Lab session Using the Datapath

Group A: November 23, 2009
Group B: November 24, 2009

Work in the given groups of two. Submit your solutions to the respective assignment on Blackboard. The file name is:
`a07_s0XXXXX_s0XXXXX.tar.gz`
One of the group members commits your solution. Keep an eye on the deadline (see Blackboard)!

1 Project

Finally the datapath will be put to work. An assembler `asm.pl`, written in the scripting language Perl, can be found on the website. It takes assembler code as input, and produces hexadecimal machine code. A file containing this code can be directly loaded into your Logisim RAM memory: right click the RAM element, select "Edit contents", "File", "Open". An input file `input.s` can be assembled in Linux as follows:

```
perl asm.pl < input.s > output.hex
```

`output.hex` then contains the hexadecimal machine code which can be loaded into Logisim RAM memory. Your input file is of the form:

```
lui $r0, 85
ori $r0, $r0, 68
lui $r1, 85
ori $r1, $r1, 68
beq $r0, $r1, -1
```

Note that labels are not supported, so you have to hard-code the branches in your assembler code. The script file `asm.pl` will produce something like this:

```
2055
3044
2155
3544
91ff
```

These machine instructions can be loaded into the program counter memory.

This week's project consists of two parts:

1. The script file `asm.pl` uses regular expressions to recognize the lines of assembler code. However, your have to adapt `asm.pl` such that the operation codes and funct codes that are generated match the ones of your datapath. Also, not all operations are present in the script file, so you have to add (i.e. copy and adapt) some lines of code in the script file.

2. Write an assembler file for your datapath that implements multiplication. You can make the following assumptions: before the execution of the program is started, the input `x` is present in memory at a designated address A (which you can choose yourself), and input `y` is present at address A+1word. Then `x*y` is calculated and the result is put in memory at address A+2words. So for example, memory has the following values:

address	value
A	3
A+1	4

   then after execution of the multiplication program, memory contains the following:

address	value
A	3
A+1	4
A+2	12

   Mind multiplication of negative numbers!