COMPUTER SCIENCE

Computer Organization and Architecture

Machine Instruction and Addressing Modes

Lecture_08

Vijay Agarwal sir

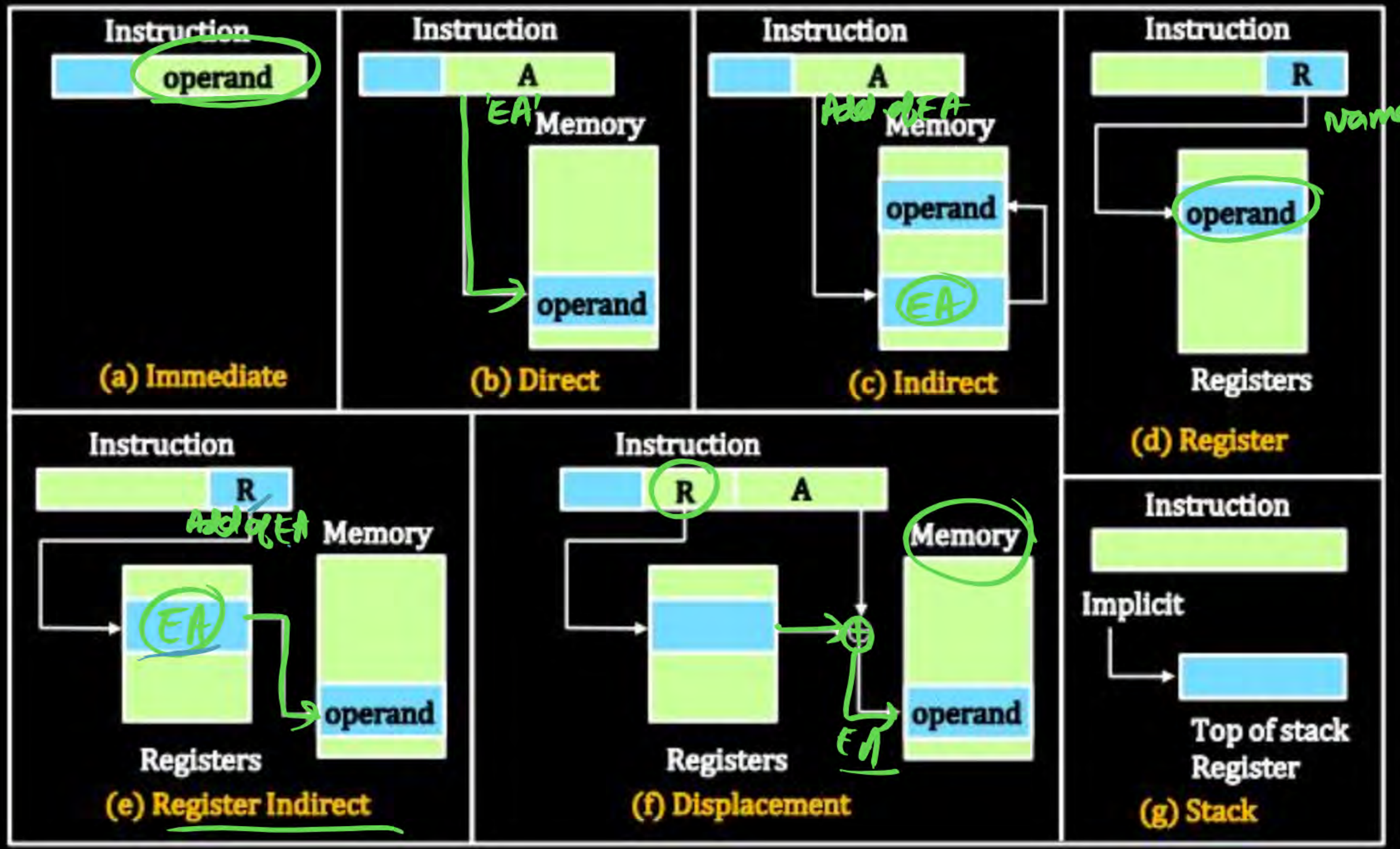
01 Addressing Modes

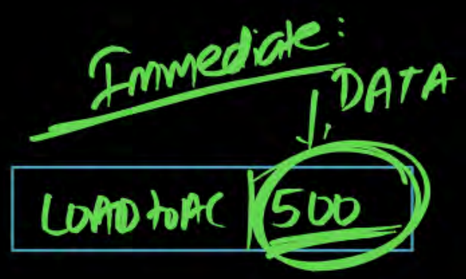
ADDRESSING MODE :

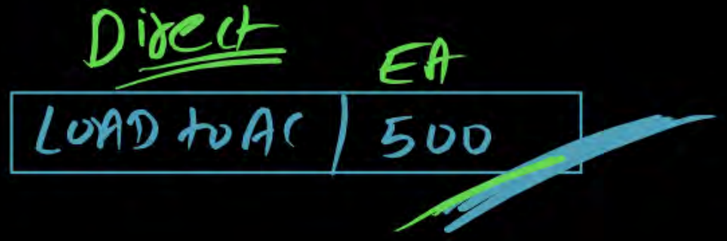
1. Immediate AM
2. Direct / Absolute AM
3. Mem. Indirect AM
4. Register Direct AM
5. Register Indirect AM
6. PC - Relative AM
7. Base - Register AM
8. Index Reg AM
9. Implied / Implict AM
10. Auto Decrement AM
11. Auto Increment AM.

Addressing Modes

- Immediate
- Direct
- Indirect
- Register
- Register Indirect
- Displacement
- Stack


Instruction
operand
(a) Immediate
Instruction
A
'EA'
Memory
operand
(b) Direct
Instruction
A
Add of EA
Memory
operand
EA
(c) Indirect
Instruction
R
name
operand
Registers
(d) Register
Instruction
R
Add of EA
Memory
EA
operand
Registers
(e) Register Indirect
Instruction
R
A
Memory
Registers
operand
EA
(f) Displacement
Instruction
Implicit
Top of stack
Register
(g) Stack





Q Consider a 16 bit Instruction 'LOAD to AC' with Address field = 500. Starts at Memory Location 200 Onwards. Opcode is 8 bit & Memory Byte Addressable & Following Details given below:

Identify [Calculate] EA & Operand using Various Am ?

R_1 Register

Addressing Mode	Effective Address	Content Of AC
(2) Direct address	500	800
(1) Immediate Operand	201	500
(3) Indirect Address	800	300
(6) Relative address	702	325
(7) Indexed address	600	900
(4) Register	R1	400
(5) Register Indirect	400	700
(8) Autoincrement	400	700
(9) Autodecrement	399	450

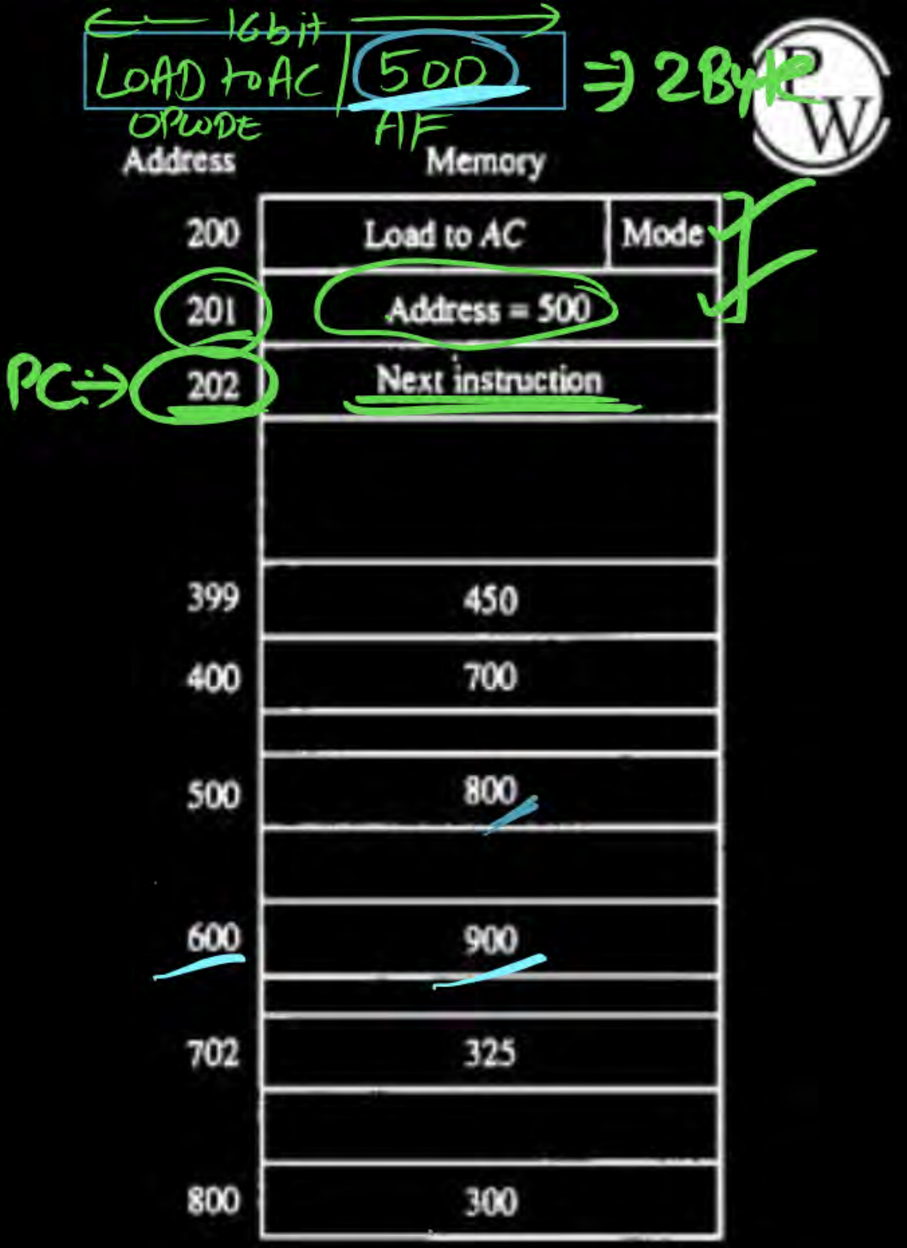
Operand

PC = 200

R1 = 400

XR = 100

AC



Numerical example for addressing modes.

PC Relative AM :

$$EA = \text{Current PC Value} + AF\ [OFFSET].$$

$$EA = 202 + 500$$

$$EA = 702$$

Index AM

$$EA = \text{Index Register (XR) Value} + A.F$$

$$100 + 500$$

$$EA = 600.$$

Auto Decrement & Increment AM: Similar to Register Indirect AM, in which Register Content (Data) Decrement or Increment.

Auto Decrement → Pre decrement

Auto Increment → Post Increment.

Pre decrement

400 ⇒ EA 399

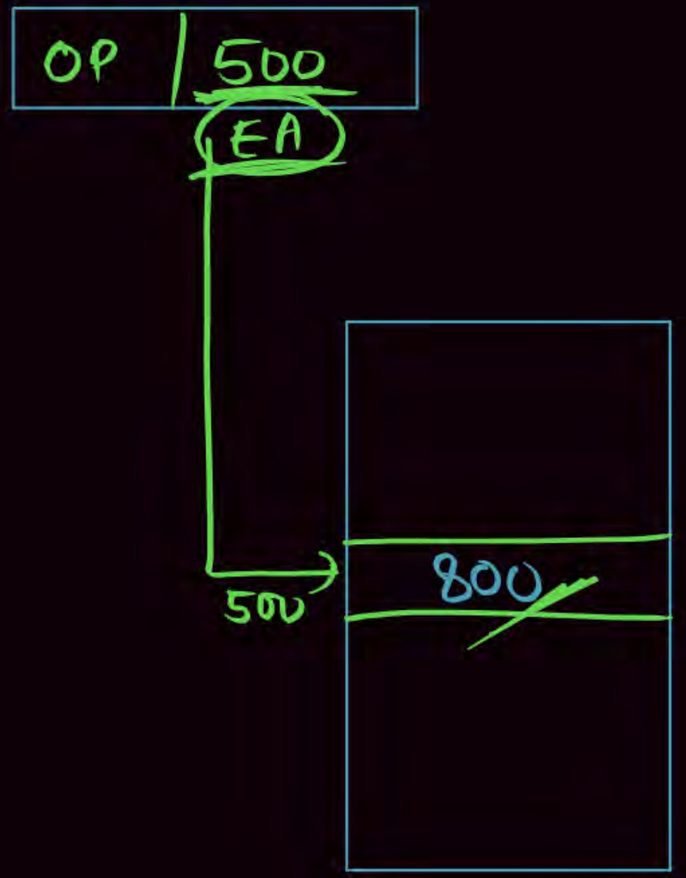
Increment: 400 ⇒ 400

Post Next Round

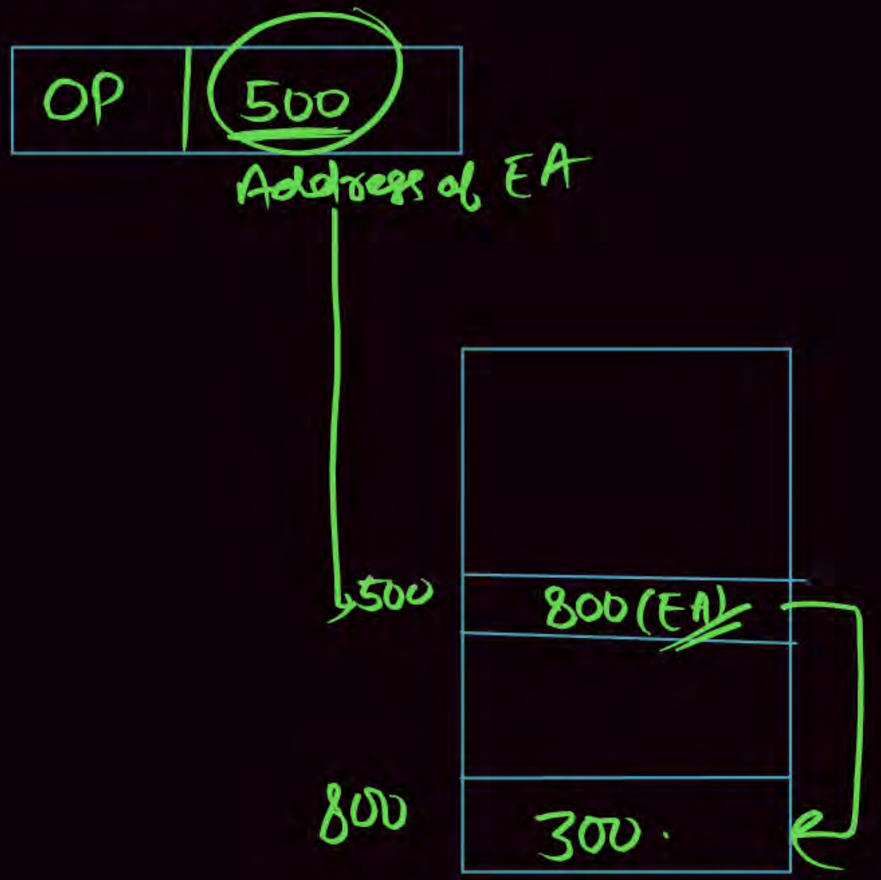
401

Direct AM

I_1:



Memory Indirect AM



Register Contain DATA

↑

Register Direct

EA

Reg Name (R_1)

Content / DATA (operand)

400

Register Indirect AM

R_1 Contain Effective Address

400

operand

700

Addressing Mode	Effective Address	Content Of AC
Direct address	500	800
Immediate Operand	201	500
Indirect Address	800	300
Relative address	702	325
Indexed address	600	900
Register	R1	400
Register Indirect	400	700
Autoincrement	400	700
Autodecrement	399	450

PC = 200

R1 = 400

XR = 100

AC

Address	Memory	
200	Load to AC	Mode
201	Address = 500	
202	Next instruction	
399	450	
400	700	
500	800	
600	900	
702	325	
800	300	

Numerical example for addressing modes.

Eight addressing modes for the load instruction

Mode	Assembly Convention	Register Transfer
Direct address	LD ADR	AC ← M[ADR]
Indirect address	LD @ADR	AC ← M[M[ADR]]
Relative address	LD $ADR	AC ← M[PC + ADR]
Immediate operand	LD#NBR	AC ← NBR
Index addressing	LD ADR(X)	AC ← M[ADR + XR]
Register	LD R1	AC ← R1
Register indirect	LD (R1)	AC ← M[R1]
Autoincrement	LD (R1)+	AC ← M[R1], R1 ← R1 + 1

Q. In which of the following addressing modes, operand is NOT A part of instruction? [MSQ]

A Immediate

B Direct

C Indirect

D Register

Q. Consider a 16 bit hypothetical processor which support 1 Address Instruction Design with various addressing mode. It contain 8 bit OPCODE. It supports 1 word Instruction stored in the Memory with a starting address of $(745)_{10}$(Decimal) onwards. Address field value of the instruction is 222. Register r_1 contain 111 memory content of [222] is 155. Which of the following is/are correct about effective address of various Addressing Mode (AM)? (all values are in decimal) [MSQ]

(I) In the Immediate AM Effective Address is 745.

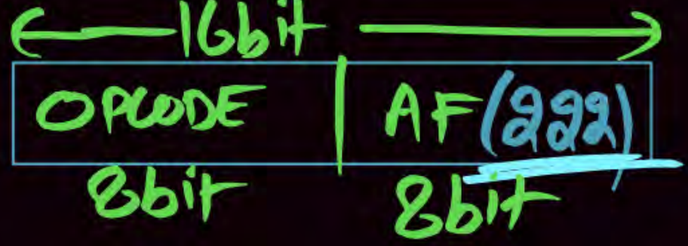
(II) In the Immediate AM Effective Address is 746.

(III) In the Memory Indirect AM Effective Address is 155.

(IV) In the Index AM effective Address is 333 (r_1 as index register)

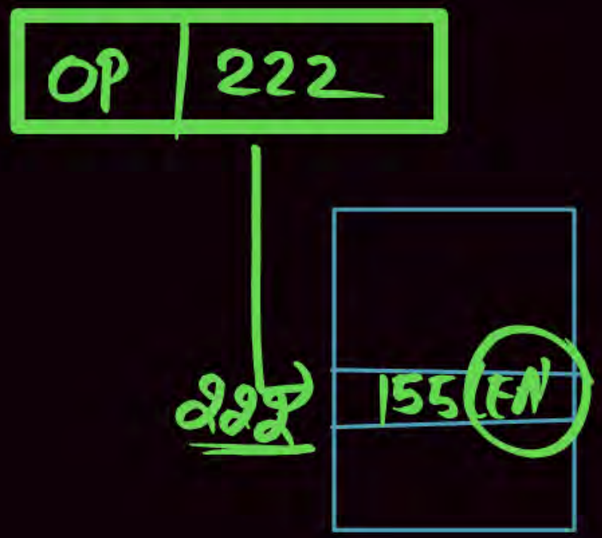
16bit Processor

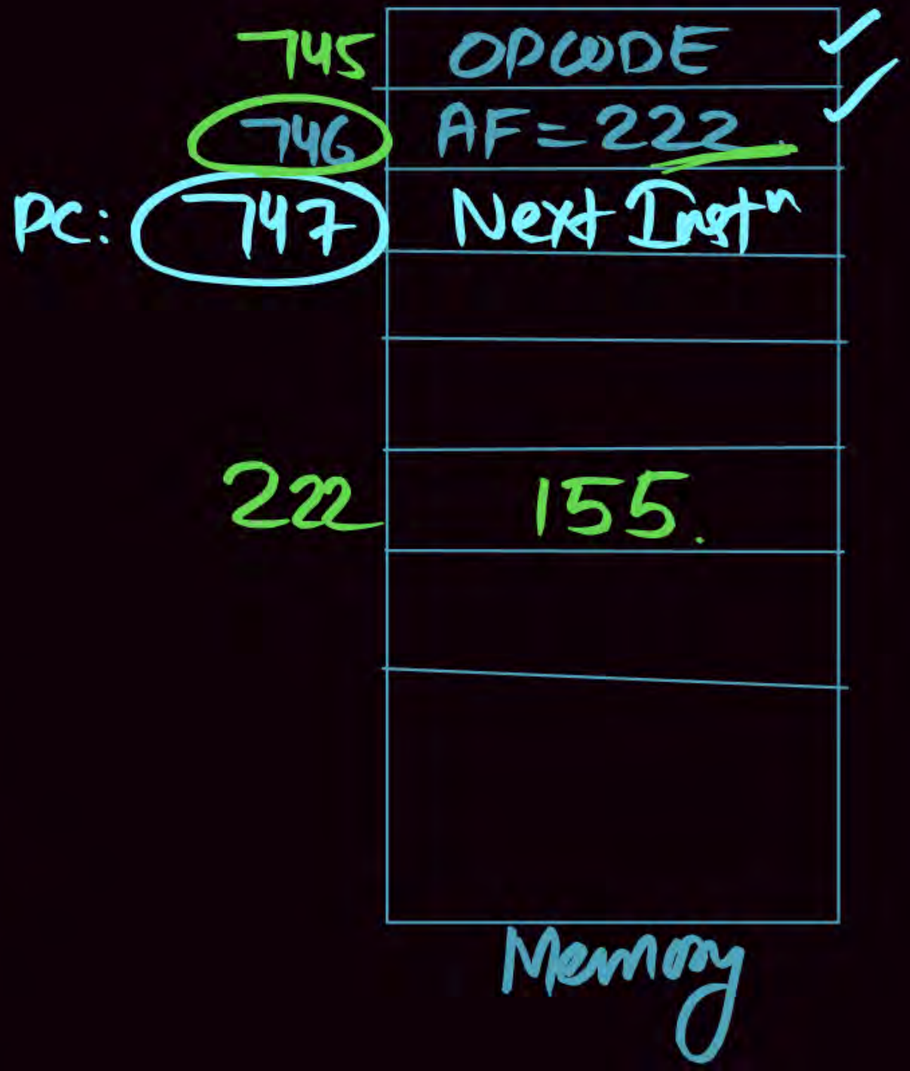
Instn Size = 1 word
= 16bit



Register (R_1) = 111

	EA
① Immediate AM :	746.
② Memory Indirect	
⇓	
EA = 155.	





$R_1 = 111$

Index AM

$$EA = \text{Index Reg Value}(R_1) + AF$$

$$111 + 222$$

$$\boxed{EA = 333.}$$ Ans

Q.1 The most appropriate matching for the following pairs

X. Indirect addressing - 2	1. Loops
Y. Immediate addressing -3	2. Pointers
Z. Auto decrement addressing -1	3. Constants

[GATE - 2000: 1 Mark]

(A) X – 3 Y – 2 Z – 1

(B) X – 1 Y – 3 Z–2

(C) X – 2 Y – 3 Z – 1

(D) X – 3 Y – 1 Z – 2

Indirect – Pointer

Immediate – Constant

Q.3 If we use internal data forwarding to speed up the performance of a CPU (R1, R2 and R3 are registers and M[100] is a memory reference), then the sequence of operations.

$R1 \rightarrow M[100]$
$M[100] \rightarrow R2$
$M[100] \rightarrow R3$

can be replaced by

[GATE - 2004: 2 Mark]

A
$R1 \rightarrow R3$
$R2 \rightarrow M[100]$

B
$M[100] \rightarrow R2$
$R1 \rightarrow R2$
$R1 \rightarrow R3$

C
$R1 \rightarrow M[100]$
$R2 \rightarrow R3$

D
$R1 \rightarrow R2$
$R1 \rightarrow R3$
$R1 \rightarrow M[100]$

$R_1 = 10$
$R_2 = 20$
$R_3 = 30$
$M[100] = 40$

$R_1 \rightarrow M[100]$
$M[100] \rightarrow R_2$
$M[100] \rightarrow R_3$

$\Rightarrow$

$R_1 = 10$
$M[100] = 10$
$R_2 = 10$
$R_3 = 10$

Ans

(a) $R_1 \rightarrow R_3$
$R_2 \rightarrow M[100]$

Register R_2 Not updated & $M[100]$ wrong Updation.

(b) $M[100] \rightarrow R_2$
$R_1 \rightarrow R_2$
$R_1 \rightarrow R_3$

$M[100]$ Not Update.

(c) $R_1 \rightarrow M[100]$
$R_2 \rightarrow R_3$

Register R_2 Not update & Register R_3 wrong Updation.

(d) $R_1 \rightarrow R_2$
$R_1 \rightarrow R_3$
$R_1 \rightarrow M[100]$

R_2, R_3 & $M[100]$ Correct Updation by R_1 content

Q.4 Match List-I with List-II and select the correct answer using the codes given below the lists:

[GATE - 2005: 2 Mark]

List-I	List-II
A. A[I] = B[J]; -2	1. Indirect addressing
B. while[*A++];-3	2. Indexed addressing
C. int temp = *X; -1	3. Auto increment

Codes:

	A	B	C
A	3	2	1
B	1	3	2
C	2	3	1
D	1	2	3

A - Index

B: Increment

C: Indirect

Q.5 The memory locations 1000, 1001 and 1020 have data values 18, 1 and 16 respectively before the following program is executed. [GATE - 2006: 2 Mark]

I_1 : MOVI	Rs, 1	Move immediate
I_2 : LOAD	Rd, 1000(Rs)	Load from memory
I_3 : ADD I	Rd, 1000	Add immediate
I_4 : STOREI	0(Rd), 20	Store immediate

Which of the statements below is TRUE after the program is executed?

A. Memory location 1000 has value 20

B. Memory location 1020 has value 20

C. Memory location 1021 has value 20

D. Memory location 1001 has value 20

m[1000] = 18
m[1001] = 1
m[1020] = 16

I_1: MOVI R_s, 1;

$R_s = 1$

I_2: LOAD R_d, 1000(R_s)

$R_d \in m[1000 + R_s]$

$m[1000 + 1]$

$R_d \in m[1001]$

$R_d = 1$

I_3: ADDI R_d, 1000

$R_d \in R_d + 1000$

$\in 1 + 1000$

$R_d = 1001$

I_4: STOREI 0(R_d), 20.

$m[0 + R_d] \in 20$

$m[0 + 1001] \in 20$

$m[1001] \in 20$. Ans

Direct AM.

Q.7 The absolute addressing mode [GATE - 2002: 1 Mark]

(A) The operand is inside the instruction

(B) The address of the operand is inside the instruction

(C) The register containing the address of the operand is specified inside the instruction.

(D) The location of the operand is implicit.

Q. 8 A CPU has 24-bit instructions. A program starts at address 300 (in decimal). Which one of the following is a legal program counter (all values in decimal)?

[GATE-1 Marks]

(a) 400 (b) 500 (c) 600 (d) 700

24 bit Instn = 3 Byte.

$300 + 3x = 600$

$3x = 300$

$x = 300/3 = 100$ ✓

$300 + 3x = 400$

$3x = 100$

$x = 100/3 = 33.3$

$300 + 3x = 700$

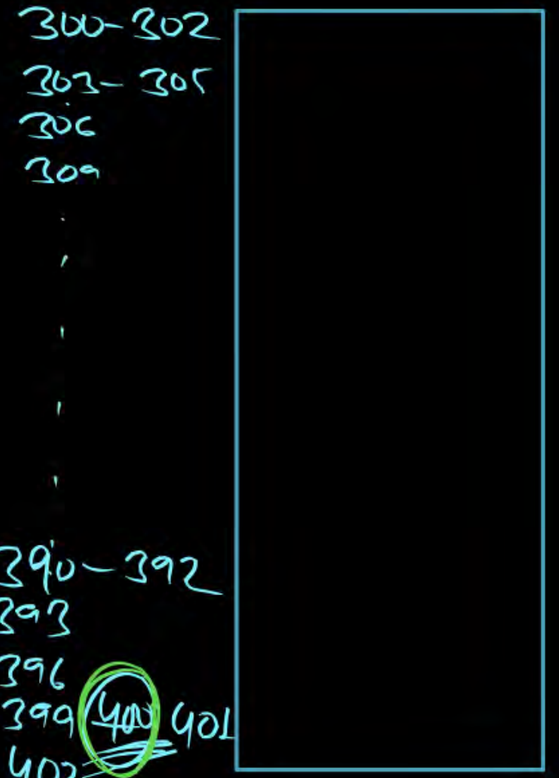
$3x = 400$

$x = 400/3 = 133.3$

$300 + 3x = 500$

$3x = 200$

$x = 200/3 = 66.6$



COMMON DATA QUESTION (9 -10)

M[3000] = 10
R_3 = 2000.

Consider the following program segment, Here R1, R2 and R3 are the general purpose register.

Instruction	Operation	Instruction size (no. of words)
MOV R1, (3000)	R1←M[3000]	2 $R_1 = 10$
LOOP;		
MOV R2, M[R3] ADD R2, R1	R2←M[R3] R2←R1 + R2	1 $R_2 \leftarrow M[R_3]$: $R_2 \leftarrow M[2000]$ $R_2 = 100$ 1 $R_2 = 10 + 100 = 110$
MOV (R3), R2	M[R3] ←R2	1 M[2000] = 110.
INC R3	R3←R3+1	1 $R_3 = 2000 + 1 = 2001 = R_3$
DEC R1	R1←R1-1	1 $R_1 = 10 - 1 = R_1 = 9$
BNZ LOOP	Branch on not zero	2
HALT		Stop

$R_2 \leftarrow M[2001]$
$R_2 = 100$
$R_2 = 9 + 100 \Rightarrow R_2 = 109$
M[2001] = 109
$R_3 = 2001 + 1 \Rightarrow R_3 = 2002$
$R_1 = 9 - 1 \Rightarrow R_1 = 8$

Assume that the content of memory location 3000 is 10 and the content of the Register R3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the Numbers are in decimal.

COMMON DATA QUESTION (9 -10)

Consider the following program segment, Here R1, R2 and R3 are the general purpose register.

Instruction	Operation	Instruction size (no. of words)
MOV R1, (3000)	R1←M[3000]	2
LOOP;		
MOV R2, M[R3]	R2←M[R3]	1
ADD R2, R1	R2←R1 + R2	1
MOV (R3), R2	M[R3] ←R2	1
INC R3	R3←R3+1	1
DEC R1	R1←R1-1	1
BNZ LOOP	Branch on not zero	2
HALT		Stop

Increment

Decrement

$R_2 \leftarrow M[2002] \Rightarrow R_2 = 100$

$R_2 = 8 + 100 = 108$

$R_2 = 108$

$M[2002] = 108$

$R_3 = 2002 + 1 \Rightarrow R_3 = 2003$

$R_1 = 8 - 1 \Rightarrow R_1 = 7$

$R_2 \leftarrow M[2003]$

$R_2 = 100$

$R_2 \leftarrow R_1 + R_2 = 7 + 100$

$R_2 = 107$

$M[2003] = 107$

$R_3 \leftarrow 2003 + 1 \Rightarrow R_3 = 2004$

$R_1 = 7 - 1 = R_1 = 6$

Assume that the content of memory location 3000 is 10 and the content of the Register R3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the Numbers are in decimal.

$M[3000] = 10$

$R_3 = 2000$

2000	~~100~~ 110
2001	~~100~~ 109
2002	~~100~~ 108
2003	~~100~~ 107
2004	100 106
2005	100 105
2006	100 104
2007	100 103
2008	100 102
2009	100 101
2010	100

(2) $M[2010] = 100$ Ans

(1) Memory Reference

Total 10 Iteration

In Each Iteration: 2 Mem Ref

In the beginning → (1) + 2*10

= 21 Mem Ref

Q.9 Assume that the memory is word addressable. The number of memory references for accessing the data in executing the program completely is

[2 marks]

(a) 10 (b) 11 (c) 20 (d) 21

COMMON DATA QUESTION (9 - 10)

Consider the following program segment, Here R1, R2 and R3 are the general purpose register.

Instruction	Operation	Instruction size (no. of words)
MOV R1, (3000)	R1←M[3000]	2
LOOP:		
MOV R2, M[R3] ADD R2, R1	R2←M[R3] R2←R1 + R2	1 1
MOV (R3), R2	M[R3] ←R2	1
INC R3	R3←R3 + 1	1
DEC R1	R1←R1 - 1	1
BNZ LOOP	Branch on not zero	2
HALT		Stop

Assume that the content of memory location 3000 is 10 and the content of the Register R3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the Numbers are in decimal.

Q.10 Assume that the memory is word addressable. After the execution of this program, the content of memory location 2010 is

[2 marks]

(a) 100 (b) 101 (c) 102 (d) 110

① Auto Decrement / Auto Increment AM.

② Index Reg AM

③ PC - Relative AM

④ Base Register AM.

(1) Auto Decrement & Increment AM: Similar to Register Indirect AM, in which

Register Content (Data) Decrement (or) Increment.
to Access the Data Sequentially.

Auto Decrement → Pre decrement

Auto Increment → Post Increment.

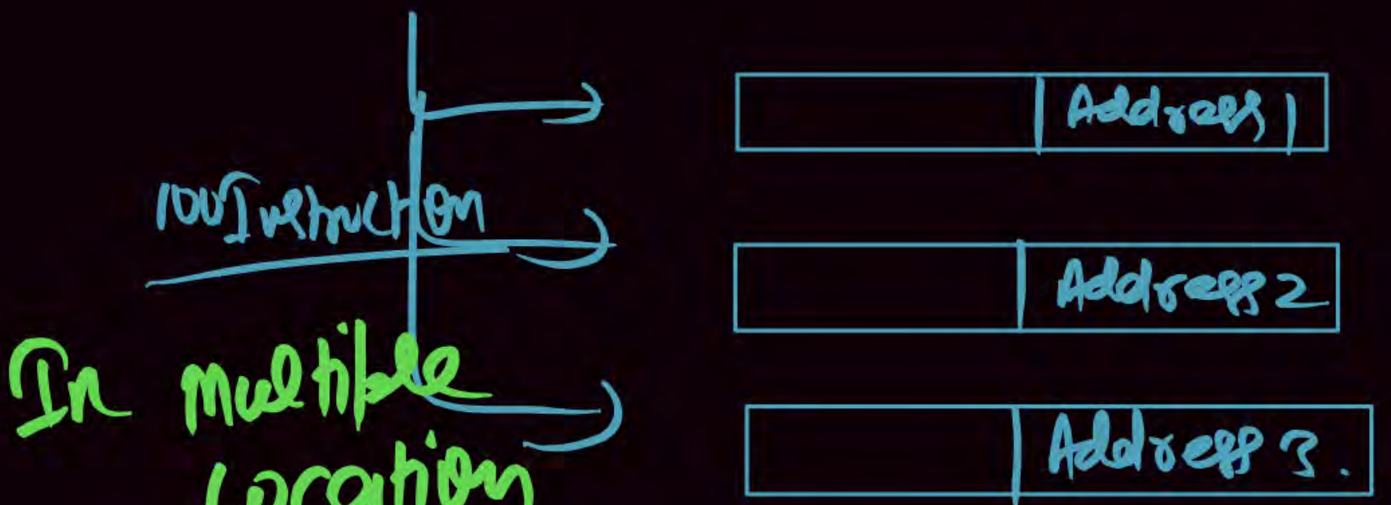
Q) Why Auto Decrement, Auto Increment AM Used : ?

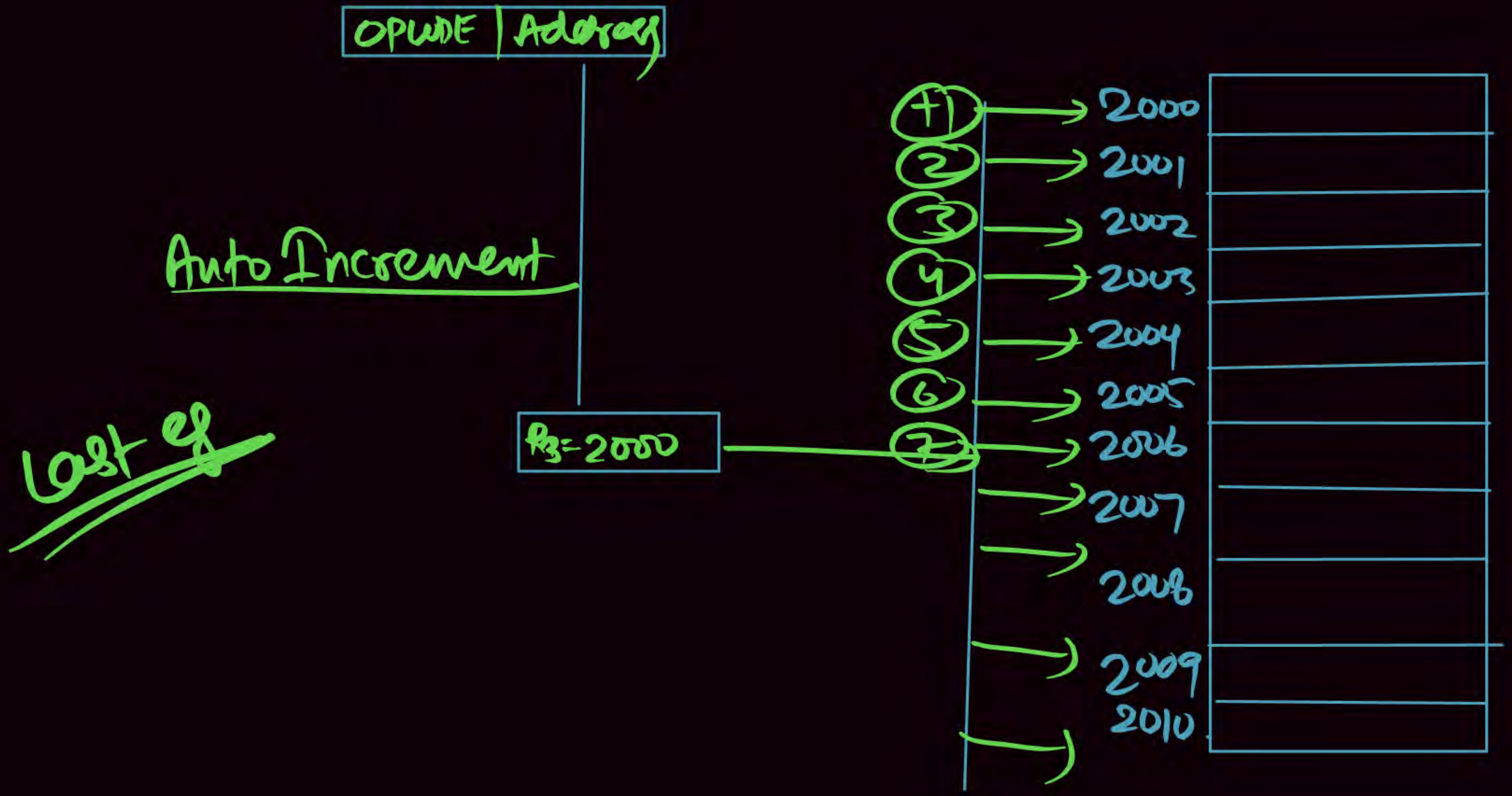
Soln) When we want to Access / Perform Same operation in Multiple Location (Assume 100 location) this Different - Different Instruction are Required.

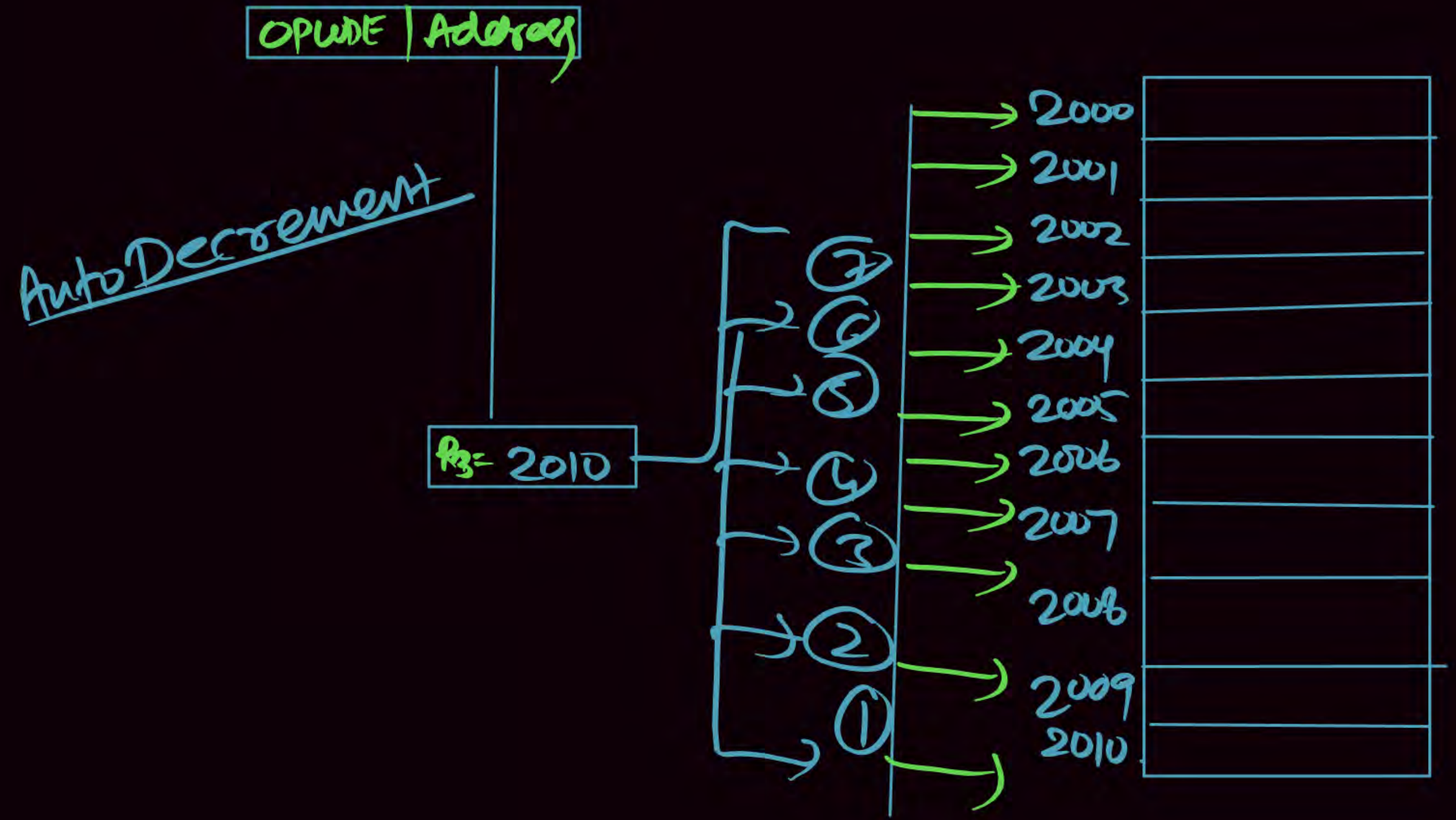
- But with the Help of Auto Decrement (or) Auto Increment AM it is possible with Only One Instruction we can Perform Same Operation (or) Multiple Locations.

HOW ?

for (i=1; i<=100; i++)


100 Instruction
In multiple location
Same operation
Address 1
Address 2
Address 3.


OPCODE | Address
Auto Increment
Last eg
R3=2000
1
2
3
4
5
6
7
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010


OPCODE
Address
Auto Decrement
R3 = 2010
7
6
5
4
3
2
1
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010

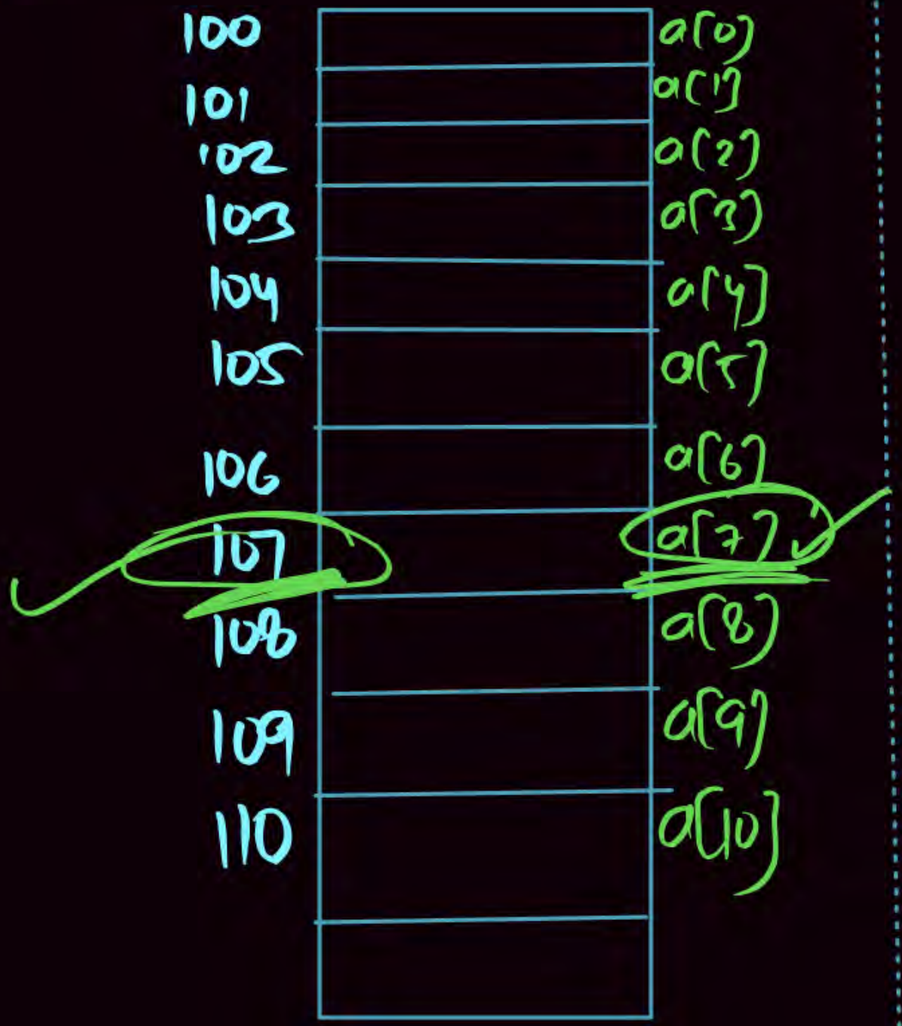
(2) Index Addressing Mode:

This AM is used to Implement the Array.

This AM is used to Access Specific (Random) element of the Array.

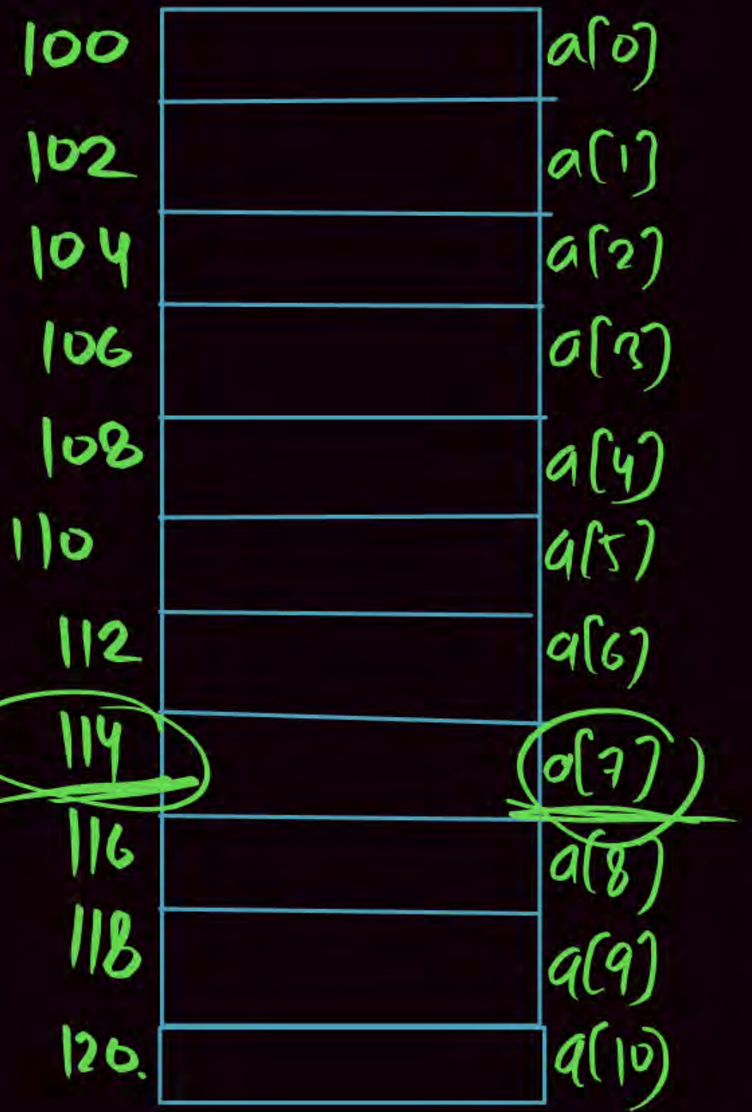
Char = 1 Byte

Char a[10].



int a[10]. int = 2 Byte

Assume starts from 100.



A[n] = Starting address (Base address) + n * Size of Datatype → Index Value

Char

a[7] = 100 + (7 * 1) → Index Value

a[7] = 107

Integer

a[7] = 100 + (7 * 2) → Index Value

= 100 + 14

a[7] = 114

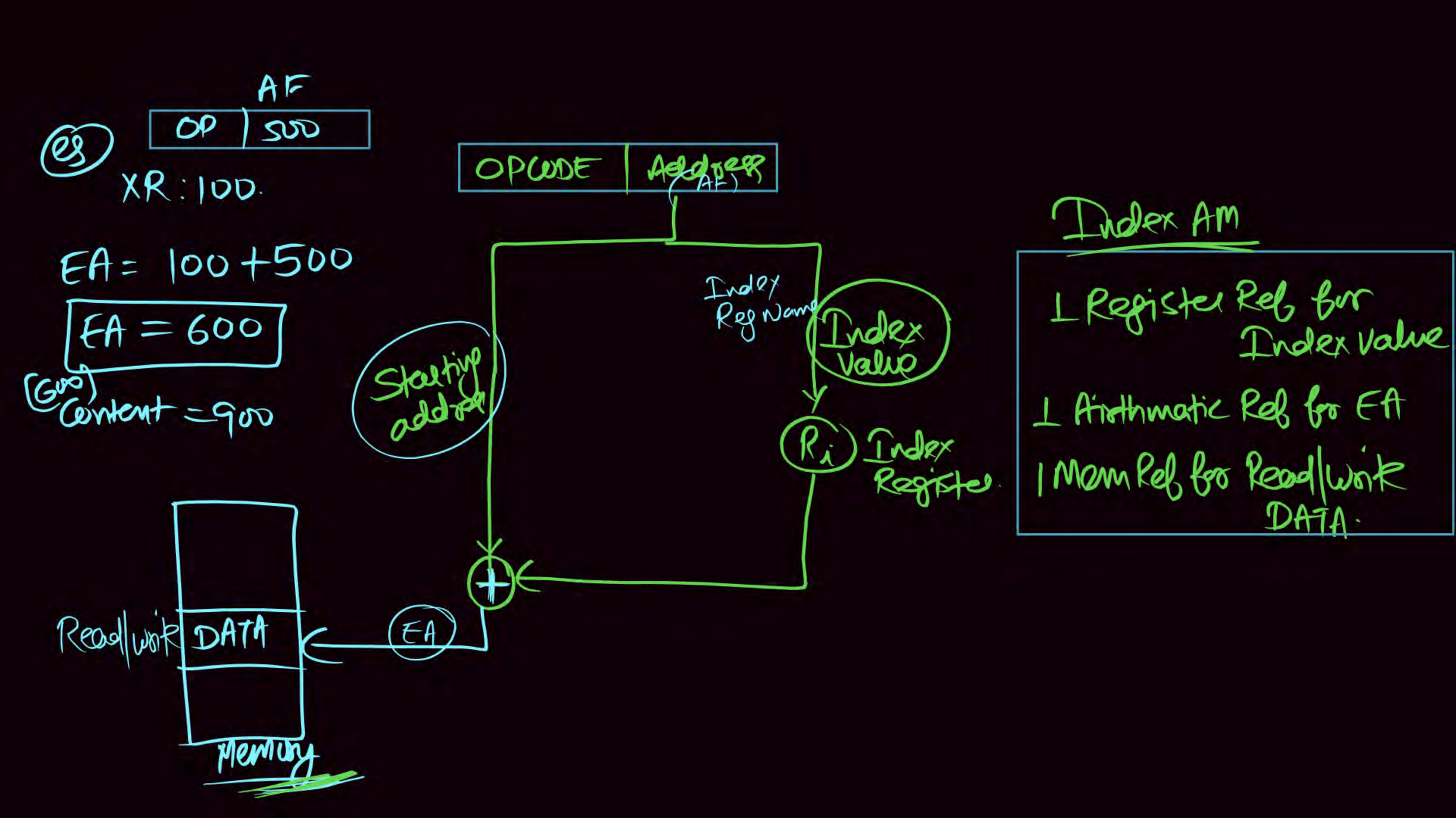
Stored in Index Register.

Note

① Index Value Stored in Index Register.

② Starting address Present in the Address field of the Instruction.

$$EA = \text{Index Reg Value} + AF$$


AF
OP
500
eg
XR : 100.
EA = 100 + 500
EA = 600
Content = 900
OPCODE
Address (AF)
Index Reg Name
Index Value
Starting address
Ri
Index Register
Read/write
DATA
EA
Memory
Index AM
1 Register Ref for Index value
1 Arithmatic Ref for EA
1 Mem Ref for Read/write DATA.

Index Register

① At the CPU Design time One Special Register Made as Index Register. than in this they implict Access the Index Value from the Index Register. (Wastage of One Register Bcz Index Register use very-2less)

② In Most of the Case One General Purpose Register Made [Designated] as Index Register. & mentioned that Reg Name as Index Register. [eg R_0 as a Index Register]

Advantage: we Can use that Register as General Purpose Register also.

PC Relative AM

EA = Current PC Value + Address field value(Relative Value)

In this Mode Effective Address(EA) is obtained by adding the Relative Value to Program Counter(PC)

Relative Value means distance between current location to target location . It is a Constant(Signed Constant), present in the address field of the instruction.

PC Relative AM is used intra segment transfer of control(branching), when target address is present in same segment then during program execution control will be transferred with in the segment called intra segment branching.

Base Register AM

EA = Base register Value + Address field value

Base Register AM is used inter segment transfer of control(branching), when target address is present in different segment then during program execution control will be transferred between the segment called inter segment branching.

Note: Both PC relative AM & Base Register AM are suitable for program reallocation at run time.

Q.

Consider a 4 Byte long pc relative instruction is located in the memory with the starting address of 243048(Decimal). A -8 sign Displacement is present in the address field of the instruction . What is the branch address(Target address)?

Q.

Consider a 4 Byte long Jump instruction stored in the word addressable memory with a word size of 16bits. The starting address of instruction is 900(Decimal). A address field contain ---32. content of base register is 500. What is the branch address(Target address) when the instruction is designed with

(i) PC Relative Addressing Modes

(ii) Base Register Addressing Modes

Q.

Consider a 16bits which support 1 word long instruction, stored in the memory with a starting address of 900(Decimal). Instruction format contain 8bit Opcode & Address field. Instruction is designed with PC Relative JMP Operation.

During its execution control(Branch) will be transferred to an address 614(Decimal) then What is

(i) What is the Relative Value in Address field of the instruction?

(ii) What is the PC Value before instruction fetch, after instruction fetch and after Execution phase?

NAT Q.11

Consider the following instruction sequence where registers R1, R2 and R3 are general purpose and MEMORY [X] denotes the content at the memory location X.

Instruction	Semanties	Instruction Size (bytes)
MOV R1, (5000)	R1 ← MEMORY [5000] $R_1 = 10$	4
MOV R2, (R3)	R2 ← MEMORY [R3] $R_2 \leftarrow M[3000] \Rightarrow R_2 = 50$	4 $R_2 \leftarrow M[3001] = 50$
ADD R2, R1	R2 ← R1 + R2 $R_2 = 10 + 50 = 60$	2 $R_2 = 9 + 50 = 59$
MOV (R3), R2	MEMORY [R3] ← R2 $M[3000] = 60$	4 $M[3001] = 59$
INC R3	R3 ← R3 + 1 $R_3 = 3000 + 1 \Rightarrow R_3 = 3001$	2 $R_3 = 3001 + 1 = R_3 = 3002$
DEC R1	R1 ← R1 - 1 $R_1 = 10 - 1 = 9$	2 $R_1 = 9 - 1 \Rightarrow R_1 = 8$
BNZ 1004	Branch if not zero to the given absolute address	2
HALT	Stop	1

1000-1005
1004-1005

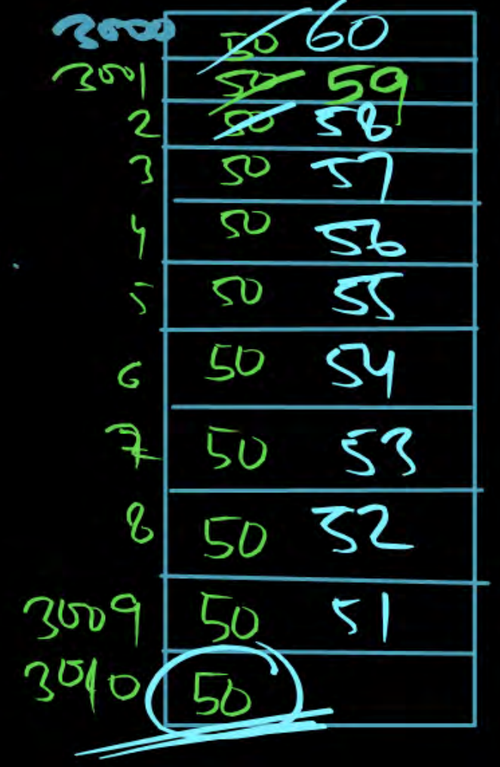
Assume that the content of the memory location 5000 is 10, and the content of the register R3 is 3000. The content of each of the memory locations from 3000 to 3010 is 50. The instruction sequence starts from the memory location 1000. All the numbers are in decimal format. Assume that the memory is byte addressable.

After the execution of the program, the content of memory location 3010 is 50 Ans

[GATE-2021(Set-1)-CS: 2M]

M(5000) = 10

R3 = 3000.



NAT **Q.12**

Consider a RISC machine where each instruction is exactly 4 bytes long. Conditional and unconditional branch instructions use PC-relative addressing mode with Offset specified in bytes to the target location of the branch instruction. Further the Offset is always with respect to the address of the next instruction in the program sequence. Consider the following instruction sequence

Instr. No	Instruction
i	add R2, R3, R4
i +1	sub R5, R6, R7
i + 2	cmp R1, R9, R10
i + 3	beq R1, Offset

If the target of the branch instruction is i, then the decimal value of the Offset is _____.

[GATE-2017(Set-1)-CS: 2M]

MCQ Q.13

For computers based on three-address instruction formats, each address field can be used to specify which of the following:

S1: A memory operand

S2: A processor register

S3: An implied accumulator register

[GATE-2015(Set-1)-CS: 1M]

A Either S1 or S2

B Either S2 or S3

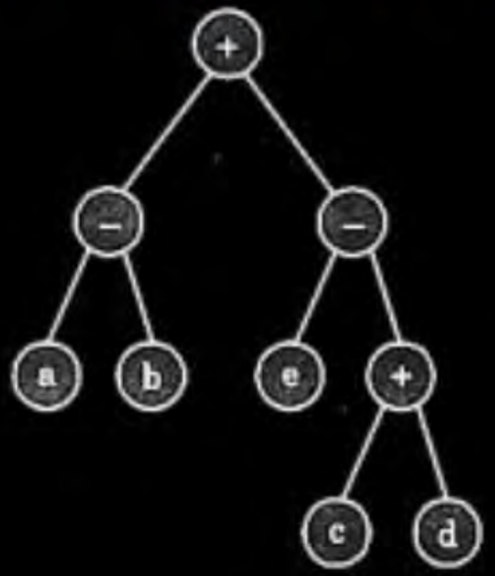
C Only S2 and S3

D All of S1, S2 and S3

MCQ **Q.14**

Consider evaluating the following expression tree on a machine with load-store architecture in which memory can be accessed only through load and store instructions. The variables a, b, c, d and e are initially stored in memory. The binary operators used in this expression tree can be evaluated by the machine only when the operands are in registers. The instructions produce result only in a register. If no intermediate results can be stored in memory, what is the minimum number of registers needed to evaluate this expression?

[GATE-2011-CS: 2M]



A 2

B 9

C 5

D 3

MCQ Q.15

The program below uses six temporary variables
a, b, c, d, e, f.

```
a = 1
b = 10
c = 20
d = a + b
e = c + d
f = c + e
b = c + e
e = b + f
d= 5 + e
return d + f
```

Assuming that all operations take their operands from registers, what is the minimum number of registers needed to execute this program without spilling?

[GATE-2010-CS: 2M]

(A) 2

(B) 3

(C) 4

(D) 6

MCQ **Q.16**

Assume that EA = (X)+ is the effective address equal to the contents of location X, with X Incremented by one word length after the effective address is calculated; EA =-(X) is the effective address equal to the contents of location X, with X decremented by one word length before the effective address is calculated; EA = (X)- is the effective address equal to the contents of location X, with X decremented by one word length after the effective address is calculated. The format of the instruction is (opcode, source, destination), which means (destination ← source op destination). Using X as a stack pointer, which of the following instructions can pop the top two elements from the stack, perform the addition operation and push the result back to the stack.

[GATE-2008-CS: 1M]

A ADD (X)-, (X)

B ADD (X), (X)-

C ADD –(X), (X)+

D ADD –(X), (X)

THANK YOU!