COMPUTER SCIENCE

Computer Organization and Architecture

Machine Instruction and Addressing Modes

Addressing Modes-4

Lecture_08

Vijay Agarwal sir

01 Addressing Modes

Addressing Modes

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

Addressing Modes

Auto Decrement | Increment AM:

Register Indirect

in this change in the

Value [Content] of Register

& Accessing | Perform same Operation in 'n' Memory Location.

Addressing Modes

Index Req AM

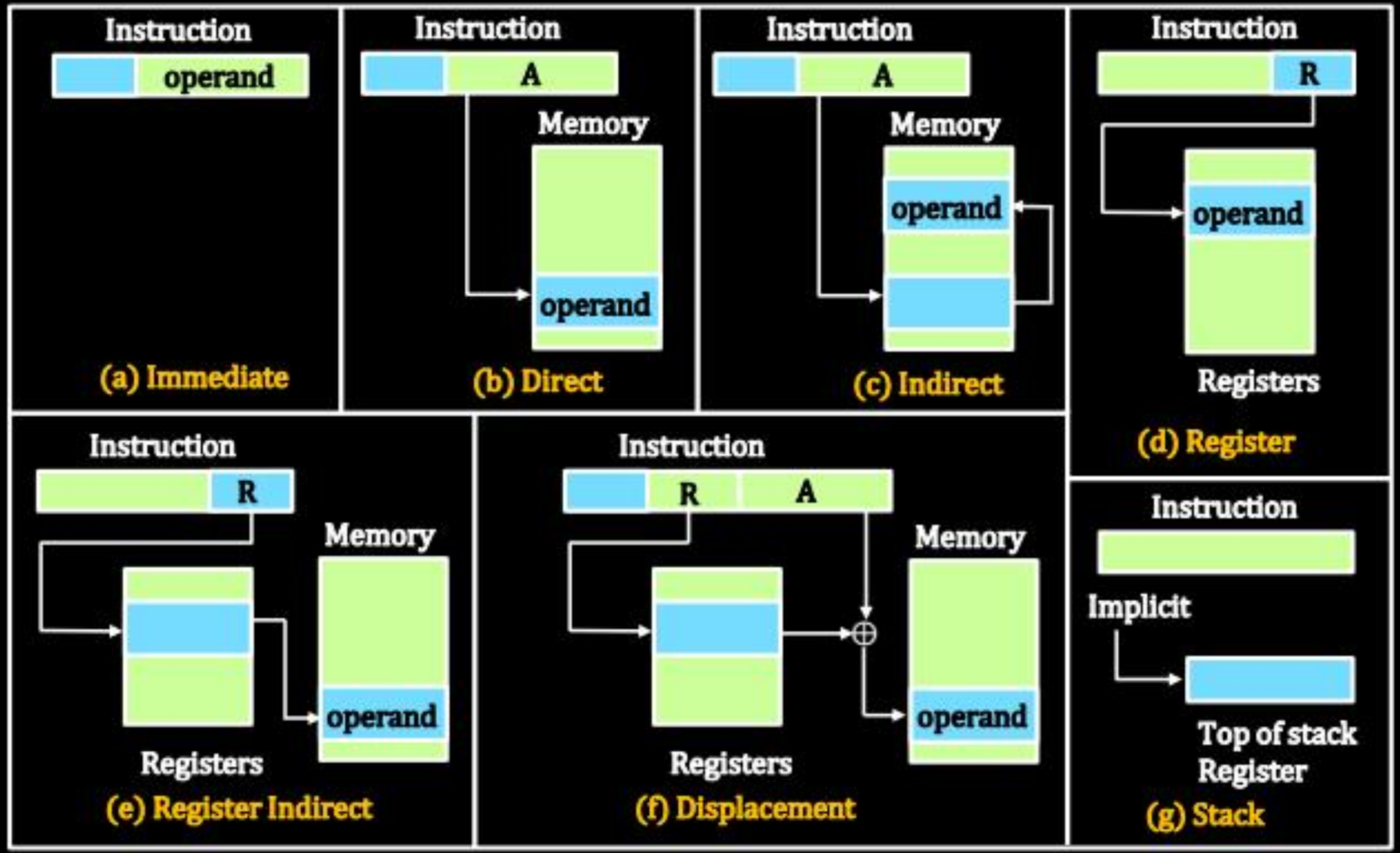
EA = A.F + Index Register

A(R0)

M[A + R0]

Addressing Modes

- Immediate
- Direct
- Indirect
- Register
- Register Indirect
- Displacement
- 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
operand
Registers
(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	702	Target Instr.
Indexed address		
Register		
Register Indirect		
Autoincrement		
Autodecrement		

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.

PC-Relative AM = PC Value + A.F (or) Offset [Relative Value]

= 202 + 500

= 702

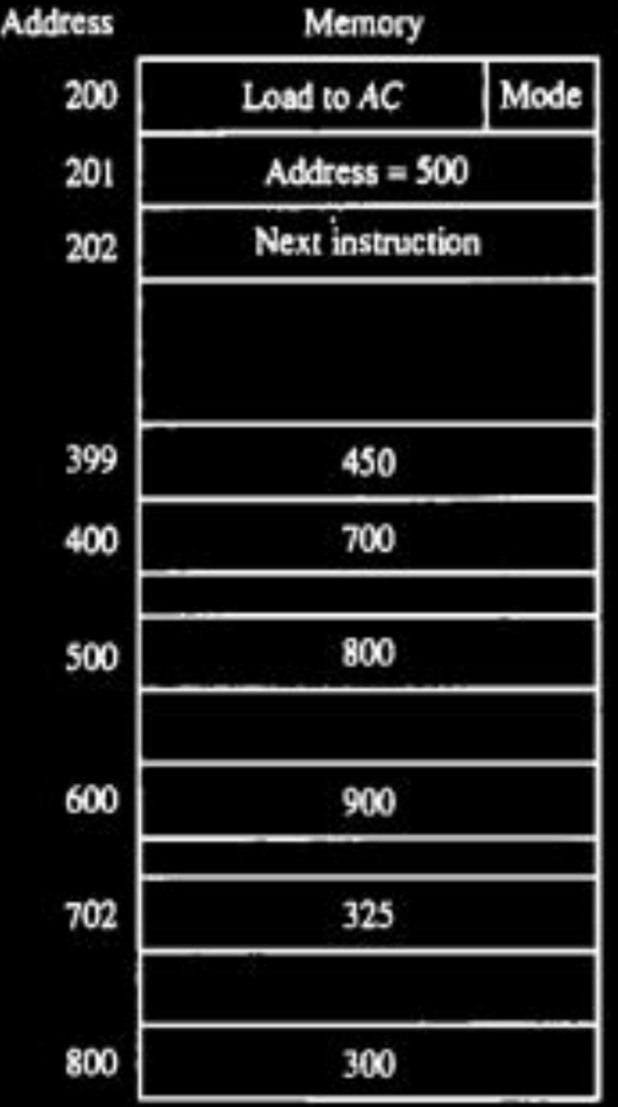
Addressing Mode	Effective Address	Content Of AC
Direct address	500	800
Immediate Operand	201	500
Indirect Address	800	300
Relative address	702	325
Indexed address	600	900
Register	--	400
Register Indirect	400	700
Autoincrement	400	700
Autodecrement	399	450

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.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]

M(1000+Rs).

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

M(0+Rd)

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

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

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.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]	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

Transfer of Control AM

TOC AM focus on Instruction. → PC Relative AM | Relative AM
→ Base Reg AM

Conditional TOC

- eg JNZ
- eg JZ
- ⋮

Unconditional TOC

- eg JMP x ← Target Address
- HALT
- goto
- .

Branch Instruction

JMP
SKIP
Branch
Goto

TA : Target Address
BA : Branch Address
EA : Effective Address

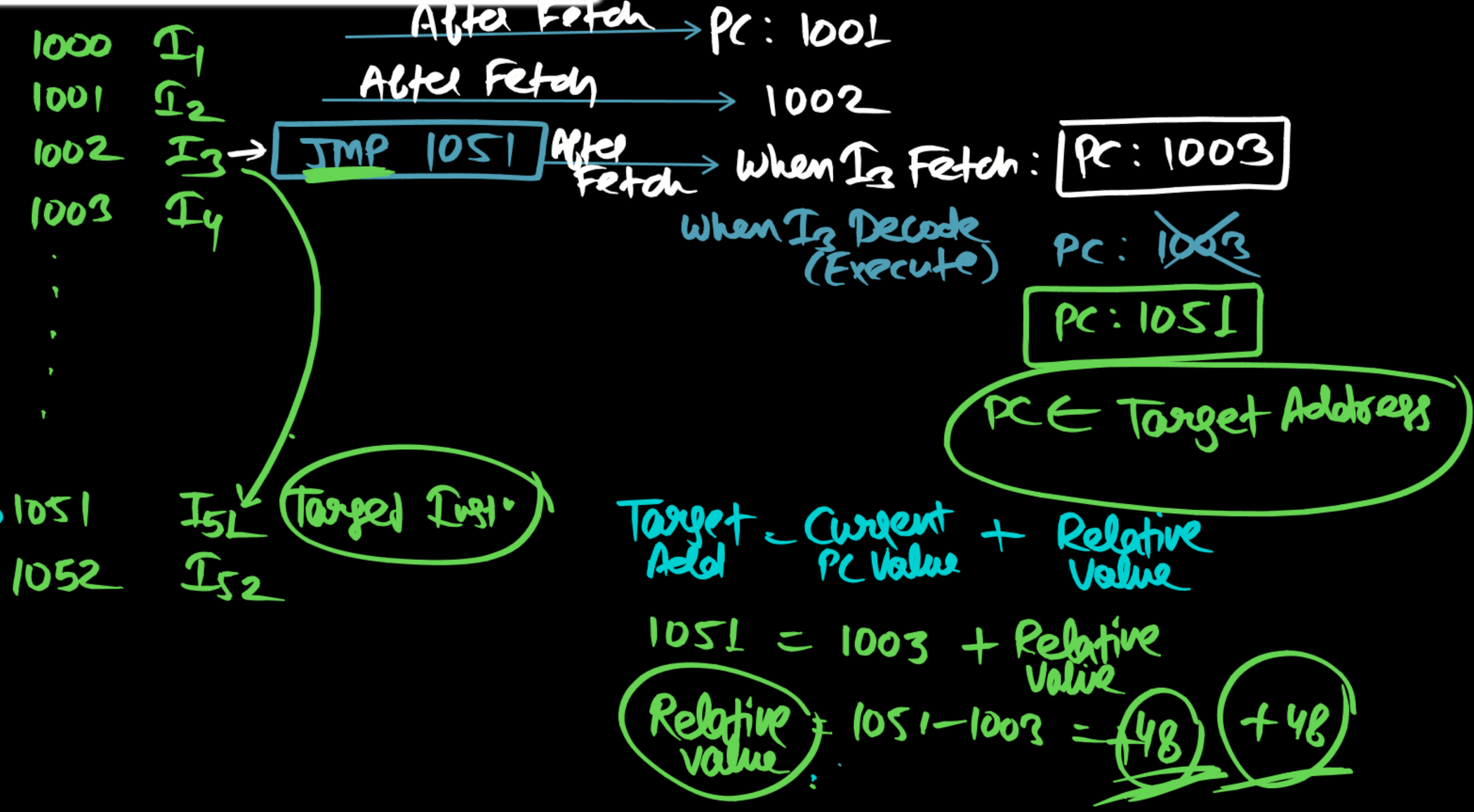
PC Relative AM

Target Address = Current PC Value + A.F [OFFSET] (Relative Value)

Relative Value = +ve [Forward Jumping/Branching]
= -ve [Backward Jumping/Branching]

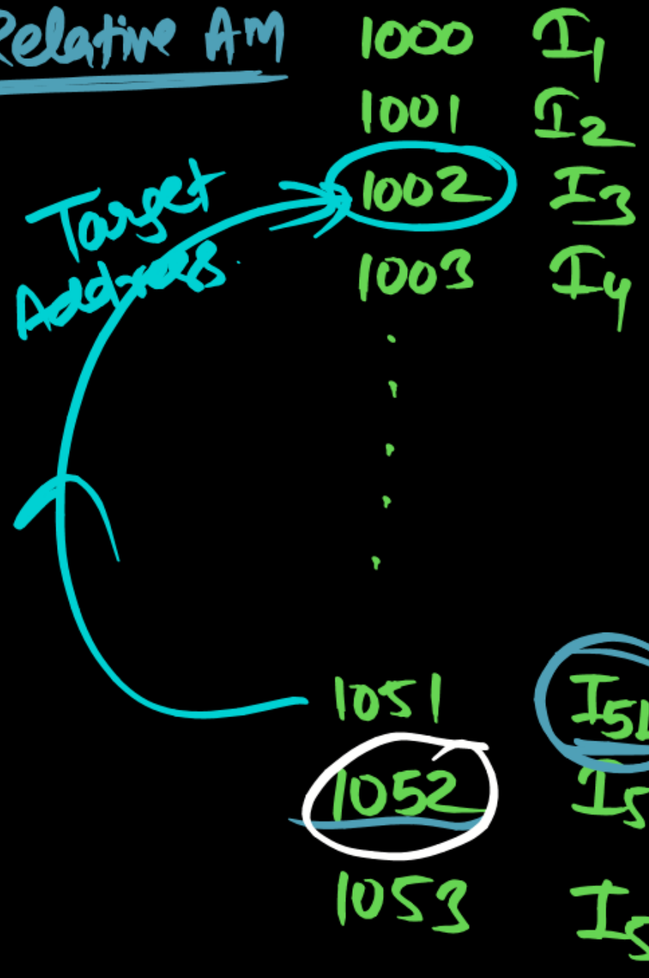
Transfer of Control AM

PC Relative AM



Transfer of Control AM

PC Relative AM



$T.A = PC\ Value + Relative\ Value$

$1002 = 1052 + Relative\ Value$

$Relative\ Value = 1002 - 1052$

$= -50$ Backward Jumping.

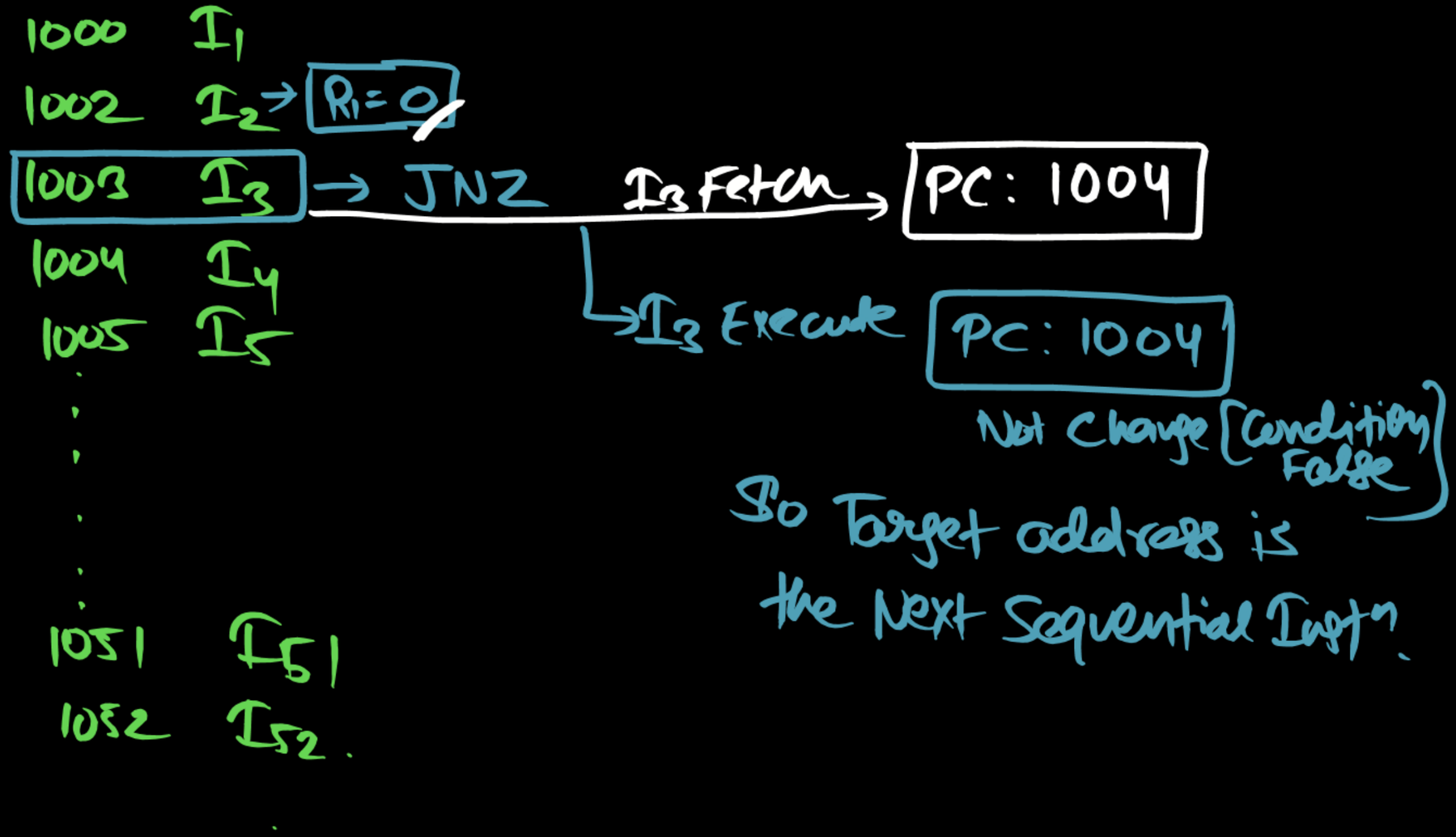
I_{51} Fetch PC : 1052

But I_{51} Execute : PC : ~~1052~~ 1002

Transfer of Control AM

Note: Condition checking on Status of Previous Instruction

Conditional Branch



Not change [Condition False]

So Target address is the Next Sequential Instn.

Instruction Cycle

① Fetch Cycle

② Decode the Instn

Execute

⇓ if

Branch Instruction

B.I
- JMP
- goto
- skip
- JNZ
- JZ etc

Conditional Branch (Toc)
(e.g. JNZ, JZ)

Unconditional Toc.
(JMP, HALT)

At the time execution Condition True (or) False.

PC Relative AM

Unconditional Toc (Branch)

PC Value is updated as Target Address & Now PC Execute Target Instruction as Next Instruction.

Conditional Toc (Branch)

If Condition True then PC value is updated as Target Address & Execute the Target Instruction as Next Instruction.

If Condition False then PC value Not update as Target Address & Execute the Next Sequential Insta as a Next Instruction.

PC Relative AM

In Relative AM

Target Address = Current PC value + Relative value.

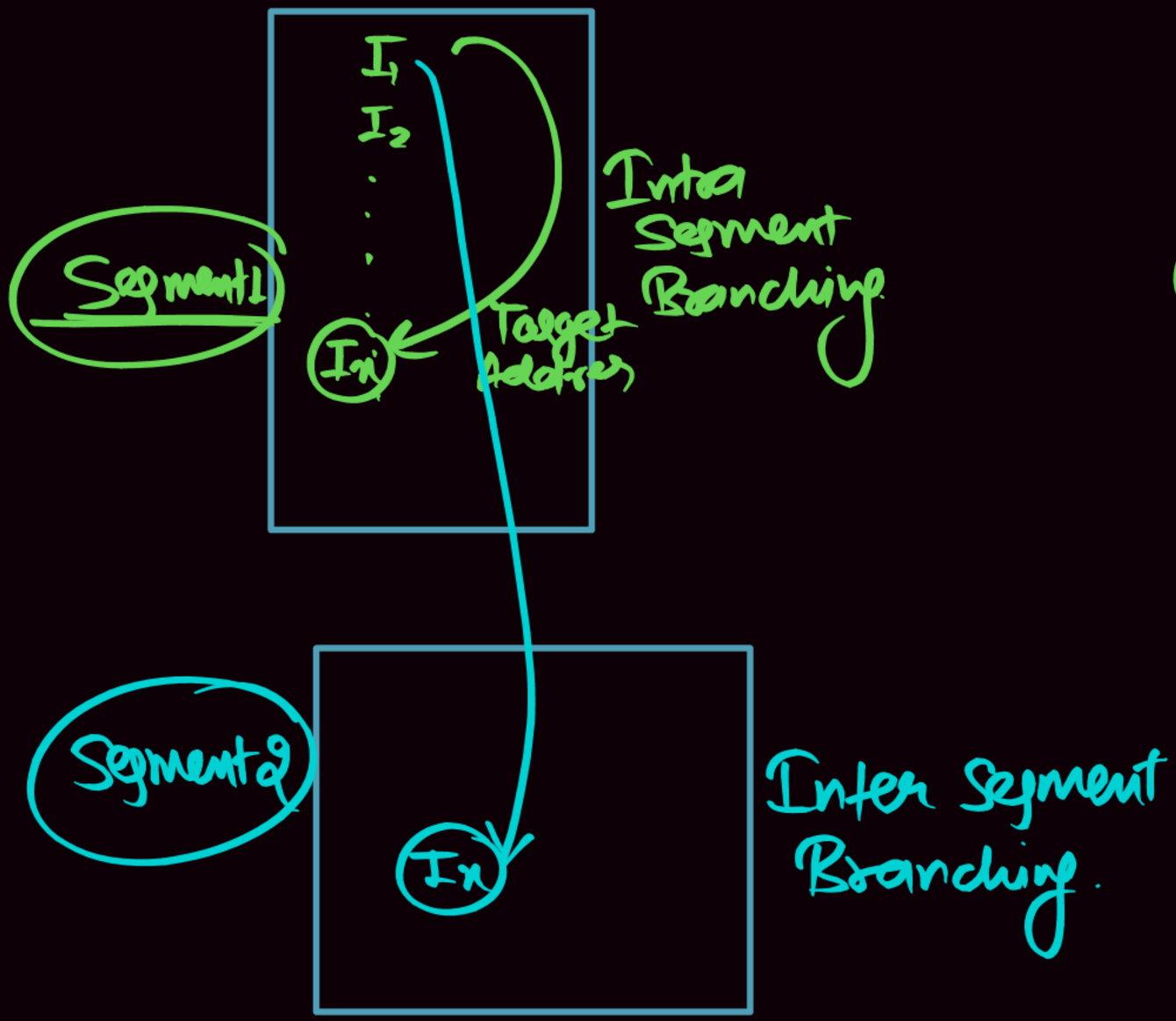
PC Relative AM

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.

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.



2^{20} cells

(eg) 8086 Uprocessor has 1MB Memory its divided into 16 Segment each with 64KB

16 X 64 KB

2^4 2^{16}

2^{20}

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)?

Starting Address = 243048

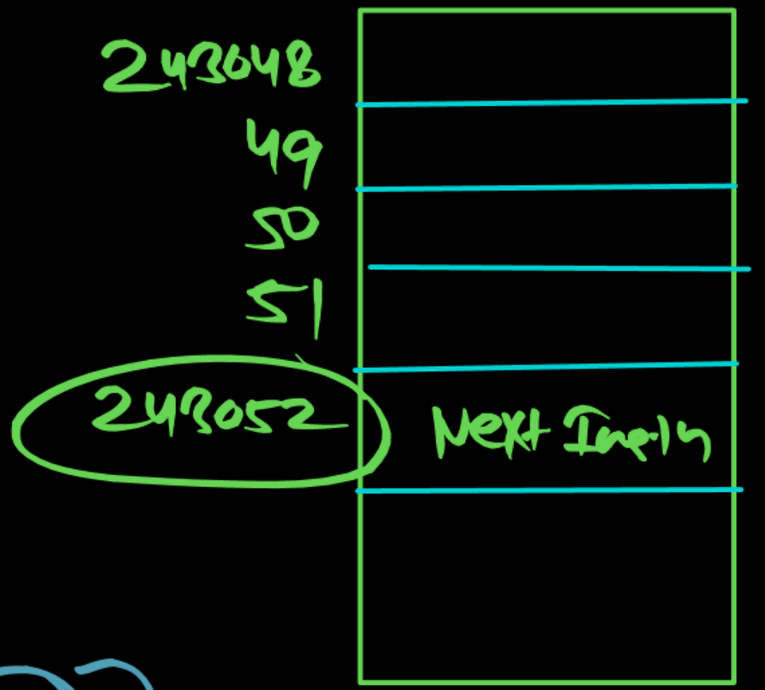
Instr Size = 4 Byte

Relative Value = −8

Target Addr = PC Value + Relative value

= 243052 − 8

= 243044 → (Backward Branching)



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

Instruction Size = 4Byte

Word Size = 16bit [2Byte]

Starting address = 900

Word Addressable Mem.

1 Instn Size = 2 Words

Relative Value = -32

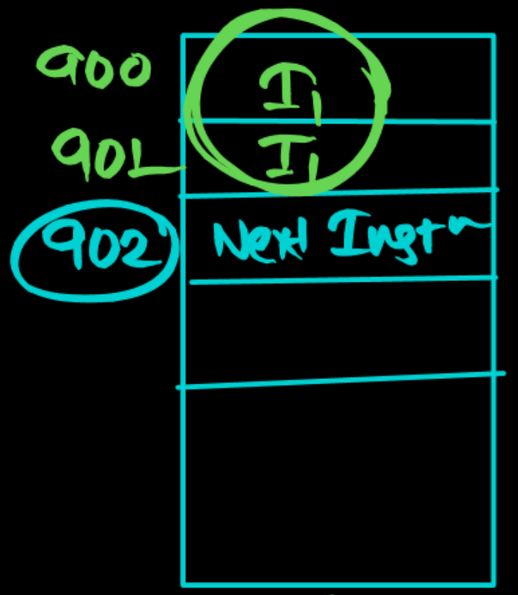
PC Value = 902

Target Add = PC value + Relative Value

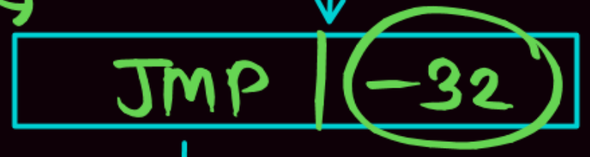
= 902 + (-32)

Target address = 870 Any

Any



Decoding → $I_1 \Rightarrow$ Decoder



Branch Instn

PC-Relative AM.

$$\text{Target Address} = \text{PC Value} + \text{A.F [Relative Value]}$$

$$= 902 - 32.$$

Target Adel = 870

Base Register = 500.

$$\text{Target address} = \text{B.R Value} + \text{Relative value}$$

$$= 500 + (-32)$$

$$= 500 - 32$$

$$= 468 \text{ Ans}$$

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?

Sol^n

16 bit Instn = 1 Word = 16 bit

Starting Add = 900

OPCODE	AF
8 bit	8 bit

PC Value = 902

Target Addr = 614

Target Addr = PC Value + Relative Value

614 = 902 + Relative Value

Relative Value = 614 − 902

= −288 Ans

(i) Befor Feten PC: 900

After Instn Feten PC: 902

AFTER Execution PC: 614

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.

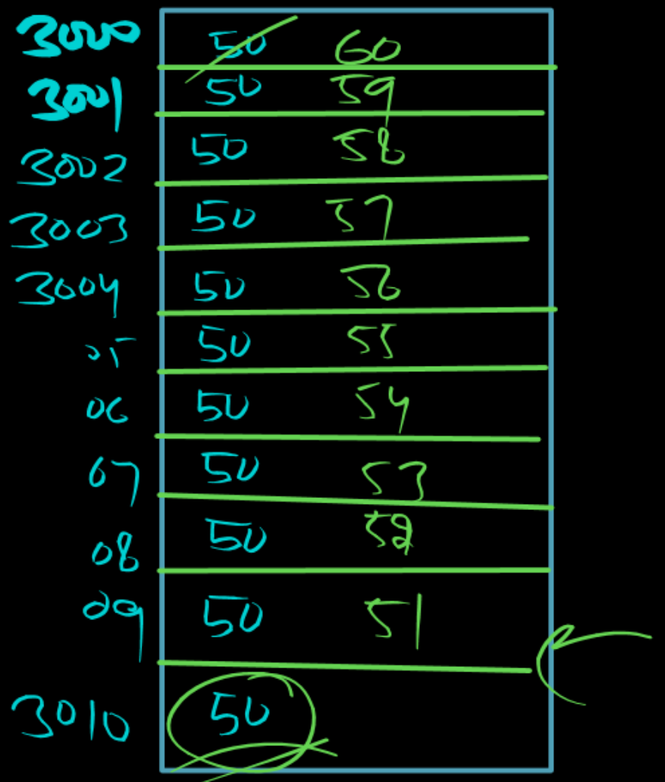
Instruction	Semanties	Instruction Size (bytes)
MOV R1, (5000)	R1 ← MEMORY [5000]	4
MOV R2, (R3)	R2 ← MEMORY [R3]	4
ADD R2, R1	R2 ← R1 + R2	2
MOV (R3), R2	MEMORY [R3] ← R2	4
INC R3	R3 ← R3 + 1	2
DEC R1	R1 ← R1 - 1	2
BNZ 1004	Branch if not zero to the given absolute address	2
HALT	Stop	1

1000 – 1003
1004 – 1007
1008 – 1009
1010 – 1013
1014 – 1015
1016 – 1017
1018 - 1019
1020

Condition Branching

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]

Ans (50)

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

Assume Starting address 1000.

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

1000 - 1003

1004 - 1007

1008 - 1011

1012 - 1015

PC = 1016

Target address = 1000

1000 = 1016 + Relative value (Offset)

Offset = -16 Ans

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

OPCODE Itself.

Acc. is Special Purpose Register Implicitly Access. No explicit Address Required.

[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?

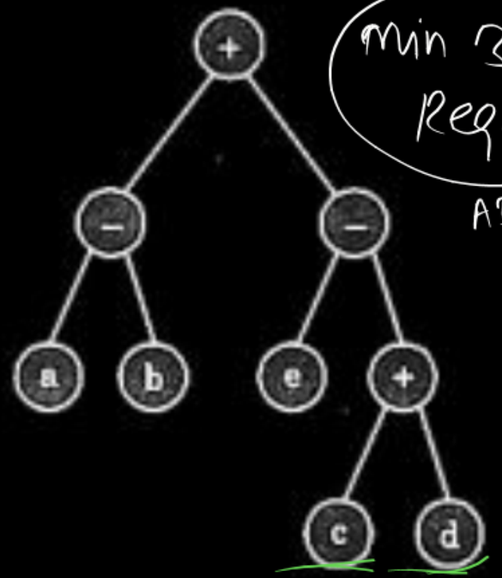
[GATE-2011-CS: 2M]

A 2

B 9

C 5

D 3



Min 3 Reg Required

ADD $R_1 R_2$

LOAD R_1 C
LOAD R_2 D
ADD $R_1 R_1 R_2$; $R_1 \in c+d$
LOAD R_2 e
SUB $R_2 R_2 R_1$ $R_2 \in e-(c+d)$
Load R_1 a
load R_3 b
SUB $R_1 R_1 R_3$; $R_1 = a-b$

ADD $R_3 R_1 R_2$
OR
ADD $R_1 R_1 R_2$
OR
ADD $R_2 R_1 R_2$

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

$R_1 = a$ (1)

$R_2 = b$ (10)

$R_3 = c$ (20)

$a = 1$

$b = 10$

$c = 20$

I. $d = a + b$

II. $e = c + d$

III $f = c + e$

IV $b = c + e$

V $e = b + f$

VI $d = 5 + e$

VII return $d + F$

I. $R_1 \leftarrow R_1 + R_2$ $R_1(d)$

II. $R_1 \leftarrow R_3 + R_L$ $R_1(e)$

III: $R_2 \leftarrow R_3 + R_1$ $R_2(f)$

IV $R_2 \leftarrow R_3 + R_1$ $R_2(b)$

V $R_1 \leftarrow R_2 + R_2$ $R_1(e)$

VI $R_3 \leftarrow 5 + R_L$ $R_3(d)$

VII Return R_3 R_2

3 Register

OR

III $R_2 \leftarrow R_3 + R_1$

IV $R_3 \leftarrow R_3 + R_1$

R_3 R_2

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)

ⓐ ADD (X)– , X

ⓑ ADD (X) , (X)–

ⓒ ADD (–X) , (X)+

ⓓ ADD (–X) , X

STACK

ADD ⇒ POP (TOS)
POP (TOS)
⊕
PUSH TOS.

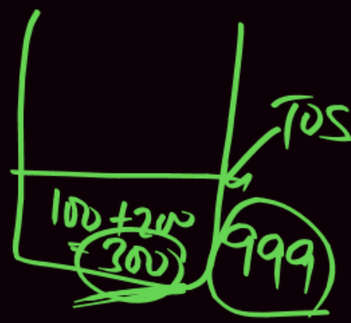
ⓐ ADD (X)–

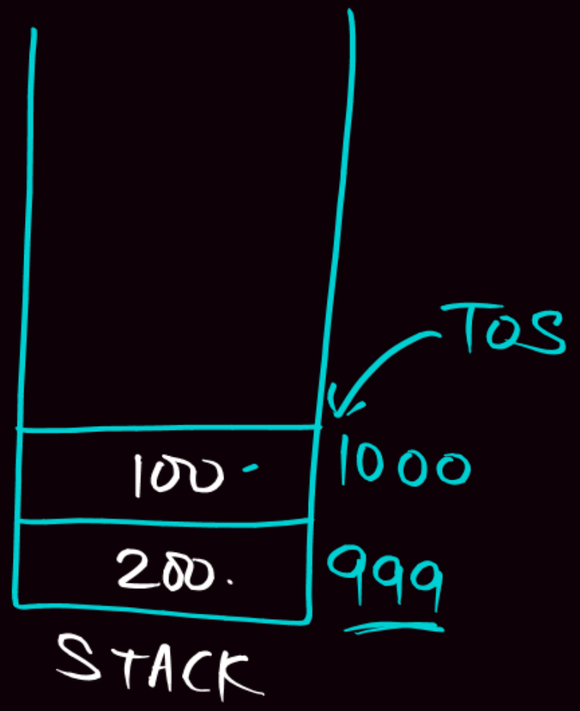
M[1000] ∈ TOS (POP) 100 then X = 999

Now M[999] ∈ TOS (POP) 200

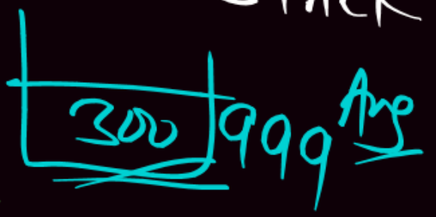
ALU

PUSH (X) ⇒ M(999) = 300 Ans





Actually Require



Ans

(a) ADD (X)−, X

(b) ADD (X), (X)−

(c) ADD (−X), (X)+

(d) ADD (−X), X.

X = 1000

(b) ADD [1000] = 100

X-(1000) = 100 then X = 999

M(999) = 200 ✗

(c) −X : M(999) = 200

X+ ⇒ M[999] = 200 then X = 1000

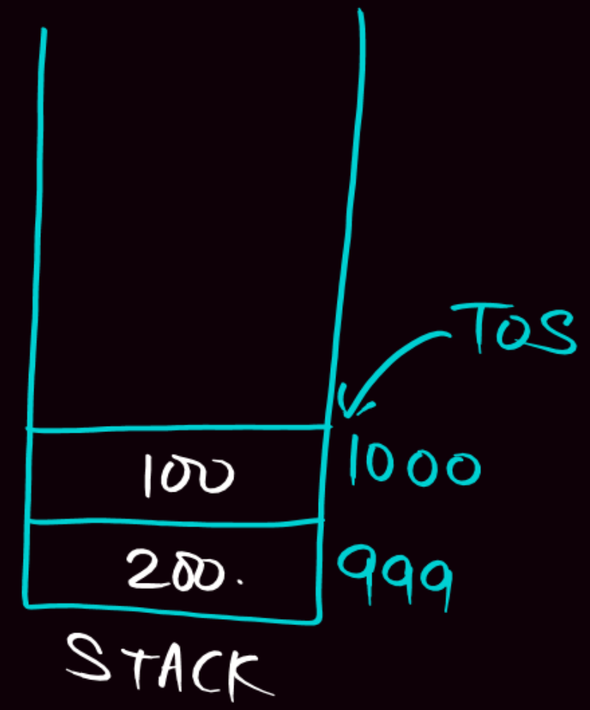
M(1000) = 200 + 200 ✗

(d) ADD (−X), X

−X: 999 ⇒ 200

X: 999 = 200

M(X) = 200 + 200 ✗



X+ : Post Increment (first Fetch the operand from location X, then Increment)

+X : Pre Increment (first Increment & Fetch the operand from X.

X− : Post Decrement

−X : Pre Decrement.

Type of Instn/operation (Instruction Set)

1. Data Transfer Instn/operation

2. Data Manipulation Instn/Operation
   - (i) Airthmatic operation (ADD, SUB, DIV, MUL, Inc etc)
   - (ii) Logical operation (OR, NOR, AND XOR etc)
   - (iii) Shift & Rotate operation [Airthmatical & logical shift]

3. Program Contad Instn (TOC Instn)

THANK YOU!