Informe de laboratorio 4 Electrónica Digital 2

Luis Eduardo Rodriguez

José David González

Abstract:
Se descargó y programó la FPGA Nexys 3 con el hardware del procesador LC3. Este se implementó de manera que tuviera un vínculo con un módulo de comunicación serial con un protocolo dado por el profesor. Dicho módulo de comunicación serial se comunica con una terminal llamada cutecom y a partir de esta terminal se le pasan los datos a la memoria del LC3 según su arquitectura para los registros KBRS, KBDR, DSR, DDR, procesarlos y devolverlos a la terminal por medio del serial, siendo el serial la pantalla y el teclado al mismo tiempo.
El programa en ASM para el lc3 se modificó de manera que se pudiera ejecutar en la FPGA debido a que esta no tiene las instrucciones de TRAP se necesitó implementar estas funciones además de implementar las demás funcionalidades faltantes y cambiar la manera como se ingresaban los datos, que solamente cuando se oprima enter se capturen los datos.

Implementación
Funciones nuevas:
GETC: Se carga en el registro R0 el valor del KDRS sólo cuando haya un valor, de lo contrario él sigue evaluando cuando ya hay un dato disponible.

```
LDI R0,KBRS1
BRZP GETCHAR1
LDI R0,KBDR1;
RET
```

OUT: Se almacena el carácter y se carga en R0 si la pantalla ya está disponible para recibir un dato y mostrarlo, luego de que pase eso se almacena el valor del carácter en R0 y este valor se escribe en DDR para ser mostrado

```
LDI R0,DSR1   ;
BRZP PUTCHAR21
LD R0,PCR01
STI R0,DDR1
RET
```

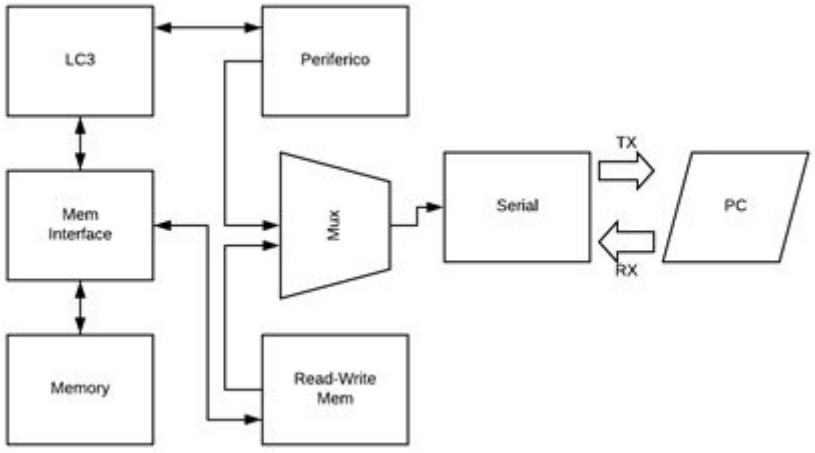
PUTS: Es un ciclo repetitivo de OUT, se aumenta la dirección de memoria que apunta al carácter y hasta que el valor que está almacenado en esta posición sea 0 se imprime en pantalla de la misma manera que el PUTS.

```
ST  R0,PMR01
   LDR R0,R0,#0
   BRz PUTSMSGE1
   ST R7,PMR71
   JSR PUTCHAR1
   LD R7,PMR71
```

```
    LD R0,PMR01
    ADD R0,R0,#1
    BRnzp PUTSMSG1
PUTSMSGE1    RET
```

Los demás TRAP no se implementaron en nuestro código ya que no era necesario.

Diagrama de flujo comunicación serial



Las demás funciones se implementan de la siguiente manera: Para el ordenamiento y para tener los valores máximos y mínimos se utilizó el mismo código de ordenamiento que en la práctica pasada, la única diferencia es en la manera en la que se leen los datos, primero se ordenaron los datos de manera ascendente como en la práctica anterior y dependiendo de la función que el usuario elija estos datos se leerán de manera ascendente o descendentemente, para escoger el menor número se elige el número que está en la primera posición del BLKW ordenado, mientras que para el elegir el mayor número se elige el número que está en la posición N del BLKW, siendo N el número de números que el usuario eligió que va a escoger.
Para los múltiplos se hizo un razonamiento igual al hecho en el laboratorio pasado, sólo que en lugar de hacer una AND con el 7 para hallar múltiplos de 8, se hace una AND con 3 para encontrar los múltiplos de 4 y una con 1 para encontrar los múltiplos de 2, si estas operaciones dan 0 entonces son múltiplos del número dado.

Conclusiones
Se nota que hay mayor libertad al trabajar en la LC3 de la FPGA en lugar del simulador, ya que se tienen disponibles todas las posiciones de memoria menos las 4 de comunicación con los periféricos lo que permite mayor capacidad de almacenamiento, claro está que se deben implementar las funciones del LC3 que necesitemos, ya que en la FPGA sólo están las instrucciones básicas(sin los TRAP), esto requiere un pequeño esfuerzo mental extra pero es un costo que se debe pagar por la libertad de uso.
Según el análisis anterior para nuestro programa hay disponibles FFFF-(Número de líneas del programa + 4) posiciones de memoria para los datos numéricos ingresados por el usuario. Esto se puede implementar poniendo el BLKW en el final del código y se accede a él cargando en un registro

el valor de la posición de memoria donde empieza y se hacen las demás funciones de manera normal. La primera barrera encontrada para utilizar toda esta memoria es que para hacer la validación de los datos se debe comparar constantemente con un valor y este valor puede tener un valor máximo de 32767, esto se puede hacer con una doble comparación, se compara primero un valor en un registro y luego por otro valor en otro, de esta manera por cada valor habrán 32767 números, por lo tanto en esta condición el número se elevaría a 32767*32767, eliminando el problema de la comparación.

Al utilizar maneras muy genéricas de codificar las funciones en la práctica 3 se simplifica enormemente el trabajo en esta práctica