**1.1 ADDITION OF TWO 16 BITS NUMBERS SIGNED & UN SIGNED**

ASSUME CS:CODE,DS:DATA

DATA SEGMENT

OPR1 DW 4269H

OPR2 DW 1000H

RES DW ?

DATA END

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

ADD AX,OPR2

MOV RES,AX

MOV AH,4CH (or) MOV AX,004CH

INT 21H

CODE ENDS

END START

END

**RESULT**: -

UNSIGNED:

INPUT: OPR1=4269H, OPR2= 1000H

OUTPUT:- 5269H

SIGNED :-

INPUT:- OPR1=9763H,OPR2= A973H

RES= 40D6H,CF=1

Or

**Mov Ax,7010H**

**Mov DS,AX**

**MOV AX,1000h (or ) MOV AX,[0001H]**

**ADD AX,2000H**

**MOV AX,004CH**

**HLT**

**RESULT**: -

AX=3000h

**1.2. SUBTRACTION OF TWO 16 BITS NO:- SIGNED & UNSIGNED**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 4269H

OPR2 DW 1000H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

SUB AX,OPR2

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

Or

**Mov Ax,7010H**

**Mov DS,AX**

**Mov AX,2200H**

**SUB AX,2000H**

**MOV AX,004CH**

**HLT**

RESULT: -

UNSIGNED:

INPUT: OPR1=4269H, OPR2= 1000H

OUTPUT:- 3269H

SIGNED :-

INPUT:- OPR1=9763H,OPR2= 8973H

RES= 0DF0H,

**1.3. MULTIPLICATION OF TWO 16 BITS UNSIGNED**

ASSUME CS:CODE,DS:DATA

DATA SEGMENT

OPR1 DW 2000H

OPR2 DW 4000H

RESLW DW ?

RESHW DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

MUL OPR2

MOV RESLW,AX

MOV RESHW,DX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

(OR)

**MOV AX,7000**

**MOV BX,2000H**

**MOV DS,AX**

**MOV AX,1000H**

**MUL BX**

**MOV [1000H],AX**

**MOV [1005H],DX**

**MOV AH,4CH**

**INT 21H**

RESULT: -

UNSIGNED:

INPUT: OPR1=2000H, OPR2= 4000H

OUTPUT:- RESLW=0000H(AX)

RESHW=0800H(DX)

**1.4.MULTIPLICATION OF TWO 16 BITS SIGNED NUMBERS**

ASSUME CS:CODE,DS:DATA

DATA SEGMENT

OPR1 DW 7593H

OPR2 DW 6845H

RESLW DW ?

RESHW DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

IMUL OPR2

MOV RESLW,AX

MOV RESHW,DX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

RESULT:

**CASE (1) :----TWO POSITIVE : INPUTS:** OPR1: 7593H

OPR2 : 6845H

**OUTPUT:**

RESLW=689FH

RESHW=2FE3H

CASE(2): ----ONE POSITIVE NUMBER& ONE NEGITIVE NUMBER:

**INPUTS:** OPR1 = 846DH 🡨 2’S COMPLEMENT IS (-7593H)

OPR2 = 6845H

**OUTPUTS:** RESLW= 9761H <- 2’S COMPLEMENT

RESHW= D01CH 🡨 OF (-2FE3689FH)

CASE(3):-----TWO NEGITATIVE NUMBERS

**INPUTS:** OPR1 = 846DH 🡨 2’S COMPLEMENT IS (-7593H)

OPR2 = 97BBH

**OUTPUTS:** RESLW= 689FH <- 2’S COMPLEMENT

RESHW= 2FE3H 🡨 OF (-2FE3689FH)

**1.5. DIVISION OF UN SIGNED NUMBERS**

ASSUME CS: CODE, DS:DATA

DATA SEGMENT

OPR1 DW 2C58H

OPR2 DW 56H

RESQ DW ?

RESR DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

DIV OPR2

MOV RESQ,AX

MOV RESR,DX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

RESULT:

**CASE (1) :--- INPUTS:** OPR1: 2C58H

OPR2 : 56H

**OUTPUT:**

RESQ=H == 0084H

RESR=H==0000H

**1.6. DIVISION OF SIGNED NUMBERS**

ASSUME CS: CODE, DS:DATA

DATA SEGMENT

OPR1 DW 2658H

OPR2 DW 0AAH

RESQ DW ?

RESR DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

IDIV OPR2

MOV RESQ,AX

MOV RESR,DX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

RESULT:

**CASE (1) :--- INPUTS:** OPR1: 2658H

OPR2 : AAH

**OUTPUT:**

RESQ == 0039H

RESR == 007EH

CASE(2):----- ONE POSITVE NUMBER & ONE NEGITIVE NUMBER

INPUT:-- OPR1 = D908H 🡨 2’S COMPLETE OF (-26F8H)

OPR2 = 56H

OUTPUT :---- RESQ= 8CH (AL) 🡨 2’S COMPLETE OF (-74H)

RESR= 00H (AH)

**2.1. ASCII ADDITION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

Char Db 8

Char1 Db 6

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AH,00H

MOV AL,CHAR

ADD AL,CHAR1

AAA

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : CHAR=8**

**CHAR1=6**

**OUTPUT:= RES= 0104(AX) 🡨 UNPACKED BCD OF 14**

**2.2 ASCII SUBTRACTION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

Char Db 9 NO NEED INVERTED COMAS

Char1 Db 5

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AH,00H

MOV AL,CHAR

SUB AL,CHAR1

AAS

MOV RES,AX

MOV AH,4CH

\*INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : CHAR=9**

**CHAR1=5**

**OUTPUT:= RES= 0004(AX)**

**CASE(II):- CHAR=5**

**CHAR1=9**

**RES=00FC(AX) 🡨 2’S COMPLEMENT(-4)**

**2.3. ASCII MULTIPLICATION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

NUM1 Db 09H

NUM2 Db 05H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AH,00H

MOV AL,NUM1

MUL NUM2

AAM

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : NUM1=09**

**NUM2=05**

**OUTPUT:= RES= 0405(AX) 🡨 UN PACKED BCD OF 45**

**2.4. ASCII DIVISION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

DIVIDEND DW 0607H

DIVISIOR DB 09H

RESQ DB ?

RESR DB ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,DIVIDEND

AAD

MOV CH,DIVISIOR

DIV CH

MOV RESQ,AL

MOV RESR,AH

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : DIVIDEND=0607H 🡨**

**UN PACKED BCD OF 67**

**DIVISIOR=09H**

**OUTPUT:= RESQ= 07(AL)**

**RESR=04(AH)**

**3.1. LOGICAL AND OPERATION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 6493H

OPR2 DW 1936H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

AND AX,OPR2

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=6493H**

**OPR2=1936H**

**OUTPUT:= RES= 0012H**

**3.2. LOGICAL OR OPERATION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 6493H

OPR2 DW 1936H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

OR AX,OPR2

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=6493H**

**OPR2=1936H**

**OUTPUT:= RES= 7DB7H**

**3.3. LOGICAL XOR OPERATION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 6493H

OPR2 DW 1936H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

XOR AX,OPR2

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=6493H**

**OPR2=1936H**

**OUTPUT:= RES= 7DA5H**

**3.4. LOGICAL NOT OPERATION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 6493H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

NOT AX

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=6493H**

**OUTPUT:= RES= 9B6CH**

**2’S COMPLEMENT**

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

**DATA SEGMENT**

**NUM1 DW 1234H**

**RESULT DW ?**

**DATA ENDS**

**CODE SEGMENT**

**START:MOV AX,DATA**

**MOV DS,AX**

**MOV AX,NUM1**

**NOT AX**

**ADD AX,01H**

**MOV RESULT,AX**

**INT 21H**

**END START**

**CODE ENDS**

**4.1.SHIFT ARITHEMATIC/LOGICAL LEFT OPERATION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 1639H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

SAL AX,01H--------🡪 (or) 🡨------------ SHL AX,01H

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=1639H**

**OUTPUT:= RES= 2C72H**

**4.2. SHIFT LOGICAL RIGHT OPERATION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 8639H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

SHR AX,01H

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=8639H**

**OUTPUT:= RES= 431CH**

**4.3. SHIFT ARTHEMATIC RIGHT OPERATION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 8639H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

SAR AX,01H

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=8639H**

**OUTPUT:= RES= C31CH**

**4.4. ROTATE RIGHT WITH OUT CARRY**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 1639H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

ROR AX,01H

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=1639H**

**OUTPUT:= RES= 8B1CH**

**4.5. ROTATE RIGHT WITH CARRY**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 1639H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

RCR AX,01H

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=1639H**

**OUTPUT:= RES= 0B1CH**

**4.6. ROTATE LEFT WITH OUT CARRY**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 8097H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

ROL AX,01H

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=8097H**

**OUTPUT:= RES= 012FH**

**4.7. ROTATE LEFT WITH CARRY**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 8097H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

RCL AX,01H

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : OPR1=8097H**

**OUTPUT:= RES= 012EH**

**5.1. MOVE BLOCK**

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

DATA SEGMENT

STR DB 04H,0F9H,0BCH,98H,40H

COUNT EQU 05H

DATA ENDS

EXTRA SEGMENT

ORG 0010H

STR1 DB 05H DUP(?)

EXTRA ENDS

CODE SEGMENT

START:

mov ax,DATA

MOV DS,AX

MOV ES,AX

MOV SI,OFFSET STR

MOV DI,OFFSET STR1

MOV CL,COUNT

CLD

REP MOVSB

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : STR(DS:0000H)=04H,F9H,BCH,98H,40H**

**OUTPUT:= STR1(DS:0010H)= 04H,F9H,BCH,98H,40H**

**5.2. REVERSE STRING**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

STR DB 01H,02H,03H,04H

COUNT EQU 02H

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV Cx,COUNT

MOV SI,OFFSET STR

MOV DI,0003H

BACK: MoV AL,[SI]

XCHG [DI],AL

MOV [SI],AL

INC SI

DEC DI

DEC CL

JNZ BACK

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : STR(DS:0000H)=01H,02H,03H,04H**

**OUTPUT:= STR(DS:0000H)= 04H,03H,02H,01H**

**5.3. LENGTH OF THE STRING**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

STR DB 01H,03H,08H,09H,05H,07H,02H

LENGTH DB ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AL,00H

MOV CL,00H

MOV SI,OFFSET STR

BACK:CMP AL,[SI]

JNC GO

INC CL

INC SI

JNZ BACK

GO:MOV LENGTH,CL

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : STR(DS:0000H)=01H,03H,08H,09H,05H,07H,02H**

**OUTPUT:= LENGTH=07H[CL]**

**5.4. STRING COMPARISION**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

STR DB 04H,05H,07H,08H

COUNT EQU 04H

ORG 0010H

STR1 DB 04H,06H,07H,09H

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV SI,OFFSET STR

MOV DI,OFFSET STR1

MOV CL,COUNT

CLD

REP CMPSB

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**INPUT : STR(DS:0000H)=04H,05H,07H,08H**

**STR(DS:0000H)= 04H,06H,07H,09H**

**OUTPUT:= IF STR=STR1 THEN ZF=1**

**IF STR =\ STR1 THEN ZF=0**

**5.5. DOS/BIOS PROGRAMMING**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

MSG DB ODH,0AH,”WELCOME TO MICRO PROCESSOR LAB”, 0DB,0AH,”$”

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,09H

MOV DX,OFFSET MSG

INT 21H

CODE ENDS

END START

END

**RESULT:-**

**WELCOME TO MICRO PROCESSORS LAB**

**6.1. PACKED BCD TO UNPACKED BCD**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

BCD DB 48H

UBCD DB ?

UBCD2 DB ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AL,BCD

MOV BL,AL

AND AL,0FH

MOV UBCD1,AL

MOV AL,BL

AND AL,0F0H

MOV CL,04H

ROR AL,CL

MOV UBCD2,AL

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

RESULT:-

INPUT: 48

OUTPUT:- 0408

**6.2. PACKED BCD TO ASCII**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

BCD DB 49H

ASCII1 DB ?

ASCII2 DB ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AL,BCD

MOV BL,AL

AND AL,0FH

OR AL,30H

MOV ASCII1,AL

MOV AL,BL

AND AL,0F0H

MOV CL,04H

ROR AL,CL

MOV ASCII2,AL

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

RESULT:-

INPUT: 49

OUTPUT:- 3439

**7.1. ASCENDING ORDER**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

NUMS DW 5H,4H,3H,2H,1H

COUNT EQU 05H

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,0000H

MOV DL,COUNT-1

BACK1:MOV CL,DL

MOV SI,OFFSET NUMS

BACK: MOV AX,[SI]

CMP AX,[SI+2]

JC GO

XCHG [SI+2],AX

MOV [SI],AX

GO:INC SI

INC SI

LOOP BACK

DEC DL

JNZ BACK1

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

RESULT:-

INPUT: 5H,4H,3H,2H,1H

OUTPUT:- 1H,2H,3H,4H,5H

**7.2. DESCENDING ORDER**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

NUMS DW 1H,2H,3H,4H,5H

COUNT EQU 05H

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,0000H

MOV DL,COUNT-1

BACK1:

MOV CL,DL

MOV SI,OFFSET NUMS

BACK: MOV AX,[SI]

CMP AX,[SI+2]

JNC GO

XCHG AX,[SI+2]

MOV [SI],AX

GO:

INC SI

INC SI

LOOP BACK

DEC DL

JNZ BACK1

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

RESULT:-

INPUT: 1H,2H,3H,4H,5H

OUTPUT:- 5H,4H,3H,2H,1H

**8.1. MAXIMUM NUMBER**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

DLMS DW 0001H,0009H,0008H,0005H,0010H

COUNT EQU 05H

MAX DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV CX,COUNT-1

MOV SI,OFFSET DLMS

MOV AX,[SI]

BACK : CMP AX,[SI+2]

JNC GO

XCHG AX,[SI+2]

GO: INC SI

INC SI

LOOP BACK

MOV MAX,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

RESULT:-

INPUT: 0001H,0009H,0008H,0005H,0010H

OUTPUT:- STORED IN A&B LOCATION OF DS

**8.2. MINIMUM NUMBER**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

DLMS DW 0007H,0009H,000FH,0008H,0005H,0006H

COUNT EQU 06H

MIN DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV CX,COUNT-1

MOV SI,OFFSET DLMS

MOV AX,[SI]

BACK : CMP AX,[SI+2]

JC GO

XCHG AX,[SI+2]

GO: INC SI

INC SI

LOOP BACK

MOV MIN,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

RESULT:-

INPUT: 0007H,0009H,000FH,0008H,0005H,0006H

OUTPUT:- 0005H IS IN C&D LOCATION

**9.1. 2’S COMPLEMENT**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

OPR1 DW 45H

RES DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

NEG OPR1

MOV RES,AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

END

RESULT:-

INPUT: OPR1=0045H

OUTPUT:- FFBBH

**9.2. AVERAGE OF TWO NUMBERS**

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

NO1 DB 0FH

NO2 DB 05H

AVG DW ?

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AX,00H

MOV AL,NO1

MOV AL,NO2

ADD AL,NO2

SAR AX,01H

MOV AVG,AX

INT 21H

CODE ENDS

END START

END

RESULT:-

INPUT: NO1=0FH,, NO2=05H

OUTPUT:- 0AH IS IN ACCUMULATOR REGISTER

Additional programs

String Operations

**Left shift operation**

mov si,7800H

mov cl,[si]

mov ax,si

add ax,cx

mov si,ax

mov al,[si]

mov [si],00h

dec cl

back : dec si

mov bl,[si]

mov [si],al

mov al,bl

dec cl

jnz back

hlt

**RIGHT shift operation**

mov si,7800h

mov cl,[si]

inc si

mov al,[si]

mov [si],00h

dec ci

back : inc si

mov bl,[si]

mov [si],al

mov al,bl

dec cl

jnz back

hlt

**COUNTING OF OCCURRENCE OF A LETTER IN A GIVEN STRING**

Data Segment

STR DB ‘AIMTOBECOMEAASTRONUT’

A DB 0H

MSG1 DB 10,13,’COUNT OF A IS:$’

DATA ENDS

DISPLAY MACRO MSG

MOV AH,9

LEA DX,MSG

INT 21H

ENDM

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:

MOV AX,DATA

MOV DS,AX

LEA SI,STR1

MOV CX,10

CHECK:

MOV AL,[SI]

CMP AL,’A’

JNE N1

INC A

N1:

CMP AL,’A’

JNE N2

INC A

N2:INC SI

LOOP CHECK

MOV AL,A

DISPLAY MSG1

MOV DL,A

ADD DL,30H

MOV AH,2

INT 21H

MOV AH,4CH

INT 21H

CODE ENDS

END START

RESULT:

Count A is 2

**Reversing a String**

Data segment

N1 db ‘communication’

Len equ ($-n1)

N2 db len dup(00)

Data ends

Code segment

Assume cs:code,ds:data

Start:

mov ax,data

mov ds,ax

mov bx,offset n1

mov si,bx

mov di,offset n2

add di,len

cld

mov cx,len

a1:mov al,[si]

mov [di],al

inc si

dec di

loop a1

mov [di],’$’

mov dx,offset n1

mov ah,09

int 21h

mov dx,offset n2+1

mov ah,09

int 21h

hlt

code end

end start

Result :

Communication

**Display a Given String**

Prog:

.model small

.stack 100h

.data

String1 db ‘Electronics and Communication Engineering $’

.code

Main proc

Mov AX,@data

Mov DS,AX

MOV AH,09H

MOV DX,OFFSET STRING1

INT 21H

MOV AH,4CH

INT 21H

MAIN ENDP

END MAIN

RESULT :

Electronics and Communication Engineering

**FACTORIAL NUMBER**

**MOV AX,06H**

**MOV CX,AX**

**AHEAD:DEC CX**

**JNZ COPY**

**PROCEED:MUL CX**

**JNZ AHEAD**

**COPY:MOV DX,AX**

**HLT**

RESULT:

AX=0006 BX=0000 CX=0005 DX=0006 SP=FFFE BP=0000 SI=0000 DI=0000

DS=0100 ES=0100 SS=0100 CS=0100 IP=000E NV UP EI PL NZ NA PE NC

0100:000E F4 HLT

SUM OF NUMBERS OF AN ARRAY

MOV SI,7800H

MOV CX,[SI]

INC SI

INC SI

MOV AX,000H

LOOP: ADD AX,[SI]

INC SI

INC SI

DEC CI

JNZ LOOP

MOV [SI],AX

INT 21H

AX=0000 BX=0000 CX=00A0 DX=0000 SP=FFFE BP=0000 SI=78C4 DI=0000

DS=0100 ES=0100 SS=0100 CS=0100 IP=000E NV UP EI PL NZ NA PO NC

0100:000E FEC9 DEC CL