**Ex.No.1 DATA TRANSFER Date:**

**Aim:**

**A. Write an ALP to transfer the data stored in consecutive memory locations, in the forward direction.**

**B. Write an ALP to transfer the data stored in consecutive memory locations, in the reverse direction.**

**C. Write an ALP to transfer the data stored in consecutive memory locations, in the forward direction overlapping.**

**Software Used: Computer system with TASM.**

**PROGRAM:**

**A. Forward Direction**

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

ARR1 DB 0H,1H,2H,3H,4H,5H,6H,7H,8H,9H

COUNT EQU 10D

ORG 3000H

ARR2 DB 10D DUP (0H)

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV SI, 2000H

MOV DI, 3000H

MOV CX, COUNT

BACK: MOV AH, [SI]

MOV [DI], AH

INC SI

INC DI

LOOP BACK

MOV AH, 4CH

INT21H

CODE ENDS

END START

**B.Reverse Direction**

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

ARR1 DB 0H,1H,2H,3H,4H,5H,6H,7H,8H,9H

COUNT EQU 10D

ORG 3000H

ARR2 DB 10D DUP (0H)

DATA ENDS

CODE SEGMENT

START: MOV AX, DATA

MOV DS, AX

MOV SI, 2000H

MOV DI, 3000H

MOV CX, COUNT

ADD DI, COUNT-1

BACK: MOV AH, [SI]

MOV [DI], AH

INC SI

DEC DI

LOOP BACK

MOV AH, 4CH

INT21H

CODE ENDS

END START

**C.Forward Direction with Overlapping**

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

ARR1 DB 0H,1H,2H,3H,4H,5H,6H,7H,8H,9H

COUNT EQU $-ARR1

OVERLAP EQU 06D

ORG 3000H

ARR2 DB 10D DUP (0H)

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV SI, 2000H

MOV DI, 3000H

MOV CX, COUNT-OVERLAP

NXTP: MOV AH, [SI]

MOV [DI], AH

INC SI

INC DI

LOOP NXTP

AGAIN: LEA SI, ARR1

MOV CX, COUNT

MOV AH, [SI]

MOV [DI], AH

INC SI

INC DI

LOOP AGAIN

MOV AH, 4CH

INT21H

CODE ENDS

END START

**Result: Data stored in consecutive memory locations is transferred from 2000h memory**

**location to 3000h memory location (a) in the forward direction (i.e. in the same order), (b) in**

**the reverse direction and (c) with overlapping in the forward direction.**

**Ex.No.2 ARITHMETIC OPERATIONS Date:**

**Aim:**

**A. Write an ALP to perform addition on 16-bit data stored in consecutive memory locations and store the result from the next location onwards.**

**B. Write an ALP to perform subtraction on 16-bit data stored in consecutive memory locations and store the result from the next location onwards.**

**C. Write an ALP to perform multiplication on 16-bit data stored in consecutive memory locations and store the result from the next locations onwards.**

**D. Write an ALP to perform division on 16-bit data stored in consecutive memory locations and store the result from the next location onwards.**

**Software Used: Computer system with TASM.**

**PROGRAM:**

1. **Addition**

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

ADDEND DW 8765H

ADDER DW 9876H

SUM DW 0H

CARRY DB 0H

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV AX, ADDEND

ADD AX, ADDER

JNC SKIP

INC CARRY

SKIP: MOV SUM, AX

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**B. Subtraction**

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

SUBTRAHEND DW 8765H

SUBTRACTOR DW 9876H

DIFFERENCE DW 0H

BARROW DB 0H

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV AX,SUBTRAHEND

SUB AX, SUBTRACTOR

JNC SKIP

INC BARROW

SKIP: MOV DIFFERENCE, AX

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**C.Multiplication**

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

MULTIPLICANT DW 0FFFFH

MULTIPLIER DW 123AH

RES DD 0H

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV AX, MULTIPLICANT

MOV BX, MULTIPLIER

MUL BX

MOV WORD PTR [RES], AX

MOV WORD PTR [RES+2], DX

MOV AH, 4CH

INT21H

CODE ENDS

END START

**D.Division**

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

DIVIDEND DW 8765H

DIVISOR DW 1234H

QUOTIENT DW 0H

REMAINDER DW 0H

DATA ENDS

CODE SEGMENT

START: MOV AX,DATA

MOV DS, AX

MOV AX, DIVIDEND

MOV BX, DIVISOR

DIV BX

MOV QUOTIENT, AX

MOV REMAINDER, DX

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**Result:- An ALP is written to perform a) addition, (b) subtraction, (c) multiplication and (d) division operations using arithmetic instructions and the same is verified.**

**Ex.No.3 LOGICAL OPERATIONS Date:**

**Aim:**

**A. Write an ALP to find number of 1’s in a given word.**

**B. Write an ALP to find the number of even and odd numbers in the given array.**

**C. Write an ALP to find the number of elements in the array having “1” in their 5 th bit position.**

**Software Used: Computer system with TASM.**

**PROGRAM:**

**A. Number of 1’s in a word**

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

NUM DW 5464

COUNT EQU 16D

BITCOUNT DB 0H

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV CL, COUNT

MOV AX, NUM

NXTP: ROR AX, 01H

JNC GO

INC BITCOUNT

GO: DEC CL

JNZ NXTP

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**B.Number of even and odd numbers**

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

SERIES DW 3456H,4533H,1234H,1567H,0FFFFH,145AH,56D7,4E34H,3421H,

89C5H

COUNT EQU 0 AH

ODDCOUNT DB 00H

EVENCOUNT DB 00H

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

LEA SI, SERIES

MOV CL, COUNT

NXTP: MOV AX, [SI]

ROR AX, 01H

JC ODD

INC EVENCOUNT

JMP OTHER

ODD: INC ODDCOUNT

OTHER: INC SI

DEC CL

JNZ NXTP

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**C.Number of elements having 1’s in their 5th bit position**

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

SERIES DB 21H,54H,05H,34H,32H,14H,18H,17H,53H,58H

COUNT EQU 0AH

BITCOUNT DB 00H

DATA ENDA

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

LEA SI, SERIES

MOV CL, COUNT

NXTP: MOV AX, [SI]

TEST AX, 10H

JZ GO

INC BITCOUNT

GO: INC SI

DEC CL

JNZ NXTP

MOV AH, 4CH

INT 21H

CODE ENDS

END START,

**Result: Logical operations such as Shift, rotate and test are used to find the a) no. of 1’s in the given byte, b) no. of even and add numbers, c)no. of positive and negative numbers and d) no. of elements having 1’s in their 5 th bit position.**

**Ex.No.4 STRING MANIPULATIONS Date:**

**Aim:**

**A. Write an ALP to transfer the data in forward direction using string instructions.**

**B. Write an ALP to transfer the data in reverse direction using string instructions.**

**Software Used: Computer system with TASM.**

**PROGRAM:**

**A. Forward Direction using String Instructions**

ASSUME DS: DATA, CS: CODE, ES: EXTRA

DATA SEGMENT

ORG 2000H

STRING1 DB 'MICROPROCESSOR'

COUNT EQU $-STRING1

DATA ENDS

EXTRA SEGMENT

ORG 3000H

STRING2 DB 14D DUP (00H)

EXTRA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV AX, EXTRA

MOV ES, AX

MOV CX, COUNT

LEA SI, STRING1

LEA DI, STRING2

CLD

REP MOVSB

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**B.Reverse Direction using String Instructions**

ASSUME DS: DATA, CS: CODE, ES: EXTRA

DATA SEGMENT

ORG 2000H

STRING1 DB 'MICROPROCESSOR'

LENGTH\_STRING DW $-STRING1

DATA ENDS

EXTRA SEGMENT

ORG 3000H

STRING2 DB 14D DUP (00H)

EXTRA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV AX, EXTRA

MOV ES, AX

LEA SI, STRING1

LEA DI, STRING2

MOV CX, LENGTH\_STRING

ADD DI, CX

DEC DI

BACK: MOVSB

SUB DI, 02H

LOOP BACK

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**Result: String manipulations such as a) forward string, b) reverse string.**

**Ex.No.5 SORTING Date:**

**Aim:**

**A. Write an ALP to sort the given array in signed ascending order.**

**B. Write an ALP to sort the given array in unsigned descending order.**

**C. Write an ALP to find the maximum and the minimum element in the given array.**

**Software Used: Computer system with TASM.**

**PROGRAM:**

**A. Signed ascending order**

ASSUME DS: DATA, CS: CODE

DATA SEGMENT

ORG 3000H

ARRAY DB 03H,07H,05H,01H,09H,04H,06H,02H,08H

COUNT EQU 09H

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV DX, COUNT

DEC DX

BEGIN: MOV CX, DX

LEA SI, ARRAY

BACK: MOV AL, [SI]

CMP AL,[SI+01 ]

JNZ SKIP

XCHG AL, [SI+01 ]

XCHG AL, [SI]

SKIP: INC SI

LOOP BACK

DEC DX

JNZ BEGIN

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**B.Unsigned descending order**

ASSUME DS: DATA, CS: CODE

DATA SEGMENT

ORG 3000H

ARRAY DB 03H,07H,05H,01H,09H,04H,06H,02H,08H

COUNT EQU 09H

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV DX, COUNT

DEC DX

BEGIN: MOV CX, DX

LEA SI, ARRAY

BACK: MOV AL, [SI]

CMP AL,[SI+01 ]

JNB SKIP

XCHG AL, [SI+01 ]

XCHG AL, [SI]

SKIP: INC SI

LOOP BACK

DEC DX

JNZ BEGIN

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**C.Maximum and Minimum elements**

**ASSUME CS: CODE, DS: DATA**

DATA SEGMENT

ORG 2000H

ARRAY DW 5555H,9999H, 7777H, 2222H, 1111H, 8888H, 6666H

A\_LENTH EQU ($-ARRAY)/2

MAX\_NO DW 0H

MIN\_NO DW 0H

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV CX, A\_LENTH-1

LEA SI, ARRAY

MOV AX, [SI]

MOV BX, AX

BACK: INC SI

INC SI

CMP AX, [SI]

JNC SKIP

MOV AX, [SI]

JMP NEXT

SKIP: CMP BX, [SI]

JC NEXT

MOV BX, [SI]

NEXT: LOOP BACK

MOV MAX\_NO,AX

MOV MIN\_NO, BX

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**Result: Sorting the array in a) signed ascending order, b) unsigned descending order and c) searching for the maximum and minimum elements in the unsigned array, are implemented.**

**Ex.No.6 PROCEDURE AND MACROS Date:**

**Aim:**

**A. Write an ALP to convert Hexadecimal numbers to BCD numbers.**

**B. Write an ALP to find the factorial of given number.**

**Software Used: Computer system with TASM.**

**Flow Chart:**

**PROGRAM**

**A. Hexadecimal to BCD conversion**

ASSUME DS: DATA, CS: CODE

DATA SEGMENT

ORG 2000H

HEXA DB 25H, 57H, 89H, 0A4H

COUNT EQU 04

DECI DB 12 DUP (00H)

BCD DW 04 DUP (00H)

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

MOV SI, OFFSET HEXA

MOV DI, OFFSET DECI

MOV CX, COUNT

BACK: XOR AX, AX

MOV AL, [SI]

MOV BL, 64H

DIV BL

MOV [DI], AL

INC DI

MOV AL, AH

XOR AH, AH

MOV BH, 0AH

DIV BH

MOV [DI], AL

INC DI

MOV [DI], AH

INC SI

INC DI

LOOP BACK

CALL UP2P

MOV AH, 4CH

INT 21H

UP2P PROC NEAR

MOV CH,COUNT

MOV DI,OFFSET DECI

MOV BP,OFFSET BCD

CONTINUE: XOR AX,AX

MOV AH,[DI]

INC DI

MOV AL,[DI]

MOV CL,4

SHL AL,CL

INC DI

ADD AL,[DI]

MOV DS:BP,AX

INC DI

INC BP

INC BP

DEC CH

JNZ CONTINUE

RET

UP2P ENDP

CODE ENDS

END START

**B.Factorial of a given number**

FACTORIAL MACRO

XOR CX, CX

XOR AX, AX

INC AX

MOV CL, NUMBER

CMP CL, 0

JE GO

REPEAT: MUL CX

LOOP REPEAT

GO: MOV FACT, AX

ENDM

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

ORG 2000H

NUMBER DB 08H

FACT DW 0

DATA ENDS

CODE SEGMENT

START: MOV AX, DATA

MOV DS, AX

FACTORIAL

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**Result: Programs on a) Procedure for converting Hexadecimal numbers to packed BCD and b) Macro for finding the factorial of a given number are implemented.**

**Ex.No.7 INTERRUP TS Date:**

**Aim:**

**A. Write an ALP to find whether the given string is a palindrome or not.**

**B. Write an ALP to enter the string through keyboard and display it.**

**Software Used: Computer system with TASM.**

**PROGRAM:**

**A. String is palindrome or not**

ASSUME DS: DATA, CS: CODE

DATA SEGMENT

ORG 2000H

STRING DB 'RADAR'

COUNT EQU $-STRING-1

MSG1 DB "STRING IS PALINDROME $"

MSG2 DB "STRING IS NOT PALINDROME $"

DATA ENDS

CODE SEGMENT

ORG 1000H

START: MOV AX, DATA

MOV DS, AX

LEA SI, STRING

LEA DX, MSG2

MOV CX, COUNT

MOV DI, SI

ADD DI, COUNT-1

SHR CX, 1

BACK: MOV AL, [SI]

CMP AL, [DI]

JNZ NEXT

INC SI

DEC DI

LOOP BACK

LEA DX, MSG1

NEXT: MOV AH, 09H

INT 21H

MOV AH, 4CH

INT 21H

CODE ENDS

END START

**B.Enter and Display the string**

ASSUME DS: DATA, CS: CODE

DATA SEGMENT

ORG 2000H

INPUT DB “ENTER THE STRING”, 0DH, 0AH, “$”

OUTPUT DB “THE ENTERED STRING IS: ” 0DH, 0AH, “$”

S\_LENTH DB 0

BUFFER DB 80 DUP (0)

DATA ENDS

CODE SEGMENT

START: MOV AX, DATA

MOV DS, AX

XOR CL, CL

LEA DX, INPUT

MOV AH, 09H

INT 21H

LEA BX, BUFFER

REPEAT: MOV AH, 01H

INT 21H

CMP AL, 0DH

JZ EXIT

INC CL

MOV [BX], AL

INC BX

JMP REPEAT

EXIT: MOV S\_LENTH, CL

LEA BX, BUFFER

ADD BL, CL

MOV AL,’$’

MOV [BX], AL

LEA DX, OUTPUT

MOV AH, 09H

INT 21H

LEA DX, BUFFER

MOV AH, 09H

INT 21H

CODE ENDS

ENDF START

Result: Interrupts on a) whether the given string is a palindrome or not and b) entering the string through

keyboard and displaying the same, are verified.