2.Write an Assembly Language Program to add an array of five,16 bit numbers

**.MODEL SMALL**

**.DATA**

**ARRAY DW 5555H,6666H,7777H,8888H,9999H**

**SUM DW 0**

**CARRY DB 0**

**.CODE**

**MOV AX,@DATA** ;Initialize DS register

**MOV DS,AX**

**MOV SI, OFFSET ARRAY** ;Place offset of ARRAY in SI register

**MOV CX ,0005H** ;Initialize count for no. of elements

**UP : MOV AX,[SI] ;**Load no. from memory to AX register

**ADD SUM,AX ;**add no. with sum

**JNC NEXT** ;Jump if Sum > 16 bit

**INC CARRY** ;store carry

**NEXT: INC SI** ;Increment memory pointer

**INC SI**

**DEC CX** ;Decrement count

**JNZ UP** ;Jump if count is not zero

**MOV AH, 4CH** ;Terminate program

**INT 21H**

**END**

3.Write an Assembly Language Program to implement multiplication and division

1] WAP for 8 bit by 8 bit unsigned multiplication

2] WAP for 16 bit by 8 bit signed multiplication

3] WAP for 16 bit by 8 bit unsigned division

4] WAP for 16 bit by 16 bit signed division

1] WAP for 8 bit by 8 bit unsigned multiplication

**.MODEL SMALL**

**.DATA**

**MULTIPLICAND DB 18H**

**MULTIPLIER DB 12H**

**PRODUCT DW ?**

**.CODE**

**MOV AX,@DATA**

**MOV DS, AX**

**MOV AL, MULTIPLICAND**

**MOV BL, MULTIPLIER**

**MUL BL**

**MOV PRODUCT,AX**

**MOV AH,4CH**

**INT 21H**

**END**

2] WAP for 16 bit by 8 bit signed multiplication

**.MODEL SMALL**

**.DATA**

**MULTIPLICAND DB -18H**

**MULTIPLIER DW 1312H**

**PRODL DW ?**

**PRODH DW ?**

**.CODE**

**MOV AX,@DATA**

**MOV DS, AX**

**MOV AL, MULTIPLICAND**

**CBW**

**MOV BX, MULTIPLIER**

**MUL BX**

**MOV PRODL,AX**

**MOV PRODH,DX**

**MOV AH,4CH**

**INT 21H**

**END**

3] WAP for 16 bit by 8 bit unsigned division

**.MODEL SMALL**

**.DATA**

**DIVIDEND DW 4567H**

**DIVISOR DB 88H**

**QUOTIENT DB ?**

**REMENDER DB ?**

**.CODE**

**MOV AX,@DATA**

**MOV DS,AX**

**MOV AX,DIVIDEND**

**MOV BL,DIVISOR**

**DIV BL**

**MOV QUOTIENT,AL**

**MOV REMENDER,AH**

**MOV AH,4CH**

**INT 21H**

**END**

4] WAP for 16 bit by 16 bit signed division

**.MODEL SMALL**

**.DATA**

**DIVIDEND DW FDCAH**

**DIVISOR DW -1234H**

**QUOTIENT DW ?**

**REMENDER DW ?**

**.CODE**

**MOV AX,@DATA**

**MOV DS,AX**

**MOV AX,DIVIDEND**

**CWD**

**MOV BX,DIVISOR**

**DIV BX**

**MOV QUOTIENT,AX**

**MOV REMENDER,DX**

**MOV AH,4CH**

**INT 21H**

**END**

**4.** Write an Assembly Language Program to find Smallest/Largest number from an array of N numbers

**.MODEL SMALL**

**.DATA**

**ARRAY DB 98H,23H,34H,0AH,10H**

**SMALLEST DB ?**

**.CODE**

**MOV AX,@DATA**

**MOV DS,AX**

**MOV CX,04H**

**MOV SI,OFFSET ARRAY**

**MOV AL,[SI]**

**UP: INC SI**

**CMP AL,[SI]**

**JNC NEXT**

**MOV AL,[SI]**

**NEXT: DEC CX**

**JNZ UP**

**MOV SMALLEST,AL**

**MOV AH,4CH**

**INT 21H**

**END**

**5.** Write an Assembly language program to exchange block of data bytes using string instructions

**.MODEL SMALL**

**.DATA**

**BLK1 DB 11H,22H,33H,44H,55H**

**BLK2 DB 0AAH,0BBH,0CCH,0DDH,EEH**

**.CODE**

**MOV AX,@DATA**

**MOV DS,AX**

**CLD**

**LEA SI,BLK1**

**LEA DI, BLK2**

**MOV CX,05H**

**UP : MOV BL,[DI]**

**MOVSB**

**MOV [SI-1],BL**

**LOOP UP**

**MOV AH,4CH**

**INT 21H**

**END**

**6.** Write an Assembly Language Program to arrange numbers in the given array in ascending/descending order

**.MODEL SMALL**

**.DATA**

**ARRAY DB 33H,88H,99H,22H,55H**

**.CODE**

**MOV AX,@DATA**

**MOV DS,AX**

**MOV BL,05H**

**TOP:MOV SI,OFFSET ARRAY**

**MOV CL,04H**

**UP:MOV AL,[SI]**

**INC SI**

**CMP AL,[SI]**

**JC DOWN**

**XCHG AL,[SI]**

**XCHG AL,[SI-1]**

**DOWN: LOOP UP**

**DEC BL**

**JNZ TOP**

**MOV AH,4CH**

**INT 21H**

**END**

**7.** Write an Assembly Language Program to check whether string is palindrome or not. Display the appropriate message.

**.MODEL SMALL**

**.DATA**

**MS1 DB 10,13,’ENTER THE STRING:$’**

**MS2 DB 10,13,’STRING IS PALINDROME:$’**

**MS3 DB 10,13,’STRING IS NOT PALINDROME:$**

**BUFF DB 80**

**DB 0**

**DB 80 DUP(0)**

**.CODE**

**MOV AX,@DATA**

**MOV DS, AX**

**MOV AH, 09H** ;display MS1

**LEA DX, MS1**

**INT 21H**

**MOV AH, 0AH** ;accept string from keyboard

**LEA DX, BUFF**

**INT 21H**

**LEA BX, BUFF+2** ;offset of string in BX reg.

**MOV CH, 00H**

**MOV CL, BUFF+1** ;length of string in CL reg.

**MOV DI, CX** ;DI points last character

**DEC DI**

**SAR CL, 1** ;divide count by 2

**MOV SI, 0000H**

**BACK: MOV AL, [BX+SI]**

**MOV AH, [BX+DI]**

**CMP AL, AH**

**JNZ LAST**

**INC SI**

**DEC DI**

**DEC CL**

**JNZ BACK**

**MOV AH, 09H** ;display MS2

**LEA DX, MS2**

**INT 21H**

**JMP TER**

**LAST: MOV AH, 09H** ;display MS3

**LEA DX, MS3**

**INT 21H**

**TER:END**

**8.** Write an Assembly Language Program to compare two strings.(Using MACRO)

Software required: MASM (Macro Assembler from Microsoft Corp.)

PRINT MACRO MES ; Macro to display string

MOV AH,09H

LEA DX, MES

INT 21H

ENDM

.MODEL SMALL

.DATA

MS1 DB 10,13,”ENTER FIRST STRING:$”

MS2 DB 10,13,”ENTER SECOND STRING:$”

MS3 DB 10,13,”EQUAL STRINGS$”

MS4 DB 10,13,”UNEQUAL STRINGS$”

BUFF1 DB 10 DUP(‘$’)

BUFF2 DB 10 DUP(‘$’)

.CODE

MOV AX,@DATA

MOV DS,AX

MOV ES,AX

PRINT MS1

MOV AH, 0AH

LEA DX, BUFF1

INT 21H

PRINT MS2

MOV AH, 0AH

LEA DX, BUFF2

INT 21H

MOV CL, LENGTH BUFF1

MOV CH, LENGTH BUFF2

CMP CL,CH

JNE UNEQ

MOV CH,00H

LEA SI, BUFF1

LEA DI, BUFF2

CLD

REPE CMPSB

JNZ UNEQ

PRINT MS3

JMP FINISH

UNEQ: PRINT MS4

FINISH: MOV AH, 4CH

INT 21H

END

**9.** Write an Assembly Language Program to find the factorial of a number

.MODEL SMALL

.DATA

NUM DW 0005H

FACTLSW DW ?

FACTMSW DW ?

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AX,01H

MOV BX,NUM

CALL FACT

MOV FACTLSW,AX

MOV FACTMSW,DX

MOV AH,4CH

INT 21H

FACT PROC NEAR

CMP BX,01H

JZ LAST

UP: MUL BX

DEC BX

CMP BX,01H

JNZ UP

RET

LAST:MOV AX,01H

RET

FACT ENDP

END

**10.** Write an Assembly Language Program to find the GCD of two 16 bit unsigned numbers

.MODEL SMALL

.DATA

NO1 DW 0120

NO2 DW 0090

GCD DW 0H

.CODE

MOV AX,@DATA

MOV DS, AX

MOV AX,NO1

MOV BX,NO2

AGAIN: CMP AX,BX

JE FINISH

JB EXCHG

UP: MOV DX,0

DIV BX

CMP DX,0

JE FINISH

MOV AX,DX

JMP AGAIN

EXCHG: XCHG AX,BX

JMP UP

FINISH: MOV GCD,BX

mov ch,04h ;count of digits to be displayed

mov cl,04h ;count to roll by 4 bits

L1: rol bx,cl ;roll bl so that msb comes to lsb

mov dl,bl ;load dl with data to be displayed

and dl,0fh ;get only lsb

cmp dl,09 ;check if digit is 0-9 or letter A-F

jbe L2

add dl,07 ;if letter add 37H else add 30H

L2: add dl,30h

mov ah,02h ;display character

int 21h

dec ch ;decrement count

jnz L1

mov ah,4ch

int 21h

END

**11.** Write an assembly language program to display the contents of 16 bit flag register.

.MODEL SMALL

.DATA

MSG DB 0DH,0AH,”-- -- -- -- OF DF IF TF SF -- ZF --AF -- PF -- CF $”

NEWL DB 0DH,0AH,” “

FLAG DW ?

.CODE

MOV AX,@DATA

MOV DS,AX

MOV DX,OFFSET MSG

MOV AH,09H

INT 21H

MOV DX,OFFSET NEWL

MOV AH,09H

INT 21H

CLI

STC

STD

PUSHF

POP BX

MOV FLAG,BX

MOV CX,16

MOV BX,8000H

LOOPS:MOV AX,FLAG

AND AX,BX

JZ ZERO

MOV DL,31H

MOV AH,02H

INT 21H

JMP SPACE

ZERO:MOV DL,30H

MOV AH,02H

INT 21H

SPACE:MOV DL,” “

MOV AH,02H

INT 21H

MOV AH,02H

INT 21H

ROR BX,1

LOOP LOOPS

MOV AH,4CH

INT 21H

END

**12** To Study and Interface Traffic Light Controller Using PPI 8255

APPARATUS: 1. 8086 Trainer kit 2. Key board 3. SMPS

**Address Opcode Mneumonics Comments**

**0400 B0 80 MOV AL,80H ;INIT 8255 CWR**

**0402 E6 76 OUT 76H,AL ;SET ALL PORTS(OUTPUT)**

**0404 B0 11 MOV AL,11H ;SET ALL SQUARE RED**

**0406 E6 70 OUT 70H,AL ;OUT AT PORT- A**

**0408 E6 74 OUT 74H,AL ;OUT AT PORT- C**

**040A E8 42 00 CALL DELAY 1 ;CALL DELAY 10msec**

**040D B0 44 UP:MOV AL,44H ;SET GREEN LED OF N& S**

**040F E6 70 OUT 70H,AL ;SET RED LED OF E,W**

**0411 E8,3B,00 CALL DELAY1 ;CALL DELAY 10 MSEC**

**0414 B0 22 MOV AL,22H ;SET YELLOW OF N&S**

**0416 E6 70 0UT 70H,AL ;OUT AT PORT -A**

**0418 E8 41 00 CALL DELAY 2 ;CALL DELAY 5 MSEC**

**041B B0 99 MOV AL,99H ;SET ALL SQUARE RED**

**041D E6 70 OUT 70H,AL ;SET GREEN(GO LEFT LED)**

**041F E8 2D 00 CALL DELAY 1 ;CALL DELAY 5 MSEC**

**0422 B0 22 MOV AL,22H ;SET YELLOW LED N&S**

**0424 E6 70 OUT 70H,AL ;OUT AT PORT-A**

**0426 E8 33 00 CALL DELAY 2 ;CALL DELAY 5 MSEC**

**0429 B0 11 MOV AL,11H ;SET ALL SQUARE RED**

**042B E6 70 OUT 70H,AL ;OUT AT PORT-A**

**042D B0 44 MOV AL,44H ;SET GREEN OF E&W**

**042F E6 74 OUT 74H,AL ;OUT AT PORT-C**

**0431 E8 1B 00 CALL DELAY 1 ;CALL DELAY 10 MSEC**

**0434 B0 22 MOV AL,22H ;SET YELLOW LED E&W**

**0436 E6 74 OUT 74H,AL ;OUT AT PORT-C**

**0438 E8 21 00 CALL DELAY 2 ;CALL DELAY 5MSEC**

**043B B0 99 MOV AL,99H ;SET ALL SQUARE RED**

**043D E6 74 OUT 74H,AL ;SET GREEN OF E&W**

**043F E8 0D 00 CALL DELAY1 ;CALL DELAY 10 MSEC**

**0442 B0 22 MOV AL,22H ;SET YELLOW LED E&W**

**0444 E6 74 OUT 74H,AL ;OUT AT PORT -C**

**0446 E8 13 00 CALL DELAY 2 ;CALL DELAY 5 MSEC**

**0449 B0 11 MOV AL,11H ;SET ALL SQUARE RED**

**044B E6 74 OUT 74H,AL ;OUT AT PORT-C**

**044D EB BE JMP UP ;JUMP TO START**

**044F BB 0F 00 DELAY1:MOV BX,000FH ;10 MSEC DELAY ROUTINE**

**0452 B9 FF FF DL2:MOV CX,0FFFFH**

**0455 49 DL1:DEC CX**

**0456 75 FD JNZ DL1**

**0458 DB DEC BX**

**0459 75 F7 JNZ DL2**

**045B C3 RET**

**045C BB 05 00 DL2:MOV BX,0005H ;5 MSEC DELAY ROUTINE**

**045F B9 FF FF DL4:MOV CX,0FFFFFH**

**0462 49 DL3:DEC CX**

**0463 75 FD JNZ DL3**

**0465 4B DEC BX**

**0466 75 F7 JNZ DL4**

**0468 C3 RET**

**13** Write mixed language program to separate even and odd numbers from an array .

**Program to separate even and odd numbers from an array :.**

**#include <conio.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**Void main( )**

**{**

**int arr[10],evn[10],odd[10];**

**int no ;**

**char rem**

**int i, j=0, k=0, l1, l2 ;**

**clrscr( )**

**printf (“/n Enter the Array Elements:’);**

**for (i=0; i<10; i++)**

**scanf (“%d”,& arr[i])**

**asm lea si,arr**

**asm mov cx,0ah**

**back : asm mov ax,[si]**

**asm mov no,ax**

**asm mov bl,02h**

**asm div bl**

**asm mov rem,ah**

**if (rem=1)**

**{**

**odd{j]=no;**

**j++ ;**

**l1=j;**

**}**

**else**

**{**

**evn[k]=no;**

**K++;**

**l2=k;**

**}**

**asm add si,2**

**asm loop back**

**printf(“\n Even Array) ;**

**for (i=0; i< l2;i++)**

**printf (“%d”,evn[i]);**

**printf (”\n Odd Array:’);**

**for (i=0; i< l1;i++)**

**printf (“%d”,odd[i]);**

**getch( );**

**}**