


GUÍA PRÁCTICA 01

1. DATOS GENERALES	
Asignatura: Taller de Programación en Bajo Nivel	Código de la Asignatura: inf-475
Carrera: Ingeniería en Informática y Electrónica	
Curso: A	Semestre: Cuarto
Contenido Analítico: <ul style="list-style-type: none"> • Introductoria • De los circuitos al procesador • Representación de datos 	Unidad Temática: INTRODUCCIÓN AL ENSAMBLADOR
Docente: Msc. Víctor Rodríguez Estévez	Email: v.rodriguez.umss.edu
Bibliografía a seguir: <ul style="list-style-type: none"> • PROGRAMACIÓN EN ASSEMBLER 	
Práctica: 1	Título: Introducción al lenguaje Ensamblador
Material de Apoyo: Diapositivas, lecturas	Carga horaria: 6

2. OBJETIVO
<ul style="list-style-type: none"> • Comprensión de la representación de datos.

3. MATERIAL Y/O EQUIPOS	
Detalle	Cantidad
Hoja de papel	1
Bolígrafos	78

1 Introducción

Antes de la era de los editores de código y los entornos de desarrollo interactivos (IDEs), los programas se codificaban de una manera física y secuencial. Una de las tecnologías más emblemáticas fueron las **tarjetas perforadas**. Cada tarjeta representaba una única línea de código o instrucción. El orden de la pila de tarjetas definía el flujo del programa.

Esta práctica tiene como objetivo entender los conceptos básicos de la programación (**secuencia**, **sintaxis**, **bucles** y **condiciones**) a través de la creación manual de “programas” usando tarjetas de cartulina.

2 Instrucciones Generales

- Representación de Enteros y Operaciones Bit a Bit:
 - Dado el número entero 0x2A en hex, representarlo en binario de 8 bits.
 - Realizar las operaciones AND, OR y XOR con la máscara 0x3F.

-
- (c) Explicar cómo estas operaciones se utilizan en algoritmos de cifrado como AES para mezclar datos.
2. Representación de Punto Flotante y Precisión
- (a) Convertir el número 12.375 a su representación IEEE 754 de 32 bits.
- (b) Analizar cómo los errores de precisión en cálculos flotantes podrían explotarse en vulnerabilidades como desbordamientos en software crítico.
3. Representación de cadenas
- (a) Tomar la cadena "Hello" en ASCII y representarla en hexadecimal y binario.
- (b) Como se codificaría como ascii z, como cadena fija, como cadena estructurada?
- (c) Considerar la endianness little-endian: ¿cómo se almacenaría "Hello" en memoria en un sistema x86?
- (d) Explicar cómo esta representación es explotada en ataques de inyección de código (ej. buffer overflows).
4. Representación de Fechas y Timestamps
- (a) Convertir la fecha "2023-10-05 12:00:00".
- (b) Explicar cómo los timestamps son utilizados en autenticación (ej. tokens TOTP) y cómo un ataque de replay podría manipularlos.
5. Manipulación de Bytes y Endianness
- (a) Dado el número 0x12345678, representarlo en big-endian y little-endian.
- (b) Convertir el valor little-endian 0x78563412 a big-endian.
- (c) Explicar por qué la endianness es crucial en la forense digital (ej. analizar volcados de memoria).
6. Representación de Datos Encriptados
- (a) Cifrar la cadena "SECRET" usando XOR con la clave 0x41.
- (b) Representar el resultado en hexadecimal y binario.
- (c) Explicar cómo el cifrado XOR se utiliza en malware para ofuscar cadenas.
7. Análisis de Hashes Criptográficos
- (a) Calcular el hash MD5 de la cadena "password" y representarlo en hexadecimal.
- (b) Compararlo con el hash de "passw0rd" y discutir cómo los hashes protegen las contraseñas y son vulnerables a ataques de rainbow tables.
8. Representación de Direcciones de Memoria
- (a) Dado un programa simple en C que declara una variable `int x = 10`, mostrar la dirección de memoria de x en hexadecimal.
- (b) Explicar cómo un desbordamiento de búfer podría sobrescribir esta dirección para ejecutar código arbitrario.