CSCE 312: Computer Organization - Final Project

Texas A&M University, Spring 2020

Date: 05/04/2020

Student 1 name: Ryan Mendoza **Student 2 name:**	**Student 1 UIN: 627003909** **Student 2 UIN:**

Introduction

This report discusses the design and implementation of the cache simulator. It will start with the design of the simulator before describing the step-by-step process of the implementation of that design. Afterwards, a review of the simulator, an analysis of the experience in creating the simulator, and the possible improvements to the simulator is included.

Design and implementation

In the simulator, a Python dictionary is used to simulate the RAM while a series of lists and sub-lists are used to simulate the cache. According to the given addresses and replacement, write hit, and write miss policies by the user, the cache will retrieve and store values from the RAM.

Step 1. The RAM's design was implemented first using a Python dictionary where the key is the memory address while the value was the value stored at that memory address. This is created and filled up by reading each line of the input file a generating a new dictionary item for each line.

Step 2. The cache configuration menu was implemented next, prompting the user to input values while storing those values for internal calculations to define the cache's behavior. Values such as number of sets, number of blocks, and replacement policy are stored and defined here.

Step 3. The cache was then implemented to follow the user inputs given in the cache menu. Specifically, cache size, block size, and associativity were used to define the structure of the cache and was filled with zeroes to emulate a cold cache upon creation.

Step 4. The cache menu was then implemented, following a structure that splits the user's input into a space separated list where the first value is expected to follow one of the command prompts and the rest are used as parameters for that command's behavior. All commands were created in order of its menu listing except for "quit" which was created first. This was done so the menu can serve as a checklist for implementation while also having a functional "quit" command in the case of any errors. Due to the nature of the command menu, the implementation of one command often used much of the same logic as the previous command, making implementation of each command simpler over time. Within each command, the structure of the cache is considered, in addition to the user's given policies.

Conclusion

Overall, the simulator successfully simulates the structure and behavior of the cache according to the user's inputs. As an experience, the creation of this simulator was moderately stressful due to the lack of a partner, resulting in design flaws that may have been considered with the ideas, collaboration, time, and knowledge of another individual. Such design flaws, and possible improvements, includes the usage of classes and methods since the cache's information and structure is strictly within the cache's series of lists and sub-lists with no outside information while many commands use copy-and-pasted code from another command. The usage of classes and methods would result in the code being presented in a much clearer way since the current code design heavily relies on my personal knowledge of its structure, making it difficult for others to understand. Despite these internal design flaws, the simulator was a success.