Homework 6: Computer Architecture

Objective:

Build all gates described in Chapter 5 (see table below), which will test your understanding of our HACK computer architecture.

Grading method:

If the chip passes *all* the tests specified in the supplied test script, it receives 60% of the grade. 30% goes to it being well built (the lowest number of chips to implement), with the remaining 10% going towards documentation provided for each chip. Generally speaking, we prefer implementations that *use as few chip **parts** as possible,* even if it implies a less efficient chip design (in term of # of AND/OR/NOT chips). Higher-level chips are considered as one chip part (ex. Mux, DMux, Or8Way, etc.)

What do you turn in?

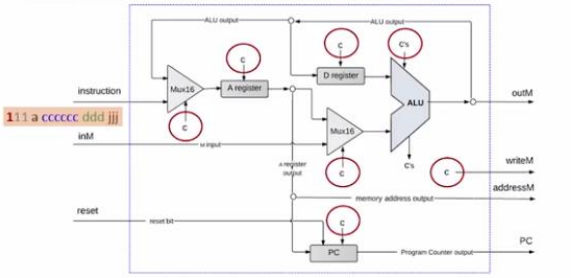
A Word document (or PDF) with screen shots of each of the working (or not) logic gates. You should also upload the documentation.pdf file (see Documentation Instructions for guidelines on how to do this) per Project Submission Guidelines.

Reminder: Make sure you use the ROM32K, Keyboard and Screen chips!

Chip	***Working?***	***Well built?***	
Memory	/ 15	/ 15	5 of these well-built points are for handling KBD addressing
CPU	/ 30	/ 10	
Computer	/ 15	/ 5	
Subtotal	/ 60	/ 25	
Documentation	/	10	

See http://nand2tetris.org/05.php for some tips/resources/tools (note that the assignment on the website may be substantially different from the assignment that is described above, if you need clarification email your instructor. You will be graded based on this documents requirements).

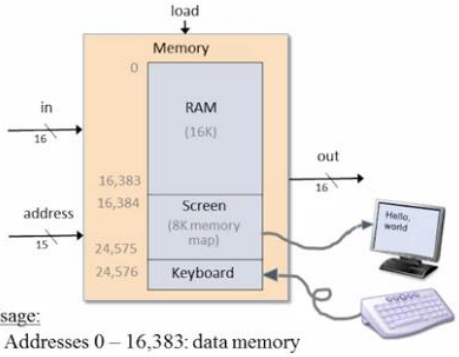
CPU Implementation



Implementation tips:

- Chip-parts: `Mux16, ARegister, DRegister, PC, ALU, ...`
- Control: use HDL subscripting to route instruction bits to the control bits of the relevant chip-parts.

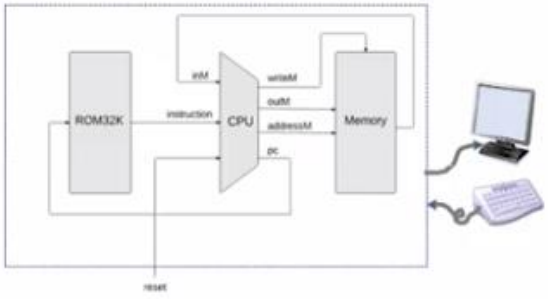
Memory implementation



Usage:

- Addresses 0 – 16,383: data memory
- Addresses 16,384 – 24,575: screen memory map
- Address 24,576: keyboard memory map

Hack Computer implementation



Computer.hdl

```
/**
 * The HACK computer, including CPU, ROM and RAM.
 * When reset is 0, the program stored in the computer's ROM executes.
 * When reset is 1, the execution of the program restarts.
*/

CHIP Computer {

    IN reset;

    PARTS:
    // Put your code here.
}
```

See http://nand2tetris.org/05.php for some tips/resources/tools (note that the assignment on the website may be substantially different from the assignment that is described above, if you need clarification email your instructor. You will be graded based on this documents requirements).