

CSC-110

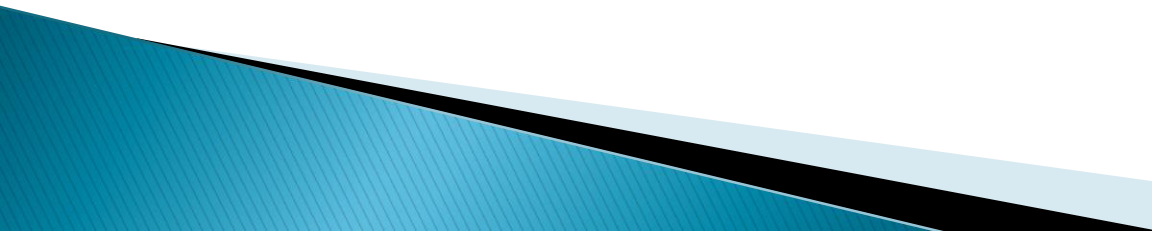
Computing FUNDAMENTALS

# THE COMPUTER SYSTEM HARDWARE

ENGR.MAHAWISH



# Computer Registers

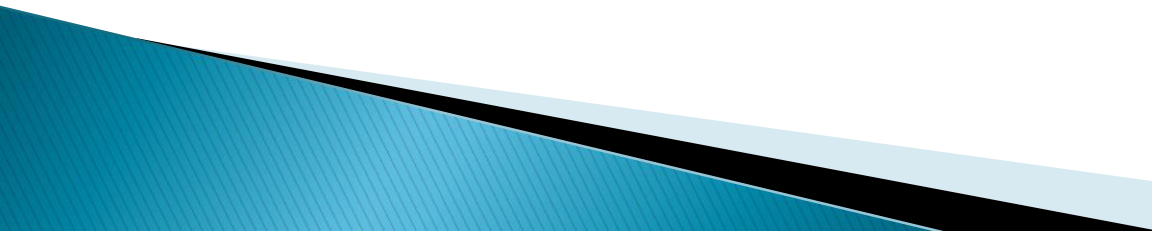
- ▶ Registers are a type of computer memory used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU. The registers used by the CPU are often termed as Processor registers.
  - ▶ A processor register may hold an instruction, a storage address, or any data (such as bit sequence or individual characters).
  - ▶ The computer needs processor registers for manipulating data and a register for holding a memory address. The register holding the memory location is used to calculate the address of the next instruction after the execution of the current instruction is completed.
- 

# Cont.

- ▶ Following is the list of some of the most common registers used in a basic computer:

Register	Symbol	Number of bits	Function
Data register	DR	16	Holds memory operand
Address register	AR	12	Holds address for the memory
Accumulator	AC	16	Processor register
Instruction register	IR	16	Holds instruction code
Program counter	PC	12	Holds address of the instruction
Temporary register	TR	16	Holds temporary data
Input register	INPR	8	Carries input character
Output register	OUTR	8	Carries output character

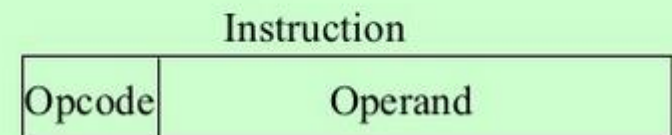
# Addressing modes

- ▶ The term *addressing modes* refers to the way in which the operand of an instruction is taken from memory or register.
  - ▶ Most CPUs have at least four addressing modes:
    - ▶ Immediate
    - ▶ Direct
    - ▶ Indirect
    - ▶ Indexed
- 

# Immediate

- ▶ Immediate Addressing • Operand is part of instruction •
- ▶ Operand = address field •
- ▶ e.g. **ADD 5** —Add 5 to contents of accumulator—5 is operand •
- ▶
- ▶ No memory reference to fetch data •
- ▶ Fast •
- ▶ Limited range

## Immediate Addressing Diagram



# Example

The following examples use the following piece of computer memory.

Location	0	1	2	3	4	5	6
Value				17		3	

**Immediate addressing supplies the actual value and is normally prefixed with a #.**

LD Acc, #5 ;Load the actual value 5 into the accumulator

Ax becomes 5.

# Register Mode

- ▶ In this mode the operand is stored in the register and this register is present in CPU. The instruction has the address of the Register where the operand is stored.

## **Advantages**

Shorter instructions and faster instruction fetch. Faster memory access to the operand(s)

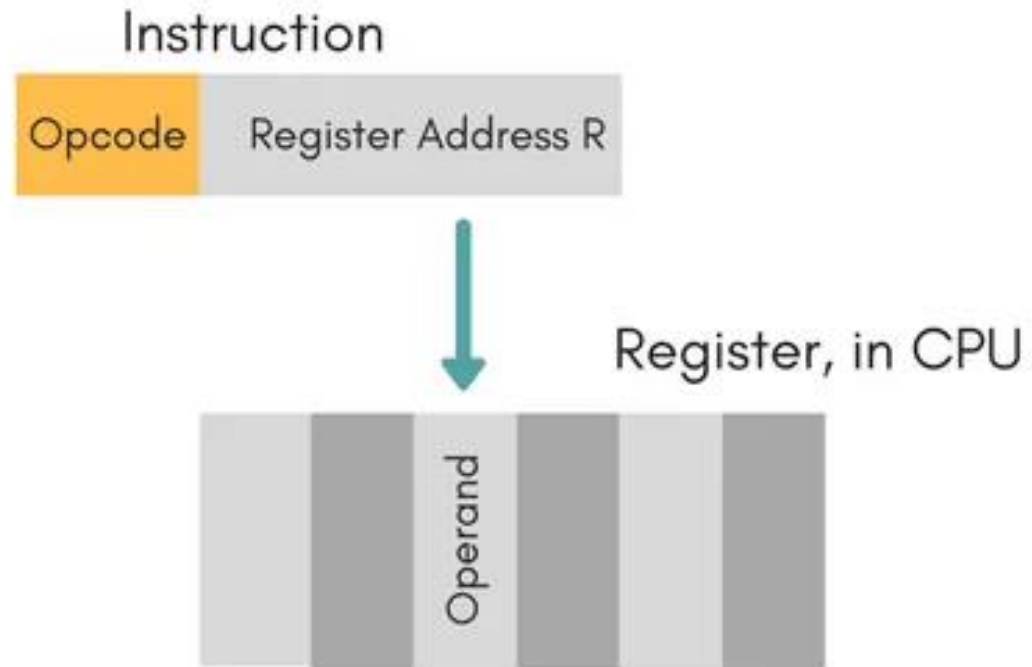
## **Disadvantages**

Very limited address space

Using multiple registers helps performance but it complicates the instructions.

# CONT.

Example: MOV AX,CX (move the contents of CX register to AX register)





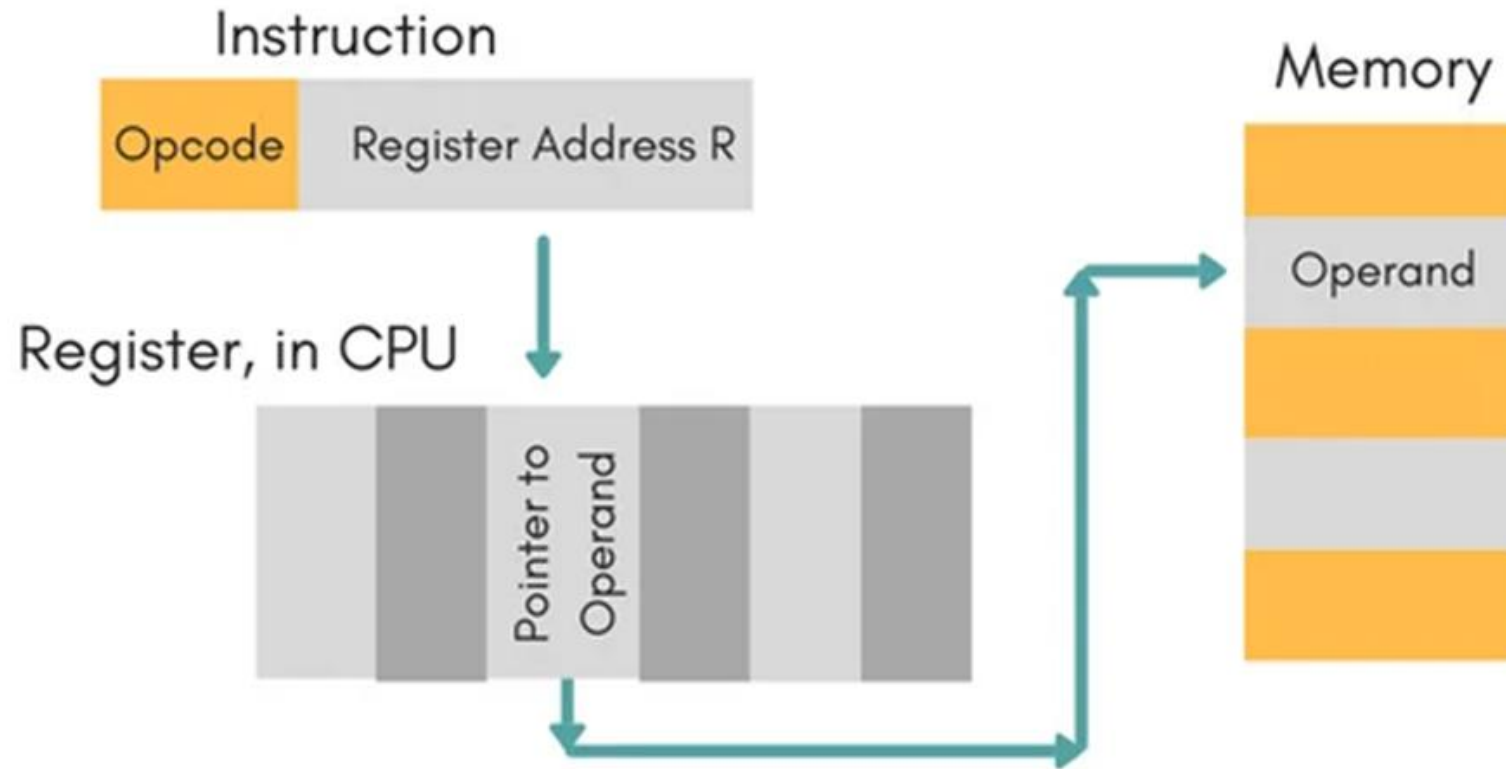
# Register Indirect Mode

- ▶ In this mode, the instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather than the operand itself.

`MOV AX, [BX]`

(move the contents of memory location addressed by the register BX to the register AX)

# CONT.



# Direct Addressing Mode

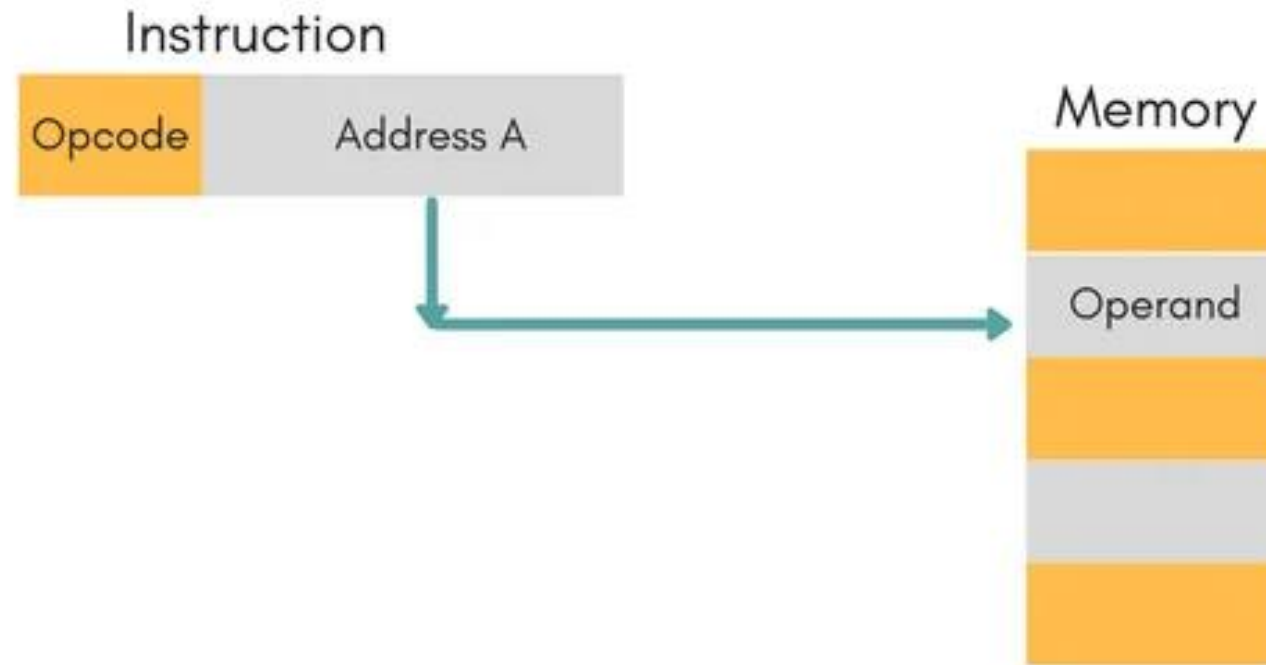
- ▶ In this mode, effective address of operand is present in instruction itself.
- ▶ Single memory reference to access data.
- ▶ No additional calculations to find the effective address of the operand.

**For Example:** **ADD R1, 4000** - In this the 4000 is effective address of operand.

**NOTE:** *Effective Address is the location where operand is present.*

# CONT.

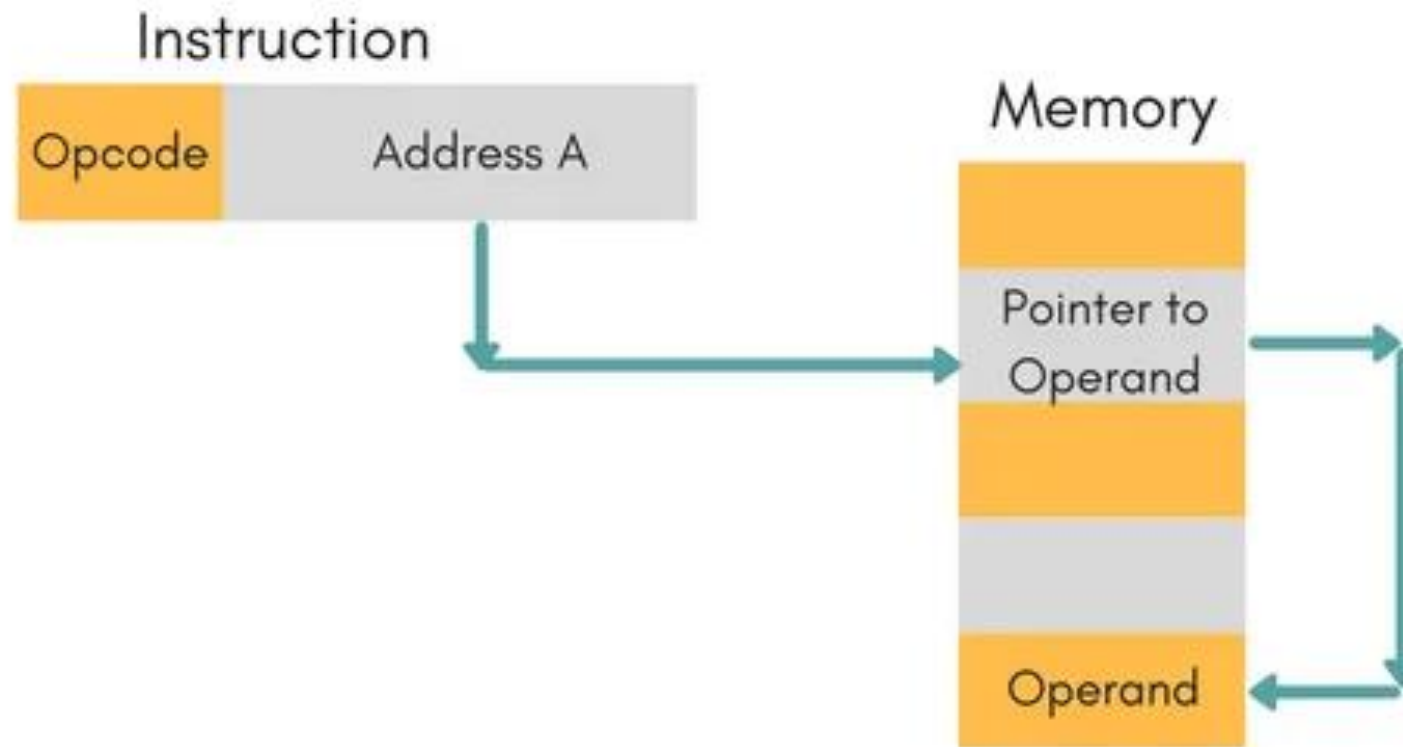
Example: `ADD AL,[0301]` //add the contents of offset address 0301 to AL



# Indirect Addressing Mode

- ▶ In this, the address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as this includes multiple memory lookups to find the operand.

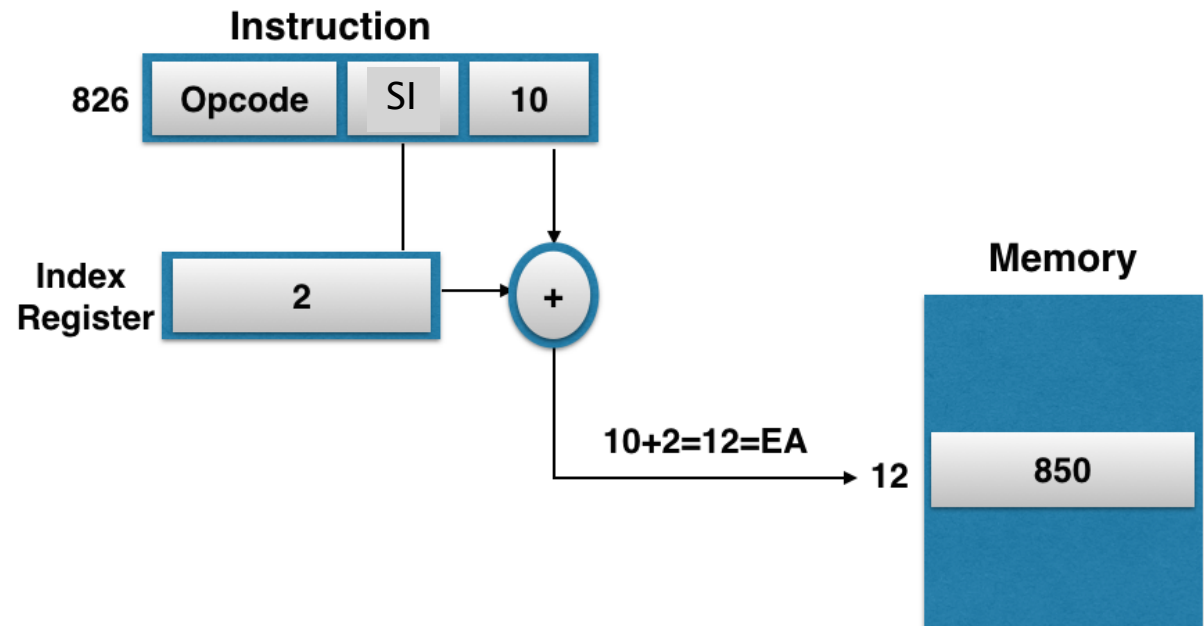
# CONT.



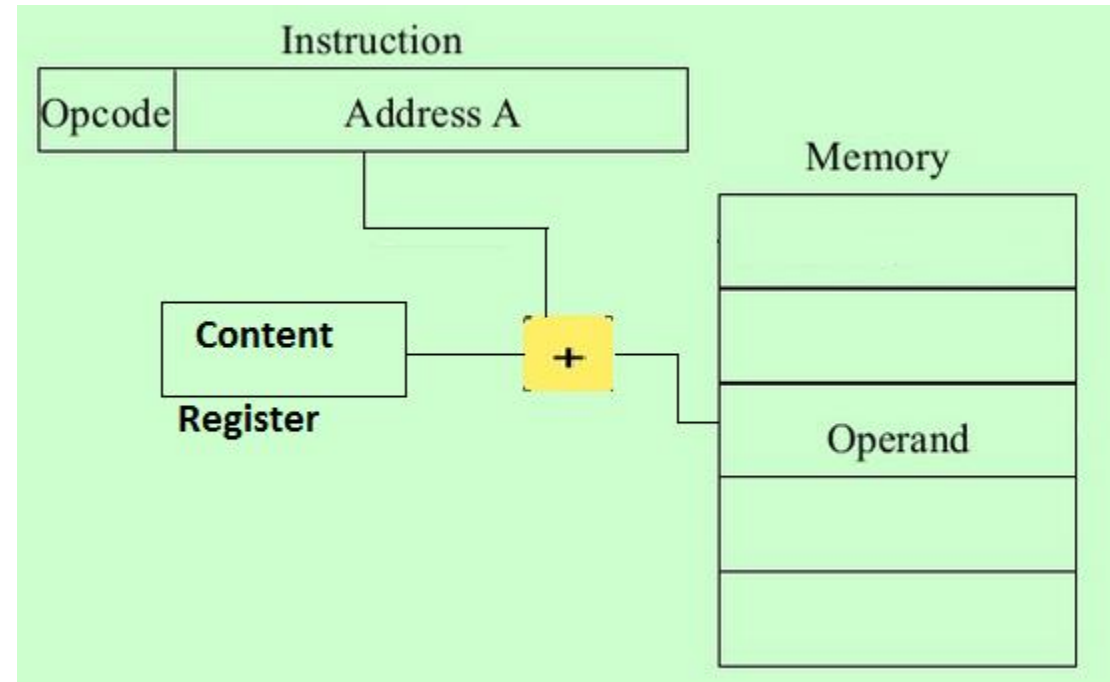
# Index Addressing

In this mode the content of an index register is added to the address part of the instruction to obtain the effective address.

Example: MOV AX, [SI + 10]



# CONT.





# EXAMPLE

**Immediate mode**

```
load r2, #800
```

**Direct**

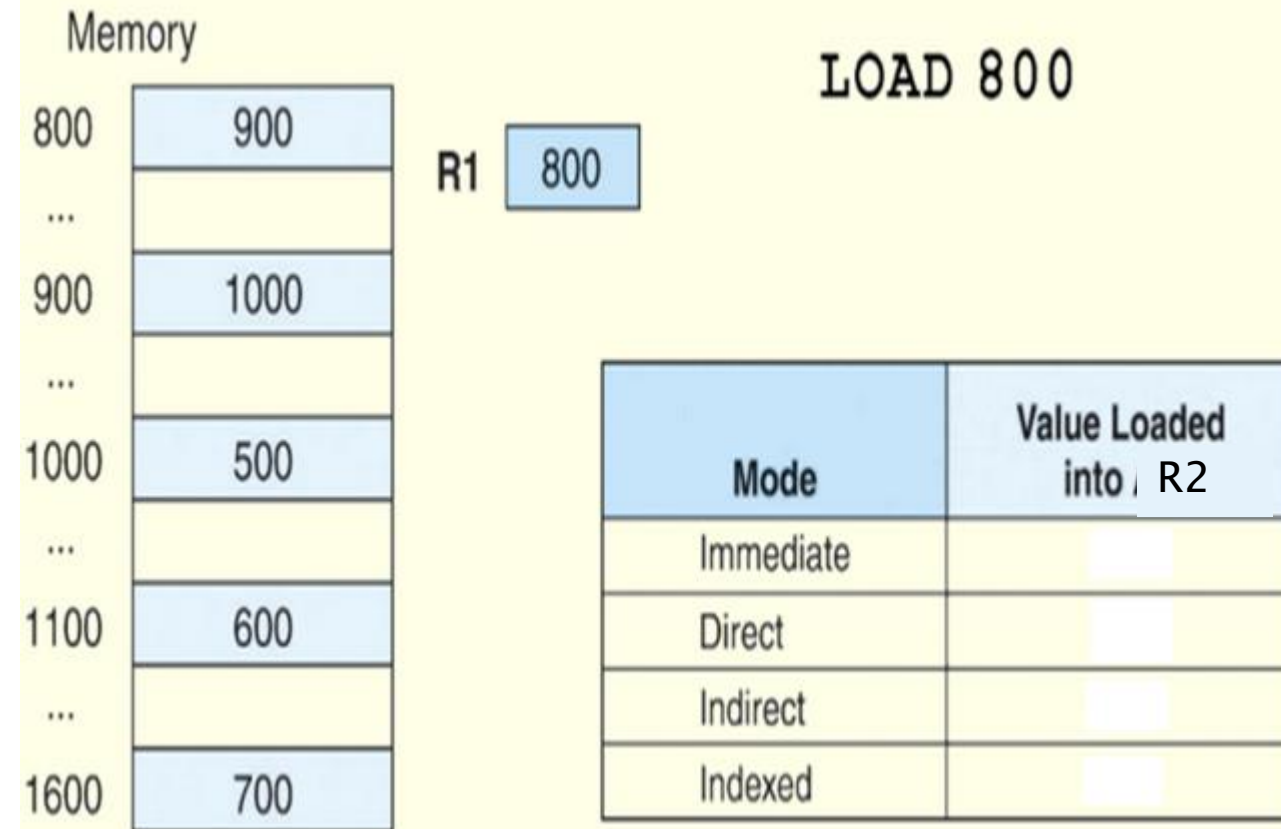
```
load r2, 800
```

**Indirect**

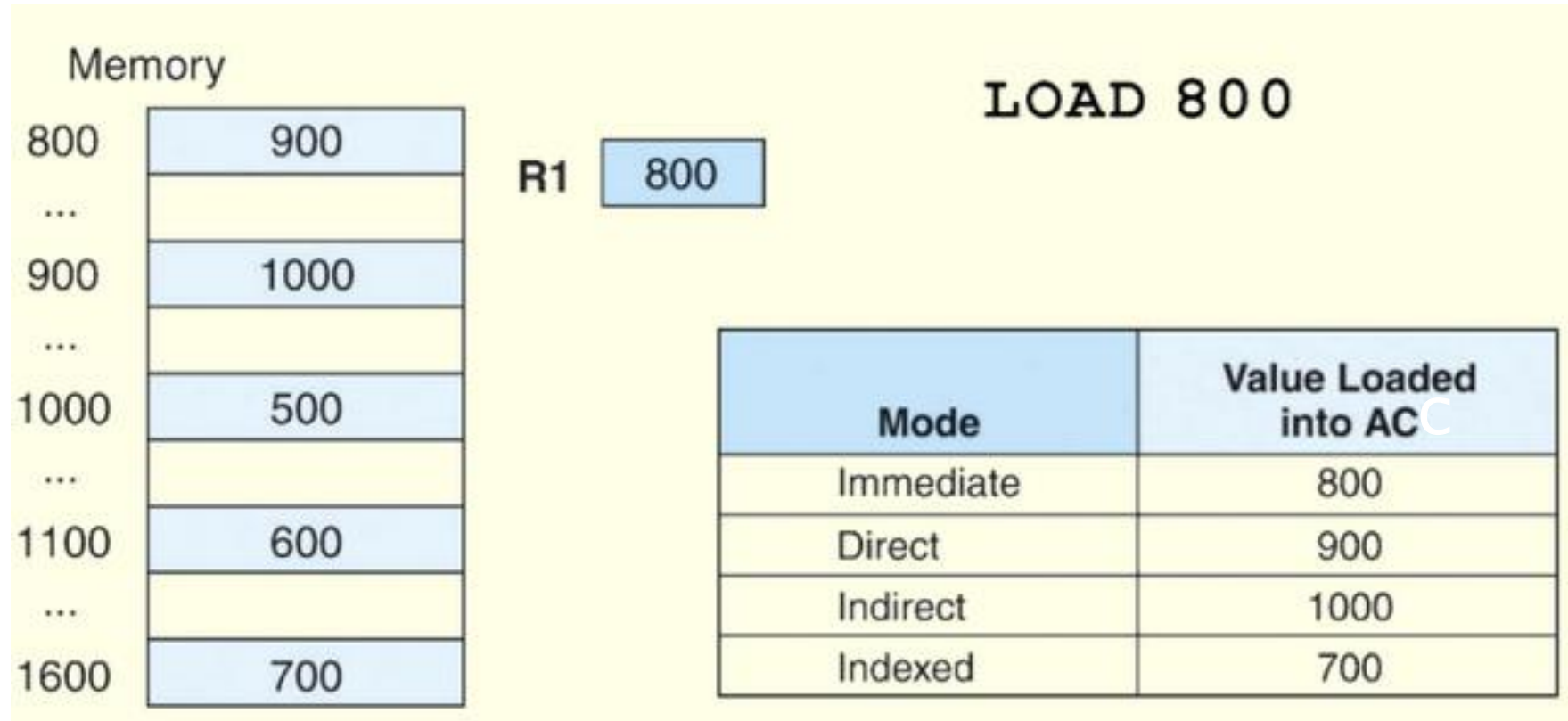
```
load r2, [ 800]
```

**Index**

```
load r2, (r1,#800)
```




# EXAMPLE



# System Bus Types and Functions

- ▶ The CPU moves data around the computer on pathways that interconnect it to all the other components on the motherboard. These pathways are called 'buses'.
- ▶ The internal bus carries data within the motherboard.
- ▶ External buses carry data to peripherals and other devices attached to the motherboard.


The lines or pins of a bus are of three types:

- ▶ Address Bus - the components pass memory addresses to one another over the address bus.
  - ▶ Control Bus- used to send out signals to coordinate and manage the activities of the motherboard components.
  - ▶ Data Bus- Data transferred between peripherals, memory and the CPU. Obviously, the data bus can be a very busy pathway.
- 

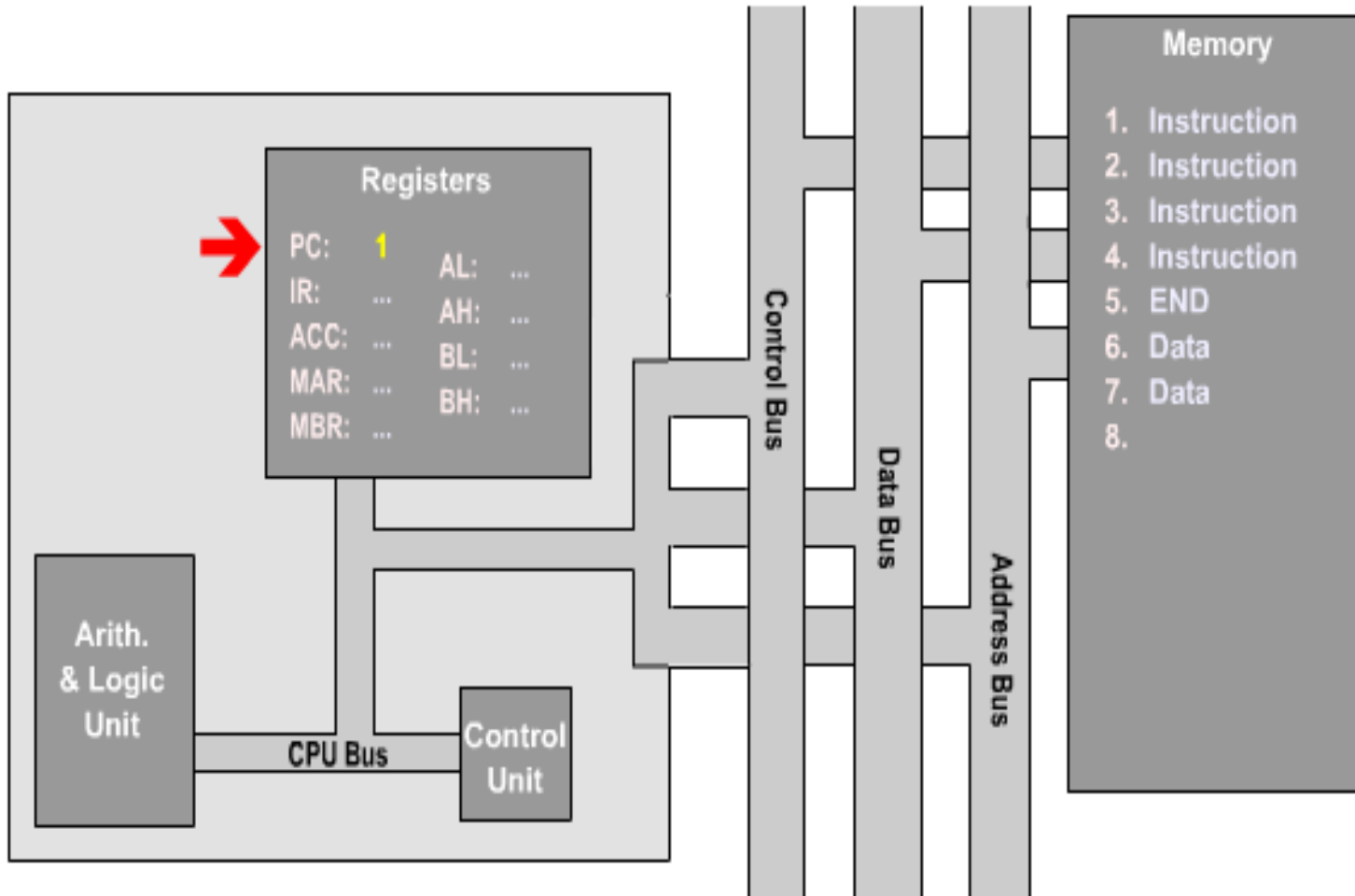
# Instruction execution cycle

- ▶ An instruction **cycle** (sometimes called a **fetch**–decode–execute **cycle**) is the basic operational process of a computer. It is the process by which a computer retrieves a program instruction from its memory, determines what actions the instruction dictates, and carries out those actions.
- ▶ This cycle is repeated continuously by a computer's central processing unit (CPU), from boot-up to when the computer is shut down.

The instruction execution cycle can be clearly divided into three different parts:

- ▶ Fetch Cycle
  - ▶ Decode Cycle
  - ▶ Execute Cycle
- 

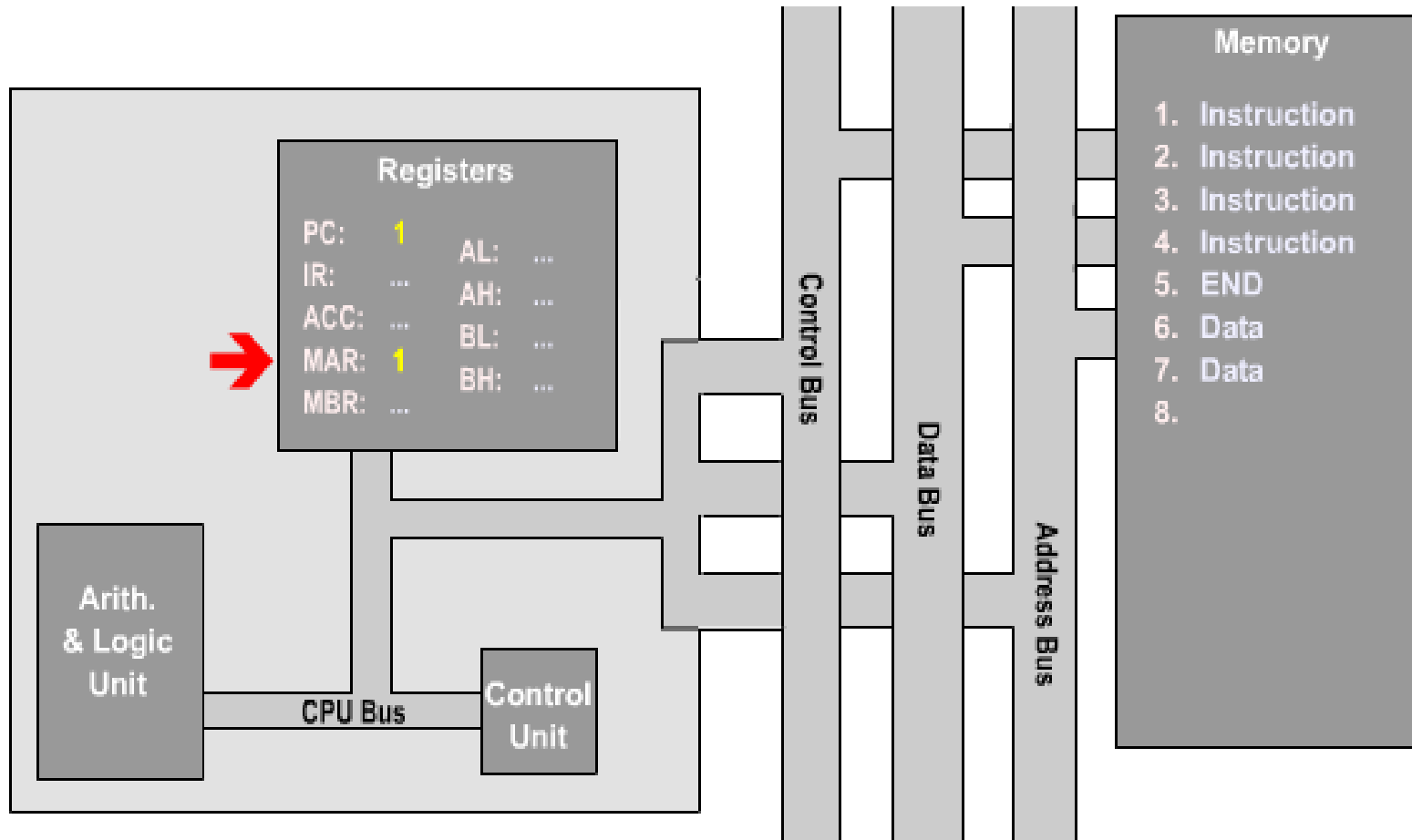
# Instruction Cycle– Fetch Cycle



## STEP 1:

- ▶ The value held in the program counter register corresponds to the address of the instruction in memory which is next to be processed.

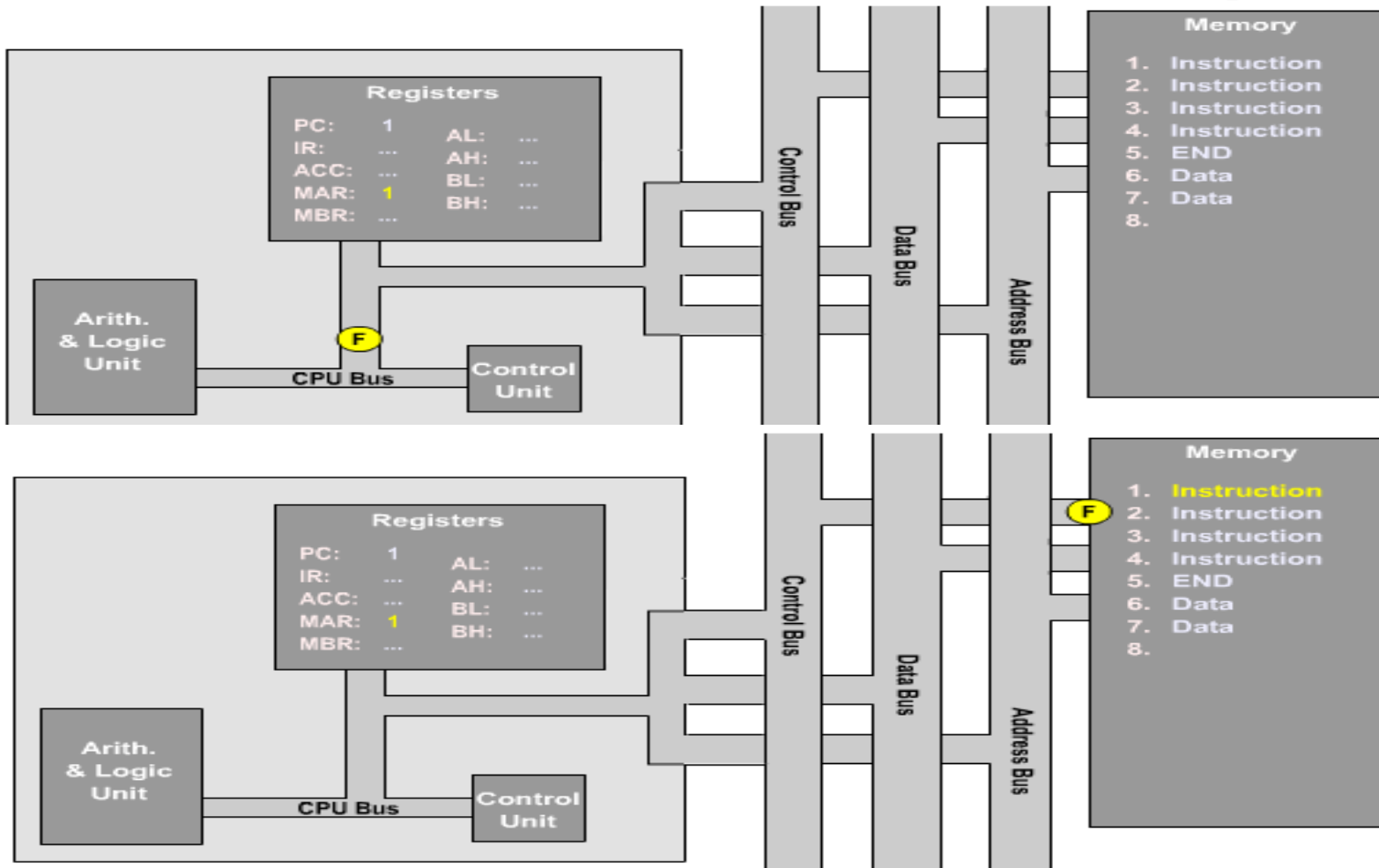
# Instruction execution Cycle– Fetch Cycle



## STEP 2:

For this value to be used, it is first transferred across to the memory address register.

# Instruction execution Cycle– Fetch Cycle

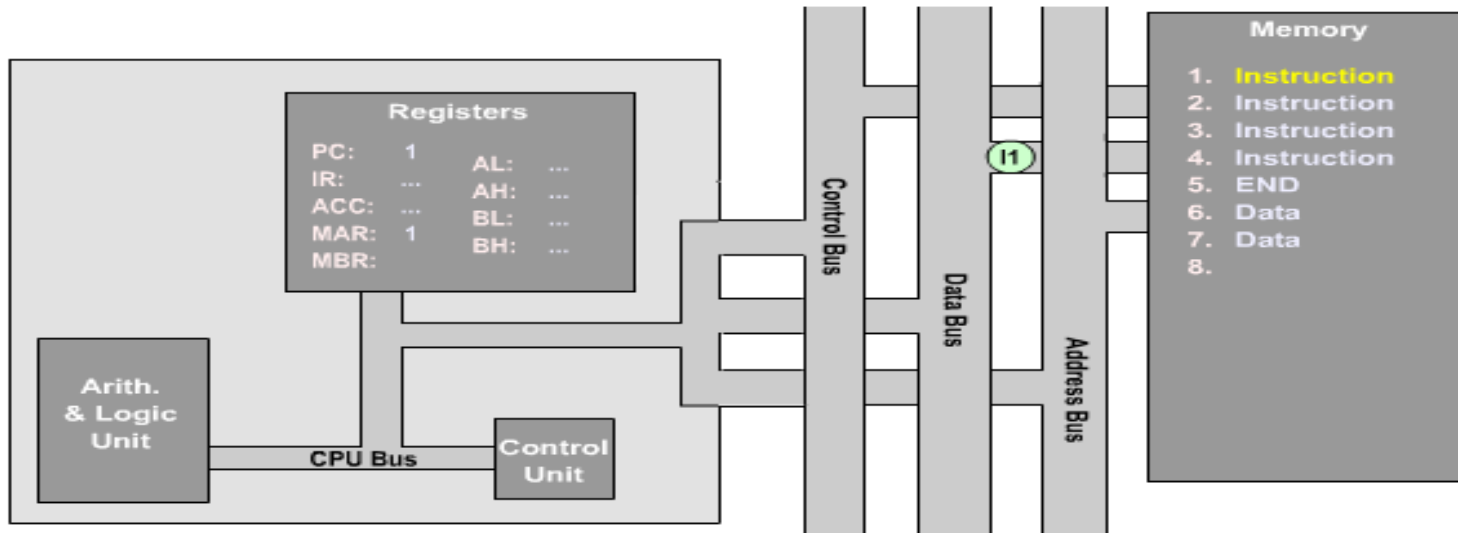


## STEP 3:

- ▶ The control unit checks the value in the memory address register, before fetching corresponding instruction from memory.

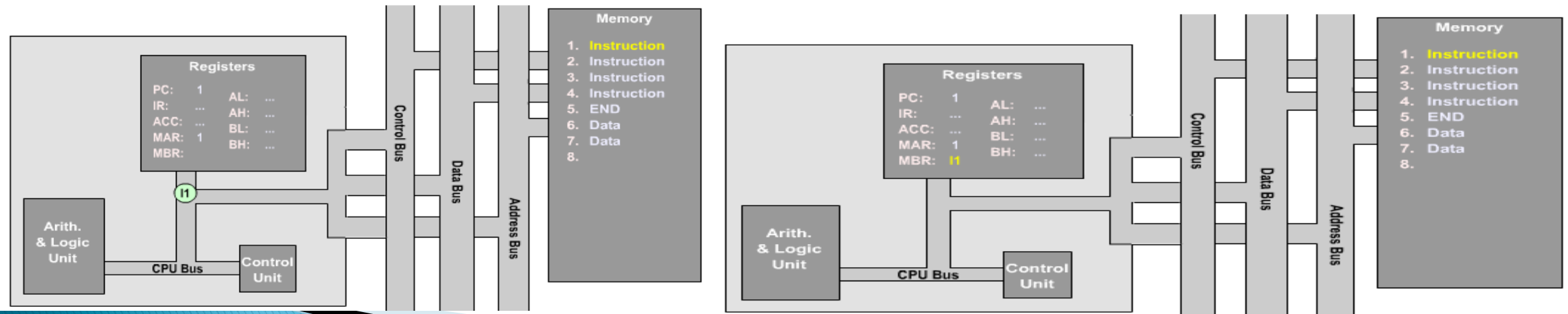


# Instruction execution Cycle– Fetch Cycle



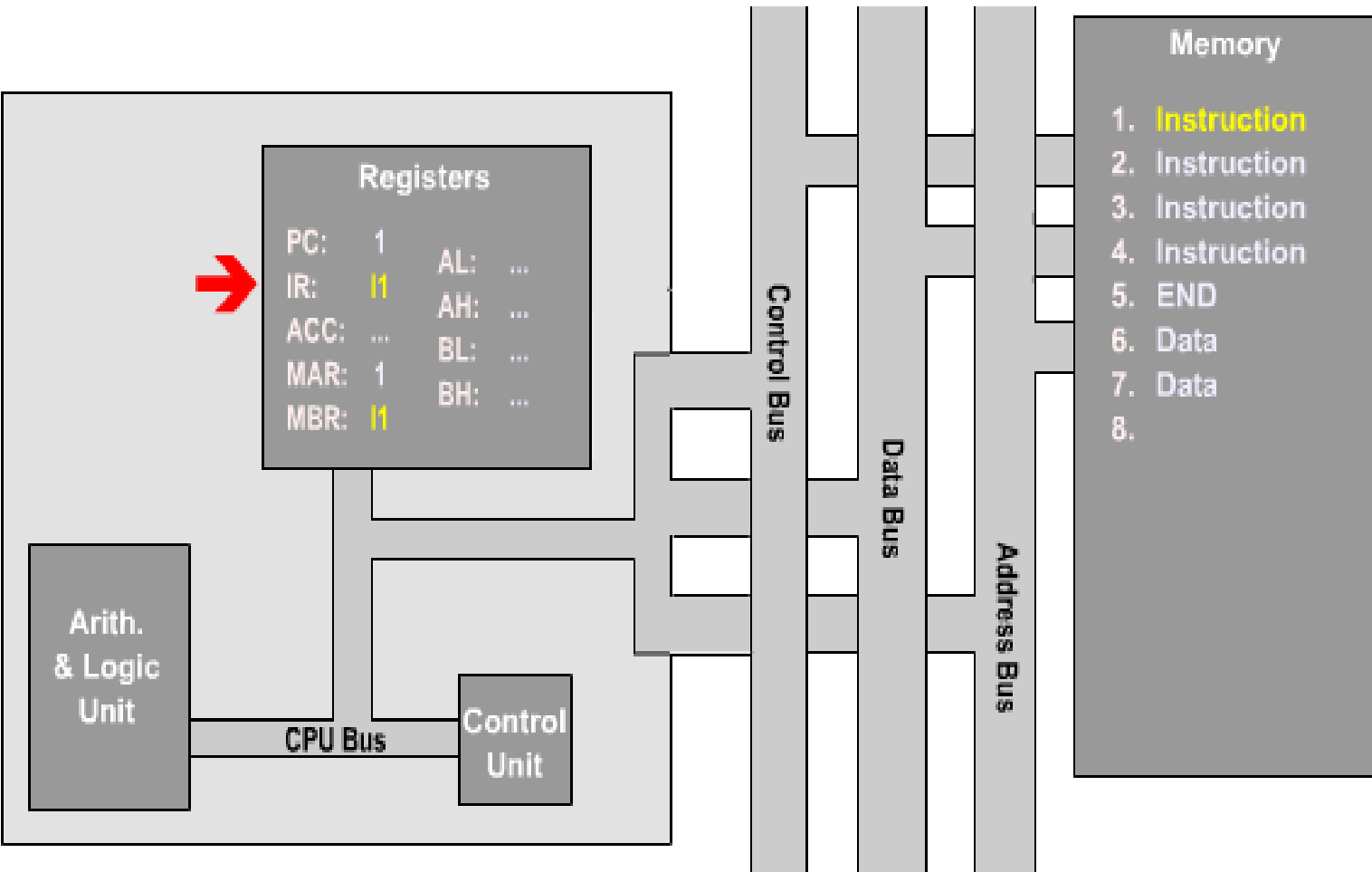
STEP 4:

- ▶ The fetched instruction is stored in the memory buffer register.





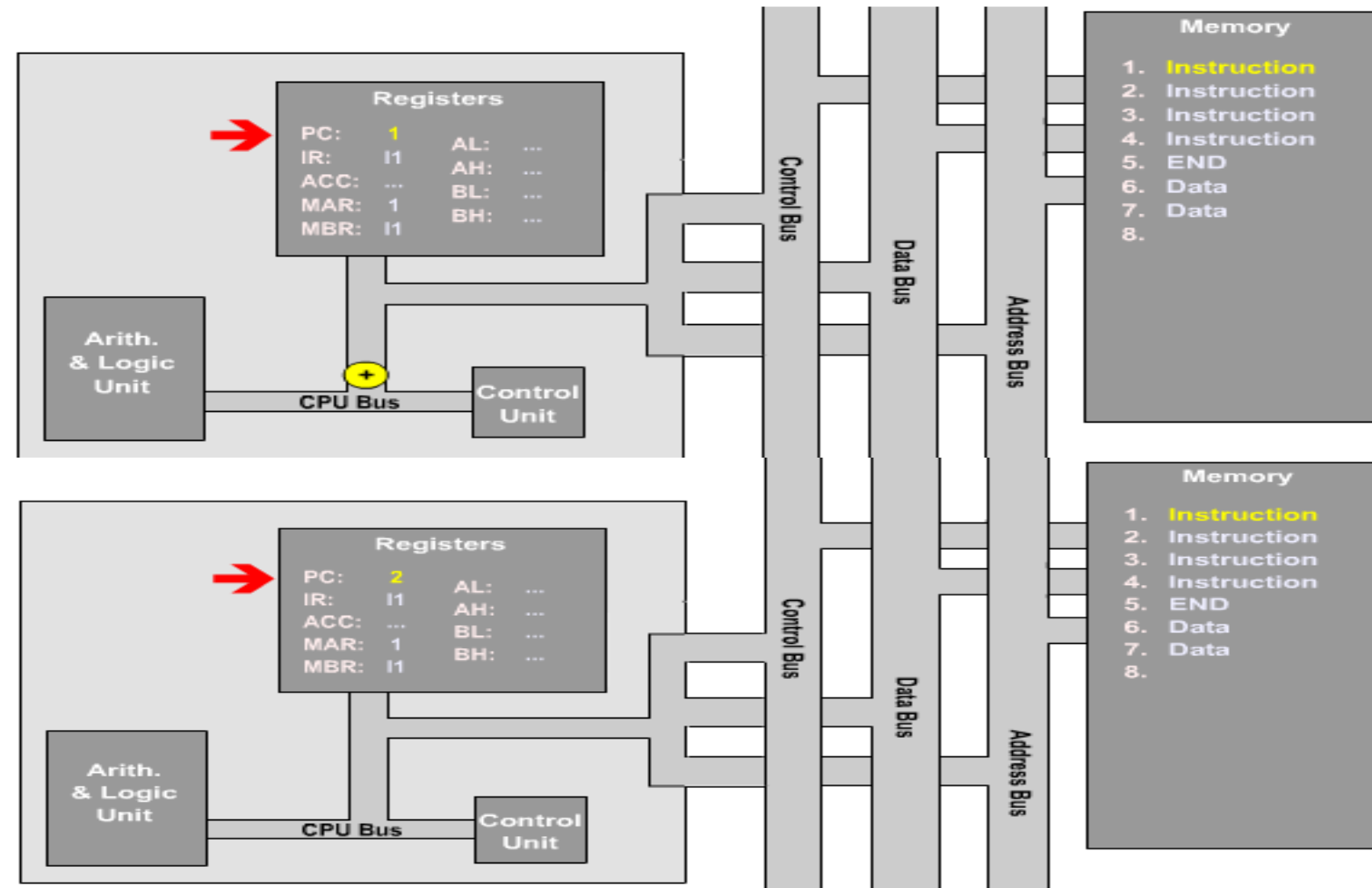
# Instruction execution Cycle– Fetch Cycle



STEP 5:

- It is then transferred to the instruction register.

# Instruction execution Cycle– Fetch Cycle



## STEP 6:

- ▶ The program counter register is advanced by the control unit, so that it corresponds to the memory location of the next instruction in the program being executed.

# Instruction execution Cycle– decode Cycle and execute cycle

## STEP 7:

- ▶ During this cycle the encoded instruction present in the IR (instruction register) is interpreted by the decoder. . It determines which opcode and addressing mode have been used.

## STEP 8:

- ▶ The control unit of the CPU passes the decoded information as a sequence of control signals to the relevant function units of the CPU to perform the actions required by the instruction such as reading values from registers, passing them to the ALU to perform mathematical or logic functions on them, and writing the result back to a register.
- 