Computer Structure Laboratory
Course 2019-2020

Task 4:
Microprocessor design MIPS

Exercise 1. Adaptation to the set of instructions

In task 3 a program that performed operations between vectors was designed (exercise 2). For the implementation of this program, named Vectors, the entire set of instructions of the Mars simulator was used. However, the microprocessor that is being developed in VHDL has a reduced set of instructions. For this reason, it is necessary to adapt the code of the file "vectors.asm" that was delivered in the previous task, only using the instructions described in Table II, but without changing the functionality of the program.

The file "vectors.asm" must be modified and assembled using the Mars simulator. Afterward, the content of the code (.text) and data (.data) segments must be exported and the data and program memories (files "MemDataVectores.vhd" and "MemProgVectores.vhd") must be completed. Some memories ("MemDataPlantilla.vhd" and "MemProgPlantilla.vhd") are provided, which will serve as an example to implement the memories requested in this exercise. Do not forget to change, both the names of the files and the names of the entities.

Later in exercise 3, the memories "MemDataVectors.vhd" and "MemProgVectores.vhd" will be used to test the operation of the program in the developed processor.

The values of vectors and, therefore, the content of the data memory should be as follows:

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	

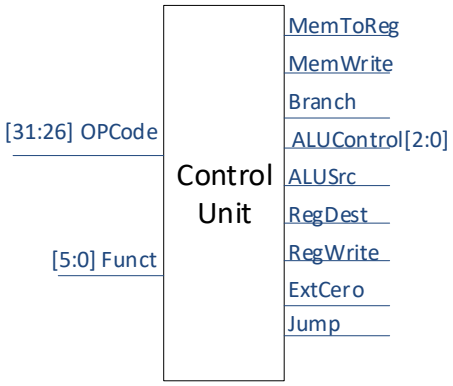
Table I

Objective

If the Vectors program performed in task 3 had an error, you must correct it. Subsequently, the Vectors program of task 3 must be adapted to be compatible with the set of instructions that will be implemented in the microprocessor done in this practice. Finally, you must prepare the program memories, "MemProgVectores.vhd", and data, "MemDataVectores.vhd", to simulate the processor.

Exercise 2. Design of the control unit

Design in VHDL the control unit of the unicycle microprocessor MIPS. The interface of this module is presented in the image on the right. This module generates 9 control signals. These control signals define the behavior of the rest of the elements that set the microprocessor.
The received inputs are: The OPCode field and Funct of the instruction.
Below are the instructions that the microprocessor must execute in VHDL:



OPCode	Name	Description	Operation
000000	R-Type	R-Type format instructions	Several. They will be seen in the following table
000010	j	Unconditional jump	PC = JTA
000100	beq	Burnch if it is equal (Z = 1)	Si ([rs] == [rt]); PC =BTA
001000	addi	Add with immediate data	[rt] = [rs] + SigImm
001100	andi	AND with immediate data	[rt] = [rs] & ZeroImm
001101	ori	OR with immediate data	[rt] = [rs] | ZeroImm
100011	lw	Read a word from memory	MEM ([rs]+SigImm) => [rt]
101011	sw	Write a word in memory	[rt] => MEM ([rs]+SigImm)
001010	slti	Set on less than (immediate)	[rs]<[SigImm] ? [rt]=1 : [rt]=0

R-TYPE

Funct	Name	Description	Operation
100100	and	Function AND	[rd] = [rs] & [rt]
100000	add	Add	[rd] = [rs] + [rt]
100010	sub	Subtract	[rd] = [rs] - [rt]
100111	nor	NOR Function	[rd] = ~ ([rs] | [rt])
100101	or	OR Function	[rd] = [rs] | [rt]
101010	slt	Set on less than	[rs]<[rt] ? [rd]=1 : [rd]=0

Table II

To facilitate the implementation of the control unit, it is recommended to fill in the following table with the values that each control signal must have for every instruction:

INSTRUCTION	MemToReg	MemWrite	Branch	ALUControl	ALUSrc	RegDest	RegWrite	ExtCero	Jump
R-Type	0	0	0	ddd	0	1	1	d	0
Lw	1	0	0	010(+)	1	0	1	0	0
Sw	x	1	0	010(+)	1	x	0	0	0
Beq	x	0	1	110 (-)	1	x	0	0	0
Logic Imm	0	0	0	ddd	1	0	1	1	0
Arithmetic Imm	0	0	0	ddd	1	0	1	0	0
J	x	0	0	xxx	x	x	0	x	1

Table III

Objective

To understand the proposed problem. To set out and implement a solution to this problem. To test the functionality of the control unit a test bench is provided “UnidadControlTb.vhd”.

Exercise 3. Complete design of the microprocessor

In this exercise, the complete design of the microprocessor must be carried out. To accomplish this exercise, the modules of the ALU (ALUMIPS.vhd) and the register bank (RegsMIPS.vhd) must be included (correcting errors that were detected in task 2).

The system that is being developed follows the Harvard architecture, with one memory for the code and another for the data. The inputs of the microprocessor "MicroMIPS" are a low active Reset signal (NRst), the clock (Clk), the program memory input (MemProgData) and the data memory input (MemDataDataRead). As outputs, the signal that enables data memory writing (MemDataWe), the program memory address (MemProgAddr), the data memory address (MemDataAddr) and the write memory bus (MemDataDataWrite).

At the end of this guideline you can find a schematic corresponding to the microprocessor that should be implemented. This schematic corresponds to the theory slides of unit 4, although the logical elements necessary to implement the instructions (jal and jr) have been added.

Double validation

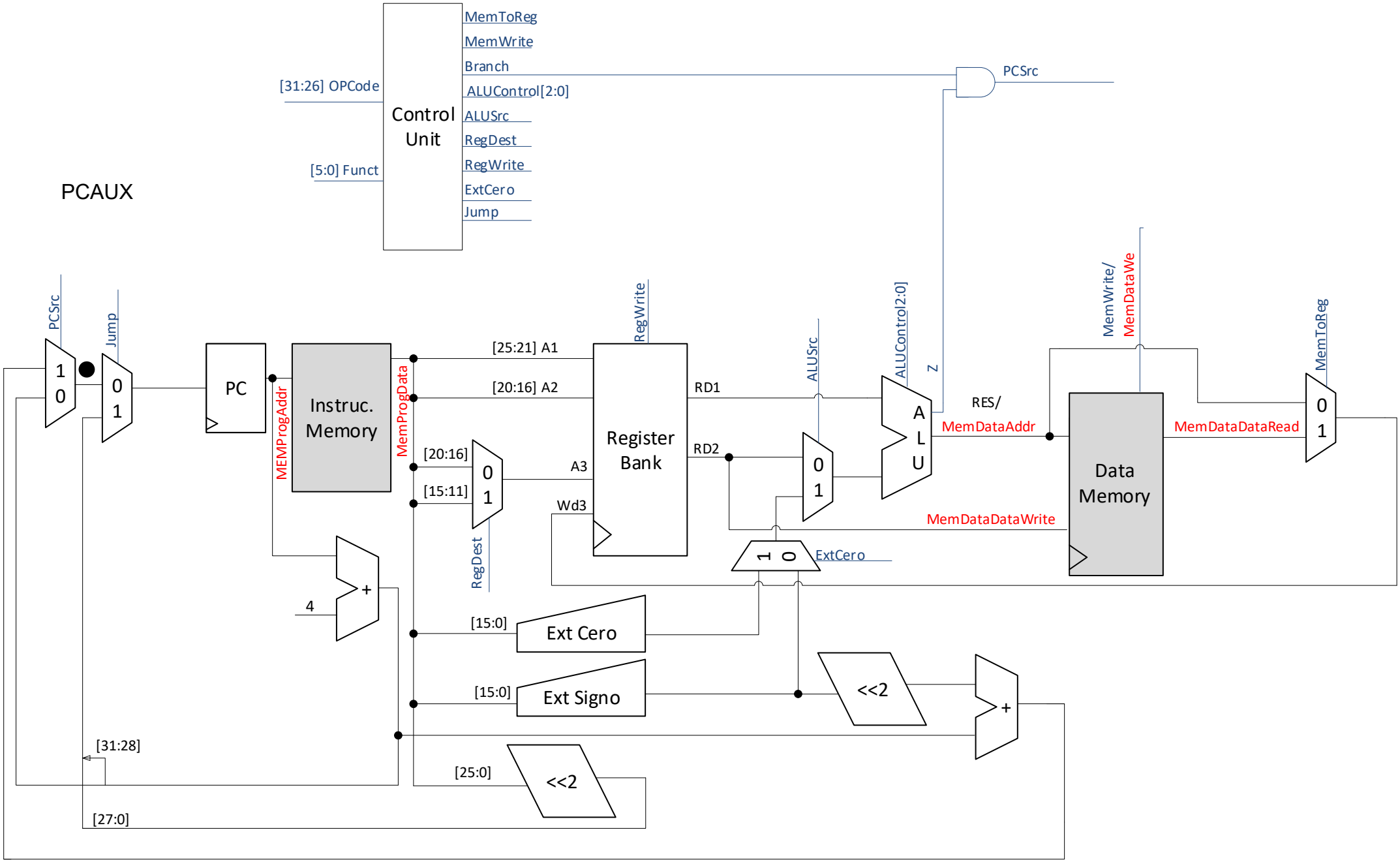
Entities and the architecture of the program and data memories are provided to perform the microprocessor tests. The program memory ("MemProgMIPS.vhd") describes a ROM memory that contains the instructions of the program. The data memory ("MemDataMIPS.vhd") is an asynchronous reading memory and synchronous rising edge writing. The given testbench ("MicroMIPSTb.vhd") instantiates the CPU to be designed by the student in exercise 3 and the supplied memories.

In this exercise, you must run the program and verify that the result obtained by the designed microprocessor is the same as that obtained using the MARS simulator when the assembly file Exercise3.asm is executed. To verify the proper performance of the micro it is suggested to use the waveforms included in "ej3inicial_wave.do" (script for the visualization of signals and register bank in ModelSim), adding the signals that you consider appropriate.

In addition, it is necessary to carry out the validation of the Vectors program done in exercise 1 of the present task. To do this, the processor must be simulated with the memories prepared in exercise 1 ("MemProgVectores.vhd" and "MemDataVectores.vhd") and the testbench provided in "MicroMipsVectoresTb". In order to check the vector where the result will be stored, it is suggested to use the file "waveVectores.do", adding the signals that you consider appropriate.

Objective

To understand the proposed problem. To set out and implement a solution to this problem. To run the testbench with the provided memories to correct the mistakes made during the design phase. Subsequently, to run the testbench with the memories developed in exercise 1 ("MemDataVectores.vhd" and "MemProgVectores.vhd").



Control signals **Inputs/Outputs** **Data**