CSCE-312 | Spring 2019

Project 5

Computer Architecture

Due Date: Submit on eCampus by **Thursday, March 28th, 11:59 PM**

Grading:

(A) Project Demo [60%]: TBA

You will be graded for correctness of the chips (hdl) you have designed and coded. We will be running offline test of all your HDL codes using Nand2tetris software (Hardware Simulator). The same simulator you will be using to check your chips in the course of this project. So, make sure to test and verify your codes before finally submitting on eCampus.

(B) Lab Quiz [40%]: TBA

***Rubric**:* The rubric allocation for chips is shown at the end of this document. Each chip needs to pass all its test cases to get the points, else you will receive 0 on that chip.

Deliverables & Submission: Turn in a zip file in the format *FirstName-LastName-UIN.zip* containing the 3 HDL files(Memory.hdl, CPU.hdl and Computer.hdl). Put your full name in the introductory comments present in each HDL file. Use relevant code comments and indentation in your code. Submit this zip file on eCampus.

Late Submission Policy: Refer to the Syllabus

Background

In previous projects we've built the computer's basic *processing* and *storage* devices (*ALU* and *RAM*, respectively). In this project we will put everything together, yielding the complete *Hack Hardware Platform*. The result will be a general-purpose computer that can run any program that you fancy.

Objective

Complete the construction of the Hack CPU and computer platform, leading up to the top-most Computer chip.

Chips

Chip (HDL)	Description	Testing
Memory.hdl	Entire RAM address space	Test this chip using Memory.tst and Memory.cmp
CPU.hdl	The Hack CPU	Recommended test files: CPU.tst and CPU.cmp. Alternative test files (less thorough but do not require using the built-in DRegister): CPU-external.tst and CPU-external.cmp.
Computer.hdl	The platform's top-most chip	Test by running some Hack programs on the constructed chip. See more instructions below.

Contract

The computer platform that you build should be capable of executing programs written in the *Hack machine language*, specified in Chapter 4. Demonstrate this capability by having your Computer chip run the three test programs given below.

Testing

Testing the Memory and CPU chips: It's important to unit-test these chips before proceeding to build the overall Computer chip. Use the the test scripts and compare files listed above.

Testing the Computer chip: A natural way to test the overall Computer chip implementation is to have it execute some sample programs written in the Hack machine language. In order to perform such a test, one can write a test script that (i) loads the Computer.hdl chip description into the supplied *Hardware Simulator*, (ii) loads a machine-level program from an external .hack file into the ROM chip-part of the loaded Computer.hdl chip, and then (iii) runs the clock enough cycles to execute the loaded instructions. We supply all the files necessary to run three such tests, as follows:

Program	Comments
Add.hack	Adds up the two constants 2 and 3 and writes the result in RAM[0]. Recommended test:ComputerAdd.tst and ComputerAdd.cmp. Alternative test (less thorough but only requires usage of the built-in RAM16K): ComputerAdd-external.tst and ComputerAdd-external.cmp.
Max.hack	Computes the maximum of RAM[0] and RAM[1] and writes the result in RAM[2]. Recommended test:ComputerMax.tst and ComputerMax.cmp. Alternative test (less thorough but only requires usage of the built-in RAM16K): ComputerMax-external.tst and ComputerMax-external.cmp.
Rect.hack	Draws a rectangle of width 16 pixels and length RAM[0] at the top left of the screen. Recommended test ComputerRect.tst and ComputerRect.cmp. Alternative test (less thorough but does not require usage of any built-in chips): ComputerRect-external.tst and ComputerRect-external.cmp.

Before testing your Computer chip on any one of the above programs, read the relevant .tst file and be sure to understand the instructions given to the simulator. Appendix B of the book may be a useful reference here.

Resources

The relevant reading for this project are Chapter 5, Appendix A, and Appendix B (as a reference, and use TAMU mail id to access). Specifically, all the chips described in Chapter 5 should be implemented in the *Hardware Description Language* (HDL) specified in Appendix A.
The resources that you need for this project are the supplied *Hardware Simulator* and the files listed above.

Implementation Tips

Complete the computer's construction in the following order:

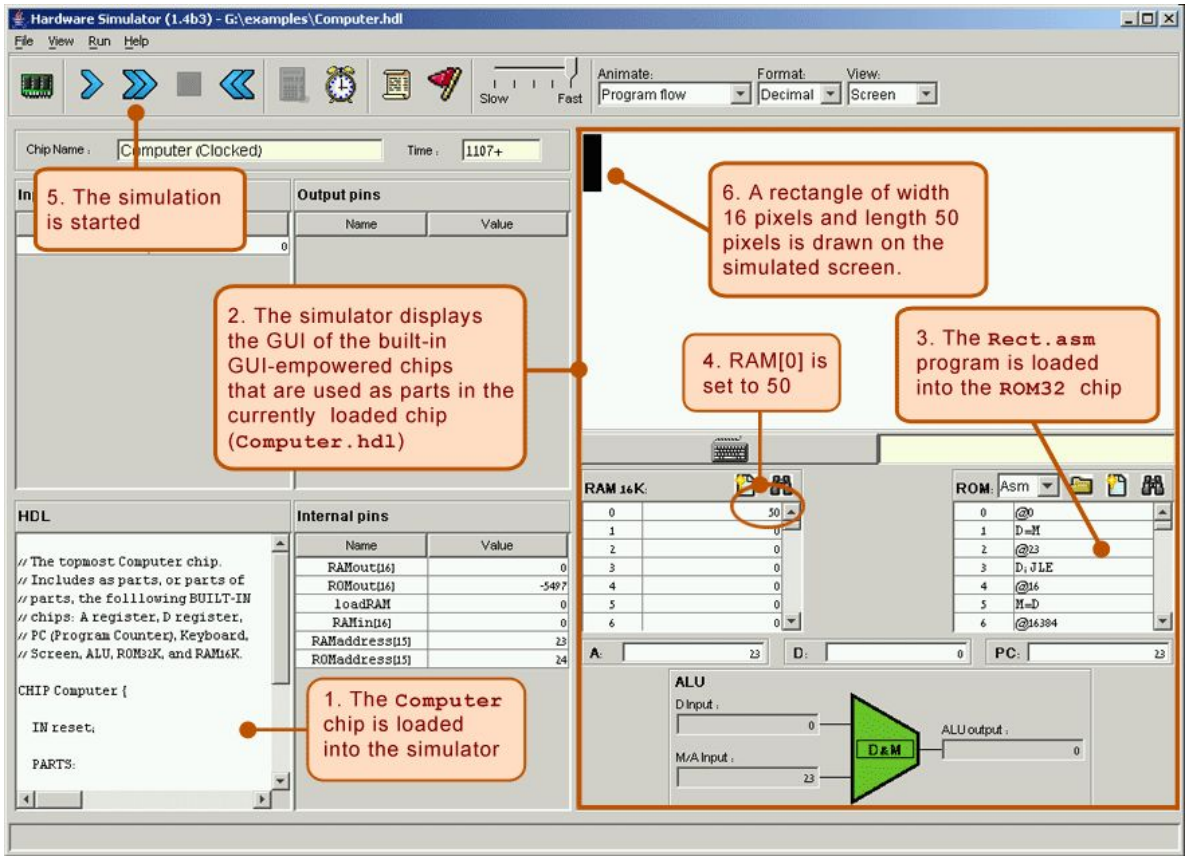
Memory: This chip includes three chip-parts: RAM16K, Screen, and Keyboard. The Screen and the Keyboard are available as built-in chips, and thus there is no need to implement them. Although the RAM16K chip was built in Project 3, we recommend using its built-in version, as it provides a debugging-friendly GUI.

CPU: This chip can be constructed according to the proposed CPU implementation given in Figure 5.9 of Chapter 5, using the ALU and register chips built in Projects 2 and 3, respectively. We recommend though using built-in chip-parts instead, in particular **A Register** and **D Register**. The built-in versions of these two chips have exactly the same interface and functionality as those of the Register chip specified in Chapter 3; however, they feature GUI side-effects that come handy for testing purposes.
In principle, your CPU implementation may include internal chips of your own specification, i.e. chips not mentioned in Figure 5.9 of Chapter 5. However, this is not recommended, and will most likely yield a less efficient CPU design. If you choose to create new chips not mentioned in the book, be sure to document and unit-test them carefully before you plug them into the architecture.

Instruction memory: Use the built-in ROM32K chip.
Computer: The top-most Computer chip can be constructed according to the proposed implementation shown in Figure 5.10 of Chapter 5.

Tools

All the chips mentioned in this project, including the topmost Computer chip, can be implemented and tested using the supplied *Hardware Simulator*. Here is a screenshot of testing the Rect.hack program on a Computer chip implementation.



The Rect program illustrated above draws a rectangle of width 16 pixels and length RAM[0] at the top-left of the screen. Now here is an interesting observation: normally, when you run a program on some computer, and you don't get the desired result, you conclude that the program is buggy. In our case though, the supplied Rect program is *bug-free*. Thus, if running this program yields unexpected results, it means that the computer platform on which it runs (Computer.hdl and/or some of its lower-level chip parts) is buggy. If that is the case, you have to debug your chips.

Rubric(60 Points):

Your turned in files will be graded based on the following tests:

CPU.tst: 15

Memory.tst: 10

CPU-external: 5

ComputerAdd: 5

ComputerAdd-external: 5

ComputerMax: 5

ComputerMax-external: 5

ComputerRect: 5

CmputerRect-external: 5

Make sure you did the tests on each of them