Que:- Consider the following Statement, Execute on a Register to Reg reference CPU, all the variables are present in the m/y. How many minimum Registers and machine instructions are required to evaluate the Statement.: $X = (A * B) + A$.

I_1: Load r_0, A ;

I_2 ; Load r_1, B ;

I_3 ; MUL , r_1, r_0, r_1 ;

I_4 ; ADD , r_0, r_0, r_1 ;

I_5: store X, r_0 ;

{ 2 Registers and 5 Instⁿ }

2 { 3 Addr Instⁿ } 3 { 2 Addr Instⁿ }

Instruction - Execution Process :-

⇒ To Analyze the execution sequence of an Instruction, let us conpider 8085 micro-processor ap a Reference CPU. It is a Accumulator. CPU with a word length of 8 bit.

Code :

Statement : (A + B)

memory space :

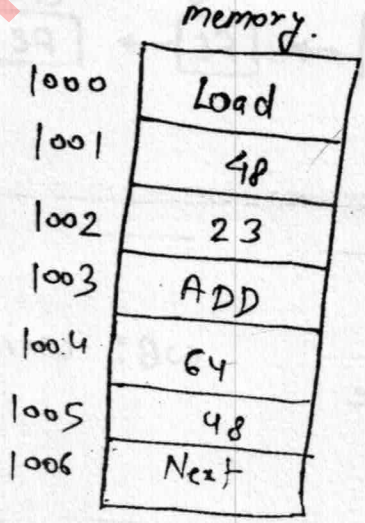
Variable	Addr	cell
A	2348	FE
B	4864	C8

machine code :-

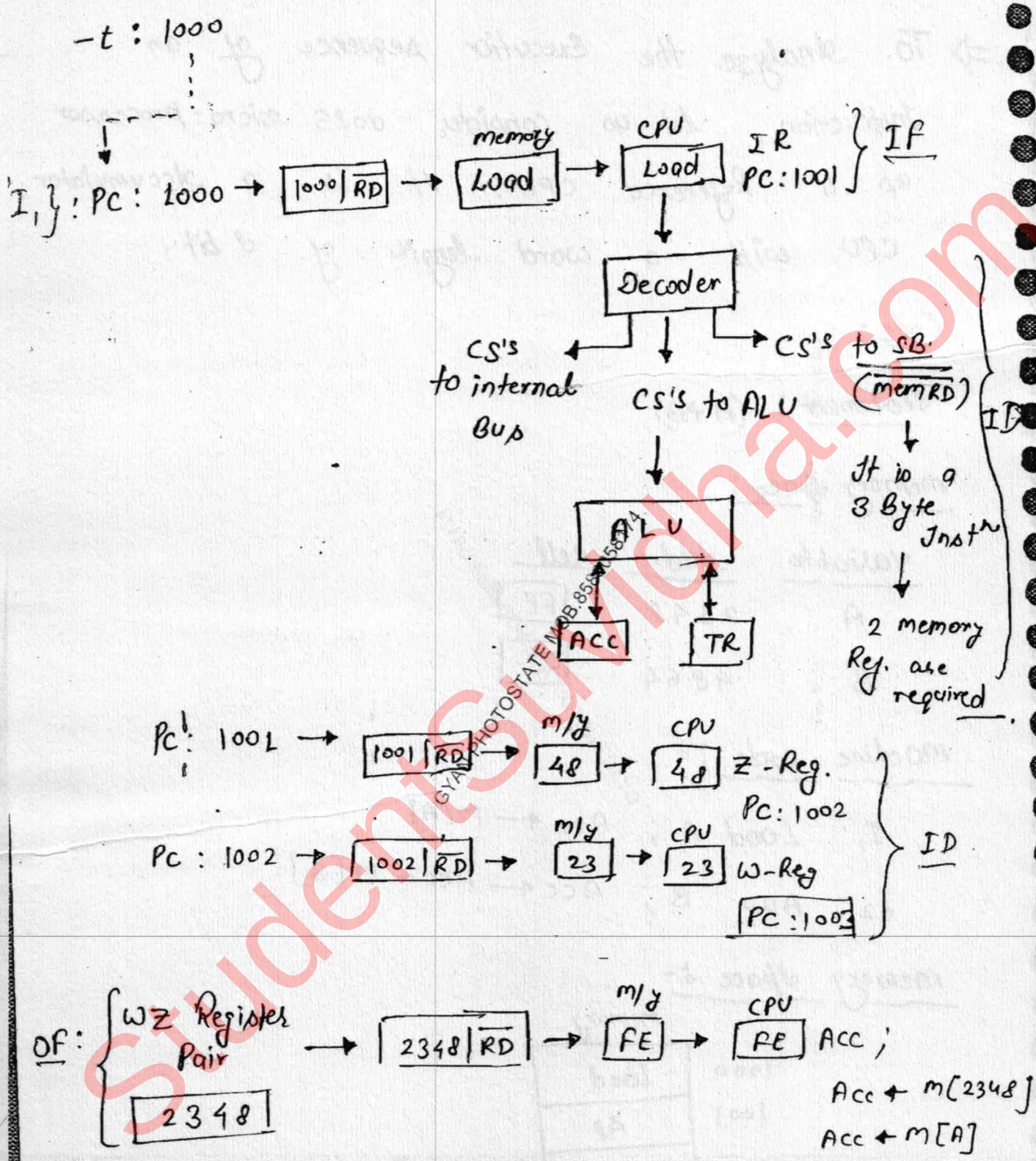
I_1 : Load A ; Acc ← M[A]

I_2 : ADD B ; Acc ← Acc + M[B]

Memory Space :-

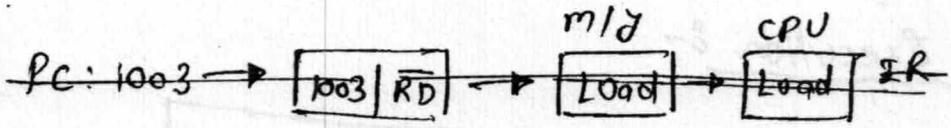


Execution :-

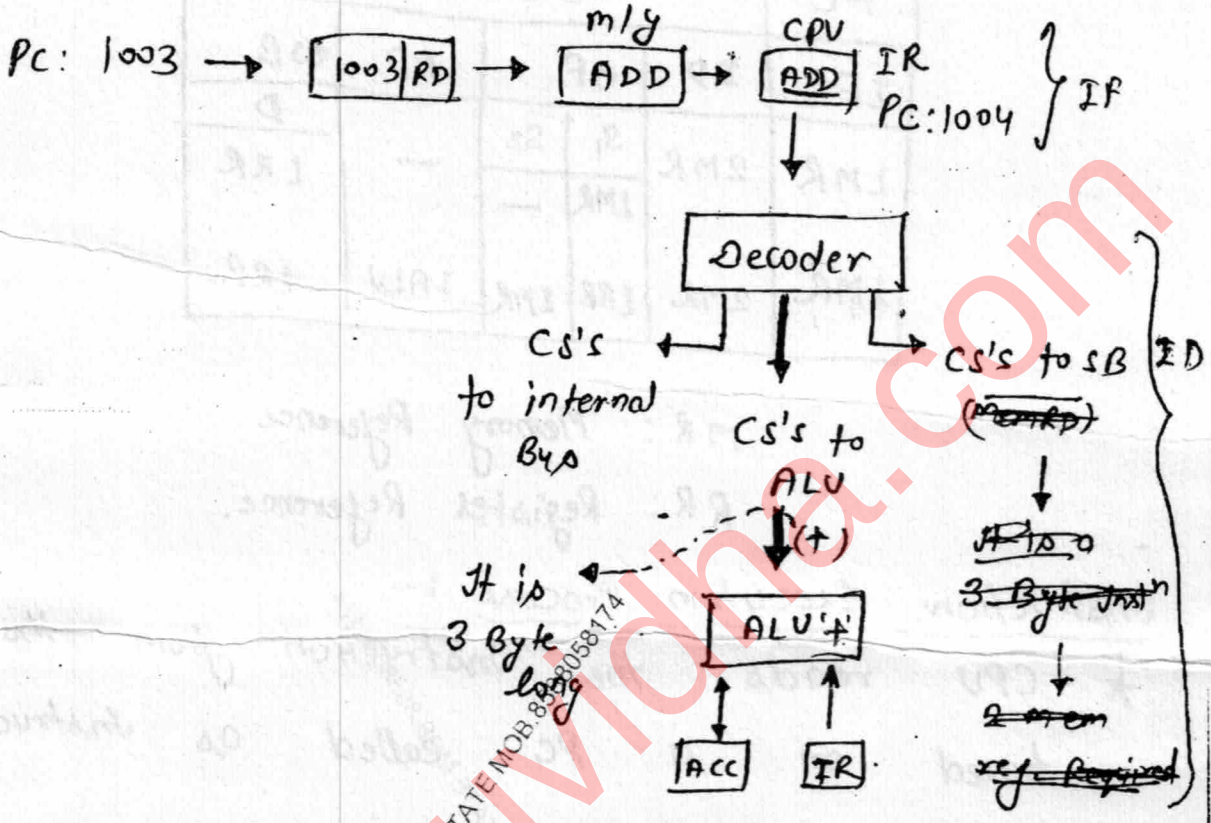


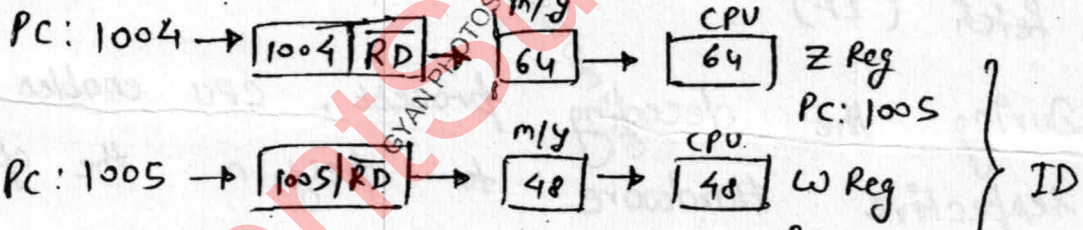
IF : Instruction fetch
ID : Instruction Decode
OF : operand fetch
PD : Process Data
WB : write Back

{I2}:

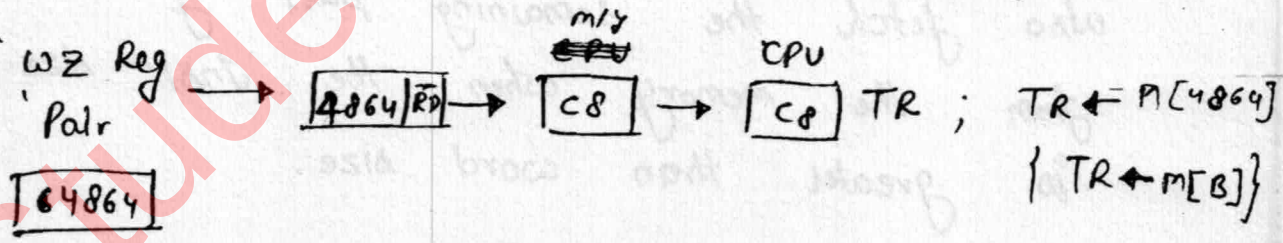


{I2}:

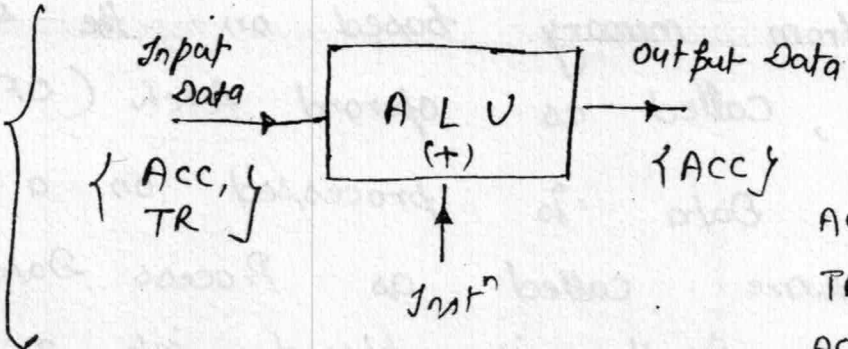




OF { WZ Reg Pair



PD & WB {



ACC : 11111110
TR : 11001000
ACC : (1)11000110

Program Execution :-

FC	EC				
IF	ID	OF		PD	WB
		S_1	S_2		D
LMR	2MR	1MR	—	—	1RR
1MR	2MR	1RR	1MR	1ALU	1RR

MR : Memory Reference
RR : Register Reference.

Instruction Execution Process :-

★ CPU reads the instruction from the memory based on a PC called as Instruction fetch (IF)

★ During the decoding process, CPU enables the respective Hardware to perform the operation. also fetch the remaining part of a instⁿ from the memory when the Instⁿ size is greater than word size.

★ CPU fetch the data either from register or from memory based on the addressing modes, called as operand fetch (OF).

★ Input Data is processed on a enabled Hardware called as Process Data (PD) Later Result is placed into a Destination

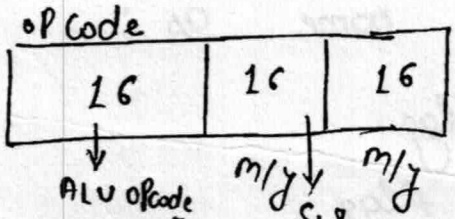
called as write Back (WB).

★ In the CPU design, Extra storage is provided to store the exceptions of a instructions, called as flag Register.

Que:- Consider a 16 bit Hypothetical processor which supports 1 word opcode with, 2 m/y operands. when the opcode is ALU opcode then operand 1 also acts as a Destination. CPU supports 64 kw memory. memory Reference consumes 8 cycles, ALU operation takes 2 cycles. How many cycles are required to complete the Instruction when the opcode is ALU opcode.

Soln:

word size = 16 bit



m/y reg = 8 cycle

ALU = 2 cycles

Instn = 3 w

FC		EC			
IF	ID	OF		PD	WB
~~1MR~~	~~2MR~~	S_1 ~~1MR~~	S_2 ~~1MR~~	—	+1
1MR	2MR	1AR	1MR	1ALU	1MR

~~1MR = 8c~~
~~1ALU = +2~~
=

= 6MR + 1ALU = 50 cycles

PSW (Program Status word) :-

PSW : | ACC & flag Register Pair |

★ Flag is a flip-flop, It is a bi-state component, used to store the 1-bit information

★ Flags are categoriged into two groups :

1) Conditional flags.
2) Control flags.

1) Conditional flags :-

These flags are set or Reset based on the Result nature of a ALU. With Reference to 8086 PSW, 6 conditional flags are present name as :

i) carry flag
ii) Parity flag
iii) Auxiliary carry flag
iv) Zero flag.
v) Sign flag
vi) overflow flag.

① Carry :-

:- Condition : If there is an extra bit out of MSB
- T = Set = 1 = C
- F = Reset = 0 = NC

② Parity :-

:- Condition : If the ALU operation (Acc) contain even number of 1's
- T = Set = 1 = PE ; Even parity
- F = Reset = 0 = PO ; odd Parity

* This flag is used in the Serial data Communication to prepare the error correction & error detection codes

③ Auxilaury carry :-

:- Condition : Is there an extra bit from the Lower nibble [3rd Position] to Higher Nibble [4th position]
- T = set = 1 = AC
- F = Reset = 0 = NA

* This flag is used in the BCD Arithmatic to indicate the Range exceeding condition of a Lower Nibble. It is set when the Lower Nibble is > F

{ Carry flag is also used in the BCD Arithmatic to indicating the Range exceeding condition of a Higher Nibble. It is set when the Higher Nibble is greater than F }

★ BCD format is used to represent the Decimal Data in the Computer. It is two kinds:

i) unpacked BCD
ii) Packed BCD

★ In the unpacked BCD, each Digit is represented with 8 bits.

eg: 23: 00000010 00000011
2 3

In this format 246 codes are unused.

$2^8 \Rightarrow 256 - 10 = 246$

so alternative required that is packed BCD.

★ In a Packed BCD, each Digit is represented with 4 bits.

eg: 23: 0010 0011
2 3

In this format 6 codes are unused {10-15 A-F}

so, Adjustment is required when the ~~Result~~ Result, nibble is greater than 9. i.e. Add 6 to a Result nibble to get the Data into a Decimal format.

eg 1:

$$\begin{array}{r} {\scriptstyle 1\ \ 1} \\ 26 \\ 37 \\ \hline 63 \end{array}$$

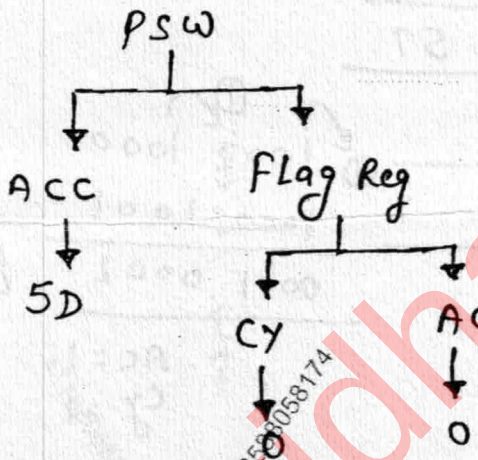
$$\begin{array}{cc} 0010 & 0110 \\ 0011 & 0111 \\ \hline 0101 & 1101 \end{array} \quad [5DH]$$

LN: 13 {13<F} ∴ AC=0

HN: 6 {6<F} ∴ CY=0

AC=0
CY=0

Justify:



{LN: D>9 ∴ Add 6 to LN
{HN: 5<9 ∴ Actual HN is 5 only ∴ No Adjustment to HN

$$\begin{array}{r} {\scriptstyle 1\ \ 1} \\ 5D \\ 6 \\ \hline 63 \end{array}$$

D+6 = 19 ⇒

16	19
1 (Qu)	3 (R)

$(13)_6$

eg 2:

$$\begin{array}{r} 1 \\ 28 \\ 29 \\ \hline 57 \end{array}$$

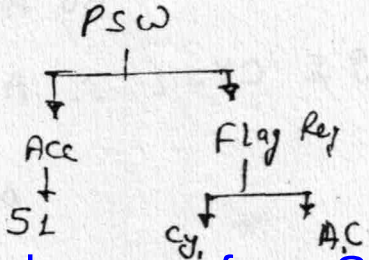
$$\begin{array}{cc} {\scriptstyle (1)} & \\ 0010 & 1000 \\ 0010 & 1001 \\ \hline 0101 & 0001 \end{array} \quad [51H]$$

AC=1
CY=0

LN: 17 {17>F} ∴ AC=1

HN: 5 {5<F} ∴ CY=0

Justify:



{ LN: L<9 & AC=1 ∴ Actual LN > F, so ADD 6 to 'LN'

{ HN: 5<9 & cy=0 ∴ Actual is '5' only so No adjustment needed

51
6
57

Eg 3:

88
89
177

1000 1000
1000 1001
0001 0001 [11H]

① ①

AC = 1
Cy = 1

LN: 17 { 17 > F } ∴ AC = 1

HN: 17 { 17 > F } ∴ CY = 1

Justify:-

PSW

ACC → 11

FLAG Register → CY = 1, AC = 1

{ LN: 1<9 & AC = 1 ∴ Actual LN > F
So Add '6' to LN }

{ HN: 1<9 & CY = 1 ∴ Actual LN > F
so Add '6' to HN }

(cy) 11
66
177

Date 12/08/2017

④ Zero Flag :-

Condition : "Is the ALU (o/p) (Acc) value zero"

→ T = set = 1 = Z
→ F = Reset = 0 = NZ

value	Status
0	1
≠0	0

★ This flag is used in the CPU to implement the various control structures in the base Hardware and also used to control the some device functionality.

⑤ Sign flag :-

Condition : "Is the MSB bit of a ALU O/P (Acc) contain 1"

→ T = set = 1 = NG (-ve logic)
→ F = Reset = 0 = PL (+ve logic)

It is used to preserve the sign of a data in the CPU.

⑥ overflow Flag :-

Condition : "there is a carry into the MSB & no carry out of the MSB (or) vice versa"

→ T = set = 1 = OV
→ F = Reset = 0 = NV

Control Flags :-

Based on the Status of these flags, some of the operations are Controlled in the Hardware. with reference to a 8086 PSW, three Control flags are present :

8086 PSW :

① Trap Flag :- (execution flag)

- 0; At a time Program Exec^{n} (go); -g
- 1; Single step program execution (trace); -t

② Interrupt Flag :-

- 0; Disable the Interrupt (DI)
- 1; Enable the Interrupt (EI)

Only the maskable interrupts are Controlled by the Interrupt flag.

③ Direction Flag :- (Accessing Flag)

- 0; Auto increament (CLD) ---> Clear Direction
- 1; Auto Decreament (STD) ---> Set Direction.

Que :- Consider the following Signed Data and perform the Arithmatic addition operation what is the Status of a carry, zero, and overflow FLAGS in the Computation:

Data : 7 6 5 4 3 2 1 0
1 0 1 1 0 1 1 0
1 1 0 1 1 0 0 1

Acc : → 1 0 0 0 1 1 1 1

CY = 1
OV = 0
Zero = 0

Ac = 0
Parity = 0 (in Acc 5 1's)
Sign = 1

Ans : (1, 0, 0)

Que :- Consider the following 8 bit data computation (-72) + (-83) what is the status of a carry, sign, zero and overflow flags in the computation ?

-72 :- ~~0 1 0 0~~ ~~0 1 1 0~~ 8 4 2 1

-72 :- 0 1 0 0 1 0 0 0

1 0 1 1 0 1 1 1
+1

1 0 1 1 1 0 0 0 ← 2's compliment

-83 :- 0 1 0 1 0 0 1 1

1 0 1 0 1 1 0 0
+1

1 0 1 0 1 1 0 1

-72 : 1 0 1 1 1 0 0 0
-83 : 1 0 1 0 1 1 0 1

Acc : 0 1 1 0 0 1 0 1

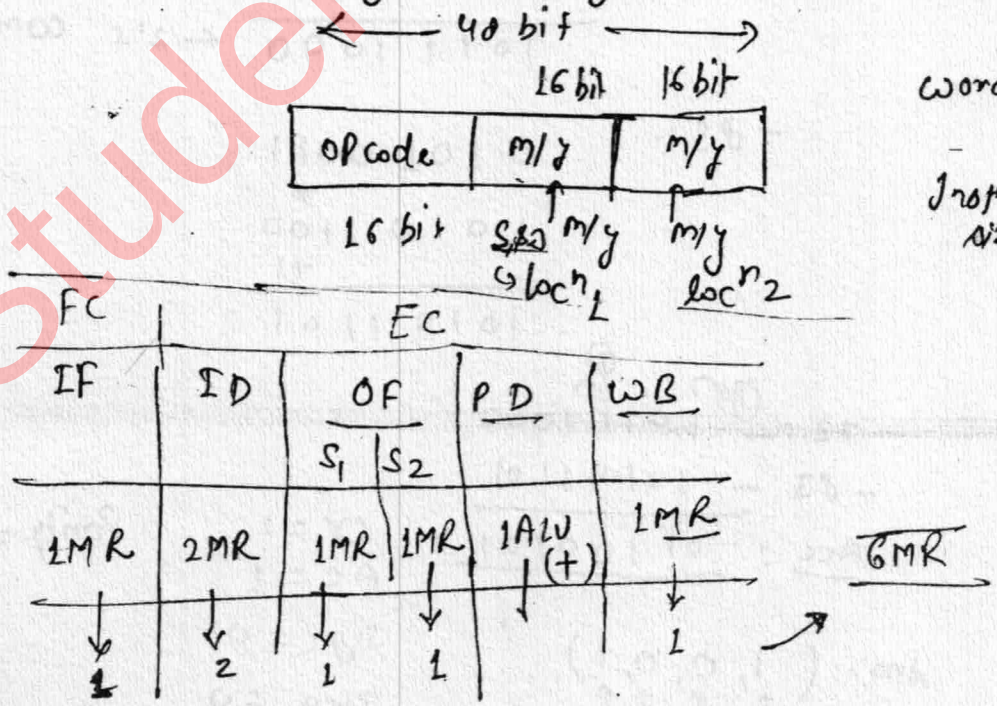
CY = 1
AC = 1
Sign = 0
Zero = 0
OV = 1
Parity = 1

Ans :- (1, 0, 0, 1)
↑ ↑ ↑ ↑
carry sign Zero OV

Que:- Consider a 16 bit Hypothetical processor which supports 3 word instruction with 1 word opcode and 2 m/y operands. 1st m/y operand is $(F2)_H$ and 2nd m/y operand is $(3C4)_H$. During the Execut^n of the instruction Addition op^n is takes place on a above data and report the result into operand 1 location.

a) How many m/y references are required to complete the inst^n when it is stored in the m/y with a address of $(78CA)_H$

b) what is the status of a carry, Zero, overflow flag in the computation.



word size = 16 bit

Instn size = 48 bit = 3w

A - 10
B - 11
C = 12

(b)

$(3C4)_H$ — 0000 0011 1100 0100 — Check this only for AC

$(F2)_H$ — 0000 0000 1111 0011

Acc :- 0000 0100 0011 0111 ← 16 bit Acc

Cy = 0
OV = 0
Zero = 0

Imp: (0, 0, 0)

AC = 0
Parity = 1 (6 1's)
Sign = 0

(2)

"Addressing Modes" :-

"Addressing mode shows the location of a required object in the computer."

★ object may be a instruction or data.

★ output of a addressing mode is Effective Address (E.A.)

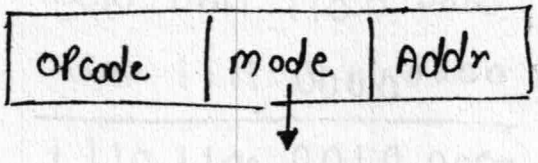
★ E.A. is the actual address of a object.
therefore object = [EA] → Content of

★ In the Instruction design, addressing mode is implemented in two ways :

① implicit [Addressing mode is implemented in the opcode itself]

② Explicit [Mode field is used in the Instn to specify the addressing mode]

eg:



(Explicit A.M. }

Type of addressing mode

$\log_2$ # AM's

★ Different symbols are used to refer the various addressing modes :

① #/I — Immediate Addressing mode

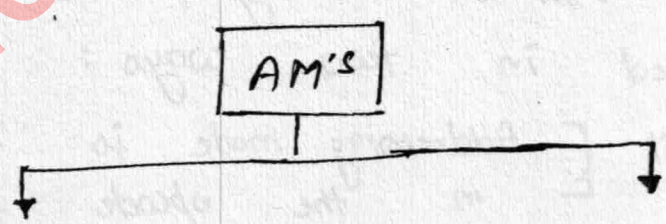
② Reg. Name — Register A.M.

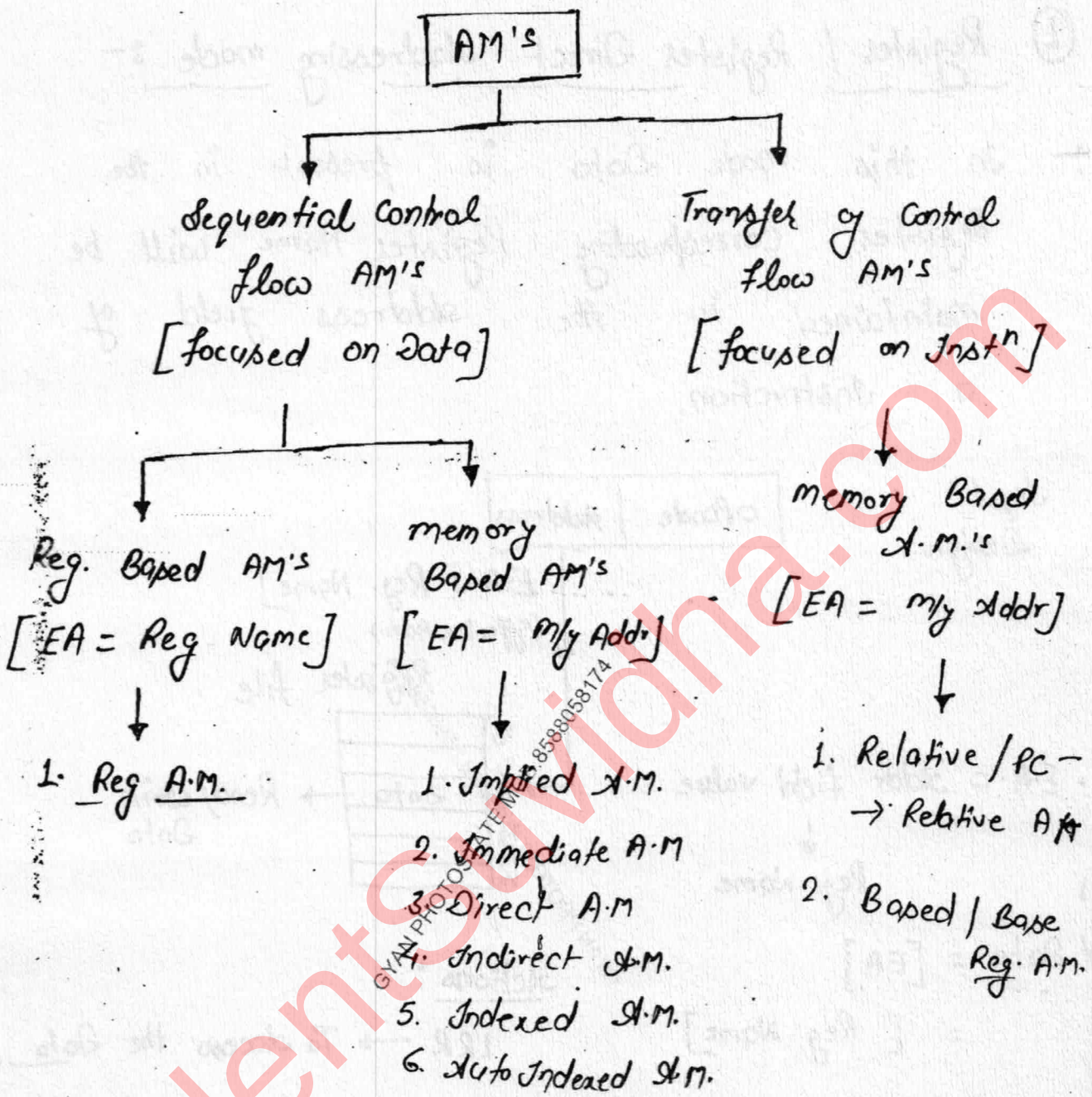
③ [ ] — Direct A.M.

④ ( )/@ — Indirect A.M.

⑤ Index Reg. Name — Indexed A.M.

⑥ +/- — Auto Indexed A.M.





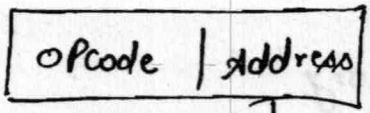
Sequential Control flow :- (Addressing Mode)

These modes are concentrate on the Location of the Data, so Data Transfer and Data manipulation, both of the Instructions are designed with these addressing mode.

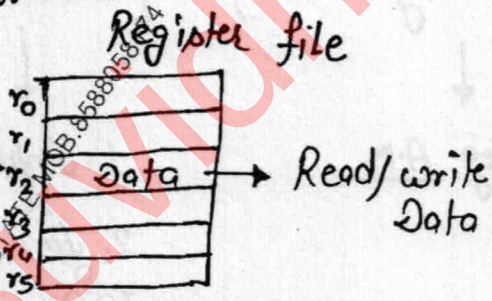
① Register / Register Direct Addressing mode :-

:- In this mode Data is present in the Register, corresponding Register Name will be maintained in the address field of a Instruction.

Instⁿ Design :



E.A. [Reg. Name]
(effective Addr.)



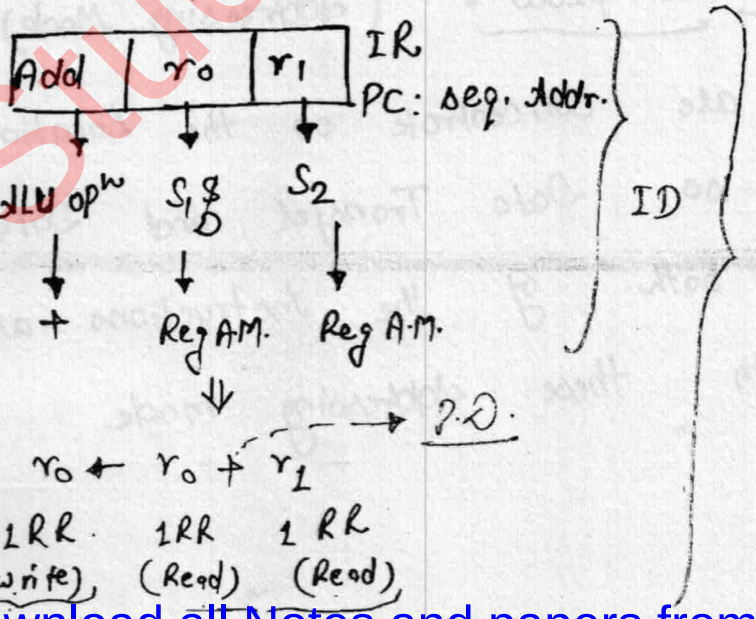
- E.A. = Addr. field value
  ↓
  Reg. Name
- Data = [EA]
  = [Reg. Name]

Actions :

LRR → To access the Data.

eg :-

4 word Instⁿ



IC (Instⁿ cycle) :-

FC	EC				
IF	ID	OF		PD	WB
		S_1	S_2		D
1MR	3MR	1RR	1RR	1ALU	1RR

Que:- If the above instⁿ is 32 byte long executed on a 16 bit CPU. How many m/y references are required to complete when the instⁿ is stored in the Byte addressible m/y.

∴ word size = 16 bit

∴ Instⁿ size = 32 byte = $\frac{256}{16} = 16w$

FC	EC				
IF	ID	OF		PD	WB
		S_1	S_2		D
1MR	15MR	1RR	1RR	1ALU	1RR

= 16MR

fetch cycle = 1MR

Execution cycle = 15MR

Instⁿ cycle = 1MR + 15MR = 16MR

Note ★

Access operand = OF + WB

Fetch operand = only OF