

# CSE 560M Computer Systems Architecture I

## Assignment 5, due Friday, Dec. 6, 2024

In this lab assignment we will explore using the `gem5` Out of Order (OoO) CPU model to look at factors within a single core processor and see how their variation can effect performance. You will be asked to sweep parameters that will change the sizes of a few microarchitecture parameters and explain how and why they affect performance.

1. To vary the parameters in the simulation we will be using the option parser class to pass parameter values into the setup options of our CPU. This makes it easy to set parameters on your simulation items from the command line. To make things straightforward we have provided a file to add this functionality to your simulation. To start, copy over the files into your own working directory (e.g. `hw5`).

```
cp -r $GEM5/hw5_2023/* .
```

Take a moment to review the files. The `hw5config.py` file is similar to the previous setup files you've created, however here we have added the `opts` to the options parser that was used in the previous lab to set parameters for the cache. In this assignment, the caches have already been set up for you. They can be set by using the switches

```
--caches and --l2cache
```

The input binary is specified with the `--c` switch and specifying the binary path. For example,

```
--c /absolute/path/to/binary
```

We will also be using an Out of Order (OoO) processor, and this parameter is set by the following switch:

```
--cpu-type="DerivO3CPU"
```

For this lab the options that you will be changing are directly related to the CPU. The `opts.py` file contains a skeleton of the `opts` parser that will be used to set these parameters without having to change the source code every run.

You'll be sweeping three parameters in this lab: (1) the number of reorder buffer entries, (2) the number of instruction queue entries, and (3) the number of physical integer registers. In the `addOpts` function the parser options will be added, which is an instance of the `OptionParser` class.

When declaring an option for the parser we will use the function `parser.add_option()` as we did in Assignment 2. Use this to add configurable options for the three parameters above. The table below gives the default options for these parameters:

Parameter	Value
num-rob-entries	192
num-iq-entries	64
num-phys-int-regs	256

The other function declared in `opts.py` is the `set_config()` function which receives a list of CPUs and the options which are used to set values. Under the scope of the for loop already declared, set the values of the parameters that you are changing for the given CPU like so:

```
cpu.parameter = options.parameter_name
```

The three parameters that you have to set in the CPU class are `numROBEntries`, `numIQEntries`, and `numPhysIntRegs`.

- Now that the options are set properly we can sweep across parameters and take measurements! Similar to previous assignments we are going to run `daxpy_arm_big` program (which is `daxpy` with a larger input size) using the ARM ISA and save the simulator output files as you vary the parameters of the options you've created. Using a **power of 4** step size (as opposed to a linear one, each entry is approximately quadruple of the previous one) to perform a measurement of each permutation of configurations as you sweep: (1) the number of physical integer registers from 64 to 1024, (2) the number of instruction queue entries from 4 to 256, and (3) the number of reorder buffer entries from 4 to 256. This should result in a total of around 50 runs that take much longer than previous simulator runs so it is **HIGHLY** suggested that you create a script to automate the process. The simulation can be started by running the following command:

```
$GEM5/build/ARM/gem5.opt hw5config.py -c $GEM5/./test_progs/daxpy/daxpy_arm_big
--cpu-type="Deriv03CPU" --caches --l2cache
--num-phys-int-regs=$INT_VALUE
--num-rob-entries=$INT_VALUE
--num-iq-entries=$INT_VALUE
```

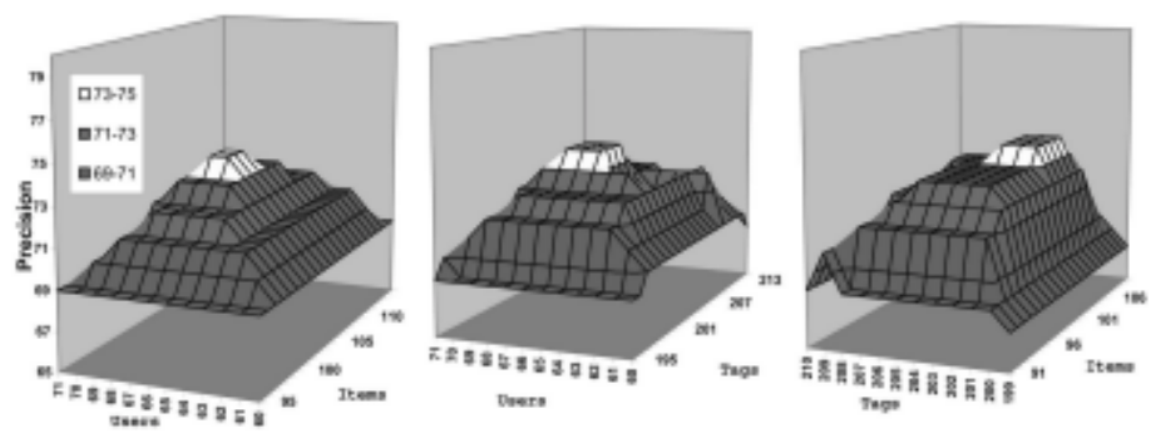
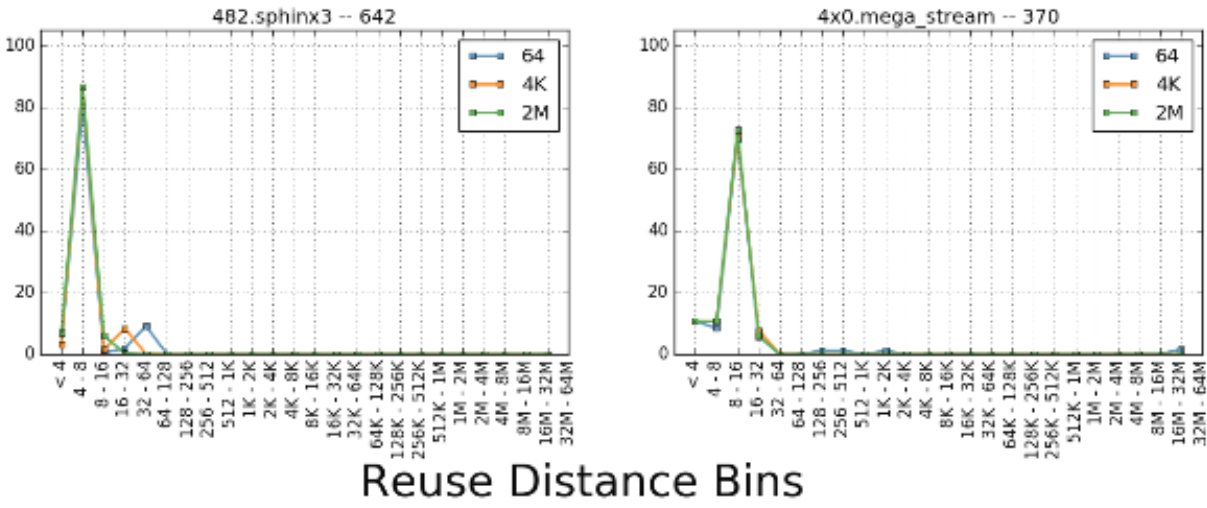
Please note the above is a one line command and you may have to change the `daxpy` path depending on your location. Also, make sure you don't overwrite your data each time you run as the default `m5out` folder is replaced each time (add `--outdir` to the command above). The following table shows the values that you should record for each run.

Parameters to record.
Number of seconds simulated
Number of cycles rename is blocking
Number of integer rename lookups
Instruction Issue Rate
Idle cycles from register renaming
Number of times rename has blocked due to ROB full
Cache miss rates

What other parameters could offer insight into the workings of this out-of-order processor?

Same as last lab, pick two parameters from those recorded above and plot the results from them based on the data from the various configurations of our CPU to show some kind of meaningful conclusion. Give a brief summary of what you think is happening and what the data is telling you.

Note: this is a 3-dimensional space as you are varying the parameters so a typical line/bar graph won't necessarily be the *best* way to represent the data. Try to think of something creative! Below are some examples:



**Fig. 15. Precision of Tensor Reduction as dimensions of core tensor vary for BibSonomy data set.**

