**Operating Systems**

Shyam Rajendren

Operating Systems Environments & Administration

1. Explain why it is possible for an operating system to load only a couple of pages of a program rather than the entire program?

Answer: The program is divided into pages of equal sizes. These program pages are loaded into noncontiguous page frames. This means that only a few pages of a program are loaded at a time rather than the entire program. The advantage of this is more efficient memory use and the elimination of external fragmentation.

1. Describe the purpose of the **Job Table** (JT), **Page Map Table** (PMT) and **Memory Map Table** (MMT).

Answer: Job Table stores the information for each active job including the job size and the memory location of the Page Map Table. The Page Map Table stores the information of each page including memory address. The Memory Map Table has an entry for each page frame including its location and whether it is free or busy.

1. Explain how the **Job Table** (JT), **Page Map Table** (PMT), and **Memory Map Table** (MMT) are each used to manage a program stored in memory.

Answer: The Job Table initially has one entry for each job. When the initial job ends, its entry in the table is released and then replaced by the entry for the next job. The Job Table is a dynamic list that grows as jobs are loaded into the system and shrinks as they are completed. Each active job has its own Page Map Table (PMT), which contains the page number and its corresponding memory address. These page numbers are sequential so it isn’t necessary to list each page number in the PMT. In other words, the first entry in the PMT lists the page frame memory

address for Page 0, the second entry is the address for Page 1, and so on.

A simple Memory Map Table has an entry for each page frame, and shows its location and its free or busy status. Let us take a program that is 250 bytes long. Let us assume that each page is divided into 100 bytes. This would mean Page 0 contains bytes 0-99, Page 1 contains bytes 100-199, and Page 3 contains bytes 200-249. Though the program has 250 bytes, the system stores them as numbers starting from 0 to 249. Thus, they are referred to by the system as Byte 0 through Byte 249.

1. What are some advantages and disadvantages of selecting a page frame size of **1 MB**? List at least one (1) advantage and disadvantage.

Answer: An advantage is that there is less overhead in the reading and writing of pages. A disadvantage is that there is more internal fragmentation.

1. What are some advantages and disadvantages of selecting a page frame size of **100 bytes**? List at least one (1) advantage and disadvantage.

Answer: An advantage is that there is less internal fragmentation. A disadvantage is that there is more overhead in the reading and writing of pages.

1. Answer the following questions about a program containing **512 bytes** and an operating system using a page frame size of **200 bytes**.  
   1. How many page frames are necessary to store the entire program in memory?

Answer: 3

* 1. How many bytes of **external** fragmentation exists when the entire program is stored in memory?

Answer: Zero, external fragmentation does not occur in paging.

* 1. How many bytes of **internal** fragmentation exists in page frame containing the first page of the program?

Answer: Zero, internal fragmentation occurs on the last page.

* 1. How many bytes of **internal** fragmentation exists in page frame containing the last page of the program?

Answer: 512 bytes = 2 pages + 112 bytes. Number of bytes of internal fragmentation = 200 - 112 = 88 bytes.

* 1. On which page would you find the program instruction at byte **256**?

Answer: Page 1

* 1. What is the displacement within the page of the program instruction at byte **256**?

Answer: 56 bytes

1. Answer the following questions about **thrashing**.
   1. What is thrashing?

Answer: Trashing occurs when an excessive amount of page swapping back and forth between main memory and secondary storage results in higher overhead and little work being done.

* 1. Explain how thrashing improves or degrades system performance.

Answer: Trashing severely degrades system performance. For example, when there is large number of jobs vying for a few free pages then pages will be frequently removed from memory only to be called back a short time after. This leads to excessive page swapping, and the time done in doing this leads to negligible work being done.

* 1. Provide at least two (2) causes of thrashing.

Answer: 1. There is an under allocation of the minimum number of pages required by a process forcing a continuous page fault. 2. There are many jobs to be executed but only a relatively small number of free pages.

1. Refer to this page request sequence A, C, B, D, A, C, E, A, C, B, D, E to answer the following questions.
   1. Perform a page trace analysis using three (3) page frames and the **First-In First-Out** (FIFO) page replacement algorithm.

Answer: Success Rate = 1 – Number of Interrupts/Page Requests Made = 1 - 6/12 = 6/12.

* 1. Perform a page trace analysis using three (3) page frames and the **Least Recently Used** (LRU) page replacement algorithm.

Answer: Success Rate = 1 – Number of Interrupts/Page Requests Made = 1 - 7/12 = 5/12.

* 1. Which page replacement algorithm performed best for this page request sequence?

Answer: First-In-First-Out.

* 1. Will one-page replacement algorithm **always** perform better than the other?

Answer: No.

* 1. Perform a page trace analysis using four (4) page frames and the **First-In First-Out** (FIFO) page replacement algorithm.

Answer: Success Rate = 1 – Number of Interrupts/Page Requests Made = 1 - 6/12 = 6/12.

* 1. Did increasing the number of available page frames increase or decrease the Success Rate of the **First-In First-Out** (FIFO) page replacement algorithm? Explain the cause of this increase or decrease.

Answer: The success rate remained the same. Usually, increasing the number of page frames would increase the success rate as more page frames would means less chance of a page needing to be swapped. However, it is possible in FIFO for the success rate to decrease with an increase in the number of page frames though it is rare.

* 1. What is the fewest number of page faults that could occur using either the First-In First-Out (FIFO) or Least Recently Used (LRU) page replacement algorithms?

Answer: Zero. If there are page frames equivalent to or greater than the number of tasks then a page fault will not occur.

1. Answer the following questions about cache memory.
   1. Which type of memory (cache or main memory) is **faster**?

Answer: Cache memory.

* 1. Which type of memory (cache or main memory) is **more expensive**?

Answer: Cache memory.

* 1. Which type of memory (cache or main memory) is **larger**?

Answer: Main memory.

1. Answer the following questions about cache memory using a computer system with the following characteristics: **Total Memory Requests**: 5000, **Memory Requests Found in Cache**: 3500, **Average Main Memory Access Time**: 1000 nsec, and **Average Cache Memory Access Time**: 200 nsec.
   1. What is the **Hit Ratio** for the system?

Answer: Hit Ratio = Memory Requests Found in Cache/Total Number of Requests \* 100 = 3500/5000 \* 100 = 70%.

* 1. What is the **Average Memory Access Time** for the system?

Answer: Average Memory Access Time = Average Cache Access Time + (1 – Hit Ratio) \* Average Main Memory Access Time = 200 nsec + (1 – 0.7) \* 1000 nsec = 200 + 0.3 \* 1000 = 200 + 300 = 500 nsec.

* 1. Explain why the Average Cache Memory Access Time is a constant value in the **Average Memory Access Time**formula.

Answer: This is based on the assumption that the system will always check the cache memory first.

* 1. Explain why the Average Main Memory Access Time is dependent on the **Hit Ratio** value in the **Average Memory Access Time**formula.

Answer: The system will access the main memory on certain average time and it will achieve a hit a certain number of times (hit ratio). Thus, multiplying these two quantities together, we can deduce how long a system will spend on finding a certain address in main memory.

* 1. What affect, if any, would doubling the available **cache** memory have on the **Average Memory Access Time**?

Answer: Doubling the available cache memory would increase the average cache memory access time as the system would have a larger memory to go through. Therefore, the average memory access time would also increase.