COMPUTER SCIENCE

Computer Organization and Architecture

Machine Instruction and Addressing Modes

Addressing Modes-3

Lecture_07

Vijay Agarwal sir

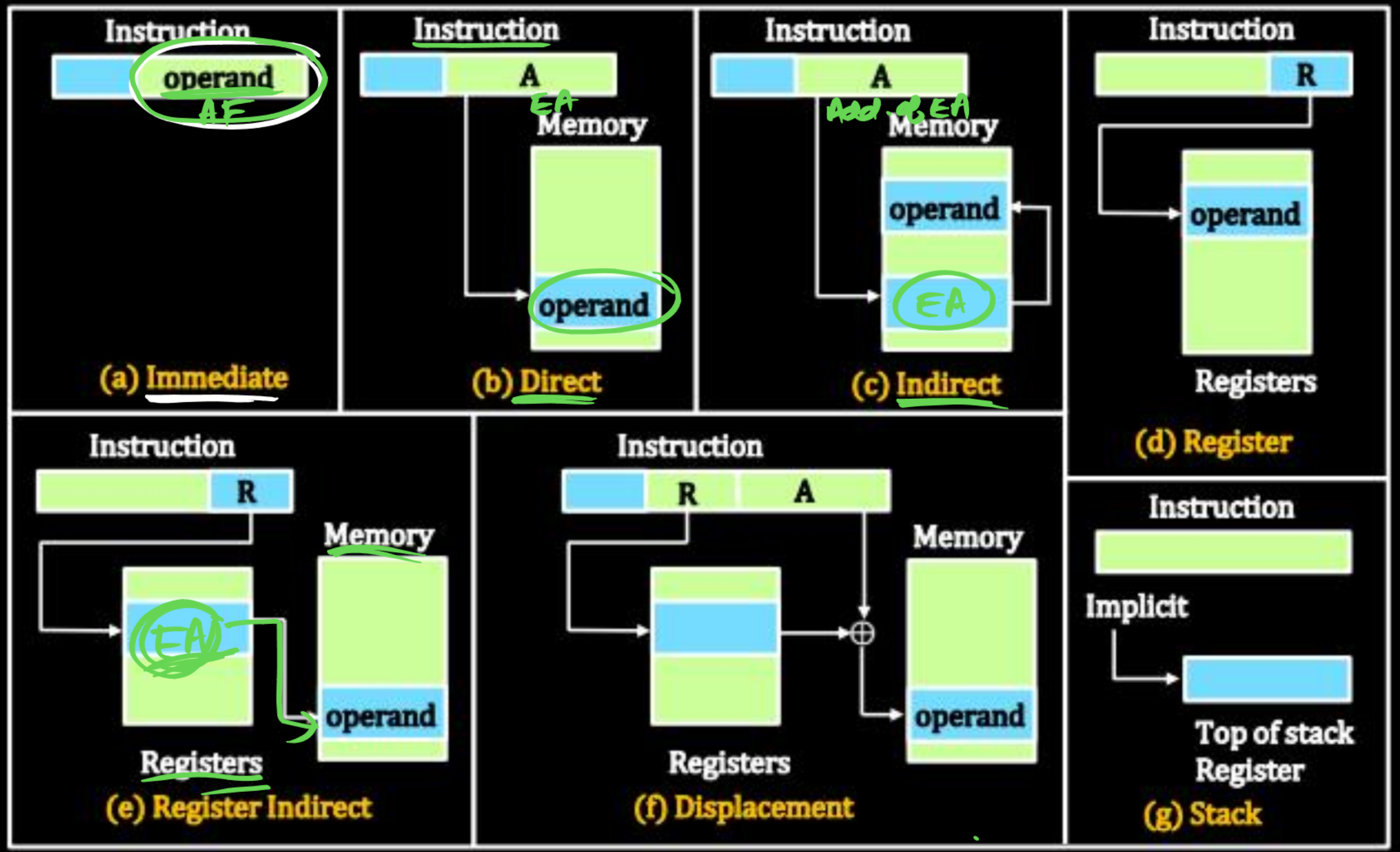
01 Addressing Modes

Addressing Mode.

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

Addressing Modes

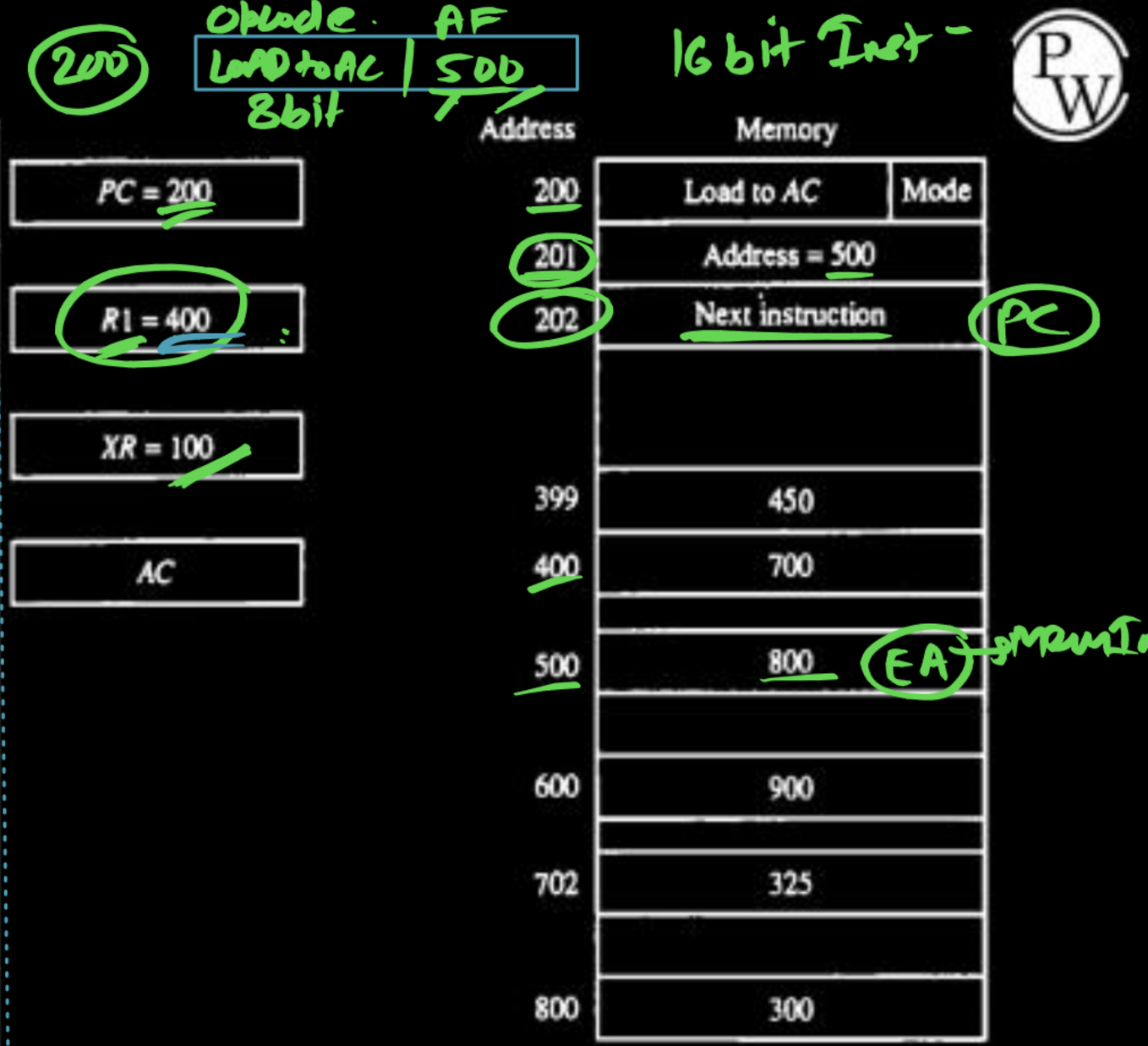
- [x] Immediate
- [x] Direct
- [x] Indirect
- [x] Register
- [x] Register Indirect
- [x] Displacement
- [x] Stack


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

Q 16 bit Instⁿ LOAD to AC with Address = 500
Start from Location 200, 8 bit Opcode. With
the following Details

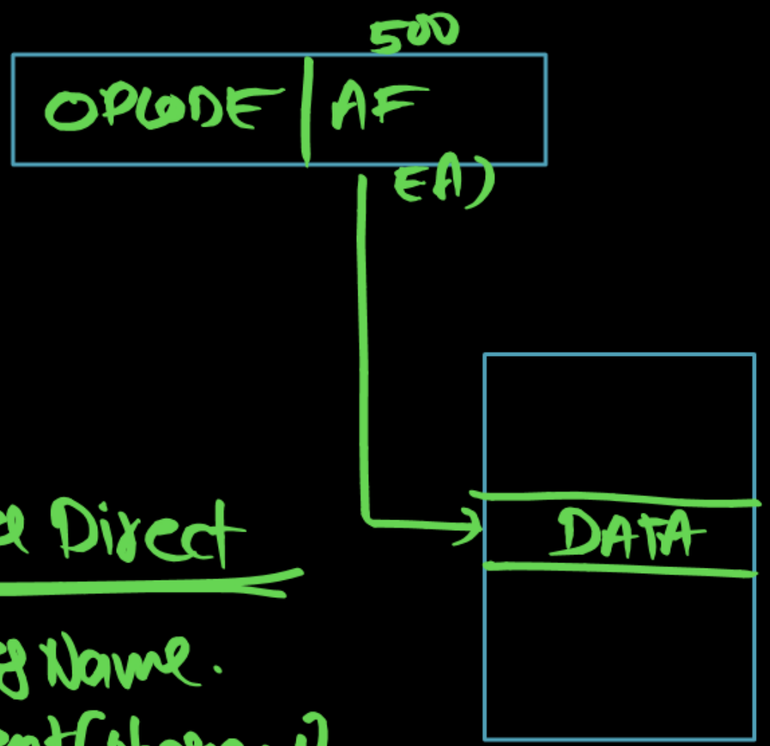
XR : Index Register.

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	R_1	400
Register Indirect	400	700
Autoincrement	400	700
Autodecrement	399	450



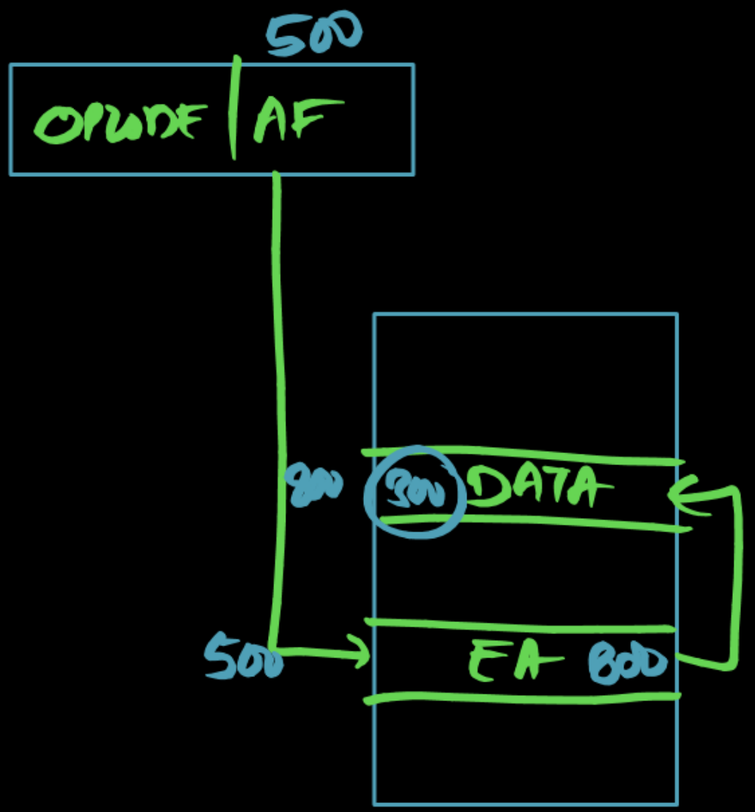
Numerical example for addressing modes.

Direct | Absolute AM



Register Direct

EA = Reg Name.
Content (operand) in the Register



Auto Decrement / Auto Increment AM is a similar to Register Indirect in which Value [Content] of Register is automatically Decrement / Increment.

Decrement – Pre Decrement

Increment → Post Increment

Index Reg AM

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

$$= 100 + 500$$

$$\boxed{EA = 600}$$

PC - Relative AM

$$EA = \underset{\text{Value}}{PC} + AF$$

$$= 202 + 500$$

$$\boxed{EA = 702}$$

Sol^n

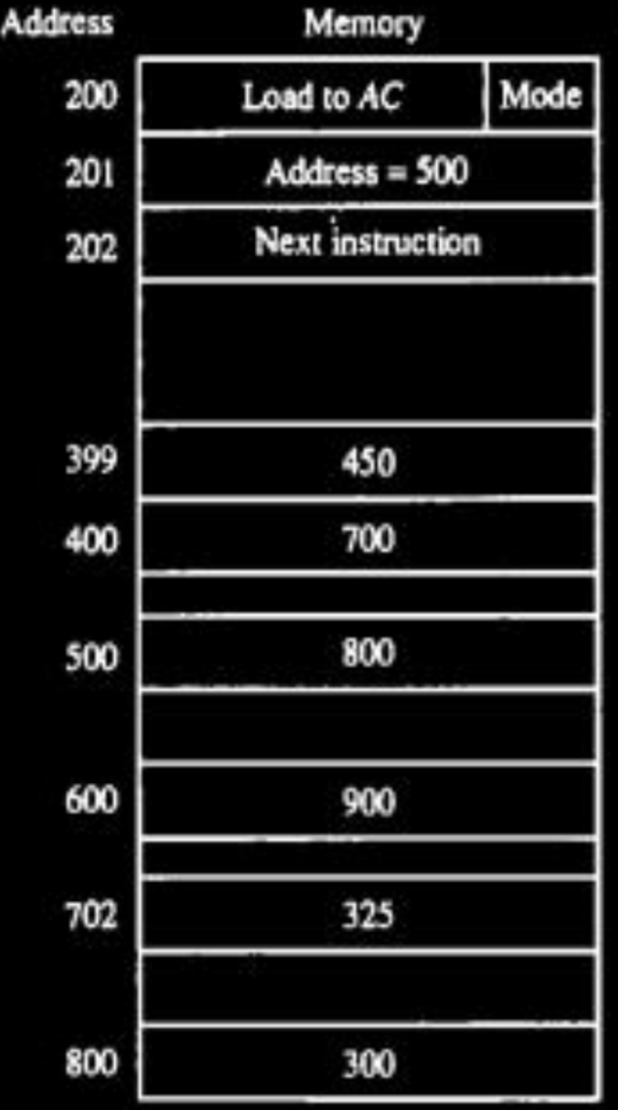
Addressing Mode	Effective Address	Content Of AC
Direct address	500	800
Immediate Operand	201	500
Indirect Address	800	300
Relative address	702	325
Indexed address	600	900
Register	--	400
Register Indirect	400	700
Autoincrement	400	700
Autodecrement	399	450

PC = 200

*R*1 = 400

XR = 100

AC



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

(A) Immediate

(B) Direct

(C) Indirect

(D) Register

Immediate AM

'DATA' (operand)

Ans (B, C & D).

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.

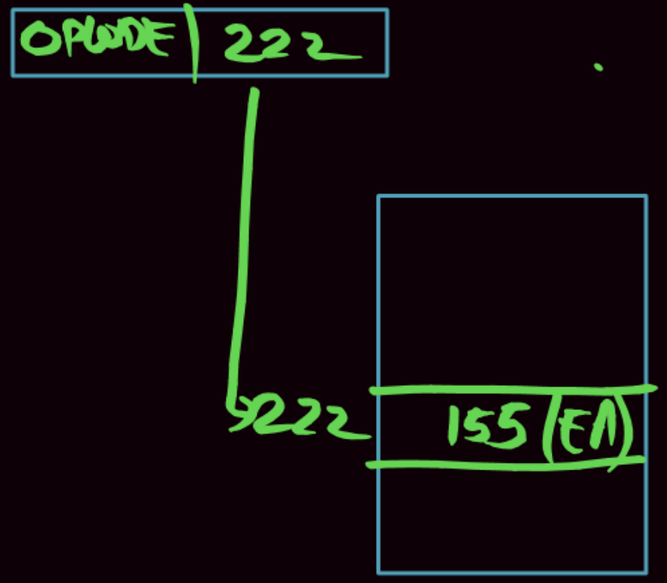
(IV) In the Index AM effective Address is 333
(r_1 as index register)

Ans (B, C & D)

① Immediate AM

EA = 746

(ii) Memory Indirect



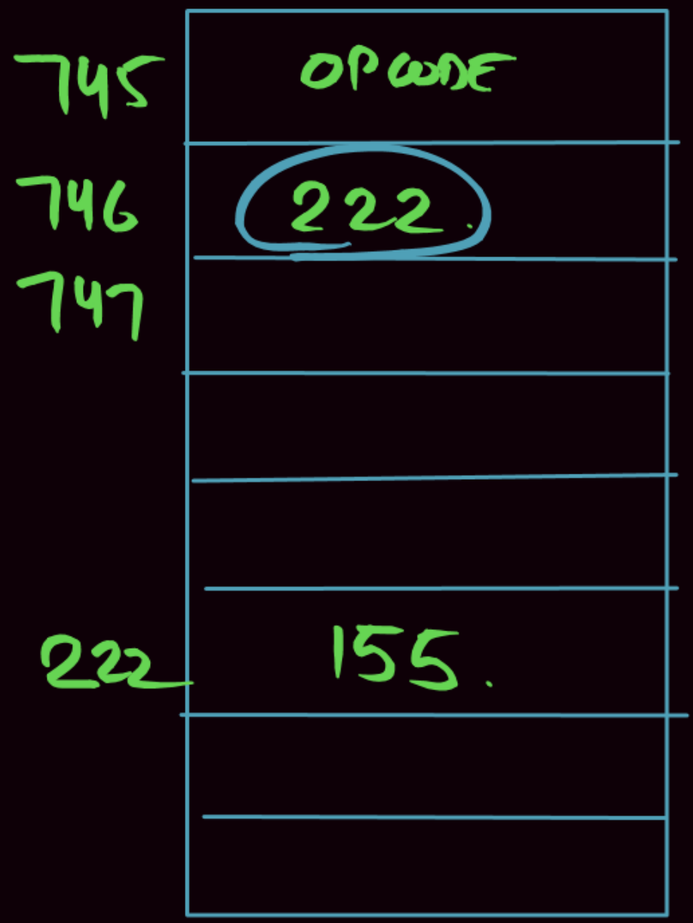
$R_1 = 111$

Index AM

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

$= 222 + 111$

EA = 333



Q.1 The most appropriate matching for the following pairs

X. Indirect addressing	1. Loops
Y. Immediate addressing	2. Pointers
Z. Auto decrement addressing	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 AM – Pointer.
Immediate AM – Constant
Auto Decr. AM – Loops

X - 2 Y - 3 Z - 1

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

Ans (D).

eg $R_1 = 10$

$R_2 = 10$.
$R_3 = 10$
$M(100) = 10$

Eg

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

Assume

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

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

Ans

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

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

Here R_2 Not update
& $M[100]$ Wrong Updation

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

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

$M[100]$ Not updated.

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

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

Register R_2 Not Update
& R_3 Wrong Updation.

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

Codes:

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

A B C

2 3 1

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

Ans (D).

M(1000) = 18

M(1001) = 1

M(1020) = 16

I_1 : MOVI Rs, 1 → Rs = 1

I_2 LOAD Rd, 1000(Rs)

Rd ← M(1000 + Rs)

M(1000 + 1)

Rd ← M(1001) → Rd = 1

I_3 : ADDI Rd, 1000

Rd ← Rd + 1000

1 + 1000

Rd = 1001

I_4 STOREI 0(Rd), 20

M(0 + Rd) ← 20

M[0 + 1001] ← 20

M[1001] = 20 Ans

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

24bit = 3Byte

(a) $300 + 3x = 400$

$3x = 100$

$x = 100/3 = 33.3$ ✗

(b) $300 + 3x = 500$

$3x = 500 - 300$

$x = 200/3 = 66.6$

(c) $300 + 3x = 600$

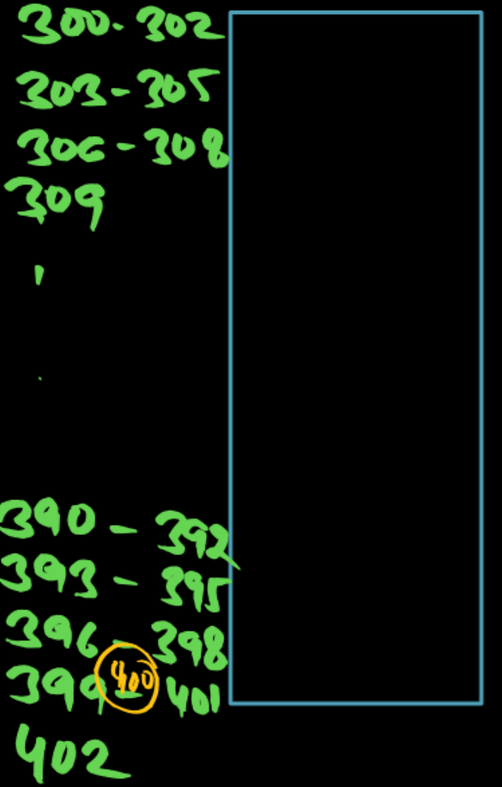
$3x = 300$

$x = 300/3 = 100$ ✓

(d) $300 + 3x = 700$

$3x = 400$

$x = 400/3 = 133.3$ ✗



COMMON DATA QUESTION (9 -10)

$R_3 = 2000$
$M(3000) = 10$

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] $R_1 = 10$	2
LOOP;		
MOV R2, M[R3]	R2←M[R3] $R_2 \in M(2000)$, $R_2 = 100$	1
ADD R2, R1	R2←R1 + R2 $R_2 = 10+100 = 110$, $R_2 = 110$	
MOV (R3), R2	M[R3] ←R2 $M(2000) = 110$	1 $M(2001) = 109$
INC R3	R3←R3+1 $R_3 = 2000+1$, $R_3 = 2001$	1; $R_3 \in R_3+1 < 2001+1 \Rightarrow R_3 = 2002$
DEC R1	R1←R1-1 $R_1 = 10-1 = 9$, $R_1 = 9$	1 $R_1 < R_1-1 \Rightarrow 9-1 = R_1 < 8$
BNZ LOOP	Branch on not	2
HALT		Stop

$R_2 \in M(R_3) \Rightarrow R_2 \in M[2000]$

$R_2 = 100$

$R_2 \in M(R_3)$; $R_2 \in M[2001]$, $R_2 = 100$

$R_2 \in R_1 + R_2 \Rightarrow 8 + 100$, $R_2 = 109$

$R_2 = 108$

$M(R_3) \in R_2 \Rightarrow$

$M(2002) = 108$

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

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

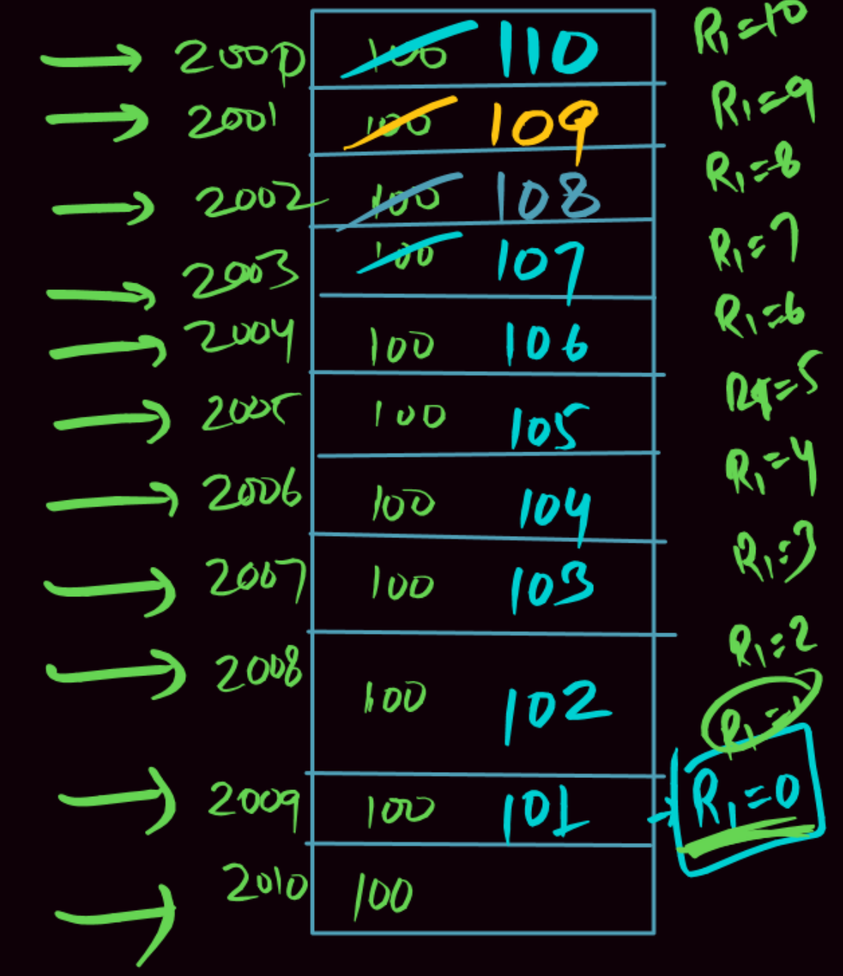
Assume that the content of memory location 3000 is 10 and the content of the Register R3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the Numbers are in decimal.

M[3000] = 10
R_3 = 2000

Stall 10

M[2010] Content Not UPDATE

∵ at M(2009) ⇒ $R_1 = 0$



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

Loop Execute Total 10 time (Iteration)

In Each Iteration 2 Times Memory Access (Reference)

Total Memory Reference = $1 + 2 * 10$

$= 1 + 20 = 21$ Ans

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

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 / Increment AM

② Indexed AM

③ PC-Relative AM

④ Base-Register AM

Auto Decrement | Auto Increment AM:

It is simial to Register Indirect AM in which Register Value (Content) automatically Decrement (or) Increment to Access the Data Sequentially.

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

Same operation Reform in ALL LOCATION from 2000 onwards.

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.

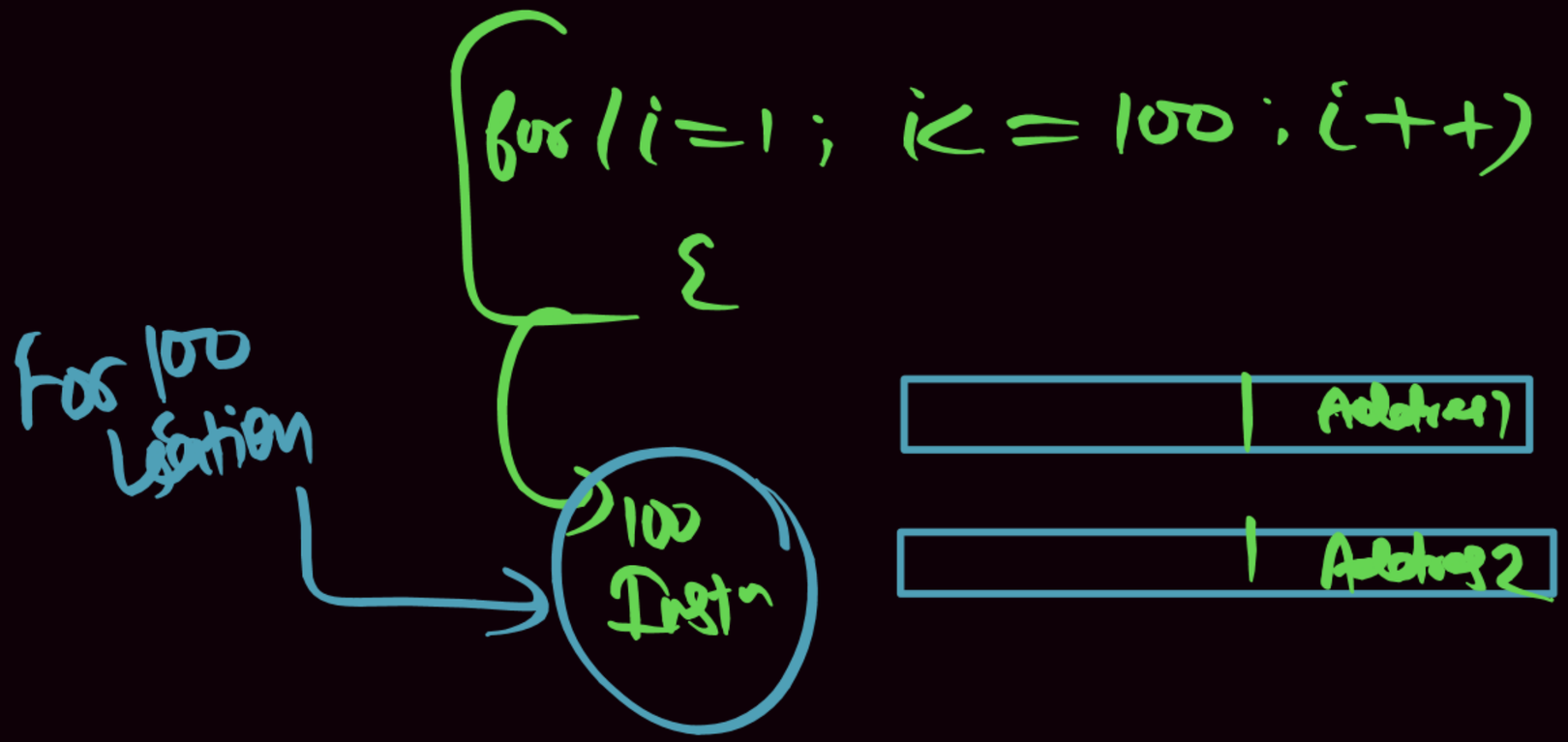
Q.1 WHY Auto Decrement (or) Auto Increment AM are USED ?

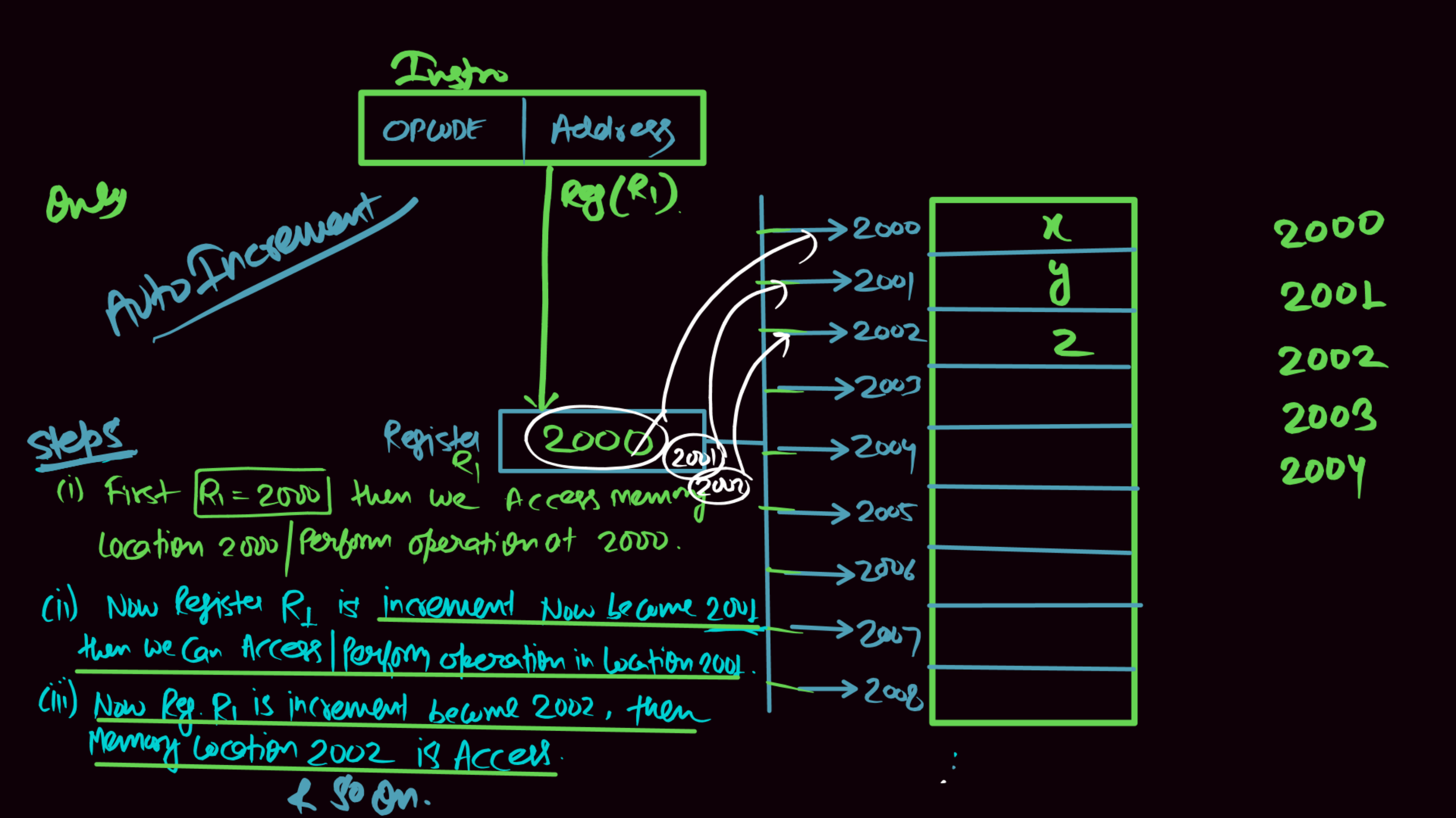
Soln

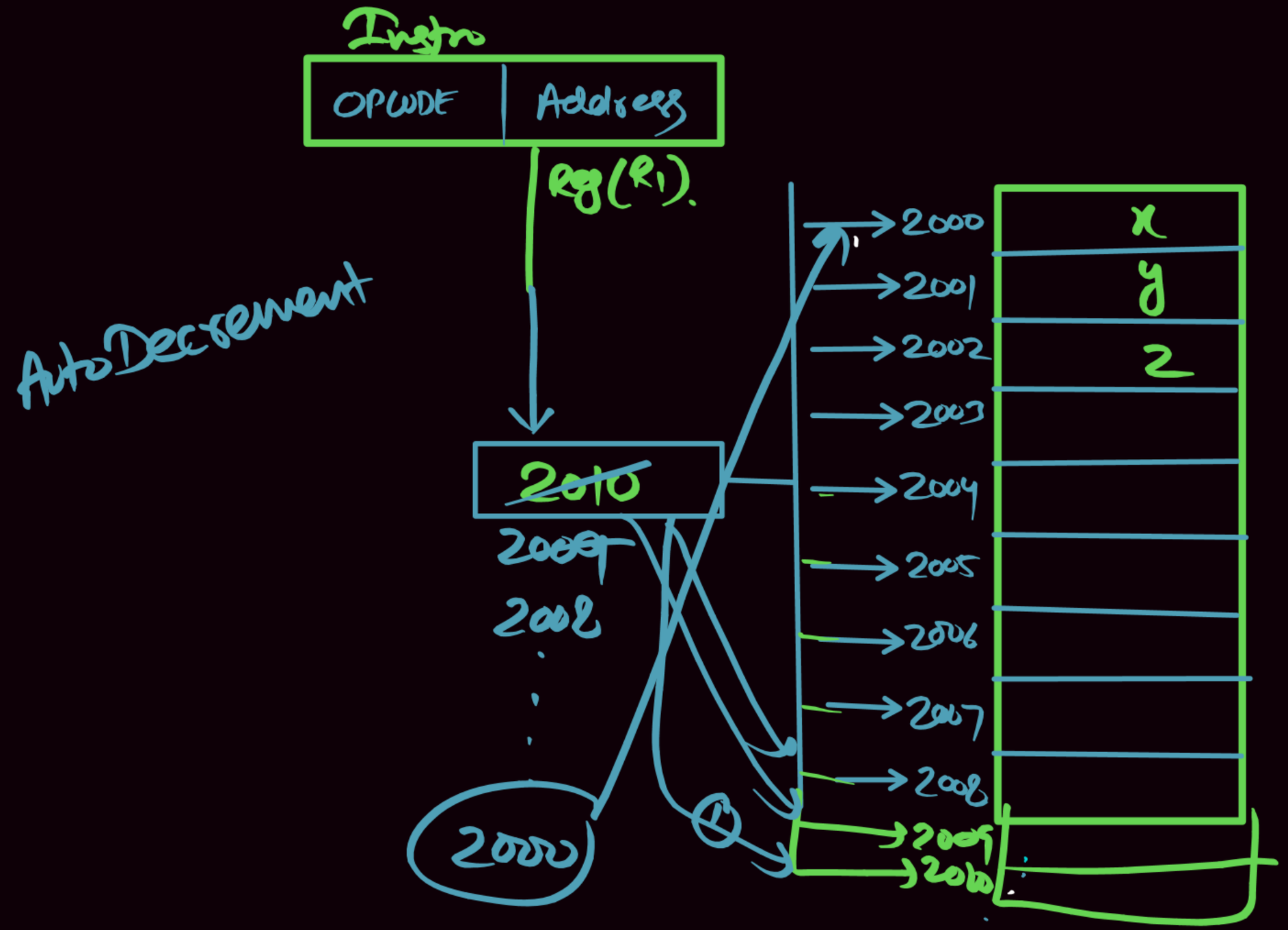
When we want to Access / Perform same operation in Multiple Location [Assume 100 Location] then Different-2 Instruction are Required.

- But with the Help of Auto Decrement / Increment AM it is possible by using Only 1 Instn we Perform same operation / Access the Multiple Location.

  HOW ?


for (i=1; i<=100; i++)
{
For 100 Location
100 Dist
Address1
Address2


Instr
OPCODE
Address
Reg(R1)
Only
Auto Increment
Register R1
2000
2001
2002
2000
2001
2002
2003
2004
2005
2006
2007
2008
x
y
z
2000
2001
2002
2003
2004
steps
(i) First R1 = 2000 then we Access memory location 2000 / Perform operation at 2000.
(ii) Now Register R1 is increment Now become 2001 then we can Access / Perform operation in location 2001.
(iii) Now Reg. R1 is increment become 2002, then memory location 2002 is Access.
& so on.


Instr
OPCODE
Address
Reg(R1).
AutoDecrement
2010
2009
2002
2000
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
x
y
z

By Using Auto Decrement / Auto Increment.

With the Help of Only One Instruction & increment/Decrement in the Register value we Accessing Multible Location OR

(OR) Performing the Same operation in Multible Location.

Index Based AM. : This AM to Implement the Array.

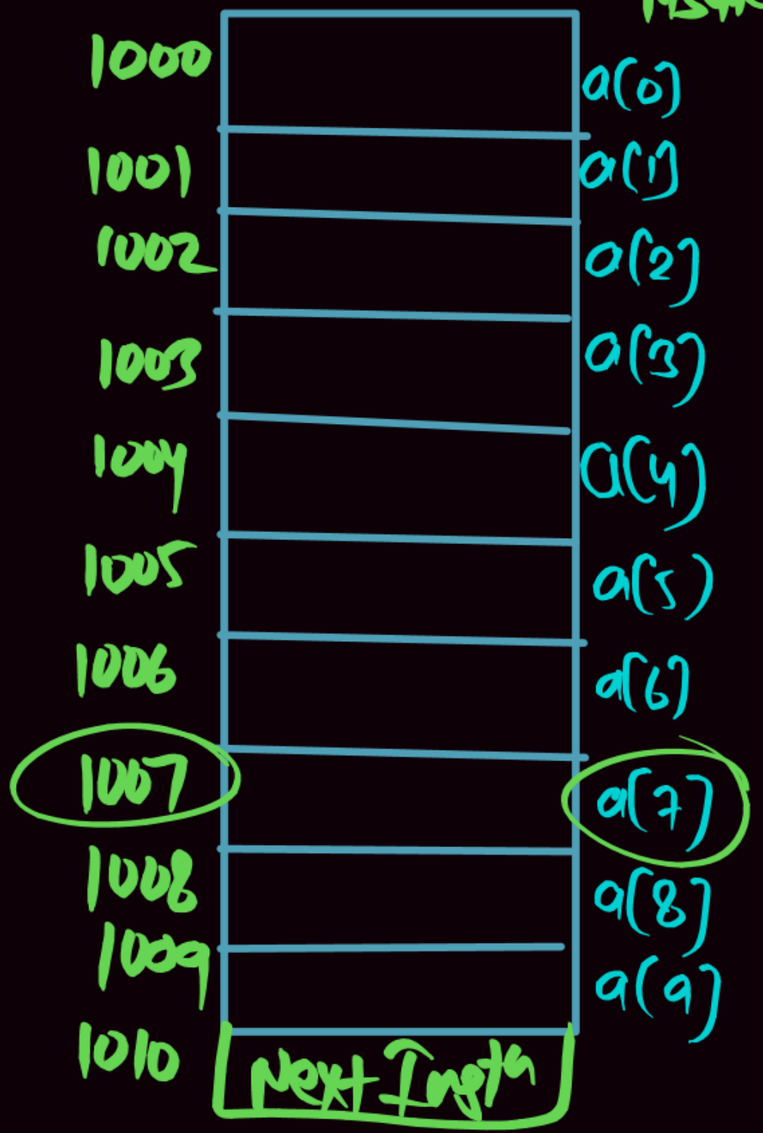
(OR)

This AM are Used to Access Specify (Random) element of the Array.

Assuming Starting addr. is 1000

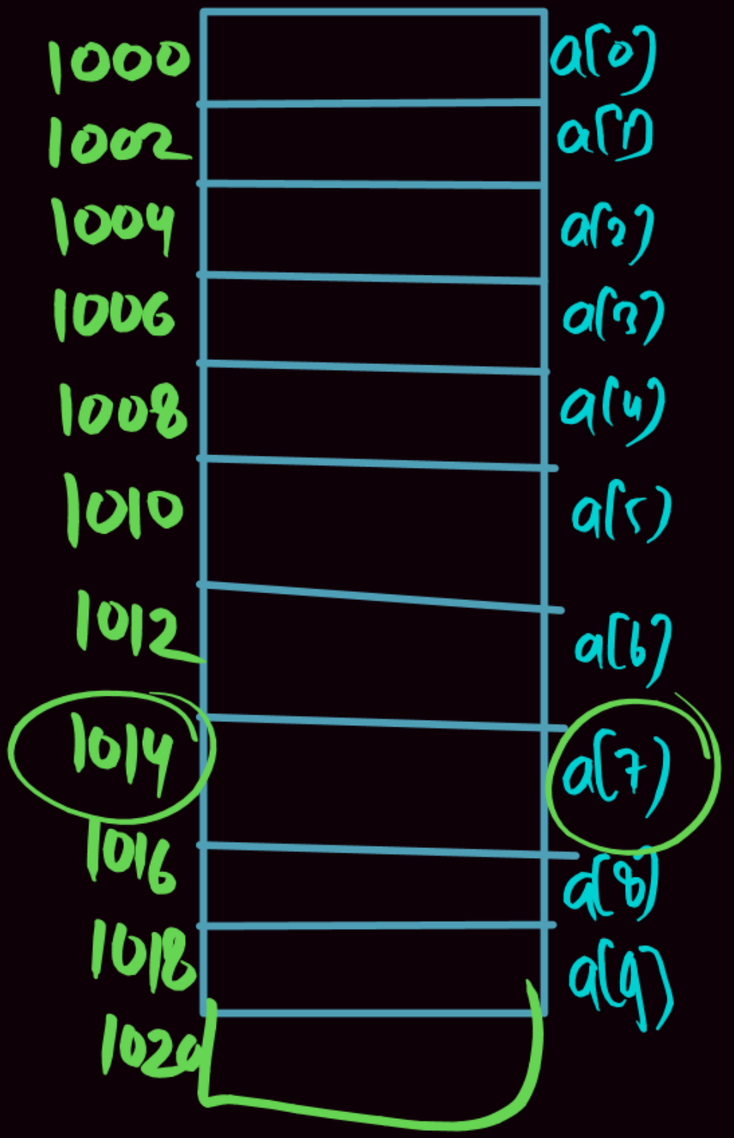
char a[10]

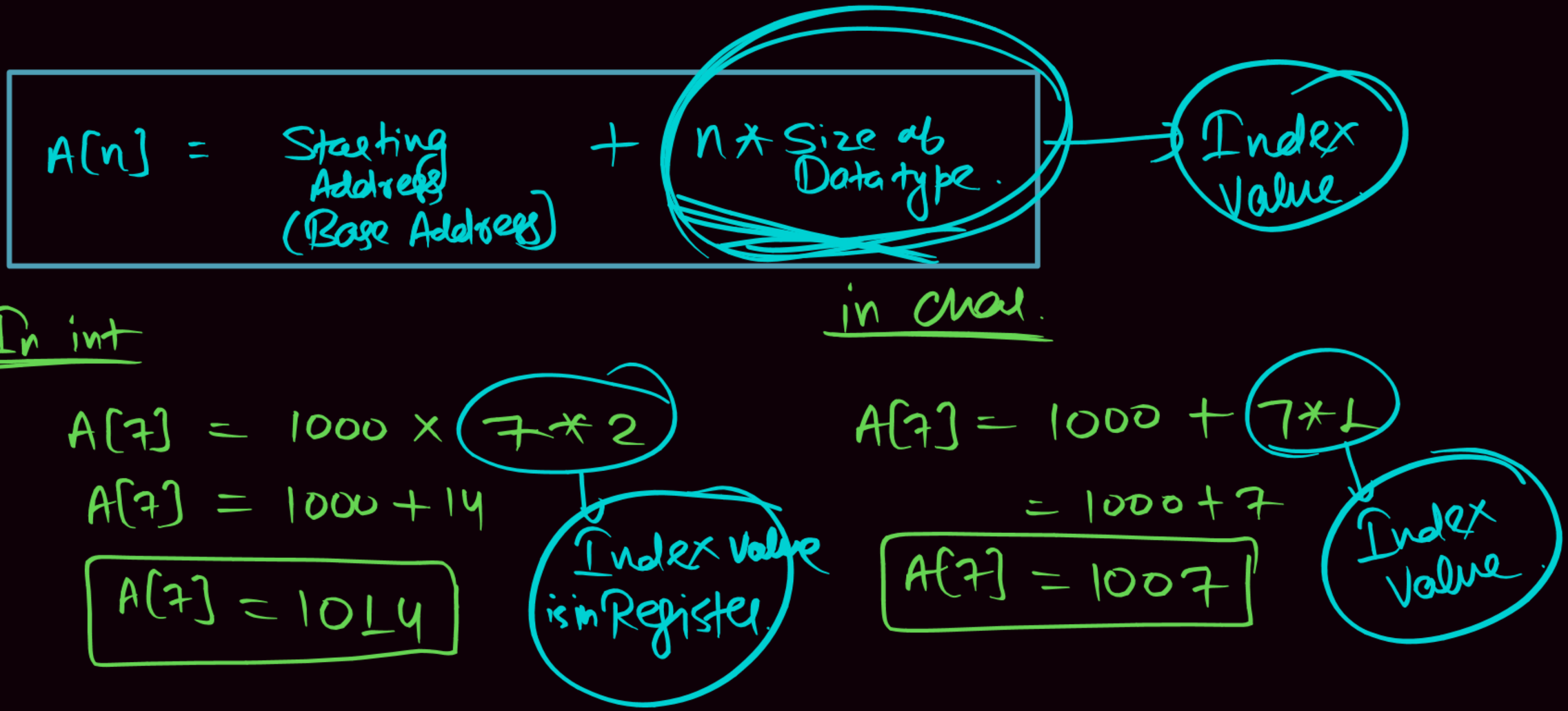
1char = 1Byte



int a[10]

1int = 2Byte

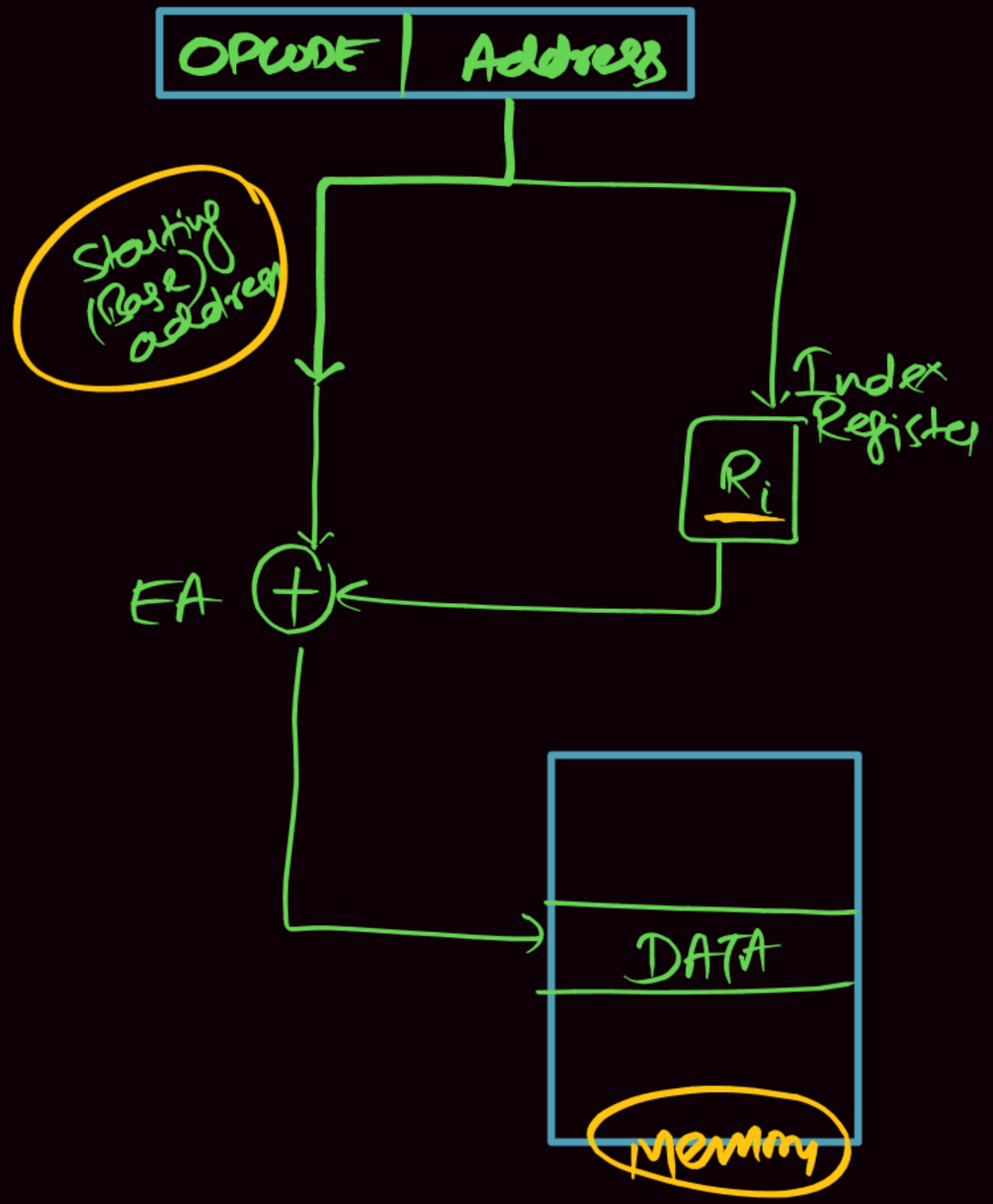



A[n] = Starting Address (Base Address) + n * Size of Datatype.
Index Value
In int
A[7] = 1000 x 7*2
A[7] = 1000 + 14
A[7] = 1014
Index value is in Register.
in char.
A[7] = 1000 + 7*1
= 1000 + 7
A[7] = 1007
Index Value

- Index Value are Present in the Index Register.
- Starting Address (Base address) is given in the Address field of the Instruction.

So

$$\text{Effective Address} = \text{Index Reg Value} + \text{Address field}$$



1 Register Ref for Index Reg

1 ALU Ref for EA Calculation

1 mem Ref for Read/Write Data

Index Register

At the CPU Design One Special Register is made as Index Register. then in this implicitly they Access the Index Register for Index Value.
(One Register waste → Bcz Not Heavy Utilization)

Most of the Case, Any General purpose Register is designated as a Index Register.
& that Register Name Mentioned (eg R1 as Index Register).

Adv:: We can Use that Register in Futuras as General Purpose Register.

Transfer of Control AM (TOC AM)

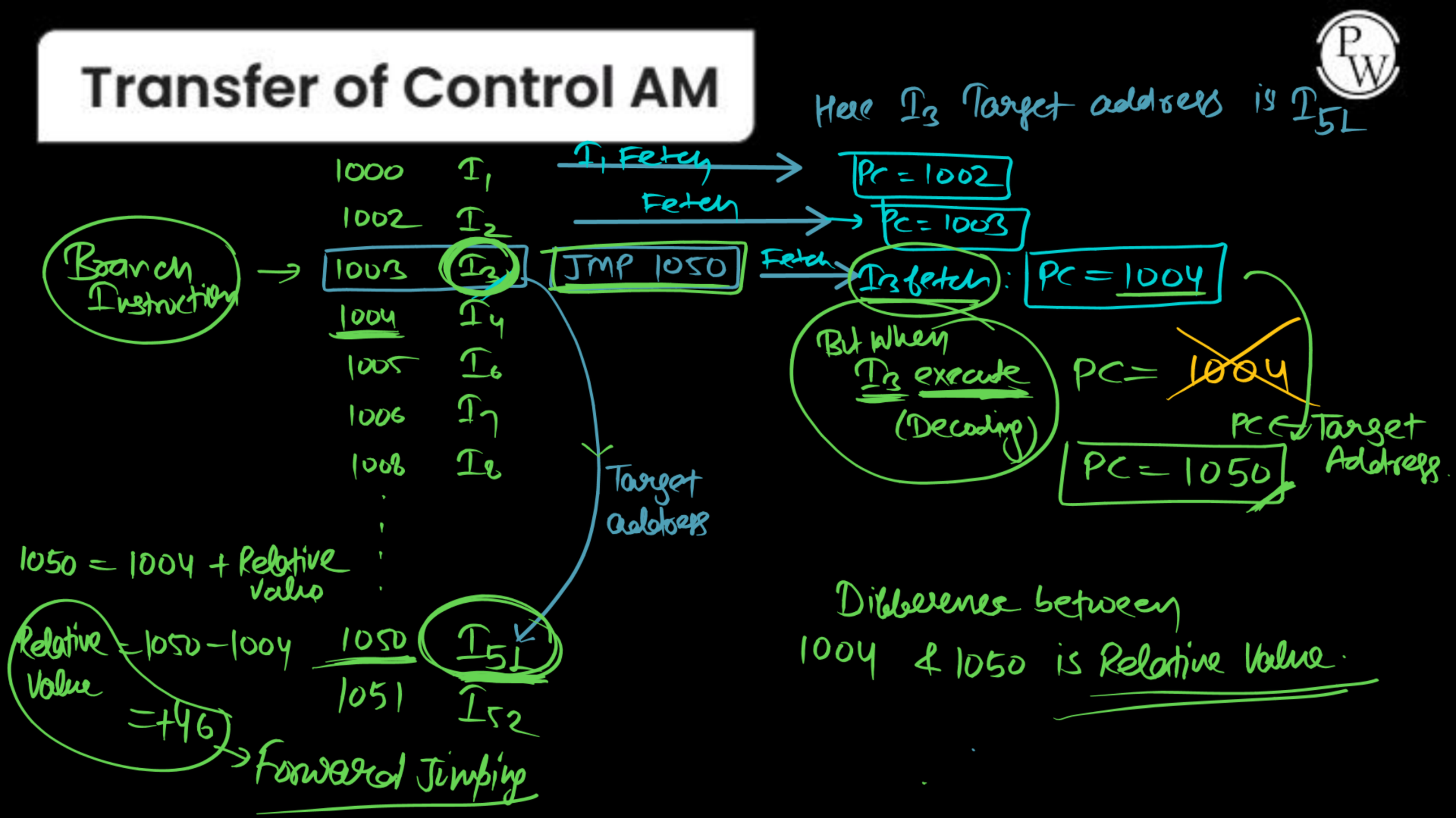
TOC AM focus on Instruction.

- During the execution of TOC Instn Program Control has been changed from One Location to Another Location.

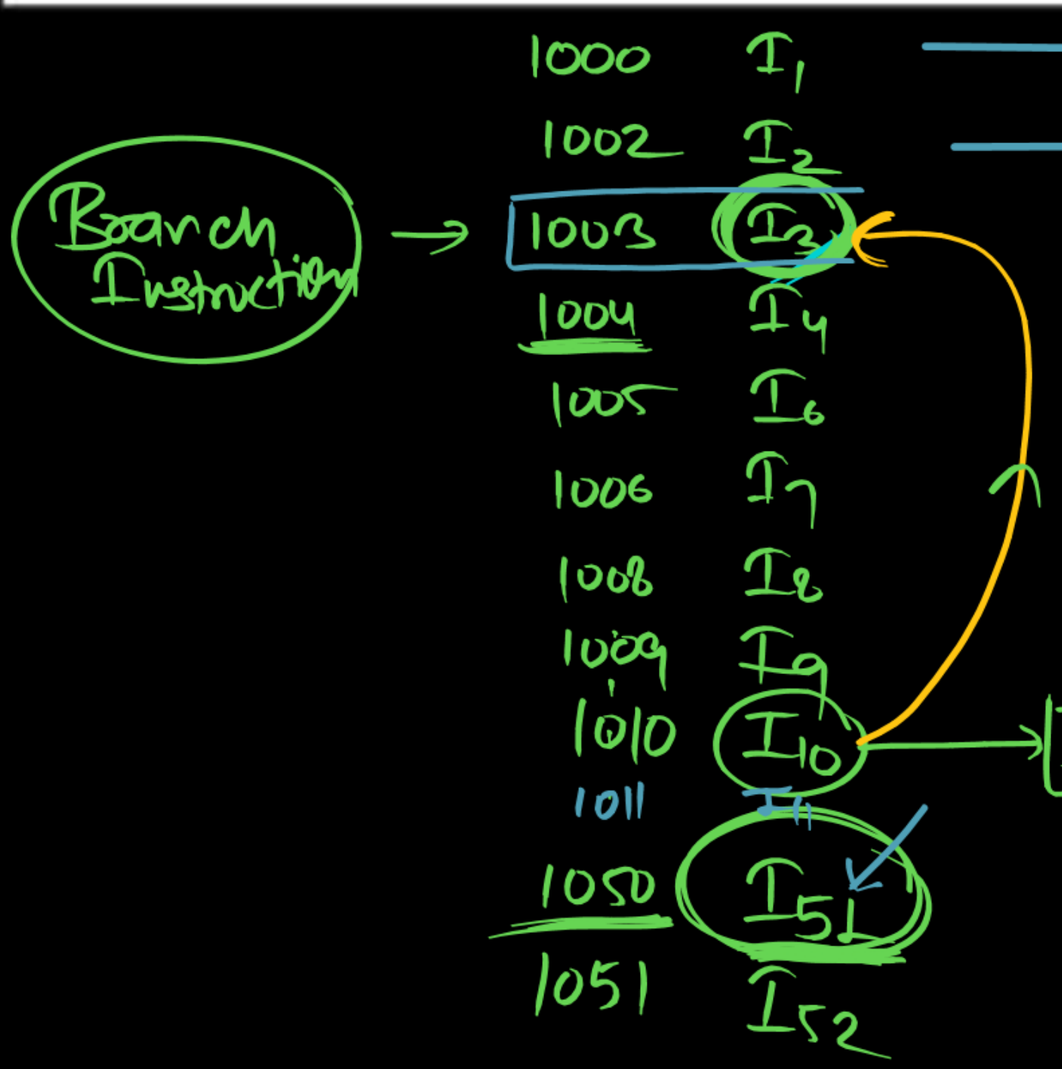
TA: Target Address
BA: Branch Address

Branch Instruction: JMP, Skip, goto.

Transfer of Control AM



Transfer of Control AM



I_{10} Target address is I_3.

$1003 = 1011 + x$

Relative value $= -8$

Ans Backward Jumping

When I_{10} Fetch PC = 1011

But when I_{10} Decode (Execute) PC = ~~1011~~

PC = 1003

Transfer of Control AM

$$\text{Target Address} = \text{PC Value} + \underset{\text{(Relative Value)}}{\text{AF (offset)}}$$

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)

+ : forward Direction (Jumping)
− : Backward Jumping (Branching)

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

(+ or −)

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)?

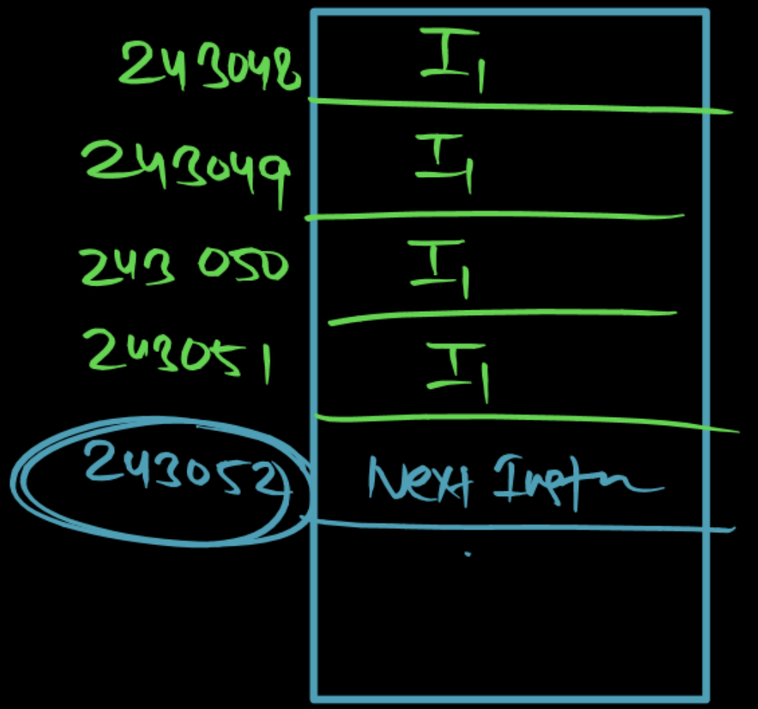
PC Value = 243052

Target Address = PC Value + Relative Value.

Backward Branching/Jumping. ⇒ 243052 + (−8)

= 243052 − 8

= 243044 Ans



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]	4
MOV R2, (R3)	R2 ← MEMORY [R3]	4
ADD R2, R1	R2 ← R1 + R2	2
MOV (R3), R2	MEMORY [R3] ← R2	4
INC R3	R3 ← R3 + 1	2
DEC R1	R1 ← R1 – 1	2
BNZ 1004	Branch if not zero to the given absolute address	2
HALT	Stop	1

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 ____.

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

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_1 i	add R2, R3, R4
I_2 i+1	sub R5, R6, R7
I_3 i+2	cmp R1, R9, R10
I_4 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]

Assume

I_1 → 1000 – 1003

I_2 1004 – 1007

I_3 1008 – 1011

I_4 1012 – 1015

$PC = 1016$

Target Address = 1000

Target Address = PC + R.V(offset)

$1000 = 1016 + x$

$x = -16$ Ans

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

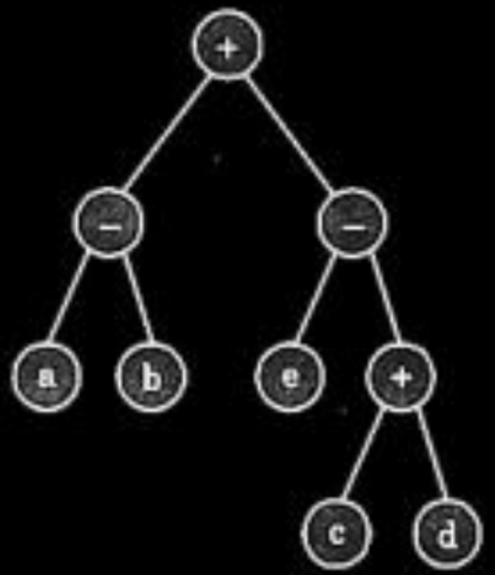
C Only S2 and S3

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!