MEMORIA TETRIS

CURSO 18/19

AUTOR: COSMIN MARIAN PADURARU

Proyecto de programación en ensamblador de Estructura y Tecnología de Computadores

ÍNDICE

El archivo con el código fuente en ensamblador se encuentra en el archivo *Tetris.s*

1.0 EJERCICIOS DE TRADUCCIÓN

En los ejercicios de traducción lo que he hecho ha sido traducir literalmente el código del archivo *tetris.c* que se nos da dentro del archivo *ensamblador-proyecto-materiales.tar.gz* que se encuentra en el Aula Virtual. Para probar el correcto funcionamiento de ciertas funciones a traducir he creado nuevos "mains". Aquí un ejemplo con el imagen_copy:

```
main_copy:    #Para probar el imagen copy

        addiu   $sp, $sp, -4
        sw      $ra, 0($sp)
        la      $a0,pieza_barra
        li      $a1,4
        li      $a2,5
        li      $a3,'P'
        jal     imagen_init
        la      $a0,pieza_barra
        la      $a1,pieza_jota
        jal     imagen_copy
        la      $a0,pieza_barra
        jal     imagen_print
        jal     mips_exit
        lw      $ra, 0($sp)
        addiu   $sp, $sp, 4
        jr      $ra
```

2.0 EJERCICIOS DE IMPLEMENTACIÓN

Para los ejercicios de implementación en los más complicados primero he creado un código en alto nivel, respectivamente en C posteriormente lo he pasado a ensamblador una vez que ya había visualizado la idea en el código C.

2.1 Marcador de puntuación

He usado el procedimiento *integer_to_string* que se realizó en clase en el *Boletín 2*, por otra parte, este procedimiento recibe tres parámetros. Pero lo he modificado, ya que siempre querremos que sea en base 10, por lo que el parámetro $a3 (base) lo he eliminado y he usado un *li $t7,10* en vez de cargarlo como parámetro de entrada. El procedimiento se llama *imagen_dibuja_cadena.* Por otra parte, he creado una variable que contendrá la cadena de texto puntuación, que se encuentra en el *.data* y se llama *puntuacion_cadena,* además he creado otra

variable que contendrá la puntuación del dígito que se irá actualizando. Como en el pdf se indica que: *Cada nueva pieza que aparezca proporcionará 1 punto.* Mi versión del Tetris empieza directamente con la puntuación uno. Pero si quisiéramos que la puntuación empezase en cero, sería suficiente inicializar la variable en -1 y luego en jugar partida también inicializarla en -1.

He modificado también *jugar_partida*, nos traemos de memoria la variable puntuación, la cargamos con cero y la volvemos a subir a memoria. Y luego también se ha modificado *actualizar_pantalla.*

2.2 Final de la partida

En primer lugar, he creado en el .data igual que en el proceso anterior un .asciiz con el cuadro de texto que pone "FIN DE PARTIDA pulse una tecla" que está situado en *fin_de_partida.* La función se llama *final_de_la_partida.* Hace una llamada a *imagen_dibuja_cadena* para dibujar la cadena de texto. Para implementar el correcto funcionamiento de esta función he modificado *jugar_partida*. Por lo que cada vez que *probar_pieza* de falso (es decir cero) se sabrá que la pieza no se puede colocar y por lo tanto se deberá terminar la partida, de aquí salta a final_partida y de ahí, al pulsar una tecla vuelves al menú principal.

2.3 Completado líneas

En una primera instancia completado_líneas no era exactamente así, pero posteriormente la tuve que modificar, para su uso en eliminando líneas que es la función siguiente. Primero realicé esta función en C, aquí dejo su código correspondiente.

```
void completado_lineas()
{
   for(int i=0;i<campo->alto;i++)
   {
      for(int j=0;j<campo->ancho;j++)
         if(imagen_get_pixel(campo,j,i)==' ')
            {
               return;
            }
   }
      puntuacion=+10;
      eliminando_lineas(i);
}
```

Lo único que he hecho para implementar la función en todo el juego de tetris ha sido hacer un *jal completado_lineas* en la parte de *jugar_juego* que va comprobando cada vez si se completa una línea o no. En la función *completado_lineas* se hace una llamada a *eliminando_lineas* que se verá a continuación.

2.4 Eliminando líneas

Mi forma de implementar esta idea es que cada vez que se complete una línea también habrá que eliminarla. Por lo tanto, cada vez que se consigan 10 puntos también llamaré a la función *eliminando_lineas.* A esta función se le pasará una altura como parámetro que será la altura donde se encuentra la línea completada, obtenida en el bucle de la función anterior. Igual que en la función anterior primero lo he hecho en C aquí el código correspondiente:

```
void eliminando_lineas(int desde)
{
   for(int i=desde,i>0,i--)
   {
      for (j=0,j<=campo->ancho,j++)
      {
         pixel p=imagen_get_pixel(campo,x,y-1)      //x=j   y=i
         imagen_set_pixel(campo,x,y, p)
      }
   }
}
```

Consiste básicamente en que iremos obteniendo los píxeles que hay en la posición superior y nos los traeremos en la posición donde se ha completado la línea, así hasta llegar a la fila cero del campo, donde ya pararemos.

2.5 Mostrar la siguiente pieza

Igual que en los ejercicios anteriores hemos creado otra variable *.asciiz* que contiene las líneas horizontales y verticales que usaremos para dibujar el rectángulo correspondiente, estas dos variables están en *pieza_sig1* y *pieza_sig2* además he creado un *pieza_sig*, que consiste una estructura que guarda la siguiente pieza. Por otra parte, he tenido que crear un procedimiento auxiliar llamado a *nueva_pieza_siguiente,* el cual nos generará una pieza aleatoria y la guardaremos en *pieza_sig*, por otra parte, tenemos el procedimiento *mostrar_la_pieza_siguiente*, que consiste básicamente en un conjunto de llamadas a *imagen_dibuja_cadena* para el rectángulo y una llamada a *imagen_dibuja_imagen* para la pieza. *mostrar_la_pieza_siguiente* está puesto en *actualizar_pantalla.*

También se modifica *nueva_pieza_actual* para que la *pieza_actual* sea la pieza_siguiente en vez de generar una pieza aleatoria cada vez.

3. BIBLIOGRAFÍA

-Apuntes de ensamblador del Aula Virtual.