Interface two 4Kx8 EPROMS and 4Kx8 RAM chips. Select suitable maps.

Table 5.1 *Memory Map for Problem 5.1*

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
							EPROM						8K × 8							

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FE000H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
FDFFFH	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
							RAM				8K × 8									
FC000H	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.2 *Memory Chip Selection for Problem 5.1*

Decoder I/P → *Address/ $\overline{BHE}$ →*	A_2 A_{13}	A_1 A_0	A_0 $\overline{BHE}$	*Selection/ Comment*
Word transfer on $D_0 - D_{15}$	0	0	0	Even and odd addresses in RAM
Byte transfer on $D_7 - D_0$	0	0	1	Only even address in RAM
Byte transfer on $D_8 - D_{15}$	0	1	0	Only odd address in RAM
Word transfer on $D_0 - D_{15}$	1	0	0	Even and odd addresses in ROM
Byte transfer on $D_0 - D_7$	1	0	1	Only even address in ROM
Byte transfer on $D_8 - D_{15}$	1	1	0	Only odd address in ROM

Interface 16 bit 8255 ports with 8086 at 80H as an I/O address of port A. Interface five 7 segment displays with 8255. Write a sequence of instructions to display 1, 2, 3, 4 and 5 over five displays continuously as per their position starting with 1 at the least significant position.

Number to be displayed	PA_7 *dp*	PA_6 *a*	PA_5 *b*	PA_4 *c*	PA_3 *d*	PA_2 *e*	PA_1 *f*	PA_0 *g*	*Code*
1	1	1	0	0	1	1	1	1	CF
2	1	0	0	1	0	0	1	0	92
3	1	0	0	0	0	1	1	0	86
4	1	1	0	0	1	1	0	0	CC
5	1	0	1	0	0	1	0	0	A4

```
AGAIN:  MOV CL, 05H           ; Count for displays
        MOV BX, 2000H         ; Initialise data segment
        MOV DS, BX            ; for look up table
        MOV CH, 01H           ; 1st number to be displayed
        MOV AL, 80H           ; Load control word in the
        OUT 86H,AL            ; CWR
        MOV DL,01H            ; Enable code for Least significant
                              ; 7-seg display
NXTDGT : MOV BX, 0000H        ; Set pointer to look up table
        MOV AL, CH            ; First no to display
                              ; Store number to be displayed in AL.
        XLAT                  ; Find code from look up table
        OUT 80H,AL            ; Display the code
        MOV AL, DL            ; Enable the display
        OUT 81H,AL            ;
        ROL DL                ; Go for selecting the next display


        INC CH                ; Next number to display
        DEC CL                ; Decrement count.
        JNZ NXTDGT            ; Go for next digit display
        JMP AGAIN             ; Repeat the procedure
```

Program 5.7 *ALP for Problem 5.13*

Write an assembly language program in 8086 to find how many times 80H appear in an array of 100 bytes of data starting at ARR1.

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
ARR1 DB 01H, 02H, 80H, 04H, 05H
DATA ENDS

CODE SEGMENT

; Initialize variables
   MOV CX, 5
   MOV SI, OFFSET ARR1
   MOV BX, 0

; Loop through the array

COUNT_LOOP:
   MOV AL, [SI]
   CMP AL, 80h
   JNE NOT_FOUND
   INC BX
NOT_FOUND:
   INC SI
   LOOP COUNT_LOOP
   MOV AH, 4CH
   INT 21H

CODE ENDS
```

Write an 8086 program which scans a string of 80 characters looking for all occurrences character 'L'. If 'L' is found, output the number of times L is found otherwise print the message' not found'.

```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT

STR DB 80 DUP(0)
COUNT DW 0000H

DATA ENDS

CODE SEGMENT

START:

; Initialize variables
MOV CX, 80
MOV SI, OFFSET STR

; Loop through the string

COUNT_LOOP:
MOV AL, [SI]
CMP AL, 'L'
JZ FOUND
INC SI
LOOP COUNT_LOOP

; String not found
MOV DX, OFFSET NOT_FOUND_MSG
MOV AH, 09H
INT 21H
JMP EXIT

; String found
FOUND:
INC COUNT
JMP COUNT_LOOP

; Print the number of times 'L' is found
NOT_FOUND_MSG DB 'L not found', 0DH, 0AH, '$'

EXIT:
MOV AH, 4CH
INT 21H

CODE ENDS
```

Write an assembly language procedure BCD_TO_BIN to convert a two-digit BCD number into Binary. The number should be passed as a parameter on stack.

```
BCD_TO_BIN PROC

; Save registers
PUSH BP
PUSH BX

; Get the parameters
MOV BP, SP
MOV BX, [BP + 2]

; Convert the BCD number to Binary
MOV AL, BL
AND AL, 0FH
MOV CL, 04H
SHL AL, CL
OR AL, BH

; Restore registers
POP BX
POP BP

; Return
RET

BCD_TO_BIN ENDP
```

Give an example program which uses DI flag in auto incrementing mode.

```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ARR DB 10H, 20H, 30H, 40H, 50H

DATA ENDS

CODE SEGMENT

START:

; Initialize DI
MOV DI, OFFSET ARR

; Loop through the array
LOOP_1:
MOV AL, [DI]
INC DI
LOOP LOOP_1

; Print the array
MOV AH, 09H
MOV DX, OFFSET ARR
INT 21H

; Exit the program
MOV AH, 4CH
INT 21H

CODE ENDS
```

Write an assemble language program to arrange a given series of hexadecimal bytes in ascending order.

```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT
ARRAY DB 01H, 02H, 03H, 04H, 05H
COUNT DB 5
DATA ENDS

CODE SEGMENT

START:
   MOV AX, DATA
   MOV DS, AX

   MOV SI, 0
   MOV CL, COUNT

NEXT_BYTE:
   MOV AL, ARRAY[SI]
   INC SI

   CMP AL, ARRAY[SI]
   JB NO_SWAP

   MOV AH, ARRAY[SI]
   MOV ARRAY[SI], AL
   MOV ARRAY[SI - 1], AH

NO_SWAP:
   LOOP NEXT_BYTE

   MOV AH, 4CH
   INT 21H

CODE ENDS

END START
```

Write an assemble language program to find square root of two-digit number. Assume that the number is a perfect square.

```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT
NUM DB 16
DATA ENDS

CODE SEGMENT

START:
   MOV AX, DATA
   MOV DS, AX

   MOV AL, NUM
   MOV BL, 01H

SQRT_LOOP:
   CMP AL, BL
   JB NO_SQUARE

   INC BL
   MOV AH, BL
   MUL AH

   CMP AH, AL
   JB SQRT_LOOP

NO_SQUARE:
   MOV AH, 4CH
   INT 21H

CODE ENDS

END START
```

Use 8086 string instructions to write an assembly language program to match a password string.

```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT
PASSWORD DB "password"
TEST_STRING DB "password123"
NO_MATCH_MSG DB "Passwords do not match.", 0DH, 0AH, '$'
DATA ENDS

CODE SEGMENT

START:
   MOV AX, DATA
   MOV DS, AX

   MOV SI, 0
   MOV DI, 0

   MOV CX, 8

MATCH_LOOP:
   MOV AL, TEST_STRING[DI]
   CMP PASSWORD[SI], AL
   JNE NO_MATCH

   INC SI
   INC DI

   LOOP MATCH_LOOP

   MOV AH, 4CH
   INT 21H

NO_MATCH:
   MOV DX, OFFSET NO_MATCH_MSG
   MOV AH, 09H
   INT 21H

END START
```

Write an 8086 Assembly Language Program to compare whether two strings stored in memory are equal or not.

```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT
STR1 DB "Hello, world"
STR2 DB "Hello, world"
NOT_EQUAL_MSG DB "Strings are not equal.", 0DH, 0AH, '$'
DATA ENDS

CODE SEGMENT

START:
   MOV AX, DATA
   MOV DS, AX

   MOV SI, OFFSET STR1
   MOV DI, OFFSET STR2

   MOV CX, 12

LOOP:
   MOV AL, STR1[SI]
   CMP AL, STR2[DI]
   JNE NOT_EQUAL

   INC SI
   INC DI

   LOOP LOOP

EQUAL:
   MOV AH, 4CH
   INT 21H

NOT_EQUAL:
   MOV DX, OFFSET NOT_EQUAL_MSG
   MOV AH, 09H
   INT 21H

END START
```

Write an assembly language program to multiply two 16 bit numbers

```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT
NUM1 DW 1234H
NUM2 DW 5678H
RESULT DW ?
DATA ENDS

CODE SEGMENT

START:
   MOV AX, NUM1
   MOV BX, NUM2
   MUL BX
   MOV RESULT, AX

   MOV AH, 4CH
   INT 21H

END START
```

Write an assembly language program to convert a binary number to its equivalent BCD number.

```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT
BIN_NUM DB 10101011B
BCD_NUM DB ?
DATA ENDS

CODE SEGMENT

START:
MOV AX, DATA
MOV DS, AX

MOV AL, BIN_NUM
MOV CL, 4

BCD_LOOP:
MOV AH, 0
AND AL, 00001111B
ADD AH, AL
MOVSB
SHR AL, 1
DEC CL
JNZ BCD_LOOP

MOV AH, 4CH
INT 21H

END START
```

Show the command words to initialize 8279 as (i) 8- character display right-entry, encoded scan Key-board N-key rollover (ii) 1-MHz input clock divided to 100 KHz and FFH as blanking character.

(i)

0	**0**	**0**	**1**	**0**	**0**	**1**	**0**

(ii)

0	**0**	**1**	**0**	**1**	**0**	**1**	**0**

For more details, read page no 258 of book (page no 277 of pdf)

Explain the instruction:

LEA, ADR

LES BX, 5000H

LEA instruction will load the effective address of the memory location 5000H into the ADR register. The LES BX, ADR instruction will then load the effective address that is stored in the ADR register into the BX segment register.