CS & IT ENGINEERING

COMPUTER ORGANIZATION AND ARCHITECTURE

Instruction & Addressing Modes

Lecture No.- 03

By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic Instructions

Topic Variable Length Instructions

Topic Register Spill

Topics to be Covered

Topic Effective Address

Topic Addressing Modes

[NAT] GATE-2018

#Q. A processor has 16 integer registers (R0, R1, … , R15) and 64 floating point registers (F0, F1, … , F63). It uses a 2-byte instruction format. There are four categories of instructions: Type-1, Type-2, Type-3, and Type 4. Type-1 category consists of four instructions, each with 3 integer register operands (3Rs). Type-2 category consists of eight instructions, each with 2 floating point register operands (2Fs). Type-3 category consists of fourteen instructions, each with one integer register operand and one floating point register operand (1R+1F). Type-4 category consists of N instructions, each with a floating-point register operand (1F).

The maximum value of N is 32 ?

Topic : Effective Address

- Address of operand in a computation-type instruction or
- The target address in a branch-type instruction.

Topic : Branch Instruction

Currently $inst^n$ I2 is in execution

PC = 204

I2 is a branch $inst^n$

Branch not taken

Condition false

Next $inst^n$ to be executed ⇒ I3

⇓

No change in PC

Branch taken

Condition True

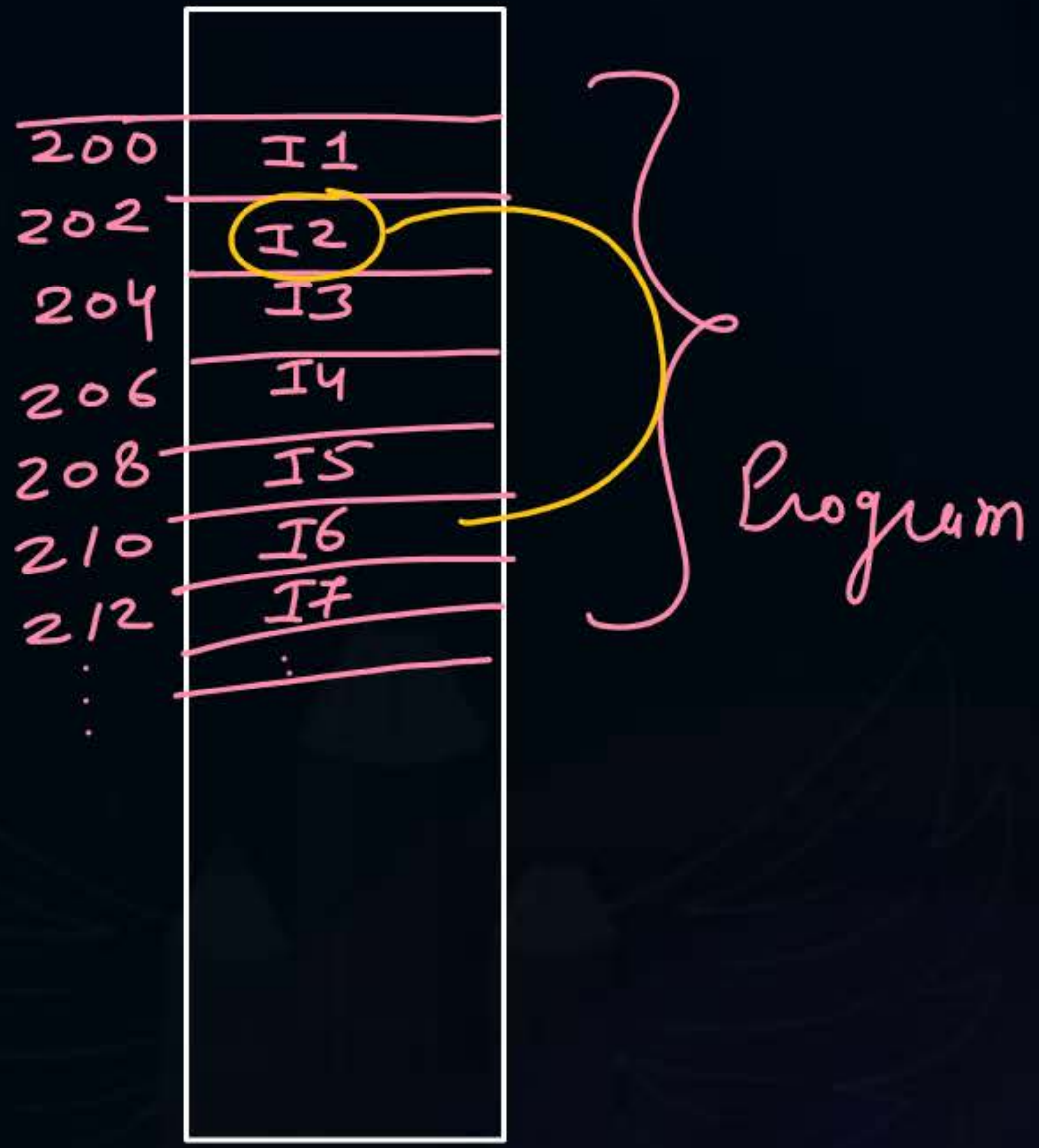
Next $inst^n$ to be executed, is Target $inst^n$.

ex:- Assume target is I6.

⇓

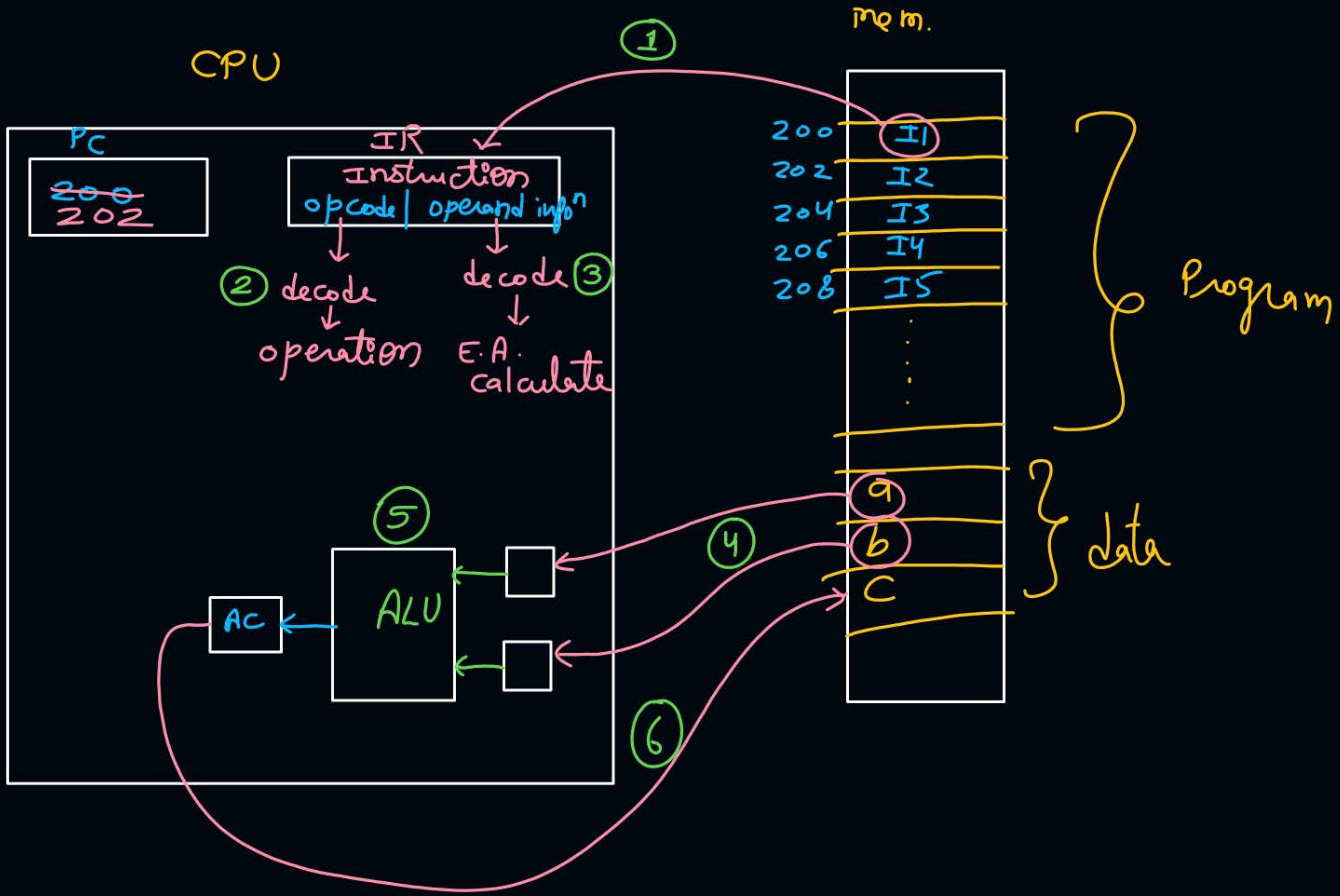
PC = address of I6

= 210 ← Target add. (Effective add.)



Topic : Instruction Cycle

1. Instruction Fetch
2. Instruction Decode
3. Effective Address Calculation
4. Operand Fetch
5. Execution
6. Write Back Result



CPU
PC
~~200~~
202
IR
Instruction
opcode | operand infon
1
2 decode
operation
decode 3
E.A. calculate
4
5
ALU
AC
6
mem.
200
202
204
206
208
I1
I2
I3
I4
I5
Program
a
b
c
data

Topic : Fetch Cycle & Execution Cycle

Inst^n fetch

decode

E.A. calculation

operand fetch

Execution

write back

Topic : Computation vs Branch Type Instruction

Instn fetch :- CPU fetches instn using PC value and increments PC.

ex :- ——— I1 ——— I2 and PC = 204

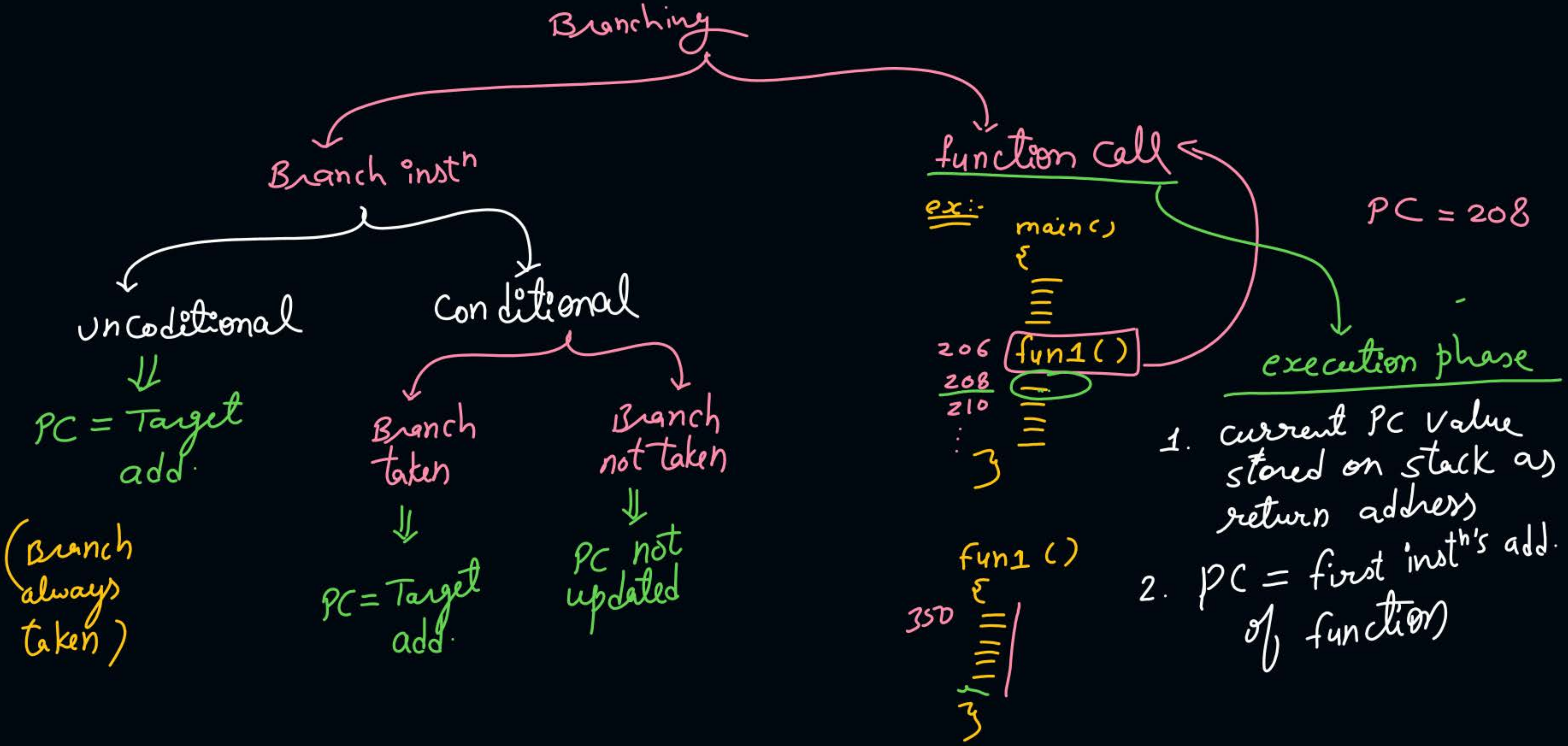
Instn decode :- CPU decodes opcode and gets the operation.

ex :- CPU decodes I2 as branch instn.

Eff. add. calculation :- CPU calculates target address.

ex :- Target address = 210

Execution :- CPU checks condition and updates PC by target add. if condition is true.



Branching

Branch inst^n

Uncoditional

⇓

PC = Target add.

(Branch always taken)

Conditional

Branch taken

⇓

PC = Target add.

Branch not taken

⇓

PC not updated

function call

ex:-

```
    main()
    {
      ===
206   fun1()
208   -
210   ===
 :
    }

    fun1()
    {
350   ===
      ===
      -
    }
```

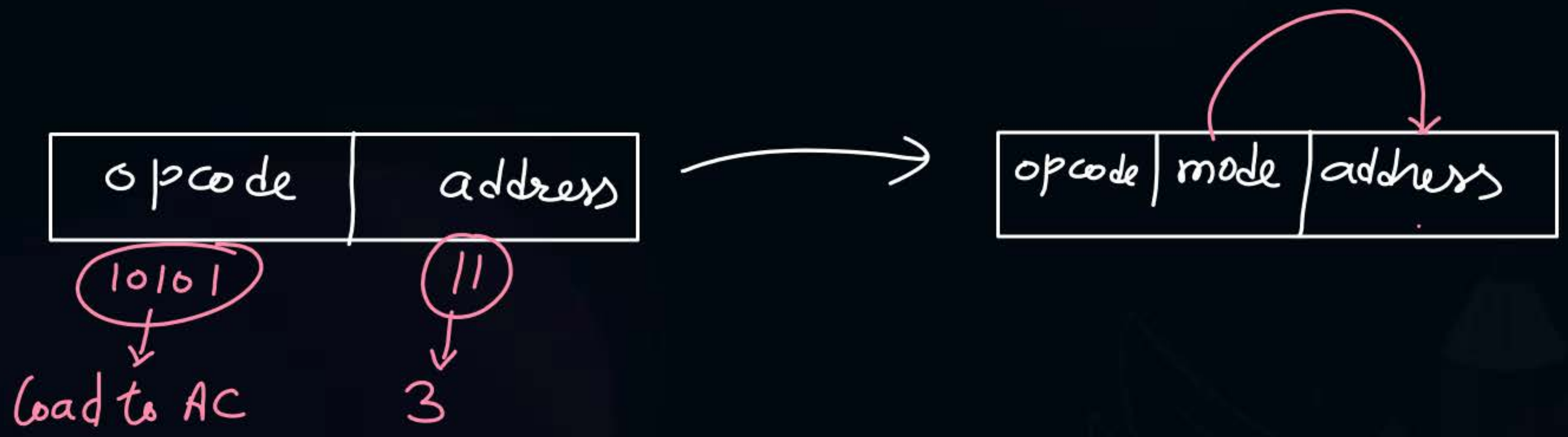
PC = 208

execution phase

1. current PC value stored on stack as return address
2. PC = first inst^n's add. of function

Topic : Addressing Modes

- It specifies how and from where the operands are obtained for an instruction



ex:-

$AC \leftarrow \#3$

or

$AC \leftarrow R3$

or

$AC \leftarrow M[3]$

Topic : Implied Mode

The opcode definition itself defines the operand

Opcode	Mode	~~Address~~
	Implied	

Opcode ↓

operation
+
operand

Ex:- INC A ⇒ Increment AC

Topic : Immediate Mode

The address field of instruction specifies the operand value

Opcode	Mode	Address

↓ operand

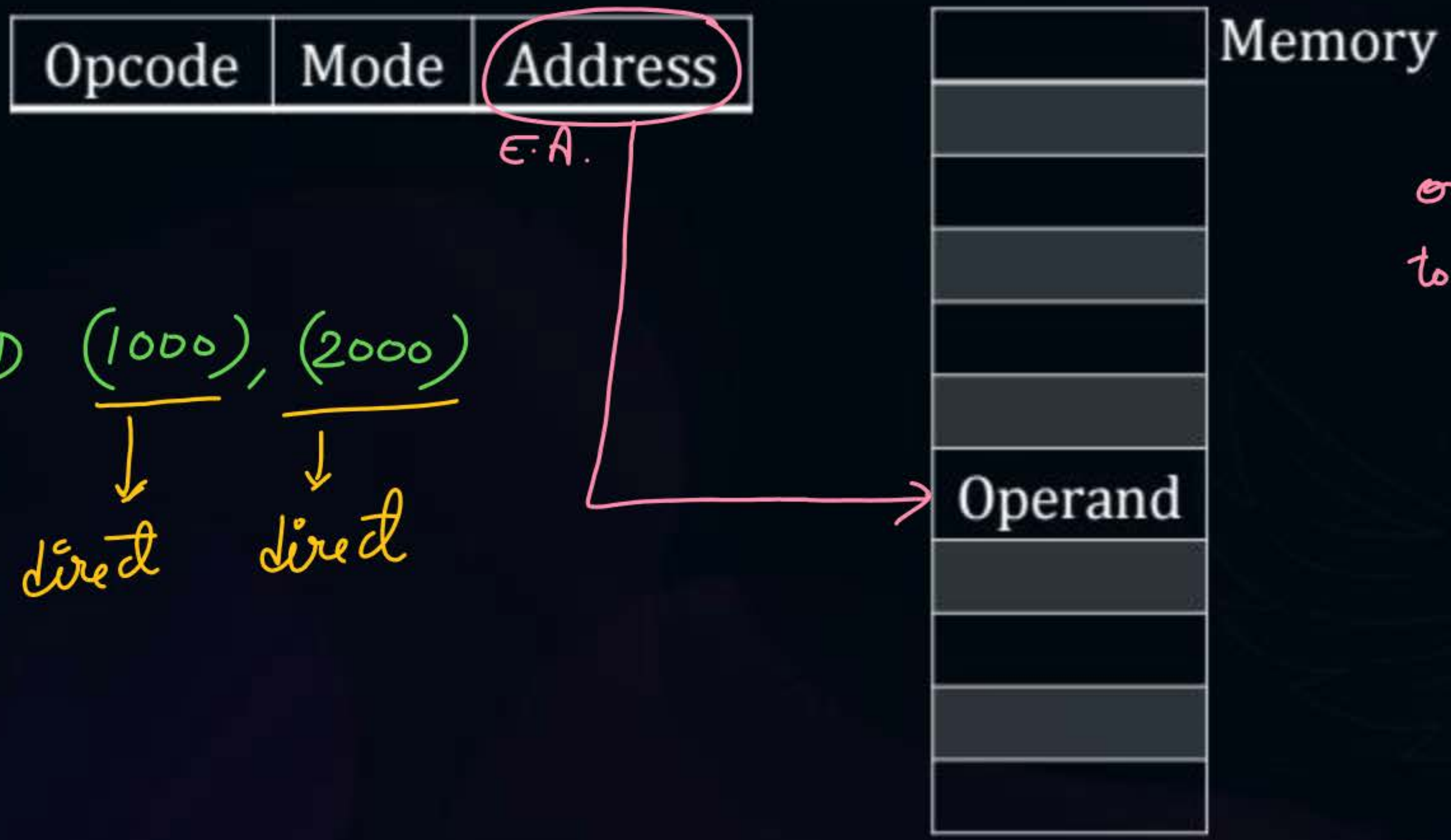
This mode is used to use Constant value within instn.

ADD R1, #12

↓ Immdiate value

Topic : Direct Mode /absolute mode

The address field of instruction specifies the effective address



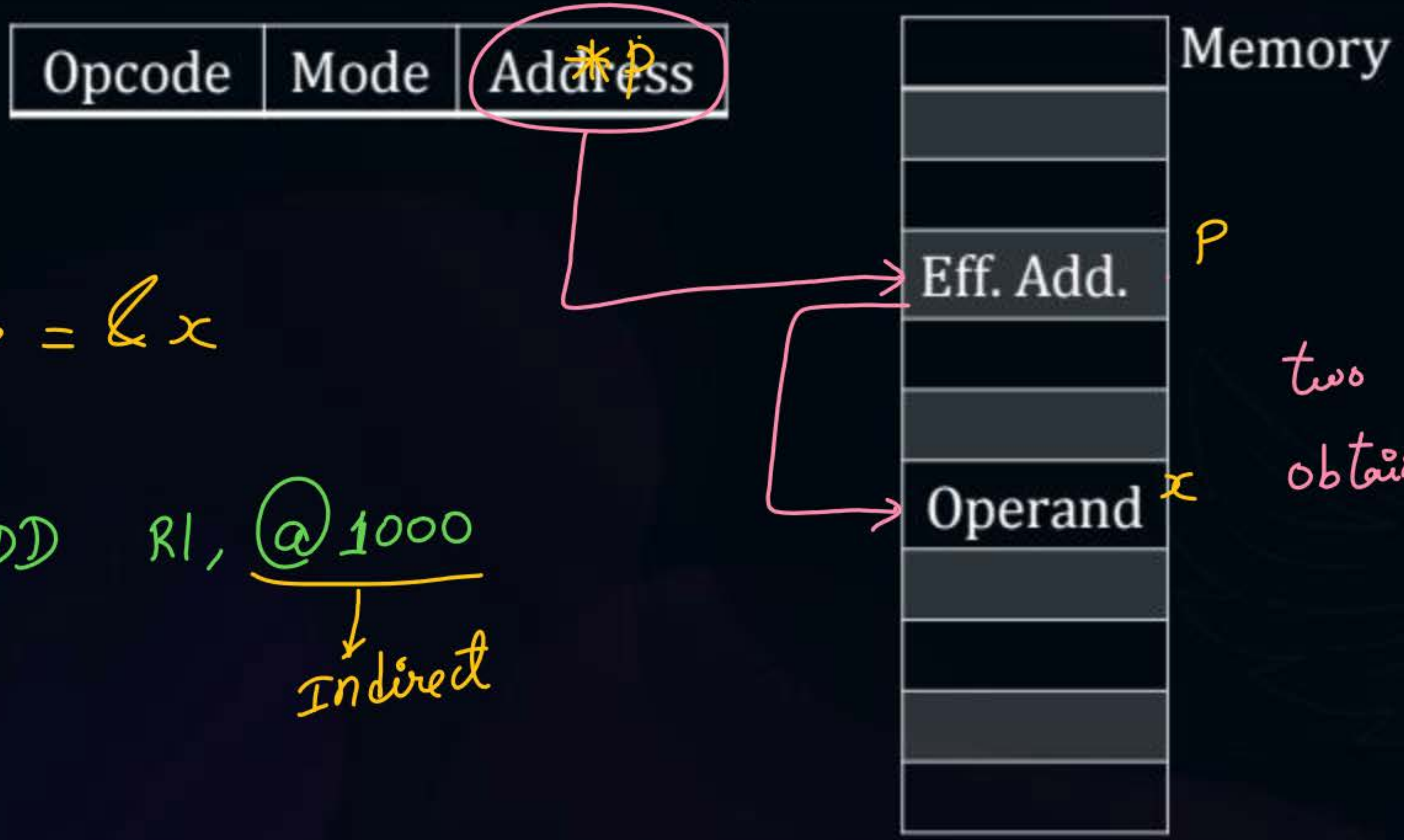
one mem. access needed to obtain operand.

$$\text{ADD } \underbrace{(1000)}_{\downarrow\ \text{direct}},\ \underbrace{(2000)}_{\downarrow\ \text{direct}}$$

Topic : Indirect Mode

This mode is used to implement pointers.

The address field of instruction specifies the address of effective address



Opcode	Mode	Address

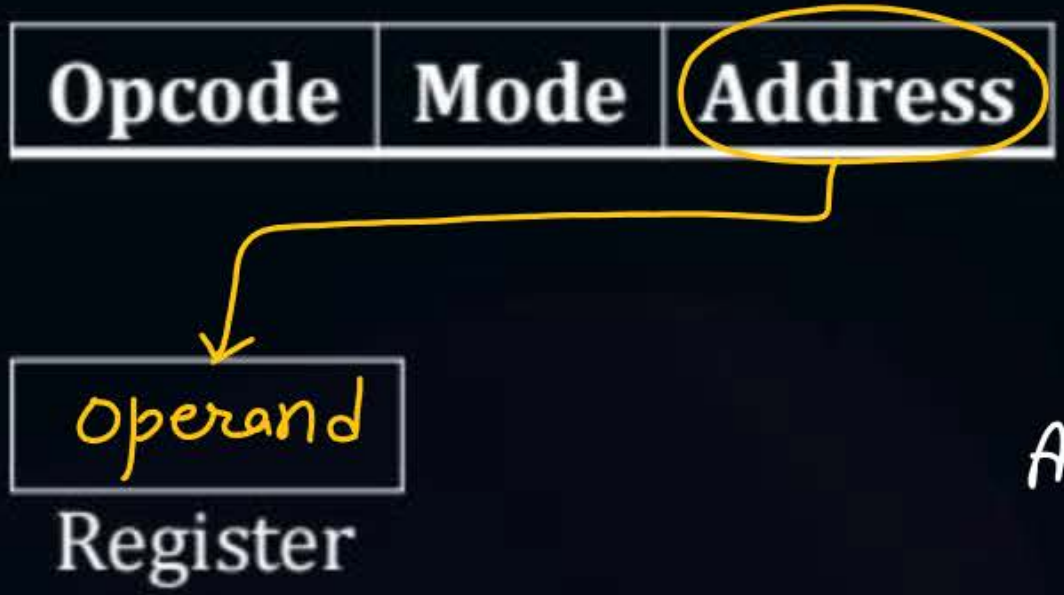
$*p = \&x$

two mem. accesses needed to obtain operand.

ADD R1, $\underbrace{@1000}_{\text{Indirect}}$

Topic : Register Mode (Reg. Direct)

The address field of instruction specifies a register which holds operand



Opcode	Mode	Address

ADD R1, R2

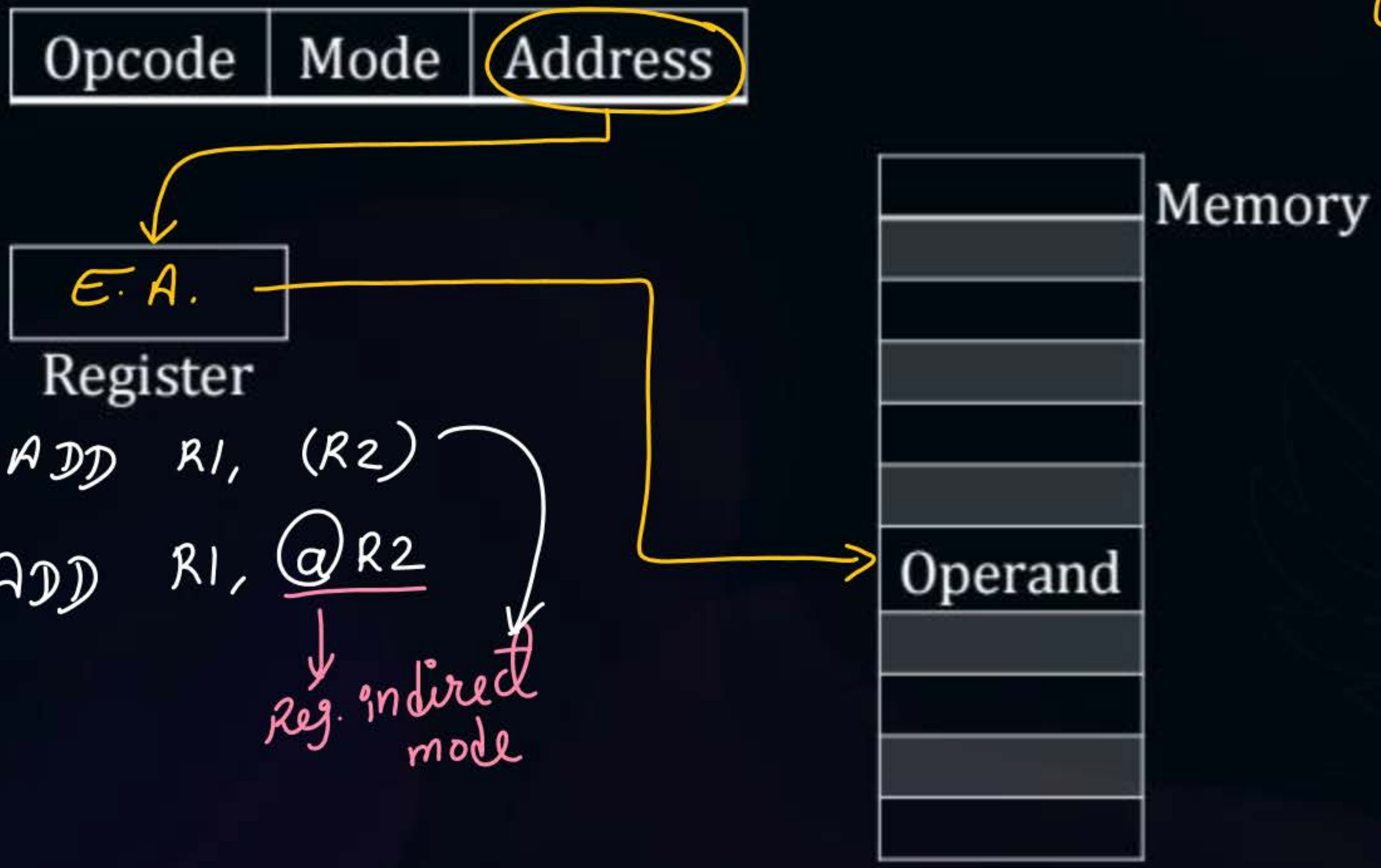
R1 → Reg. mode

R2 → Reg. mode

Topic : Register Indirect Mode

The address field of instruction specifies a register which holds effective address

used to shorten inst^n length



ADD R1, (R2)

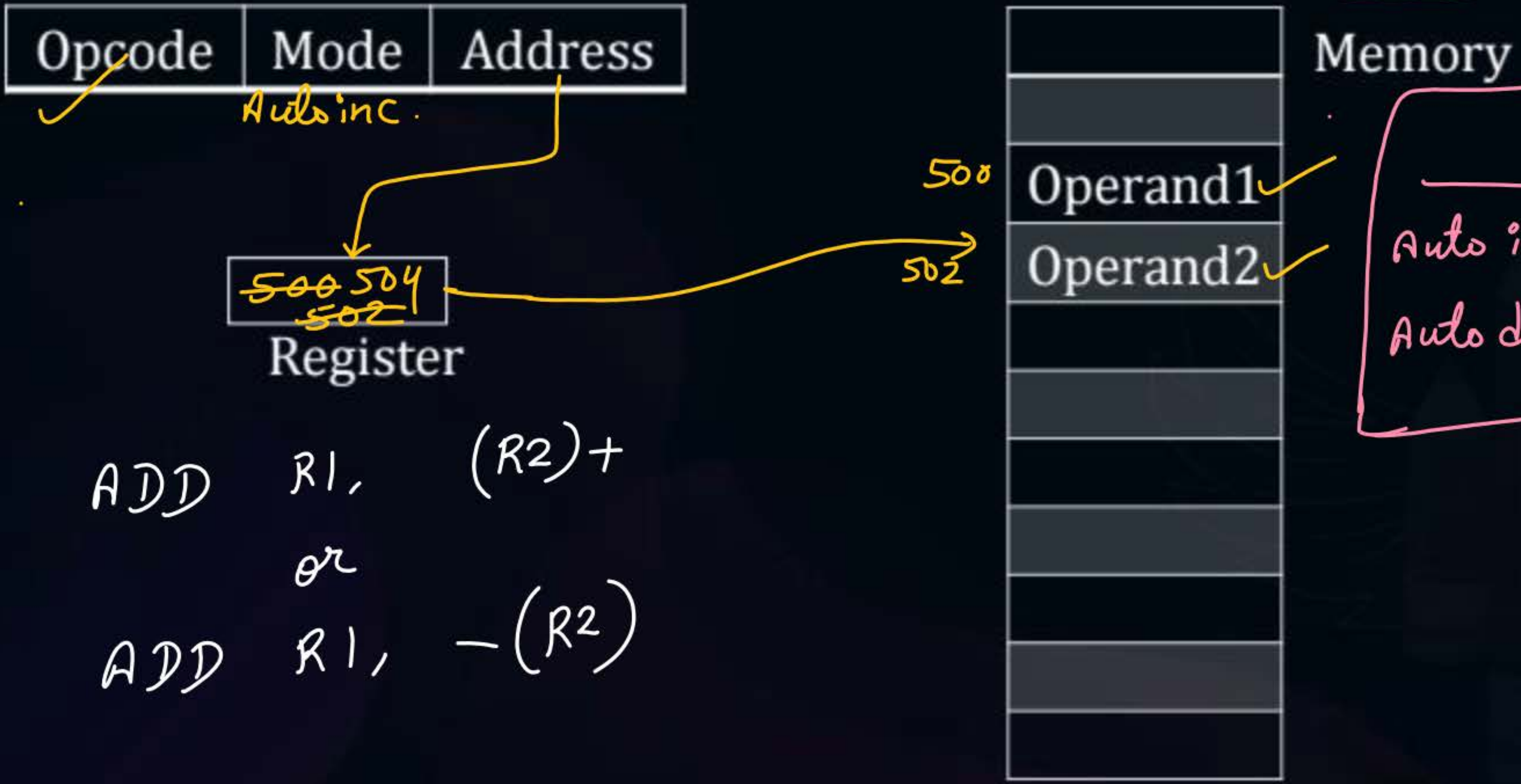
ADD R1, @R2

Reg. indirect mode

Topic : Autoincrement/Autodecrement Mode

Variant of register indirect mode, in which content of register is automatically incremented or decremented to access a table of content sequentially.

→ array



	By default
Auto inc.	Post increment
Auto dec.	Pre decrement

ADD R1, (R2)+

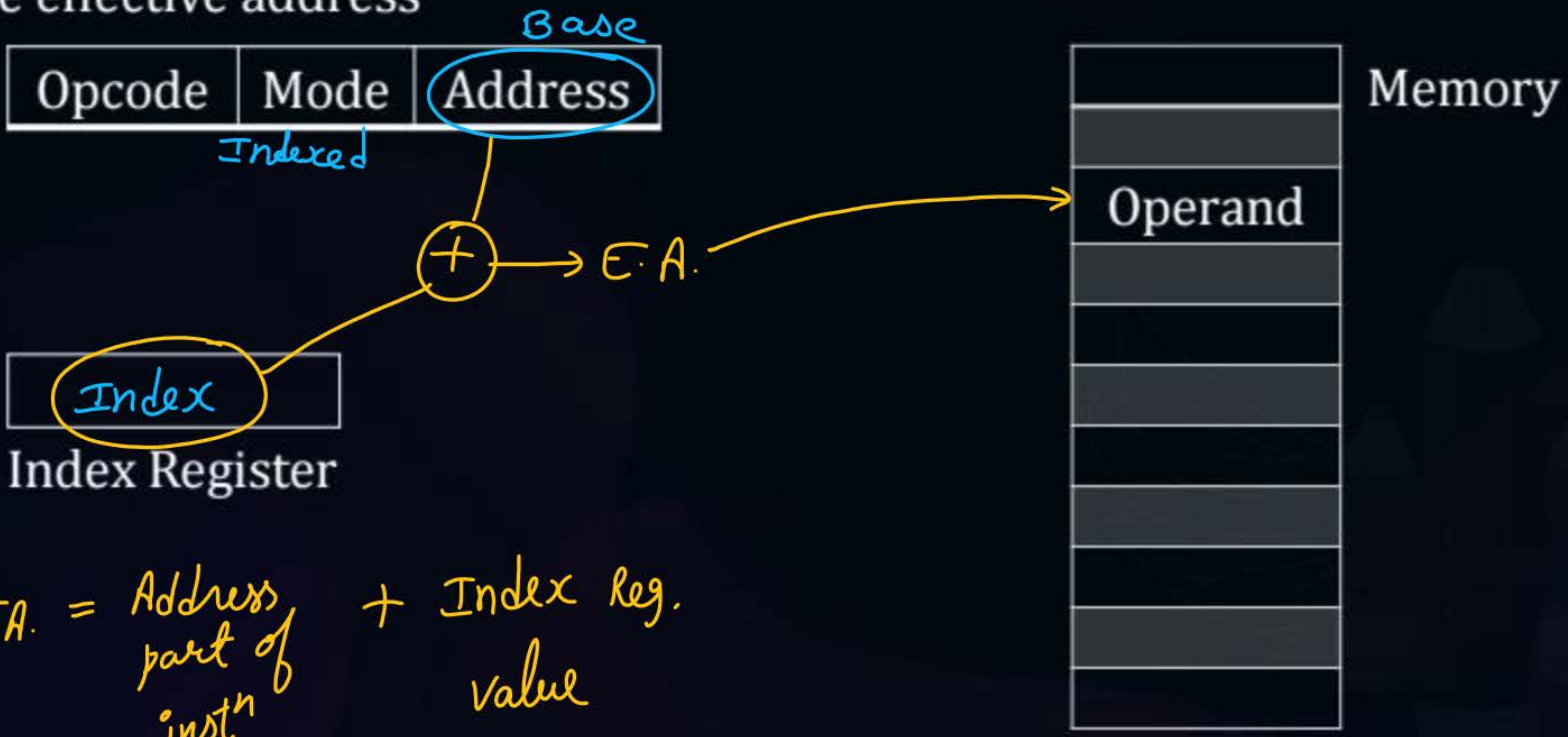
or

ADD R1, −(R2)

Topic : Indexed Mode

(Index Reg. mode) ⇒ Used to access a specific element of array

Address part of instruction (base address) is added to index register value to get the effective address



E.A. = Address part of instn + Index Reg. value

$$\underbrace{loc(A[i])}_{E.A.} = \underbrace{Base}_{\substack{inst^{n}'s \\ add.\ part}} + \underbrace{\text{size of element} * \overset{\text{Index}}{i}}_{Index\ Reg.}$$

$$\left.\begin{array}{l} \text{operation on } A[i] \implies \text{Index Reg} \leftarrow i * \text{size of element} \\ \qquad\qquad\qquad\quad \text{Indexed mode inst}^{n} \end{array}\right\} 2\ \text{inst}^{ns}$$

Actual $inst^n$

opcode	mode	Index Reg. number (designation)	add.

ADD R1, $\underline{1000(R2)}$

R2 ⇒ Index Reg.

1000 ⇒ Base add.

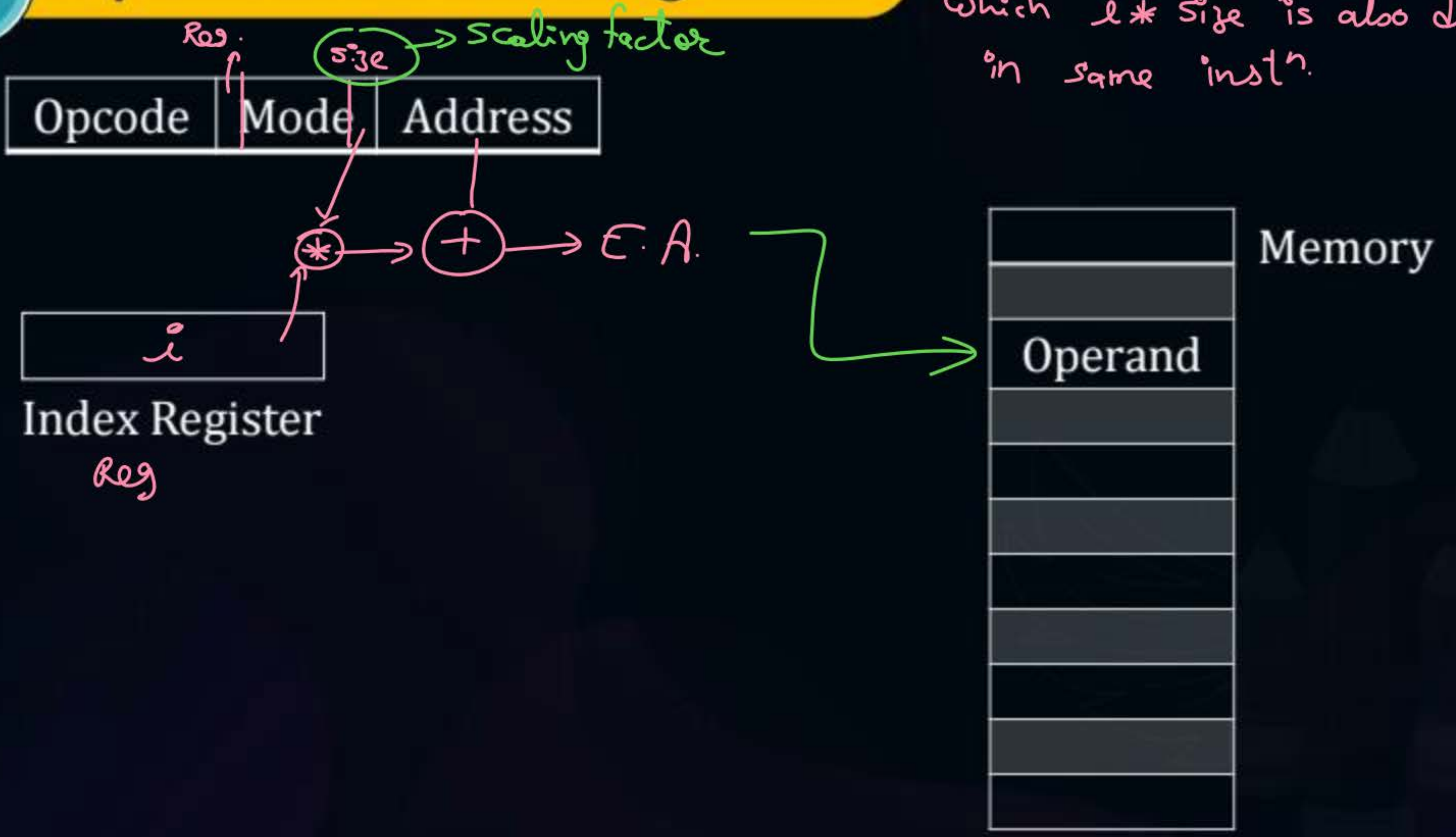
or

ADD R1, R2(1000)

$$EA = 1000 + R2$$

Topic : Scaled Addressing Mode

→ Advanced indexed mode, in which i * size is also done in same instⁿ.

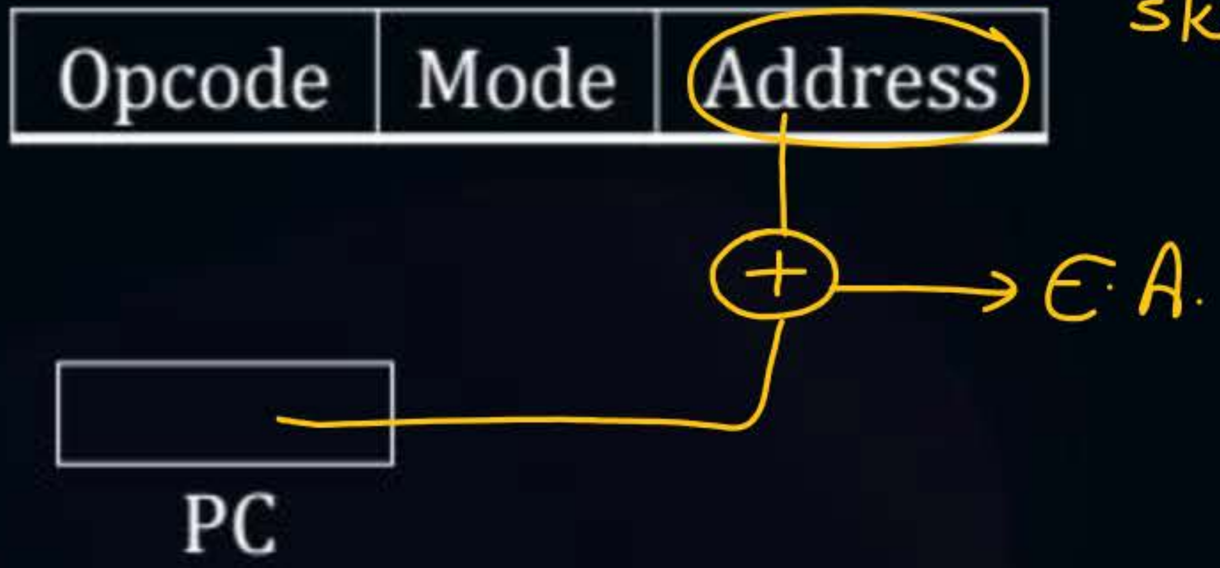


Topic : PC-Relative Mode → Used for intra-segment branching

(Position-Independent mode)

Address part of instruction (offset) is added to PC register value to get the effective address

Relative add. to skip (offset)



	offset
forw jump	+ve
Backward jump	−ve

$$E.A. = \underset{\text{of inst}^n}{\text{Address part}} + PC$$

I2 in CPU

PC = 204

CPU decodes I2 as branch instⁿ

⇓

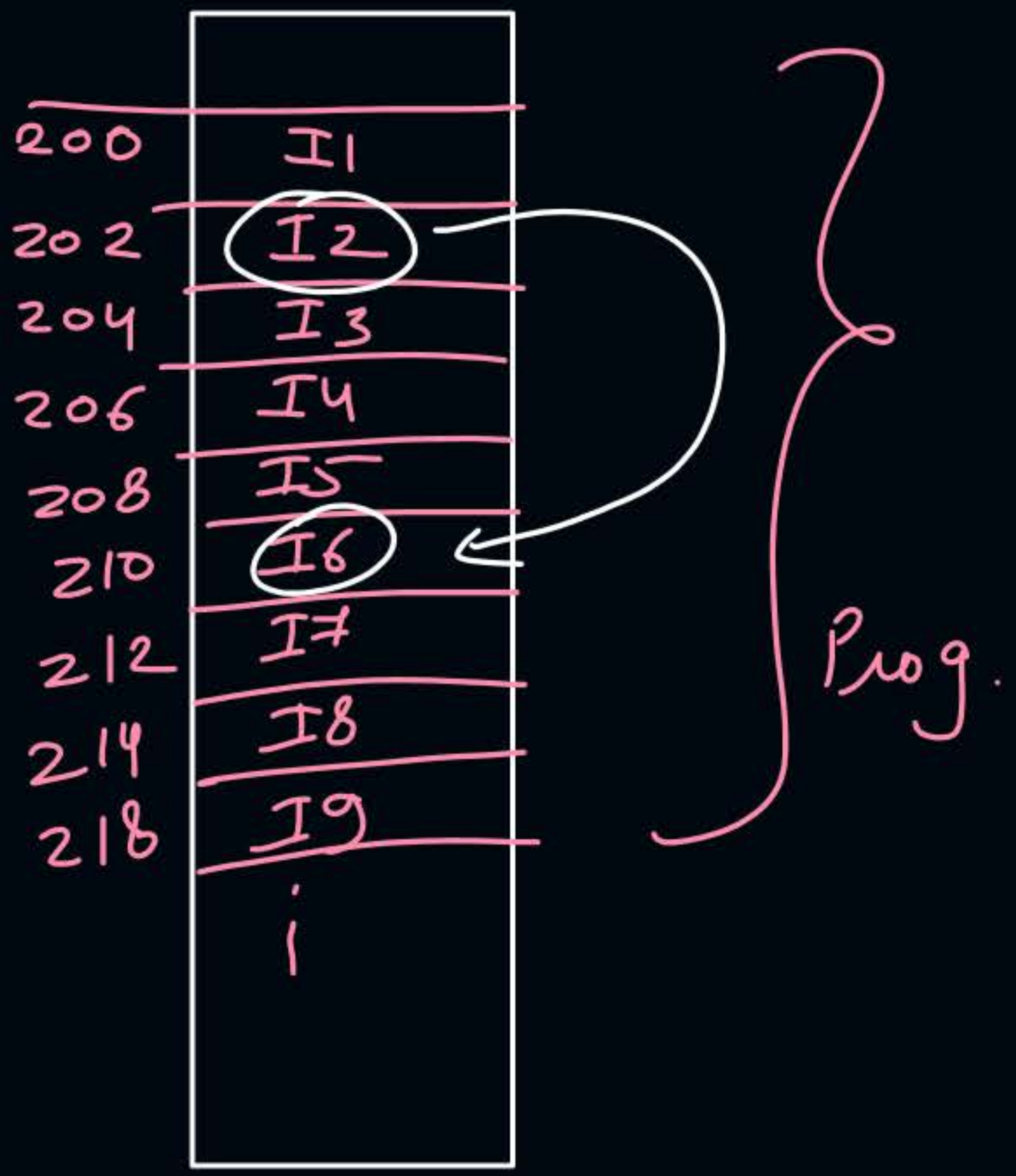
Target of I2 is I6

⇓

Target add. = $\underline{\underline{210}}$

Target add. = 204 + 6 ← Relative no. of locations to skip

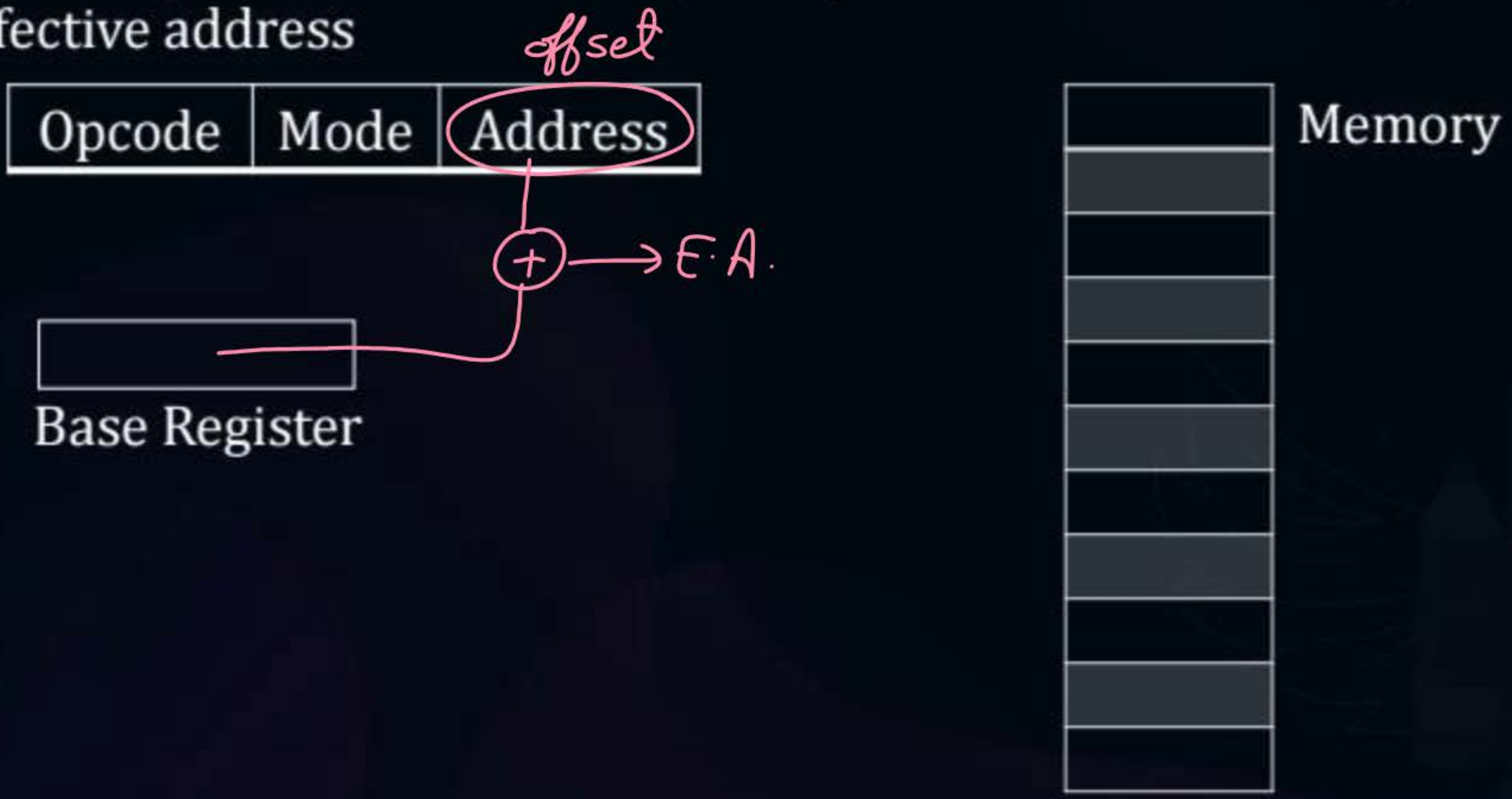
= 210



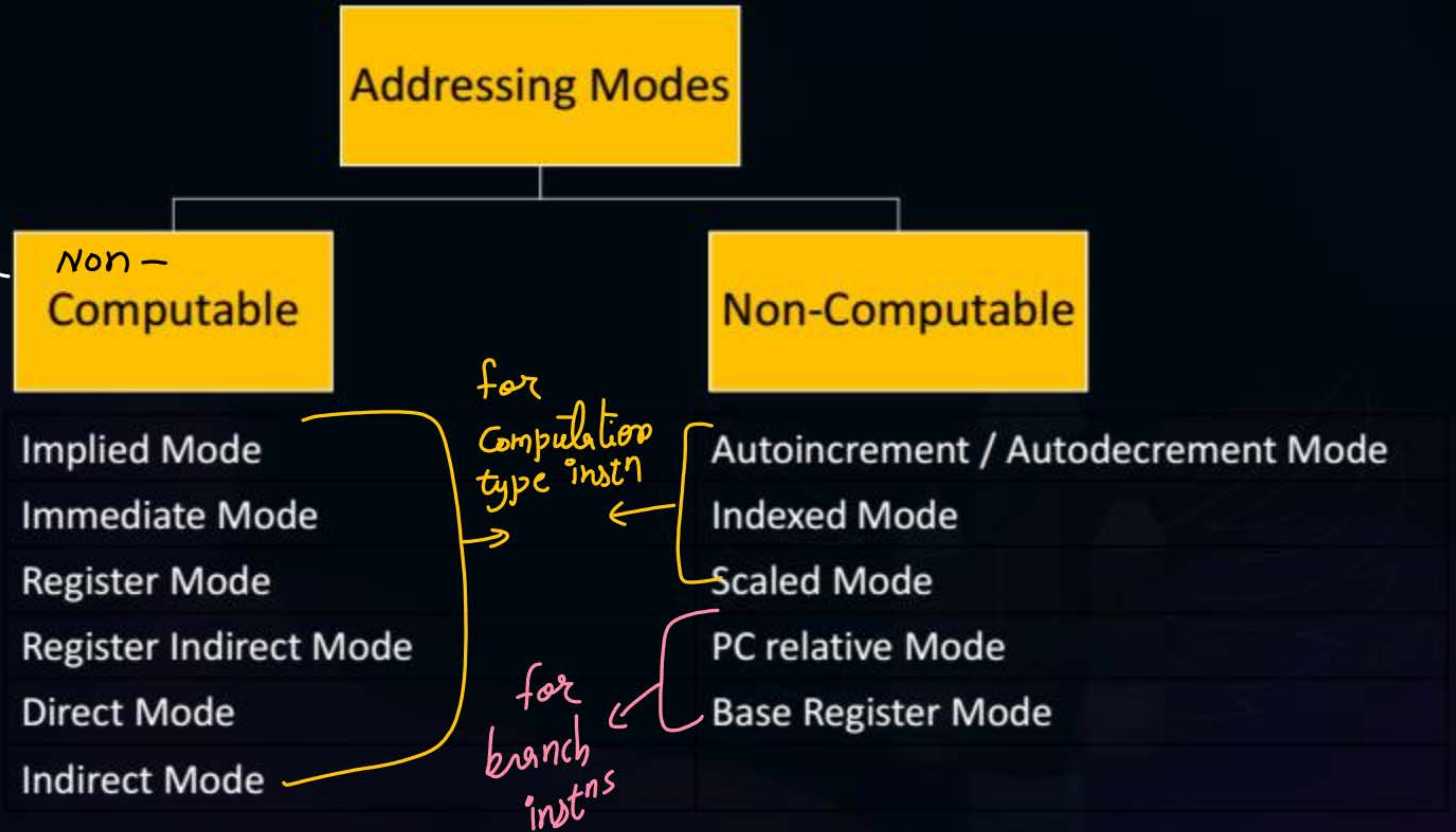
Topic : Base Register Mode

→ Inter - segment branching

Address part of instruction (offset) is added to Base register value to get the effective address



Topic : Types of Addressing Modes



Addressing Modes

Non – Computable	Non-Computable
Implied Mode	Autoincrement / Autodecrement Mode
Immediate Mode	Indexed Mode
Register Mode	Scaled Mode
Register Indirect Mode	PC relative Mode
Direct Mode	Base Register Mode
Indirect Mode	

No any computation to get E.A.

for computation type instn

for branch instns

Topic : Example

← Inst^n →

Op Code	mode	add. = 50

	Memory	
20	*Opcode*	Mode
21	Address = 50	
22	Next Instruction	
39	45	
40	70	
50	80	
60	90	
72	-----------------	
80	30	

PC = ~~20~~ 22

R50 = ~~40~~ 39

Index
XR = 10

AC

mem. add.
↑

Mode	Effective Address	Operand
1. Immediate Mode	21	50
2. Direct Mode	50	80
3. Indirect Mode	80	30
4. Register Mode	—	40
5. Register Indirect Mode	40	70
6. Autodecrement Mode (pre)	39	45
7. Indexed Mode	50+10= 60	90
8. PC- Relative Mode	22+50=72	—

[MCQ]

H.W.

#Q. An instruction is stored at Location 300 with its address field at location 301. The address field has the value 400. A processor register contains the number 150. Evaluate the effective address, if addressing mode is:

A Direct

B Immediate

C Relative

D Register Indirect

[MCQ]

H.W.

#Q. In case the code is position independent, the most suitable addressing mode is

A Direct mode

B Indirect mode

C Relative mode

D Indexed mode

[MCQ]

H. W.

#Q. The addressing mode that permits relocation, without any change whatsoever in the code, is

A Indirect addressing

B Base register addressing

C Indexed addressing

D PC relative addressing

[MCQ]

H.W.

#Q. A relative branch mode type instruction is stored in memory at address 250. The branch is made to an address 450.

1. What should be the value of relative address field of the instruction?
2. Determine the value of PC before instruction fetch, after the fetch and after execution phase?

GATE
WALLAH

2 mins Summary

Topic Effective Address

Topic Addressing Modes

Happy Learning

THANK - YOU