University of Connecticut
CSE 4302 / CSE 5302 / ECE 5402: Computer Architecture
Fall 2024

Programming Assignment 0:
Getting Started with *riscv-uconn* simulator

Due September 3, 2024 (Tuesday) @ 11:59 PM on HuskyCT

The objective of this introductory PA is to guide the student through setting up the development environment that will be used for the remainder of the course. The student will also gain familiarity with obtaining PA materials, using *riscv-uconn*, and submitting course deliverables through HuskyCT.

1 Install Visual Studio Code

Visual Studio Code (VS Code) is a cross-platform Integrated Development Environment (IDE) with powerful built-in developer tooling. VS Code provides first-party support for several popular programming languages (C, C++, Python, etc.) and developer tools (git, gcc, gdb, etc.).

Those interested in trying out VS Code can install it from the following webpage:

`https://code.visualstudio.com/download`

Instructions for using VS Code with C/C++ can be found on the following webpage:

`https://code.visualstudio.com/docs/languages/cpp`

Windows users can learn how to use VS Code with WSL on the following webpage:

`https://code.visualstudio.com/docs/remote/wsl-tutorial`

2 Environment Setup

The PA materials for this course will be provided through a git repository hosted on UConn's GitHub server. They will be written in C code and require an appropriate toolchain (compiler, build system, etc.). Follow the instructions below to setup the required development environment for your platform.

NOTE: For terminal commands, the “$" character denotes the shell prompt and should not be typed literally.

Windows 10/11

Windows users will leverage the Windows Subsystem for Linux (WSL). WSL is essentially a lightweight Linux Virtual Machine (VM) that runs as a native Windows application.

The following instructions describe how to install WSL, Ubuntu Linux, and the required developer tools:

1. Ensure that your machine is completely up to date.
2. Follow the **Manual Installation Steps** on the following webpage to install WSL:

   `https://docs.microsoft.com/en-us/windows/wsl/install`

   and install **Ubuntu 20.04.6 LTS** from the Microsoft Store.
3. After installation, launch the Visual Studio Code IDE.
4. In the options on the top right of the screen, Click on *View* → *Terminal.*

5. Click on the ▼ sign right next to "+". And select Ubuntu-20.04 (WSL).
6. Enter the following commands in the Terminal to install GNU Make, Git, the GNU Compiler Collection (GCC), and the GNU Project Debugger (GDB):

   ```
   $ sudo apt update
   ```

   ```
   $ sudo apt install make git gcc gdb
   ```

Mac OS

Mac OS users will install the Xcode command line tools, which includes GNU Make, Git, the Clang C/C++ compiler, and the Clang debugger. Instructions to do so are as follows:

1. launch the Visual Studio Code IDE.
2. In the options on the top right of the screen, Click on *View* → *Terminal*.
3. Install the Xcode command line tools with the following command on the Terminal:

   ```
   $ xcode-select -install
   ```

   Follow the Finder dialog prompts to finish installing the tools.

3 Getting *riscv-uconn*

All PA materials will be available at the following GitHub repository:

`https://github.uconn.edu/omk12001/cse4302`

You can clone this repository by executing the following command in your platform's terminal:

```
$ git clone https://github.uconn.edu/omk12001/cse4302.git
```

The above command will create a new copy of the directory `cse4302` in the current working directory (the directory the command was executed in).

When a new assignment is released, you can pull the contents by executing the following command anywhere within the `cse4302` directory:

```
$ git pull
```

You will never `push` anything to this repository.

4 Getting Started with *riscv-uconn*

Once the repository is cloned, you will find two subdirectories underneath `cse4302`: `assembler` and `pa0`. In this programming assignment, you will explore a 5-stage pipelined simulator implementing the ADDI instruction in the RISC-V *riscv-uconn* Instruction Set Architecture (ISA). Refer to the Introduction to *riscv-uconn* document for more details.

Navigate to the `assembler` directory and execute `make` to build the assembler binary. Periodically, the assembler may be updated with new functionality, so it is important to ensure it is up-to-date between assignments.

The following is a brief description of the relevant materials in `pa0`:

`src/`	Simulator source code
`unittests/`	Simulator unit tests (test programs)
`README.md`	Simulator and unit test build instructions
`assemble_all.sh`	Bash script to automatically assemble the tests in `unittests/`

4.1 Assembling a Unit Test

For `pa0`, a unit test for the ADDI instruction is assembled, `IType_ADD.asm`. After the assembler is built, navigate back to the `pa0` directory and assemble the `IType_ADD.asm` file into binary machine code using the following command:

```
$ ./assemble_all.sh
```

Running this script will create a directory `assembled_tests`, and populate it with the assembled unit test. Verify that the contents of `IType_ADD.out` file match the following:

```
00000000000100101000001010010011
00000000000100000000000000010011
11111111111111111111111111111111
00000000000000000001000011001110
00000000000000000001010100011010
00000000000000000000011111100111
```

You can also examine the `IType_ADD.asm` file to see how it is structured.

4.2 Simulating a Unit Test

The `src` directory contains a basic 5-stage pipeline simulator that only supports the ADDI instruction. Build the simulator by executing the `make` command in the `pa0` directory. Next, execute the simulator using the following command:

```
$ ./simulator ./assembled_tests/IType_ADD.out > PA0.output
```

The simulator output will be piped to PA0.output file. Verify that the output includes the following lines:

```
TOTAL INSTRUCTIONS COMMITED: 2
TOTAL CYCLES SIMULATED: 6
AVERAGE CPI: 3.000
```

Also verify that Register Dump section shows t0 register value as 1. If all verification steps pass, you are now ready to proceed with future PAs.

Please submit the PA0.output file in HuskyCT to complete this assignment.