18-640: Foundations of Computer Architecture

gem5 Start-up

1. Logistics

We expected you to sign-up for your group by now. Please do it today itself. Link to sign-up sheet is available in *BlackBoard* Announcements.

Assumptions: The projects based on gem5 assume that you are proficient in C++. Some knowledge of Python will be helpful which you should be able to pick up during the course.

2. About gem5

gem5 is written primarily in C++ and python, and most components are provided under a BSD style license. It can simulate a complete system with devices and an operating system in **full system mode (FS mode)**, or user space only programs where system services are provided directly by the simulator in **syscall emulation mode (SE mode)**. There are varying levels of support for executing Alpha, ARM, MIPS, Power, SPARC, and 64 bit x86 binaries on CPU models including two simple single CPI models, an out of order model, and an in order pipelined model.
You should go through the documentation at *www.gem5.org*.

3. Simulator Setup

1. You could run the simulator on any operating system, but **Linux Ubuntu** is highly recommended. Since the build generated is huge, you should run Linux on your local system and avoid AFS for this use. If you want to run on any other OS, you may do so, but the following instructions may be different.
   You could consider installing *VirtualBox* (or any other Virtual Machine) to run Linux Ubuntu.
2. Download gem5 from our private release.
   Copy `/afs/ece.cmu.edu/class/ece640/project/gem5.tar.bz2` into your folder.
3. Untar it using: `tar –xvjf gem5.tar.bz2`
4. `sudo apt-get update; sudo apt-get upgrade`
5. `sudo apt-get install mercurial scons swig gcc m4 python python-dev libgoogle-perftools-dev g++ zlib1g-dev`
6. Build gem5 binary: `scons build/X86/gem5.opt`
   This will build `opt` version of gem5 for x86 Architecture.

Please go through *http://www.gem5.org/Introduction* for details. The "-j" option can be used to enable parallel builds (useful on multiprocessor hosts). For example, "–j4" makes the build using 4 concurrent processes. Thus, reduces the build time.

7. You may now test your build, by running hello world program in SE mode
   `build/X86/gem5.opt configs/example/se.py -c tests/test-progs/hello/bin/x86/linux/hello`

8. Please proceed only if you get the expected output.

```
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 compiled Jul 17 2011 19:16:28
gem5 started Jul 17 2011 19:18:16
gem5 executing on zizzer
command line: ./build/ARM/m5.opt configs/example/se.py
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...
Hello world!
Exiting @ tick 3188500 because target called exit()
```

4. Benchmark:

We will be using MiBench (*http://www.eecs.umich.edu/mibench/*) for projects using syscall emulation mode (SE mode).

- The benchmarks are included in the distribution under `mibench/`
- You can run `benchmarks-compile.py` to untar and compile the following benchmarks but you are also free to compile other benchmarks in the distribution:
  1. `consumer/jpeg`
  2. `automotive/qsort`
  3. `network/dijkstra`
  4. `office/stringsearch`
  5. `telecomm/FFT`
- gem5 doesn't yet support dynamic linking; so all benchmarks must be statically linked. Thus at compile time make sure `-static` flag is passed to `gcc`. For the above 5 benchmarks we have ensured `-static` flag.

5. Understanding gem5 in SE mode:

Example command that you will be implementing:

```
build/X86/gem5.opt configs/example/se.py -c binary [-o options] --cpu-type=detailed --caches
```

Explanation:

- `build/X86/gem5.opt` - gem5 binary
- `configs/example/se.py` - simulation script
- `-c binary` - binary to run in SE mode
- `-o options` - cmd line options for the binary
- `--cpu-type=detailed` - specify cpu type (In Order, Out of Order etc.)
- `--caches` - caches enable

After simulation, the stats will be generated in `m5out/` folder.

`.config` file is the configuration of the simulation performed

`.stat` file is the statistics generated at the end of simulation. You will have to extract data from this file to analyze information.