Project 1 Instructions

Objective: Become familiar with gem5 and learn implications of different processor aspects like Fetch Width, Commit Width etc., and different execution units.

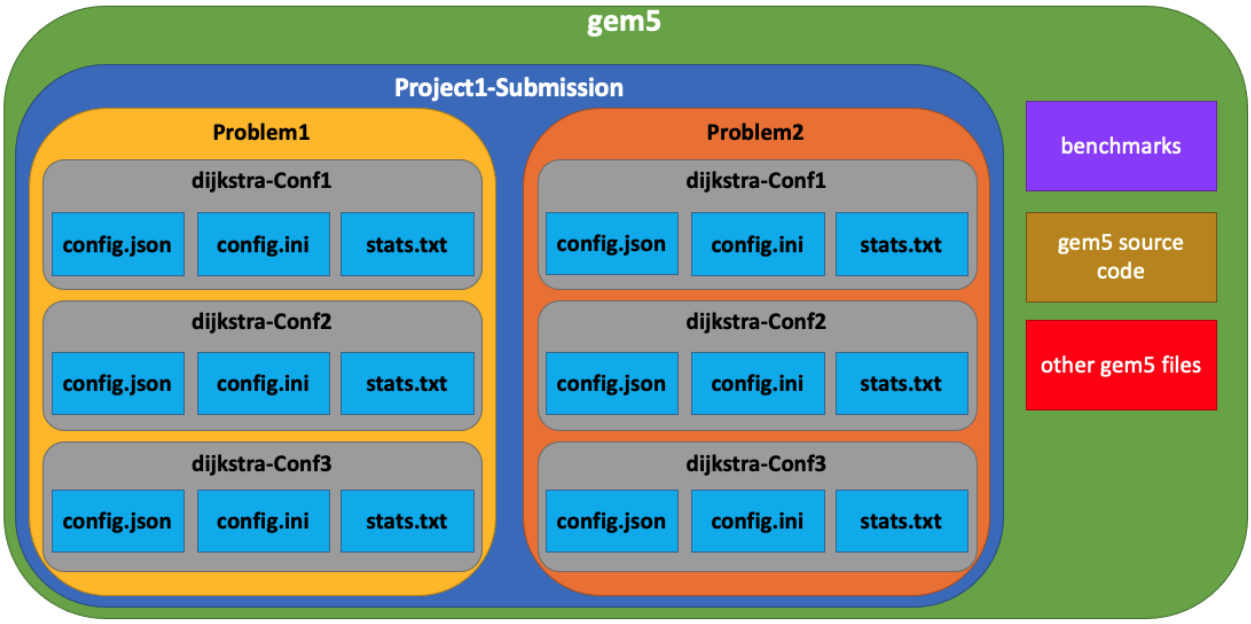
In this project, you will be required to use the gem5 processor simulator, which is popular in academia. You will do experiments with different benchmarks and will learn how to simulate the performance on gem5. For various benchmarks, you will analyze the impact of different cache parameters on performance metrics. Please read and complete the "Getting Started with gem5" document to set up your system with gem5 with the correct specifications before starting Project 1 (this document).

Note: We are using Gem5 to simulate the ARM processor. If you followed all steps in the Getting Started document, this should already be configured.

Benchmarks: Performance based system-design has made benchmarking a crucial aspect of the design process. Different benchmarks target specific areas of the computation. For example, widely used SPEC (Standard Performance Evaluation Corporation) CPU benchmarks characterize workloads for general purpose computers. They are used to evaluate the efficacy of different microarchitecture design aspects such as different data or instruction-level parallelism techniques, sophisticated caching, or branch-prediction techniques etc. For this project, we will use the benchmarks from MiBench suite consisting of benchmarks for embedded processors. Mainly, we will deal with networking benchmark dijkstra, binary of which is provided to you. You may also simulate other benchmarks such as FFT, qsort or basicmath on gem5 and analyze the performance statistics.

CPU Model: Gem5 can simulate several CPU models. Please refer to https://www.gem5.org/documentation/general_docs/cpu_models/SimpleCPU to learn more about them. AtomicSimpleCPU model is a functional simulator that uses atomic memory accesses, and can be used to profile instructions, and collect simulation statistics. The TimingSimpleCPU model uses timing memory accesses and models the memory accesses in greater detail – which is needed here while dealing with different cache configurations. In this project we will be using an Out of Order CPU model called DerivO3CPU. The O3 CPU model has various factors like fetch width, issue width, integer and floating-point arithmetic logic units (IntALU, FP_ALU), integer and floating point Multiply/Divide unit, SIMD etc.

Folder Setup: From the "Introduction to gem5" slides in lecture, remember that the purpose of this project is to change the processor to 3 different configurations of parameter values and then do the same thing for execution unit parameters. Therefore, we must run the benchmark command on each new configuration. The result of running the benchmark are 3 new files: "config.ini", "config.json", and "stats.txt". We need to store these files somewhere unique to each configuration, so we must make the directory structure as such:



Initially the config folders (the gray folders in the diagram) will be empty, as the config files will only be added automatically after running the benchmarks. To create such a structure, go to the gem5 directory and run the following command:

```
mkdir Project1-Submission Project1-Submission/Problem1
Project1-Submission/Problem1/dijkstra-Conf1
Project1-Submission/Problem1/dijkstra-Conf2
Project1-Submission/Problem1/dijkstra-Conf3
Project1-Submission/Problem2
Project1-Submission/Problem2/dijkstra-Conf1
Project1-Submission/Problem2/dijkstra-Conf2
Project1-Submission/Problem2/dijkstra-Conf3
```

Now that the structure is set up, every time you run a benchmark, you just need to specify the path to the target directory, it will automatically populate with the correct output files. First, you must move the benchmarks directory into the gem5 directory with the following command:

```
mv benchmarks gem5
```

Problem 1 [5 Points]: Impact of Various Widths of O3 processor.

An Out of Order processor consists of fetch width, decode width, rename width, dispatch width, issue width, writeback width, commit width, and squash width. They affect the performance of the processor like the execution time, IPC, etc. In this project you will be changing the following parameters to observe three different configurations.

Task: Change the following parameters in O3CPU.py.

1. fetchWidth
2. decodeWidth
3. renameWidth
4. dispatchWidth
5. issueWidth
6. wbWidth
7. commitWidth
8. squashWidth

Configuration #	Fetch Width	Decode Width	Rename Width	Dispatch Width	Issue Width	WB Width	Commit Width	Squash Width
1 (default)	8	8	8	8	8	8	8	8
2	4	4	4	4	4	4	4	4
3	2	2	2	2	2	2	2	2

If you are not using a hypervisor and only have access to a terminal/command line, you can use any built-in text editor like Vim, Emacs, or Nano. Vim is the most popular, so here is a brief explanation on how to use it. Install Vim with the command "`sudo apt install vim`". You must first display the file with the command "vim [file name]". Vim by default shows the text in a read only mode. If you want to edit a file, you then press the "i" key to enable "insert mode". To leave "insert mode", just press the "esc" escape key. To quit without saving you must first leave "insert mode" by pressing the "esc" escape key, then press ":q" (colon - q - enter). To quit after saving, you must first leave "insert mode" by pressing the "esc" escape key, then press ":wq" (colon - w - q - enter).

```
vim src/cpu/o3/O3CPU.py
```

After modifying the O3CPU.py with the correct values, you need to rebuild the gem5 for each configuration. This build process allows gem5 to create the object files to incorporate the changes made to the source code. To (re)build gem5, run the following in the gem5 directory:

```
scons build/ARM/gem5.opt
```

Then, run the Dijkstra benchmark command on gem5 with the modified configurations:

```
./build/ARM/gem5.opt -d [path to target directory]
./configs/example/se.py --cpu-type=DerivO3CPU --caches --l2cache -c
./benchmarks/dijkstra/dijkstra_small -o ./benchmarks/dijkstra/input.dat
```

For the parameter *[path to target directory field]*, put the path to the correct dijkstra-Config folder for the respective configuration described by the table above. For example, put "Project1-Submission/Problem1/Dijkstra-Conf1" for configuration 1, "Project1-Submission/Problem1/Dijkstra-Conf2" for configuration 2, and so on. This will make the relevant stats files automatically deposit in each config folder.

You may notice that additional files have been stored in the specified config directory.

```
config.ini  config.json  fs  stats.txt
```

Since we only need "config.ini", "config.json", and "stats.txt", you can delete the other files with the command:

```
rm -rf [file or folder name]
```

```
config.ini  config.json  stats.txt
```

Repeat this entire process (edit, build, benchmark) for all 3 configurations. The expected end result should be as depicted in the "Project Deliverables" section below. **Remember to rebuild gem5 every time you change the values of the parameters and before you run the benchmark command**.

Problem 2 [5 Points]: Impact of Execution Unit.

****NOTE****: *Please return O3CPU.py back to the default configuration (back to '8') for all of the parameters.*

There are multiple execution units in the Out of Order processor in gem5, like IntALU (for integer computation), IntMultDiv (for integer multiply and divide), FP_ALU (floating point computation), FP_MultDiv (floating multiply and Divide), SIMD etc. You can find these in the following file:

```
vim src/cpu/o3/FuncUnitConfig.py
```

Configuration #	IntALU count	IntMultDiv count	FP_ALU count	FP_MultDiv count
1 (default)	6	2	4	2
2	4	1	4	1
3	8	4	6	4

Just as you did with Problem 1, modify the FuncUnitConfig.py file for each configuration, rebuild gem5, and run the Dijkstra benchmark on the modified file for each of the 3 configurations shown in the table above. The target directories have already been created with the first command of this document.

Once you are done with Problem 1 and Problem 2, don't forget to set modified parameters back to their default configurations (for next projects).

Project Deliverables

1. Your project submission should contain the following:
    a. A parent directory: Project1-Submission.
    b. Two directories inside Project1-Submission: Problem1 and Problem2.
    c. Problem1 and Problem2 directories should each contain three directories: dijkstraConf1, dijkstra-Conf2, and dijkstra-Conf3.
    d. Each of these 6 configuration directories should contain config.ini, config.json, and stats.txt files (generated by gem5). The directory structure should be as following:

```
Project1-Submission
     |
     +---Problem1
     |   +---dijkstra-Conf1
     |   |   +---config.ini
     |   |   +---config.json
     |   |   +---stats.txt
     |   |
     |   +---dijkstra-Conf2
     |   |   +---config.ini
     |   |   +---config.json
     |   |   +---stats.txt
     |   |
     |   +---dijkstra-Conf3
     |       +---config.ini
     |       +---config.json
     |       +---stats.txt
     |
     +---Problem2
         +---dijkstra-Conf1
         |   +---config.ini
         |   +---config.json
         |   +---stats.txt
         |
         +---dijkstra-Conf2
         |   +---config.ini
         |   +---config.json
         |   +---stats.txt
         |
```

```
+---dijkstra-Conf3
    +---config.ini
    +---config.json
    +---stats.txt
```

Submission Instructions

Submit to Canvas a single zip file containing all the files from the project. You should be zipping just the folder "Project1-Submission" and then submitting the resulting zip file entitled "LastName-Project1-Submission.zip" where 'LastName' is the last name of the group member submitting the file to Canvas. **Only one group member should make the submission through Canvas**. All team members in the group will receive the same score. Please follow the naming conventions correctly so that the auto grader will correctly grade your assignment.

If you are using the terminal on GCP, then use the command "`zip LastName-Project1-Submission.zip -r Project1-Submission`" to obtain the correct zip file output. On GCP, you need to download it to your local machine for submission.

Rubric

1. **Problem 1 [5 points]**
    a. Correct configuration from the table from config.ini file. [2 points]
    b. Correct stats in the stats.txt file [3 points]
2. **Problem 2 [5 points]**
    a. Correct configuration from the table from config.ini file. [2 points]
    b. Correct stats in the stats.txt file [3 points]