Lab session Datapath 2

Group A: November 16, 2009
Group B: November 17, 2009

Work in the given groups of two. Submit your solutions to the respective assignment on Blackboard. The file name is:
`a06_s0XXXXX_s0XXXXX.tar.gz`
One of the group members commits your solution. Keep an eye on the deadline (see Blackboard)!

1 Project

Continue building your datapath until it can read, interpret and execute a stored program from your program counter RAM memory. Once done, your datapath can correctly run a program written in machine language, so the behaviour of arithmetic, branching and memory operations is fully implemented.

Implement the instructions described in the table below. Additionally, 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.

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

[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]Adapt your branch method if it behaves like $pc=$pc+imm (this can be simply done by putting the branch addition *behind* the default +1 addition).