# **SHYAMA PRASAD MUKHERJI COLLEGE FOR WOMEN**

****

**UNIVERSITY OF DELHI**

**Practical File**

**Microprocessors**

**Submitted to : Dr. Reema Thareja**

**Submitted by : Tanya Maryam**

**College Roll No. : 21/5104**

**Course : B.Sc. Computer Science(H)**

**Semester : V**

**Year : 3rd**

**Date : 30th November, 2023**

**Q1)Write a program to implement Linear Search.**

*; LINEAR SEARCH*

.MODEL SMALL

.DATA

ARR DB 9 DUP(?)

MES1 DB 13,10,10, "ENTER THE ARRAY ELEMENTS : $"

MES2 DB 13,10,10, "ENTER THE VALUE TO BE SEARCHED : $"

MES3 DB 13,10,10, "VALUE FOUND AT LOCATION : $"

MES4 DB 13,10,10, "VALUE NOT PRESENT IN THE ARRAY $"

.CODE

.STARTUP

MOV AH,9

MOV DX, OFFSET MES1

INT 21H

MOV SI,0

MOV CX,9

L1:

MOV AH,1

INT 21H

SUB AL,30H

MOV ARR[SI],AL

INC SI

LOOP L1

MOV AH,9

MOV DX, OFFSET MES2

INT 21H

MOV AH,1

INT 21H

SUB AL,30H

MOV SI,0

MOV CX,9

L2:

CMP AL,ARR[SI]

JZ EQUAL

INC SI

LOOP L2

MOV DX,OFFSET MES4

MOV AH,9

INT 21H

JMP STOP

EQUAL:

MOV AH,9

MOV DX, OFFSET MES3

INT 21H

MOV DX,SI

ADD DL,30H

MOV AH,2

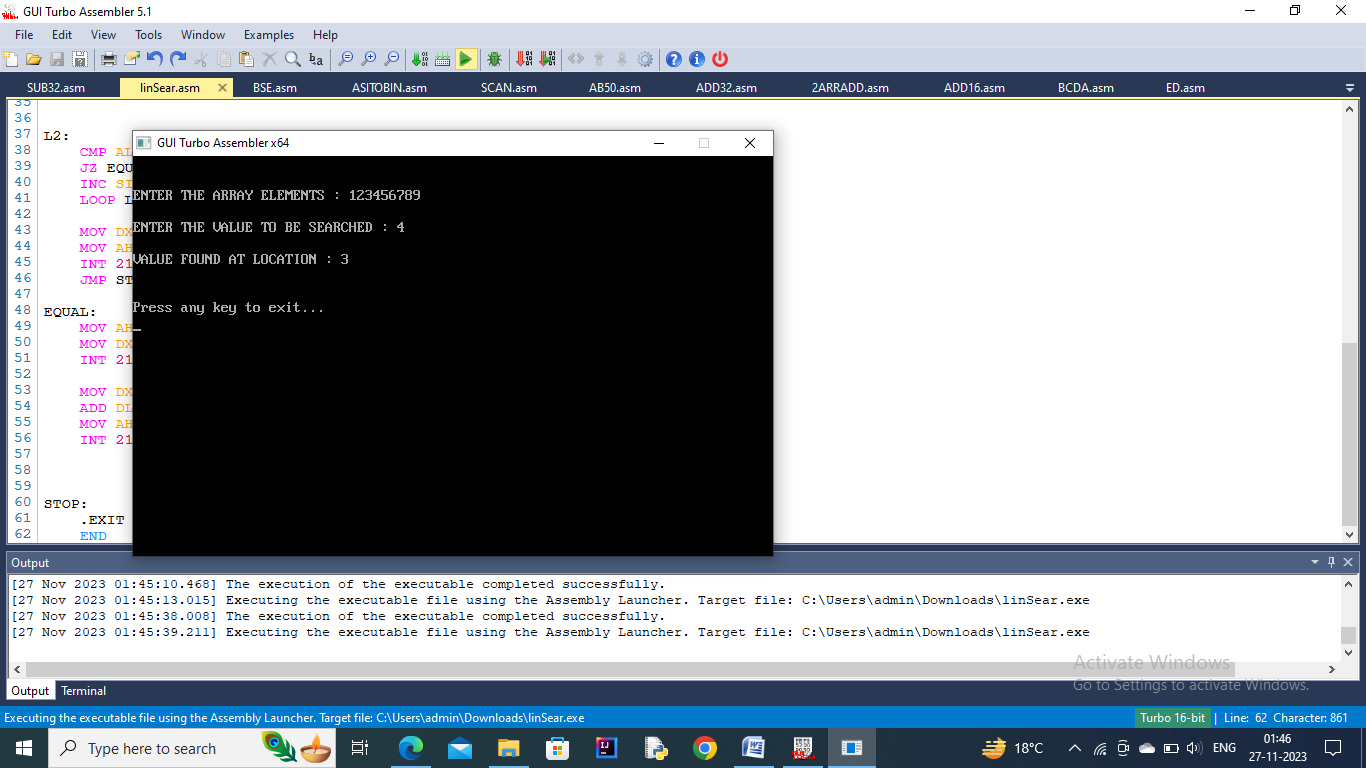
INT 21H

STOP:

.EXIT

END

**OUTPUT –**

****

**Q2) Write a program to implement Binary Search.**

*; BINARY SEARCH*

.MODEL SMALL

.DATA

ARR DB 9 DUP(?)

MES1 DB 13,10,10,"ENTER THE ARRAY ELEMENTS:$"

MES2 DB 13,10,10,"ENTER THE VALUE TO BE SEARCHED :$"

MES3 DB 13,10,10,"VALUE FOUND AT LOCATION :$"

MES4 DB 13,10,10,"VALUE NOT PRESENT IN THE ARRAY :$"

*;BEG DB ?*

*;END DB ?*

NUM DB ?

.CODE

.STARTUP

MOV AH,9

MOV DX,OFFSET MES1

INT 21H

MOV SI,0

MOV CX,9

L1:

MOV AH,1

INT 21H

SUB AL,30H

MOV ARR[SI],AL

INC SI

LOOP L1

MOV AH,9

MOV DX,OFFSET MES2

INT 21H

MOV AH,1

INT 21H

SUB AL,30H

MOV NUM,AL

MOV CH,2

MOV BL,0 *; START INDEX*

MOV BH,8 *; END INDEX*

L2:

CMP BL,BH

JAE NOTTHERE

XOR AX,AX

MOV AL,BL

ADD AL,BH *; [ AL = AL+BH] AND ALSO KNOW THAT*

MOV AH,0 *; AX = AL+BH*

DIV CH *; [AL = (AL+BH)/2] AND ALSO (AX = AX/2) SO HERE IS MID VALUE*

MOV AH,0

MOV SI,AX *; SI = MID INDEX*

MOV CL,ARR[SI]

CMP NUM,CL *; NUM IS MID*

JA GREATER

JE EQUAL

DEC AL

MOV BH,AL

JMP NEXT

GREATER:

INC AL

MOV BL,AL

NEXT: LOOP L2

NOTTHERE:

MOV DX,OFFSET MES4

MOV AH,9

INT 21H

JMP STOP

EQUAL:

MOV DX,OFFSET MES3

MOV AH,9

INT 21H

MOV DX,SI

ADD DX,30H

MOV AH,2

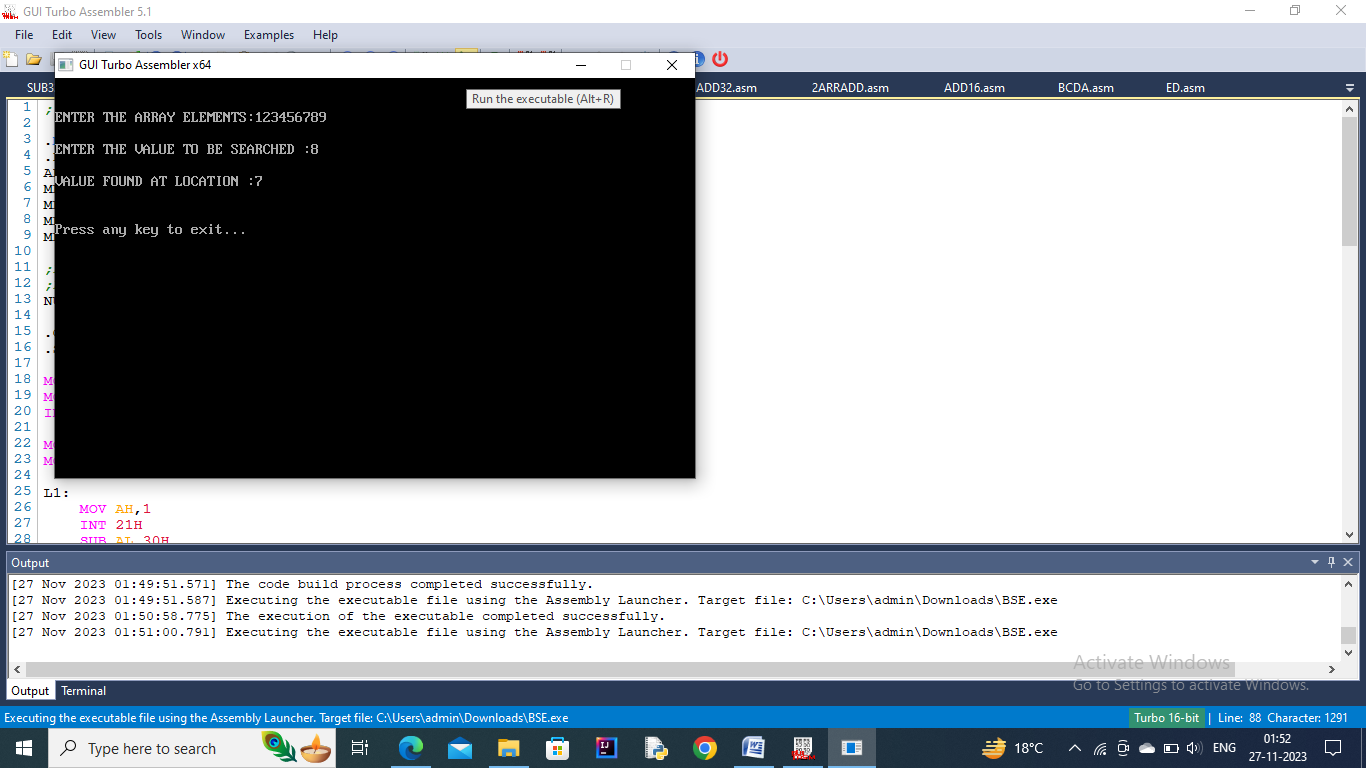
INT 21H

STOP:

.EXIT

END

**OUTPUT –**



**Q3) Write a program to add two arrays.**

*; ADDITION OF TWO ARRAY*

.MODEL SMALL

.DATA

ARR1 DB 9 DUP(?)

ARR2 DB 9 DUP(?)

ADDARR DB 9 DUP(?)

SUBARR DB 9 DUP(?)

AL1MES DB 13,10,10, "ENTER THE ARRAY1 ELEMENTS : $"

AL2MES DB 13,10,10, "ENTER THE ARRAY2 ELEMENTS : $"

ADDL3MES DB 13,10,10, "ADDITION TIME : $"

ADDL4MES DB 13,10,10, "ADDITION OF TWO ARRAY ARE : $"

SUBL5MES DB 13,10,10, "SUBTRACTION OF TWO ARRAY ARE : $"

SUBL6MES DB 13,10,10, "SUBTRACTION OF TWO ARRAY ARE : $"

.CODE

.STARTUP

MOV AH,9

MOV DX,OFFSET AL1MES

INT 21H

MOV SI,0

MOV CX,9

L1: *; TAKING ELEMENT IN FIRST ARRAY*

MOV AH,1

INT 21H

SUB AL,30H

MOV ARR1[SI],AL

INC SI

LOOP L1

MOV AH,9

MOV DX,OFFSET AL2MES

INT 21H

MOV SI,0

MOV CX,9

L2: *; TAKING ELEMENT OF SECOND ARRAY*

MOV AH,1

INT 21H

SUB AL,30H

MOV ARR2[SI],AL

INC SI

LOOP L2

MOV AH,9

MOV DX,OFFSET ADDL3MES

INT 21H

MOV SI,0

MOV CX,9

L3: *; HERE WE ARE DOING ADDITION OT TWO ARRAY*

MOV AL,ARR1[SI]

ADD AL,ARR2[SI]

MOV ADDARR[SI],AL

INC SI

LOOP L3

MOV AH,9

MOV DX,OFFSET ADDL4MES

INT 21H

MOV SI,0

MOV CX,9

L4: *; HERE WE PRINT THE ADDING OF TWO ARRAY*

MOV DL,ADDARR[SI]

ADD DL,30H

MOV AH,2

INT 21H

INC SI

LOOP L4

MOV AH,9

MOV DX,OFFSET SUBL5MES

INT 21H

MOV SI,0

MOV CX,9

L5: *; HERE ADDTION TIME OF TWO ARRAY PERFORM*

MOV AL,ARR1[SI]

SUB AL,ARR2[SI]

MOV SUBARR[SI],AL

INC SI

LOOP L5

MOV AH,9

MOV DX,OFFSET SUBL6MES

INT 21H

MOV SI,0

MOV CX,9

L6: *; PRINT THE SUBTRACTION OF TWO ARRAY*

MOV DL,SUBARR[SI]

ADD DL,30H

MOV AH,2H

INT 21H

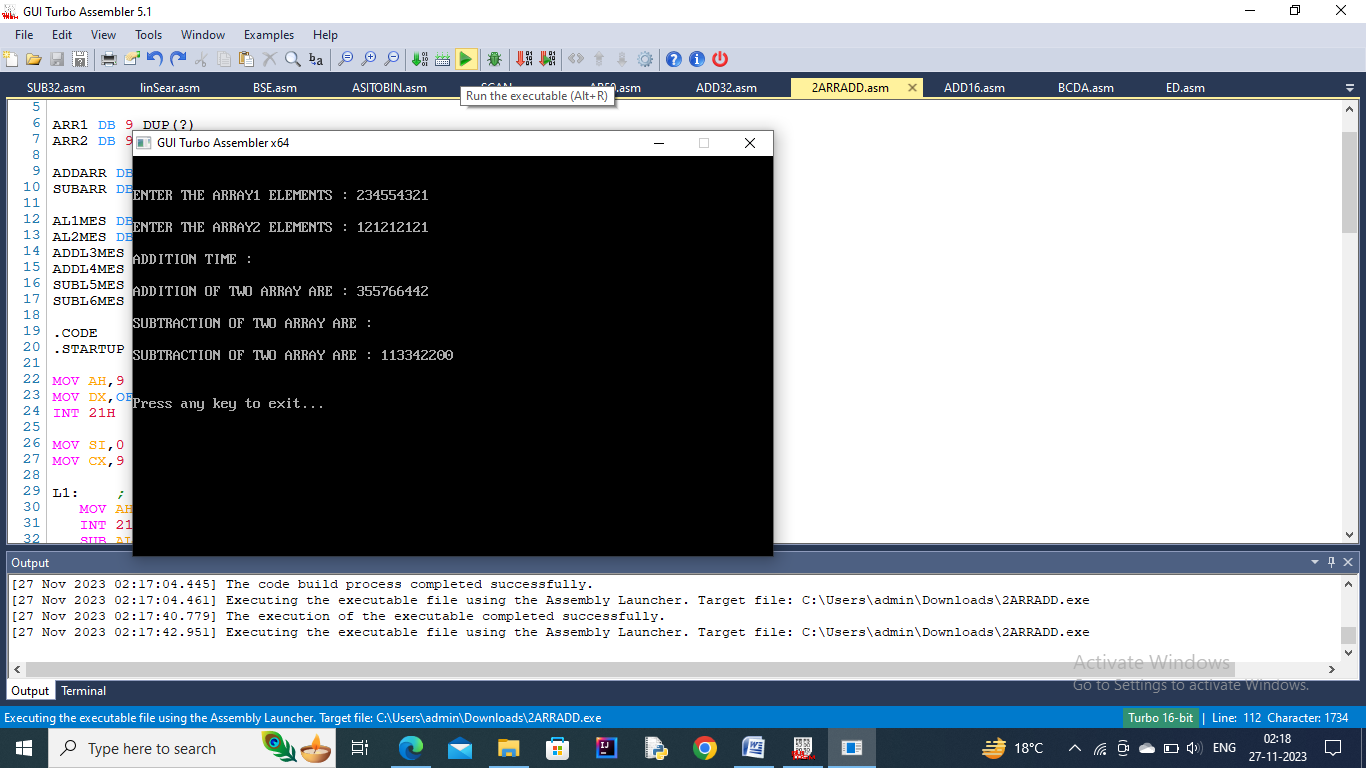
INC SI

LOOP L6

.EXIT

END

**OUTPUT –**



**Q4) Write a program to subtract two arrays.**

*; SUBTRACTION OF TWO ARRAY*

.MODEL SMALL

.DATA

ARR1 DB 9 DUP(?)

ARR2 DB 9 DUP(?)

ADDARR DB 9 DUP(?)

SUBARR DB 9 DUP(?)

AL1MES DB 13,10,10, "ENTER THE ARRAY1 ELEMENTS : $"

AL2MES DB 13,10,10, "ENTER THE ARRAY2 ELEMENTS : $"

ADDL3MES DB 13,10,10, "ADDITION TIME : $"

ADDL4MES DB 13,10,10, "ADDITION OF TWO ARRAY ARE : $"

SUBL5MES DB 13,10,10, "SUBTRACTION OF TWO ARRAY ARE : $"

SUBL6MES DB 13,10,10, "SUBTRACTION OF TWO ARRAY ARE : $"

.CODE

.STARTUP

MOV AH,9

MOV DX,OFFSET AL1MES

INT 21H

MOV SI,0

MOV CX,9

L1: *; TAKING ELEMENT IN FIRST ARRAY*

MOV AH,1

INT 21H

SUB AL,30H

MOV ARR1[SI],AL

INC SI

LOOP L1

MOV AH,9

MOV DX,OFFSET AL2MES

INT 21H

MOV SI,0

MOV CX,9

L2: *; TAKING ELEMENT OF SECOND ARRAY*

MOV AH,1

INT 21H

SUB AL,30H

MOV ARR2[SI],AL

INC SI

LOOP L2

MOV AH,9

MOV DX,OFFSET ADDL3MES

INT 21H

MOV SI,0

MOV CX,9

L3: *; HERE WE ARE DOING ADDITION OT TWO ARRAY*

MOV AL,ARR1[SI]

ADD AL,ARR2[SI]

MOV ADDARR[SI],AL

INC SI

LOOP L3

MOV AH,9

MOV DX,OFFSET ADDL4MES

INT 21H

MOV SI,0

MOV CX,9

L4: *; HERE WE PRINT THE ADDING OF TWO ARRAY*

MOV DL,ADDARR[SI]

ADD DL,30H

MOV AH,2

INT 21H

INC SI

LOOP L4

MOV AH,9

MOV DX,OFFSET SUBL5MES

INT 21H

MOV SI,0

MOV CX,9

L5: *; HERE ADDTION TIME OF TWO ARRAY PERFORM*

MOV AL,ARR1[SI]

SUB AL,ARR2[SI]

MOV SUBARR[SI],AL

INC SI

LOOP L5

MOV AH,9

MOV DX,OFFSET SUBL6MES

INT 21H

MOV SI,0

MOV CX,9

L6: *; PRINT THE SUBTRACTION OF TWO ARRAY*

MOV DL,SUBARR[SI]

ADD DL,30H

MOV AH,2H

INT 21H

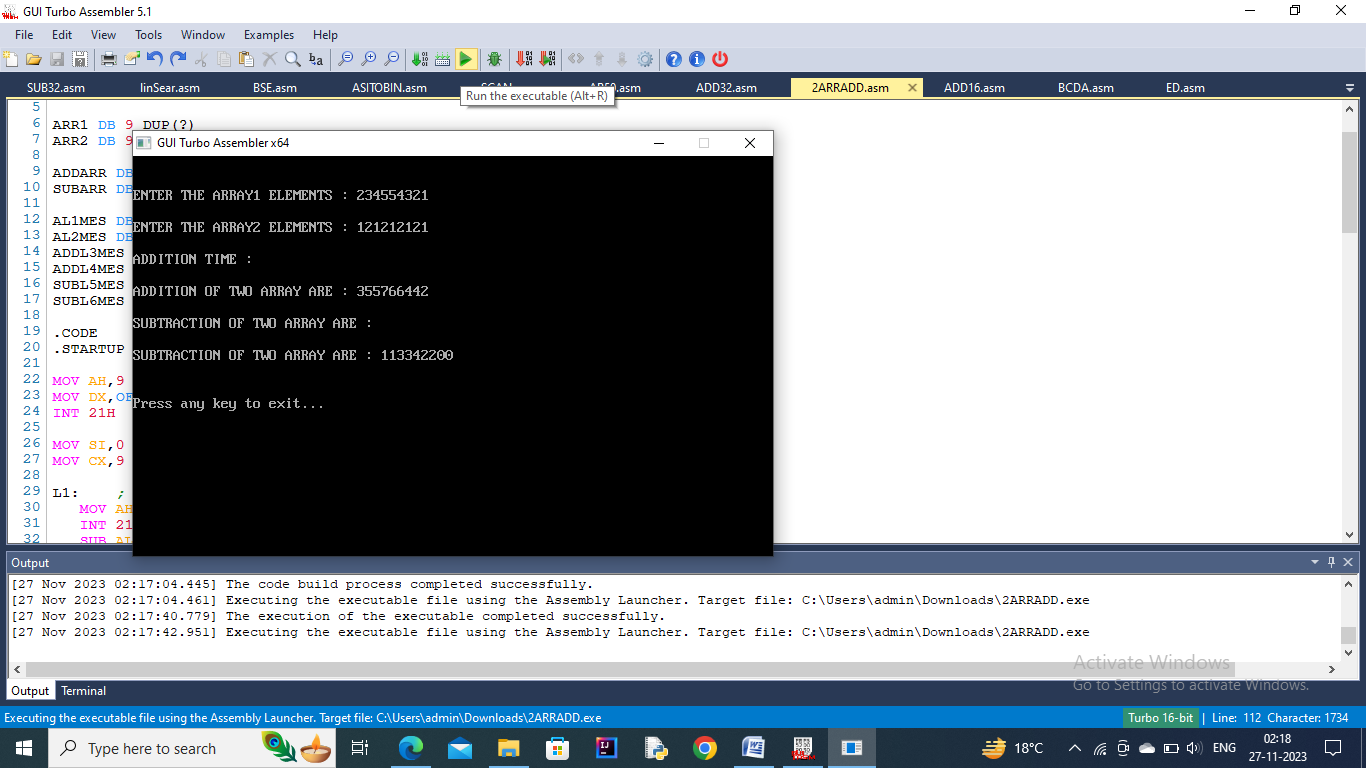
INC SI

LOOP L6

.EXIT

END

**OUTPUT –**



**Q5) Write a program to Compare two Strings.**

*; COMPARE TWO STRING*

.MODEL SMALL

.DATA

MES1 DB 13,10,10, "STRINGS ARE SAME $"

MES2 DB 13,10,10, "STRINGS ARE DIFFERENT $"

S1 DB 'HELLO$'

S2 DB 'HELLO$'

.CODE

.STARTUP

MOV DX,OFFSET MES2

MOV SI,0

MOV CX,5

L1:

MOV AL,S1[SI]

CMP AL,S2[SI]

JNZ NE

INC SI

LOOP L1

MOV DX, OFFSET MES1

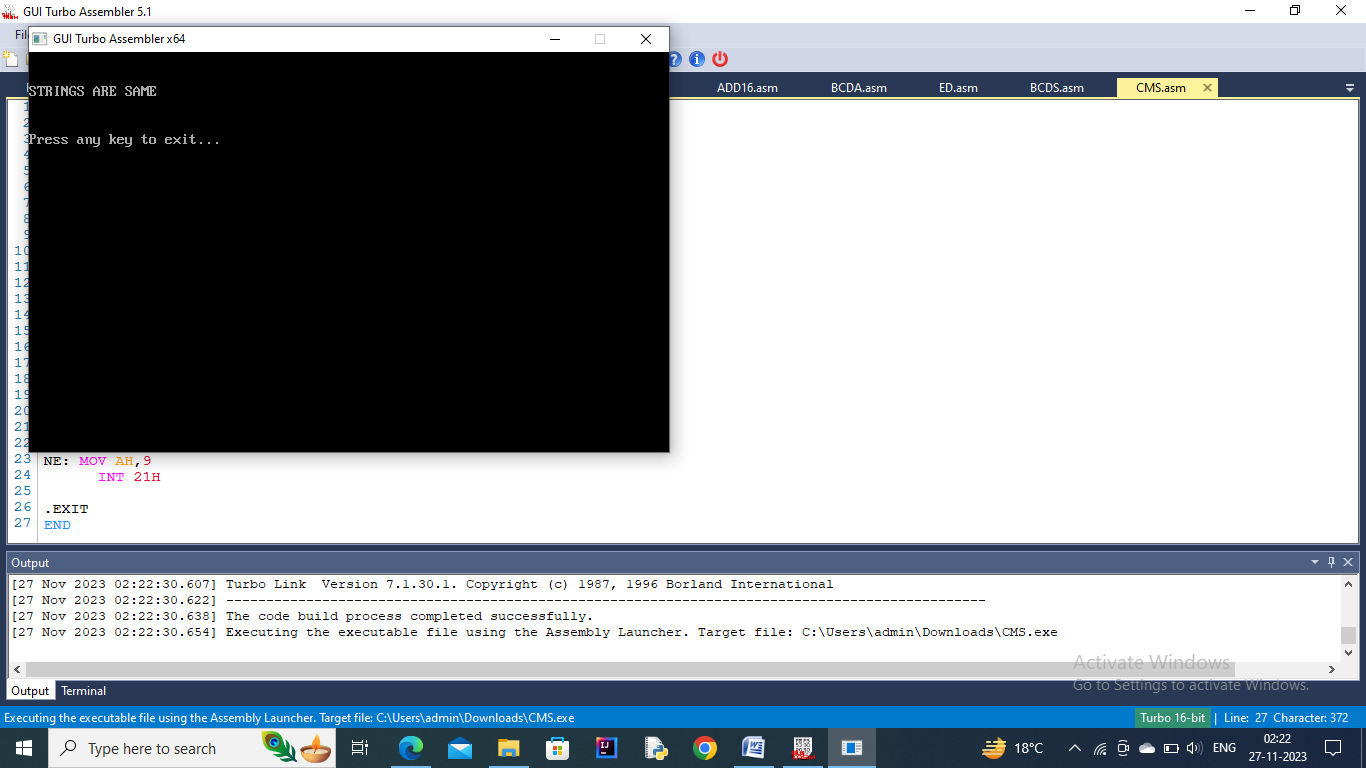
NE: MOV AH,9

INT 21H

.EXIT

END

**OUTPUT –**



**Q7) Write a program to convert a string into Upper Case.**

*; CONVERT LOWER TO UPPER STRING*

.MODEL SMALL

.DATA

MES1 DB 13,10,10, "THE ORIGINAL STRING IS IN LOWER STRING :$"

MES2 DB 13,10,10, "THE OUTPUT IN UPPER STRING IS:$"

S1 DB 'hello$'

s2 DB 6 DUP('$')

.CODE

.STARTUP

MOV AX,DS

MOV ES,AX

MOV SI, OFFSET S1

MOV DI, OFFSET S2

MOV CX,5

MOV DX, OFFSET MES1

MOV AH,9

INT 21H

MOV DX, OFFSET S1

MOV AH,9

INT 21H

MOV DX, OFFSET MES2

MOV AH,9

INT 21H

L1:

LODSB

SUB AL,20H

STOSB

LOOP L1

MOV AH,9

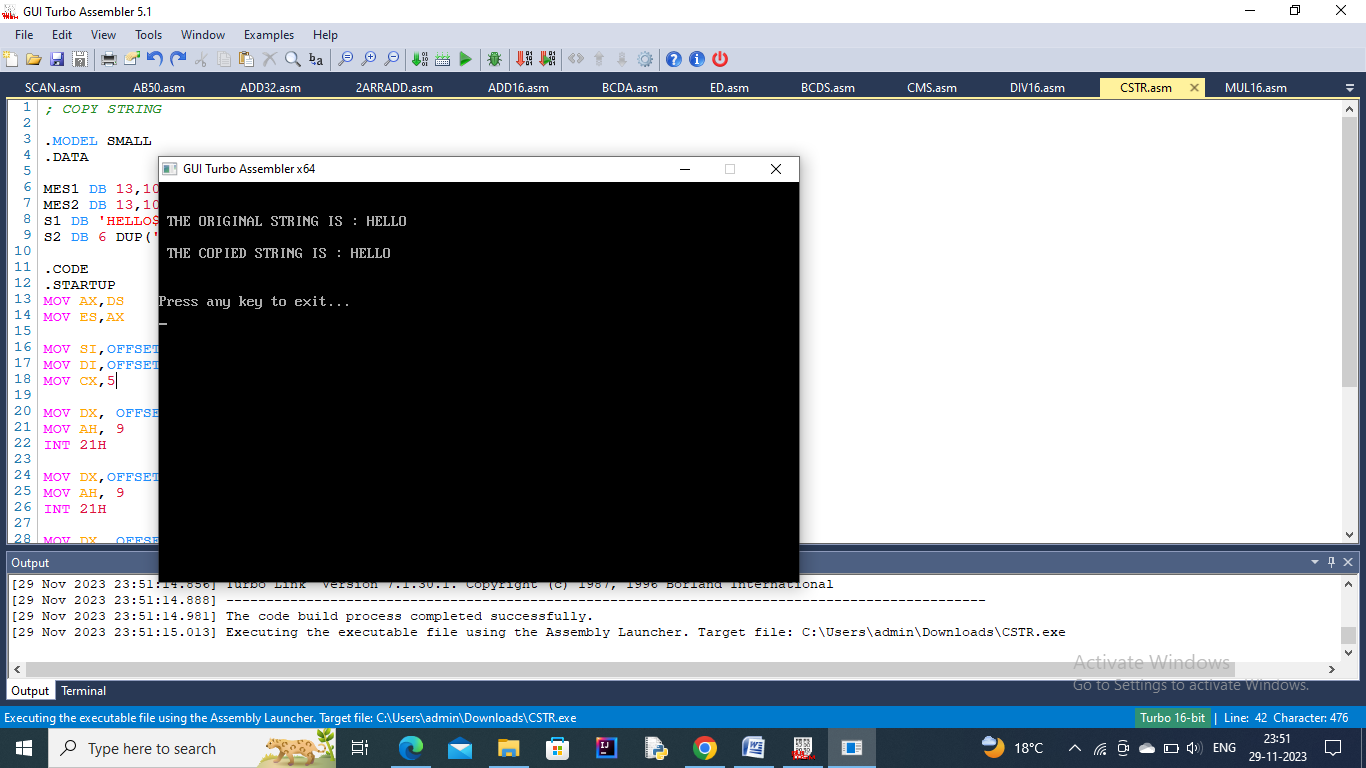
MOV DX,OFFSET S2

INT 21H

.EXIT

END

**OUTPUT –**

****

**Q8) Write a program to convert a string into lower case.**

*; CONVERT UPPER TO LOWER STRING*

.MODEL SMALL

.DATA

MES1 DB 13,10,10, "THE ORIGINAL STRING IS IN UPPER STRING :$"

MES2 DB 13,10,10, "THE OUTPUT IN LOWER STRING IS:$"

S1 DB 'HELLO$'

s2 DB 6 DUP('$')

.CODE

.STARTUP

MOV AX,DS

MOV ES,AX

MOV SI, OFFSET S1

MOV DI, OFFSET S2

MOV CX,5

MOV DX, OFFSET MES1

MOV AH,9

INT 21H

MOV DX, OFFSET S1

MOV AH,9

INT 21H

MOV DX, OFFSET MES2

MOV AH,9

INT 21H

L1:

MOVSB

ADD AL,20H

STOSB

LOOP L1

MOV AH,9

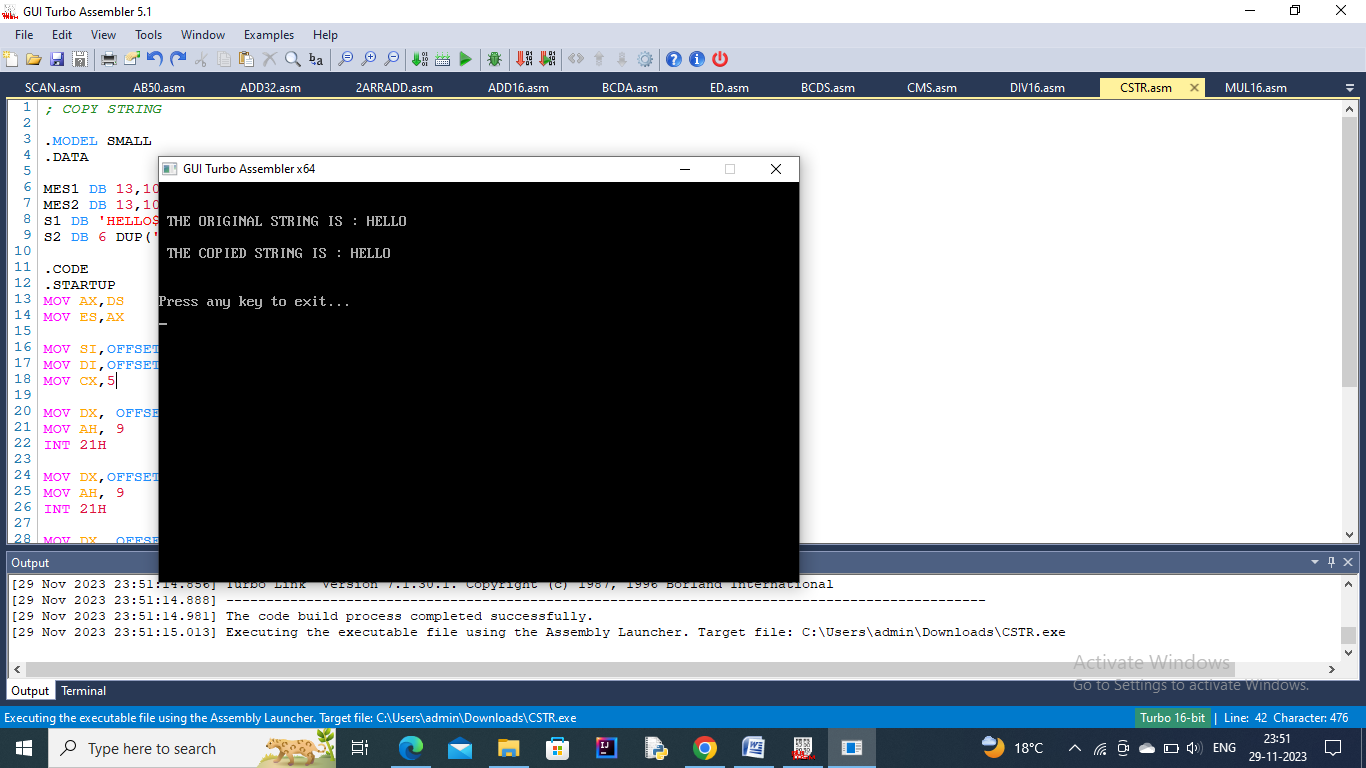
MOV DX,OFFSET S2

INT 21H

.EXIT

END

**OUTPUT –**

****

**Q9) Write a program to reverse a string.**

*; REVERSER STRING*

.MODEL SMALL

.DATA

MES1 DB 13,10,10, " THE ORIGINAL STRING IS : $ "

MES2 DB 13,10,10, " THE REVERSED STRING IS : $ "

S1 DB 'H','E','L','L','O','$'

S2 DB 6 DUP('$')

.CODE

.STARTUP

MOV DX, OFFSET MES1

MOV AH, 9

INT 21H

MOV DX,OFFSET S1

MOV AH, 9

INT 21H

MOV DX, OFFSET MES2

MOV AH, 9

INT 21H

MOV CX,5

MOV SI,4

MOV DI,0

L1:

MOV AL, S1[SI]

MOV S2[DI], AL

DEC SI

INC DI

LOOP L1

MOV AH,9

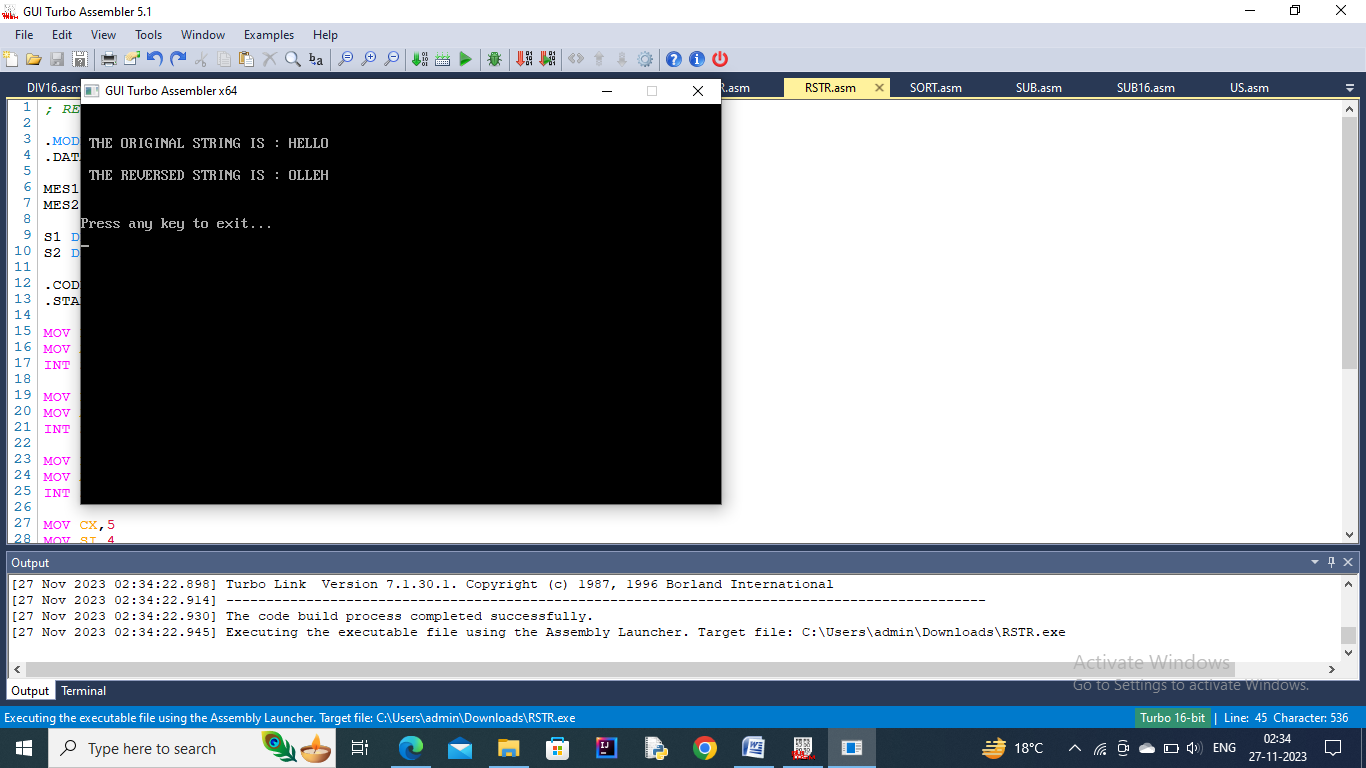
MOV DX,OFFSET S2

INT 21H

.EXIT

END

**OUTPUT –**



**Q10) Write a program to add two 32-bit binary numbers.**

*; PROGRAM TO FIND SUM OF 32 BIT NUMBERS*

.MODEL SMALL

.DATA

NUM1 DW 1234

NUM2 DW 5673

NUM3 DW 1111

NUM4 DW 2222

RES DW ?

MES3 DB 13,10,10,"SUM OF 32 BIT NUMBERS: $"

.CODE

.STARTUP

MOV DX,OFFSET MES3

MOV AH,9

INT 21H

CLC

MOV AX,NUM1

ADD AX,NUM3

CALL DISPX

MOV AX,NUM2

ADC AX,NUM4

CALL DISPX

.EXIT

DISPX PROC NEAR

MOV CX,0

MOV BX,10

DISPX1:

MOV DX,0

DIV BX

PUSH DX

INC CX

OR AX,AX

JNZ DISPX1

DISPX2:

POP DX

MOV AH,2

ADD DL,30H

INT 21H

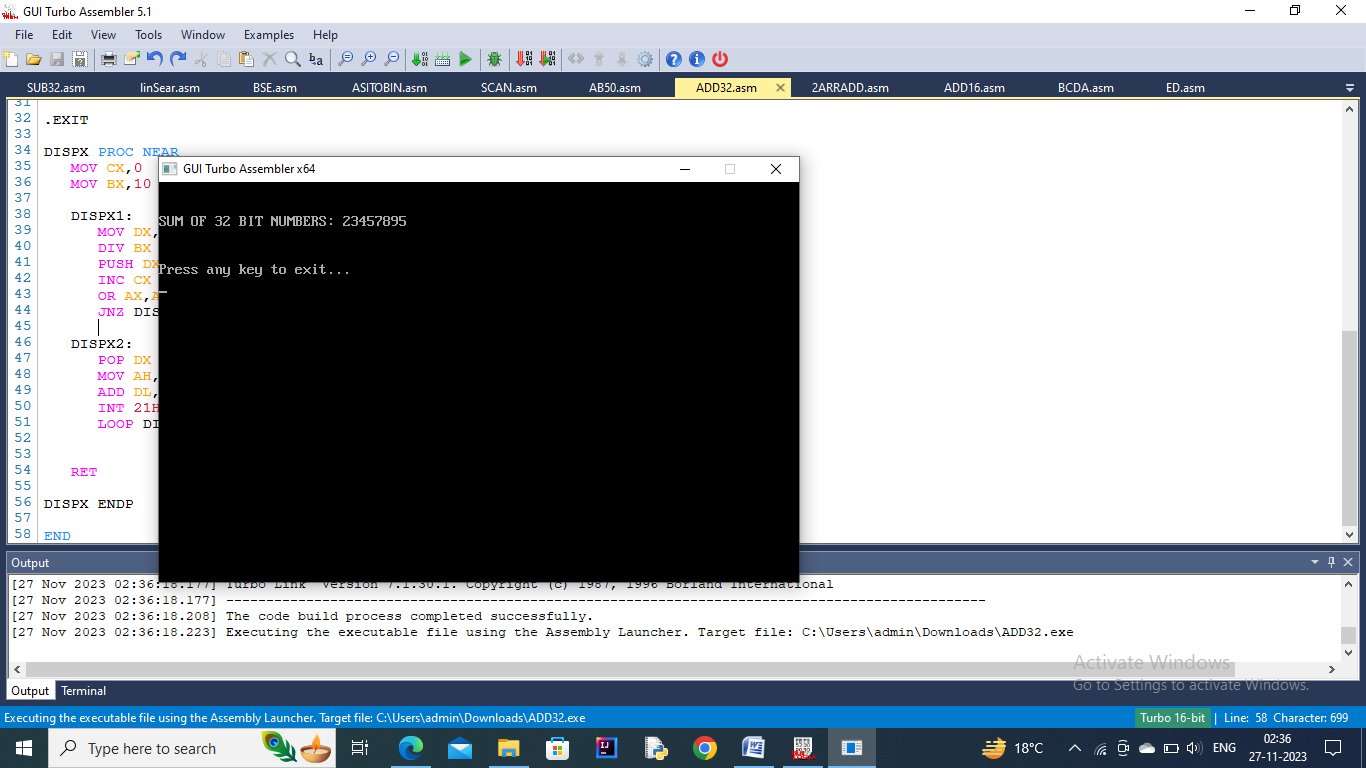
LOOP DISPX2

RET

DISPX ENDP

END

**OUTPUT –**



**Q11) Write a program to subtract two 32-bit binary numbers.**

*; PROGRAM TO FIND SUBTRACTION OF 32 BIT NUMBERS*

.MODEL SMALL

.DATA

NUM1 DW 1234

NUM2 DW 5673

NUM3 DW 1111

NUM4 DW 2222

RES DW ?

MES3 DB 13,10,10,"DIFFERENCE OF 32 BIT NUMBERS: $"

.CODE

.STARTUP

MOV DX,OFFSET MES3

MOV AH,9

INT 21H

CLC

MOV AX,NUM1

SUB AX,NUM3

CALL DISPX

MOV AX,NUM2

SUB AX,NUM4

CALL DISPX

.EXIT

DISPX PROC NEAR

MOV CX,0

MOV BX,10

DISPX1:

MOV DX,0

DIV BX

PUSH DX

INC CX

OR AX,AX

JNZ DISPX1

DISPX2:

POP DX

MOV AH,2

ADD DL,30H

INT 21H

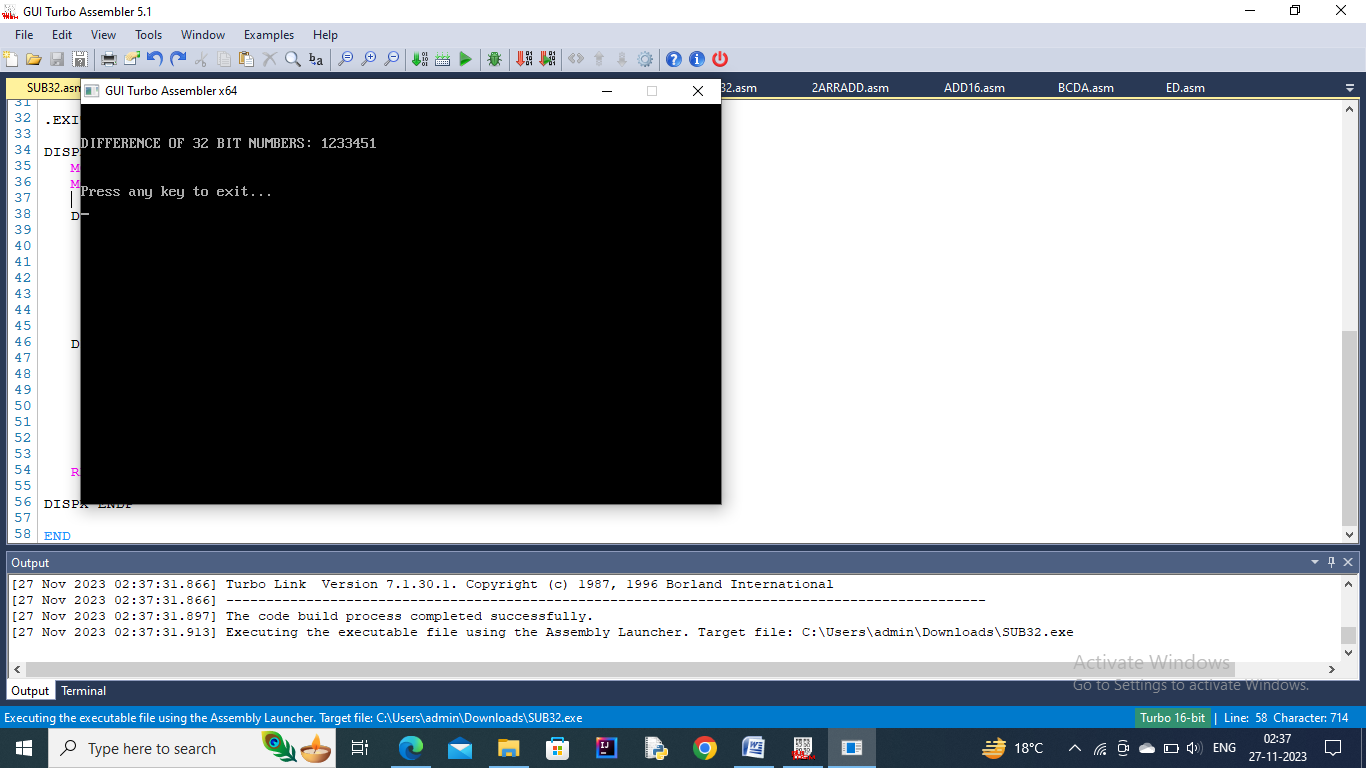
LOOP DISPX2

RET

DISPX ENDP

END

**OUTPUT –**



**Q12) Write a program to add two 32-bit BCD numbers.**

*; BCD ADDITION 32 BIT*

.MODEL SMALL

.DATA

NUM1 DB 12,34,56,78

NUM2 DB 26,32,12,21

NUM3 DB 4 DUP(?)

.CODE

.STARTUP

MOV CX,4

MOV SI,0

L1:

MOV AL,NUM1[SI]

ADD AL,NUM2[SI]

MOV NUM3[SI],AL

INC SI

LOOP L1

MOV SI,0

MOV CX,4

L2:

MOV AL,NUM3[SI]

AAM

ADD AX,3030H

MOV BX,AX

MOV AH,2

MOV DL,BH

INT 21H

MOV AH,2

MOV DL,BL

INT 21H

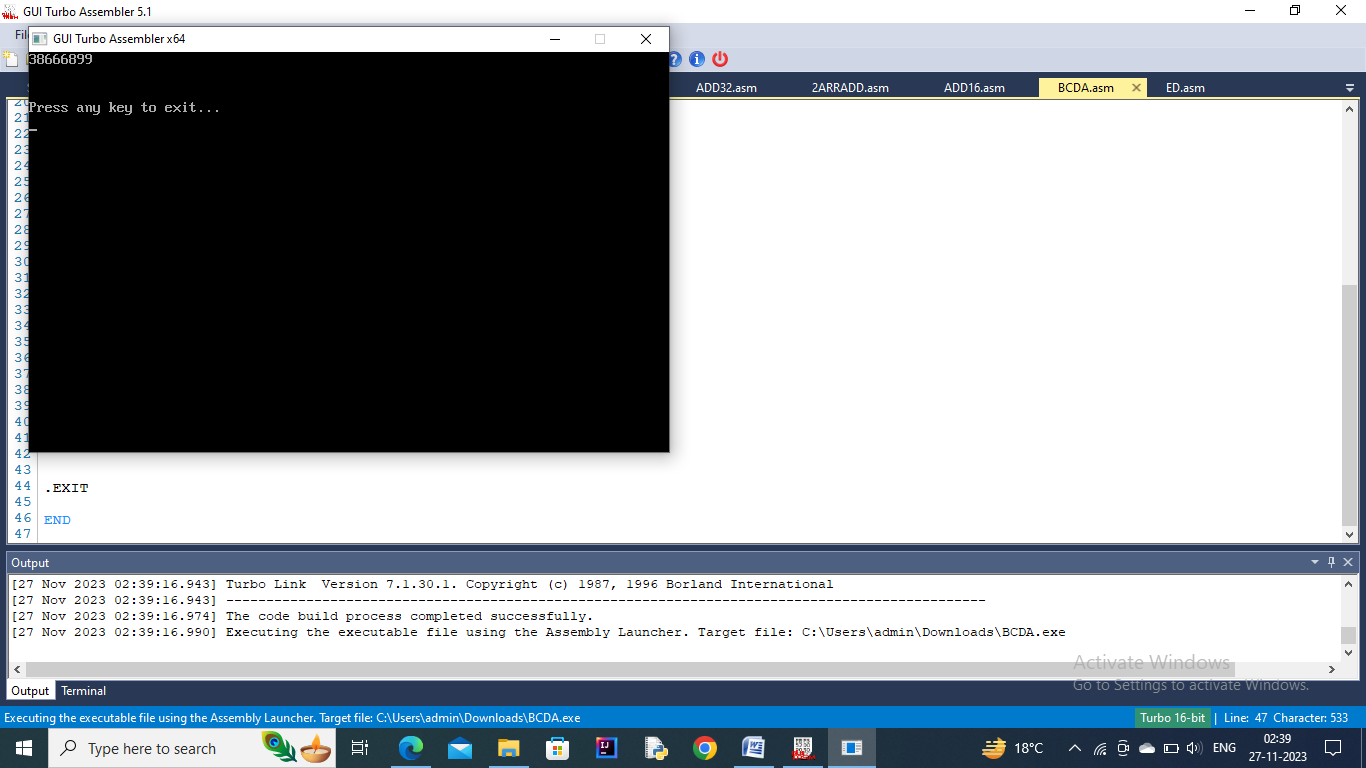
INC SI

LOOP L2

.EXIT

END

**OUTPUT –**



**Q13) Write a program to subtract two 32-bit BCD numbers.**

*; BCD SUBTRACTION 32 BIT*

.MODEL SMALL

.DATA

NUM1 DB 28,34,56,78

NUM2 DB 26,32,12,21

NUM3 DB 4 DUP(?)

.CODE

.STARTUP

MOV CX,4

MOV SI,0

L1:

MOV AL,NUM1[SI]

SUB AL,NUM2[SI]

MOV NUM3[SI],AL

INC SI

LOOP L1

MOV SI,0

MOV CX,4

L2:

MOV AL,NUM3[SI]

AAM

ADD AX,3030H

MOV BX,AX

MOV AH,2

MOV DL,BH

INT 21H

MOV AH,2

MOV DL,BL

INT 21H

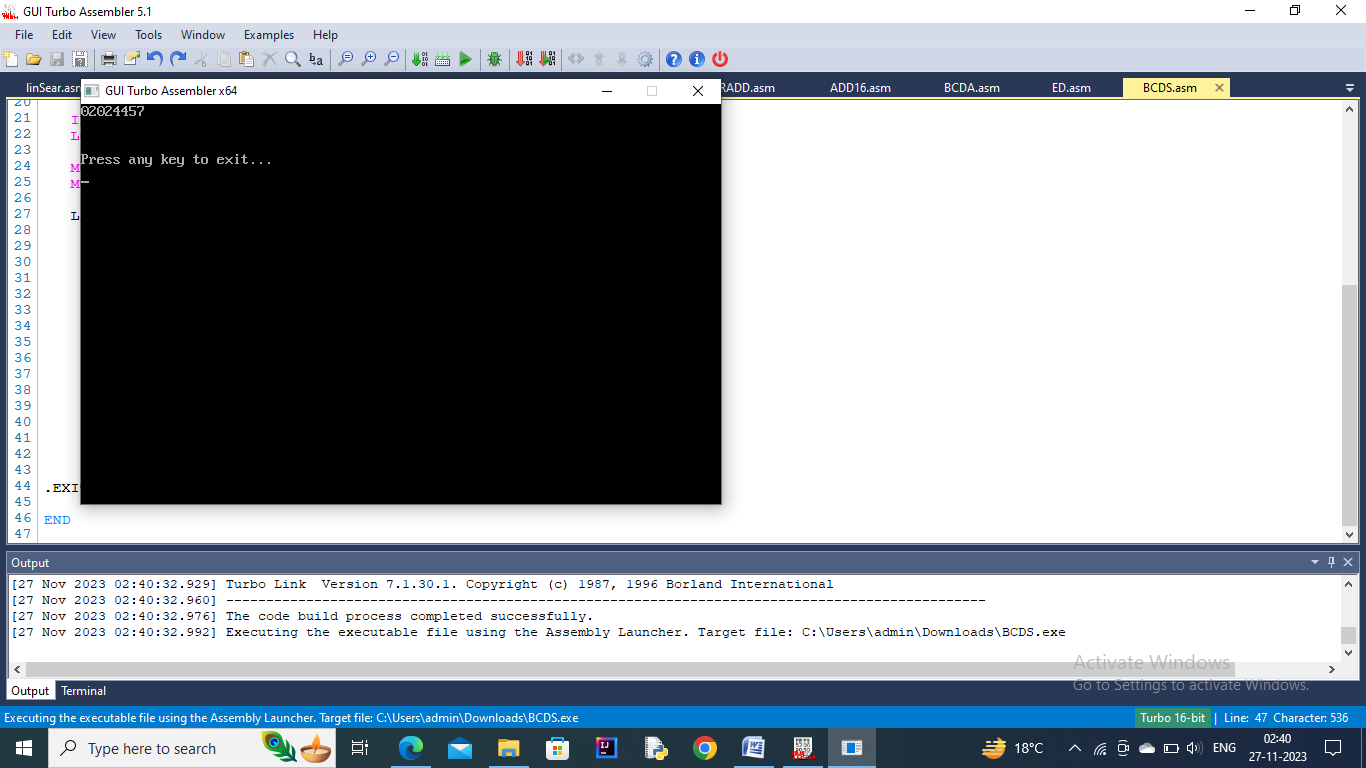
INC SI

LOOP L2

.EXIT

END

**OUTPUT –**



**Q14) Write a program to sort an array.**

*; SORTING*

.MODEL SMALL

.DATA

ARR DB 9 DUP(?)

MES1 DB 13,10,10,"ENTER THE ARRAY ELEMENTS:$"

MES2 DB 13,10,10,"THE SORTED ARRAY:$"

.CODE

.STARTUP

MOV AH,9

MOV DX,OFFSET MES1

INT 21H

MOV SI,0

MOV CX,9

L1:

MOV AH,1

INT 21H

SUB AL,30H

MOV ARR[SI],AL

INC SI

LOOP L1

MOV CX,9

DEC CX

L2:

MOV DI,CX

MOV SI,0

L3:

MOV AL,ARR[SI]

CMP AL,ARR[SI+1]

JL CONTINUE

XCHG AL,ARR[SI+1]

MOV ARR[SI],AL

CONTINUE:

INC SI

LOOP L3

MOV CX,DI

LOOP L2

MOV AH,9

MOV DX,OFFSET MES2

INT 21H

MOV SI,0

MOV CX,9

L4:

MOV DL,ARR[SI]

ADD DL,30H

MOV AH,2

INC SI

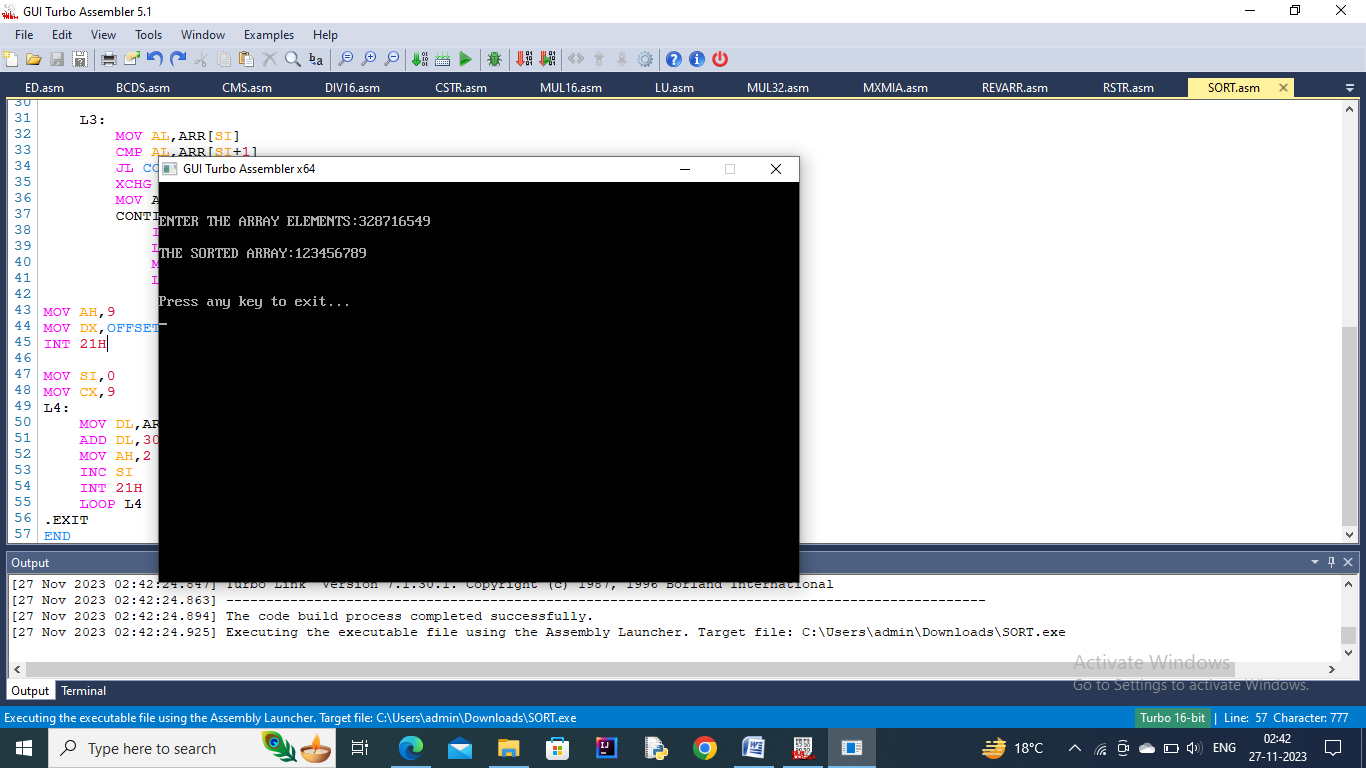
INT 21H

LOOP L4

.EXIT

END

**OUTPUT –**



**Q15) Write a program to perform an ASCII to Binary conversion.**

*; ASCII TO BINARY*

.MODEL SMALL

.DATA

TEMP DW ?

MES1 DB 13,10,10,"ENTER A NUMBER : $"

MES2 DB 13,10,10,"THE NUMBER ENTERED IS : $"

.CODE

.STARTUP

MOV DX,OFFSET MES1

MOV AH,9

INT 21H

XOR AX,AX

CALL READN

MOV TEMP, AX

MOV AH,9

MOV DX, OFFSET MES2

INT 21H

MOV AX, TEMP

CALL DISPX

.EXIT

READN PROC NEAR

PUSH BX

PUSH CX

MOV CX,10

MOV BX, 0

READN1:

MOV AH,1

INT 21H

CMP AL,30H

JB READN2

CMP AL, 39H

JA READN2

SUB AL,30H

PUSH AX

MOV AX,BX

MUL CX

MOV BX,AX

POP AX

MOV AH,0

ADD BX,AX

JMP READN1

READN2:

MOV AX,BX

POP CX

POP BX

RET

READN ENDP

DISPX PROC NEAR

PUSH DX

PUSH CX

PUSH BX

MOV CX,0

MOV BX,10

DISPX1:

MOV DX,0

DIV BX

PUSH DX

INC CX

OR AX,AX

JNZ DISPX1

DISPX2:

POP DX

MOV AH,2

ADD DL,30H

INT 21H

LOOP DISPX2

POP BX

POP CX

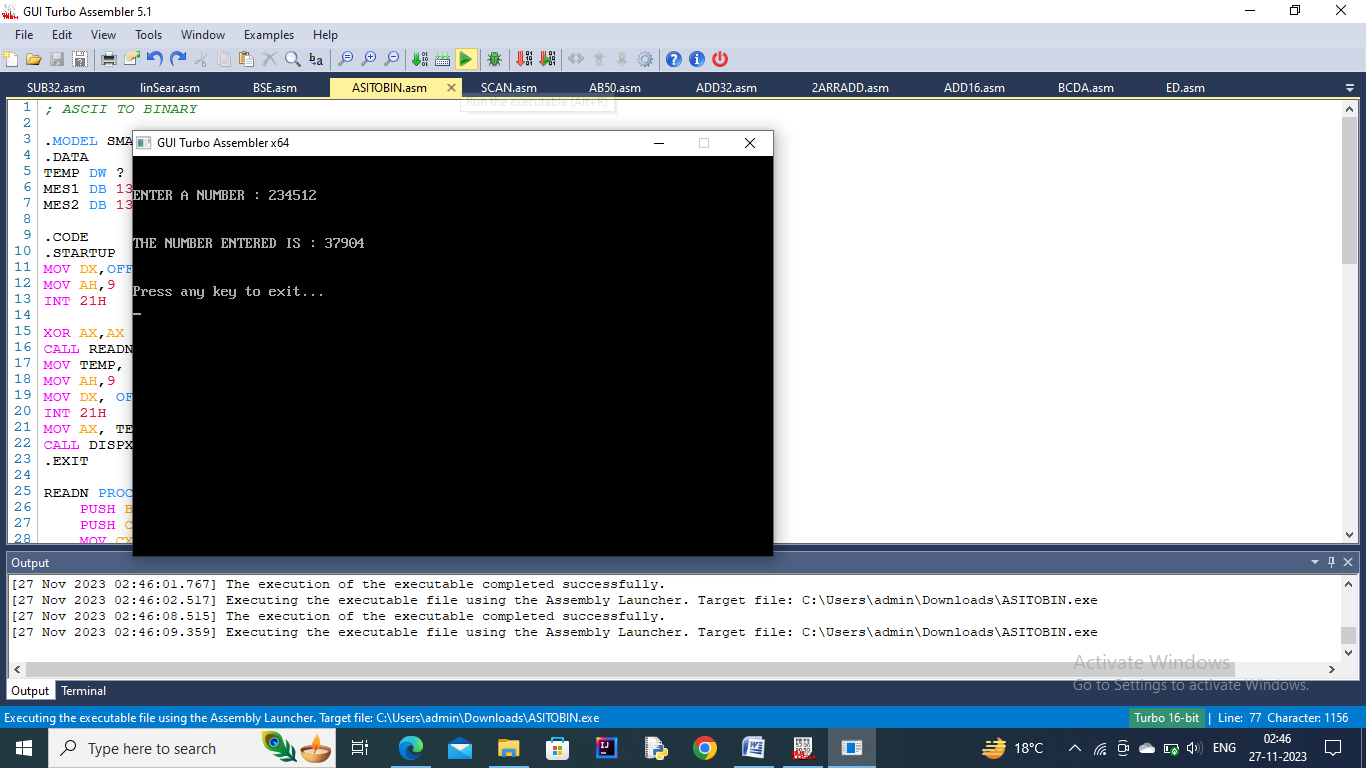
POP DX

RET

DISPX ENDP

END

**OUTPUT –**



**Q16) Write a program to perform a Binary to ASCII conversion.**

*; BINARY TO ASCII*

.MODEL SMALL

.CODE

.STARTUP

MOV AX,1234

CALL DISPX

.EXIT

DISPX PROC NEAR

PUSH DX

PUSH CX

PUSH BX

MOV CX,0

MOV BX,10

DISPX1:

MOV DX,0

DIV BX

PUSH DX

INC CX

OR AX,AX

JNZ DISPX1

DISPX2:

POP DX

MOV AH,2

ADD DL,30H

INT 21H

LOOP DISPX2

POP BX

POP CX

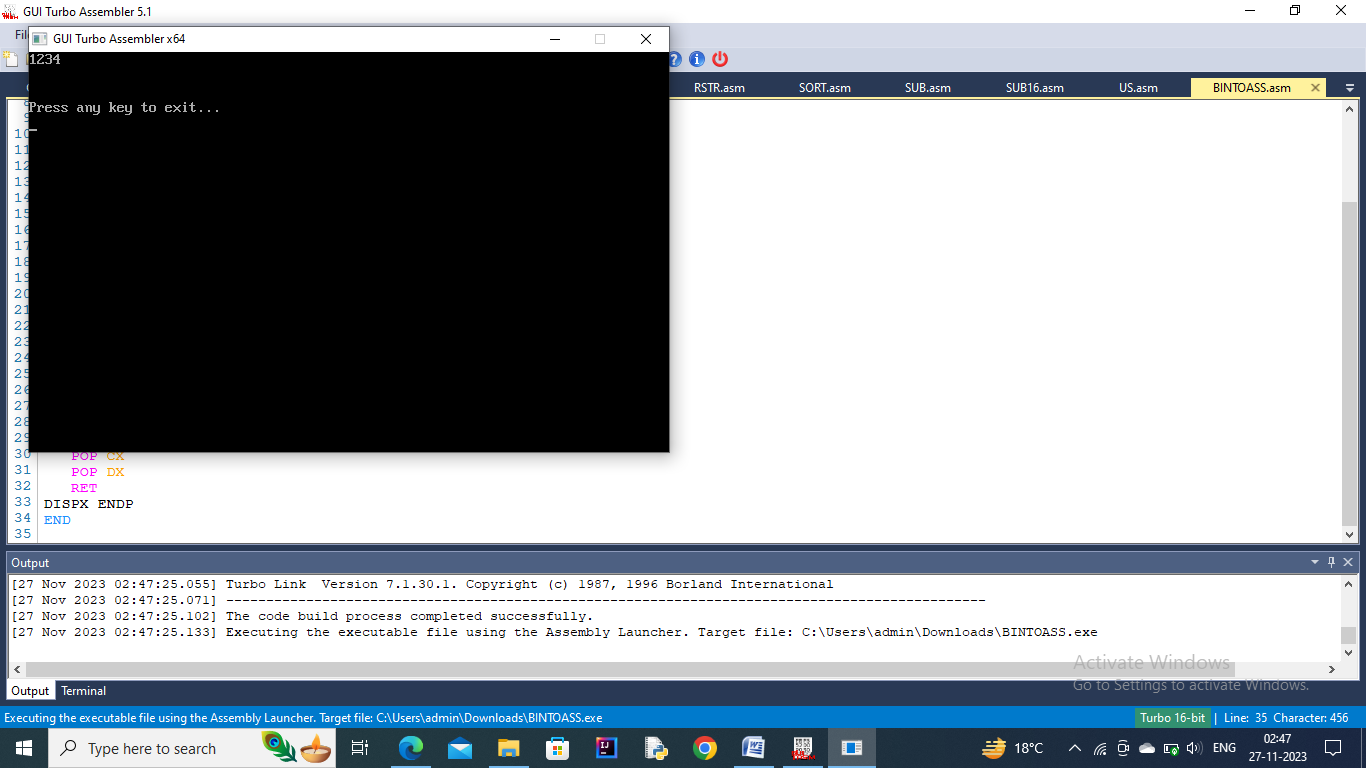
POP DX

RET

DISPX ENDP

END

**OUTPUT –**



**Q17) Write a program to count the number of times a character appears in a given string.**

*;count the number of times a character appears in a given string*

DATA SEGMENT

MSG1 DB 10,13,’ENTER ANY STRING :- $’

MSG2 DB 10,13,’ENTER ANY CHARACTER :- $’

MSG3 DB 10,13,’ $’

MSG4 DB 10,13,’NO, CHARACTER FOUND IN THE GIVEN STRING $’

MSG5 DB ‘ CHARACTER(S) FOUND IN THE GIVEN STRING $’

CHAR DB ?

COUNT DB 0

P1 LABEL BYTE

M1 DB 0FFH

L1 DB ?

P11 DB 0FFH DUP (‘$’)

DATA ENDS

DISPLAY MACRO MSG

MOV AH,9

LEA DX,MSG

INT 21H

ENDM

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:

MOV AX,DATA

MOV DS,AX

DISPLAY MSG1

LEA DX,P1

MOV AH,0AH

INT 21H

DISPLAY MSG2

MOV AH,1

INT 21H

MOV CHAR,AL

DISPLAY MSG3

LEA SI,P11

MOV CL,L1

MOV CH,0

CHECK:

MOV AL,[SI]

CMP CHAR,AL

JNE SKIP

INC COUNT

SKIP:

INC SI

LOOP CHECK

CMP COUNT,0

JE NOTFOUND

DISPLAY MSG3

MOV DL,COUNT

ADD DL,30H

MOV AH,2

INT 21H

DISPLAY MSG5

JMP EXIT

NOTFOUND:

DISPLAY MSG4

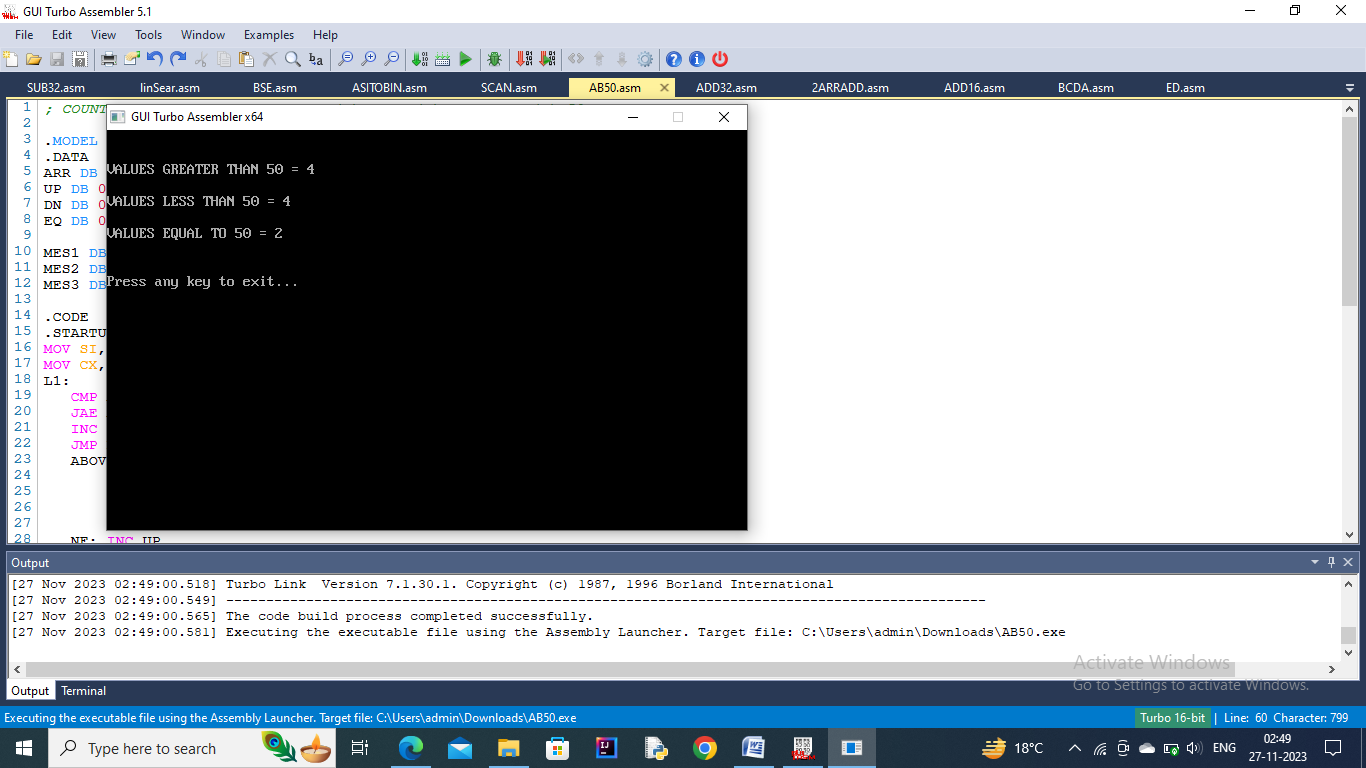
EXIT: MOV AH,4CH

INT 21H

CODE ENDS

END START

**OUTPUT –**

****

**Q18) Write a program to count the number of elements in an array that are greater than a given value.**

*; COUNT NUMBER OF VALUES GREATER(>), LESS(<) AND EQUAL(=) 50.*

.MODEL SMALL

.DATA

ARR DB 78,23,45,76,90,50,23,45,89,50

UP DB 0

DN DB 0

EQ DB 0

MES1 DB 13,10,10, "VALUES GREATER THAN 50 = $"

MES2 DB 13,10,10, "VALUES LESS THAN 50 = $"

MES3 DB 13,10,10, "VALUES EQUAL TO 50 = $"

.CODE

.STARTUP

MOV SI,0

MOV CX,10

L1:

CMP ARR[SI],50

JAE ABOVE

INC DN

JMP NEXT

ABOVE:

JNZ NE

INC EQ

JMP NEXT

NE: INC UP

NEXT:

INC SI

LOOP L1

MOV DX, OFFSET MES1

MOV AH,9

INT 21H

MOV DL, UP

ADD DL,30H

MOV AH,2

INT 21H

MOV DX, OFFSET MES2

MOV AH,9

INT 21H

MOV DL, DN

ADD DL,30H

MOV AH,2

INT 21H

MOV DX, OFFSET MES3

MOV AH, 9

INT 21H

MOV DL, EQ

ADD DL,30H

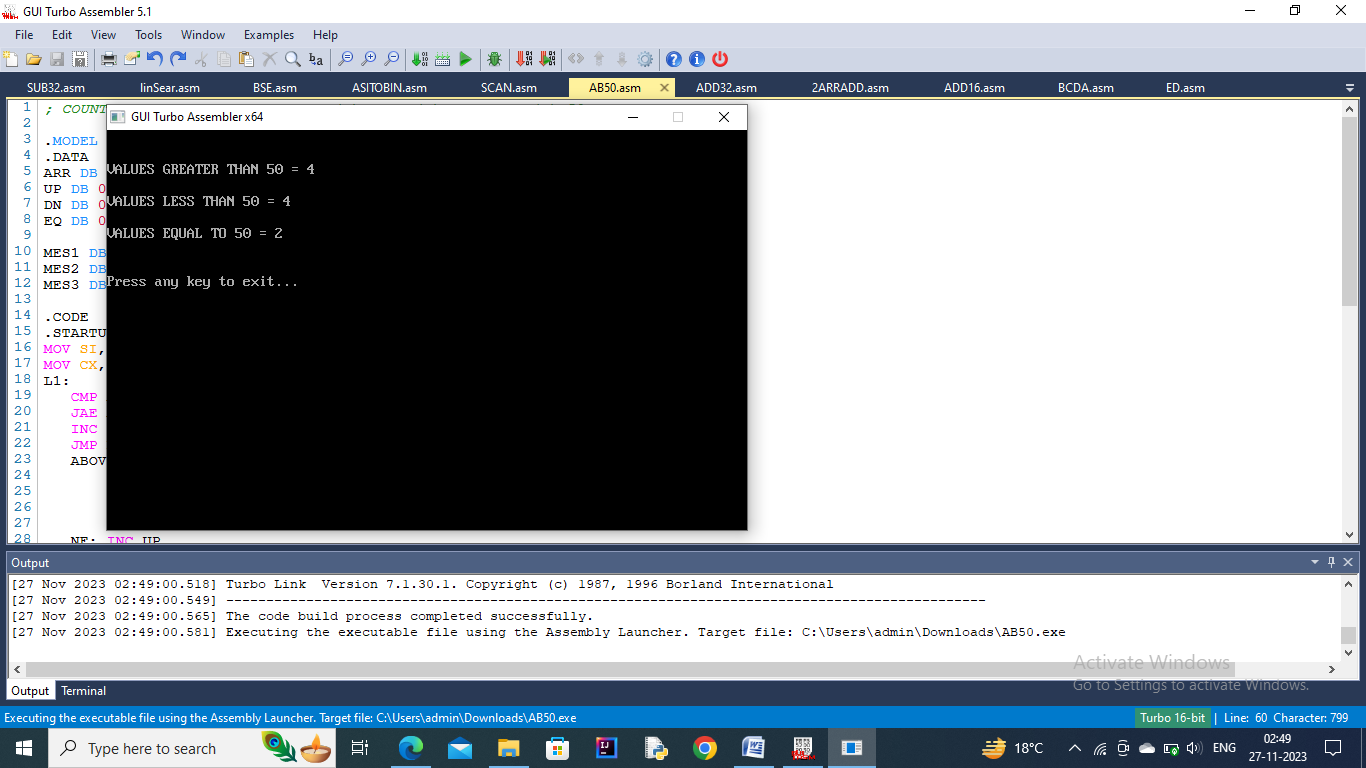
MOV AH,2

INT 21H

.EXIT

END

**OUTPUT –**



**Q19) Write a program to print the length of a string.**

*;length of a string*

DATA SEGMENT

STR DB ‘GANGADHAR$’

MSG1 DB 10,13,’THE STRING IN THE MEMORY IS : $’

MSG2 DB 10,13,’LENGTH OF THE STRING IS :- $’

LEN DB 0H

DATA ENDS

DISPLAY MACRO MSG

MOV AH,9

LEA DX,MSG

INT 21H

ENDM

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:

MOV AX,DATA

MOV DS,AX

DISPLAY MSG1

DISPLAY STR

LEA SI,STR

NEXT:

CMP [SI],’$’

JE DONE

INC LEN

INC SI

JMP NEXT

DONE:

DISPLAY MSG2

MOV AL,LEN

ADD AL,30H

MOV DL,AL

MOV AH,2

INT 21H

MOV AH,4CH

INT 21H

CODE ENDS

END START

**OUTPUT –**

