Getting Started with gem5

Step 1: Setting up required software modules for gem5

This is an important step to download all the required software for your simulations.

sudo apt-get install mercurial scons swig gcc m4 python python-dev libgoogle-perftools-dev g++ zlib1g-dev git

Step 2: Download gem5

1) Create a work directory for the project by using the following command and press Enter. Or determine the work folder where you want to download gem5.

```
>$ mkdir project
>$ cd project
```

2) Download the gem5 directory from the Canvas CSE -ca Files. Move the gem5 from Downloads to the project target work directory,

```
>$ mv ~/Downloads/gem5  /path_to_project/gem5-cse-ca
```

3) You may rename the gem5-directory, for ease of access. For example:

```
>$ cp -r gem5 gem5-cse-ca
```

If you find any trouble downloading the software, please first refer to the gem5 webpage – http://gem5.org/Download

Step 3: Build gem5 (Make gem5 ready for use)

1) Enter the gem5 directory:

```
>$ cd gem5-cse-ca
```

2) Build the gem5 simulator for the target processor (here we choose the ALPHA processor, which will be used for all the projects for this class). Gem5 also supports other ISAs such as x86, ARM etc. For more details, you may (optionally) refer to the gem5 documentation (http://gem5.org/Documentation).

```
>$ scons build/ALPHA/gem5.opt
```

Wait for the build process to complete!

Note: It will take a while for the build process to complete. You can make the process fast by allowing multiple jobs to do so. For example, if your machine can run up to 8 threads, you may prefer to apply the command

```
>$ scons build/ALPHA/gem5.opt -j4
```

Step 4: Testing gem5 (Hello world!)

Once the gem5 is built for ALPHA processor, we can simulate the in-built sample test program "hello world" on gem5. To do so, type on the terminal –

```
>$  build/ALPHA/gem5.opt ./configs/example/se.py
-c tests/test-progs/hello/bin/alpha/linux/hello
```

You will see output Hello world! on the console. Output statistics are created in the directory m5out, by default. We will explore later various command-line options for gem5 simulation.

Step 5: Download and install the cross-compiler

It is often hard to obtain the binary of the benchmarks for some architectures (say ALPHA) and we need a cross-compiler to compile the source code. So, we will download and install the cross-compiler for ALPHA.

1) Download the cross-compiler

```
>$ wget  http://www.m5sim.org/dist/current/alpha_crosstool.tar.bz2
```

2) Untar the cross-compiler files.

```
>$ tar xjf alpha_crosstool.tar.bz2
```

3) The following directory (if you have followed the instructions above as is) contains the alpha cross-compiler binaries.

```
~/project/opt/crosstool/gcc-3.4.3-glibc-2.3.5/alpha-
unknown-linux-gnu/bin
```

No installation is required!

For the remainder of this document, we will refer to above directory as: **<alpha_tool_bin>**

Note: For detailed installation tips visit www.gem5.org/Introduction and watch a short six-minute video. Also, read the documentation at -

www.gem5.org/Main_Page and www.gem5.org/Documentation

Step 6: Cross-compile "hello.c" and Run it on gem5

1) Open the gem5 directory

```
>$ cd gem5-cse-ca
```

2) Create a hello world program, and save it as ***hello.c***, inside this directory.

Note: If you do this in a different directory, remember to provide the ***Path*** to that directory before the file name while executing the following commands (compiler and simulator). The similar procedure goes for benchmark programs, etc.

```
>$ <alpha_tool_bin>/alpha-unknown-linux-gnu-gcc
-static -o hello_world hello.c
```

NOTE: Please do not type **<alpha_tool_bin>** literally. This is a path shown in 3rd box of Step 5. If you have followed the document, this step should work fine. If it throws a "no such file or directory found" error the **opt** directory is not in **~/project** directory. Please type **ls** in your **gem5-cse-ca** directory and try to locate **opt** directory.

3) Now a binary file called hello_world is created in your gem5 (or working) directory. Now test your hello world program using the following command in the gem5 directory

```
>$ build/ALPHA/gem5.opt
./configs/example/se.py -c hello_world
```

4) You will get output something like this –

```
gem5 compiled Sep  3 2014 14:23:34
gem5 started Sep  4 2014 22:27:18
gem5 executing on hphn1
command line: build/ALPHA/gem5.opt configs/example/se.py -c hello
Global frequency set at 1000000000000 ticks per second
0: system.remote_gdb.listener: listening for remote gdb #0 on port 7000
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
info: Increasing stack size by one page.
Hello, world!  LONG_MIN is -9223372036854775808, PATH_MAX is 4096
Exiting @ tick 4829500 because target called exit()
```

5) The output statistics of your simulation will be available in the *"stats.txt"* file:

```
>$ less m5out/stats.txt
```

6) To include L1 and L2 caches use --caches and --l2cache flags.

```
>$  build/ALPHA/gem5.opt ./configs/example/se.py
--caches --l2cache -c hello_world
```

Note – You are given four different benchmarks in a zip folder along with the command line options to execute them. Consider the already provided cross-compiled binaries or try to compile your own binary (with step 5). Simulate the binaries of the benchmarks without and along with caches, to get a kick-starter experience.