Laboratorio 4 Electrónica Digital 2

1st Camilo Álvarez Muñoz *Facultad de ingeniería*
Universidad de Antioquia
Medellín, Colombia
camilo.alvarezm@udea.edu.co
2nd Juan Camilo Arteaga Ibarra *Facultad de ingeniería*
Universidad de Antioquia
Medellín, Colombia
camilo.arteaga1@udea.edu.co

***Resumen*—En este laboratorio se diseñó e implementó una versión simplificada del procesador ARM con ejecución de instrucciones en múltiples ciclos mediante el lenguaje de descripción de hardware SystemVerilog y el sistema de desarrollo DE10-Lite. En primer lugar se implementó un procesador ARM con ejecución de instrucciones de un solo ciclo y posteriormente se transformó a un procesador multi-ciclo capaz de ejecutar un programa de prueba desarrollado en el lenguaje ensamblador con ayuda de los periféricos de la FPGA.**

***Index Terms*—CPU, ALU, registros, ARM, multi-ciclo, mono-ciclo, datapath, control, módulos, procesador.**

I. Introducción

Una microarquitectura es la disposición específica de registros, ALUs (Unidades Aritmético-Lógicas), máquinas de estados finitos (FSMs), memorias y otros bloques lógicos necesarios para implementar una arquitectura. Para la arquitectura ARM hay microarquitecturas mono-ciclo, multi-ciclo y Pipelined.

Este documento contiene el reporte de laboratorio sobre el desarrollo de un procesador ARM en su versión mono-ciclo y su versión multi-ciclo. Estas dos micro arquitecturas son representaciones del procesador ARM que pueden brindar diferentes ventajas o desventajas en su implementación; estas fueron conocidas durante el desarrollo del curso y exploradas a fondo en el libro guía. Con ayuda del libro guía se crearon y adaptaron las unidades de datapath y Control necesarias para la implementación de cada procesador y se realizo un programa de prueba capaz de registrar datos de entrada enviados mediante los periféricos de la FPGA DE10-lite en donde se esperaba visualizar un resultado a la operación $3x - 5y + 2z$ en formato hexadecimal y decimal con signo haciendo uso de los displays siete segmentos de la tarjeta de desarrollo.

II. Diseño e implementación

II-A. Procesador mono-ciclo:

En un procesador mono-ciclo, todas las etapas de ejecución de una instrucción, como la búsqueda de la instrucción, la decodificación, la ejecución y el almacenamiento de resultados, se llevan a cabo en un solo ciclo de reloj. Esto significa que cada instrucción debería tardar la misma cantidad de tiempo en completarse, pero realmente este tiempo es limitado por la instrucción mas lenta.

Este tipo de diseño es relativamente simple y fácil de implementar, ya que no requiere de técnicas de pipeline o ejecución fuera de orden para mejorar el rendimiento. Sin embargo, también puede ser menos eficiente en términos de utilización de recursos, ya que las etapas del procesador pueden permanecer inactivas durante ciertos ciclos de reloj cuando no hay instrucciones que ejecutar. Otra clara desventaja del procesador mono-ciclo es el uso de dos memorias, una de instrucciones y otra de datos, el uso de estas dos memorias no es realista.

Para implementar este procesador mono-ciclo, estudiamos y usamos los códigos suministrados en la sección 7.6 del libro guía. Para la construcción de la unidad de datapath fue necesario crear un bloque de memoria de instrucciones (imem) de 64 registros (256 bytes) en la que se abre un archivo de formato ".dat"que contiene las 38 instrucciones necesarias para ejecutar el programa de prueba, donde cada instrucción es de 32 bits. Ademas de esto, fue necesario crear una memoria de datos de 256 bytes capaz de almacenar información y también recibir y manipular los datos leídos de los switches y botones, así como los datos llevados a los displays. Estos periféricos son accedidos por fuera de las 256 direcciones de memoria y fue extremadamente importante inicializar la dirección de memoria cero con un dato C0000000, de manera que un registro guardase ese valor para poder acceder a esa dirección de memoria que es a la cual, junto con las direcciones C0000004 y C0000008 , se les asignan el uso de los periféricos, lectura y escritura según sea el caso. La dirección de memoria C0000000 es la que se utiliza para la escritura en los displays siete segmentos, la C0000004 es la asignada para la lectura del botón y la C0000008 para la lectura de los switches. El mapeo de la memoria es descrita a detalle en la siguiente tabla:

Dirección (Hex)	Operación	Datos
00000000 a 000000FF	Lectura	$rd = RAM[a[31:2]]$
00000000 a 000000FF	Escritura	$RAM[a[31:2]] <= a$
C0000000	Lectura	$rd = 32'b0$
C0000000	Escritura	$deco\_displays\_seven\_segments <= wd[7:0]$
C0000004	Lectura	$rd = \{31'b0, P\}$, siendo P el valor del pulsador
C0000004	Escritura	No hay escritura ni en memoria ni en periféricos
C0000008	Lectura	$rd = \{24'b0, SW\}$, siendo SW el valor de los 8 suiches
C0000008	Escritura	No hay escritura ni en memoria ni en periféricos

Figura 1: Mapeo de la memoria: RAM y Periféricos

También fue necesario que creáramos una unidad aritmética lógica (ALU) para este procesador mono-ciclo. Este procesador solo hace uso de las instrucciones ADD,SUB, AND, OR, LDR, STR y Bxx, asi que reutilizamos la ALU creada para la CPU simple pero para esta ocasión solo se usaron las instrucciones de suma, resta, and y or; esto debido a que las instrucciones de lectura y carga de datos y las instrucciones de salto ya vienen implementadas en el procesador gracias a las señales en la unidad de control.

Para mostrar el resultado en los displays de la FPGA, fue necesario incluir el decodificador a 7 segmentos utilizado en el laboratorio 2.

Con todas estas consideraciones, este es el esquemático de procesador mono-ciclo implementado:

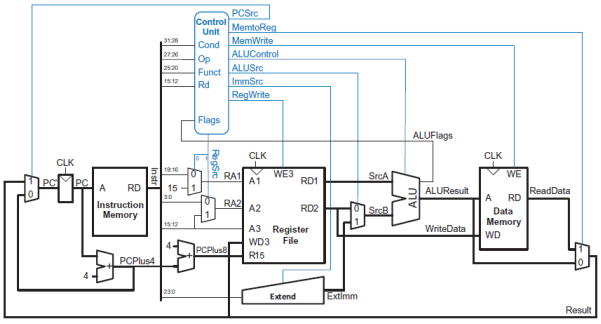

Figura 2: procesador mono-ciclo.

La unidad de control puede ser visualizada a mayor detalle en el siguiente esquemático:

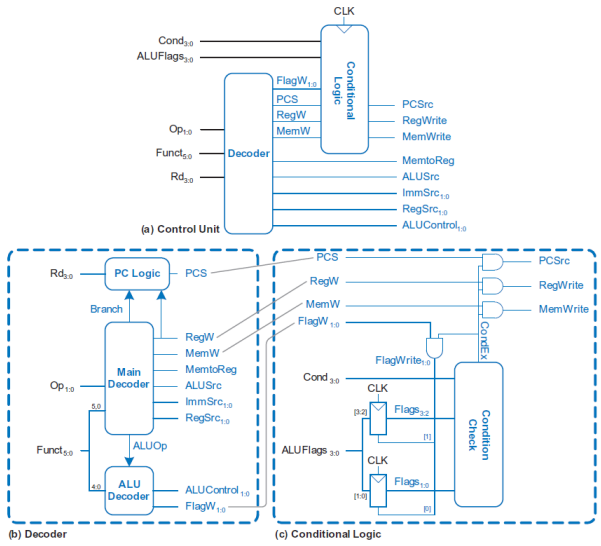

Figura 3: Unidad de control.

Para comprobar el funcionamiento de los módulos implementados y el correcto funcionamiento del procesador completo, incluimos el programa de prueba suministrado en la sección 7.6.3 del texto guía en el archivo .dat leído en el modulo ïmem".

Figura 4: señales de salida del simulador en mono-ciclo.

Al realizar la simulación del testbench propuesto por el libro, se imprimió en la consola del simulador ModelSIM un mensaje que indica la correcta simulación y ejecución de todas las instrucciones introducidas a la memoria, solo a través de este mensaje es posible verificar que el procesador creado funciona a la perfección. Es posible verificar que al ejecutarse la ultima instrucción del testbench (STR R2, [R0, #100]) el valor 7 será escrito en la dirección de memoria 32'h00000064 (100 en decimal).

Al comprobar el correcto funcionamiento del procesador implementado, procedimos a realizar el programa de prueba para la función $3x-5y+2z$. Para esto, se utilizó la herramienta de simulación CPUlator en su versión ARMV7, con este simulador es posible programar en lenguaje ensamblador. Debido a que no existen las instrucción MOV, se debe inicializar los registros necesarios en cero realizando una resta del registro 15 consigo mismo. En el procesador mono-ciclo no es posible utilizar la instrucción CMP, por lo tanto, para utilizar los Mnemonic condicionales es necesario utilizar la instrucción

SUBS. Al ingresar el programa de prueba a la memoria de instrucciones, se comprobó la correcta operación de la función $3x-5y+2z$ al ingresar los valores de X, Y y Z con los switches de la FPGA.

II-B. Procesador multi-ciclo:

El procesador mono-ciclo tiene algunas desventajas, como lo es el uso de dos memorias para los datos y las instrucciones, mientras que la mayoría de los procesadores de la actualidad cuentan con una única memoria externa que contiene las instrucciones y datos, por otro lado, se necesitan 3 adders (sumadores), los cuales suelen ser circuitos relativamente costosos. Y por último, se debe definir una frecuencia de reloj que cumpla con el tiempo necesario para soportar la instrucción que más tiempo requiere, la lectura de memoria (LDR).

Estos problemas los solucionamos dividiendo las instrucciones en pequeños pasos, de modo que las instrucciones más simples tomarán menor tiempo que las más largas. Para el diseño del procesador multi-ciclo se realizan los mismos procesos que para el mono-ciclo, definiendo en una primera instancia el datapath con cada uno de sus componentes y conexiones, con la diferencia de que se añaden registros que permiten mantener los datos entre cada estado. Luego se realiza la unidad de control, que es la encargada de definir múltiples señales que indican las operaciones a realizarse en la unidad del datapath, teniendo como base principal para esta unidad, una máquina de estados finita.

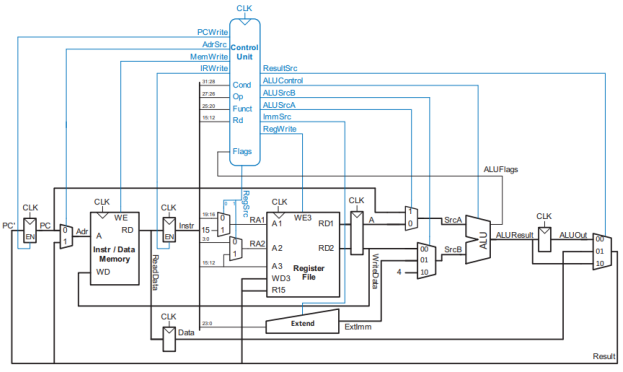

Figura 5: procesador multi-ciclo.

Partiendo de los componentes descritos en el libro para el procesador mono-ciclo (páginas 444 a 456), realizamos el montaje del procesador multi-ciclo, agregando componentes y conexiones a la unidad datapath, y cambiando el módulo de decodificación, que ahora estaría regido por la máquina de estados finitos mostrada a continuación:

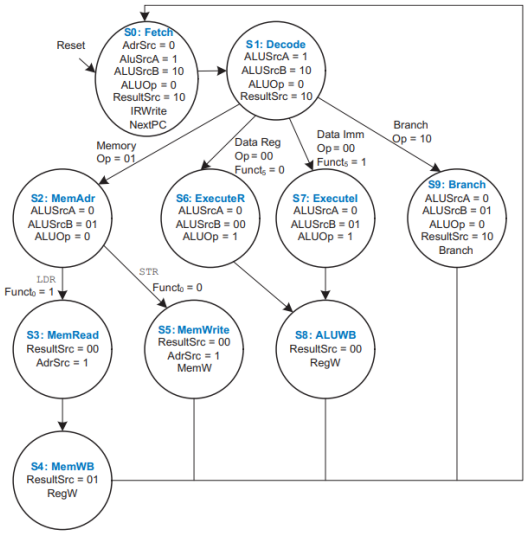

Figura 6: máquina de estados finita.

Cada señal definida en la máquina de estados permite controlar un proceso para la instrucción que se esté ejecutando, permitiendo así acotar el número de ciclos utilizados según sean necesarios. Cuando realizamos el módulo de la máquina de estados, no tuvimos en cuenta que a todas las señales se les debe dar un valor ”don’t care.[en] caso de no ser usada en alguno de los estados o procesos definidos, esto hizo que al momento de probar el funcionamiento del procesador aparecieran errores, ya que se debido a que lo definimos como un proceso no completamente combinacional, el software Quartus hiciera conexiones y asimilaciones de datos incorrectos, lo cual se corrigió posteriormente. También cambiamos algunas pocas condiciones en el módulo de la lógica condicional.

Por otro lado, en la unidad del datapath realizamos bastantes cambios, empezando por la unidad de memoria, que para ella, además de tener las direcciones de memoria propuestas por la guía de laboratorio para el uso de periféricos, se debía acondicionar para que almacenara las instrucciones a ejecutar y los datos que fuesen necesarios guardar en cada proceso. También la aparición de flip-flops, muxes, registros y un extensor de datos que permiten la buena ejecución de cada instrucción. La explicación del por qué cada uno de los módulos está descrita en el libro en la sección 7.4.1. Cuando realizamos la simulación del procesador, notamos que la unidad de datapath tenía errores de conexión y una mala definición de la bandera overflow de la ALU, lo cual nos tomó tiempo notar pero fue corregido.

Al simular y ejecutar el código realizado apreciamos las siguientes señales de salida para el testbench propuesto en el libro.

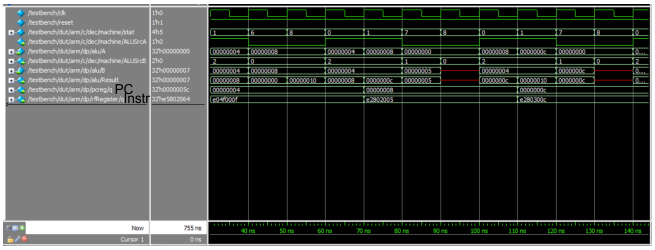

Figura 7: señales de salida del simulador.

Nosotros no implementamos las dos instrucciones CMP y MOV, sin embargo, sabemos que para la implementación de estas instrucciones es necesario modificar la ALU, describiendo su lógica interna, el tamaño de la señal ALUCntr, que le indica a la ALU la operación a realizar. También, modificar la unidad de control, con más precisión la máquina de estados y la lógica condicional para la salida de las señales de control que indiquen los procesos que se deben realizar y los cuáles no en caso de ser alguna de las dos instrucciones no implementadas. En el caso del programa de prueba en ensamblador, toda instrucción SUBS debe ser remplazada por la instrucción CMP, y de la misma forma toda instrucción que requiera inicializar una variable en un valor inmediato será reemplaza por MOV.

Para la práctica, también se pidió un decodificador siete segmentos que mostrara los valores en formato decimal en un rango de -128 a 127, para lo cual definimos un módulo que convirtiera los números hexadecimales que se usaban en la versión del procesador mono-ciclo a decimales con sino, usando las operaciones de módulo y división, lo cual arrojaría las centenas, decenas y unidades del número a observar en los periféricos de la tarjeta DE-10 Lite.

III. Conclusiones

- Para la realización de la práctica de laboratorio, parte 1 y 2, necesitamos al rededor de 32 horas.

- El uso del simulador del software Quartus nos permitió corregir muchos de los errores que se presentaron durante el desarrollo de la práctica, tal como lo fue una mala definición de la bandera de Overflow. Tambien hubieron múltiples errores en las conexión de señales en el datapath del procesador multi-ciclo.

- En muchas ocasiones es de gran ayuda realizar un testbench para cada uno de los componentes que hacen parte de un procesador diseñado, permitiéndonos observar los cambios que realiza cada una de las señales que influyen en el componente, a manera que cuando se deba hacer la unión de componentes, podamos tener la certeza de que cada uno por separado funciona correctamente.

- Debido a cuestiones de tiempo, no implementamos las instrucciones CMP y MOV en el procesador, pero tenemos totalmente claro el procedimiento necesario para incluir estas instrucciones tanto en el procesador como en el programa de prueba.

- Es importante conocer el tipo de instrucciones que deseamos utilizar en un procesador y su forma de codificación, para poder definir un sistema que soporte los datos sobre las instrucciones y opera con cada una de ellas de manera correcta y eficiente.

- El procesador mono-ciclo es facil de implementar y entender, pero los problemas relacionados con el tiempo de ejecución y el critical path hacen que la implementación de procesadores multi-ciclo sean mas factibles en la implementación de sistemas debido a su flexibilidad, reducción de costo y desempeño. La microarquitectura multi-ciclo reduce el costo del hardware considerablemente al reutilizar bloques de hardware costosos como sumadores y memorias y también al ejecutar algunas instrucciones en menos ciclos.

Referencias

[1] S. L. Harris and D. Harris, “Digital design and computer architecture: Arm edition,” 2015.