CSE 490/590, Project 2, Summer 2025

Analysis Of Cache Parameters And Trade-Offs Using Gem5

DUE: Tuesday, July 1, 2025 @ 11:59pm

This is a group project. We will use the same groups from Project 1 for Project 2. If you would like to make changes to your group, please email the professor or one of the TA listed on the course website..

There must be **equal contribution to the project from all members of the group**. Individual contributions and specific tasks will be checked in the final project report

Cache parameters influence the performance of a processor to a great extent. In this project you are asked to tune various cache parameters and analyze performance in each case.

Gem5 Introduction

The Gem5 simulator is a modular platform for computer-architecture research. We will be *indirectly* using Gem5 (via a shell script) to observe the changes in performance of an x86 processor as certain parameters, such as cache size, associativity, and block size are changed.

Accessing Gem5 on the CSE server "cerf" or "turing"

The CSE servers on which Gem5 and the shell script has been installed for your use are 'cerf.cse.buffalo.edu' or 'turing.cse.buffalo.edu', which are accessible via VPN from off-campus machines. Specifically (if you are off-campus) you need to use the AnyConnect VPN to connect to UB's CSE servers. You can get AnyConnect from here:
https://www.buffalo.edu/ubit/service-guides/software/by-title/anyconnect.html

See Appendix A for information on how to connect to the CSE servers in a terminal.

We will provide you a shell script called 'gemTest.csh' which simplifies running CPU benchmarks with Gem5. See Appendix B for how to use the script. (See Appendix B for where these files will be located).

CPU Benchmarks

CPU benchmarks are used to simulate the architectural designs of processors to evaluate their performance. In this project there are five programs (listed below) which are used as benchmarks. What Gem5 does is simulate running these programs on a virtual CPU and cache, and measures various statistics of the cache utilization, generating a report at the end of these statistics which can be analyzed.

The benchmarks that will be used are: **(your group will be assigned three of these)**

1. 401.bzip2
2. 429.mcf
3. 456.hmmer
4. 458.sjeng
5. 470.lbm

Once you are in a group, we will assign your group **three** of the listed benchmarks that are to be used for your analysis.

Cache Parameters

The cache parameters are the characteristics that can be changed to yield different performance profiles and behavior.

The parameters that can be varied are: **(your group will be assigned five of these)**

1. L1 D Cache size
2. L1 I Cache size
3. L2 Cache size
4. L1 D Associativity
5. L1 I Associativity
6. L2 Associativity
7. Block Size

Once you are in a group, we will assign your group **five** of the listed parameters that are to be used for your analysis.

Experiment Execution

When running your tests you will follow this process:

Important Terminology:

“**Benchmark Test Set”:** a collection of **Parameter Test Sets**.
“Parameter Test Set”: a collection of **test cases**.
“**Test Case”:** a single test; in the context of Project 2, a single run of Gem5.

For each of your three assigned benchmarks, each of you will run a **Benchmark Test Set.** For each **Benchmark Test Set** you will set a baseline for **all seven** parameters. (Input values are still needed for the parameters that your group was not assigned)

In a **Benchmark Test Set** there will be five **Parameter Test Sets**, one for each of your assigned parameters. In each **Parameter Test Set** changes will be made to **only** a single parameter associated with the current **Parameter Test Set**.

In a **Parameter Test Set**, there will be five **Test Cases**. In each test case **only** the value of the parameter associated with the **Parameter Test Set** will be changed. Each **Test Case** should have a different value for the associated parameter from the other test **Test Cases**. You are responsible for choosing how the parameters are varied.

For a single **Test Case**, you will run Gem5 with the provided shell script (gemTest.csh) with the parameters for that test **Test Case**. Once the results are printed, you will record those results for later analysis.

CSE 490/590, Project 2, Summer 2025

Here is a table for a single **Benchmark Test Set** to help visualize the tests to be done:

Benchmark	**Changed Parameter**	**Test Case 1**	**Test Case 2**	**Test Case 3**	**Test Case 4**	**Test Case 5**
X	A					
	B					
	C					
	D					
	E					

Analysis

For each of your assigned test cases, calculate the following information:

1. Hit rate of L1 D cache
2. Hit rate of L1 I cache
3. Hit rate of L2 cache
4. CPI

To approximate the CPI use the formula given below:

$$CPI = 1 + \frac{(IL1.miss\_num + DL1.miss\_num) \times 6 + L2.miss\_num \times 50}{Total\_Inst\_num}$$

With all the information that you collect and compute: **create graphs, charts, and/or plots of the CPI** to show trends in how changes in each parameter effect the CPI of the simulated x86 processor.

Project Report

The requirements for the project report are as follows:

- **[Most Important]** An **explanation for why** the trends in the graphs are there in the first place (how did changes in the cache parameters result in the trends that are observed).
- A step by step explanation of how you completed this project.
- Tabulated benchmarks with the hit rates, miss rates and CPI obtained in each test case.
- Any graphs, charts, and/or plots created for the CPI.

Useful points to note

1. Memory Background Lecture:
    a. https://cse.buffalo.edu/~rsridhar/cse490-590/lec/ch2.pdf
    b. https://cse.buffalo.edu/~rsridhar/cse490-590/lec/Mem-background-updated.pdf

Submission

On completion, each project team should turn in a compressed folder with the following items:

1. Your project report.
2. Another compressed folder that contains all the stats.txt files, with proper naming for identification. (See Appendix B for where to find your stats files.)

Appendix A - Connecting to the CSE server

Windows CSE server Connection

To access the sever on Windows you can either use the Putty terminal, available at: https://www.buffalo.edu/ubit/service-guides/software/downloading/windows-software/managing-your-software/putty.html

Or in Powershell (a built in terminal in Windows 10 and 11) you can use the command:

```
ssh YOUR_UBIT@cerf.cse.buffalo.edu
```

Where "`YOUR_UBIT`" is replaced by your UBIT name (the beginning part of your UB email). You will then have to provide your UBIT password.

MacOS and Linux CSE server Connection

On MacOS the is an application called "Terminal" which provides a terminal emulator where you can input commands. A similar utility should be available on many Linux distributions.

On MacOS and Linux the same command can be used in the terminal:

```
ssh YOUR_UBIT@cerf.cse.buffalo.edu
```

Where "`YOUR_UBIT`" is replaced by your UBIT name (the beginning part of your UB email). You will then have to provide your UBIT password.

Appendix B - Using the shell script

For this project, we provide you a shell script called 'gemTest.csh' which simplifies running CPU benchmarks with Gem5. The shell script is available on cerf in the `/shared/projects/CSE490/project2` directory.

The shell script provided for this project has multiple arguments that are needed to use the script. The general format for using the script is as follows:

```
/shared/projects/CSE490/project2/gemTest.csh <benchmark name> <output name>
<L1D Size> <L1I Size> <L2 Size> <L1D Assoc> <L1I Assoc> <L2 Assoc> <Block
Size>
```

Where:

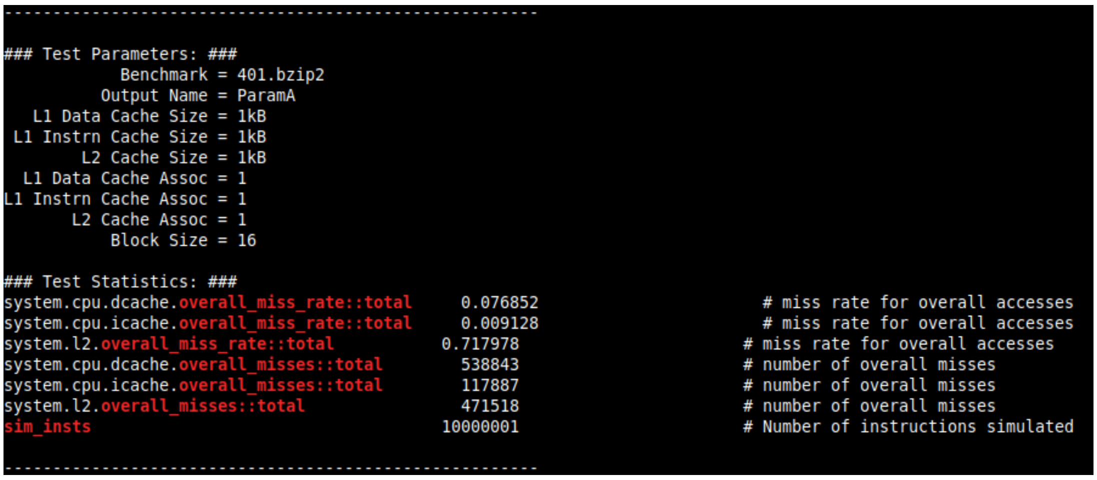
<test name>	is the name of the benchmark that is to be run
<output name>	is the name of the stats file produced from the cache simulation
<L1D Size>	is the L1 data cache size
<L1I Size>	is the L1 instruction cache size
<L2 Size>	is the L2 cache size
<L1D Assoc>	is the L1 data cache associativity
<L1I Assoc>	is th L1 instruction cache associativity
<L2 Assoc>	is the L2 cache associativity
<Block Size>	is the block size

Note: For the cache sizes, the value must end exactly with 'kB', such as: '1kB'.

Example command:

```
/shared/projects/CSE490/project2/gemTest.csh 401.bzip2 ParamA 1kB 1kB 1kB 1 1 1 16
```

When the script is run it will produce an output that looks like this:



```
-------------------------------------------------------

### Test Parameters: ###
            Benchmark = 401.bzip2
          Output Name = ParamA
   L1 Data Cache Size = 1kB
 L1 Instrn Cache Size = 1kB
        L2 Cache Size = 1kB
  L1 Data Cache Assoc = 1
L1 Instrn Cache Assoc = 1
       L2 Cache Assoc = 1
           Block Size = 16

### Test Statistics: ###
system.cpu.dcache.overall_miss_rate::total     0.076852                      # miss rate for overall accesses
system.cpu.icache.overall_miss_rate::total     0.009128                      # miss rate for overall accesses
system.l2.overall_miss_rate::total           0.717978                      # miss rate for overall accesses
system.cpu.dcache.overall_misses::total        538843                      # number of overall misses
system.cpu.icache.overall_misses::total        117887                      # number of overall misses
system.l2.overall_misses::total                471518                      # number of overall misses
sim_insts                                    10000001                      # Number of instructions simulated

-------------------------------------------------------
```

In the "Test Statistics" section, the information listed is as follows: L1 data cache misses, L1 instruction cache misses, L2 cache misses, total instructions simulated. This is the information that is important for this project.

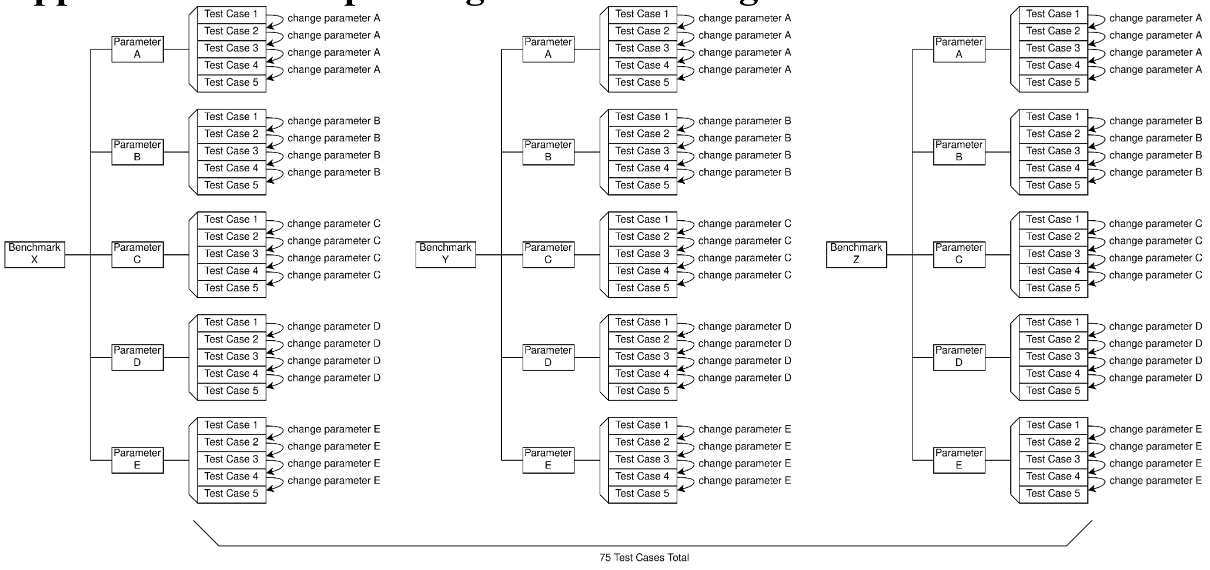
The script will also produce output in a file called "stat_files". This directory contains all the stats files produced from each Gem5 simulation that was run (these are the files that need to be uploaded as part of the final submission). For example, for the command:

```
/shared/projects/CSE490/project2/gemTest.csh 401.bzip2 ParamA 1kB 1kB 1kB 1 1 1 16
```

the following file is created: `stat_files/401.bzip2/ParamA_stats.txt` (this is relative to where you execute the script). Thus after running multiple simulations you may see a directory that looks something like this:

```
[dir] stat_files
      |--  [dir] 401.bzip2
      |          |-- [file] ParamA_stats.txt
      |          |-- [file] ParamB_stats.txt
      |          |-- [file] ParamC_stats.txt
      |--  [dir] 429.mcf
                 |-- [file] ParamA_stats.txt
                 |-- [file] ParamB_stats.txt
                 |-- [file] ParamC_stats.txt
```

Appendix C - Example Diagram of Running Test Cases


Benchmark X
Benchmark Y
Benchmark Z
Parameter A
Parameter B
Parameter C
Parameter D
Parameter E
Test Case 1
Test Case 2
Test Case 3
Test Case 4
Test Case 5
change parameter A
change parameter B
change parameter C
change parameter D
change parameter E
75 Test Cases Total