# Computer Systems and -architecture

Project Exam Retake

*1 Ba INF 2017-2018*

Brent van Bladel
brent.vanbladel@uantwerpen.be

*Don't hesitate to contact the teaching assistant of this course. You can reach him in room M.G.305 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 one tgz archive, as explained on the course's website, and submit your report to the exercises on Blackboard.

- Report deadline: **31 August 2017**
- Evaluation and feedback: **7 September 2017**

## 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;
- 16 registers. Register r0 and r15 are reserved. r0 is always 0, r15 is used for storing the link address;
- a data memory with address width of 8 bits.;
- instructions that are 16 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. As always, if you have

questions about the script, cannot get it to work or suspect that there is a bug, contact the teaching assistant.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | name | instruction | description |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | | | | rd | | | | unused | | | | unused | | | | zero[1] | `zero rd` | $rd := 0 |
| 0001 | | | | rd | | | | rs | | | | rt | | | | and[1] | `and rd rs rt` | $rd := $rs & $rt |
| 0010 | | | | rd | | | | rs | | | | rt | | | | or[1] | `or rd rs rt` | $rd := $rs \| $rt |
| 0011 | | | | rd | | | | rs | | | | unused | | | | not[1] | `not rd rs` | $rd := !$rs |
| 0100 | | | | rd | | | | rs | | | | unused | | | | inv[1] | `inv rd rs` | $rd := -$rs |
| 0101 | | | | rd | | | | rs | | | | rt | | | | add[1] | `add rd rs rt` | $rd := $rs + $rt |
| 0110 | | | | rd | | | | rs | | | | rt | | | | sub[1] | `sub rd rs rt` | $rd := $rs - $rt |
| 0111 | | | | rd | | | | rs | | | | unused | | | | sla[1] | `sla rd rs` | $rd := $rs * 2 |
| 1000 | | | | rd | | | | rs | | | | unused | | | | sra[1,2] | `sra rd rs` | $rd := $rs / 2 |
| 1001 | | | | rd | | | | rs | | | | rt | | | | lt[1] | `lt rd rs rt` | $rd := $rs < $rt ? 1 : 0 |
| 1010 | | | | rd | | | | rs | | | | rt | | | | gt[1] | `gt rd rs rt` | $rd := $rs > $rt ? 1 : 0 |
| 1011 | | | | rd | | | | rs | | | | rt | | | | eq[1] | `eq rd rs rt` | $rd := $rs = $rt ? 1 : 0 |
| 1100 | | | | rd | | | | rs | | | | imm (uns.) | | | | lw | `lw rd rs imm` | $rd := MEM[$rs+imm] |
| 1101 | | | | rd | | | | rs | | | | imm (uns.) | | | | sw | `sw rd rs imm` | MEM[$rs+imm] := $rd |
| 1110 | | | | rd | | | | rs | | | | imm (sign.[2]) | | | | bne | `bne rd rs imm` | $rd != $rs ? $pc := $pc + 1 + imm |
| 1111 | | | | rd | | | | imm (uns.) | | | | 0000 | | | | ori | `ori rd imm` | $rd := $rd\|imm |
| 1111 | | | | rd | | | | imm (uns.) | | | | 0001 | | | | lui[3] | `lui rd imm` | $rd := imm << 4 |
| 1111 | | | | rd | | | | imm (uns.) | | | | 0100 | | | | addi | `addi rd imm` | $rd := $rd - imm |
| 1111 | | | | rd | | | | imm (uns.) | | | | 0101 | | | | subi | `subi rd imm` | $rd := $rd + imm |
| 1111 | | | | rd | | | | immediate (signed[2]) | | | | | | 10 | | jr | `jr rd imm` | $pc := $rd + imm |
| 1111 | | | | target address | | | | | | | | | | 11 | | jal | `jal imm` | $r15 := $pc + 1; $pc := addr |

[1] R-type instruction.

[2] Integer division.

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