CS & IT ENGINEERING

COMPUTER ORGANIZATION AND ARCHITECTURE

Instruction & Addressing Modes

Lecture No.- 05

By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic Instruction Construction for CPU

Topics to be Covered

Topic Instruction Cycle

Topic Fetch & Execution Cycle

Topic Branch Instruction

Addressing mod

Topic : Effective Address

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

Topic : Branch Instruction

CPU is currently executing inst^n I2.

PC = 502

CPU decodes I2 as branch type inst^n

branch inst^n condition

True

Branch taken

Target inst^n on which jump to be taken, is executed next.

False

Branch not taken

Next inst^n in the sequence I3 executes next.

PC value remains same 502.

True

Assume Target inst^n is I7.

I7 called as Target inst^n and it's mem. address 506 is called as target address.

Branch inst^n while executing changes PC = Target add.

= 506

Topic : Instruction Cycle → 6 steps to execute an inst^n

(IF) 1. Instruction Fetch

(ID) 2. Instruction Decode

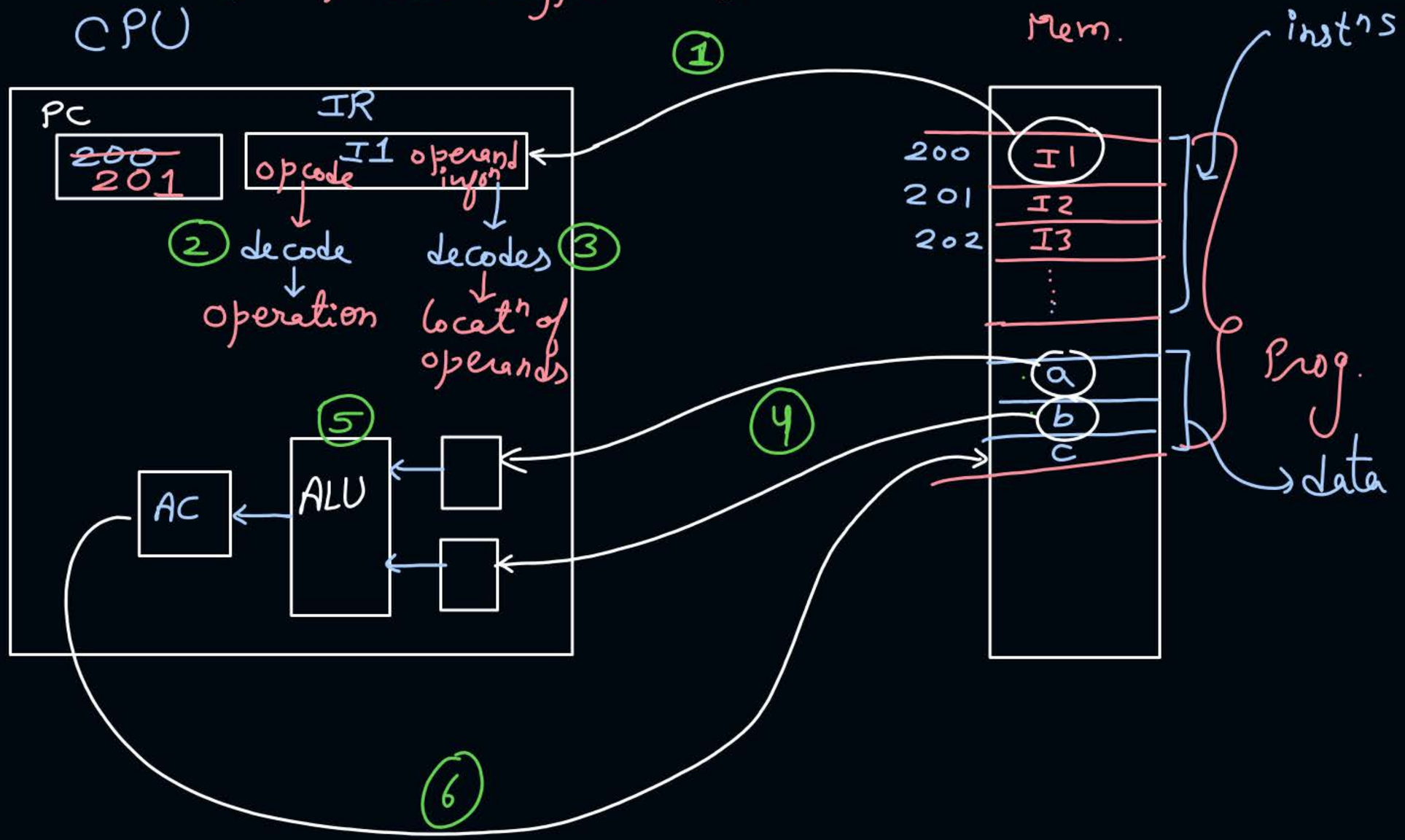
(EAC) 3. Effective Address Calculation

OF 4. Operand Fetch

Ex 5. Execution

WB 6. Write Back Result

(Computation type instn)



Topic : Fetch Cycle & Execution Cycle

only inst^{n} fetch

from decode

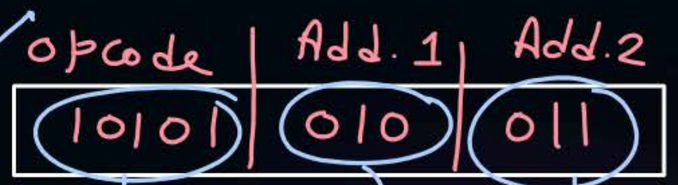
to write back

Topic : Computation vs Branch Type Instruction

	Computation Type Instruction	Branch Type Instruction
IF	Fetch the inst^n using PC value & increment PC by size of inst^n	Fetch the inst^n using PC value & inc. PC.
ID	decode opcode	decode opcode
EAC	Address/Locat^n of operand calculated	Target address calculated.
OF	operands are fetched	—
EX	operation is performed & result generated	condition evaluation & PC updation if needed
WB	Result written back to destinat^n	—

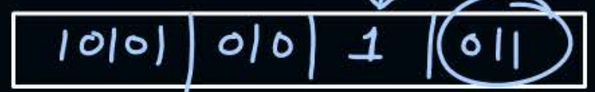
Topic : Why Addressing Modes

Assume for a Reg. – memory based architecture CPU an inst^n is as:



Sol^n ⇒ add mode bit in inst^n

decodes add. is used for Reg.



mem. add.

Addition operation

definitely R2

? 2 possibilities

$(011)_2 = (3)_{10}$ ⇒ Reg. R3

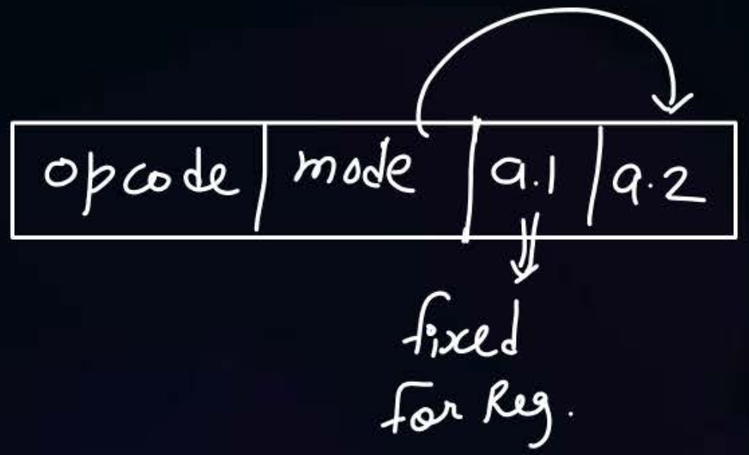
$(011)_2 = (3)_{10}$ ⇒ memory add. 3

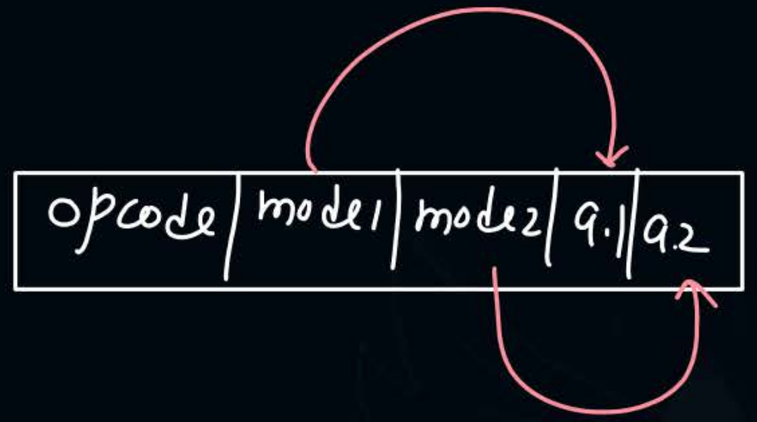
---

int a;

Topic : Addressing Modes

- It specifies how and from where the operands are obtained for an instruction using address part of inst^{n}.





Topic : Implied Mode

The opcode definition itself defines the operand

Opcode	Mode	~~Address~~

↳ Implied mode

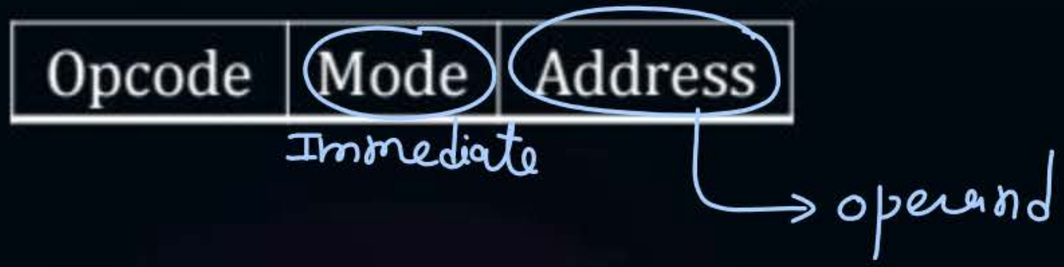
operation + operand

for any special purpose register access, no any explicit address must be needed in address part ; instⁿ.

ex:- opcode = INCA ⇒ Increment AC

Topic : Immediate Mode

The address field of instruction specifies the operand value



Use :- To perform operation on constant value.

ex:- $R2 \leftarrow R2 + \#3$

To initialize Registers with constant value.

ex:- $R3 \leftarrow \#4$

Topic : Direct Mode (Absolute mode)

The address field of instruction specifies the effective address

Opcode	Mode	Address

Direct

No. of times memory accessed to obtain operand

= 1

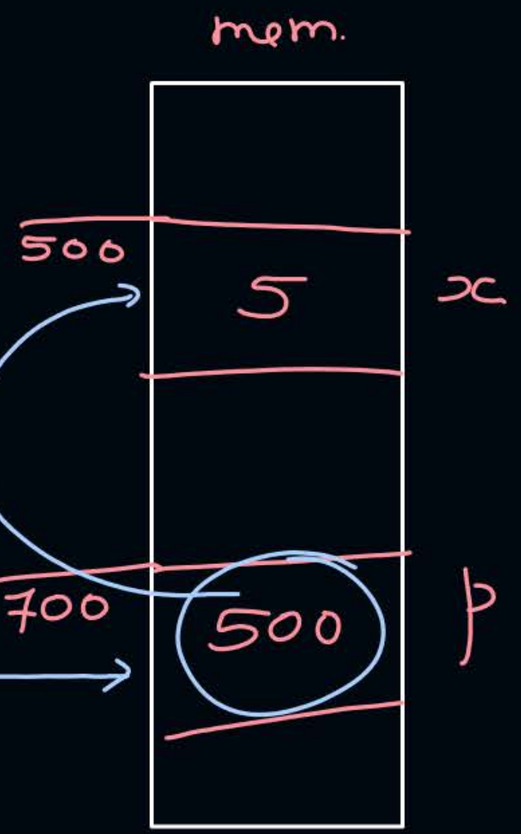
Topic : Indirect Mode

The address field of instruction specifies the address of effective address

No. of times memory is accessed to get operand = 2

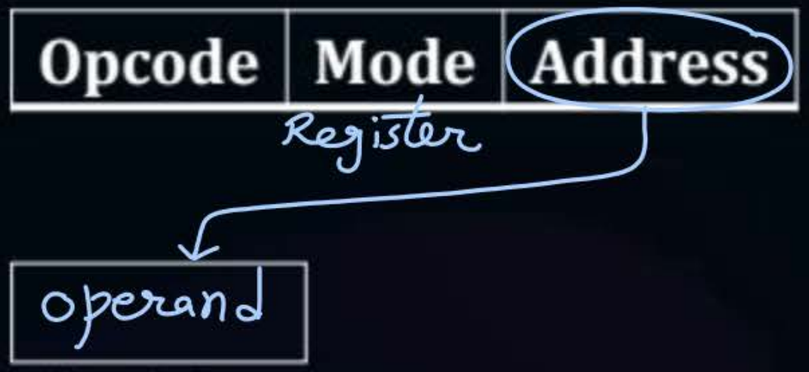
Use :- Used to implemented pointer

```
int x = 5;
int *p = &x;
printf("%d", *p);
```



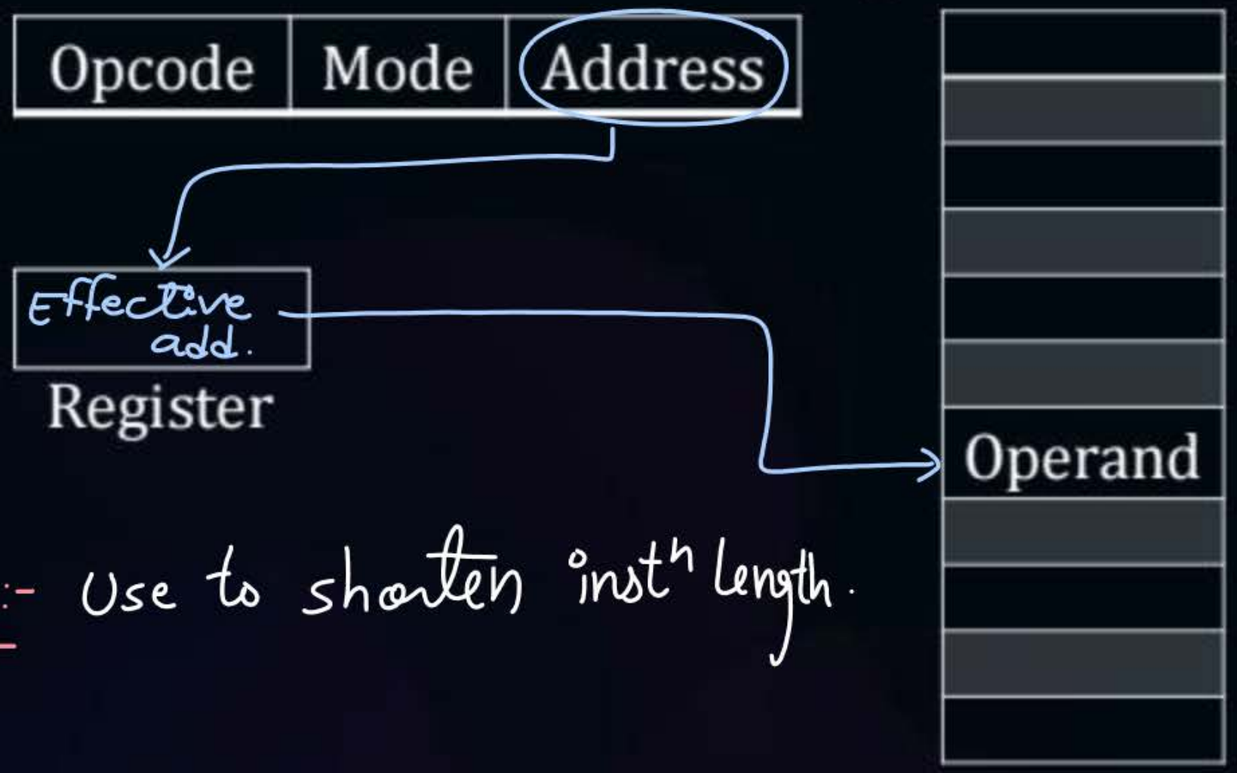
Topic : Register Mode

The address field of instruction specifies a register which holds operand



Topic : Register Indirect Mode

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



No. of times memory accessed to get operand = 1

Use :- Use to shorten inst^{n} length.

Direct mode :-

$\Rightarrow (x+y+32)$ bits

Time to get operand = 1 mem. access

Reg. Indirect mode :-

$\Rightarrow (x+y+6)$ bits

Time to get operand = 1 Reg. access
+
1 mem access

ex:-
system has 64 GPR's & 4GB RAM.

4GB
↓
mem. add. = 32 bits

reg. field = 6 bits

2 mins Summary

Topic Instruction Cycle

Topic Fetch & Execution Cycle

Topic Branch Instruction

Happy Learning

THANK - YOU