**Organización del Computador**

**TP - ARM**

**Propósito:**

En este trabajo se van a desarrollar y poner en práctica los conceptos de arquitectura ARM que se ven durante la segunda parte de la materia después del primer parcial. En particular se presta atención a los siguientes conceptos:

* Datos almacenados en registros, pila, memoria
* Modos de direccionamiento
* Llamada a procedimientos del usuario e interrupciones del sistema

Estos puntos se ponen en práctica en el contexto de un juego de consola o terminal. Este contexto también permitirá implementar algunos conceptos vistos durante la primera parte de la materia, por ejemplo:

* Codificación de caracteres ASCII
* Conversión entre bases Decimal -> Binario, Binario -> Decimal
* Operaciones en Complemento A2

**Producto final:**

Al finalizar este trabajo tendremos un juego de consola escrito en lenguaje ensamblador ARM. Además tendremos una serie de procedimientos que nos permitirán implementar otros juegos y programas.

**Evaluación:**

Para acreditar y aprobar esta actividad se solicita:

* Un archivo en formato PDF conteniendo un informe del trabajo realizado y las dificultades encontradas. Incluir un pseudocódigo de su juego con su explicación correspondiente. No pegar código en su informe. El nombre del archivo debe ser **TP\_Final\_Orga\_NumeroDeGrupo.pdf**
* También incluir el código fuente de su implementación pero no incluir ejecutables.

Esta actividad es grupal, obligatoria y será calificada con la siguiente escala:

* A+ (Aprobado más) = 10/10 puntos
* A (aprobado) = 6 a 9/10 puntos
* A- (aprobado menos) = 4 a 5/10 puntos
* I (insuficiente) = < 4/10 puntos

La nota de este trabajo se promedia con los parciales.

Si obtienen en el trabajo una I (insuficiente), se solicitará al alumno que re-entregue los ejercicios con las correcciones pertinentes pero no se calificará con A+ en instancia de recuperatorio.

Las herramientas disponibles para realizar el TP son las siguientes:

* Emulador de Raspbian y procesador ARM ([instrucciones para instalar](http://24.232.104.248/doku.php?id=arm_qemu))
* Acceso remoto a una placa Raspberry real (solicitar acceso a su docente, [instrucciones para acceder de manera remota](https://docs.google.com/document/d/1Sr5b_6Jyavk2AMitB_Tf83RGspvgSKRup0RMsEJN-Ts/edit?usp=sharing))

**Fecha de entrega:** 15/6

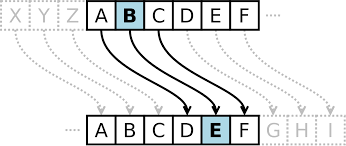
**Modo de entrega:** Moodle

**Cifrado César - ¡Enviando mensajes secretos!**

El método de cifrado César fue usado en los tiempos del emperador Cayo Julio Cesar (100 - 44 AC) para enviar mensajes secretos a sus generales en los confines del imperio romano.

|  |
| --- |
| **hola este mensaje esta en texto plano sin encriptar**  **firma pepito** |

El método consiste en hacer un corrimiento de las letras, por ejemplo, la letra A se convierte en la letra D, la letra B se convierte en la letra E, así como muestra la siguiente figura:



El resultado de este proceso de aplicar shift a cada una de las letras de un mensaje es un nuevo mensaje que a simple vista no se puede entender.



|  |
| --- |
| **krod hvwh phqvdmh hvqd hq whawr sodqr vlq hqfulswdu**  **ilupd shslwr** |

Si bien el cifrado César es un método muy básico y fácil de quebrar en comparación con otros algoritmos de cifrado de la actualidad, este mecanismo de cifrar los mensajes fue muy efectivo en su tiempo y ayudó a mantener en secreto las estrategias militares del César.

Se pide realizar un programa que reciba como input un mensaje normal (que se desea enviar en secreto) y devuelva el mismo mensaje pero cifrado (aplicando el algoritmo de cifrado César). Además se va a agregar un chequeo de paridad par al final de la cadena que tomará el valor de 0 o 1 dependiendo de la paridad de la cadena resultante.

Mi programa

texto normal

texto cifrado +bit de paridad

También pedimos que el mismo programa se pueda usar para descifrar un código que nos envía un amigo por ejemplo. En este caso el programa recibe un texto cifrado y devuelve el texto original:

Mi programa

texto cifrado +bit de paridad

texto original

Se debe tener en cuenta al descifrarlo que la paridad debe ser correcta, en caso de que no lo sea deberá indicarlo con un mensaje de error.

**Etapa 1**

Se pide que el programa reciba un solo string con 3 datos:

* el mensaje,
* la clave (c/d) para codificar/decodificar (tener en cuenta que para decodificar los mensajes se debe incluir el bit de paridad al final del mensaje como si fuera un carácter más)
* y una opción que sirve para determinar si se codifica o se decodifica el mensaje usando la clave proporcionada.

Gráficamente el input debe ser el siguiente string:

***“Mensaje; clave; opción;”***

Cada parte del string debe estar separada de la otra con un separador dado por el carácter ;

Puede asumir que los mensajes no tendrán en su contenido símbolos de puntuación. No se debe codificar los espacios.

**Ejemplo de input**

Por ejemplo un input puede ser el siguiente:

***“hola este mensaje esta en texto plano sin encriptar;3;c;”***

Donde

* el mensaje es “hola este mensaje esta en texto plano sin encriptar”
* la clave es 3
* y la opción es c (codificar)

La clave se usa para saber cuánto debe desplazarse cada letra del mensaje (no se deben cifrar/descifrar los espacios)

Para el input de este ejemplo el programa debe imprimir por pantalla el mensaje encriptado:

***krod hvwh phqvdmh hvqd hq whawr sodqr vlq hqfulswdu***

Adicionalmente, el programa debe imprimir la cantidad de letras que fueron procesadas. Por ejemplo, con el input anterior se debe imprimir, además del mensaje codificado, en una nueva línea con el mensaje: Se procesaron 51 caracteres

**Etapa 2**

A partir de un mensaje cifrado, donde no conozco la clave, pero si tengo una pista. Se pide decodificarlo.

Se debe ingresar un string, compuesto por el texto codificado y una palabra clave, en base a esa palabra clave se determina el desplazamiento utilizado.

Luego se decodifica el mensaje. Se debe mostrar por pantalla el mensaje decodificado y el desplazamiento usado.

Ejemplo

String ingresado: mtqf jxyj rjsxfoj jxyÑ jshwnuyfit; este

Texto codificado: mtqf jxyj rjsxfoj jxyÑ jshwnuyfit

Palabra clave: este

Para este input el programa debe imprimir por pantalla:

este mensaje está encriptado

desplazamiento usado = -5

Mensaje: “Hola este mensaje está encriptado”

**Pseudocódigo**

Para la **Etapa 1**, la lógica general del programa de se puede describir con el siguiente pseudocódigo

|  |
| --- |
| main()  {  inputUsuario= leer input del usuario  mensaje= extraer mensaje de inputUsuario  clave= extraer clave de inputUsuario  opcion= extraer opción de inputUsuario    if opcion==’c’  output= codificar el mensaje  if opcion==’d’  output= decodificar el mensaje    Imprimir output  Imprimir cantidad  } |

Para implementar este pseudocódigo se sugiere implementar las siguientes subrutinas que ayudarán a repartir el trabajo entre los miembros del grupo y facilitarán la tarea de debugging:

Subrutinas principales

|  |  |
| --- | --- |
| extraer\_mensaje | Debe recorrer el string del input del usuario y devolver solo el mensaje, sin la clave ni la opción |
| extraer\_clave | Debe recorrer el string del input del usuario y devolver la clave que será usada para codificar/decodificar |
| extraer\_opcion | Debe recorrer el string del input del usuario y devolver la opción |
| codificar | Debe aplicar al mensaje el cifrado César usando la clave extraída en pasos anteriores |
| decodificar | Debe aplicar al mensaje el cifrado César pero a la inversa, para extraer el mensaje original. Prestar atención al bit de paridad |

Subrutinas secundarias

|  |  |
| --- | --- |
| convertir\_ascii\_a\_entero | Para convertir la clave a un entero. Recordar que el input es un string y está codificado en ascii, por lo tanto si se ingresa una clave 3, por ejemplo, hay que convertir el ascii 3 al entero 3 |
| converir\_entero\_a\_ascii | Para poder imprimir por pantalla la cantidad de letras procesadas |

Para la **Etapa 2,** elaborar el algoritmo necesario para resolver la situación, antes de codificar en assembler, puede ser en pseudocódigo o en un lenguaje de alto nivel. Se sugiere pensar en las acciones necesarias.

**Comentarios generales**

Es importante definir previamente cuál es el alfabeto que será usado para los mensajes y cómo se comporta el cifrado en los bordes. Por ejemplo, pueden limitarse a las letras mayúsculas de ASCII, y que sea módulo Z, así en caso de pasarse siempre tienen otra letra del mismo alfabeto. En caso de elegir otra codificación y otro alfabeto debe consultar previamente con sus docentes antes de empezar a implementar.

Finalmente, es importante que el texto se lea desde la entrada standard, así podrían codificar o decodificar mensajes escritos directamente en la terminal. Para poder realizar esta interacción con la terminal pueden tomar como ejemplo el código que está en el siguiente anexo:

