Course : Operating Systems **Instructor:** 黃能富

Student name: Tuguldur Tserenbaljir **Assignment:** Assignment 6

Student ID: 109006271

9.17)

a1) What is the initial value of the counters?

The counters should be initialized to zero at the start. This setting implies that initially, no pages are loaded into the frames. As pages begin to be loaded into memory, the counters will be updated accordingly.

a2) When are counters increased?

When a page is loaded into a frame: Every time a new page is loaded into a page frame, the counter for that frame is incremented by one, indicating that an additional page is now with that frame. Meaning that the counter is increased.

When a page is accessed: The corresponding frame's counter is also incremented to reflect the increased activity or relevance of the pages it contains.

a3)When are counters decreased?

Counters are decreased when a page is removed from a frame. That is when a page is replaced or the page is no longer needed. When a page within a frame is replaced by another page due to a page fault, the counter for that frame is decreased by one. On the other hand, If a page in a frame is closed or no longer needed the counter is decreased accordingly.

a4) How is the page to be replaced selected?

It can be either the algorithm of Least Recently Used (LRU) or the First-In-First-Out strategy (FIFO). They can explained as replace the page that has not been used for the longest period within that frame or that has been in the frame the longest respectively (LRU or FIFO).

b) 1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2. (Please Look at the Table Below **Table 9.17b**)

Course : Operating Systems **Instructor:** 黃能富

Student name: Tuguldur Tserenbaljir **Assignment:** Assignment 6

Student ID: 109006271

Ref	**Frame 1**	**Frame 2**	**Frame 3**	**Frame 4**	**Fault/Hit**
1	1	-	-	-	Fault
2	1	2	-	-	Fault
3	1	2	3	-	Fault
4	1	2	3	4	Fault
5	5	2	3	4	Fault
3	5	2	3	4	Hit
4	5	2	3	4	Hit
1	5	1	3	4	Fault
6	5	1	6	4	Fault
7	5	1	6	7	Fault
8	8	1	6	7	Fault
7	8	1	6	7	Hit
8	8	1	6	7	Hit
9	8	9	6	7	Fault
7	8	9	6	7	Hit
8	8	9	6	7	Hit
9	8	9	6	7	Hit
5	8	9	5	7	Fault
4	8	9	5	4	Fault
5	8	9	5	4	Hit
4	8	9	5	4	Hit
2	2	9	5	4	Fault

(Table 9.17b)

The algorithm that it follows is that:

- For each number in the reference string, check if the number is already in one of the frames.
- If the number is not in the frames, this is a page fault. Load the number into a frame.
- If all frames are full, replace the least recently used (LRU) page. The LRU page is determined by the longest time since it was last referenced.

Course : Operating Systems **Instructor:** 黃能富

Student name: Tuguldur Tserenbaljir **Assignment:** Assignment 6

Student ID: 109006271

For example, until all 4 frames are loaded in and are full it is Fault (Ref: 1,2,3,4). Then, 5: Not in frame. Replace the least recently used (1). **Fault**. Next is checks for the next **Ref** which is 3,4 and it is already in Frame, so **Hit**. It follows the algorithm above until there is no more reference to check. Thus, making it 13 **Fault** in total.

c) What is the minimum number of page faults for an optimal page replacement strategy for the reference string in part b with four page frames?

(Below Table 9.17c)

Ref	**Frame 1**	**Frame 2**	**Frame 3**	**Frame 4**	**Fault/Hit**
1	1	-	-	-	Fault
2	1	2	-	-	Fault
3	1	2	3	-	Fault
4	1	2	3	4	Fault
5	1	5	3	4	Fault
3	1	5	3	4	Hit
4	1	5	3	4	Hit
1	1	5	3	4	Hit
6	6	5	3	4	Fault
7	6	5	7	4	Fault
8	8	5	7	4	Fault
7	8	5	7	4	Hit
8	8	5	7	4	Hit
9	8	5	7	9	Fault
7	8	5	7	9	Hit
8	8	5	7	9	Hit
9	8	5	7	9	Hit
5	8	5	7	9	Hit
4	8	5	4	9	Fault
5	8	5	4	9	Hit
4	8	5	4	9	Hit
2	2	5	4	9	Fault

Course : Operating Systems **Instructor:** 黃能富

Student name: Tuguldur Tserenbaljir **Assignment:** Assignment 6

Student ID: 109006271

To explain the contents of table 9.17c:

- Until all 4 frames are loaded in and are full it is **Fault** (Ref: 1,2,3,4).
- 5: Not in frame. Looking ahead, pages 1, 2, 3, and 4 are used again, but 1 is used last among them. Replace 1. **Fault**.
- 3: Already in frame. **Hit**.
- 4: Already in frame. **Hit**.
- 1: In frame. **Hit.**
- 6: Not in frame. (1,5,3,4) 1 is used last among them. Replace 1. Fault. Making it (6,5,3,4). **Fault**
- Repeat.

This table follows the optimal strategy by always replacing the page that will be needed last, minimizing the number of page faults. There are 11 **FAULT** in total.

9.18)

EMAT=> (**AMHR**×**AM Access Time**) +
(**Non-Page Fault Rate within NAMHR**×**Memory Access Time for 2 accesses**) +
(**PFR**×**Disk Access Time**)

AMHR =80%
AM Access Time = 1 μ
Non-page fault rate within non-AM accesses = 18% (90% of 20%)
Page fault rate (PFR): 10% of the 20% non-AM accesses, which is 2% of the total

EMAT => (0.8 * 1μ) + (0.18 * 2 μ) + 0.02 * (20000 μ + 2 ns) =>
EMAT => 401.2 μ

Course : Operating Systems **Instructor:** 黃能富

Student name: Tuguldur Tserenbaljir **Assignment:** Assignment 6

Student ID: 109006271

9.19)

Thrashing occurs when a system or program spends excessive time swapping data between memory and disk due to high demand or insufficient resources.

a) What is the cause of thrashing?

Thrashing mainly arises from memory resource configuration with less capacity to meet the system's requirements. It arises in situations where there is inadequate main memory to hold all the required tangible pages for the running processes, and it forces a system to swap the pages repeatedly. This can also be made worse by running too many processes at once such that, the total amount of memory requested is bigger than the physical memory available and instead of clearing the memory up the processes will keep on fighting for the available memory.

Also, the improper management of memory causes the pages to be loaded and unloaded more often and in a more inefficient manner. A third factor, that can also be considered as being of considerable importance, focuses on the improper allocation of the working sets of processes. A working set consists of the absolute minimum number of pages that a process needs to function optimally in operation. If a process is active and its working set is not fully in memory, then the situation leads to paging faults and, thus, thrashing.

b) How does the system detect thrashing?

Operating systems monitor several indicators to recognize thrashing conditions. A high rate of paging activities, such as excessive page faults and swapping, is a primary indicator. Moreover, a noticeable drop in CPU utilization in conjunction with high paging rates can signal thrashing, as the CPU spends more time handling page faults than executing processes.

c) Once it detects thrashing, what can the system do to eliminate this problem?

Once detected, various strategies can be employed to eliminate thrashing. One straightforward approach is to increase the physical memory available, which directly reduces the need for swapping. Alternatively, operating systems might temporarily suspend one or more processes to reduce the load on the system's memory. This suspension allows the system to provide more memory for the remaining active processes, ensuring that their working sets are completely resident in memory.

Also, djusting the working set size of processes can also be beneficial by increasing the memory frames allocated to each process to accommodate its entire working set. Additionally, systems can implement load balancing to distribute process loads more evenly across available resources, or adjust process priorities to favor less memory-intensive processes.