# PRACTICA 4 – JUEGO DE PAREJAS

## INTRODUCCIÓN A LOS COMPUTADORES

**Trabajo realizado por Pablo Izquierdo González**
**y Esmeralda Madrazo Quintana**

Práctica nº: 4 Fecha: 22/05/2019 – 25/05/2019

Autores: Pablo Izquierdo González y Esmeralda Madrazo Quintana

---

## Introducción

Nuestra forma de afrontar esta practica fue declarando dos matrices una oculta (con #) que el jugador vería al iniciar el juego y otra correcta (con las letras) que contiene las cartas del juego.
Después, declaramos una serie de datos para comunicarnos con el jugador a lo largo de la partida.
Además, declaramos un vector que se encuentra vacío al comienzo de la partida y que se va rellenando con las coordenadas de las cartas a medida que el jugador las va acertando, y que nos sirve para la comprobación de errores.

Lo primero que hacemos es limpiar el vector de coordenadas de las almacenadas en anteriores ejecuciones del juego, para evitar que de errores inexistentes. Para ello usamos la subrutina **LimpiaMemoria,** a la que le pasamos la dirección del vector de coordenadas(R0), el tamaño de dicho vector(R1) y un carácter que no se usa en ninguna de las cartas(R2). Esta subrutina no devuelve nada.

Después, barajeamos la matriz de coordenadas para que el juego no sea demasiado simple mediante el uso de la subrutina **BARAJEA**, la cual usa la función **random_integer** que os ha sido facilitada. A **BARAJEA**, le pasamos como parámetro la dirección de la matriz correcta. Esta subrutina no devuelve nada.

Movemos el valor 0 a R8 ya que lo vamos a usar como desplazamiento del vector coordenadas, que es el que contiene las coordenadas de las cartas acertadas por el jugador.

Comienza el juego y aisa al jugador de ello mostrándole un texto e imprimiendo la matriz oculta mediante la subrutina **Muestra_Matriz**, a la que le pasamos como parámetros la dirección de la matriz oculta(R0), el número de filas(R1) y el número de columnas(R2). Además, inicializamos el contador que se encuentra en R11 a 0. Este nos servirá para llevar cuenta del número de aciertos del jugador y así sabremos que al llegar a 21 se habrá terminado el juego.

Pedimos las coordenadas de la primera carta y comprobamos que se encuentren en el rango de nuestra matriz y que no se correspondan con las de ninguna carta ya acertada por el jugador mediante la subrutina **Comprueba_Error**, a la que le pasamos como parámetros una palabra que contiene a las coordenadas x e y(R0) la cual hemos generado mediante la llamada a la subrutina **Genera_Palabra**, que recibe como parámetro las coordenadas x(R0) e y (R1) y también 2 valores que serian los de z y k pero que al no haber sido introducidas aun sustituimos por 2 valores que no se encuentran en

la matriz correcta(R2 y R3); la dirección del vector coordenadas(R1) y el desplazamiento que tenemos guardado en R8(R2).

En caso de que la subrutina anterior devuelva 1 (lo cual significa que no se corresponden con ninguna de las coordenadas correctas) continuamos con la ejecución normal del código, en caso contrario, imprimimos un mensaje de error y volvemos a solicitar las coordenadas x e y.

Al continuar con el código, volteamos la carta introducida por el usuario mediante el uso de la subrutina **Voltear**, que recibe como parámetros una palabra que contiene las coordenadas de la carta que se desea voltear(R0), la dirección de la matriz oculta(R1), la longitud del elemento(R2), la dirección de la matriz correcta(R3) y el número de columnas(R12). Esta función devuelve la letra volteada, la cual almacenamos en R9 para una vez realizados los mismos pasos con las coordenadas de la segunda carta poder comprobar que si esta se corresponde con la otra letra que hemos almacenado en R10 y que hemos obtenido mediante la función **Voltear**.

Esta comprobación la hacemos mediante el uso de la subrutina **CompruebaLetra**, (que es en la que usaríamos los leds pero que debido a un problema con la imagen de la Raspberry no hemos podido llevar a cabo ya que no nos era posible comprobar si funcionaba, ya que congelaba la Raspberry al intentar ejecutarla) a la que le pasamos como parámetros, la letra almacenada en R9(R0), la letra almacenada en R10(R1). Esta función devuelve 1 si son correctas y 0 si no lo son.

En caso afirmativo, sumamos uno al contador de puntuación(R11) y almacenamos dichas coordenadas en el vector de coordenadas mediante el uso de la subrutina **Guarda_Pareja**, que usa como parámetros una palabra (que hemos obtenido usando el **Genera_Palabra**) que contiene las coordenadas de ambas cartas(R0) y la dirección del vector de coordenadas con el desplazamiento adecuado que se encuentra en R8(R1). Después, sumamos 4 al desplazamiento(R8).

En caso de fallo, volteamos de nuevo las cartas para que oculten su valor.

En ambos casos repetimos este proceso hasta la finalización del juego, que se lleva a cabo al llegar el contador de aciertos(R11) a 21.

## Pseudo-Código

**Muestra_Matriz:**

```
void muestraMatriz(@MatrixO, NF, NC, LE) {
        int k = 0;
        while (k < NF) {
                int i = 0;
                while (i < NC) {
                        imprimeLetra (k, i, NC, LE);
                        i++;
                }
                printf(\n);
                k++;
        }
}
```

**imprime_Letra:**

```
void imprime_letra(k, i) {
        int pos = (k * NC + i) * LE;
        printf(elemento en la posición pos);
}
```

**Voltear:**

```
char Voltear(palabra, @matrixO, LE, @matrixC, NC) {
        R0 = x;
        R8 = y;
        R4 = (x*NC + y) * LE;
        R6 = letra1 (matrixO con desplazamiento R4);
        R7 = letra2 (matrixC con desplazamiento R4);
        // Guardo R6 en la matrixC con desplazamiento R4;
        // Guardo R7 en la matrixO con desplazamiento R4;
        return R7;
}
```

**Genera_Palabra:**

```
int Genera_Palabra(x, y, z, k) {
        palabra = 000k;
        palabra += 00z0;
        palabra += 0y00;
        palabra += x000;
        return palabra;
}
```

**Guarda_Pareja:**

```
void Guarda_Pareja(palabra, @vector_coordenadas) {
    // guardo la palabra (que son las coordenadas de las cartas acertadas) en
    la dirección especificada;
}
```

**Comprueba_Error:**

```
bool Comprueba_Error(palabra, @Vector_Coordenadas, desplazamiento) {
    R4 = 0000 primera mitad de la palabra;
    R5 = 0000 segunda mitad de la palabra;
    R6 = @Vector_Coordenadas;
    R0 = desplazamiento
    R0 = R0/2 +1
    for(int R7 = 0; R7 < R0; R7++) {
        @return TRUE (1) si se encuentra volteada
        @return FALSE (0) si no se encuentra volteada
    }
}
```

**Limpia_Memoria:**

```
void Limpia_Memoria(@Vector_Coordenadas, elementoInexistente) {
    for (int i = 0; i < 84; i++) {
        // Guardamos el elementoInexistente en la dirección del
        vector_Coordenadas con desplazamiento i.
    }
 }
```

**CompruebaLetra:**

```
bool CompruebaLetra (letra1, letra2) {
    if(letra1 == letra2) {
        return True(1);
    }
    return False(0);
}
```

**BARAJEAR:**

```
void Barajear(@matrixC) {
        int i = 0;
        while(i < 42) {
                R12 = @matrixC;
                X = // Genero X aleatoria;
                Y = // Genero Y aleatoria;
                Z = // Genero Z aleatoria;
                K = // Genera K aleatoria;
                Cambio(X, Y, Z, K, @matrixC);
                i++;
        }
}
```

**Cambio:**

```
void cambio(X, Y, Z, K, @matrixC) {
        //Carga una letra
        //Carga otra letra
        //Guarda letra 1 en posición 2
        //Guarda letra 2 en posición 1
}
```

# Código

```
.data
matrixO : .asciz  "#","#","#","#","#","#","#", "#","#","#","#","#","#","#","#", "#","#","#","#","#","#","#","#",
"#","#","#","#","#","#","#", "#","#","#","#","#","#","#", "#","#","#","#","#","#","#"
matrixC : .asciz  "A","B","C","D","E","F","G", "H","I","J","K","L","M","N", "O","P","Q","R","S","T","U",
"A","B","C","D","E","F","G", "H","I","J","K","L","M","N", "O","P","Q","R","S","T","U"

ESPACIO : .asciz "\n"
      .align 2
vectorCoor: .space 84


.equ LE, 2
.equ NC, 7
.equ NF, 6
.equ LM, 84

Texto1: .asciz "\nBienvenido a este increible juego\n"
Texto2: .asciz "Introduzca coordenadas de la primera carta:\n"
Texto3: .asciz "Coordenada X: "
Texto4: .asciz "\nCoordenada Y: "
Texto5: .asciz "\nIntroduzca coordenadas de la segunda carta:\n"
Texto6: .asciz "Coordenada Z: "
Texto7: .asciz "\nCoordenada K: "
Texto8: .asciz "\nError vuelva a introducir coordenadas\n"
Texto9: .asciz "\nHas fallado, vuelve a intentarlo"
Texto10: .asciz "\nHas acertado, sigue asi"

TextoFin: .asciz "\nFelicidades has ganado"

.text
.globl _start
_start:

LDR R0, =vectorCoor
MOV R1,#84
MOV R2,#0x50

BL LimpiaMemoria

LDR R0,=matrixC
BL BARAJEA

MOV R8,#0

LDR R0,=Texto1
SWI 0x02

MOV R11,#0

WhileJuego:
CMP R11,#21     @bucle principal
BEQ FinJuego

LDR R0, =matrixO
MOV R1,#NF
MOV R2,#NC
MOV R3,#LE
```

```
BL Muestra_Matriz

Coordenadas:

LDR R0,=Texto2 @FRASE
SWI 0x02

LDR R0,=Texto3 @X
SWI 0x02

SWI 0x04
SUB R4,R0,#0x30

LDR R0,=Texto4 @Y
SWI 0x02

SWI 0x04
SUB R5,R0,#0x30

LDR R0,=ESPACIO
SWI 0x02

@Compruebo rango de coordenadas
MOV R0,#NF
SUB R0,R0,#1 @Maximo valor
CMP R4,R0
LDRGT R0,=Texto8
SWIGT 0x02
BGT Coordenadas

MOV R0,#NC
SUB R0,R0,#1 @Maximo valor
CMP R5,R0
LDRGT R0,=Texto8
SWIGT 0x02
BGT Coordenadas


MOV R0,R4
MOV R1,R5
MOV R2,#0x51
MOV R3,#0x51
BL Genera_Palabra

LDR R1,=vectorCoor
MOV R2,R8
BL Comprueba_Error

CMP R0,#1
BNE SIGUE
   LDR R0,=Texto8
   SWI 0x02
   B Coordenadas

SIGUE:
MOV R0,R4
MOV R1,R5
MOV R2,#0
```

```
MOV R3,#0
BL Genera_Palabra

LDR R1,=matrixO
MOV R2,#LE
LDR R3,=matrixC
MOV R12,#NC
BL Voltear
MOV R9,R0


LDR R0,=matrixO
MOV R1,#NF
MOV R2,#NC
MOV R3,#LE

BL Muestra_Matriz

Coordenadas2:

LDR R0,=Texto5 @FRASE
SWI 0x02

LDR R0,=Texto6 @Z
SWI 0x02

SWI 0x04
SUB R6,R0,#0x30

LDR R0,=Texto7 @K
SWI 0x02

SWI 0x04
SUB R7,R0,#0x30

@Compruebo rango de coordenadas
MOV R0,#NF
SUB R0,R0,#1 @Maximo valor
CMP R6,R0
LDRGT R0,=Texto8
SWIGT 0x02
BGT Coordenadas2

MOV R0,#NC
SUB R0,R0,#1 @Maximo valor
CMP R7,R0
LDRGT R0,=Texto8
SWIGT 0x02
BGT Coordenadas2

MOV R0,R4
MOV R1,R5
MOV R2,R6
MOV R3,R7
BL Genera_Palabra

LDR R1,=vectorCoor
MOV R2,R8
BL Comprueba_Error
```

```
CMP R0,#1
BNE SIGUE2
   LDR R0,=Texto8
   SWI 0x02
   B Coordenadas2

SIGUE2:
MOV R0,R6
MOV R1,R7
MOV R2,#0
MOV R3,#0
BL Genera_Palabra

LDR R1,=matrixO
MOV R2,#LE
LDR R3,=matrixC
MOV R12,#NC
BL Voltear
MOV R10,R0


MOV R0,R4
MOV R1,R5
MOV R2,R6
MOV R3,R7
BL Genera_Palabra

LDR R1,=vectorCoor
MOV R2,R8
BL Comprueba_Error

CMP R0,#1
BNE CONTINUA_COOR
   LDR R0,=Texto8
   SWI 0x02
   B Coordenadas

CONTINUA_COOR:

MOV R0,R9
MOV R1,R10
BL CompruebaLetra

CMP R0,#1
BNE CONTINUA_FALLO
   MOV R0,R4
   MOV R1,R5
   MOV R2,R6
   MOV R3,R7
   BL Genera_Palabra

   LDR R1,=vectorCoor
   ADD R1,R1,R8

   BL Guarda_Pareja

   ADD R8,R8,#4
   ADD R11,R11,#1
```

```
  B CONTINUA_CORRECTO

CONTINUA_FALLO:

LDR R0,=ESPACIO
SWI 0x02

LDR R0,=matrixO
MOV R1,#NF
MOV R2,#NC
MOV R3,#LE

BL Muestra_Matriz

@doy vuelta a las cartas fallidas
MOV R0,R4
MOV R1,R5
MOV R2,#0
MOV R3,#0
BL Genera_Palabra

LDR R1,=matrixO
MOV R2,#LE
LDR R3,=matrixC
MOV R12,#NC

BL Voltear  @fallo las vuelvo a su sitio

MOV R0,R6
MOV R1,R7
MOV R2,#0
MOV R3,#0
BL Genera_Palabra

LDR R1,=matrixO
MOV R2,#LE
LDR R3,=matrixC
MOV R12,#NC

BL Voltear  @fallo las vuelvo a su sitio

LDR R0,=Texto9
SWI 0x02
B REPITE

CONTINUA_CORRECTO:

LDR R0,=Texto10
SWI 0x02

REPITE:
LDR R0,=ESPACIO
SWI 0x02

B WhileJuego

FinJuego:

LDR R0,=TextoFin
```

```
SWI 0x02

SWI 0x11



Muestra_Matriz:
STMDB SP!, {R4-R8, LR}

MOV R6,R1 @NF
MOV R7,R2 @NC
MOV R8,R3 @LE

ADD R2,R0,#0
MOV R4,#0


WHILEMM1:
        CMP R4,R6
        BGE FINMM

                MOV R5,#0
                WHILEMM2:
                        CMP R5,R7
                        BGE FWHILEMM2
                                MOV R0,R4  @X
                                MOV R1,R5  @Y
                                        @R2 DIR MATRIZ O
                                MOV R3, R7   @NC
                                MOV R12, R8     @LE
                                BL Imprime_Letra
                                ADD R5,R5,#1
                                B WHILEMM2

                        FWHILEMM2:
                        LDR R0,=ESPACIO
                        SWI 0x02
                        ADD R4,R4,#1
                        B WHILEMM1
        FINMM:
                LDMIA SP!, {R4-R8, PC}

Imprime_Letra:

        STMDB SP!, {R4-R5}

        MOV R5,R3
        MUL R4,R0,R5

        ADD R4,R4,R1

        MOV R5,R12
        MUL R1,R4,R5

        ADD R0,R2,R1
        SWI 0x02
```

```
        LDMIA SP!, {R4-R5}
        MOV PC,LR

CompruebaLetra:

CMP R0,R1
BNE ELSECL
  MOV R0,#1 @SON IGUALES
  B FINCL

ELSECL:
MOV R0,#0    @SON DISTINTAS

FINCL:
MOV PC,LR

Voltear:

STMDB SP!, {R4-R8, LR}

LSR R8,R0,#8
AND R0,R0,#0xFF


MUL R4,R0,R12
ADD R4,R4,R8


MUL R4,R2,R4

LDRB R6, [R1,R4]
LDRB R7, [R3,R4]

STRB R7, [R1,R4]
STRB R6, [R3,R4]

MOV R0,R7
LDMIA SP!, {R4-R8, PC}

Genera_Palabra:

LSL R1,#8
ORR R0,R1
LSL R2,#16
ORR R0,R2
LSL R3,#24
ORR R0,R3

MOV PC,LR


Guarda_Pareja:

STR R0,[R1]

MOV PC,LR

Comprueba_Error:
```

```
STMDB SP!, {R4-R10, LR}

MOV R4,#0xFF00
ORR R4,#0xFF

AND R4,R0,R4
LSR R5,R0,#16
MOV R6,R1

CMP R4,R5
BEQ ENCONTRADO


MOV R0,R2
MOV R1,#2

BL Cociente

ADD R0,R0,#1
MOV R7,#0

ForCE:
CMP R7,R0
BGE FForCE
   MOV R10,#2
   MUL R8,R7,R10
   LDRH R9, [R6,R8]
   CMP R9,R4
   BNE ElseCE
      B ENCONTRADO

   ElseCE:
   CMP R9,R5
   BEQ ENCONTRADO
      ADD R7,R7,#1
      B ForCE



ENCONTRADO:
MOV R0,#1
B FinCE


FForCE:   @No Encontrado
MOV R0,#0
FinCE:
LDMIA SP!, {R4-R10,PC}


Cociente:
STMDB SP!, {R4,LR}
MOV R4,#0

WhileCO:
CMP R0,R1
BLE FinwhileCO
   SUB R0,R0,R1
```

```
   ADD R4,R4,#1
   B WhileCO
FinwhileCO:
MOV R0,R4
LDMIA SP!, {R4,PC}


LimpiaMemoria:

MOV R3,#0

FORLM:
CMP R3,R1
BGE FINLM
   STRB R2,[R0,R3]
   ADD R3,R3,#1
   B FORLM

FINLM:
MOV PC,LR



BARAJEA:

STMDB SP!, {R4-R9,LR}

ADD R9,R0,#0  @DIRECCION

MOV R8,#0

FORBAR:
CMP R8,#42
BEQ FINFORBAR

MOV R0,#6
BL random_integer
MOV R4,R0     @X
MOV R0,#6
BL random_integer
MOV R5,R0     @Z
MOV R0,#7
BL random_integer
MOV R6,R0     @Y
MOV R0,#7
BL random_integer
MOV R7,R0     @K

MOV R0,R4 @X
MOV R1,R6 @Y
MOV R2,R5 @Z
MOV R3,R7 @K
ADD R12,R9,#0
BL CAMBIO

ADD R8,R8,#1

B FORBAR
```

```
FINFORBAR:

LDMIA SP!, {R4-R9, PC}

CAMBIO:

STMDB SP!, {R4-R7,LR}

MOV R5,#NC
MUL R4,R0,R5
ADD R4,R4,R1

MOV R5,#LE
MUL R4,R5,R4 @DESPLAZAMIENTO 1

MOV R5,#NC
MUL R6,R2,R5
ADD R6,R6,R3

MOV R5,#LE
MUL R5,R6,R5 @DESPLAZAMIENTO 2

LDRB R6, [R12,R4]
LDRB R7, [R12,R5]

STRB R7, [R12,R4]
STRB R6, [R12,R5]

LDMIA SP!, {R4-R7,PC}
```

## Observaciones

A la hora de depurar el código nos hemos ayudado el UC Debug y la terminal de la propia Raspberry. Hemos ido depurando subrutina por subrutina, comprobando que todas ellas iban funcionando correctamente antes de pasar a la siguiente, y completando la main al mismo tiempo para poder llamarlas correctamente.

En los casos en los que nuestras subrutinas necesitaban llamar a otras que aún no habíamos creado las agrupábamos y las redactábamos y depurábamos en conjunto en el UC Debug ya que al ser varias era necesario el uso de breakpoints para detectar sus errores.

Una vez detectados los fallos de sintaxis y los saltos a direcciones de memoria inexistentes, nos propusimos iniciar una partida para comprobar su correcto funcionamiento, de esta manera detectamos un par de errores en la funcionalidad que solventamos modificando la subrutina **Comprueba_Error** y la main.

El único problema que no hemos podido solventar es uno que concierne la implementación de los leds. En un principio decidimos realiza un código de prueba para comprobar el funcionamiento de los leds. Gracias a esto, observamos que al ser ejecutados en el UC Debug la Raspberry quedaba totalmente congelado y era necesario desconectarla completamente para que volviera a funcionar. Después, decidimos probar los leds en la terminal, pero aquí observamos un segundo problema, que el tiempo que estaban encendidos era demasiado pequeño para poder ser percibido de manera adecuada.

En caso de haber sido capaces de implementar los leds de manera adecuada hubiéramos colocado las llamadas en los puntos en los que notificamos el acierto (led verde) y el error (led rojo). También hubiéramos aprovechado el led amarillo para notificar al usuario que no ha introducido las coordenadas de manera correcta, así que la hubiéramos encendido junto a los mensajes de ‘Error, vuelva a introducir las coordenadas’.

Este es el código que realizamos para probar los leds:

```
.text
.globl leds
_start:

MOV R0,#10
MOV R1,#9

BL leds


MOV R0,#17
MOV R1,#11

BL leds

MOV R0,#27
MOV R1,#22

BL leds

SWI 0x11
```