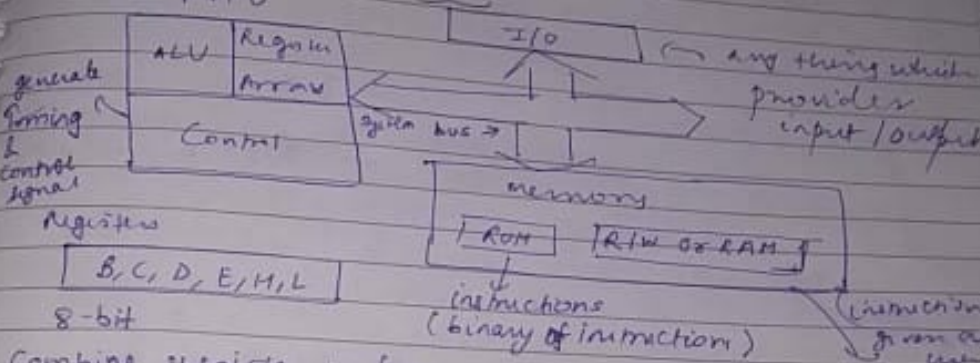
A4 SIZE

Delta

CEC10 · MICROPROCESSORS

(Basic block diagram) [8085]



instruction → binary of instruction → MPU.
from ROM (Only understand binary instruction)

We are using one peripheral
↳ MPU is interacting with one peripheral at a time. Inp from peripherals are stored sequentially to memory.
It can ~~use~~ be used as input or output not both.
control unit specifies whether inp or out etc

Inp → Keyboard / keypad

o/p → Seven segment display.

o/p also goes to memory → If further operation is to be done

Delta

* It is programmable.
* It is clock-driven as it needs to be synchronized
* It is register based → push data on in registers.

* Memory  → 8 bit register

2^{10} → 1024 → 1 kByte locations

* Application

(1) Reprogrammable → Data & operations programmed on it.

(2) Embedded System → part of bigger system (Washing Machine etc) user cannot access it as it is part of bigger System.

* Registers can be used in pairs for 16 bit

BC } 16 bit regis
DE
HL }

* Control Unit :

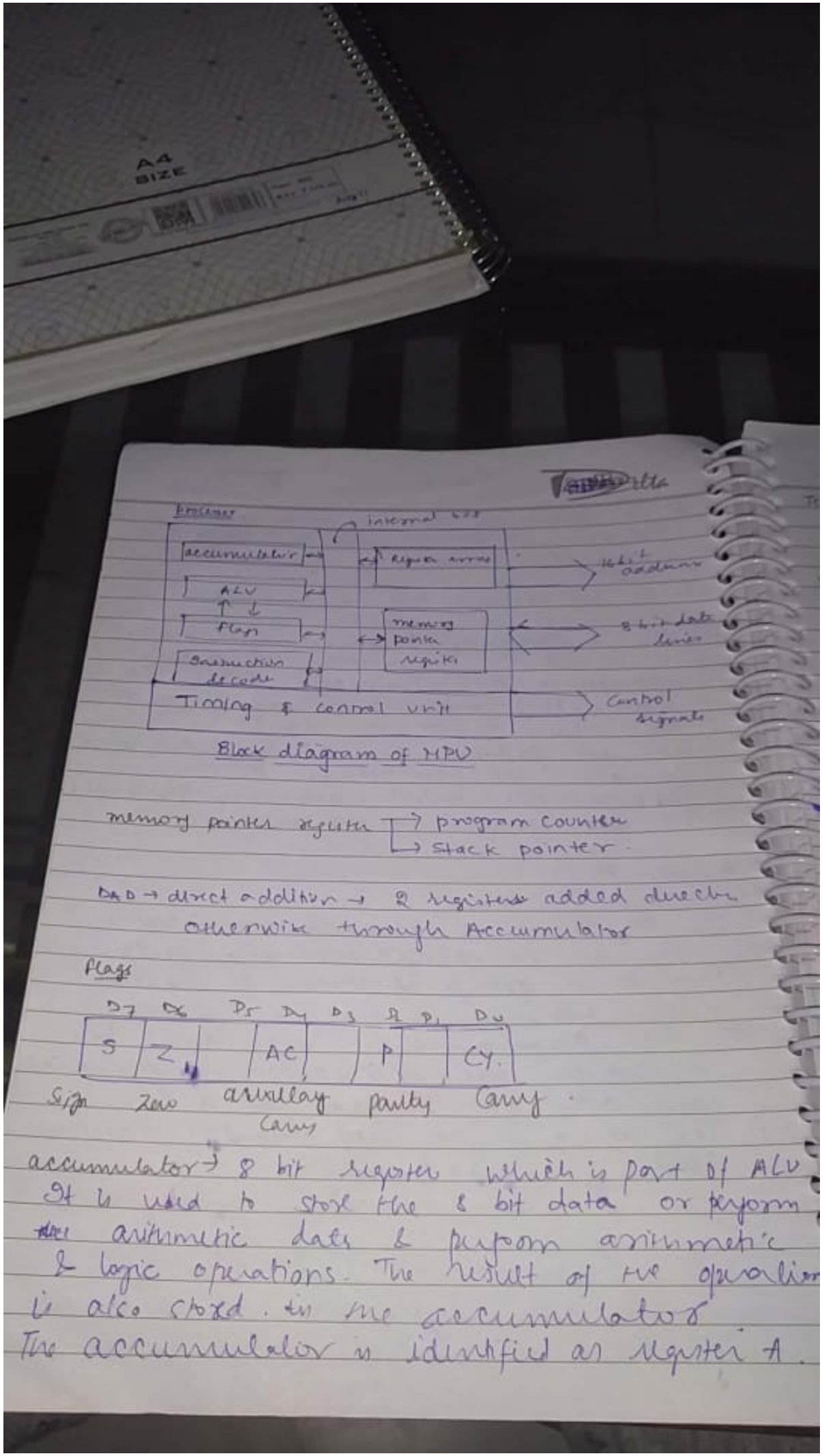
(1) It provides the necessary timing & control signals to all operations in the micro-computer

② It controls the flow of data between the microprocessor, the memory and the peripherals.

③ The bit patterns required to initiate the microprogram instructions are given to the programmer in the form of an instruction set. Programmer selects the appropriate bit pattern and enters it sequentially in the memory through an input device. The CPU/MPU reads these bit patterns one at a time and initiates the appropriate microgram through the control unit and performs the task specified in the instruction.

* System Bus:

① It, can is a communication path b/w the microprocessor and peripherals.
② It is a group of wires to carry the bits.
③ All peripherals share the same bus although the microprocessor communicate with only one peripheral at a time.
④ The timing is provided by the control unit.



Block diagram of MPU

memory pointer registers → program counter
→ stack pointer.

DAD → direct addition → 2 registers added directly otherwise through Accumulator

Flags

D7	D6	D5	D4	D3	D2	D1	D0
S	Z		AC		P		CY.

Sign Zero auxiliary carry parity Carry.
Carry

accumulator → 8 bit register which is part of ALU. It is used to store the 8 bit data or perform arithmetic data & perform arithmetic & logic operations. The result of the operation is also stored in the accumulator.
The accumulator is identified as register A.

JC (Jump on carry)
if (CY = 1)
PC = PC + M(xx).

Program Counter (PC) → Points indicating to a particular memory location in stack from where a particular data or instruction is stored.
next instruction to be fetched. (ROM)

Stack Pointer (SP) → Read / write memory top of the memory stack.
(both are 16 bit registers)

PC → Microprocessor uses the PC register to sequence the execution of instructions. It points to the memory address from where the next byte has to be fetched. When the byte is being fetched the program counter is incremented by 1 to point to the next memory location.

SP → It is used as the memory pointer. It points to the memory location in the read / write memory called the stack. The beginning of the stack is defined by loading a 16-bit address in the stack-pointer.

Delta

Instruction sets (8085):

① Data transfer → (copy instruction)

Type:	Example
a) Register to register	Copy content of B to D
b) Specific data byte in memory location	Load register A with data byte 2000H.
c) b/w register & memory location	A [2000H] to B. B ← M[2000H]
d) b/w input device & accumulator	AC ← INP(0-7)

Logical Operations ⇒ AND, OR

① AND, OR, XOR

8 bit data, register or a memory location
↳ ANDed, ORed, or XORed with A (Accumulator)

② Rotate – Content of A are shifted either left / right by 1 bit.

③ Compare – Contents of 8 bit data, register, memory locn is A compared for greater than, less than equal to.

④ Complement - Complements content of A ie 1→0, 0→1

Branching Operation

Lower special service routine + Sub routine
→ transfer to that sub-routine as it has higher priority
When There can be jump based on some condition of on flag.
There can be unconditional jump also

① JUMP: Conditional jumps are important in a decision making process in programming. The instruction tells for certain conditions ie the zero, carry flag etc and there could also be an unconditional jump.

② Call Return or Restart: return from special routine back to main program.

Three types of Instructions

① 1 byte - opcode & operand stored in same memory location
② 2 byte
③ 3 byte

One Byte - instruction

One byte instructions include the operand and the opcode in the same byte.

1-byte

MOV	C, A	Hex code 4FH
(MOV Rd, Rs)		
ADD	B	80H.
CMA		2FH.

Two-Byte instruction

		Hex-code	
MVI	C, F2H.	3EF2H	C ← F2H.
MVI	B, 32H.		

(move immediate)

(2 memory locations to store instruction
one - opcode
two → data/address)

These instructions require two memory locations for the opcode and the operand to store the binary code

Three-byte instruction.

Delta

opcode → 1 byte
others 2 → 16 bit address

2nd Byte → Lower Order Address
3rd Byte → higher order Address

eg LDA 2050H — Hex code
3A
50 H — 2nd byte
20H — 3rd byte

JMP 2080H

NOTE
=> The internal architechture of microprocessor decides the operations to be performed
There are:
1. store 8 bit data
2. perform arithmetic / logical operations
3. test for conditions (flag = 0 / etc)
4. sequence the execution of instructions.
5. Store the data temporarily during execution in the read-write memory location called the Stack.

# Memory	Hex Code		
PC → 2000	06	MVI B, 78H	B ← 78H A ← F2H A ← A+B
2001	78		
2002	3E	MVI A, F2H	
2003	F2		
2004	80	ADD B	78H + F2H → 16AH
2005	76	HLT	

Carry

Delta

-> Peripheral or Externally Initiated operations

① RESET : PC back to 0000H

When activated externally, all internal operations are suspended and the program Counter is cleared to 0000H so the program execution can begin from the zero memory address

② INTERRUPT : When interrupts are made then PC goes to special service routine

③ READY: (active low) If low than the microprocessor enters into a ~~weight~~ wait state, and this is used to synchronize the microprocessor (μP) with the slower peripherals.

④ HOLD: When activated, the microprocessor relinquishes (gives up) the control of the buses and allows and external peripheral to use them.
DMA (Direct memory Access) data transfer.

Delta

Instruction set

R - 8 bit register
R_d → destination Register
R_s → source Register
M → memory location
R_p → Register pair

BC } Register
DE } pairs
HL }

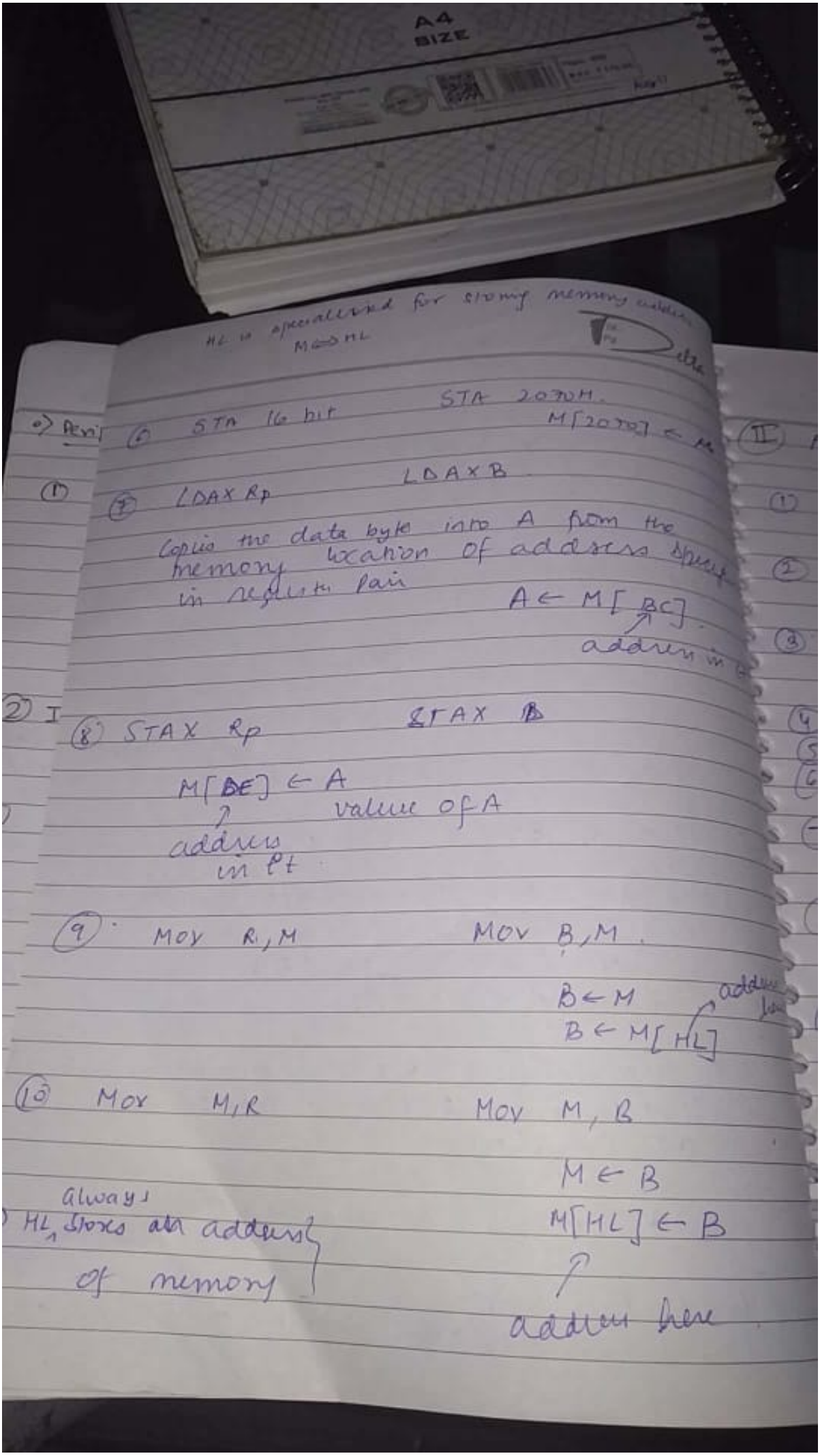
I Data Transfer Instructions.

	Mnemonic	Eg
(1)	MVI A, 8 bit	MVI B, 32 H
(2)	MOV Rd, Rs	MOV B, A
(3)	LXI R_p, 16 Bit (load immediate). (load 16 bit number into Register pair)	LXI B, 2050H. B ← 20H C ← 50H
(4)	IN 8 bit (from external input).	IN 07 H.

It accepts the data bit from an input device and places in it accumulator.

(5)	LDA 16 bit	LDA 2050H. A ← M[2050H].

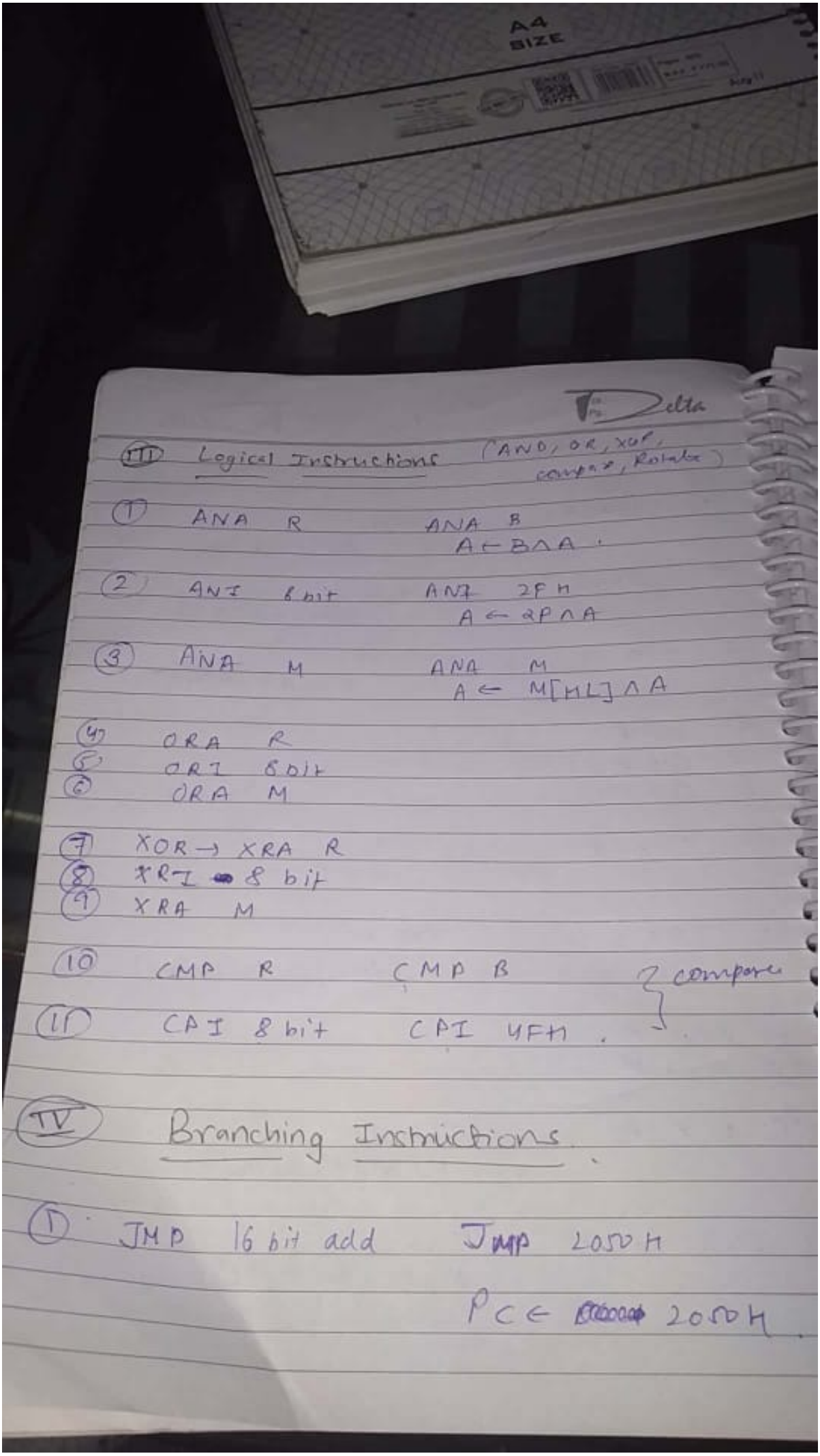
(load data from memory location to accumulator



Delta

II. Arithmetic Instructions

1. ADD R — ADD B — A ← A + B
2. ADD B bit — ADD 32 M — A ← A + 32 M
3. ADD M — ADD M — A ← A + M[HL]
4. SUB R.
5. SUB 8 bit
6. SUB M.
7. INR R. — INR B — B ← B + 1
8. INR M — INR M — M[HL] ← M[HL] + 1
9. DCR R.
10. DCR M
11. DCX Rp — DCX B. — M[BC] ← M[BC] - 1
12. INX Rp — INX B. — M[BC] ← M[BC] + 1.



{Appendix → instruction set}

(2) JZ 16 bit add — JZ 2080 H; if (Z==1) PC ← 2080 H

(3) JNZ 16 bit add — JNZ 2032 H; if (Z==0) PC ← 2032 H

(4) JC 16 bit — JC 2075 H; If (CC==1) PC ← 2075 H

(5) JNC 16 bit — JNC 2082 H; If (CC==0) PC ← 2082 H

(6) CALL 16 bit add. — CALL 2075 H. (current PC is stored) before jumping

(7) RET (going back) — RET.

(V) Machine Control Instruction

HLT — Halt (stop)

NOP — No operation (Suspended state no operation is performed).

Delta

Write the instructions to subtract two bytes already stored in memory register 2051H and 2052H. The location 2051H holds the byte 49H & location 2052H holds the byte 9FH. Subtract the first byte 49H from 2nd byte & store the result in 2053H. Write the instruction beginning at 2030H.

AM

2030 LDA 2052H
2033 SUB 2051H
2036 STA 2053H

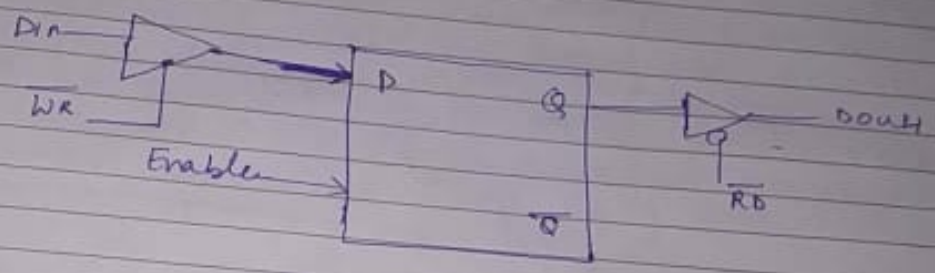
A ← 9FH
A ← A − M[2051H]

2030H LDA 2051H.
2033H MOV B, A.
2034H LDA 2052H
2037H SUB B
2038 STA 2053H.
2041 HLT.

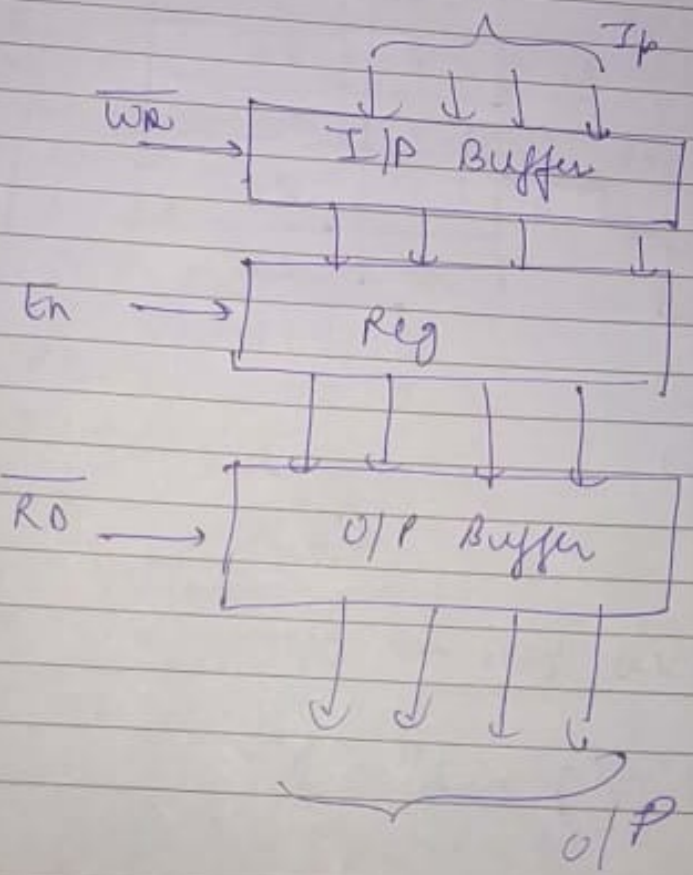
Memory → Two types

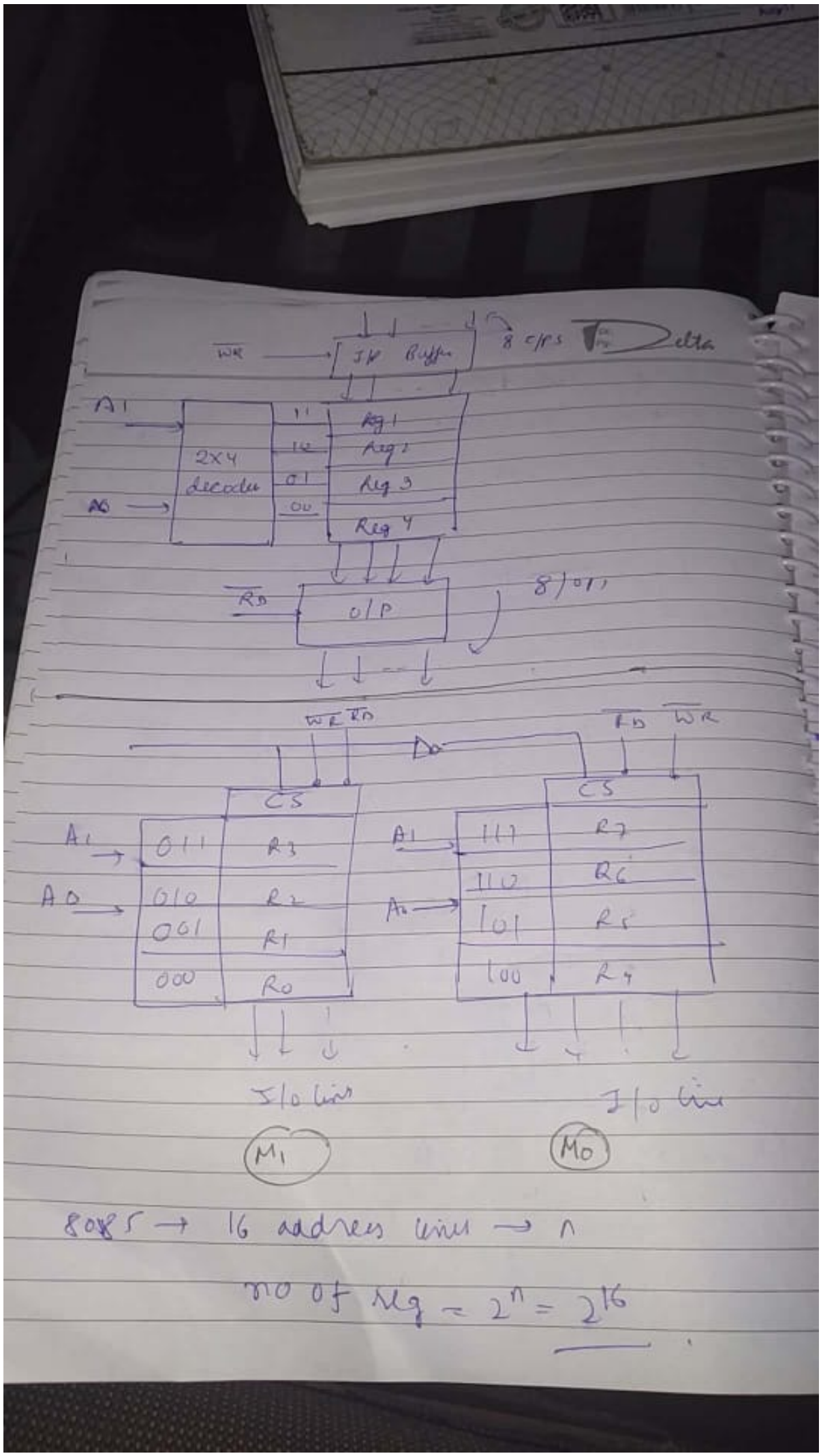
① ROM → Diodes

② RAM → Flipflops / FETs



Memory cell of RAM.





8085 → 16 address line → n

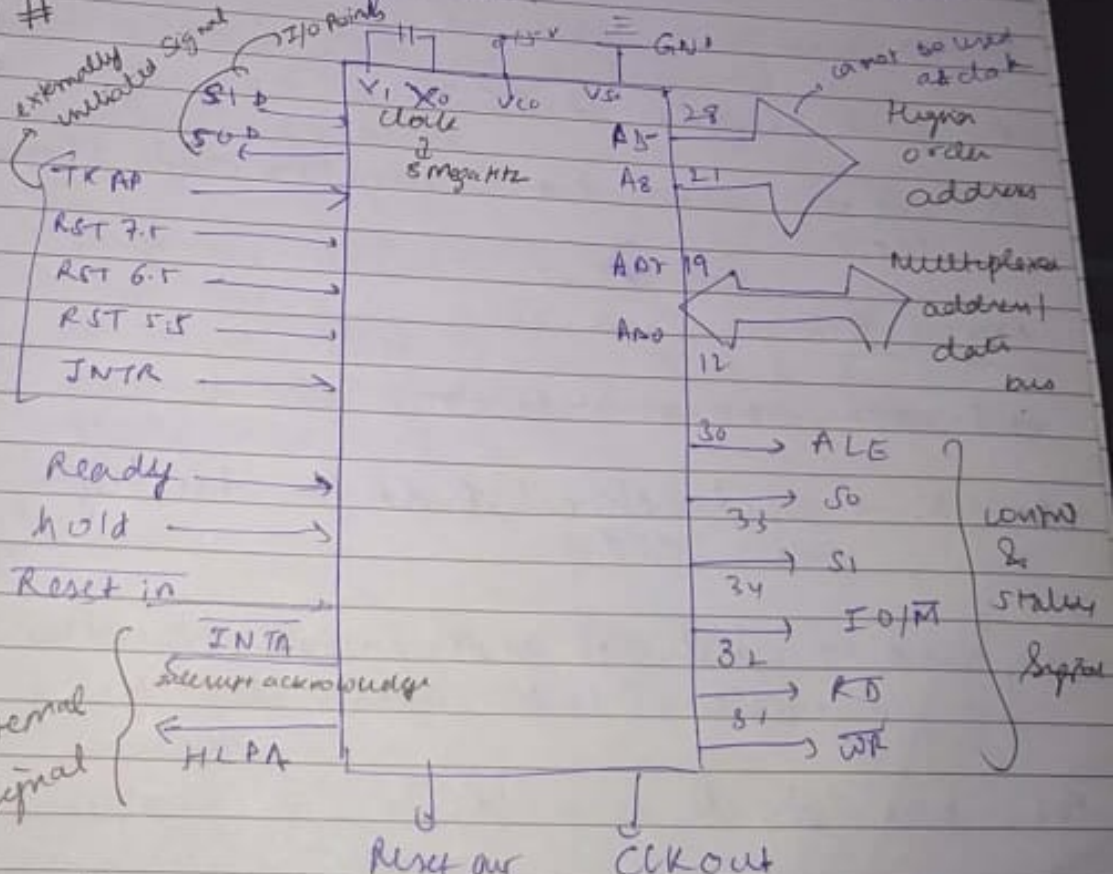
no of Mg = $2^n = 2^{16}$

-> A memory chip requires address lines to identify a memory register. 8085 has 16 address lines. The no. of registers that can be addressed is 2^{16}.

-> A memory chip requires CS to enable the chip and the address lines of the memory chip selects the register. Thus the memory address of register is determined by the logic level of address lines.

-> The control signals RD and WR enable the O/P / I/P buffers so that the data can be read from or written into the memory cell.

#



{8085 pin diagram}

A1 1) All signals used in 8085 can be classified into
a) address bus
b) Data Bus
c) control & status signals
d) power supply & frequency signal
A2 e) externally initiated signal
f) serial I/O ports

2) 8 bit microprocessor

3) capable of addressing 64 K Byte of memory

4) 40 pins

5) power supply is 5 V

6) operates with the 3 mega Hz single phase clock.

A1

A2 ADDRESS BUS

~~Synchronous~~ ~~A15 to A8~~ →

A15 to A8 → unidirectional, used for the high order address

AD7 to AD0 → bidirectional, used for the lower order address

for data also it is used.

Delta

Control signals - $\overline{RD}$, $\overline{WR}$

status signal - S.S., IO/$\overline{M}$

special signal - ALF

ALF - address latch enable

=> This is the positive going pulse generated everytime the 8085 begins the operation (machine cycle).

=> It indicates that AD7 to AD0 are the address bits.

=> This is used to latch the lower order address from the multiplexed bus and generate a seperate set of 8 address lines A7 to A0.

$\overline{RD}$ -

=> It is an active low signal

=> This indicates that the selected memory loc or the I/O device is to be read from & the data is available on the data bus.

$\overline{WR}$ -

=> active low

=> indicates that data has to be written into some memory / IO location

Opposite of $\overline{RD}$

Delta

IO/$\overline{M}$
- Status signal used to differentiate b/w I/O & Memory operation.
  High → I/O
  Low → Memory.
- It can be combined with RD and WR

Clock X_1, X_0, CLK OUT
- This is the crystal or the RC-LC network
- The frequency is internally divide by 2 hence to operate the system at 3 MegaHz the crystal should have frequency of 6 MegaHz
- Clock out is the signal that can be used as the system clock for some external device.

INTR
- Interrupt request
- general purpose interrupt (input signal).

$\overline{INTA}$
- Interrupt acknowledge.
- used to acknowlege an interrupt - active low, O/P signal

RST 7.5, RST 6.5, RST 5.5 (Restart interrupt)
- vectored interrupts that transfer the program counter to the specific memory location
- higher priority than INTR.
- $X_1 > > 6.5 > > 5.5$

Delta

TRAP
-> Input, non maskable interrupt & has a highest priority.

HOLD
-> Input signal
-> Signal indicates that the peripheral such as a DMA controller is requesting the use of address and the data buses

HOLDA (Hold acknowledge)
-> acknowledge O/P signal.
-> acknowledges the had request

Ready
-> I/P pin
-> used to delay the microprocessor read or write operation. tells a slower peripheral is ready to send or accept the data.
-> When the signal goes low the microprocessor waits for an integrated no of clock cycles until it goes high again.

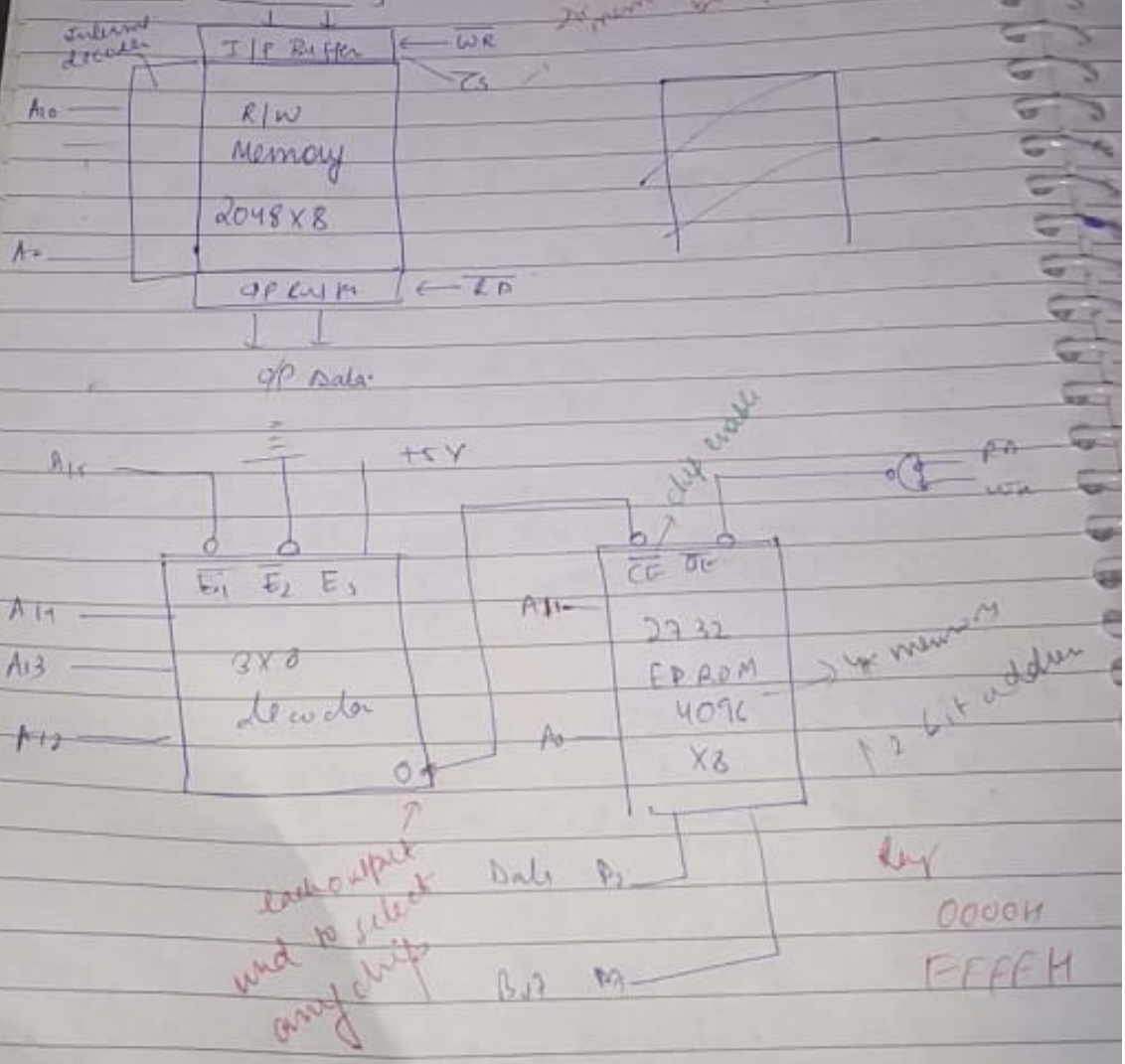
Reset out
-> Signals indicates that the microprocessor is being reset & the signal can be used to reset other devices.

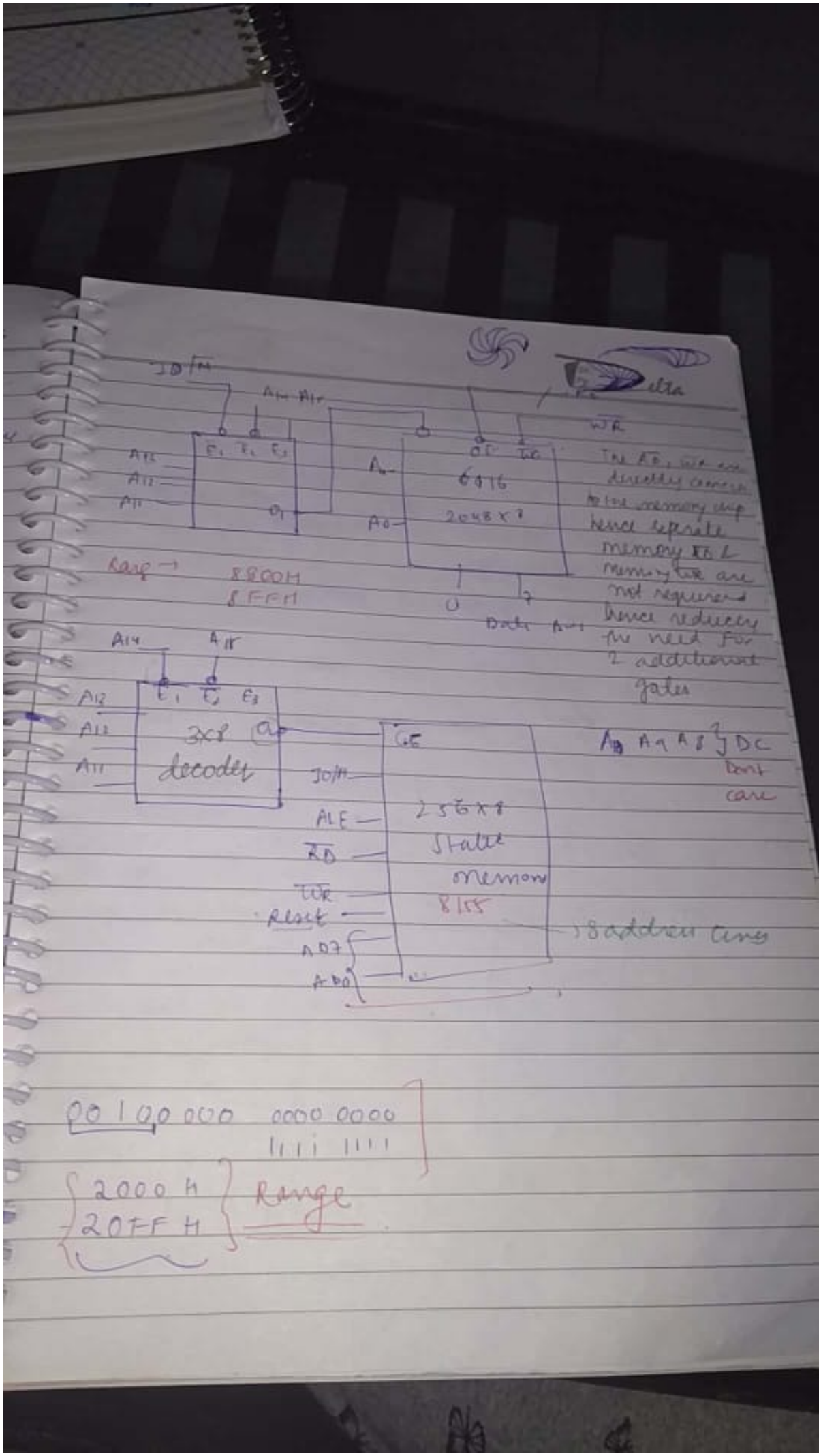
Reset in
-> When this goes low, PC is set to 0, the buses are tristated, the microprocessor resets

SID → Serial I/P data
SOD → Serial O/P data

The data bits are sent over the single line one bit at a time.

Address Decoding





Delta

Some case address lines are can be assumed to have any d/c combination 000 to 111 Thus each combination can generate 1 set of complete address. The address given assuming the don't care all lines is by convention this is specified as the primary address. The remaining address range is known as foldback memory or mirror memory

Ans

mirror memory 2100H → 27FFH

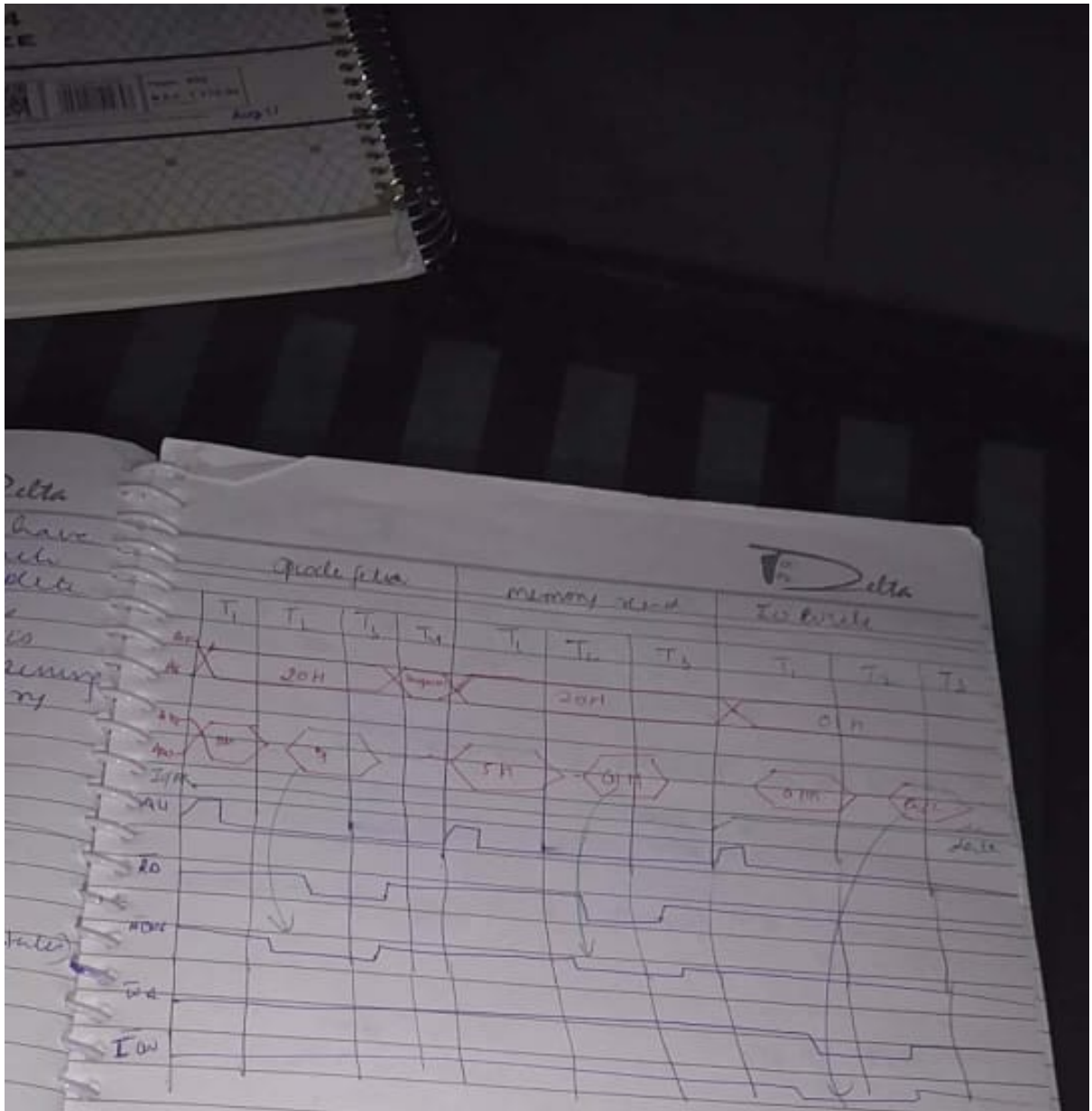
Ans- OUT 8 bit port address OUT 01H

opcode
① OUT D3 → fetch and decode (4T time states)

② Fetch 01H from the memory (R). (3T)

③ The data from accumulator (A_{15}) → 01H. (3T) (IO-write)

T_1	T_2	T_3	T_4



WAP & verify the O/P

1) Load the number 8BH into register D
   6FH in C
2) Increments the content of register C by 1.
3) Add the contents of C & D
4) & display the sum at O/P port 1

Ans

00	MVI D, 8BH	06 ADD D
01		07 OUT PORT 1
02	MVI C 6FH	08
03		09 HLT.
04	INR C	
05	MOV A, C	Flag → CY=0, S=1, Z=0

~~Ignore for~~ Unsigned nos

Delta

Since addition of signed nos is not taking place the S=1 flag can be ignored. It is 1 only because the bit D7 has been assigned to represent the signed flag It needs to be ignored

Q WAP to do the following to load the A.

B → 30H
C → 39H

Subtract 39H from 30H & display the contents of Port one.

```
00  MVI  B,30H
01
02  MVI  C,39H
03  MOV  A,B
04  SUB  C
05  OUT  PORT1
06
07  HLT
```

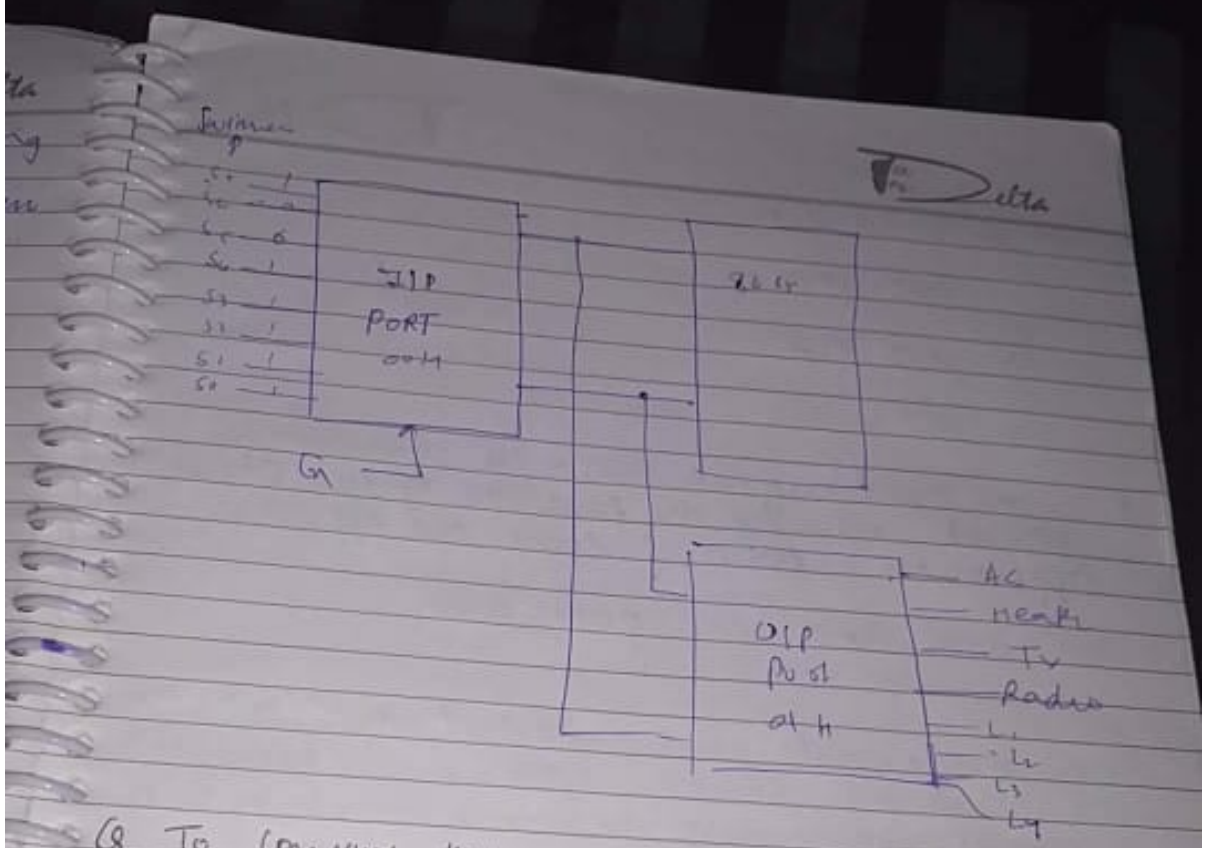
0011	0001	31H
1100	0110	→ 1's comp
	1	
1100	0111	2's comp
0011	0000	→ 30H
1111	0111	

0 CY

D7 → Sign bit

number but not signed ~~bit~~

So we will ignore sign bit.



Q To conserve the energy in the house

1) Turn on the airconditioning at the switch S7 on the input port. Ignore
2) Ignore all the other switches on the I/P port even if someone attempts to turn the on by turn on any other appliance.

mask it → any number and it with 80H.

1001 1111
& 1000 0000
1000 0000

We will get S7.

XX00 MVI A, data
01
02 ANI 80H
03
04 OUT 01H
05
06 HLT

Q In the figure keep the radio on continuously without affecting the function of the other appliances even if some one off it.

OR with 10H : 0001 0000

XX00	MVI A, data	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
01		0 0 0 1 0 0 0 0
02	ORI 10H	D_7 D_6 D_5 1 D_3 D_2 D_1 D_0
03		
04	OUT 04H	
05	HLT	
06	HLT	

Q Conditional Jump

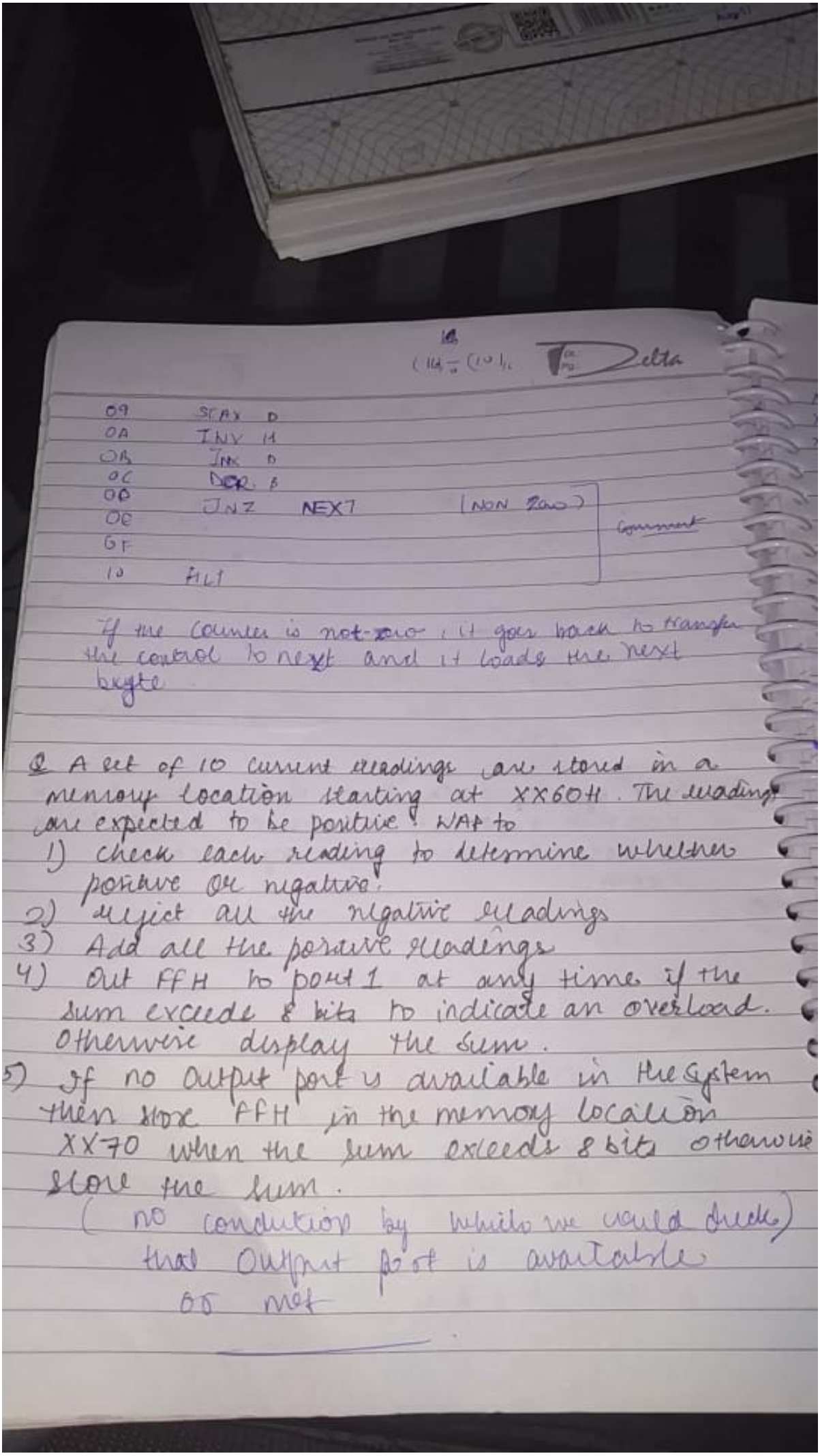
Load the registers A & C with 9B and A7H add the numbers. If sum is greater than FFH display 01 at the output port else display the sum

```
00      MVI D, 78H
01
02      MVI E, 87H
03
04      MOV A, D
05      ADD E
06      JNC DISPLAY     (Jump no carry).
07
08
09      MVI A, 01H
0A
0B      OUT 01H DISPLAY
0C
0D      HLT
```

Q Some 16 bytes of data stored in memory locations XX50H to XX5FH, transfer the entire block of data to a new memory location XX70H.

Ans

```
        Label
00      LXI H,  XX50H
01
02
03      LXI D,  XX70H
04
05
06      MVI B, 10H      → Setting Counter
07
08
09 NEXT MOV A, M
```



```
09      SUB D
0A      INR H
0B      INR D
0C      DCR B
0D      JNZ NEXT        [NON ZERO]
0E                      Comment
0F
10      HLT
```

If the counter is not zero, it goes back to transfer the control to next and it loads the next byte.

Q A set of 10 current readings are stored in a memory location starting at XX60H. The readings are expected to be positive. WAP to

1) check each reading to determine whether positive or negative.
2) reject all the negative readings
3) Add all the positive readings
4) Out FFH to port 1 at any time if the sum excede 8 bits to indicate an overload. Otherwise display the sum.
5) If no output port is available in the system then store FFH in the memory location XX70 when the sum exceeds 8 bits otherwise store the sum.

(no condition by which we could decide) that Output port is available or not

Rotate without carry → RRC, RLC

Take

XX00		LXI H, XX60H	
XX01			
XX02			
XX03		MVI B, 00H	// Register for time delay
XX04			
05		MVI C, 0AH	// Register to counter
06			
07	NEXT	MOV A, M	
08			
09		RAL	// Rotate left to check carry
0A		JC REJECT	// Jump on carry with carry
0B			
0C		RAR	// Rotate right with carry
0D		ADD B	
0E		JC OVERLOAD > FFH	
0F			
10			
11		MOV B, A	
12	REJECT	INX H	
13		DCR C	
14		JNZ Next	// Jump if not Zero.
15			
16			
17		MOV A, B	
18		OUT PORT 1	
19			
1A		HLT	
1B		NOP	// No operation for time delay
1C	OVALUE	MVI A, FFH	
1D			
1E		Output 1.	
1F			
20		HLT	

Q A set of readings is stored in the memory location at XX50H. The end of the data string is indicated by the byte 00H. Find the set of readings the answer may be larger than FFH. Display the sum at PORT1 and PORT2.

```
XX00H  LXI H XX50        19  MOV A,B
XX 01                    1A  OUT Port 2
   02                    1B
   03  MVI C,00H //Sum   1C  HLT
   04
   05 Next MOV B,C → Save carry
   06      MOV A,M
   07      CPI 00H       → if 00 → then carry = 1
   08
   09      JC Display
   0A
   0B
   0C      ADC
   0D      JNC Save      (Carry=0)
   0E
   0F
   10      INR B         (Carry ⇒ Carry = 1)
   11 Save MOV C,A
   12      INX H
   13      JMP Next
   14
   15
   16 Display MOV A,C
   17      Output port 1
   18
```

Assignment

Delta

Q A set of 5 readings are stored in memory location starting at XX50H. Sort the readings in the ascending order & wait for 1s. Also display the output.

50→07, 51→25, 52→42

Q The following block of data is stored in a memory location XX55H to XX5AH. Transfer the data to the memory location XX80 to XX85 in the reverse order

Data: 22, AF, B2, 99, 7E, 37

Counters and Delays

Label		T States	
	MVI C, FF H	7T	(4+3)
LOOP	DCR C	4T	(4)
	JNZ LOOP	10T/7T	(4+3+3)

↳ if condition is false so only 7 T state used. will not use last 3 T states to go back to loop.

Clock frequency = 2 MHz

Time Period = $\frac{1}{2}$ = 0.5 μs

f system clock

4 → load opcode
3 → load address
3 → go back to address

not used if condition is false

Delta

T_o → outside the loop
T_L → for loop delay

[Total time delay = $T_o + T_L$]

FFH = $(255)_{10}$

[TL] Total loop time = Tclock x no of T-states x N_{10}

N_{10} → decimal representation of time for which loop was executed.
In terms of → $(FFH)_{16}$ → $(ST)_{16}$

1) MVI C, FFH

$T_o = T_c \times 7T = 0.5\mu s \times 7 = 3.5\mu s$

2) $T_L = 0.5 \times 255 \times 14T$
$T_L = 1785\ \mu s - (8T \times T_c)$ → for last case at there are 3 less T states in the last condition

$= 1785 - 1.5$
$= 1783.5\ \mu s$

↑ ignore 1.5μs as if time is converted to microsecond 1.5μs will be negligible

Delta

10/7 → 10 T states we are required to execute a conditional jump when it jumps to change the sequence

7 T states are required when it goes to the instruction following jump on non zero i.e the last cycle will have 7 T states and loop delay would be calculated by subtracting the time required for three T states

$$T_{Total} = T_0 + T_1$$
$$= 783.5 + 3.5$$
$$= 787\,\mu s = 1.5\,ms$$

Time Delay using a register Pair

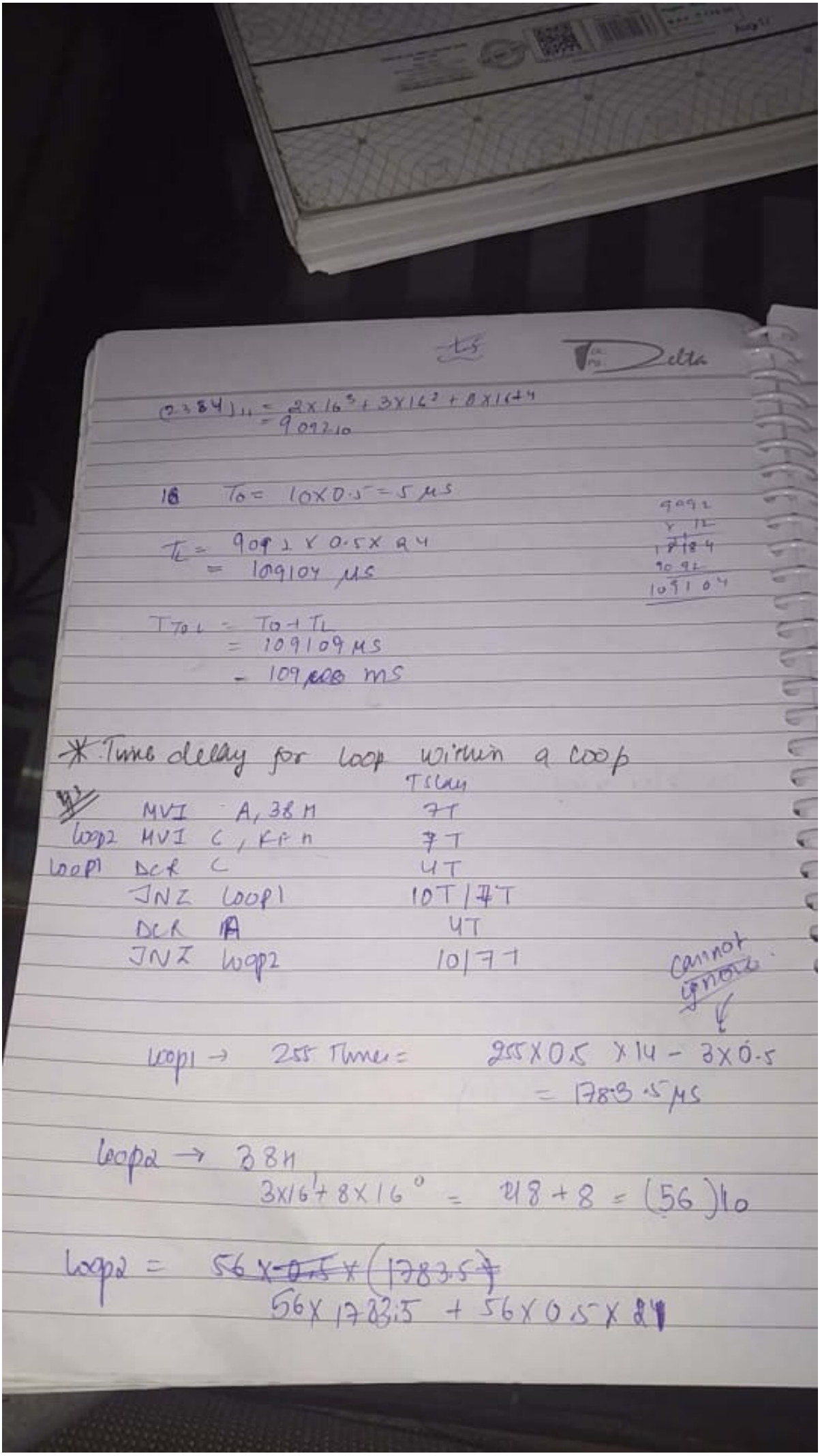
Eg 2

			T state
	LXI	B, 2384 H	10T
Loop			
	DCX	B	6T
	MOV	A, C	4T
	ORA	B	4T
	JNZ	Loop.	10/7T.

B → 23 22
C → 84 A ← 84

00/0M0
1008 100

Clock Period T = 0



Loop = 56 [1783.5 + 10]

= 56 [1794]

= 100464 μs

= 100.46 ms

$T_0 = 7 \times 0.5 = 3.5 \mu s$.

$T_{Total} = 100464 \mu s + 3.5 \mu s \rightarrow 100.46 ms$

Draw Backs.

1) The accuracy of time delay depends upon accuracy of the system clock.
2) Microprocessor is occupied simply in a waiting loop Otherwise it could be employed to perform other functions.
3) Task of calculating accurate time delays is tedious.

Solution →

To incorporate the Intel 8254 programmable timer chip which can be interfaced with the microprocessor but it adds on to the cost & space required.

↗

p Same programming but outside the microprocessor → μp can perform other functions.

Hexadecimal Counter

WAP to count continuously a hexadecimal count from FFH to 00H in a system where the System clock is 0.5 μs. Use the register C to set up a 1 ms delay between each count and display the number at one of the output ports.

T = 0.5 μs, C → 1ms delay

```
00      MVI B, 00H      (counter)
01
02  Next  DCR B         (FF H)
03        MVI C, Count  (Load C with delay count)
04
05  Delay DCR C       } loop  4T
06        JNZ Delay   }       10T
07
08
09        MOV A, B
0A        OUT PORT 1
0B
0C        JNZ Next  (JMP Next)
0D                  ↑
0E                infinite.
```

14 × 0.5 × Count = 1000

7 × Count = 1000

$Count = \frac{1000}{7} = (143)_{10}$

16	143	F
16	8 8	

$= (8F)_{16}$

Delta

$T_{loop} = 14T \times 0.5 \times 10^{-6} \times Count$

Taking:

DCR B	4T
MVI C, Count	7T
MOV A, B	4T
ORI FOR I	10T
JNZ Next	10T
	35 T states

$T_D = (35T \times 0.5 \times 10^{-6})$

$T_D (Total) = T_D + T_{loop}$

$1\,ms = T_D + T_{loop}$

$1000 = 35 \times 0.5 + 7 \times Count$

$1000 = 17.5 + 7 \times Count$

$Count = \frac{1000 - 17.5}{7}$

$Count = \frac{982.5}{7}$

$Count = 140.35 \approx (140)_{10} = (8CH)_{16}$

16	140	C
	8	

$Count = 8CH$

Delta

WAP to count from 0 to 9 with a 1 second delay between each count At the count of 9 counter to set itself to zero & repeat the sequence continuosly. Use HL pair to for delay & display each count at one of the output port. Assume the clock frequency of microprocessor is 1 Mega Hz
Tc = 1μs

```
00 (Start). MVI B, 00H
01
02 Display  OUT Port 1
03
04    LXI H, 16 bit          (Delay)
05           find value
06
07 Loop  DCX H          → Does not reflect on zero flag.
08       MOV A, L       Zero flag can not be read.
09       ORA H
0A       JNZ Loop
0B
0C
0D       INR B
0E       MOV A, B       Carry
0F       CPI 0AH        If 0A non zero → not equal to 0AH
10
11       JNZ Display    Carry Zero → equal to 0AH.
12       JNZ
13
14       JZ  start      When equal zero carry flag is set
15       JZ
16.
```

Is delay → can not be accomdated in 8 bit so need 16 bit

Delta

$T_L = (6+4+4+10) * T_C \times$ (Count)
$= 24 \times 0 \rightarrow 1 \times$ Count
$= 24 \times$ Count

$T_D = 10 + 10 + 4 + 4 + 7 + 10 + 10$
$= 55$

$T_D = T_L + T_H$
$1000000 = 24 \times$ Count $+ 55$

approx Count $= \frac{1000000}{24}$
$= (41666)_{10}$

Total $= (41665)_{10}$
≈ 41666
Can be ignored

16	41666	2
16	2604	4 12 (C)
16	162	2
	10 (A)	

$(41666)_{10} \rightarrow (A2C2)_{10}$
↑
Count

WAP to generate a square wave (continuous) with period 500 ns. Assume that clock is 325 ns & use the bit D0 to o/p the square wave.

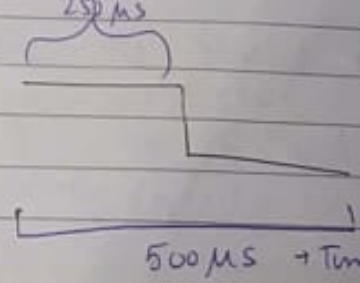
(Conditions where T_D can not be ground)

```
01      MVI D, AAH
02
03 Relay  MOV A,D
04        RLC                  // Rotate, left Circular
05        MOV D,A
06        ANI 01H
07
08        OUT PORT1
09
0A        MVI B, Count  // Delay
0B
0C  Delay DCR B          4T  } Loop
0D        JNZ Delay      10T/7T
0E
0F
10        JMP Restart
```

AAH	10101010	
RLC	01010101	// without carry.
ANI	00000001	
	00000001	

as D0 bit show the O/P

So to get only D0 we and with 01H.



1st → D0 = 1

2nd → D0 = 0.

Delta

500 µs is total time period so → 250µs each for high & low

So we need a delay of 250µs b/w each iteration

$T_L = 325 \times (4+10) \times Count = 325 \times 14 \times Count$

$T_L = 325 \times 14T \times (Count-1) + 325 \times 11T$ } Last loop 7

$T_D = 46T \times 325$ Clock → 325 ns

$T_D = T_D + T_L = 250\,\mu s$ → Calculate count & convert it into hex.

Stacks and Subroutines

SP → Stack Pointer
PUSH
POP
PSW → Program Status Word (Value of flags & accumulator)
↗
Can be used into Stack.

It works in register pair

PUSH B → Push BC
16 bit

Delta

H → | | Peek
L → | |

Stack Pointer will inc/dec by 2 not 1

LXI SP address
Stack pointer value stack has to be placed

1) Once the stack location is defined storing the data begins at a memory address that is one less than the address in the stack pointer register.

2) PUSH → data bytes in the register pair of the microprocessor can be stored in the stack two at a time in reverse order (Decreasing memory address).

3) POP → data bytes can be transferred from the stack to the register pair by incrementing the memory address.

4) Because two bytes are being stored at a time sixteen bit address in the stack Pointer is decremented by 2.

5) Address in the stack Pointer indicates the next two memory locations.

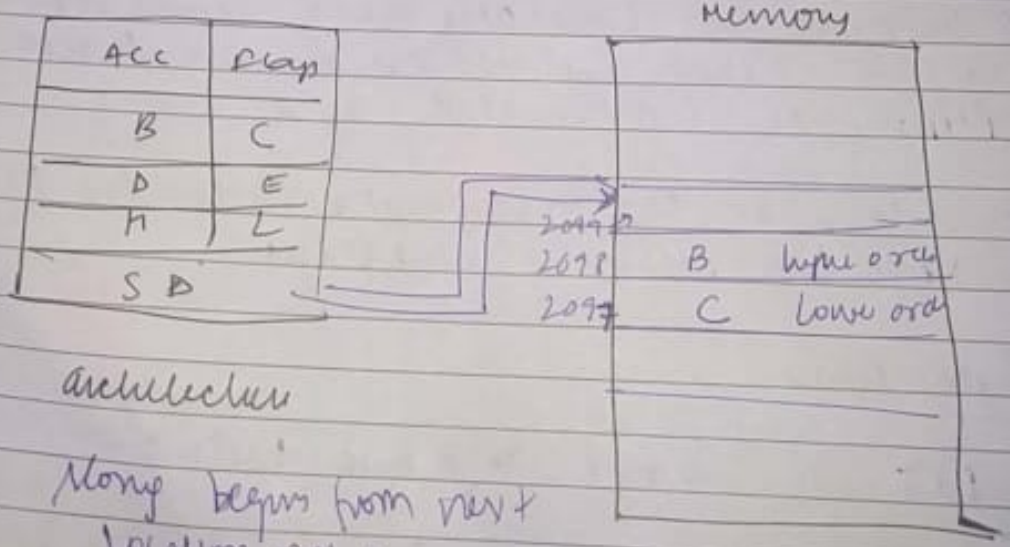
Instructions

1) LXI SP, 16 bit → 3 byte instruction

loading SP with starting of stack
where stack should begin in memory

2) Push register Pair (RP)
PUSH H
PUSH B
PUSH D
PUSH PSW

a) It is a one byte instruction
b) It copies the contents of the specified register pair as follows
i) Stack pointer register is decremented by 1 and the contents of the higher order register (eg B) are copied in the location
ii) SP decremented by 1
iii) Contents of lower order register (Eg C) are copied to the given location in the stack pointer
iv) PSW includes the contents of the Accumulator & the flags.



architecture

Memory begins from next location not 2099 but 2098

3) POP register pair (RP)
POP H
POP B
POP D
POP PSW

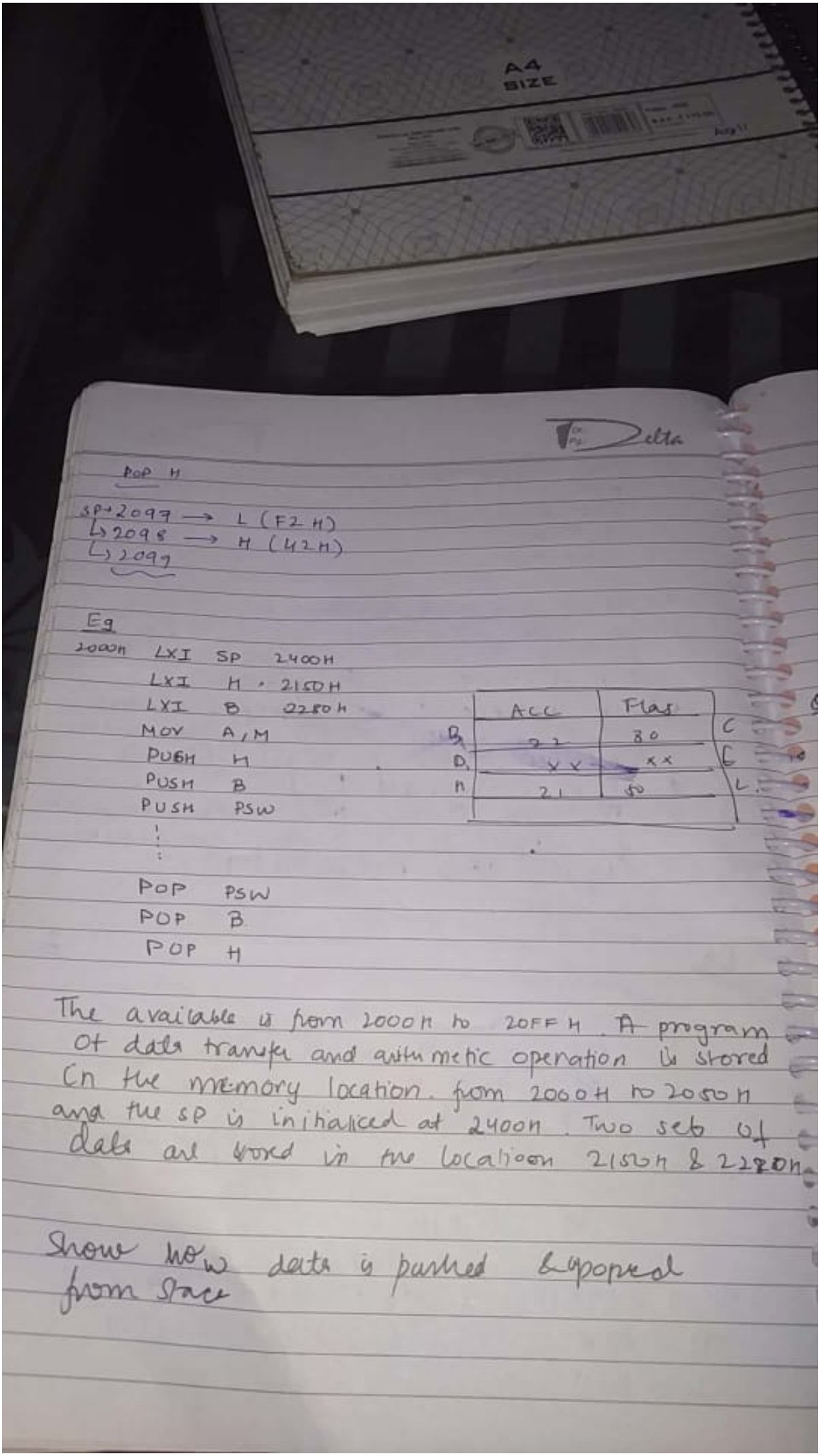
a) 1 byte instruction
b) It copies the contents of the top two memory locations of the stack into the specified register pair
c) The contents of the memory location indicated by the SP are copied into the lower order register (eg L)
d) SP is incremented by 1
d) Contents of the next memory location are copied to the higher order register (eg H). SP is incremented by 1

Eg

```
2000    LXI SP, 2099 H
03      LXI H, 42F2 H
06      POSH H
07      Delay Counter   } providing delay
        .
        .
10      POP H.
```

POSH H
SP → 2099 H
H → 42H
L → F2H

POSH H
SP → 2098 ← H ← 42H
L → 2099 ← L ← F2H



PUSH. Stack

2400	
23FF →	H
23FE →	L
23FD →	B
23FC →	C
23FB →	A
23FA →	flags

Data

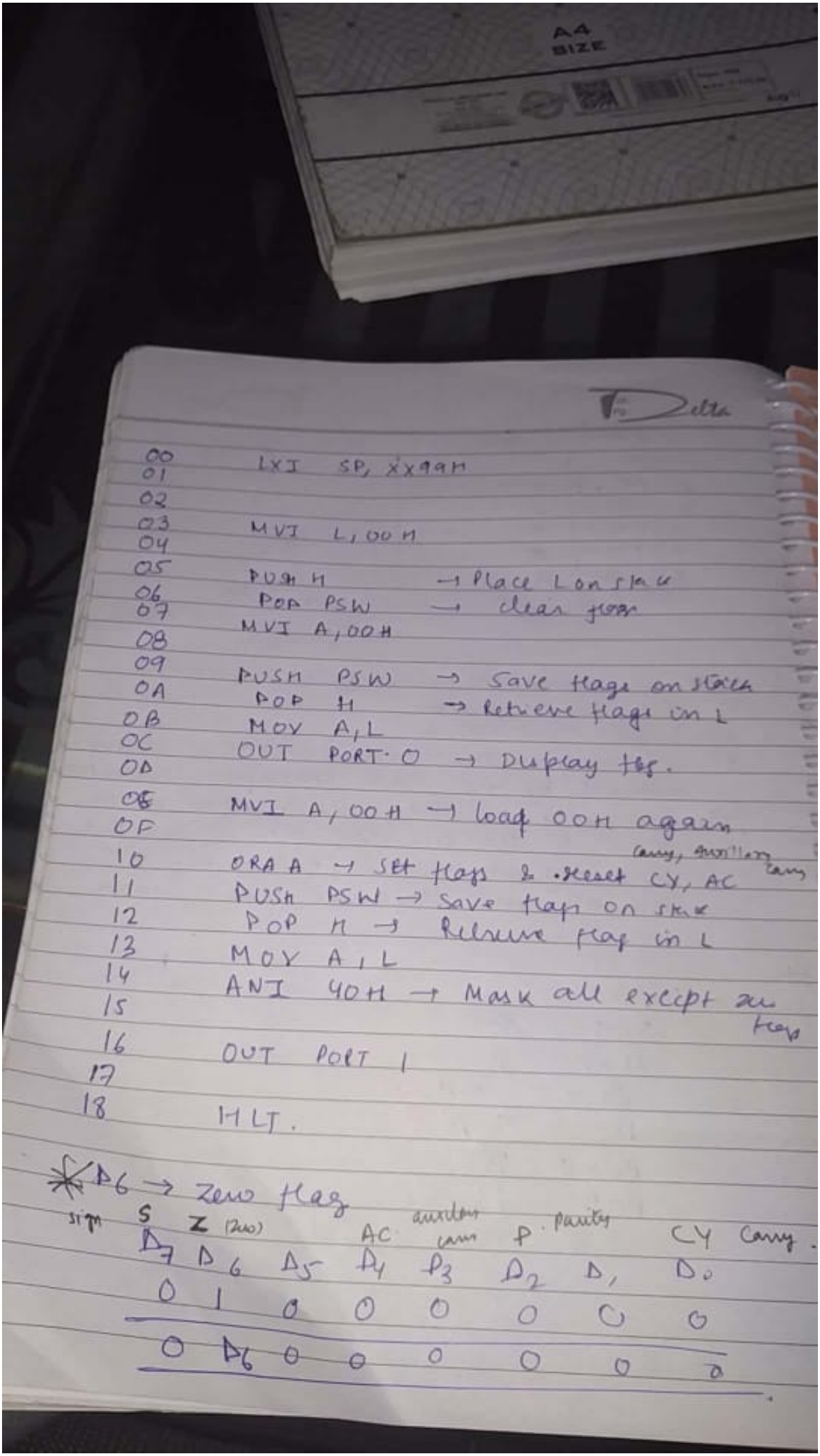
H
L
B
C
A
flags

POP

Q WAP to perform the following instruction
1) Clear all the flags
2) Load 00H in A.
3) Demonstrate that the zero flag is not affected by the data transfer instruction
4) Logically OR the accumulator with itself to set zero flag & display the zero flag at the OUTPUT PORT 01 or Store all the flags on the stack.

Ans

PUSH PSW
POP B.
MOV A,C.
ANI 00H.
MOV C,A.
PUSH B.
POP PSW.



Delta

```
00      LXI  SP, XX99H
01
02
03      MVI  L, 00 H
04
05      PUSH H          → Place L on stack
06      POP  PSW        → clear flags
07      MVI A, 00H
08
09      PUSH PSW        → Save flags on stack
0A      POP  H          → Retrieve flags in L
0B      MOV  A, L
0C      OUT  PORT 0     → Display flags.
0D
0E      MVI  A, 00 H    → load 00H again
0F                                  carry, auxiliary carry
10      ORA A  → SET flags & Reset CY, AC
11      PUSH PSW → Save flags on stack
12      POP  H  → Retrieve flag in L
13      MOV  A, L
14      ANI  40H → Mask all except zero
15                                  flag
16      OUT  PORT 1
17
18      HLT.
```

* D6 → Zero flag

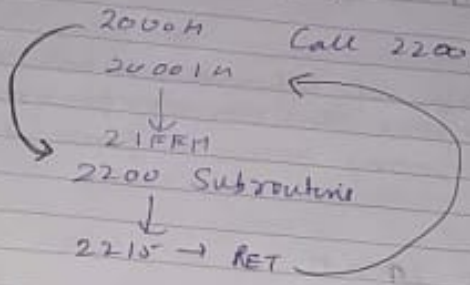
sign S Z (zero) AC auxiliary carry P parity CY Carry.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	1	0	0	0	0	0	0
0	D_6	0	0	0	0	0	0

CEC 10: Microprocessors

SUBROUTINES

group of instructions



① CALL 3 byte instruction
② RET 1 byte instruction

CALL: The contents of the program counter which is the sixteen bit address of the instruction following the call instruction is stored in the stack and the program execution is transferred to the subroutine address.

- It is a 3 byte instruction
- It saves the contents of PC on the stack
- It decrements the stack register by 2
- Transfers / Jumps unconditionally to the memory location specified by the 2nd and the third byte.
- This instruction is accompanied by return instruction

RET: → It is a one byte instruction
→ It inserts the 2 bytes from the top of the stack into the PC and increments the SP register by 2
→ It unconditionally returns from the subroutine.

Indicate the exchange of information between the stack and PC for the following program if the available memory range is from 2000H to 23FFH.

2000 LXI SP 2400 H

↓

2040 CALL 2070 H → W & Z registers.
2041
2042 → 2070 H
2043 Next instruction ← 207F H
↓
205F HLT
↓
2070 First Subroutine instruction
↓
207F RET
2080 ↓
2098 Other subroutine
↓
23FE
23FF Empty space] address pushed here

2400 → SP is initialized.

W, Z → Temporary registers

CALL → opcode fetch, load address, go to stack
5 Machine cycles, 18 T-States

M_1 → Opcode fetch

1) Contents of the PC are placed on address bus
2) The Instruction code is fetched
3) The Program Counter is incremented to 2041
4) The instruction decoded & executed
5) The Stack pointer is decremented by 1 to 23FF H

M_2 → Memory read
M_3

1) The sixteen bit address 2070H of the call instruction is fetched
2) 70H is fetched first and placed in the internal Z register.
3) 20 H is fetched and placed in the internal W register
4) During M3 the PC is upgraded to 2043H

M_4 and M_5 → Storing of PC

1) At the beginning of M_4, the normal operation of placing the content of the PC on the address line is suspended. The
2) The contents of SP register 23FFH are placed on the address bus

3) The higher order byte of the PC is placed in the 23FF H
4) The stack pointer is decremented by 1 to 23FE H
5) During Mc the contents of the SP are placed on the address bus (23FE H)
6) The lower address byte (43H) is placed in the data bus and is stored in the stack location 23FE H

Next instruction cycle

On this the program execution sequence is transfred to the call location 2070 H by placing the contents of the W and Z register onto the address bus

During the next instruction cycle PC is upgraded to the location 2070 H

* **Return** RET

(M1) → Opcode fetch and decoding of the return instruction takes place

(M2) → Contents of SP placed on the address bus
1) 43H placed on top of stack is fetched & stored in Z register
2) SP incremented by 1

(43) → 1) 20H is copied from stack & stored in W register
2) SP is incremented by 1
3) Program sequence is then transferred by placing the contents of the WZ register (2048H) on the address bus at the beginning of next instruction cycle.

Conditional Call

1) CC → Call if carry flag is set (1)
2) CNC → Call if Carry flag is reset (0)
3) CZ → Zero flag set (1)
4) CNZ → Zero flag reset (0)
5) CM → Sign flag set (S=1) Number → -ve
6) CP → Sign flag reset (S=0) number → +ve
7) CPE → even parity
8) CPO → odd parity.

Conditional Return.

1) RC
2) RNC
3) RZ
4) RNZ
5) RM
6) RP
7) RPE
8) RPO

comparison b/w call-ret & push-pop

Call-Ret	Push-pop
1) In this instruction HP automatically stores the 16 bit address of the next instruction on the stack	1) The programmer uses the instruction push to save the contents of a register pair on the stack
2) During call execution the SP is decremented by 2	2) When push is executed SP is decremented by 2
3) The Return transfers the contents of the top 2 locations of the stack to the Program Counter (PC)	3) Pop transfers the contents of the top 2 locations of the stack to the specified register pair
4) During the ret execution SP is incremented by 2	4) In Both Pop, SP is incremented by 2
5) Condition instruction are available while using call & return.	5) For push & pop they are not available