

## Práctica 03

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Fundamentos de Lenguajes de Programación

PRÁCTICA	TEMA	DURACIÓN
03	Ensamblador	3 horas

### 1. Datos de los estudiantes

- Grupo: 03
- Integrantes:
  - Andree Alvarez Guzman
  - Jonathan Aguirre Soto

### 2. Ejercicios

1. Implementar un programa que muestre la suma, la diferencia, la multiplicación, la división y el promedio de dos números ingresados por teclado.

```
>> Ingrese un numero: 3
>> Ingrese otro numero: 2
>> La suma es: 5
>> La diferencia es: 1
>> La multiplicacion es: 6
>> La division es: 1.5
>> El promedio es: 2.5
```

```
.data
out_string: .ascii "\Ingrese un primer numero\n"
out_string2: .ascii "\Ingrese otro numero\n"
out_string3: .ascii "\nLa suma es\n"
out_string4: .ascii "\nLa diferencia es\n"
out_string5: .ascii "\nLa multiplicacion es\n"
out_string6: .ascii "\nLa division es\n"
out_string7: .ascii "\nEl promedio es\n"

.text

main:
    li $s, $f5, 2.0

    li $v0 4 #ingreso 1
```

```
la $a0, out_string
syscall

li $v0, 6 #muevo
syscall
mov.s $f1, $f0

li $v0 4 #ingreso 2
la $a0, out_string2
syscall

li $v0, 6 #muevo
syscall
mov.s $f2, $f0

li $v0 4 #res1
la $a0, out_string3
syscall

li $v0, 2 #suma
add.s $f12, $f1, $f2
#move.s $a0 $t1
syscall

li $v0 4 #res2
la $a0, out_string4
syscall

li $v0, 2 #diferencia
sub.s $f12, $f1, $f2
syscall

li $v0 4 #res3
la $a0, out_string5
syscall

li $v0, 2 #multipl
mul.s $f12, $f1, $f2
syscall

li $v0 4 #res4
la $a0, out_string6
syscall

li $v0, 2 #dividir
div.s $f12, $f1, $f2
syscall

li $v0 4 #res5
la $a0, out_string7
syscall
```

```
#li $v0, 2 #promedio  
add.s $f12, $f1, $f2  
#div.s $f12, $f12, $f5
```

```
li $v0, 2 #promedio  
div.s $f12, $f12, $f5  
syscall
```

```
jr $ra
```

- Ejecución del programa.



```
Console  
\  
Ingrese un primer numero  
3  
\  
Ingrese otro numero  
2  
  
La suma es  
5.00000000  
La diferencia es  
1.00000000  
La multiplicacion es  
6.00000000  
La division es  
1.50000000  
El promedio es  
2.50000000|
```

2. Implementar un programa que solicite una cantidad  $n$  de números y luego retorne: la suma de estos, el promedio, el mayor y el menor.

---

```
>> Ingrese la cantidad de numeros: 4
>> Ingrese un numero: 3
>> Ingrese un numero: 2
>> Ingrese un numero: 2
>> Ingrese un numero: 3
>> La suma es: 10
>> El promedio es: 2.5
>> El mayor es: 3
>> El menor es: 2
```

---

```
.data
texto1: .asciiz "\Ingrese la cantidad de numeros: "
texto2: .asciiz "\Ingrese un numero: "
texto3: .asciiz "\nLa suma es: "
texto4: .asciiz "\nEl promedio es: "
texto5: .asciiz "\nEl mayor es: "
texto6: .asciiz "\nEl menor es: "

.text

main:
    li $v0, 4 #texto 1
    la $a0, texto1
    syscall

    li $v0, 5
    syscall
    move $t1, $v0

    #li.s $f1, 0.0
    li.s $f4, 1.0 #solo es para sumar de 1 en 1
    li.s $f5, 0.0
    li $t2, 0 #contador
Loop:
    beq $t1, $t2 Exit
    li $v0, 4 #texto 2
    la $a0, texto2
    syscall

    li $v0, 6
    syscall

    #mayor y menor
    #mayor#
    #c.lt.d $f#0, $f1, LABEL_IF
    #LABEL_IF:#
    #      mov.s $f5, $f0
    mov.s $f1, $f0
```

```
add.s $f2, $f2, $f1 #suma de n numeros
add $t2, $t2, 1 #contador del loop
add.s $f3, $f3, $f4 #para el promedio
```

```
j Loop
```

```
Exit:
```

```
li $v0, 4 #texto 3
```

```
la $a0, texto3
```

```
syscall
```

```
li $v0, 2
```

```
mov.s $f12, $f2
```

```
syscall
```

```
li $v0, 4 #texto 4
```

```
la $a0, texto4
```

```
syscall
```

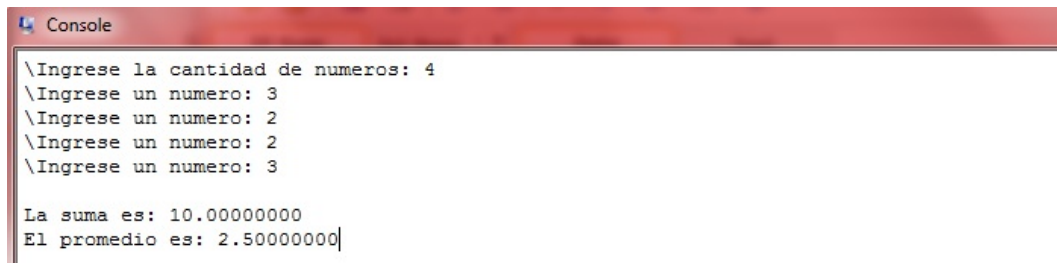
```
li $v0, 2
```

```
div.s $f12, $f2, $f3
```

```
syscall
```

```
jr $ra
```

- Ejecución del programa.



```
Console
\Ingrese la cantidad de numeros: 4
\Ingrese un numero: 3
\Ingrese un numero: 2
\Ingrese un numero: 2
\Ingrese un numero: 3

La suma es: 10.00000000
El promedio es: 2.50000000|
```

3. Implemente un programa que solicite por teclado: la longitud de los tres lados de un triángulo. Luego el programa debe indicar si es un triángulo válido.

---

```
>> Ingrese el primer lado del triángulo: 2
>> Ingrese el segundo lado del triángulo: 3
>> Ingrese el tercer lado del triángulo: 2
>> El triángulo es válido
```

---

```
.data
text_1: .asciiz "\nEl primer lado del triángulo\n"
text_2: .asciiz "\nEl segundo lado del triángulo\n"
text_3: .asciiz "\nEl tercer lado del triángulo\n"
text_4: .asciiz "\nTriángulo no válido\n"
text_5: .asciiz "\nTriángulo válido\n"

.text

main:

    li $v0, 4
    la $a0, text_1
    syscall

    li $v0, 5
    syscall

    move $a1, $v0

    li $v0, 4
    la $a0, text_2
    syscall

    li $v0, 5
    syscall

    move $t2, $v0

    li $v0, 4
    la $a0, text_3
    syscall

    li $v0, 5
    syscall

    move $t3, $v0

    add $a0, $a1, $t2
    bgtu $a0, $t3, et0 #si la suma de los dos primeros es mayor al ultimo
    #et0 = IF
```

```

LABEL_ELSE_0:
    la $a0, text_4
    b END_LABEL_IF

et0:
    la $a0, text_5

add $a0, $t2, $t3
bgtu $a1, $a0, et1 #si la suma de los dos ultimos es mayor al primero
#et1 = IF

LABEL_ELSE_1:
    la $a0, text_4
    b END_LABEL_IF

et1:
    la $a0, text_5

add $a0, $a1, $t3
bgtu $t2, $a0, et2 #si la suma de el primero y el ultimo es mayor al numero de
#et2 = IF

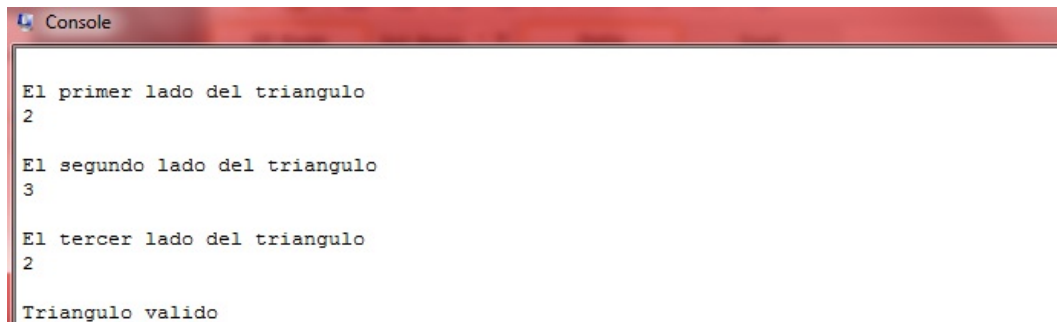
LABEL_ELSE_2:
    la $a0, text_4
    b END_LABEL_IF

et2:
    la $a0, text_5

END_LABEL_IF:
    li $v0, 4
    syscall

jr $ra
```

- Ejecución del programa.



```

Console

El primer lado del triangulo
2

El segundo lado del triangulo
3

El tercer lado del triangulo
2

Triangulo valido
```

4. Implemente un programa que solicite un número n, luego este debe mostrar que números desde 1 a 20, son múltiplos de n.

---

```
>> Ingrese un numero: 3
>> El numero 1 no es multiplo de 3
>> El numero 2 no es multiplo de 3
>> El numero 3 si es multiplo de 3
>> El numero 4 no es multiplo de 3
>> El numero 5 no es multiplo de 3
>> El numero 6 si es multiplo de 3
...
```

---

```
.data
texto1: .asciiz "Ingrese un numero: "
texto2: .asciiz "\nEl numero "
texto3: .asciiz " si es multiplo de "
texto4: .asciiz " no es multiplo de "

.text

main:
    li $v0, 4 #texto 1
    la $a0, texto1
    syscall

    li $v0, 5
    syscall
    move $t0, $v0 #numero n
    li $t1, 1 #contador
    Loop:
        beq $t1, 21 Exit
        div $t1, $t0
        mfhi $a0
        beq $a0, 0, LABEL_IF
        LABEL_ELSE:
            li $v0, 4
            la $a0, texto2
            syscall

            li $v0, 1
            move $a0, $t1
            syscall

            li $v0, 4
            la $a0, texto4
            syscall

            li $v0, 1
            move $a0, $t0
            syscall
            b END_LABEL_IF
        LABEL_IF:
        b END_LABEL_IF
    END_LABEL_IF
```



```
        li $v0, 4
        la $a0, texto2
        syscall

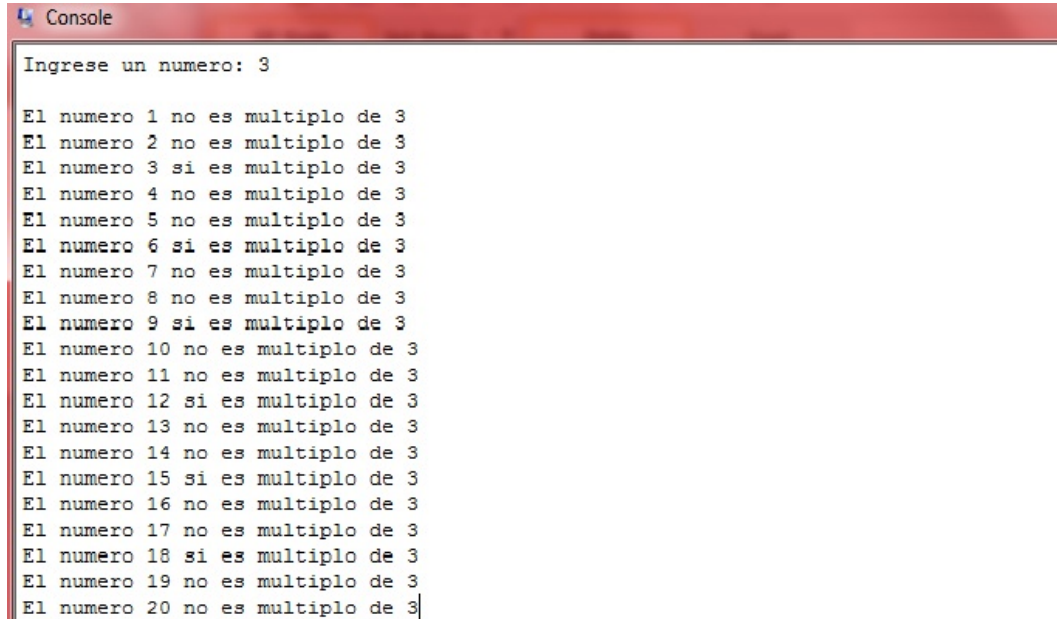
        li $v0, 1
        move $a0, $t1
        syscall

        li $v0, 4
        la $a0, texto3
        syscall

        li $v0, 1
        move $a0, $t0
        syscall
    ENDLABELIF:
        add $t1, $t1, 1 #contador del loop
    j Loop

Exit:
    jr $ra
```

- Ejecución del programa.



```
Console
Ingrese un numero: 3

El numero 1 no es multiplo de 3
El numero 2 no es multiplo de 3
El numero 3 si es multiplo de 3
El numero 4 no es multiplo de 3
El numero 5 no es multiplo de 3
El numero 6 si es multiplo de 3
El numero 7 no es multiplo de 3
El numero 8 no es multiplo de 3
El numero 9 si es multiplo de 3
El numero 10 no es multiplo de 3
El numero 11 no es multiplo de 3
El numero 12 si es multiplo de 3
El numero 13 no es multiplo de 3
El numero 14 no es multiplo de 3
El numero 15 si es multiplo de 3
El numero 16 no es multiplo de 3
El numero 17 no es multiplo de 3
El numero 18 si es multiplo de 3
El numero 19 no es multiplo de 3
El numero 20 no es multiplo de 3
```

5. Enlace de Github:

<https://github.com/Jona2010/Fundamentos-de-Lenguaje-de-la-Programaci-n>