Computer Structure Laboratory

Course 2019-2020

Task 2:

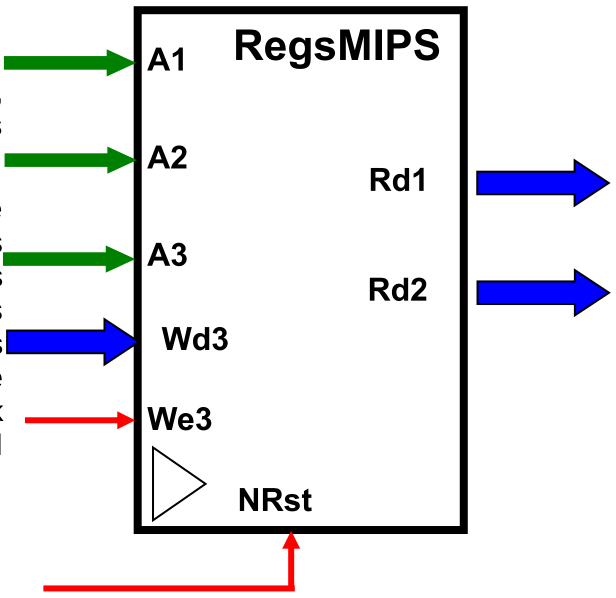
Simplified microprocessor, GPR and ALU

Exercise 1. Design of the register bank.

Design in VHDL the bank of general purpose registers (entity "RegsMIPS") of the microprocessor MIPS, whose interface is shown in the attached figure. The complete test is provided (file "*RegsMIPSTb.vhd*").

This bank consists of 32 registers of 32-bit each one, which are identified from 0 to 31. The 0 register always has a value of 0.

The bank allows the synchronous writing of a single record (destination record). The address of this register is given by the 5-bit A3 signal, which can take values between 0 and 31. The value that the register will take is indicated in the 32-bit input Wd3. The writing becomes effective when a rising edge of the clock occurs and the enabling signal We3 is activated. The register bank includes a low active NRst signal that will initialize all registers to value 0 when activated.



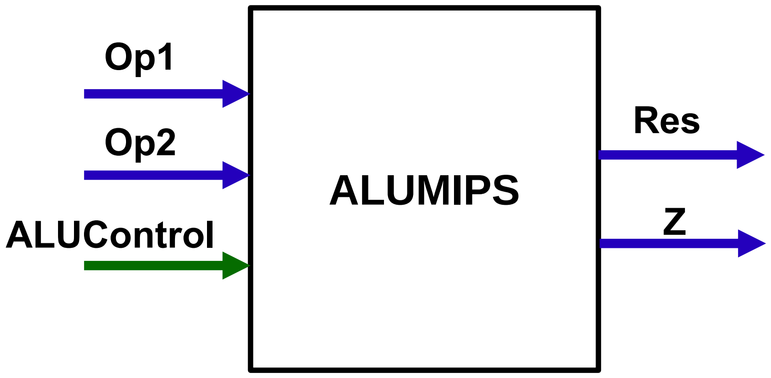
Additionally, this bank allows simultaneous asynchronous reading of two independent registers. The addresses of these registers are indicated respectively in the 5-bit input signals: A1 and A2. The values of the selected registers are copied to the outputs Rd1 and Rd2, respectively.

Objective

Design and verify the operation of the register bank through the provided testbench ("*RegsMIPSTb.vhd*"). This register bank will be used in the implementation of the complete microprocessor (Task 4).

Exercise 2. Design of the ALU.

Design a combinational arithmetic-logic unit ("ALUMIPS" entity) with two 32-bit data inputs (signals OP1 and OP2), a 3-bit operation selection input (ALUControl signal), a 32-bit data output (Res) and a one-bit status output (Z). This status signal corresponds to the flag of Z, which must be activated when the result is equal to zero.



The selection input values are encoding the following operations:

ALUControl[2:0]	Operación
000	OP1 AND OP2
001	OP1 OR OP2
010	OP1 + OP2
011	OP1 XOR OP2
100	SIN USAR
101	OP1 NOR OP2
110	OP1 - OP2
111	OP1 SLT OP2

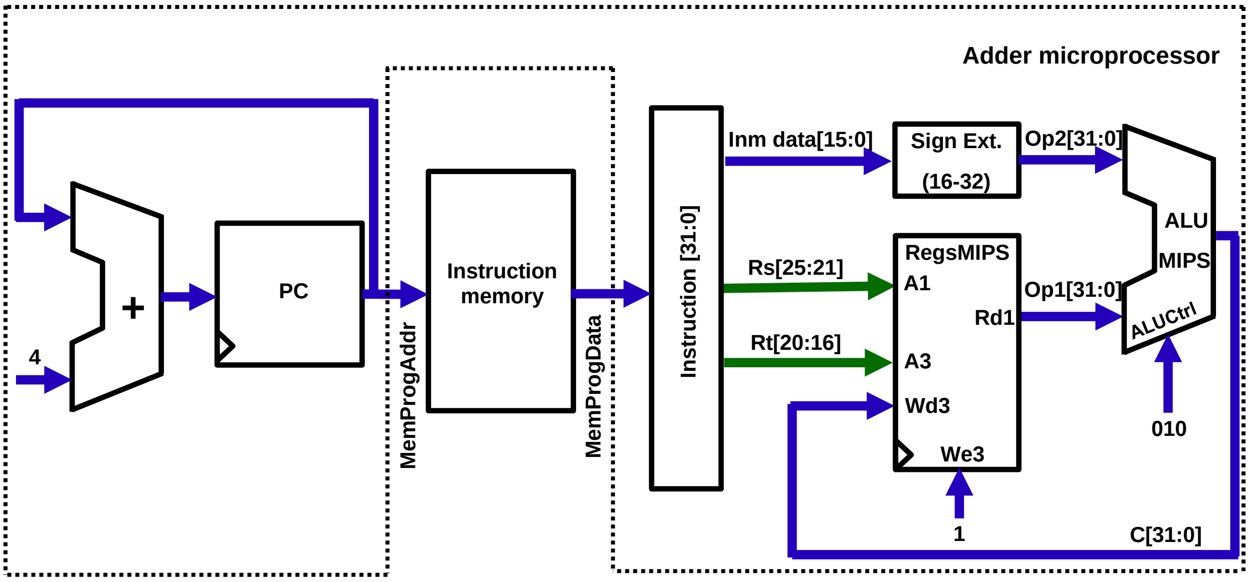
The slt (set less than) instruction sets the result to 0x1 when operand 1 is less than operand 2. Otherwise, the result will be 0x0. This instruction takes into account the sign of the input operands, that are two's complement numbers. To verify the correct design of the ALU, a testbench ("ALUMIPSTb.vhd") is provided and verifies the correct functioning of the ALU. The testbench will indicate if the ALU does not perform any operations correctly.

Objective

Design and verify the operation of an ALU through a testbench.

Exercise 3. Design of the adder microprocessor.

In this exercise, you should complete the design of a microprocessor that allows performing only sums between a register and an immediate data. To carry out this exercise, the different elements that provide the necessary functionality must be added to the register bank and connected as shown in the following figure.



The microprocessor to be developed does not include the instruction memory – supplied in the file "MemProgSUMA.vhd" – which must be incorporated into the project as an additional file, but which is only necessary to run the testbench test circuit.

The entity of the microprocessor is supplied in the file "MicroSuma.vhd". Apart from the clock (Clk) and the Reset (NRst, asynchronous and low-active that resets all the registers and the program counter to 0), the only entry of this entity is the 32-bit signal MemProgData, where the code of each instruction arrives from the memory. And the only output is the 32-bit signal MemProgAddr, where the program counter, PC, must be supplied. Therefore, the program counter must also have 32 bits. The tasks to develop inside this file are the following:

1. Instantiate the register bank module of exercise 1. The A2 address is not used, so it can receive any value. Likewise, the output Rd2 is not used so the term “open” can be used in the instantiation. Finally, the We3 entry must always be activated, since the results of the sums must be written in the register bank.
2. Instantiate the ALU written in exercise 2. The ALU will always add, so the ALUControl input should always have the value 010. Besides, the flag Z is not used, so the term “open” can be used in the instantiation.
3. Implement a sign extension from 16 to 32 bits.
4. The data path for the program counter (PC) register. The PC consists of a 32-bit register and the circuitry that increases it by 4 each clock cycle (counter type).

To verify the correct functioning of the microprocessor, a test bench is provided, "MicroSumaTb.vhd". In it the micro and the supplied memory are instantiated, causing the program to be executed, as follows:

```
R1 = R0 +10;
R2 = R1 + 5;
R3 = R2 + 25;
R0 = R0 + 5;
R4 = R3 - 5;
```

The correct functioning of the microprocessor must be checked by reading the values of the registers R0, R1, R2, R3, and R4, which are internal signals of the register bank.

Objective

Understand the microprocessor design, propose and implement a solution for the design. It is recommended to consult the Moodle document Computer Structure, which is located in Unit 2.

Other exercises.

- What changes would the register bank need if the reset were synchronous?
- What changes would the register bank need if the readings were synchronous?
- What changes would the ALU need if the N (negative) flag was implemented? Flag N should only be activated when the result is negative.
- Would the size of any input or output change if the ALU implemented 10 different operations?
- What changes should be made in the microprocessor to perform AND operations knowing that this operation requires that the immediate data is zero-extended?
- How to implement in the ALU the shift operations sll (*shift left logic*), srl(*shift right logic*) and sra (*shift right arithmetic*), where the maximum 31 positions shift is pointed out by 5 lsb of second operator?
- How much should the PC counter sum if each instruction was written in 64 bits?