

Práctica 05

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

PRÁCTICA	TEMA	DURACIÓN
05	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
- Conexión 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 así 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. Escribir un programa que pida al usuario un número entero y muestre por pantalla si es par o impar. (3 puntos)

```
>> Ingrese un numero: 3
>> El numero 3 es impar
```

```
>> Ingrese un numero: 4
>> El numero 4 es par
```

```
.data
    text_par: .asciiz "El numero es par"
    tex_impar:.asciiz "El numero es impar"
    text_numero:.asciiz "El numero introducido es: "

.text

main:
    addi $t0, $0, 2
    la $a0, text_numero #mostrando mensaje para pedir el numero
    li $v0, 4
    syscall

    li $v0, 5 #leemos el numero
    syscall

    div $2, $t0 #hacemos la division para saber si el residuo es 0 o 1
    mfhi $t1

    beq $t1, $0, print_1 #verifica si el contenido del registro es igual a 0

    la $a0, tex_impar #Si es impar
    li $v0, 4
    syscall
    j fin

print_1: la $a0, text_par #si es par
    li $v0, 4
    syscall
    j fin

fin: li $v0, 10 #system call 10; exit
    syscall
```

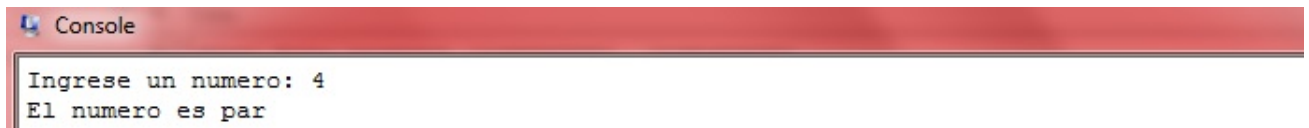
- Ejecución del programa.



A screenshot of a console window with a red title bar labeled "Console". The window contains two lines of text: "Ingrese un numero: 3" and "El numero es impar".

```
Console
Ingrese un numero: 3
El numero es impar
```

- Ejecución del programa.



A screenshot of a console window with a red title bar labeled "Console". The window contains two lines of text: "Ingrese un numero: 4" and "El numero es par".

```
Console
Ingrese un numero: 4
El numero es par
```

2. Escribir un programa que pida al usuario un número entero positivo y muestre por pantalla todos los números impares desde 1 hasta ese número. (4 puntos)

```
>> Ingrese un numero: 10
>> Los numeros impares hasta 10 son: 1 3 5 7 9
```

```
.data
    str1: .asciiz "Ingrese un numero: "
    str2: .asciiz "\nLos numeros impares hasta "
    str3: .asciiz " son: "
.text
main:
    li $v0, 4
    la $a0, str1 #primer texto
    syscall

    li $v0, 5
    syscall
    move $t0, $v0

    li $v0, 4
    la $a0, str2 #segundo texto
    syscall

    li $v0, 1
    move $a0, $t0
    syscall

    li $v0, 4
    la $a0, str3 #tercer texto
    syscall

    li $t1, 1
while1:
    bge $t1, $t0 end_while1 #saber si es el registro $t1 es mayor o igual al
        registro $t0

    li $v0, 1
    move $a0, $t1 #moviendo el registro $t1 al $a0
    syscall

    li $a0, 32 #print space
    li $v0, 11 #codigo del sistema para imprimir un carcter
    syscall

    add $t1, $t1, 2
    j while1

end_while1:
    li $v0, 10 #system call 10; exit
    syscall
```

- Ejecución del programa.

```
Console
Ingrese un numero: 10
Los numeros impares hasta 10 son: 1 3 5 7 9
```

3. Escribir un programa que pida al usuario un número entero y muestre por pantalla si es un número primo o no. (5 puntos)

```
>> Ingrese un numero: 10
>> El numero 10 no es primo
```

```
>> Ingrese un numero: 7
>> El numero 7 es primo
```

```
.data
    numero: .asciiz "\n Ingrese un numero: "
    noprimo: .asciiz "\n El numero no es primo"
    primo: .asciiz "\n El numero es primo"

.text
main:
    li $v0, 4
    la $a0, numero #primer texto
    syscall
    li $v0, 5 #pidiendo numero
    syscall
    move $t0, $v0 #moviendolo al registro $t0
    li $t1, 2

while1:
    beq $t0, $t1 es_primo #verifica si el contenido del registro de $t0 es
                           igual al del $t1

    div $t0, $t1 #division de los dos registros
    mfhi $t2 #mueve el contenido del registro a $t2
    beqz $t2, no_primo #si el contenido es mayor igual que 0
    addi $t1, $t1 1
    j while1

no_primo:
    li $v0, 4
    la $a0, noprimo
    syscall
    j exit

es_primo:
    li $v0, 4
    la $a0, primo
    syscall
    j exit
```

```
exit:
    li $v0, 10 #system call 10; exit
    syscall
```

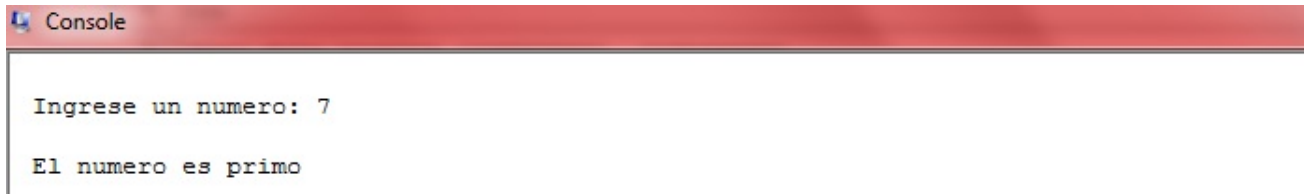
- Ejecución del programa.



The screenshot shows a terminal window titled "Console" with a red header bar. The text inside the terminal is as follows:

```
Ingrese un numero: 10
El numero no es primo|
```

- Ejecución del programa.

A screenshot of a console window with a red title bar labeled "Console". The window contains two lines of text: "Ingrese un numero: 7" and "El numero es primo".

```
Console  
Ingrese un numero: 7  
El numero es primo
```

4. Escriba un programa que entregue el máximo común divisor de dos enteros ingresados por teclado.
(5 puntos)

```
>> Ingrese un numero: 5
>> Ingrese un numero: 15
>> El maximo comun divisor es: 5
```

```
>> Ingrese un numero: 8
>> Ingrese un numero: 12
>> El maximo comun divisor es: 4
```

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

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 solicitando que se ingrese los números, etc.) por consola.	5	2.5	0
Errores ortográficos: Por cada error ortográfico, se le descontara 1 punto.	-	-	-