Microprocessor Technologies (102045610)

MODULE 4

PROGRAMMING TECHNIQUES

Module 4

- Stack & Subroutines
- Developing Counters and Time Delay Routines
- Code Conversion
- BCD Arithmetic and 16-Bit Data operations

Stack

- Stack is a group of memory location in the R/W memory that is used for temporary storage of binary information during execution of a program.
- The starting memory location of the stack can be defined in program and space is reserved usually at the high end of memory map.

E.g.: **LXI SP,FFF8**; loads 16-bit memory address in stack pointer

Stack

Instruction necessary for stack are as follows:

LXI SP, 2095	**Load the stack pointer register with a 16-bit address.**
PUSH B/D/H	**It copies contents of register pair on the stack**
PUSH PSW	**Operand PSW represents Program status word meaning. i.e. content of accumulator and flags.**
POP B/D/H	**It copies content of top two memory locations of the stack in to specified register pair**
POP PSW	**It copies content of top two memory locations of the stack in to accumulator and flags respectively.**

Subroutine

- A subroutine is a group of instruction that performs a subtask of repeated occurrence.
- A subroutine can be used repeatedly in different locations of the program.

Subroutine

Advantage of using Subroutine

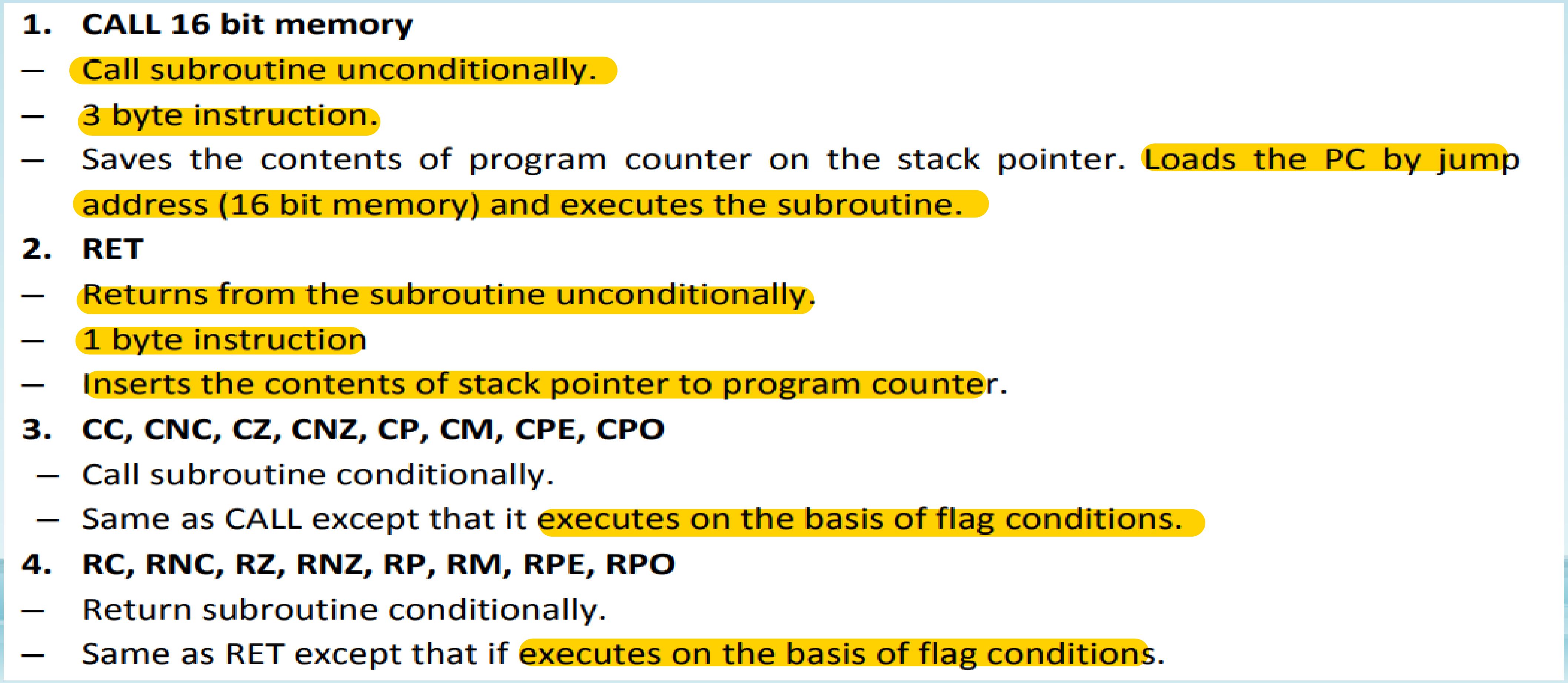
- Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations.

Where to write Subroutine?

- In Assembly language, a subroutine can exist anywhere in the code.
- However, it is customary to place subroutines separately from the main program.

Subroutine

- The 8085 has two instructions for dealing with subroutines.
  - The **CALL** instruction is used to redirect program execution to the subroutine.
  - The **RET** instruction is used to return.



1. **CALL 16 bit memory**
   - Call subroutine unconditionally.
   - 3 byte instruction.
   - Saves the contents of program counter on the stack pointer. Loads the PC by jump address (16 bit memory) and executes the subroutine.
2. **RET**
   - Returns from the subroutine unconditionally.
   - 1 byte instruction
   - Inserts the contents of stack pointer to program counter.
3. **CC, CNC, CZ, CNZ, CP, CM, CPE, CPO**
   - Call subroutine conditionally.
   - Same as CALL except that it executes on the basis of flag conditions.
4. **RC, RNC, RZ, RNZ, RP, RM, RPE, RPO**
   - Return subroutine conditionally.
   - Same as RET except that if executes on the basis of flag conditions.

Subroutine

Things to be considered in Subroutine

- Number of **PUSH** and **POP** instruction used in the subroutine must be same, otherwise, **RET** instruction will pick wrong value of the return address from the stack and program will fail.

E.g. **Write an ALP to add two numbers using subroutines.**

```
2000        MVI B, 4AH
2002        MVI C, A0H
2004        CALL 3000H
2007        MOV B, A
2008        HLT
```

SP = 2007 (i.e. PC)

PC = 3000 (i.e. 16bit)

```
3000    MOV A, B
3001    ADD C
3002    RET
```

PC = 2007 (i. e. sp)

Counters and Time Delays

Counters and Time Delays are important techniques.

Applications of Counters and Time Delays

They are commonly used in

1. Traffic Signal
2. Digital Clocks
3. Process Control
4. Serial data transfer

Counters and Time Delays

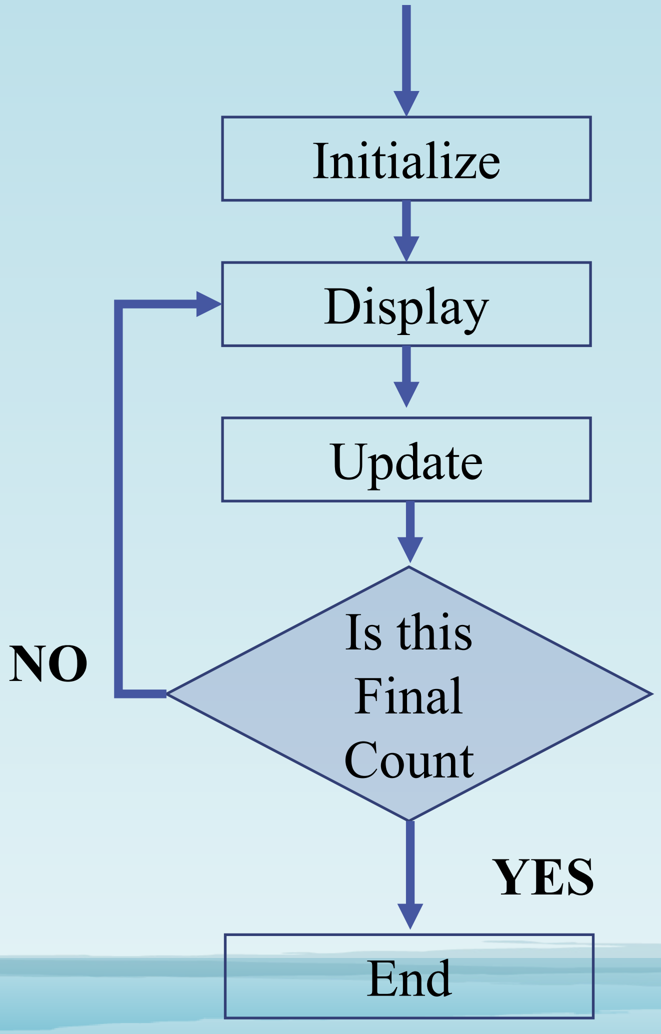
- A counter is designed simply by loading appropriate number into one of the registers and using INR or DCR instructions.

- Loop is established to update the count.

- Each count is checked to determine whether it has reached final number ;if not, the loop is repeated.

Counters

- How to create counters?



```
1.  MVI C,05
2.  LOOP: MOV A,C
3.        OUT 01
4.        DCR C
5.         JNZ LOOP
6.  HLT
```

Counters

Label	Opcode	Operand	Comment	T-states
	MVI	C,05h	; Load Counter	7
LOOP:	DCR	C	; Decrement Counter	4
	JNZ	LOOP	; Jump back to Decr. C	10/7

MVI C, 05h

Machine Cycle: **F + R = 2**

T-States: **4T + 3T = 7T**

DCR C

Machine Cycle: **F = 1**

T-States: **= 4T**

JNZ LOOP (false)

Machine Cycle: **F + R = 2**

T-States: **4T + 3T =7T**

JNZ LOOP (true)

Machine Cycle: **F + R + R =3**

T-States: **4T + 3T + 3T=10T**

Counters and Time Delays

- Instruction **MVI C, 05h** requires **7 T-States** to execute.
- Assuming, 8085 Microprocessor with **2MHz** clock frequency.
- How much time it will take to execute above instruction?

Clock frequency of the system (f)= 2 MHz

Clock period (T) = 1/f = ½ * 10^{-6} = 0.5 µs

Time to execute MVI = 7 T-states * 0.5 µs

= 3.5 µs

How much time it will take to execute above instruction with 1 MHz clock frequency?

Counters and Time Delays

Label	Opcode	Operand	Comment	T-states
	MVI	C,05h	; Load Counter	7
LOOP:	DCR	C	; Decrement Counter	4
	JNZ	LOOP	; Jump back to Decr. C	10/7

- Now to calculate time delay in loop, we must account for the T-states required for each instruction, and for the number of times instructions are executed in the loop.

- The for the next two instructions:

DCR:		4 T-States
JNZ :	+	10 T-States
		14 T-States

- Here, the loop is repeated for 5 times.

Counters and Time Delays

How to calculate time delay for given loop?

- Time delay in loop T_L with 2MHz clock frequency is calculated as:

$$\mathbf{T_L = T * Loop\ T\text{-}states * N_{10}} \text{ ------------------(1)}$$

T_L	: Time Delay in Loop
T	: Clock Period
N_{10}	: Equivalent decimal number of hexadecimal count loaded in the delay register

Substituting value in equation (1)

$$\mathbf{T_L = (0.5 * 10^{-6} * 14 * 5)}$$

$$= \mathbf{35}\ \mu s$$

Counters and Time Delays

- If we want to calculate delay more accurately, we need to accurately calculate execution of JNZ instruction

i.e

If **JNZ = true**, then **T-States = 10**

Else if **JNZ =false**, then **T-States = 7**

Counters and Time Delays

- Now, according to our program:

```
1.   MVI C,05
2.   LOOP:      DCR C
3.              JNZ LOOP
4.   HLT
```

Here, the last cycle will be executed in **7 T-States**; when **JNZ = false**

Therefore, there is difference of (10T – 7T) 3T-states:

Counters and Time Delays

Therefore, there is difference of (10T – 7T) 3T-states:

- Delay generated by last clock cycle:

= 3T * Clock Period

= 3T * ($1/2 * 10^{-6}$)

= 1.5 µs

- Now, the accurate loop delay is:

T_{LA}=T_L - Delay generated by last clock cycle

= 35 µs - 1.5 µs

= 33.5 µs

T_L= (0.5 * 10^{-6} * 14 * 5)= 35 µs

Counters and Time Delays

Now, to calculate total time delay

Total Delay = Time taken to execute instruction outside loop

+

Time taken to execute loop instructions

$$\mathbf{T_D = T_O + T_{LA}}$$

$$= (7 * 0.5\ \mu s) + 33.5\ \mu s$$

$$= 3.5\ \mu s + 33.5\ \mu s$$

$$\mathbf{= 37\ \mu s}$$

Counters and Time Delays

Calculate time delay and accurate time delay for given loop with

Counter value =255 (FF h) and

Clock frequency =2MHz

T_L= T * Loop T-states * N_{10} -----------------(1)

$= 0.5 * 10^{-6} * 14 * 255$

$= 1785\ \mu s = 1.8\ ms$

T_{LA}= Time to execute loop instructions

$= T_L$ – (3T states*clock period)

$= 1785 - (3 * ½ * 10^{-6})$

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

Exercise

1. How much time the 8085 microprocessor will take to execute the MOV B, A instruction, if the **crystal frequency** is 4MHz?
   ANS : 1μs

1. How much time will be required to execute the STAX B instruction if the **clock frequency** is 4 MHz?

2. How much time will be required to execute the MVI M,25h instruction if the **clock frequency** is 6 MHz?

Time Delay using a Register Pair

- Time delay can be considerably increased by setting a loop and using a register pair with a 16-bit number (FFFF h).
- A 16-bit is decremented by using DCX instruction.

Problem with DCX instruction

- DCX instruction doesn't set Zero flag.
- Without test flag, Jump instruction can't check desired conditions.
- Additional technique must be used to set Zero flag.

Time Delay using a Register Pair

Label	Opcode	Operand	Comment	T-states
	LXI	B,2384 h	; Load BC with 16-bit counter	**10**
LOOP:	DCX	B	; Decrement BC by 1	**6**
	MOV	A, C	; Place contents of C in A	**4**
	ORA	B	; OR B with C to chk Zero flag	**4**
	JNZ	LOOP	; if result not equal to 0, 10/7 jump back to loop	**10/7**

- Here the loop includes four instructions:

  Total T-States = 6T + 4T + 4T + 10T

  = **24 T-states**

Time Delay using a Register Pair

- The loop is repeated for 2384 h times.
- Converting $(2384)_{16} \longrightarrow (\_\_\_\_\_)_{10}$

$$2384\ h = (2 * 16^3) + (3 * 16^2) + (8 * 16^1) + (4 * 16^0)$$

$$= 8192 + 768 + 128 + 4$$

$$= \mathbf{9092}$$

- Clock frequency of the system (f)= 2 MHz
- Clock period (T) = $1/f = ½ * 10^{-6} = 0.5\ \mu s$

Time Delay using a Register Pair

Now, to find delay in the loop

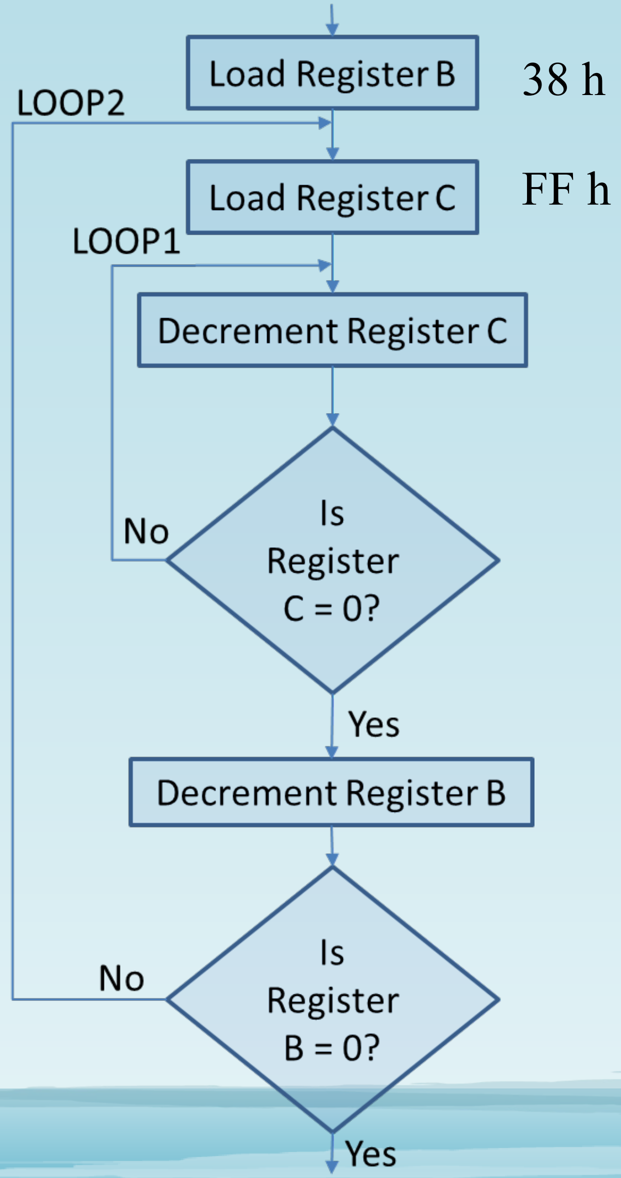
T_L= T * Loop T-sates * N_{10}

= 0.5 *10^{-6}* 24 * 9092

= 109104 μs

= **109 ms** (without adjusting last cycle)

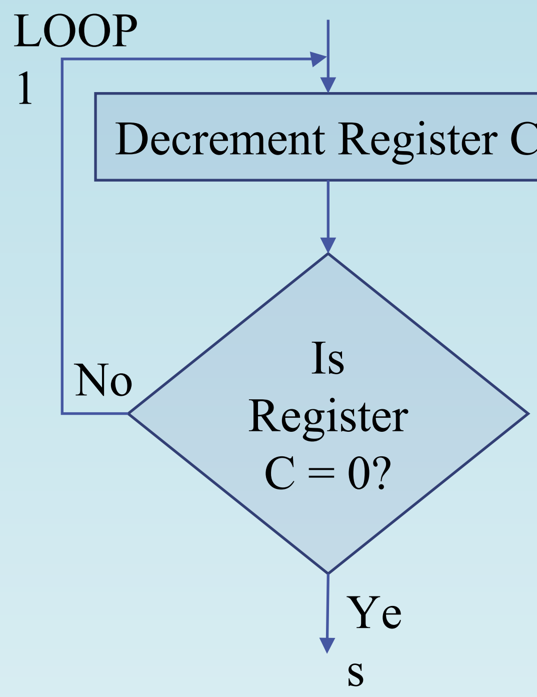
Time Delay using a LOOP within a LOOP



Label	Opcode	Operand	T-states
	MVI	B,38h	7T
LOOP2:	MVI	C,FFh	7T
LOOP1:	DCR	C	4T
	JNZ	LOOP1	10/7 T
	DCR	B	4T
	JNZ	LOOP2	10/7 T

Time Delay using a LOOP within a LOOP

- Calculating delay of inner LOOP1: T_{L1}



Label	Opcode	Operand	T-states
LOOP 1:	DCR	C	4T
	JNZ	**LOOP1**	10/7 T

$\mathbf{T_L = T * Loop\ T\text{-}states * N_{10}}$

$= 0.5 * 10^{-6} * 14 * 255$

$= 1785\ \mu s = 1.8\ ms$

$T_{L1} = TL - (3T\ states * clock\ period)$

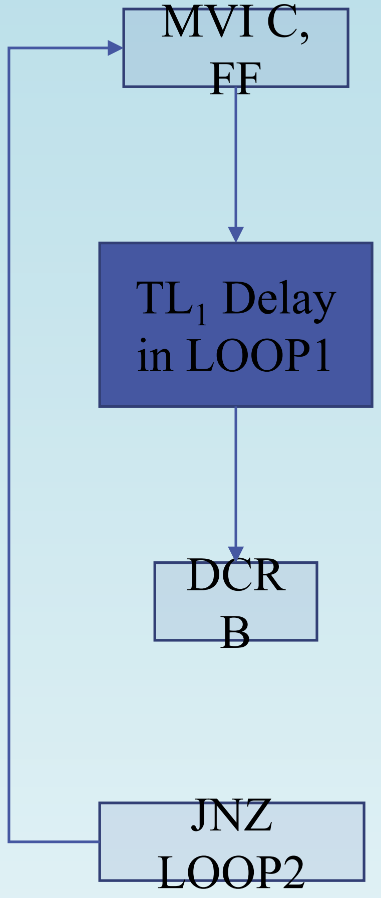
$= 1785 - (3 * ½ * 10^{-6})$

$= 1785 - 1.5 = \mathbf{1783.5\ \mu s}$

Delay of Loop1 $T_{L1} = \mathbf{1783.5\ \mu s}$

Time Delay using a LOOP within a LOOP

- Now, Calculating delay of outer LOOP2: T_{L2}



Label	Opcode	Operand	T-states
	MVI	B,38h	7T
LOOP2	MVI	C,FFh	7T
Delay of Loop1 T_{L1}= 1783.5 µs			
	DCR	B	4T
	JNZ	LOOP2	10/7 T

Counter B : $(38)_{16}$ = **$(56)_{10}$**

Loop2 is executed for 56 times

T-States = 7 + 4 + 10 = 21 T-States

$$\mathbf{T_{L2} = 56 (T_{L1} + 21\ T\text{-}States * 0.5)}$$

$= 56(1783.5\ \mu s + 10.5)$

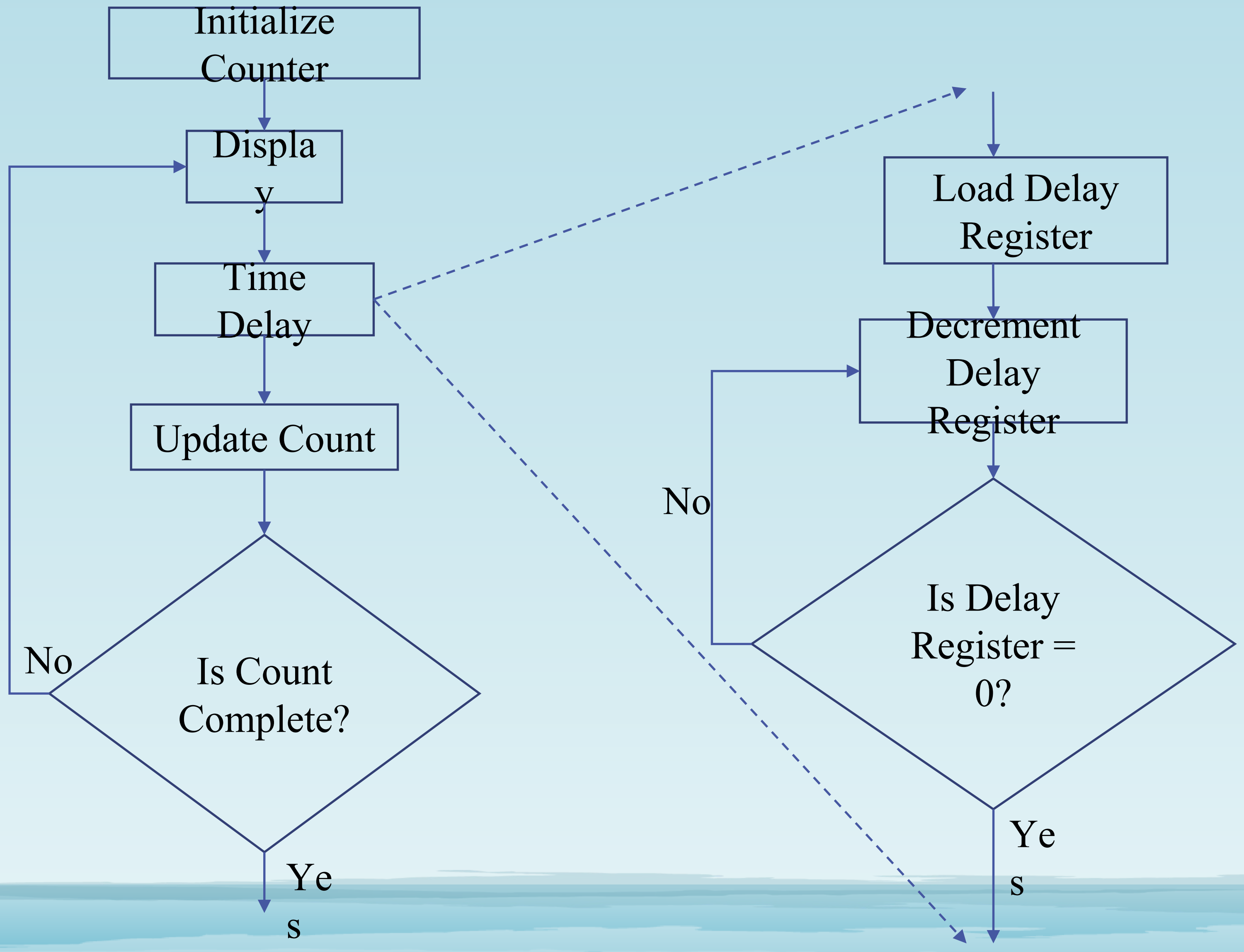
$= 100464\ \mu s$

$$\mathbf{T_{L2} = 100.46\ ms}$$

Disadvantage of using software delay

- Accuracy of time delay depends on the accuracy of system clock.
- Microprocessor is occupied simply in a waiting loop; otherwise, it could be employed to perform other functions.
- The task of calculating accurate time delays is tedious.
- In real time applications timers are commonly used.
- Intel 8254 is a programmable timer chip, that can be interfaced with microprocessor to provide timing accuracy.
- The disadvantage of using hardware chip include the additional expense and the need for extra chip in the system.

Counter design with time delay



Hexadecimal counter program

Write a program to count continuously in hexadecimal from **FFh** to **00h** with **0.5 µs** clock period. Use **register C** to set up **1ms** delay between each count and display the number at one of the output port.

Given:

Counter= FF h

Clock Period T=0.5 µs

Total Delay = 1ms

Output:

To find value of delay counter

Hexadecimal counter program

Program

```
1.  MVI B,FF
2.  LOOP:MOV A,B
3.          OUT 01
4.      MVI C, COUNT; need to calculate delay count
5.          DELAY: DCR C
6.            JNZ DELAY
7.      DCR B
8.      JNZ LOOP
```

Hexadecimal counter program

Delay Calculations

```
1. MVI B,FF;

2. LOOP:MOV A,B

3.      OUT 01

4.      MVI C, COUNT

5.      DELAY: DCR C

6.             JNZ DELAY

7.      DCR B

8.      JNZ LOOP
```

Instruction	T-States
DCR C	4
JNZ DELAY	10
Total	**14 T**

Calculate Delay for Internal Loop

$$T_I = \text{T-States} * \text{Clock Period} * \text{COUNT}$$

$$= 14 * 0.5 * 10^{-6} * \text{COUNT}$$

$$T_I = (7.0 * 10^{-6}) * \text{COUNT}$$

Hexadecimal counter program

Calculate Delay for Outer Loop:

T_O = T-States * Clock Period

$= 35 * 0.5 * 10^{-6}$

$T_O = 17.5\ \mu s$

Instruction	T-States
MOV A,B	4
OUT 01H	10
MVI C, COUNT	7
DCR B	4
JNZ LOOP	10
TOTAL	35 T

Calculate Total Time Delay:

$T_D = T_O + T_L$

$1\text{ ms} = 17.5 * 10^{-6} + (7.0 * 10^{-6}) * \text{COUNT}$

$1 * 10^{-3} = 17.5 * 10^{-6} + (7.0 * 10^{-6}) * \text{COUNT}$

$$\text{COUNT} = \frac{1 * 10^{-3} - 17.5 * 10^{-6}}{7.0 * 10^{-6}} \cong (140)_{10} = (8C)_{16}$$

0-9 up/down counter program

Write an 8085 assembly language program to generate a decimal counter (which counts 0 to 9 continuously) with a one second delay in between. The counter should reset itself to zero and repeat continuously. Assume a **Clock frequency** of 1MHz.

0-9 up counter program

Program

```
1.    START: MVI B,00H
2.    DISPLAY: OUT 01
3.           LXI H, COUNT
4.    LOOP: DCX H
5.           MOV A,L
6.           ORA H
7.           JNZ LOOP
8.      INR B
9.      MOV A,B
10.     CPI 0A
11.     JNZ DISPLAY
12.   JZ START
```

Instruction	T-States
DCX	6
MOV A,L	4
ORA H	4
JNZ	10
TOTAL	24 T

37

0-9 up counter program

Delay Calculation:

As show in previous program of register pair, assuming

T-states = 24

Loop Delay T_L = 1 sec

Clock Period T =1 * 10^{-6} sec

Find Count

$$\mathbf{T_L = T * Loop\ T\text{-}states * Count}$$

$$1\ \text{sec} = (1.0 * 10^{-6}\ \text{sec}) * 24 * \mathbf{Count}$$

$$\mathbf{Count} = \frac{1}{24 * 10^{-6}}$$

$$= (41666)_{10}$$

Exercise

Calculate delay in following loop, assuming clock period = 0.33µs

Label	**Instruction**	**T-states**
	LXI B, 12FF H	10
DELAY:	DCX B	6
	XTHL	16
	XTHL	16
	NOP	4
	NOP	4
	MOV A,C	4
	ORA B	4
	JNZ DELAY	10/7

TL= T * T-states * Count

Ans=102ms

Exercise

Write a set of 8085 assembly language instructions to generate a 1 second delay, if the **crystal frequency** is 6 MHz.

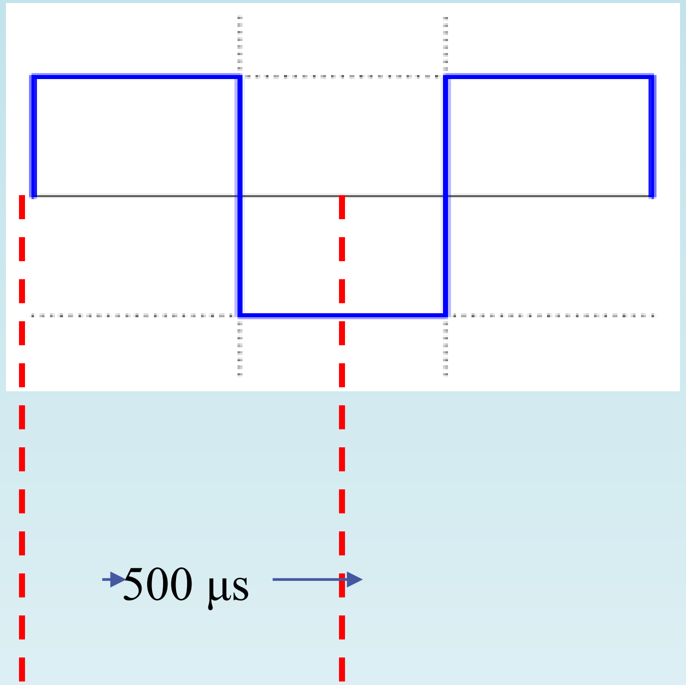
Note:

Clock Frequency(Operating Frequency)=Crystal Frequency/2

Clock Period = 1/Clock Frequency

Square wave program

Write a program to generate a continuous square wave with the period of 500 µs with clock period of 325ns, and use bit D_0 to output square wave.



Square wave program

Problem Analysis

- In this problem, the period of square wave is 500 μs; therefore, the pulse should be ON(logic 1) for 250 μs and OFF(logic 0) for remaining 250 μs.
- Therefore, the alternate pattern of 0/1 bits can be provided by loading Accumulator with AA h (1010 1010).
- Now rotating the pattern once through each delay loop.
- Bit D_0 of output port is used to provide logic 0 and 1.
- The delay of 250μs can be easily obtained with an 8-bit delay count and one register.

Square wave program

Program

```
1. MVI D, AA
2. ROTATE:   MOV A,D
3.          RLC
4.           MOV D,A
5.           ANI 01
6.           OUT 01
7.           MVI B, COUNT
8. DELAY:    DCR B
9.           JNZ DELAY
10.          JMP ROTATE
```

Logic 1

(A)	1 0 1 0 1 0 1 0
After RLC	0 1 0 1 0 1 0 1
ANI 01h	0 0 0 0 0 0 0 1
After AND	**0 0 0 0 0 0 0 1**

Logic 0

(A)	0 1 0 1 0 1 0 1
After RLC	1 0 1 0 1 0 1 0
ANI 01h	0 0 0 0 0 0 0 1
After AND	**0 0 0 0 0 0 0 0**

Square wave program

Delay Calculation:

- In this problem, the pulse width is relatively small (250 μs); therefore, to obtain accurate output pulse, we should take into account for all the T-states.
- The total delay should include the delay in the loop and execution time of the instruction outside the loop.

Square wave program

Delay Calculation:

- No. of instruction outside the loop is seven.

  Delay outside the loop $\mathbf{T_O}$=46 T-states * 325 ns =**14.95µs**

- Delay Loop includes two instruction, with 14 T-States, except for last cycle 11 T-States

  Loop Delay T_L= 14 T * 325ns * (**COUNT** -1) + [11 T *325ns]

  $$\mathbf{T_L = 4.5\ \mu s\ (COUNT - 1) + 3.575\ \mu s}$$

- **Total delay required = 250** µs

  $$\mathbf{T_D = T_O + T_L}$$

  250 µs = 14.95µs + 4.5 µs (**COUNT** -1) + 3.575 µs

  $$\mathbf{Count = (52)_{10}}$$

  $$\mathbf{= (34)_{16}}$$

Exercise

Write a program to generate a square wave with the period of 400µs with clock period of 325ns. Use bit D_0 to output square wave.

ANS = $(42)_{10}$

To convert a given decimal number to hexadecimal

```
LXI H,4300H
MOV A,M
MOV B,A
ANI 0FH
MOV C,A
MOV A,B
RRC
RRC
RRC
RRC
ANI 0FH
MOV B,A
XRA A
CMP B
JZ LAST
BACK:ADI 0AH
DCR B
JNZ BACK
LAST:ADD C
INX H
MOV M,A
HLT
```

To convert a given HEXADECIMAL TO DECIMAL.

```
LXI H, 4150H          ; Point to data
LXI B, 0000H          ; Initialize hundreds= 0, Tens=0
MOV A, M              ; Get hex data to A
LOOP: SUI 64H
    JC LOOP1
     INR B                ; hundreds= hundreds+1
     JMP LOOP
LOOP1: ADI 64H           ; if subtracted extra, add it clear carry flag
LOOP2: SUI 0AH
        JC LOOP3
        INR C             ; Tens=tens+1
        JMP LOOP2
LOOP3: ADI 0AH           ; If subtracted extra, add it again
       INX H              ; A = Units
       MOV M, B           ; store hundreds
       MOV B, A           ; Combine Tens in C &
       MOV A, C           ; Units in A to form a
       RLC                ; Single 8-bit number
       RLC
       RLC
       RLC
       ADD B
       INX H
       MOV M, A           ; Store tens & Units
       HLT
```

Note: In this experiment the number is converted to its equivalent decimal number using the following logic. First count the number of hundreds, the number of tens & units present in that hex number. Then add up to get the equivalent decimal number.

Converting A9 we get:

A9 /64=45   Hundreds = 01

Since 64(100 decimal) cannot be subtracted from 45 no. of hundreds = 01. Now count tens

45/0A=3B   Tens = 01

Now from 09, 0A cannot be subtracted. Hence tens = 06 the decimal equivalent of A9 is

PROGRAM 1: Convert BCD TO BINARY OR BCD TO HEX.

Note: Use principal of positional weighing in given no.

For example: 60=6 x 0A+00   72=7x0A+02

=3C+00   =48

```
LDA D000H
MOV B,A
ANI 0FH
MOV C,A
MOV A,B
ANI F0H
RRC
RRC
RRC
RRC
MOV B,A
XRA A
MVI D,0AH
L1:ADD D
DCR B
JNZ L1
ADD C
STA D100H
HLT
```

Binary to BCD conversion or HEX to BCD CONVERSION

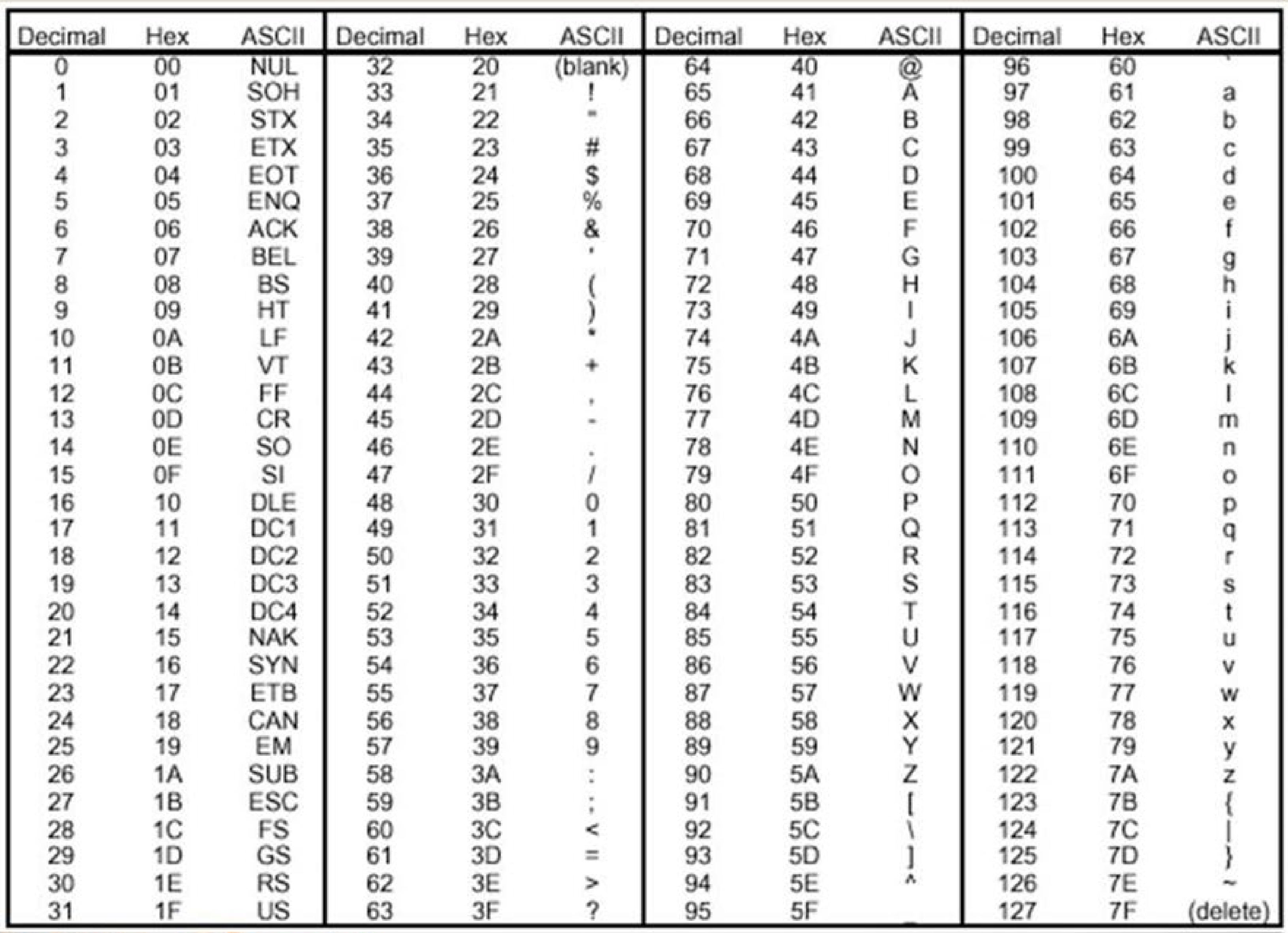
Note: Binary to BCD is done by dividing no. by power often.

Step-1if no. is equal to or greater than 100 ,divide the no. By 100(i.e. subtract 100 repeatedly till remainder is less than 100) the quotient gives the MSB ,DIGIT 2 of BCD no. if no.<100 follow step.2

STEP -2 IF NO. i.e. remainder of 1st division is equal to or greater than 10 divide no. by 10 repeatedly until remainder is less than 10 quotient , the digit 1, if no. is less than 10, go to step 3.

Step-3 Remainder it gives digit 3

```
LDA 3040H
MVI B,64H
MVI C,0AH
MVI D,00H
MVI E,00H
L1:CMP B
   JC L2
   SUB B
   INR E
   JMP L1
L2:CMP C
   JC L3
   SUB C
   INR D
   JMP L2
L3: STA 3041H
    MOV A,D
    STA 3042H
    MOV A,E
    STA 3043H
    HLT
```



Decimal	Hex	ASCII	Decimal	Hex	ASCII	Decimal	Hex	ASCII	Decimal	Hex	ASCII
0	00	NUL	32	20	(blank)	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	(delete)

ASCII Chart

HEX or Binary to ASCII HEX Code

```
LXI H,0050H
LXI D,0052H
MOV A,M
MOV B,A
RRC
RRC
RRC
RRC
CALL ASCII
STAX D
INX D
MOV A,B
CALL ASCII
STAX D
HLT

ASCII: ANI 0FH
      CPI 0AH
      JC CODE
      ADI 07H
CODE: ADI 30H
RET
```

Write 8085 Assembly language program to convert ASCII to Hexadecimal character values.

```
LXI H,8000H
MOV A, M
CPI 58H
JNC NUM
SUI 37H
JMP STORE
NUM:SUI 30H
STORE:INX H
MOV M, A
HLT
```

To find the square of the number from 0 to 9 using a Table of Square.

ALGORITHM:

1. Initialize HL pair to point Look up table
2. Get the data .
3. Check whether the given input is less than 9.
4. If yes go to next step else halt the program
5. Add the desired address with the accumulator content
6. Store the result

PROGRAM:

```
                LXI     H,4125              Initialsie Look up table address
                LDA     4150                Get the data
                CPI     0A                  Check input > 9
                JC      AFTER               if yes error
                MVI     A,FF                Error Indication
                STA     4151
                HLT
AFTER:          MOV     C,A     Add the desired Address
                MVI     B,00
                DAD     B
                MOV     A,M
                STA     4151    Store the result
                HLT             Terminate the program
```

LOOKUP TABLE:

4125	01
4126	04
4127	09
4128	16
4129	25
4130	36
4131	49
4132	64
4133	81

OBSERVATION:

Input:	4150:	05
Output:	4151	25 (Square)
Input :	4150:	11

Add two 8 bit BCD numbers stored at consecutive memory locations.

```
START: MVI       C, 00H
         LXI     H, 4500H
         MOV     A, M
         INX     H
         ADD     M
         DAA
         JNC     L1
         INR     C
L1:INX   H
        MOV      M, A
        INX      H
        MOV      M, C
        HLT
```