Project 5: Computer

In this project you will integrate the ALU, registers and RAM devices built in projects 2 and 3 into a CPU and a computer system capable of executing programs written in the machine language introduced in project 4.

Objective

Build the following three chips:

Memory

CPU

Computer

Files: For each one of the three Xxx chips, we provide a skeletal Xxx.hdl program, also called *stub file*, with a missing PARTS section. In addition, for each chip we provide one or more Xxx.tst scripts that tell the hardware simulator how to test the chip, along with an Xxx.cmp compare files containing the correct outputs that the supplied test is expected to generate. Your task is writing, and testing, the chip implementations (specifically: Completing the supplied Xxx.hdl files).

Contract: For each chip in the list, your chip implementation (modified Xxx.hdl file), tested by the supplied Xxx.tst files, must generate the outputs listed in the supplied Xxx.cmp files. If the actual outputs generated by your chip disagree with the desired outputs, the simulator will report error messages. For a final test, your Computer chip will have to correctly execute the three test programs described below.

Test Programs

A natural way to test your topmost Computer-on-a-chip implementation is to have it execute correctly sample programs written in the Hack machine language. In order to run such a test, one can write a test script that loads a machine language program into the Computer's ROM32K chip-part (the instruction memory), and then runs the clock enough cycles to execute the program. We provide three such test programs, along with corresponding test scripts and compare files:

Add: Adds the constants 2 and 3 and writes the result in RAM[0].

Max: Computes the maximum of RAM[0] and RAM[1] and writes the result in RAM[2].

Rect: Draws on the screen a rectangle of RAM[0] rows of 16 pixels each. The rectangle's top left corner is located at the top left corner of the screen.

Before testing your Computer chip on any one of these programs, review the test script associated with each program, and be sure to understand the instructions given to the hardware simulator. If needed, consult the Test Description Language Guide (link below).

If you wish to inspect the binary code of any one of the three test programs (e.g. Add.hack), use the CPU emulator to open the symbolic version of this program (e.g. Add.asm) from the projects/6 folder, and change the display mode from 'asm' to 'bin'.

Implementation Plan

Build the computer platform in the following order:

Memory: The three main chip-parts of this chip are RAM16K, Screen, and Keyboard. They are given as builtin chips, and there is no need to implement them. The main implementation task is combining these three memory chip-parts into a single 32K address space.

CPU: The Central Processing Unit can be built from the ALU built in project 2, the Register and PC chips built in project 3, and logic gates built in project 1. As usual though, we'll use their builtin versions. In particular, your HDL code should use the ARegister and DRegister chips. These builtin chips have exactly the same functionality as the Register and PC chips built in project 3.

Instruction Memory: Use the builtin ROM32K chip. This chip is essentially the same as the RAM chips built in project 3, but the builtin version has an interface that allows loading a program from a text file into the memory.

Computer: Can be built from three chip-parts: CPU, Memory, and ROM32K.

Building the chips

A new online IDE (Integrated Development Environment) was recently launched for learners of Nand to Tetris courses. Therefore, there are now two options for completing project 5:

If you are using the Nand2Tetris IDE Online (which is recommended), all the Xxx.hdl, Xxx.tst and Xxx.cmp files are available in your browser memory. To develop and test a particular chip, select the project / chip from the simulator's drop-down menus. Your edited HDL code will be saved automatically. To download the HDL files to your local PC, click the *download* button. The current version of all the project's Xxx.hdl files will be downloaded as one zip file.

Using the desktop Nand2Tetris hardware simulator is also possible. If you've downloaded the Nand to Tetris software suite from www.nand2tetris.org, and extracted it into a folder named nand2tetris on your computer, the nand2tetris/tools folder contains the desktop version of the hardware simulator, and the nand2tetris/projects/5 folder contains all the files needed for completing this project. You can write/edit the HDL code of each Xxx.hdl file using any plain text editor, and then test your code using the desktop simulator.

References

Hardware Description Language

Test Description Language (to be used as needed, for understanding the supplied test scripts)

Hack Chip Set API

Tutorials

The tutorials below focus on using the desktop version of the hardware simulator. Tutorials for the online simulator, the preferred tool for this project, will be available soon. However, you can apply the principles from these tutorials to perform similar actions in the online simulator (a major difference is that there is no need to load any files in the online simulator).

Screen chip demo

Keyboard chip demo

ROM32K chip demo

Implementation Tips

0. Before implementing a chip, it is recommended to experiment with its builtin implementation. If you are using the online simulator, simply click the *builtin* toggle; If you are using the desktop version, load the builtin chip from the tools/builtInChips folder.
1. Each chip can be implemented in more than one way. The simpler the implementation, the better. As a general rule, strive to use as few chip-parts as possible.
2. In the course of implementing the CPU or other chips in this project, you may be tempted to specify and build some internal ("helper") chips of your own. Be advised that there is no need to do so; The Hack CPU can be implemented elegantly and efficiently using only the chip-parts mentioned above, plus some of the elementary logic gates from project 1.
3. Each chip can be tested interactively, or using a test script. In the online simulator, simply run the test script. In the desktop simulator, you must first load the test script, and then run it.
4. The three test programs with which you have to test your Computer chip are Add.hack, Max.hack, and Rect.hack. These programs are written in the binary Hack machine language. In the online simulator, you can inspect the symbolic versions of these programs in the visualized ROM image; In the desktop simulator, you can inspect the respective .asm files in the projects/6 folder.
5. Normally, when running a program on some computer, and not getting expected results, we conclude that the program is buggy. In this project, the supplied test programs are bug-free. Therefore, if running a test program yields unexpected results, it means that the computer chip implementation (Computer.hdl) is buggy. If that is the case, debug the chip and keep testing.
6. How to make sure that you are using builtin chip-parts? In the online hardware simulator, this is the default, without further ado; In the desktop simulator, use only the .hdl files stored in the projects/5 folder. Don’t add any other .hdl file to this folder.