Cache Final Assignment

Computer Organisation (CSE112)

Vaibhav Saxena
vaibhav19342@iiitd.ac.in

Introduction

The project was to implement virtual primary and secondary cache for storing most recently used data. According to Wikipedia.org, in computing, a cache is a hardware or software component that stores data so that future requests for that data can be served faste. There are different kinds of mappings used for cache. We use the direct, fully associative and set associative mappings in this project.

Primary Input

1. **Size of one Block**: that defines how many data items a block can store
2. **Number of Primary Cache lines**: defines the size of the primary cache; the size of the secondary cache is taken as double of this
3. **Number of total Blocks in Main memory**: the total number of blocks that can be written

All of these are integer values and represent powers of 2, which give the actual value.

Output

The program outputs many things including some Menus to select which type of mapping is wanted, or to choose whether a read or write instruction is to be given. It takes appropriate inputs, checks them for errors and displays the suitable outputs. Any write statement outputs both the caches and any read statement outputs all the processes that occurred and displays the final caches. Every Menu is repetitive and asks whether the user would like to go back to the Menu, when they reach the end of the instruction.

Functioning

1. Input

The program takes the primary inputs and also some other inputs, like the number of sets in set associative mapping. When taking an instruction, it knows whether it is taking a read instruction or write instruction, with the help of the menu. So it knows what all inputs it needs. The Block tags and offsets are taken as integers instead of binary numbers to provide ease to the user. Each Block data is taken as a single string without any spaces.

2. Menu for Type of Mapping

The user gets the option to choose which type of mapping he/she wants to apply to the caches. This menu is repetitive and provides an option to go back to the menu at the end of instructions.

3. Menu for Type of Instruction

The user must choose what type of instruction he/she wants to give before every instruction- whether it is a write or a read instruction. This menu is repetitive and provides an option to go back to the menu at the end of instructions.

4. Algorithms and Output

Based on the type of mapping that was selected by the user, the program writes, reads and replaces the data in the caches appropriately. For the associative part, there are many replacement algorithms available but we have used the FIFO (First In First Out) algorithm in this program. The program also implements the Main Memory completely. The output of any instruction gives the full details of all the processes that took place and the final values in the caches.

Functions and Classes used

1. **displaycache()** and **displaycachesetasso():**

   Displays the entire cache with set and cache line numbering

2. **Direct()**, **Asso()** and **SetAsso():**

   Contain the main algorithms of the different mappings

3. **Input():**

   Used in all three algorithms to take the input instruction i.e. read or write

4. **Class Block:**

   To store one Block of data with an integer tag and string data items

5. **Class Instruct:**

   To store one instruction - read or write, with the block tag, offset or data

Errors handled

1. **Wrong input format**: shows an error if the input is not in the format that is specified, this ensures that the parameters are in the proper format and no runtime errors occur
2. **Input out of range**: shows an error if any input parameter is not in the specified range, this ensures that the parameters are properly in range and no logical errors occur