Computer Architecture - CS2323

Lab-6 (Cache Miss Simulator)

Rishitha Surineni
CS22BTECH11050

November 23, 2023

1 Introduction

The aim of this assignment is to simulate a cache with user specified configuration and check which accesses would result in a cache hit and which would result in a cache miss.

2 Methodology

User will be asked to provide two files.The first one would contain the following attributes.
SIZE-OF-CACHE (number)
BLOCK-SIZE (number)
ASSOCIATIVITY (number)
REPLACEMENT-POLICY (FIFO or LRU or RANDOM)
WRITEBACK-POLICY (WB or WT)
And the second one will be the file containing 32-bit hexadecimal (starting with 0x) addresses of the accesses along with mode (Read or Write).
After reading the configuration file,
The number of sets and number of bits for offset,tag,setindex are calculated.
Cache is created with the given parameters.It is implemented using an array of structures (of size equal to number of sets) where each structure represents a set in the cache.Each of these sets stores the values of tags and valid bits based on associativity.
Then the accesses file is read and for each address,
It is converted into a binary number and stored in an interger array.
The Cache and this binary number are sent into a function along with other parameters like writepolicy,access mode,associativity and number of bits for tag,setindex,offset .
For Direct Mapped cache and Set Associative cache
The corresponding set index and tag for the address is calculated and that set of cache is checked if the tag matches with any of the tags already present in the cache. Valid bits are also used so that an wrong hits don't happen.
For Fully Associative Cache each structure in Cache array contains only one tag.Each structure's tag is checked with present tag.If there is a match then it is a hit.

3 If there is a hit

If the replacement polity is LRU then this address would be the recently used one. So the tags in the set from the one which gave hit are moved one index lower and this tag is placed at the end of the present tags.

4 If there is a miss

If the access mode is write and write policy is Write Through Without Allocate then nothing has to be done as the data is written directly into main memory.
If the number of tags in the set are less than associativity then this tag is added to the set.
If the set is full then one of tags is to be replaced,this is done based on Replacement policy
If Replacement policy is FIFO
The tag which is placed first into the Set is to be replaced.
The tag at 0th index in the set is removed and all of the tags next to it are moved one index lower and the new tag is palced at the last index.
If Replacement policy is LRU
Similar to FIFO beacuse we are storing tags here in the order of most recently accessed to least recently accessed(as everytime a hit is encountered that tag is placed at last by shifting other tags)
If Replacement policy is Random
A random index is found usind rand() function and the tag at this index is replaced with current tag.

5 Validation

First I have implemented Read access for a direct mapped cache where the associativity is 1.
Here there is no need of replacement policy as each set can store only one tag and data.
After this I have implemented set associative cache and checked the cases when replacement is not needed.
Then I implemented FIFO policy and checked my code with that configuration.
Then I tested with LRU by making changes when a hit is encountered and keeping the miss condition same as the one done in FIFO.
I then checked for write access and modified the code accordingly.
Then I implemented Fully Associative cache and Random replacement policy and run the code using different cache configurations and multiple testcases.
I verified the results by checking the outputs and analyzing them.