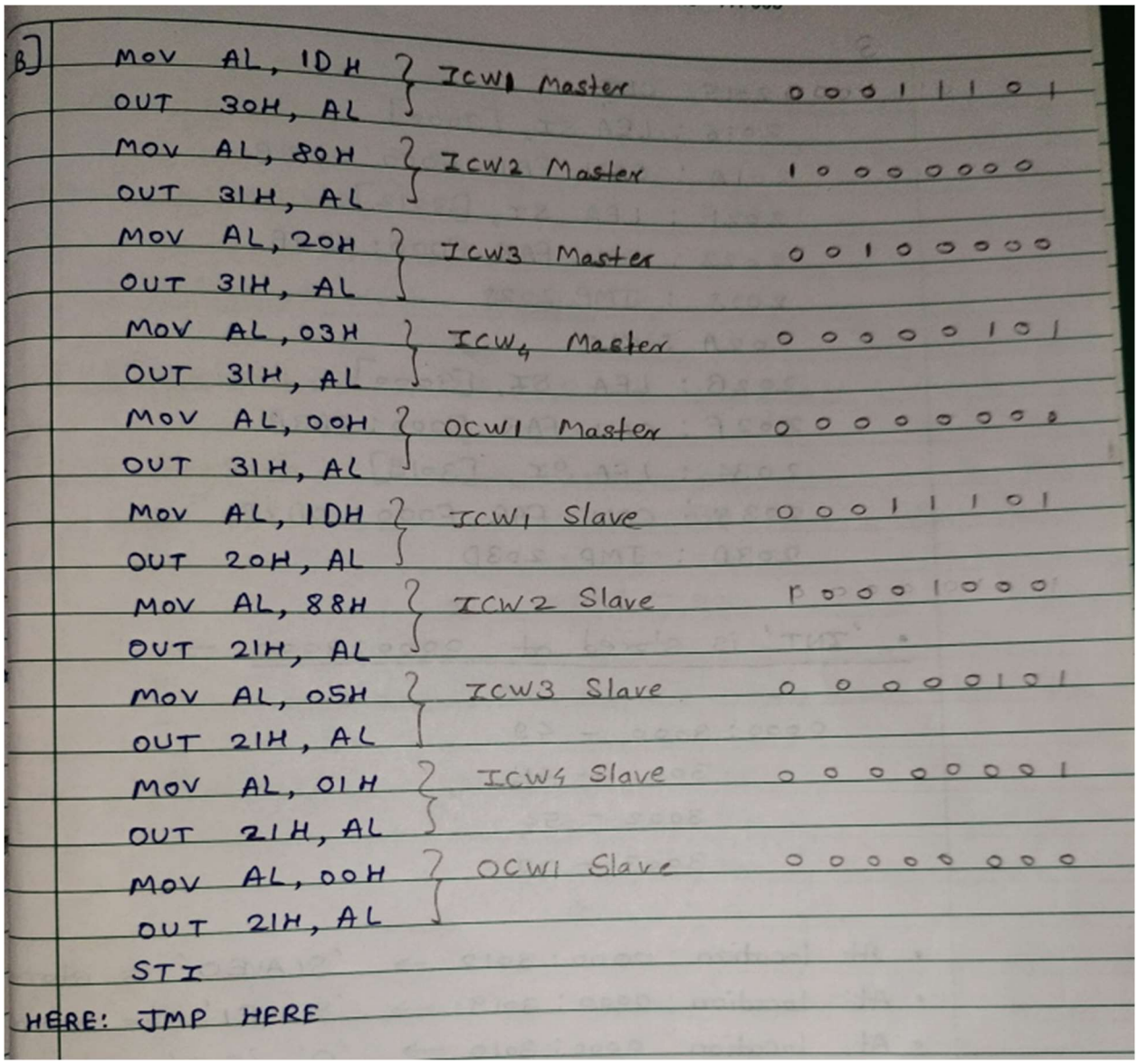
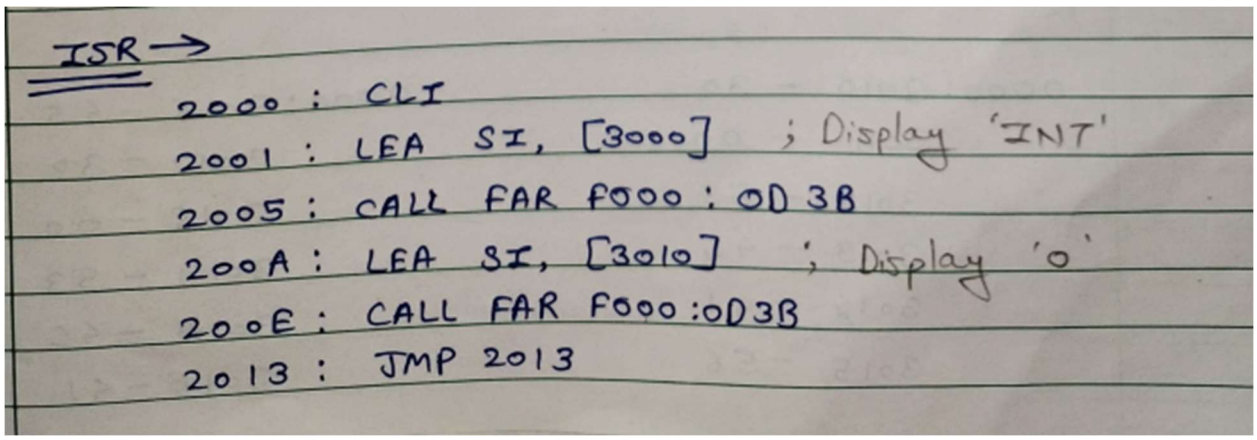
1. Write a program in 8086 assembly language to demonstrate Fully nested mode



B]

```
MOV AL, 1DH   } ICW1 Master      0 0 0 1 1 1 0 1
OUT 30H, AL   }
MOV AL, 80H   } ICW2 Master      1 0 0 0 0 0 0 0
OUT 31H, AL   }
MOV AL, 20H   } ICW3 Master      0 0 1 0 0 0 0 0
OUT 31H, AL   }
MOV AL, 03H   } ICW4 Master      0 0 0 0 0 1 0 1
OUT 31H, AL   }
MOV AL, 00H   } OCW1 Master      0 0 0 0 0 0 0 0
OUT 31H, AL   }
MOV AL, 1DH   } ICW1 Slave       0 0 0 1 1 1 0 1
OUT 20H, AL   }
MOV AL, 88H   } ICW2 Slave       1 0 0 0 1 0 0 0
OUT 21H, AL   }
MOV AL, 05H   } ICW3 Slave       0 0 0 0 0 1 0 1
OUT 21H, AL   }
MOV AL, 01H   } ICW4 Slave       0 0 0 0 0 0 0 1
OUT 21H, AL   }
MOV AL, 00H   } OCW1 Slave       0 0 0 0 0 0 0 0
OUT 21H, AL   }
STI
HERE: JMP HERE
```



ISR →

```
2000 : CLI
2001 : LEA SI, [3000]   ; Display 'INT'
2005 : CALL FAR F000 : 0D 3B
200A : LEA SI, [3010]   ; Display '0'
200E : CALL FAR F000 :0D3B
2013 : JMP 2013
```

```
2015: CLI
2016 : LEA SI, [3000]  ; Display 'INT'
201A : CALL FAR F000:0D3B
202F : LEA SI, [3012] ; Display 'SLAVE0'
2023 : CALL FAR F000:0D3B
2028 : JMP 2028
202A : CLI
202B : LEA SI, [3000] ; Display 'INT'
202F : CALL FAR F000:0D3B
2034 : LEA SI, [3019] ; Display 'SLAVE1'
2038 : CALL FAR F000:0D3B
203D : JMP 203D
```

- 'INT' is stored at 0000:3000 →

```
0000:3000 - 49
     3001 - 4E
     3002 - 54
     3003 - 00
```

- At location 0000:3012 → 'SLAVE0' is stored
- At location 0000:3019 → 'SLAVE1' is stored
- At location 0000:3010 → '0' is stored.

```
0000:3010 - 30
     3011 - 00
     3012 - 53
     3013 - 4C
     3014 - 41
     3015 - 56
```

```
0000:3016 - 45
     3017 - 30
     3018 - 00
     3019 - 53
     301A - 4C
     301B - 41
```

0000 : 301C - 56
301D - 45
301E - 31
301F - 00

• Interrupt Vector Table -

Type no. → 80H ⇒ 80 × 4 = 0200H

Therefore, 0000 : 0200 - 00
0201 - 20
0202 - 00
0203 - 00

Type no. → 88H ⇒ 88 × 4 = 0220H

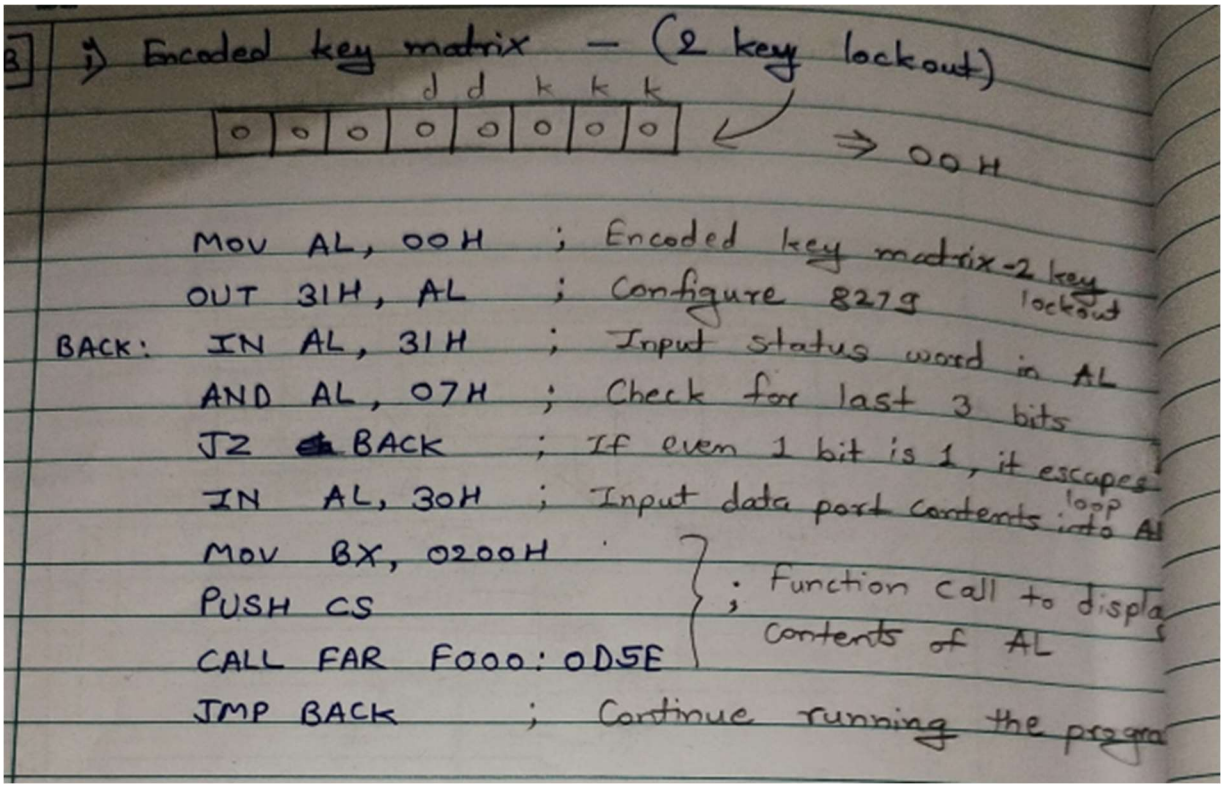
Therefore, 0000 : 0220 - 15
0221 - 20
0222 - 00
0223 - 00

Type no. → 89H ⇒ 89 × 4 = 0224H

Therefore, 0000 : 0224 - 2A
0225 - 20
0226 - 00
0227 - 00

2. Write a program in 8086 assembly language to demonstrate

3. Write a program in 8086 assembly language to demonstrate Encoded key Matrix of 8279



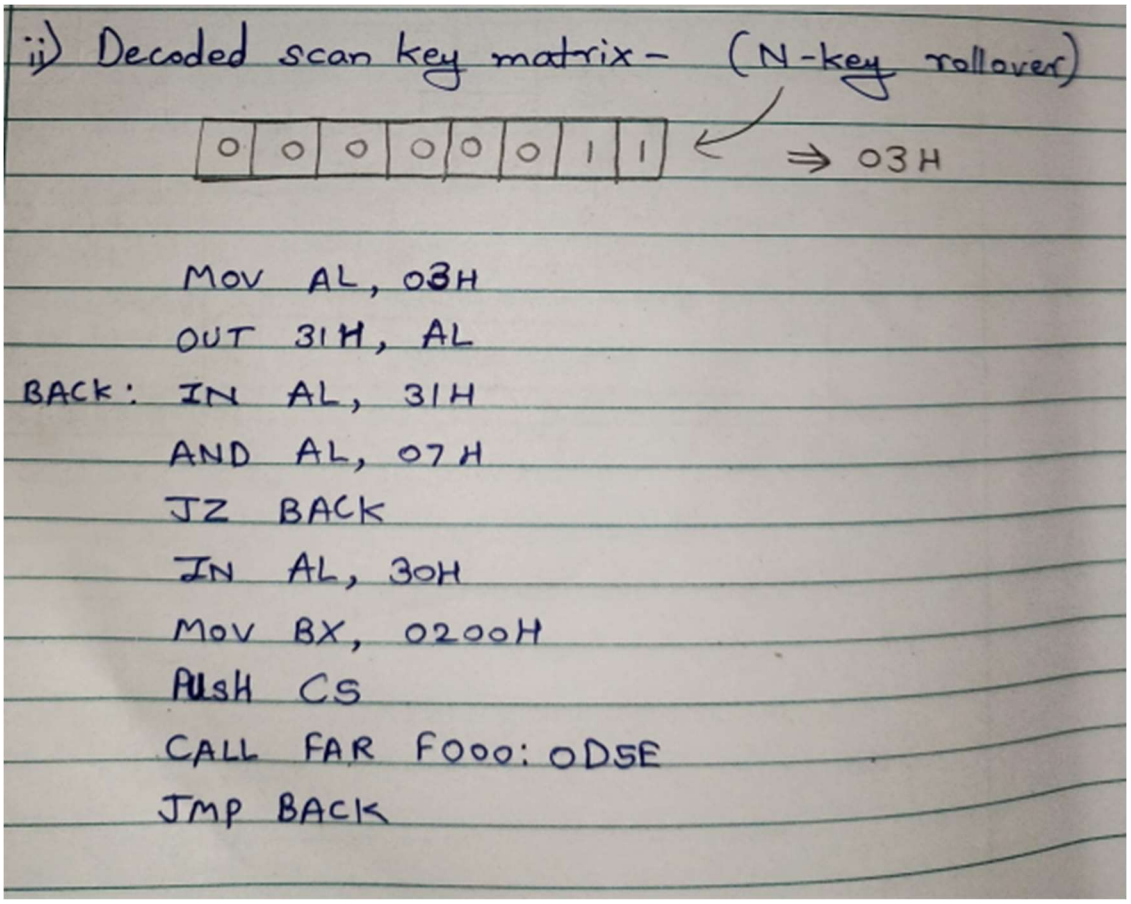
i) Encoded key matrix – (2 key lockout)

d d k k k

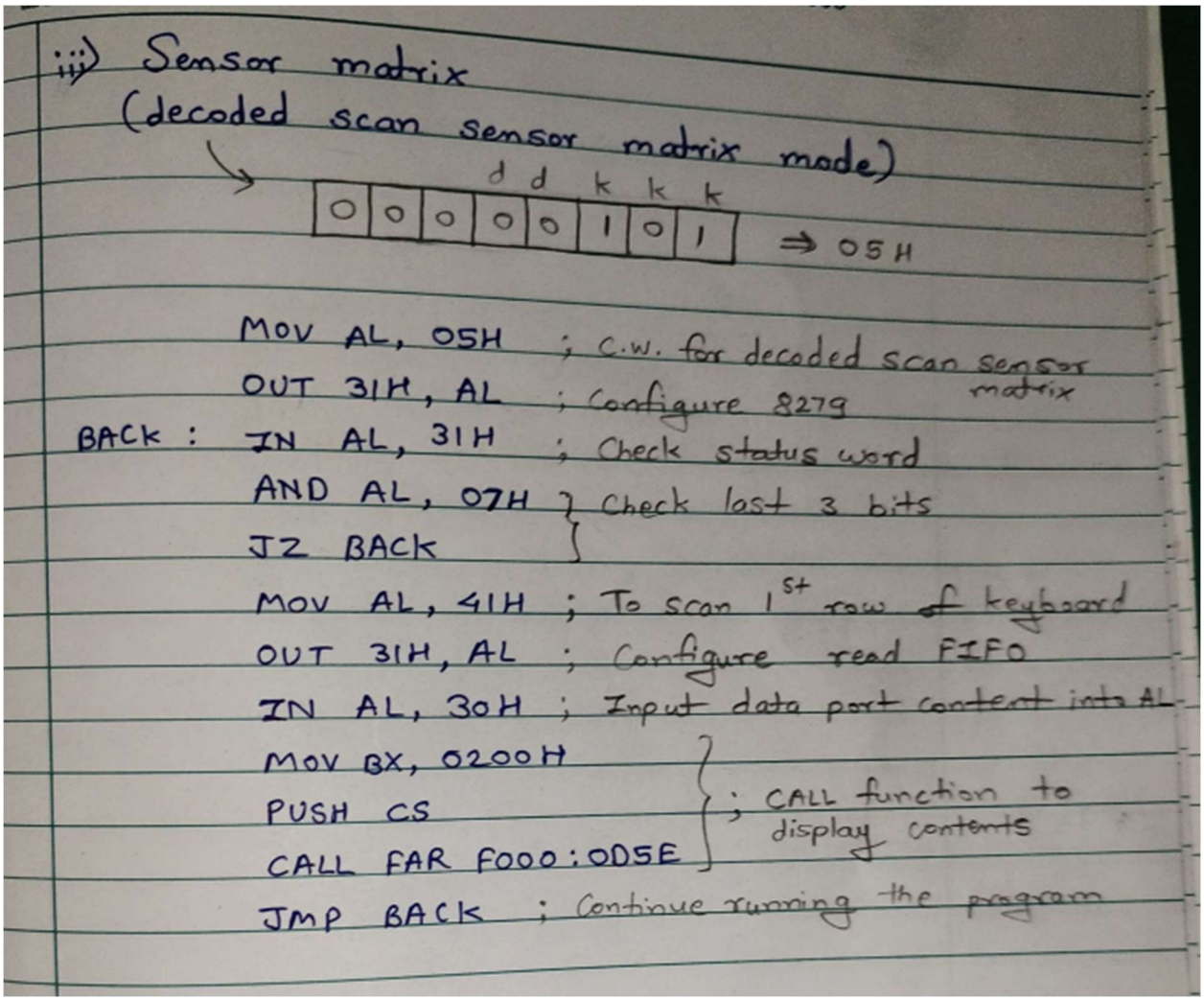
0 0 0 0 0 0 0 0 ⇒ 00 H

```
        MOV AL, 00H   ; Encoded key matrix-2 key lockout
        OUT 31H, AL   ; Configure 8279
BACK:   IN AL, 31H    ; Input status word in AL
        AND AL, 07H   ; Check for last 3 bits
        JZ BACK       ; If even 1 bit is 1, it escapes loop
        IN AL, 30H    ; Input data port contents into AL
        MOV BX, 0200H
        PUSH CS
        CALL FAR F000:0D5E   ; Function call to display contents of AL
        JMP BACK      ; Continue running the program
```

4. Write a program in 8086 assembly language to demonstrate Decoded scan key Matrix of 8279



5. Write a program in 8086 assembly language to demonstrate Sensor Matrix of 8279



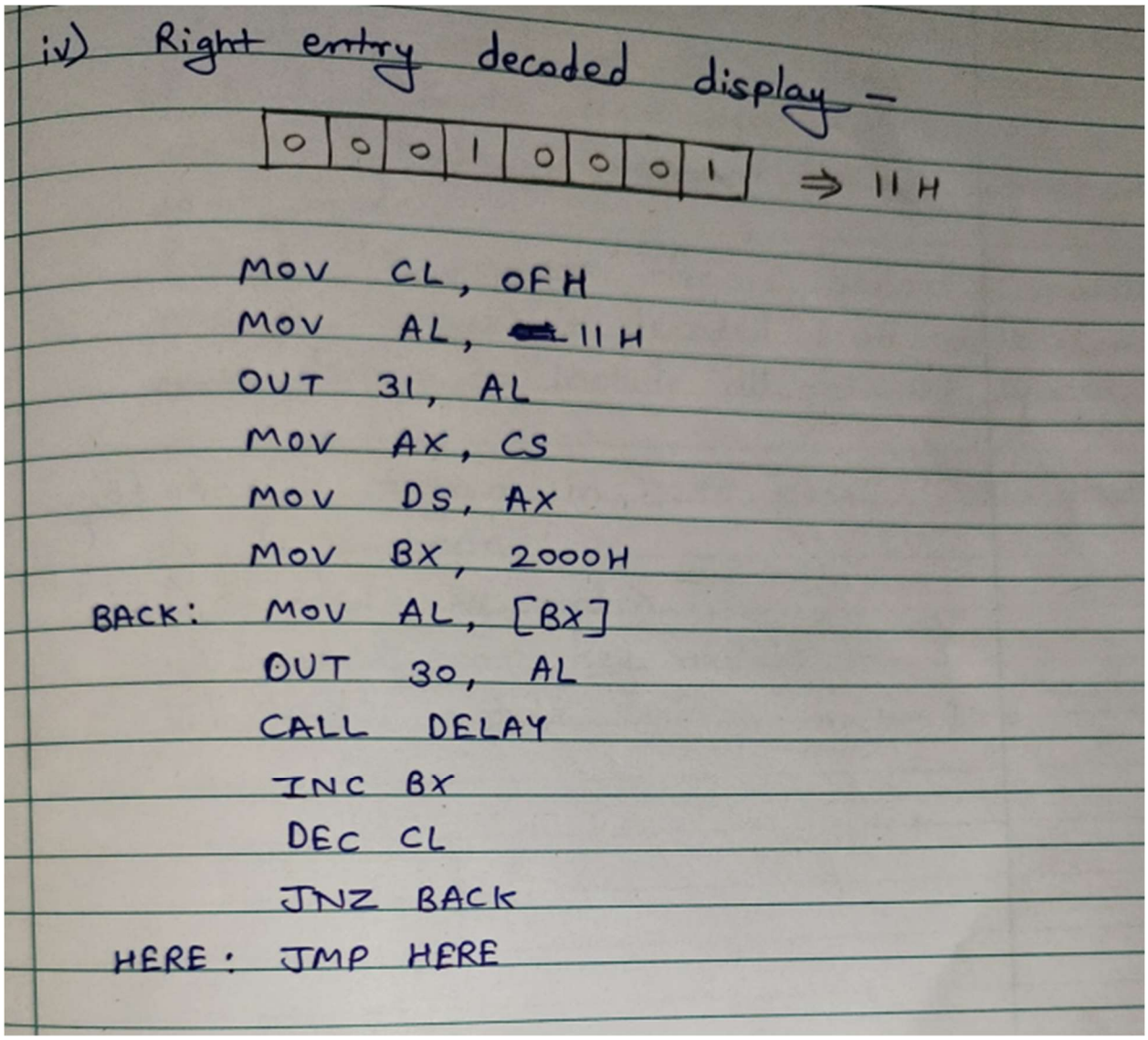
iii) Sensor matrix

(decoded scan sensor matrix mode)

					d	d	k	k	k
	0	0	0	0	0	1	0	1	⇒ 05H

```
        MOV AL, 05H   ; C.W. for decoded scan sensor matrix
        OUT 31H, AL   ; Configure 8279
BACK :  IN AL, 31H    ; Check status word
        AND AL, 07H   } Check last 3 bits
        JZ BACK       }
        MOV AL, 41H   ; To scan 1st row of keyboard
        OUT 31H, AL   ; Configure read FIFO
        IN AL, 30H    ; Input data port content into AL
        MOV BX, 0200H }
        PUSH CS       } ; CALL function to display contents
        CALL FAR F000:0D5E }
        JMP BACK      ; Continue running the program
```

6. Write a program in 8086 assembly language to demonstrate following 8279 seven segment display interface modes. Display digits 0 onwards in a right entry decoded scan



7. Write a program in 8086 assembly language to demonstrate following 8279 seven segment display interface modes. Display digits 0 onwards in a right entry mode in encoded scan

iii) Right entry encoded 16·8 character display –

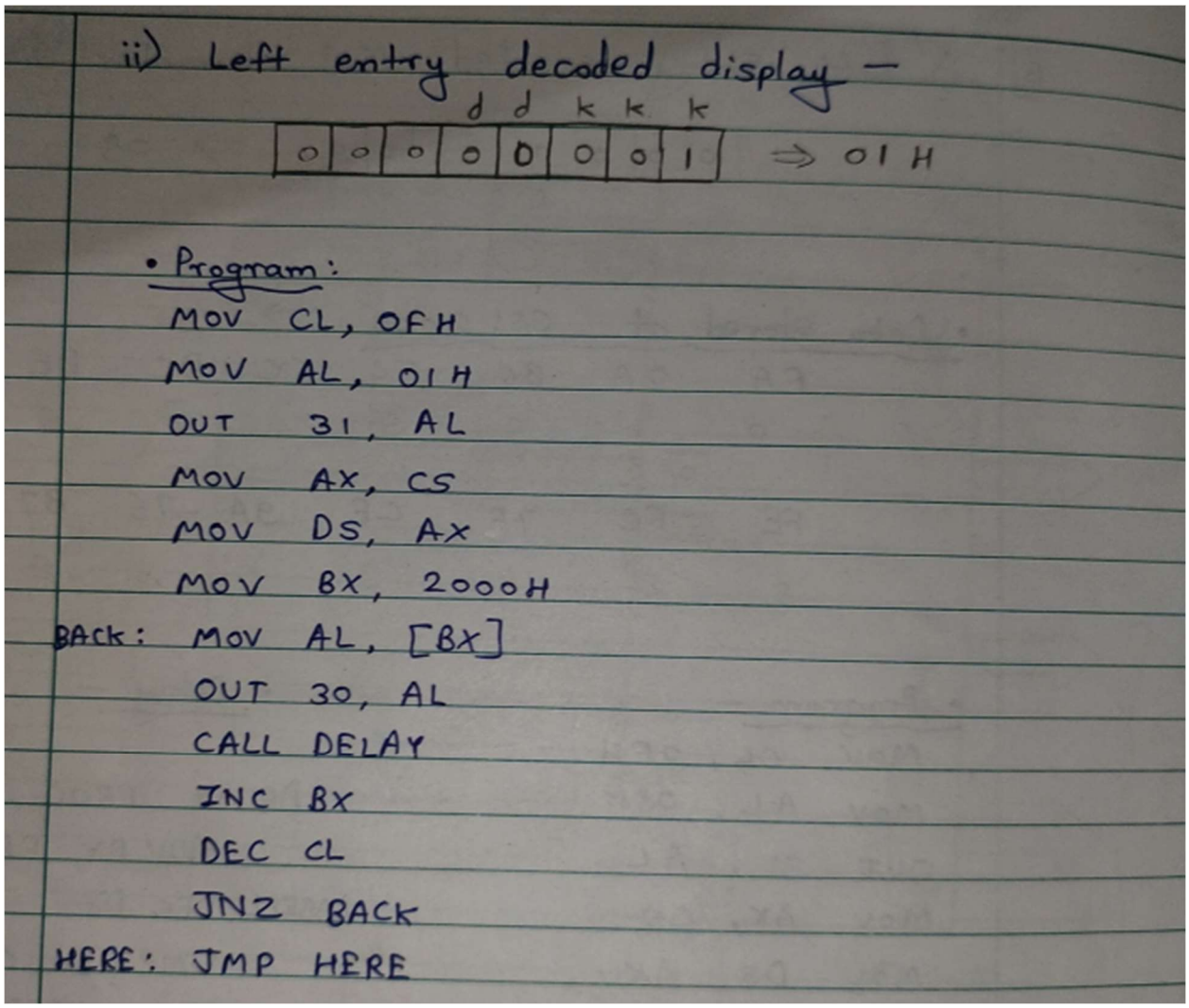
0	0	0	1	1	0	0	0

⇒ 18H

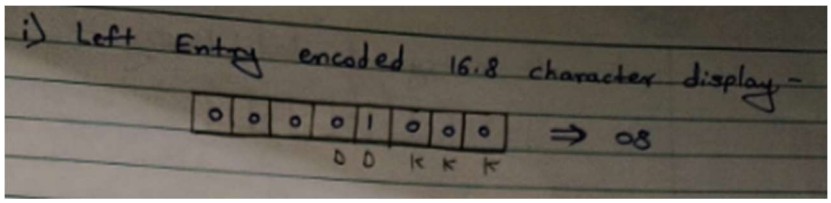
```
        MOV  CL, 0FH
        MOV  AL, 18H
        OUT  31, AL
        MOV  AX, CS
        MOV  DS, AX
        MOV  BX, 2000H
BACK:   MOV  AL, [BX]
```

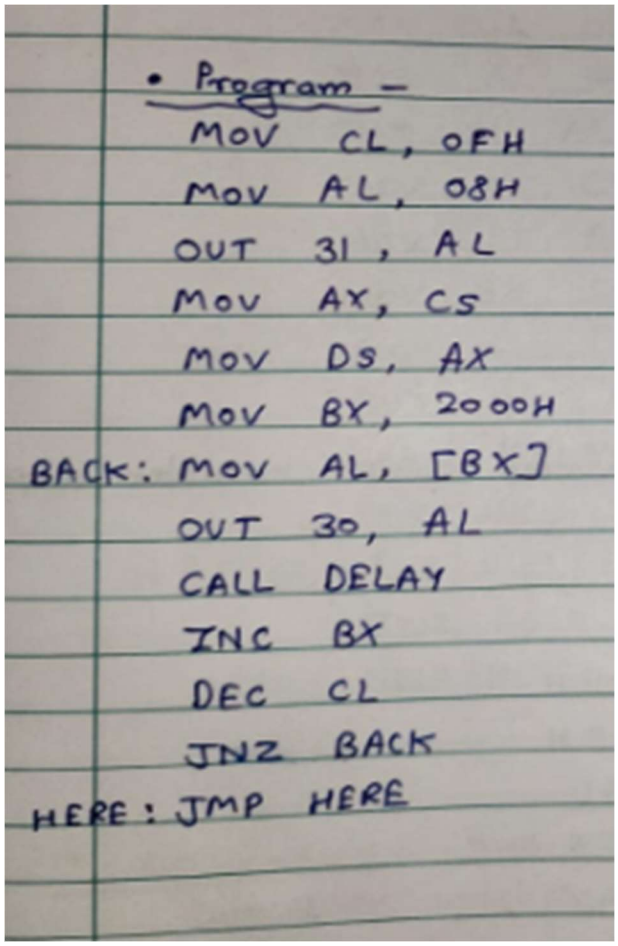
```
        OUT  30, AL
        CALL DELAY
        INC  BX
        DEC  CL
        JNZ  BACK
HERE:   JMP  HERE
```

8. a program in 8086 assembly language to demonstrate following 8279 seven segment display interface modes. Display digits 0 onwards in a left entry mode in decoded scan



9. Write a program in 8086 assembly language to demonstrate following 8279 seven segment display interface modes. Display digits 0 onwards in a left entry mode in both encoded scan

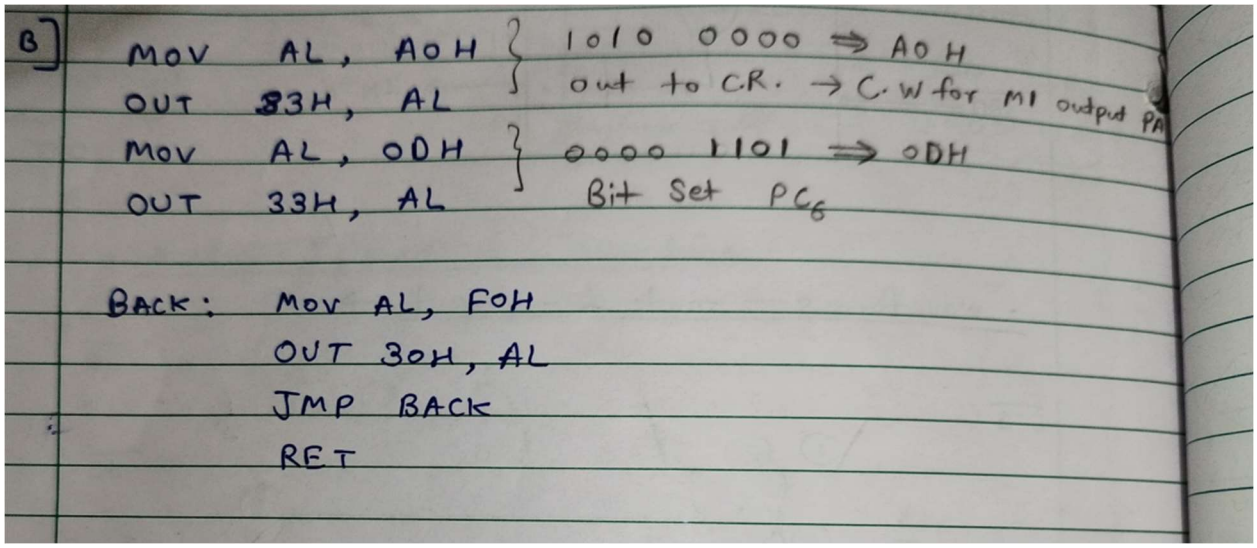




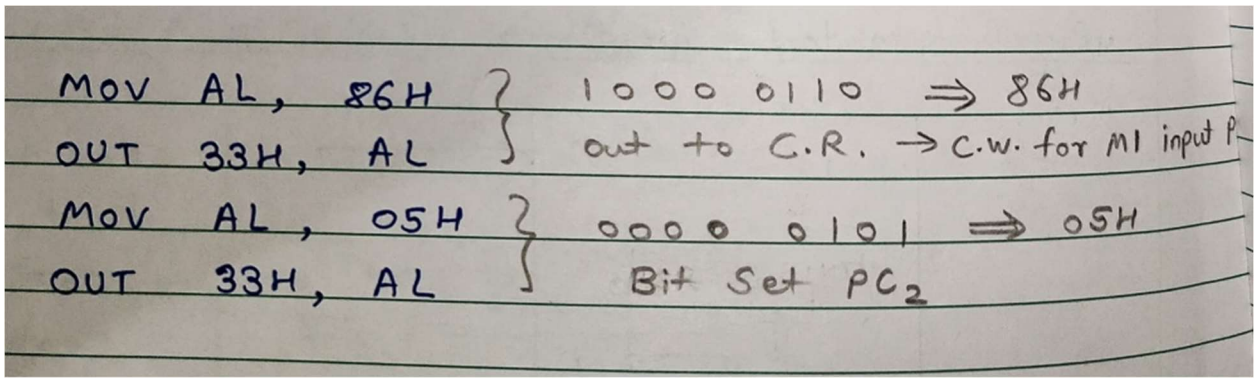
- Program –

```
      MOV  CL, 0FH
      MOV  AL, 08H
      OUT  31 , AL
      MOV  AX, CS
      MOV  DS, AX
      MOV  BX, 2000H
BACK: MOV  AL, [BX]
      OUT  30, AL
      CALL DELAY
      INC  BX
      DEC  CL
      JNZ  BACK
HERE: JMP  HERE
```

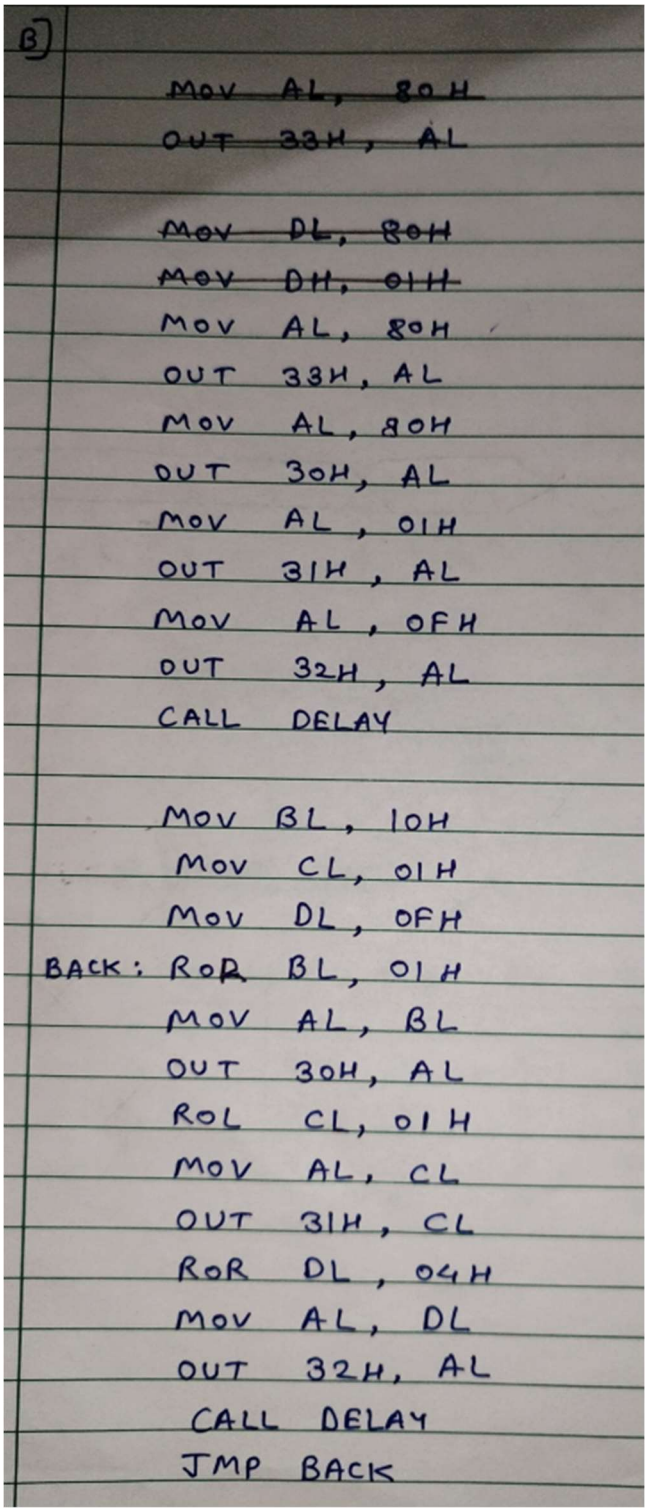
10. Write a program in 8086 assembly language to configure Port A of 8255 as output port in mode1. Verify operation of this mode by writing a program to send output byte to port A under the control of handshake signals



11. Write a program in 8086 assembly language to configure port B of 8255 in mode 1 input, verify it's operation under the control of relevant handshake signals



12. Write a program in 8086 assembly language to configure Port A, B and C as output ports in simple I/O mode. Generate flashing LEDs at Port C lines (with PCU and PCL turning on and off alternately), running lights from right to left at Port B and running lights from left to right at Port Fine tune the delay between flashing and running lights to 0.5 seconds



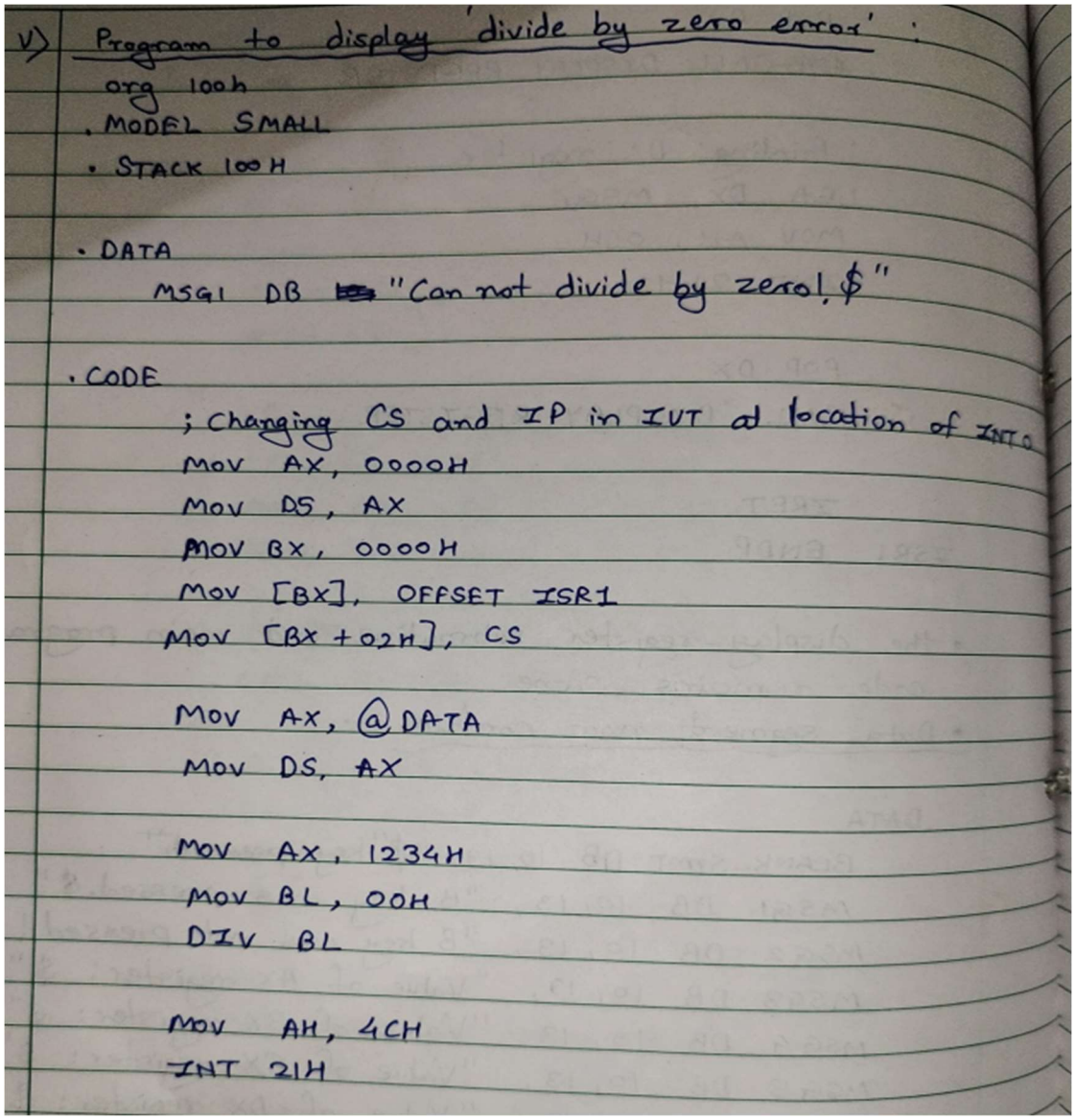
```
B)
        MOV AL, 80H
        OUT 33H, AL

        MOV DL, 80H
        MOV DH, 01H
        MOV AL, 80H
        OUT 33H, AL
        MOV AL, 80H
        OUT 30H, AL
        MOV AL, 01H
        OUT 31H, AL
        MOV AL, 0FH
        OUT 32H, AL
        CALL DELAY

        MOV BL, 10H
        MOV CL, 01H
        MOV DL, 0FH
BACK:   ROR BL, 01H
        MOV AL, BL
        OUT 30H, AL
        ROL CL, 01H
        MOV AL, CL
        OUT 31H, CL
        ROR DL, 04H
        MOV AL, DL
        OUT 32H, AL
        CALL DELAY
        JMP BACK
```

```
DELAY   PROC
        MOV   AX, FFFFH
HERE :  DEC   AX
        CMP   AX, 00H
        JNZ   HERE
        RET
DELAY   ENDP
```

c]

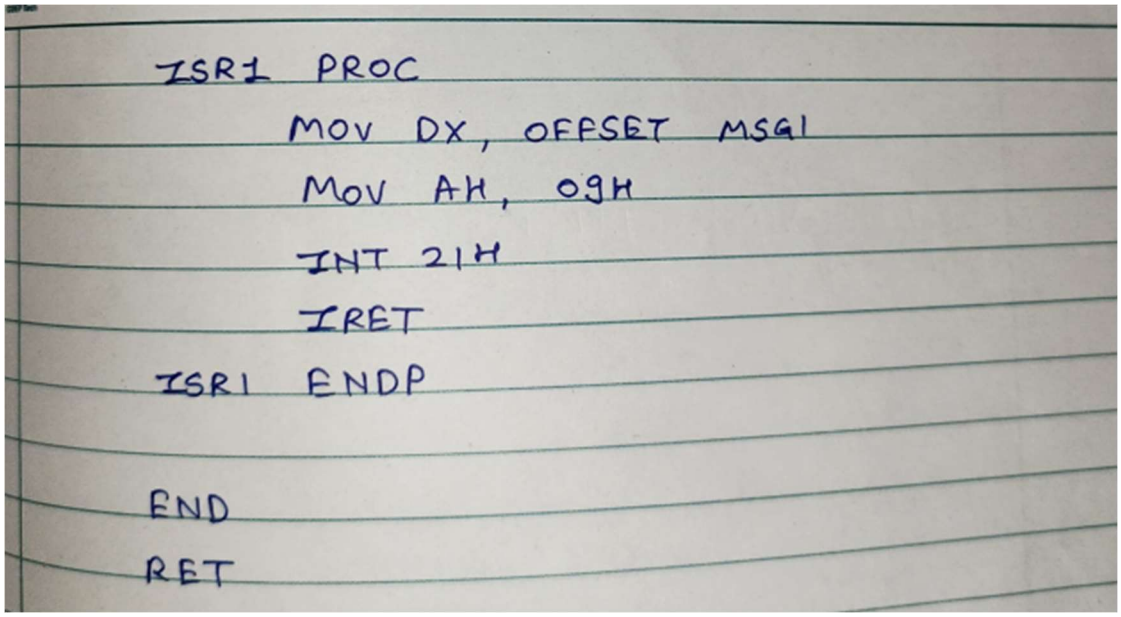
```
DELAY   PROC                    T-states
        MOV   DX, 05H           4 x 1
LABEL1: MOV   CX, AD9BH         4 x 5
LABEL2: DEC   CX                2 x 5 x N
        JNZ   LABEL2            16 x 5 x N
        DEC   DX                2 x 5
        JNZ   LABEL1            16 x 5
        RET                     8 x 1
DELAY   ENDP
```

13. Write ISS for divide by zero interrupt to display ÷ ZERO ERROR



v) Program to display 'divide by zero error' :

```
org 100h
.MODEL SMALL
.STACK 100H

.DATA
    MSG1 DB "Can not divide by zero!,$"

.CODE
    ; Changing CS and IP in IVT at location of INT 0
    MOV AX, 0000H
    MOV DS, AX
    MOV BX, 0000H
    MOV [BX], OFFSET ISR1
    MOV [BX+02H], CS

    MOV AX, @DATA
    MOV DS, AX

    MOV AX, 1234H
    MOV BL, 00H
    DIV BL

    MOV AH, 4CH
    INT 21H
```



```
ISR1 PROC
    MOV DX, OFFSET MSG1
    MOV AH, 09H
    INT 21H
    IRET
ISR1 ENDP

END
RET
```

14. Write a Interrupt Service Subroutine (ISS) for INT3 to increment CX by two and display contents of AX register on screen using INT 10 function call 0A of BIOS interrupt

iii)

```
org 100h
.MODEL SMALL
.STACK 100H
.DATA
  BLANK_STMT DB 10,13, "$"
  MSG1 DB 10,13, "B key was pressed.$"
  MSG2 DB 10,13, "B key was not pressed.$"
  MSG3 DB 10,13, "Value in CX is : $"
  MSG4 DB 10,13, "Value of AX register: $"
  COUNT DB 04H

.CODE
  ; Changing CS and IP in IVT at location of INT3
  MOV AX,0000H
  MOV DS, AX
  MOV BX, 0000H
  MOV [BX], ISR1
  MOV [BX+02H], CS

  MOV AX, @DATA
  MOV DS, AX

  MOV CX, 0000H
```

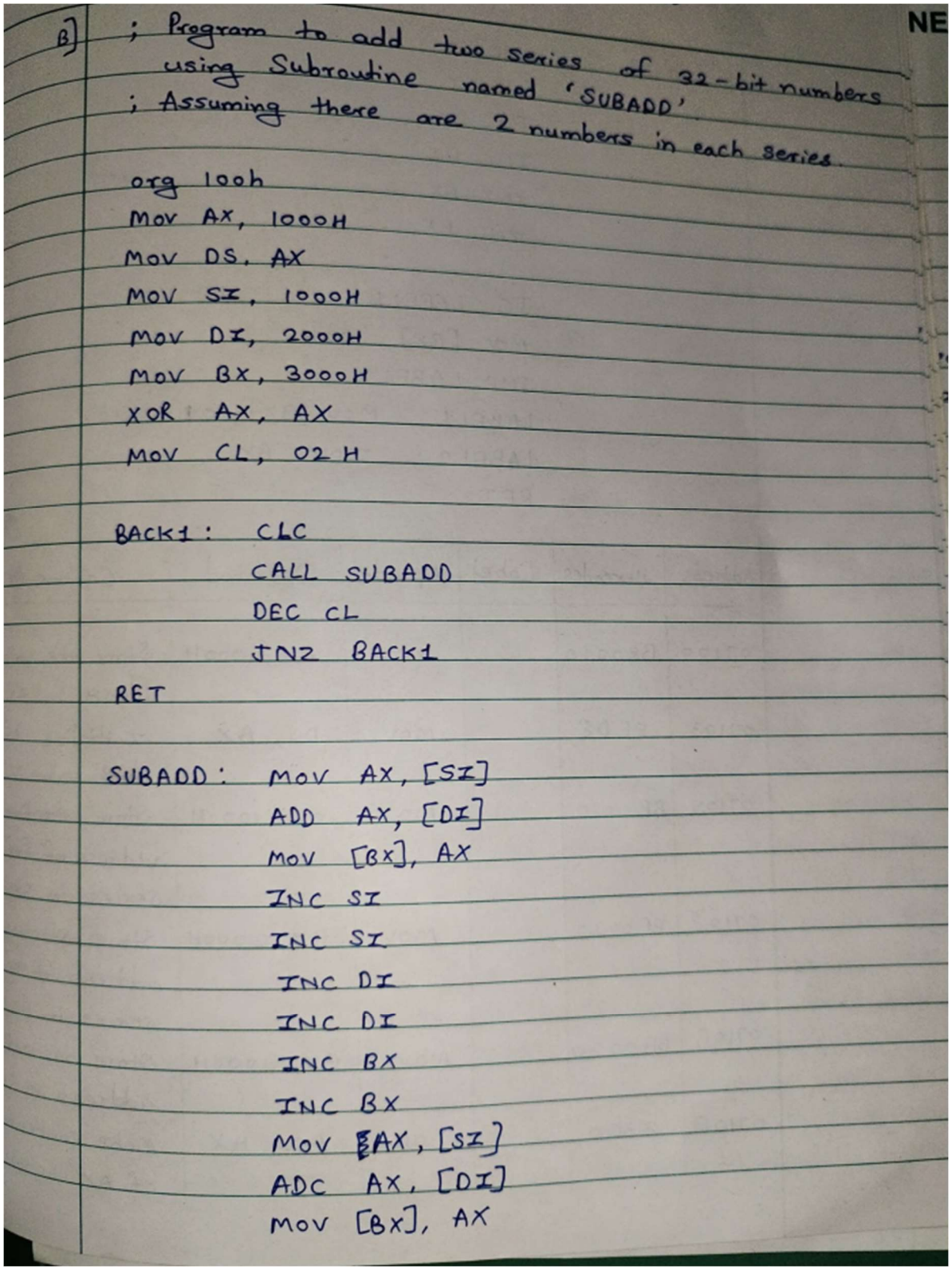
```
BACK: MOV AH,00H
      INT 16H
      MOV AH,0EH
      INT 10H
      CMP AL,42H
      JE NEXT
      MOV DX, OFFSET MSG2
      MOV AH, 09H
      INT 21H
      MOV DX, OFFSET BLANK_STMT
      MOV AH, 09H
      INT 21H
      JNE BACK
NEXT: MOV DX, OFFSET MSG1
      MOV AH, 09H
      INT 21H
      JNE BACK
      MOV DX, OFFSET BLANK_STMT
      MOV AX, 09H
      INT 21H

      MOV BX, OFFSET LAST
      MOV [BX], 00CCH

LAST: NOP

MOV AH, 4CH
INT 21H
```

```
ISR1 PROC:
    PUSH AX
    INC CX
    INC CX
    LEA DX, BLANK_STMT
    Mov AH, 09H
    INT 21H
    LEA DX, MSG3
    MOV AH, 09H
    INT 21H
    MOV DL, CL
    ADD DL, 30H
    MOV AH, 02H
    INT 21H

    ; Printing AX
    LEA DX, MSG4
    Mov AH, 09H
    INT 21H

    POP DX ; Store contents of AX in DX to display
    CALL DISPLAY_REGISTER

    IRET
ISR1 ENDP
```

```
DISPLAY_REGISTER PROC
    BACK2: ROL DX, 04H
           MOV AL, DL
           AND AL, 0FH
           CMP AL, 0AH
           JB DOWN2
           ADD AL, 07H
           DOWN2 : ADD . AL, 30H

           MOV BH, 00H
           MOV AH, 0AH
           MOV CX, 0001H
           INT 10H

           PUSH DX

           ; Get cursor position
           MOV AH, 03H
           XOR BH, BH
           INT 10H
           ; Move cursor one position right
           INC DL
           ; Set cursor position
           MOV AH, 02H
           INT 10H

           POP DX
           SUB COUNT, 01H
           JNZ BACK 2

       MOV COUNT, 04H
       RET
DISPLAY_REGISTER ENDP
```

15. Write ISS for detection of overflow to display OVERFLOW ERROR

16. Write a program in assembly language of 8086 to add two 32 bit numbers. First number is stored from 0000:2000, second number is stored from location 0000:3000. Store the result from 0000:4000 onwards

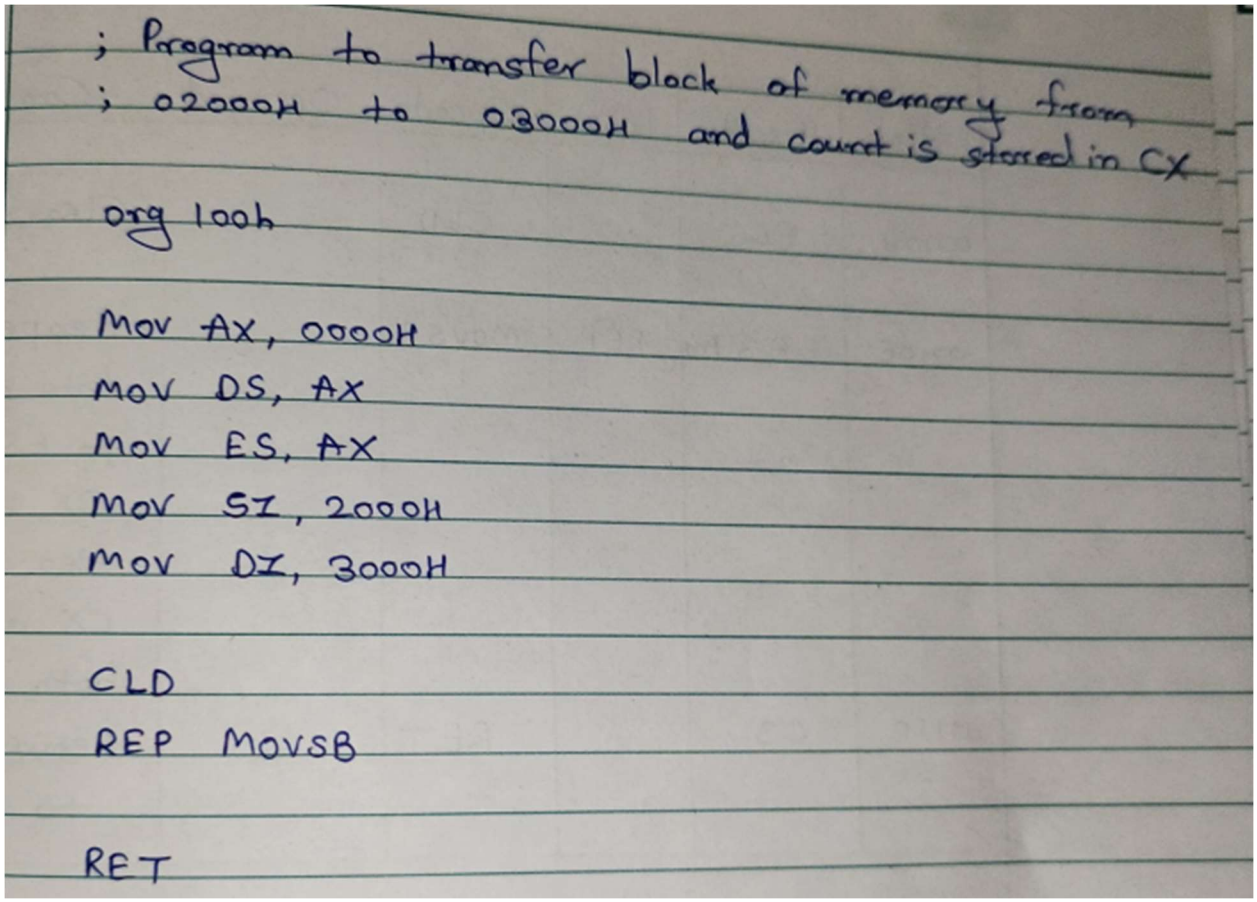


B]

```
; Program to add two series of 32-bit numbers
  using Subroutine named 'SUBADD'
; Assuming there are 2 numbers in each series.

org 100h
MOV AX, 1000H
MOV DS, AX
MOV SI, 1000H
MOV DI, 2000H
MOV BX, 3000H
XOR AX, AX
MOV CL, 02 H

BACK1: CLC
       CALL SUBADD
       DEC CL
       JNZ BACK1
RET

SUBADD: MOV AX, [SI]
        ADD AX, [DI]
        MOV [BX], AX
        INC SI
        INC SI
        INC DI
        INC DI
        INC BX
        INC BX
        MOV AX, [SI]
        ADC AX, [DI]
        MOV [BX], AX
```

```
INC SI
INC SI
INC DI
INC DI
INC BX
INC BX

JC LABEL1
MOV [BX], 00H
JMP LABEL2
LABEL1: MOV [BX], 01H
LABEL2: INC BX
RET
```

17. Write a program in assembly language of 8086 to transfer a block of memory. Starting address of source block is 0000:2000. Starting address of destination block is 0000:3000. Assume length of the block is stored in CX register



```
; Program to transfer block of memory from
; 02000H to 03000H and count is stored in CX

org 100h

Mov AX, 0000H
MOV DS, AX
Mov ES, AX
Mov SI, 2000H
Mov DI, 3000H

CLD
REP MOVSB

RET
```

18. Write a program in assembly language of 8086 to find number of negative elements in a block of memory. Starting address of source block is 0000:2000. Assume length of the block is stored in 0000:3000

19. Write a program in assembly language of 8086 to find number of positive elements in a block of memory. Starting address of source block is 0000:2000. Assume length of the block is stored in 0000:3000

20. Write a program in assembly language of 8086 to find number of even elements in a block of memory. Starting address of source block is 0000:2000. Assume length of the block is stored in 0000:3000