ECEC 621
Simulation Project – Introduction to gem5

Due November 5th 2012

1 Overview

This quarter you will be working on a research project which relates to architecture design and evaluation. Consequently, you will need to be proficient with using a processor simulator. This assignment is intended to be a gentle introduction to processor simulation.

2 Getting Started

In this class, we will use the GEM5 simulator. To work with the GEM5 simulator, you will need to download the simulator and untar it by following these steps:

1. Login to `xunil.coe.drexel.edu` via SSH with your Drexel user id and password. Note: You should use either the linux tool "screen" or computer with a reliable Internet connection; both build and simulation tasks can take over an hour.
2. download the GEM5 simulator from gem5.org to your home directory using the following command:
   `> wget http://repo.gem5.org/gem5-stable/archive/tip.tar.bz2`
3. Unpack the archive with the command:
   `> tar -xjvf tip.tar.bz2` (execute `man tar` to get the manual-page for an explanation of the `tar` command)

   You will now have a gem5-stable directory located in your home directory that contains the GEM5 simulator.
4. Navigate to the gem5-stable directory.
   `> cd gem5-stable`

3 Building an Architecture

GEM5 is a robust architecture simulator that allows you to model many different ISAs, including ALPHA, ARM, POWER, MIPS, and x86. For many of these ISAs, you can configure either in-order and out-of-order (OoO or O3) processor models. The list of available ISAs is in the `gem5-stable/build` directory.

For this project, we will be building an out-of-order ALPHA processor. From your gem5-stable directory, execute the following command:

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

`scons` is a similar to `make`, and compiles the code for the simulator, building the gem5.opt executable in the ~`/gem5-stable/build/ALPHA` directory

4 Basic Usage

Now we will get started with the GEM5 simulator through a very basic usage example. Here, we will execute a set of benchmarks on GEM5 called Splash2. Splash2 contains several standard algorithms:

`Cholesky`, `FFT`, `LUContig`, `LUNoncontig`, `Radix`, `Barnes`, `FFM`, `OceanContig`, `OceanNoncontig`, `Raytrace`, `WaterNSquared`, and `WaterSpatial`.

Here are the steps to run a splash-2 benchmark on GEM5:

1. Navigate to your home directory using the following command:
   ```
   > cd ~
   ```

2. Download the splash-2 benchmarks using the following command:
   ```
   > wget http://www.gem5.org/dist/m5_benchmarks/v1-splash-alpha.tgz
   ```

3. Extract the splash-2 benchmarks:
   ```
   > tar -xzvf v1-splash-alpha.tgz
   ```
   You now have a directory `~/splash2` containing the benchmarks

4. Navigate to your build directory `> cd ~/gem5-stable/build/ALPHA`

5. Copy the `runSplash.py` file from my folder to your build directory
   ```
   > cp /home/DREXEL/sjb328/runSplash.py .
   ```

6. Invoke the GEM5 simulator with the following command line:
   ```
   > gem5.opt runSplash.py -d -n 4 --l1size 32kB --cacheblocksize 64 -b FFT
   --rootdir=/home/DREXEL/< drexel_id >/splash2/codes
   ```
   –l1size configures the L1 cache size –cacheblocksize configures the cache line size, the amount of data transferred from memory on a cache miss. -d tells GEM5 to use the detailed out-of-order model.
   -n 4 tells GEM5 to create a system with 4 CPUs
   -b FFT tells GEM5 to run the FFT benchmark
   –rootdir tells GEM5 where to find the benchmarks, replace < *drexel_id* > with your username

The simulator will run and generate output in the directory ./m5out. Here you will find `config.ini` containing the simulation parameters and `stats.txt` containing the results of the simulation.

5 Experiment

There are two parts to this assignment. First, you will evaluate how cache configurations affect benchmark performance. Next, you will modify an architectural parameter, and see how that affects performance.

5.1 Cache

In this experiment you will use GEM5 to investigate cache miss rates, bus traffic, and execution time under different system configurations. You will be provided with a base cache/processor configuration and you will perform a set of experiments designed to test the effect of scalability and cache capacity. Pick one of the following experiments:

- keep L1 cache capacity fixed at 32KB, block size fixed at 64B and vary number of processors (2, 4, 8)

- keep number of processors fixed at 4, block size fixed at 64B and vary L1 cache capacity (16KB, 32KB, 64KB)
- keep L1 cache capacity fixed at 32KB, number of processors fixed at 4 and vary L1 cache block size (32B, 64B, 128B)

You should use the default cache coherence protocol (MSI), L2, and memory configurations. Pick any two benchmarks of your choosing out of the Splash2 suite: `Cholesky, FFT,LUContig, LUNoncontig, Radix, Barnes, FFM, OceanContig, OceanNoncontig, Raytrace, WaterNSquared,` or `WaterSpatial`. You should submit a brief project report which explains your configuration and reports:

- Data L1 cache miss rate (in each experiment, use the avg over all processors)
- L2 bus traffic (transactions per processor)
- Total performance (execution cycles), and IPC

You should explain how varying the parameters impacts your results. Then offer your guess as to why you see the results you do. What do the results tell you about your workloads? Plot IPC, miss rate, bus traffic, and execution cycles for each benchmark across your configurations.

5.2 Architecture

In this experiment, you will use GEM5 to investigate how architecture affects performance. You will use the default cache configuration `--l1size 32kB --cacheblocksize 64` and modify the base processor configuration.
The architecture is defined in `~/gem5-stable/src/cpu/o3/O3CPU.py`. When you build an architecture using `scons`, it reads this file.

Pick one of the following parameters and sweep across the range given below:

- `predType` : change from "tournament" to "local", and sweep `localPredictorSize` : range [512, 1024 ,2048, 4096], default 2048
- `numPhysIntRegs` : range [64, 128, 256, 512], default 256
- `issueWidth` : range [4, 8, 16, 32], default 8
- `commitWidth` : range [4, 8, 16, 32], default 8
- `LQEntries` : range [8, 16, 32, 64], default 32

1. Edit the `~/gem5-stable/src/cpu/o3/O3CPU.py` file to change the chosen parameter and value.
2. Rebuild the processor simulator by executing the following from your `~/gem5-stable` directory
   ```
   > scons build/ALPHA/gem5.opt
   ```
3. Simulate 2 benchmarks, record execution cycles, IPC, and mispredicted conditional branches.
4. Go to 1 and configure the next value in the range.

You should explain how varying the parameters impacts your results, then offer your guess as to why you see the results you do, what the trade-offs are involved when varying your chosen parameter. Does anything else in the architecture need to change along with the parameter you have chosen? What do the results tell you about your workloads? Plot execution cycles, IPC, and branch mispredicts for each benchmark across your configurations.