PRÁCTICA 3: DISEÑO DE UNA PLATAFORMA INDEPENDIENTE

OBJETIVOS:

El objetivo principal de esta práctica es finalizar el proyecto en el que habéis estado trabajando para conseguir un sistema empotrado autónomo de los ordenadores del laboratorio con el que se pueda jugar directamente. Para ello vais a:

- Utilizar la pantalla LCD táctil para visualizar e introducir movimientos en el tablero.
- Cargar vuestro código en la memoria Flash de la placa mediante el estándar JTAG, de forma que al encenderla se pueda jugar sin necesidad de conectarse ni descargar el programa.

ESTRUCTURA DE LA PRÁCTICA:

La práctica consta de cuatro apartados obligatorios:

A. JUEGO COMPLETO Y PANTALLA LCD

Vamos a dotar al juego de una interfaz gráfica. Para ello vamos a introducir dos periféricos nuevos: la pantalla LCD para mostrar información y la pantalla táctil o TouchPad (TSP) para validar la jugada.

Para realizar este apartado tenéis disponible en Moodle un guion de prácticas en el que se explica cómo funciona el LCD y un ejemplo en el que se utiliza una biblioteca con funciones básicas del LCD y del TSP. Por un lado, os facilitamos un fichero **`main.c`** con ejemplo de uso del LCD y TSP que también hace uso de la línea serie (opcional). Además se suministran ficheros ya existentes en vuestros proyectos que añaden funcionalidad nueva, son los ficheros facilitados por el fabricante de la placa, que pueden tener bugs o malas prácticas. Deberéis estudiarlos con sumo cuidado y decidir que hacéis. Como siempre, tratad de entender lo que uséis porque os podrá evitar muchos errores.

Mostrar información en la pantalla: Las bibliotecas del LCD dibujan sobre una zona de memoria RAM principal a la que llamaremos ***frame buffer***. Cuando queráis actualizar el contenido en el LCD se debe transferir del *frame buffer* a la memoria interna del LCD. Esta operación se realiza programando un nuevo dispositivo denominado **DMA** (*Direct Memory Access*). El procesador programa la operación del DMA (@origen y destino y la cantidad de memoria a transferir) y queda libre para realizar otras tareas a la espera de que el DMA acabe. Al acabar, el DMA interrumpe la ejecución del procesador para notificárselo.

Actualmente la biblioteca tiene una espera activa. Deberéis modificar la biblioteca para eliminar esa espera activa y hacer que el DMA inserte un evento de fin de transferencia ***`ev_finLCD`*** (similar a los eventos de la práctica anterior). Mientras se transfiere, el procesador puede dormirse o realizar otras tareas, pero no debe pintar sobre el *frame buffer* (ya que todavía se está copiando su contenido y podríamos mostrar parte de la pantalla vieja y parte de la nueva.

Los ficheros del LCD de bajo nivel tienen primitivas genéricas para ayudar a pintar. Podéis añadir primitivas nuevas a ese fichero, pero solo genéricas que puedan ser útiles para más programas. Las funciones específicas relacionadas con el juego se deben poner en ficheros aparte.

En la versión definitiva se debe jugar tanto con los botones como utilizando la pantalla táctil. Para ello, debéis tener en cuenta los siguientes requisitos:

1. Al encender la placa, se mostrará una **pantalla inicial** con las instrucciones básicas para jugar y la leyenda "Toque la pantalla para jugar". El sistema quedará a la espera de que se pulse sobre la pantalla táctil o se pulse un botón para continuar.
2. A continuación, a la derecha del LCD aparecerá el **tablero**, que será un cuadrado del máximo tamaño posible para que se vea entero en el **LCD**. Las columnas y las filas estarán numeradas (1...8).
3. Durante el turno del jugador humano, se mostrará una ficha en **gris** en el tablero. La ficha se moverá lateralmente (columnas) con el botón izquierdo y verticalmente (filas) con el botón derecho. Si se llega al final del tablero se dará la vuelta empezando de nuevo (*round-robin*). Para validar la jugada, una vez el usuario se coloque sobre la casilla elegida, se deberá pulsar el TSP. Si el usuario elige una casilla no valida (ya hay ficha o no se voltea ninguna del contrario) se entenderá que "pasa" y no puede realizar ningún otro movimiento.
4. Debajo del tablero aparecerá la **leyenda**: "Pulse para jugar".
5. Se mostrará a la izquierda de la pantalla la fila y la columna, el **tiempo total** transcurrido en la partida en segundos. También se mostrará información de "*profiling*", en este caso queremos poder ver en microsegundos el tiempo acumulado invertido en los **cálculos** de vuestro programa (no incluye el tiempo dedicado por el usuario a pensar e introducir un valor), el tiempo acumulado en la ejecución de la función **`patron_volteo`**, y el número de veces total que ha sido invocada. **Nota:** Se debe ejecutar el programa en el máximo nivel de optimización y usando la función de **`patron_volteo`** más eficiente (Práctica 1).
6. Cuando se acabe la partida (tablero completado o terminada prematuramente por el usuario), se visualizará el resultado y la leyenda "Toque la pantalla para jugar". **Lógicamente si el usuario pulsa comenzará una nueva partida**.
7. Podéis añadir cualquier otra mejora de jugabilidad que os parezca interesante, siempre que respetéis los puntos anteriores. Por ejemplo, poner un botón en pantalla para mostrar de nuevo las instrucciones o para pasar o finalizar la partida.

Nota: El TSP puede generar rebotes al igual que los botones, ya sabéis como solucionarlos). Es necesario calibrar un mínimo la pantalla táctil si se desea implementar botones en el LCD. Diseña un método inicial de calibración.

B. MODO USUARIO

La ejecución de todo el programa hasta ahora se está realizando mayoritariamente con el procesador en **modo supervisor**. Realiza las modificaciones necesarias para que el reversi **se ejecute en modo usuario**. Para ello se debe cambiar el modo del procesador a usuario antes de llamar a la función ***reversi_main***. Ciertas funciones de E/S se ejecutarán en modo IRQ, FIQ, o supervisor (system) allí donde sea necesario pero retornaran la ejecución a modo usuario tras la ejecución de la parte privilegiada. Debeis prestar especial atención a la pila del modo de usuario y la pila de depuración creada en la práctica anterior.

C. FIQ, FAST INTERRUPTS

Los procesadores ARM ofrecen dos tipos de interrupción y no los estamos aprovechando. Para hacerlo, vamos a darle al temporizador que uséis para leer el tiempo la máxima prioridad utilizando para ello una interrupción **FIQ**. Las FIQ se usan de manera muy similar a las IRQ:

- Repasad de nuevo los registros del controlador de interrupciones para saber qué son y cómo se habilitan y gestionan las FIQ. Fijaos que hay registros distintos para las IRQ y las FIQ.
- Tened en cuenta que las FIQ no soportan el modo autovectorizado en el que cada fuente de interrupción tiene su propia rutina de servicio de interrupción (RSI). En su lugar hay que usar la RSI genérica de las FIQ (**pISR_FIQ**). Si tuviésemos varios dispositivos capaces de generar una FIQ habría que hacer una encuesta para identificarlos. En este caso no es necesario porque sólo habrá una fuente de interrupción FIQ.

D. PLATAFORMA AUTÓNOMA

Queremos que nuestra plataforma sea autónoma y se pueda utilizar sin conectarse a ningún PC. Para ello vamos a cargar nuestro programa en la memoria Flash de la placa utilizando *openOCD y JTAG*. Los pasos a seguir son los siguientes:

1. Debemos generar el binario a escribir en la memoria Flash a partir del fichero `.elf`. Esto se hace con la utilidad de gcc `objcopy`. Lo podemos hacer desde la línea de comandos como en el siguiente ejemplo:

   ```
   arm-none-eabi-objcopy -O binary prueba.elf prueba.bin
   ```

2. A continuación, volcaremos el binario a la Flash con el siguiente comando (openocd está en "C:\Programas-Practicas-ISA\EclipseARM\openocd-0.7.0\bin"):

   ```
   openocd-0.7.0.exe -f test/arm-fdi-ucm.cfg -c "program
   prueba.bin 0x00000000"
   ```

Al encender la placa de nuevo, nuestro código estará almacenado en la Flash, pero allí no podemos ejecutarlo correctamente porque no funcionarán las escrituras y porque las direcciones reales no cuadrarán con las que ha utilizado el linker. Así que debemos copiarlo a la RAM y a partir de ahí podremos jugar al reversi como antes.

Para ello hay que añadir el código que: (1) inicialice el controlador de memoria (que antes estaba dentro de un script que se ejecutaba al conectarse a la placa), (2) copie el contenido de la ROM a la memoria RAM al comienzo de la ejecución, y, por último, (3) salte al Main recién copiado en la RAM. Esta nueva funcionalidad estará dentro de la rutina de tratamiento del `reset`. En moodle encontraréis un ejemplo que muestra cómo se hacen estos pasos. Ese ejemplo incluye otras funcionalidades que no son necesarias porque las hace la función de inicio de nuestro proyecto que seguiremos utilizando. Los pasos a seguir son:

1. Estudiar el ejemplo disponible en moodle
2. Identificar el código que inicializa el controlador de memoria (**tanto las instrucciones como los datos guardados en "*SMRDATA*"**)
3. Identificar el código que copie el contenido de la ROM a la memoria RAM al comienzo de la ejecución. Este código utiliza dos etiquetas definidas en el linker script. **Debéis entender qué hace con ellas.**
4. Añadir esos códigos a vuestro `44binit`, justo en el mismo lugar en el que estaban en el ejemplo (en `Resethandler`, tras desactivar las interrupciones)
5. Después tenéis que eliminar la parte del `44binit` que actualizaba el controlador de memoria, porque ya lo hace el código que habéis añadido
6. Al cargar vuestro código en la memoria flash y resetear podéis observar cómo la ejecución empezará en las direcciones de la memoria flash y al llevar al salto al Main pasaréis a las direcciones de la RAM. A partir de ahí vuestro código se ejecutará como siempre
7. **Antes de devolver la placa, debe ser re-escrito el código por defecto.** Debéis restaurar el fichero backup_flash.bin original, disponible en Moodle.

APARTADOS OPCIONALES:

1. Como apartado opcional podéis utilizar otros métodos para interactuar con el humano, tales como el teclado o uno de los puertos serie de las placas para enviar los movimientos desde un PC del laboratorio.

   Tendréis disponibles todas las bibliotecas de la placa, y se valorará el uso de cualquier periférico que aporte valor al dispositivo.

2. Diseñar un esquema que permita hacer simultáneamente los cálculos del juego con el refresco de pantalla. Por ejemplo, actualice el LCD tras la jugada humana y en paralelo se esté realizando el cómputo de la jugada del ordenador.

MATERIAL DISPONIBLE:

En la página de la asignatura en Moodle podéis encontrar el siguiente material de apoyo:

- Un guion dónde se explica el funcionamiento del LCD.
- Códigos fuentes para usar el LCD con algunas funciones básicas.
- Un guion en el que se explica cómo programar la memoria flash.

- Códigos fuente con un ejemplo en el que se programa la memoria flash de la placa.
- Para los apartados opcionales:
  - Un guion adicional que explica el puerto serie
  - Códigos fuente con las funciones básicas.

EVALUACIÓN DE LA PRÁCTICA

Está práctica se presentará el 19 de diciembre. La memoria el viernes 17 de enero.
La evaluación de esta práctica y la presentación será individual (la memoria es única y debe incluir la práctica 2).

ANEXO 1: REALIZACIÓN DE LA MEMORIA FINAL

Esta memoria incluirá el trabajo realizado en las prácticas 2 y 3. La memoria de las prácticas tiene diseño libre, pero debe incluir los apartados própios de una memoria técnica (recomendable revisar el documento Redacción de una Memoria Técnica, disponible en la web de la asignatura). Es obligatorio que incluya los siguientes contenidos:

1. Resumen ejecutivo (una cara como máximo). El resumen ejecutivo es un página independiente al inicio de la memoria que describe brevemente qué habéis hecho, por qué lo habéis hecho, qué resultados obtenéis y cuáles son vuestras conclusiones.
2. Explicación de la estructura de vuestro proyecto.
3. Descripción de la gestión de la entrada/salida de la versión final de vuestro proyecto incluyendo los diagramas de estados pertinentes en el que se vea cómo funciona, detallando que acciones se realizan en cada estado y como se genera la salida correspondiente en la pantalla.
4. Describe la gestión de excepciones.
5. Explicar cómo conseguís que vuestro código se cargue en la memoria RAM.
6. Método de calibración de la pantalla táctil.
7. Código fuente comentado (sólo el que habéis desarrollado nuevo y sea relevante de las prácticas 2 y 3). Como siempre, cada función debe incluir una cabecera en la que se explique qué hace, qué parámetros recibe..., ese apartado, varlorad su inclusión como anexo. Todo el código fuente lo debéis adjuntar con extensión .txt en el fichero zip en la entrega final.
8. Descripción de los problemas encontrados en la realización de la práctica y sus soluciones.
9. Resultado final.
10. Conclusiones valorando vuestro proyecto final.
11. Las correcciones solicitadas, en su caso, en las entregas previas.

Se valorará que el texto sea **claro y conciso,** con el correspondiente rigor técnico. Incluir en la memoria las partes de vuestro código que habéis desarrollado para cada

sección. Cuánto más fácil sea entender el funcionamiento del código y vuestro trabajo, mejor.

ANEXO 2: FORMA DE ENTREGA DE LA MEMORIA

La forma de entrega de la memoria será idéntica a las anteriores (revisar el documento de cómo escribir una memoria técnica). Debéis entregar la memoria en formato pdf junto con las fuentes originales (texto plano, extensión .txt) en un fichero comprimido en formato zip. Se mandará un único fichero por pareja con el siguiente nombre:

p2-3_NIP-Apellidos_Alumno1_NIP-Apellidos_Alumno2.zip

Por ejemplo: p2-3_345456-Gracia_Esteban_45632-Arribas_Murillo.zip

El archivo que contiene la memoria debe llamarse p2-3_memoria_NIP-Apellidos_Alumno1_NIP-Apellidos_Alumno2.pdf