

Ejercicio 1 Sumar dos números 7 y 5 y mostrar por pantalla

.data

```
msg:          .asciiz "Suma de dos numeros"
resultado:    .asciiz "Resultado es : "
retorno:      .asciiz "\n"
```

.text

```
.globl main
main: li $v0, 4    # syscall 4 (print_str)
      la $a0, msg  # argument: cadena string
      syscall      # llamada al sistema

      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: cadena string
      syscall      # llamada al sistema

      li $t1, 5     #carga de la constante 5 en el registro
      li $t2, 7     #carga de la constante 7 en el registro
      add $t1, $t1, $t2 #suma de los valores de los registros

      li $v0, 4     # syscall 4 (print_str)
      la $a0, resultado # argument: string
      syscall      # llamada al sistema

      li $v0, 4     # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string

      li $v0, 1     # syscall 4 (print_int)
      move $a0, $t1  # copiamos el registro en $a0
      syscall      # print the string

      jr $ra        # devolucion llamada punto siguiente invocacion
```

Ejercicio2. Solicite uno de los dos números y súmelos. Mostrar por pantalla el resultado

`.data`

```
msg: .asciiz "Sumar a 7 el numero:"
resultado: .asciiz "Resultado es : "
retorno: .asciiz "\n"
```

`.text`

```
.globl main
main: li $v0, 4    # syscall 4 (print_str)
      la $a0, msg  # argument: string
      syscall      # print the string
      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string

      #Tomar por pantalla un valor entero
      li $v0, 5     #syscall 5 (read_int)
      syscall
      move $t1, $v0

      li $t2, 7
      add $t1, $t1, $t2

      li $v0, 4     # syscall 4 (print_str)
      la $a0, resultado # argument: string
      syscall      # print the string

      li $v0, 4     # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string

      li $v0, 1     # syscall 4 (print_int)
      move $a0, $t1  # Copiamos valor en el registro esperado
      syscall      # print the string

      jr $ra        # Devolucion llamada
```

Ejercicio 3 . Crear un programa que permita listar los elementos de un array.

.data

```
msg:          .ascii "Listado array"
resultado:    .ascii "Valores almacenados son : "
retorno:      .ascii "\n"
arr: .word 100,200,300,400
indice: .word 1
```

.text

.globl main

```
main: li $v0, 4    # syscall 4 (print_str)
      la $a0, msg   # argument: string
      syscall      # print the string
      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string
      li $v0, 1    # syscall 4 (print_int)
      lw $t3,arr($zero)
      move $a0,$t3
      syscall      # print the string
      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string
      li $v0, 1    # syscall 4 (print_int)
      li $t2, 1    # argument: 1
      mul $t2,$t2,4
      lw $t3,arr($t2)
      move $a0,$t3 #Copiar registro
      syscall      # print the string
      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string
      li $v0, 1    # syscall 4 (print_int)
      li $t2, 2    # argument: 2
      mul $t2,$t2,4
      lw $t3,arr($t2)
      move $a0,$t3
      syscall      # print the string
      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string
      li $v0, 1    # syscall 4 (print_int)
      li $t2, 3    # argument: 3
      mul $t2,$t2,4
      lw $t3,arr($t2)
      move $a0,$t3
      syscall      # print the string
      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string
```

```

li $v0, 1    # syscall 4 (print_int)

li $t2, 4    # argument: 4
mul $t2,$t2,4
lw $t3,arr($t2)
move $a0,$t3
syscall      # print the string
jr $ra       # Devolucion llamada a punto invocacion

```

Ejercicio 4 . Multiplicar dos números positivos recogidos por pantalla.Si uno de los números es negativo no hacer nada y mostrar mensaje por pantalla.

Multiplicamos los dos valores leidos desde el terminal del teclado

.data

```

msg:          .asciiz "Multiplicacion 2 numeros por teclado"
valor1:       .asciiz "Leer valor 1"
valor2:       .asciiz "Leer valor 2"
resultado:    .asciiz "Resultado es : "
retorno:      .asciiz "\n"

```

.text

```

.globl main
main: li $v0, 4    # syscall 4 (print_str)
      la $a0, msg  # argument: string
      syscall      # print the string
      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string

      li $v0, 4    # syscall 4 (print_str)
      la $a0, valor1 # argument: string
      syscall      # print the string
      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string

      #tomar por pantalla un valor entero
      li $v0,5     #syscall 5 (read_int)
      syscall
      move $t1, $v0

      li $v0, 4    # syscall 4 (print_str)
      la $a0, valor2 # argument: string
      syscall      # print the string

      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string

```

```

syscall      # print the string

#tomar por pantalla segundo entero
li $v0,5     #syscall 5 (read_int)
syscall
move $t2, $v0

li $v0, 4     # syscall 4 (print_str)
la $a0, resultado # argument: string
syscall      # print the string
li $v0, 4     # syscall 4 (print_str)
la $a0, retorno # argument: string
syscall      # print the string

mul $t2,$t1,$t2 #multiplicamos dos enteros
li $v0, 1     # syscall 4 (print_int)

move $a0,$t2
syscall      # print the string

jr $ra      # Devolucion llamada

```

Añadimos al código las comprobaciones oportunas sobre los valores recogidos y verificamos que sean mayores que cero.

`.data`

```

msg:         .asciiz "Multiplicacion 2 numeros por teclado"
valor1:      .asciiz "Leer valor 1"
valor2:      .asciiz "Leer valor 2"
resultado:   .asciiz "Resultado es : "
serror:      .asciiz "Valor incorrecto !! "
retorno:     .asciiz "\n"

```

`.text`

```

.globl main
main: li $v0, 4     # syscall 4 (print_str)
      la $a0, msg   # argument: string
      syscall      # print the string
      li $v0, 4     # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string

      li $v0, 4     # syscall 4 (print_str)
      la $a0, valor1 # argument: string
      syscall      # print the string
      li $v0, 4     # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string

      #tomar por pantalla un valor entero

```

```
li $v0,5    #syscall 5 (read_int)
syscall
move $t1, $v0
bltz $t1,error
```

```
li $v0, 4    # syscall 4 (print_str)
la $a0, valor2 # argument: string
syscall      # print the string
```

```
li $v0, 4    # syscall 4 (print_str)
la $a0, retorno # argument: string
syscall      # print the string
```

```
#tomar por pantalla segundo entero
li $v0,5    #syscall 5 (read_int)
syscall
move $t2, $v0
bltz $t2,error
```

```
li $v0, 4    # syscall 4 (print_str)
la $a0, resultado # argument: string
syscall      # print the string
li $v0, 4    # syscall 4 (print_str)
la $a0, retorno # argument: string
syscall      # print the string
```

```
mul $t2,$t1,$t2    #multiplicamos dos enteros
li $v0, 1    # syscall 4 (print_int)
```

```
move $a0,$t2
syscall      # print the string
b final
```

```
error:
li $v0, 4    # syscall 4 (print_str)
la $a0, serror # argument: string
syscall      # print the string
li $v0, 4    # syscall 4 (print_str)
la $a0, retorno # argument: string
syscall      # print the string
```

```
final:
jr $ra      # Devolucion llamada
```

Ejercicio 5 En un determinado array de valores, identifique el mayor valor y muéstrelolo por pantalla.

.data

```
msg:          .asciiz "Listado array"
resultado:    .asciiz "Valores almacenados son : "
maximo:       .asciiz "Valor maximo es : "
retorno:      .asciiz "\n"
arr: .word 10,200,30,40
indice: .word 1
```

.text

```
.globl main
main: li $v0, 4    # syscall 4 (print_str)
      la $a0, msg  # argument: string
      syscall      # print the string

      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string

      li $v0, 1    # syscall 4 (print_int)

      lw $t3,arr($zero)
      move $t0,$t3
      move $a0,$t3
      syscall      # print the string

      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string

      li $v0, 1    # syscall 4 (print_int)

      li $t2, 1    # argument: 1
      mul $t2,$t2,4
      lw $t3,arr($t2)
      move $a0,$t3
      syscall      # print the string

      bgt $t3,$t0,update1
      b after1
      update1:
      move $t0,$t3
      after1:

      li $v0, 4    # syscall 4 (print_str)
      la $a0, retorno # argument: string
      syscall      # print the string

      li $v0, 1    # syscall 4 (print_int)
```

```
li $t2, 2    # argument: 2
mul $t2,$t2,4
lw $t3,arr($t2)
move $a0,$t3
syscall      # print the string
```

```
bgt $t3,$t0,update2
b after2
update2:
move $t0,$t3
after2:
```

```
li $v0, 4    # syscall 4 (print_str)
la $a0, retorno # argument: string
syscall      # print the string
```

```
li $v0, 1    # syscall 4 (print_int)
```

```
li $t2, 3    # argument: 3
mul $t2,$t2,4
lw $t3,arr($t2)
move $a0,$t3
syscall      # print the string
```

```
bgt $t3,$t0,update3
b after3
update3:
move $t0,$t3
after3:
```

```
li $v0, 4    # syscall 4 (print_str)
la $a0, retorno # argument: string
syscall      # print the string
```

```
li $v0, 1    # syscall 4 (print_int)
```

```
li $t2, 4    # argument: 4
mul $t2,$t2,4
lw $t3,arr($t2)
move $a0,$t3
syscall      # print the string
```

```
bgt $t3,$t0,update4
b fin
update4:
move $t0,$t3
fin:
```

```
li $v0, 4    # syscall 4 (print_str)
la $a0, retorno # argument: string
syscall      # print the string
```

```
li $v0, 4    # syscall 4 (print_str)
la $a0, maximo # argument: string
syscall      # print the string

li $v0, 1    # syscall 4 (print_int)

move $a0,$t0
syscall      # print the string

jr $ra
```

Optimizar el código mostrado mediante el uso de un número menor de saltos y etiquetas

Ejercicio 6 Crear un sistema de selección de butacas. El usuario podrá escoger el tipo de butaca a partir de una lista de posibles opciones

Ejercicio Seleccion

.data

```
# Incluimos el salto de linea en la propia cadena
# a diferencia ejemplos anteriores que era independiente
opciones: .ascii "Leer opcion 1-Fumador 2-No Fumador 3-VIP -Resto \n"
fumador: .ascii "Asiento Fumador\n"
Nofumador: .ascii "Asiento no fumador\n"
vip: .ascii "Acceso VIP\n"
otros: .ascii "Otros\n"
```

.text

```
.globl main
main:
    li $v0, 4
    la $a0, opciones
    syscall

    #tomar por pantalla un valor entero
    li $v0, 5    #syscall 5 (read_character)
    syscall
    move $t1, $v0

opcionUno:    bne $t1, 1, opcionDos
              la $a0, fumador
              b fin
opcionDos:    bne $t1, 2, opcionTres
              la $a0, Nofumador
              b fin
opcionTres:    bne $t1, 3, opcionResto
              la $a0, vip
              b fin
opcionResto:    la $a0, otros

    fin: li $v0, 4
          syscall
          li $v0, 10 #terminar ejecucion del programa
          syscall
```