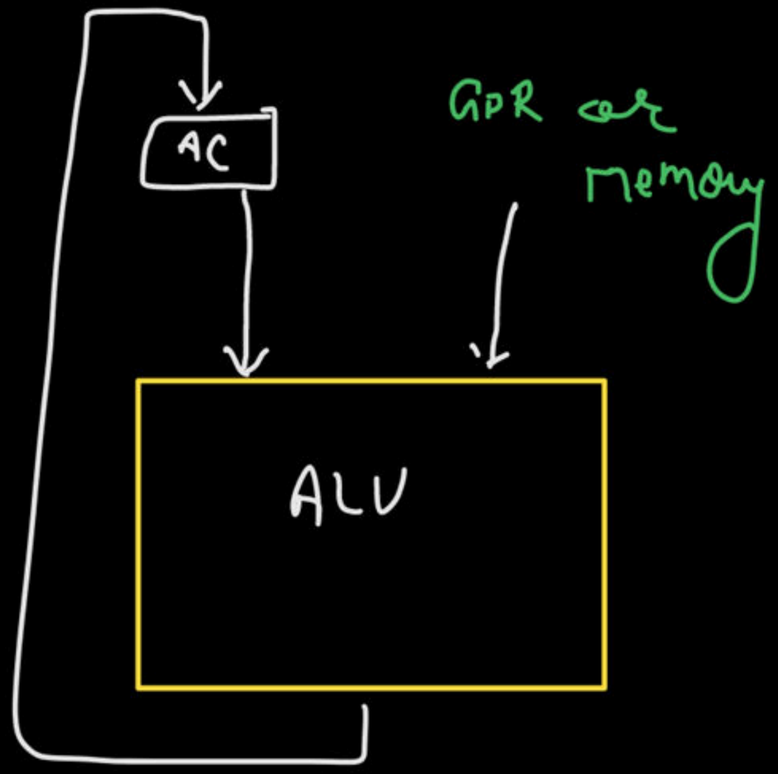



# Types of Architecture

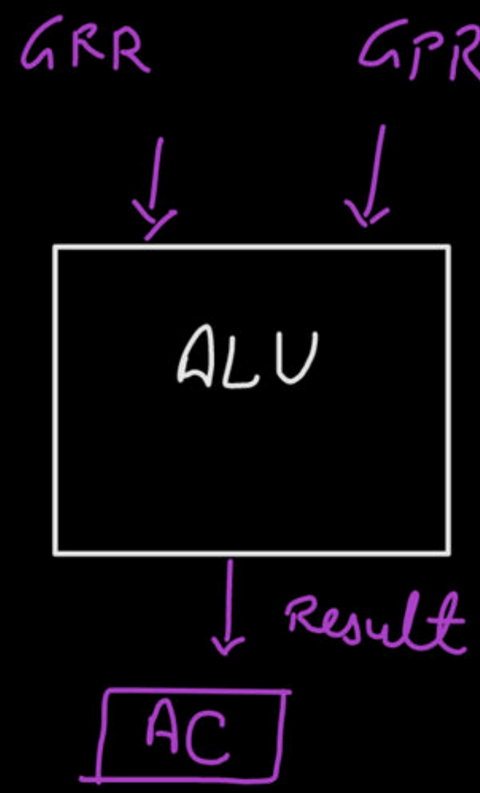
**Based on ALU input:** from where ALU takes 2 inputs

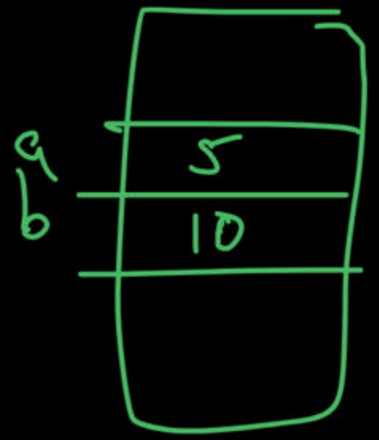
1. AC-Based Architecture
2. Register Based Architecture
3. Register-Memory Based Architecture
4. Complex System Architecture
5. Stack Based Architecture

# AC-Based Architecture



# Register-Based Architecture



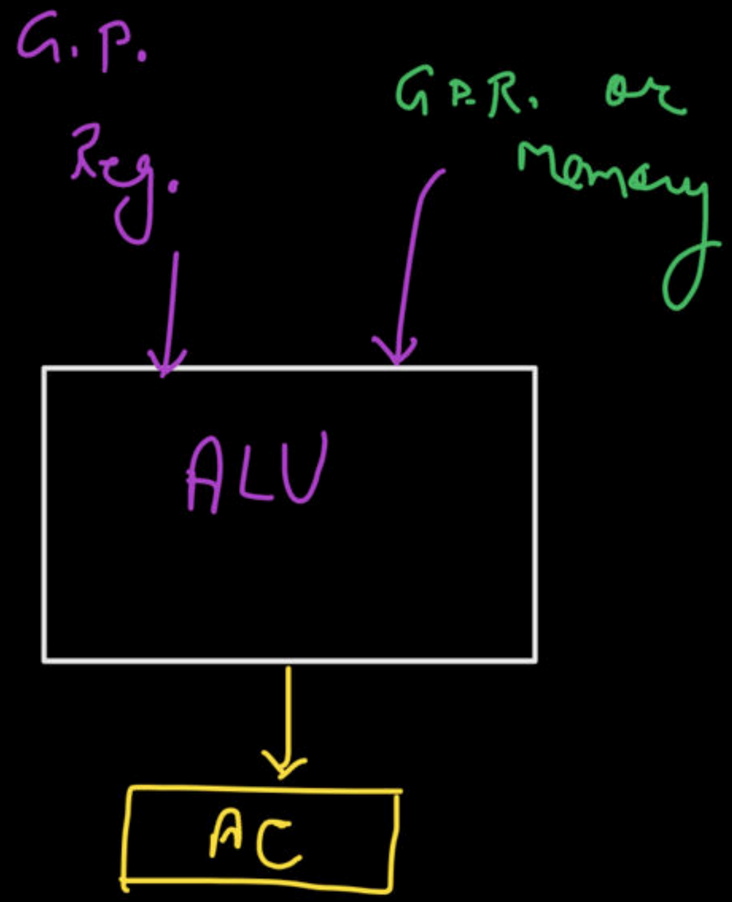

a+b

R1 = a

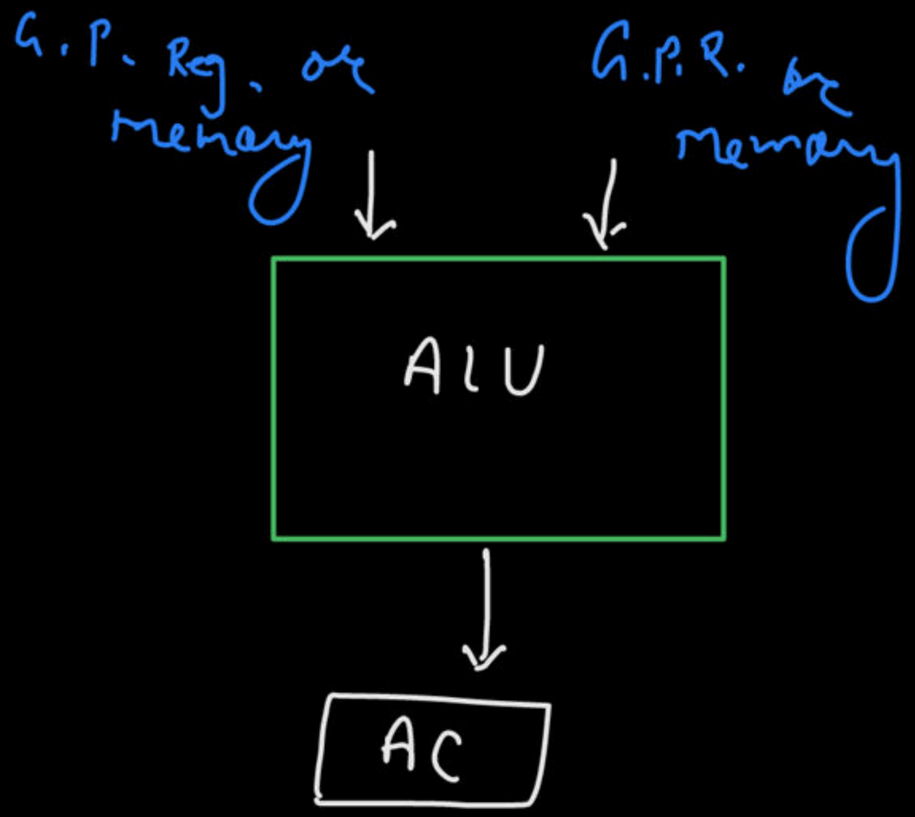
R2 = b

R1+R2

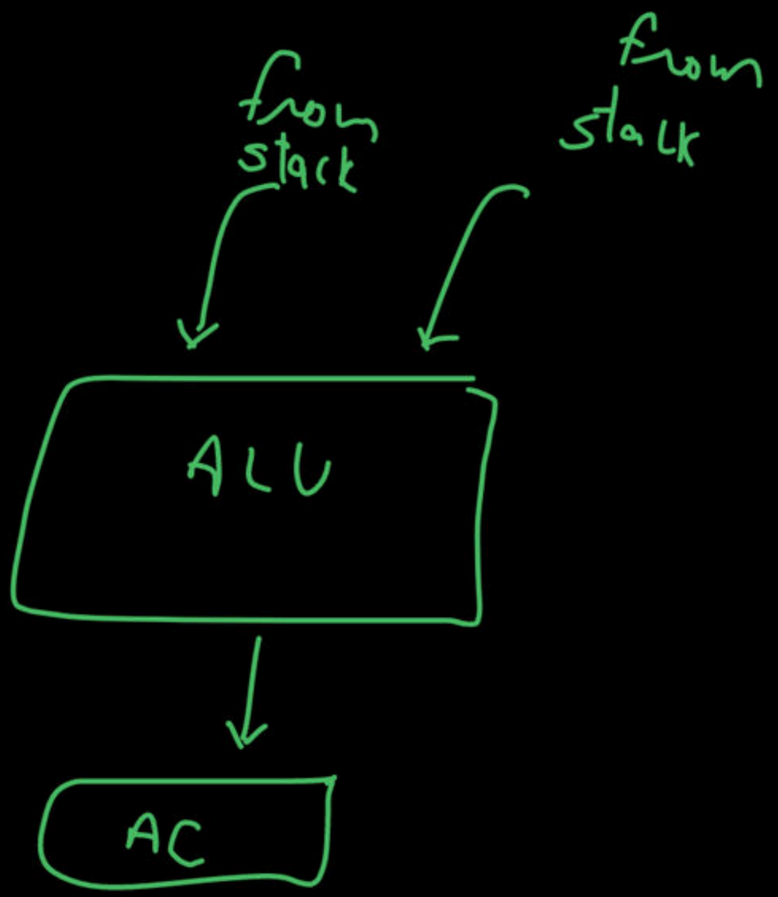
# Register-Memory Based Architecture



# Complex System Architecture

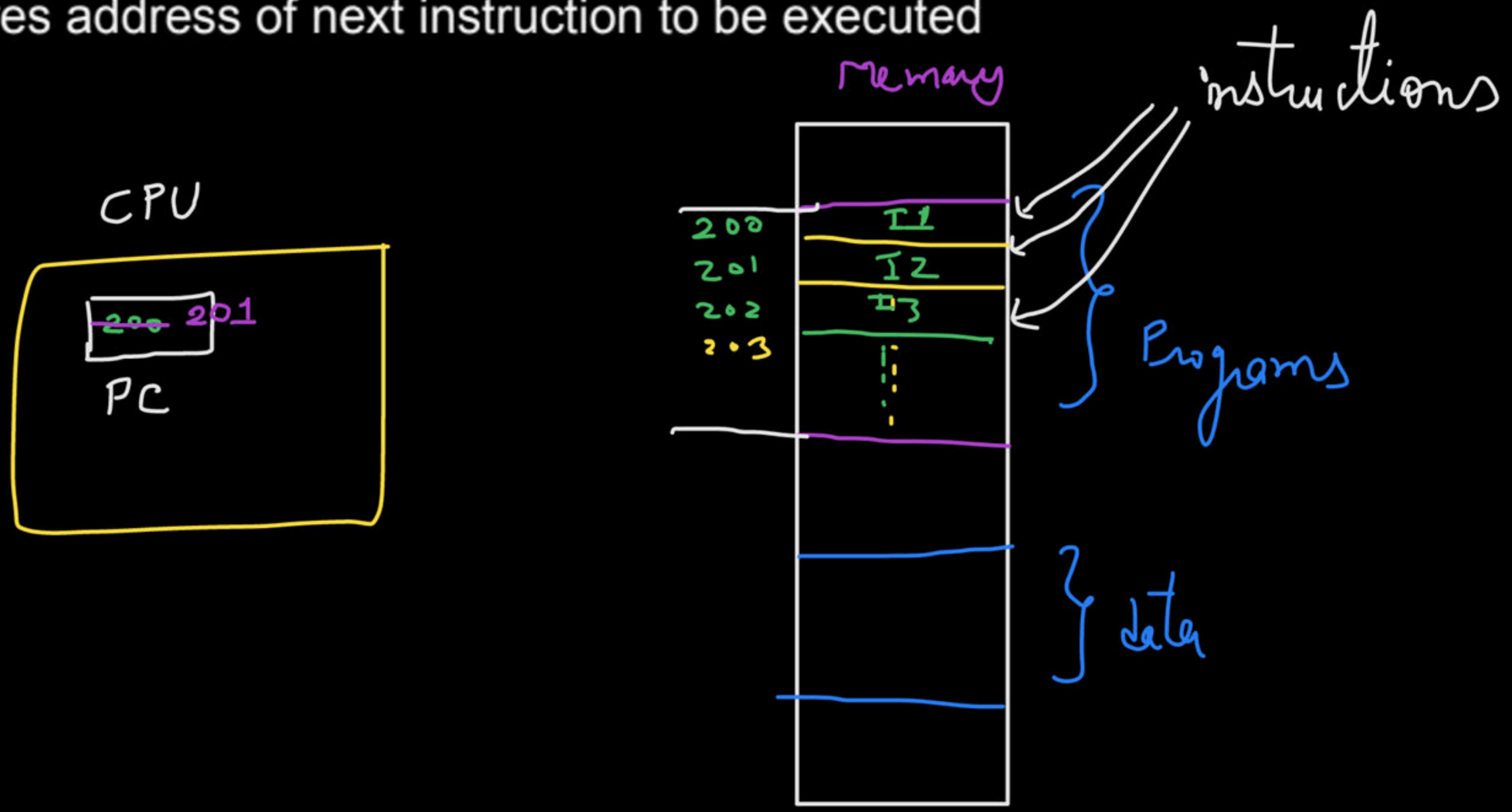


# Stack-Based Architecture



# Program Counter

Stores address of next instruction to be executed



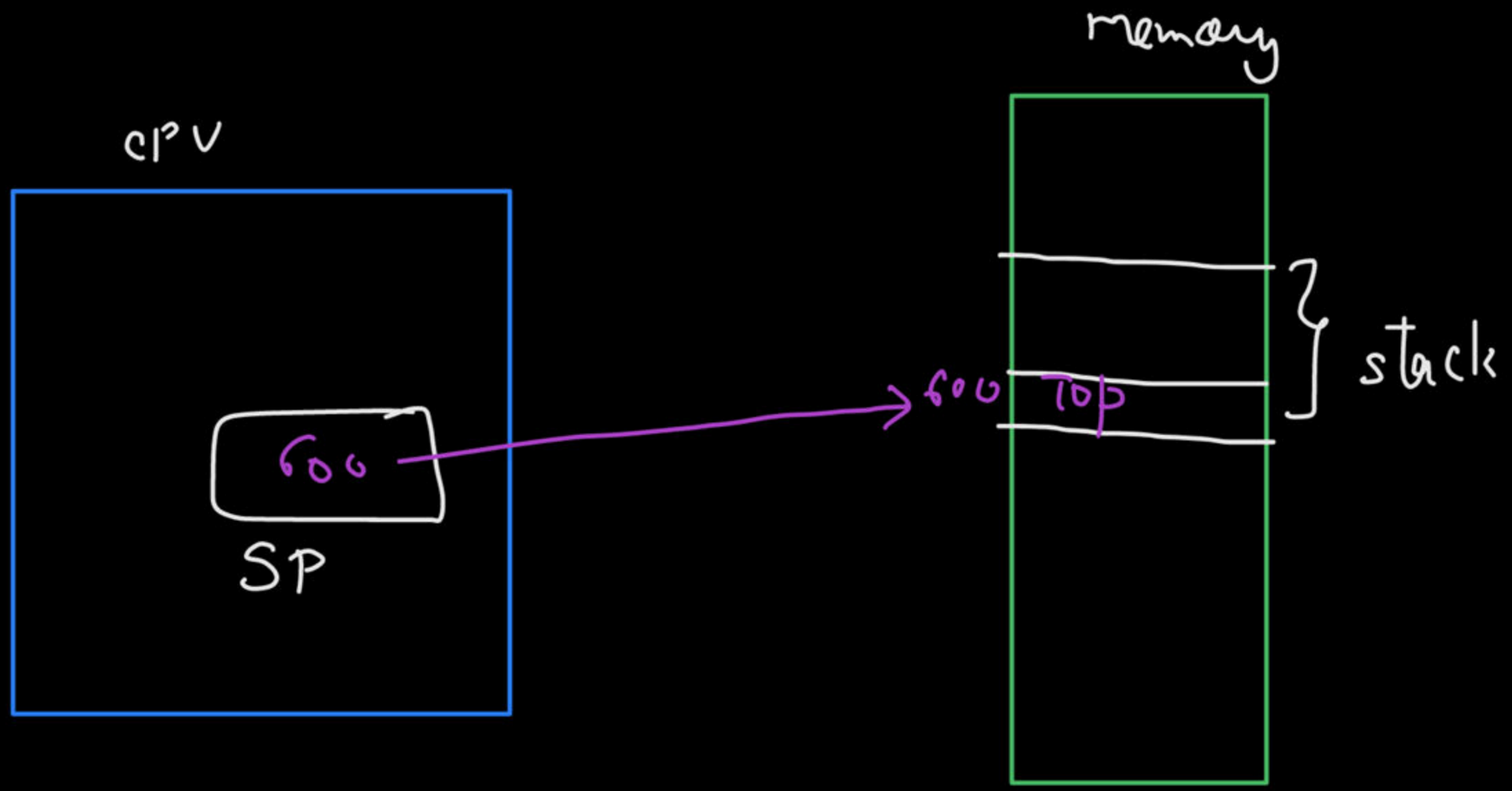
# Instruction Register

Stores the current instruction to be executed

# Stack Pointer

Stores the address of the top of the stack

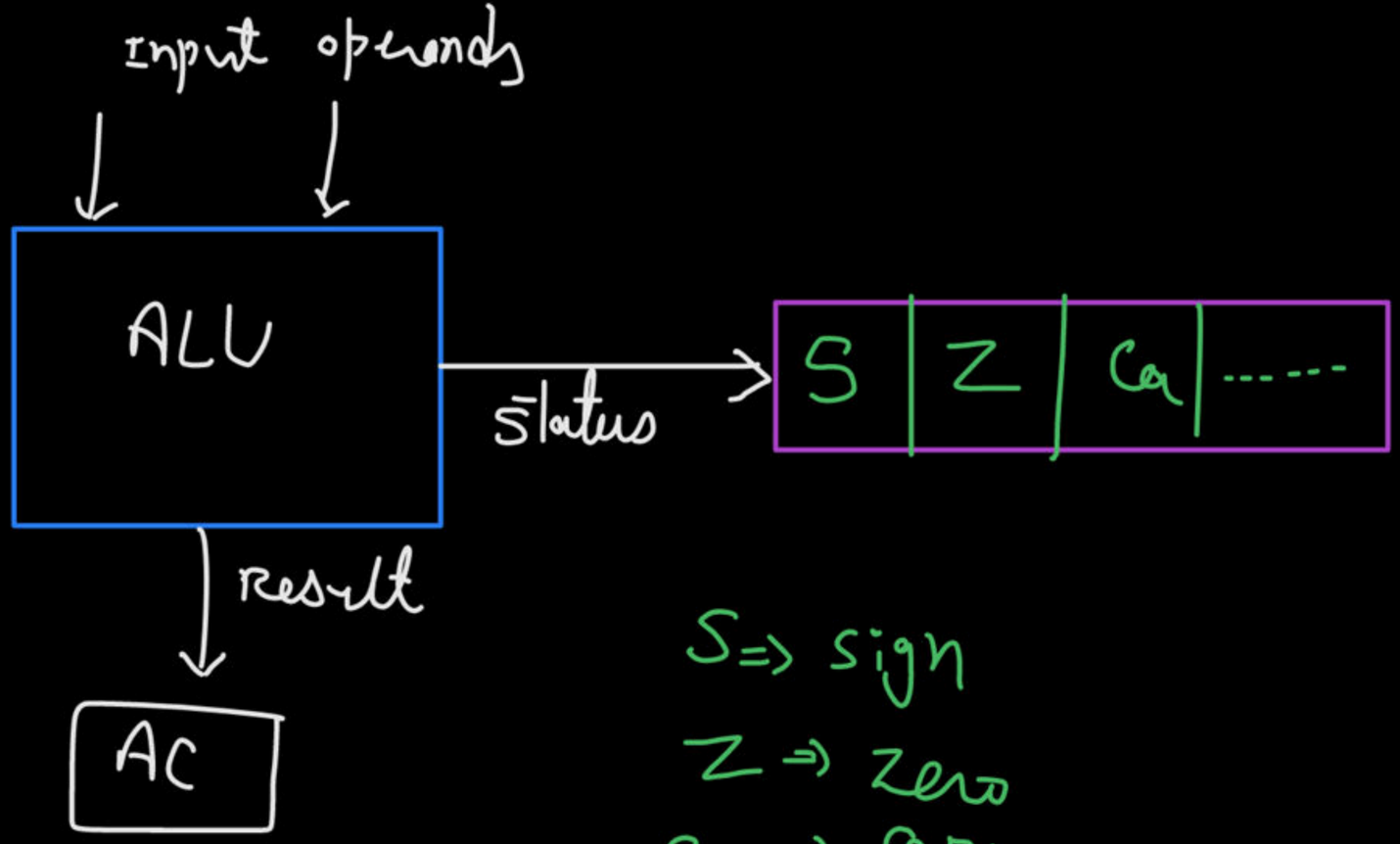
top of stack



# Flag or Status Register

Stores the status of the ALU result

↳ for implementations of conditions




S => sign

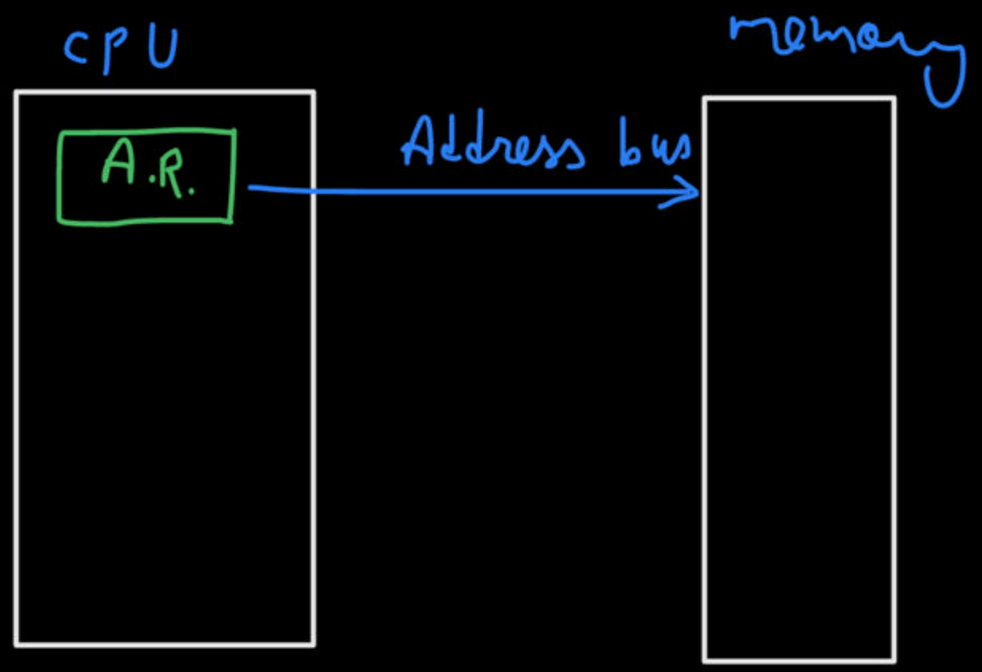
Z => zero

Ca => carry

```
if (a > b)
if (a == b)
if (a == 0)
```

# Address Register or MAR

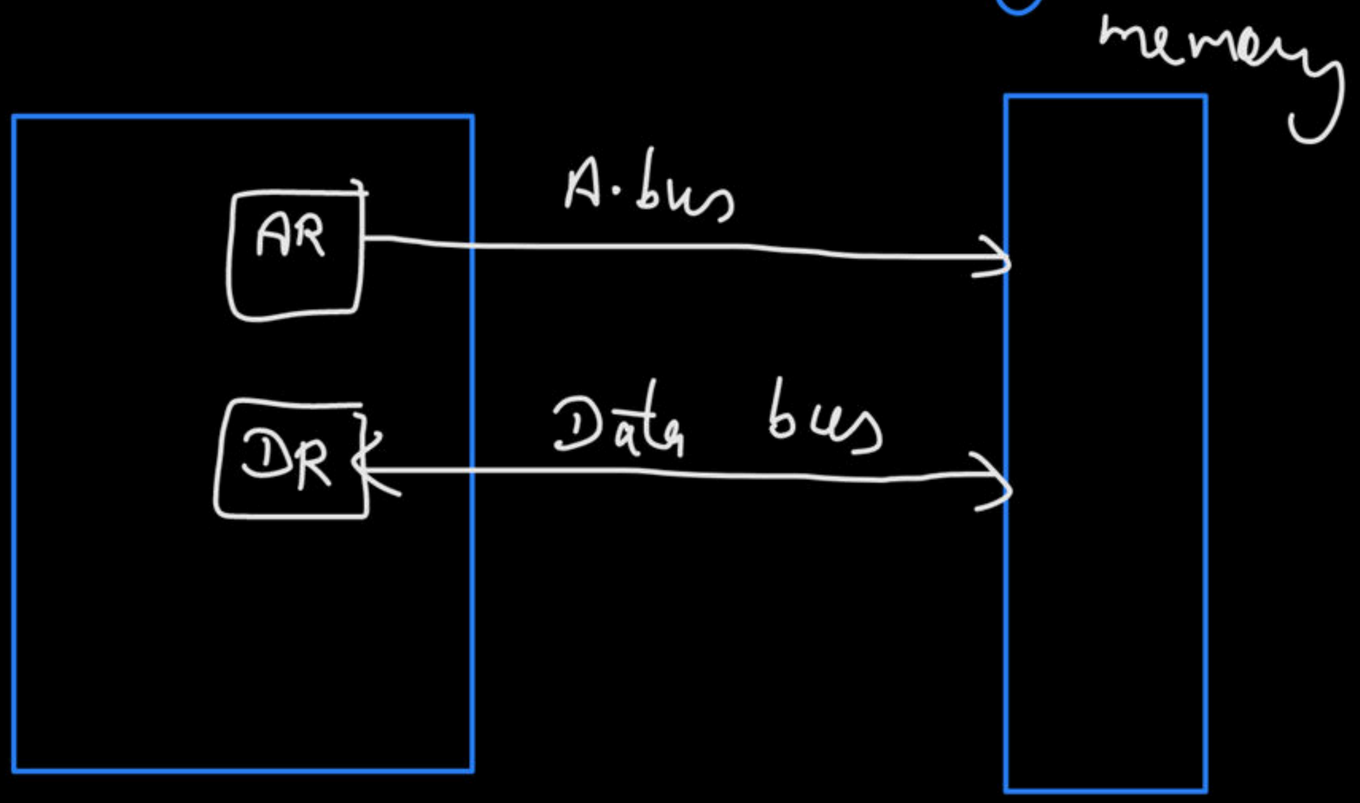
Used to send address to memory



# Data Register or MDR

Used to send data to memory (memory write)

And to receive data from memory (memory Read)

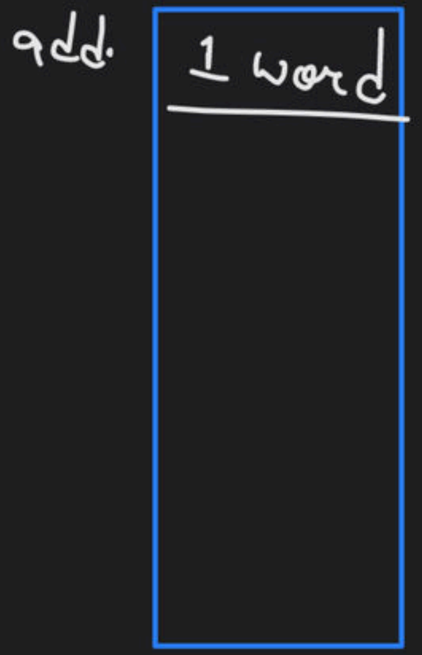


# Memory types

## Byte addressable (default)


1 byte = 8 bits

## Word addressable




1 word =

1 bit, 2 bits, 4 bits,

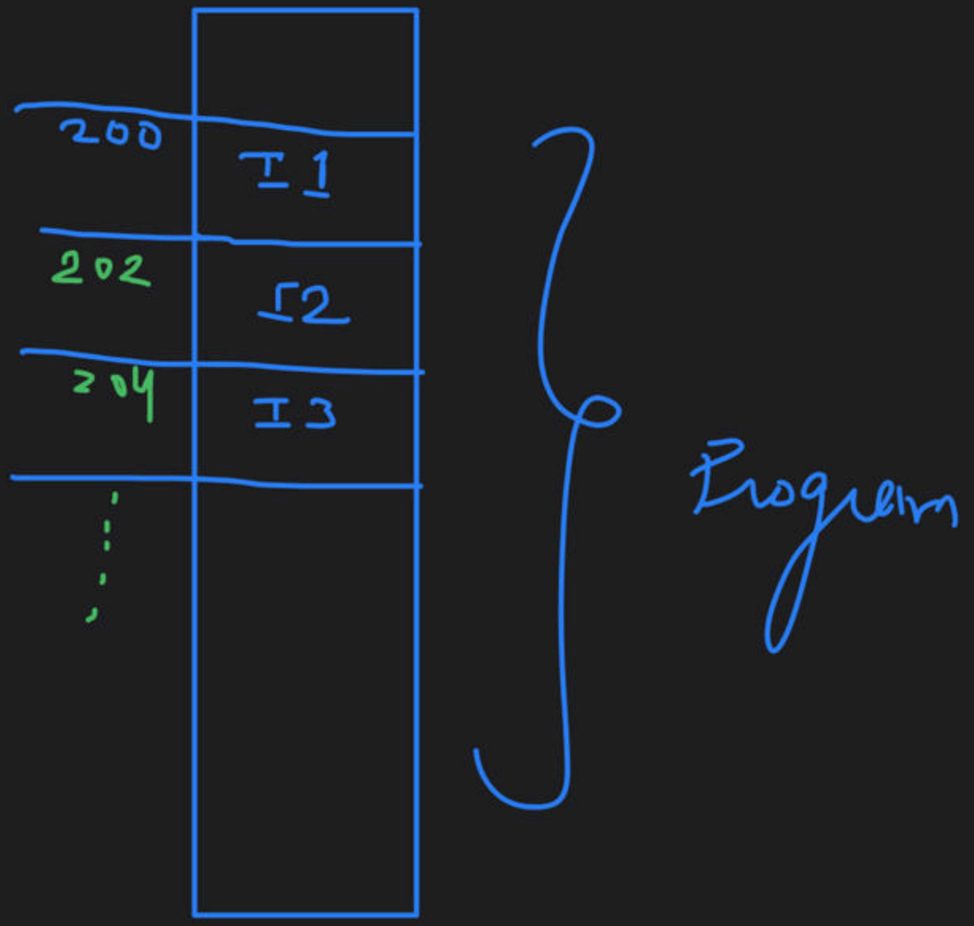
1 byte, 2 byte,

4 byte, 8 byte ....

→ memory byte addressable

→ 1 instⁿ = 2 bytes

PC = ~~200~~ ~~202~~ ~~204~~ 206



# Question

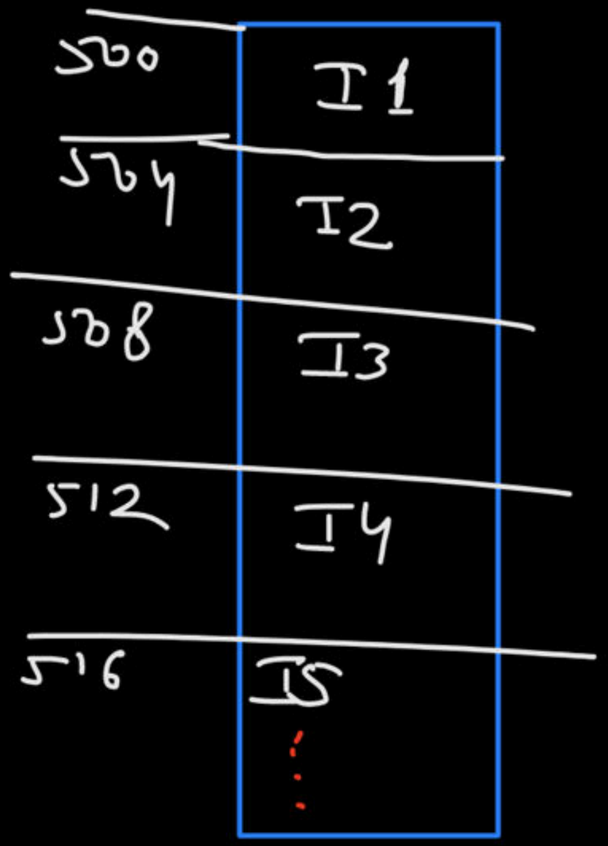
A CPU has 4 bytes instructions. A program (Instructions I1 to I200) starts at address 500 (in decimal). Find the address of following instructions:

1. I1 ⇒ 500
2. I5 ⇒ 516
3. I120 ⇒ 976

500 + 4(120−1)

= 500 + 476

= 976



# Question

A CPU has 4 bytes instructions. A program (Instructions I1 to I200) starts at address 500 (in decimal). What should be the PC value when instruction I6 will be executing in CPU?

if I6 is in execution then add. of I7 will be in PC

$= 500 + 4 * (7 - 1)$

$= 524$

# Question

A CPU has 4 bytes instructions. A program (Instructions I1 to I200) starts at address 500 (in decimal). What should be the PC value when instruction $i$ will be executing in CPU?

if instⁿ $i$ is in execution then, add. of $(i+1)$ will be in PC

$= 500 + 4 * (i + 1 - 1)$

$= 500 + 4i$

# Happy Learning.!