Proyecto 2 - Arquitectura de Computadores

Sebastián Quesada Rojas, *estudiante,* Ingeniería en Computadores, *Instituto Tecnológico de Costa Rica*
Mario Carranza Castillo, *estudiante,* Ingeniería en Computadores, *Instituto Tecnológico de Costa Rica*
Daniel Rayo Diaz, *estudiante,* Ingeniería en Computadores, *Instituto Tecnológico de Costa Rica*
Andrés Martinez Vargas, *estudiante,* Ingeniería en Computadores, *Instituto Tecnológico de Costa Rica*

1 Listado de Requerimientos del Sistema

Como primera parte, el sistema deberá tener un procesador de tipo Pipeline creado utilizando el HDL SystemVerilog. Además, este procesador debe ser implmentado para poder ser utilizado y sintetizado por una FPGA de tipo Terasic DE1-SoC. Es necesario tomar en cuenta los riesgos que tiene poder implementar un procesador de tipo Pipeline los cuales son:

- **Dependencia de datos:** Se sabe que en este tipo de procesadores, las instrucciones se ejecutan en etapas distintas. Cuando una instrucción necesite de un dato que aún no ha terminado de procesarse, se debe tener la consideración de los STALLS.
- **Control:** Se debe ser cuidadoso sobre todo en instrucciones de condiciones, pues cuando se deba implementar un salto, se debe tener certeza que el programa se va a ejecutar donde debería.
- **Recursos:** Como también se va a interactuar con la memoria, se deberá tener en consideración el gasto de la misma, esto para poder garantizar un correcto funcionamiento del programa.

El procesador debe tener la capacidad de poder crear la segmentación de memoria en datos y en instrucciones. Para esto, se probará en simulación cada uno de los elementos del mismo con unit tests.

Como segunda parte (software), se deberá crear un ISA propio, con el tamaño que se considere correcto para las instrucciones. Luego del proceso de diseño del ISA, se procederá con la creación de un compilador, esto para poder leer cualquier código que el usuario desee ingresar para ser ejecutado por el procesador. Por lo tanto, se tendrá un archivo compilado en lenguaje ensamblador.

2 Elaboración de opciones de solución al problema

En el caso del procesador con Pipeline, se sugirió que se debía implementar el procesador basándose en el procesador uniciclo dado por el libro [1]. De esta manera, se procede a añadir los registros necesarios luego de cada etapa y se dividen lo módulos por etapa para tener un mayor orden.

En el caso del ISA, se planteó que la idea principal sería basar este mismo en el ISA de ARM. Al tratarse de un conjunto de pocas instrucciones, se plantea utilizar 16 bits, donde para los 4 bits menos significativos se tiene que:

- **1er bit menos significativo:** Este bit será utilizado como el de overflow donde un 1 significa que la operación supera los 16 bits.
- **2do bit menos significativo:** Este bit será utilizado como el de carry, donde un 1 significa que es necesario hacer un acarreo.
- **3er bit menos significativo:** Este bit será utilizado como el de zero, donde un 1 significa que el resultado de una operación es 0.
- **4to bit menos significativo:** Este bit será utilizado como el de negativo, donde un 1 significa que el resultado de una operación es negativo.

Además, el Op.Code será de 4 bits. Este se encontrará en los 4 bits mas significativos de la cadena de 16 bits. Para este se tiene la siguiente tabla de igualdades:

Op.Code	Significado
0000	ADD
0001	SUB
0010	AND
0011	ORR
0100	LSL
0101	CMP
0110	SET
0111	LDR
1000	STR
1001	BAL
1010	BEQ
1011	BGE
1100	NOT
1101	END

TABLE 1
Igualdades para el OpCode

La segunda opción planteada fue utilizar un ISA basado en RISC V. La variación con respecto a la primera opción basada en ARM, es que el op.code se encuentra en los 4 bits menos significativos, manteniendo el estándar de la tabla [1]. Igualmente se mantienen las instrucciones de un tamaño de 16 bits.

3 Comparación de opciones de solución

En este caso, ambas alternativas presentadas son muy similares. A nivel económico, al tratarse de una solución basada en 16 bits, se espera que no sea de alto consumo de potencia, por lo que es una excelente alternativa para un bajo presupuesto. A nivel de seguridad, el manejo de información dentro del procesador es cerrado, por lo que costaría una filtración de la misma. Viendo esta información, la escogencia de una de las opciones dependería de la preferencia del estándar a utilizar; es decir, a la posición dentro de la cadena de 16 bits donde se encuentre la información necesaria, como lo es el valor de los registros y el opcode.

4 Selección de la propuesta final:

Para lo planteado a nivel de software, se decidió a utilizar la opción del ISA basado en ARM. Tomando en consideración que ambas opciones eran bastante similares (basarse en ARM o en RISC V), se planteó que tener el Op.Code en las primeras posiciones de la cadena de 16 bits era la mejor opción, esto pues es el estándar al que se está mas acostumbrado por todos los miembros del grupo. Esto también siguiente nuestra principal fuente bibliográfica [1] y en cual se basó el procesador Pipeline, ya que este está diseñado para utilizar con ARM.

Para el caso del procesador, se siguió los lineamientos del Uniciclo presentado en [1], al cual se le crearon los registros y las etapas necesarias. De esta manera, se obtuvo la solución presentada en los anexos [1][2][3] y [4].

5 Anexos

A continuación se presenta el diagrama de bloquess divido por etapas:

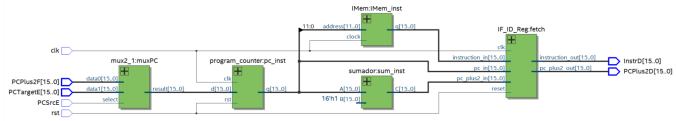

Fig. 1. Etapa de **Fetch** del procesador

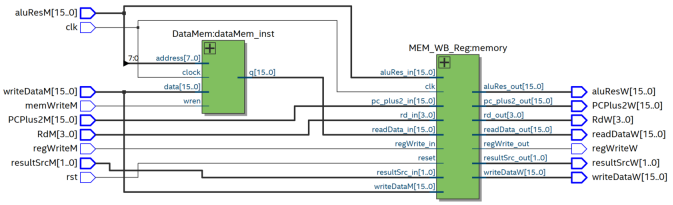

Fig. 2. Etapa de **Memoria** del procesador

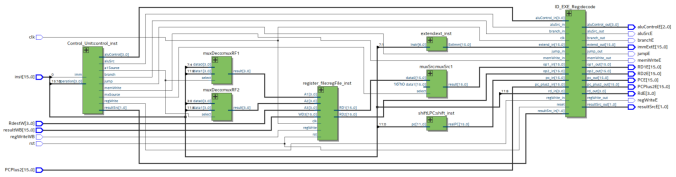

Fig. 3. Etapa de **Decode** del procesador

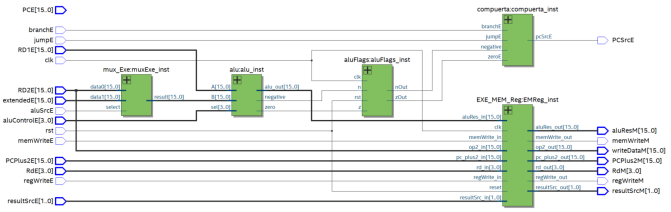

Fig. 4. Etapa de **Execute** del procesador

References

[1] Sarah Harris y David Harris. Digital design and computer architecture: arm edition. Morgan Kaufmann, 2015.