TE2002B Challenge

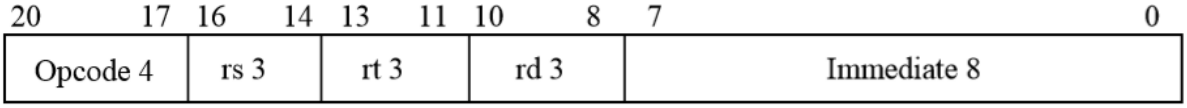
Semester: February-June 2021 (first 5-week period)
Campus: ITESM – Campus Qro.
Profesores: Rick L. Swenson y Jesús Cienfuegos
Challenge: Implement a fully functional embedded RISC Soft-processor on a DE10-Lite FPGA Board

Objective:

During this challenge, you will design, with your team, a single cycle microprocessor loosely based on the knowledge you have acquired to this point.

Instructions:

You will be designing a microprocessor that executes 16 different instructions. Each instruction is 21 bits long and consists of the following five fields: a 4-bit opcode, two 3-bit fields rs and rt that denote operand registers, one 3-bit field rd that denotes the destination register and an 8-bit immediate field. Note that although all the instructions share the same layout most instructions do not use all of the available fields.



The instructions that the processor will execute are given below:

1. **add rs rt rd:** (opcode = 0) Add the contents of the two registers denoted by the rs and rt fields and store the result in the register indicated in the rd field

2. **sub rs rt rd:** (opcode = 1) Subtract the contents of the register denoted by the rt field from the register denoted by the rs field and store the result in the register indicated in the rd field

3. **and rs rt rd:** (opcode = 2) And the contents of the two registers denoted by the rs and rt fields and store the result in the register indicated in the rd field

4. **or rs rt rd:** (opcode = 3) Or the contents of the two registers denoted by the rs and rt fields and store the result in the register indicated in the rd field

5. **xor rs rt rd:** (opcode = 4) Xor the contents of the two registers denoted by the rs and rt fields and store the result in the register indicated in the rd field

6. **not rs rd:** (opcode = 5) Invert the contents of the register denoted by the rs field and store the result in the register indicated in the rd field

7. **shl rs rd:** (opcode=6) Shift the contents of the register denoted by the rs field left one position (filling the vacated slot on the right with a zero) and store the result in the register indicated in the rd field

8. **shr rs rd:** (opcode=7) Shift the contents of the register denoted by the rs field right one position (filling the vacated slot on the left with a zero) and store the result in the register indicated in the rd field

9. **ld rd i:** (opcode=8) Load the register denoted by the rd field with the contents of the immediate field of the instruction

10. **spc rd:** (opcode=9) Store the current contents of the 8-bit program counter in the register indicated in the rd field

11. **beq rs rt i:** (opcode=10) If the contents of the two registers denoted by the rs and rt fields are equal then increment the program counter with the value in the immediate field

12. **ji i:** (opcode=11) Reset the program counter to the value found in the immediate field

13. **jr rs:** (opcode=12) Reset the program counter to the value stored in the register denoted by the rs field

14. **w7seg rs:** (opcode = 13) Write the contents of the register denoted by the rs field into another 8-bit register connected to the seven segment displays

15. **wleds rs:** (opcode = 14) Write the contents of the register denoted by the rs field into another 8-bit register connected to the 8 LEDs

16. **rsw rd:** (opcode = 15) Read the value of the least significant 8-switches and store the contents in the register denoted by the rd field

Question 1: For each of the 16 instructions indicate how the next value of the program counter (PC) should be determined.

Question 2: The conditional branch instruction requires us to compare the contents of two registers using the ALU. What ALU operation could we use to implement this comparison? Are there any other options?

Question 3: The PCInc signal controls whether the current PC value is added to the constant 1 or the immediate field of the current instruction. How should this signal be set during a beq instruction?

Based on your answers to these questions check complete the datapath for the processor for all instructions, giving one diagram per instruction.

In order to check the design of this processor we must provide an implementation of the control unit. The control unit is a combinational circuit which chooses the settings of all of the control lines in the circuit based on the opcode of the current instruction.

Question 4: Provide a table which indicates what the settings of all of the control signals in your circuit should be for each of the 16 instructions.