Lab 5: gem5 CPU Models & SW/HW Interaction on Performance

EEL 5734 – Fall 2022

Introduction:

After two labs using gem5 to explore basic characteristics of caches, the objectives of Lab 5 are:

- Explore the directory structure and contents of ~/gem5/ in preparation for the project
- Explore gem5.org and, in particular, the gem5 bootcamp videos.
- Experiment with parameters of a gem5 CPU model (**O3**CPU: **O**ut-**O**f-**O**rder CPU)
- Explore the effect of software/hardware interaction on performance

Part 0 – Explore directory structure and contents of ~/gem5/

From https://www.gem5.org/ "The gem5 simulator is a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor microarchitecture". In the previous labs in the class, we have created the configuration scripts "from scratch", following the tutorials from gem5.org. This is very useful, as it allows you to specify every single system parameter. However, realistic systems are very complex to set up. For that reason, the gem5 developers have provided many scripts to bootstrap the process of building systems for research (and for your class projects)

For Part 0, we will, in class, browse through the directory structure ~/gem5/ and explore their contents in preparation for the project. In particular, we will explore what is provided in the following directories:

- ~/gem5/src/
- ~/gem5/src/cpu
- ~/gem5/src/arch

We will also explore gem5.org and, in particular, the gem5 bootcamp videos:
https://www.youtube.com/channel/UCCpCGEj_835WYmbB0g96lZw/videos

Part A – Experimentation with O3CPU parameters

gem5 provides several CPU models, including a detailed model of an out-of-order CPU (**O3CPU**), which we will explore in this lab. Although not necessary for this lab, but if you are interested, you can find documentation for the gem5 O3CPU at:

https://www.gem5.org/documentation/general_docs/cpu_models/O3CPU,

Part A(1): In this part of Lab 5, you will vary 3 parameters for O3CPU, the details of these parameters can be found in the configuration script at: ~/gem5/src/cpu/o3/BaseO3CPU.py

** **IMPORTANT**: You have to make it so these parameters can be changed in the **"build" command line** (details below)

(1) **numIQEntries** (Reservation Station size: "number of instruction queue entries")
- Defined in the class **BaseO3CPU** within the **BaseO3CPU.py** configuration file.
- Note its data type and its default value

(2) **LQEntries** ("number of load queue entries")
- Defined in the class **BaseO3CPU** within the **BaseO3CPU.py** configuration file.
- Note its data type and its default value

(3) **numROBEntries** (ROB size: "number of reorder buffer entries")
- Defined in the class **BaseO3CPU** within the **BaseO3CPU.py** configuration file.
- Note its data type and its default value

Helpful instructions:

As you did in Lab 4, make it so these parameters can be specified in the "build" command line.

- Use your two-levelLab4Part3.py as a starting point. Copy it to **lab5O3CPU.py**
  - Where you added parser.add_argument (…) statement for l1i_data_latency (in Lab 4), do the same for the three parameters for Lab 5.
  - You need to set the system.cpu to O3CPU (instead of TimingSimpleCPU)
  - **Important note: In Lab 4**, where you added an if statement **in the caches.py** to see if a parameter was used in the command. For example:

    ```
    if options and options.l1i_size:
        self.size = options.l1i_size
    ```

    In Lab 5, you also have to add an if statement for each of the 3 new parameters. However, you don't add those if statements in caches.py, you add them here in Lab5O3CPU.py, right after the system.cpu = O3CPU() statement. For example, for parameter xxx, the if statement would be in the form of:

    ```
    if options.xxx:
        system.cpu.xxx = options.xxx
    ```

- Then from the command line (as you did in Lab 4), you can specify values for these parameters. For example:
  build/X86/gem5.opt configs/tutorial/Lab5a/lab5Minor.py --xxx=somevalue

Part A(2): Given (as a part of this assignment) is the matmult code (000-matmult.c) used in the previous labs, except most of the print statements have been taken out. Perform experiments on the three parameters in the following manner, using the **provided matmult code**.

Using the command line, collect the performance for:

- Vary the value of the **numIQEntries** parameter as follows: 1, 32, 48, 64, 96, 128
- Vary the value of the **LQEntries** parameter as follows: 1, 16, 24, 32, 48, 64
- Vary the value of the **numROBEntries** parameter as follows: 1, 48, 96, 192, 288, 384

For submission, the outputs should be summarized in **three tables** (one for each parameter). No observation paragraph is necessary.

Part B – Software/hardware interaction on performance

In Part B, we will explore the connection between software (application, compiler) and a given hardware architecture with respect to performance.

Part B(1): For this part, also use the provided matmult code (000-matmult.c). Note that the initialization of matrices A and B, and the calculation of C, are performed in "row-major" order:

```
for (i=0; i<nrows; i++)
     for (j=0; j<ncols; j++)
```

You will simulate 8 different variations of the matmult. For all 8 cases, using the "build" command line, simulate the code with default values for all parameters, **except l1d_size=1kB**

For simplicity, I will label the 8 cases from 000, 001, … , 111 (corresponding to matrix names A, B, C); where "0" represents the case where the i and j are kept in the original order and "1" indicates reverse order, iterating j first as shown here:

```
for (j=0; j<ncols; j++)
     for (i=0; i<,nrows; i++)
```

For example, the label 101 (corresponding to ABC) represents the case:

- (1) Reverse the order in initializing matrix A
- (0) Keep the initialization of B in the original i,j order
- (1) Reverse the order in calculating matrix C

Thus, 000 represents the original given matmult code (000-matmult.c) where the initialization of matrices A and B, and the calculation of C, are in "row major". And 111 represents the case where the initialization of matrices A and B, and the calculation of C, are all "reversed".

Notes:

- For each of the 8 cases, you need to compile using the following gcc command, where XXX is 000, 001, …, 111:

  gcc -O0 -ggdb3 -std=c99 -static -o matmult **XXX**-matmult.c

- Again, simulate the code with default values for all parameters, **except l1d_size=1kB**

  build/X86/gem5.opt configs/tutorial/lab5Part2.py --l1d_size=1kB

For submission, the outputs of all 8 cases should be summarized in a table:

A B C	Number of ticks
0 0 0	
0 0 1	
. . .	
1 1 1	

Also include a short paragraph giving your observation of the pattern of the results.

Part B(2): In Part B(1), you observe the change in performance based on how you structure the code in your application. In this part, let's see how the optimization(s) in a compiler can change the performance.

Using the original given matmult code (000-matmult.c), compile and simulate the code using different level of optimization of gcc. Also, keep **l1d_size=1kB** in the build command.

- Optimization level 0: gcc -**O0** -ggdb3 -std=c99 -static -o matmult 000-matmult.c
- Optimization level 1: gcc -**O1** -ggdb3 -std=c99 -static -o matmult 000-matmult.c
- Optimization level 2: gcc -**O2** -ggdb3 -std=c99 -static -o matmult 000-matmult.c
- Optimization level 3: gcc -**O3** -ggdb3 -std=c99 -static -o matmult 000-matmult.c

For submission, put the outputs in a table of 4 rows, **followed by a sentence summarizing** your observation of the result.

Lab 5: gem5 CPU Models & SW/HW Interaction on Performance

EEL 5734 – Fall 2022

SUBMISSION INSTRUCTIONS: Turn in **one pdf file** and **no zip file**.

The pdf file should have the following naming convention:

LastNameFirstNameLab5.pdf ex: LamHermanLab5.pdf

The **pdf file** contains the following **5 pages**.

Pages 1 and 2: From Part A(1)

- From Part A(1) - Printout of your **lab5O3CPU.py** file, showing all the changes you made so you can specify the parameter values from a "build" command line.

Page 3: From Part A(2)

- Three tables (one for each parameter) summarizing the parameter values and the performance (number of ticks)
- No observation paragraph is necessary

Page 4: From Part B(1)

- The performance of all 8 cases should be summarized in a table (as illustrated above)
- A **short paragraph** giving your observation of the pattern of the results

Page 5: From Part B(2)

- One table of 4 rows, for the 4 optimization levels and their performance (number of ticks)
- **One sentence** summarizing your observation