IPCL

**1. Display the results in terminal (STDOUT) for addition**

section .data

msg db 'Enter first number :',10

msglen: equ $-msg

msg1 db 'Enter second number :',10

msglen1: equ $-msg1

msg2 db 'Addition is :',10

msglen2: equ $-msg2

section .bss

num1 resb 1

num2 resb 1

res resb 1

section .text

global \_start

\_start:

mov eax,4

mov ebx,1

mov ecx,msg

mov edx,msglen

int 80h

mov eax,3

mov ebx,0

mov ecx,num1

mov edx,2

int 80h

mov eax,4

mov ebx,1

mov ecx,msg1

mov edx,msglen1

int 80h

mov eax,3

mov ebx,0

mov ecx,num2

mov edx,2

int 80h

mov eax,4

mov ebx,1

mov ecx,msg2

mov edx,msglen2

int 80h

mov eax,[num1]

sub eax,'0'

mov ebx,[num2]

sub ebx,'0'

add eax,ebx

add eax,'0'

mov [res],eax

int 80H

mov eax,4

mov ebx,1

mov ecx,res

mov edx,1

int 80h

mov eax,01H

mov ebx,10

int 80H

**2. SUBTRACTION**

section .data

msg db 'Enter first number :',10

msglen: equ $-msg

msg1 db 'Enter second number :',10

msglen1: equ $-msg1

msg2 db 'Subtraction is :',10

msglen2: equ $-msg2

section .bss

num1 resb 1

num2 resb 1

res resb 1

section .text

global \_start

\_start:

mov eax,4

mov ebx,1

mov ecx,msg

mov edx,msglen

int 80h

mov eax,3

mov ebx,0

mov ecx,num1

mov edx,2

int 80h

mov eax,4

mov ebx,1

mov ecx,msg1

mov edx,msglen1

int 80h

mov eax,3

mov ebx,0

mov ecx,num2

mov edx,2

int 80h

mov eax,4

mov ebx,1

mov ecx,msg2

mov edx,msglen2

int 80h

mov eax,[num1]

sub eax,'0'

mov ebx,[num2]

sub ebx,'0'

sub eax,ebx

add eax,'0'

mov [res],eax

int 80H

mov eax,4

mov ebx,1

mov ecx,res

mov edx,1

int 80h

mov eax,01H

mov ebx,10

int 80H

**3. MULTIPLICATION**

section .data

msg db 'Enter first number :',10

msglen: equ $-msg

msg1 db 'Enter second number :',10

msglen1: equ $-msg1

msg2 db 'Multiplication is :',10

msglen2: equ $-msg2

section .bss

num1 resb 4

num2 resb 4

res resb 4

section .text

global \_start

\_start:

mov eax,4

mov ebx,1

mov ecx,msg

mov edx,msglen

int 80h

mov eax,3

mov ebx,0

mov ecx,num1

mov edx,2

int 80h

mov eax,4

mov ebx,1

mov ecx,msg1

mov edx,msglen1

int 80h

mov eax,3

mov ebx,0

mov ecx,num2

mov edx,2

int 80h

mov eax,4

mov ebx,1

mov ecx,msg2

mov edx,msglen2

int 80h

mov eax,[num1]

sub eax,'0'

mov ebx,[num2]

sub ebx,'0'

mul ebx

add eax,'0'

mov [res],eax

int 80H

mov eax,4

mov ebx,1

mov ecx,res

mov edx,1

int 80h

mov eax,01H

mov ebx,10

int 80H

**4. DIVISION**

**section .data**

**msg1: db 'Enter the Dividend: ',10 ;**

**msg1Len equ $-msg1 ;**

**msg2: db 'Enter the divisor: ',10 ;**

**msg2Len equ $-msg2 ;**

**msg3: db 'Quotient of two numbers: ',10 ;**

**msg3Len equ $-msg3 ;**

**msg4: db 10, 'Reminder of two numbers: ',10 ;**

**msg4Len equ $-msg4 ;**

**section .bss**

**num1 resb 2**

**num2 resb 2**

**quo resb 2**

**rem resb 2**

**section .text**

**global \_start**

**\_start:**

**mov eax,4**

**mov ebx,1**

**mov ecx,msg1 ;Printing first message**

**mov edx,msg1Len**

**int 80h**

**mov eax,3**

**mov ebx,0**

**mov ecx,num1 ;Getting first input**

**mov edx,2**

**int 80h**

**mov eax,4**

**mov ebx,1**

**mov ecx,msg2 ;Printing second message**

**mov edx,msg2Len**

**int 80h**

**mov eax,3**

**mov ebx,0 ;Getting second input**

**mov ecx,num2**

**mov edx,2**

**int 80h**

**mov al, [num1] ;Converting ASCII to decimal**

**sub al,'0'**

**mov bl, [num2]**

**sub bl,'0'**

**div bl**

**add al,'0'**

**mov [quo],al**

**add ah,'0'**

**mov [rem],ah**

**mov eax,4**

**mov ebx,1**

**mov ecx,msg3 ;Printing third message**

**mov edx,msg3Len**

**int 80h**

**mov eax,4**

**mov ebx,1**

**mov ecx,quo ;Printing quo**

**mov edx,1**

**int 80h**

**mov eax,4**

**mov ebx,1**

**mov ecx,msg4 ;Printing fourth message**

**mov edx,msg4Len**

**int 80h**

**mov eax,4**

**mov ebx,1**

**mov ecx,rem ;Printing rem**

**mov edx,1**

**int 80h**

**mov eax,1**

**mov ebx,0**

**int 80h**

**5. Array addition**

section .data

msg db 10,13,9,'Addition is:'

msglen: equ $-msg

global x ;declaration and initialization of an array

x:

db 1

db 2

db 3

db 1

db 1

sum:

db 0

section .text

global \_start

\_start:

mov eax,4 ;SYS Call Write

mov ebx,1 ;STDOUT

mov ecx,msg ;Msg display

mov edx,msglen ;message length

int 80h

mov eax,x ;eax will point to the current element to

be summed

mov ebx,0 ;ebx will store result (Sum)

mov ecx,5 ;Length of an array

t:

add ebx,[eax] ;add array element with ebx

add eax,1 ;move pointer to next element

loop t ;if counter is not zero repeat loop

cmp bl,9 ;compare result with 9

jna t1 ;jna= jump if not above or jbe= jump

if below or equal

add bl,07 ;if result is >9, add 37H to convert

in to ASCII

t1: add bl,30h ;if result is <9, add 30H to convert

in to ASCII

mov [sum],bl ;move result to sum

mov edx,1 ;message length

mov ecx,sum ;message to write (Sum)

mov eax,4 ;system call number (System write)

mov ebx,1 ;stdout

int 80h

mov eax,1 ; The system call for exit (sys-Exit)

mov ebx,0 ; Exit with return code 0 (no error)

int 80h

**6. Division on two 8-bit numbers, input by user.**

section .data

msg db 'Enter the dividend :',10

msglen: equ $-msg

msg1 db 'Enter the divisor :',10

msglen1: equ $-msg1

msg2 db 'Quotient of division is = ',10

msglen2: equ $-msg2

msg3 db 10,'Remainder of division is = ',10

msglen3: equ $-msg3

section .bss

dividend resb 1

divisor resb 1

quotient resb 1

remainder resb 1

%macro RW 4

mov eax,%1

mov ebx,%2

mov ecx,%3

mov edx,%4

int 80h

%endmacro

section .text

global \_start

\_start:

RW 4,1,msg,msglen

RW 3,0,dividend,2

RW 4,1,msg1,msglen1

RW 3,0,divisor,2

call procdiv

RW 4,1,msg2,msglen2

RW 4,1,quotient,1

RW 4,1,msg3,msglen3

RW 4,1,remainder ,1

mov eax,1

mov ebx,0

int 80h

procdiv: mov al,[dividend]

sub al,'0'

mov ah,0

mov bl,[divisor]

sub bl,'0'

div bl ; Q(al) & R(ah)=ax/bl

add al,'0' ;number in to ascii

mov [quotient],al

add ah,'0' ;number in to ascii

mov [remainder],ah

ret

**aurdino**

**blinking light**

int led=13;

void setup()

{

pinMode(led, OUTPUT);

}

void loop()

{

digitalWrite(led, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(led, LOW);

delay(1000); // Wait for 1000 millisecond(s)

}

**temp sensor**

int temperaturePin =0;

void setup()

{

Serial.begin(9600);

}

void loop()

{

float voltage,degreesC, degreesF;

voltage =getVoltage(temperaturePin);

degreesC=(voltage-0.5)\*100.0;

degreesF= degreesC-(9.0/5.0)-32.0;

Serial.print("Voltage: ");

Serial.print(voltage);

Serial.print("Degree C: ");

Serial.print(degreesC);

Serial.print("Degree F: ");

Serial.print(degreesF);

delay(1000);

}

float getVoltage(int pin)

{

return(analogRead(pin)\*0.004882814);

}

**pir**

int pinsensor=2;

int pinled=12;

int pinbuzzer=13;

int pirsensor=0;

void setup()

{

pinMode(pinsensor, INPUT);

pinMode(pinled, OUTPUT);

pinMode(pinbuzzer, OUTPUT);

}

void loop()

{

pinsensor=digitalRead(pinsensor);

if(pinsensor==HIGH)

{

digitalWrite(pinled, HIGH);

tone(pinbuzzer,1000,500);

}

else

{

digitalWrite(pinled, LOW);

}

delay(10);

}

**ultrasonic**

#define echoPin 2

#define trigPin 3

long duration;

int distance;

void setup()

{

Serial.begin(9600);

pinMode(trigPin,OUTPUT);

pinMode(echoPin,INPUT);

}

void loop()

{

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration=pulseIn(echoPin,HIGH);

distance=(duration\*0.034/2);

Serial.print("Distance : ");

Serial.print(distance);

Serial.print("cm ");

delay(1000);

}