CS425: Computer Systems Architecture

Simulation Assignment 2
Assignment Date: 12/12/2025
Due Date: 22/12/2025 - 23:59

The Limits of ILP

The purpose of this assignment is to introduce you to architectural exploration using simulation and help you familiarise with some of the details of Out-of-Order CPU by exploring some of its design points and potential performance improvements. You have to run simulations with different parameters (the actual list of parameters can be found below) and see the impact of alternative design choices in terms of Instructions Per Cycle (IPC). Note at this point, that you are free to refer to other metrics, as well.

For the simulations you will use the gem5 simulator (www.gem5.org). Although the assignment does not require prior experience with gem5, it is recommended that you take some time to follow the gem5 tutorial:
www.gem5.org/documentation/learning_gem5/introduction/

Simulation Assignment 2 – Code Repository

For the infrastructure of this assignment, we have created personal git repositories on the Department's gitlab server for each student enrolled in the course. Your personal repository is in the form:

https://gitlab-csd.datacenter.uoc.gr/hy425_2025f/sim2_submits/sim2-csdXYZWA

Get a clone of your repository with `git clone` using your username (csdXYZWA@csd.uoc.gr). Access is only available via the University network and the VPN. *(Do not fork the repo – Work directly on the cloned git repo)*.

The repo includes the following:

- `README.md` : Quick installation and run instructions
- `benchmarks/` : contains five precompiled benchmarks for you to run
- `config/` : gem5 configuration of an out-of-order core based on the O3CPU model
- `setup.sh` : Shell script for setting environment variables and PATHs
- `runall.sh` : Shell script for quickly running a gem5 simulation with the given configuration

Study the folders and the files to get an idea of the repo and feel free to create your own scripts as you see fit.

In this assignment you will compile gem5 to target the **x86 ISA**. The build requirements can be found here: https://www.gem5.org/documentation/general_docs/building

Check the **`README.md`** file which provides a quick installation and running guide. The gem5 compilation step takes time so complete it as soon as possible!

(You can use your previous gem5 installation from Simulation Assignment 1 but you still need to compile for x86).

Assignment Tasks

First read the details of the O3CPU model to understand the architecture that is documented here: https://www.gem5.org/documentation/general_docs/cpu_models/O3CPU

To run a batch of five (5) benchmarks with the simulator you have to use the following command inside the `sim2-username` directory:

```
./runall.sh [List of arguments]
```

This command will run all the benchmarks while simulating a setup with an Out-Of-Order CPU (O3CPU), a good performing branch predictor (TournamentBP) and reasonable settings for caches and other components. You are encouraged to examine thoroughly the `runall.sh` to figure out the different arguments that you are allowed to change. Note that the `--Insts` *[Instruction Count]* argument means that for each benchmark, gem5 simulates *[Instruction Count]* instructions. You can change this parameter during early testing but for generating final numbers you should have at-least 20 million instructions. If you set a very large number of instructions the simulation time will increase dramatically. Ensure that your comparisons are based on the same instruction count between different benchmarks and/or settings to extract uniform and fair results. Also, ensure that this parameter is large enough to extract meaningful results. An estimated number would be between 20-50 million instructions and this should take 2-5 minutes per benchmark. In any case, this is another reason to begin the assignment as soon as possible!

Explore OoO CPU Design Points

Your main task is to explore the given design space of an OoO CPU by simulating benchmarks and observe mainly the corresponding IPC values. Study the slides presented in the class and the paper from David W. Wall: “Limits of Instruction-Level Parallelism” - you can find it on the website of the course.

The list of the specified design parameters is the following:

- ROB_ENTRIES: Number of Reorder Buffer entries
    - Range: 64, 128, 256, 384, 512, 640
    - Default: 64
    - Today: up-to 512

- FUINT_NUM: Number of Simple Int Functional Units (ADD etc.)
    - Range: 2, 4, 6, 8
    - Default: 2
    - Today: up-to 6

- FULS_NUM: Number of Load/Store Functional Units (same value for both)
    - Range: 1, 2, 3
    - Default: 1
    - Today: up-to 2

- LSQ_ENTRIES: Number of LoadQueue/StoreQueue entries (same value for both)
    - Range: 16, 32, 64, 128, 160
    - Default: 16
    - Today: up-to 128

Provide your report in the `answers.md` file and use tables and/or graphs for the following:

1. Explore and report IPC improvements using the ranges of these parameters compared to the default values. One suggested plan is to tweak each parameter while having the others set to default. Once you pick the highest performing value for each parameter, you can use this value to continue the exploration. However, you are free to suggest your own exploration plan!

2. Find the combination of parameter values with the minimum product MiP (i.e. MiP = ROB_ENTRIES * FUINT_NUM * FULS_NUM * LSQ_ENTRIES) that maximises IPC using only realistic ranges. However, you are encouraged to report any performance gain with non-realistic parameters (if any).

3. Is there a saturation point for each benchmark's IPC? Explain your answers.

It is important to get familiar with the generated m5out directories under the `logs/benchmarkN` directories after each run. These directories include files showing the simulated configuration and their statistical information.

Bonus Question

Use the generated statistics (e.g. instruction mix) to infer information and/or other characteristics about the behaviour of each benchmark. Does your answer to question 3 also help to infer such characteristics? Explain your approach and observations.

Assignment Delivery

Commit and push the following files on your designated gitlab repo:

- `answers.md` with your report and comments
- All the `logs` (`benchmark1, benchmark2`, …) folders/subfolders generated by your simulations
- Any other files or scripts you consider useful

Indicative git commands:

- git status (see the changes)
- git config user.email csdXYZW@csd.uoc.gr
- git config user.name "Name Surname"
- git add answers.md logs* (add all related folders)
- git commit -m "sim2 delivery" (commit and message)
- git push (use your username and password)

Make sure that the push was successfully done and that your files have been uploaded on the gitlab repo.