CSE 520 Computer Architecture II
Lab 0: Getting Familiar with Pin

Assigned: January 15, 2026

Due on: Thursday January 28th by 11:59pm

1 Summary

This term in CSE 520 we will be using Pin, a binary instrumentation tool provided by Intel. The purpose of this lab is to familiarize you with compiling and running Pin tools, as well as to make sure that you are able to use the class infrastructure and computers. To test the infrastructure, you will build and run a small Pin tool. In order to develop architectures that run programs quickly architects must thoroughly understand the properties of representative programs. Architects experiment with different programs to help them make design decisions about features that will be included in processors. There are a few approaches that can be used to evaluate the behavior a program. One is simulation. In this case, a software model of a processor is built, and the program executed on the model. Simulators have the advantage of being arbitrarily detail – in theory one could build a SPICE processor simulator. Typically, architectural simulators give cycle-accurate timing estimates. The penalty for this level of detail is simulation speed: the more detailed a software simulator is, the slower it can execute programs; advanced software simulators can simulate at rates of tens of KIPS (kilo instructions per second). A second option is code instrumentation. Code instrumentation collects information about program characteristics. In general, code instrumentation is less detailed than simulation, but code instrumentation is faster to implement and enjoys faster program execution. Thus code instrumentation which can be very useful in guiding architectural decisions early in the development process, before detailed simulators are available. The simplest form of code instrumentation, terminal display statements (e.g. printf), is used by almost every programmer, computer architect or otherwise. Clearly such manual instrumentation is time-consuming, both in terms of writing code and collecting execution results. A better choice is to use a meta language to describe the code instrumentation and develop a tool that will efficiently instrument the target program at compile or runtime.

Pin is a free (though not open), industrial grade binary instrumentation tool produced by Intel and used widely in industry and academia. Pin accepts as inputs a compiled Pin tool and a generic binary executable. A Pin tool is a C++ program that makes a series of calls to an instrumentation API and provides code for Pin to execute at instrumented locations. The executable

itself is just-in-time compiled by Pin, and the instruments are inserted. The code is then executed natively on the host machine. A major advantage of Pin is that it can instrument programs without requiring recompilation. Thus, even legacy binaries can be analyzed. Since Pin executes large portions of the target program natively, it can be very fast, however, this constrains the programs analyzed to the host architecture, namely x86. Yet, Pin is surprisingly versatile: new instructions can be emulated by hijacking unused x86 opcodes.

1.1 Preliminaries:

Although you can develop Pin tools on your home machine, we suggest using either the provided VM or the Sol compute server. If you are running an Intel or AMD CPU and would like to use the VM, please follow the provided "VM Setup" instructions. If you are running a device with Apple Silicon, you will need to utilize Sol. Please follow the "Sol Setup" instructions below.

1.2 Sol Setup:

If you are using your personal machine (which utilizes an Intel or AMD CPU) or plan to use the provided VM, please skip this section.

To utilize ASU's compute infrastructure, you first need to ensure you are connected to the ASU network. If you are off-campus, you will need to use the ASU VPN.

We are currently waiting for Sol accounts to be setup. In the meantime, please utilize the general ASU compute platform by running

```
ssh <asurite>@general.asu.edu
```

where you replace `<asurite>` with your own asurite ID. You may need to verify your identity using your preferred two-factor authentication method.

Once the Sol accounts for this class have been created, we will send out an announcement on Canvas. If you have completed Lab 0 on `general.asu.edu`, you can transfer the Lab 0 files to Sol without any issues. To log in to Sol via ssh, open a terminal and run

```
ssh <asurite>@login.sol.rc.asu.edu
```

You may need to verify your identity using your preferred two-factor authentication method.

Once you have logged in to Sol, you can then start an 8-hour interactive terminal session with 8 CPU Cores and 16 GB of RAM by running the following within Sol:

```
interactive -c 8 --mem=16gb -t 0-08:00:00
```

1.3 Setting up:

Once you have decided which machine to use, download the Zip file we have provided through Canvas.

If you are using Sol or `general.asu.edu`, you will first need to download the file on your personal machine and upload it to the compute platform. We recommend doing this via scp. On your host machine, for Sol run

```
scp <Path to CSE520.zip> <asurite>@login.sol.rc.asu.edu:~
```

for `general.asu.edu` run

```
scp <Path to CSE520.zip> <asurite>@general.asu.edu:~
```

Once you have downloaded the file, extract it. You can do this by running the following command in the same directory where the file is located:

```
unzip CSE520.zip
```

The command will create the directory structure we will use for this course. Inside the "CSE520/Lab0" directory created, you will find the script to setup the Intel Pin tool and a basic test program. Run the following commands to setup Pin:

```
cd CSE520/Lab0
chmod +x setup.sh run.sh
./setup.sh
```

You will notice that `setup.sh` fails to compile. While your instructor is an expert in Verilog, his C is not so good. Fix his silly parse error, and run `setup.sh` again. Once everything is setup, run

```
./run.sh
```

to verify that Pin has been installed correctly.

Since the `inscount0` Pin tool counts the number of instructions in the instrumented program, the output file will contain the number of instructions executed when running the test program. If you are interested in the inner workings of a real Pin tool, examine `inscount0.cpp`. The included scripts will enable you to test your Pin tool, but will not verify correctness. The expected results of the test cases will not be released. Furthermore, the TAs may use additional test cases to verify the correctness of your solution. You are encouraged to compare your results with your lab mates and craft your own test cases.

1.4 Submitting your lab:

When you have completed the lab to your satisfaction, rename your `inscount0.cpp` according to

```
<ASURITE_ID>_lab0.cpp
```

and submit to Canvas by the submission deadline. No late submissions will be accepted!

If you are using Sol or `general.asu.edu`, you will need to transfer `inscount0.cpp` from the compute server back to your personal machine. If you are using Sol run the following command from your personal device

```
scp <asurite>@login.sol.rc.asu.edu:<path-to-inscount0.cpp> \
    <path-to-save-file-on-personal-machine>
```

If you are using `general.asu.edu`, run the following command from your personal device

```
scp <asurite>@general.asu.edu:<path-to-inscount0.cpp> \
    <path-to-save-file-on-personal-machine>
```

1.5 Guides for the perplexed:

- Pin home page
- Pin documentation