Assignment 3 Concepts: Dimitry Koutchine 101114229 & Phil Baird 101117932

Question 1 i)

Bold = page fault

LRU

201 302 203	404	302 201	205	206	302	201	302 203	207	206	203 302	201	302 203 206
201	**404**	404	**205**	205	205	**201**	201	**207**	207	**302**	302	302
302	302	302	302	**206**	206	206	**203**	203	203	203	203	203
203	203	**201**	201	201	**302**	302	302	302	**206**	206	**201**	**206**

15 page faults

FIFO

201 302 203	404 302 201 205	206 302 201	302 203 207 206	203 302 201 302 203	206
201	**404**	**206**	**203**	**302**	**206**
302	**201**	**302**	**207**	**201**	
203	**205**	**201**	**206**	**203**	

16 page faults

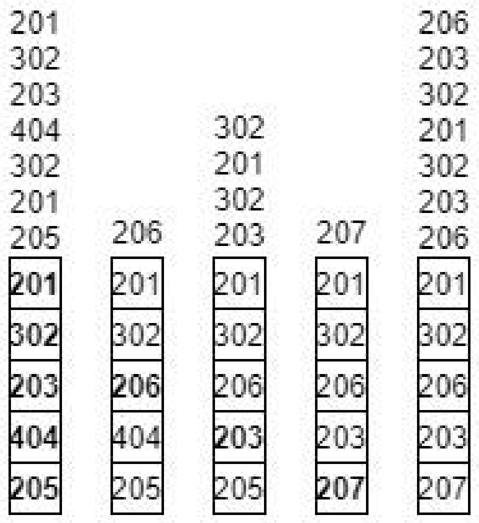
Optimal

201 302 203	404	302 201 205	206	302 201 302 203	207	206 203 302	201	302 203 206
201	201	201	201	**203**	203	203	203	**206**
302	302	302	302	302	**207**	**302**	302	302
203	**404**	**205**	**206**	206	206	206	**201**	201

11 page faults

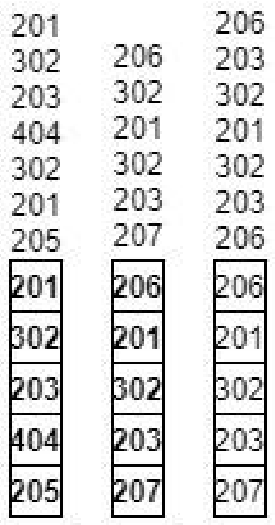
ii)

LRU



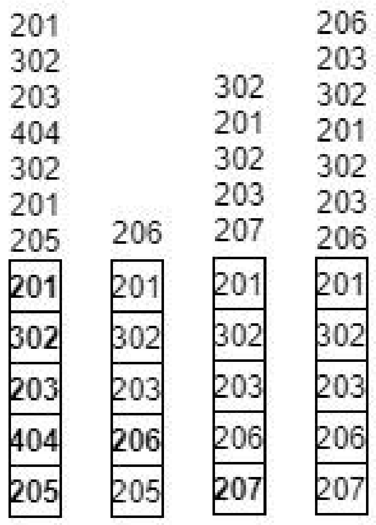
8 page faults

Fifo



10 page faults

Optimal



7 page faults

Question 2.

a) It would take 500ns since when performing a paged memory reference it must first access memory to get the reference from the table, and then proceed to access memory again to get required data from the required byte in memory which results in two memory references of 250ns.
b) Effective access time = hit chance x memory ref time+ miss chance x TLB overhead
Effective access time =0.8 x 280ns + 0.2 x 580ns = 397 ns
c) Adding a TLB eliminates the need for accessing memory twice in order to get a desired result which as a result significantly lowers the time it takes to get the needed data. It accomplishes this by keeping track of important memory locations which allows for almost immediate access to locations of data that are stored in the TLB. However it does have costs associated with it that may even in the worst case cause it to be less efficient then not having it. The two main costs are the overhead associated with it and the hit chance. Overhead is the time it takes to traverse the TLB and Hit chance are the odds that the wanted memory address is in the TLB. Since not everything can be in the tlb there will always be a chance that the required data will not be found in the tlb and therefore will require 2 accesses to memory. If the hit chance is low and the overhead is high enough it may be better to not even have a tlb as the Effective access time would be greater than not having it. However in the previous question with the implemented TLB the EAT is 103 ns faster then not having it.

Question 3.
a) The address will be split up into page number and page offset. Since 32 = 2^5 the page number part of the address will take up 5 bits and since 2000 = 2^11 the page offset will take up 11 bits.
b)The length is 32, and since 1000/2 is approx 2^9 therefore the width is 9 bits.
c) The length will stay the same however since 500/2 is approx 2^8 therefore the width will be 8 bits

Question 4.

a) A race condition occurs when there is concurrency in the system and multiple processes access a shared variable at the same time and cause an error. Errors due to race conditions can vary from the unexpected change to a variable to entire threads being locked out. A specific example of a race condition would 2 threads sharing a variable and while thread 1 is in the process of changing the variable it gets interrupted, in the meantime thread 2 takes this value expecting it to be already altered by thread 1, and thus receiving an inaccurate value.
b) While disabling interrupts might help a process not leave a half finished job, in the long run it would cause many more issues as interrupts are essential for kernel level processes , and thus turning it off is not an effective way to counteract race conditions.

Question 5.
In the most simple terms a semaphore is an integer that is initialized at 0 on the startup of a program and can only be altered through the use of 2 specific operations, wait() and signal().

The signal () operation increases the semaphore by 1, while the wait() operation suspends the process that calls it until the semaphore value is 1 or greater. The semaphore value can only be altered automatically so only one process can alter it at a time. The main use of semaphores is to synchronize two or more processes by having them share a semaphore. A process that needs to run after another will call the wait() operation and essentially wait for the other process to use signal() letting it know it can now run.

Question 6.
In order to be able to do other operations like append or read, the open operation must be used. Instead of having to move the required file into memory every time you use a command, you use the open() command with a file name as an argument, which moves it into memory and returns the reference that other operations like append() will take as an argument to make changes to the file in memory. After you are done editing the file the close() operation must be called, which will remove the file from memory and save it to the directory.

Question 7.

a)
Disk block = 8Kb = 1KB
12 Direct Disk Block = 1KB *12 = 12 KB
Single Indirect Disk = 1KB / 4B *1KB= 2^8B * 1KB =256000B =256KB
Double Indirect Disk = 256B * 256B *1000B = 65536 KB =
Triple Indirect Disk = 256B * 256B * 256B * 1000B= 16777216 KB
Total = 12 + 256 + 65536 + 16777216 = 16843020 KB = 16.8 GB = 16 GB

Therefore it can store around 16 GB.

b)
The reason for indirect disk block is that with direct disk blocks, in a 32 bit system it would only be able to store up to 4 GBs of pointer values for blocks. Depending on how many blocks you would have this would be insufficient for most systems and thus indirect disk blocks are used to be able to be able to support more blocks. However if you are only required to store 1 file that's larger than the allocated space, direct disk blocks would be sufficient since you can have that file in one block. The maximum file that can be stored would be in one big block so pointers storage wouldn't be necessary. The maximum storage would be...
Disk block = 8Kb = 1KB
12 Direct Disk Block = 1KB *12 = 12 KB
Single Indirect Disk = 1KB *1KB= 1MB
Double Indirect Disk = 1KB * 1KB *1KB =1GB
Triple Indirect Disk = 1KB * 1KB * 1KB * 1KB= 1TB
Total = 12KB + 1MB + 1GB + 1TB = 1TB
Therefore a 1TB file would be the max size.