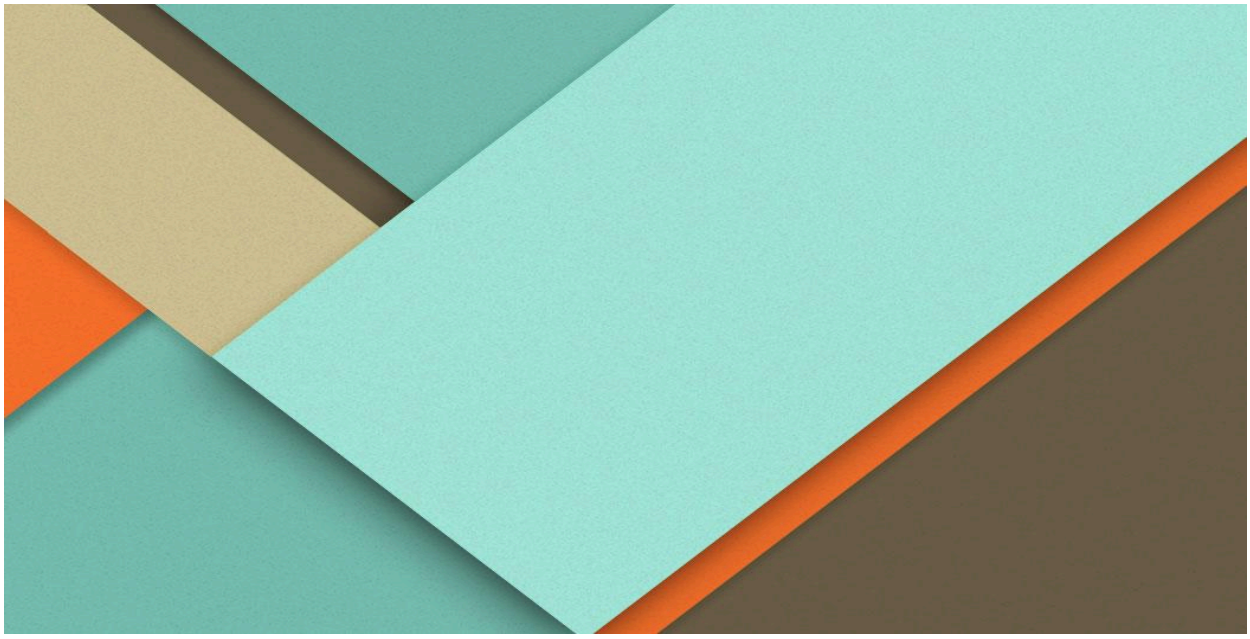

Algoritmo de Desencriptación AES-128 en Ensamblador ARM64

Proyecto 2: MANUAL DE USUARIO

—

Arquitetura de computadores y ensambladores 1 Grupo #12

Michael Cristian Itzep Ixcayau 201610975

Luigi Anderson López Suy 202104727

Fernando Andhré González Espinoza 202202055

Vasti Abigail González Pereira 202300652

28/12/2025

Vac. Dic. 2do Semestre

1. Introducción

Este manual describe el uso de la aplicación AES-128 Decrypt, desarrollada en lenguaje ensamblador ARM64, cuyo propósito es realizar el descifrado de bloques de 128 bits utilizando el algoritmo criptográfico simétrico AES-128.

La aplicación permite al usuario ingresar una clave de ronda 10 y un texto cifrado, procesarlos mediante las rondas inversas del algoritmo AES, y obtener como salida el texto original en formato hexadecimal.

Este programa fue desarrollado como parte del Proyecto 2 del curso Arquitectura de Computadores y Ensambladores 1, de la Universidad de San Carlos de Guatemala

2. Requisitos del Sistema

Para ejecutar correctamente el programa, el usuario debe contar con:

2.1 Software requerido

- Sistema operativo **Linux** (preferentemente Ubuntu).
- Ensamblador **aarch64-linux-gnu-as**.
- Enlazador **aarch64-linux-gnu-ld**.
- Emulador **QEMU ARM64 (qemu-aarch64)**.

2.2 Arquitectura

- Arquitectura anfitriona x86_64, con emulación de ARM64 mediante QEMU.

3. Archivos del Proyecto

El proyecto está compuesto por los siguientes archivos:

inverse_decrypt_final.s

Código principal del algoritmo de desencriptación AES-128

constants_new.s

Tablas constantes (S-Box, Inverse S-Box, Rcon, matriz InvMixColumns)

4. Compilación del Programa

Para compilar el programa, el usuario debe ubicarse en el directorio donde se encuentran los archivos y ejecutar los siguientes comandos:

- **aarch64-linux-gnu-as inverse_decrypt_final.s -o proyecto2.o**
- **aarch64-linux-gnu-ld proyecto2.o -o proyecto2**

5. Ejecución del Programa

La ejecución del programa se realiza utilizando el emulador QEMU:

- **qemu-aarch64 ./proyecto2**

Al ejecutar el programa, el sistema solicitará al usuario el ingreso de dos valores.

6. Uso del Programa

El programa solicita la clave de la ronda 10 del algoritmo AES-128.

Formato requerido:

- 32 caracteres hexadecimales
- Sin espacios
- Sin prefijos (no escribir “0x”)

Ejemplo correcto:

28FDDEF86DA4244ACCC0A4FE3B316F26

```
andhre@andhre-VirtualBox:~/lab_arqui$ qemu-aarch64 ./proyecto2
Ingrese la última clave (ronda 10): 28FDDEF86DA4244ACCC0A4FE3B316F26
 EXPANSIÓN INVERSA DE CLAVES
Clave Ronda 10:
28 6D CC 3B
FD A4 C0 31
DE 24 A4 6F
F8 4A FE 26
```

6.2 Ingreso del texto cifrado

A continuación, el programa solicita el texto cifrado de 128 bits.

Formato requerido:

- 32 caracteres hexadecimales
- Sin espacios

Ejemplo correcto:

29C3505F571420F6402299B31A02D73A

```
Ingrese el texto cifrado (32 hex / 16 bytes): 29C3505F571420F6402299B31A02D73A

CIPHERTEXT cargado (hex 4x4):
29 57 40 1A
C3 14 22 02
50 20 99 D7
5F F6 B3 3A
```

7. Salida del Programa

El programa mostrará el resultado del descifrado en formato matriz 4x4 hexadecimal, correspondiente al estado final del algoritmo AES.

7.1 Ejemplo de salida

```
PLAINTEXT (hex 4x4):
54 4F 4E 20
77 6E 69 54
6F 65 6E 77
20 20 65 6F
```

Este resultado corresponde al texto original:

Two One Nine Two

8. Funcionamiento Interno (Resumen)

El programa implementa la estructura completa del algoritmo AES-128 de desencriptación descrita en el enunciado del proyecto

8.1 Ronda Inicial (Ronda 10)

- AddRoundKey con la última subclave.

```
Ingrese el texto cifrado (32 hex / 16 bytes): 29C3505F571420F6402299B31A02D73A

CIPHERTEXT cargado (hex 4x4):
29 57 40 1A
C3 14 22 02
50 20 99 D7
5F F6 B3 3A


--- RONDA 10 despues AddRoundKey

ESTADO (hex 4x4):
01 3A 8C 21
3E B0 E2 33
8E 04 3D B8
A7 BC 4D 1C
```

8.2 Rondas Intermedias (Rondas 9 a 1)

Para cada ronda:

- InvShiftRows

- InvSubBytes
- AddRoundKey
- InvMixColumns

```
--- RONDA 9 despues InvShiftRows

ESTADO (hex 4x4):
01 3A 8C 21
33 3E B0 E2
3D B8 8E 04
BC 4D 1C A7

 despues InvSubBytes

ESTADO (hex 4x4):
09 A2 F0 7B
66 D1 FC 3B
8B 9A E6 30
78 65 C4 89

 despues AddRoundKey

ESTADO (hex 4x4):
B6 E7 51 8C
84 88 98 CA
34 60 66 FB
E8 D7 70 51

 despues InvMixColumns

ESTADO (hex 4x4):
33 51 79 0A
8B 66 8F 3F
76 7D EB BE
20 92 C2 67
```

8.3 Ronda Final (Ronda 0)

- InvShiftRows
- InvSubBytes
- AddRoundKey

```
--- RONDA 0 despues InvShiftRows

ESTADO (hex 4x4):
63 EB 9F A0
C0 2F 93 92
AB 30 AF C7
20 CB 2B A2

 despues InvSubBytes

ESTADO (hex 4x4):
00 3C 6E 47
1F 4E 22 74
0E 08 1B 31
54 59 0B 1A

 despues AddRoundKey

ESTADO (hex 4x4):
54 4F 4E 20
77 6E 69 54
6F 65 6E 77
20 20 65 6F


PLAINTEXT (hex 4x4):
54 4F 4E 20
77 6E 69 54
6F 65 6E 77
20 20 65 6F
```

9. Consideraciones Importantes

- El algoritmo opera a nivel de bytes, utilizando aritmética del campo de Galois GF(2^8).
- Las subclaves se generan mediante Key Expansion estándar, pero se utilizan en orden inverso.
- El programa sigue el formato column-major para representar la matriz de estado.
- No se utiliza ninguna librería externa; todo el procesamiento se realiza en ensamblador ARM64.

10. Conclusión

Este programa demuestra la correcta implementación del algoritmo AES-128 de desencriptación en un lenguaje de bajo nivel, cumpliendo con los requerimientos técnicos y académicos del proyecto. La aplicación valida la viabilidad del uso de ensamblador ARM64 para algoritmos criptográficos complejos, destacando el control directo sobre memoria, registros y operaciones matemáticas.