Práctica 2

Estructura de Computadores

Modos de ejecución, gestión de excepciones y entrada/salida

Índice general

2.1. Objetivos de la práctica

En esta práctica completaremos nuestro estudio del procesador ARM7TDMI analizando sus modos de ejecución, sus excepciones y su sistema de entrada/salida, que gestionaremos mediante interrupciones. Los principales objetivos de la práctica son:

- Conocer los modos de ejecución del procesador.
- Entender, conocer y saber manejar el sistema de E/S por interrupciones.
- Conocer y saber manejar dispositivos básicos como leds, pulsadores, displays y temporizadores.

En esta práctica abordaremos el estudio del sistema de E/S de forma progresiva. Primero aprenderemos a realizar una E/S programada por espera de respuesta utilizando los dispositivos más sencillos. Después analizaremos el sistema de interrupciones nativo del ARM7TDMI y sus limitaciones, incluyendo para ello un dispositivo muy importante como son los temporizadores.

2.2. Lecturas previas obligatorias

Es imprescindible leer los capítulos 1 a 8 del documento titulado *Descripción del sistema de memoria y de entrada/salida en la placa S3CEV40* ([TPGb]) publicado a través del Campus Virtual. Sin leer y comprender completamente ese documento, es absolutamente imposible hacer esta práctica. Puedes consultar toda la documentación en [arm] y [um-].

2.3. Mapa de memoria de un programa de prácticas

Los detalles del sistema de memoria del S3CEV40 se detallan en el capítulo 2 del documento [TPGb] (lectura obligatoria).

De aquí en adelante vamos a utilizar un mapa de memoria muy similar para todos los programas, que está ilustrado en la figura 2.1. Como vemos tendremos varias regiones diferentes en el mapa de memoria de un programa, que son:

- Región de código
- Región de datos
- Región de heap (nosotros no usaremos, pero se colocaría ahí)
- Región de pilas
- Tabla de ISRs o RTIs

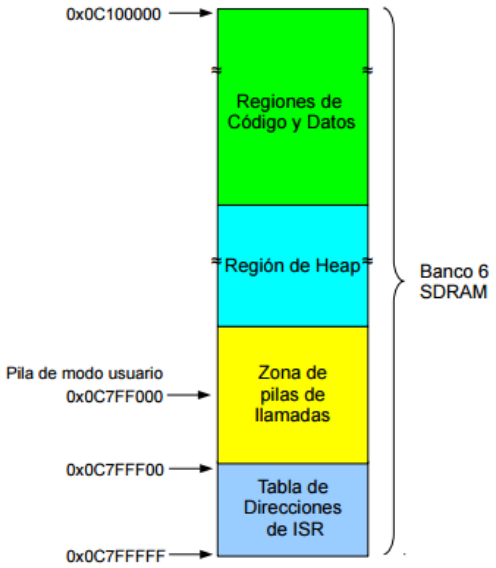

Figura 2.1: Mapa de memoria utilizado para los programas de las prácticas.

Como vemos, son direcciones que corresponden todas al banco 6 del controlador de memoria, en el que está ubicado el único chip SDRAM de la placa S3CEV40.

La tabla de ISRs tendrá una entrada por vector de interrupción, que se utilizará para almacenar la dirección de la rutina que debe tratar dicha interrupción (o excepción interna).

La región de pilas reserva espacio para ubicar las pilas de los distintos modos de ejecución del ARM7TDMI (generalmente inicializadas por el programa de test de la placa residente en la memoria flash).

La región de heap, de existir, es gestionada por la biblioteca del sistema (en este caso newlib) para asignar dinámicamente trozos de esta región por medio de llamadas a *malloc*. Nosotros no utilizaremos memoria dinámica en este laboratorio.

El espacio restante será utilizado para ubicar las distintas secciones de nuestro programa (**text**, **data**, **rodata** y **bss**) comenzando en la dirección más baja del banco 6.

Práctica 2

E/S programada con espera de respuesta

En esta primera parte vamos a manejar dispositivos de E/S sencillos, realizando espera activa para atenderlos. Concretamente vamos a trabajar con leds y pulsadores, gestionados a través de pines multifunción de los puertos B y G del GPIO (consultar el capítulo 5 de [TPGb]); y el display de 8 segmentos, ubicado en el banco 1 (consultar el capítulo 6 de [TPGb]).

Haremos un sencillo programa que permitirá usar dos pulsadores (botones) para realizar las siguientes acciones:

- **Encender/apagar leds:** Cada uno de los botones tendrá asociado un led que cambiará su estado (apagado/encendido) a cada pulsación del botón.

- **Mover un segmento del Display de 8 segmentos** (haciéndolo girar bordeando el símbolo 0)**:** Uno de los botones (botón1) se usará para parar o activar el movimiento del segmento y el otro (botón2) se usará para cambiar la dirección de giro.

Los ficheros con los que trabajaremos en esta primera parte de la práctica son:

- **44b.h:** fichero de cabecera con definiciones de macros para facilitar el acceso a los controladores de los dispositivos de nuestro sistema.

- **gpio.h** y **gpio.c**: interfaz e implementación de funciones para el manejo de los puertos multifunción. El alumno deberá **implementar todas las funciones del fichero gpio.c**, siguiendo las indicaciones de los comentarios y consultando la documentación [TPGb, um-].

- **button.h** y **button.c**: interfaz e implementación de funciones para el manejo de los pulsadores. El alumno deberá **completar la implementación de la función read_button** utilizando el interfaz del puerto G definido en gpio.h.

- **D8Led.h** y **D8Led.c**: interfaz e implementación de funciones para el manejo del display de 8 segmentos. El alumno deberá **completar la implementación de las funciones D8Led_segment** y **D8Led_digit**.

- **leds.h** y **leds.c**: interfaz e implementación de funciones para el manejo de los leds. El alumno deberá **completar la implementación de las funciones leds_init** y **leds_display**, usando el interfaz del puerto B definido en **gpio.h**.

- **utils.h** y **utils.c**: interfaz e implementación de funciones auxiliares. En este caso sólo tenemos implementada una función para realizar una espera activa **Delay**. Esta función debe ser invocada en la fase de configuración (*setup*) con el argumento 0 para que se auto ajuste. Tras esta calibración el argumento a la función estará en unidades

de 0.1 ms. Así, por ejemplo, **Delay** (1000) esperaría 100ms. Este fichero está completo y no necesita ser modificado.

- **init.asm (init.S)**: fichero de inicialización. Contiene el símbolo *start*, es por donde comenzará la ejecución de nuestro programa. Su misión es realizar una configuración básica mínima y luego invocar la rutina *main* de nuestro programa. Entre otras cosas: configura las pilas de los distintos modos de ejecución, habilita las interrupciones e inicializa la región de datos sin valor inicial a 0. Este fichero está completo y no necesita ser modificado.

- **ld_script.ld**: script de enlazado que deberemos utilizar (igual que se hacía en las prácticas de FC).

- **main.c**: fichero con el código del programa principal. **Este fichero deberá ser codificado en su mayoría por el alumno, siguiendo las instrucciones que se darán a continuación.**

Para la codificación del programa principal seguiremos la estructura de un sketch de arduino, que tiene las siguientes funciones:

- **setup**: cuya misión es configurar el sistema para su correcto funcionamiento. Su misión principal será configurar los controladores HW de los dispositivos que vamos a manejar, que en nuestro caso serán:
  - Leds: debemos configurar el puerto B para que los pines 9 y 10 sean pines de salida. Esto lo haremos en la función **leds_init** por lo que setup sólo tendrá que invocar dicha función.
  - Pulsadores: debemos configurar el puerto G para que los pines 6 y 7 sean de entrada. Esto lo haremos utilizando el interfaz del puerto G definido en **gpio.h**.
  - Display 8 segmentos: no necesita configuración inicial. Si queremos, podemos encender el led en la posición inicial deseada.
  - Rutina **Delay**: debe ser invocada con el valor 0 para su calibración.

- **loop:** representa el cuerpo de un bucle infinito. Es la función principal de nuestro programa, que tendrá que codificar el alumno. Al final de la rutina se hará una espera activa de 200ms (con esto eliminaremos los rebotes de los pulsadores y tendremos una base para determinar la velocidad de giro del led). Antes de esta espera la función deberá:
  - Comprobar si se ha pulsado alguno de los pulsadores invocando la función **read_button.** Esta función devolverá un entero en el que los dos bits menos significativos indican si se ha pulsado o no alguno de los pulsadores. El módulo button exporta, además, las macros **BUT1** y **BUT2,** que pueden usarse para comprobar si se ha pulsado el botón 1 o el botón 2 (haciendo una *and* con el valor devuelto por **read_button**).

- Si se ha pulsado el pulsador 1, debe permutar el estado del led 1 (lo apagará si estaba encendido y lo encenderá si estaba apagado) y la dirección de giro del segmento en el display de 8 segmentos.
- Si se ha pulsado el pulsador 2, debe permutar el estado del led 2 y el estado de movimiento (en marcha o parado) del segmento en el display.
- Si el segmento no está detenido, decrementará un contador de las iteraciones que lleva en marcha y si se hace 0 deberá desplazar el segmento una posición y reiniciar dicho contador.

La función main del programa será por tanto:

```
int main(void)
{
      setup();
      while (1) {
            loop();
      }
      return 0;
}
```

Para facilitar la codificación de nuestro programa, el fichero main.c tiene declarada una variable global que es una estructura que representa el estado del *led rotante* sobre el display de 8 segmentos, cuyos campos son:

- **moving**: usada como variable booleana, a 0 si el led está quieto y a 1 si está rotando.
- **speed**: indica la velocidad actual del led en número de vueltas del bucle principal (aproximamos a que una iteración del bucle principal son 200ms, asumiendo que el tiempo de espera domina sobre el tiempo de cómputo).
- **iter**: contador de iteraciones del bucle principal desde el último movimiento del led. Se va decrementando en cada iteración del bucle si está en movimiento el led. Cuando llega a 0, el led se desplaza y este contador debe ser entonces inicializado al valor del campo **speed.**
- **direction**: variable que indica la dirección de giro del led rotante. Un valor 0 indica sentido antihorario y un valor 1 indica sentido horario.
- **position**: indica el segmento que debe estar encendido actualmente (codificado como una posición en el **array Segments** del módulo **D8Led**).

Bibliografía

[arm] Arm architecture reference manual. Accesible en http://www.arm.com/miscPDFs/14128.pdf. Hay una copia en el campus virtual.

[TPGa] Christian Tenllado, Luis Piñuel, and José Ignacio Gómez. Introducción al entorno de desarrollo eclipse-arm.

[TPGb] Christian Tenllado, Luis Piñuel, and José Ignacio Gómez. Descripción del sistema de memoria y de entrada/salida en la placa S3CEV40.

[um-] S3c44b0x risc microprocessor product overview. Accesible en http://www.samsung.com/global/business/semiconductor/productInfo.do?fmly_id=229&partnum=S3C44B0. Hay una copia en el campus virtual.