Laboratorio de Estructura de Computadores
Curso 2020-2021

Práctica 4:
Diseño del Microprocesador MIPS

Ejercicio 1. Adaptación al juego de instrucciones presente

En la práctica 3 se escribió un programa que realizaba operaciones entre vectores (Otros Ejercicios). 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 "*OtrosEjercicios_BucleFor.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 *"OtrosEjercicios_BucleFor.asm"* deberá ser modificado y ensamblado utilizando el simulador Mars. Después se deberá exportar el contenido de los segmentos de código (*.text*) para completar la memoria de programa (fichero "*MemProgVectores.vhd*") utilizando como ejemplo "*MemProgMIPS.vhd*".

A diferencia del ejercicio propuesto en la Práctica 3 anterior, mientras que el programa debe ser, aunque adaptado, el mismo que desarrolló en dicha práctica, para este ejercicio el contenido de la memoria de datos "*MemDataVectores*", contiene los datos indicados en la tabla adjunta, usando como ejemplo "*MemDataMIPS.vhd*".

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	2,4,6,8,10,12
x201C	B	-1,-5,4,10,1,-5
x2034	C	

Tabla I

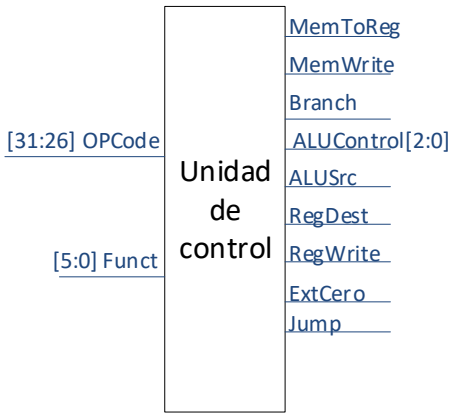
Posteriormente en el ejercicio 3, las memorias "*MemDataVectores.vhd*" y "*MemProgVectores.vhd*" serán utilizadas para probar el funcionamiento del programa en el procesador desarrollado.

Objetivo

Si el programa de vectores con bucle for realizado en la práctica 3 tuviera algún error, debe corregirlo. Posteriormente, se debe adaptar el programa "*OtrosEjercicios_BucleFor.asm"* de la práctica 3 para que sea compatible con el conjunto de instrucciones que se van a implementar en el microprocesador realizado en esta práctica (ver Tabla I). Por último, se debe preparar las memorias de programa, "*MemProgVectores.vhd*", y datos, "*MemDataVectores.vhd*", para posibilitar la simulación de procesador.

Ejercicio 2. Diseño de la unidad de control

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 9 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
001101	ori	OR 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]

100110	xor	Función XOR	[rd] = [rs] XOR [rt]
100101	or	Función OR	[rd] = [rs] | [rt]
101010	slt	Set on less than	[rs]<[rt] ? [rd]=1 : [rd]=0

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
R-Type									
Lw									
Sw									
Beq									
Lógicas Inm									
Aritméticas Inm									
J									

Tabla III

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”.

Ejercicio 3. Diseño completo del microprocesador

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 2).

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.

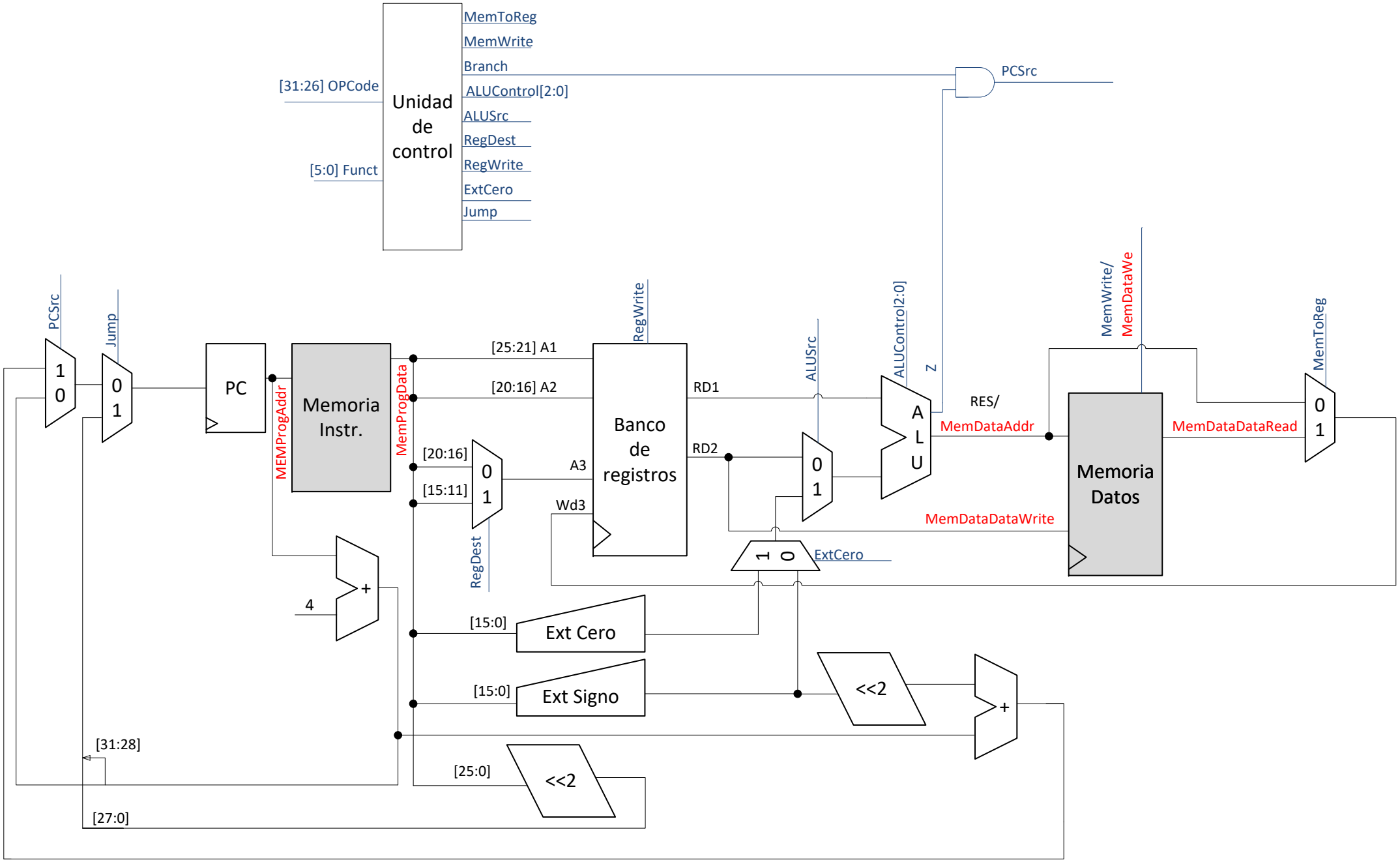
Nota: En la corrección de la práctica, para poder utilizar el archivo **.do** facilitado, al instanciar el Baco de Registros en el archivo MicroMIPS.vhd, es necesario que el nombre de dicha instancia sea GPR [GPR: RegsMIPS port map {…].

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 "Wave_Ejercicio3.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 de vectores con bucle for 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 "Wave_Vectores.do", añadiendo las señales que se considere oportuno. El cronograma facilitado solo representa los 10 primeros registros del GPR, del $0 al $9, por tanto, si ha utilizado otros registros, debe incorporarlos al cronograma dado.

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*").



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