Final Practical File

Name: Ravish Ranjan

Roll no.: 21058570040

Paper: Microprocessors

Semester: 5th Semester

Course: BSc. Hons. Computer Science

Question 1. Write a program for 32-bit Binary Addition, Subtraction, Division, and Multiplication.

Question 2. Write a program for 32-bit BCD Addition, and Subtraction.

Question 3. Write a program for sorting.

Question 4. Write a program for linear search and binary search.

Question 5. Write a program to add and subtract two arrays.

Question 6. Write a program for binary to ASCII conversion.

Question 7. Write a program for ASCII to binary conversion.

Q1) Write a program for 32 bit Binary Addition, Subtraction, Division & Multiplication.

Ans

Addition

```
.MODEL SMALL

.386

.data

    MSG1    DB  'Program to add 2 multi-digit number using array',13,10,'$'

    MSG2    DB  13,10,'Enter the 1st number: $'

    MSG3    DB  13,10,'Enter the 2nd number: $'

    MSG4    DB  13,10,'The result:  $'

    NUM1    DD  0

    NUM2    DD  0

    MUL_FAC DD  10

    COUNTER DD  0

    ARR_NUM DD  50  DUP(0)


    .code

    .STARTUP

    INITIAL_MSG:

            MOV     AH, 09

            MOV     EDX, OFFSET MSG1

            INT     21H


    FIRST_MSG:

            MOV     AH, 09
```

```
        MOV     EDX, OFFSET MSG2
        INT     21H

GET_1ST_NUM:
        MOV     AH, 01
        INT     21H
        CMP     AL,13
        JZ      SECOND_MSG

CONVERT_1ST_NUM:
        SUB     AL, 48
        MOV     BL,AL
        MOV     BH,0
        MOV     EAX,NUM1
        MUL     MUL_FAC
        ADD     EBX,EAX
        MOV     NUM1,EBX
        JMP     GET_1ST_NUM

SECOND_MSG:
        MOV     AH, 09
        MOV     EDX, OFFSET MSG3
        INT     21H


GET_2ND_NUM:
        MOV     AH, 01

        INT     21H

        CMP     AL,13

        JZ      ADD_NUMS


CONVERT_2ND_NUM:

        SUB     AL, 48

        MOV     BL,AL

        MOV     BH,0

        MOV     EAX,NUM2

        MUL     MUL_FAC

        ADD     EBX,EAX

        MOV     NUM2,EBX

        JMP     GET_2ND_NUM


    ADD_NUMS:

        MOV     EAX, NUM1

        MOV     ECX, NUM2

        ADD     EAX,ECX

        MOV     EBX,EAX

        DAA


        MOV     AH, 09

        MOV     EDX, OFFSET MSG4

        INT     21H
```

```
DISPLAYING:

        MOV     EAX,EBX

        MOV     EBX,10

        MOV     EDI, OFFSET ARR_NUM

        MOV     NUM1, EDI


BREAKING_NUM:

        MOV     EDX,0

        DIV     EBX

        ADD     EDX,48

        MOV     [EDI],EDX

        INC     EDI

        CMP     EAX,0

        JZ      JOIN_N_DISPLAY

        JMP     BREAKING_NUM



JOIN_N_DISPLAY:

        MOV     EDX, [EDI]

        MOV     AH,02

        INT     21H

        CMP     NUM1,EDI

        JZ      EXIT_PROGRAM

        DEC     EDI

        JMP     JOIN_N_DISPLAY
```

```
    EXIT_PROGRAM:

            MOV     AH, 4CH

            MOV     AL, 0

            INT     21H

            .EXIT

            END
```

Subtraction

```
.MODEL SMALL

.386

.data

    MSG1    DB  'Program to add 2 multi-digit number using array',13,10,'$'

    MSG2    DB  13,10,'Enter the 1st number: $'

    MSG3    DB  13,10,'Enter the 2nd number: $'

    MSG4    DB  13,10,'The result:  $'

    NUM1    DD  0

    NUM2    DD  0

    MUL_FAC DD  10

    COUNTER DD  0

    ARR_NUM DD  50  DUP(0)


    .code

    .STARTUP

    INITIAL_MSG:

            MOV     AH, 09

            MOV     EDX, OFFSET MSG1

            INT     21H
```

```
FIRST_MSG:

        MOV     AH, 09

        MOV     EDX, OFFSET MSG2

        INT     21H


GET_1ST_NUM:

        MOV     AH, 01

        INT     21H

        CMP     AL,13

        JZ      SECOND_MSG


CONVERT_1ST_NUM:

        SUB     AL, 48

        MOV     BL,AL

        MOV     BH,0

        MOV     EAX,NUM1

        MUL     MUL_FAC

        ADD     EBX,EAX

        MOV     NUM1,EBX

        JMP     GET_1ST_NUM


SECOND_MSG:

        MOV     AH, 09

        MOV     EDX, OFFSET MSG3

        INT     21H
```

```
GET_2ND_NUM:

        MOV     AH, 01

        INT     21H

        CMP     AL,13

        JZ      SUB_NUMS


CONVERT_2ND_NUM:

        SUB     AL, 48

        MOV     BL,AL

        MOV     BH,0

        MOV     EAX,NUM2

        MUL     MUL_FAC

        ADD     EBX,EAX

        MOV     NUM2,EBX

        JMP     GET_2ND_NUM


    SUB_NUMS:

        MOV     EBP,00H

        MOV     EBX, NUM1

        MOV     ECX, NUM2

        SUB     EBX,ECX

        JGE     NORMAL

        MOV     EBP,01H

        NEG     EBX
```

```
NORMAL:

        MOV     AH, 09

        MOV     EDX, OFFSET MSG4

        INT     21H

        CMP     EBP,0H

        JZ      DISPLAYING

        MOV     AH,02

        MOV     DL,'-'

        INT     21H


DISPLAYING:

        MOV     EAX,EBX

        MOV     EBX,10

        MOV     EDI, OFFSET ARR_NUM

        MOV     NUM1, EDI


BREAKING_NUM:

        MOV     EDX,0

        DIV     EBX

        ADD     EDX,48

        MOV     [EDI],EDX

        INC     EDI

        CMP     EAX,0

        JZ      JOIN_N_DISPLAY

        JMP     BREAKING_NUM
```

```
    JOIN_N_DISPLAY:
            MOV     EDX, [EDI]
            MOV     AH,02
            INT     21H
            CMP     NUM1,EDI
            JZ      EXIT_PROGRAM
            DEC     EDI
            JMP     JOIN_N_DISPLAY

    EXIT_PROGRAM:
            MOV     AH, 4CH
            MOV     AL, 0
            INT     21H
            .EXIT
            END
```

Division

```
.MODEL SMALL
.386
.data
    MSG1    DB  'Program to add 2 multi-digit number using array',13,10,'$'
    MSG2    DB  13,10,'Enter the 1st number: $'
    MSG3    DB  13,10,'Enter the 2nd number: $'
    MSG4    DB  13,10,'The result:  $'
    NUM1    DD  0
    NUM2    DD  0
```

```
MUL_FAC DD  10

COUNTER DD  0

ARR_NUM DD  50  DUP(0)


.code

.STARTUP

INITIAL_MSG:

        MOV     AH, 09

        MOV     EDX, OFFSET MSG1

        INT     21H


FIRST_MSG:

        MOV     AH, 09

        MOV     EDX, OFFSET MSG2

        INT     21H


GET_1ST_NUM:

        MOV     AH, 01

        INT     21H

        CMP     AL,13

        JZ      SECOND_MSG


CONVERT_1ST_NUM:

        SUB     AL, 48

        MOV     BL,AL

        MOV     BH,0
        MOV     EAX,NUM1
        MUL     MUL_FAC
        ADD     EBX,EAX
        MOV     NUM1,EBX
        JMP     GET_1ST_NUM

SECOND_MSG:
        MOV     AH, 09
        MOV     EDX, OFFSET MSG3
        INT     21H



GET_2ND_NUM:
        MOV     AH, 01
        INT     21H
        CMP     AL,13
        JZ      DIV_NUMS

CONVERT_2ND_NUM:
        SUB     AL, 48
        MOV     BL,AL
        MOV     BH,0
        MOV     EAX,NUM2
        MUL     MUL_FAC
        ADD     EBX,EAX
        MOV     NUM2,EBX
        JMP     GET_2ND_NUM


    DIV_NUMS:

        MOV     EAX, NUM1

        MOV     ECX, NUM2

        DIV     ECX

        MOV     EBX,EAX

        DAA


        MOV     AH, 09

        MOV     EDX, OFFSET MSG4

        INT     21H


DISPLAYING:

        MOV     EAX,EBX

        MOV     EBX,10

        MOV     EDI, OFFSET ARR_NUM

        MOV     NUM1, EDI


BREAKING_NUM:

        MOV     EDX,0

        DIV     EBX

        ADD     EDX,48

        MOV     [EDI],EDX

        INC     EDI

        CMP     EAX,0
        JZ      JOIN_N_DISPLAY
        JMP     BREAKING_NUM


    JOIN_N_DISPLAY:
        MOV     EDX, [EDI]
        MOV     AH,02
        INT     21H
        CMP     NUM1,EDI
        JZ      EXIT_PROGRAM
        DEC     EDI
        JMP     JOIN_N_DISPLAY

    EXIT_PROGRAM:
        MOV     AH, 4CH
        MOV     AL, 0
        INT     21H
        .EXIT
        END
```

Multiplication

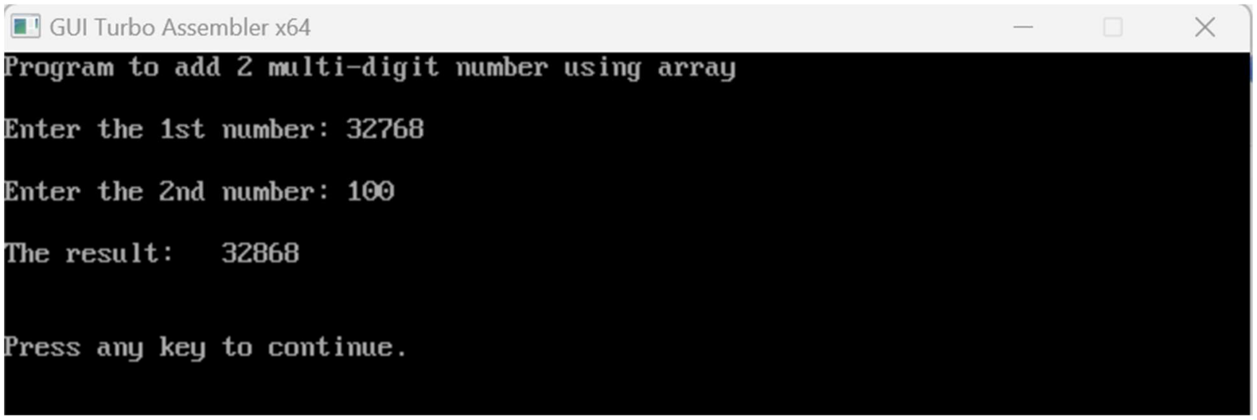
```
.MODEL SMALL
.386

.data
    MSG1    DB  'Program to add 2 multi-digit number using array',13,10,'$'
    MSG2    DB  13,10,'Enter the 1st number: $'
```

```
MSG3    DB  13,10,'Enter the 2nd number: $'

MSG4    DB  13,10,'The product of the numbers is $'

NUM1    DD  0

NUM2    DD  0

MUL_FAC DD  10

COUNTER DD  0

ARR_NUM DD  50  DUP(0)


.code

.STARTUP

INITIAL_MSG:

        MOV     AH, 09

        MOV     EDX, OFFSET MSG1

        INT     21H


FIRST_MSG:

        MOV     AH, 09

        MOV     EDX, OFFSET MSG2

        INT     21H


GET_1ST_NUM:

        MOV     AH, 01

        INT     21H

        CMP     AL,13

        JZ      SECOND_MSG
```

```
CONVERT_1ST_NUM:

        SUB     AL, 48

        MOV     BL,AL

        MOV     BH,0

        MOV     EAX,NUM1

        MUL     MUL_FAC

        ADD     EBX,EAX

        MOV     NUM1,EBX

        JMP     GET_1ST_NUM


SECOND_MSG:

        MOV     AH, 09

        MOV     EDX, OFFSET MSG3

        INT     21H


GET_2ND_NUM:

        MOV     AH, 01

        INT     21H

        CMP     AL,13

        JZ      MUL_NUMS


CONVERT_2ND_NUM:

        SUB     AL, 48

        MOV     BL,AL

        MOV     BH,0

        MOV     EAX,NUM2
        MUL     MUL_FAC

        ADD     EBX,EAX

        MOV     NUM2,EBX

        JMP     GET_2ND_NUM


    MUL_NUMS:

        MOV     EAX, NUM1

        MOV     ECX, NUM2

        MUL     ECX

        MOV     EBX,EAX

        DAA

        MOV     AH, 09

        MOV     EDX, OFFSET MSG4

        INT     21H


DISPLAYING:

        MOV     EAX,EBX

        MOV     EBX,10

        MOV     EDI, OFFSET ARR_NUM

        MOV     NUM1, EDI


BREAKING_NUM:

        MOV     EDX,0

        DIV     EBX

        ADD     EDX,48

        MOV     [EDI],EDX
        INC     EDI
        CMP     EAX,0
        JZ      JOIN_N_DISPLAY
        JMP     BREAKING_NUM

JOIN_N_DISPLAY:
        MOV     EDX, [EDI]
        MOV     AH,02
        INT     21H
        CMP     NUM1,EDI
        JZ      EXIT_PROGRAM
        DEC     EDI
        JMP     JOIN_N_DISPLAY

EXIT_PROGRAM:
        MOV     AH, 4CH
        MOV     AL, 0
        INT     21H
        .EXIT
        END
```

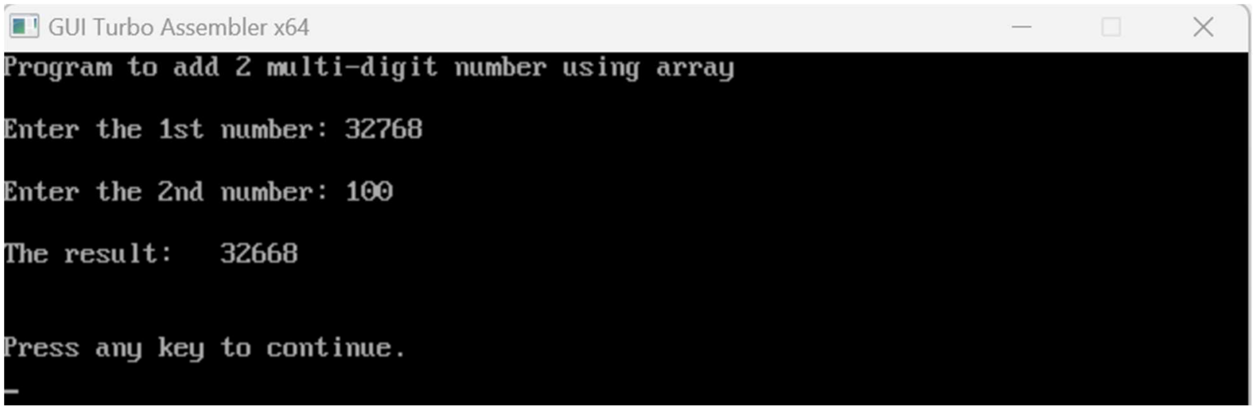
#output

Addition:



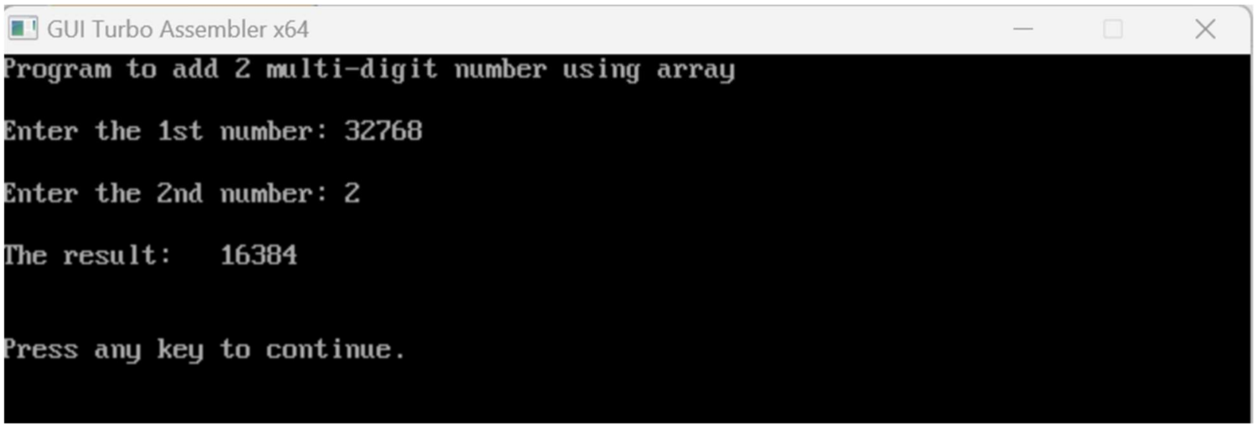
```
Program to add 2 multi-digit number using array

Enter the 1st number: 32768

Enter the 2nd number: 100

The result:   32868


Press any key to continue.
```

Subtraction:



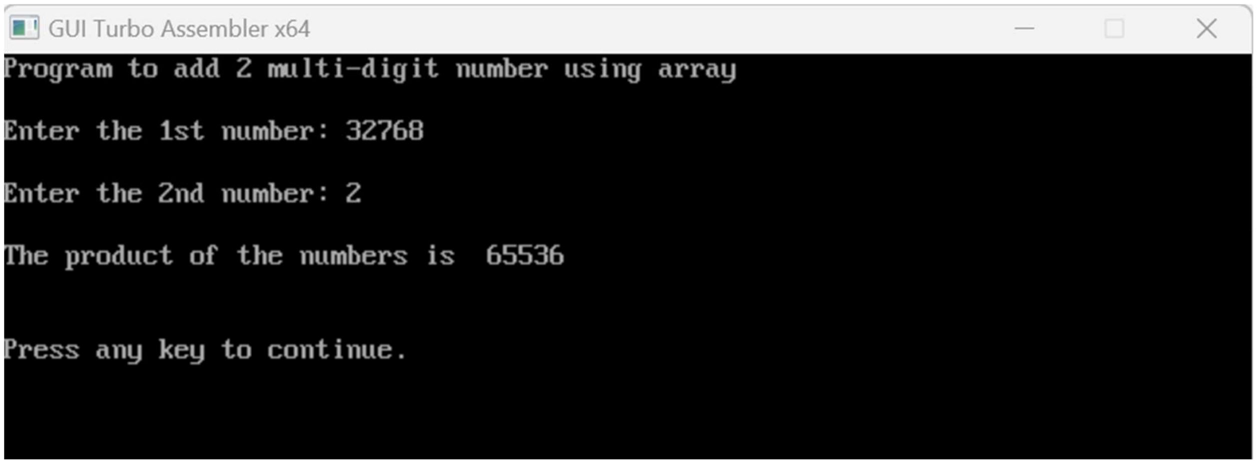
```
Program to add 2 multi-digit number using array

Enter the 1st number: 32768

Enter the 2nd number: 100

The result:   32668


Press any key to continue.
_
```

Division:



```
Program to add 2 multi-digit number using array

Enter the 1st number: 32768

Enter the 2nd number: 2

The result:   16384


Press any key to continue.
```

Multiplication:



GUI Turbo Assembler x64

```
Program to add 2 multi-digit number using array

Enter the 1st number: 32768

Enter the 2nd number: 2

The product of the numbers is  65536


Press any key to continue.
```

Q2) Write a program for 32 bit BCD Addition & Subtraction

Ans

Addition

```
.MODEL SMALL

.386

.data

num1 DD 0H

num2 DD 0H

num3 DD 0H

msg db 10,13,"Enter the first no.:: $"

msg1 db 10,13,"Enter the second no.:: $"

msg2 db 10,13,"The Resultant sum is :: $"

.code

.startup

MOV AH,09

MOV DX,OFFSET msg

INT 21H
```

```
MOV EBX, 0

GET_NUM_1:
MOV AH,01
INT 21H
CMP AL,'A'
JGE EXIT
CMP AL,13
JZ BREAK1
SUB AL,30H
SHL EBX,4
ADD BL,AL
LOOP GET_NUM_1

BREAK1:MOV num1,EBX
MOV AH,09
MOV DX,OFFSET msg1
INT 21H
MOV EBX,0

GET_NUM_2:
MOV AH,01
INT 21H
CMP AL,'A'
JGE EXIT
CMP AL,13
```

```
JZ BREAK2

SUB AL,30H

SHL EBX,4

ADD BL,AL

LOOP GET_NUM_2


BREAK2:

MOV num2, EBX

MOV AX, word PTR num1

MOV DX, word PTR num2

ADD AL,DL

DAA

MOV BL, AL

MOV AL, AH

ADC AL,DH

DAA

MOV BH, AL


MOV word PTR num3,BX

MOV AX, word PTR num1+2

MOV DX, word PTR num2+2

ADC AL,DL

DAA

MOV BL, AL

MOV AL, AH

ADC AL,DH
```

```
DAA
MOV BH,AL
MOV word PTR num3+2,BX
MOV EBX,num3

MOV AH,09
MOV DX,OFFSET msg2
INT 21H

JNC A
MOV AH, 02H
MOV DL, "1"
INT 21H

A: MOV CX,8
DISPLAY_NUM: ROL EBX,4
MOV DL,BL
AND DL,0FH
ADD DL,30H
MOV AH,02
INT 21H
LOOP DISPLAY_NUM

EXIT:
MOV AH,4CH
int 21H
```

```
.EXIT

END
```

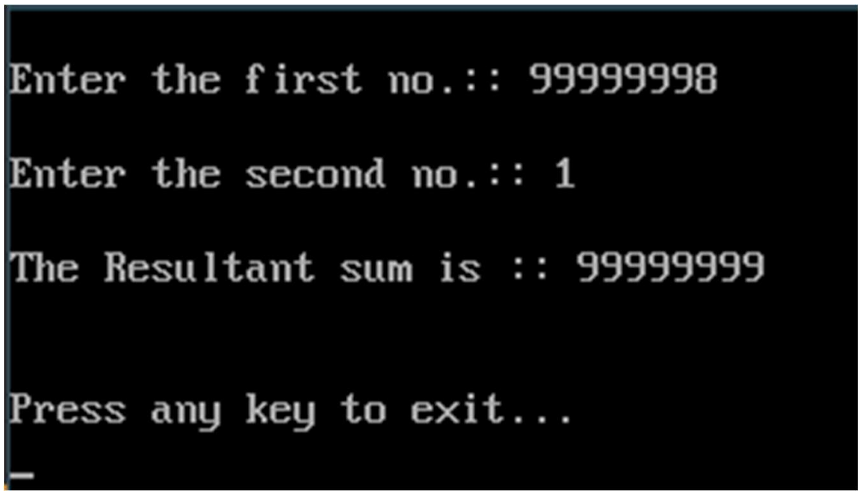
Subtraction

```
DATA_SEG    SEGMENT

    MSG1    DB  'Program to add 2 multi-digit number using array',13,10,'$'

    MSG2    DB  13,10,'Enter the 2nd number: $'

    MSG3    DB  13,10,'Enter the 1st number: $'

    MSG4    DB  13,10,'The diff of the number is $'

    NUM1    DW  0

    NUM2    DW  0

    MUL_FAC DB  10

    COUNTER DB  0

    ARR_NUM DW  50  DUP(0)

DATA_SEG    ENDS

CODE_SEG    SEGMENT

   ASSUME CS: CODE_SEG, DS:DATA_SEG

    START:  MOV     AX, DATA_SEG

            MOV     DS, AX

    INITIAL_MSG:

            MOV     AH, 09

            MOV     DX, OFFSET MSG1

            INT     21H

    FIRST_MSG:

            MOV     AH, 09

            MOV     DX, OFFSET MSG2

            INT     21H
```

```
GET_1ST_NUM:
        MOV     AH, 01
        INT     21H
        CMP     AL,13
        JZ      SECOND_MSG
CONVERT_1ST_NUM:
        SUB     AL, 48
        MOV     BL,AL
        MOV     BH,0
        MOV     AX,NUM1
        MUL     MUL_FAC
        ADD     BX,AX
        MOV     NUM1,BX
        JMP     GET_1ST_NUM
SECOND_MSG:
        MOV     AH, 09
        MOV     DX, OFFSET MSG3
        INT     21H
GET_2ND_NUM:
        MOV     AH, 01
        INT     21H
        CMP     AL,13
        JZ      ADD_NUMS
CONVERT_2ND_NUM:
        SUB     AL, 48
        MOV     BL,AL
        MOV     BH,0
        MOV     AX,NUM2
        MUL     MUL_FAC
        ADD     BX,AX
        MOV     NUM2,BX
        JMP     GET_2ND_NUM
ADD_NUMS:
        MOV     BP,00H
        MOV     BX, NUM1
        MOV     CX, NUM2
        SUB     CX,BX
        JGE     NORMAL
        MOV     BP,01H
        NEG     CX
NORMAL:
        MOV     AH, 09
        MOV     DX, OFFSET MSG4
        INT     21H
        CMP     BP,0H
        JZ      DISPLAYING
        MOV     AH,02
        MOV     DL,'-'
        INT     21H
DISPLAYING:
        MOV     AX,CX
        MOV     BX,10
        MOV     DI, OFFSET ARR_NUM

        MOV     NUM1, DI

    BREAKING_NUM:

        MOV     DX,0

        DIV     BX

        ADD     DX,48

        MOV     [DI],DX

        INC     DI

        CMP     AX,0

        JZ      JOIN_N_DISPLAY

        JMP     BREAKING_NUM

    JOIN_N_DISPLAY:

        MOV     DX, [DI]

        MOV     AH,02

        INT     21H

        CMP     NUM1,DI

        JZ      EXIT_PROGRAM

        DEC     DI

        JMP     JOIN_N_DISPLAY

    EXIT_PROGRAM:

        MOV     AH, 4CH

        MOV     AL, 0

        INT     21H

CODE_SEG    ENDS

    END START
```

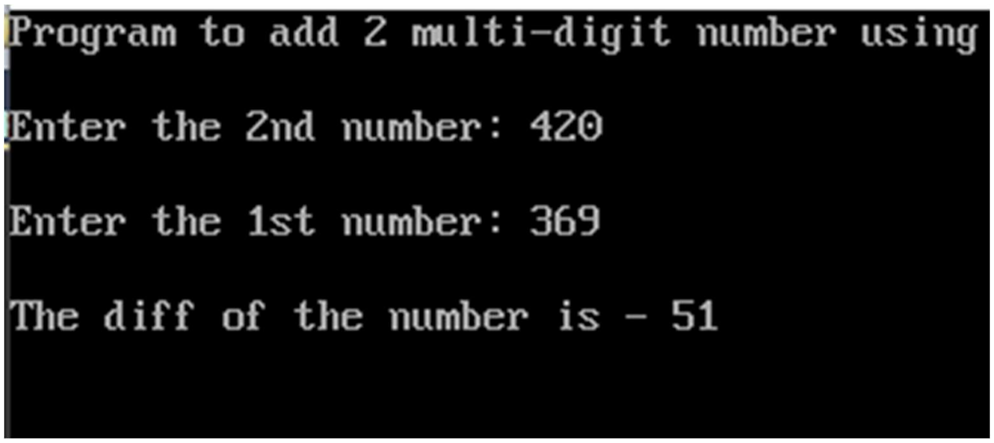
#output

Addition



```
Enter the first no.:: 99999998

Enter the second no.:: 1

The Resultant sum is :: 99999999


Press any key to exit...
_
```

Subtraction



```
Program to add 2 multi-digit number using

Enter the 2nd number: 420

Enter the 1st number: 369

The diff of the number is - 51
```

Q3) Write a program for sorting.

Ans

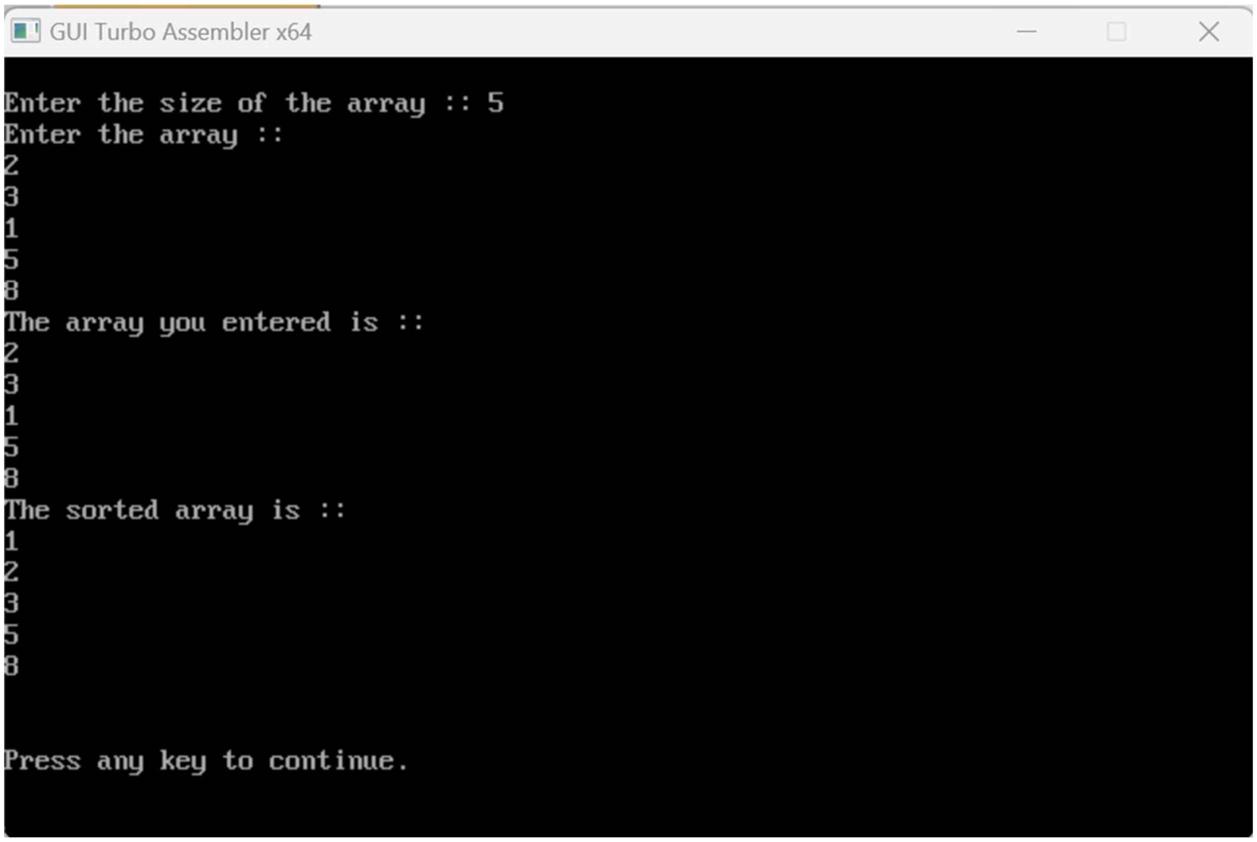
```
.model small

.386

.data

ARRAY DB 100 DUP (?)

DATA1 dw 0000H
```

```
DATA2 dw 0000H

NUMB  dw 0000H

msg   db 10,13,"Enter the size of the array :: $"

msg2  db 10,13,"Enter the array :: $"

msg3  db 10,13,"The sorted array is :: $"

msg4  db 10,13, "The array you entered is ::$"


.code

.startup

MOV AH,09

MOV DX,OFFSET msg

INT 21H


MOV AH,01

INT 21H

SUB AL,30H

MOV AH,0

MOV CX,AX

MOV DATA1,AX


MOV AH,09

MOV DX,OFFSET msg2

INT 21H
```

```
MOV AH,0

MOV SI, 0

MOV BX, OFFSET ARRAY

L1: MOV DL, 0AH

MOV AH, 02H

INT 21H

MOV AH, 01H

INT 21H

SUB AL,30H

MOV [BX + SI], AL

INC SI

LOOP L1

MOV AH,09

MOV DX,OFFSET msg4

INT 21H


MOV CX, DATA1

MOV SI, OFFSET ARRAY

L50: MOV DL, 0AH

MOV AH, 02H

INT 21H

MOV DX, [SI]

INC SI

ADD DL, 30H
```

```
MOV AH, 02

INT 21H

LOOP L50


MOV CX, DATA1

MOV BX, OFFSET ARRAY

L2: MOV SI, 0

MOV AX, SI

INC AX

MOV DI, AX

MOV DATA2, CX


MOV CX, DATA1

MOV NUMB, CX

DEC NUMB

MOV CX, NUMB

L3: MOV AL, [BX + SI]

CMP AL, [BX + DI]

JL L4

XCHG AL, [BX + DI]

MOV [BX + SI], AL

L4: INC SI

    INC DI

    LOOP L3
    MOV CX, DATA2
LOOP L2


MOV CX, DATA1
LEA SI, ARRAY
MOV AH,09
MOV DX,OFFSET msg3
INT 21H
L5: MOV DL, 0AH
MOV AH, 02H
INT 21H
MOV DX, [SI]
INC SI
ADD DL, 30H
MOV AH, 02
INT 21H
LOOP L5
.EXIT
END
```

#output


GUI Turbo Assembler x64

```
Enter the size of the array :: 5
Enter the array ::
2
3
1
5
8
The array you entered is ::
2
3
1
5
8
The sorted array is ::
1
2
3
5
8


Press any key to continue.
```

Q4) Write a program for Linear Search & Binary Search

Ans

Linear Search

```
.model small

.386

.data

ARRAY DW 20 DUP (?)

DATA1 dw 0000H

success db 10,13,"Element is present in the array $"

fail db 10,13,"Element is not present in the arary $"

msg db 10,13,"Enter the size of the array :: $"

msg2 db 10,13,"Enter the array :: $"
```

```
msg3 db 10,13,"Enter the element to be searched :: $"

.code

.startup

MOV AH,09

MOV DX,OFFSET msg

INT 21H

MOV AH,01

INT 21H

SUB AL,30H

MOV AH,0

MOV CX,AX

MOV DATA1,AX

MOV AH,09

MOV DX,OFFSET msg2

INT 21H

MOV AH,0

MOV SI, 0

MOV BX, OFFSET ARRAY


L1: MOV DL, 0AH

MOV AH, 02H

INT 21H

MOV DX, SI

MOV AH, 01H

INT 21H

SUB AL,30H
```

```
MOV SI, DX

MOV [BX + SI], AX

INC SI

LOOP L1


MOV CX,DATA1

MOV AH,09

MOV DX,OFFSET msg3

INT 21H

MOV AH,01

INT 21H

SUB AL,30H

MOV SI, 0

MOV BX, OFFSET ARRAY


L2: CMP [BX + SI], AL

JZ L3

INC SI

LOOP L2

MOV AH,09

MOV DX,OFFSET fail

INT 21H

MOV AH, 4CH

INT 21H


L3: MOV AH, 09H
```

```
MOV DX,OFFSET success

INT 21H

MOV AH, 4CH

INT 21H

.EXIT

END
```

Binary Search

```
.model small

.386

.data

ARRAY DW 20 DUP (?)

DATA1 dw 0000H

DATA2 dw 0000H

success db 10,13,"Element is present in the array $"

fail db 10,13,"Element is not present in the array $"

msg db 10,13,"Enter the size of the array :: $"

msg2 db 10,13,"Enter the array :: $"

msg3 db 10,13,"Enter the element to be searched :: $"


.code

.startup

MOV AH,09

MOV DX,OFFSET msg

INT 21H
```

```
MOV AH,01

INT 21H

SUB AL,30H

MOV AH,0

MOV CX,AX

MOV DATA1,AX


MOV AH,09

MOV DX,OFFSET msg2

INT 21H


MOV AH,0

MOV SI, 0

MOV BX, OFFSET ARRAY


L1: MOV DL, 0AH

MOV AH, 02H

INT 21H

MOV DX, SI

MOV AH, 01H

INT 21H

SUB AL,30H

MOV SI, DX

MOV [BX + SI], AX

INC SI

LOOP L1
```

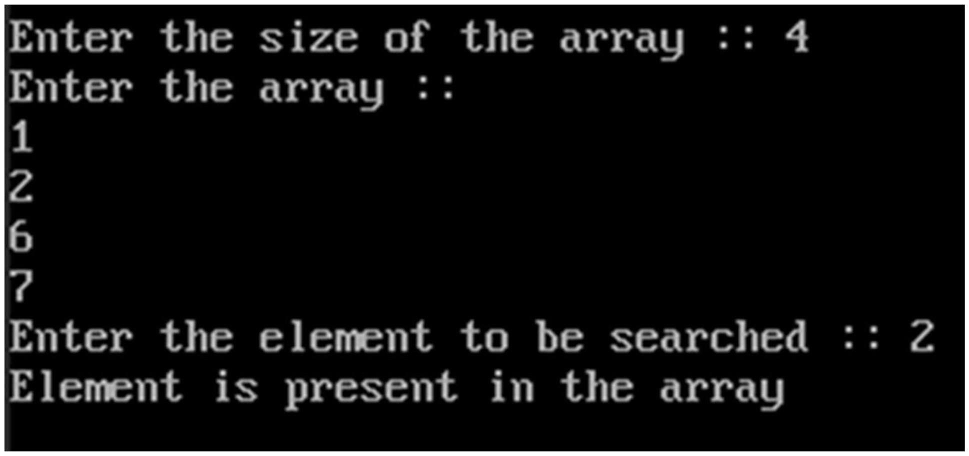
```
MOV AH,09

MOV DX,OFFSET msg3

INT 21H

MOV AH,01

INT 21H

SUB AL,30H


MOV DATA2, AX

MOV CX,DATA1

MOV SI,0

MOV DI, DATA1

MOV BP, 0

MOV BX, OFFSET ARRAY

MOV AX, DATA1


L2: MOV SI, DI

ADD SI, BP

MOV AX, SI

MOV DL, 2

DIV DL


MOV AH,0

MOV DX,0

MOV SI,AX

MOV DX,DATA2
```

```
CMP [BX + SI],DL

JZ L3

CALL L4

LOOP L2


MOV AH, 09H

MOV DX,OFFSET fail

INT 21H

MOV AH, 4CH

INT 21H


L3: MOV AH, 09H

MOV DX,OFFSET success

INT 21H

MOV AH, 4CH

INT 21H


L4 PROC NEAR

CMP [BX+SI], DL

JL L6

MOV DI, SI

RET

L6: MOV BP,SI

RET

L4 ENDP

.EXIT
```

```
END
```

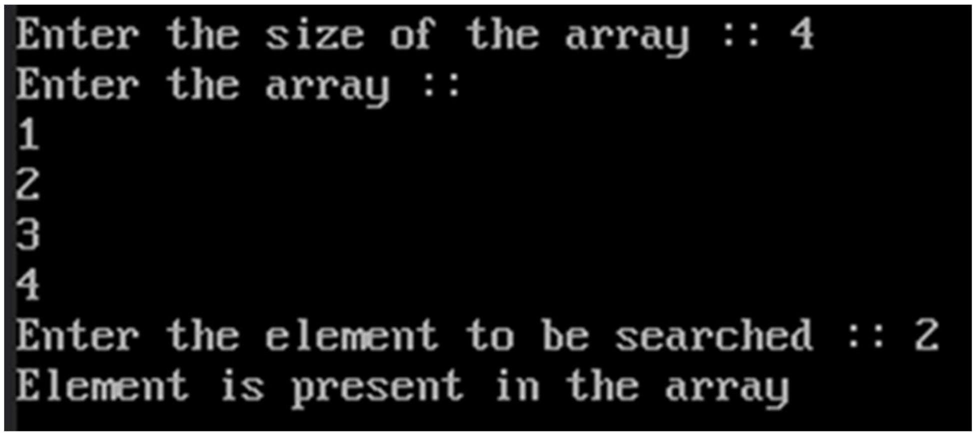
#output

Linear Search



```
Enter the size of the array :: 4
Enter the array ::
1
2
6
7
Enter the element to be searched :: 2
Element is present in the array
```

Binary Search



```
Enter the size of the array :: 4
Enter the array ::
1
2
3
4
Enter the element to be searched :: 2
Element is present in the array
```

Q5) Write a program to add & subtract two arrays.

Ans

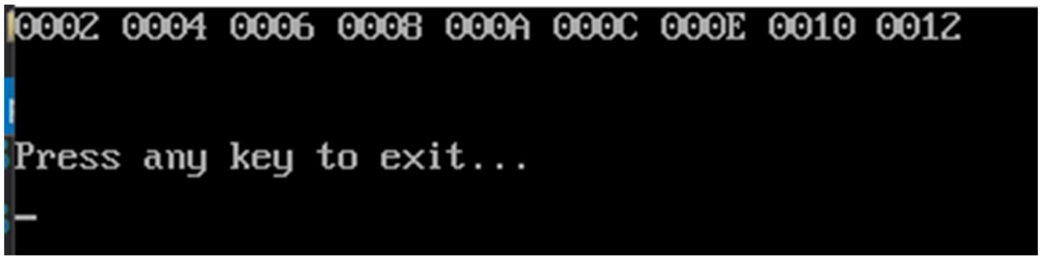
```
.model small

.386

.data
```

```
array1 db 1H,2H,3H,4H,5H,6H,7H,8H,9H

array2 db 1H,2H,3H,4H,5H,6H,7H,8H,9H

result dw 9 dup (?)

.code

.startup

MOV AX, @data

MOV DS, AX

MOV CX, 09H

MOV DI, OFFSET array1

MOV BX, OFFSET array2

MOV SI, OFFSET result


back:

MOV AH, 0

MOV AL, [DI]

ADD AL, [BX]

ADC AH, 00

MOV [SI], AX

INC DI

INC BX

INC SI

INC SI

loop back


MOV SI, OFFSET result

MOV DH, 9
```

```
l10:

MOV CH, 04H

MOV CL, 04H

MOV BX, [SI]


l2:

ROL BX, CL

MOV DL, BL

AND DL, 15

CMP DL, 09

JBE l4

ADD DL, 07H


l4:

ADD DL, 30H

MOV AH, 02

INT 21H

DEC CH

JNZ l2

MOV DL, ' '

INT 21H

INC SI

INC SI

DEC DH

JNZ l10
```

```
MOV AH, 4CH

INT 21


.EXIT

END
```

#output



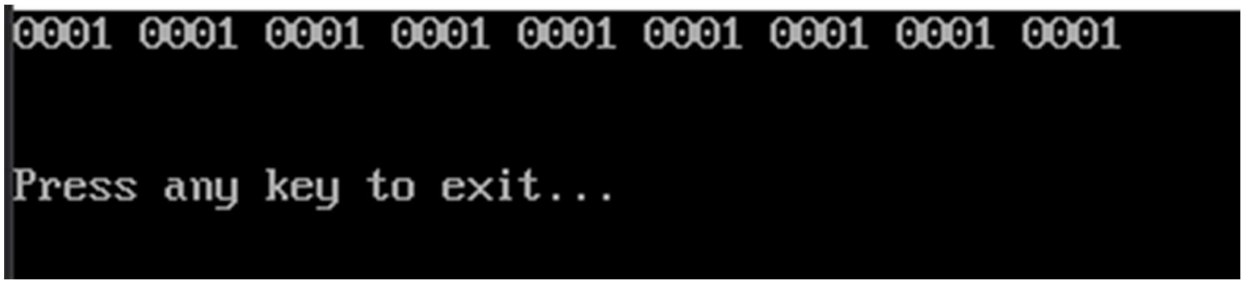
```
0002 0004 0006 0008 000A 000C 000E 0010 0012


Press any key to exit...
_
```

Subtraction

```
.model small

.386

.data

array1 db 1H,2H,3H,4H,5H,6H,7H,8H,9h

array2 db 0H,1H,2H,3H,4H,5H,6H,7H,8H

result dw 9 dup (?)

.code

.startup

MOV AX, @data

MOV DS, AX

MOV CX, 09H
MOV DI, OFFSET array1

MOV BX, OFFSET array2

MOV SI, OFFSET result


back:

MOV AH, 0

MOV AL, [DI]

SUB AL, [BX]

SBB AH, 00

MOV [SI], AX

INC DI

INC BX

INC SI

INC SI

loop back


MOV SI, OFFSET result

MOV DH, 9


l10:

MOV CH, 04H

MOV CL, 04H

MOV BX, [SI]


l2:

ROL BX, CL
```

```
MOV DL, BL

AND DL, 15

CMP DL, 09

JBE l4

ADD DL, 07H


l4:

ADD DL, 30H

MOV AH, 02

INT 21H

DEC CH

JNZ l2

MOV DL, ' '

INT 21H

INC SI

INC SI

DEC DH

JNZ l10


MOV AH, 4CH

INT 21


.EXIT

END
```

#output



```
0001 0001 0001 0001 0001 0001 0001 0001 0001


Press any key to exit...
```

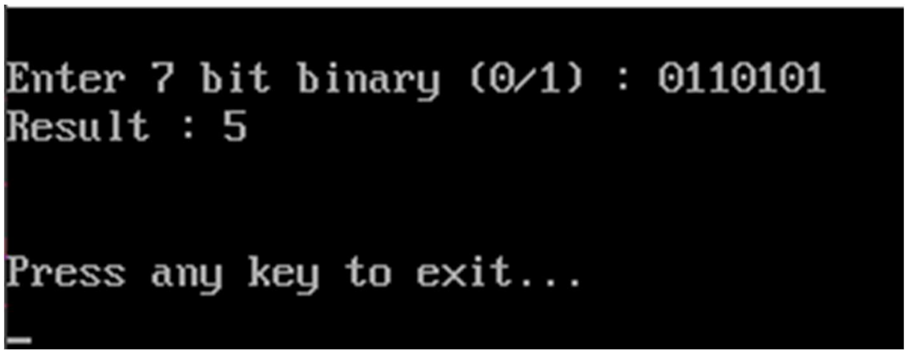
Q6) Write a program for binary to ASCII conversion.

Ans

```
.model small
.data
msg1 db 10,13,"Enter 7 bit binary (0/1) : $"
msg2 db 10,13,"Result : $"
bin db 0
.code
.startup
mov dx,offset msg1
mov ah,9
int 21h

mov bl,0

mov cx,7

getbin:
mov ah,1
int 21h
cmp al,'2'
jge ext
sub al,48
shl bl,1
```

```
add bl,al
loop getbin

mov ah,9
mov dx,offset msg2
int 21h

mov dl,bl
mov ah,2
int 21h

ext:
mov ah,4CH
int 21h

.exit
end
```

#output



```
Enter 7 bit binary (0/1) : 0110101
Result : 5


Press any key to exit...
_
```

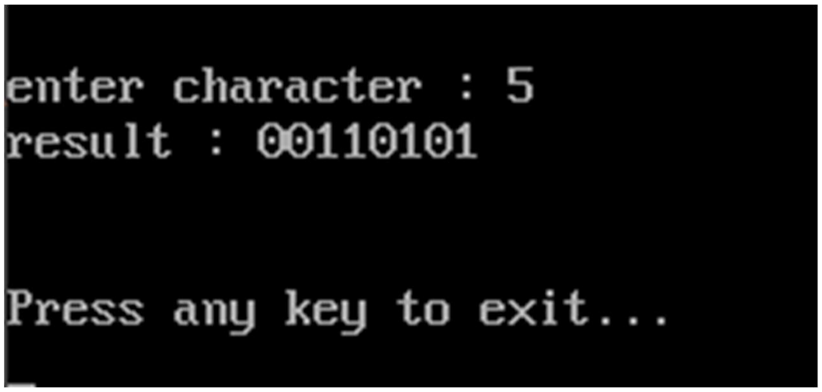
Q7) Write a program for ASCII to binary conversion.

Ans

```
.model small
.data
num1 db 0
msg1 db 10,13,"enter character : $"
msg2 db 10,13,"result : $"

.code
.startup
mov dx,offset msg1
mov ah,9
int 21h

mov ah,01
int 21h
mov num1,al

mov dx,offset msg2
mov ah,9
int 21h

mov cx,8

prt:
mov al,num1
mov bl,80h
and bl,al
cmp bl,0
je  prt0
mov dl,'1'
jmp prt1
prt0:
mov dl,'0'
```

```
prt1:
mov ah,2
int 21h
mov al,num1
shl al,1
mov num1,al
loop prt

ext:
mov ah,4ch
int 21h
.exit
end
```

#output



```
enter character : 5
result : 00110101


Press any key to exit...
```