COMPUTER SCIENCE

Computer Organization and Architecture

Machine Instruction and Addressing Modes

Lecture_07

Vijay Agarwal sir

01 Addressing Modes

ADDRESSING MODE (AM)

WHEN &

WHY AM USED

Type of AM.

Addressing Modes [AM]

⇒ 'EA'.

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

Addressing Modes

Symbols of AM's

AM	Symbolsy
Immediate AM	I (or) #
Direct AM	[ ]
Indirect AM	@ (or) [( )]
Register AM	Register Name
Index Reg AM	Index Reg Name.

Addressing Modes

'MODE field'

OPERAND(DATA) ⇒ Effective Address [EA] (where it is operand ?)

⇓

Actual Address of the Operand.

Address
- → Memory Address
- → Register Address
- → Immediate field [constant value].

Addressing Modes [AM]

1. Immediate AM
2. Direct / Absolute AM.
3. Memory Indirect AM.
4. Register Direct AM
5. Register Indirect AM.
6. PC-Relative AM
7. Base-Reg AM
8. Index-Reg AM

Displacement AM.

9. Implied / Implict AM
10. Auto Decrement AM
11. Auto Increment AM.

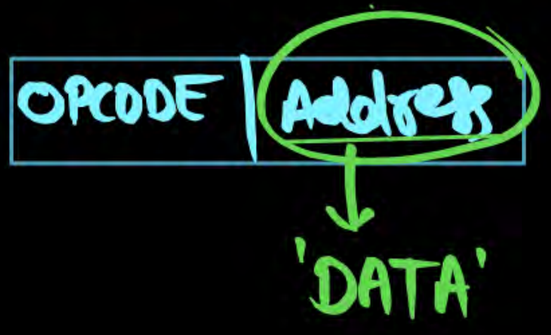
Addressing Modes

① Immediate AM: [I or #]

In this Addressing operand are present in the Address field of the Instruction.

OR

In this AM operand are present in the Instruction itself.

I: 

Note: Immediate AM are Used to Access the Constant & Initialize the Register With Constant Value.

Addressing Modes

ADD R_1, #500 ← Constant.

(or)

ADDI	R_1	500.

$R_1 \leftarrow R_1 + 500$

MVI

MOV R_0 #50

(or)

MOVI R_0, 50

MVI R_0, 50

$R_0 \leftarrow 50$

Addressing Modes

Immediate AM:

MVI R_2, 4000

OR

MOV R_2, #4000.

$R_2 \leftarrow 4000$

Eg MVI R_0, 600

MOV R_0, #600

OR

MOVI R_0, 600

$R_0 \leftarrow 600$.

Eg ADDI R_1, 4000

OR

ADD R_1, #4000.

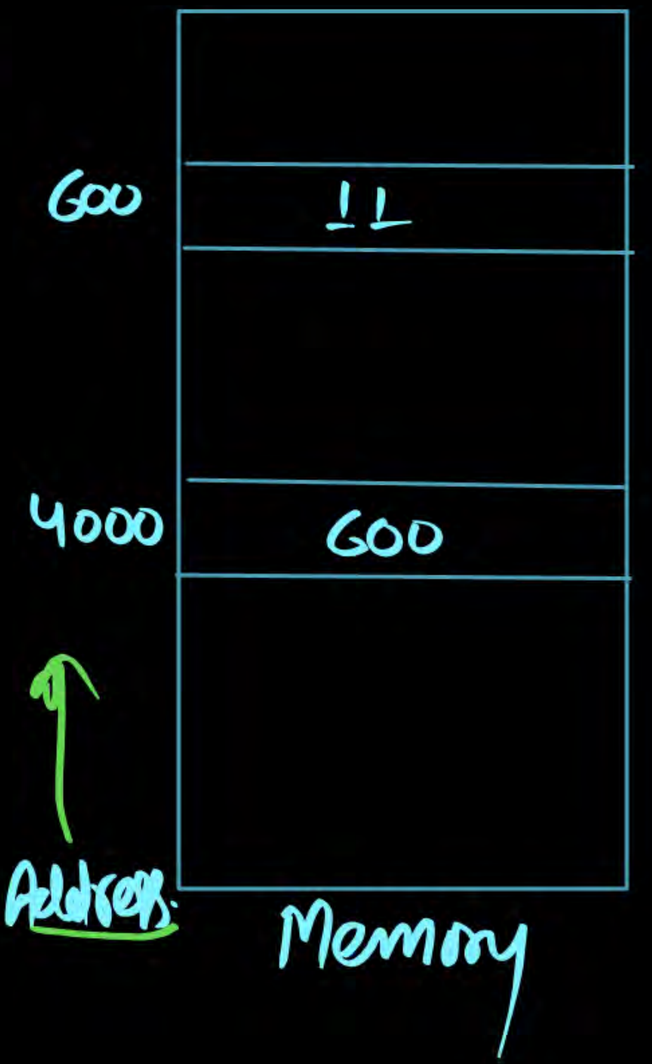
$R_1 \leftarrow R_1 + 4000$.

Eg ADDI R_4, 600

OR

ADD R_4, #600

$R_4 \leftarrow R_4 + 600$.



Immediate AM has Some Limitations:

① Immediate AM Can never be Used as a Destination address It Can be Used only Source Address, Because Destination must Required Storage & Constant does not have Any Storage.

Eg ADDI R_1, 500
$R_1 \leftarrow R_1 + 500$;

Eg ADDI 500 R_1
$500 \leftarrow 500 + R_1$ ✗

MOVI 1000 R_2 ✗
$1000 \leftarrow R_2$

② The Range of Constant is limited by Address field.
n bit Immediate field Unsigned Range = 0 to $2^n - 1$
Signed Range = -2^{n-1} to $+2^{n-1} - 1$

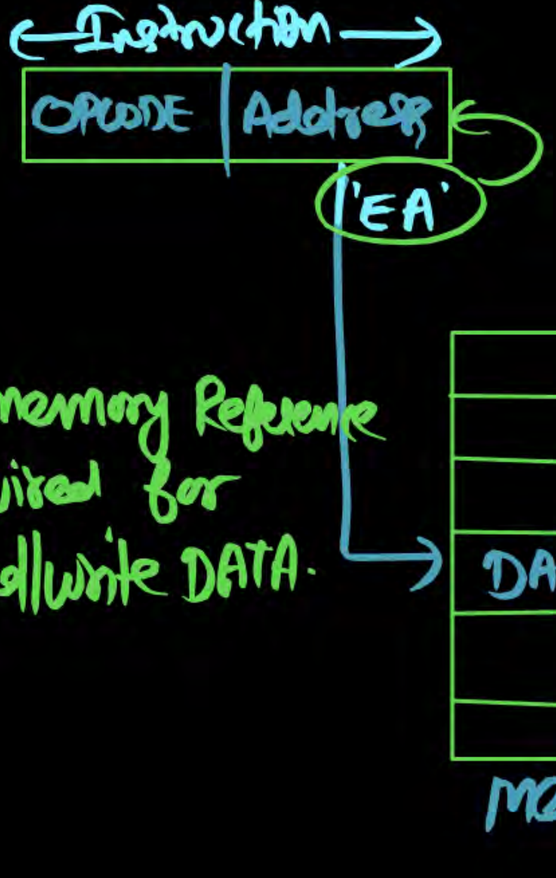
Addressing Modes

② Memory Direct (or) Absolute AM:

In this Addressing Mode operand are present in the memory & Address field of the Instruction Contain the Effective Address.

(OR)

Instruction Contain the effective Address.



Note: This AM is used to Access the Variable

Note: 1 Memory Reference Required for Read/write DATA.

Addressing Modes

Absolute / Direct AM.

ADD R_1 500

$R_1 \leftarrow R_1 + M[500]$

eg

MOV R_1 [50]

$R_1 \leftarrow M[50]$

Addressing Modes

Absolute / Direct AM.

(eg) MOV R_2 [4000]

$R_2 \in M[4000]$

$R_2 \in 600$

(eg) MOV R_0 [600]

$R_0 \in M[600]$

$R_0 \in 11$

(eg) ADD R_1, [4000]

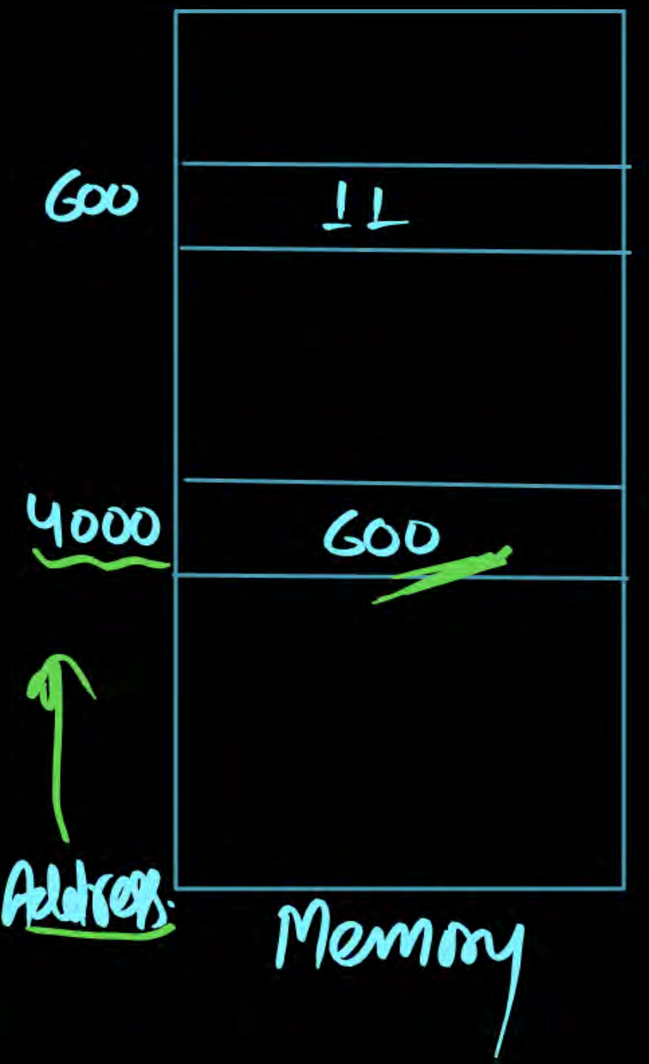
$R_1 \in R_1 + M[4000]$

$R_1 \in R_1 + 600$

(eg) ADD R_1 [600]

$R_1 \in R_1 + M[600]$

$R_1 \in R_1 + 11$



Direct Am

ADD [6000] [4000] [5000]

M[6000] ← M[4000] + M[5000]

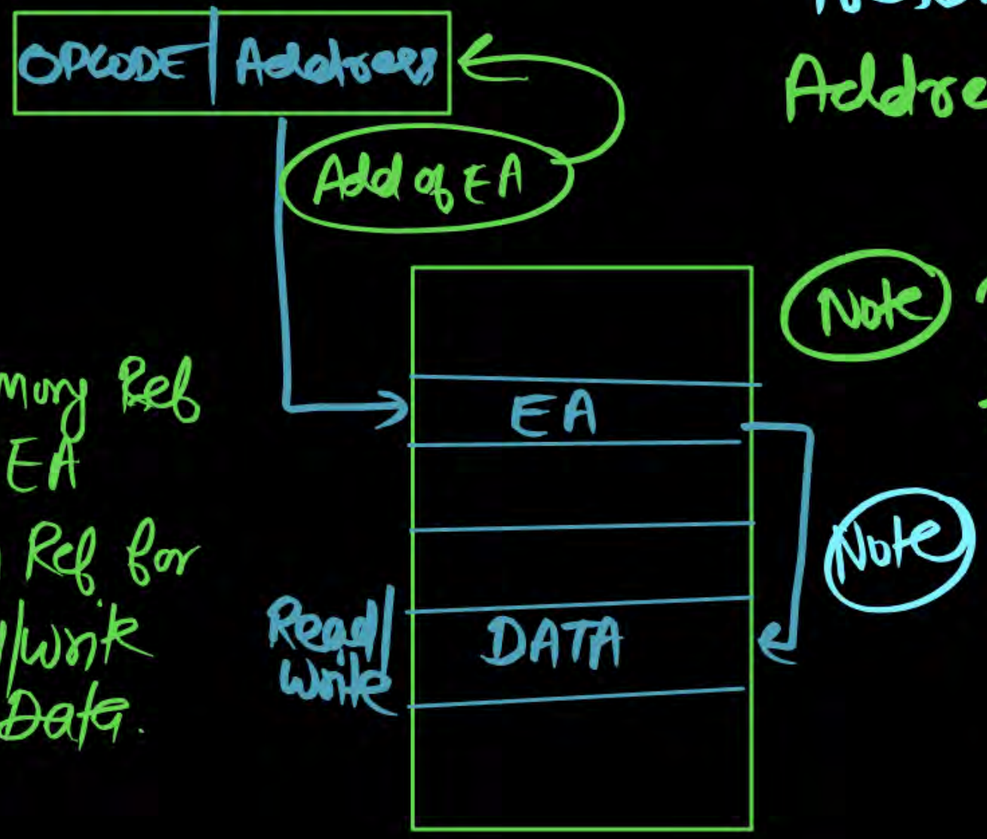
↑ 1 Mem Ref for Write

↑ 1 Mem Ref for Read

↑ 1 Mem Ref for Read

Addressing Modes

③ Memory Indirect AM: In this AM operand are present in the Memory, Effective Address is also present in memory, Instruction Contain Address of Effective Address.



Note Indirect AM are used to Access the Pointers.

2 Memory Reference is Required in Memory Indirect AM.

Addressing Modes

Memory Indirect

(eg) MOV R_0 @500

$R_0 \leftarrow M[[500]]$

(eg) ADD R_1 @500

$R_1 \leftarrow R_1 + M[[500]]$

Addressing Modes

Memory Indirect AM.

eg MOV R_2 @4000

$R_2 \in m[[4000]]$

$R_2 \in M[600]$

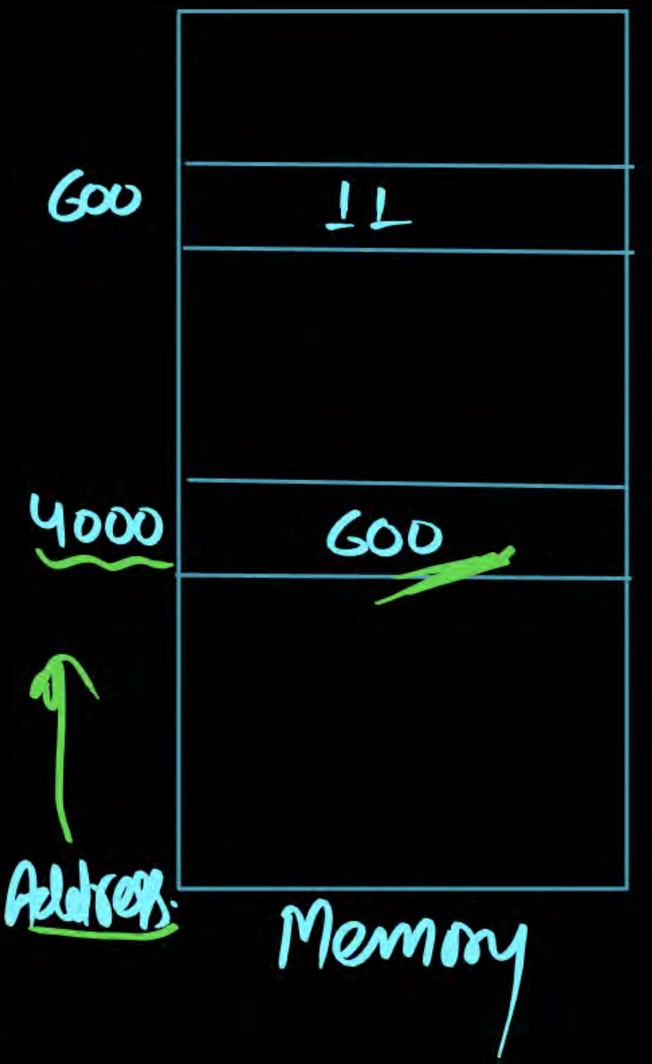
$R_2 \in 11$

eg ADD R_1, @4000

$R_1 \in R_1 + [[4000]]$

$R_1 \in R_1 + m[600]$

$R_1 \in R_1 + 11$



Addressing Modes

ADD [6000] @7000 ((8000))

[6000] ↑ Direct AM

@7000 ↓ Memory Indirect AM

((8000)) ↑ Memory Indirect AM.

$M[6000] \leftarrow M[[7000]] + M[[8000]]$

↑ 1 Mem Ref for Write DATA.

1 Mem Ref for EA
1 Mem Ref for Read

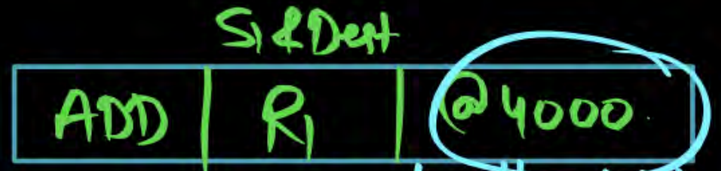
9 1 Mem Ref for EA
1 Mem Ref for Read DATA.

---

5 Mem Ref.

Addressing Modes

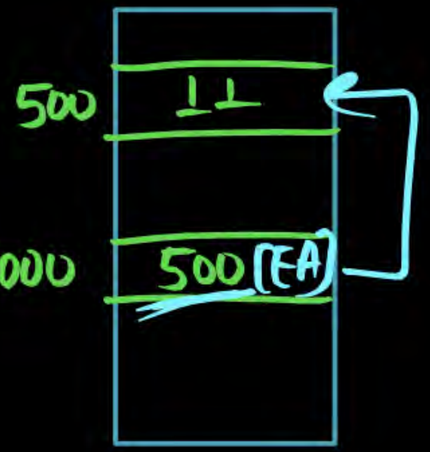
① ADD R_1 @4000



$R_1 \leftarrow R_1 + [[4000]]$

$R_1 \leftarrow R_1 + M[500]$

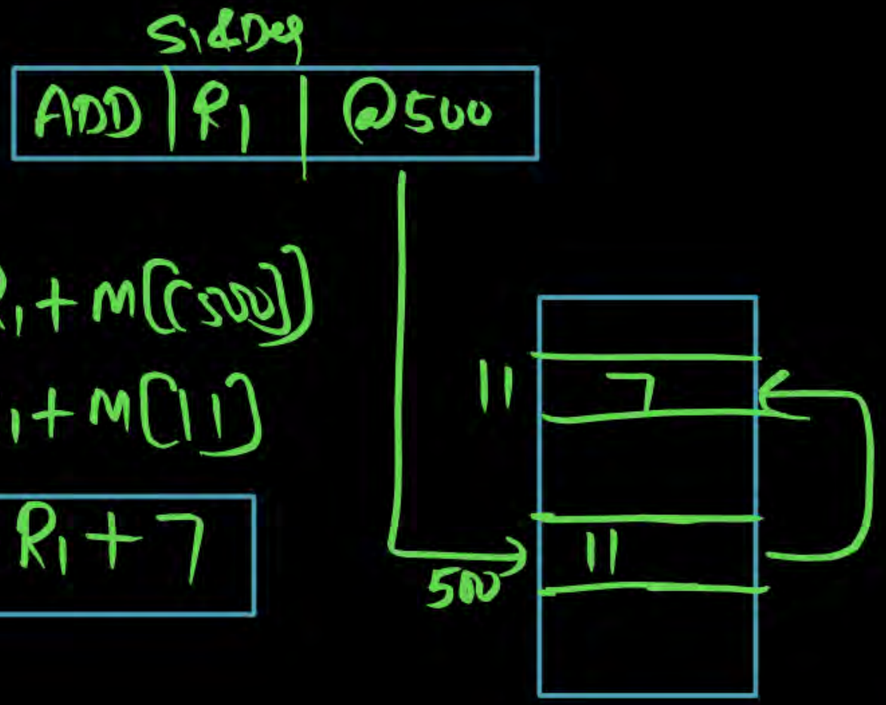
$R_1 \leftarrow R_1 + 11$.

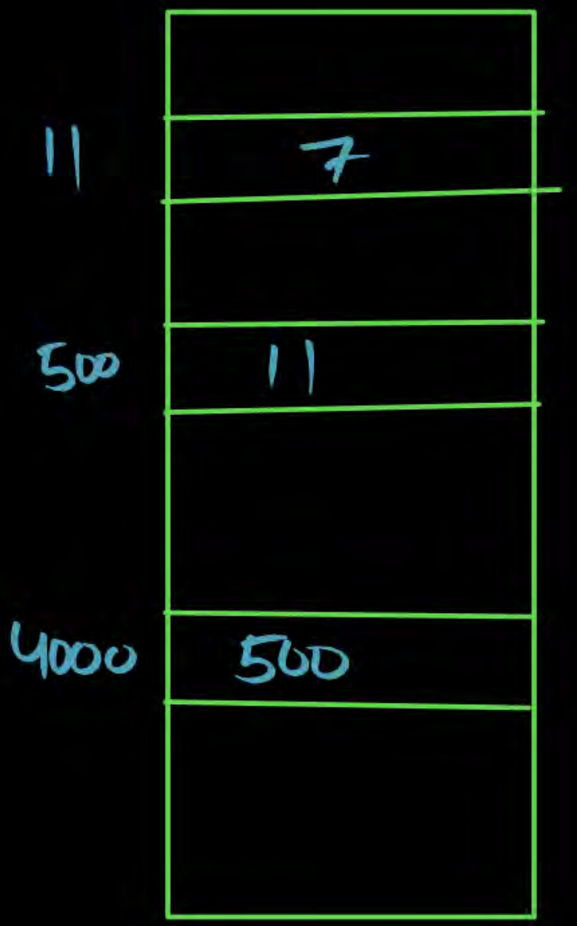


2 Mem Ref.

Addressing Modes

② ADD R_1 @500





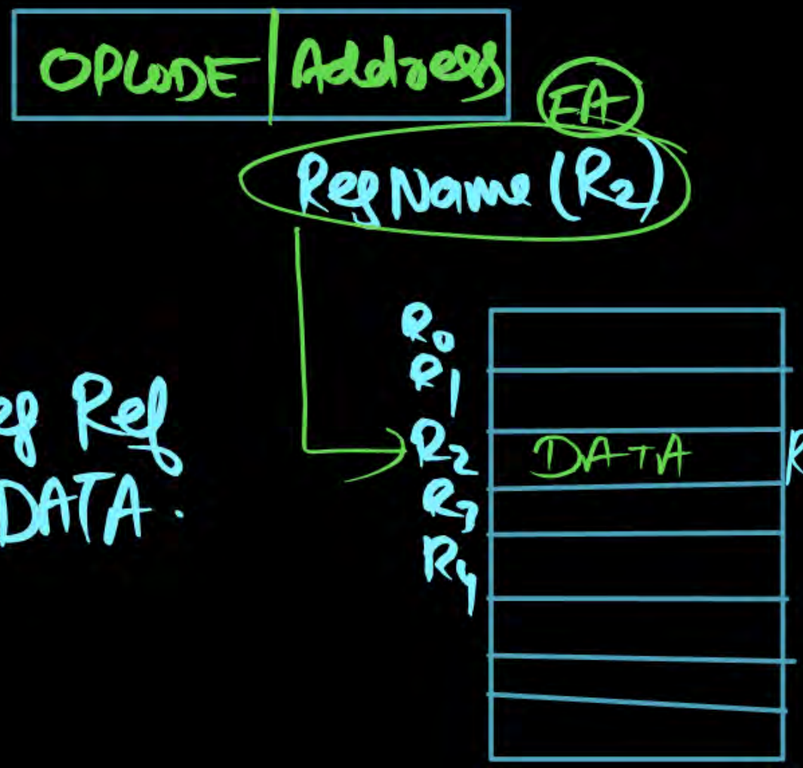
Addressing Modes

(4) Register Direct AM :

This AM is Similar to memory Direct AM But in this AM operand are present in the Registers instead of memory.

(OR)

In this AM operand are present in the Register that Register Address (Register Name) Maintained in the Address field of the Instruction.



1 Reg Ref for DATA.

1 Register Reference Required for Read/write DATA.

Addressing Modes

Eg MOV R_0 R_1

$R_0 \leftarrow R_1$

1 Reg Ref for write.

1 Reg Ref for Read.

Addressing Modes

ADD [6000] R_1 @7000

[6000] → Mem Direct

R_1 → Reg Direct

@7000 → Memory Indirect

$M[6000] \leftarrow R_1 + @7000$

$M[6000]$ → 1 Mem Ref for Write DATA

R_1 → 1 Reg Ref for Read DATA

@7000 → 1 Mem Ref for EA, 1 Mem Ref for Read DATA

---

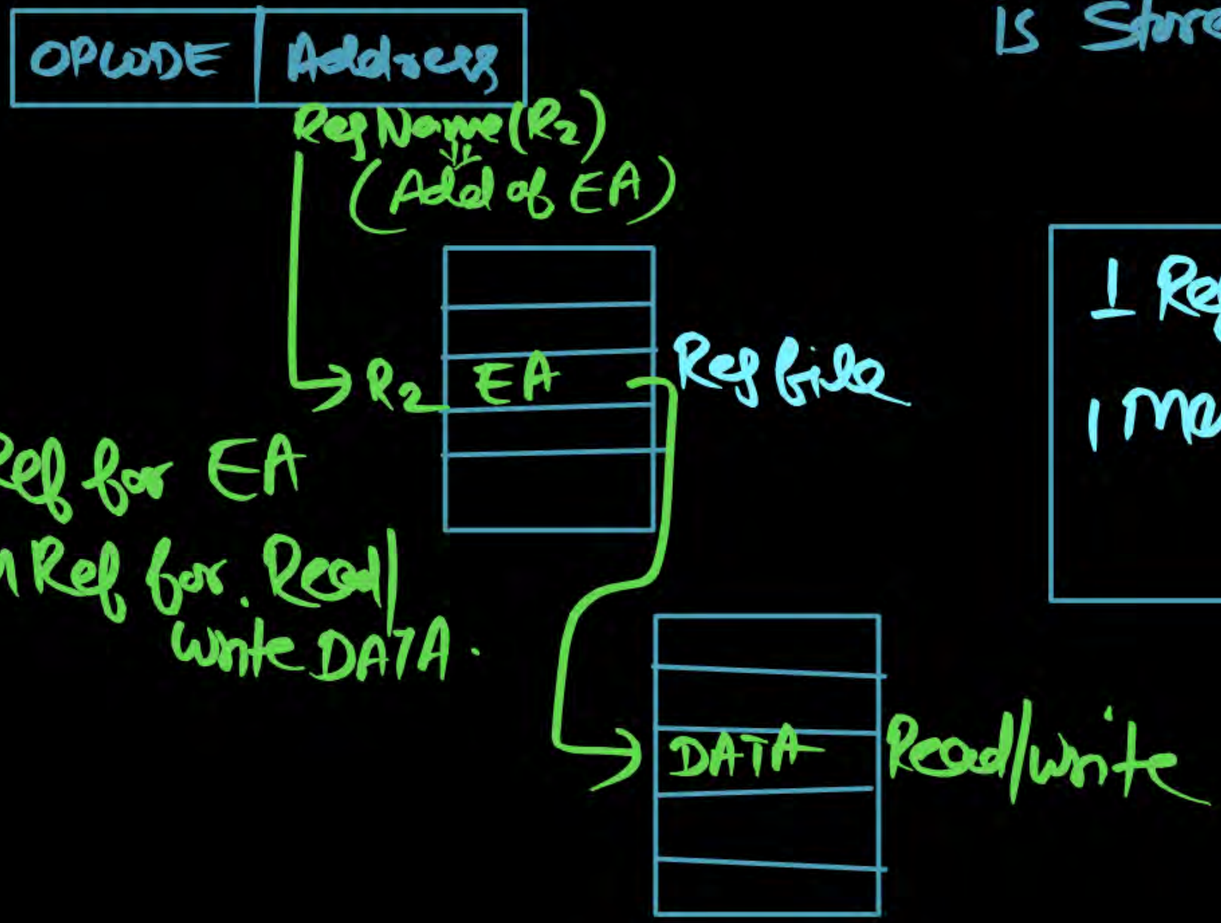
3 Memory Reference

1 Register Reference

Addressing Modes

⑤ Register - Indirect AM :

In this AM operand are present in the Memory & Effective Address is stored in Register.



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

1 Register Ref for EA
1 memory Ref for Read/write DATA.

Addressing Modes

$R_2 = 400$

Register Indirect

MOV R_1 @R_2

$R_1 \in M[R_2]$

$R_1 \in M[400]$

$R_1 \in 9$

Register Indirect @R_2 = $M[R_2]$ (1 Reg Ref for R_2, 1 mem Ref)

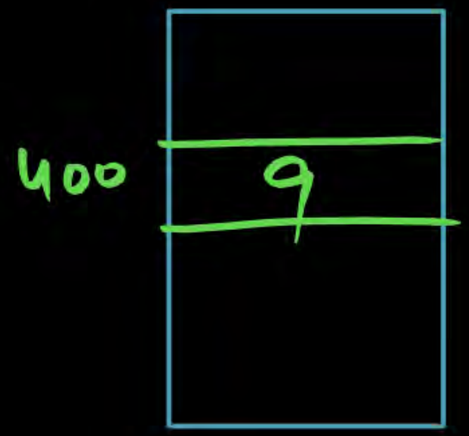
mem Indirect @1000 = $M[[1000]]$

Register Direct

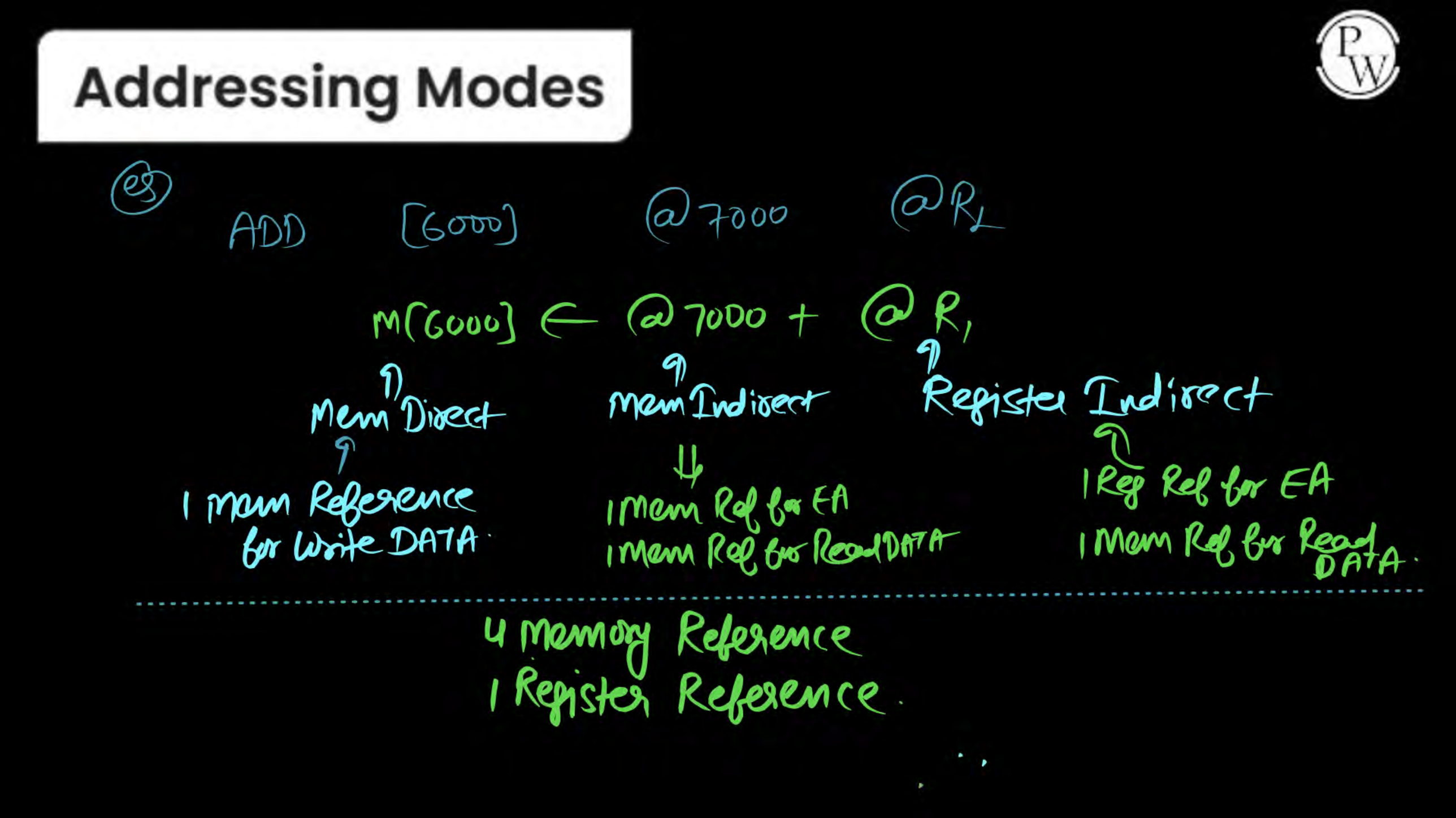
MOV R_1, R_2

$R_1 \in R_2$

$R_1 \in 400$.



Addressing Modes



Addressing Modes

Q) Why Register Indirect AM used instead of Memory Indirect AM ?

Soln ① To Shorten the Instruction Length (Size).

② Accessing the Register is very fast Compare to Memory.

Eg) If Memory = 16GB then Memory AF = 34 bit the Instruction size large.
But in the Processor we have 32 or 64 Register so we need 5 or 6 bit Respectively.

So 2 Advantage.

① Instruction Size Reduced.

② Faster Access.

Addressing Modes

⑥, ⑦ & ⑧.

BR : Base Register

R_i or XR : Index Register.

Displacement AM.

① PC Relative AM

$EA = \text{Current PC value} + \text{A.F (OFFSET)}$

OR

$EA = \text{Current PC value} + \text{Relative value}$.

② Base Register AM

$EA = \text{Base Register value} + \text{AF (OFFSET)}$

③ Index - Register AM.

Index Reg. AM

$EA = \text{Index Reg value} + \text{OFFSET [AF]}$

⇓

Array Implementation

Addressing Modes

Index AM ⇒ (Array Implementation)

EA = Index Reg Value + A.F [OFFSET]

'DATA'

1 Reg. Reference for Index Register value.

1 Airthmatic [ALU] Reference for EA Calculation.

1 Memory Ref for Read/Write DATA.

∴,

Addressing Modes

(9) & (10) **Auto Decrement & Increment AM** : This AM is similar to Register Indirect in which Register value Decrement or Increment.

Decrement : Pre decrement [First Decrement Register value then Access the DATA.

Increment : Post Increment. [First Access the Data then Increment in Register Value].

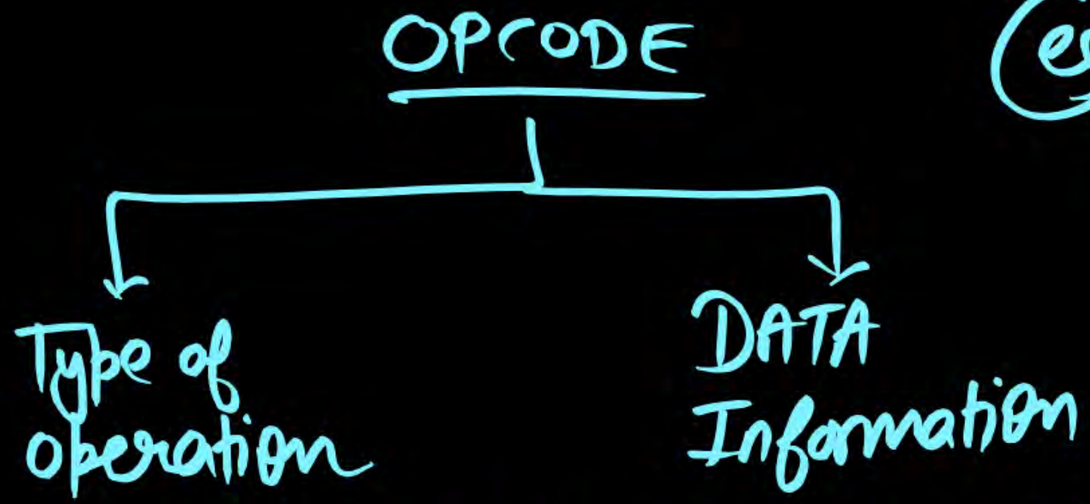
Hint (Job Salary)

Decrement : Negotiation (bagging)

Increment : After 1 year Salary Increased

Addressing Modes

(11) Implied / Implicit AM: In this AM operand (DATA info.) are present in the opcode itself.



(eg) STC : Set Carry (Carry = 1)

CLC : clear Carry (Carry = 0).

Stack Based org used Implied / Implicit AM. INC A

Addressing Modes

V. V. Imp.

Constant — Immediate AM

Variable — Direct AM

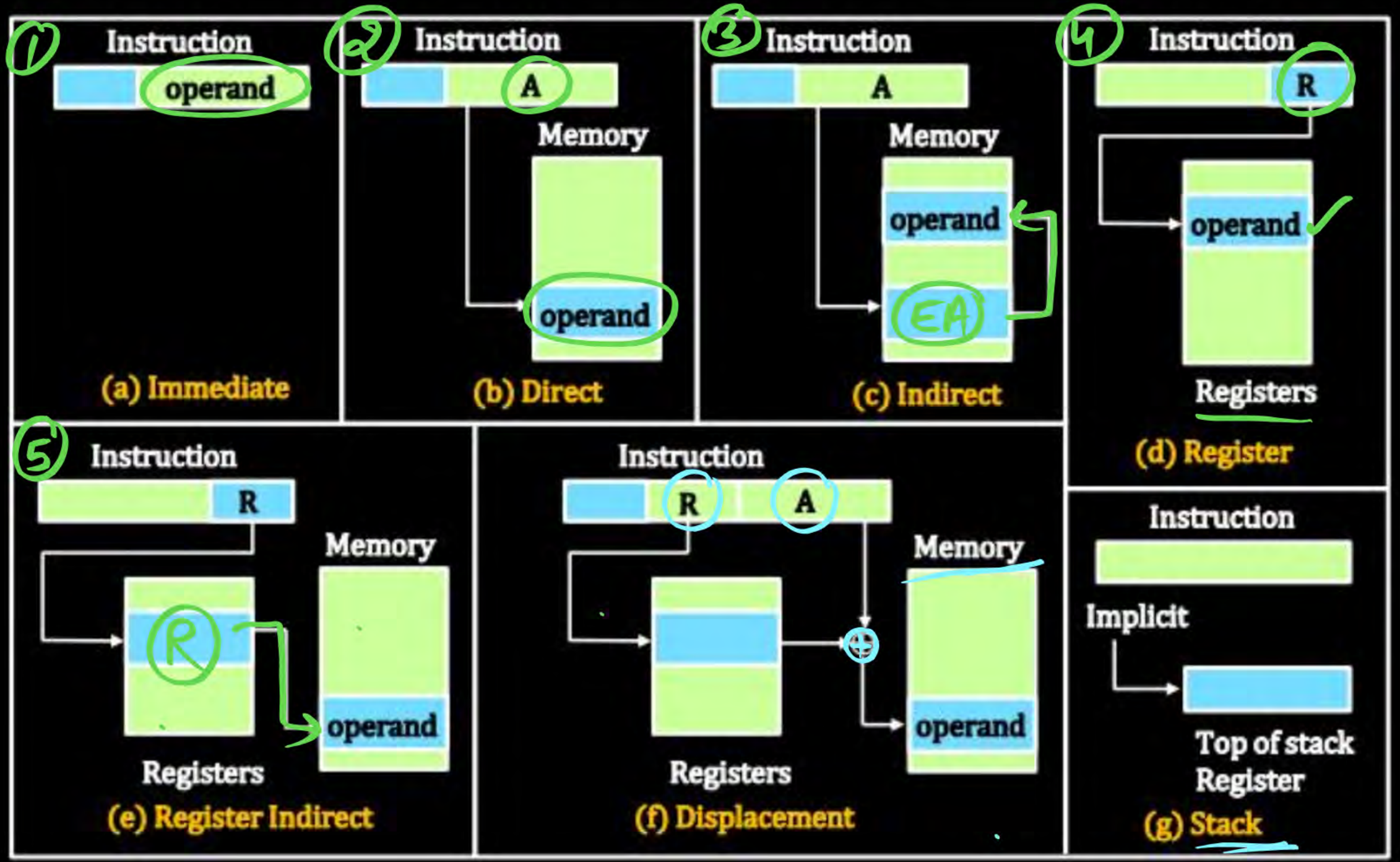
Pointer — Indirect AM

Array — Index AM

Reallocation — Base Reg. AM.

Addressing Modes

1. Immediate
2. Direct
3. Indirect
4. Register
5. Register Indirect
6. Displacement
7. Stack
8. Auto Decrement AM.
9. Auto Increment AM


1
Instruction
operand
(a) Immediate
2
Instruction
A
Memory
operand
(b) Direct
3
Instruction
A
Memory
operand
EA
(c) Indirect
4
Instruction
R
operand
Registers
(d) Register
5
Instruction
R
Memory
R
operand
Registers
(e) Register Indirect
Instruction
R
A
Memory
Registers
operand
(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. Here Memory is Byte Addressable & OPCODE is 8 bit in Size. (All Numbers are in Decimal) With the following Details

Q Calculate / Tell the operand (DATA) & EA in various AM's ?

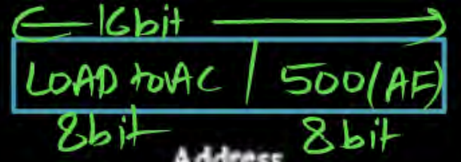
Addressing Mode	Effective Address	Content Of AC
Direct address		
Immediate Operand		
Indirect Address		
Relative address		
Indexed address		
Register		
Register Indirect		
Autoincrement		
Autodecrement		

PC = 200

R1 = 400

XR = 100

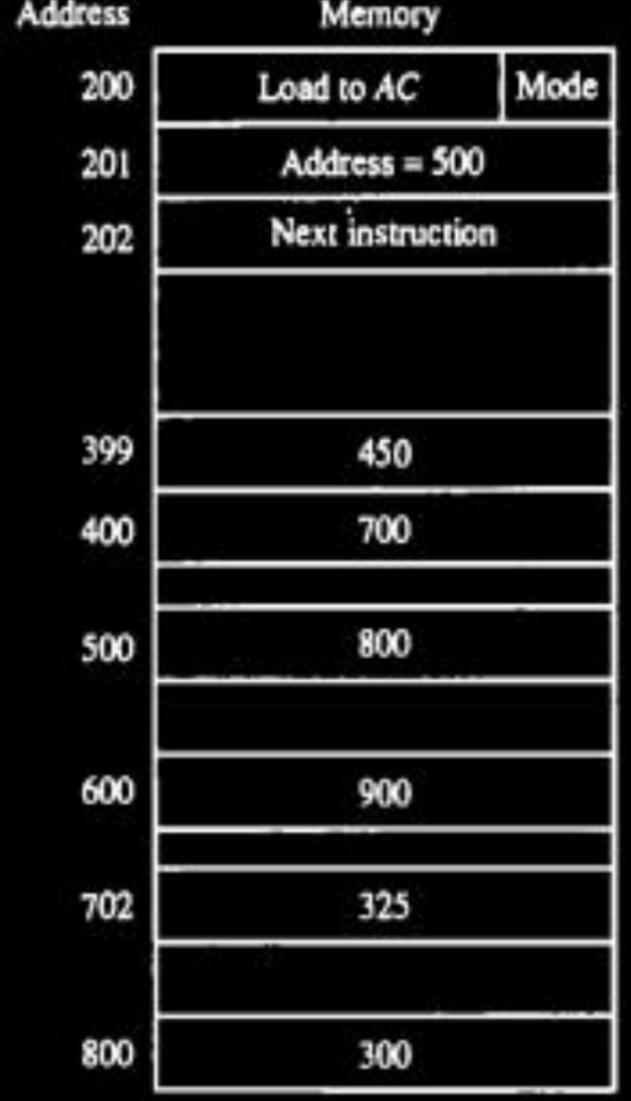
AC



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.

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

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)

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

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

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

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]

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

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] 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

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]	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	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

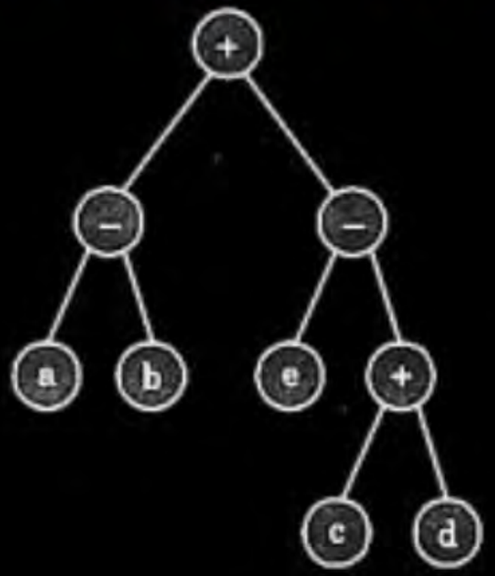
C Only S2 and S3

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!