ALEJANDRO REYES ALBILLAR 45931406-S ara65@alu.ua.es

PRÁCTICA 4.1 ESTRUCTURAS DE LOS COMPUTADORES

1º) La práctica nos pide que implementemos con la ayuda de los apuntes y lo aprendido en clase el algoritmo de codificación TEA (Tiny Encryption Algorithm). Para poder utilizar dicho algoritmo de codificación es necesaria una clave de codificación que posee el que codifica el mensaje y que debe conocer el que quiere recibir el mensaje para poder leerlo, de ese modo el mensaje está seguro de todas las personas ajenas a él.

Para implementar dicho algoritmo nos hemos servido de los códigos implementados en C que nos han proporcionado en la práctica:

void CIFRAR (unsigned long\* v, unsigned long\* k) {

unsigned long y=v [0], z=v [1], sum=0, i=0, /\* inicialización \*/

delta=0x9e3779b9, /\* constante necesaria \*/

n=32; /\* número de rondas de cifrado \*/

for (i=0; i< n; i++) { /\* Ciclo básico \*/

sum += delta;

y += ((z<<4)+k[0]) xor (z+sum) xor ((z>>5)+k[1]);

z += ((y<<4)+k[2]) xor (y+sum) xor ((y>>5)+k[3]);

} /\* fin del ciclo \*/

v [0]=y;

v [1]=z;

}

void DESCIFRAR (unsigned long\* v, unsigned long\* k) {

unsigned long y=v [0], z=v [1], sum=0xc6ef3720, i=0, /\* inicialización \*/

delta=0x9e3779b9, /\* constante necesaria \*/

n=1; /\* número de rondas de cifrado \*/

for (i=0; i< n; i++) { /\* Ciclo básico \*/

z -= ((y<<4)+k[2]) xor (y+sum) xor ((y>>5)+k[3]);

y -= ((z<<4)+k[0]) xor (z+sum) xor ((z>>5)+k[1]);

sum -= delta;

} /\* fin del ciclo \*/

v [0]=y;

v [1]=z;

}

Donde v es un vector que almacena una palabra de 8 caracteres en código ASCII, por lo que convierte los caracteres en valores hexadecimales del tipo 0x012345687. Por este motivo, y dado a que una sola celda del vector únicamente puede almacenar 4 de estos caracteres, son necesarias 2 celdas en este vector, que son v[0] y v[1].

Además la clave de codificación que hemos llamado k, consta también de un vector aunque, en este caso, es de 4 celdas numeradas del 0 al 3.

Por último, es necesaria una constante llamada delta que posee el valor hexadecimal 0x9e3779b9 y que es utilizada también en el proceso de codificación/descodificación.

Para poder implementar esta práctica hemos utilizado el ensamblador MARS basado en MIPS.

En él hemos implementado el código de un menú para escoger la opción de cifrado, descifrado, cambio de clave de cifrado y salida con sus consiguientes códigos de codificación/decodificación/cambio que se detallan a continuación:

.data 0x10010000

v: .word 0x00000000, 0x00000000 #vector v de 2 componentes de 32 bits a cifrar

.data 0x10010020

k: .word 0x11111111, 0x22222222, 0x33333333, 0x44444444 #clave de cifrado de 128 bits, 4 vectores de 32 bits

.data 0x10010040

delta: .word 0x9e3779b9

.align 2

opcion: .word

.align 2

.data 0x10001001

menu: .asciiz "\nMENU PRINCIPAL \n 1.Cifrar un texto.\* \n 2.Descifrar un texto.\* \n 3.Modificar la clave de cifrado. \n 4.Salir del programa. \n \*La operacion finaliza el programa. \nIntroduzcca opcion(1-4):"

.align 2

merror: .asciiz "HA SELECCIONADO UNA OPCION INCORRECTA"

.align 2

introducido:.asciiz "\nIntroduce el texto:"

.align 2

cifrado:.asciiz "\nTexto cifrado:"

.align 2

descifrado:.asciiz "\nTexto descifrado:"

.align 2

nuevaclave:.asciiz "\nIntroduce la nueva clave de cifrado:"

.align 2

cambiada: .asciiz "\nLa nueva clave de cifrado es:"

.align 2

.text

main: li $t0, 32 #$t0=32, iterador n

li $t1, 1 #$t1=1, iterador i

li $t2, 0 #$t2=0, sum(codificado)

lw $t3, delta #$t3=delta

li $t4, 0xc6ef3720 #St4= valor hexadecimal, sum(decodificado)

la $s0, v #$s0=v carga el vector entero

la $s1, k #$s1=k carga el vector entro

lw $t5, 0($s0)#$t5=v[0]

lw $t6, 4($s0) #$t6=v[1]

lw $t7, 0($s1) #$t7=k[0]

lw $t8, 4($s1) #$t8=k[1]

lw $t9, 8($s1)#$t9=k[2]

lw $s0, 12($s1) #$s0=k[3] como ya no necesitamos el valor de v que esta cargado en los registros $t5 y $t6 actualizamos $s0

imprimemenu: li $v0,4 # codigo imprimir cadena

la $a0, menu # imprime el menu

syscall # Llamada al sistema

j pideopcion #salta a pideopcion

pideopcion: li $v0, 5 # codigo de leer entero

syscall # Llamada al sistema

sw $v0, opcion # guarda la opcion elegida en la etiqueta opcion

beq $v0, 1, pidecifrar #Salta a pidecifrar cuando se elige la opcion 1

beq $v0, 2, pidedescifrar #Salta a pidedescifrar cuando se introduce la opcion 2

beq $v0, 3, cambiar #Salta a cambiar cuando se elige la opcion 3

beq $v0, 4, fin #Salta a fin cuando se elige la opcion 4

blez $v0, error # salta a error cuando se elige una opcion menor o igual que 0

bgt $v0,4, error # salta a error cuando se elige una opcion mayor que 3

error:li $v0, 4 # codigo imprimir cadena

la $a0, merror # imprime el contenido de la etiqueta merror

syscall # Llamada al sistema

j imprimemenu # Salto a imprimemenu

pidecifrar:li $v0,4 # codigo imprimir cadena

la $a0, introducido # imprime el menu

syscall # Llamada al sistema

la $a0, v #carga el vector v donde se guardara lo que se quiere guardar

li $a1, 9 #hace que la iteracion guarde un tamanyo de 8 caracteres

li $v0, 8 #codigo leer string

syscall # Llamada al sistema

la $s1, v #$s1=k carga el vector entro

lw $t5, 0($s1)#$t5=v[0]=y

lw $t6, 4($s1) #$t6=v[1]=z

j cifrar #Salta a cifrar

cifrar:bgt $t1,$t0,imprimeclavecif #si $t1>$t0 saltar imprimeclavecif

addu $t2, $t2, $t3 #sum += delta;

sll $s1, $t6,4 #(z<<4)

addu $s2, $s1, $t7 #((z<<4)+k[0])

addu $s3, $t6, $t2 #(z+sum)

srl $s4, $t6,5 #(z>>5)

addu $s5, $s4, $t8 #((z>>5)+k[1])

xor $s6, $s2, $s3 #(z<<4)+k[0]) xor (z+sum)

xor $s7, $s6, $s5 #((z<<4)+k[0]) xor (z+sum) xor ((z>>5)+k[1])

addu $t5, $t5, $s7 #v[0]=y += ((z<<4)+k[0]) xor (z+sum) xor ((z>>5)+k[1]);

sll $s1, $t5,4 #(y<<4)

addu $s2, $s1, $t9 #((y<<4)+k[2])

addu $s3, $t5, $t2 #(y+sum)

srl $s4, $t5,5 #(y>>5)

addu $s5, $s4, $s0 #((y>>5)+k[3])

xor $s6, $s2, $s3 #(y<<4)+k[2]) xor (y+sum)

xor $s7, $s6, $s5 #((y<<4)+k[2]) xor (y+sum) xor ((z>>5)+k[3])

addu $t6, $t6, $s7 #v[1]=z += ((y<<4)+k[2]) xor (y+sum) xor ((y>>5)+k[3]);

addi $t1, $t1, 1 #aumenta en 1 el iterador i del bucle

j cifrar #Salta a cifrar

pidedescifrar:li $v0,4 # codigo imprimir cadena

la $a0, introducido # imprime el menu

syscall # Llamada al sistema

la $a0, v #carga el vector v donde se guardara lo que se quiere guardar

li $a1, 9 #hace que la iteracion guarde un tamanyo de 8 caracteres

li $v0, 8 #codigo leer string

syscall # Llamada al sistema

la $s1, v #$s1=k carga el vector entro

lw $t5, 0($s1)#$t5=v[0]=y

lw $t6, 4($s1) #$t6=v[1]=z

j descifrar #Salta a descifrar

descifrar:bgt $t1,$t0,imprimeclavedescif #si $t1>$t0 saltar imrimeclave

sll $s1, $t5,4 #(y<<4)

addu $s2, $s1, $t9 #((y<<4)+k[2])

addu $s3, $t5, $t4 #(y+sum)

srl $s4, $t5,5 #(y>>5)

addu $s5, $s4, $s0 #((y>>5)+k[3])

xor $s6, $s2, $s3 #(y<<4)+k[2]) xor (y+sum)

xor $s7, $s6, $s5 #((y<<4)+k[2]) xor (y+sum) xor ((z>>5)+k[3])

subu $t6, $t6, $s7 # z= v[1] -= ((y<<4)+k[2]) xor (y+sum) xor ((y>>5)+k[3]);

sll $s1, $t6,4 #(z<<4)

addu $s2, $s1, $t7 #((z<<4)+k[0])

addu $s3, $t6, $t4 #(z+sum)

srl $s4, $t6,5 #(z>>5)

addu $s5, $s4, $t8 #((z>>5)+k[1])

xor $s6, $s2, $s3 #(z<<4)+k[0]) xor (z+sum)

xor $s7, $s6, $s5 #((z<<4)+k[0]) xor (z+sum) xor ((z>>5)+k[1])

subu $t5, $t5, $s7 #y=v[0] -= ((z<<4)+k[0]) xor (z+sum) xor ((z>>5)+k[1]);

subu $t4, $t4, $t3 #sum -= delta;

addi $t1, $t1, 1 #aumenta en 1 el iterador i del bucle

j descifrar #Salta a descifrar

cambiar:li $v0, 4 # codigo imprimir cadena

la $a0, nuevaclave # imprime el contenido de la etiqueta nuevaclave

syscall # Llamada al sistema

la $a0, k # Carga el vector k que es donde s guardar? la nueva clave

li $a1, 17 #hace que la iteracion guarde un tamanyo de 16 caracteres

li $v0, 8 #codigo leer string

syscall # Llamada al sistema

li $v0,4 # codigo imprimir cadena

la $a0, cambiada # imprime el contenido de la etiqueta cambiada

syscall # Llamada al sistema

li $v0,4 # codigo imprimir cadena

la $a0, k # imprime el contenido de la etiqueta cambiada

syscall # Llamada al sistema

j main #Salta al main

imprimeclavecif:

li $t9, 4 #$t9=4

sw $t5,v #v [0]=y

sw $t6,v($t9) #v [1]=z;

li $v0,4 # codigo imprimir cadena

la $a0, cifrado # imprime el contenido de la etiqueta cifrado

syscall # Llamada al sistema

li $v0,4 # codigo imprimir cadena

la $a0, v # imprime el contenido del vector v

syscall # Llamada al sistema

j fin #Salto a fin

imprimeclavedescif:

li $t9,4 #$t9=4

sw $t5,v #v[0]=y

sw $t6,v($t9) #v[1]=z;

li $v0,4 # codigo imprimir cadena

la $a0, descifrado # imprime el contenido de la etiqueta descifrado

syscall # Llamada al sistema

li $v0,4 # codigo imprimir cadena

la $a0, v # imprime el contenido del vector v

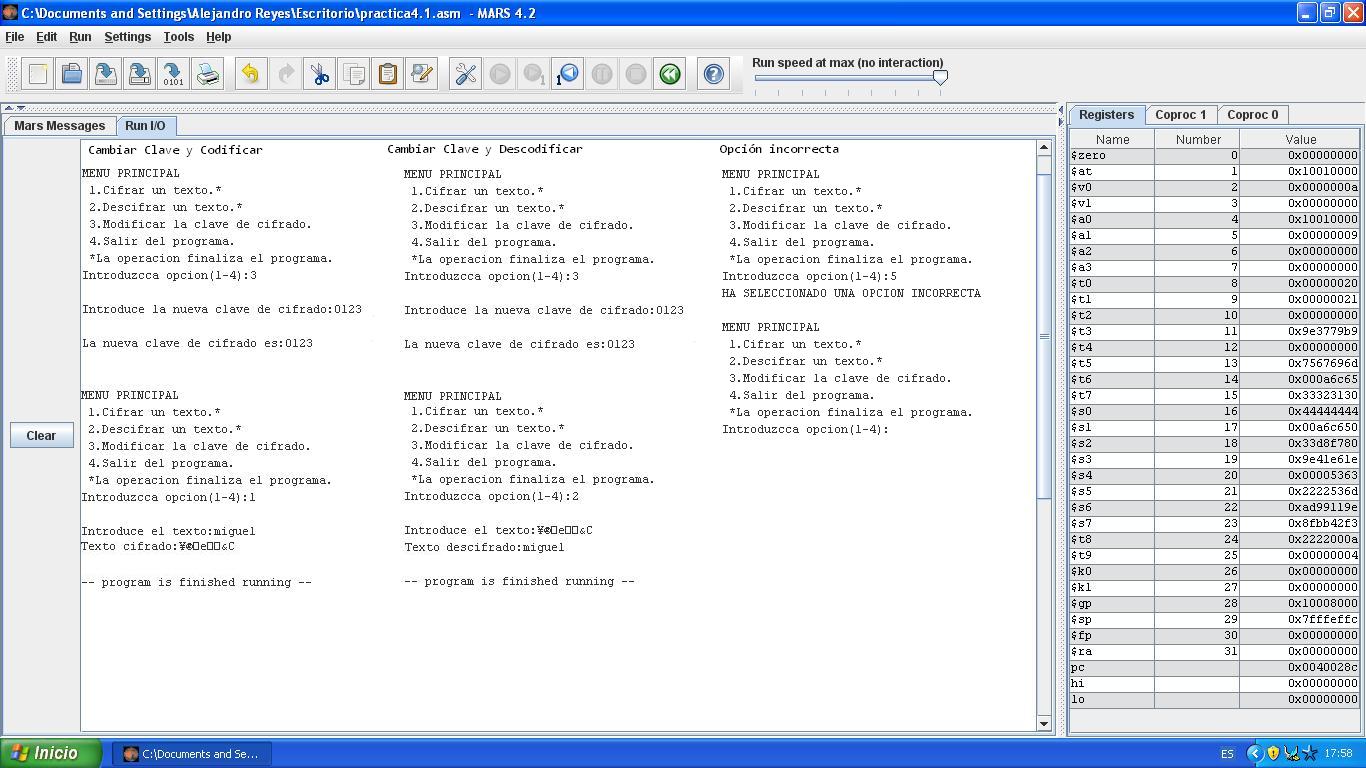
syscall # Llamada al sistema

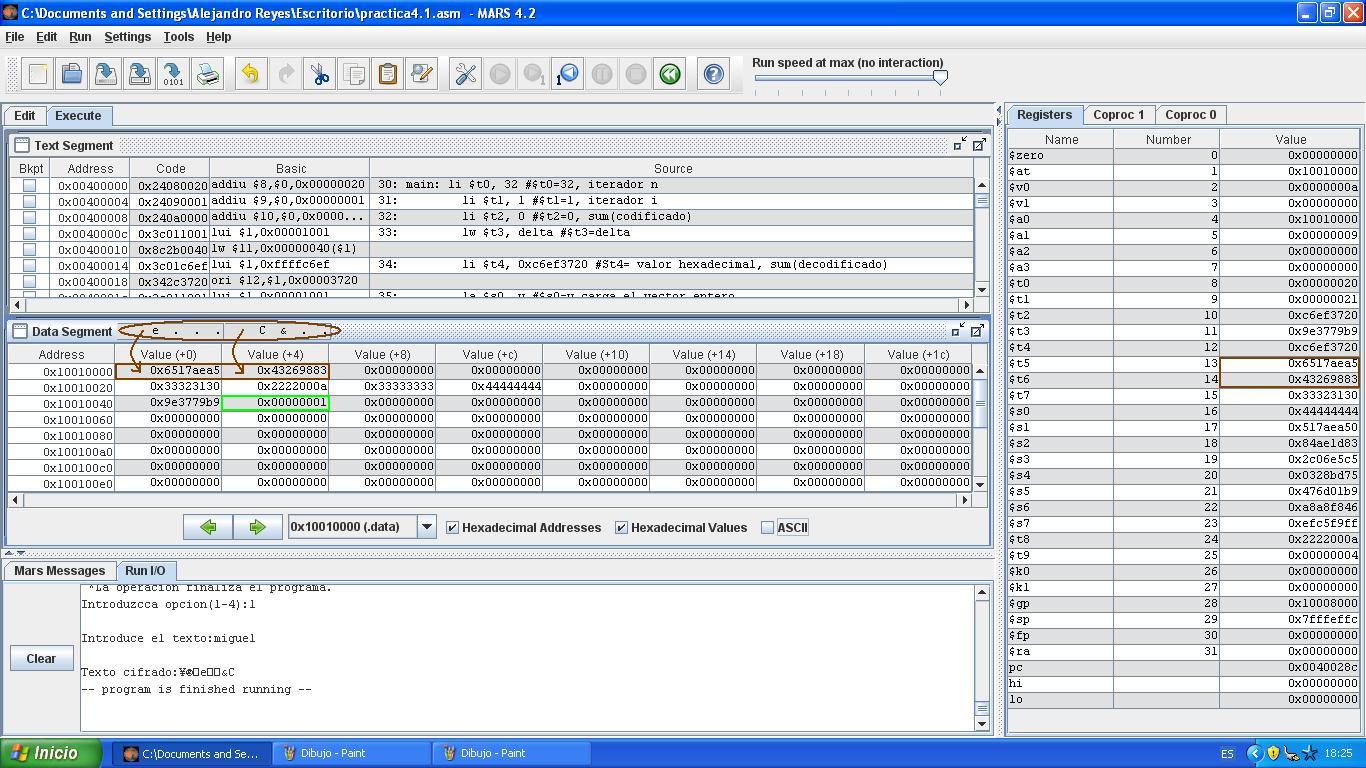
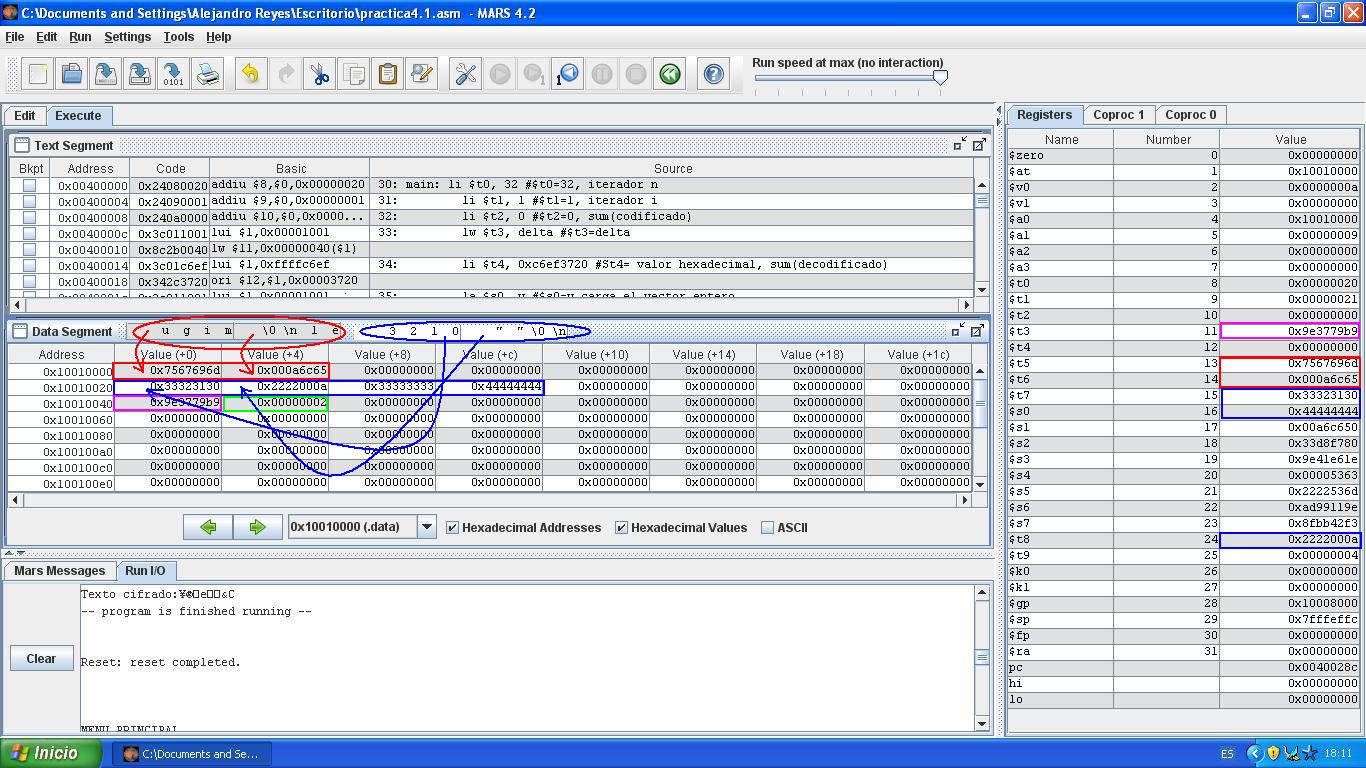
j fin #Salto a fin

fin: li $v0, 10 # Fin programa

syscall # Llamada al sistema

A continuación mostramos una ejecución del programa del ensamblador:





Las imágenes anteriores muestran diferentes ejecuciones del programa.

La primera imagen ha sido modificada para que se puedan observar las distintas ejecuciones del programa y de esta manera comprobar su funcionamiento.

En las dos siguientes imágenes podemos observar los registros con diferentes resaltos en colores diversos. Por ejemplo, en ambas imágenes la opción escogida está mostrada con un recuadro verde, en la segunda imagen, la cual corresponde a la ejecución del algoritmo de cifrado podemos ver en un recuadro marrón los registros que guardan la nueva clave cifrada, así como su equivalente en ASCII redondeado en una elipse del mismo color.

En la primera imagen, al igual que en la anteriormente mencionada, podemos observar en color rojo la clave anteriormente cifrada una vez que ya ha sido descifrada, con su correspondencia en ASCII, y en azul podemos ver la clave de cifrado utilizada en el algoritmo.

Como dicha clave es la misma para las dos imágenes únicamente está resaltada en una de ellas.

Podemos observar que únicamente hay dos flechas apuntando a 2 partes del vector clave, esto es debido a que hemos cambiado la clave de cifrado con la opción 3 del menú, es por esto que los valores de las otras dos celdas recuadradas no han sido modificados de la predeterminada por el usuario del sistema.

Por último, he querido resaltar con color fucsia el valor de delta, que es utilizado por las opciones 1 y 2 del programa.

**TANTO EL CÓDIGO COMO LAS IMÁGENES (INCLUYENDO EL CÓDIGO EN C Y SU EJECUTABLE QUE DEVUELVE VALORES EN DECIMAL) SE ENCUENTRAN ADJUNTOS EN EL FICHERO COMPRIMIDO .ZIP QUE CONTIENE A ESTA MEMORIA.**