Computer Systems and -architecture

Project Exam Retake

1 Ba INF 2015-2016

Bart Meyers
bart.meyers@uantwerpen.be

Don't hesitate to contact the teaching assistant of this course. You can reach him in room M.G.3.17 or by e-mail.

Time Schedule

Projects are solved individually. Projects build on each other, to converge into a unified whole at the end of the semester. At the evaluation moment, you will present your solution by giving a demo and answering some questions.

You will submit a solution for all seven projects from the first semester, with the differences explained in this project description. Covering all seven projects, you submit one report 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: **31 August 2016**
- Evaluation and feedback: **7 September 2016**

Project

Complete all seven projects from the first semester, with the differences explained below. If there is no mention of a certain assignment (e.g., carry-lookahead addition or finite state automata), you solve the original assignment.

Your datapath should support:

- data words (in register and data memory) of 8 bits;
- 8 registers. Register r0 and r7 are reserved. r0 is always 0, r7 is used for storing the link address;
- a data memory that stores 8 data words;
- instructions that are 13 bits wide, stored in an instruction memory with address width of 8 bits.

Implement the instructions described in the table below ("imm" stands for "immediate", "uns" stands for "unsigned" and "sig" stands for "signed").

Carefully read the following instruction table, as there are a number of differences with the previous assignment. Make sure you use `TestRetake.py` for this project.

12	11	10	9	8	7	6	5	4	3	2	1	0	Description
`0000`				rs			rt			imm (uns.)			lw: $rt = MEM[$rs+imm]
`0001`				rs			rt			imm (uns.)			sw: MEM[$rs+imm] = $rt
`0010`				rs			rt			imm (sig.[2])			bne[5]: $rs!=$rt ? pc := pc+1+imm
`0011`				rs			rt			imm (sig.[2])			be[5]: $rs==$rt ? pc := pc+1+imm
`0100`				rs			imm (sig.[2])						bgtz[5]: $rs>0 ? pc := pc+1+imm
`0101`				rs			imm (sig.[2])						jr: pc := $rs+imm
`0110`				target address								`0`	j: pc := addr"
`0110`				target address								`1`	jal: $r7:= pc+1 ; pc := addr"
`0111`				rs			imm (uns.)				`00`		ori[4]: $rs = $rs|imm
`0111`				rs			imm (uns.)				`01`		lui[3,4]: $rs = imm<<4
`0111`				rs			imm (uns.)				`10`		addi: $rs = $rs+imm
`0111`				rs			imm (uns.)				`11`		subi: $rs = $rs-imm
`1000`				rd			rs			rt			zero[1]: $rd = 0
`1001`				rd			rs			rt			or[1]: $rd = $rs|rt
`1010`				rd			rs			rt			and[1]: $rd = $rs&rt
`1011`				rd			rs			rt			add[1,2]: $rd = $rs+$rt
`1100`				rd			rs			rt			sub[1,2]: $rd = $rs-$rt
`1101`				rd			rs			rt			lt[1,2,5]: $rd = $rs<$rt
`1110`				rd			rs			rt			gt[1,2,5]: $rd = $rs>$rt
`1111`				rd			rs			rt			eq[1,5]: $rd = $rs==$rt

[1] R-type instructions from your ALU.

[2] Signed, two's complement.

[3] "Load upper immediate": put the 4-bit immediate in the upper 4 bits.

[4] The `lui` and `ori` instructions can be used together to implement a `li` pseudo-instruction which loads a 8-bit immediate into a register.

[5] Boolean operation. True yields value 0, false yields value 1.