Problem Statement - Implementation of Buffer Cache Simulation

Design and Analysis

Class buffer
{
Data member : blockNumber, processID, status {free ,busy ,delay-write}
buffer *freelistnext *freelistprevious, *hashnext, *hashprevious;
}

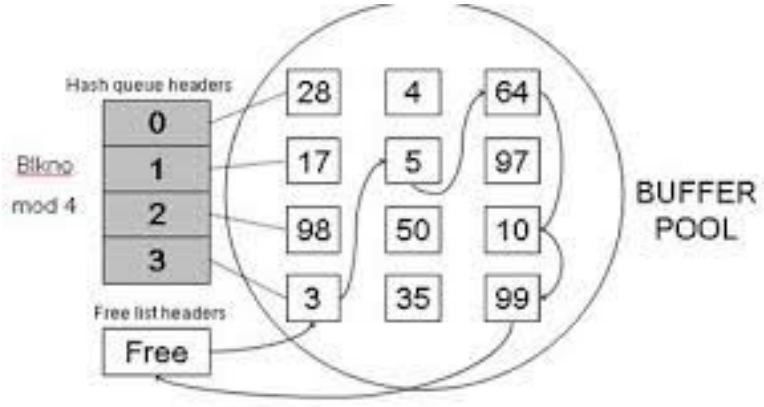
Class Buffer_cache
{
Data members : buffer *tail, *head;
Methods :

1. insertBufferAtTail()
2. insertBufferAtHeadFreelist()
3. removeBufferFromHeadFreelist()
4. removeSpecificBuffer()
5. isEmpty()
6. searchBuffer()
7. printHash()

}

Note:

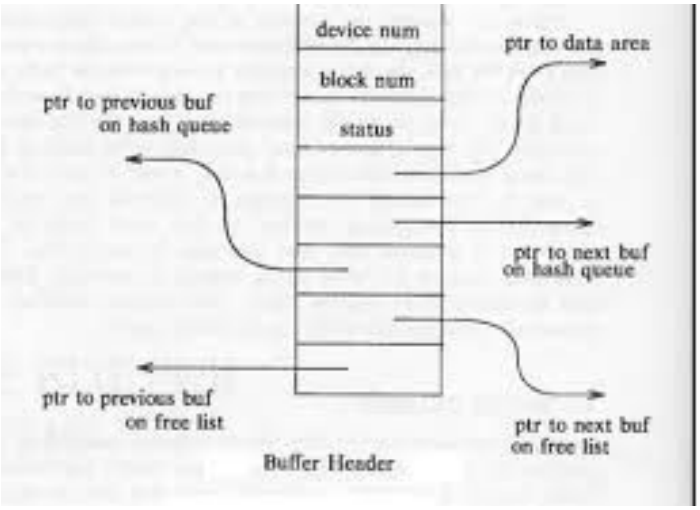
Free list and hash queue will be maintained as doubly circular linked list :



1. Two objects has create is created of Buffer_cahce class one for HashQueue and another for freelist i.e, hashQueue[4], freelist
   Initially at a beginning initialize free list with some initial buffer, let say 20 buffers having initial value of block number is 0.
   At the beginning the hash queue will be empty.
   When multiple threads will execute, demanded buffer will be inserted at the tail of corresponding hash queue.

2. Multiple theads is created where each thread call processManager() where each thread will race for getting block number by calling **getblk()** method.

3. Implemention of getblk() algorithm , brealse() algorithm



Structure of Buffer

Programing language used - C++

AOS concepts implemented

1) **Multithreading**
2) **Locking and Unlocking** : Mutex Lock are used for synchronisation among multiple thread.
3) **Signal Handling :** Signal handling is done via conditional variable. Wait() and Signal() function are used. Wait() function will be called when any thread is waiting for buffer to become free. Wait causes the current thread to block until the condition variable is notified. The thread will be unblocked when notify_all() or notify_one() is executed.
   Signal() is used to wake up all the the waiting process waiting for free buffer or particular buffer.