**Workshop 02**

**Workshop - More Combinatorial Hardware**

**Make sure you are marked off during this session. Immediately before marking flag yourself in the pracmarker:**

[**https://cs.adelaide.edu.au/services/pracmarker/**](https://cs.adelaide.edu.au/services/pracmarker/)

This workshop is worth 0.8% of your course mark.  These marks will be given for your participation in your session. As with the first session the purpose of this session is to give you the chance to work with the design of hardware. In this case building simple components from basic gates that you saw in chapter 1.

In this workshop you will be producing HDL code.  For the HDL you may find t[he HDL Survival Guide](http://www.nand2tetris.org/software/HDL%20Survival%20Guide.html) useful.

The questions below are derived from the second assignment of the Nand2teris course. See [this link](http://nand2tetris.org/02.php) for more information and test scripts.

Answer the following questions.

**Question 1  (elementary)**

Write HDL code for a **half-adder** gate. (make sure you include comments describing your design).

**In your log for this workshop**write some brief notes your development process for this half-adder gate.

**Question 2 (intermediate)**

Write HDL code for a **full-adder** gate (again, include comments about design)

**Write brief notes**  logging your development process for this full-adder gate.

**Question 3 (more challenging)**

Look at [table 2.6 in the textbook](http://www.nand2tetris.org/chapters/chapter%2002.pdf). This table describes the behaviour of an ALU in terms of input switches. Answer the following:

1. Consider only the rows for the operations x&y and x+y. **Draw** the gates for an ALU that implements these two operations. Note that this means the only control input needed is **f**. You must assume that you have two 16-bit inputs **x** and **y.**In your answer you may assume that you can use gates for a 16 bit adder and 16-bit versions of any gate specified in your[first assignment](http://nand2tetris.org/01.php).  This means you should be able to write your answer using just a small number of gates.
2. Copy your diagram from part 1 above and add an input for the last control value **no**. Connect this wire to logic that implements the behaviour specified at the top of the **no** column. What operations would be implemented if **no=1**. (hint: they are not any of the operations currently in the table).

**Include notes** logging your development process for this full-adder gate.

At the end of your session go to the [pracmarker system](https://cs.adelaide.edu.au/services/pracmarker/?sub_year=2016&sub_period=s1&sub_course=intrope&sub_assign=workshop1&sub_alt_user=a1065958&sub_flagme) (navigate to workshop 2 of this course)  and, when the supervisor comes to check your work  flag your work for marking.