9.4 A certain computer provides its users with a virtual-memory space of 232 bytes. The computer has 2_{18} bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations.

Answer: The virtual address in binary form is
0001 0001 0001 0010 0011 0100 0101 0110
Since the page size is 2_{12}, the page table size is 2_{20}. Therefore the low-order 12 bits "0100 0101 0110" are used as the displacement into the page, while the remaining 20 bits "0001 0001 0001 0010 0011" are used as the displacement in the page table.

9.5 Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?

Answer:

$0.2 \_\text{sec} = (1 - P) \times 0.1 \_\text{sec} + (0.3P) \times 8 \text{ millisec} + (0.7P) \times 20 \text{ millisec}$

$0.1 = -0.1P + 2400\,P + 14000\,P$

$0.1 \_ 16{,}400\,P$

$P \_ 0.000006$

9.9 The VAX/VMS system uses a FIFO replacement algorithm for resident pages and a free-frame pool of recently used pages. Assume that the free-frame pool is managed using the least recently used replacement policy. Answer the following questions:
a. If a page fault occurs and if the page does not exist in the free frame pool, how is free space generated for the newly requested page?
b. If a page fault occurs and if the page exists in the free-frame pool, how is the resident page set and the free-frame pool managed to make space for the requested page?
c. What does the system degenerate to if the number of resident pages is set to one?
d. What does the system degenerate to if the number of pages in the free-frame pool is zero?

Answer:
• When a page fault occurs and if the page does not exist in the free-frame pool, then one of the pages in the free-frame pool is evicted to disk, creating space for one of the resident pages to be moved to the free-frame pool. The accessed page is then moved to the resident set.
• When a page fault occurs and if the page exists in the free-frame pool, then it is moved into the set of resident pages, while one of the resident pages is moved to the free-frame pool.
• When the number of resident pages is set to one, then the system degenerates into the page replacement algorithm used in the free-frame pool, which is typically managed in a LRU fashion.
• When the number of pages in the free-frame pool is zero, then the system degenerates into a FIFO page replacement algorithm.

9.10 Consider a demand-paging system with the following time-measured utilizations:
CPU utilization 20% Paging disk 97.7% Other I/O devices 5%
Which (if any) of the following will (probably) improve CPU utilization?
Explain your answer.
a. Install a faster CPU.
b. Install a bigger paging disk.

c. Increase the degree of multiprogramming.
d. Decrease the degree of multiprogramming.
e. Install more main memory.
f. Install a faster hard disk or multiple controllers with multiple hard disks.
g. Add prepaging to the page fetch algorithms.
h. Increase the page size.

Answer: The system obviously is spending most of its time paging, indicating over-allocation of memory. If the level of multiprogramming is reduced resident processes would page fault less frequently and the CPU utilization would improve. Another way to improve performance would be to get more physical memory or a faster paging drum.

a. Get a faster CPU—No.
b. Get a bigger paging drum—No.
c. Increase the degree of multiprogramming—No.
d. Decrease the degree of multiprogramming—Yes.
e. Install more main memory—Likely to improve CPU utilization as more pages can remain resident and not require paging to or from the disks.
f. Install a faster hard disk, or multiple controllers with multiple hard disks—Also an improvement, for as the disk bottleneck is removed by faster response and more throughput to the disks, the CPU will get more data more quickly.
g. Add prepaging to the page fetch algorithms—Again, the CPU will get more data faster, so it will be more in use. This is only the case if the paging action is amenable to prefetching (i.e., some of the access is sequential).
h. Increase the page size—Increasing the page size will result in fewer page faults if data is being accessed sequentially. If data access is more or less random, more paging action could ensue because fewer pages can be kept in memory and more data is transferred per page fault. So this change is as likely to decrease utilization as it is to increase it.

9.14 Consider a demand-paging system with a paging disk that has an average access and transfer time of 20milliseconds.Addresses are translated through a page table in main memory, with an access time of 1microsecond per memory access. Thus, each memory reference through the page table takes two accesses. To improve this time, we have added an associative memory that reduces access time to one memory reference, if the page-table entry is in the associative memory. Assume that 80 percent of the accesses are in the associative memory and that, of the remaining, 10 percent (or 2 percent of the total) cause page faults. What is the effective memory access time?

Answer:

effective access time = (0.8) × (1 _sec)

+ (0.1) × (2 _sec) + (0.1) × (5002 _sec)

= 501.2 _sec
= 0.5 millisec