18-344 Recitation 1

Setting up Lab0 toolchains

Office Hours:

Hour starting at	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
9:00 AM							
10:00 AM			M		M		
11:00 AM			M		M	M	
12:00 PM							
1:00 PM						Jordan	
2:00 PM		Mathew	Jeffrey	Mathew	Jeffrey	Jordan	
3:00 PM		Mathew	Jeffrey	Mathew	Jeffrey	Jordan	
4:00 PM							

Step 1: Set up environment variables

Add this to your .bashrc file

```
source /afs/ece.cmu.edu/class/ece344/bin/setup344
```

You may also need to add these to the top of your .bashrc file:

```
aklog ece.cmu.edu
aklog andrew.cmu.edu
```

Step 2: Test the toolchain

To test the toolchain, we will use the files provided in the lab0.tar.gz tarball.

/afs/ece.cmu.edu/class/ece344/assign

Extract these files on in a local directory using `tar -xvzf <file_name>.tar.gz`

Note that the tarball contains only the files, so make sure to extract it in an empty directory.

Navigate to the directory where you've extracted the tarball.

Type `make`.

This should result in a folder named `obj-intel64`, with a file called `mem-count.so`.

We can now test the toolchain.

Step 2a: Testing Pin

Type the following command:

```
pin -t <pintool-path>/<pintool-name>.so -o test.stats -- curl https://google.com
```

here,

`-t` flag specifies the pintool with its relative path

`-o` flag specifies the name for the output file

`--` specifies the binary to instrument

This should create an empty file named `test.stats` in your current directory

Step 2b: Testing Destiny

Type the following command:

```
destiny SRAM_template.cfg
```

Destiny takes in a configuration file that specifies details about the memory it will model, e.g. size, associativity, technology node, etc.

This should output the details of the SRAM cache being modeled.

Step 2c: Testing SPEC2017

Setting up and testing the SPEC2017 toolchain is slightly more work.

First, open the `18344-f23-template.cfg` configuration file in the /`afs/ece.cmu.edu/class/ece344/opt/spec2017/config` and copy to your home directory.

Next, navigate to the `Global Settings` section that defines two fields we will modify `-- output_root` and `submit`.

Set `output_root` to: `/scratch/ece344-<andrewid>`

Step 2c: Testing SPEC2017

Set `submit` to:

```
<full-path-to-HOME>/run.sh ${benchmark} $command
```

or

```
<full-path-to-HOME>/run.py ${benchmark} $command
```

Step 2c: Testing SPEC2017

Now we need to write the script files for SPEC2017 to find:

Sample run.sh file:

```
#!/bin/bash

BENCHMARK=$1
COMMAND=${@:2}
PINTOOL="<path-to-pintool>/<pintool_name>.so"
RESULT_FILE="<your-results-dir>/${BENCHMARK}.stats"

pin -t ${PINTOOL} -o ${RESULT_FILE} -- ${COMMAND}
```

Sample run.py file:

```
import os
import sys

benchmark = sys.argv[1]
command = ‘ ‘.join(sys.argv[2:])

pintool = os.path.join( <path-to-pintool>,
‘<pintool-name>.so’ )
results_file = os.path.join( <your-results-dir>, benchmark +
‘.stats’)

pin_cmd = ‘pin -t %s -o %s -- %s’ % (pintool,
results_file, command)
os.system(pin_cmd)
```

Step 2c: Testing SPEC2017

Okay, now we have an intermediate script that will intercept the SPEC2017 information, and can customize it before passing it to Pin.

This can be used to provide benchmark-specific output filenames, and also to sweep across different Pin configurations (e.g. multiple cache sizes) for the same

SPEC2017 benchmark command, by adding simple loops in your script file.

Step 2c: Testing SPEC2017

Finally, we can run the SPEC2017 suite!

Here's the command for that:

```
runcpu -c /afs/ece.cmu.edu/class/ece344/opt/spec2017/config/18344-f21-<andrewid>.cfg --action=onlyrun
--noreportable --size=test <selected-benchmark-suite>
```

`<selected-benchmark-suite>` can be used to select single benchmarks or suites of multiple benchmarks.

For this session, we will use the `gcc_s` benchmark. This will provide a quick test.

For the rest of the class, you will use the `intspeed` benchmark suite. Remember that these benchmarks will take some time to build (~30mins), so provision time accordingly!

If all runs correctly, you should have a file named 602.gcc_s.stats in the results directory you specified in the script file. It will be an empty file, since the pintool isn't writing any output yet.

Your task for this lab will be to add the logic for counting loads and stores, as well as writing the results to the output file. Good luck!

Note: The following `intspeed` benchmarks will fail: `perlbench_s`, `deepsjeng_s`, and `specrand_s`

GDB with PinTool!

Update your Bashrc (again)

Add:

```
export PATH="/afs/ece.cmu.edu/class/ece344/opt/pin/pin-3.18-98332-gaebd7b1e6-gcc-linux/intel64/bin":$PATH
```

Alternate:

```
source /afs/ece.cmu.edu/class/ece344/bin/setup344
```

(don’t forget to remove all the stuff we told you to add earlier… sorry)

Step 1: Start GDB with PinBin

```
$ gdb pinbin
```

```
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-120.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from
/afs/ece.cmu.edu/class/ece344/opt/pin/pin-3.18-98332-gaebd7b1e6-gcc-linux/intel64/bin/pinbin...(
no debugging symbols found)...done.
(gdb)
```

Step 2: start your Pintool in another terminal

(make sure it’s on the same machine)

```
$ pin -pause_tool 10 -t obj-intel64/pintool.so -o stats.txt
-- cmd
```

```
Pausing for 10 seconds to attach to process with pid [18344]
To load the debug info to gdb use:
******************************************************************
set sysroot /not/existing/dir
file
add-symbol-file /path/to/obj-intel64/bp_pintool.so 0x7f95279d70a0
******************************************************************
```

Step 3: Add Symbol File

```
(gdb) attach 18344

(gdb) add-symbol-file /path/to/obj-intel64/bp_pintool.so
0x7f95279d70a0

add symbol table from file
"/path/to/obj-intel64/bp_pintool.so" at
    .text_addr = 0x7f7eaa8f60a0
    .data_addr = 0x7f7eaacf1880
    .bss_addr = 0x7f7eaacf2240
(y or n) y
Reading symbols from
/path/to/obj-intel64/bp_pintool.so...(no debugging symbols
found)...done.
```