Computer Systems and -architecture

Project 7

1 Ba INF 2011-2012

Ruben Van den Bossche
ruben.vandenbossche@ua.ac.be

Sam Verboven
sam.verboven@ua.ac.be

Don't hesitate to contact the teaching assistants of this course. You can reach them in room M.G.2.12 or by e-mail.

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 500 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, 18 2011, 23u55**
- Evaluation and feedback: **December, 20 2011**

Project

1. Exceptions are a very important part of a datapath and control. In this exercise, you will add a basic form of exception handling to your datapath: when an exception is detected, your program counter should halt at the instruction that caused the exception. Both arithmetic overflow and undefined instructions should be detected and supported.

   Think about enhanced versions of exception control. What is necessary in order to add a more advanced form of exception handling to a datapath with our instruction set?

2. Demonstrate the proper operation of your datapath by providing a number of small RASM-programs. Try to use subroutines at least once. Don't forget to initialize the stack pointer. Provide the programs below.

   (a) A program that calculates the Fibonacci numbers and stores them in memory. After which number does overflow occur?

   (b) A program that calculates the greatest common divisor (using the Euclidean algorithm, but without recursion) of two integers read from memory. Store the result back in memory.

(c) A program that finds the biggest element in an array of integers stored in memory. Store the biggest element back in memory.

(d) A program that writes a 1 to memory if an array of integers in memory contains duplicates, and writes 0 if it doesn't.

(e) A program that sorts an array of integers in memory. You can use a sort algorithm of your own choice. If you want a challenge, you can try to implement the quick sort algorithm.