COMPUTER SCIENCE

Computer Organization and Architecture

Machine Instruction and Addressing Modes

Lecture_ 09

Vijay Agarwal sir

01 Addressing Modes

Addressing Mode

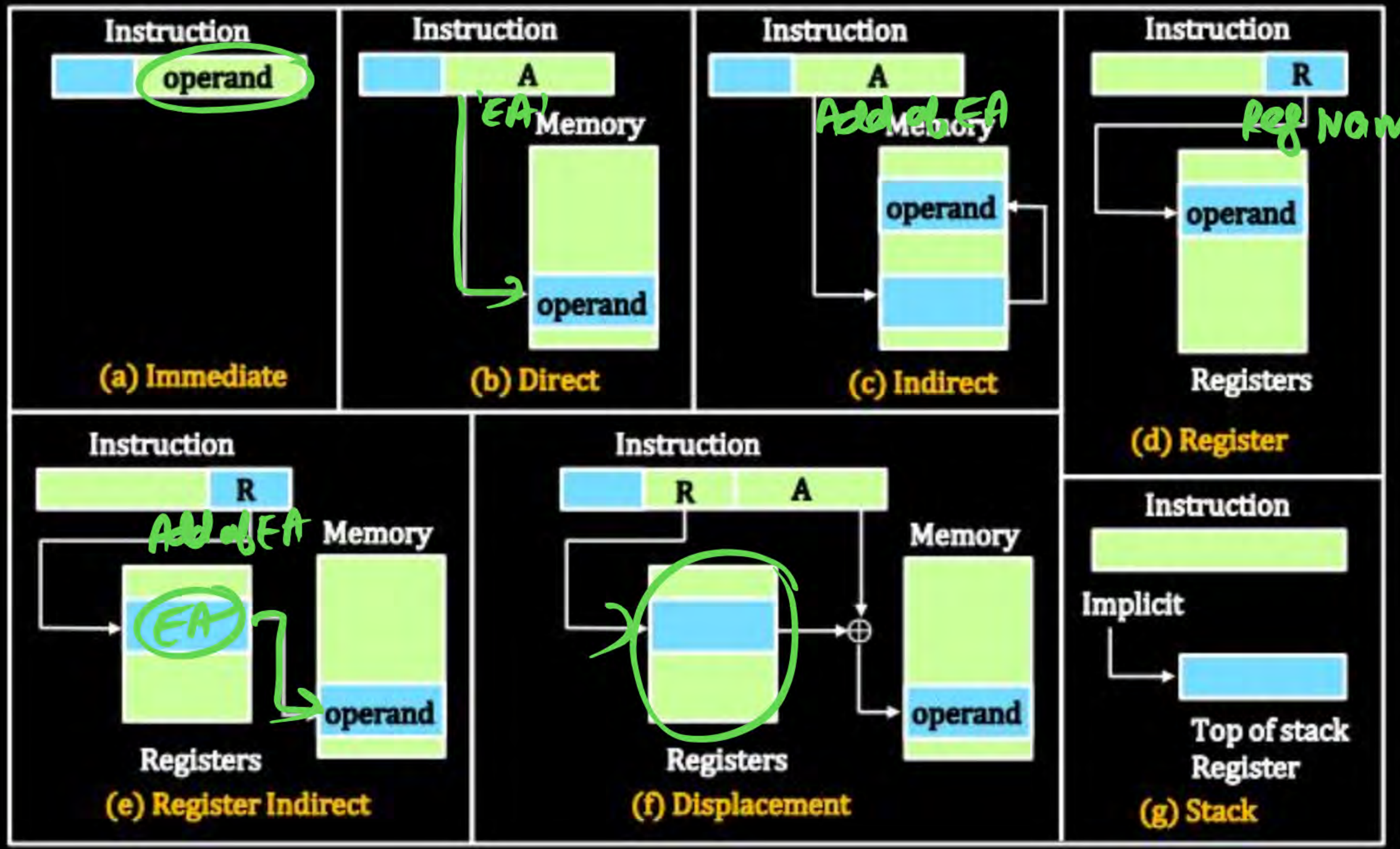
↳ 'EA'

Addressing Modes

1. Auto Decrement & Increment AM.
2. Index Register AM
3. PC-Relative AM
4. Base-Reg AM.

Addressing Modes

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


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

Addressing Mode	Effective Address	Content Of AC
Direct address		
Immediate Operand		
Indirect Address		
Relative address		
Indexed address		
Register		
Register Indirect		
Autoincrement		
Autodecrement		

PC = 200

*R*1 = 400

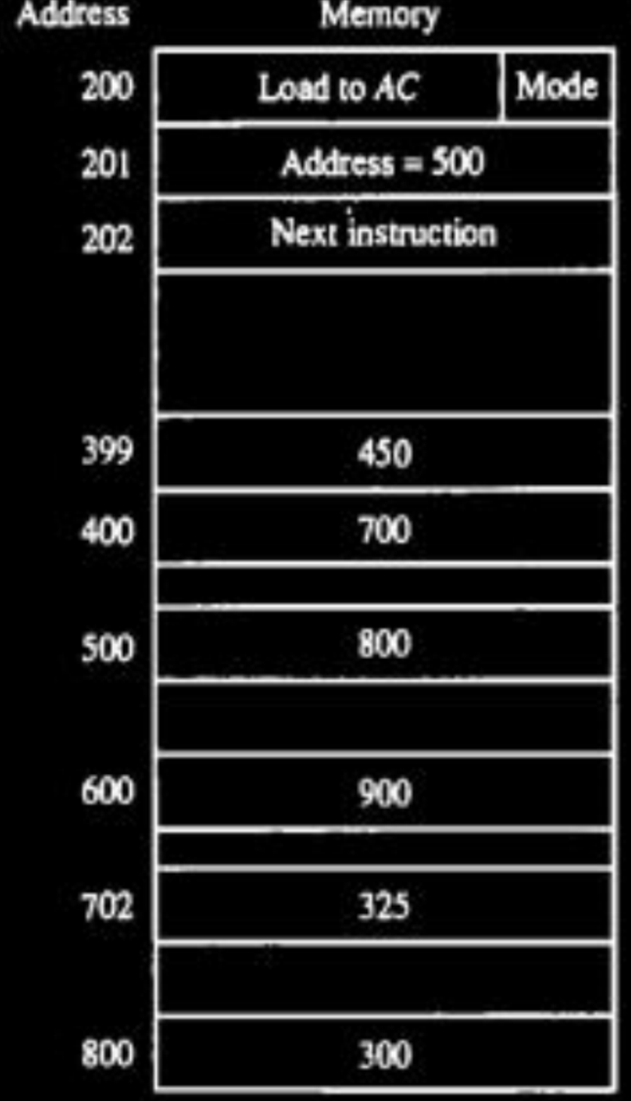
XR = 100

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.

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



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

① Indexed AM

② PC-Relative AM

③ Base-Register AM.

Index AM

Scaled Index AM.

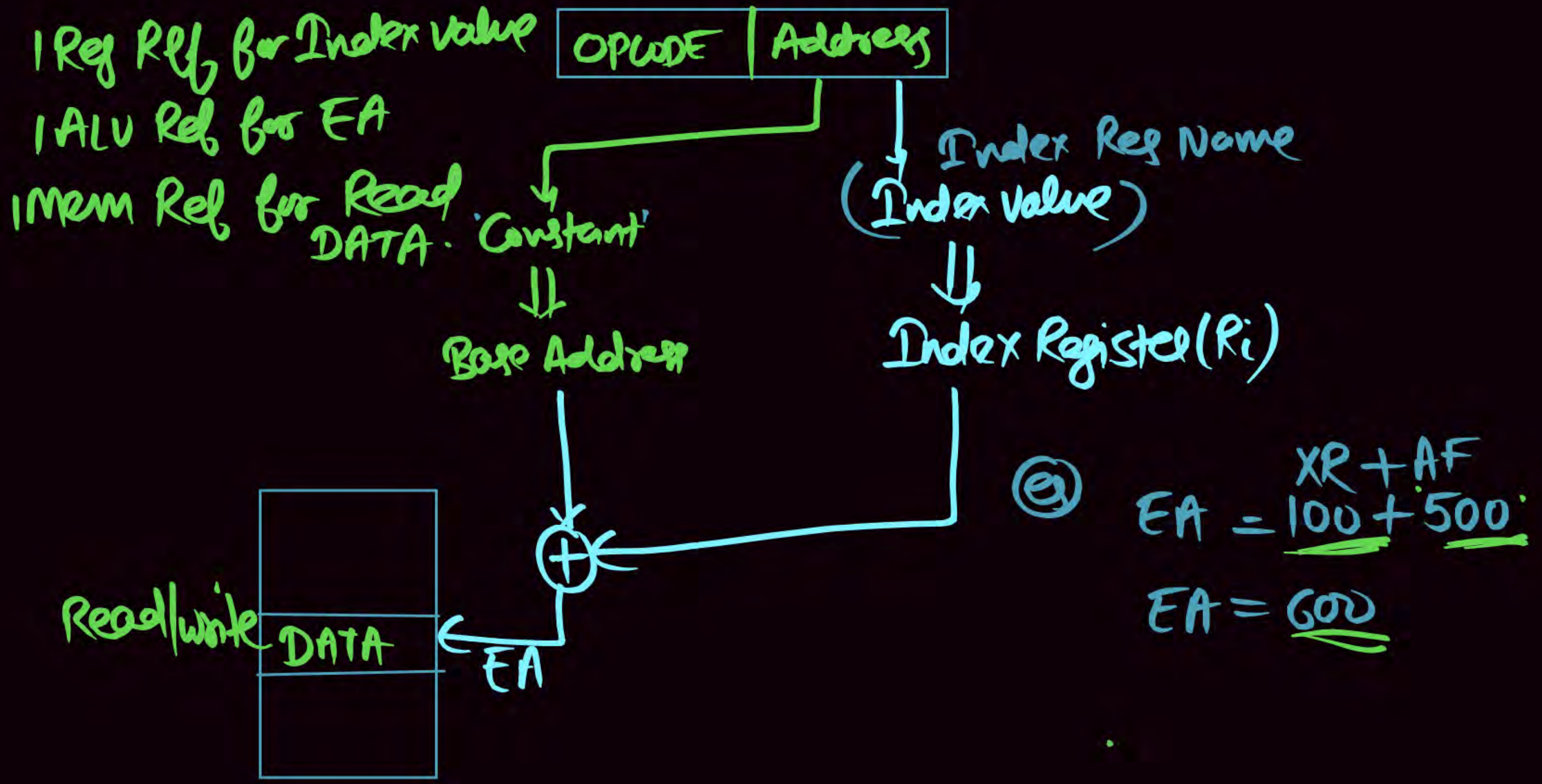
1. Index AM
2. Index Based AM | Base Index AM.
3. Base Index with Displacement AM.
4. Index Indirect AM.
5. Auto Indexed AM.

① Indexed AM :

Index AM is used to Access the Random element of 'Array'.

↓
Starting address of the Instruction
↓↓
Present in the address field of the Instruction as a Constant.

→ Position of array element
↓
Index Register.


1 Reg Ref. for Index value
1 ALU Ref. for EA
1 Mem Ref. for Read DATA
OPCODE
Address
'Constant'
Base Address
Index Reg Name
(Index value)
Index Register (Ri)
Read/write
DATA
EA
@
XR + AF
EA = 100 + 500
EA = 600

Index AM

e.g.

MOV Ax 1000(R_0)

$Ax \in m[1000 + R_0]$

$Ax \in m[1000 + 20]$

$Ax \in m[1020]$

R_0 as a Index Register.

$R_0 = 20$.

Index AM

Scaled Index AM.

1. Index AM
2. Index Based AM / Base Index AM.
3. Base Index with Displacement AM.
4. Index Indirect AM.
5. Auto Indexed AM.
   - Auto Incrment / Decrment

② Base Index AM

Some time CPU Contain Separate Base Register & Index Register to Store the Base Value & Index Value Respectively

(e) **8086 Up.**

BX Reg ⇒ Base Register

SI|DI : Index Register [Source Index | Destination Index].

```
MOV Bx, #1000
MOV SI, #20.
```

```
MOV Ax [BX][SI]
```

Ax ∈ [BX + SI]

Ax ∈ M[1000+20]

∴ Ax ∈ M[1020]

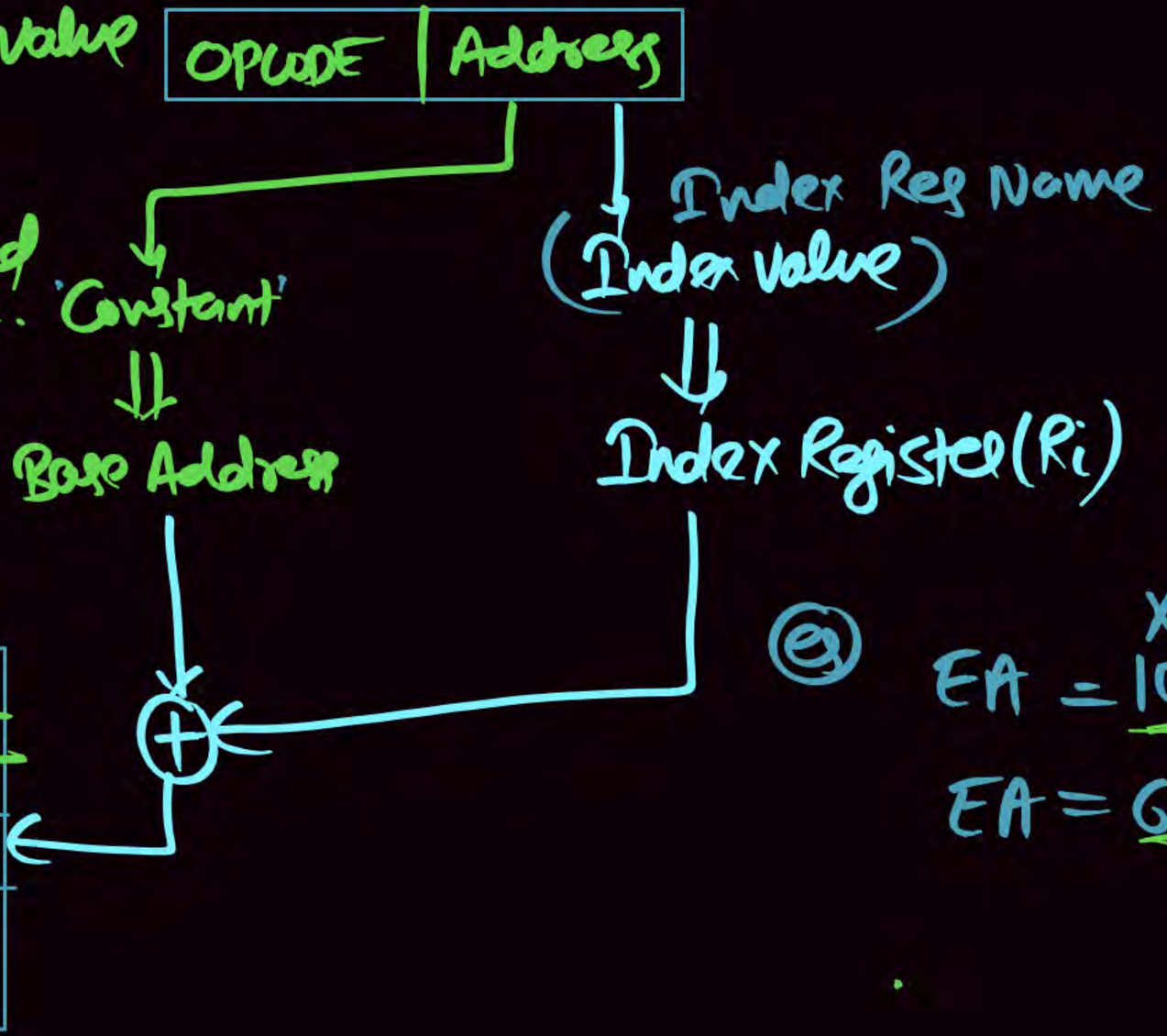
③ Base Index with Displacement :

mov Ax 50 ([BX] [SI])

$$Ax \in M[([BX] + [SI] + 50)]$$

④ Indirect Index AM.

1 Reg Ref for Index value
1 ALU Ref for EA
1 Mem Ref for Read DATA.



e.g.

XR + AF
EA = 100 + 500
EA = 600

Not in used.

Register Indirect

1 Reg Ref EA
1 Mem Ref DATA

Memory Indirect

1 Mem Ref for EA
1 Mem Ref for Read/write DATA

Index Indirect

1 Reg Ref for Index Value
1 Arithmatic Ref. for EA calculation
1 Mem Ref for EA
1 Mem Ref for Read/write DATA

Transfer of Control AM [TOC]

- Transfer of Control [TOC] Flow AM focused on 'Instruction'.
- During the execution of TOC Instruction Program Control has been changed from One Location to Another Location (target)

TA: Target Address
B.A: Branch Address
EA: Effective Address.

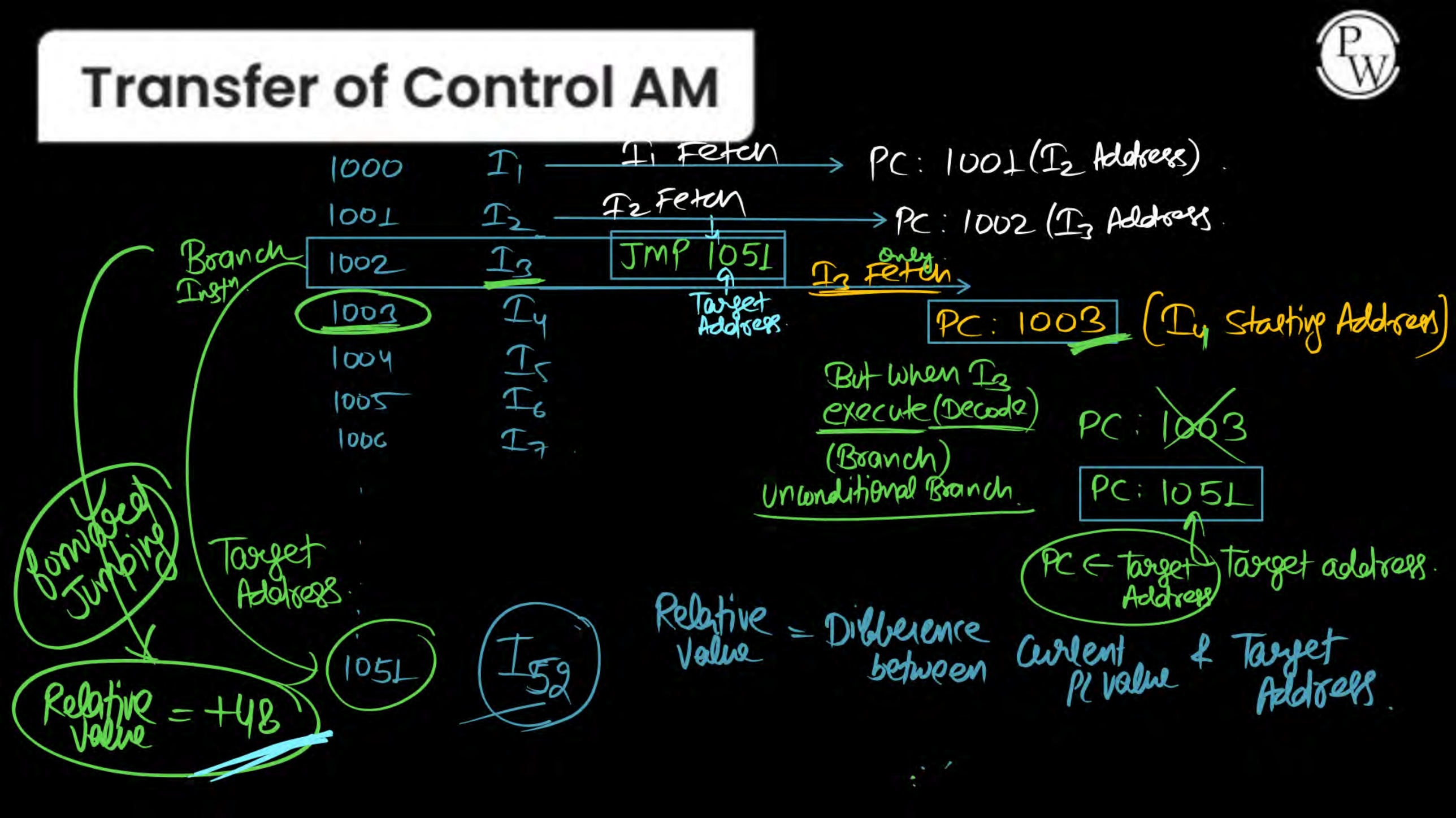
Branch Instruction → JMP, SKIP, goto etc.

Transfer of Control AM

TOC flow AM.

1. Unconditional TOC.
   - eg
     1. JMP x (Address)
     2. HALT
2. Conditional TOC.
   - JNZ (Jump On Not Zero)
   - BNZ (Branch On Not Zero)
   - BZ (Branch On Zero)
   - JZ (Jump On Zero).

Transfer of Control AM



Relative Value is Difference between Current PC Value & Target Address

Note

Relative Value = +ve (Forward Jumping)

Relative Value = −ve (Backward Jumping)

PC Relative AM.

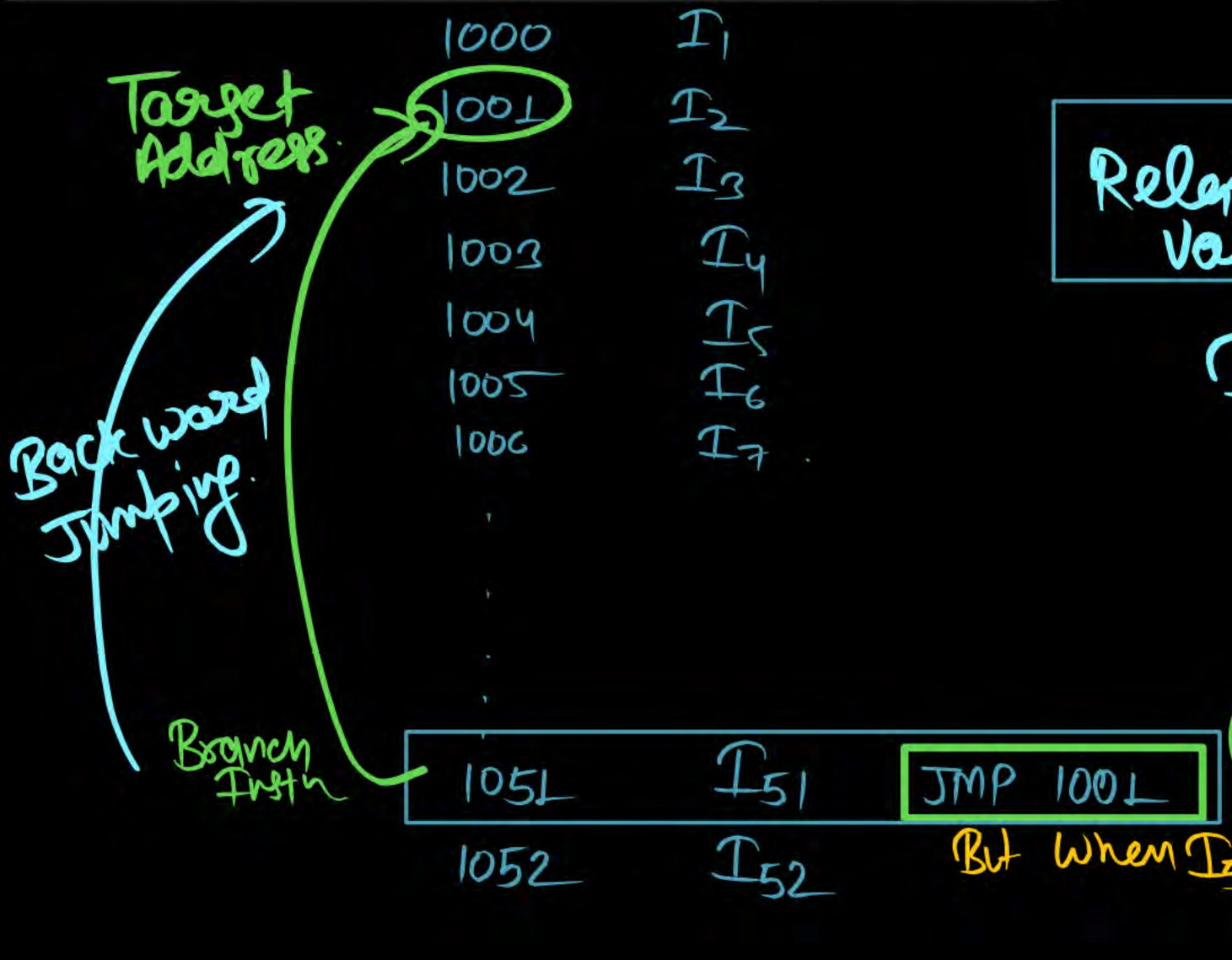
$$\text{Target Address} = \text{Current PC Value} + \text{OFFSET (Relative Value)}$$

$$1051 = 1003 + \text{Relative value}$$

$$\text{Relative value} = 1051 - 1003$$

$$\text{Relative value} = (+48) \text{ forward Jumping.}$$

Transfer of Control AM



Relative Value = −51

Backward Jumping.

$$\text{Target Address} = \text{Current PC Value} + \text{Relative Value}.$$

$$1001 = 1052 + \text{Relative Value}.$$

$$\text{Relative Value} = 1001 - 1052$$

$$= -51$$

$$\text{Relative Value} = -51$$

$\therefore$

Conditional TOC : In Conditional TOC condition Checking Done on the Status of Previous Instruction output.

Transfer of Control AM

Jump on Not zero.

$R_1 = 0$ So Condition false.

Conditional Branch.

1000	I_1	
1001	I_2	$R_1 = 0$
1002	I_3	JNZ 1051
1003	I_4	
1004	I_5	
1051	I_{101}	

I_3 : Fetch

PC : 1003

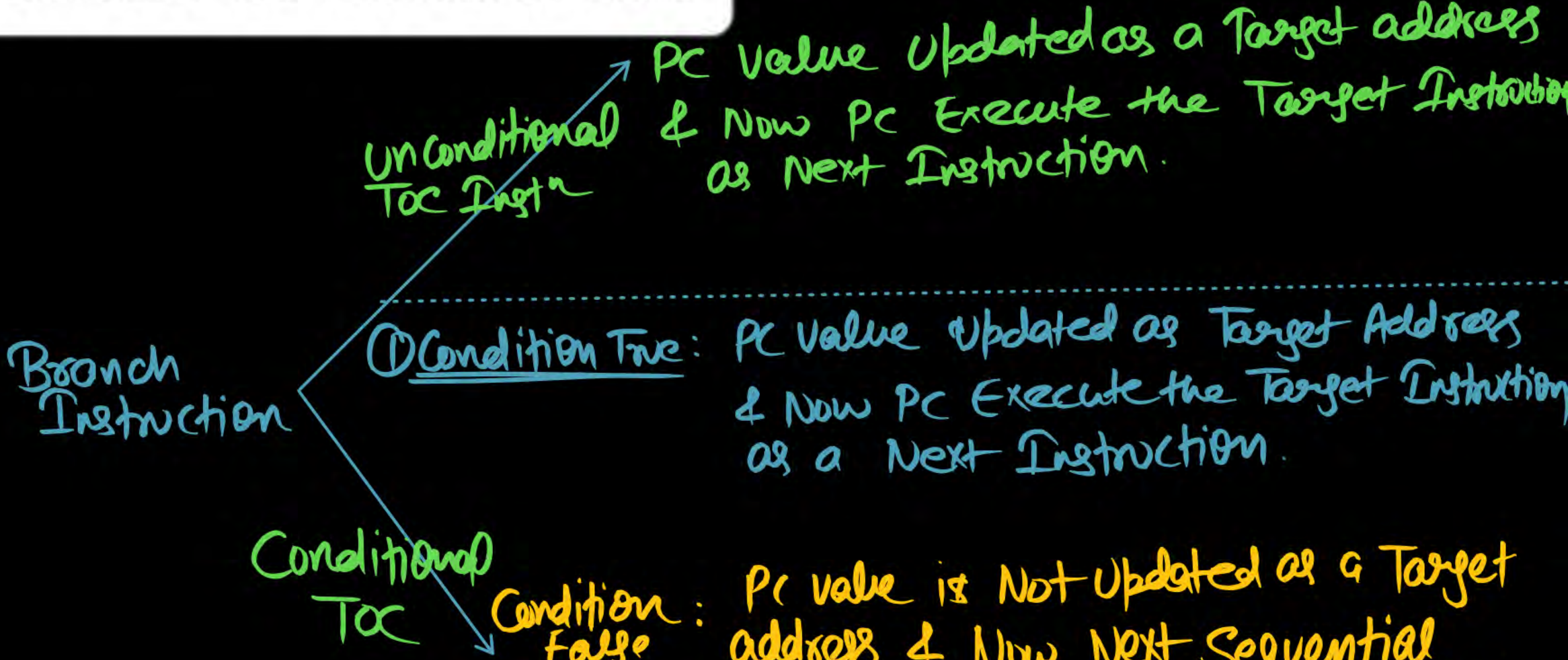
But when I_3 execute then PC Value.

PC : 1003

No Change Bcz Condition false.

So Target address is the Next Sequential Instruction address.

Transfer of Control AM



PC Relative AM

Target Address = Current PC value + Relative Value.

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.

Note

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.

Q 8086 up 1MB memory Capacity. Divided into 16 Segment of 64KB.

$16 \times 64KB$

$2^4 \times 2^{16} B$

2^{20} Byte.

Base Register AM

Target Address = Base Register Value + OFFSET (AF) or Relative Value.

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.

Base Register AM ⇒ Reallocation.

PC-Relative AM ⇒ Intra Segment Branching

Q. 1

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)? (243044) Ans

Instruction Size = 4 Byte

Starting address = 243048

Current PC value = 243052

Displacement (Relative value) = −8 (Backward jumping)

Target Address = Current PC value + Relative value

= 243052 + (−8)

= 243052 − 8 = (243044) Ans

Q.2

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

Ans 870

Ans 468

4Byte JMP Instruction

Memory Word Addressable.

1 Word Size = 16 bit ≡ 2 Byte

Instn Size = 4Byte ≡ 2 Words.

4B / 2B Words

Target Address = Current PC value + Relative value.

$$902 + (-32)$$

$$\Rightarrow 902 - 32$$

Target Address = 870

Ans

Decoder

JMP	-32

(1) PC Relative AM

Target Addr = Current PC Value + AF (OFFSET) Relative Value

902 - 32

= 870 Ans

(2) Base Register AM

Target addr = Base Register Value + Relative Value

500 + (-32)

500 - 32

Target Address = 468 Ans

Q.3 2Byte.

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?

16 bit Instruction = 2Byte.

8bit	8bit
OPCODE	AF

900	OPCODE } IL
901	AF }
902	Next Instn

PC = 902

Target Addr = 614

(i) Target Address = Current PC Value + Relative Value.

614 = 902 + Relative Value ⇒ R.V = 614 − 902.

(i) **Relative value = −288** Ans

(ii) PC value Before Instn Fetch : 900
After Instruction Fetch : 902
After Execution : 614 (Target Address)

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 50 Ans

[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
1000-1003 i	add R2, R3, R4
1004-1007 i +1	sub R5, R6, R7
1008-1011 i + 2	cmp R1, R9, R10
1012-1015 i + 3	beq R1, Offset

PC Value = 1016
Target Address = 1000

Target Address = Current PC Value + OFFSET (Relative Value)

1000 = 1016 + offset

offset = -16 Backward jumping

If the target of the branch instruction is i, then the decimal value of the Offset is -16. Ans

[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

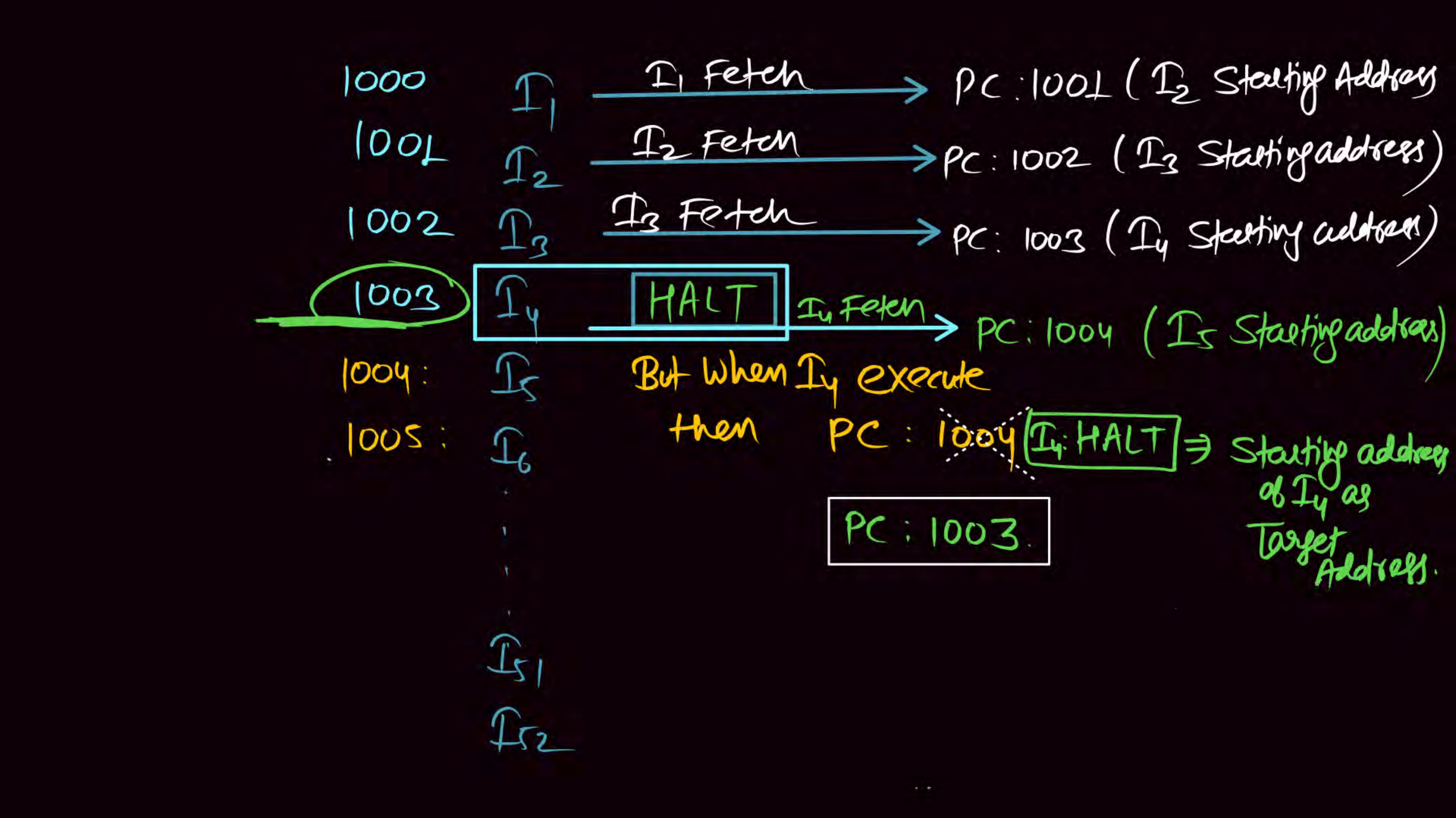
(D) All of S1, S2 and S3

Unconditional JOC

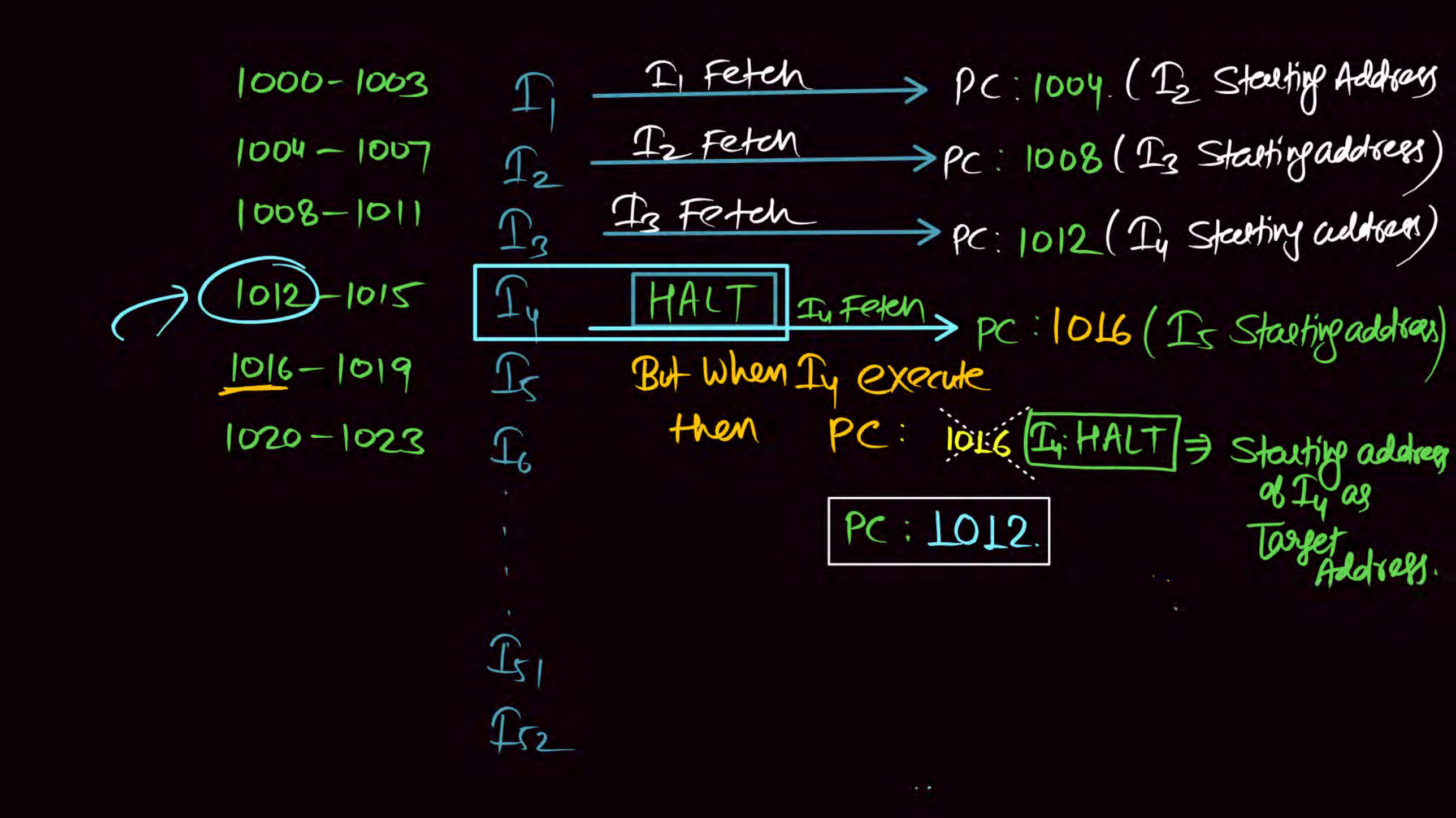
JMP x ← Target Address

Eg: JMP 4000 ← Target address.

HALT: Uncondition TOC Instruction in which Target Address is the Starting Address of the Instruction which Called HALT Instruction.


1000
I1
I1 Fetch
PC : 1001 (I2 Starting Address
1001
I2
I2 Fetch
PC : 1002 (I3 Starting address)
1002
I3
I3 Fetch
PC : 1003 (I4 Starting address)
1003
I4
HALT
I4 Fetch
PC : 1004 (I5 Starting address)
1004 :
I5
But when I4 execute
then PC : 1004
I4 : HALT
⇒ Starting address of I4 as Target Address.
1005 :
I6
PC : 1003
I51
I52

Another example of HAL-


1000-1003
1004-1007
1008-1011
1012-1015
1016-1019
1020-1023
I1
I2
I3
I4
I5
I6
I51
I52
I1 Fetch
PC: 1004. (I2 Starting Address
I2 Fetch
PC: 1008 (I3 Starting address)
I3 Fetch
PC: 1012 (I4 Starting address)
HALT
I4 Fetch
PC: 1016 (I5 Starting address)
But when I4 execute
then PC: 1016 I4: HALT ⇒ Starting address of I4 as Target Address.
PC: 1012.

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?

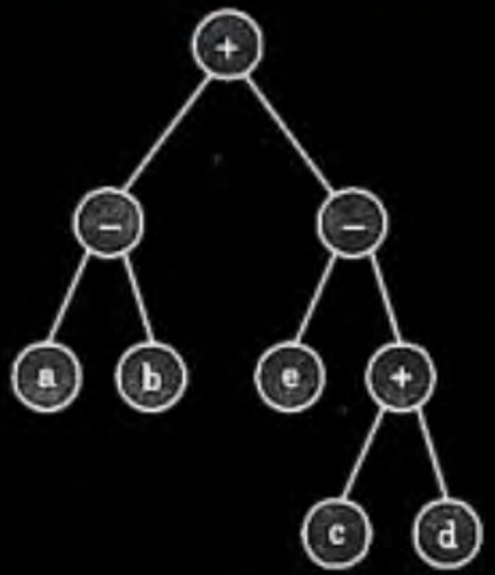
[GATE-2011-CS: 2M]

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)

Instruction Set

1. DATA Transfer Instn / operation.
2. DATA Manipulation Instn / operation
   - (i) Airthmatic (eg ADD, SUB, DIV, MUL, INC, etc.)
   - (ii) Logical (OR, NOT, AND, XOR, NAND etc)
   - (iii) Shift & Rotate operation (Airthmatic & Logical Shift Rotate with & without Carry)
3. Program Control Instn / operation (TOC Instn) (Unconditional & conditional TOC)

D.E

Data Transfer Instn/operation: Data transferred from one location to another location.

(i) Mov:

Destination	Source
mem	Reg
Reg	mem
Reg	Reg
mem	Immediate
Reg	Immediate

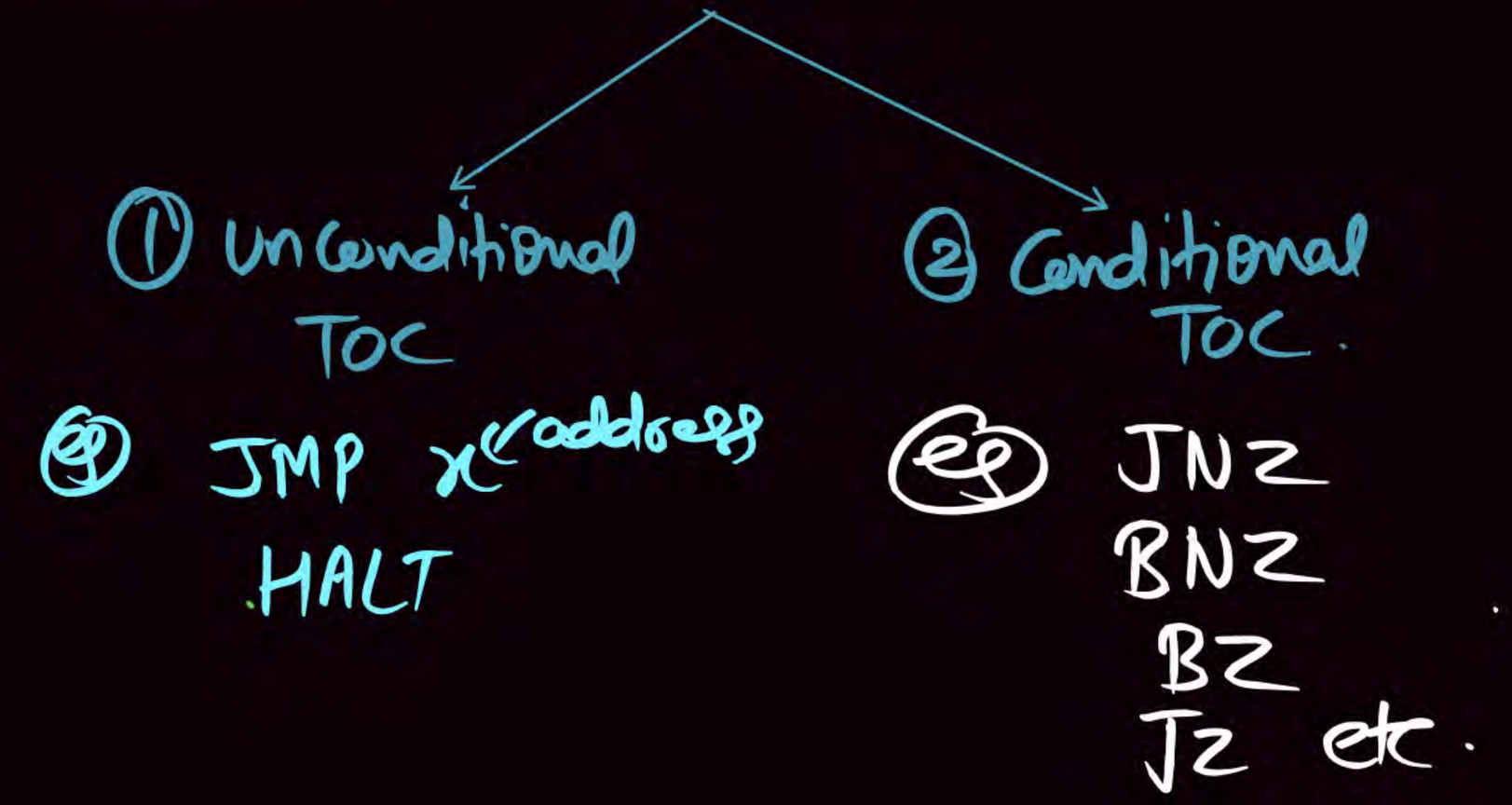
Data Transfer Instn/operation:

② LOAD, STORE } memory specific → Memory Read
→ Memory write

③ PUSH, POP } Stack Specific → Insert
→ Delete

④ IN, OUT } IO specific → IO Read
→ IO write.

③ Program Control Instn (TOC Instn)



COA PYQ's

Digital Electronic

1. Airthmatic, Logical & Shift, Rotate operation.

   RLC, RRC

2. Flag
   - Carry
   - Sign
   - zero
   - Auxillary carry
   - Parity
   - overflow Flag.

THANK YOU!