Laboratorio de Electrónica Digital II
Práctica No. 7: Procesador ARM *Pipelined*

Profesores
Luis Fernando Castaño L. (luis.castanol@udea.edu.co)
Luis Germán García M. (german.garcia@udea.edu.co)

Octubre 31, 2023

Fecha de entrega:	Del 14 al 17 de Noviembre de 2023
Medio de entrega:	https://virtualingenieriaudea.co/
Sustentación:	Horario de Laboratorio
Valor Práctica:	12% del curso

1 Introducción

En esta práctica de laboratorio, el grupo de trabajo (máximo dos integrantes) llevará a cabo la implementación de una versión simplificada del procesador ARM con ejecución de instrucciones en *pipeline*. El desarrollo será llevado a cabo en dos partes o etapas. La primera parte se realizará durante la primera semana y consistirá en implementar el procesador ARM *pipelined* sin unidad de *Hazards*. Para la prueba, el grupo de estudiantes escribirá un corto programa en ensamblador, lo montará en las memorias de instrucciones y datos, y lo probará mediante la herramienta de simulación ModelSIM. La segunda parte se realizará durante la segunda semana y consistirá en agregar al procesador ARM *pipelined* implementado previamente, la unidad de *Hazards*. El grupo

de trabajo empleará el programa de secuencias realizado en la práctica no. 6 para probar la versión modificada del procesador, empleando tanto la herramienta de simulación ModelSIM, como el despliegue en el sistema DE10-Lite. La descripción del hardware será realizada mediante el uso del lenguaje de descripción de hardware SystemVerilog, mientras que los programas de prueba serán escritos en ensamblador, con la ayuda de CPUlator.

2 Objetivo de la Práctica

Implementar una versión simplificada del procesador ARM con ejecución de instrucciones en *pipeline* en el sistema de desarrollo DE10-Lite, mediante el uso del lenguaje de descripción de hardware SystemVerilog, con el propósito de ejecutar un conjunto de programas escritos en lenguaje ensamblador que haga uso de los periféricos del sistema.

3 Consideraciones

Para el diseño del procesador ARM *pipelined*, el grupo de estudiantes primero describirá las unidades de Control y *Datapath* del procesador de acuerdo con la información disponible en las secciones 7.5.1 - *Datapath* y 7.5.2 - *Control* del texto guía del curso, *Digital Design And Computer Architecture - ARMEdition*. Como paso siguiente, escribirá un corto programa en ensamblador y lo probará, mediante simulación, en el procesador ARM *pipelined*. A continuación, agregará al procesador la unidad de *Hazards* con el propósito de resolver los problemas de dependencias de datos y de control, tal y como se indica en la sección 7.5.3 - *Hazards* del texto guía. Finalmente, con el procesador ARM *Pipelined* completo, el grupo de trabajo probará el funcionamiento del mismo empleando el programa de secuencias generado en la práctica previa, mediante simulación y despliegue en el sistema DE10-Lite. Para la implementación, tenga en cuenta las siguientes consideraciones:

a. El diseño y construcción del procesador ARM *pipelined* está basado en la sección 7.5 (7.5.1 - *Datapath*, 7.5.2 - *Control* y 7.5.3 - *Hazards*) del texto guía del curso, *Digital Design And Computer Architecture - ARMEdition*. El entendimiento de dicha sección es imprescindible para poder continuar con el diseño y construcción del procesador. Recuerde que el procesador ARM *pipelined* también fue explicado en clase.

b. El esquema completo del procesador a diseñar se muestra en la Figura 7.58 del texto guía del curso.

c. El código en *SystemVerilog* para las unidades *Control*, *Datapath* y *Hazards* del procesador ARM *pipelined* NO está disponible en el texto guía. Su trabajo será describirlo a partir del código que usted implementó en la práctica de laboratorio no. 6 y probarlo mediante simulación e implementación en la FPGA.

d. Así como con el procesador de un solo ciclo, existirá un módulo **top** que instancia el procesador **arm** y las memorias de instrucciones **imem** y datos **dmem**, respectivamente. La memoria de datos deberá tener el mapeo de los periféricos descritos en la práctica previa.

e. Es mandatorio realizar un diseño modular.

3.1 Parte 1: implementación del procesador ARM simplificado

Antes de comenzar, cree una copia al código del procesador ARM con ejecución de instrucciones de un solo ciclo, que desarrolló en la segunda parte de la práctica no. 6 y llámela 'ARM-Pipelined-Simple'. A continuación, realice el desarrollo de esta primera parte sobre él.

Para describir la unidad de Control (controller) para el procesador ARM *Pipelined*, tenga en cuenta que ésta es bastante similar a la del **procesador de un solo ciclo**, pero con algunas modificaciones en la forma como se conecta y en la localización de la unidad de condición. Recuerde que la unidad de Control incluye los módulos decodificador (decode) y lógica condicional (condlogic). Observe cuidadosamente la Figura 7.47 del texto guía para desarrollar la mencionada unidad. Por otro lado, el *Datapath* del procesador ARM *pipelined* comparte muchas de las características de la misma unidad del procesador ARM de un solo ciclo. Observe cuidadosamente la Figura 7.47 del texto guía para desarrollar el *Datapath*. Tenga en cuenta que se podrá reutilizar la gran mayoría de módulos que se implementaron en la práctica de laboratorio no. 6. Recuerde que todos los registros deben disponer de una entrada de Reset para llevarlos a un estado conocido (0).

Una vez tenga listo tanto la unidad de Control como el *Datapath*, pruebe el funcionamiento del procesador ARM con el código en ensamblador dado en la Fig. 1, al cual se le deberán agregar instrucciones NOP debido a que el procesador ARM *Pipelined* todavía no tiene la capacidad de resolver problemas de dependencias. El grupo de trabajo será responsable de comprender qué hace el código, agregar las instrucciones NOP requeridas (sin agregar más de las necesarias), codificar las instrucciones en código de máquina e incorporarlas en las memorias de instrucciones y datos, preparar un testbench en SystemVerilog que realice la simulación hasta que se escriba en memoria de datos y realizar la simulación para probar el funcionamiento del procesador. Una vez realizada la simulación, responda las siguientes preguntas: ¿Qué valor se escribió en los registros R0 a R5 y LR durante la ejecución del programa? ¿Qué valor escribió la instrucción STR y en qué dirección? La simulación deberá incluir por lo menos las siguientes señales: clk, reset, PC, Instr, RF[0-14] y dmem[0-8].

Las instrucciones que deberá soportar el procesador ARM *Pipelined* en esta primera parte de la implementación son las mismas que soporta el procesador implementado en la práctica de laboratorio no. 6, además de la instrucción NOP:

a. Aritméticas, lógicas y datos: ADD, SUB, AND, ORR, MOV, LSL, LSR, ASR y ROR, empleando registros y valor inmediato sin desplazamientos.

b. Carga y almacenamiento: LDR y STR, con desplazamiento inmediato positivo.

c. Salto B y BL.

```
.global _start
_start:
    MOV     R0, #0
    LDR     R1, [R0, #0]    // Load value 0x80000000 from Memory Address 0x00000000
    ASR     R2, R1, #31
    LSL     R3, R2, #31
    LSR     R4, R3, #15
    ROR     R5, R4, #31
    BL      Store
End:
    B       End
Store:
    STR     R5, [R0, #4]    // Store value to Memory Address 0x00000004
    MOV     PC, LR
```

Fig. 1: Plantilla para el código de prueba

d. Otras: NOP. Tenga en cuenta que CPUlator codifica esta instrucción con el valor 0x0320F000. Esta instrucción no deberá modificar el estado de la arquitectura del procesador. Solamente se debe actualizar el registro R15 para poder ejecutar la instrucción siguiente.

3.2 Parte 2: Implementación de la Unidad de Hazards en el Procesador ARM *Pipelined*

Antes de comenzar, cree una copia al código del procesador ARM que desarrolló en la primera parte de esta práctica y llámela 'ARM-Pipelined-Hazards'. A continuación, realice el desarrollo de esta segunda parte sobre él. No vaya a borrar el código 'ARM-Pipelined-Simple', lo necesitará para la sustentación.

Para resolver los problemas de dependencia de datos (*data hazards*) y control (*control hazards*) explicados en clase, es necesario implementar la unidad de *Hazards* en el procesador ARM *Pipelined* junto con la lógica requerida en las unidades de Control y *Datapath*. Primero, añada la funcionalidad *Data Forwarding* para resolver los problemas de dependencia de datos con la mayoría de instrucciones. Tenga en cuenta la Figura 7.51 del texto guía para agregar esta funcionalidad. A continuación, agregue la capacidad al procesador ARM *Pipelined* de realizar un *Stall* de un ciclo cuando se presente la instrucción LDR. Tenga en cuenta la Figura 7.54 del texto guía para agregar esta funcionalidad. Finalmente, añada la funcionalidad para manejar los problemas de dependencias de control (*control hazards*) que involucran el uso de las instrucciones de salto Bxx y BLxx o escrituras al registro R15 por parte de cualquiera de las instrucciones disponibles. Tenga en cuenta la Figura 7.57 del texto guía para agregar esta funcionalidad.

Una vez tenga listo la unidad de *Hazards*, pruebe el funcionamiento del procesador ARM con el programa dado en la Fig. 1 (recuerde que los NOPs ya no serán necesarios), tal y como lo hizo en la primera parte de esta práctica. El resultado deberá ser similar al que obtuvo con el procesador ARM *Pipelined* sin unidad de *Hazards*, pero con un número menor de instrucciones ejecutadas

¿Por qué? La simulación deberá incluir por lo menos las siguientes señales: clk, reset, PC, Instr, RF[0-14] y dmem[0-8].

Finalmente, incorpore a las memorias de instrucciones y datos, el programa de secuencias que realizó para la práctica no. 6, compile el diseño y realice el despliegue en el sistema DE10-Lite. ¿Cuál es el resultado obtenido?

4 Entrega

El grupo de trabajo deberá escribir un breve reporte en formato IEEE que contenga los siguientes elementos (ver guía para reportes en la página del curso):

a. **Abstract**: resumen del diseño e implementación de la práctica.

b. **Respuestas a preguntas**: respuestas a las preguntas de la primera y segunda parte del desarrollo de la práctica.

c. **Esquemas de HW**: esquemáticos del hardware del procesador ARM *pipelined* con unidad de *Hazards* para soportar las instrucciones de desplazamiento, movimiento de datos y saltos a funciones (solo la parte que involucran las instrucciones LSL, LSR, ASR, ROR, MOV y BL).

d. **Simulación**: simulaciones con el programa dado en la Fig. 1 para ambas partes del desarrollo de la práctica.

e. **Conclusiones**: dos o tres conclusiones sobre el trabajo realizado por el grupo de trabajo. Indicar el tiempo que les tomó realizar la práctica en las conclusiones.

Crear un archivo comprimido que incluya el reporte y los archivos importantes de su código en ensamblador y SystemVerilog como se describe a continuación:

a. **Reporte**: archivo con extensión .pdf

b. **Archivos esenciales**: proyecto en QuartusPrime y códigos en ensamblador.

El nombre del archivo comprimido deberá tener el siguiente formato:
p7_primerapellidointegrante1_primerapellidointegrante2_horariolaboratorio.zip.
Ejemplo: si el primer apellido de ambos integrantes es **Castano** y **Garcia**, respectivamente, y el laboratorio es el Martes 9-12, entonces el archivo debe ser nombrado: *p7_castano_garcia_m9-12.zip*.

5 Evaluación

La evaluación de la práctica se divide en tres partes: funcionamiento (40%), sustentación (50%) y reporte (10%). Las notas de funcionamiento y reporte se asignan por igual a todos los integrantes del grupo de trabajo (máximo dos personas por equipo), mientras que la nota de sustentación es individual. La sustentación podrá realizarse de cuatro maneras posibles:

a. Un examen corto al finalizar la sesión de laboratorio.

b. Solicitud de cambios al código por parte del profesor de laboratorio.

c. Un par de preguntas orales que pongan a prueba los conocimientos del estudiante sobre el desarrollo de la práctica.

d. Una presentación corta a todo el grupo de laboratorio.

En caso un estudiante obtenga una nota inferior a 3.0 en la sustentación, la nota final de la práctica para el estudiante en mención será la que obtuvo en la sustentación, es decir, no se tendrá en cuenta el funcionamiento en el cálculo.

Cada grupo de trabajo deberá sustentar la práctica en un tiempo de 15 minutos, 8 minutos para revisar la funcionalidad y 7 minutos para la sustentación en cualquiera de las cuatro modalidades presentadas. Es importante tener abierto el CPUlator y el Quartus Prime con el programa listo para cuando el profesor llegue a su puesto de trabajo. No habrá tiempo para hacer correcciones de último momento.

6 Referencias

a. Harris, Sarah and Harris, David. Digital Design and Computer Architecture: ARM Edition. Morgan Kaufmann Publishers Inc. 2015. ISBN: 0128000562

b. CPUlator Computer System Simulator
`https://cpulator.01xz.net/?sys=arm`

c. Quartus Prime Lite Edition
`https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime.html`

d. Sistema de desarrollo DE10-Lite
`http://de10-lite.terasic.com/`

e. Tutorial de SystemVerilog
`https://verilogguide.readthedocs.io/en/latest/`