Instrucciones para escribir y testar el código

En Moodle podrás encontrar 2 ficheros con código SystemC.

El archivo **`codigoFFT-DF-completo.rar`** contiene la arquitectura, programas y archivos de testbench necesarios para realizar la transformada directa, filtrar e invertir 2 bloques de datos. **Se suministra sólo con fines informativos** y también porque se puede utilizar para montar el sistema final (sin embargo, no contiene el código utilizado para guardarSonido).

El fichero que os interesa ahora es el llamado `codigoModulos-DF.rar`. Cada grupo deberá modificar algunos ficheros de código para que implemente correctamente la parte que se le ha asignado. Esto se hará en 3 fases:

1. Modificar mainMIPS-DF.cpp para que el productor y el consumidor utilicen los archivos de testbench apropiados para vuestro módulo. Los archivos de testbench también se suministran. Modificar MIPScore.h para que SC_THREAD lance el fichero de código que se corresponde con vuestra parte. Compilar y comprobar que funciona.
2. Para la segunda parte, debéis sustituir el código en C de MIPScore.cpp por seudocódigo ensamblador. Compilad y simulad el programa, y depuradlo hasta que funcione.
3. Cuando vuestro código funcione, reescribirlo para que parezca auténtico código ensamblador (básicamente es eliminar los paréntesis y reescribir las llamadas a LW y SW). Grabad ese código en un fichero y utilizar el programa **`ensamblaMIPS`** (disponible en Moodle para Windows y Linux) para obtener el código máquina. Encontraréis instrucciones más detalladas más adelante.

Escribir pseudocódigo ensamblador

Resulta muy similar a escribir el código real con las siguientes salvedades y aclaraciones:

- Las instrucciones son en realidad macros, así que es necesario incluir los parámetros (registros, etc.) entre paréntesis.
- Para referirnos a un registro se puede utilizar la nomenclatura `$t0` o `$8` indistintamente. Pero si queremos leer o escribir directamente, la sintaxis correcta sería `Rt0`.
- Para utilizar `lw` o `sw`, se invoca a las funciones `LW` y `SW`. El primer parámetro es el registro donde se va a cargar el dato (si `lw`) o el que se va a guardar en memoria (si `sw`), y debe indicarse como `$t0`, `$t1`, etc. El segundo parámetro es la dirección (siempre múltiplo de 4). La dirección se especifica como un único número, pero debéis recordar que la versión final deberá especificar las direcciones como la combinación de un registro y un offset. Así que lo mejor es que especifiquéis las direcciones de la forma `Ra0 + 8` (por ejemplo). De esta forma no escribiréis código que luego no sepáis convertir a ensamblador real.
- Los saltos se escriben de forma casi idéntica a como se hace en ensamblador real, utilizando etiquetas dentro del código
- No existen directivas `.data` ni `.text`. El programa comienza en la dirección 0x0000, los datos internos en 0x0800, los datos que se van a procesar en 0x1000, los resultados en 0x2000, y la E/S mapeada en memoria en 0x3000. Como programadores, debéis usar registros y desplazamientos para apuntar a esas zonas de memoria.

- El mapeo de E/S es el siguiente: La dirección 0x3000 recibe datos del procesador anterior (o del productor). La dirección 0x3004 envía datos al procesador siguiente (o al consumidor). La dirección 0x3008 recibe datos del procesador siguiente (o del consumidor). La dirección 0x300c envía datos al procesador anterior (o al productor). Por último, la dirección 0x3010 sólo la utiliza el procesador de filtrado para recibir datos externos.

Convertir a código ensamblador real

Principalmente es necesario eliminar los paréntesis y puntos y coma. La única complicación se dará con `LW` y `SW`. Ya que habrá que sustituir `LW( $t0, Ra0 + 4)` por `lw $t0, 4($a0)` de forma manual.

Obtener el código máquina con el ensamblador externo

En Moodle podréis encontrar un programa compilado para Windows y para Linux que convierte un archivo de código ensamblador puro (sin directivas ni zona de datos) en código máquina al gusto del simulador. Se deben suministrar como argumentos los nombres de losl archivos fuente y destino.

Cargar y simular el código máquina en el simulador

Para estar seguros de que vuestro código máquina funciona, lo simularéis a nivel de instrucción en el simulador. El procedimiento es el siguiente:

- Modificar `mainMIPS-DF.cpp` para que el nombre del archivo de código sea el de vuestro código máquina. Así el simulador sabrá que tiene que cargarlo y ejecutarlo instrucción por instrucción.
- Modificar `MIPScore.h` y descomentar `SC_THREAD(procesar)`. Será necesario que comentéis el `SC_THREAD` que veníais utilizando, de lo contrario habrá 2 procesos en marcha intentando hacer lo mismo.
- Compilar y ejecutar el programa. Si se produce algún error habrá que depurarlo. El error podría surgir en el proceso "`procesar`" o ser una discordancia en los resultados. En el primer caso habría que poner un punto de ruptura y comprobar qué error se os ha colado en el código. En el segundo, habría que comprobar que el vuestro código ensamblador coincide con el que en su momento funcionó.