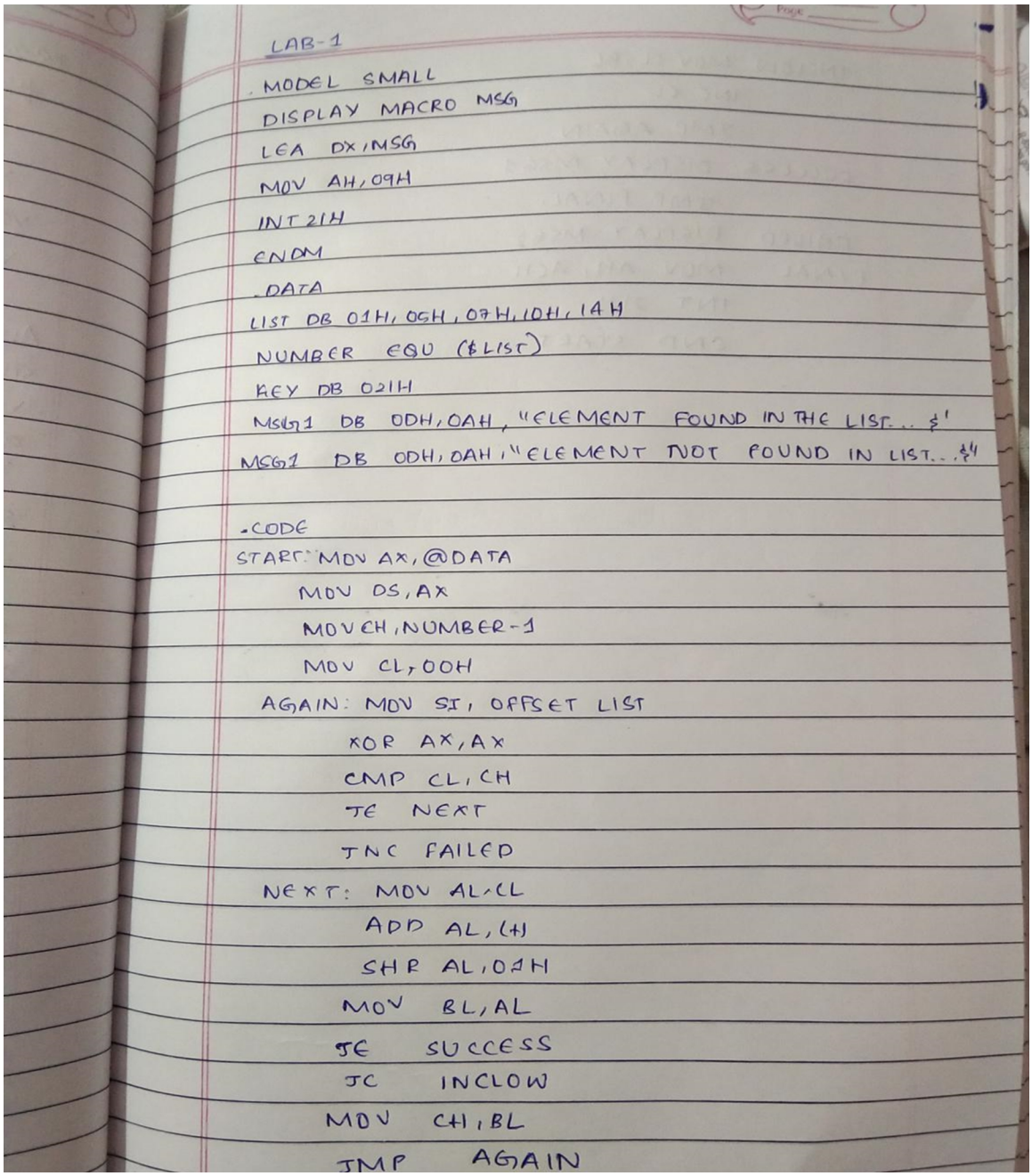
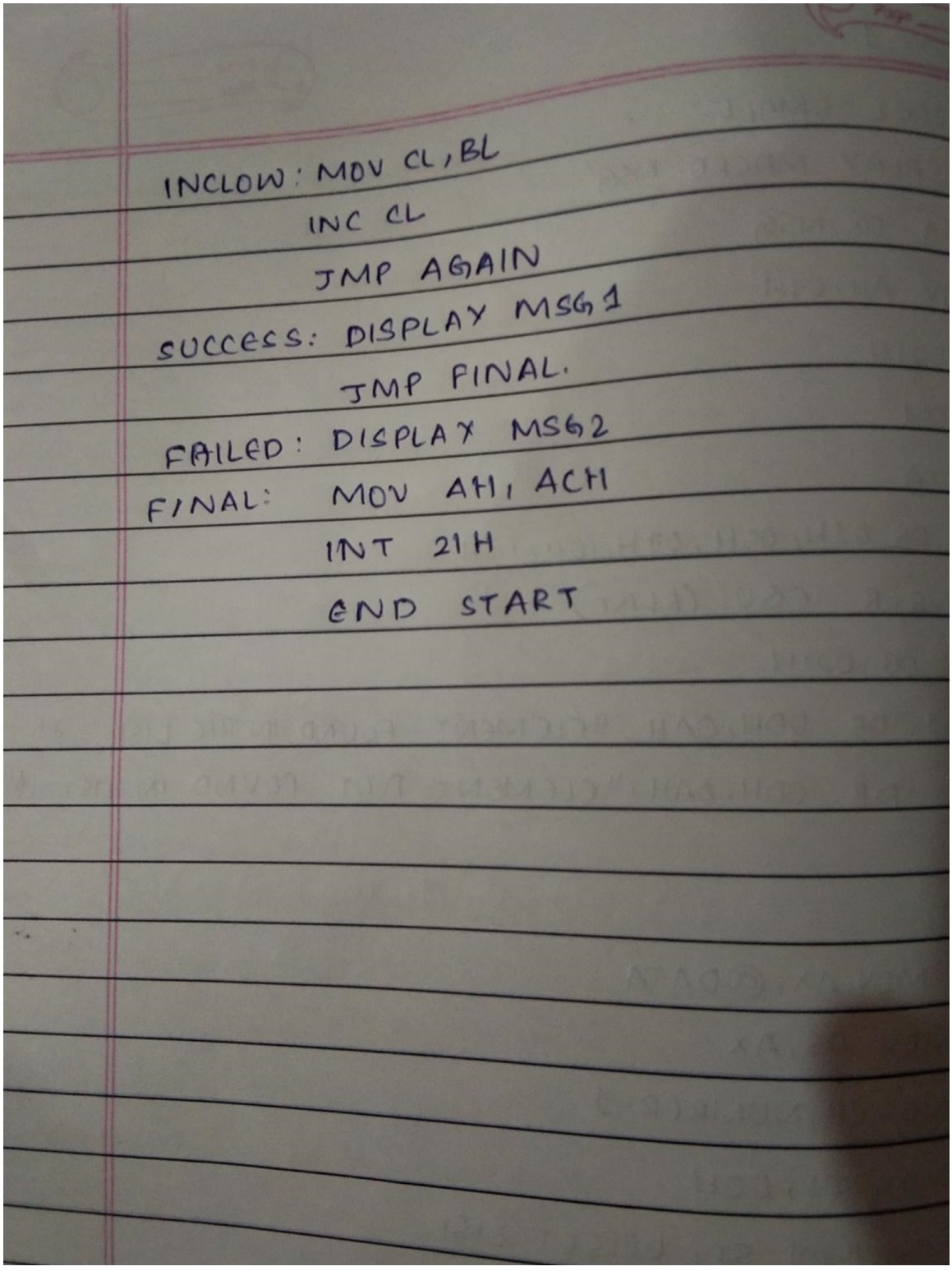
8086 Microprocessor Programs...

1.Design and develop an assembly language program to search a key element “X” in a list of ‘n’ 16-bit numbers. Adopt Binary search algorithm in your program for searching.

Program:

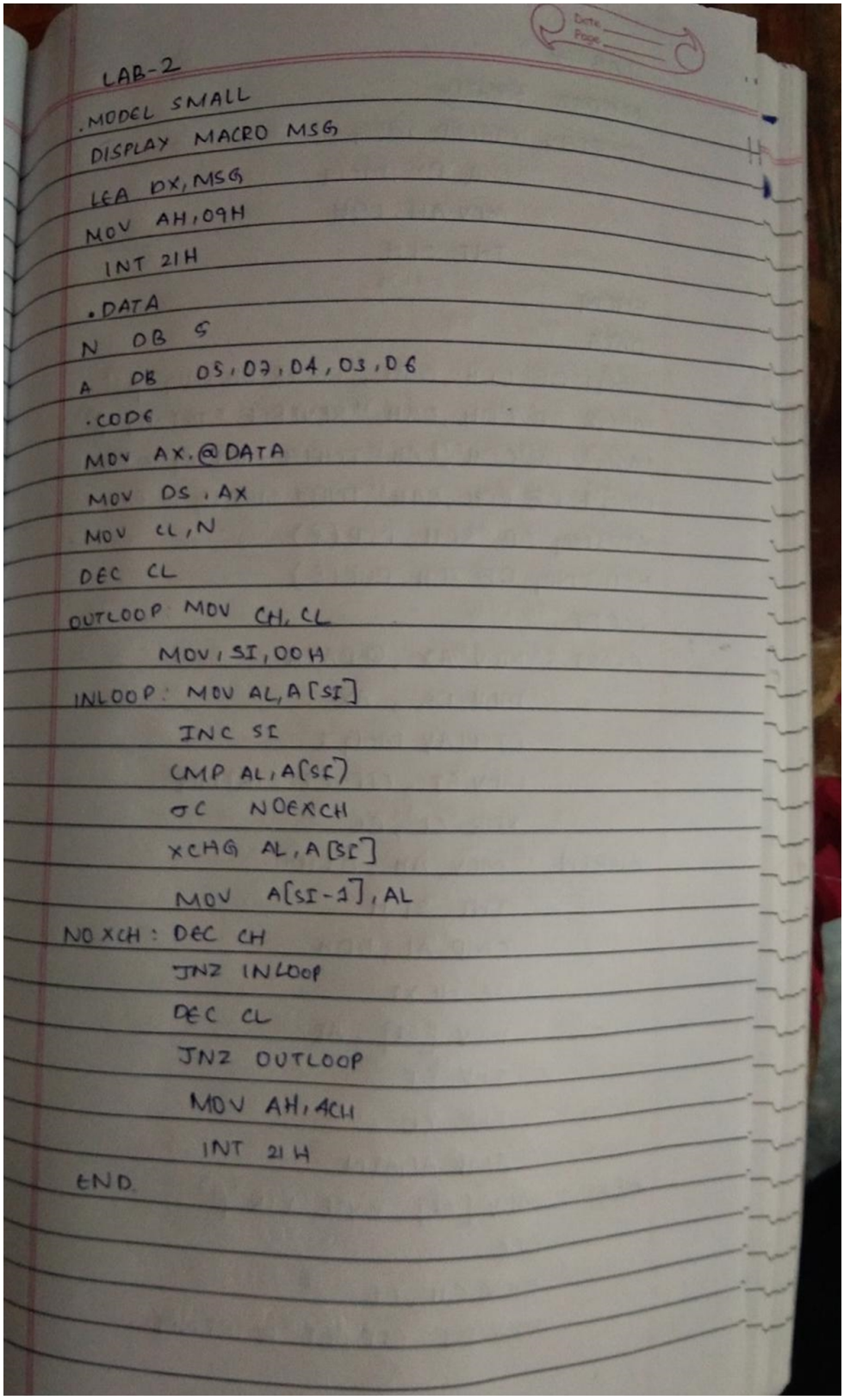


```
LAB-1
.MODEL SMALL
DISPLAY MACRO MSG
LEA DX,MSG
MOV AH,09H
INT 21H
ENDM
.DATA
LIST DB 01H,05H,07H,10H,1AH
NUMBER EQU ($-LIST)
KEY DB 021H
MSG1 DB 0DH,0AH,"ELEMENT FOUND IN THE LIST..$"
MSG1 DB 0DH,0AH,"ELEMENT NOT FOUND IN LIST..$"

.CODE
START: MOV AX,@DATA
    MOV DS,AX
    MOV CH,NUMBER-1
    MOV CL,00H
AGAIN: MOV SI, OFFSET LIST
       XOR AX,AX
       CMP CL,CH
       JE NEXT
       JNC FAILED
NEXT: MOV AL,CL
      ADD AL,[SI]
      SHR AL,01H
      MOV BL,AL
      JE SUCCESS
      JC INCLOW
      MOV CH,BL
      JMP AGAIN
```



2. Design and develop an assembly program to sort a given set of ‘n’ 16-bit numbers in ascending order. Adopt Bubble Sort algorithm to sort given elements

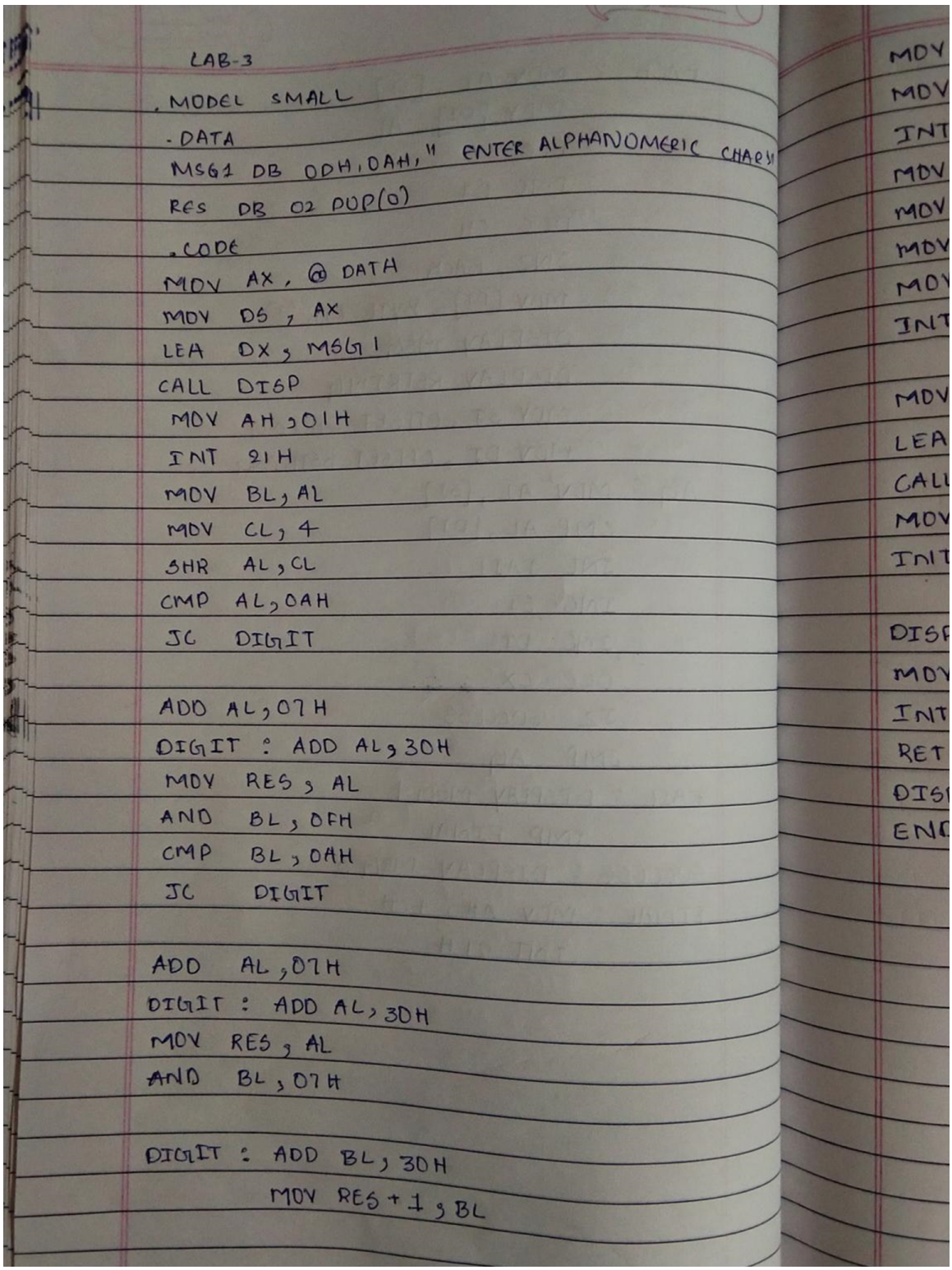
Program:



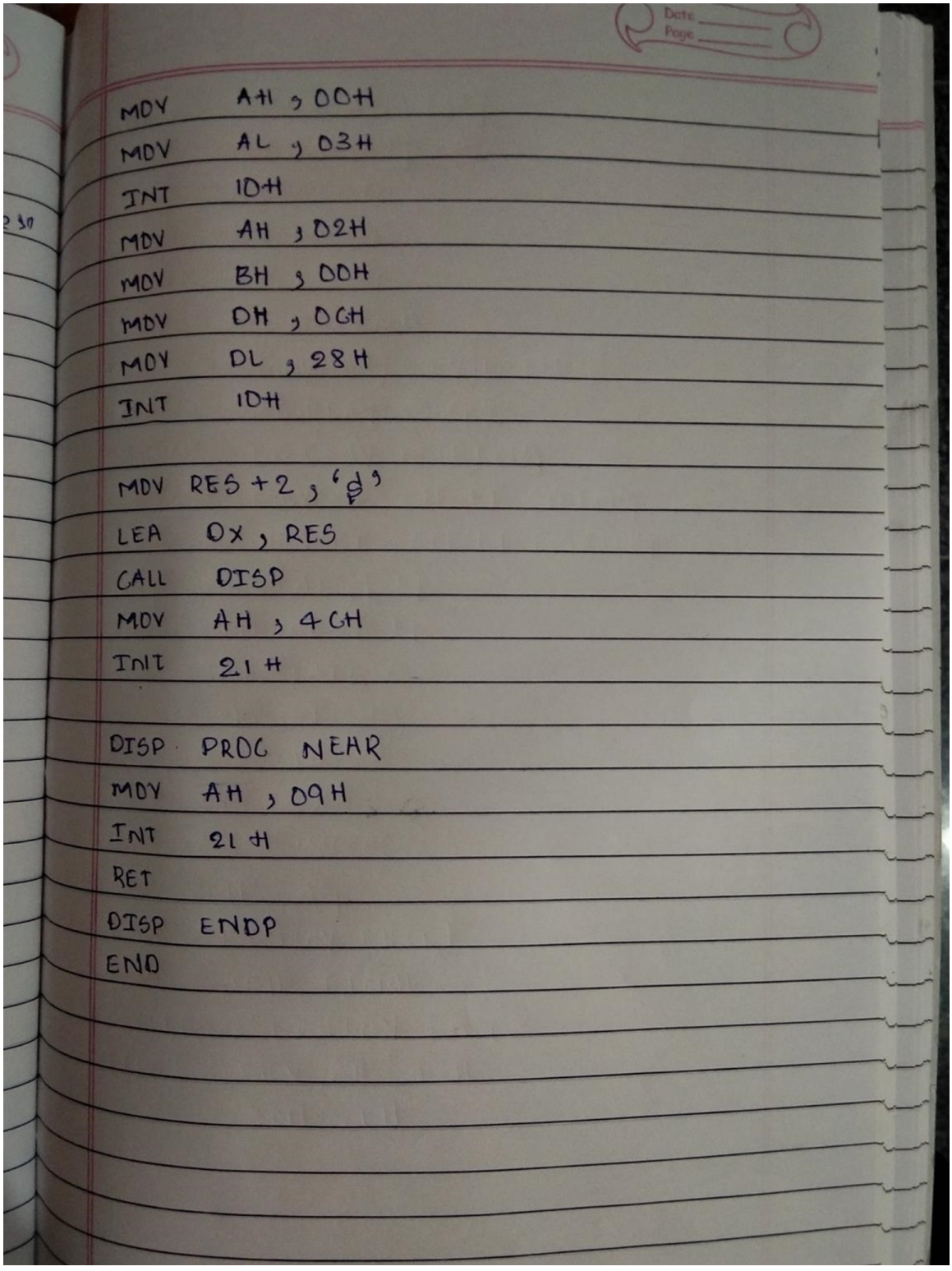
```
LAB-2
.MODEL SMALL
DISPLAY MACRO MSG
LEA DX,MSG
MOV AH,09H
INT 21H
.DATA
N DB 5
A DB 05,07,04,03,06
.CODE
MOV AX,@DATA
MOV DS,AX
MOV CL,N
DEC CL
OUTLOOP MOV CH,CL
        MOV,SI,00H
INLOOP: MOV AL,A[SI]
        INC SI
        CMP AL,A[SI]
        JC NOEXCH
        XCHG AL,A[SI]
        MOV A[SI-1],AL
NOXCH: DEC CH
        JNZ INLOOP
        DEC CL
        JNZ OUTLOOP
        MOV AH,4CH
        INT 21H
END
```

3. Read an alphanumeric character and display its equivalent ASCII code at the center of the screen.

Program:



```
LAB-3
.MODEL SMALL
.DATA
MSG1 DB 0DH,0AH," ENTER ALPHANUMERIC CHAR$"
RES DB 02 DUP(0)
.CODE
MOV AX, @DATA
MOV DS, AX
LEA DX, MSG1
CALL DISP
MOV AH,01H
INT 21H
MOV BL, AL
MOV CL, 4
SHR AL, CL
CMP AL, 0AH
JC DIGIT

ADD AL,07H
DIGIT : ADD AL, 30H
MOV RES, AL
AND BL, 0FH
CMP BL, 0AH
JC DIGIT

ADD AL,07H
DIGIT : ADD AL, 30H
MOV RES, AL
AND BL, 07H

DIGIT : ADD BL, 30H
        MOV RES+1, BL
```



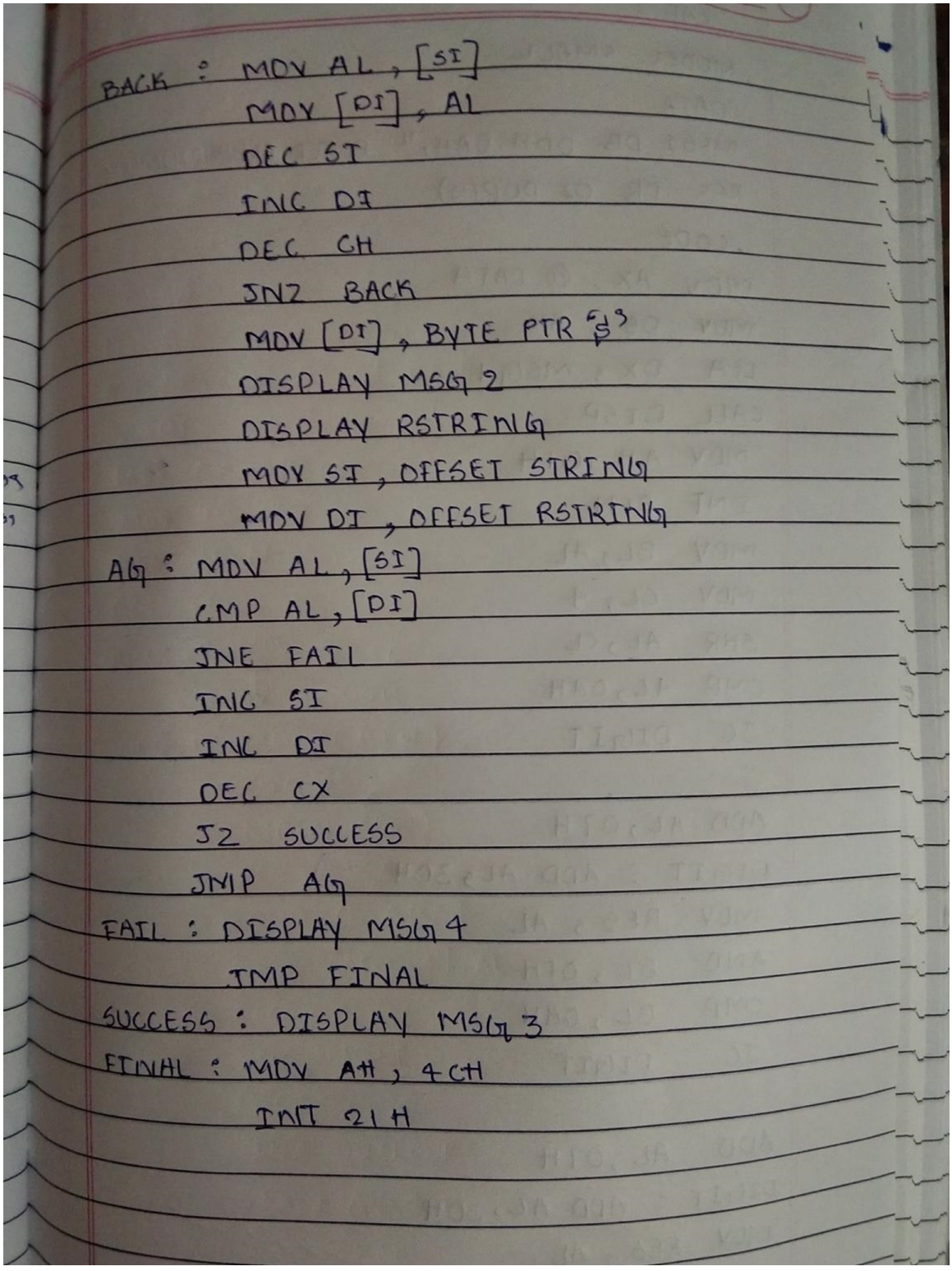
```
MOV     AH , 00H
MOV     AL , 03H
INT     10H
MOV     AH , 02H
MOV     BH , 00H
MOV     DH , 0CH
MOV     DL , 28H
INT     10H

MOV RES+2 , 'd'
LEA     DX , RES
CALL    DISP
MOV     AH , 4CH
INT     21H

DISP PROC NEAR
MOV     AH , 09H
INT     21H
RET
DISP ENDP
END
```

4. Reverse a given string and check whether it is a palindrome or not

Program:

LAB - 4

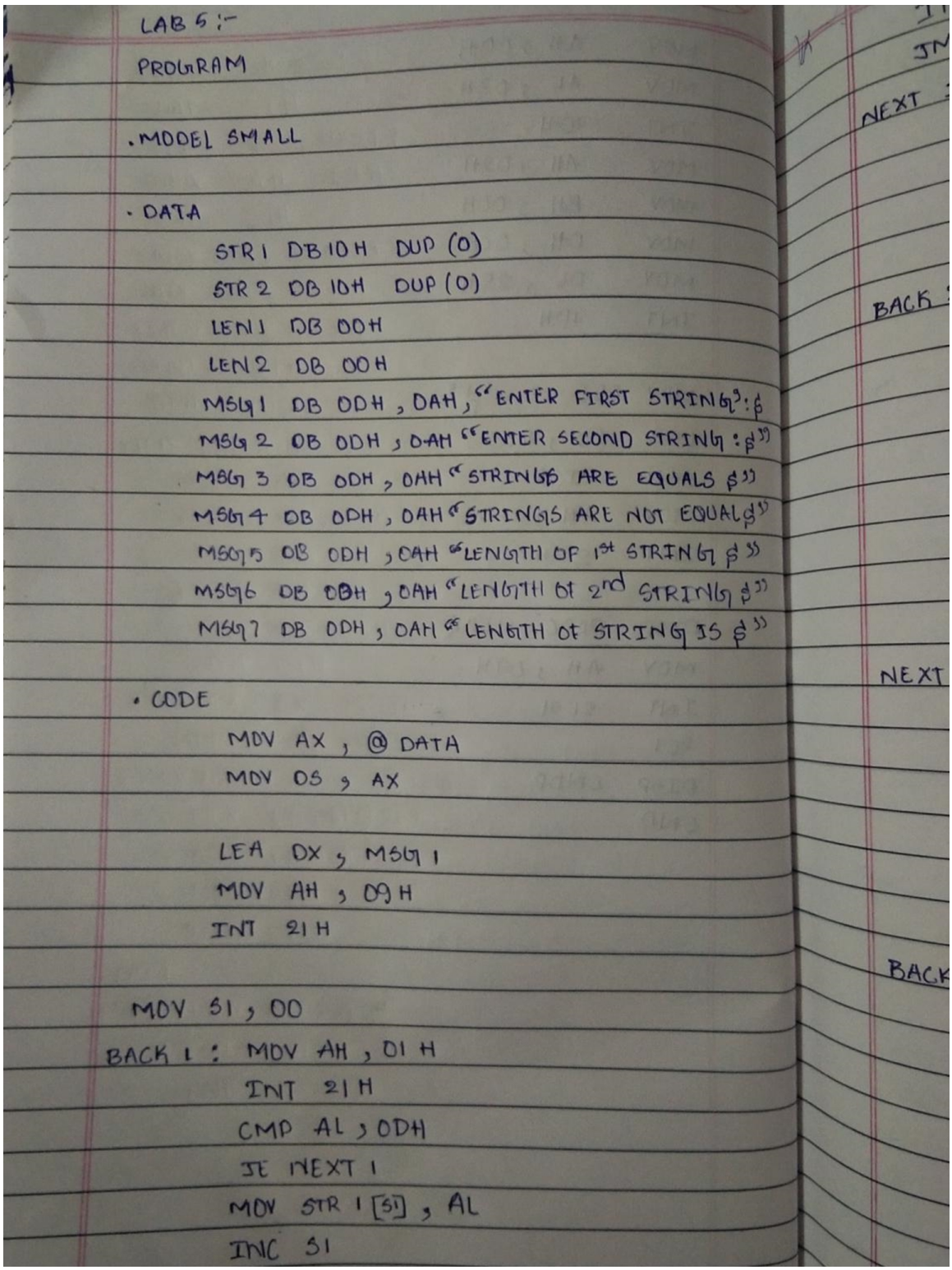
```
.MODEL SMALL
DISPLAY MACRO MSG
        LEA DX, MSG
        MOV AH, 09H
        INT 21H
ENDM
DATA
MSG1 DB 0DH, 0AH, "ENTER STRING: $"
MSG2 DB 0DH, 0AH "REVERSE STRING: $"
MSG3 DB 0DH, 0AH "INPUT STRING IS Palindrome: $"
MSG4 DB 0DH, 0AH "INPUT STRING IS PALINDROME $"
STRING DB 80H DUP(?)
RSTRING DB 80H DUP(?)
.CODE
START : MOV AX, @DATA
        MOV DS, AX
        DISPLAY MSG1
        MOV SI, OFFSET STRING
        XOR CL, CL
AGAIN : MOV AH, 01H
        INT 21H
        CMP AL, 0DH
        JE NEXT
        MOV [SI], AL
        INC SI
        INC CL
        JMP AGAIN
NEXT : MOV [SI], BYTE PTR '$'
       DEC
       MOV CH, CL
       MOV DI, OFF
```



```
BACK : MOV AL, [SI]
       MOV [DI], AL
       DEC SI
       INC DI
       DEC CH
       JNZ BACK
       MOV [DI], BYTE PTR '$'
       DISPLAY MSG 2
       DISPLAY RSTRING
       MOV SI, OFFSET STRING
       MOV DI, OFFSET RSTRING
AG : MOV AL, [SI]
     CMP AL, [DI]
     JNE FAIL
     INC SI
     INC DI
     DEC CX
     JZ SUCCESS
     JMP AG
FAIL : DISPLAY MSG 4
       JMP FINAL
SUCCESS : DISPLAY MSG 3
FINAL : MOV AH, 4CH
        INT 21H
```

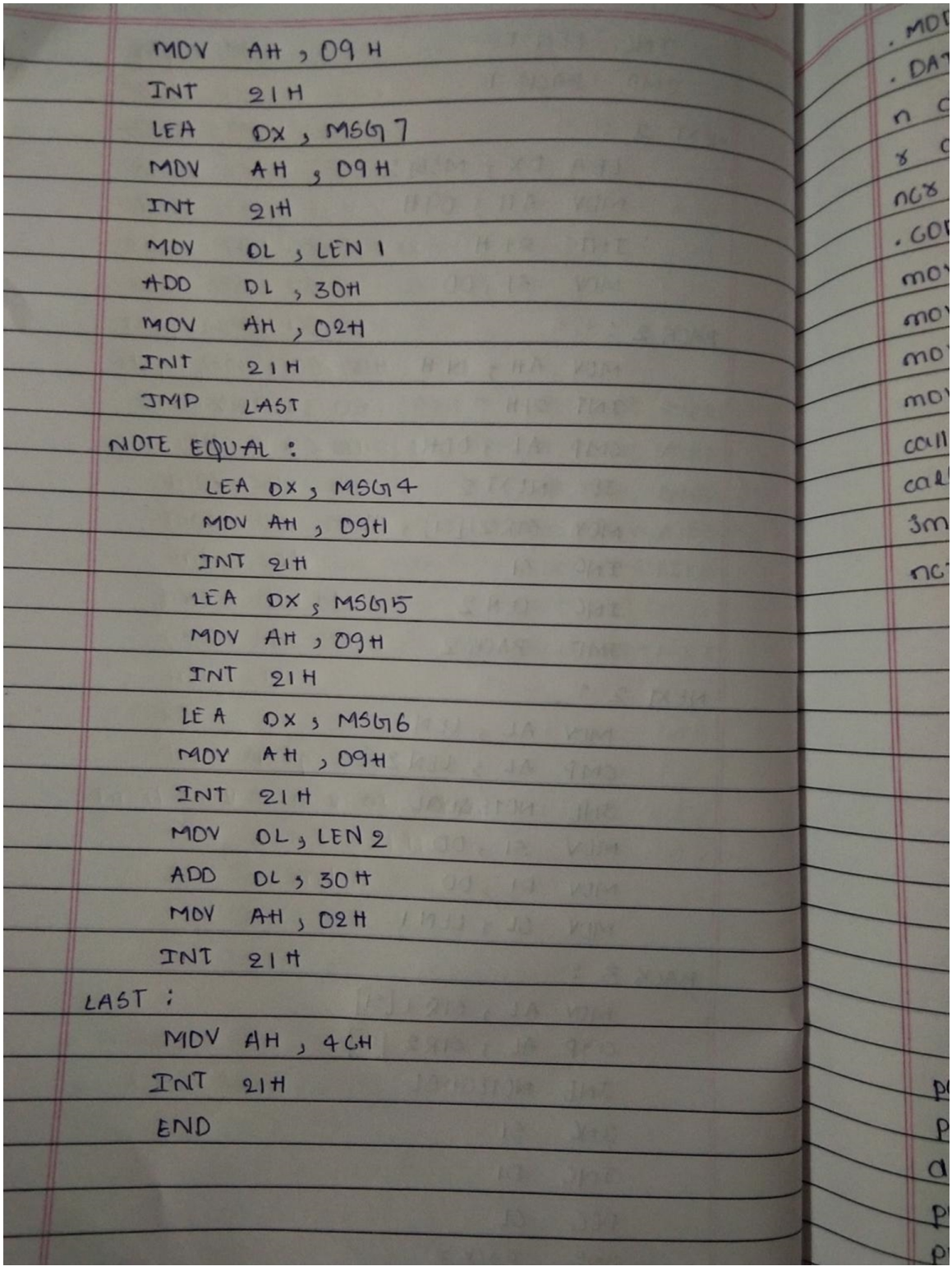
5.Read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriate messages. Also display the length of the stored strings.

Program:



```
LAB 5 :-
PROGRAM

.MODEL SMALL

.DATA
        STR1 DB 10H DUP (0)
        STR2 DB 10H DUP (0)
        LEN1 DB 00H
        LEN2 DB 00H
        MSG1 DB 0DH, 0AH, "ENTER FIRST STRING: $"
        MSG2 DB 0DH, 0AH "ENTER SECOND STRING: $"
        MSG3 DB 0DH, 0AH "STRINGS ARE EQUALS $"
        MSG4 DB 0DH, 0AH "STRINGS ARE NOT EQUAL$"
        MSG5 DB 0DH, 0AH "LENGTH OF 1st STRING $"
        MSG6 DB 0DH, 0AH "LENGTH OF 2nd STRING $"
        MSG7 DB 0DH, 0AH "LENGTH OF STRING IS $"

.CODE
        MOV AX, @DATA
        MOV DS, AX

        LEA DX, MSG1
        MOV AH, 09H
        INT 21H

MOV SI, 00
BACK1 : MOV AH, 01H
        INT 21H
        CMP AL, 0DH
        JE NEXT1
        MOV STR1[SI], AL
        INC SI
```

```
        INC   LEN 1
        JMP   BACK 1
NEXT 1 :
        LEA   DX , MSG 2
        MOV   AH , 09H
        INT   21 H
        MOV   SI , 00
BACK 2 :
        MOV   AH , 01 H
        INT   21H
        CMP   AL , 00H
        JE    NEXT 2
        MOV   STR2 [SI] , AL
        INC   SI
        INC   LEN 2
        JMP   BACK 2
NEXT 2 :
        MOV   AL , LEN 1
        CMP   AL , LEN 2
        JNE   NOTEQUAL
        MOV   SI , 00
        MOV   DI , 00
        MOV   CL , LEN 1
BACK 3 :
        MOV   AL , STR 1 [SI]
        CMP   AL , STR 2 [DI]
        JNE   NOTEQUAL
        INC   SI
        INC   DI
        DEC   CL
        JNE   BACK 3
        LEA   DX , MSG 3
```



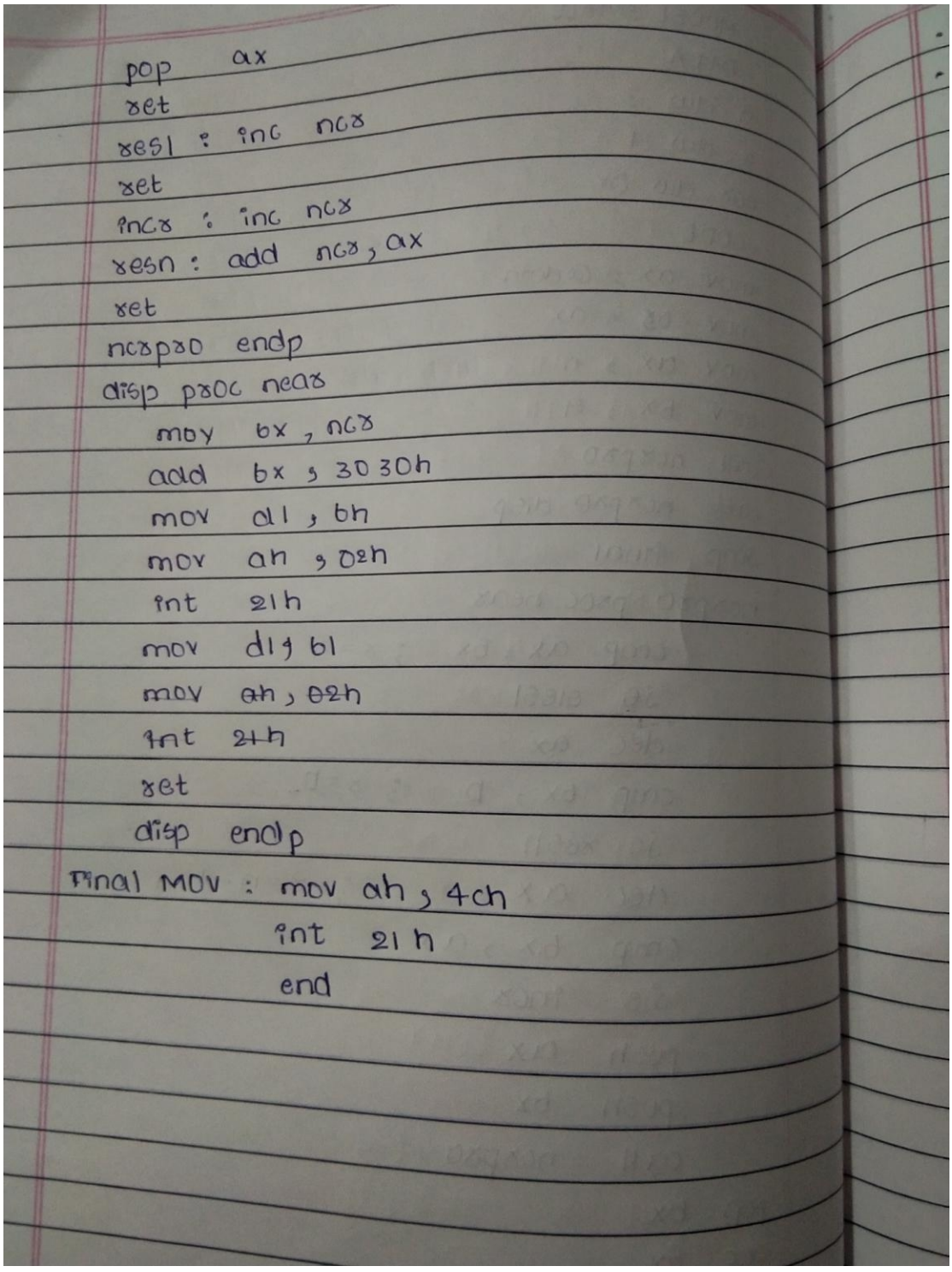
```
    MOV  AH , 09 H
    INT  21 H
    LEA  DX , MSG7
    MOV  AH , 09 H
    INT  21H
    MOV  DL , LEN 1
    ADD  DL , 30H
    MOV  AH , 02H
    INT  21 H
    JMP  LAST
NOTE EQUAL :
        LEA DX , MSG4
        MOV AH , 09H
        INT 21H
        LEA  DX , MSG5
        MOV AH , 09H
        INT  21 H
        LE A  DX , MSG6
        MOV  AH , 09H
        INT  21 H
        MOV  DL , LEN 2
        ADD  DL , 30 H
        MOV  AH , 02 H
        INT  21 H
LAST :
        MOV AH , 4CH
        INT  21H
        END
```

6.Develop an assembly language program to compute nCr using recursive procedure. Assume that ‘n’ and ‘r’ are non-negative integers

Program:

LAB 6

```
.MODEL SMALL
.DATA
n dw 3
r dw 1
ncr dw 0
.CODE
mov ax, @data
mov ds, ax
mov ax, n
mov bx, r
call ncrproc
call ncrproc disp
jmp final
ncrproc proc near
        cmp ax, bx   ; r = n
        je equal
        dec ax
        cmp bx, 0    ; r = 0
        je zero
        dec ax       ; r = n-1
        cmp bx, ax
        je incr
        push ax
        push bx
        call ncrproc
pop bx
pop bx
dec bx
push ax
push bx
call ncrproc
pop bx
```

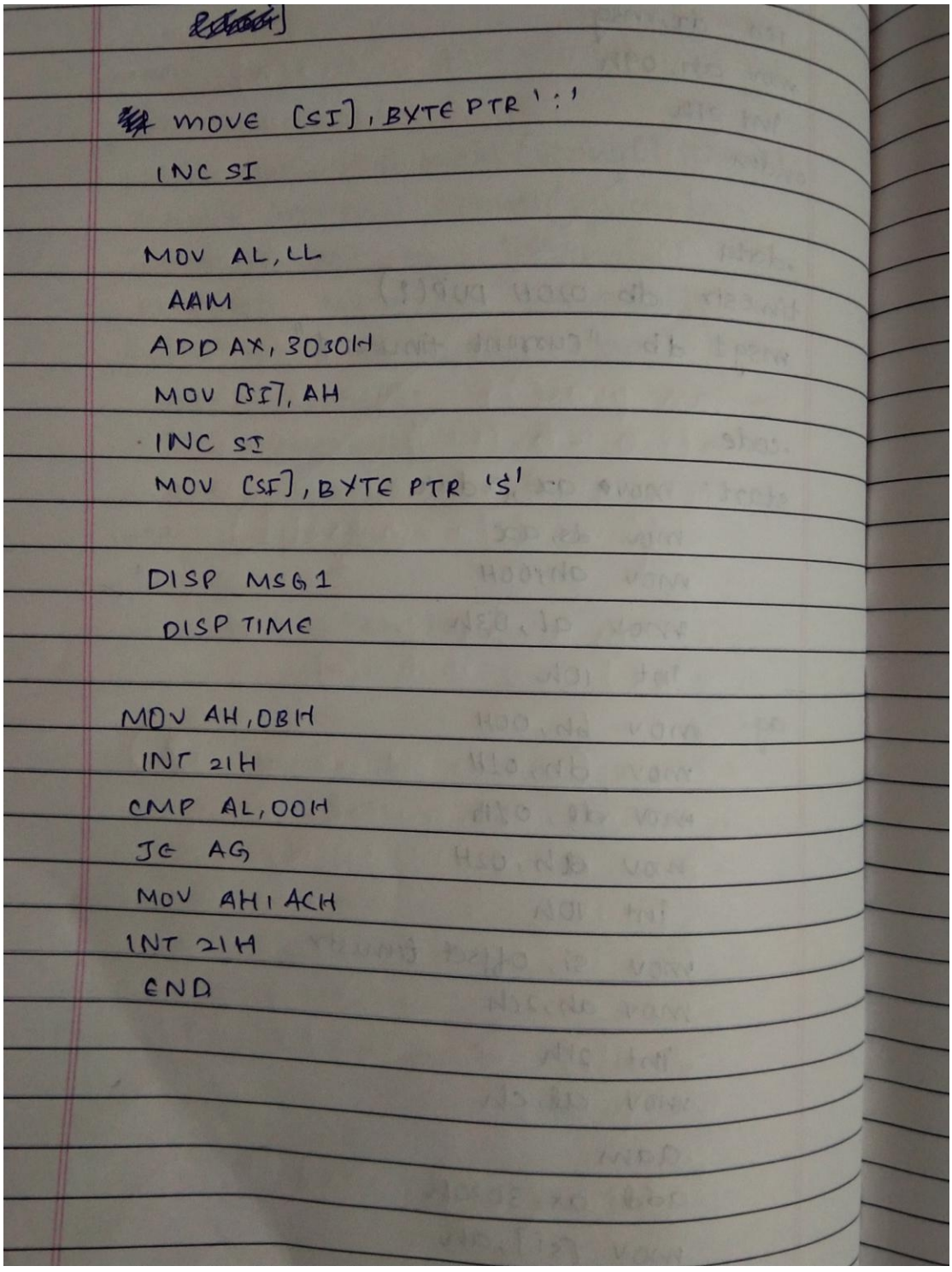


```
pop   ax
ret
resl : inc  ncx
ret
incx : inc ncx
resn : add  ncx, ax
ret
ncxpx0 endp
disp proc near
    mov  bx , ncx
    add  bx , 3030h
    mov  dl , bh
    mov  ah , 02h
    int  21h
    mov  dl, bl
    mov  ah, 02h
    int  21h
    ret
    disp endp
final MOV : mov ah, 4ch
            int  21 h
            end
```

7. Read the current time from the system and display it in the standard format on the screen

LAB 7.

```
.model small
display macro msg
 lea  dx,msg
 mov  ah, 09h
  int 21h
endm

.data
timestr  db 020H DUP(?)
 msg1 db  "current time :: $"

.code
start:  move ax ,@data
        mov  ds, ax
        mov  ah,00H
        mov  al, 03h
        int  10h
ag:     mov  bh, 00H
        mov  dh, 01H
        mov  dl, 01h
        mov  ah, 02H
        int  10h
        mov  si, offset timustr
        mov  ah,2ch
        int  21h
        mov  al, ch
        aam
        add  ax, 3030h
        mov  [si], ah
        inc  si
        mov  [si], al
        inc  si
```

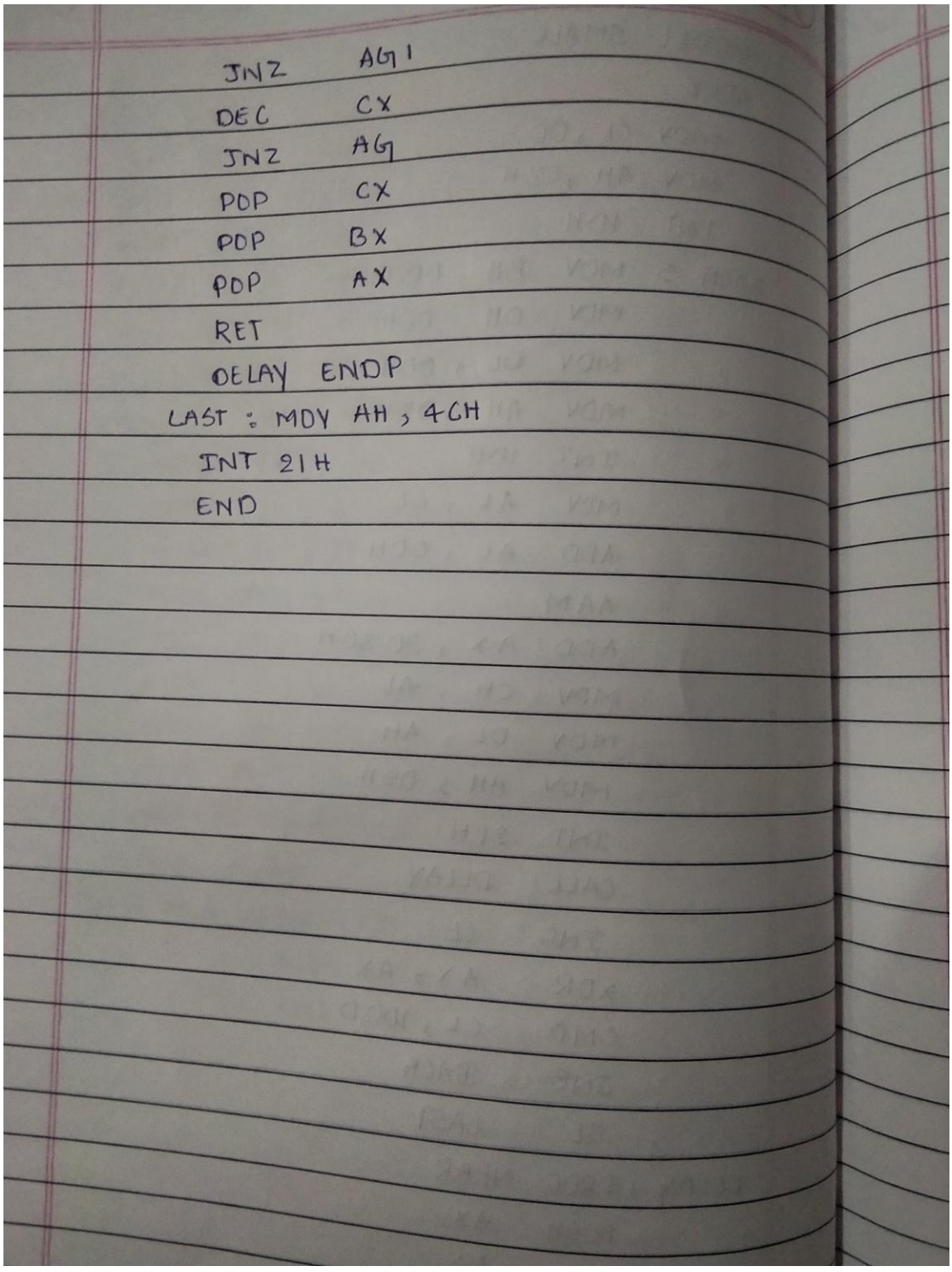


```
MOVE [SI], BYTE PTR ':'
INC SI

MOV AL, CL
AAM
ADD AX, 3030H
MOV [SI], AH
INC SI
MOV [SI], BYTE PTR 'S'

DISP MSG1
DISP TIME

MOV AH, 0BH
INT 21H
CMP AL, 00H
JE AG
MOV AH, ACH
INT 21H
END
```

8. Write a program to simulate a Decimal Up-counter to display 00-99

LAB 8

```
.MODEL SMALL
.CODE
    MOV CL, 00
    MOV AH, 03H
    INT 10H
BACK: MOV BH, 00H
    MOV DH, 00H
    MOV DL, 00H
    MOV AH, 02H
    INT 10H
    MOV AL, CL
    ADD AL, 0DH
    AAM
    ADD AX, 3030H
    MOV CH, AL
    MOV DL, AH
    MOV AH, 02H
    INT 21H
    CALL DELAY
    INC CL
    XOR AX, AX
    CMP CL, 100D
    JNE BACK
    JE LAST
DELAY PROC NEAR
    PUSH AX
    PUSH BX
    PUSH CX
    MOV CX, 00FFH
AG: MOV BX, 0FFH
AG1: NOP
    DEC BX
```

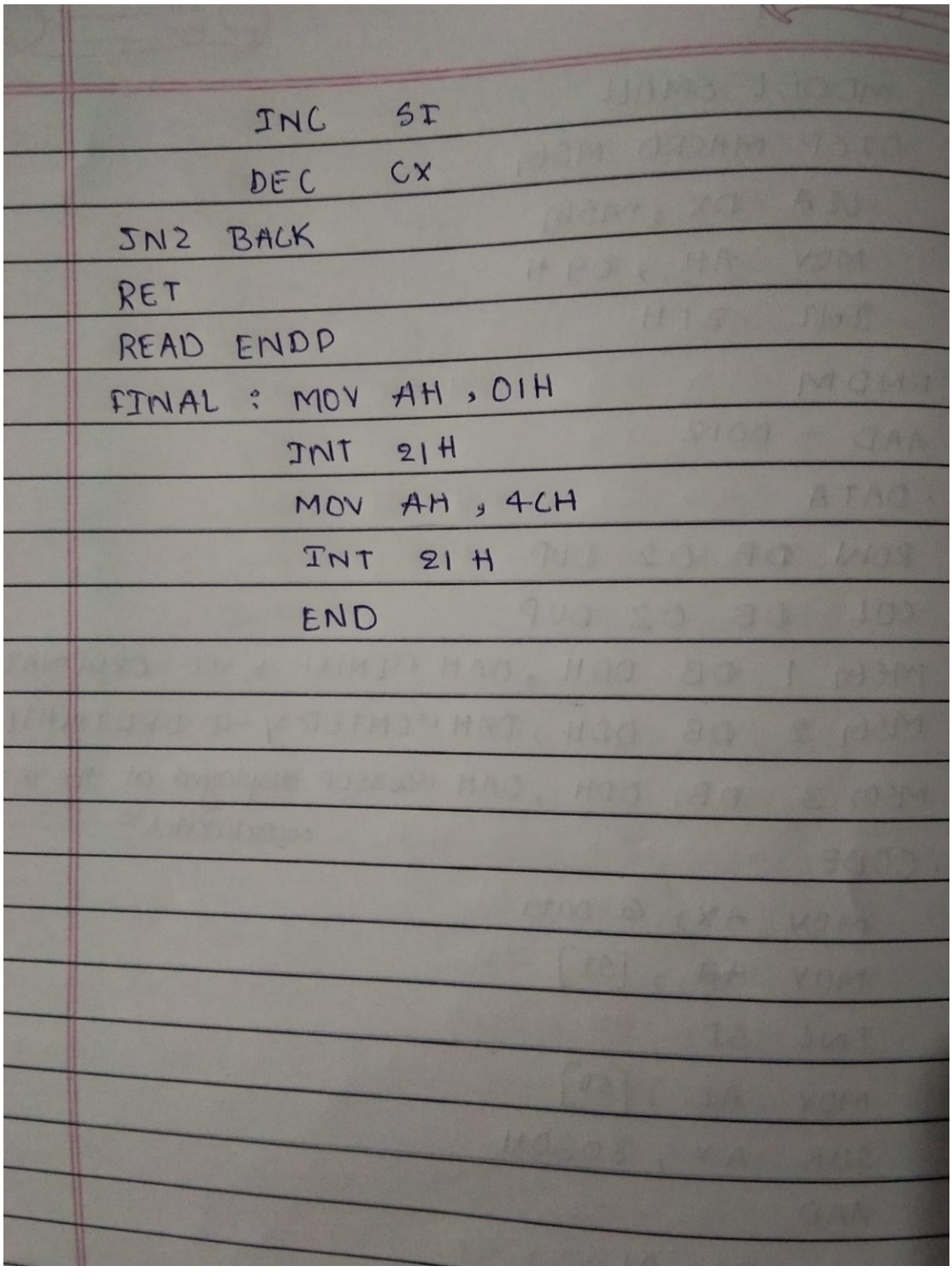


```
JNZ     AG1
DEC     CX
JNZ     AG
POP     CX
POP     BX
POP     AX
RET
DELAY ENDP
LAST : MOV AH, 4CH
INT 21H
END
```

9. Read a pair of input co-ordinates in BCD and move the cursor to the specified location on the screen.

LAD - 9

```
.MODEL SMALL
DISP MACRO MSG
  LEA DX, MSG
  MOV AH, 09H
  INT 21H
ENDM
AAD - 0012
.DATA
ROW DB 02 DUP
COL DB 02 DUP
MSG 1 DB 0DH, 0AH "ENTER X-CO-ORDINATE"
MSG 2 DB 0DH, 0AH "ENTER Y-CO-ORDINATE"
MSG 3 DB 0DH, 0AH "CURSOR displayed at the correct
                   coordinates"
.CODE
  MOV AX, @data
  MOV AH, [SI]
  INC SI
  MOV AL, [SI]
  SUB AX, 3030H
  AAD
  MOV DL, AL
  MOV AH, 00
  MOV AL, 03H
  INT 10H
  MOV AH, 02H
  INT 10H
  JMP FINAL
READ PROC NEAR
  MOV CX, 02H
BACK: MOV AH, 01H
      INT 21H
      MOV [SI], AL
```



10. Write a program to create a file (input file) and to delete an existing file

LAB,10

```
.model small

disp macro msg
 lea dx,msg
 mov ah,09H
 int 21h
endm

.data
msg1 db 0DH,0AH, "enter the file name for creation:$"
msg2 db 0DH,0AH, "File created succussfully$"
msg3 db 0DH,0AH, "creation failed$"
msg4 db 0DH,0AH, "enter file name for deletion:$"
msg5 db 0DH,0AH, "file deleted succussfully$"
msg6 db 0DH,0AH, "deletion failed$"
Fname1 DB 10 DUP(0)
Fname2 DB 10 DUP(0)

.code
 MOV AX, @data
 mov ds,AX
 disp MSG1
 MOV SI,00
back1: mov ah,01h
int 21h
 cmp al,0dh
 JE NEXT1
 MOV Fname1[SI],AL
 inc SI
 JMP BACK1
NEXT1: MOV FNAME1[SI],'$'
LEA DX, FNAME1
```

```
MOV CX,00
MOV AH,3CH
INT 21H
JC CFAIL
DISP MSG2
JMP DEL

CFAIL: DISP MSG3
DEL: DISP MSG4

MOV SI,00
BACK2: MOV AH,01H
INT 21H
CMP AL,0DH
JE NEXT2
MOV FNAME2[SI],AL
INC SI
JMP BACK2

NEXT2: MOV FNAME2[SI],'$'
LEA DX,FNAME2
MOV AH,41H
INT 21H
cmp AL,0DH
JE NEXT2
MOV FNAME2[SI],AL
INC SI
JMP BACK2

NEXT2: MOV FNAME2[SI],'$'
LEA DX,FNAME2
MOV AH,41H
```

```
INT 21H
JC DFAIL
DISP MSG5
JMP LAST
DFAIL: DISP MSG6
LAST: MOV AH, 4CH
      INT 21H
END.
```