

**8085**

**PROGRAMS**

## Experiment - 1

### REARRANGING BYTES:

16 bytes are residing in locations starting from 0C20H / 2400H. Write a program to transfer these bytes in locations starting from 0C40H / 2500H in such a way that first 8 bytes in the first block should appear at last 8 locations of the second block and the last 8 bytes in the first block at first 8 locations of the second block .

### CODE:

8000	— 11	LXI D, 8700H	3
8001	— 00		
8002	— 84		
8003	— 01	LXI B, 8508H	3
8004	— 08		
8005	— 85		
8006	— 26	MVI H, 08	2
8007	— 08		1
8008	— 1A	LDAX D	1
8009	— 02	STAX B	1
800A	— 13	INX D	1
800B	— 03	INX B	1
800C	— 25	DCR H	1
800D	— C2	JNZ 8008	3
800E	— 08		
800F	— 80		
8010	— 11	LXI D	8708
8011	— 08		3
8012	— 84		
8013	— 01	LXI B, 8500	3
8014	— 00		
8015	— 85		

8016 — 26 } MVI H @8 2  
 8017 — 08  
 8018 — 1A LDAX D /  
 8019 — 02 STAX B /  
 801A — 1B INX D /  
 801B — 03 INXB /  
 801C — 25 DCRH 1  
 801D — C2 } JNZ 8018 3  
 801E — 18  
 801F — 80  
 8020 — 76 HLT 1

I/Pt

8400 - 00  
 8401 - 01  
 8402 - 02  
 8403 - 03  
 8404 - 04  
 8405 - 05  
 8406 - 06  
 8407 - 07  
 8408 - 08  
 8409 - 09  
 840A - 0A  
 840B - 0B  
 840C - 0C  
 840D - 0D  
 840E - 0E  
 840F - 0F

16bit

*last 8 bytes*

O/Pt

8500 - 08  
 8501 - 09  
 8502 - 0A  
 8503 - 0B  
 8504 - 0C  
 8505 - 0D  
 8506 - 0E  
 8507 - 0F

8508 - 00  
 8509 - 01  
 850A - 02  
 850B - 03  
 850C - 04  
 850D - 05  
 850E - 06  
 850F - 07

## Experiment - 2

### CHECKING BITS OF A WORD:

A word is residing in location 0C40H / 2400H. Write a program to check each bit of the word starting from Ms bit and fill 16 locations starting from 0C20H / 2500H, with either 00H or FFH depending on the bit, FFH if bit is '1' and 00H if the bit is '0'. Also count the no. of 1's and 0's (count in BCD) in the word and store them respectively at 0C30H / 2420H and 0C31H / 2421H

### CODE:

8000 — AF      XRA A      I      XOR with Atau  
[To reset the flag  
[if any one is set]  
before]

8001 — 32 }      STA 8420      3

8002 — 20 }  
8003 — 84 }

8004 — 32 }      STA 8421      3

8005 — 21 }  
8006 — 84 }

8007 — 21 }      LXI H 8400      3

8008 — 00 }  
8009 — 84 }

800A — 11 }      LXI D, 16bit  
[8500]      3

800B — 00 }  
800C — 85 }

800D — 0E }      MVI C 02      2

800E — 0R }  
800F — 06 }      MVI B 07      2

8010 — 08 }

8011 — 7E      MOV A,M      I

8012 — 07      RLC      I

8013 — 77      MOV M,A      I

8014 — D2 }      JNC 8026      3

8015 — 26 }  
8016 — 80 }

8017 — 3E } MVI A (FF) 2  
8018 — FF }

8019 — 12 STAX D 1

801A — 3A } LDA (8420) 3  
801B — 20 }  
801C — 84 }

801D — C6 } ADI <sup>8421</sup> (01) 2  
801E — 01 }

801F — 27 DAA 1

8020 — 32 } STA 8420 3  
8021 — 20 }  
8022 — 84 }

8023 — C3 } JMP 8034 3  
8024 — 34 }  
8025 — 80 }

8026 — 3E } MVI A 2  
8027 — 00 }

8028 — 12 STAX D 1

8029 — 3A } LDA 8421 3  
802A — 21 }  
802B — 84 }

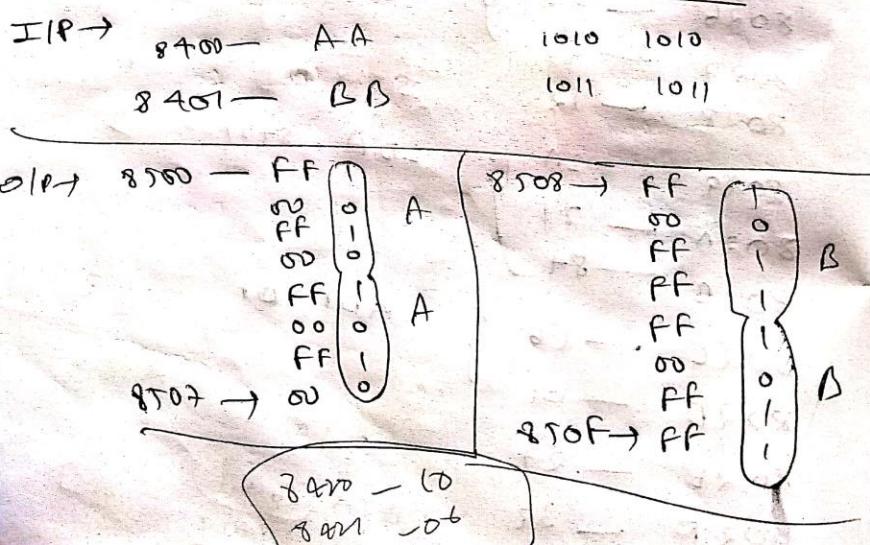
802C — C6 } ADI <sup>8421</sup> (01) 2  
802D — 01 }

802E — 27 DAA 1

802F — 32 STA (16 bit) 8421 3  
80

(7) A NM

8030 — 21		
8031 — 84		
8032 — C3	JMP 8035	3
8033 — 35		
8034 — 80	ADD B	
8035 — 13	INX D	1
8036 — 05	DCR B	1
8037 — C2	JNZ 8011	
8038 — 41		
8039 — 70		
 803A — 23	INX H	1
 803B — 0D	DCR C	1
803C — C2	JNZ 800F	3
803D — 0F		
803E — 80		
 803F — 71	HLT	1



## Experiment-3

### FILLING UP 128 LOCATIONS:

Write a program to fill up 128 locations starting from 0C00H / 2400H with bytes in the following pattern: first 64 locations to be filled up as 00H, 11H, 22H, -----, FFH, 00H, 11H, 22H, ----- and so on and the last 64 locations to be filled up as FFH, EEH, DDH, -----, 00H, FFH, EEH, DDH and so on....

### CODE :

8000	—21	LXI H, 8400	3
8001	—00		
8002	—84		
8003	—0E	MVI C, 8bit	2
8004	—07		
8005	—06		
8006	—10	MVI B, 8bit	2
8007	—3E		
8008	—00		
8009	—77	MOV M, A	1
800A	—C6	ADI 8bit	2
800B	—11		
800C	—23	INX H	1
800D	—05	DCR B	1
800E	—C2	JNZ 16bit 8009	3
800F	—09		
8010	—80		
8011	—0D	DCA C	1
8012	—C2	JNZ 8005	3
8013	—05		
8014	—80		
8015	—0E	MVI C, 8bit	2
8016	—07		
8017	—06		
8018	—10	MVI B, 8bit	2
8019	—3E		
801A	—FF		
801B	—77	MOV M, A	1

801C	— DG	}	SUI Data (J1)	2
801D	— 11			
801E	— 23		INX H	I
801F	— 05		DCR B	1
8020	— C2	}	JNZ 801B	3
8021	— 1B			
8022	— 80			
8023	— 0D		DCR C	I
8024	— C2	}	JNZ 8017	3
8025	— 17			
8026	— 80			
8027	— 76		HLT	

<del>DIPS</del>	O/P →	8400 — 00	16
		! ! 22	
		! !	
	—	840A — AA	16
		! !	
		840F — FF	
	—	8410 — 00	16
		! !	
		841F — FF	
	—	8420 — 00	16
		! !	
		842F — FF	
	—	8450 — 00	16
		! !	
		843F — FF	
	—	8440 — FF	64
		! !	
		848F — 00	

## EXPERIMENT-4

### **ADDITION OF 12 BYTES:**

Write a program to add 12 bytes residing in locations starting from 0C20H / 2400H, and store the sum in location 0C30H / 2500H (result space 2 bytes)

### **CODE:**

8000	— 21	LXI H, 8-150	3
8001	— 00		
8002	— 84		
8003	— 3E	MVI A, 8bit	2
8004	— 00		
8005	— 06		
8006	— 0C	MVI B, 8bit	2
8007	— 0E		
8008	— 00		
8009	— 86	ADD, M	1
800A	— D2		
800B	— 0E		
800C	— 80	JNC <del>1600H</del> 800E	3
800D	— 0C		
800E	— 23		
800F	— 05	INR C INX H DCR B JNZ ,8009	1
8010	— C2		
8011	— 09		
8012	— 80	STA 8500	3
8013	— 32		
8014	— 00		
8015	— 85	MOV A,C STA , 8501	1
8016	— 79		
8017	— 32		
8018	— 01	STA , 8501	3
8019	— 85		
801A	— 76		
		M ET.	1
<hr/> <i>I(P :- 8400 - FF ; 8401 - FF ; 8402 - FF)</i>		<i>O/P :-</i>	
		<i>8500 - FF</i>	
		<i>8501 - 08</i>	

## EXPERIMENT-5

### SORTING IN DESCENDING ORDER

Write a program to sort 16 bytes residing in locations starting from 0C20H / 2400H, in descending order in the same location .

#### CODE:

CODE:  
MVI C, 10H  
DCR C  
REPEAT: MOV D,C  
LXI H, 4200H  
LOOP: MOV A,M  
INX H  
CMP M  
JNC SKIP  
MOV B,M  
MOV M,A  
DCX H  
MOV M,B  
INX H  
SKIP: DCR D  
JNZ LOOP  
DCR C  
JNZ REPEAT  
HLT

## INPUT :

Memory View																↻	0x	4200
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
420	00	00	0F	11	22	22	33	34	34	56	76	88	99	AB	FF	FF		
421	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
422	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
423	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
424	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
425	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
426	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
427	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
428	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
429	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

Start Address at: 0x 3300 ▾

## OUTPUT :

Memory View																↻	0x	4200
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
420	FF	FF	AB	99	88	76	56	34	34	33	22	22	11	0F	00	00		
421	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
422	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
423	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
424	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
425	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
426	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
427	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
428	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
429	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
42F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

Start Address at: 0x 3300 ▾

## Experiment -6

### MULTIPLICATION OF SINGLE BYTE BY SINGLE BYTE:

Write a program to multiply a single byte residing in location 0C20H / 2400H by another byte residing in location 0C21H / 2401H and store the product in 0C22H / 2402H (Product space 2 bytes)

#### CODE:

```
CODE:  
    LXI SP 23FFH  
    LXI H, 2400H  
    MVI D, OOH  
    MOV A, M  
    ADI OOH  
    JZ EXIT  
    MOV B,M  
    INX H  
    MOV A,M  
    ADI OOH  
    JZ EXIT  
    MOV C,M  
    MVI A,OOH  
LOOP: ADD B  
    JNC NEXT  
    INR D  
NEXT: DCRC  
    JNZ LOOP  
EXIT: LXIH, 2402H  
    MOV M,A  
    INX H  
    MOV M,D  
    HLT
```

## **INPUT/OUTPUT:**

## Experiment - 7

### SORTING EVEN AND ODD PARITY BYTES:

Sixteen bytes reside in locations starting from 0C20H / 2400H. Write a program to sort them in the same locations according to parity, odd parity bytes first and then even parity bytes next. Keep the count (in BCD) of odd parity and even parity bytes at 0C30H / 2420H and 0C31H / 2421H respectively

### CODE:

CODE :

```
LXI SP, 23FFH
LXI D, 2450H
PUSH D
MVI D, 00H
MVI E, 00H
LXI H, 2400H
MVI B, 10H
LOOP1: MOV A,M
ADI 00H
JPE NEXT1
XTHL
MOV M,A
INX H
XTHL
MOV A,D
ADI 01H
DAA
MOV D,A
NEXT1: INX H
DCR B
JNZ LOOP1
LXI H, 2400H
MVI B, 10H
LOOP2: MOV A,M
ADI 00H
JPO NEXT2
XTHL
```

MOV M,A  
INX H  
XT HL  
MOV A,E  
ADI 01H  
DAA  
MOV E,A  
NEXT2: INX H  
DCR B  
JNZ LOOP2  
LXI H ,2420H  
MOV M,D  
INX H  
MOV M,E  
LXI H ,2400H  
LXI D, 2450H  
MVI B, 10H  
LOOP3: LDAX D  
MOV M,A  
INX D  
INX H  
DCR B  
JNZ LOOP3  
HLT

## INPUT/OUTPUT :

Memory View																↻	0x	2400
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
240	01	02	04	07	08	0B	0D	0E	03	05	06	09	0A	0C	0F	00		
241	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
242	2C	38	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
243	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
244	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
245	01	02	04	07	08	0B	0D	0E	03	05	06	09	0A	0C	0F	00		
246	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
247	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
248	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
249	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

Start Address at: 0x

## Experiment - 8

### **BCD TO BINARY CONVERSION:**

Write a program to convert a BCD number residing at 0C20H / 2400H to its corresponding binary number and store it at 0C21H / 2401H

### **CODE:**

CODE:

```
START: LXI SP, 2099H
        LXI H, 2400H
        LXI B, 2401H
        MOV A, M
        PUSH B
        MOV B,A
        ANI 0F H
        MOV C,A
        MOV A,B
        ANI F0H
J2    BCD1
        RRC
        RRC
        RRC
        RRC
        MOV D,A
        XRA A
        MVI E,0AH
SUM: ADD E
        DCR D
        JNZ SUM
BCD1: ADD C
        POP B
        STAX B
        HLT
```

## INPUT/OUTPUT:

Memory View  0x 2400

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
240	99	63	00	00	00	00	00	00	00	00	00	00	00	00	00	00
241	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
242	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
243	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
244	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
245	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
246	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
247	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
248	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
249	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Start Address at: 0x  

## Experiment - 9

### **NUMBER OF BITS IN BYTES:**

A block of 16 bytes resides in locations starting from 0C20H / 2400H. Write a program to form a second block of size 16 with the no. of 1's in the corresponding byte of the first block, at locations starting from 0C30H / 2420H. (For e.g. the first entry in the second block will be the no. of 1's in the first entry of the first block.)

### **CODE:**

CODE:

```
LXI SP, 23FFH
LXI H, 2400H
LXI D, 2420H
PUSH D
MOVJ B, 10H
LOOP1: MVI D, 00H
MVIC C, 08H
MOV A, M
LOOP2: MOV E, A
ANI 01H
JZ NEXT
INR D
NEXT: MOV A, E
RAR
DCR C
JNZ LOOP2
XTHL
MOV M, D
INX H
XTHL
INX H
DCR B
JNZ LOOP1
HLT
```

## INPUT/OUTPUT:

## Memory View



0x

2400

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
240	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
241	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
242	00	01	01	02	01	02	02	03	01	02	02	03	02	03	03	04
243	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
244	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
245	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
246	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
247	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
248	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
249	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Start Address at: 0x 

## Experiment -10

### NUMBER OF TIMES FFH OCCURS AS PAIR:

A block starts at 0C20H / 2400H and ends at 0C2FH / 240FH. Write a program to count the number of times FFH occur as memory pairs and keep the count (BCD) in location 0C30H / 2410H. If four consecutive locations are filled up with FFH the count shall be 2 and so on

#### CODE:

CODE:

```
LXI H, 2400H
MVI B, 0FH
MVI C, 00H
LOOP: MOV A,M
      CPI  FFH
      JNZ NEXT
      INX H
      MOV A,M
      CPI  FFH
      JNZ NEXT
      MOV A,C
      ADI  01H
      DAA
      MOV C,A
      DCR B
NEXT: INX H
      DCR B
      JNZ LOOP
LXI H, 2420H
MOV M,C
HLT
```

## INPUT/OUTPUT :

Memory View																	0x	2400
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
240	FF	00	FF	FF	23	12	11	FF	FF	56	78	FF	FF	12	34	69		
241	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
242	03	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
243	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
244	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
245	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
246	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
247	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
248	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
249	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
24F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

Start Address at: 0x



## EXPERIMENT-11

### **CONSECUTIVE MEMORY LOCATIONS WITH IDENTICAL DATA:**

A block starts at 0C20H / 2400H and ends at 0C3FH / 241FH. Write a program to determine the number of consecutive locations with identical data and store the count (in BCD) in location 0C40H / 2420H. If three consecutive locations have identical data, the count shall be 2 and so on

#### **CODE:**

CODE:

```
LXI H,2400H
MVI B,1FH
MVI C,00H
LOOP: MOV D,M
      INX H
      MOV A,M
      CMP D
      JNZ NEXT
      MOV A,C
      ADJ 01H
      DAA
      MOV C,A
NEXT:  DCRB
      JNZ LOOP
      LXI H,2420H
      MOV M,C
      HLT
```

**INPUT/OUTPUT:**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
240	FF	12	12	FF	99	FF	23	23	00	00	99	99	12	0A	AB	CD
241	AA	AA	34	69	90	45	22	22	11	23	56	65	34	34	12	12
242	08	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
243	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
244	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
245	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
246	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
247	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
248	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
249	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Start Address at: 0x  

## **EXPERIMENT-12**

### **FORMATION OF A THIRD BLOCK:**

Two blocks of 10 bytes each are residing in locations 0C20H / 2400H and 0C30H / 2410H respectively, Write a program to form a third block of 10 bytes at locations starting from 0C40H / 2420H, such that each entry in that block is the one's complement of the larger of the corresponding entries of the first two blocks (the  $n^{\text{th}}$  entry in the third block shall be the one's complement of the larger of the  $n^{\text{th}}$  entry of the first block and the  $n^{\text{th}}$  entry of the second block)

### **CODE:**

CODE:

```
LXI SP, 23FFH
LXI H, 2420H
PUSH H
LXI H, 2400H
LXI D, 2410H
MVI B, 0AH
LOOP: LDAXD
      CMP M
      JNC NEXT1
      MOV A,M
NEXT1: CMA
      XTHL
      MOV M,A
      INX H
      XTHL
      INX H
      INX D
      DCR B
      JNZ LOOP
      HLT
```

**INPUT/OUTPUT:**

## Memory View



0x

2400

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
240	FF	12	12	FF	99	FF	23	23	45	54	00	00	00	00	00	00
241	AA	AA	34	69	90	45	22	22	11	23	00	00	00	00	00	00
242	00	55	CB	00	66	00	DC	DC	BA	AB	00	00	00	00	00	00
243	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
244	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
245	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
246	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
247	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
248	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
249	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Start Address at: 0x 2200 ▾



## Experiment -13

### **INSERTION TO A LIST:**

Write a program to insert the content at the location 0C20H / 2400H to a list which begins at 0C22H / 2402H, if not already present in the list and only if byte at location 0C20H / 2400H is divisible by two. The number of bytes in the list is given at location 0C21H / 2401H (in BCD). If the insertion is valid, insert the byte at the end of the list and modify the count accordingly

### **CODE:**

```
CODE: LXI H, 2401H
      MOV A,M
      CALL BCD
      CPI 00H
      JZ EXIT
      MOV B,A
      LDA 2400H
      RRC
      JC EXIT
      RLC
      LOOP: INX H
      CMP M
      JZ EXIT
      DCR B
      JNZ LOOP
      INX H
      MOV M,A
      LXI H , 2401H
      MOV A,M
      ADI 01H
      DAA
      MOV M,A
      EXIT : HLT
```

## Experiment -13

### **INSERTION TO A LIST:**

Write a program to insert the content at the location 0C20H / 2400H to a list which begins at 0C22H / 2402H, if not already present in the list and only if byte at location 0C20H / 2400H is divisible by two. The number of bytes in the list is given at location 0C21H / 2401H (in BCD). If the insertion is valid, insert the byte at the end of the list and modify the count accordingly

### **CODE:**

```
CODE: LXI H, 2401H
      MOV A,M
      CALL BCD
      CPI 00H
      JZ EXIT
      MOV B,A
      LDA 2400H
      RRC
      JC EXIT
      RLC
      LOOP: INX H
      CMP M
      JZ EXIT
      DCR B
      JNZ LOOP
      INX H
      MOV M,A
      LXI H , 2401H
      MOV A,M
      ADI 01H
      DAA
      MOV M,A
      EXIT : HLT
```

BCD: ANI OFH

MOV B,A

MOV A,M

RRC

RRC

RRC

RRC

ANI OFH

MOV C,A

MOV A,B

INR C

MYLOOP: DCRC

JZ EXIT1

ADI OAH

JMP MYLOOP

EXIT1: RET

**INPUT:**

Memory View																	0x	2400
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
240	10	09	01	02	03	FF	23	56	77	AB	CC	00	00	00	00	00	00	00
241	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
242	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
243	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
244	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
245	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
246	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
247	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
248	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
249	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Start Address at: 0x

## OUTPUT:

Memory View																	0x	2400
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
240	10	10	01	02	03	FF	23	56	77	AB	CC	10	00	00	00	00	00	00
241	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
242	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
243	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
244	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
245	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
246	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
247	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
248	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
249	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
24F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Start Address at: 0x

# 8086

## PROGRAMS

### Experiment – 14

#### CHECKING BITS OF A WORD:

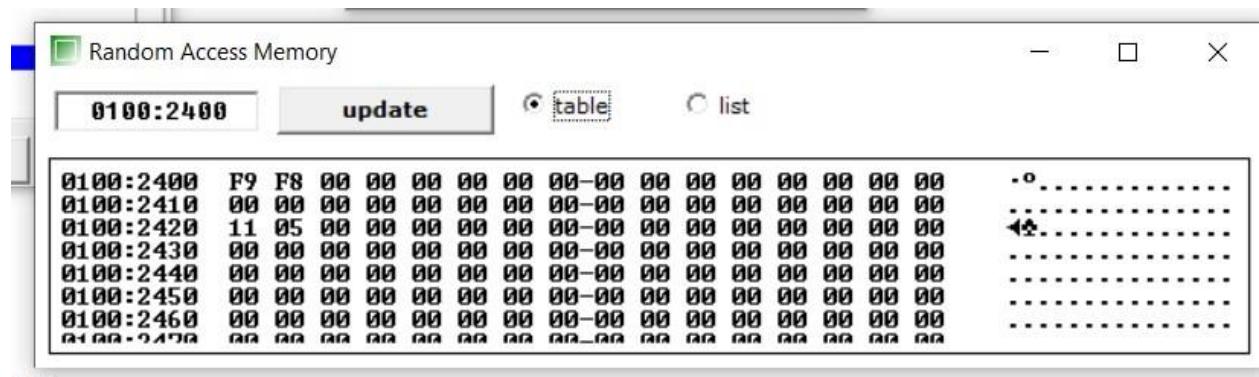
A word is residing in location 0C40H / 2400H .Write a program to check each bit of the word starting from Ms bit and fill 16locations starting from 0C20H / 2500H, with either 00H or FFH depending on the bit, FFH if bit is '1' and 00H if the is '0'. Also count the no. of 1's and 0's(count in BCD) in the word and store them respectively at 0C30H / 2420H and 0C31H / 2421H.

**SOFTWARE USED:** EMU8086 Simulator

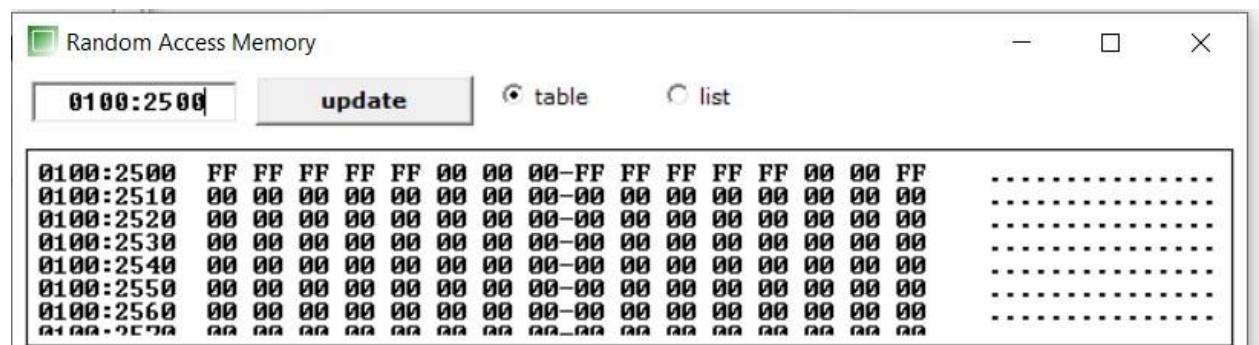
#### CODE:

```
CODE:  
    MOV SI, 2500H  
    MOV BL, [2400H]  
    MOV BH, [2401H]  
    MOV CL, 10H  
    MOV AL, 00H  
    MOV AH, 00H  
BACK: ROL BX, 01H  
    JC NEXT  
    MOV [SI], 00H  
    INC SI  
    ADD AL, 01H  
    DAA  
    DEC CL  
    JNZ BACK  
    JMP FINAL  
NEXT: MOV [SI], 00FFH  
    INC SI  
    XCHG AL, AH  
    ADD AL, 01H  
    DAA  
    XCHG AL, AH  
    DEC CL  
    JNZ BACK  
FINAL: MOV [2420H], AH  
      MOV [2421H], AL  
      HLT
```

**INPUT:**



## **OUTPUT :**



## EXPERIMENT - 15

### SORTING IN DESCENDING ORDER:

Write a program to sort 16 bytes residing in locations starting from 0C20H / 2400H, in descending order in the same location.

SOFTWARE USED: EMU8086 Simulator

CODE:

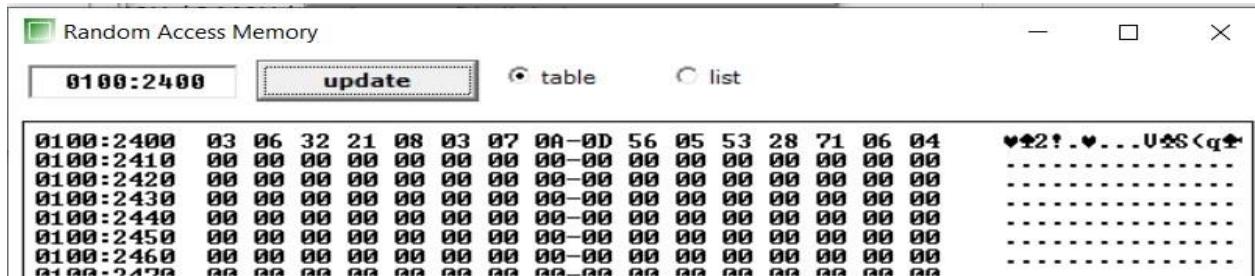
CODE:

```
START: MOV SI, 2400H
        MOV BH, 10H
        MOV CL, 0DH

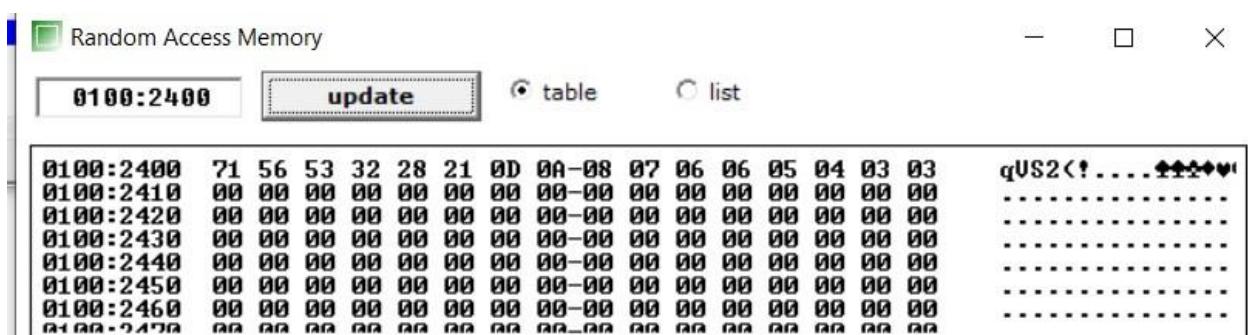
CHECK: MOV AL, [SI]
        INC SI
        MOV BL, [SI]
        CMP AL, BL
        JNC NEXTBYTE
        JZ NEXT BYTE
        XCHG AL, [SI]
        DEC SI
        MOV [SI], AL
        INC SI
        MOV CL, 01H

NEXTBYTE: DEC BH
        JNZ CHECK
        CHP CL, 01H
        JZ START
        HLT
```

## **INPUT:**



### **OUTPUT :**



## EXPERIMENT NO : 16

### ADDITION OF TWO 20-DIGIT BCD NO's:

Write a program to add a 20-digit BCD number residing in location 0C20H / 2400H, with another 20-digit BCD number residing in location 0C30H / 2420H and store the result at location 0C40H / 2440H (result space 21 digits).

SOFTWARE USED: EMU8086 Simulator

CODE:

CODE :

```
MOV CL,0AH
MOV SI,2400H
MOV DI, 2420H
MOV BX, 2440H
NEXT: MOV AH,[SI]
       MOV AL, [DI]
       ADC AL, AH
       DAA
       MOV [BX], AL
       INC SI
       INC DI
       INC BX
       DEC CL
       JNZ NEXT
       MOV AL,00H
       ADC AL,00H
       MOV [BX], AL
       HLT
```



## EXPERIMENT - 17

### MULTIPLICATION OF TWO 5-BYTE BINARY NUMBERS:

Write a program to multiply a 5byte binary number residing in location at 0C20H / 2400H by another 5-byte number residing in location at 0C30H / 2410H and store the result at 0C40H / 2440H (result space 10 bytes).

SOFTWARE USED :EMU8086 Simulator

CODE :

CODE:

```
MOV SI, 2400H  
MOV DI, 2410H  
MOV BP, 2440H  
MOV BL, 05H  
NEXT: MOV AL,[SI]  
      MOV BH,[DI]  
      MUL BH  
      MOV [BP],AL  
      INC BP  
      MOV [BP],AH  
      INC BP  
      INC SI  
      INC DI  
      DEC BL  
      JNZ NEXT  
      HLT
```

## INPUT AND OUTPUT:

The screenshot shows a window titled "Random Access Memory". At the top, there are three buttons: "0100:2400" (highlighted in yellow), "update" (highlighted in red), and two radio buttons for "table" and "list" modes. The main area displays a memory dump starting at address 0100:2400. The data is presented in a grid format with columns for address and data bytes. The first few rows of data are as follows:

Address	0100:2400	0100:2410	0100:2420	0100:2430	0100:2440	0100:2450	0100:2460	0100:2470
Data	03 01 02 01 02 00 00 00-00 00 00 00 00 00 00 00 00	89 A6 D2 F2 F8 00 00 00-00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00	9B 01 A6 00 A4 01 F2 00-F0 01 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00

## EXPERIMENT - 18

### **SIZE OF A BLOCK ENDING WITH A SPECIFIED BYTE:**

Write a program to estimate the size of a block which starts at 0C21H / 2401H and ends up with a data specified at 0C20H / 2400H. Keep the count (in BCD) at 0C90H / 2490H. The specified byte is also a part of the block. If the specified byte is not encountered within 99 numbers, put EEH at 0C90H / 2490H.

**SOFTWARE USED: EMU8086 Simulator**

**CODE:**

```
CODE:  
MOV SI, 2400H  
MOV CL, 63H  
MOV CH, [SI]  
MOV AL, 00H  
INC SI  
JUMP: MOV AH, [SI]  
CMP AH, CH  
JZ END  
ADD AL, 01H  
DAA  
INC SI  
DEC CL  
JNZ JUMP  
MOV [2490H], 00EEH  
HLT  
END: ADD AL, 01H  
DAA  
MOV [2490H], AL  
HLT
```

**INPUT:**

A screenshot of a computer application window titled "Random Access Memory". The window has a toolbar with a green square icon, a title bar with standard window controls, and a main area containing memory dump data. The data is presented in a table format with two columns: address and value. The address column shows values from 0100:2400 to 0100:2470. The value column shows binary data (e.g., 01 25 32 05 21 39 43 42-45 38 01 00 00 00 00 00) followed by a hex dump (e.g., @:2419CBE8@....). A radio button labeled "table" is selected.

0100:2400	01 25 32 05 21 39 43 42-45 38 01 00 00 00 00 00	@:2419CBE8@....
0100:2410	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2420	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2430	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2440	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2450	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2460	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2470	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....

**OUTPUT :**

A screenshot of a computer application window titled "Random Access Memory". The window has a toolbar with a green square icon, a title bar with standard window controls, and a main area containing memory dump data. The data is presented in a table format with two columns: address and value. The address column shows values from 0100:2490 to 0100:2500. The value column shows binary data (e.g., 10 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00) followed by a hex dump (e.g., ►.....). A radio button labeled "table" is selected.

0100:2490	10 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	►.....
0100:24A0	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:24B0	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:24C0	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:24D0	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:24E0	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:24F0	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2500	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....

## EXPERIMENT - 19

### FORMATION OF A THIRD BLOCK:

Two blocks of 10 bytes each are residing locations 0C20H / 2400H and 0C30H / 2410H respectively, Write a program to form a third block of 10 bytes at locations starting from 0C40H / 2420H, such that each entry in that block is the one's complement of the larger of the corresponding entries of the first two blocks (the  $n^{\text{th}}$  entry in the third block shall be the  $n^{\text{th}}$  entry of the first block and the  $n^{\text{th}}$  entry of the second block).

**SOFTWARE USED:** EMU8086 Simulator

**CODE:**

Code:-

```
MOV SI, 2400H
MOV DI, 2410H
MOV BX, 2420H
MOV CL, 0AH
LOOP: MOV AL, [SI]
       MOV AH, [DI]
       CMP AL, AH
       JNC JUMP
       NOT AH
       MOV [BX], AH
       INC SI BX
       INC SI
       INC DI
       DEC CL
       JNZ LOOP
       HLT
JUMP: NOT AL
       MOV [BX], AL
       INC BX
       INC SI
       INC DI
       DEC CL
       JNZ LOOP
       HLT
```

## INPUT AND OUTPUT :

The screenshot shows a window titled "Random Access Memory" with the following interface elements:

- Address bar: 0100:2400
- Action button: update
- View mode selector: table (selected)
- View mode selector: list

The main area displays memory contents from address 0100:2400 to 0100:2470. The data is presented in two columns: hex values and ASCII characters.

Address	Hex Data	ASCII Data
0100:2400	68 12 15 08 09 38 12 05-28 21 00 00 00 00 00 00	ht§..8†*†.....
0100:2410	38 32 05 03 19 28 42 25-38 25 00 00 00 00 00 00	82‡▼↓<B%8%.....
0100:2420	97 CD EA F7 E6 C7 BD DA-C7 DA 00 00 00 00 00 00	ù=Ü§ü ¶r  r.....
0100:2430	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2440	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2450	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2460	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....
0100:2470	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	.....

## EXPERIMENT-20

### **INSERTION TO A LIST:**

Write a program to insert the content at the location 0C20H / 2400H to a list which begins at 0C22H / 2402H, if not already present in the list and only if byte at location 0C20H / 2400H is divisible by two. The number of bytes in the list is given at location 0C21H / 2401H (in BCD). If the insertion is valid, insert the byte at the end of the list and modify the count accordingly.

**SOFTWARE USED: EMU8086 Simulator**

**CODE:**

```
CODE:  
        MOV SI, 2400H  
        MOV CH, [SI]  
        INC SI  
        MOV AL, [SI]  
        AND AL, 10H  
        ROR AL, 04H  
        ADD AL, AL  
        MOV AH, AL  
        ADD AL, AL  
        ADD AL, AH  
        MOV AH, AL  
        MOV AL, [SI]  
        AND AL, 0FH  
        ADD AL, AH  
        MOV CL, AL  
        INC SI  
LOOPJ:  MOV AL, [SI]  
        CMP AL, CH  
        JZ JUMP  
        INC SI  
        DEC CL  
        JNZ LOOPJ  
        MOV AL, CH  
LOOPR:  SUB AL, 02H  
        CMP AL, 02H  
        JZ JUMP1
```

JNC LOOP2

HLT

JUMP1: MOV [SI], CH

MOV AL, [2401H]

ADD AL, OH

DAA

MOV [2401H], AL

JUMP: HLT

## **INPUT :**

0100:2400	update	<input checked="" type="radio"/> table	<input type="radio"/> list
0100:2400	04 09 32 35 21 39 43 42-45 38 09 00 00 00 00 00	♦.25†9CBE8.....	-
0100:2410	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	-	-
0100:2420	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	-	-
0100:2430	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	-	-
0100:2440	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	-	-
0100:2450	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	-	-
0100:2460	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	-	-
0100:2470	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00	-	-

