***DOCUMENTATION***

***CACHE – PROJECT***

**Introduction to the Objective of the Program:**

The objective of the program is to create a cache structure with **Direct Mapping**, **Associative Mapping** and **Set-Associative Mapping.** The cache Memory is easy to access and is time efficient. So, the computer looks for the address required in the **cache Memory** first and then if the Address is missing then it searches the address in the **Main Memory**. This increases the efficiency and the time to access data decreases.

**Functions used**

1. **to\_bin ():**

This function is used to take hexadecimal addresses in form of string and then return the binary form of the same. It takes two string in form of input one the hexadecimal address in the empty string.

Example if to\_bin (FFFF) is called, it returns 1111111111111111 as an output

1. **to\_dec ():**

This function is used to calculate the decimal form of a string entered in binary form. It takes the binary form a number in input and returns the decimal form of the number.

Example if to\_dec (111) is called it returns 7 as an output.

1. **power\_2 ():**

This function is used to calculate the power of 2 in the particular number. It takes an integer as input and returns the power of 2 in that number.

Example if power\_2 (16) Is called then it would return 4 as an output.

1. **Search ()**

This function is used to search whether the particular tag Is present in the cache structure or not.0

Some of the inbuilt functions such as **int ()**, **str ()** and functions related to the **Module Math** are used in this program.

**ASSUMPTIONS:**

* It is assumed that the address is of **16 Bit.**
* The input of address of each word in taken in **Hexadecimal Base.**
* All inputs are assumed to be in the **power of two** starting from 2.
* It is assumed that there is **no main memory** i.e. it is a stand - alone cache
* It is assumed that data to be stored is always greater than or equal to 0. **Data>=0.**
* 1 word = 1Byte

**INPUT FORMAT**

The input format is the same as described in the level1 cache with two types of statement “READ” and “WRITE” and the purpose of the statements are also the same.

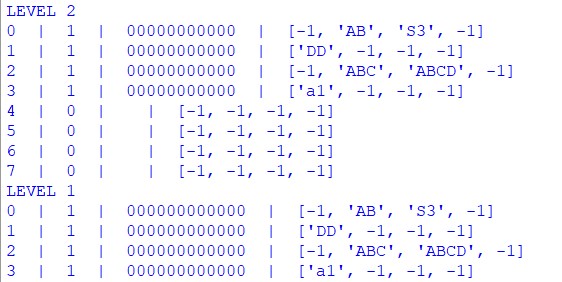
**OUTPUT FORMAT**

The output format is also the same with data and the replacement and Miss statements.

**THE WORKING OF THE PROGRAM:**

1. **Direct Mapping:**

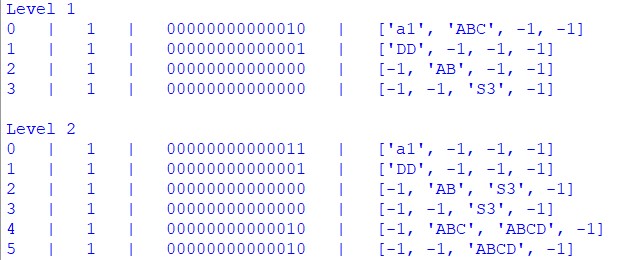
This mapping follows the same procedure for the tag bits, the word bit, and the offset bit with the difference being in the coordination between the 2 levels of the cache. When the WRITE statement is encountered by the program, then if checks which of the cache is empty. If both are empty, then the program enters the data in the both the levels in the respective positions of the cache. When one of them is empty then it writes the data in the empty cache structure. When both of them are full, then the cache writes the data in both of them at their respective positions. The output shows the cache structure of both the levels with the data and tags present in them. When the “READ” statement is encountered then the cache first checks in the level1 and then the level 2 and gives the output as hit if it is present in either of them, otherwise gives an Address missing message. In case where the address is found in the level 2 the program copies it into the level 1.



1. **Fully Associative Mapping:**

This type of mapping follows the same procedure as the level 1 cache in terms of the tag bits. The Difference here lies in the write and read statement of the two levels of the cache. When the write Statement is encountered by the program, it adds the data and the tag in the both the level 1 and the level 2 of the cache. If the all the lines of the level 1 cache get filled, then the data continues to be added in the level 2 cache and when the level 2 of cache is filled, then the program uses the FIFO method to the replace in the level 2 of the cache.

When the read statement is encountered then the program first checks the level 1 of the cache and then the level 2. If the data is found then it is considered as a hit whereas in the other scenario it is a Miss. And if the data is found in the level 2 cache, the data is copied in the cache of level 1. The output finally shows both level of cache



**ERRORS ENCOUNTERED:**

The errors that can be encountered and the reasons for them are the same as described in the level 1 cache.