Procesador ARM monociclo Simplificado de 32 bits.
Informe

Alexander Passo Polanco (1038139046)
Víctor Manuel Jiménez García (1036688794)
Universidad de Antioquia

Abstract:

Para la implementación del procesador ARM monociclo, se partió de la explicación en cada una de las unidades del libro propuesto por la guía correspondiente a la práctica, en donde se logró entender el funcionamiento de cada uno de los bloques para así agregar las instrucciones MOV y CMP en la alu. Con las nuevas implementaciones, se debió modificar el controlador y el decodificador, de tal manera que las nuevas instrucciones fueran entendidas por el programa. El funcionamiento de la alu, del procesador original y modificado, se mostrará en el desarrollo de este documento.

Tabla y simulación ALU:

Para el desarrollo de la ALU, fue clave el esquema de la ALU del libro (Pág 250), ya que permitía entender su funcionamiento de forma combinacional, además de obtener los flags que activan las banderas según la instrucción dada. Dicho esto, se procedió a simular la alu, para los diferentes AluControl.

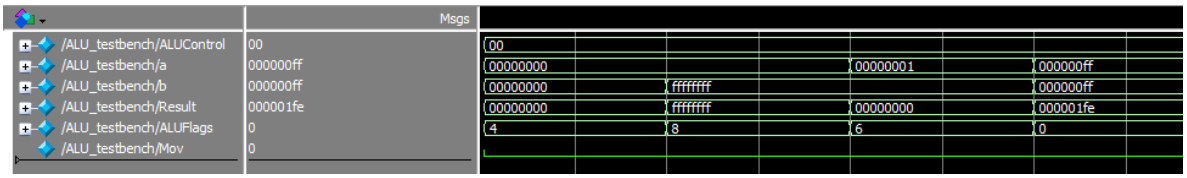

Figura 1 Simulación ALU con AluControl=0

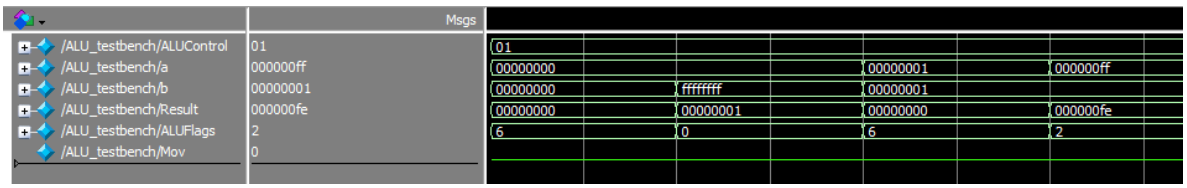

Figura 2 Simulación ALU con AluControl=1

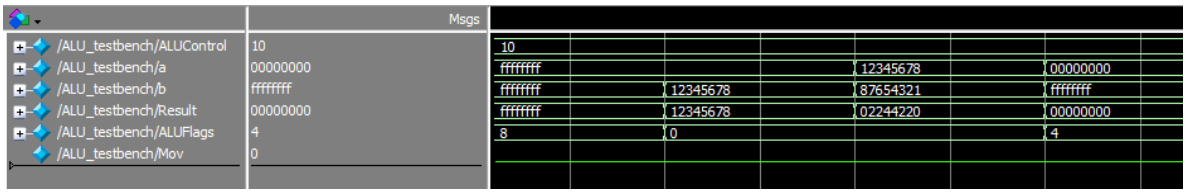

Figura 3 Simulación ALU con AluControl=2

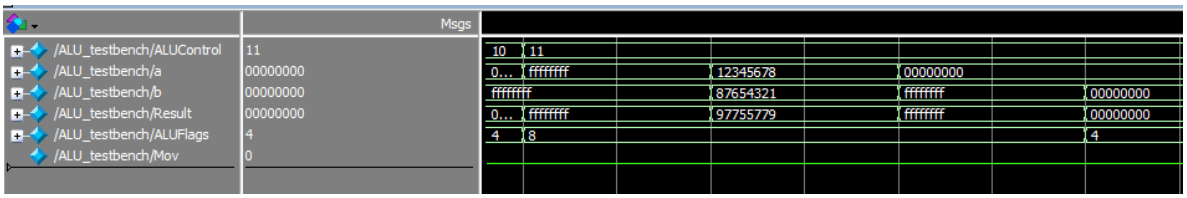

Figura 4 Simulación ALU con AluControl=3

A partir de la simulación obtenida, se completó la tabla dada en la guía la cual se muestra a continuación.

Prueba	ALUControl	A	B	Result	ALUFlags
ADD 0+0	**0**	**0**	**0**	**00000000**	**4**
ADD 0+(-1)		0	FFFFFFFF	FFFFFFFF	8
ADD 1+(-1)		1	FFFFFFFF	00000000	6
ADD FF+FF		000000FF	000000FF	000001FE	0
SUB 0-0	**1**	**00000000**	**00000000**	**00000000**	**6**
SUB 0-(-1)		00000000	FFFFFFFF	00000001	0
SUB 1-1		00000001	00000001	00000000	6
SUB FF-1		000000FF	00000001	000000FE	2
AND FFFFFFFF, FFFFFFFF	**2**	**FFFFFFFF**	**FFFFFFFF**	**FFFFFFFF**	**8**
AND FFFFFFFF, 12345678		FFFFFFFF	12345678	12345678	0
AND 12345678, 87654321		12345678	87654321	02244220	0
AND 00000000, FFFFFFFF		00000000	FFFFFFFF	00000000	4
OR FFFFFFFF, FFFFFFFF	**3**	**FFFFFFFF**	**FFFFFFFF**	**FFFFFFFF**	**8**
OR 12345678, 87654321		12345678	87654321	97755779	8
OR 00000000, FFFFFFFF		00000000	FFFFFFFF	FFFFFFFF	8
OR 00000000, 00000000		00000000	00000000	00000000	4

Figura 5 Tabla de entrada y salida ALU.

Simulación del procesador ARM:

Para este caso, se tomo el procesador diseñado en el libro guía, y se evidencio su funcionamiento a partir del testbench propuesto por el libro, en el que se busca asignar el numero 7, en la dirección de memoria 24. Está instrucción esta dada en la tabla que se muestra en la guía respectiva de la práctica. Por tanto, se puede observar en la siguiente imagen que el procesador esta funcionando correctamente.

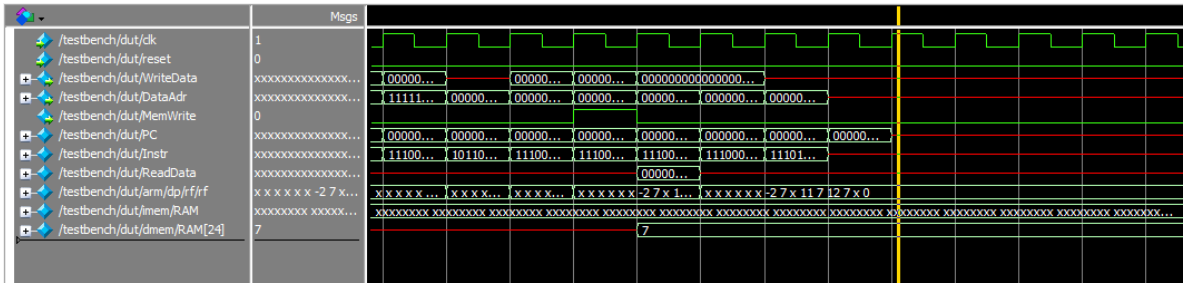

Figura 6 Simulación del procesador ARM.

Simulación del procesador ARM (Modificado):

A continuación, se procede a mostrar en ModelSIM las señales: clk, reset, WriteData, MemWrite, PC, Instr, AluResult y ReadData.

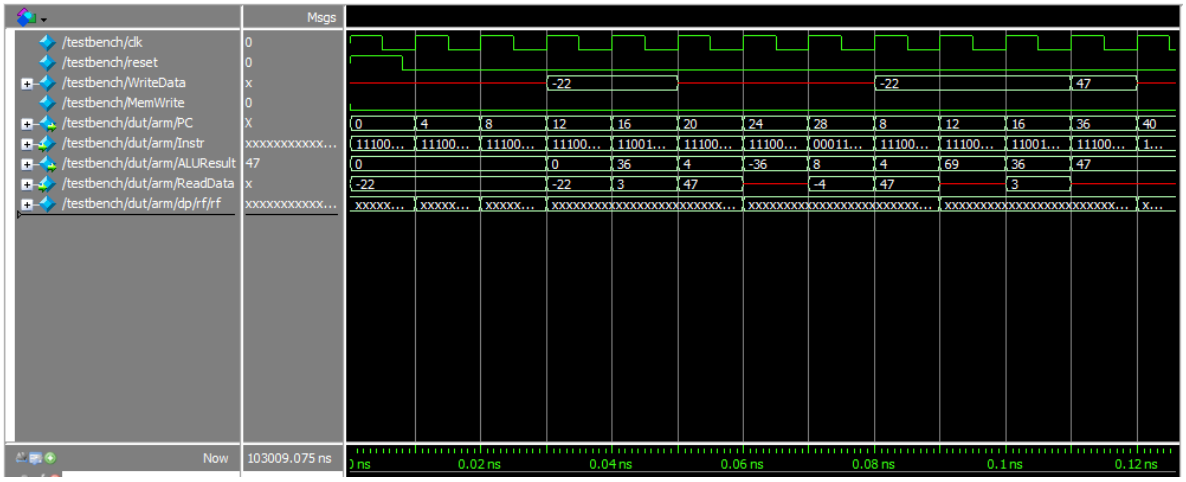

Figura 6. Simulación de las señales del procesador ARM.

Luego, para verificar el funcionamiento se opta por hacer una validación en los registros, así:

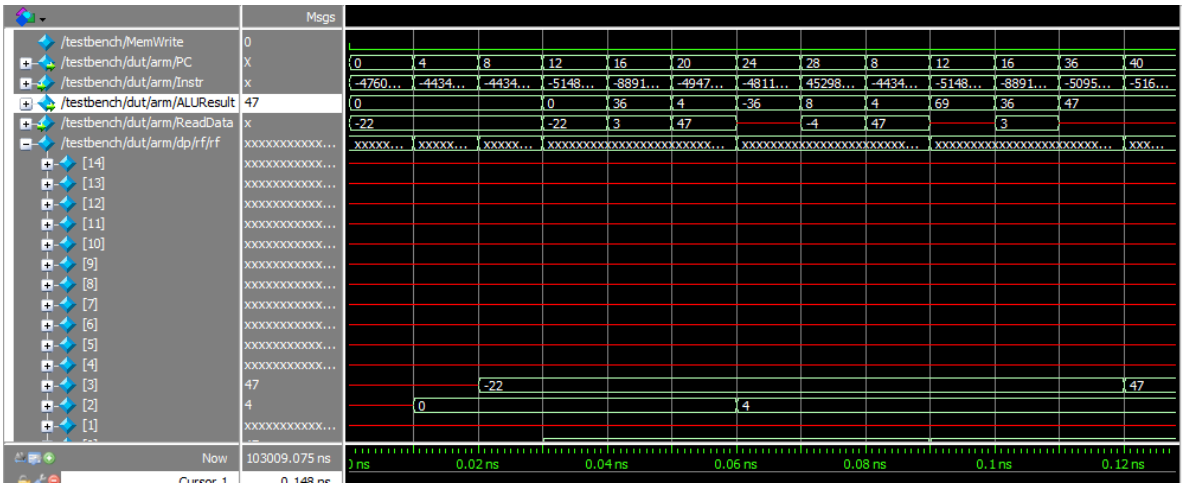

Figura 7. Verificación por medio de registros.

Para la implementación de la instrucción MOV, se hizo a partir de la suma, pero para no tener inconvenientes con la instrucción ADD, se agrego un condicional para que activara esta instrucción.

Con respecto a la instrucción CMP, se implemento a partir de la resta, con una codificación diferente a las demás, que depende de los valores que tenga la señal Funct, y de la nueva señal que se agregó, NoWrite, para que el programa sepa cuando se hace la operación resta o cuando se hace la operación CMP.

Conclusiones

- Para la implementación de la ALU, se debió diseñar a partir de las compuertas AND, OR, XOR, y XNOR para realizar las banderas del aluflags.

- Se logró implementar la instrucción MOV a partir de la suma, teniendo en cuenta una nueva señal (mov) que permitía activar a partir de una condición activar esta instrucción

- Para hacer que los diferentes módulos quedaran unidos en uno solo, fue necesario establecer las entradas y salidas de una forma específica y obedecer a dicha colocación porque de lo contrario, se generarían errores en el sistema.

- Hay un atraso generado por las componentes secuenciales y para llevarlo a lo más mínimo posible, es bueno contar con una microarquitectura que permita dicha reducción.

- El uso del libro guía fue fundamental, no solo por los códigos que proveyó sino que también ayudó a entender internamente el funcionamiento de todo el montaje.