1. **(i) .Write an alp to move a block of data (byte) from source to destination**

**Without overlap.**

.MODEL SMALL ;

.DATA ;

X DB 11H, 22H, 33H, 44H, 55H ;

Y DB 05 DUP(?) ;

.CODE ; [initialize

MOV AX, @DATA ; data segment]

MOV DS, AX ;

MOV CX, 5 ; count = 05h

LEA BX,X ; initialize BX for X

LEA SI,Y ; initialize destination index for Y

L1: MOV AL,[BX] ; get the number in al register from source

MOV [SI], AL ; get the number in destination from al reg

INC BX ; increment pointer

INC SI ; increment pointer

LOOP L1 ; count=0,if NO go to L1,YES come out of L1

MOV AH, 4CH ; [program

INT 21H ; termination]

END

**RESULT:**

**Before Execution:**

**Source :** **11h,22h,33h,44h,55h**

**Destination:** **00h,00h,00h,00h,00h**

**After Execution:**

**Source :** **11h,22h,33h,44h,hh5**

**Destination:** **11h,22h,33h,44h,55h**

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load data & length

Step4: initialize the counters

Step5: load the effective address of memory x into BX, and y into SI

Step6: move data to al pointed by BX.

Step7: move data from al to address pointed by SI.

Step8: increment SI & BX

Step9: repeat till counter becomes zero

Step10: Terminate program

1. **(ii).Write an alp to move a block of data (word) from source to destination**

**without overlap.**

.MODEL SMALL

.DATA

X DW 1111H, 2222H, 3333H, 4444H, 5555H

Y DW 05 DUP(?)

.CODE ;

MOV AX, @DATA ; [initialize

MOV DS, AX ; data segment]

MOV CX, 05h ; count = 05h

LEA BX,X ; initialize BX for X

LEA SI,Y ; initialize destination index for Y

L1: MOV AX,[BX] ; get the number in ax register from source

MOV [SI], AX ; get the number in destination from al reg

ADD BX, 02H ; increment pointer

ADD SI, 02H ; increment pointer

LOOP L1 ; count=0,if NO go to L1,YES come out of L1

MOV AH,4CH ; [program

INT 21H ; termination]

END

**RESULT:**

**Before Execution:**

**Source :** **1111h,2222h,3333h,4444h,5555h**

**Destination:** **0000h,0000h,0000h,0000h,0000h**

**After Execution:**

**Source :** **1111h,2222h,3333h,4444h,5555h**

**Destination:** **1111h,2222h,3333h,4444h ,5555h**

;

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load data & length

Step4: initialize the counters

Step5: load the effective address of memory x into BX, and y into SI

Step6: move data to AX pointed by BX.

Step7: move data from AX to address pointed by SI.

Step8: increment SI & BX ,by two times

Step9: repeat till counter becomes zero

Step10: Terminate program

1. **(i) Write an alp to move a block of data (byte) from source to destination**

**with overlap.**

.MODEL SMALL

.DATA

Y DB 03 DUP(0)

X DB 11H,22H,33H,44H,55H

.CODE

mov ax,@data; [initialize

mov ds,ax; data segment]

MOV CX, 5 ; count = 05h i,e length of the block of data

LEA SI,X ; initialize source index for src

LEA DI,Y ; initialize destination index for dst

L1: MOV AL,[SI] ; get the number in al register from source

MOV [DI], AL ; get the number in destination from al reg

INC SI ; Increment pointer

INC DI ; Increment pointer

LOOP L1 ; count=0,if NO go to L1,YES come out of loop

MOV AH,4CH ; [program

INT 21H ; termination]

END

**RESULT:**

**Before Execution:**

**Source :** **11h,22h,33h,44h,55h**

**After Execution:**

**Source : 11h,22h,33h,44h,55h,33h, 44h,55h**

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load data & length

Step4: initialize the counters

Step5: load the effective address of memory x into SI, and y into DI

Step6: move data to AL from location pointed by SI.

Step7: move data from AL to address pointed by DI.

Step8: increment SI & DI

Step9: repeat till counter becomes zero

Step10: Terminate program

**2. (ii) Write an alp to move a block of data (word) from source to destination**

**with overlap.**

.MODEL SMALL

.DATA

Y DW 03H DUP (0)

X DW 1111H, 2222H, 3333H, 4444H, 5555H

.CODE

MOV AX, @DATA ; [initialize

MOV DS, AX ; data segment]

MOV CX, 5 ; count = 05h i,e length of the block of data

LEA SI, X ; initialize source index for src

LEA DI, Y ; initialize destination index for dst

L1: MOV AX,[SI] ; get the number in al register from source

MOV [DI], AX ; get the number in destination from al reg

ADD SI, 02H ; Increment pointer by twice

ADD DI, 02H ; Increment pointer by twice

LOOP L1 ; count=0,if NO go to L1,YES come out of loop

MOV AH, 4CH ; [program

INT 21H ; termination]

END

**RESULT:**

**Before Execution:**

**Source :** 11**11h,2222h,3333h,4444h,5555h**

**After Execution:**

**Source : 1111h,2222h,3333h,4444h,5555h,3333h, 4444h,5555h**

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load data & length

Step4: initialize the counters

Step5: load the effective address of memory x into SI, and y into DI

Step6: move data to AX pointed by SI.

Step7: move data from AX to address pointed by DI.

Step8: increment SI & DI by two times

Step9: repeat till counter becomes zero

Step10: Terminate program

1. **(i) Write an alp to interchange a block of data (byte) between source and destination.**

.MODEL SMALL

.DATA

X DB 01H, 02H, 03H, 04H , 05H

Y DB 10H, 20H, 30H, 40H , 50H

.CODE

MOV AX, @DATA ; [initialize

MOV DS, AX ; data segment]

MOV CX,05 ;count = 05h

LEA SI,X ; initialize source index for source

LEA DI,Y

L1: MOV AL,[SI]

XCHG AL,[DI]

MOV [SI], AL

INC SI

INC DI

LOOP L1

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load data & length

Step4: initialize the counters

Step5: load the effective address of memory x into SI, and y into DI

Step6: move data to AL pointed by SI.

Step7: exchange data from AL to address pointed by DI.

Step8: move data from AL to memory location pointed by SI

Step9: increment SI & DI

Step10: repeat till counter becomes zero

Step11: Terminate program

**3. (ii) Write an alp to interchange a block of data (word) between source and**

**destination.**

.MODEL SMALL

.DATA

X DW 1111H, 2222H, 3333H, 4444H , 5555H

Y DW 0AAAAH, 0BBBBH, 0CCCCH, 0DDDDH , 0EEEEH

.CODE

MOV AX, @DATA

MOV DS, AX

MOV CX,05

LEA SI,X

LEA DI,Y

L1: MOV AX,[SI]

XCHG AX,[DI]

MOV [SI], AX

ADD SI,02H

ADD DI,02H

DEC CX

JNZ L1

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load data & length

Step4: initialize the counters

Step5: load the effective address of memory x into SI, and y into DI

Step6: move data to AX pointed by SI.

Step7: exchange data from AX to address pointed by DI.

Step8: move data from AX to memory location pointed by SI

Step9: add SI & DI with 2

Step10: repeat till counter becomes zero

Step11: Terminate program

1. **Write an alp to perform byte/word data transfer in different addressing modes.**

.MODEL SMALL

.DATA

X DW 1342H

Y DW 01 DUP(?)

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AX,1506H

MOV BX,AX

MOV DX,X

LEA BX,X

MOV CX,[BX]

MOV [DI+2],AX

MOV AX,[BX+SI]

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2:Initialize data segment

Step3: Load data into Register AX

Step4: Move the content of AX into BX register

Step5:Load data in memory location x into DX register

Step6: Load effective address of x in to BX

Step7: Move data pointed by BX into CX register

Step8: Move data from AX to location to DI+2

Step9: Move data stored in location pointed by [BX+SI] to AX register

Step10: Terminate the Program

1. **Write an alp to convert BINARY to ASCII value.**

.MODEL SMALL

.DATA

BIN DB 0AH

ASC DB 1 DUP (?)

.CODE

MOV AX, @DATA

MOV DS, AX

MOV DL, BIN

CMP DL, 0AH

JC L1

ADD DL, 07H

L1: ADD DL, 30H

MOV ASC, DL

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move data from BIN to DL

Step4: Compare DL content with 0AH data

Step5: If carry is generated, add 30H, else add 37H

Step6: Store result into ASC location from DL register

Step7: Terminate the Program

1. **Write An Alp To Convert ASCII to BINARY Value.**

.MODEL SMALL

.DATA

ASC DB 39H

BIN DB 1 DUP(?)

.CODE

MOV AX, @DATA

MOV DS, AX

MOV DL, ASC

CMP DL, 03AH

JC L1

SUB DL, 37H

LOOP L2

L1: SUB DL, 30H

L2: MOV BIN, DL

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move data from ASI to DL

Step4: Compare DL content with 3AH data

Step5: If carry is generated, subtract 30H and store result in Bin

Step6: If no carry subtract 37H and store result in Bin

Step7: Terminate the Program

1. **Write an alp to convert BCD to BINARY (byte).**

.MODEL SMALL

.DATA

X DB 98H

Y DB ?

.CODE

MOV AX, @DATA

MOV DS,AX

MOV AL,X

MOV CL, 4

SHR AL, CL

MOV BL,X

AND BL,0FH

MOV CL, 0AH

MUL CL

ADD AL, BL

MOV Y, AL

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move data from X to AL register

Step4: Initialize counter (CL)

Step5: unpack the number

Step6: Store first number in AL register

Step7: Multiply the number stored in AL by 10

Step8: Add AL content with BL register

Step9: Move result to y memory location

Step10: Terminate the Program

1. **(i) Write an alp to convert BINARY to BCD.(Word)**

.MODEL SMALL

.DATA

BIN DW 0FFFFH

BCD DW 2 DUP(?)

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AX, BIN

MOV BX, 10000

MOV DX, 0

DIV BX

MOV BCD, AX

MOV AX, DX

MOV DX, 0

MOV BX, 1000

DIV BX

MOV CL, 04

ROR AL, CL

MOV BCD+1, AL

MOV AX, DX

MOV DX, 0

MOV BX, 100

DIV BX

ADD BCD+1, AX

MOV AX, DX

MOV DX, 0

MOV BX, 10

DIV BX

MOV CL, 04

ROR AL, CL

ADD AL, DL

MOV BCD+2, AL

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move data from X to register AX

Step4: move data 10,000 and 00 into registers BX and DX respectively

Step5: Divide data in DX- AX with BX content

Step6: Move data in AX into locations BCD

Step7: Move data in DX into register AX register

Step8: Move data 00 and 1000 into registers DX and BX respectively

Step9: Divide data in DX- AX with BX content

Step 10: Move data 04 into register CL and rotate right data in AL by 4 times

Step11: Move data in AL into location BCD+1

Step12: move data in register DX into register AX

Step13: Move data 00 and 100 into registers DX and BX respectively

Step14: Divide data in DX- AX with BX content

Step15: Add data in AX with data in BCD+1 and move data in AX into location Y+1

Step16: Move data register DX into register AX

Step17: Move data 00 and 10 into registers DX and BX respectively

Step18: Divide data in DX-AX with AX

Step19: Move data 04 into register CL

Step20: Rotate right data in AL by 4-times

Steps21: Logical ‘or’ the data in AL by DL

Step22: move data in AL to BCD+2

Steps23: Terminate the program

**8. (ii) Write an alp to convert BCD to BINARY (word).**

.MODEL SMALL

.DATA

X DW 9876H

Y DW ?

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AX, X

MOV CL, 12

SHR AX, CL

MOV BX, 1000

MUL BX

ADD Y, AX

MOV AX, X

MOV CL, 08

SHR AX, CL

AND AL, 0FH

MOV BX, 100

MUL BX

ADD Y, AX

MOV AX, X

MOV CL, 4

SHR AX, CL

AND AX, 00FH

MOV BX, 10

MUL BX

ADD Y, AX

MOV AX, X

AND AX, 000FH

ADD Y, AX

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move data from BIN to register AX

Step4: Move cl with data 12 and perform shift right operation on AX content by 12 times

Step5: Move data 00 and 1000 into registers DX and BX respectively

Step6: Multiply data in DX- AX registers with BX register content

Step7: Store or move result in BCD memory location

Step8: Move data 08 into register CL and shift right data in AL by 8 times

Step9: Move data in AL into location BCD+1

Step10: move data in register DX into register AX

Step11: Move data 00 and 100 into registers DX and BX respectively

Step12: Multiply data in DX- AX with BX

Step13: Add data in AX with data in BCD+1 and move data in AX into location Y+1 location

Step13: Move data register DX into register AX

Step15: Move data 04 into register CL and shift right data in AL by 4 times

Step16: Move data 00 and 10 into registers DX and BX respectively

Step17: Multiply data in DX-AX with AX

Step18: Logical or data in AL by DL

Step19: move data in AL to BCD+2

Step20: Terminate the program

1. **(i) Write an alp to sort an array in ascending order using bubble sort technique.(byte)**

.MODEL SMALL

.DATA

X DB 55H,45H,35H, 25H, 15H

LEN DW($-X)

.CODE

MOV CX,@DATA

MOV DS, CX

MOV CX, LEN

DEC CX

L3: MOV DX,CX

LEA SI,X

L2: MOV AL,[SI]

CMP AL, [SI+1]

JC L1

XCHG AL, [SI+1]

MOV [SI], AL

L1: INC SI

DEC DX

JNZ L2

LOOP L3

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: obtain length: number of elements

Step4: load counters CX with length

Step5: Decrement CX

Step6: Move CX to DX

Step7: load effective address of X in to SI register

Step8: move contents location pointed SI to AL register

Step9: Compare with next value. If carry is generated then increment SI and decrement DX and repeat the loop till counter ends

Step10: If no carry is generated then exchange the contents and store back in Xarray with next element to it and go back to step8

Step11: Terminate the Program

**9. (ii) Write an alp to sort an array in descending order using bubble sort technique.(byte)**

.MODEL SMALL

.DATA

X DB 15H,25H,35H, 45H, 55H

LEN DW($-X)

.CODE

MOV CX,@DATA

MOV DS, CX

MOV CX, LEN

DEC CX

L3: MOV DX,CX

LEA SI,X

L2: MOV AL,[SI]

CMP AL, [SI+1]

JNC L1

XCHG AL, [SI+1]

MOV [SI], AL

L1: INC SI

DEC DX

JNZ L2

LOOP L3

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: obtain length: no of elements

Step4: load counters CX with length

Step5: Decrement CX

Step6: Move CX content to DX register

Step7: load effective value of X into SI register

Step8: move contents pointed by location SI in AL register

Step9: Compare with next value. If carry is generated then increment SI and decrement DX and repeat the loop till counter ends

Step10: If carry is generated then exchange the contents and store back in Xarray with next element to it and go back to step8

Step11: Terminate the program

**9.(iii) Write an alp to sort an array in ascending order using bubble sort technique(word)**

.MODEL SMALL

.DATA

X DW 5555H, 4444H, 3333H, 2222H, 1111H

LEN DW ($-X)/2

.CODE

MOV AX, @DATA

MOV DS, AX

MOV CX, LEN

DEC CX

L3:MOV DX, CX

LEA SI, X

L2:MOV AX, [SI]

CMP AX, [SI+2]

JC L1

XCHG AX, [SI+2]

MOV [SI], AX

L1:INC SI

INC SI

DEC DX

JNZ L2

LOOP L3

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: obtain length: number of elements

Step4: load counters CX with length

Step5: Decrement CX

Step6: Move CX content to DX register

Step7: load effective value of X in SI register

Step8: move contents location pointed by SI in AL register

Step9: Compare with next value. If carry is generated then increment SI twice and decrement DX

and repeat the loop till counter-1 ends

Step10: If no carry, then exchange the contents and store back in X and repeat the step till counter ends

Step11: Terminate the program

1. **(iv) Write an alp to sort an array in descending order using bubble sort technique.(word)**

.MODEL SMALL

.DATA

X DW 1111H, 4444H, 2222H, 5555H, 3333H

LEN DW ($-X)/2

.CODE

MOV AX, @DATA

MOV DS, AX

MOV CX, LEN

DEC CX

L3:MOV DX, CX

LEA SI, X

L2:MOV AX, [SI]

CMP AX, [SI+2]

JNC L1

XCHG AX, [SI+2]

MOV [SI], AX

L1:INC SI

INC SI

DEC DX

JNZ L2

LOOP L3

MOV AH, 4CH

INT 21H

END

1. **Write an alp to check whether the given number is odd or even.**

.MODEL SMALL

.DATA

NUM DB 01H

MSG1 DB "IT IS EVEN $"

MSG2 DB "IT IS ODD $"

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AL, NUM

ROR AL, 1

JC NEXT

LEA DX, MSG1

JMP NEXT1

NEXT:LEA DX, MSG2

NEXT1:MOV AH, 09H

INT 21H

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move data from num to register AL

Step4: Rotate right the data in AL by one bit

Step5: If carry flag is set then display the message saying that the given no is odd

Step6: If carry flag is not stet then display the message saying that the given no is even

Step7: Terminate the program

1. **Write an alp to check whether the given number is positive or negative.**

.MODEL SMALL

.DATA

NUM DB 88H

MSG1 DB "IT IS POSITIVE $"

MSG2 DB "IT IS NEGATIVE $"

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AL, NUM

ROL AL, 1

JC NEXT

LEA DX, MSG1

JMP NEXT1

NEXT:LEA DX, MSG2

NEXT1:MOV AH, 09H

INT 21H

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move data from num to register AL

Step4: Rotate left the data in AL by one bit

Step5: If carry flag is set then display the message saying that the given no is negative

Step6: If carry flag is not stet then display the message saying that the given no is positive

Step7: Terminate the program

1. **Write an alp to find the number of 1’s and 0’s in a given number.**

..model small

.data

src dw 0003h,10 dup(0)

.code

mov ax,@data ;[initialize

mov ds,ax ; data segment]

lea si,src ; initialize source index for src

mov ax,[si] ; get the number

mov cx,10h ;load the count=16

up:ror ax,01 ;rotate right the content ax by one bit

jc l1 ;go to l1 if cy=1

inc word ptr [si+2] ;increment the number of 0’s

l2: dec cx ; decrement count

jnz up ; repeat loop if zf=0

jmp exit ; jump un conditionally to exit

l1: inc word ptr [si+4] ; increment the number of 1’s

jmp l2 ; jump un conditionally to l2

exit: mov ah,4ch ;[program

int 21h

end ; termination]

.MODEL SMALL

.DATA

N DB 18H

ONES DB 1 DUP (?)

ZEROS DB 1 DUP (?)

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AL, X

MOV CX,08H

L2:ROR AL,01

JC INCONES

INC ZEROS

JMP L1

INCONES :INC ONES

L1:LOOP L2

MOV AH,4CH

INT 21H

END

**RESULT:**

**Source: 0003 h**

**The number of 0’s: 000e h**

**The number of 1’s: 0002 h**

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load counter CX: number of bits

Step4: move data in num to register AL

Step5: move data 00H into register BX

Step6: Rotate right contents of register AL by 1 bit

Step7: If carry flag is set then increment BH else increment BL, and repeat till counter loop ends

Step8: move contents of register BL to zeros and contents of register BH to ones

Step9: Terminate the program

1. **Write an alp to check the given number is 2 out of 5 code.**

.MODEL SMALL

.DATA

N DB 18H

MSG1 DB 13,10 "2 OUT OF 5 CODE $"

MSG2 DB 13,10 "NOT 2 OUT OF 5 CODE $"

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AL, X

AND AL, 0E0H

JNZ FAILS

MOV AL, N

MOV BL, 00H

MOV CX, 05

AGAIN: MOV AL, 1

JC INCBIT

JMP REPEAT

INCBIT:INC BL

REPEAT:LOOP AGAIN

CMP BL, 02H

JNZ FAILS

LEA DX, MSG1

JMP NEXT

FAILS:LEA DX, MSG2

NEXT:MOV AH, 09H

INT 21H

MOV AH, 4CH

INT 21H

END

1. **Write an alp to check whether the given number is bitwise palindrome or not.**

.MODEL SMALL

.DATA

NUM DB 50H

MSG1 DB "IT IS BITWISE PALINDROME $"

MSG2 DB "IT IS NOT BITWISE PALINDROME $"

.CODE

MOV AX, @DATA

MOV DS, AX

MOV CX, 08H

MOV BL, 00H

MOV AL, NUM

UP:ROR AL, 01H

RCL BL, 01H

LOOP UP

CMP AL, BL

JE DISPMSG1

LEA DX, MSG2

JMP DOWN

DISPMSG1:LEA DX, MSG1

DOWN:MOV AH, 09H

INT 21H

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load counter CX and move data 00H into BL

Step4: move contents in num to register AL

Step5: Rotate right contents of AL by 1 bit

Step6: Rotate left with carry contents of BL by 1 bit

Step7: Repeat the loop till CX and compare contents of AL with BL registers

Step8: If equal display the message it is palindrome

Step9: If not equal display the message it is not palindrome

Step10: Terminate the Program

1. **Write an alp to check whether the given number is nibble wise palindrome or not.**

.MODEL SMALL

.DATA

NUM DW 1818H

MSG1 DB "NIBBLEWISE PALINDROME$"

MSG2 DB "NOT A NIBBLEWISE PALINDROME$"

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AX, NUM

MOV CL, 04H

ROR AL, CL

CMP AH, AL

JE DISPMSG1

LEA DX, MSG2

JMP DOWN

DISPMSG1:LEA DX, MSG1

DOWN:MOV AH, 09H

INT 21H

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move contents in num to register AX

Step4: load counter CL i.e 04

Step5: Rotate right contents of AL, CL times

Step6: Compare contents in AH with AL register

Step7: If equal display the message Nibble wise palindrome

Step9: If not equal display the message not a nibble wise palindrome

Step10: Terminate the program

1. **(i) Write an alp to add N numbers in an array (BCD).**

.MODEL SMALL

.DATA

ARRAY DB 99H, 98H, 95H, 93H, 92H

LEN DW ($-ARRAY)

SUM DW 02H DUP(0)

.CODE

MOV AX,@DATA

MOV DS, AX

MOV CX, LEN

DEC CX

MOV AH, 00H

CLC

LEA SI, ARRAY

MOV AL, [SI]

NEXT: ADD AL,[SI+1]

DAA

ADC AH,00H

DAA

INC SI

LOOP NEXT

JC DISPARRAY

DISPARRAY: ADC AH, 00H

DAA

MOV SUM,AX

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load counter CX: number of array elements

Step4: move data 00H into register Ah

Step5: Clear the carry flag content

Step6: load effective address of array X in register SI and move the first element to AL register

Step7: Add contents of register AL with data pointed by SI location; perform decimal adjust after addition

Step8: Add with carry data in AH with 00H data; perform decimal adjust after addition

Step9: Increment SI and repeat till loop end counter

Step10: Move the result to sum location

Step11: Terminate the program

1. **(ii) Write an alp to subtract N numbers in an array(BCD)**

.MODEL SMALL

.DATA

ARRAY DB 0FAH, 12H, 0A4H, 56H, 0FFH

LEN DW ($-ARRAY)

DIFF DW 02H DUP(0)

.CODE

MOV AX,@DATA

MOV DS, AX

MOV CX, LEN

DEC CX

MOV AH, 00H

CLC

LEA SI, ARRAY

MOV AL, [SI]

NEXT: SUB AL,[SI+1]

DAS

SBB AH,00H

DAS

INC SI

LOOP NEXT

JC DISPARRAY

DISPARRAY: SBB AH, 00H

MOV DIFF,AX

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load counter CX and decrement CX by,

Step4: move data 00H into register AH

Step5: load effective address of array in register SI

Step6: Move data pointed by SI to register AL

Step7: Subtract data in AL wit data pointed by [SI]; perform decimal adjust after Subtraction

Step8: subtract with borrow content of AH with 00H data; perform decimal adjust after subtraction

Step8: Increment SI and loop L1

Step10: Move the result from AX to diff memory location

Step11: Terminate the program

1. **(i) Write an alp to add N numbers in an array (hexa).**

.MODEL SMALL

.DATA

ARRAY DB 0FAH, 12H, 0A4H, 56H, 0FFH

LEN DW ($-ARRAY)

SUM DW 02H DUP(0)

.CODE

MOV AX,@DATA

MOV DS, AX

MOV CX, LEN

DEC CX

MOV AH, 00H

CLC

LEA SI, ARRAY

MOV AL, [SI]

NEXT: ADD AL,[SI+1]

ADC AH,00H

INC SI

LOOP NEXT

JC DISPARRAY

DISPARRAY: ADC AH, 00H

MOV SUM,AX

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load counter CX: number of array elements

Step4: move data 00H into register Ah

Step5: Clear the carry flag content

Step6: load effective address of array X in register SI and move the first element to AL register

Step7: Add contents of register AL with data pointed by SI location

Step8: Add with carry data in AH with 00H data

Step9: Increment SI and loop L1

Step10: Move the result to sum location

Step11: Terminate the program

1. **(ii) Write an alp to subtract N numbers in an array (hexa).**

.MODEL SMALL

.DATA

ARRAY DB 0FAH, 12H, 0A4H, 56H, 0FFH

LEN DW ($-ARRAY)

DIFF DW 02H DUP(0)

.CODE

MOV AX,@DATA

MOV DS, AX

MOV CX, LEN

DEC CX

MOV AH, 00H

CLC

LEA SI, ARRAY

MOV AL, [SI]

NEXT: SUB AL,[SI+1]

SBB AH,00H

INC SI

LOOP NEXT

JC DISPARRAY

DISPARRAY: SBB AH, 00H

MOV DIFF,AX

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load counter CX and decrement CX by,

Step4: move data 00H into register AH

Step5: load effective address of array in register SI

Step6: Move data pointed by SI to register AL

Step7: Subtract data in AL wit data pointed by [SI]

Step8: subtract with borrow content of AH with 00H data

Step8: Increment SI and loop L1

Step10: Move the result from AX to diff memory location

Step11: Terminate the program

1. **(i) Write an alp for multi-byte addition of two 32 bit numbers( BCD)**

.MODEL SMALL

.DATA

X DB 12H,32H,64H,56H

Y DB 56H,64H,32H,12H

Z DB 4 DUP (?)

CARRY DB ?

.CODE

MOV AX,@DATA

MOV DS,AX

MOV SI,3

MOV CL,4

CLC

L1:MOV AL,X[SI]

ADC AL,Y[SI]

DAA

MOV Z[SI],AL

DEC SI

LOOP L1

MOV AH,00H

ADC AH,00H

MOV CARRY,AH

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load the counter in CX and Index SI with content counter-1

Step4: move the first index content of X into AL register

Step5: Add with carry content to AL with Indexed value of Y

Step6: perform Decimal adjust after addition

Step7: Store the result in Index value if Z memory location

Step8: Decrement SI and repeat the loop till Counter ends

Step9: Obtain the Carry content in DH and store the result in CARRY

Step10: Terminate the program

**18.(ii) Write an alp for multi-byte subtraction of two 32 bit numbers(bcd)**

.MODEL SMALL

.DATA

X DB 12H,32H,64H,56H

Y DB 56H,64H,32H,12H

Z DB 4 DUP (?)

CARRY DB ?

.CODE

MOV AX,@DATA

MOV DS,AX

MOV SI,3

MOV CL,4

CLC

L1:MOV AL,X[SI]

SBB AL,Y[SI]

DAS

MOV Z[SI],AL

DEC SI

LOOP L1

MOV DH,00H

ADC DH,00H

MOV CARRY, DH

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load the counter in CX and Index SI with content counter-1

Step4: move the first index content of X into AL register

Step5: Subtract with borrow content to AL with Indexed value of Y

Step6: perform Decimal adjust after Subtraction

Step7: Store the result in Index value if Z memory location

Step8: Decrement SI and repeat the loop till Counter ends

Step9: Obtain the Carry content in DH and store the result in CARRY

Step10: Terminate the program

1. **(i) Write an alp for multi-byte addition of two 32 bit numbers(hexa)**

.MODEL SMALL

.DATA

X DB 0ABH,96H,46H,6CH

Y DB 45H,94H,0CH,46H

Z DB 4 DUP (?)

CARRY DB ?

.CODE

MOV AX,@DATA

MOV DS,AX

MOV SI,3

MOV CL,4

CLC

L1:MOV AL,X[SI]

ADC AL,Y[SI]

MOV Z[SI],AL

DEC SI

LOOP L1

MOV AH,00H

ADC AH,00H

MOV CARRY,AH

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load the counter in CX and Index SI with content counter-1

Step4: move the first index content of X into AL register

Step5: Add with carry content to AL with Indexed value of Y

Step6: Store the result in Index value if Z memory location

Step7: Decrement SI and repeat the loop till Counter ends

Step8: Obtain the Carry content in DH and store the result in Carry location

Step9: Terminate the program

1. **(ii) Write an alp for multi-byte subtraction of two 32 bit numbers(hexa)**

.MODEL SMALL

.DATA

X DB 0ABH, 0BCH, 80H, 91H

Y DB 0BH, 0CBH, 0ABH, 41H

Z DB 4 DUP (?)

CARRY DB ?

.CODE

MOV AX,@DATA

MOV DS,AX

MOV SI,3

MOV CL,4

CLC

L1:MOV AL,X[SI]

SBB AL,Y[SI]

MOV Z[SI],AL

DEC SI

LOOP L1

MOV AH,00H

ADC AH,00H

MOV CARRY,AH

MOV AH,4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load the counter in CX and Index SI with content counter-1

Step4: move the first index content of X into AL register

Step5: Subtract with borrow content to AL with Indexed value of Y

Step6: Store the result in Index value if Z memory location

Step7: Decrement SI and repeat the loop till Counter ends

Step8: Obtain the Carry content in DH and store the result in CARRY

Step9: Terminate the program

1. **Write an alp to find the largest number in the given array of data.**

.MODEL SMALL

.DATA

ARRAY DB 56H, 32H, 43H, 15H, 72H

LEN DB($-ARRAY)

Y DB ?

.CODE

MOV AX, @DATA

MOV DS, AX

LEA SI, ARRAY

MOV CX, LEN

MOV AL, [SI]

L2:CMP AL, [SI+1]

JNC L1

MOV AL, [SI+1]

L1:INC SI

LOOP L2

MOV Y, AL

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: load the values of counter

Step4: move contents pointed by SI to AL register

Step5: Compare the contents of AL with Contents pointed by [SI+1] Memory location

Step6: Jump if no carry to next and increment SI, decrement CX and loop again if zero flag is not set

Step7: If carry flag is set then move contents pointed by [SI] to Al register and inl SI and repeat again

Step8: display the appropriate message

Step9: Terminate the program

1. **(ii) Write an alp to find the smallest number in the given array of data.**

.MODEL SMALL

.DATA

ARRAY DB 56H, 32H, 43H, 15H, 72H

LEN DB($-ARRAY)

Y DB ?

.CODE

MOV AX, @DATA

MOV DS, AX

LEA SI, ARRAY

MOV CX, LEN

DEC CX

MOV AL, [SI]

L2:CMP AL, [SI+1]

JC L1

MOV AL, [SI+1]

L1:INC SI

LOOP L2

MOV Y, AL

MOV AH, 4CH

INT 21H

END

1. **Write an alp to find the factorial of a given number.**

.MODEL SMALL

.DATA

NUM DB 06H

Y DW ?

.CODE

MOV AX, @DATA

MOV DS, AX

MOV BX, NUM

MOV CX, 01H

MOV AX, 01H

L1:MUL CX

INC CX

DEC BX

JNZ L1

MOV Y, AX

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move contents in num to BX

Step4: move data 01 to register CX

Step5: move data 01 to register AX

Step6: Multiply contents in AX content with contents in CX register

Step7: Increment CX and decrement BX

Step8: If zero flag is not set then go to loop l1

Step9: Move result contents in AX to location y

Step10: Terminate the program

**24. Write an alp to find the GCD of two numbers.**

.MODEL SMALL

.DATA

X DW 0024H, 0036H

Y DW 01H DUP (?)

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AX, X

MOV BX, X+2

CMP AX, BX

JE GCDFND

JC L1

AGN:MOV DX, 00H

DIV BX

CMP DX, 00H

JE GCDFND

MOV AX, BX

MOV BX, DX

JMP AGN

L1:XCHG AX, BX

JMP AGN

GCDFND:MOV Y, BX

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: Move contents of x location into register AX

Step4: Move contents of x+2 locations into register Bx

Step5: Compare data in register Ax and BX

Step6: Jump if equal to exit and move contents of Bx to gcd. If not equal jump to L1 and exchange the contents of Ax and Bx and jump to L2

Step7: Move data 00 into register Dx

Step8: Divide contents in Bx with contents in Ax

Step9: compare data in Bx with 00H. If equal jump to exit

Step10: Move contents in Bx to Ax and move contents in Dx to Bx and jump to L2

Step11: Terminate the Program

**25**. **Write an alp to find the LCM of two numbers.**

.MODEL SMALL

.DATA

NUM DW 0ABCDH,0FFFFH

LCM DW 02H DUP (?)

.CODE

MOV AX, @DATA

MOV DS, AX

MOV DX, 00H

MOV AX, NUM

MOV BX, NUM+2

UP:PUSH AX

PUSH DX

DIV BX

CMP DX, 00H

JE EXIT

POP DX

POP AX

ADD AX, NUM

JNC DOWN

INC DX

DOWN:JMP UP

EXIT:POP DX

POP AX

MOV LCM+2, AX

MOV LCM, DX

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: Move data 00 into register Dx and contents of x into register Ax

Step4: Move data in x+2 into register Bx

Step5: Push Ax and push Dx

Step6: Divide data in Bx with data in Ax

Step7: Compare data in Dx with data 0

Step8: Jump if equal to exit. if not equal pop dx and pop Ax

Step9: Add data in Ax with data in x and jump it no carry to down

Step10: Increment Dx

Step11: Jump up

Step12: POP LCM+2 and POP LCM i.e Retrieve back the result to LCM & LCM+2

Step13: Terminate the Program

1. **Write an alp to demonstrate ASCII**

**(i) ASCII addition**

.MODEL SMALL

.DATA

X DB 9H

Y DB 9H

Z DB 02 DUP (?)

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AH, 00H

MOV AL, X

ADD AL, Y

AAA

ADD AX, 3030H

MOV Z, AL

MOV Z+1, AH

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move data 00H into register AH

Step4: move data from X to register AL

Step5: Add data in AL with data in location y

Step6: ASCII adjust after addition the result

Step7: Add contents in AX with data 3030H

Step8: move contents in AL and AH to locations and Z+1 respectively

Step9: Terminate the Program

**(ii) ASCII subtraction**

.MODEL SMALL

.DATA

X DB '9'

Y DB '9'

Z DB 02 DUP (?)

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AH, 00H

MOV AL, X

SUB AL, Y

AAS

ADD AX, 3030H

MOV Z, AL

MOV Z+1, AH

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move data 00H into register AH

Step4: move data from X to register AL

Step5: Subtraction data in AL with contents in y

Step6: ASCII adjust after Subtraction

Step7: Add contents in AX with data 3030H

Step8: move data in AL and AH to locations Z and Z+1 respectively

Step9: Terminate the program

**27. Write an alp to demonstrate ASCII multiplication**

.MODEL SMALL

.DATA

X DB 8H

Y DB 8H

Z DB 02 DUP (?)

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AH, 00H

MOV AL, X

MUL Y

AAM

ADD AX, 3030H

MOV Z, AL

MOV Z+1, AH

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move data 00H into register AX

Step4: move data from X to register AL

Step5: Multiply the contents in AX with y

Step6: ASCII adjust after Multiplication

Step7: Add data 3030 with contents in AX

Step8: move contents in AL and AH to locations and Z+1 respectively

Step10: Terminate the Program

**28. Write an alp to demonstrate ASCII division**

.MODEL SMALL

.DATA

X DW 0604H

Y DB 08H

Z DB 02 DUP (?)

.CODE

MOV AX, @DATA

MOV DS, AX

MOV AH, 00H

MOV BL, Y

MOV AX, X

AAD

DIV BL

ADD AX, 3030H

MOV Z, AL

MOV Z+1, AH

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: move contents from X to register AX

Step4: ASCII adjust before division

Step5: Divide the contents of AX with contents in y

Step6: Add data 3030 with register AX

Step7: move data in AL and AL to locations and Z+1 respectively

Step8: Terminate the Program

1. **Write an alp for string transfer.**

.MODEL SMALL

.DATA

X DB "CAMBRIDGE"

LEN DW $-X

Y DB(LEN-X) DUP(?)

.CODE

MOV CX, @DATA

MOV DS, CX

MOV ES, CX

CLD

MOV CX, LEN

LEA SI, X

LEA DI, Y

L1: MOVSB

LOOP L1

MOV AH, 4CH

INT 21H

END

Step1: Start

Step2: initialize the data segment

Step3: loads effective addresses of x and y in SI and DI respectively

Step4: move data in len into register cx

Step5: Move the complete byte and loop L1

Step6: Terminate the Program

1. **Write an alp to reverse the given string.**

.MODEL SMALL

.DATA

X DB "POOL"

LEN DW $-X

Y DB(LEN-X) DUP(?)

.CODE

MOV CX, @DATA

MOV DS, CX

MOV ES, CX

CLD

MOV CX, LEN

LEA SI, X

LEA DI, Y

ADD DI, CX

DEC DI

L1:MOVSB

SUB DI,2

LOOP L1

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: initialize the data segment

Step3: loads effective addresses of x and y in SI and DI respectively

Step4: move data in len into register cx

Step5: Move the complete byte and loop L1

Step6: Terminate the Program

1. **Write an alp to check whether the given string is palindrome or not.**

.MODEL SMALL

.DATA

X DB "ROAR"

LEN DW $-X

Y DB(LEN-X) DUP(?)

MSG1 DB 13,10, "PALINDROME$"

MSG2 DB 13,10, "NOT PALINDROME$"

.CODE

MOV CX, @DATA

MOV DS, CX

MOV ES, CX

CLD

LEA SI, X

LEA DI, Y

MOV CX, LEN

ADD DI, CX

DEC DI

L1:MOVSB

SUB DI,2

LOOP L1

LEA SI,X

LEA DI, Y

MOV CX, LEN

L2: CMPSB

JNE FAILS

LOOP L2

LEA DX, MSG1

JMP L3

FAILS: LEA DX, MSG2

L3: MOV AH, 9

INT 21H

MOV AH, 4CH

INT 21H

END

**Algorithm:**

Step1: Start

Step2: Initialize the data segment

Step3: Clear directional flag

Step4: load effective addresses of x and y in SI and DI respectively

Step5: Move data in len into register CX

Step6: Add address of CX with address of SI

Step7: Decrement SI

Step8: Move the complete byte (byte operation)

Step9: Decrement DI twice and loop L1

Step10: load effective addresses of x and y in SI and DI respectively

Step11: move data in len into register Cx

Step12: compares the complete byte

Step13: If they are not equal then display the message not a palindrome

Step14: If they are equal then display the message palindrome

Step15: Terminate the Program

**INTERFACING HARDWARE PROGRAMS**

1. **Write an alp to read number from port B through logic controller interface**

**board. And verify the number is 2 out of 5 code or not. And then display the result on port A.**

.MODEL SMALL

.DATA

PA EQU 0CE00H

PB EQU 0CE01H

CR EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV DX,CR

MOV AL,82H

OUT DX,AL

MOV DX,PB

IN AL,DX

TEST AL,0E0H

JNZ DISPLAY

MOV DX,00H

MOV CX,05H

UP:ROR AL,01H

JNC DOWN

INC DX

DOWN:LOOP UP

CMP DX,02H

JNE DISPLAY

MOV AL,0FFH

MOV DX,PA

OUT DX,AL

JMP EXIT

DISPLAY:MOV AL,00

MOV DX, PA

OUT DX,AL

EXIT:MOV AH,4CH

INT 21H

END

**2) Write an alp to perform BCD up and down counter using logic controller**

**interface board.**

.MODEL SMALL

.DATA

PA EQU 0CE00H

CR EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV DX,CR

MOV AL,80H

OUT DX,AL

MOV AL,00H

MOV DX,PA

LOOP1:OUT DX,AL

CMP AL,09H

JGE NEXT1

CALL DELAY

ADD AL,01H

JMP LOOP1

NEXT1:OUT DX,AL

CALL DELAY

SUB AL,01H

JNZ NEXT1

OUT DX,AL

MOV AH,4CH

INT 21H

DELAY PROC

MOV BX,2FFFH

B2:MOV CX,0FFFFH

B1:LOOP B1

DEC BX

JNZ B2

RET

DELAY ENDP

END

**3) Write an alp to perform binary up and down counter using logic controller**

**interface board.**

.MODEL SMALL

.DATA

PA EQU 0CE00H

CR EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV DX,CR

MOV AL,80H

OUT DX,AL

MOV AL,00H

MOV DX,PA

LOOP1:OUT DX,AL

CMP AL,0fH

JGE NEXT1

CALL DELAY

ADD AL,01H

JMP LOOP1

NEXT1:OUT DX,AL

CALL DELAY

SUB AL,01H

JNZ NEXT1

OUT DX,AL

MOV AH,4CH

INT 21H

DELAY PROC

MOV BX,2FFFH

B2:MOV CX,0FFFFH

B1:LOOP B1

DEC BX

JNZ B2

RET

DELAY ENDP

END

**4) Write an alp to read number from port B through logic controller interface**

**board. And**

**verify the number is even or odd. And then display the result on port A.**

.MODEL SMALL

.STACK 30

.DATA

PORTA EQU 0CE00H

PORTB EQU 0CE01H

PORTC EQU 0CE02H

CW EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV DX,CW

MOV AL,82H

OUT DX,AL

MOV DX,PORTB

IN AL, DX

BACK: ROR AL,01H

JC ODD

MOV DX, PORTA

MOV AL,00H

OUT DX, AL

JMP EXIT

ODD: MOV DX, PORTA

MOV AL,0FFH

OUT DX, AL

EXIT: MOV AH,4CH

INT 21H

END

**5) Write an alp to read number from port B through logic controller interface**

**board. And verify the number is even parity or odd parity. And then display**

**the result on port** **A**.

.MODEL SMALL

.STACK 30

.DATA

PORTA EQU 0CE00H

PORTB EQU 0CE01H

PORTC EQU 0CE02H

CW EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV DX,CW

MOV AL,82H

OUT DX,AL

MOV DX,PORTB

IN AL, DX

MOV BL,00H

MOV CX, 08H

BACK: ROL AL,01H

JNC L1

INC BL

L1: LOOP BACK

TEST BL,01H

JNZ ODD

MOV DX, PORTA

MOV AL,00H

OUT DX, AL

JMP EXIT

ODD: MOV DX, PORTA

MOV AL,0FFH

OUT DX, AL

EXIT: MOV AH, 4CH

INT 21H

END

**6) Write an alp to read number from port B through logic controller interface**

**board. And verify the number is Positive or negative. And then display the**

**result on port A.**

.MODEL SMALL

.STACK 30

.DATA

PORTA EQU 0CE00H

PORTB EQU 0CE01H

PORTC EQU 0CE02H

CW EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV DX,CW

MOV AL,82H

OUT DX,AL

MOV DX,PORTB

IN AL, DX

BACK: ROL AL,01H

JC NEGA

MOV DX, PORTA

MOV AL,00H

OUT DX, AL

JMP EXIT

NEGA: MOV DX, PORTA

MOV AL,0FFH

OUT DX, AL

EXIT: MOV AH,4CH

INT 21H

END

**7) Write an alp to perform ring counter using logic controller interface board.**

.MODEL SMALL

.STACK 30

.DATA

PORTA EQU 0CE00H

PORTB EQU 0CE01H

PORTC EQU 0CE02H

CW EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV DX,CW

MOV AL,80H

OUT DX,AL

MOV AL,00H

MOV DX,PORTA

MOV CL,08H

MOV AL,01H

BACK: ROR AL,01H

OUT DX,AL

CALL DELAY

LOOP BACK

MOV AH,4CH

INT 21H

DELAY PROC NEAR

PUSH CX

PUSH BX

MOV CX,06FFFH

L2: MOV BX,0FFFFH

L1:DEC BX

JNZ L1

LOOP L2

POP BX

POP CX

RET

DELAY ENDP

END

**8) Write an alp to display the day & month of any day of the year using**

**keyboard/display interfacing board. (Example: 06/04/2016 is displayed as**

**0604)**

.MODEL SMALL

.STACK 20

.DATA

NUM DB 0,6,0,4

DISPTBL DB 3,9FH,25H,0DH,99H,49H,41H,1FH,1,9

PA EQU 0CE00H

PB EQU 0CE01H

PC EQU 0CE02H

CTRL EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AL,90H

MOV DX,CTRL

OUT DX,AL

CALL BCD4DISP

MOV AH,4CH

INT 21H

BCD4DISP PROC

MOV SI,3

NXTCHAR: MOV AH,8

MOV AL,NUM[SI]

LEA BX,DISPTBL

XLAT

NXTSEG:MOV DX,PB

OUT DX,AL

MOV CH,AL

MOV AL,0

MOV DX,PC

OUT DX,AL

MOV AL,0F0H

OUT DX,AL

DEC AH

JZ OVRCHK

MOV AL,CH

ROR AL,1

JMP NXTSEG

OVRCHK: DEC SI

CMP SI,-1

JNE NXTCHAR

RET

BCD4DISP ENDP

END

**9) Write an alp for keyboard interfacing to display the key pressed along with its**

**row and column number.**

.MODEL SMALL

.STACK 20

.DATA

PROMPT DB 'PRESS ANY KEY ONT THE COMPUTER',13,10,"$"

MSG DB 'THE SCAN CODE OF THE KEY PRESSED OF KBD I/P IS:'

ASKIRES DB ?,?,13,10,'$'

ASKICOD DB '0123456789ABCDEF'

T1 DB ?

T2 DB ?

PA EQU 0CE00H

PB EQU 0CE01H

PC EQU 0CE02H

CTRL EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AL,90H

MOV DX,CTRL

OUT DX,AL

MOV AH,9

LEA DX,PROMPT

INT 21H

MOV T1,20H

AGAIN:MOV AH,6

MOV DL,0FFH

INT 21H

JNZ QUIT

CALL SCAN

CMP SI,0

JE AGAIN

MOV CL,3

ROL BH,CL

ADD BH,AH

MOV AL,BH

MOV AH,0

MOV CL,10H

DIV CL

LEA BX,ASKICOD

XLAT

MOV ASKIRES,AL

MOV AL,AH

XLAT

MOV ASKIRES+1,AL

MOV AH,9

LEA DX,MSG

INT 21H

PUSH AX

MOV AX,0FFFH

HERE1:MOV CX,0FFFFH

HERE:LOOP HERE

DEC AX

JNZ HERE1

POP AX

DN1:JMP AGAIN

QUIT:MOV AH,4CH

INT 21H

SCAN PROC

MOV SI,0

MOV CX,3

MOV BH,0

MOV AL,80H

NXTROW:ROL AL,1

MOV BL,AL

MOV DX,PC

OUT DX,AL

MOV DX,PA

IN AL,DX

CMP AL,0

JNZ KEYID

MOV AL,BL

INC BH

LOOP NXTROW

RET

KEYID:MOV SI,1

MOV CX,8

MOV AH,0

AGN2:ROR AL,1

JC SKIP

INC AH

LOOP AGN2

SKIP:RET

SCAN ENDP

END

**10) Write an alp to flash/scroll a message FIRE& HELP using keyboard/display**

**interfacing board.**

.MODEL SMALL

.STACK 20

.DATA

M DB "EXIT$"

M1 DB 86H,8FH,0F9H,8EH

M2 DB 8CH,0C7H,86H,89H

PA EQU 0CE00H

PB EQU 0CE01H

PC EQU 0CE02H

CTRL EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AL,80H

MOV DX,CTRL

OUT DX,AL

LEA DX,M

MOV AH,09H

INT 21H

START:LEA SI,M1

CALL DISP

CALL DELAY

LEA SI,M2

CALL DISP

CALL DELAY

MOV AH,01H

INT 16H

JZ START

MOV AH,4CH

INT 21H

DISP PROC

MOV CX,04H

NC:MOV BL,08H

MOV AL,[SI]

NB:ROL AL,01H

MOV DX,PB

OUT DX,AL

PUSH AX

MOV AL,00H

MOV DX,PC

OUT DX,AL

MOV AL,0FFH

OUT DX,AL

POP AX

DEC BL

JNZ NB

INC SI

LOOP NC

RET

DISP ENDP

DELAY PROC

PUSH CX

PUSH BX

MOV BX,2FFFH

L2:MOV CX,0FFFFH

L1:LOOP L1

DEC BX

JNZ L2

POP BX

POP CX

RET

DELAY ENDP

END

**11) Write an alp to drive a stepper motor in clockwise direction using Stepper**

**motor interface board.**

.MODEL SMALL

.DATA

PORTC EQU 0CE02H

CR EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV DX,CR

MOV AL,80H

OUT DX,AL

MOV AL,99H

RPT1: MOV DX,PORTC

OUT DX,AL

CALL DELAY

PUSH AX

MOV AH,0BH

INT 21H

OR AL,AL

JNZ QUIT

POP AX

ROR AL,01H

JMP RPT1

QUIT:MOV AH,4CH

INT 21H

DELAY PROC

MOV DX,0FFH

D2: MOV CX,0FFFFH

D1: LOOP D1

DEC DX

JNZ D2

RET

DELAY ENDP

END

**12) Write an alp to drive a stepper motor in anticlockwise direction using Stepper**

**motor interface board.**

.MODEL SMALL

.DATA

PORTC EQU 0CE02H

CR EQU 0CE03H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV DX,CR

MOV AL,80H

OUT DX,AL

MOV AL,99H

RPT1: MOV DX,PORTC

OUT DX,AL

CALL DELAY

PUSH AX

MOV AH,0BH

INT 21H

OR AL,AL

JNZ QUIT

POP AX

ROL AL, 01H

JMP RPT1

QUIT: MOV AH,4CH

INT 21H

DELAY PROC

MOV DX, 0FFH

D2: MOV CX, 0FFFFH

D1: LOOP D1

DEC DX

JNZ D2

RET

DELAY ENDP END

; PROGRAM FOR LOGIC CONTROL INTERFACE

;EQUATES

PORT\_A: EQU 0FFC0H

PORT\_B: EQU 0FFC2H

PORT\_C: EQU 0FFC4H

CTL\_PORT: EQU 0FFC6H

CSEG SEGMENT

ASSUME CS:CSEG DS:CSEG

ORG 0:5000H

START1:

MOV AL,8AH

MOV DX,CTL\_PORT

OUT DX,AL ;INITIALIZE 8255,PORTS A & C LOWER

MOV AL,00H ;AS OUT AND PORT B & C HIGHER AS INPUT

MOV DX,PORT\_A ; BLANK OUT PORT A

OUT DX,AL ; AND PORTC

MOV DX,PORT\_C ; AND PORTC

OUT DX,AL

READ\_PORTB:

MOV DX,PORT\_B

IN AL,DX

MOV DX,PORT\_A

OUT DX,AL

READ\_PORTC:

MOV DX,PORT\_C

IN AL,DX

ROR AL,1

ROR AL,1

ROR AL,1

ROR AL,1

AND AL,0FH

MOV DX,PORT\_C

OUT DX,AL

JMP READ\_PORTB

INT 3

CSEG ENDS

END

; PROGRAM TO STUDY ADC-0809 IN INTERRUPT MODE

CONTROL EQU FFC6H ;control port address for 8255

PORTA EQU FFC0H ;porta address for 8255

PORTB EQU FFC2H ;portb address for 8255

PORTC EQU FFC4H ;portc address for 8255

DBDTA EQU F800:4F1FH

CMD59 EQU FFD8H

DATA59 EQU FFDAH

DSEG SEGMENT

ORG 0:4000H

IBYTE DB 0

DSEG ENDS

CSEG SEGMENT

ORG 0000:5000H

ASSUME CS:CSEG,DS:DSEG

START:

MOV AX,00H ;initialisation of stack pointer

MOV SS,AX

CLI

CLD

MOV BX,202H ;initalisation of interrupt vector

PUSH CS

POP AX

MOV [BX],AX

MOV BX,200H

LEA AX,CS:ISS

MOV [BX],AX

MOV DX,CMD59 ;ICW1

MOV AL,13H

OUT DX,AL

MOV DX,DATA59 ;ICW2(interrupt vector address)

MOV AL,80H

OUT DX,AL

MOV AL,0FH

OUT DX,AL ;ICW4

MOV AL,0FEH

OUT DX,AL ;OCW1(IR0 mask reset)

MOV AL,IBYTE

MOV CL,AL

MOV BL,00H

MOV DX,CONTROL

MOV AL,90H ;control word for PPI,portA is

OUT DX,AL ;set as input port & portB,portC

;are set as output ports.

STI

AD00:

MOV AL,IBYTE ;output channel number

MOV DX,PORTC

OUT DX,AL

;start conversion

MOV DX,CONTROL

MOV AL,0FH ;PC7 (START/ALE) set

OUT DX,AL

PUSH CX

MOV CX,FFH

WAT:

LOOP WAT

POP CX

MOV DX,CONTROL

MOV AL,0EH ;PC7 reset

OUT DX,AL

;wait for interrupt to come

AD05: ;look for flag, flag will

CMP BL,00H ;set in ISR,if it is set again

JZ AD05 ;read the channel and display

MOV BL,00H ;digital value

JMP AD00

;INTERRUPT SERVICE ROUTINE

;--------------------------

;EOC line is connected to IR0 line of PIC

;through D F/F,hence EOC interrupts CPU.

;In ISR, read the specified channel and display

;the read value in data field.

ISS:

MOV DX,CONTROL

MOV AL,0DH ;PC6 (OE) set

OUT DX,AL

;before reading data from ADC set PC6 line

MOV DX,PORTA

IN AL,DX ;read digital value

MOV AH,00H

MOV SI,AX

PUSH CX

CALL FAR DBDTA ;display digital value

POP CX

MOV AL,IBYTE

MOV AH,0

MOV SI,AX

; CALL FAR DBDTA ;display digital value

MOV BL,FFH

STI

IRET

CSEG ENDS

END

;PROGRAM TO STUDY ADC-0809 IN POLLED MODE

; SELECTING CHANNEL NO. 0

CONTROL EQU FFC6H ;control port address for 8255

PORTA EQU FFC0H ;porta address for 8255

PORTB EQU FFC2H ;portb address for 8255

PORTC EQU FFC4H ;portc address for 8255

KBDT EQU F800:4EECH ;4EEDH

DBDTA EQU F800:4F1FH

DSPLY EQU F800:4FC0H

CLRDSP EQU F800:4BB1H

DSEG SEGMENT

ORG 0000:4000H

CH\_NO DB 0

DSEG ENDS

CSEG SEGMENT

ASSUME CS:CSEG,DS:DSEG

ORG 0000:5000H

MOV AX,0000H

MOV DS,AX

;AD00:

MOV AL,90H ;control word for PPI

MOV DX,CONTROL

OUT DX,AL ;portA->i/p port,portB->o/p port

;portC->o/p port.

AD00:

MOV AL,CH\_NO ;output channel number

MOV DX,PORTC

OUT DX,AL

;start conversion

MOV AL,0FH ;PC7 (START/ALE) set

MOV DX,CONTROL

OUT DX,AL

PUSH CX

MOV CX,3FFFH

DEL1:

LOOP DEL1

POP CX

MOV AL,0EH ;PC7 reset

MOV DX,CONTROL

OUT DX,AL

;look for EOC

MOV AL,0CH ;reset PC6 to read EOC

OUT DX,AL

AD01:

MOV DX,PORTA

IN AL,DX ;poll the EOC line which

AND AL,80H ;is connected to PA7 line

CMP AL,80H

JNZ AD01

;if EOC (PA7) is high read the digital value otherwise

;again check for EOC (PA7) line

MOV AL,0DH ;set OE (PC6) to read value

MOV DX,CONTROL

OUT DX,AL

;before reading data from ADC set PC6 line

MOV DX,PORTA

IN AL,DX ;read digital value

MOV AH,00H

MOV SI,AX

PUSH CX

CALL FAR DBDTA ;display digital value

POP CX

JMP AD00

CSEG ENDS

END

;PROGRAM FOR LDR-BUZZER-RELAY OPERATION

; PC1 CONNECTED TO LDR I/P

; PC4 CONNECTED TO BUZZER-RELAY

.OUTPUT 2500AD

CSEG SEGMENT

ORG 0:6000H

PORTA EQU 0FFC0H

PORTB EQU 0FFC2H

PORTC EQU 0FFC4H

CNTRL EQU 0FFC6H

ST:

;CONFIGURE PORTCL AS I/P & OTHER PORTS AS O/P

MOV DX,CNTRL

MOV AL,81H

OUT DX,AL

MOV DX,CNTRL

MOV AL,08H; BZR RLY OFF

OUT DX,AL

CHKLDR:

MOV DX,PORTC

IN AL,DX

AND AL,02H

JNZ BZRLYON

MOV DX,CNTRL

MOV AL,08H ; BZR RLY OFF

OUT DX,AL

JMP CHKLDR

BZRLYON:

MOV DX,CNTRL

MOV AL,09H ; BZR RLY ON

OUT DX,AL

;DELAY

MOV CX,FFFFH

DLY: LOOP DLY

JMP CHKLDR

CSEG ENDS

END