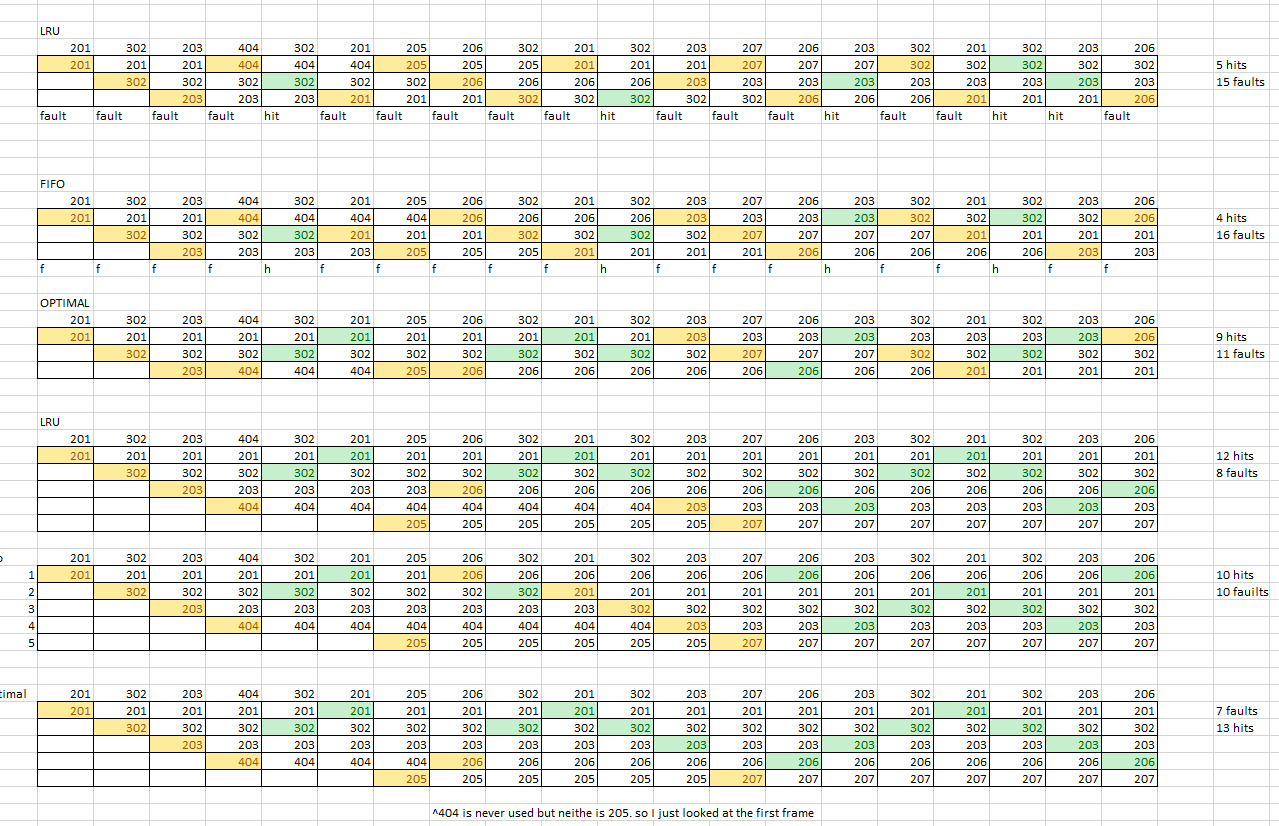
Question 1: i), ii).



Question 2:

1. If the memory reference takes 250ns, then a paged memory reference should take 250 to access the page table and then another 250ns to find the page in memory.
2. Effective Access Time = (1+e)(a) +(2+e)(1-a)

(1+30)(0.8) +(2+30)(1-0.8)

31.2ns

1. TLB improves performance because it is much faster than memory on a hardware level. Also, accessing a single bit in the page table may take multiple passes.

Question 3:

1. With 32 pages, you need 5 bits, 2n = 32, n = 5 for the address. For each page, 2 KB are required, and 214 bits is 2048 Bytes.
2. The page table is 32 pages long. The memory is 1Mbyte, which is 223 bits. The width is the difference between the the offset of the page and the size of the memory. So, 223 – 214 = 29. Nine bits wide.
3. As long as the page table is still less than the size of the memory device it should be fine. And if some entries get cut out, there would be an ordinary page fault, from the perspective of the the processor, get cut off is equivalent to what was seen in Q1, so it would cause a page fault.

Question 4:

1. A relevant example of a race condition is in a system with processes that run multithreaded. Imagine two processes that run on with threading, one will increment an integer in memory up and the other down then print the value. Under ideal conditions, you’d expect the printed value to stay around 0 and 1. However, in real execution, an interrupt may cause one process to run less than the other, and you may see that the value becomes negative, or way greater than 1. Due to the non-synchronous nature of multiple threads, race conditions like these exist in code that needs to complete in a sequential manner.
2. Disabling interrupts would not solve the race condition described above. It is possible that in runtime, the length of those operations may differ slightly and overall the problem would still persist, so disabling interrupts is not a mechanism to solve the race condition. To solve this particular condition, a lock or a wait or a similar flag that prevents the threads from outpacing each other. For example, Thread A decrements the integer. Thread B increments the integer. A lock will be on Thread A that prevents it from running more than Thread B runs, that way, the printed value should never be anything besides 0.

Question 5:

A semaphore is an integer value or a binary value that is used to achieve synchronization in multithreaded processes. Using the same example as before for the race condition, we could introduce a Boolean semaphore. Thread A decrements the integer, then changes the value of the semaphore in the Thread B to allow it to commence. Thread B increments the integer then changes the value of the semaphore in Thread A to run.

Question 6:

First, the location of the file on disk is known, through its location in subfolders. The content of the file is moved to the memory and then a pointer is made to the location of the file in memory. The location of the file on the disk is also needed. Permission for write are needed to append to the file. You would then use sequential access to write to the end of the file.

Question 7:

1. Single disk = 512 \* 2KB = 512\*29 = 220

Double disk = 512 \*512 \* 2KB = 229

12 Disk Block = 12 \* 2KN = 216

The maximum size of a file would be slightly less than 229 Bytes on the double disk, the largest block.

1. You couldn’t store a larger file unless you somehow would somehow fragmented the file across multiple blocks