CS&A: Lab Sessions

Project: Datapath (2)

Ruben Van den Bossche

1BA INF - 2010-2011

1 Time Schedule

Projects are solved in pairs of two students. Projects build on each other, to converge into a unified whole at the end of the semester. During the semester, you will be evaluated three times. At these evaluation moments, you will present your solution of the past projects by giving a demo and answering some questions. You will immediately receive feedback, which you can use to improve your solution for the following evaluations.

For every project, you submit a small report of the project you made by filling in `verslag.html` completely. A report typically consists of 1000 words and a number of drawings/screenshots. Put all your files in a tgz archive, as explained on the course's website, and submit your report to the exercises on Blackboard.

- Report deadline: **December, 19 2010, 23u55**
- Evaluation and feedback: **December, 21 2010**

2 Project

1. In the previous assignment, we used the ALU OP-codes as instruction OP-codes and added two additional instructions (`lw` and `sw`). In order to add more than 16 instructions to our datapath, we need to alter our OP-codes and instruction set: next to the 14 ALU (R-type) instructions, we also support immediate instructions as well as branch and jump instructions.

   In order to translate from the instruction OP-code to the ALU OP-codes and to get all control lines right, you will have to add a *Control Unit* circuit to your datapath.

   - Input is the instruction OP-code (4 bits).
   - Outputs are the ALU OP-code as well as all control lines for i.e. the program counter, instruction and data memory, multiplexers and the register file.

   More information on the implementation of a control unit can be found in Section 4.4 of *Computer organization and design.*

2. Implement the instructions described in the table below. You will have to alter your datapath slightly in order to get the right operation codes for the R-type instructions and the lw/sw instructions you already implemented. Once done, your datapath can correctly execute a program written in machine language, as the behaviour of arithmetic, branching and memory operations is now fully implemented!

0-3	4	5	6	7	8	9	10	11	12	13	14	15	
0000	rs		rt		rd		funct						14 R-type Instructions[1]
0001	rs		rt		immediate (unsigned)								lui[3,5]: $rt = imm << 8
0010	rs		rt		immediate (unsigned)								ori[5]: $rt = $rs | imm
0011	rs		rt		immediate (signed[2])								addi: $rt = $rs + imm
0100	rs		rt		immediate (unsigned)								andi: $rt = $rs & imm
0101	rs		rt		immediate (signed[2])								lw: $rt = MEM[$rs+imm]
0110	rs		rt		immediate (signed[2])								sw: MEM[$rs+imm] = $rt
0111	target address												jump[4]: $pc = addr
1000	rs		rt		offset (signed[2])								jr[4]: $pc = $rs+imm
1001	rs		rt		offset (signed[2])								beq[4]: ($rs=$rt) ? $pc=$pc+1+imm
1010	rs		rt		offset (signed[2])								bne[4]: ($rs≠$rt) ? $pc=$pc+1+imm

[1] 14 R-type instructions from your ALU. The ALU opcode is given in the funct field.

[2] Two's complement.

[3] "Load upper immediate": put the 8-bit immediate in the upper 8 bits (shift left x8 and store in register).

[4] You will have to adapt your branch method to support the behaviour of these instructions.

[5] The `lui` and `ori` instructions can be used together to implement a `li` pseudo-instruction which loads a 16-bit immediate into a register.