School of Computer Science and Information Technology
Computer Systems Architecture (G51CSA) Autumn 2008
Thorsten Altenkirch

Coursework 8 (cw id 124)
Monday, 1 December 2008
Deadline: 8 December 2008, 12:00

Collaborating in small groups of up to three students is permitted, but you must implement your own programs (absolutely *do not* copy and paste from others) and provide your own answers where appropriate.

The solution has to be submitted using the departmental coursework submission system, see
http://support.cs.nott.ac.uk/coursework/cwstud/.

Create a directory `ex08` and put all the files to be submitted (but nothing else) into this directory before submitting the directory.

Multiple submission before the deadline are allowed, only the last one will be taken into account.

1. Implement a simple reaction game using timer and keyboard interrupts in MIPS assembly language using SPIM's simulated interrupts.

   The game prints a sequence of dots (`.`) with a delay of 0.5s. After a number of dots it prints a star (`*`). The user is supposed to press a button as soon as possible after the star appears. If she manages to do this within the next 0.5s the program prints the number of 10 ms it has taken to user to hit the key. Otherwise it prints a message *time limit exceeded*. If the user presses a key before the star has appeared, the program prints the message *too early* and exits.

   Hints:

   - Make sure you enable memory-mapped IO in SPIM, e.g. for xspim start with `xspim -mio`.
   - Use the skeleton base08.asm as a starting point. It is based on the code related to lecture 12.
   - You can fix the number of dots, but you are also welcome to find a way to randomize this.

   Store your program in a file called `react.asm`

2. Answer the following questions regarding pipelines and store your answers in a file `pipeline.txt`.

   (a) What is a data hazard? Name and describe a hardware technique that overcomes most data hazards. What happens during the execution of the following sequence?

   ```
   lw $t0, 0($a0)
   add $s0, $s0, $t0
   ```

   (b) What is a control hazard? How can we avoid stalling the pipeline some of the time? What happens if this technique goes wrong?

   (c) Identify the data hazards in the following code fragment and reorder the instructions to avoid any pipeline stalls.

   ```
   lw $t0, ($sp)
   lw $t1, ($t0)
   addi $t1, $t1, 4
   lw $t2, ($t0)
   add $t3, $t2, $t1
   sw $t3, 4($sp)
   sw $zero, ($sp)
   ```