# Computer Architecture
# Lab Assignment 1: Gem5 Basics

Nanjing University, Spring, 2017

## I. Introduction

In this assignment, you will learn how to use the gem5 simulator, which is a modular platform for computer-system architecture research. We recommend that you run the gem5 simulator under a Linux system. We will use the gem5 to simulate a system using the X86 ISA.

## II. Install and Test gem5 on Your Machine

Please follow the gem5 tutorial provided by University of Wisconsin-Madison. You can use this link to access the tutorial: http://learning.gem5.org/book/index.html. Please finish the “Part I: Getting started with gem5” by following the steps in the tutorial. Note that as the gem5 simulator is continuously changing, you might encounter some problems when following the steps. For example, some of the scripts used in the tutorial may be outdated. You may need to modify the script based on your own understandings of the system. Hint: you can look at the scripts under the gem5 source code pulled from github, under gem5/configs/learning_gem5.

## III. Write Your Test Program

In this assignment, you will study the performance of your own program instead of the “hello.c” in the tutorial. First, please write a C program to find out the number of primes for integers $\leq$ 1,000,000. You should use the Sieve of Eratosthenes algorithm for your program. Please follow the pseudocode listed in the Wiki page. Be careful about the data types used in your program. Your program should print out a single number of 78498 as the result.

After writing your sieve program, you should first compile and run it on your machine to see if everything is alright. Then, please modify the configurations of the “two_level.py” machine, to run the compiled X86 program on the simulator. Record the number of ticks and the output “stats.txt” of your simulation .

## IV. Try Different Parameters

The sieve algorithm is not very cache friendly. You can try different settings of machines to understand the cache performance. The basic setting for your simulation should be a “Simple Timing CPU” at 1 GHz clock with L1 instruction cache size of 16 kB, L1 data cache size of 64 kB and L2 cache size of 256 kB. The default cache block size is 64 Bytes. Other parameters, such as memory type and cache hit latency, can be set as in the tutorial.

When changing the system parameters, we recommend that you fix all other parameters and only change a single parameter at a time to see the effects of the change. You can make the following changes:

- Change the clock rate of the CPU to 0.5, 1, 2, and 3 GHz
- Change the L1 instruction cache size to 2, 4, 8, 16, 32, and 64 kB.
- Change the L1 data cache size to 2, 4, 8, 16, 32, 64, 128, and 256 kB.
- Change the L2 cache size to 128, 256, 512, 1024, 2048, and 4096 kB.
- Change the system cache block size (system.cache_line_size) to 8, 16, 32, 64, 128, and 256 Bytes.

You are recommended to collect stats of your simulation, including (but not limited to):

- Total time to run the program

- Clock cycles Per Instruction (CPI)
- L1 Instruction cache miss rate (e.g., system.cpu.icache.overall_miss_rate::total)
- L1 Data cache miss rate (e.g., system.cpu.dcache.demand_miss_rate::total)
- L2 cache miss rate (e.g., system.l2cache.overall_miss_rate::total)

While doing the experiments, we recommend that you should write scripts to automatically execute your simulations. Your script should be able to automatically change the parameters of your machine, run the simulation, save/copy the results and extract stats from "stats.txt". In the ideal case, you should be able to hit the "Enter" key, then take a nap, and your script will automatically run all the experiments and draw the plots for you after a few hours.

## V. Submission Requirements

You should submit both your code and your report in a zip archive. Your codes include the test program, configuration scripts, execution scripts, and your raw "stats.txt" file.

In your assignment report, you should include the following diagrams:

1) Diagram of program running time with respect to the CPU clock rate
2) Diagram of CPI with respect to the CPU clock rate
3) Diagram of L1 instruction miss rate with respect to the L1 instruction cache size
4) Diagram of L1 data miss rate with respect to the L1 data cache size
5) Diagram of L2 miss rate with respect to the L2 cache size
6) Diagram of L1/L2 miss rate with respect to the cache block size

You should also answer the following questions based on your observations:

1) How is the running time related to the CPU clock rate, why?
2) What is the impact of the size of cache block on the miss rate, why?
3) What is the impact of cache size on cache miss rate, why?

You can write your report in Chinese.

Please submit both your code and your assignment report in a zip archive to the course web site before the deadline. Any late submission will be rejected.

## VI. Acknowledgement

This lab assignment is adapted form homework of "CS 752: Advanced Computer Architecture I", University of Wisconsin-Madison.