Date

Page

Module 2

Advanced Progr. Technique

Looping, Counting, Indexing

↓ Repetition of task no. of times

↓ How many times?

↓ Refer or pointing to the mem. address

Looping: is the basic structure which forces the MPU to repeat the sequence of instruction for a particular number of times.
eg add^n of no. of bytes, transfer mem. bytes to another mem. location. etc.

Counting: allow the programmer to count how many times the set of instruction are executed. by the the MPU.

Indexing: method that allow the programmer to point or refer the data stored at sequential memory locations.

Two types of Counter — ① 8-bit counter

MVI R, 8-bitdata
MVI C, 05H. } Initialization section.

assign value to any reg. of MPU.

max value of counter MVI C, (255)H. 2^8-1.
↓
FFH.

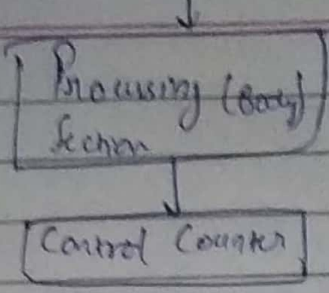
↳ Limitation upto 255 times. So we use ↙

② 16-bit counter : LXI Rp, 16 bit value
LXI B, FFFFH. } Initialization Section ↓

max $2^{16}-1$.

assign value to reg. pair of MPU.

MVI C, 05H [Initialization]

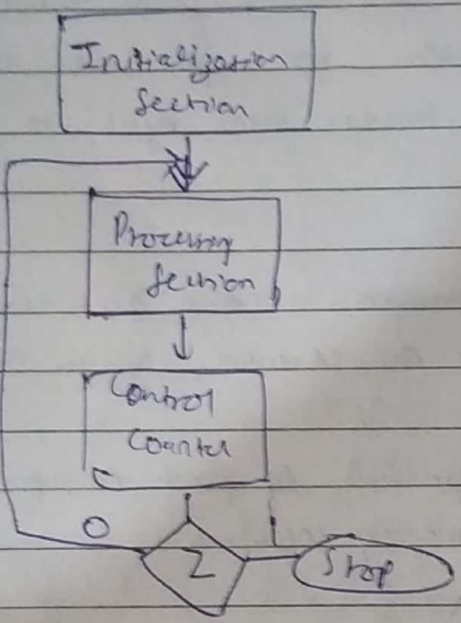


loop
≡

DCR C (decrement counter).
JNZ loop.
↓
(Jump not Z)
~~If zero~~ (Z is not 1)
Z=0 (condn true)

When Z → 1
exit.

05	Z=0
04	Z=0
03	Z=0
02	Z=0
01	Z=0
00	Z=1.



16-bit

```
     LXI  B, FFFCH
loop ≡
     DCX  B       (no flag affected).
     MOV  A, B    16-bit operation.
     ORA  C       with 8085.
     JNZ  loop.
```

B C OR operation
FFFF → FF
↓
FFFE → FF
FF00 → FF

$\left.\begin{matrix}\text{FFFF} \to \text{FF} \\ \text{FFFE} \to \text{FF} \\ \text{FF00} \to \text{FF}\end{matrix}\right\} Z=0$

$\left.0000 \to 00\right] Z=1$

F1 + 01   F1 00
0
F 1   F1
F000 = F0
00

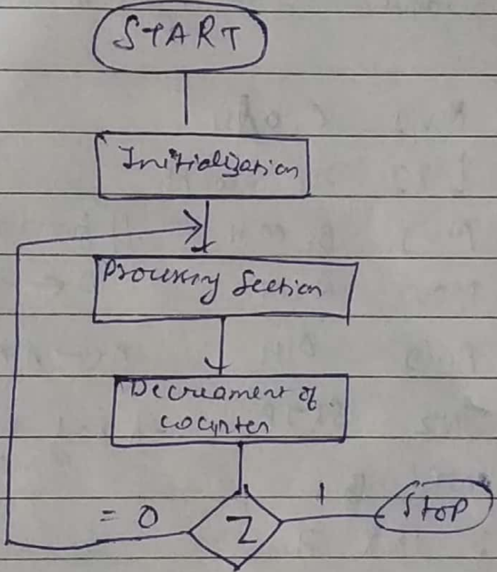
Q LXI B, F101H → How many times this loop will run?
Sol F101 times.

Q LXI B,F101H. How many times?

```
loop ≡
DCX B
MOV A,C
ORA B
JNZ loop.
```

Ans → F101.

Q if JNC loop. → infinite times.
carry (~~Carry~~ not generated in logical operation).



Q WAP to transfer memory bytes from 3000 to 3005
4000 to 4005.

Sol

6 Counter (0 to 5)

```
        MVI  C, 06H    // counter
        LXI  H, 3000   // pointer to read the value.
        LXI  D, 4005   //  "      " write  "   ".
L1 :    MOV  A, M
        STAX  D
        INX   H
        INX   D
        DCR   C
```

5
6

JNZ L1

Q1 Copy 3000 → 4005
3001 → 4004

3005 → 4000

Q2 Overlapping

3000	1F	3002	1F
3001	2F	3003	2F
3002	3F	3004	3F
3003	4F	3005	4F
3004	5F	3006	5F
3005	6F	3007	6F

Q3 Multiplication → 03 by 04.

Q Find, how many no. are even in a given list.

Starting address = 3000H   Ten bytes to be check.

```
        MVI  C, 0AH
        LXI  H, 3000H
        MVI  B, 00H     // for Hold the result.
BACK:   MOV  A, M       A ← M[HL]
        ANI  01H        A ← A ∧ 01H        = even Z=1
                                              odd Z=0
        JNZ  SKIP       if Z=1 then
                        increment B
                           otherwise skip.
        INR  B
SKIP:   INX  H
        DCR  C
        JNZ  BACK
```

JNZ ↓ change JZ ← for odd

Z=1

Q counting of ⊖ve no. → D7 → 1.

$\underbrace{1000}\ \underbrace{0000}$
80H. 01

⊖ve. $\left\{ \begin{array}{l} \text{ANI 01H} \xrightarrow{\text{change}} \text{ANI 80H} \\ \text{JNZ} \xrightarrow{\text{change}} \text{JZ.} \end{array} \right.$

for ⊕ve

$\left. \begin{array}{l} \text{JNZ} \xrightarrow{\text{change}} \text{JNZ} \\ \text{ANI 01H} \xrightarrow{\text{change}} \text{ANI 80H} \end{array} \right\}$ ⊕ve

$\frac{2\,\%}{4}$

Date
Page

Q even and odd.

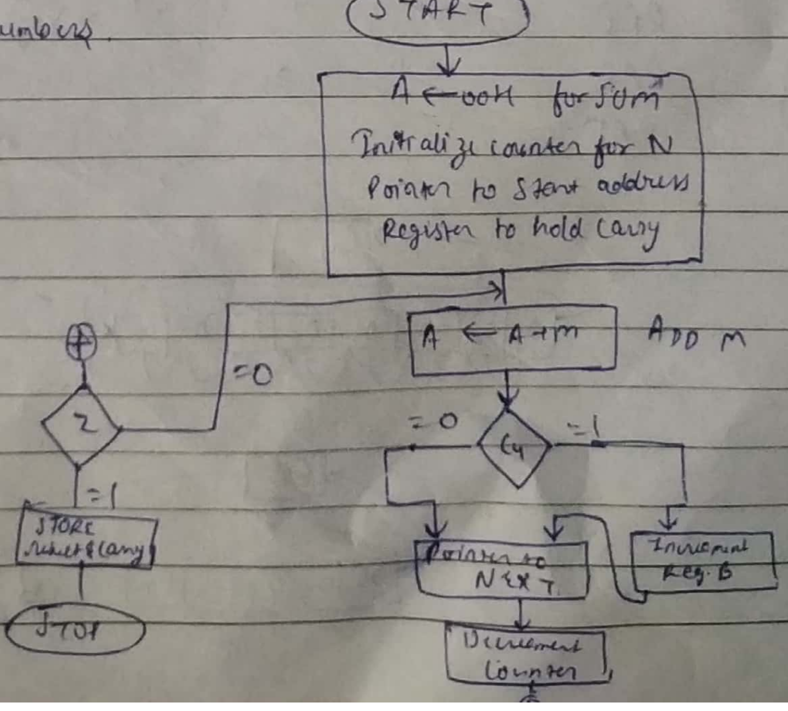
```
        MVI C, 0A H
        LXI  H, 3000
        MVI  B, 00H   — all even
        MVI  D, 00H   — all odd
BACK :  MOV  A, M
        ANI  01
        JNZ  ~~STOP~~ ODD   → if false then go to odd.
        INR  B
        JMP  NEXT.
ODD     INR  D
NEXT    INX  H
        DCR  C
        JNZ  BACK.
```

for ⊕ ⊖ve

B → all ⊕ve pos
D → " ⊖ve. neg.

ANI → ANI 80H
JNZ ~~ODD~~ NEG
ODD → NEG.

Q Sum of n numbers.



```
        MVI   A, 00H
        MVI   C, 06H *
        LXI   H, 3000H
        MVI   B, 00H    → hold carry.
        ~~ADD~~
        ~~MOV   A, M~~
BACK    ADD   M
        JNC   SKIP
        INR   B
SKIP:   INX   H
        DCR   C
        JNZ   BACK
        STA   2FFE
        MOV   A, B
        STA   2FFF
        HLT.
```

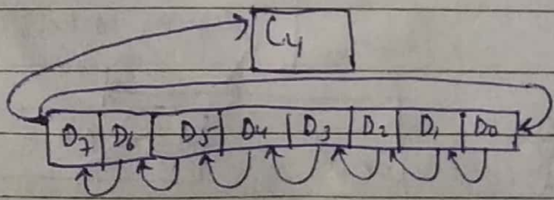
Rotate Instruction :

First transfer to accumulator

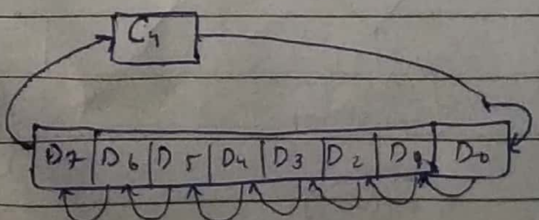
RLC	Rotate without carry in left.
RRC	" " " " right.
RAR	" with " " "
RAL	" " " " left

update

$D_0 \leftarrow D_7$
$C_y \leftarrow D_7$
$D_{n+1} \leftarrow D_n$ } left



$C_y \leftarrow D_0$
$D_7 \leftarrow D_0$
$D_n \leftarrow D_{n+1}$ } Right.



Date
Page

Q 7E → Count no. of 1's in particular no.

0111 1110

Carry flag set $\xrightarrow{\text{then}}$ count

```
        MVI C,08H
        MVI B,00H
        MVI A,7EH
BACK:   RLC
        JNC SKIP        (0 4 SKIP)
        INR B
SKIP:   DCR C
        JNZ BACK
        MOV A,B
        STA 3000H
        HLT.
```

Q WAP to Count how many 1's in 16-bit no.

```
        MVI B,00H
        LXI D,XXXXH.. DE
        ~~MVI C,16H~~                    D E
        MOV A,E
:BK:    ~~RAR~~ RLC                      Lo  Ho
        JNC SKIP.
        INR B.
SKIP :  DCR C
        JNZ BK
          MVI C,08H
          MOV A,D
        D. Repeat
          MOV A,B
          STA 3000
          HLT.
```

Scanned with CamScanner

Multiply 16 bit with 8 bits

Double add: DAD D [HL ← HL+DE]

```
     LXI  D, XXXXH
     MVI  C, XXH    //Counter
     LXI  H, 0000H
BK : DAD  D
     DCR  C
     JNZ  BK
     SHLD 3000
     HLT.
```

```
HL  0000
DE  XXXX
--------
HL  XXXX
DE  XXXX
--------
    2X 2X 2X 2X
```

for 8 bit: LXI D, ~~00~~XXXXH ⟶ LXI D, 00XX H.

Delay

Delay by using 8-bit loop

```
    MVI  B, FF  ↗max   - 7T
L1: DCR  B             - 4T → 255
    JNZ  L1            - 10/7T → 255
                          ↓  when cond^n true
                          ↘ 1  false
```

How much delay generated?
processor - frequency = 2MHz.

$T \text{(clock period)} = \frac{1}{f} = \frac{1}{2MHz} = \frac{1}{2\times10^6} = 0.5\times10^{-6}$
$= 0.5 \mu \text{sec}$.

Total T-state $= T_{MVI} + T_{loop}$
$= ~~7 + [(4+10\times255)]~~$
$= 7 + (4+10) \times 255 - 3$
$= 3574$

Date
Page

Total delay = Total T-state × CP
= 3574 × 0.5 usec
= 1787 usec
= 1.787 msec.

Delay by using 16-bit loop

```
     LXI H, 1000H    → 10
BK:  DCX H           → 6
     MOV A, H        → 4
     ORA L           → 4    loop
     JNZ BK          → 10/7
```

Total T-state = $T_{LXI} + T_{loop}$

$\nearrow 2^{16} - 1$

$= 10 + (6+4+4+10) \times (1000)_H - 3$

$= 10 + (24) \times 4096 - 3$

$= 10 +$

$= 98311$

TD = 98311 × 0.5 usec
= 49155.5 usec
= 0.05 sec

Max delay :-

LXI H, FFFFH
same as previous

$= 10 + (6+4+4+10) \times (2^{16}-1) - 3$

$= 10 + 24 \times 65536 - 3$

$=$ ~~10+~~ 1572871

TD = 1572871 × 0.5
= 786435.5
= 17864 ms

Date
Page

For 1sec use nesting.

Delay by using Nested loop -

```
      MVI C, 02H
L2:   LXI H, FFFF H
BK:   DCX H
      MOV A, H
      ORA L
      JNZ BK
      DCR C
      JNZ L2
```

Total T-state = 98311 + 14

↓

= T_{MVI} + T_{outer loop}

= 7 + (T_{inner} + 4 + 10) * (counter) - 3

= 4 + (98311 + 14) * 02

= ~~4+~~ 196654

TD = 196654 × 0.5 μs

= 98327 μsec

= ·1 sec.

Q4 Calculate the delay for nested loop both inner & outer are 8-bit nested loop.

OR

in exam.
→ Q What will be the value of counter for the delay of 1ms by using 8-bit loop.

TD = T-state * Clock period

T-state = T_{MVI} + T_{loop}

= 7 + (4+10) * (counter) - 3

= 7+14 n - 3 = 4+14n

1ms = 1000 μs

1000 μs = 4+14n × 0.5 μsec.

1000 = 4 + 7n

996 = 7n

n = 142.28 ≃ 142

Scanned with CamScanner

0001 0100 ~~0010~~

Date

Page

0001 0100 0010

$(n)_{10} = (142)_{10}$

$= \underbrace{1000}_{8}\underbrace{1110}_{E} = 8E$

Q Find the delay ~~for two loops~~ in 8 bit loop (nested)

MVI B,FFH — 7

L2 : MVI C,10H

BK : DCR C — inner 228

JNZ BK

DCR B — 4

JNZ L2 — 10/7

Step 1. Calculate the T-State of inner loop

Total t-st = $7 + \cancel{7}\ \cancel{8}\ \overset{(4+10)}{14 \times 16} - 3$

~~= 4 + 84 = 88~~ = 228.

Step 2. For outer loop

$7 + 228 + \cancel{440}\overset{(4+10)255}{} - 3$

$= 232 + 3570$

$= 3802$

$7 + \cancel{228}(228 + 4 + 10)\,255 - 3$

$= 4 + 61710$

$= 61714$

Step 3 Apply total delay formula

$TD = \text{Total T-state} \times \frac{1}{f}$

$= (61714 + 228) \times \frac{1}{2}$

$= 30971$

Q Find the value of counter for the delay of 5ms of the MUP which is working with frequency 2MHz.

Step 1 Instr for 16-bit loop with counter x.

```
      LXI H, XXXX (counter = n)   - 10
BK:   DCX H                       - 6
      MOV A, H                    - 4
      ORA L                       - 4
      JNZ BK                      - 10/7
```

$= 10 + 24x - 3$

$\Rightarrow (7 + 24n) \frac{1}{f} = 5ms$

$= (7 + 24n) \frac{1}{2} = 5ms$

$0.5 \mu s \times (7 + 24n) = 5ms$

$0.5 \mu s (7 + 24n) = 5000 \mu sec$

$7 + 24n = \frac{5000.0}{0.5} = 10000$

$24n = 10000 - 7$

$n = 416.37 \cong 416$

$= 416 \Rightarrow 01A0$ Ans

0000 0001 1010 0000

256

Q. Write assembly language prog. to get square of a no. which is stored at the location 3000 Store result at 3001.

```
              M 3000
~~MVI~~  LXI  H, XX00H
~~MVI~~ C, 9
~~XRA~~ A        (Clear acc.)
SUB
```

```
      MOV B,M
loop  ADD M
      DCR B
      JNZ Loop
      STA ~~[illegible]~~ 300 H
      HLT
```

```
     LXI H,3000
     MOV B,M
     ~~MOV C,M~~
     SUB A
BK:  ADD B
     DCR C
     JNZ BK
     STA 3009
     ~~HLT.~~
```

(not for 256)

$(16)^2 \nearrow X$

for 16-bit LXD " ,3000H

w/ DAD ADDB → DAD B

Q5 Find the min. / smallest no. in the array of size 20 starting from the mem. location 3000.

Q6 Divide a 16-bit no. by 8-bit number. Store both the results quotient & remainder at the location 3000 & 3001 respectively.

Decimal Adjust Accumulator DAA

```
 47        add 6 if result greater than 9       47
 17                                             17
 5E                         1(4)+6             1 5E
           47                                   64   Ans
           77
           8E
        1) 24
```

↳ It is useful to convert in decimal form.

Stack & Sub-Routine

Stack is a linear data structure at MPU level it is use reversal memory area in RAM for storing temporary information to perform read or write operation on to the stack it uses pop and push function respectively. It means by using pop function you can retrieve or delete the value from stack. Another side by using push function it insert the value onto the stack It also uses 16-bit stack pointer which hold the address of top element that's why it is also known as top of the stack with the designation SP (stack pointer).

① PUSH R_p
PUSH B
SP ← SP−1
M[SP] ← Higherbyte [B]
SP ← SP−1
M[SP] ← lowerbyte[C]

② POP Rp
POP D
E ← M[SP]
SP ← SP+1
D ← M[SP]
SP ← SP+1.

↳ SPHL : load content of HL into SP. SP ← HL
↳ PCHL : " " " " " PC. PC ← HL
↳ XTHL : exchange " " " . M[SP] ⟷ L
M[SP+1] ⟷ H

↳ PSW : Program status word.
PSW = Content of A + Content of Flag reg.
PUSH PSW.

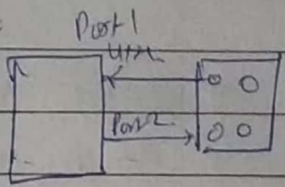
Use : ① To see status of flag reg. PUSH PSW
POP H

Sub-Routine : A specific set of Instruction useful to perform special operation is called subroutine.

or

A block of code which is called no. of times in a program that block is called routine.

Light	bit	Time
Green	D_0	50 sec
Red	D_2	30 sec
Yellow	D_4	10 sec
Don't walk	D_6	50 sec
Walk	D_7	40 sec



G DW
$\underbrace{0100}_{4}\ \underbrace{0001}_{1H.}$

R DW
$\underbrace{1000}_{8}\ \underbrace{0100}_{4H.}$

Y
$\underbrace{1001}_{9}\ \underbrace{0000}_{0H}$

```
Loop:   MVI A,41H.
        OUT PORT1.
        MVI B,32H.   } → Delay for 50 sec.
        CALL DELAY.
        MVI A,84H
        OUT PORT1
        MVI B,1EH.   }  30 sec.
        CALL DELAY
        MVI A,90H
        OUT PORT1
        MVI B,0AH.
        CALL DELAY
        JMP LOOP
```

8086 μp :

↳ 16 - bit MUP.
↳ developed in ~~1886~~ 1986
↳ Pipeline Architecture

Differences b/w 8085 & 8086

8085	8086
① 8-bit MUP - Wordsize	① 16 - bit MUP - Word size
② Address line - 16 address line	② 20 - bit address bus
③ Max. access 2^{16} = 64K	③ max. access 2^{20} = 1mB.
④ Uniprocessor Architecture It does not support pipeline architecture.	④ support pipeline architecture (2- stage " 4).
⑤ Multiprocessing Support - Not Support by 8085	⑤ It support multiproc
⑥ 256 I/O port.	⑥ using 16 I/O address lines. 2^{16} = 65536.
⑦ Coprocessor Interface : Not support	⑦ Has a coprocessor interface in the form of 8087
⑧ Addressing mode i 5 add. mode.	⑧ 8 add. mode (5 of ~~add~~ 8085) and 3 of itself.
⑨ Cost is low	⑨ Cost is high as compared to 8085.
⑩ Memory space is not segmented	⑩ Segmented (Data, stack, program segments etc.).

Scanned with CamScanner

Date
Page

Features of 8086 : It operates in two modes.

① Maximum mode ② minimum mode

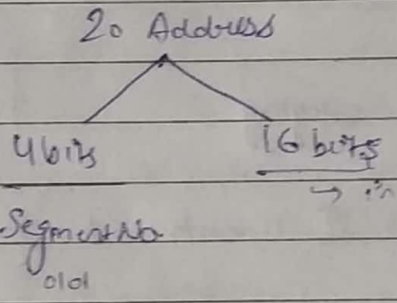
↓ use ↓

When we 8086 with coprocessor 8087 then enable ~~the~~ this mode

as 8085, working as a simple processor.

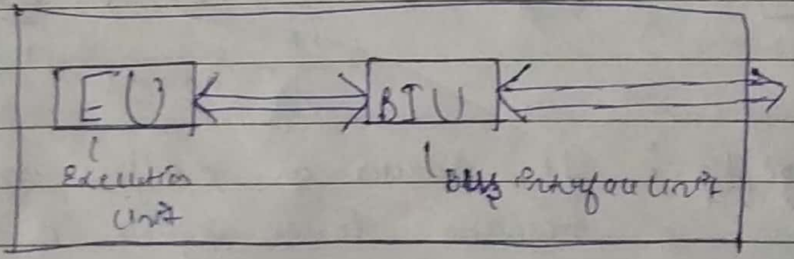
Memory Segmentation -

② Memory ~~segmented~~ divided into 16 - segments of capacity 2^{16} with the name code segment (CS), data segment (DS), stack segment (SS) or extra segment (ES).



③ It has ~~4~~ 256 vectored interrupts.

④ It also support powerful instruction set multiply, divide operation.



8086 Pipeline

BIU : perform fetching of an instruction or data It fetches from the memory or I/O device which are connected with the I/O ports. It also write the data to the memory or I/O ports. It sends the addresses to the memory and ports. So we can say BIU is called the external word interface of the processor.

EU: The execution unit tells the BIU from where to fetch the instruction or data. It decoded the fetch instruction and execute it. So we can say takes care of performing operation on the data provide by the BIU. So it is the heart of the processor.

Architecture of 8086 :

Flag - Two category

① Control flags ② Conditional flags
↓
There are three flags — ⓐ Trap flag (TF)
ⓑ Interrupt flag (IF)
ⓒ Direction " (DF)

Conditional - 6 flags $\underbrace{\text{CF, AF, ZF, SF, PF}}_{\text{previous}}$, OF (overflow)

Overflow flag : It indicates an overflow from the magnitude to the sign bit of the result. If OF is set an arithmetic overflow has occured because the size of the result exceeded the capacity of destination location. Whenever in 8086 interrupt or overflow instruction is available that will generate an interrupt to the processor.

Date

Page

<u>Trap flag</u>: Whenever trap flag will be set it is for single step debugging it means MUP execute a instr. and enter into single step 'ISR' (Interrupt Service routine).

<u>IF</u>: If user sets IF flag the CPU will recognize external ~~input~~ interrupt request clearing IF disable these interrupts.

<u>DF</u>: This bit is for string instruction in string instr. we use SI (Source index register) and DI (destination index) as offset register to point source area and destination area respectively. DF flag controls direction of SI and DI pointers. It means if DF = 1 the string instruction will automatically decrement the pointers it means it process string from high addresses to low addresses. In other case if DF = 0 the string instr. will automatically increment the pointer from lower addresses to higher addresses.

F=1 (right to left)

=0 (left to right)

Architecture —

<u>Addressing Modes</u>

① Implied Addressing
② Register "
③ Immediate "
④ Direct "
⑤ Register Indirect "
⑥ Based - 8-bit or 16-bit Instruction operand
⑦ Indexed " " " "
⑧ String Addressing
⑨ Direct I/O port addressing
⑩ Indirect " " "
⑪ Relative addressing
⑫ Implied "