1. A. LRU

For LRU there are 12 page faults (each arrow represents a page fault).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 201 -> | 404-> | 205 -> | 201 -> | 206 -> | 201 -> | 206 |
| 302 | -> | 206 -> | 203 |  |  |  |
| 203 -> | 201 -> | 302 -> | 207 -> | 302 |  |  |

B. FIFO

For FIFO there are 13 page faults (Each arrow represents a page fault).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 201 -> | 404 -> | 206 -> | 203 -> | 302 -> | 206 |
| 302 -> | 201 -> | 302 -> | 207 -> | 201 |  |
| 203 -> | 205 -> | 201 -> | 206 -> | 203 |  |

C. Optimal

For Optimal there are 8 page faults (Each arrow represents a page fault).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 201 |  |  | -> | 207 -> | 206-> | 201 -> | 206 |
| 302 |  |  |  |  |  |  |  |
| 303 -> | 404 -> | 205 -> | 206 -> | 203 |  |  |  |

D.

LRU there are 3 Page faults (each arrow represents a page fault).

|  |  |
| --- | --- |
| 201 |  |
| 302 |  |
| 203 -> | 206 |
| 404 -> | 203 |
| 205-> | 207 |

FIFO there are 5 page faults.

|  |  |
| --- | --- |
| 201 -> | 206 |
| 302 -> | 201 |
| 203 -> | 302 |
| 404 -> | 203 |
| 205 -> | 207 |

Optimal there are 2 page faults (each arrow represents a page fault.)

|  |  |
| --- | --- |
| 201 |  |
| 302 |  |
| 203 |  |
| 404 -> | 206 |
| 205 -> | 207 |

2 a.

Because there are 2 memory accesses it will take 500 ns because of the following

P =0

(1-0)(250+250) = 500 nm

2b. Assuming 2 memory reads again along with TLB overhead it will take 330 ns because of the following

(1-P)\* mem access + P\*(pagefault overhead)

= 0.8(250+30) + 0.2(250+250+30) = 330 ns.

2c. There is a low hit percentage then TLB will add more overhead than reducing than reducing time for a hit.

3a. There will be 5 bits for the logical page number because 32 = 2^5. And 11 bits for the page offset because 2K bytes is equal to 2^10 \* 2^1 = 2^11.

3b. The length of the page table will be 32 because there are 32 pages. And the width of the page table would be 9 bits because we have 1Mbyte of physical memory space and 2Kbytes offset. Therefore the width is 2^20 /2^11 = 2^9 therefore 9 bits.

3c. Physical memory is 2^20 halving that will equal 2^19. Therefore the number of bits for the width will be 8 bits because similar to the prior equation 2^19 /2^11 = 2^8. Thus the width is reduced by 1.

4. a) A race condition is a situation that is undesirable which occurs when a device or system tries to perform more than one operation simutaniously but the operations must be done in a certain sequence in order to be correct.

An example of a race condition is when two threads try to change the same value. One thread does a “check-then-act” meaning check the condition and then do an action while the other thread changes the value between checking of the condition and the action.

b) Unlike in a uniprocessor system where using CLI prevents a race condition from occuring, in a multiprocessor systems when executing a CLI instruction only disables interrupts on a particular processor thus leaving the other processors free to handle interrupts.

This does not prevent race condtions because the other processors are able to handle interrupts and are able to start executing the offending interrupt handler.

In order to prevent a race condition from occuring in a multiprocessor system, CLI and locks must be used simutaniously.

5. A semaphore is a way to lock a resource in order to prevent a race condition. A semaphore is like a mutex however a semaphore can support a fixed number of simultaneous callers. Once the limit is hit, it will block the rest of the callers until one of the slots are freed.

6. There are several steps that the open operation must do. First it must pass a file name to the logical file system and perform a search on the system-wide open-file table to see if the file is not already open. If it is, then a per-process open file table entry is created pointing to the open-file table. Otherwise, the directory stucture is searched for the given file name. Once the file is found, the FCB (File Control Block) is copied into the open-file table and a per-process open file table entry is created. After all this is done, the open opearation returns a pointer to the appropriate enty in the per-process open file table.

7. a) [Assuming there is 1 single indirect pointer, 1 double indirect pointer and 1 triple indirect pointer.](http://www2.cs.uregina.ca/~hamilton/courses/330/notes/allocate/allocate.html)

Then:

Max Num of bytes addressed by 12 direct pointers:

Num of direct pointers \* block size

= 12 \* 8Kb

= 96 Kb

Max Num of bytes addressed by 1 single indirect pointer:

Number of enteries \* block size

= ( block size / pointer size ) \* block size

= (8Kb / 4b) \* 8Kb

= 2^11 \* 2^13

= 2^24 b

= 16 Mb

Max Num of bytes addressed by 1 double indirect pointer:

(Number of enteries) ^ 2 \* block size

= (block size / pointer size) ^ 2 \* block size

= (8Kb / 4b) ^ 2 \* 8Kb

= (2^11) ^ 2 \* 2^13

= 2^22 \* 2^13

= 2^35 b

= 32 Gb

Max Num of bytes addressed by 1 double indirect pointer:

(Number of enteries) ^ 3 \* block size

= (block size / pointer size) ^ 3 \* block size

= (8Kb / 4b) ^ 3 \* 8Kb

= (2^11) ^ 3 \* 2^13

= 2^33 \* 2^13

= 2^46 b

= 64 Tb

Therefore the maximum file is 64 Tb + 32 Gb + 16 Mb

b) In the event that the file you need to save is larger than the allowed size is, then you can attempt to split the file into separate smaller files and save it.