Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
(Electronic Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores
(Licentiate Degree Program in Computer Engineering)

Curso: CE-4301 Arquitectura de Computadores I
(Course: CE-4301 Computer Architecture I)

Atributos
Proyecto 1: Decodificador de imágenes, documento de diseño
(Project 1: Image decoder, design document)

Realizado por:
(Made by:)
Richard Guillén Varela,2015023203

Profesor:
(Professor:)
Jason Leitón Jiménez.

Fecha: Cartago, octubre 18, 2020.
(Date: Cartago, October 18, 2020)

Documento de diseño

Requerimientos del sistema

Se requiere desarrollar un algoritmo de desencriptación RSA en el lenguaje ensamblador el el cual se pueda tomar un par ordenado de números y obtener el valor binario de estos números. El valor binario del número combinado será el número real al cual se le realiza la descripción mediante el algoritmo RSA.

Se requiere utilizar las propiedades de la exponenciación modular para realizar la desencriptación eficiente del número encriptado. La exponenciación modular ayuda a mantener los costos computacionales requeridos al mínimo debido a que simplifica el proceso de exponenciación convencional cuando se obtiene el módulo del número al que se eleva.

Se requiere desarrollar un algoritmo que optimice el funcionamiento del sistema. Este algoritmo es de elección libre para el estudiante, pero debe implementarse en el lenguaje ensamblador.

Se requiere desarrollar un programa que pueda ser de alto nivel y permita la visualización de los valores desencriptados calculados previamente con el algoritmo ensamblador RSA.

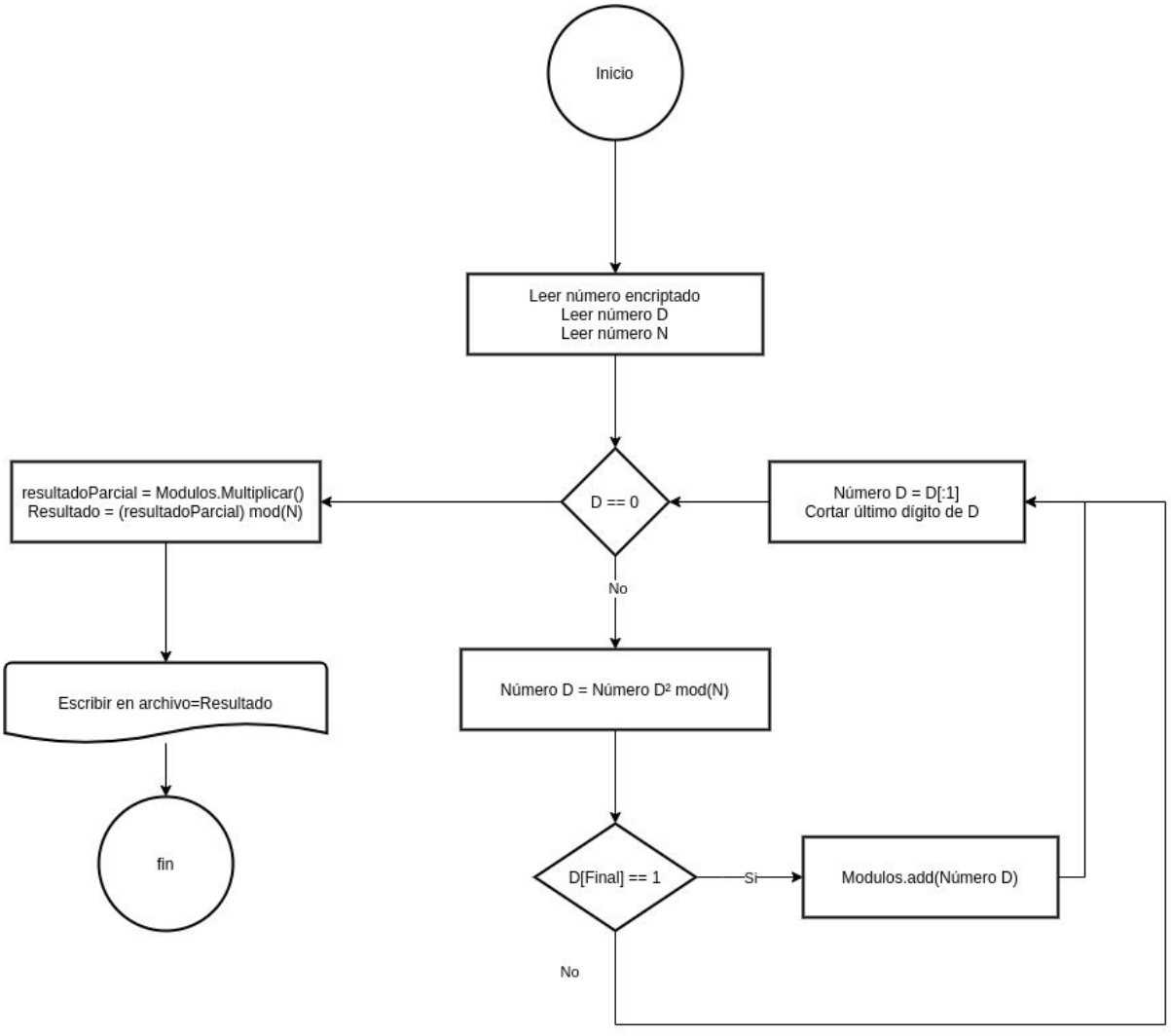
Soluciones del problema

El problema a solucionar es básicamente el siguiente: $m = (c ** d)\ mod(N)$, sin embargo, resolver esta ecuación para valores elevados de ‘C’ y ‘D’ sin ningún tipo de procesamiento diferente a la exponenciación convencional, puede ser bastante costoso de recursos computacionales.

La primera opción de solución es la exponenciación modular simple en la cual se realiza la exponenciación parcial y en potencias de dos, sin embargo, cada vez que se realiza la exponenciación se obtiene el módulo del valor elevado y el próximo valor a elevar es el módulo calculado previamente. Este módulo si el exponente convertido a binario es uno, también se agrega este módulo calculado a una lista para posteriormente multiplicar todos los valores de los módulos calculados en el que el exponente en binario sea un uno y se multiplican. Una vez realizado este proceso de multiplicación se procede a obtener el módulo del valor resultante total y este sería el dato desencriptado.

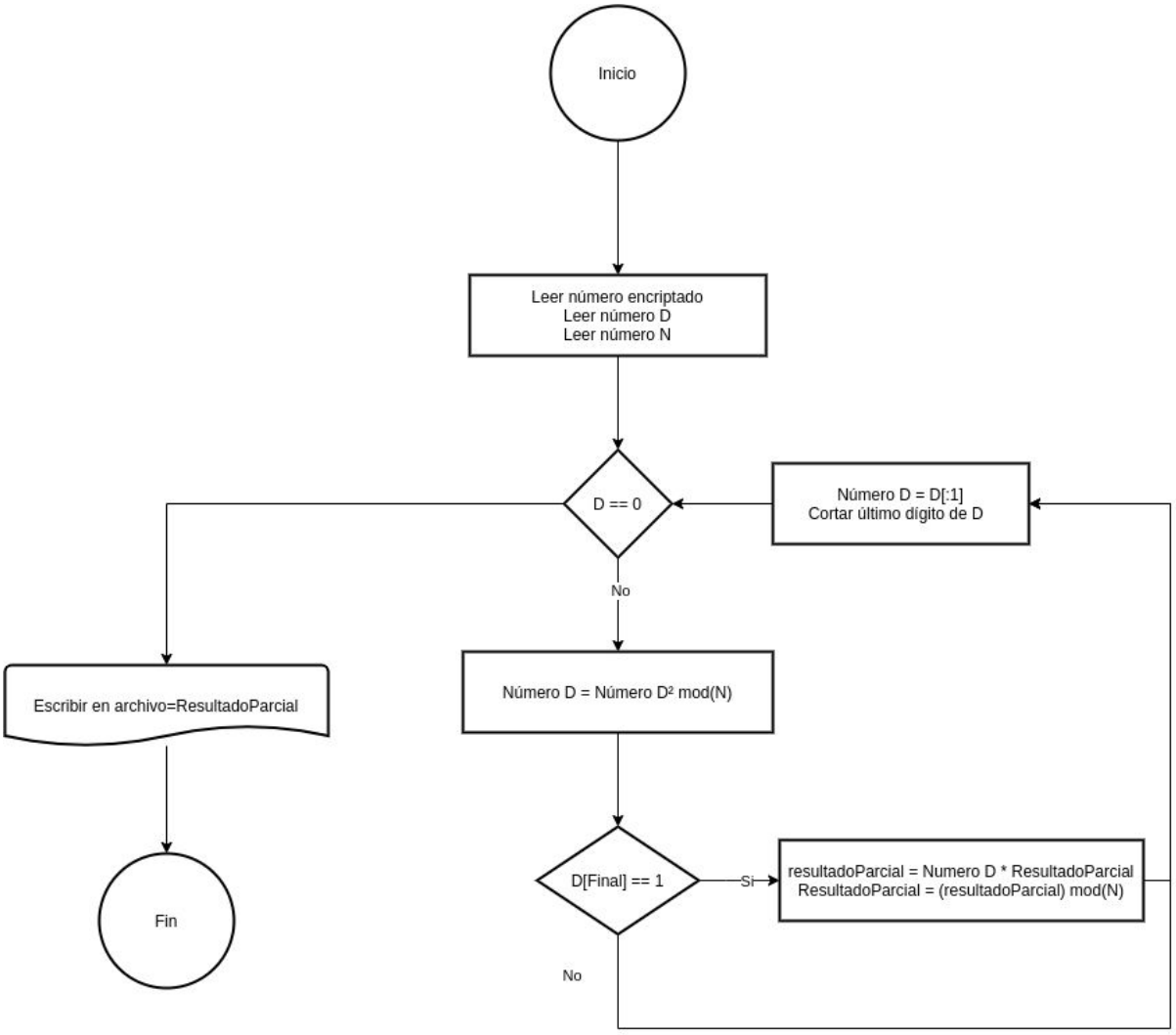
La primera solución se representa en la siguiente imagen.

Imagen 1. Solución 1



La segunda solución planteada vuelve a aplicar nuevamente las propiedades de la exponenciación modular para no plantear la necesidad de una lista o array en el cual se almacene números para posteriormente multiplicarlos, sino que simplemente cada vez que el exponente presenta una potencia de dos se multiplica el dato y se calcula nuevamente el módulo manteniendo datos acordes a la solución, sin utilizar números elevados.

Imagen 2. Solución 2



Comparación opciones de solución

	Ventajas	Desventajas
Solución 1	--Mantiene la simplicidad en el código. --No agrega cálculos parciales	--Necesidad de almacenamiento de datos durante la desencripción. --Datos grandes cuando se realizan las multiplicaciones de los datos parciales.
Solución 2	--Mantiene mayor simplicidad en los cálculos realizados por el procesador.	--Agregar cálculos extra durante el procesamiento de los datos de la imagen.

Selección de la propuesta final

Finalmente se selecciona la segunda opción debido a la simplicidad de los cálculos realizados que aporta debido a que no realiza varias multiplicaciones consecutivas generando con estos errores por overflow. Si se realizan las multiplicaciones parciales, sin embargo al aplicar el módulo cada vez que se realiza la multiplicación se mantienen los datos con un valor máximo en resultados parciales y valor final.

También se selecciona este algoritmo para la simplificación del desarrollo de código en ensamblador. Reduce la cantidad de errores por overflow de datos debido a que los valores se mantienen con un valor máximo alcanzable por lo tanto no se necesitan buscar alternativas como la extensión en la cantidad de bits para almacenar datos más extensos.