ADDITION 8 BIT

```
;addition is done in two ways 8+8 and 16+16
;The syntax is add <operand1> <operand2> the sum is stored in operand1

data segment
var1l db 09h
var1h db 01h
sum db ?
data ends

code segment
assume cs:code, ds:data
start:
 mov ax, data
 mov ds, ax
 mov ah, var1l
 mov al, var1h
 add al,ah
 mov sum,al
 int 3h
code ends
end start
```

ADDITION 16 BIT

```
;addition is done in two ways 8+8 and 16+16
;The syntax is add <operand1> <operand2> the sum is stored in operand1

data segment
var1l dw 0902h
var1h dw 0106h
sum dw ?
data ends

code segment
assume cs:code, ds:data
start:
 mov ax, data
 mov ds, ax
 mov ax, var1l
 mov bx, var1h
 add ax,bx
 mov sum,ax
 int 3h
code ends
end start
```

ADDITION Packed BCD

```
;The logic of using daa is turning a hexa number to decimal.
;All the input and calculation are done in hexa but converted to decimal when used
daa instruction

data segment
x db 23h
y db 56h
sum db ?
data ends
code segment
assume cs:code,ds:data
start:
 mov ax,data; //
 mov ds,ax; to assign the data segment
 mov al,x
 add al, y
 daa
 mov sum, al
 int 3h
```

```
code ends
end start
```

SUBSTRACTION 8 Bit

```
;substraction is done in two ways 8+8 and 16+16
;The syntax is sub <operand1> <operand2> the difference is stored in operand1

data segment
var1 db 06h
var2 db 04h
diff db ?
data ends
code segment
assume cs:code, ds:data
start:
 mov ax, data
 mov ds, ax
 mov al, var1
 mov ah, var2
 sub al, ah
 mov diff, al
 int 3h
code ends
end start
```

SUBSTRACTION 16 Bit

```
;substraction is done in two ways 8+8 and 16+16
;The syntax is sub <operand1> <operand2> the difference is stored in operand1

data segment
var1 dw 0809h
var2 dw 0605h
diff dw ?
data ends
code segment
assume cs:code, ds:data
start:
 mov ax, data
 mov ds, ax
 mov ax, var1
 mov bx, var2
 sub ax, bx
 mov diff, ax
 int 3h
code ends
end start
```

SUBSTRACTION Packed

```
;The logic of using das is turning a hexa number to decimal.
;All the input and calculation are done in hexa but converted to decimal when used
das instruction

data segment
x db 56h
y db 23h
sum db ?
data ends
code segment
assume cs:code,ds:data
start:
 mov ax,data; //
 mov ds,ax; to assign the data segment
 mov al,x
```

```
 sub al, y
 das
 mov sum, al
 int 3h
code ends
end start
```

MULTIPLICATION 8*8

```
;mul has only one operand and it support 8*8 and 16*16 only
;In 8*8 multiplication the first is stored in al and other can be specified by us
and the result is stored in ax
;In 16*16 multiplication the first is stored in ax and other can be specified by
us and the result is stored in dx(higher order bit) and ax(lower order bit)
data segment
var1 db 16h
var2 db 18h
pdt dw ?
data ends
code segment
assume cs:code, ds:data
start:
 mov ax, data
 mov ds, ax
 mov al, var1
 mov ah, var2
 mul ah  ;mul done as al*ah implicitly and saved in ax
 mov pdt, ax ;product
 int 3h
code ends
end start
```

MULTIPLICATION 16*16

```
;mul has only one operand and it support 8*8 and 16*16 only
;In 8*8 multiplication the first is stored in al and other can be specified by us
and the result is stored in ax
;In 16*16 multiplication the first is stored in ax and other can be specified by
us and the result is stored in dx(higher order bit) and ax(lower order bit)
data segment
var1 dw 0304h
var2 dw 0609h
pdtl dw ?
pdth dw ?
data ends
code segment
assume cs:code, ds:data
start:
 mov ax, data
 mov ds, ax
 mov ax, var1
 mov bx, var2
 mul bx  ;mul done as al*ah implicitly and saved in ax
 mov pdtl, ax ;lower order bit result
 mov pdth, dx ;higher order bit result
 int 3h
code ends
end start
```

DIVISION 16/8

```
;div has only one operand and it support 16/8 and 32/16 only
;In 16/8 division the first is stored ax and other can be specified by us and the
result is stored as remainder in ah and quoitent in al
;In 32/16 division the first is stored in dx(higher order bit) and ax(lower order
```

```
bit) and other can be specified by us and the result is stored as remainder in dx
and quoitent in ax

data segment
x dw 2314h
y db 26h
q db ?
r db ?
data ends
code segment
assume cs:code,ds:data
start:
 mov ax,data
 mov ds,ax
 mov ax,x
 div y  ;div done as ax/bh implicitly and saved as remainder in ah and quoitent in
al
 mov q,al
 mov r,ah
 int 3h
 code ends
end start
```

DIVISION 32/16

```
;div has only one operand and it support 16/8 and 32/16 only
;In 16/8 division the first is stored ax and other can be specified by us and the
result is stored as remainder in ah and quoitent in al
;In 32/16 division the first is stored in dx(higher order bit) and ax(lower order
bit) and other can be specified by us and the result is stored as remainder in dx
and quoitent in ax

data segment
x dw 2314h,1234h
y dw 2567h
q db ?
r db ?
data ends
code segment
assume cs:code,ds:data
start:
 mov ax,data
 mov ds,ax
 mov ax,x
 mov dx,x+2
 div y  ;div done as dxax/bx implicitly and saved as remainder in dx and quoitent
in ax
 mov q,al
 mov r,ah
 int 3h
 code ends
end start
```

FACTORIAL

```
data segment
n db 05h
res dw ?
data ends
code segment
assume cs:code,ds:data
start:
 mov cl,05h
 mov al,01h
```

```
 LABLE:mul cl
 LOOP LABLE
 mov res,ax
 int 3h
 code ends
end start
```

LENGTH OF A STRING

```
data segment
str1 db "abcde"
data ends
code segment
assume cs:code,ds:data
start:
 mov ax,data
 mov ds,ax
 mov cx,0h
 lea si,str1
 mov bl,'e'
Lable: cmp [str1+si],bl
 inc cx
 jnc cont
 inc si
 jmp Lable
cont: int 3h
 code ends
end start
```

COMPARISION OF TWO STRING

```
;Comparing two strings
;cmpsb compares the bytes at DS:SI and ES:DI and sets the status flag accordingly
;If both are same zero flag is set to 0 else 1

data segment
str1 db "dcba"
result db ?
data ends
extra segment
str2 db "abcd"
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,str1 ;loading effective address of str1 to SI
 lea di,str2 ;loading effective address of str2 to DI
 mov cx,04h ;Setting conter to string length, used by cmpsb which compares that
many bytes
 cld  ;setting direction to forward
 repe cmpsb ;comparing strings
 jnz exit ;If zero flag is clear jump to Exit
 mov result,00h
 exit:
  mov result,01h
 int 3h
 code ends
end start
```

COPYING OF STRING

```
;Copy string to string
;movsb is used to copy bytes of data from DS:SI to ES:DI
;No flags are affected

data segment
str1 db "1234"
data ends
extra segment
str2 db ?
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,str1 ;loading effective address of str1 to SI
 lea di,str2 ;loading effective address of str2 to DI
 mov cx,04h ;Setting conter to string length, used by movsb moves that many bytes
 cld  ;setting direction to forward
 repe movsb ;moving string
 int 3h
 code ends
end start


*Scanning of string and 32bit addition using loops will be added as soon as
possible ....GA2
```