ADDITION-8 bit

```
DATA SEGMENT
      A DB 03H;
      B DB 02H;
      RES DB ?
DATA ENDS

CODE SEGMENT
      START:
      ASSUME CS:CODE,DS:DATA
      MOV AX,DATA
      MOV DS,AX
      MOV AL,A
      MOV BL,B
      ADD AL,BL
      MOV RES,AL
      INT 03H
      END START
CODE ENDS
```

ADDITION-16 Bit

```
data segment
a dw 0014h
b dw 0016h
res dw ?
data ends
code segment
assume cs:code,ds:data
start:mov ax,data
mov ds,ax
mov ax,00h
mov bx,00h
mov ax,a
mov bx,b
add ax,bx
mov res,ax
int 03h
code ends
end start
```

SUBTRACTION-8 Bit

```
data segment
a dw 06h
b dw 02h
c dw ?
data ends
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov ax,a
mov bx,b
sub ax,bx
mov c,ax
int 03h
code ends
end start
```

SUBTRACTION-16 Bit

```
data segment
a dw 0006h
b dw 0004h
res dw ?
data ends
code segment
assume cs:code,ds:data
start:mov ax,data
mov ds,ax
mov ax,00h
mov bx,00h
mov ax,a
mov bx,b
sub ax,bx
mov res,ax
int 03h
code ends
end start
```

MULTIPLICATION-8 Bit

```
DATA SEGMENT
    A DW 0006H
    B DW 0003H
    RES DW ?
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
    START:
    MOV AX,DATA
    MOV DS,AX
    MOV AX,A
    MOV BX,B
    MUL BX
    MOV RES,AX
    INT 21H
    CODE ENDS
    END START
```

MULTIPLICATION-16 Bit

```
DATA SEGMENT
    A DW 12C1H
    B DW 1999H
    RES DW ?
    RES1 DW ?
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
    START:
    MOV AX,DATA
    MOV DS,AX
    MOV AX,A
    MOV BX,B
    MUL BX
    MOV RES,AX
    MOV RES1,DX
    INT 03H
    CODE ENDS
    END START
```

DIVISION-16 Bit by 8 Bit

```
data segment
a dw 0FEh
b dw 05h
quo dw ?
rem dw ?
data ends
code segment
start:
assume cs:code,ds:data
mov ax,data
mov ds,ax
mov ax,a
mov bx,b
div bx
mov quo,ax
mov rem,dx
int 03h
end start
code ends
```

LARGEST NO.

```
DATA SEGMENT
    A DB 12H,14H,03H,69H,42H, 22H,19H,20H,24H,04H
    SIZ DB 0AH
    OUTPUT DB ?
    SML DB ?
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
    START:
    MOV AX,DATA
    MOV DS,AX
    LEA SI,A
    LEA DI,SML
    MOV CL,SIZ
    MOV AH,00H
    CALL LARGEST
    JMP FINISH
    LARGEST PROC NEAR
    MOV AL,[SI]
    UP:
        DEC CL
        JZ FNS
        INC SI
        MOV BL,[SI]
        CMP AL,BL
        JG UP
        MOV AL,[SI]
        JMP UP
    FNS:
        MOV [DI],AL
        RET
        LARGEST ENDP
    FINISH:
    INT 03H
    CODE ENDS
    END START
```

FIND SMALLEST NUMBER

```
DATA SEGMENT
    A DB 12H,14H,03,69H,42H,22H,19H,20H,24H,04H
    SIZ DB 0AH
    OUTPUT DB ?
    SML DB ?
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
    START:
    MOV AX,DATA
    MOV DS,AX
    LEA SI,A
    LEA DI,SML
    MOV CL,SIZ
    CALL SMALLEST
    JMP FINISH
    SMALLEST PROC NEAR
    MOV AL,[SI]
    UP:
        DEC CL
        JZ FNS
        INC SI
        MOV BL,[SI]
        CMP AL,BL
        JNG UP
        MOV AL,[SI]
        JMP UP
    FNS:
        MOV [DI],AL
        RET
        SMALLEST ENDP
    FINISH:
    INT 03H
    CODE ENDS
    END START
```

Factorial(mixed language programming)

```
#include<iostream.h>
#include<conio.h>
void main()
{
        clrscr();
        short a;
        unsigned int c;
        cout<<"Enter a number between 0 to 8"<<endl;
        cin>>a;
        asm mov ax,0000h
        asm mov al,01h
        asm mov cx,0000h
        asm mov cx,a
        back:
                asm mul cx
                asm dec cx
                asm jnz back
                asm mov c,ax
        cout<<endl<<"The factorial of A is "<<a;
        getch();
}
```

Move the String

```
DATA SEGMENT
    STRING1 DB 0AH,"Namaste$"
    LEN DB ($-STRING1)
DATA ENDS
EXTRA SEGMENT
    STRING2 DB 20 DUP(0)
EXTRA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, ES: EXTRA
    START:
        MOV AX,DATA
        MOV DS,AX
        MOV AX,EXTRA
        MOV ES,AX
        LEA SI,STRING1
        LEA DI,STRING2
        MOV CL,LEN
        CLD
        REP MOVSB
        INT 21H
        CODE ENDS
    END START
```

COUNT NUMBER OF VOWELS

```
DATA SEGMENT
          A DB 0AH,0DH,'ENTER THE STRING','$'
          B DB 0AH,0DH,'THE NUMBER OF VOWELS:','$'
          VOWEL DB 'A','A','E','E','I','I','O','O','U','U','$'
          DATABUF DB 100,0,100  DUP('$')
DATA ENDS
CODE SEGMENT
          ASSUME CS:CODE,DS:DATA
          START:
          MOV AX,DATA
          MOV DS,AX
          LEA DX,A
          MOV AH,09H
          INT 21H
          LEA DX,DATABUF
          MOV AH,0AH
          INT 21H
          MOV SI,DX
          LEA DX,B
          MOV AH,09H
          INT 21H
          MOV BL,00H
          CHECK: LEA DI,VOWEL
          MOV CX,000AH
          MOV AL,[SI]
          CONT: CMP AL,[DI]
          JE FOUND
          INC DI
          LOOP CONT
          JMP NEXT
          FOUND: INC BL
          NEXT: INC SI
          CMP DATABUF[SI],0AH
          JNE CHECK
          MOV DL,BL
          ADD DL,30H
          MOV AH,02H
          INT 21H
          CODE ENDS
```

```
END START
```

Compare two strings using Macros

```
GETSTR MACRO STR
MOV AH,0AH
LEA DX, STR
INT 21H
ENDM
PRINTSTR MACRO STR
MOV AH, 09H
LEA DX,STR
INT 21H
ENDM
DATA SEGMENT
STR1 DB 80,80 DUP(&#39;$&#39;)
STR2 DB 80,80 DUP(&#39;$&#39;)
MSG1 DB 20H, &quot;ENTER THE FIRST STRING: $&quot;
MSG2 DB 20H, &quot;ENTER THE SECOND STRING: $&quot;
MSG3 DB 20H, &quot;THE TWO STRINGS ARE EQUAL $&quot;
MSG4 DB 20H, &quot;THE TWO STRINGS ARE NOT EQUAL $&quot;
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA, ES: DATA
START:
MOV AX,DATA
MOV ES,AX
MOV DS,AX
PRINTSTR MSG1
GETSTR STR1
PRINTSTR MSG2
GETSTR STR2
LEA SI,STR1+2
LEA DI,STR2+2
MOV CL,STR1+1
MOV CH,00H
REPE CMPSB
JNE NOTEQUAL
PRINTSTR MSG3
JMP JAY1
NOTEQUAL:
PRINTSTR MSG4
```

```
JAY1:
MOV AX,4C00H
INT 21H
CODE ENDS
END START
```

String display using interrupt(int 21h)

```
DATA SEGMENT
        STRING2 DB "NAMASTE NAMASTE$"
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE, DS:DATA
        START:
        MOV AX,DATA
        MOV DS,AX
        LEA DX, STRING2
        MOV AH,09H
        INT 21H
        MOV AH,4CH
        INT 21H
        CODE ENDS
END START
```

PALINDROME

```
DATA SEGMENT
```

```
    STR1 DB 'TINTIN'
    LEN EQU $-STR1
    STR2 DB 20 DUP(0)
    MES1 DB 10,13,'WORD IS PALINDROME$'
    MES2 DB 10,13,'WORD IS NOT PALINDROME$'
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA,ES:DATA
    START:
        MOV AX,DATA
        MOV DS,AX
        MOV ES,AX
        LEA SI,STR1
        LEA DI,STR2+LEN-1
        MOV CX,LEN
      UP: CLD
        LODSB
        STD
        STOSB
        LOOP UP
        LEA SI,STR1
        LEA DI,STR2
        CLD
        MOV CX,LEN
        REPE CMPSB
        CMP CX,0H
        JNZ NOTPALIN
        LEA DX,MES1
        MOV AH,09H
        INT 21H
        JMP EXIT
        NOTPALIN:
        LEA DX,MES2
        MOV AH,09H
        INT 21H
        EXIT:
        MOV AH,4CH
        INT 21H
        CODE ENDS
       END START
```