CS & IT ENGINEERING

COMPUTER ORGANIZATION AND ARCHITECTURE

Instruction & Addressing Modes

Lecture No.- 04

By- Vishvadeep Gothi sir

Recap of Previous Lecture

- Topic: Multiple Type Instructions Support
- Topic: Variable length Instructions
- Topic: Register Spill
- Topic: Instruction Generation for CPU

Topics to be Covered

- Topic: Effective Address
- Topic: Branch Instruction
- Topic: Instruction Cycle
- Topic: Fetch Cycle & Execution Cycle
- Topic: Addressing Modes

[NAT]

Ans = 2

#Q. Consider a register-memory architecture system (2-address instructions). For this system the following intermediate code is going to be converted in machine code. Minimum how many registers are required in system so that the code can run without register spill?

Consider first operand is always the register operand and it's the destination for operation too.

t1 = X + Y
t2 = t1 – Z
t3 = t1 + t2
t4 = M + t3

R1 ← X
R1 ← R1 + Y
R2 ← R1
R1 ← R1 − Z

R2 ← R2 + R1
R1 ← M
R1 ← R1 + R2

Assume X, Y, Z and M are memory operands

[NAT]

Ans = 3

#Q. Consider a register-based architecture system (2-address instructions). For this system the following intermediate code is going to be converted in machine code. Minimum how many registers are required in system so that the code can run without register spill?

Consider first operand is always the register operand and it's the destination for operation too.

t1 = X + Y	R1 ← X	R3 ← Z
t2 = t1 – Z	R2 ← Y	R1 ← R1 – R3
t3 = t1 + t2	R1 ← R1 + R2	R2 ← R2 + R1
t4 = M + t3	R2 ← R1	R1 ← M
		R1 ← R1 + R2

Assume X, Y, Z and M are memory operands

Topic : Effective Address (EA.)

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

Topic : Branch Instruction

Assuming CPU is currently executing inst^n I2.

PC = 502 (address of I3)

Assume CPU decodes I2 as branch (jump) inst^n.

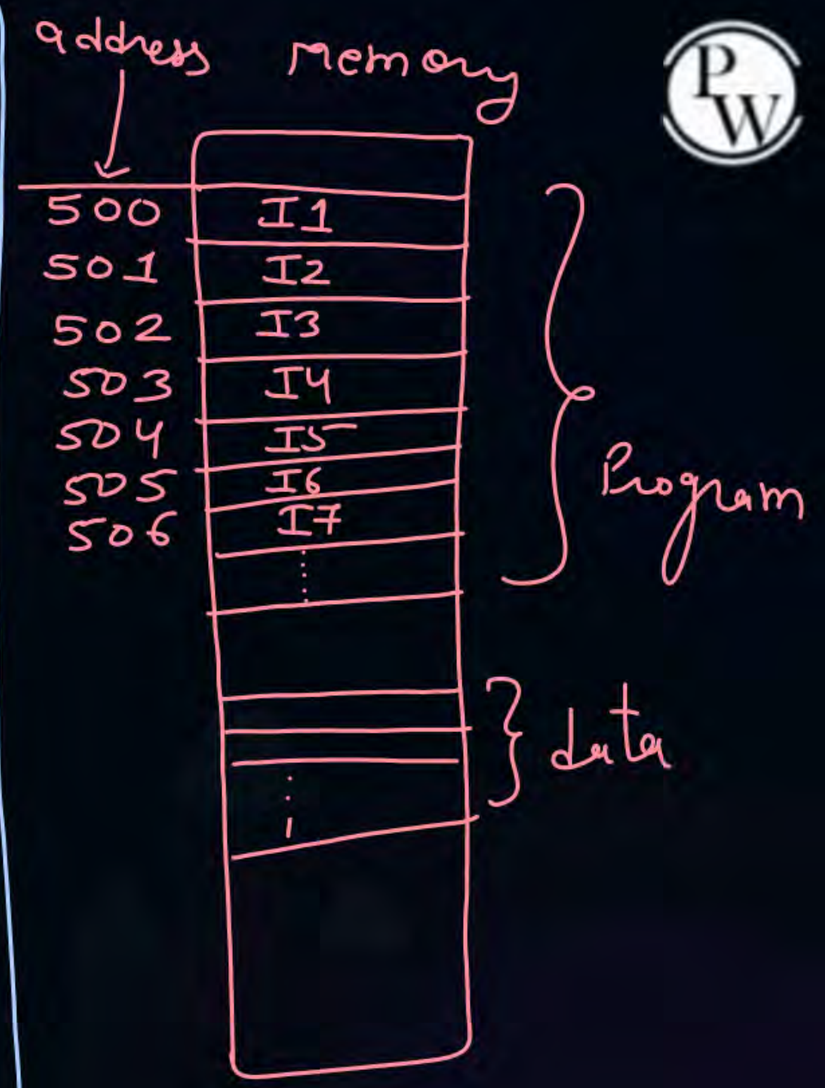
Condition

True → Take jump or Branch taken

False → Don't take jump or Branch not taken

Next inst^n in sequence, which is I3 will be executed next.

no any change in PC.



condit^{n} → True → Branch taken

Next inst^{n} executed ⇒ Target inst^{n} [where the Branch inst^{n} I2 will take branch]

assume I7 is target of I2.

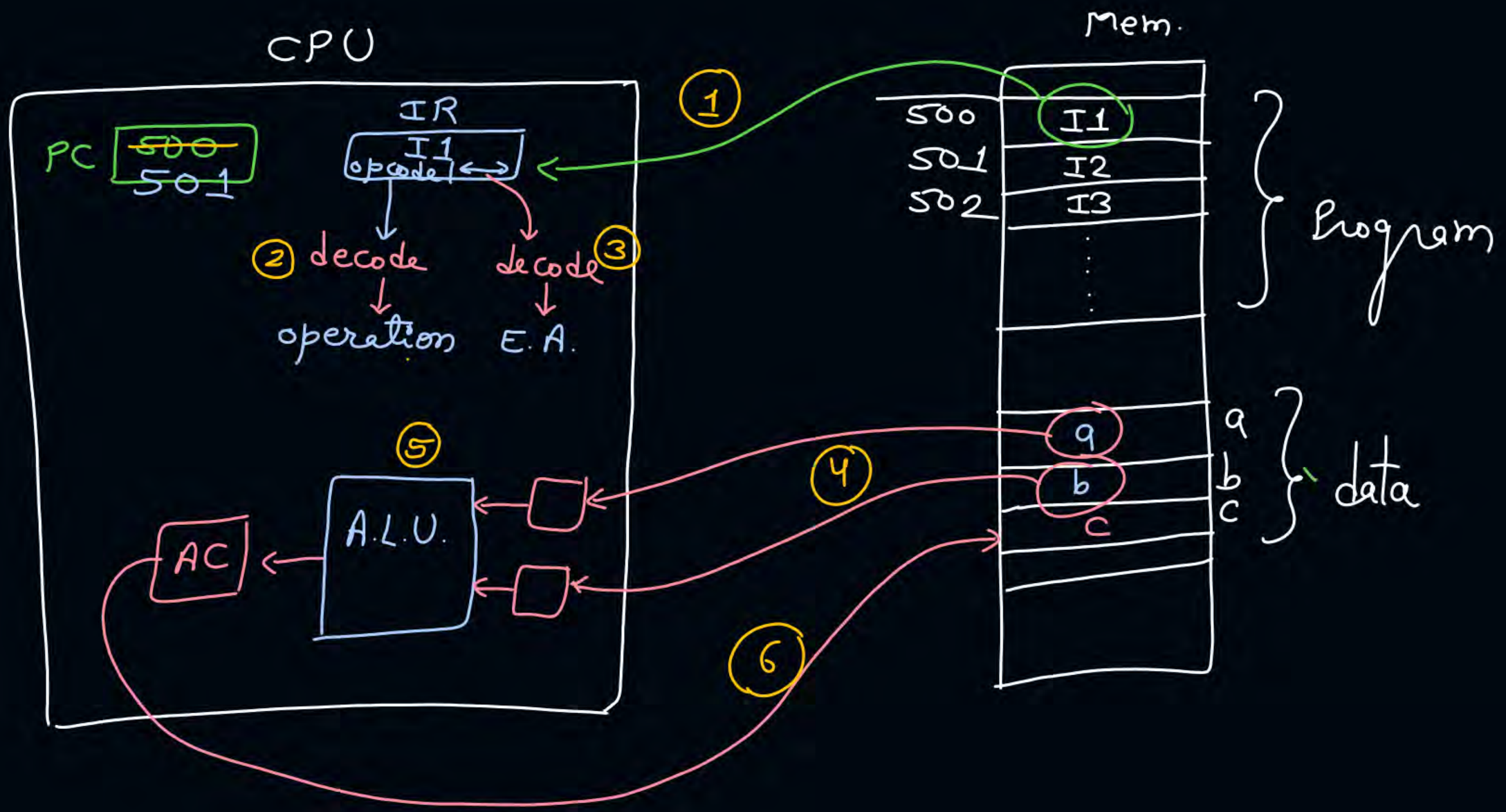
PC value will be updated by value 506

Effective address for branch inst^{n} I2 ⇒ 506

Topic : Instruction Cycle

steps to execute an instⁿ.

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


CPU
Mem.
IR
I1
opcode
PC
500
501
1
2 decode
operation
decode 3
E.A.
500
501
502
I1
I2
I3
Program
5
A.L.U.
AC
4
a
b
c
a
b
c
data
6

Instⁿ cycle for branch type instⁿ :-

1. Instⁿ fetch ⇒ Read instⁿ from mem. & bring it to IR.
   PC incremented

2. Instⁿ decode ⇒ Decodes opcode part & try to understand operatⁿ.

3. E.A. Calculation :- CPU calculates target address

4. operand fetch :- Not needed for branch instⁿ

5. Execution :- Condition check & PC updation by target add. if needed

6. write back result :- Not needed for branch inst^{n}

Topic : Fetch Cycle & Execution Cycle

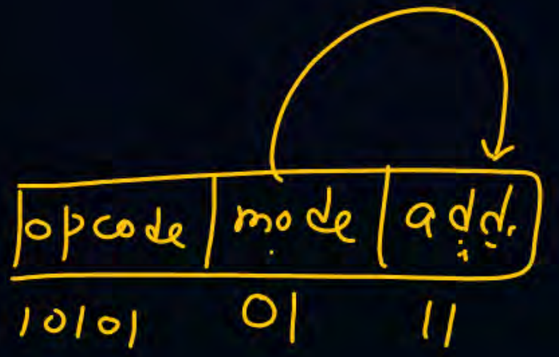
only instⁿ fetch

from decode to write back result

Topic : Computation vs Branch Type Instruction

Topic : Why Addressing Modes

ex:- 1-add. inst^{n}



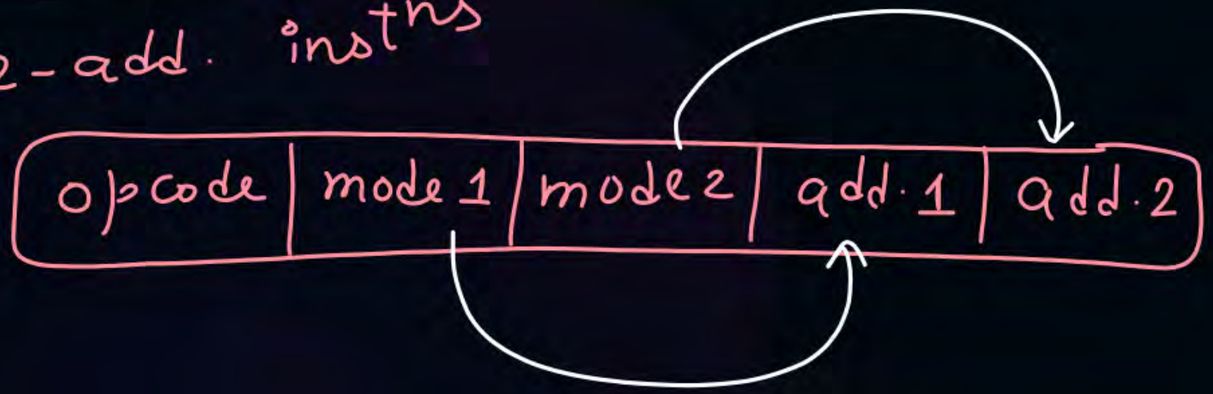
ex:- mode

AC ← #3	00
or	
AC ← R3	01
AC ← M[3]	10

Topic : Addressing Modes

- It specifies how and from where the operands are obtained for an instruction using address field.

2-add. instns



Topic : Implied Mode

The opcode definition itself defines the operand

Opcode	Mode	Address

Topic : Immediate Mode

The address field of instruction specifies the operand value

Opcode	Mode	Address

Topic : Direct Mode

The address field of instruction specifies the effective address

Opcode	Mode	Address

Topic : Indirect Mode

The address field of instruction specifies the effective address

Opcode	Mode	Address

Topic : Register Mode

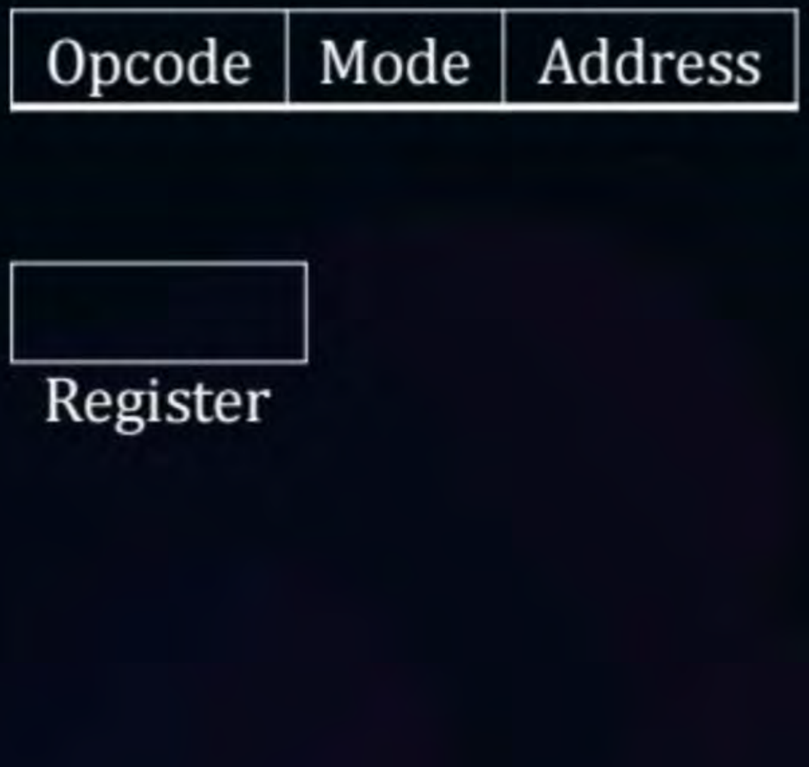
The address field of instruction specifies a register which holds operand

Opcode	Mode	Address

Register

Topic : Register Indirect Mode

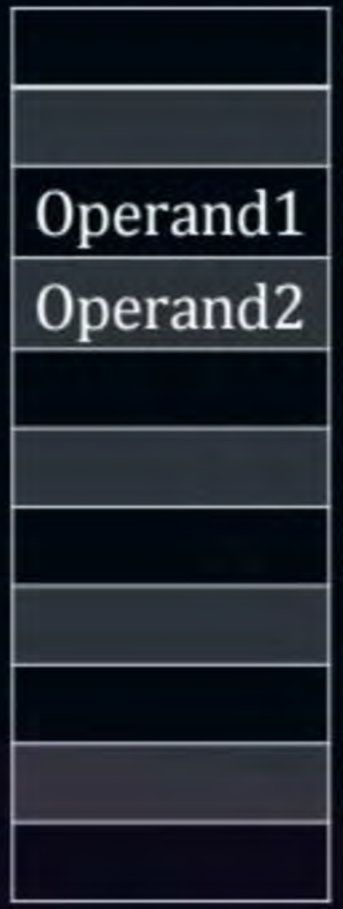
The address field of instruction specifies a register which holds operand



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.

Register

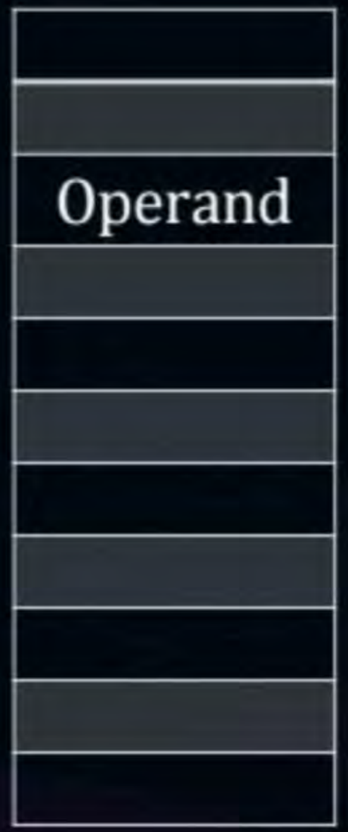


Memory

Topic : Indexed Mode

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

Index Register



Memory

Topic : PC-Relative Mode

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

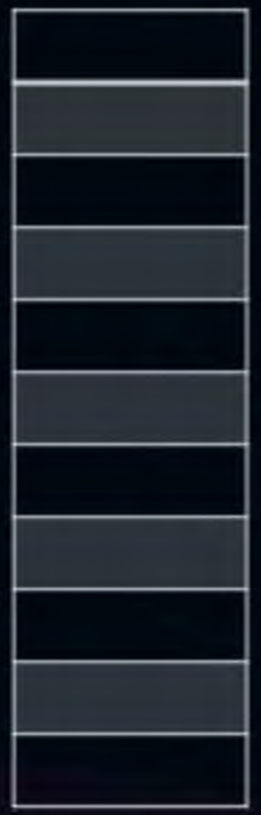
PC

Memory

Topic : Base Register Mode

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

Base Register



Memory

Topic : Example

	Memory	
200		
201	*Opcode*	Mode
202	Address = 500	
	Next Instruction	
399	450	
400	700	
500	800	
600	900	
702	----------	
800	300	

PC =200

R500 = 400

XR = 100

AC

Mode	Effective Address	Operand
1. Immediate Mode		
2. Direct Mode		
3. Indirect Mode		
4. Register Mode		
5. Register Indirect Mode		
6. Autodecrement Mode		
7. Indexed Mode		
8. PC- Relative Mode		

2 mins Summary

- Topic: Effective Address
- Topic: Branch Instruction
- Topic: Instruction Cycle
- Topic: Fetch Cycle & Execution Cycle
- Topic: Addressing Modes

Happy Learning

THANK - YOU