Laboratorio de Fundamentos de Microprocesadores
Curso 2017-2018

Práctica 5:
Diseño del Microprocesador MIPS

La entrega estará compuesta de un único fichero que incluya:

- Todos los ".vhd" y ".asm" solicitados.
- Un fichero llamado "autores.txt" que contenga los nombres completos e identificadores de la pareja.

Todos estos ficheros deben entregarse en un único fichero comprimido.
Los ejercicios deberán subirse a la actividad correspondiente de *Moodle*, identificada por el número de práctica (ej. La práctica 2 se entrega en la tarea «Entrega Práctica 2»). Sólo es necesario que suba los ficheros un miembro de la pareja. En caso de que ambos miembros de la pareja suban un fichero a *Moodle*, se corregirá sólo uno.
Todos los ficheros deberán contener en la cabecera los nombres de los autores y el identificador de la pareja. Asimismo, se recuerda que los ficheros deberán estar correctamente comentados. Dado que no se entrega una memoria adicional, los comentarios de los ficheros deben ser claros y concisos. No atenerse a estas normas implicará una penalización en la nota.

En esta práctica se realizarán **dos entregas**. La **primera entrega** consistirá en los **ejercicios 1 y 2**, y la **segunda entrega** consistirá en **todos los ejercicios** (incluyendo otra vez el 1 y el 2, cuya solución puede cambiarse). **Las parejas que no hayan entregado en plazo y forma serán calificadas con un cero en la entrega correspondiente.**

Ejercicio 1. Adaptación al juego de instrucciones presente (1 punto)

En la práctica 4 se escribió un programa que realizaba operaciones entre vectores. Para la implementación de dicho programa, llamado Vectores, se utilizó todo el juego de instrucciones posibles del simulador *Mars*. Sin embargo, el microprocesador que se está desarrollando en VHDL tiene un conjunto reducido de instrucciones. Por ello es necesario adaptar el código del fichero "*vectores.asm*" que se entregó en la práctica previa, de tal forma que sólo se utilicen las instrucciones descritas en la **Tabla II**, pero sin cambiar la funcionalidad del programa.

El fichero *"vectores.asm"* deberá ser modificado y ensamblado utilizando el simulador de la práctica 4. Después se deberá exportar el contenido de los segmentos de código (*.text*) y de datos (*.data*) y se completarán las memorias de datos y de programa (ficheros "*MemDataVectores.vhd*" y "*MemProgVectores.vhd*"). Se entregan unas memorias ("*MemDataPlantilla.vhd*" y "*MemProgPlantilla.vhd*") que servirán de ejemplo para realizar las memorias pedidas en este ejercicio. No se olvide de cambiar, tanto los nombres de los ficheros a los pedidos como los nombres de las entidades.

Posteriormente en el ejercicio 3, las memorias "*MemDataVectores.vhd*" y "*MemProgVectores.vhd*" serán utilizadas para probar el funcionamiento del programa en el procesador desarrollado.

Los valores de vectores y, por tanto, el contenido de la memoria de datos debe ser el siguiente:

Posición de memoria	Etiqueta	Valores
x2000	N	6
x2004	A	30,10,250,120,10,50
x201C	B	1,-8,50,10,-190,-250
x2034	C	

Tabla I

Tenga en cuenta que **los datos han cambiado** frente a la práctica 4. Si en la anterior práctica ha declarado alguna otra variable en el segmento de datos, debe eliminarla. Tras cambiar los vectores a sus nuevos valores y tamaños, se recomienda **probar el programa modificado con el simulador Mars. Posteriormente rellene las memorias.**

Objetivo

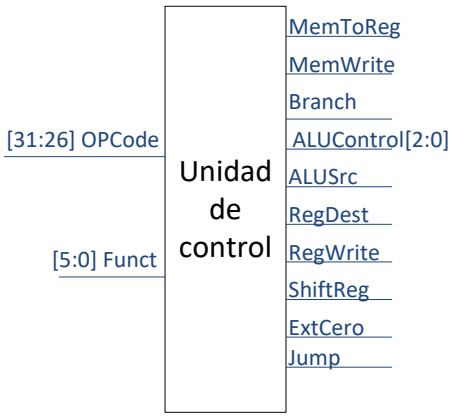
Adaptar el programa *Vectores* de la práctica 4 para que sea compatible con el conjunto de instrucciones que se van a implementar en el microprocesador realizado en esta práctica. Preparar las memorias de programa y datos para posibilitar su simulación.

Entrega

En este ejercicio se deben entregar los ficheros "*Vectores.asm*" que contendrá el código ensamblador del problema (corrigiendo errores que fueran detectados en la práctica 4), el fichero "*MemProgVectores.vhd*" que contendrá la memoria de programa de la arquitectura con el código implementado y el fichero "*MemDataVectores.vhd*" que contendrá el contenido de la memoria de datos del sistema.

Ejercicio 2. Diseño de la unidad de control (2 puntos)

Diseñar en VHDL la unidad de control del microprocesador MIPS uniciclo. La interfaz de este módulo se presenta en la imagen de la derecha.



Este módulo genera 11 señales de control. Estas señales de control definen el comportamiento del resto de elementos que componen el microprocesador.
Como entrada recibe: el campo *OPCode* y *Funct* de la instrucción.

A continuación se muestran las instrucciones que debe ejecutar el microprocesador implementado en VHDL:

OPCode	Nombre	Descripción	Operación
000000	R-Type	Instrucciones con formato R-Type	Varias. Se verán en la siguiente tabla
000010	j	Salto incondicional	PC = JTA
000100	beq	Bifurca si igual (Z = 1)	Si ([rs] == [rt]); PC =BTA
001000	addi	Suma con dato inmediato	[rt] = [rs] + SigImm
001100	andi	AND con dato inmediato	[rt] = [rs] & ZeroImm
001110	xori	XOR con dato inmediato	[rt] = [rs] ^ ZeroImm
100011	lw	Lee una palabra de memoria	MEM ([rs]+SigImm) => [rt]
101011	sw	Escribe una palabra en memoria	[rt] => MEM ([rs]+SigImm)
001010	slti	Set on less than (inmediato)	[rs]<[SigImm] ? [rt]=1 : [rt]=0

R-TYPE

Funct	Nombre	Descripción	Operación
100100	and	Función AND	[rd] = [rs] & [rt]
100000	add	Sumar	[rd] = [rs] + [rt]
100010	sub	Restar	[rd] = [rs] - [rt]

100111	nor	Función NOR	[rd] = ~ ([rs] | [rt])
100110	xor	Función XOR	[rd] = [rs] ^ [rt]
101010	slt	Set on less than	[rs]<[rt] ? [rd]=1 : [rd]=0
000011	sra	Shift right arithmetic	[rd] = [rt] >>> shamt

Tabla II

Para facilitar la implementación de la unidad de control, se recomienda rellenar la siguiente tabla con los valores que debe tener cada señal de control para cada una de las instrucciones:

INSTRUCCIÓN	MemToReg	MemWrite	Branch	ALUControl	ALUSrc	RegDest	RegWrite	ExtCero	Jump	ShiftReg
R-Type										
Lw										
Sw										
Beq										
Lógicas Inm										
Aritméticas Inm										
J										
SRA										

Tabla III

Explicación de las señales de control

Las señales de control servirán para indicar a cada elemento del microprocesador la función que debe realizar. El significado de las señales de control a implementar es el que se detalla a continuación:

- MemToReg: Indica si un dato de la memoria de datos irá al banco de registros para ser guardado. En cualquier caso, RegWrite tiene el control final sobre la escritura en el banco de registros.
- MemWrite: Write enable de la memoria de datos. Cuando se habilita, se escribe un dato en la memoria de datos.
- Branch: Indica que hay una instrucción de tipo beq, independientemente de que se cumpla la condición de salto.
- ALUControl: Código de operación para la ALU.
- ALUSrc: Indica el operando 2 de la ALU. Cuando está a ‘1’, el operando 2 de la ALU recibe un dato inmediato. Cuando está a ‘0’, recibe el registro rt.
- RegDst: Indica en qué campo de la instrucción se encuentra la dirección del registro de escritura. Cuando está a ‘1’, la dirección se encuentra en los bits [15:11] de la instrucción. Cuando está a ‘0’, se encuentra en los bits [20:16].

- RegWrite: Write enable del banco de registros. Cuando se habilita, se escribe un dato en el banco de registros. El dato que se escribirá dependerá de la señal de control MemToReg.
- ExtCero: Indica si el dato inmediato de 16 bits de una instrucción I-type debe extenderse en cero o en signo. Esta señal sólo tendrá importancia en las instrucciones que usen dicho dato inmediato, siendo indiferente en el resto.
- Jump: Indica si se va a realizar un salto j.
- ShiftReg: Indica el operando 1 de la ALU. Cuando está a '1', el operando 1 de la ALU recibe el registro RT (sólo tiene sentido en operaciones de desplazamiento). Cuando está a '0', recibe el registro rs.

Objetivo

Comprender el problema propuesto. Plantear e implementar una solución para dicho problema. Para probar la funcionalidad de la unidad de control se suministra el banco de pruebas "UnidadControlTb.vhd".

Entrega

Se debe entregar el fichero "UnidadControl.vhd", que contendrá la definición de la entidad y la arquitectura de la unidad de control.

Ejercicio 3. Diseño completo del microprocesador (7 puntos)

En este ejercicio se deberá realizar el diseño completo del microprocesador. Para llevar a cabo este ejercicio se deberán incluir los módulos de la ALU (ALUMIPS.vhd) y del banco de registros (RegsMIPS.vhd) (corrigiendo errores que fueran detectados en la práctica 3).

El sistema que se está desarrollando sigue la arquitectura Harvard, con una memoria para el código y otra para los datos. Las entradas del microprocesador "MicroMIPS" son una señal de *Reset* activa a nivel bajo (NRst), el reloj (Clk), la entrada de memoria de programa (MemProgData) y la entrada de memoria de datos (MemDataDataRead). Como salidas, la señal que habilita la escritura en memoria de datos (MemDataWe), la dirección de memoria de programa (MemProgAddr), la dirección de memoria de datos (MemDataAddr) y el bus de escritura en memoria (MemDataDataWrite).

Al final de este enunciado se puede encontrar un esquemático correspondiente al microprocesador que debe implementarse. Este esquemático corresponde a las transparencias de la unidad 4, aunque ya están añadidos los elementos lógicos necesarios para implementar las instrucciones que no figuran en dicha unidad.

Doble validación

Para realizar las pruebas del microprocesador se proporcionan las entidades y las arquitecturas de las memorias de programa y de datos. La memoria de programa ("*MemProgMIPS.vhd"*) describe una memoria ROM que recoge las instrucciones del programa. La memoria de datos ("*MemDataMIPS.vhd"*) es una memoria de lectura asíncrona y escritura síncrona en flanco de subida. El testbench que se suministra ("*MicroMIPSTb.vhd*") instancia la CPU a diseñar por el alumno en el ejercicio 3 y las memorias suministradas.

En este ejercicio se debe ejecutar el programa y comprobar que el resultado obtenido por el microprocesador diseñado es el mismo que el que se obtiene utilizando el simulador MARS cuando se ejecuta el fichero en ensamblador *Ejercicio3.asm*. Para comprobar el correcto funcionamiento del micro se sugiere utilizar las formas de onda incluidas en "ej3inicial_wave.do" (script para la visualización de señales y el banco de registros en *ModelSim*), añadiendo las señales que se considere oportuno.

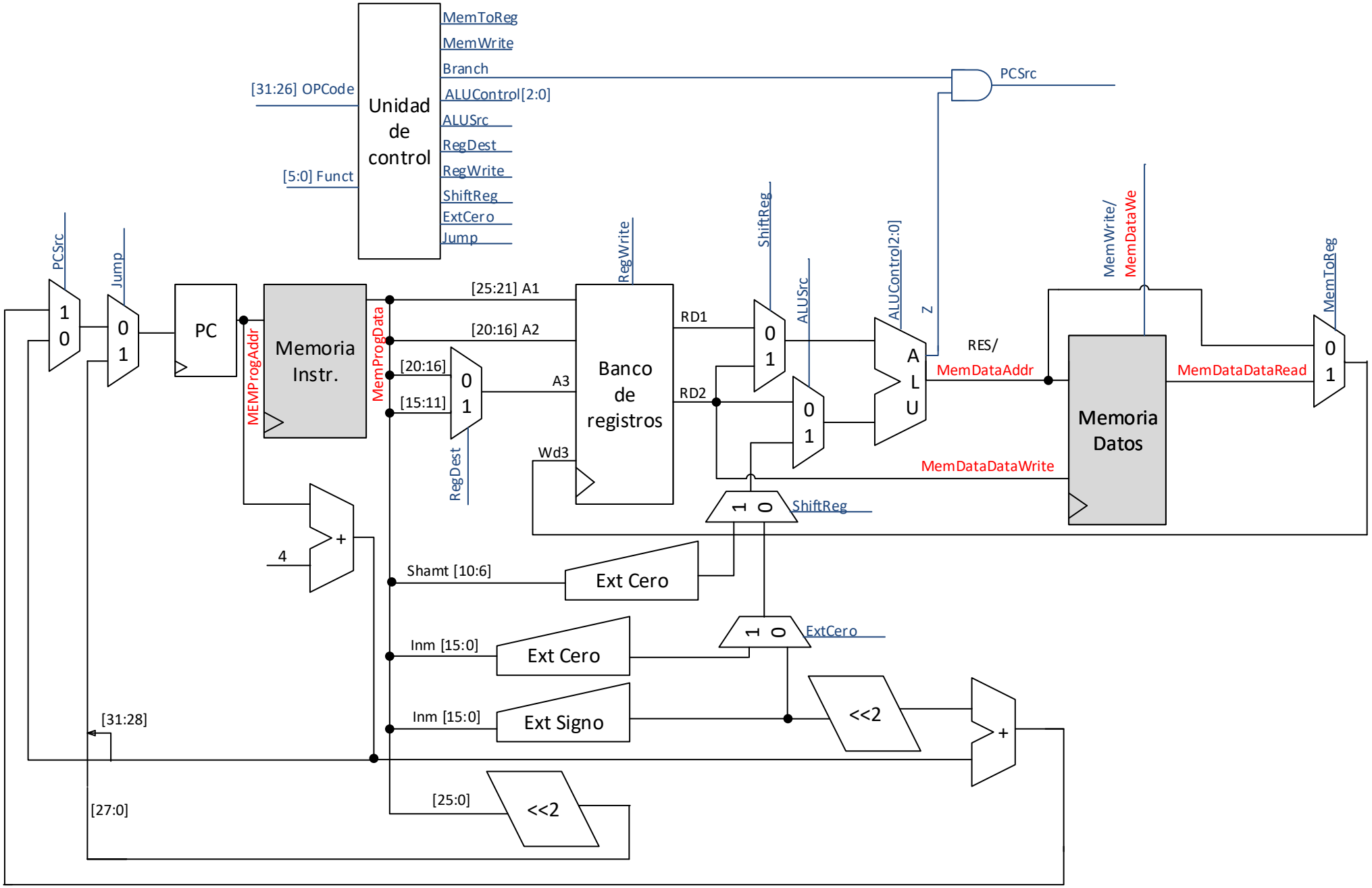
Adicionalmente, es necesario realizar la validación del programa Vectores realizado en el ejercicio 1 de la presente práctica. Para ello, se debe simular el procesador con las memorias preparadas en el ejercicio 1 ("*MemProgVectores.vhd*" y "*MemDataVectores.vhd*") y el testbench proporcionado en "MicroMipsVectoresTb". Para poder comprobar el vector donde se guardará el resultado, se sugiere utilizar el fichero "waveVectores.do", añadiendo las señales que se considere oportuno.

Objetivo

Comprender el problema propuesto. Plantear e implementar una solución para dicho problema. Ejecutar el banco de pruebas con las memorias entregadas para corregir los errores cometidos durante la fase de diseño. Posteriormente, ejecutar el banco de pruebas con las memorias desarrolladas en el ejercicio 1 ("*MemDataVectores.vhd*" y "*MemProgVectores.vhd*").

Entrega

Se debe entregar el fichero "*MicroMIPS.vhd*", que contendrá la definición de la entidad y la arquitectura del microprocesador. Además se deberán entregar los ficheros "*ALUMIPS.vhd*", "*RegsMIPS.vhd*", "*UnidadControl.vhd*" y si se han añadido otros módulos interiores en el micro, todos los ficheros VHDL de dichos módulos.



Señales de control **Entradas/Salidas** **Datos**