MIPS Datapath

Stephano Wurttele
Cesar Madera
Profesor: Jorge Luis Gonzales Reaño

Julio 2019

1 Abstract

El presente informe sirve como reporte del trabajo final del ciclo 2019-1 delcurso de Arquitectura de Computadoras, curso dictado por los profesores Renzo Bustamante y Jorge Luis Gonzales Reano. En este, se tratar a de explicar lo sprocesos por los que pasar ıa normalmente un Datapath que siga la arquitectura MIPS, y a su vez explicar como nosotros mismos lo hemos implementado digitalmente usando el simulador Verilog. A continuación haremos una breve explicación de que es MIPS y como funciona, mostraremos cuales fueron las diferencias entre los módulos que hemos implementado y los de la arquitectura MIPS, y finalmente concluiremos con algunas diferencias destacables que considerar y lo aprendido del proyecto.

2 Introduccion

Como sabemos, los computadores a lo largo de los años han ido evolucionando a velocidades sorprendentes. En realidad, hemos pasado de computadores del tamaño de cuartos enteros a computadores del tamaño de nuestras uñas. Sin embargo, nadad de esto es magia. Todo esto tuvo que ser implementado por personas muy dedicadas e inteligentes que, con tanto conocimientos fisicos como percepcion abstracta, han logrado hacer mejores controladores y microchips con los cuales se puedan hacer operaciones.

De todos modos, parte de este proceso de creacion es experimentar, y utilizar piezas una y otra vez para probar si algo funciona o falla, llegado cierto nivel de abstraccion, es no solo absurdo sino también caro. Para ello, se han creado programas para simular sistemas de hardware como es Verilog, programa que usaremos para el desarrollo de este informe. En brevedad explicaremos como es que hicimos la simulación y utilizando que componentes, al igual que su funcionamiento.

3 Metodologia: La arquitectura MIPS

La arquitectura MIPS resalta por su método de agrupación de instrucciones y registros. Las instrucciones se carácterizan por ser byte adressable (cada espacio de su respectivo Instruction Memory se dividen por bytes) mientras que los registros y data Memory son word-addressable (se dividen por words o 4 bytes). Asimismo, el motor del datapath son los clocks. Dada la división de las instrucciones, cada clock hace que la direccion de la instruccion a revisar avance de 4 en 4, para poder recoger una instruccion completa y poder descomponerla para ser evaluada a detalle por cada parte del procesador.

Utilizamos esta arquitectura como guia, ya que es la que mejor entendemos y estudiamos en clases. Primero debiamos entender en su totalidad como funcionaba el datapath de MIPS para poder emular uno similar hecho por nosotors mismos con los componentes que vieramos necesarios para que este funcionara. Aqui fue cuando identificamos que, entre todas las instrucciones, hay muchos mas detalles implicitos de los que se ve en el diagrama que mostraremos a continuacionFinalmente, hicimos pruebas con respecto a todas las instrucciones (que son solo algunas de las instrucciones implementadas en MIPS) para evaluar nuestro procesador y ver que este, con todos sus componentes y en todo momento, funcionara correctamente.

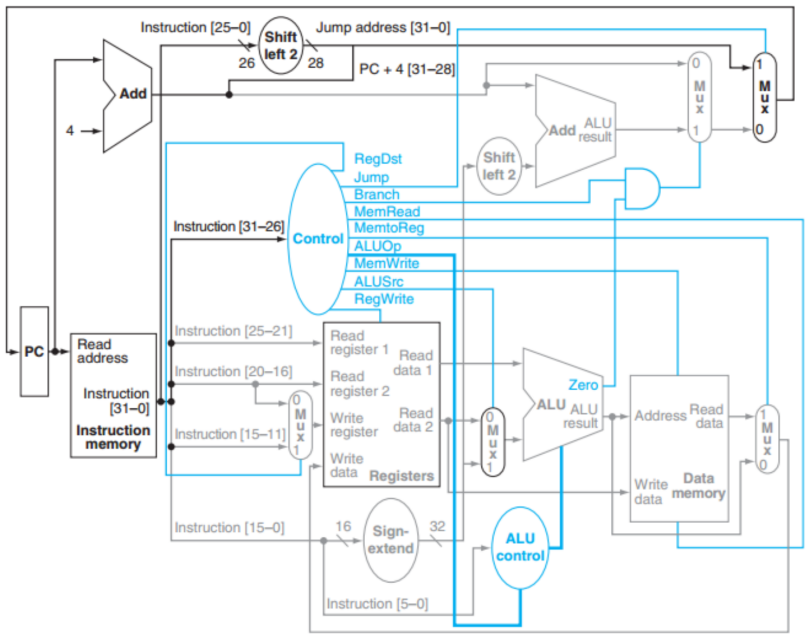
Lo especial de esta arquitectura no es unicamente la manera por la cual se obtienen las instrucciones, sino tambien los campos en los cuales se subdividen estos 4 bytes:

Type	-31-	format (bits)				-0-
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	immediate (16)		
J	opcode (6)	address (26)				

Figure 1: Imagen obtenida de Wikipedia

Como se puede ver, existen 3 tipos de instrucciones: las de tipo R, las cuales normalmente utilizan 3 registros (a excepcion del jump register), las del tipo I, las cuales usan dos registros y un inmediato y los de tipo J, los cuales se caracterizan saltar de una instruccion a otra a través del manejo del Program counter.

Ahora presentamos la imagen referencial del datapath del MIPS mencionado previamente.



6.png

Figure 2: Imagen obtenida de Wikipedia

4 Modulos

Como podemos ver en la imagen, hay un larga cantidad de modulos que queremos explicar en este inciso. Cada uno tiene una funcionalidad diferente, sin embargo funcionan en conjunto, y una larga cantidad de instrucciones podrian dejar de funcionar si dejamos de tener alguno de estos componentes. Empezaremos desde el PC, dado que es el que recibe los clocks y el motor del procesador.

4.1 PC

Este modulo se encargará de recibir el program counter y volver a lanzarlo, el cual en un inicio comenzará en 0 y luego podría variar según el tipo de instrucción que se esté usando. En caso que una instrucción no involucre al PC directamente, entonces el siguiente será PC+4. Asimismo, esta decision será tomada unicamente cuando el clock sea positivo.

4.2 Instruction Memory

Este módulo se encargará de recibir el PC, el cual señala a la ultima direccion de la instrucción contenida en el Instruction Memory, la cual permitirá obtener los 4 bytes que siguen con la estructura MIPS explicada anteriormente

4.3 Control Unit

Este módulo se encargará de recibir unicamente los 6 bits mas importantes de la instrucción generada por el Instruction Memory, los cuales se van a procesar y dará como output a: RegDst,Branch,MemtoReg,ALUSrc,RegWrite, ALUOP, Memwrite,MemRead y Jump; los cuales nos ayudarán a tomar decisiones en los módulos por venir.

4.4 Register Memory

Este módulo contiene los 32 registros necesarios para la implementacion de la arquitectura MIPS. De este módulo se pueden hacer 2 operaciones: Obtener información de 2 registros a la vez, o escribir dentro de un registro. El primer caso expuesto requiere la información de la posición de los registros los cuales se quieren leer ,mientras que en el segundo caso se necesita tanto el RegWrite, la posición de registro en donde se quiere escribir, la información que se quiere escribir (la cual vendrá en un futuro por un mux que se explicará a continuación) y los ultimos 6 bits de la instrucción (para poder discriminar a la instrucción jump register).

4.5 Mux

Este módulo se encargará de hacer una elección entre 2 parámetros teniendo en cuenta un bit, el cual será el control de este mux. Para nuestra implementación, se necesito hacer un mux que reciba parámetros de 5 bits y otro para los parámetros de 32 bits. Mayormente los control de estos mux serán las salidas del Control Unit.

4.6 PC Adder

Este módulo obtiene el PC y le agrega 4 y lo lanzará denuevo.

4.7 Sign-extension

Este moduló se encargará de alargar una entrada de 16 bits a una salida de 32 bits. Para esto, se hizo una comparación entre el bit mas significativo de la entrada, y si este era igual a 1, pues todos los bits de alargamiento serán 1's mientras que serán 0's si es que no.

4.8 Shift Left Logical 2

Este módulo se encargará de obtener una entradad y multiplicarla por 4 (lo cual en sistema binario equivaldría a desplazar 2 veces hacia la izquierda a todos los bits).

4.9 Alu

Este módulo es probablemente el más importante de todos ya que realiza todas las operaciones matemáticas y lógicas, tal como lo dice su nombre (Arithmetic-Logic Unit). Este módulo recibe la información tanto de un mux, el cual está decidiendo entre el segundo registro o un numero inmediato, el cual fue alargo por el sign-extend y el primer registro, además de un control, el cual es pasado por el Alu Control, el cual va a permitir decidir entre qué operación realizar. Los outputs de este módulo serán un overflow (el cual no va a tener utilidad en este proyecto, pero que de todas maneras importa), un "Zero" el cual unicamente servirá si es que la instrucción a ejecutar es un branch y la operación en sí.

4.10 Alu Control

Este módulo se encarga de recibir tanto los primeros como los últimos seis bits de la instrucción y decidir entre cual de ellos mandar al Alu como su controlador, para que pueda ejecutar sus operaciones. Este módulo tiene varias condicionales para poder mandar el controlador al Alu, debido a que existen varias variantes entre los diferentes tipos de instrucciones.

4.11 Data Memory

Se encarga de almacenar toda la información que no esta siendo inmediatamente utilizada ya que los registros como tal, son muy costosos por su alta velocidad. El Data Memory también permite obtener y sobreescribir data ya existente. Sus entradas vienen directamente del resultado de la ALU(el cual será la dirección de la memoria a la cual se quiere llegar), la data que se quiere escribir, el MemWrite y el Memread, los cuales serán los controladores que permitiran decidir entre las diferentes instrucciones que se pueden realizar, ya que existen diferentes maneras de sobreescribir un espacio de la memoria y a la vez, a veces no se necesita cargar todo el dato de la memoria, por lo que se resulta util que sea filtrado antes de culminar. Como salida tiene la informacion guardada en la direccion de la memoria, la cual luego pasará por un mux, y finalmente se decidirá si se debe o no escribir en el Register Memory.

5 Nuestra implementación

Nos hemos basado en la estructura de la arquitectura MIPS, aunque se han hecho modificaciones para poder solucionar algunos problemas que han surgido en el proceso de crear nuestro propio procesador. Como mencionamos antes,el procesador de MIPS presentado previamente es una version simplificada y grafica de todo lo que se tiene a nivel de hardwaree realmente. Sin embargo, tambien tiene muchas mas instrucciones de las que nosotros estamos implementando como validas en nuestro datapath. Consecuentemente, nuestra implementacion utiliza varios detalles que nosotros creimos necesarios y que, sean o no parte de

la implementacion real de MIPS, no son mostrados en la grafica anterior pero si en la siguiente:

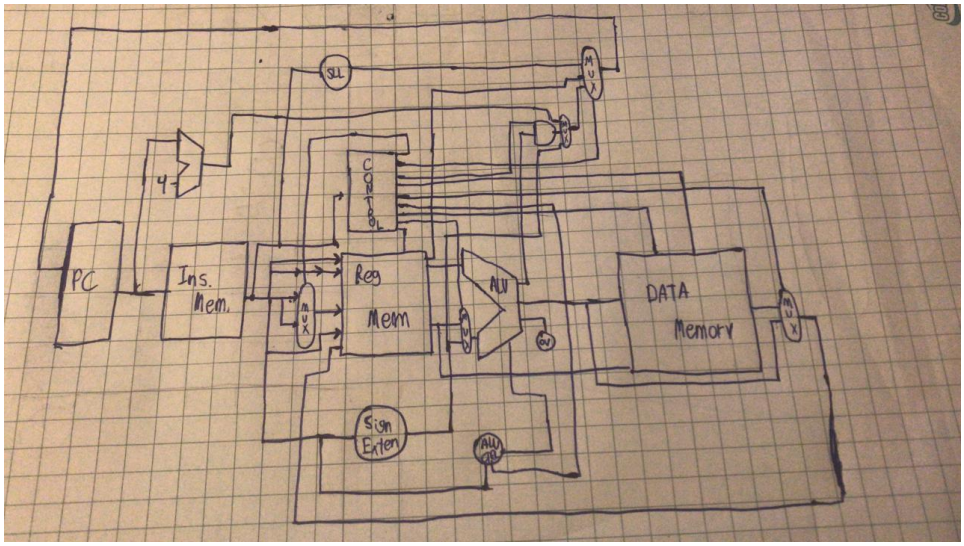


Figure 3: Gráfico dibujado por el grupo

Si bien no tenemos modulos adicionales, utilizamos un par de cables mas para poder trabajar con algunas excepciones entre instrucciones, como el uso del Jump Register que presento ciertas dificultades por su naturaleza de instruccion R pero dirigiendose al modulo de direcciones, o el cable adicional que utilizamos de las instrucciones al Register Memory para poder leer lo que se quiere cargar cuando utilizamos el Load Upper Immediate. Asimismo, del control se utilizan varios buses en vez de solo cables, como o para poder identificar las instrucciones mejores en ciertos modulos, comoen el caso de las instrucciones load y store. Especificamente, los cables que salen del Control: "Jump", "MemRead", y "MemWrite" fueron configurados a buses de dos bits para que se decida que tipo de Jump es, si necesita guardar el registro o de donde sacar el donde saltar; y en el caso de los MemWrite y MemRead se necesita el bit extra para saber que parte se guardara, o el tamaño del guardado de necesitar ser byte, word, o halfword.

6 Evaluacion

Como fue mencionado, en nuestro datapath hay muchas situaciones en las cuales necesitamos utilizar buses adicionales o cables que, si seguimos lo aprendido en clase, resultan en un costo adicional para menos instrucciones. Esto hace que el procesador, si bien funciona y es hasta cierto punto eficiente, es demasiado costoso para lo que hace. Se podrian hacer mas instrucciones de las implemen-

tadas con los cables adicionales que hemos incluido, lo cual le da un potencial extra al datapath, diferenciado de solo lo que el profesor solicito como instrucciones. Aun asi, tendrían que ser instrucciones pensadas cuidadosamente, pues lo que se busca es generalizar lo más posible las instrucciones. Una curiosidad relacionada a esto será mencionada luego en los comentarios.

7 Conclusion

Como conclusiones acerca de nuestra implementación, se puede ver que es necesario la creación de elementos extra para poder implementar una variedad de instrucciones del mismo tamaño que MIPS tiene, lo cual justifica los cambios que realizamos en nuestro caso, lo cual tambien implica un aumento en costo y en espacio ocupado por los mismos. MIPS ha logrado compactar todos los elementos necesarios en una sola arquitectura, por lo que es claramente mejor que nuestra implementación. Aun sobre todas estas afirmaciones y diferencias entre implementaciones, consideramos que el objetivo final de hacerla nosotros mismos fue alcanzada, la cual sería darnos cuenta que efectivamente, los sistemas físicos lógicos de computación incluso para alguien especializado en software es no solo de alta importancia, si no también de alta complejidad. Podemos afirmar que hemos concluido el datapath, el informe, y el curso, con una noción más completa y profesional de las implicaciones de hardware en los dispositivos tecnologicos, y la importancia de las arquitecturas de la arquitectura de computadoras.

8 Comentarios

Entre las dificultades o curiosidades mas resaltantes en la implementación de este procesador fueron: la muy pequeña cantidad de memoria dentro del datamemory y la muy pequeña cantidad de instrucciones que se han generado al hacer las pruebas. Dado que los registro de por si son de 32 bits, el numero de registros que pueden existir son muchas mas que solo 32 o 64 registros. Esto solo conlleva a futuros errores en ubicacion de memoria, pues si alguno de los registros indica un valor mayor a la cantidad de registros que tenemos, se generan errores al hacer un jump register, store o loads. Finalmente, aunque fue implementado, consideramos para este datapath podria ser innecesario hacer explicitamente un subi, pues una suma inmediata con un numero negativo debería tener el mismo funcionamiento utilizando menos hardware.