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

Coursework 5 (cw id 121)
Monday, 27 October 2008
Deadline: 3 November 08, 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.

Part 1 has to be submitted on paper to the school office before the deadline. Use the coursework submission cover sheet and write CSA, coursework 5, *your name*, *your student id*, and the email address of your CSA tutor, i.e. either *asg* or *lyh* on the top of your submission. If your submission is more than one page then the pages have to be stapled together.

The remaining parts have to be submitted using the departmental coursework submission system, see

http://support.cs.nott.ac.uk/coursework/cwstud/.

Create a directory `ex05` 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. The following exercise is about binary arithmetic. It is important that you carry out these calculations by hand on paper. Hence, the answers to this question have to be **submitted to the school office on paper!**

   Given

   $$\begin{aligned} a &= 122 \\ b &= 10 \\ c &= 63 \end{aligned}$$

   Translate a, b, c into unsigned 8-bit numbers and calculate in binary:

   (a) $a \times b$

   (b) $b \times c$

   (c) $a \div b$

   (d) $c \div b$

   The result should be either one 16 bit number ($\times$) or two 8 bit numbers ($\div$) representing quotient and remainder.

   Show all the steps of your calculation. Write clearly and use vertical pencil lines to layout your calculations.

   Check your results by translating back into decimal.

2. Implement a MIPS program that reads two integers, multiplies them using mult and prints the result, if it is representable with as a signed 32 bit number. Call you program `mult.asm`. The program should check for overflow – emulating multo – and print the result or print *overflow*, if an overflow has occured.

   Test your programs with the following inputs: -10×-20, 2000000×1000000 and -2000000×1000000.

3. The following C fragment implements unsigned binary division of two 16 bit numbers x, y calculating the quotient in z and the remainder in x using only logical operations and addition and subtraction:

```
z=0;
y = y << 16;
for(i=0;i<16;i=i++) {
  y = y >> 1;
  z = z << 1;
  if(x >= y) {
    x = x - y;
    z = z + 1;
  }
}
```

   Translate this program into MIPS assembler, i.e. implement division without using the built-in division operation. Call your program `div.asm`.

   Test your program with $122 \div 10$ and $4321 \div 123$.

4. (*) Implement exponentiation of binary numbers using only multiplication. Implement your program first in C (calling it `pow.c`) and then translate it into MIPS assembler (calling it `pow.asm`). Your program should be efficient in the sense that its running time should not depend on the input numbers (but only on their size).

   You don't have to check for overflows and you may assume that the inputs are representable with 16 bits.

   Test your program with 10^3 and 7^6.