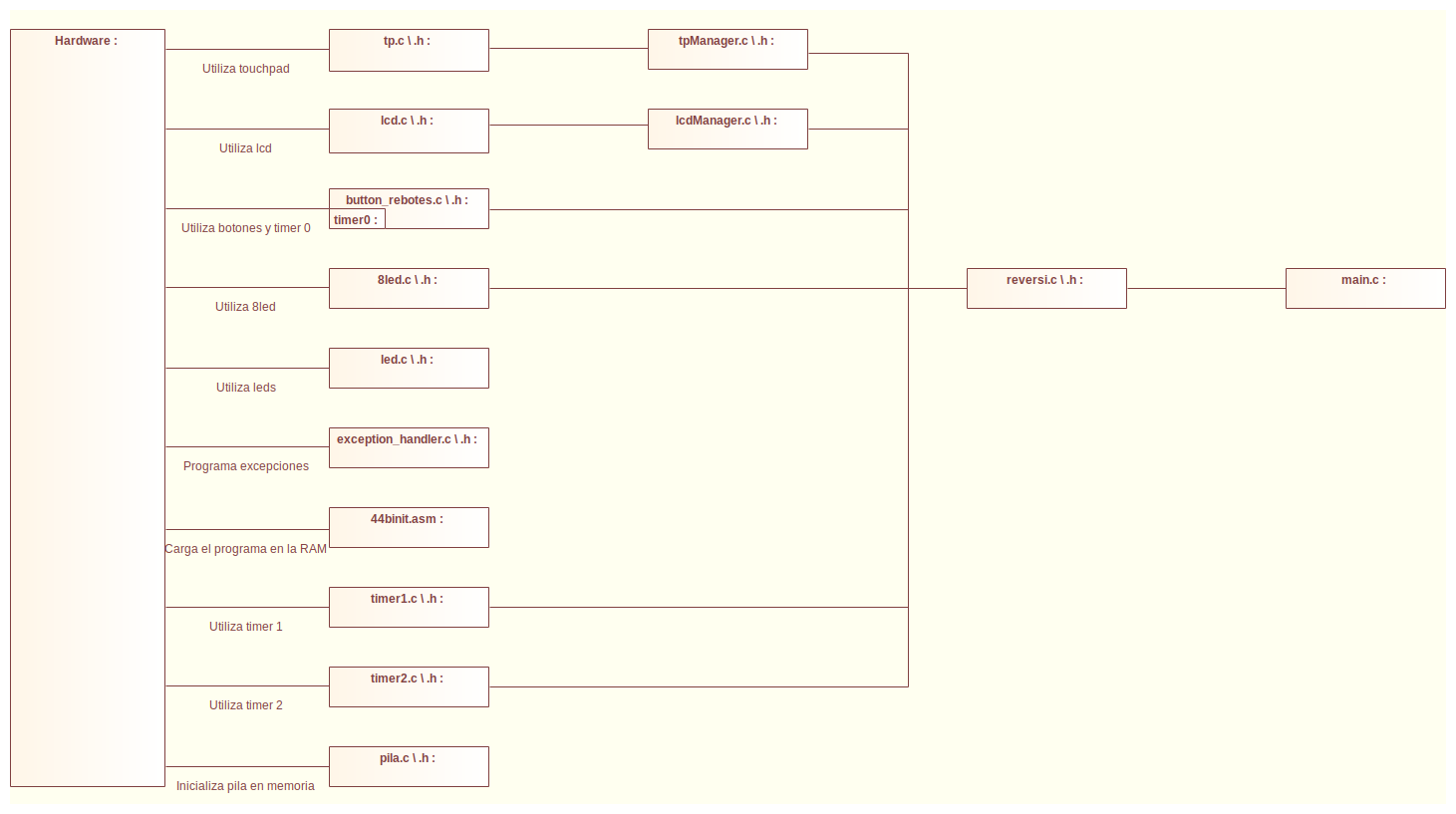
**Estructura del proyecto**

****

El proyecto, está compuesto por 4 niveles, los cuales están delimitados según su dependencia del hardware.

En el primer nivel se encuentran las bibliotecas que interactúan directamente con el hardware, que son tp, lcd, button\_rebotes, 8led, led, exception\_handler, 44binit.asm, timer2, timer1 y pila.

Luego existe un segundo nivel el cual utiliza solamente las bibliotecas del nivel 1,estando compuesto este nivel por los gestores del lcd (lcdManager) y el touchpad(tpManager). La existencia de este nivel permite crear una separación entre el juego y el hardware. Permitiendo esto que las implementaciones de bibliotecas que trabajan con el hardware puedan ser reutilizadas en otros proyectos así como poder desarrollar el juego sin haber desarrollado previamente el uso del hardware.

El tercer nivel es el juego del reversi, el cual utiliza tanto las bibliotecas del nivel 2 (las cuales se desarrollaron en la 3ª práctica) como las bibliotecas del nivel 1 que no están gestionadas por bibliotecas del nivel 2 (las cuales se desarrollaron el la 2ª práctica).

El último nivel está compuesto por el fichero main. Este se encarga tanto de inicializar todas las bibliotecas, así como inicializar el juego.

**Descripción de las distintas bibliotecas**

tp: Esta biblioteca interactúa con el hardware para poder controlar el touchpad desde ella. Posee una interfaz que será utilizada por todas las demás bibliotecas que deseen acceder al touchpad y desde la que se puede saber si se ha pulsado el touchpad y el punto en el que se ha pulsado según el sistema de coordenadas del lcd((0,0) esquina superior izquierda y (320,240) esquina inferior derecha). Por otro lado esta biblioteca también calibra el touchpad las 8 primeras veces que se pulsa sobre él.

lcd: Esta biblioteca interactuá con el hardware para poder controlar el lcd desde ella. Posee una interfaz que será utilizada por todas las demás bibliotecas que deseen acceder al lcd y desde la que se puede dibujar formas básicas (cuadrados, lineas, círculos y letras) sobre él.

button\_rebotes: Esta biblioteca interactuá con el hardware para poder controlar los botones desde ella. Posee una interfaz que será utilizada por todas las demás bibliotecas que deseen interactuar con los botones y desde la que se puede saber si se ha pulsado un botón, y en caso afirmativo cuál. Por otro lado, esta biblioteca se encarga de gestionar los rebotes producidos durante la pulsación de los botones haciendo uso para ello del timer 0.

8led: Esta biblioteca interactuá con el hardware para poder controlar el 8led.

led: Esta biblioteca interactuá con el hardware para poder controlar los 2 leds.

exception\_handler: Esta biblioteca se encarga de gestionar las excepciones de tipo data abort, prefetch abort, software interrupt y undefinned.

44binit.asm: Este fichero se encarga de inicializar la placa así como de copiar el programa de la memoria flash a la memoria ram y ejecutarlo.

timer1: Esta biblioteca se encarga de gestionar el timer 1. Posee una interfaz que será utilizada por todas las demás bibliotecas que deseen acceder al timer 1, desde la que se puede resetear la cuenta y ver el valor de la cuenta en un momento dado. Este timer tiene una baja precisión y es utilizado para controlar el paso de los segundos en el juego(tiempo de juego).

timer2: Esta biblioteca se encarga de gestionar el timer 2. Posee una interfaz que será utilizada por todas las demás bibliotecas que deseen acceder al timer 2, desde la que se puede resetear la cuenta y ver el valor de la cuenta en un momento dado. Su precisión es muy alta y es utilizada por el juego para obtener el tiempo de calculo.

pila: Esta biblioteca se encarga de gestionar la pila de depuración. Posee una interfaz que permite tanto apilar como desapilar eventos de la pila.

lcdManager: Esta biblioteca se encarga de dibujar las pantallas. Para ello hace uso de la biblioteca lcd.

tpManager: Esta biblioteca se encarga de proporcionar una interfaz desde la que se pueda saber en que elemento de la pantalla se ha pulsado(por ejemplo en el tablero, en el botón de pasar, fuera de la pantalla). Su implementación depende del diseño de las pantallas.

reversi: Esta biblioteca contiene el juego del reversi y la maquina de estados principal del programa. Desde esta, se van utilizando el resto de bibliotecas.

main: Este fichero se encarga de inicializar los diferentes componentes, cambia de modo supervisor a modo usuario y proceder a lanzar el juego.

**Descripción de los diferentes componentes del proyecto**

**Biblioteca de dibujado de pantallas**

Debido a que el dibujo y diseño de las pantallas es algo que consume mucho tiempo, para poder desarrollar la biblioteca lcdManager sin acceso a la placa, se desarrollo una biblioteca que sustituye a la biblioteca lcd proporcionando la misma interfaz, que permite crear una imagen con lo que se mostraría en el lcd.

Para hacer uso de esta biblioteca, se ha de estar sin placa y sin el emulador de arm, compilando de forma nativa para el ordenador desde el que se va a ejecutar.

La imagen resultante está en formato ppm. Esta puede ser visualizada de forma nativa desde algunas distribuciones linux como Ubuntu, en cambio en otros sistemas operativos como Windows se necesita de programas externos para visualizarla.

**Biblioteca timer 2**

La biblioteca timer 2, se encarga de controlar un timer que posea la máxima precisión posible. Para lograrlo, se utilizó una configuración en la cual se estableció un nivel de preescalado de 0 y un valor de divisor de ½. Esto produce que el contador se decremente con una frecuencia de 32MHz.

Para evitar un gran número de interrupciones, se estableció el valor inicial de la cuenta en 65535(máximo valor permitido), y el valor final de la cuenta en 0. De esta forma llega una interrupción cada 2.05 milisegundos.

Cada vez que llega una interrupción, se aumenta un contador y en el momento que se pide leer el tiempo transcurrido, se devuelve el valor de ese contador multiplicado por 65535 más el valor de la cuenta del timer en ese momento, todo ello dividido entre 32. Esta división se produce debido a que como la frecuencia con la que se decrementa el contador es de 32 Mhz, esto implica que cada 1/32 de microsegundo el contador es decremnetado. Por ello, el número total de decrementos que ha sufrido el contador, dividido por 32 es el número total de microsegundos que ha pasado.

Para tener la máxima precisión posible, antes de devolver el valor en cuestión se comprueba si el contador ha aumentado. Si es así, se devuelve el aumento del contador.

**Biblioteca de tratamiento de excepciones**

Para realizar la identificación y gestión de excepciones, se utilizan 3 funciones. Dos de ellas son utilizadas para realizar la identificación y la tercera es usada para la gestión de estas.

La primera de ellas será invocada una vez que salta un prefetch abort y la segunda lo es con el resto de excepciones. El uso de dos funciones para la identificación de excepciones es debido a que un prefetch abort y un data abort producen el mismo modo de ejecución, por lo que son indistinguibles una vez se han producido. Por ello es necesario que la rutina de interrupción de una de ellas sea diferente. En este caso se decido que el prefetch abort sea diferente ya que en el entorno que se está ejecutando, esta no se va a producir.

En estas funciones de identificación también se identifica la instrucción causante mediante el link register y se llama a otra función que es la que se encarga de gestionar la excepción.

Al gestionar la excepción, lo que se realiza es el guardado de esta en la pila de depuración y se entra en un bucle infinito en el cual se hace que parpadeen los leds y el 8 led, para de esta forma parar la ejecución del programa y avisar al usuario.

**Carga del código en la memoria RAM**

Para conseguir que el código se cargue en la RAM, desde el fichero de inicialización de la placa, primero se copia todo el programa a la región que el compilador trabaja como si fuera inicio de código en la RAM (Image\_RO\_Base).

El compilador piensa que el programa se encuentran ubicado entre Image\_RO\_Base y Image\_ZI\_Base, haciendo todas las referencias en base a estas direcciones. Por ello se copia toda la memoria RAM desde la dirección 0x0, (dirección en la que se encuera el programa al ser cargado desde la flash), a la dirección Image\_RO\_Base. Como el programa tiene un tamaño de Image\_ZI\_Base - Image\_RO\_Base, la memoria se irá copiando hasta el momento en el que se vaya a escribir sobre la dirección Image\_ZI\_Base, que se parará de copia.

Más tarde se inicializará a cero la región de memoria ubicada entre Image\_ZI\_Base y Image\_ZI\_Limit.

Por último se salta a la dirección de RAM donde se ha copiado el código para seguir la ejecución de este pero desde ahí.

**Biblioteca tp y calibración de la pantalla táctil.**

Para calibrar la pantalla táctil se exige al usuario nada más empezar el juego que pulse sobre los 4 bordes del tablero. De esta forma se puede obtener el sistema de coordenadas en el que actúa el touchpad. Esto es almacenado y cada ves que se realice una pulsación, se realiza un cambio de coordenadas del sistema que utiliza el touchpad al sistema que utiliza el lcd.

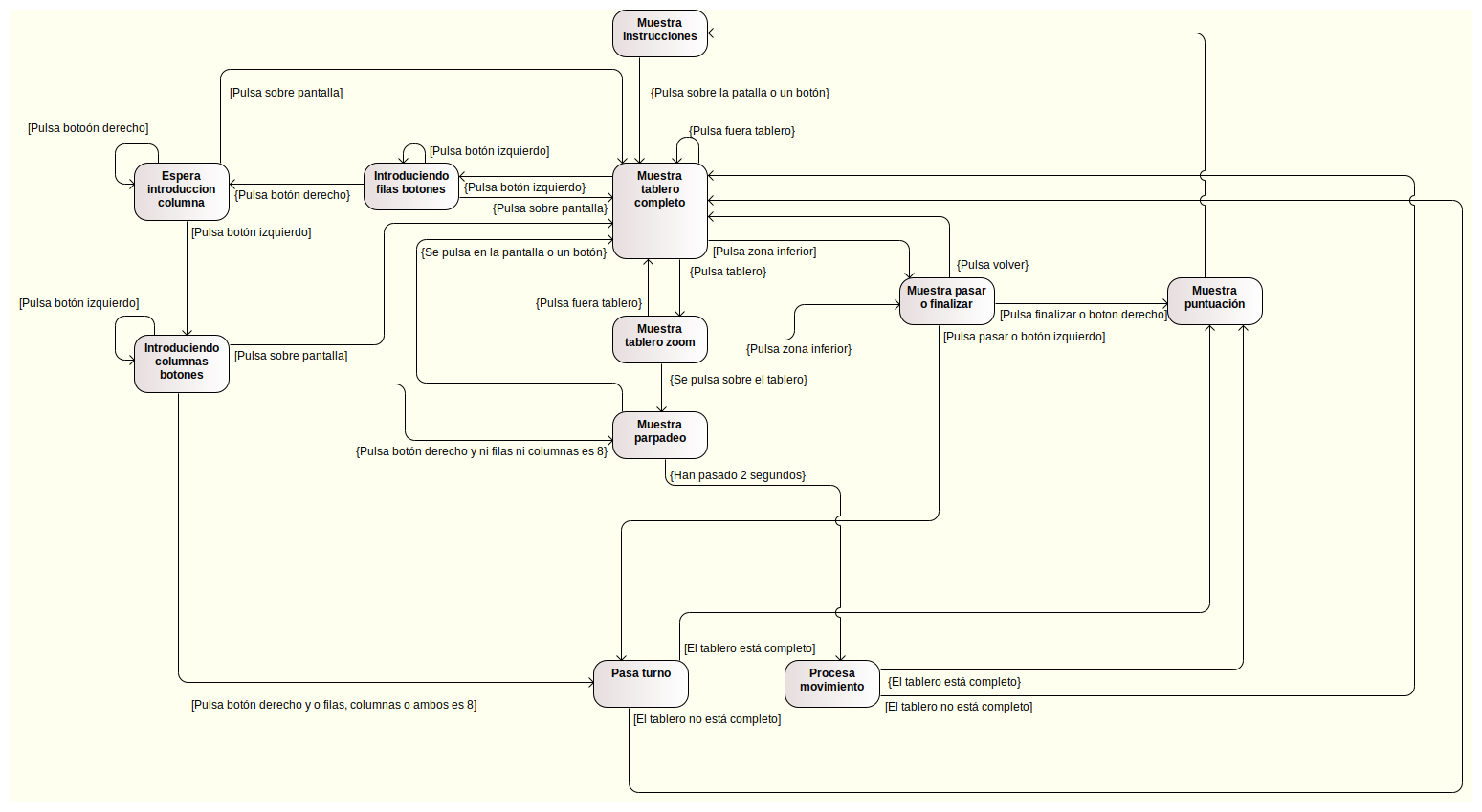
**Gestión entrada salida**

En el proyecto, el control de la entrada salida se produce en 3 módulos diferentes, los cuales son button\_rebotes, lcd y tp. Aunque la concurrencia entre el uso del touchpad y el uso de los botones se gestiona en una máquina que se encuentra en el módulo principal del juego.

El módulo button\_rebotes es el módulo que controla la gestión de los botones incluyendo los rebotes de los mismos, así como también es el que sirve de interfaz para saber si se ha pulsado algún botón y cual. Para gestionar los rebotes, se utiliza una máquina de estados la cual tiene 4 estados (esperando, gestionando rebotes iniciales, mantenido y gestionando rebotes finales). Se está en el estado esperando mientras no hay ninguna pulsación. En el momento que existe una pulsación se pasa por un estado transitorio llamado botón pulsado, en el cual se inicializa el timer que controla los rebotes y se deshabilitan las interrupciones de botones y se pasa directamente el estado gestionando rebotes iniciales. En este estado se mantiene durante 200 milisegundos, tras los cuales se vuelven a habilitar las interrupciones de botón y se realiza una encuesta periódica a este para comprobar si se ha soltado. Cada cierto periodo de tiempo que el usuario mantenga el botón pulsado, se registra en la interfaz una nueva pulsación, para de esta forma poder cumplir el requisito de que una pulsación larga equivaliese a varias pulsaciones. Una vez que se detecta que se ha soltado el botón se procede a desactivar de nuevo las interrupciones de botón, y tras 200 milisegundos, en los cuales se producirán los rebotes finales, vuelve al estado de espera.

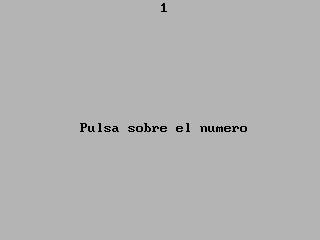
Para realizar la interacción con el lcd se utiliza el módulo lcd, aunque para el dibujado de las pantallas se utiliza el módulo lcdManager que hace uso de lcd. Este, está creado de tal forma que solamente se redibujarán las zonas de pantalla que se actualizan. Con esto se evita en situaciones como mostrar la cuenta del tiempo de juego o mostrar una ficha parpadeando, redibujar la pantalla completa.

La interacción con el touchpad se realiza en el módulo tp, el cual se encarga tanto de calibrar como de transformar las coordenadas que se producen en el tp a un formato que sea más fácil de integrar con el proyecto (formato de coordenadas que utiliza el lcd).

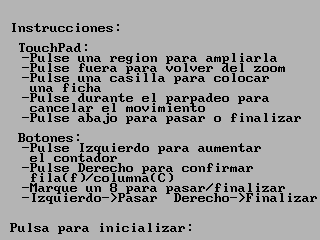
Para gestionar la concurrencia en el uso de la entrada salida en el proyecto, se utiliza una máquina de estados principal, en la cual cada estado corresponde con una pantalla a dibujar. En ella se contempla la posibilidad de utilizar el touchpad y los botones a la vez, por lo que en cada estado existen transiciones en el caso de que se haya pulsado un botón o en el caso de que se haya pulsado el touchpad. Las transiciones se producen según la máquina de estados de arriba.

**Diseño de las pantallas**

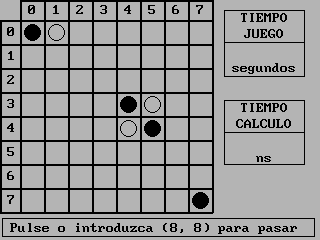
Calibrar:

Esta es la primera pantalla que se muestra en el juego, y se utiliza para calibrar el touchpad.

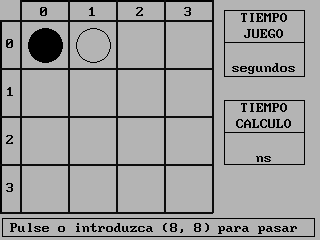
Instrucciones:

Después de calibrar el touchpad y antes de cada partida, se muestra esta pantalla, en la cual se muestran las instrucciones del juego.

Tablero completo:

En esta pantalla se muestra el tablero sin estado de zoom.

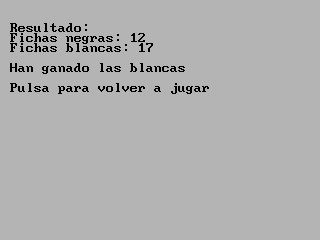
Tablero Zoom:

En esta pantalla se muestra el tablero con estado de zoom.

Pasar o finalizar:

Esta pantalla se muestra cuando el usuario introduce algún 8 mediante los botones o cuando pulsa sobre la región inferior de la pantalla, en los estados en los que se muestra el tablero.

Puntuación:

Esta es la pantalla que aparece cuando se finaliza una partida

**Comparativa en el rendimiento entre las diferentes versiones desarrolladas en la práctica 1**

Tiempos obtenidos en la ejecución de las diferentes combinaciones sobre la placa de desarrollo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | O0 | O1 | O2 | O3 | Os |
| C-C | 29 | 17 | 15 | 15 | 17 |
| C-ARM | 22 | 17 | 16 | 16 | 17 |
| C-THUMB | 23 | 18 | 17 | 17 | 19 |
| ARM-C | 25 | 14 | 13 | 13 | 14 |
| ARM-ARM | 18 | 14 | 13 | 13 | 14 |
| ARM-THUMB | 19 | 15 | 14 | 14 | 15 |

Como se puede comprobar, al ejecutar las diferentes versiones sobre la placa de desarrollo, con nivel de optimización 03, la combinación que mejor rendimiento ofrece es ARM-ARM. Por otro lado se puede ver cómo no existe diferencia en el rendimiento entre las versiones compiladas con nivel de optimización 2 y las compiladas con nivel de optimización 3 (Aunque se comprobó que el código ejecutado era diferente).

Aunque la combinación que ofrece mejor rendimiento es ARM-ARM, para la versión final del proyecto se decidió utilizar la versión C-C, debido a que de esta forma se limitaba menos al compilador a la hora de realizar optimizaciones sobre el proyecto en su conjunto, que con la combinación ARM-ARM si que se hubiese producido.

Tiempos obtenidos en la ejecución de las diferentes combinaciones sobre la simulador.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | O0 | O1 | O2 | O3 | Os |
| C-C | 7.1 | 4.8 | 2.7 | 0 | 4.87 |
| C-ARM | 5.5 | 4.8 | 4.6 | 2.6 | 4.9 |
| C-THUMB | 6.1 |  |  |  |  |
| ARM-C | 6.3 |  |  |  |  |
| ARM-ARM | 4.7 |  |  |  |  |
| ARM-THUMB | 5.3 |  |  |  |  |

Se puede comprobar como al ejecutar las diferentes versiones sobre el simulador la diferencia de rendimiento entre las versiones compiladas con nivel de optimización 2 y las compiladas con nivel de optimización 3 si que varía a diferencia de cuando se ejecuta sobre la placa.

Número de instrucciones que se ejecutan según las diferentes combinaciones.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | O0 | O1 | O2 | O3 | Os |
| C-C | 68 | 44 | 20 | 2 | 40 |
| C-ARM | 48 | 48 | 27 | 33 | 46 |
| C-THUMB | 53 |  |  |  |  |
| ARM-C | 55 |  |  |  |  |
| ARM-ARM | 33 |  |  |  |  |
| ARM-THUMB | 39 |  |  |  |  |