UTEC

Procesador MIPS 32

Arquitectura de las Computadoras

Alejandro Goicochea
alejandro.goicochea@utec.edu.pe

Profesor:
Jorge Gonzales

July 3, 2019

1 Introducción

En este reporte se va a documentar la implementación de un procesador MIPS32 en verilog. Para las instrucciones se implementaran algunas instrucciones del instruction set detallado en el MIPS reference data card del siguiente enlace: MIPS Resource Data Sheet
Las instrucciones implementadas son las siguientes:

add	addi	lb	beq
sub	andi	lh	bneq
and	ori	lw	bgez
nor	slti	sb	j
or		sh	jr
slt		sw	jal
		lui	

Para el caso de bgez se usara el opcode 000001.

2 Implementación

Para la implementación del procesador se usaron los siguientes módulos:

1. DataPath
2. InstructionMemory
3. InstructionProcess
4. Registers
5. DataMemory
6. ALU
7. ControlUnit
8. MemToReg

2.1 Datapath

Este módulo es la base del procesador, se encarga de enviar la información a todos los otros módulos mediante wires y de la declaración de todos los registros que se van a usar como las señales y registros. Además, se encarga de los incrementos del PC a través de los jumps y branches o un incremento de 4.

2.2 InstructionMemory

En este módulo se almacenan las instrucciones en un registro desde un archivo de texto y se devuelve una instrucción cada vez que llega un PC. Las instrucciones se guardan en bytes por lo que el módulo concatena 4 direcciones para devolver la instrucción completa.

2.3 InstructionProcess

En este módulo se divide la instrucción en sus diferentes partes dependiendo del tipo de instrucción. Toma siempre el opcode ya que los primeros 6 bits siempre le van a pertenecer y con eso sabe cómo partir el resto de los bits.

2.4 Registers

Este módulo se encarga de los 32 registros con los que cuenta el procesador. Está separado en dos bloques *always* uno para escritura y otro para lectura por lo que tiene dos señales que habilitan estos bloques. Para el bloque de escritura, también se cuenta con una señal para ver si se escribe a la dirección de rd o rt dependiendo si es una instrucción de tipo R o I. En caso sea una instrucción de tipo I se revisa el opcode en caso de que sea una instrucción load que tiene diferente comportamiento a las demás. Los registros están cargados inicialmente con puros 0 de un archivo de texto.

2.5 DataMemory

En el DataMemory se maneja el caché del procesador. Esta se carga de un archivo de texto lleno de 0. Cuenta con dos funciones, escritura y lectura. El módulo solo se llama para las instrucciones de load y store, mandando datos al registro en caso de loads y almacenando en la memoria en caso de store. Cuenta también con dos señales: $memWrite$ y $memRead$ para determinar la operación que se hace.

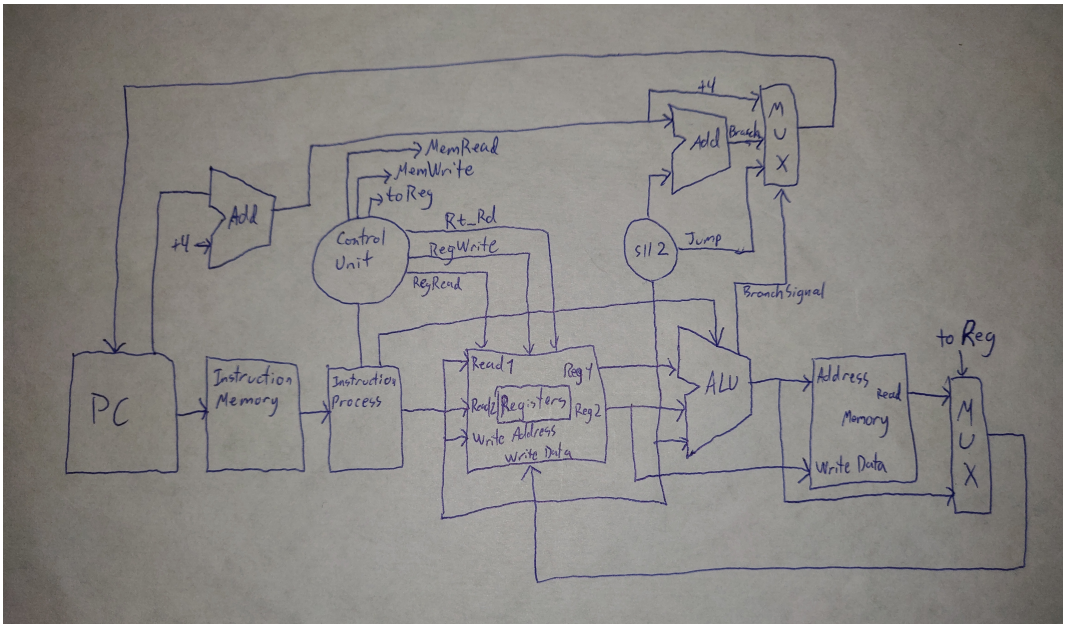
2.6 ALU

La ALU se encarga de todas las operaciones aritméticas de nuestras instrucciones. También se encarga de determinar si se toma un branch o no. Siempre se revisa el opcode primero para ver si la instrucción es de tipo R por lo que se pasa a revisar el valor de $funct$. La ALU también crea registros de tipo signed para las operaciones de suma o resta, calcula la dirección del branch, y la dirección de los stores y loads.

2.7 MemToReg

El módulo MemToReg es un multiplexer de 2 a 1 que se encarga de seleccionar que se va a mandar al módulo de registros para escritura. El selector de este multiplexer depende del tipo de instrucción que se está ejecutando.

3 Diagrama



4 Conclusiones

El procesador puede ejecutar todas las instrucciones propuestas exitosamente. Se crearon 3 programas para ejecutarlas. El primero ejecuta las dos primeras columnas de las instrucciones detalladas en la sección 1, el segundo la tercera columna y el tercer programa la cuarta columna. Para el debugging del procesador se usó el comando *display* en varios lugares los cuales terminaron siendo usados para comprobar que el datapath funcione correctamente. Cada módulo ejecuta un *display* dependiendo de las operaciones que ejecute las cuales detallan la ejecución de cada programa. El procesador implementado, aunque funcione correctamente, utiliza más hardware que un procesador común y gracias a las facilidades que ofrece el lenguaje usado se implementan algunos MUX dentro de los módulos para reducir la cantidad de wires y módulos usados simplificando su lectura y comprensión.