Informe de práctica de laboratorio:

Microprocesador MIPS segmentado

Arquitectura de Computadoras I - UNTREF - 2017

Integrantes:

- Álvarez Felipe
- Capece Cintia
- Fiorito Hernán

Docentes:

- Leiva Lucas
- Vázquez Martín

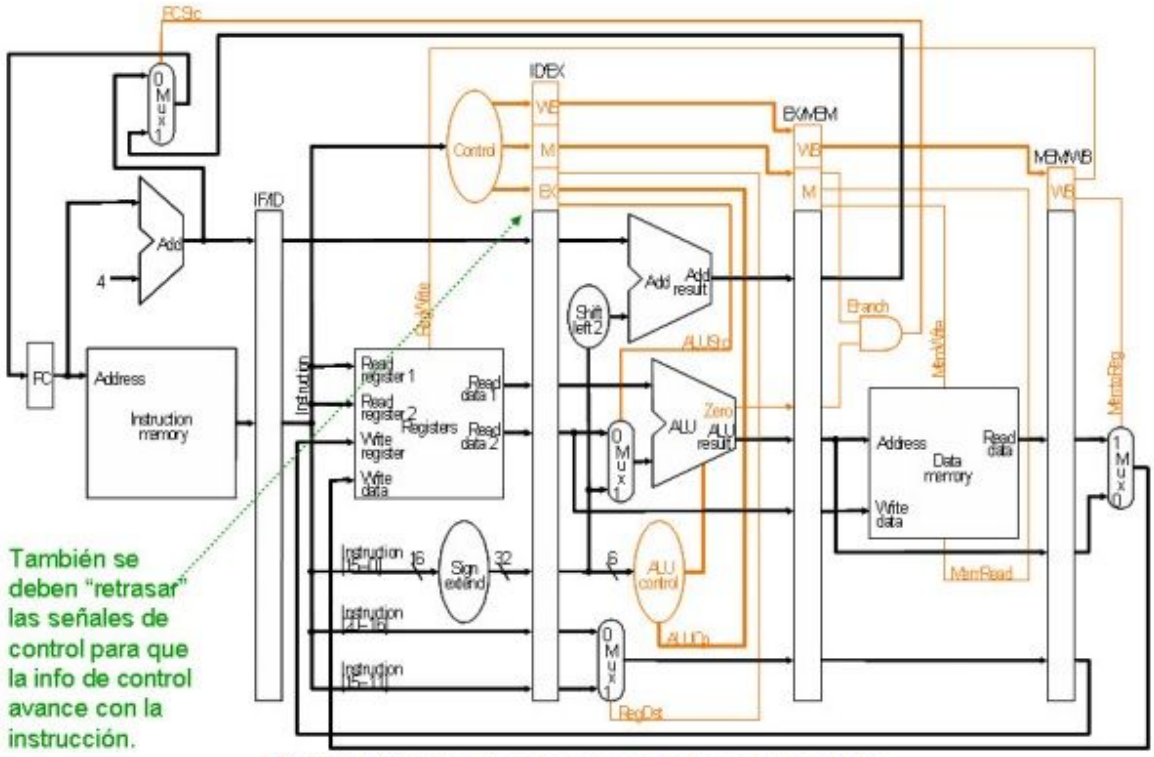
Introducción:

Un procesador MIPS segmentado se destaca por su mejor rendimiento en términos de productividad, ya que un procesador MIPS segmentado ideal alcanza a ejecutar una instrucción por ciclo (sin embargo, el tiempo de ciclo se determina basándose en la etapa que más tarda). A su vez, como desventajas, la arquitectura del MIPS introduce riesgos de datos y de control que necesitan ser salvados mediante lógica adicional, implementando una unidad de adelantamiento de datos y otra unidad de detección de riesgos respectivamente. Para los propósitos de está práctica, se supone un funcionamiento ideal y se obviaron las unidades antes mencionadas.

Los procesadores MIPS utilizan arquitecturas RISC, que implican un conjunto de instrucciones reducidos y de longitudes uniformes; y arquitecturas Harvard, que indican que la memoria de datos e instrucciones están separadas.

En el presente trabajo práctico se implementó un microprocesador MIPS segmentado mediante la implementación de los componentes del procesador a través el lenguaje de modelado de hardware VHDL y utilizando el programa "*Model Sim 10.0d"* para realizar las simulaciones y pruebas correspondientes.

El procesador a implementar, junto con sus componentes, son detallados en la siguiente figura:



El desarrollo se realizó por etapas correspondientes a cada una de las etapas de segmentación (IF, ID, EX, MEM, WB). El procesador implementado en esta práctica soporta los siguientes tipos de instrucciones: ADD, SUB, AND, OR, SLT, LW, SW, BEQ y LUI.

Decisiones de diseño:

Los componentes del procesador MIPS segmentado (exceptuando las memorias) se implementaron como componentes internos del procesador (es decir, se declararon y se instanciaron dentro del archivo VHDL del procesador "*processor.vhd*").

Las memorias de datos e instrucciones, el banco de registros y la ALU ya fueron implementados por la cátedra como base, así como también se facilitó un programa de prueba (que se cargan en la memoria de instrucciones) y un conjunto de datos (que se cargan en la memoria de datos) para probar el funcionamiento del procesador. Sin embargo, se decidió hacer otro programa que incluye instrucciones BEQ y LUI (ya que estas instrucciones estaban ausentes en el programa dado) con el propósito de mostrar que tales instrucciones son soportadas por el procesador realizado.

A continuación se detallan los componentes restantes que fueron necesarios implementar para obtener un procesador MIPS segmentado funcional:

Contador de programa (Program Counter)

El contador de programa se implementó como un registro de 32 bits con reset asíncrono. Los 32 bits son necesarios para guardar una instrucción, ya que la longitud de palabra de cada una son de 4 bytes. El componente se definió en el archivo "*pc.vhd*".

Sumador

Se declaró un sumador como un componente (en el archivo "*adder.vhd*") y el mismo es instanciado como dos componentes diferentes dentro del procesador: Por un lado, en la etapa IF donde es utilizado para calcular el PC de la siguiente instrucción del programa (PC+4). Por el otro, en la etapa EX para calcular la dirección de salto efectiva (para el caso de instrucciones de salto).

Extensión de signo

Recibe un intervalo de 16 bits de la instrucción y lo extiende a 32 bits. Este dato será utilizado por el sumador de la etapa EX para calcular la dirección de un salto, luego de ser desplazada dos posiciones a izquierda. Declarado en el archivo "*signextension.vhd*".

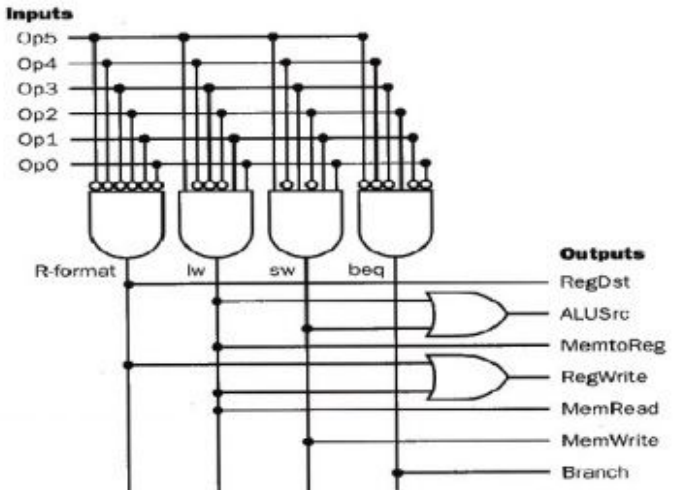
Unidad de control

Recibe el identificador de la instrucción, correspondiente a los primeros 6 bits de una instrucción, y genera las señales de control necesarias para la correcta ejecución de las instrucciones. De esta unidad salen tres buses: EX_signals que se utiliza en la etapa EX, MEM_signals que se utiliza en la etapa MEM y WB_signals que se utiliza en la etapa WB. Estos buses se componen de la concatenación de los bits con los valores de las señales que se utilizan en las tres etapas mencionadas de la siguiente manera:

EX_signals (4 bits): RegDst (bit 3), ALUSrc (bit 2), alu_op (bits 1-0).
MEM_signals (3 bits): Branch (bit 2), MemRead (bit 1), MemWrite (bit 0).
WB_signals (2 bits): RegWrite (bit 1), MemtoReg (bit 0).

La unidad de control se implementó siguiendo el modelo de compuertas lógicas y utilizando señales y se agregó lógica adicional para generar las señales de control que recibe la unidad de control de ALU.

instrucción	RegDst	ALU Src	Memto Reg	Reg Write	Mem Read	Mem Write	Branch
R-type	1	0	0	1	0	0	0
LW	0	1	1	1	1	0	0
SW	X	1	X	0	0	1	0
BEQ	X	0	X	0	0	0	1



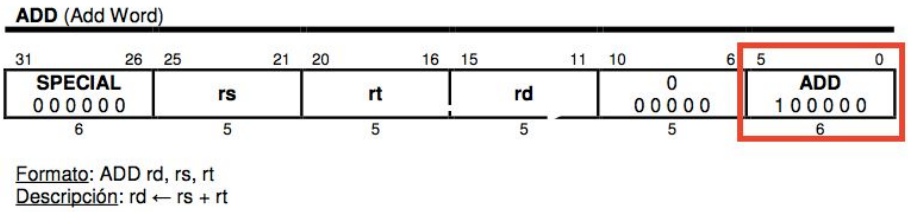
El componente está declarado en el archivo "*control_unit.vhd*".

Shift left 2

Desplaza dos posiciones a izquierda el binario de entrada, multiplicando por 4 el valor recibido. Se utiliza para calcular dirección de un salto. Se implementó como un componente y se declaró en el archivo "*shiftleft2.vhd*".

Unidad de control de la ALU

Recibe la señal de control saliente de la unidad de control y los últimos 6 bits de las instrucciones para generar una señal para la ALU que le indica qué operación realizar. La siguiente tabla muestra las operaciones indicadas para cada instrucción. Se agregó soporte para la instrucción inmediata LUI.



OP	ALUOp	INSTR	FUNCT	op deseada	ALU input
LW	0 0	load word	x x x x x x	add	0 1 0
SW	0 0	store word	x x x x x x	add	0 1 0
BEQ	0 1	beq	x x x x x x	sub	1 1 0
r-type	1 0	add	1 0 0 0 0 0	add	0 1 0
r-type	1 0	sub	1 0 0 0 1 0	sub	1 1 0
r-type	1 0	and	1 0 0 1 0 0	and	0 0 0
r-type	1 0	or	1 0 0 1 0 1	or	0 0 1
r-type	1 0	slt	1 0 1 0 1 0	slt	1 1 1

El componente fue declarado en el archivo "*alu_control.vhd*".

Registros de segmentación

Se modelaron cuatro registros de segmentación que delimitan las etapas del procesador: los registros IF/ID, ID/EX, EX/MEM y MEM/WB. Tienen como objetivo retrasar las señales de control y que éstas acompañen a la instrucción a lo largo de todas las etapas. Como decisión de diseño, se implementaron los registros como componentes y se declararon en los archivos "*regIF-ID.vhd*", "*regID-EX.vhd*", "*regEX-MEM.vhd*" y "*regMEM-WB.vhd*".

Multiplexores de 32 y 5 bits

Se modelaron como componentes. Para el MIPS segmentado son necesarios multiplexores de 32 y 5 bits (declarados en los archivos "*mux_2_1_32bits.vhd*" y "*mux_2_1_32bits.vhd*" respectivamente).

El primero se utiliza en tres etapas: IF, EX y WB. En la etapa IF se utiliza para seleccionar el PC de la siguiente instrucción, ya sea el PC de la instrucción siguiente secuencialmente en el programa o el PC de la instrucción de un salto (como dato adicional, la señal que controla este multiplexor se implementó como un proceso concurrente dentro del procesador). En la etapa EX se utiliza para seleccionar la fuente del segundo operando de la ALU, pudiendo ser los datos de un registro del banco de registros o la salida del componente de extensión de signo. Finalmente, en la etapa WB se utiliza para seleccionar qué dato se escribirá en el banco de registros, si el extraído de la memoria de datos o el resultado de una operación aritmética de la ALU.

El segundo sólo se utiliza en la etapa EX para seleccionar uno de los campos de la instrucción que indica el número del registro donde se escribirán datos en el banco de registros.