Departamento de Ingeniería de Sistemas

Arquitectura de Computadores y Laboratorio

Laboratorio No. 2

Arquitectura MIPS

2025 – 1

**Escrito por Mariana Gallego Rodas y Leonardo Jose Amaris Dominguez**

**Enlace a video explicativo:**

Informe escrito

Arquitectura MIPS

Para comenzar con la descripción de este laboratorio es importante tener en cuenta que el código escrito para su desarrollo se escribió implementando una programación estructurada, es decir, se basa en el uso de funciones que realizan una tarea específica.

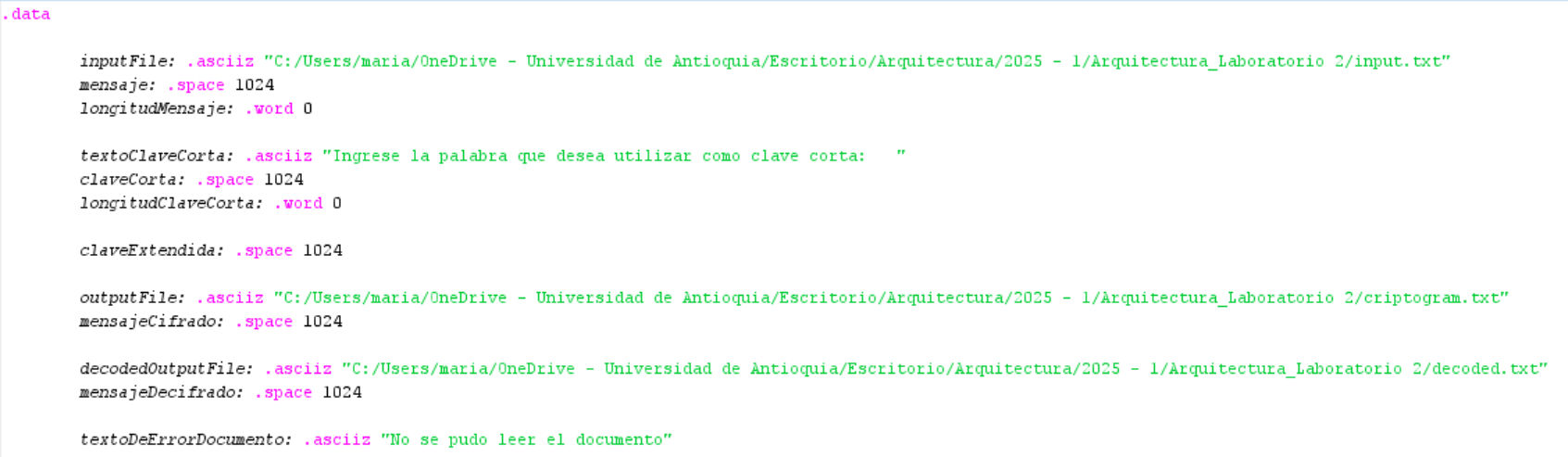
Primero se hará una descripción detallada de como está escrito es código y qué tarea desarrolla cada función y por último, se mostrarán dos pruebas de escritorio que ejemplificarán como se realizó la encriptación y desencriptación del mensaje de entrada.

**Detalle del código:**

En lenguaje MIPS, el código se divide en dos partes: *.data* y *.text*

**.data:** es donde se guarda la información global, las constantes y las cadenas de texto que se utilizarán a lo largo del programa. En el desarrollo de este laboratorio se definieron dentro de esta sección las siguientes constantes:

* **inputFile, outputFile, decodedOutputFile**: contienen la ruta a los documentos a los que tendrá acceso el programa
* **mensaje**: un buffer con un espacio específico de 1024 bytes en donde se almacenará temporalmente el mensaje que el sistema leerá del archivo inputFile o outputFile
* **claveCorta y claveExtendida**: dos buffers con un espacio específico de 1024 bytes cada uno, en donde se almacenará la clave corta y la clave extendida respectivamente
* **longitudMensaje y longitudClaveCorta**: dos buffers que almacenaran la cantidad de caracteres que tiene el mensaje y la clave corta que ingresó el usuario
* **mensajeCifrado y mensajeDecifrado**: dos buffers con un espacio específico de 1024 bytes cada uno, en donde se almacenará el mensaje cifrado y el decifrado respectivamente
* **textoClaveCorta y textoDeErrorDocumento:** almacenan los textos que se le mostrarán al usuario para solicitarle que ingrese la clave corta o para indicarle que no se pudo abrir el documento



**.text:** es donde se definen todas las instrucciones que ejecutará el programa, las funciones, las condiciones, etc. Para el desarrollo de este laboratorio esta sección se organizó en tres partes con el fin de obtener un código mucho más ordenado, la descripción de estas secciones se detalla a continuación:

**Parte 1 - main:** tal como en los lenguajes de alto nivel, es la primera función que se ejecuta por defecto. Para un mejor entendimiento, esta sección se divide en dos partes, cifrado y decifrado, en cada una se estarán invocando las funciones que se necesitan para cumplir con el objetivo del laboratorio.

Desde la parte de cifrado se llaman las siguientes funciones: ***leerMensaje*** a la que a través de $a3 se le pasa como argumento la dirección de espacio de memoria en donde está almacenada la ruta de acceso al documento que se quiere leer; ***leerClaveCorta*** con la que se le pide al usuario que ingrese la clave que se utilizará para el cifrado; ***crearClaveExtendida*** que se encarga de copiar la clave corta y completar con los caracteres del mensaje; ***cifrarMensaje*** que se encarga del cifrado del mensaje usando la técnica de cifrado Autoclave Vigenere, y ***escribir*** que escribe el mensaje cifrado o descifrado, dependiendo de cual se le pasa en el argumento $a1, en el documento que se le especifica a través del argumento que se le pasa con $a3

Desde la parte de descifrado, se llaman las funciones ***leerMensaje*** y ***escribir***, y también la función ***decifrarMensaje*** que se encarga del decifrado del mensaje usando la técnica de decifrado Autoclave Vigenere

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Parte 2 - Definición de funciones:** contiene la definición de todas las funciones que se utilizarán a lo largo de la ejecución del programa para obtener la solución. A continuación se dará una descripción de las operaciones que realiza cada una:

* **leerMensaje:** cuando se invoca esta función se le debe pasar un argumento, a través del registro **$a3**, que le indicará cual es el archivo de texto del que debe extraer el mensaje.

****

Cuando comienza la ejecución, abre el archivo que se le indicó a través del registro $a3 utilizando el *Syscall 13*, que le indica al programa que se abrirá un archivo, a $a0 se le pasa el valor de $a3, es decir, la dirección de la cadena que contiene el nombre del archivo, el valor cero para $a1 indica que el archivo se abrirá en modo de solo lectura. Al final, a $v0 se le asigna el valor -1 si el archivo no se pudo abrir, luego ese valor se almacena en $t0 y se utiliza bltz (Branch if les than zero) para llamar al error handler **errorHandlerDocumento** en caso de que sea -1; si $v0 es distinto de -1, entonces se continua con la ejecución.

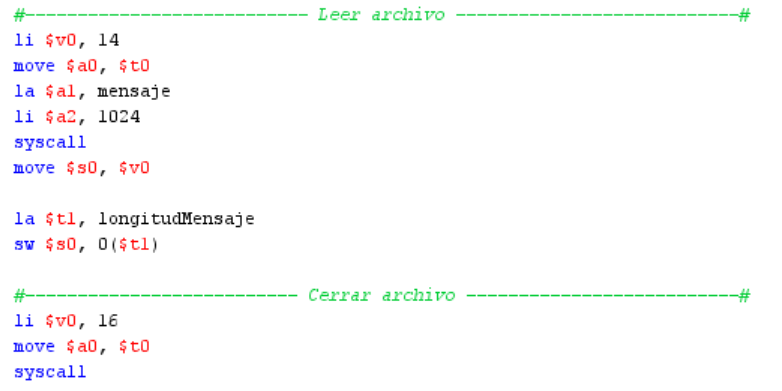
Texto

Descripción generada automáticamente con confianza baja

El segundo paso de esta función es leer el archivo que ya se pudo abrir utilizando el *Syscall 14*, que se utiliza para indicar que se leerán datos, luego el valor de $t0, que se obtuvo en el paso anterior, se le pasa al registro $a0, ese es el descriptor de archivo del cual se van a leer los datos; a $a1 se le pasa la dirección de memoria en donde se almacenarán los datos leídos, esto se definió en la sección *.data* en *mensaje* y a $a2 se le pasa el número máximo de bytes que se podrán leer.

Cuando termina de leer el archivo, se mueve el contenido de $v0, que contiene el total de bytes leídos, a $s0, para después utilizar un StoreWord para almacenar ese número en el espacio de memoria que se le reservó llamado longitudMensaje.

Al finalizar, se utiliza el Syscall 16 para cerrar el archivo.



* **leerClaveCorta:** esta función se encarga de mostrar el mensaje al usuario para pedirle que ingrese la clave que se va a usar para encriptar el mensaje leído desde el documento.

Utiliza el *Syscall 4* para mostrar el mensaje guardados en textoClaveCorta que le pide al usuario que digite la clave que quiere usar para encriptar y el Syscall 8 para leer la palabra clave. Después llama a la función ajustarClaveCorta que se encarga de que al presionar enter, este no se guarde como un carácter; más adelante se detallará sobre esta función.

Por último, se utiliza un ciclo para contar la cantidad de caracteres que quedaron almacenados en el buffer claveCorta y esta cantidad se almacena en el espacio de memoria reservado llamado longitudClaveCorta.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.Texto

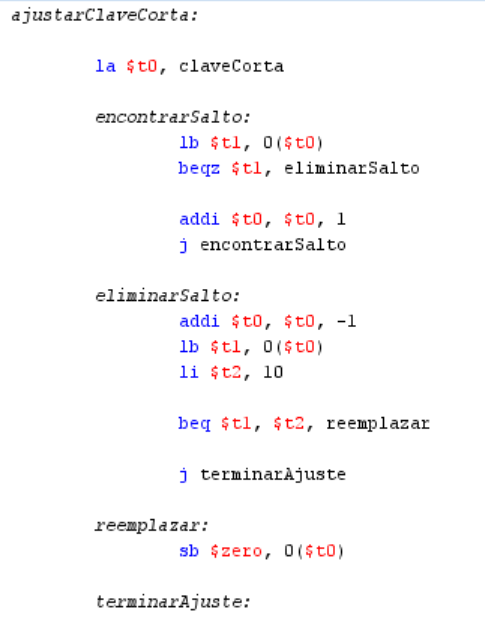
El contenido generado por IA puede ser incorrecto.

* + - * **ajustarClaveCorta**: como el usuario debe presionar enter cuando termine de digitar la clave para que la ejecución pueda continuar, esta acción se guarda como un salto de linea, o sea, como un carácter más, entonces esta función se encarga de eliminar ese salto de línea del total de caracteres que componen la clave corta.

Para hacer esto, se creó un ciclo llamado encontrarSalto, que recorre el buffer en donde está almacenada la clave corta hasta que llega al carácter nulo; cuando esto pasa, se hace un salto a eliminarSalto en donde se retrocede una posición ya que si se está en el carácter nulo, entonces el salto de línea está en la posición anterior.

Cuando ya se está en la posición del salto de línea, se carga el valor 10 en $t2, que es el código ASCII del salto de línea; luego se comparan $t1 que contiene el byte anterior al nulo y $t2 y si son iguales se hace un salto a reemplazar.

En reemplazar se almacena el valor cero en la posición donde está el salto de línea y finaliza el código de la función.



* + - * **crearClaveExtendida:** esta función se encarga de copiar, dentro de en buffer claveExtendida, la clave corta y de rellenar el resto de espacios disponibles con caracteres del mensaje que se va a cifrar.

Para esto utiliza utiliza los buffers longitudMensaje, longitudClaveCorta, claveCorta, mensaje y claveExtendida.

Primero llama a la función copiarClaveCorta y luego a la función completarClaveExtendida. El procedimiento de ambas se detallará más adelante.

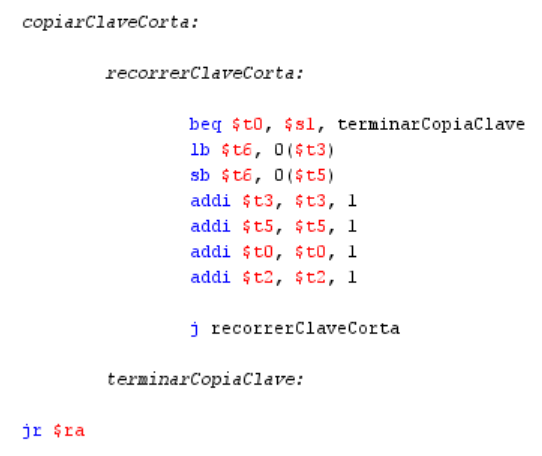
Tabla

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

* + - * **copiarClaveCorta**: esta función utiliza un ciclo para recorrer el buffer que almacena la clave corta y copiar cada carácter que encuentre en el buffer que guarda la clave extendida.

Cuando comienza el ciclo, se comparan $t0 y $s1, que contienen la cantidad de caracteres recorridos y la longitud de la clave corta, respectivamente. Cuando los dos registros son iguales, entonces significa que ya se recorrieron y copiaron todos los caracteres de la clave corta y se termina la función.

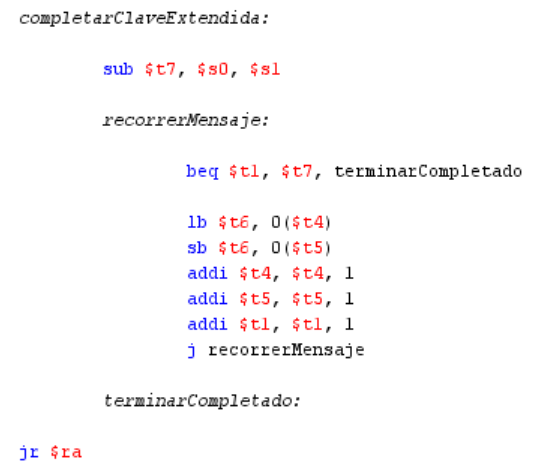


* + - * **completarClaveExtendida:** esta función se encarga de rellenar la clave extendida con caracteres del mensaje hasta que la longitud sea correcta.

Primero resta $s0 y $s1, la longitud del mensaje y la longitud de la clave corta respectivamente, para así saber cuantos caracteres del mensaje hacen falta para completar la clave extendida. El resultado de esa resta se almacena en $t7.

Después utiliza un ciclo que al principio compara $t1, que es un contador y $t7 para saber si ya se tomaron los caracteres necesarios para completar. Si no es así, entonces se procede a recorrer el buffer del mensaje y a almacenar cada carácter encontrado en el buffer de la clave extendida. Gracias a que en la función copiarClaveCorta se almacenó en $t5 la posición en la que está el último carácter en la clave extendida, entonces en esta función $t5 se encarga de indicar a partir de donde se comienzan a almacenar los caracteres a la clave extendida.

El ciclo y la función terminan cuando $t1 y $t7 son iguales.



* + - * **escribir**: recibe los parámetros $a1 y $a3, que son la ruta de un archivo de texto y la dirección de un buffer respectivamente. Su función en escribir dentro del archivo de texto los caracteres almacenados en el buffer.

El primer paso es abrir el documento en el cual se van a escribir los caracteres, para esto utiliza el *Syscall 13*, y después se comprueba si se abrió correctamente. La descripción de estas dos líneas ya se detalló en procesos anteriores con la única diferencia de que al registro $a1 se le asigna el valor 1 para indicar que el archivo se va a abrir en modo de escritura.

Después, se utiliza el *Syscall 15* para escribir en el archivo de texto y se mueve el valor de $s0, que es el que se obtuvo en el Syscall 13, a $a0; en $a1 se carga el buffer donde están los caracteres y en $a2 se almacena la cantidad de caracteres que se van a escribir.

El último paso es cerrar el archivo utilizando el *Syscall 16*, y con eso termina la ejecución de esta función.

