ECSE 425 Final Project
Proposal for the Bonus Optimisation

Amine Mallek
ECSE Department
McGill University
Montreal, Canada
amine.mallek@mail.mcgill.ca

Bowen Cui
ECSE Department
McGill University
Montreal, Canada
bowen.cui@mail.mcgill.ca

Kaan Gure
ECSE Department
McGill University
Montreal, Canada
kaan.gure@mail.mcgill.ca

James Willems
ECSE Department
McGill University
Montreal, Canada
james.willems@mail.mcgill.ca

Krishna Aroomoogon
ECSE Department
McGill University
Montreal, Canada
krishna.aroomoogon@mail.mcgill.ca

***Abstract*—This proposal outlines the general design of the proposed cache task for the optimisation of the final project.**

I. Introduction

For our optimisation task, we have decided to implement a cache for out pipelined processor. The cache consists of two 4096-bit L1 caches for instruction and data respectively, and an arbiter to arbitrate the memory operations of the two L1 caches to the memory.

The L1 cache part of this task will be similar to the cache task from the mini-project 2. Since the performance of our processor will be assessed with simulations, multi-level caches would not be beneficial to the performance of our optimisation.

A. Arbiter

Since there is only one data bus in the unified memory, an arbiter will have to be implemented to schedule the memory accesses when there are two memory accesses from the two caches at the same time. The priority of the two requests is hard-coded in the simple arbiter, and the request from the data cache will always be accommodated first. Another function of the arbiter is that when a cache is fetching data from the memory, it will signal the pipeline to stall and wait for the memory access to finish.

II. Diagram

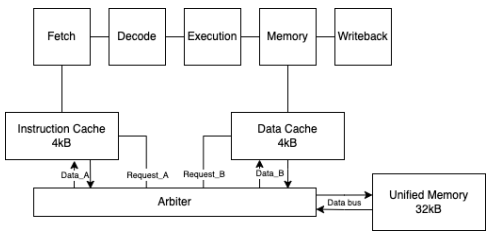

Fig. 1. The structure diagram of the cache subsystem.

III. Performance Profiling

To measure and compare the performance gain from the cache, we will measure the performance of the processor by running the same program in three different configurations.

The first configuration will be the bare processor without cache used, and the instruction memory and the data memory will be separate. In the second configuration, two 4096-bit L1 caches will be used for both instruction and data. In the third configuration, the sizes of the two caches will be increased to 8192 bits.

The code to generate the Fibonacci sequence will be used to measure the performance of the caches. The Fibonacci sequence is chosen because the speed on execution of the algorithm is dependent on fast memory accesses, and its pattern of memory accesses can fully take advantage of a data cache. The size of sequence generated will be set to a high value ($2000 \sim 3000$) to make sure that the performance difference between the three scenarios can be observed.