Name- Aakash Yadav Section-B

Roll no-2019222 Subject- CO

End sem Project

In this project I have written code that allows loading into cache and searching cache.

Language used:- JAVA

Working of Program

Firstly I am asking the user which type of mapping he/she wants to perform.

There are Three options:-

1) Direct mapping
2) Associative memory
3) n-way set associative memory where n is a power of 2.

Direct Mapping

Input format:

```
2
Enter cache size in KB for cache level1
0.25
Block size
32
Enter 1 for write and and 2 for read and 3 print tag array and block data array
1
Enter address
11111101010101011010101010101111
input data
7
1 for more use otherwise 0
1
Enter 1 for write and and 2 for read and 3 print tag array and block data array
3
tag array
0 0 0 0 0 11111101010101011010101010 0 0
data array
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
1 for more use otherwise 0
```

Direct mapping works on the basis of index. As shown in image above the last 3 bit will indicate the block offset and from last 4th 5th and 6th bit will represent the index for tag array and storing tag in tag array according to index and saving data in data array according to block offset(index at which data and tag are stored in data array and tag array are same) .

Printing tag array array and data array also.

Format of index for tag:- 000 means storing tag at index0 of tag array.

Condition:- At the time of inserting tag if a different tag is present at the same index then initialize all elements of that row of data array to zero/null then replace tag in tag array and data in data array according to its block offset.

If same tag is present at the same index than change data in data array according to its block offset.

Fully Associative Mapping:-

Input format:

```
1
Enter cache size in KB
0.25
Block size
32
Enter 1 for write and and 2 for read and 3 print tag array and block data array
1
Enter address
11111101010101011010101010101111
input data
1
Press 1 for more use otherwise 0
1
Enter 1 for write and and 2 for read and 3 print tag array and block data array
3
tag array
0 0 0 0 11111101010101011010101010101 0 0 0
data array
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Press 1 for more use otherwise 0
```

In associative mapping (example as shown in figure) the last 3 bit will indicate block offset and 29 bit from the beginning will represent a tag(this is according to example).Printing tag array array and data array also.

In this, inserting tags in tag array randomly and saving data in data array according to block offset(index at which data and tag are stored in tag array and data array are same) .

If the same tag is present at the same index then change data in the data array according to its block offset. I am doing FIFO replacement.

Set Associative Mapping:-

Input format:- upload image

```
3
Enter cache size in KB
1
Block size
64
Enter value of n
4
Enter 1 for write and and 2 for read and 3 print tag array and block data array
1
Enter address
11111101010101011010101010101111
input data
7
1 for more use otherwise 0
1
Enter 1 for write and and 2 for read and 3 print tag array and block data array
3
tag array
0 0 0 0 0 0 0 0 0 11111101010101011010101010 0 0 0 0 0 0
data array
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
Enter 1 for write and and 2 for read and 3 print tag array and block data array
1
Enter address
11111101010101011010101010101111
input data
7
1 for more use otherwise 0
1
Enter 1 for write and and 2 for read and 3 print tag array and block data array
3
tag array
0 0 0 0 0 0 0 0 0 11111101010101011010101010 0 0 0 0 0 0
data array
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 for more use otherwise 0
```

In this mapping (example as shown in figure) the last 4 bit will indicate block offset and from the last 5th and 6th bit will represent the set in the tag array. In a particular set I am inserting tag in tag array randomly and saving data in data array according to block offset(index at which data and tag are stored in tag array and data array are same) . FIFO replacement is being used in a particular set. Printing tag array array and data array also.

If the same tag is present at the same index then change data in the data array according to its block offset.

2 level Direct Mapping:-

Input format:- upload image

```
4
Enter cache size in KB
0.25
Block size
32
Enter 1 for write and and 2 for read and 3 print tag array and block data array
1
Enter address
11111101010101011010101010101111
input data
1
1 for more use otherwise 0
1
Enter 1 for write and and 2 for read and 3 print tag array and block data array
3
cache level1
tag array
0 0 0 0 0 11111101010101011010101010 0 0
```

```
data array
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0


cache level2
tag array
0 0 0 0 0 11111101010101101010101 0 0 0 0 0 0 0 0 0
data array
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

For level1 cache:-

This mapping is performed on the basis of index as shown in example. The last 3 bit will indicate the block offset and from the last 4th, 5th and 6th bits will represent the index for tag array and storing tag in tag array according to index and saving data in data array according to block offset(index at which data and tag are storing in tag array and data array are same) .

Format of index for tag:- 000 means storing tag at index0 of tagarray.

For level2 cache:-

This mapping is performed on the basis of index as shown in example. The last 3 bits will indicate the block offset and from the last 4th, 5th, 6th and 7th bits will represent theindex for tag array and storing tag in tag array according to index and saving data in data array according to block offset(index at which data and tag are storing in tag array and data array are same) .

Format of index for tag:- 0000 means storing tag at index0 of tagarray.

Size of level2 cache is double of level1 cache.

Condition:- At the time of inserting tag if a different tag is present at same index then initializing all elements of that row of data array to zero/null then replacing data in data array according to its block offset.

If the same tag is present at the same index then change data in the data array according to its block offset. Data is changing at both levels.

Reading:-If the block we are searching is present in level1 cache than cache hit otherwise we will search in level2 cache and if block is present in level2 cache than copy that block and paste it in level1 according to 4th, 5th and 6th bit of the address and if block is not present than cache miss. Printing tag array array and data array also.

2 level Fully Associative Mapping:-

Input format:- upload image

```
5
Enter cache size in KB for cache level1
0.25
Block size
32
Enter 1 for write and and 2 for read and 3 print tag array and block data array
1
Enter address
11111101010101011010101010101111
input data
7
1 for more input otherwise 0
1
Enter 1 for write and and 2 for read and 3 print tag array and block data array
3
cache level1
tag array
0 0 0 0 11111101010101011010101010101 0 0 0
```

```
data array
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0


cache level2
tag array
0 0 0 0 0 0 0 11111101010101011010101010101 0 0 0 0 0 0 0 0
data array
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

1level:-In this mapping (example as shown in figure) last 3 bits will indicate the block offset and 29 bits from the beginning will represent a tag.In this I am inserting tags in tag array randomly and saving data in data array according to block offset(index at which data and tag are storing in tag array and data array are same) .I am doing FIFO replacement.

2level:-In this I am mapping on the basis of index as shown in example last 3 bits will indicate my block offset and 29 bit from the beginning will represent a tag.In this I am inserting tags in tag array randomly and saving data in data array according to block offset(index at which data and tag are storing in tag array and data array are same) .I am doing FIFO replacement.

Size of level2 cache is double of level1 cache.

Reading:-If the block we are searching is present in level1 cache then cache hit otherwise we will search in level2 cache and if block is present in level2 cache then copy that block and paste it in level1 with FIFO replacement and if block is not present then cache miss.

Condition:-If the same tag is present at the same index then change data in the data array according to its block offset. Data is changing in both level.

Printing tag array array and data array also.

2 level Set Associative Mapping:-

Input format:- upload image

```
6
Enter cache size in KB
1
Block size
64
Enter value of n
4
Enter 1 for write and and 2 for read and 3 print tag array and block data array
1
Enter address
11111101010101011010101010101111
input data
7
1 for more input otherwise 0
1
```

```
1 for more input otherwise 0
1
Enter 1 for write and and 2 for read and 3 print tag array and block data array
3
cache level1
tag array
0 0 0 0 0 0 0 0 11111101010101011010101010 0 0 0 0 0 0 0
data array
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
cache level 2
tag array
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11111101010101011010101010 0 0 0 0 0 0 0 0 0 0 0 0 0
data array
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 for more input otherwise 0
```

Level1 cache:-In this i am mapping (example as shown in figure) last 4 bits will indicate the block offset and from last 5th and 6th bit will represent my set in the tag array. In particular set I am inserting tag in tag array randomly and saving data in data array according to block offset(index at which data and tag are storing in tag array and data array are the same) . I am doing FIFO replacement in a particular set.

Level2 :-In this i am mapping (example as shown in figure) last 4 bit will indicate my block offset and from last 5,6 bit will represent my set in tag array. In particular set I am inserting tag in tag array randomly and saving data in data array according to block offset(index at which data and tag are storing in tag array and data array are same) . I am doing FIFO replacement in a particular set.

Condition :- size of set in level2 should be double of size of set in level1.

If the same tag is present at the same index then change data in the data array according to its block offset. Data is changing at both levels.

Printing tag array array and data array also.

Constraints:-
Input address should be 32 bit long.
Block size is in bytes.