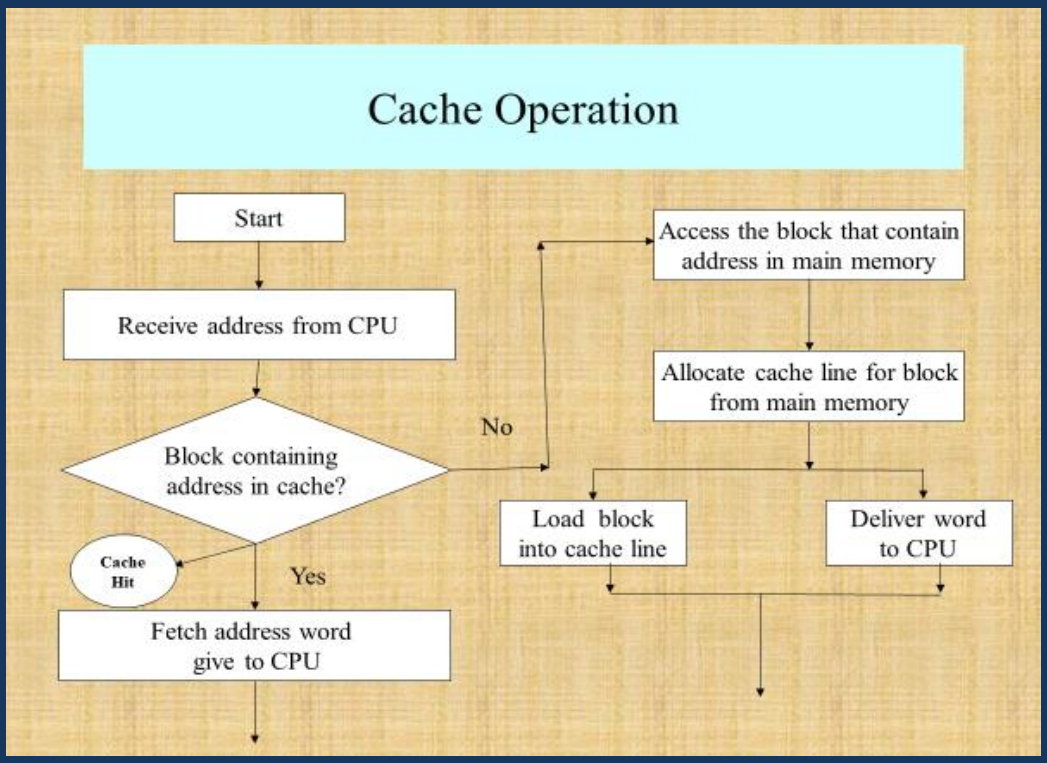
Documentation

*End Semester Assignment – **CSE112***

Manvik Arya (2019432)

- ## Assumptions

  - Main memory has not been considered in this cache simulation
  - Input is given word by word and block-wise insertion is not followed
  - A 16-bit machine has been considered
  - FIFO replacement policy has been used



- ## About Code

1. The code has to be provided with the following inputs:
   - ✓ Total Number of Cache Lines in Cache
   - ✓ Block Size
   - ✓ Memory Size
   - ✓ Required Mapping Technique
   - ✓ Operations to be performed (Read/Write/Display Cache)
   - ✓ Physical Address (if Read/ Write operation is to be performed)

2. There are 3 types of Mapping techniques available:

   ❖ **Direct Mapping**

   In the case of direct mapping, the provided physical address is interpreted into 3 parts i.e. a *tag*, a *cache line number* and a *block offset*.

   If the Read operation is called then, the tag is matched with the respective cache's line tag. If they are matched then, the value present at the given cache line at the given block offset value are printed and a cache hit happens. Otherwise, a cache miss happens.

   When the Write operation is called then, the following formula is used in order to find the cache line at which the data is to be stored and tags are matched to check whether tag replacement policy is necessary or not.

   $$CL = B \bmod C$$

   (where CL = Cache Line, B = Memory Block, C = Total Cache Lines)

   ❖ **Fully Associative Mapping**

   In the case of fully associative mapping, the provided physical address is interpreted into two parts i.e. a *tag* and a *block offset.*

For the Read operation, the tag is matched with all the cache lines and if a match is found then, the value present at that particular block offset position is printed and a cache hit occurs. Otherwise, a message is printed which indicated the required was not found and a cache miss occurs.

For the Write operation, the tag is matched with all the stored tags. If the tag matches, then the data given is stored at that particular tag's cache line. If there is no tag stored then, the given tag is stored and the provided data is inserted. Also, if there are stored tags but, none of them matches with the given tag then, the FIFO replacement policy is followed and a previously stored tag is replaced by the new tag and the data is inserted.

- **Set Associative Mapping**

  For set associative mapping, an integer 'R' is given which is equal to the number of cache lines required in a set. The physical address is interpreted in 3 parts i.e. a *tag*, a set *number*, and a *block offset.*

  For the Read operation, the set number is used to locate the specified set and then, the stored tags of that particular set are matched with the given tag. If there is a match then, the block offset is used to extract the data from the cache memory and a cache hit occurs otherwise, cache miss occurs and a message mentioning that "Address not found" is printed.

  For the Write operation, the set number is used to locate the mentioned set. The stored tags of that set are matched with the given tag. If the tag is matched then, the block offset is used to insert the data into the cache memory. If the tag doesn't match at all then, the given tag is stored in the tag storage and the data given is inserted in the particular tag's cache line. Also, if the given tag doesn't match and space is left in the cache tag storage then, FIFO replacement policy is followed and the earliest tag is replaced with the given new tag.

$$Sn = X \bmod R$$

(where Sn = New Set Number, X = Set Number, R = Number of lines in 1 set)

3. Error handling:

- ✓ An error is thrown if the physical address doesn't match the number of bits used to represent the memory size.

- ✓ An error is thrown if the memory size/ number of cache lines/ block size are not in the powers of 2.