# Laboratorio de programación en ensamblador MIPS


UNIVERSIDAD
DE ANTIOQUIA
1 8 0 3

**Estudiantes**

Karol Daniela Alzate Mejía 1192735374
Sergio Josué Rodríguez Taborda 1007398380

**Profesor**

Fredy Alexander Rivera Vélez

Arquitectura de computadores y laboratorio
Universidad de Antioquia
Ingeniería de Sistemas
2020-1

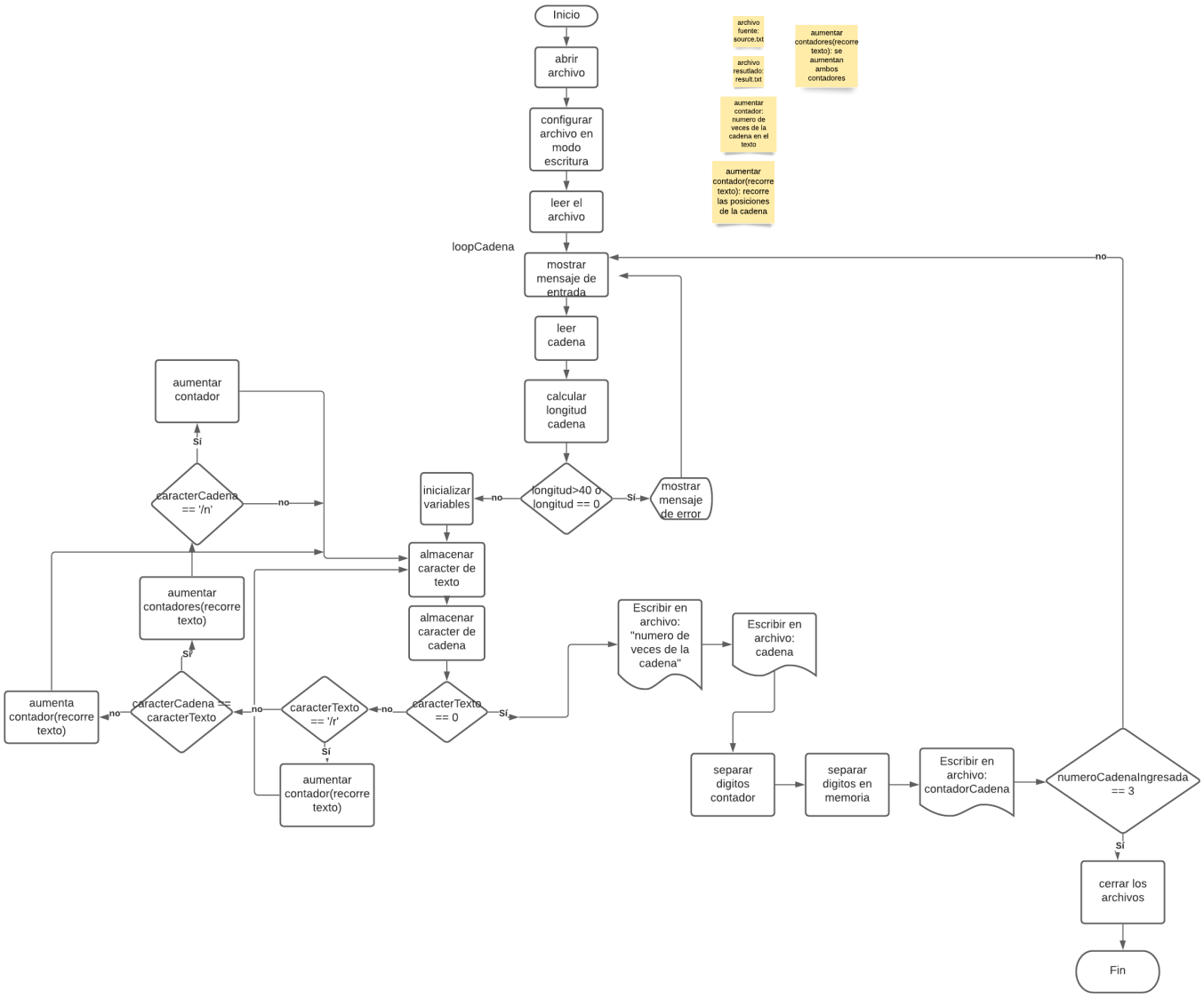
# Introducción

En este informe mostraremos el proceso mediante el cual se resolvió la práctica número 3 del curso. Para resolver esta práctica usamos el IDE MARS 4.5, donde codificamos y simulamos la ejecución del código. Se hizo uso de la arquitectura MIPS y el lenguaje ensamblador para realizar la codificación.

El código desarrollado esta en capacidad de encontrar una cadena de caracteres en un archivo de texto plano y mostrar como resultado el número de veces que aparece esta cadena de caracteres en el archivo.

Además, se mostrará una descripción de alto nivel referente al código desarrollado, la relación de los procedimientos empleados y los análisis y conclusiones obtenidos en el proceso.

# Descripción de alto nivel



## Descripción de entradas, salidas y funcionalidad

**Entradas**

- Cadena de caracteres de máximo de 40
- Archivo: source.txt

**Salidas**

- Archivo: result.txt
  Este archivo muestra un mensaje con los resultados como: el número de veces en las que están las cadenas en el texto.

**Funcionalidad**

- Ya que sólo se realizaba una vez el llamado de estas acciones, no se colocaron los métodos de abrir archivo, crearlo y de leerlo en un método como tal sino que se escribieron al inicio del archivo para que se realizara su ejecución en ese momento.

- Método LongitudCadena: es un bucle, el cual calcula la longitud de la cadena ingresada, retorna el valor de la cadena + 2 (Este valor es asignado para que al momento de realizar la impresión en el archivo se deje un espacio entre la cadena y el número que se imprime después de esta).

- Método NumeroABuffer: reserva 3 bytes para 3 caracteres, separa los dígitos realizando una división por 10 y los almacena en los bytes anteriormente reservados.

- Método TextoVacío: cuando el archivo que se ingresó está vacío, el contador toma un valor de -1. Luego pasa a textoVacío y el espacio de memoria reservado anteriormente lo va llenar con -1.

- Método LoopCadena: este es el método base que realiza el llamado a todas los métodos más influyentes en nuestro código. Se al método InputCadena en el que se ingresa la cadena, luego llama a inicializar que se encarga de generar toda la ejecución del conteo de caracteres en el texto, luego se llama a WriteFile que escribe en el archivo nuevo las líneas

de texto y se dirige a Exit que se encarga de finalizar el programa cuando sea necesario.

- Método InputCadena: recibe la cadena que ingresa el usuario.

- Método MensajeAdvertencia: imprime mensaje de alerta cuando la longitud de la cadena ingresada es mayor a 40 o menor a 0.

- Método Inicializar: se inicializa el contador de las cadenas y se toma la dirección de memoria del texto del archivo y de la cadena ingresada.

- Método Bucle: almacena los valores de los caracteres del texto y de la cadena y con ello realiza diferentes comparaciones para llamar a otros métodos que se encargarán de realizar otras funcionalidades.

- Método avanzar: aumenta el contador encargado de recorrer el texto en 2, ya que cuando se llega al final de la línea se tiene '/r' y '/n' almacenado en memoria entonces, para seguir realizando las comparaciones, se hace el aumento para omitir la lectura de estas dos instrucciones.

- Método indexSubCadena: este método se llama cuando los caracteres de las dos cadenas son semejantes por lo que se encarga de aumentar el contador que recorre ambas cadenas. Esto se realiza con el fin de que, si las cadenas siguen teniendo caracteres parecidos, se llegará a un momento en el que al recorrer la cadena que se ingresó se llegue al final de esta, lo que indicaría que el conjunto de caracteres de la cadena se encuentra en el texto; cuando esta condición se cumple se llama al método de contador.

- Método Contador: cuando se encuentra el carácter '/n' (implica que se llegó al final de la cadena que se ingresó) se aumenta el contador de cadenas en 1.

- Método Fin: redirecciona a LoopCadena con el valor final del contador de cadenas.

- Método Exit: se encarga de cerrar los dos archivos y de finalizar el programa.

## Explicación y justificación de las decisiones de diseño

- Al principio se había creado un método para realizar el proceso de conteo de cadenas por cada una de las cadenas que se estaba ingresando, pero luego se notó que por medio de un contador se podía lograr que se creara sólo un método (LoopCadena), el cual tendría una condición de parada cuando el contador fuera igual a 3 que es el que indicaría que se habían ingresado las 3 cadenas.

- Para la definición de la práctica se decidió utilizar las variables temporales para aquellos valores que estaban cambiando constantemente, estos sólo se necesitaban en momentos específicos por lo que cuando ya se había realizado su funcionalidad en el código, se reutilizaban estas variables en otros métodos. En los argumentos se almacenaban las direcciones de memoria de los elementos o se almacenaban los argumentos de las funciones. En los registros guardados se almacenaban en general los valores que se necesitaban preservar en general en toda la codificación, como por ejemplo, la dirección donde quedaba el nuevo archivo.

- Al momento de pasar el número que daba como resultado del contador de cadenas al archivo se presentaba un inconveniente dado que el valor que se estaba mandando era en realidad el un código ASCII, no el número como tal por lo que se decidió realizar el siguiente procedimiento.

    - Se reservó un espacio de 3 bytes para 3 caracteres en memoria.
    - En el primer byte de derecha a izquierda se almacenó un salto de línea para mayor legibilidad a la hora de agregar el número al archivo.
    - Luego, se procedió a hacer una división por 10 del número resultante del contador de cadenas para tomar por separado cada uno de sus dígitos.
    - *Importante:* esto genera una restricción en el código en la que no se podrán contar las cadenas en un texto si esta aparece más de 99 veces por lo que sólo se están tomando dos dígitos del contador.
    - Luego de realizar esta separación se buscó la representación del número 0 en el código ASCII que es el número 48 y se procedió a sumar el dígito a este número para encontrar su equivalente en código ASCII y así poder ser mostrado en el nuevo archivo.

- Para mayor comprensión el archivo de texto nuevo, se le agregó a este una cadena que contenía: “Número de veces de la cadena ” y se procedió a calcular la longitud de la cadena ingresada con el fin de imprimirla también en el archivo para mayor comprensión, esto se realizó porque cuando se iba a imprimir la cadena en el archivo este pedía que se diera el número de caracteres que se iba a imprimir; si se ponía un valor fijo y la cadena era menor que ese valor, podían aparecer también otros textos en el archivo dado que la cadena no estaba ocupando todo ese espacio, entonces por eso se procedió a calcular la longitud de la cadena en esta situación.

- El límite de la longitud de la cadena que se puede ingresar es de 40 caracteres. Para esto se realizó una validación en la que, al momento de ingresar la cadena que quería ser contada en el archivo, se calculaba la longitud de la cadena y si esta no cumplía con la condición debía ingresar nuevamente la cadena, también se le agregó una en esta misma sección que no permite la entrada de una cadena nula.

- El archivo que lee tiene un máximo de 10000 caracteres, ya que se considero un valor prudente para el ejercicio.

- Se creó una condición en la que, si el archivo que se había ingresado para leer el texto estaba completamente vacío, se le asigna un valor de -1 al contador de cadenas que indique que se está dando este caso en particular.

## Análisis y conclusiones

- Es importante que antes de iniciar a escribir el código, tengamos previamente pensado el proceso que vamos a realizar para evitar problemas futuros.
- La consulta constante de la documentación del IDE ayuda a elegir mejores métodos para la utilización de alguna funcionalidad que sea necesaria al momento de la implementación.
- Es importante tener una buena organización para que no se complique el entender el código.
- Debemos realizar una buena documentación para dejar las instrucciones de lo que estamos realizando, ya que es posible que se nos olvide lo que hemos hecho o también para que otra persona entienda el proceso.
- Si se realiza un buen análisis de los procesos que se manejan en el código podemos lograr la reutilización de diferentes métodos con el fin de que no se repita código de manera innecesaria.