DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

ADVANCED COMPUTER ARCHITECTURES

Project 2

Abstract: The second lab consists in the analysis of in-order and out-of-order processor architectures and their connection with the memory subsystem. For this, students are expected to develop simple programs to stress the processor architectures and to determine the influence of some design parameters.

1 INTRODUCTION

For the development of this work, students should use the gem5 processor simulator (http://gem5.org/), which is modular platform for computer-system architecture research, from the microarchitecture to the system level.

Students can opt by installing gem5 in their own computers, or rely on a previously installed version, available at:

Machine name: elsa.inesc-id.pt
Username: aac2018
Password: toKLGOZ9
Connection protocol: SSH
Folder: /homelocal/aac/gem5

To help the development of the work, students are advisor to follow the gem5 tutorial book (http://learning.gem5.org/book/), by reading the Learning gem5 webpage (http://learning.gem5.org/) or to consult online other documentation.

For the purpose of this work, student should rely on the x86 instruction set architecture and on two core microarchitecture models: `MinorCPU` (in-order) and `DerivO3CPU` (out-of-order).

To develop test programs, students should rely on C code and compile using Gcc with flags `-DUNIX -static`, e.g.,

```
gcc -DUNIX -static sample_source_code.c -o sample_binary
```

2 WORK PLAN

The objective of this work is to highlight the differences between in-order and out-of-order processors. For this, students should develop small benchmark programs that:

Program A: Maximizes the differences (in execution time) between an in-order and an out-of-order processor
Program B: Minimizes the differences (in execution time) between an in-order and an out-of-order processor

The difference in execution time should be measured by the speed-up of an out-of-order processor vs an in-order one (reference). Hence, while for program A students should focus on maximizing the speed-up of the out-of-order processor, for program B students should focus on minimizing the speed-up.

To accomplish this objective use the following steps:

1. Create a folder for your group project, e.g., under `/homelocal/aac/groupXY`.
2. Create a small basic benchmark `simple_bench.c` (e.g., a hello world program) and compile it:

   ```
   gcc -DUNIX -static simple_bench.c -o simple_bench
   ```

3. Copy the base configuration files from `gem5/configs/learning_gem5/part1/*` to `/homelocal/aac/groupXY`, and update the path on each copied file. For this change the lines:

```
# Add the common scripts to our path
m5.util.addToPath('../../')
```

to

```
# Add the common scripts to our path
m5.util.addToPath('/homelocal/aac/gem5')
m5.util.addToPath('/homelocal/aac/gem5/configs')
```

4. Create two copies of the two_level.py configuration file, one for the in-order processor (e.g., `two_level_inorder.py`) and another out-of-order (e.g.., `two_level_outoforder.py`). Adapt each configuration file so that the first uses an in-order processor (`MinorCPU`), the other the out-of-order (`DerivO3CPU`). Also adjust the binary that you want to run, in order to point to your benchmark.

5. Run the benchmark on both scripts, e.g.,

```
cd /homelocal/aac/groupXY/
../../gem5/build/X86/gem5.opt two_level_inorder.py
cp m5out/stats.txt m5out_inorder.txt
../../gem5/build/X86/gem5.opt two_level_outoforder.py
cp m5out/stats.txt m5out_outoforder.txt
```

6. Evaluate the difference in performance by comparing the file statistics.

7. Change the benchmark in order to solve the handout.

3 PROJECT DUE DATE AND REPORT FORMAT

The project should be completed until May 6, with the code and an up to 6 page report being be submitted online (via Fenix). The report should follow the template for the IEEE Computer society journals:

https://www.ieee.org/publications_standards/publications/authors/author_templates.html

The report should include:

- an abstract (single paragraph, 100-200 words) explaining the objective of the work;
- an introduction section explaining what is the characteristic of both cores that you are trying to exploit in order to solve the problem;
- a solution section describing the structure of your programs – you may use a figure, diagram or pseudo-code to help explaining the idea, but not the original C code;
- an experimental results section showing the obtained results (support your text with tables or figures); by analysing the two models, try to explain the obtained speed-ups;
- a conclusions section.