

Práctica 04

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

PRÁCTICA	TEMA	DURACIÓN	
04	Ensamblador	3 horas	

1. Competencias del curso

- Conocer el desarrollo histórico de los lenguajes de programación y los paradigmas de programación.
- Comprender el papel de los diferentes mecanismos de abstracción en la creación de facilidades definidas por el usuario así como los beneficios de los lenguajes intermedios en el proceso de compilación.

2. Competencias de la práctica

• Programar tareas básicas en lenguaje ensamblador.

3. Equipos y materiales

- Latex
- Conección a internet
- IDE de desarrollo

4. Entregables

- Se debe elaborar un informe en Latex donde se responda a cada ejercicio de la Sección 6.
- En el informe se debe agregar un enlace al repositorio Github donde esta el código.
- En el informe se debe agregar el código fuente asi como capturas de pantalla de la ejecución y resultados del mismo.
- Por cada 5 minutos de retraso, el alumno tendrá un punto menos.

5. Datos del estudiante

- Grupo: 03
- Integrante:
 - Jonathan Aguirre Soto



6. Ejercicios

1. Cree un array de 5 elementos con multiplos de un número ingresado por teclado, luego debe mostrar dicho array por pantalla. (4 puntos)

```
>> Ingrese un numero: 4
>> Los 5 primeros multiplos de 4 son:
>> 4
>> 8
>> 12
>> 16
>> 20
```

```
.data
  array_1: .word 1:5 #array len=10 of 1 values
  str1: .asciiz "Ingrese un numero: "
  str2: .asciiz "Los 5 primeros multiplos de "
  str3: .asciiz " son: "
         str4: .asciiz "\n"
 .text
 main:
  li $v0, 4
     la $a0, str1
     syscall
  li $v0, 5
     syscall
     move $t0, $v0 #Numero n
  la $t1, array_1
  li $t2, 1 #contador
      while1:
  beq $t5, $t0 end_while1
  mult $t2, $t0 #hallando los multiplos del numero n
  mflo $a0
  sw $a0, 0($t1)
  add $t1, $t1, 4
  add $t2, $t2, 1
  add $t5, $t5, 1
  j while1 # regresa al inicio del ciclo
     end_while1:
  li $v0, 4
     la $a0, str2
     syscall
     li $v0, 1
     move $a0, $t0
     syscall
     li $v0, 4
     la $a0, str3
     syscall
  la $t1, array_1
```



```
li $t5, 0
   #inicio del segundo ciclo
   while2:
      beq $t5, $t0 end_while2
lw $t3, 0($t1)
add $t1, $t1, 4
      li $v0, 4
  la $a0, str4
  syscall
li $v0, 1 #print int
move $a0, $t3
syscall
add $t5, $t5, 1 #contador
j while2 # regresa al inicio del ciclo
  end_while2:
jr $ra
```

■ Ejecución del programa.

Console

```
Ingrese un numero: 4
Los 5 primeros multiplos de 4 son:
4
8
12
16
```

2. Cree un programa que solicite 5 elementos por teclado y los almacene en un Array. Luego se debe mostrar estos elementos en orden inverso. (4 puntos)

```
>> Ingrese un numero: 1
>> Ingrese un numero: 2
>> Ingrese un numero: 3
>> Ingrese un numero: 4
>> Ingrese un numero: 5
>> Los elementos en orden inverso son:
>> 5 4 3 2 1
```

```
.data
  array_1: .word 1:5 #array
  str1: .asciiz "Ingrese un numero: "
         str2: .asciiz "Los elementos en orden inverso son: "
 .text
 main:
  la $t1, array_1
 while1:
  beq $t0, 5 end_while1
  li $v0, 4
     la $a0, str1
     syscall
     li $v0, 5
     syscall
     move $a0, $v0
     sw $a0, 0($t1)
     add $t1, $t1, 4
     add $t0, $t0, 1
  j while1 # regresa al inicio del ciclo
 end_while1:
  li $t0, 0
        li $v0, 4
     la $a0, str2
     syscall
         #inicio del segundo ciclo
  while2:
     beq $t0, 5 end_while2
     sub $t1, $t1, 4
     lw $t3, 0($t1)
     li $v0, 1 #imprime los enteros en forma inversa
     move $a0, $t3
     syscall
     li $a0, 32 #imprime un espacio
     li $v0, 11 #cdigo del sistema para imprimir un carcter
     syscall
```



```
add $t0, $t0, 1 #contador
j while2 # regresa al inicio del ciclo
end_while2:
jr $ra
```

■ Ejecución del programa.

```
Ingrese un numero: 1
Ingrese un numero: 2
Ingrese un numero: 3
Ingrese un numero: 4
Ingrese un numero: 5
Los elementos en orden inverso son: 5 4 3 2 1
```

3. Cree un programa que solite por teclado dos vectores unitarios en 3 dimensiones (un vector es un array de tamaño 3). Luego se debe mostrar el producto escalar de ambos vectores. (4 puntos)

El producto escalar se define como: $\overrightarrow{a} \cdot \overrightarrow{b} = (a_x * b_x) + (a_y * b_y) + (a_z * b_z)$

```
>> Ingrese un del vector 1: 2
>> Ingrese un del vector 1: 3
>> Ingrese un del vector 1: 4

>> Ingrese un del vector 2: 3
>> Ingrese un del vector 2: 2
>> Ingrese un del vector 2: 1
>> El producto escalar es: 15
```

```
.data
  array_1: .word 1:3 #array 1
  array_2: .word 1:3 #array 2
  str1: .asciiz "Ingrese un numero del vector 1: "
  str2: .asciiz "Ingrese un numero del vector 2: "
  str3: .asciiz "El producto escalar es: "
        str4: .asciiz "\n"
 .text
 main:
  la $t0, array_1
  la $t1, array_2
   while1:
  beq $t2, 3 end_while1
  li $v0, 4
     la $a0, str1
     syscall
     li $v0, 5
     syscall
     move $a0, $v0
     sw $a0, 0($t0)
     add $t0, $t0, 4
     add $t2, $t2, 1
  j while1 # regresa al inicio del ciclo
   end_while1:
  li $t2, 0
        li $v0, 4
     la $a0, str4
     syscall
        #inicio del segundo ciclo
  while2:
     beq $t2, 3 end_while2
     li $v0, 4
```



```
la $a0, str2
  syscall
  li $v0, 5
  syscall
  move $a0, $v0
  sw $a0, 0($t1)
  add $t1, $t1, 4
  add $t2, $t2, 1
  j while2 # regresa al inicio del ciclo
end_while2:
  li $t2, 0
  la $t0, array_1
  la $t1, array_2
  li $t6, 0
          #iniciio del tercer ciclo
  while3:
  beq $t2, 3 end_while3
  lw $t3, 0($t0) #cargo el array 1
  lw $t4, 0($t1) #cargo el array 2
  mult $t3, $t4 #hacemos la multiplicacion
  mflo $t5
  add $t6, $t6, $t5 #sumamos sus valores
  add $t0, $t0, 4
  add $t1, $t1, 4
  add $t2, $t2, 1 #contador
  j while3 # regresa al inicio del ciclo
end_while3:
                  li $v0, 4
              la $a0, str4
              syscall
  li $v0, 4
  la $a0, str3
  syscall
  li $v0, 1 #imprimer el valor de la suma
  move $a0, $t6
  syscall
  jr $ra
```

Ejecución del programa.

Console

```
Ingrese un numero del vector 1: 2
Ingrese un numero del vector 1: 3
Ingrese un numero del vector 1: 4
Ingrese un numero del vector 2: 3
Ingrese un numero del vector 2: 2
Ingrese un numero del vector 2: 1
El producto escalar es: 16
```



7. Rúbricas

Rúbrica	Cumple	Cumple con obs.	No cumple
Informe: El informe debe estar en Latex, con un formato limpio y facil de leer.	3	1.5	0
Implementación: Implementa la funcionalidad correcta de cada ejercicio.	12	6	0
Mensaje de ayuda: Agrega mensajes de ayuda (textos solici- tando que se ingrese los números, etc.) por consola.	5	2.5	0
Errores ortográficos: Por cada error ortográfico, se le descontara 1 punto.	-	-	-