**PROBLEMS**

**CHAPTER 3.**  **MEMORY MANAGEMENT**

1. 1. The IBM 360 had a scheme of locking 2-KB blocks by assigning each one a 4-bit key and having the CPU compare the key on every memory reference to the 4-bit key in the PSW. Name two drawbacks of this scheme not mentioned in the text.
2. 2. In Fig. 3-3 the base and limit registers contain the same value, 16,384. Is this just an accident, or are they always the same? It is just an accident, why are they the same in this example?
3. 3. A swapping system eliminates holes by compaction. Assuming a random distribution of many holes and many data segments and a time to read or write a 32-bit memory word of 4 nsec, about how long does it take to compact 4 GB? For simplicity, assume that word 0 is part of a hole and that the highest word in memory contains valid data.
4. 4. Consider a swapping system in which memory consists of the following hole sizes in memory order: 10 MB, 4 MB, 20 MB, 18 MB, 7 MB, 9 MB, 12 MB, and 15 MB. Which hole is taken for successive segment requests of
   1. (a) 12 MB
   2. (b) 10 MB
   3. (c) 9 MB

for first fit? Now repeat the question for best fit, worst fit, and next fit.

1. 5. What is the difference between a physical address and a virtual address?
2. 6. For each of the following decimal virtual addresses, compute the virtual page number and offset for a 4-KB page and for an 8 KB page: 20000, 32768, 60000.
3. 7. Using the page table of Fig. 3-9, give the physical address corresponding to each of the following virtual addresses:
   1. (a) 20
   2. (b) 4100
   3. (c) 8300
4. 8. The Intel 8086 processor did not have an MMU or support virtual memory. Neverthe-less, some companies sold systems that contained an unmodified 8086 CPU and did paging. Make an educated guess as to how they did it. (*Hint*: Think about the logical location of the MMU.)
5. 9. What kind of hardware support is needed for a paged virtual memory to work?
6. 10. Copy on write is an interesting idea used on server systems. Does it make any sense on a smartphone?
7. 11. Consider the following C program:

int X[N];

int step = M; /\* M is some predefined constant \*/

for (int i = 0; i < N; i += step) X[i] = X[i] + 1;

* 1. (a) If this program is run on a machine with a 4-KB page size and 64-entry TLB, what values of *M* and *N* will cause a TLB miss for every execution of the inner loop?
  2. (b) Would your answer in part (a) be different if the loop were repeated many times? Explain.

1. 12. The amount of disk space that must be available for page storage is related to the maxi-mum number of processes, *n*, the number of bytes in the virtual address space, *v*, and the number of bytes of RAM, *r*. Give an expression for the worst-case disk-space re-quirements. How realistic is this amount?
2. 13. If an instruction takes 1 nsec and a page fault takes an additional *n* nsec, give a formula for the effective instruction time if page faults occur every *k* instructions.
3. 14. A machine has a 32-bit address space and an 8-KB page. The page table is entirely in hardware, with one 32-bit word per entry. When a process starts, the page table is cop-ied to the hardware from memory, at one word every 100 nsec. If each process runs for 100 msec (including the time to load the page table), what fraction of the CPU time is devoted to loading the page tables?
4. 15. Suppose that a machine has 48-bit virtual addresses and 32-bit physical addresses.
   1. (a) If pages are 4 KB, how many entries are in the page table if it has only a single level? Explain.
   2. (b) Suppose this same system has a TLB (Translation Lookaside Buffer) with 32 en-tries. Furthermore, suppose that a program contains instructions that fit into one page and it sequentially reads long integer elements from an array that spans thou-sands of pages. How effective will the TLB be for this case?
5. 16. You are given the following data about a virtual memory system:

(a)The TLB can hold 1024 entries and can be accessed in 1 clock cycle (1 nsec).

* 1. A page table entry can be found in 100 clock cycles or 100 nsec.
  2. The average page replacement time is 6 msec.

If page references are handled by the TLB 99% of the time, and only 0.01% lead to a page fault, what is the effective address-translation time?

1. 17. Suppose that a machine has 38-bit virtual addresses and 32-bit physical addresses.
   1. (a) What is the main advantage of a multilevel page table over a single-level one?
   2. (b) With a two-level page table, 16-KB pages, and 4-byte entries, how many bits should be allocated for the top-level page table field and how many for the next-level page table field? Explain.
2. 18. Section 3.3.4 states that the Pentium Pro extended each entry in the page table hier-archy to 64 bits but still could only address only 4 GB of memory. Explain how this statement can be true when page table entries have 64 bits.
3. 19. A computer with a 32-bit address uses a two-level page table. Virtual addresses are split into a 9-bit top-level page table field, an 11-bit second-level page table field, and an offset. How large are the pages and how many are there in the address space?
4. 20. A computer has 32-bit virtual addresses and 4-KB pages. The program and data toget-her fit in the lowest page (0–4095) The stack fits in the highest page. How many en-tries are needed in the page table if traditional (one-level) paging is used? How many page table entries are needed for two-level paging, with 10 bits in each part?
5. 21. Below is an execution trace of a program fragment for a computer with 512-byte pages. The program is located at address 1020, and its stack pointer is at 8192 (the stack grows toward 0). Give the page reference string generated by this program. Each instruction occupies 4 bytes (1 word) including immediate constants. Both instruction and data references count in the reference string.

Load word 6144 into register 0

Push register 0 onto the stack

Call a procedure at 5120, stacking the return address

Subtract the immediate constant 16 from the stack pointer

Compare the actual parameter to the immediate constant 4

Jump if equal to 5152

1. 22. A computer whose processes have 1024 pages in their address spaces keeps its page tables in memory. The overhead required for reading a word from the page table is 5 nsec. To reduce this overhead, the computer has a TLB, which holds 32 (virtual page, physical page frame) pairs, and can do a lookup in 1 nsec. What hit rate is needed to reduce the mean overhead to 2 nsec?
2. 23. How can the associative memory device needed for a TLB be implemented in hard-ware, and what are the implications of such a design for expandability?
3. 24. A machine has 48-bit virtual addresses and 32-bit physical addresses. Pages are 8 KB. How many entries are needed for a single-level linear page table?
4. 25. A computer with an 8-KB page, a 256-KB main memory, and a 64-GB virtual address space uses an inverted page table to implement its virtual memory. How big should the hash table be to ensure a mean hash chain length of less than 1? Assume that the hash-table size is a power of two.
5. 26. A student in a compiler design course proposes to the professor a project of writing a compiler that will produce a list of page references that can be used to implement the optimal page replacement algorithm. Is this possible? Why or why not? Is there any-thing that could be done to improve paging efficiency at run time?
6. 27. Suppose that the virtual page reference stream contains repetitions of long sequences of page references followed occasionally by a random page reference. For example, the sequence: 0, 1, ... , 511, 431, 0, 1, ... , 511, 332, 0, 1, ... consists of repetitions of the sequence 0, 1, ... , 511 followed by a random reference to pages 431 and 332.
   1. (a) Why will the standard replacement algorithms (LRU, FIFO, clock) not be effective in handling this workload for a page allocation that is less than the sequence length?
   2. (b) If this program were allocated 500 page frames, describe a page replacement ap-proach that would perform much better than the LRU, FIFO, or clock algorithms.
7. 28. If FIFO page replacement is used with four page frames and eight pages, how many page faults will occur with the reference string 0172327103 if the four frames are ini-tially empty? Now repeat this problem for LRU.
8. 29. Consider the page sequence of Fig. 3-15(b). Suppose that the *R* bits for the pages *B* through *A* are 11011011, respectively. Which page will second chance remove?
9. 30. A small computer on a smart card has four page frames. At the first clock tick, the *R* bits are 0111 (page 0 is 0, the rest are 1). At subsequent clock ticks, the values are 1011, 1010, 1101, 0010, 1010, 1100, and 0001. If the aging algorithm is used with an 8-bit counter, give the values of the four counters after the last tick.
10. 31. Give a simple example of a page reference sequence where the first page selected for replacement will be different for the clock and LRU page replacement algorithms. As-sume that a process is allocated 3=three frames, and the reference string contains page numbers from the set 0, 1, 2, 3.

**32.** In the WSClock algorithm of Fig. 3-20(c), the hand points to a page with *R* = 0. If

*τ* 400, will this page be removed? What about if *τ* 1000?

33. Suppose that the WSClock page replacement algorithm uses a *τ* of two ticks, and the system state is the following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Page** | **Time stamp** | **V** | **R** | **M** |
|  |  |  |  |  |
| 0 | 6 | 1 | 0 | 1 |
|  |  |  |  |  |
| 1 | 9 | 1 | 1 | 0 |
|  |  |  |  |  |
| 2 | 9 | 1 | 1 | 1 |
|  |  |  |  |  |
| 3 | 7 | 1 | 0 | 0 |
|  |  |  |  |  |
| 4 | 4 | 0 | 0 | 0 |
|  |  |  |  |  |

where the three flag bits *V*, R, and *M* stand for Valid, Referenced, and Modified, re-spectively.

* 1. (a) If a clock interrupt occurs at tick 10, show the contents of the new table entries. Ex-plain. (You can omit entries that are unchanged.)
  2. (b) Suppose that instead of a clock interrupt, a page fault occurs at tick 10 due to a read request to page 4. Show the contents of the new table entries. Explain. (You can omit entries that are unchanged.)

1. 34. A student has claimed that ‘‘in the abstract, the basic page replacement algorithms (FIFO, LRU, optimal) are identical except for the attribute used for selecting the page to be replaced.’’
   1. What is that attribute for the FIFO algorithm? LRU algorithm? Optimal algorithm?
   2. Give the generic algorithm for these page replacement algorithms.
2. 35. How long does it take to load a 64-KB program from a disk whose average seek time is 5 msec, whose rotation time is 5 msec, and whose tracks hold 1 MB
   1. for a 2-KB page size?
   2. for a 4-KB page size?

The pages are spread randomly around the disk and the number of cylinders is so large that the chance of two pages being on the same cylinder is negligible.

1. 36. A computer has four page frames. The time of loading, time of last access, and the *R* and *M* bits for each page are as shown below (the times are in clock ticks):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Page** | **Loaded** | **Last ref.** | **R** | **M** |
|  |  |  |  |  |
| 0 | 126 | 280 | 1 | 0 |
|  |  |  |  |  |
| 1 | 230 | 265 | 0 | 1 |
|  |  |  |  |  |
| 2 | 140 | 270 | 0 | 0 |
|  |  |  |  |  |
| 3 | 110 | 285 | 1 | 1 |
|  |  |  |  |  |

* 1. (a) Which page will NRU replace?
  2. (b) Which page will FIFO replace?
  3. (c) Which page will LRU replace?
  4. (d) Which page will second chance replace?

1. 37. Suppose that two processes *A* and *B* share a page that is not in memory. If process *A* faults on the shared page, the page table entry for process *A* must be updated once the page is read into memory.
   1. (a) Under what conditions should the page table update for process *B* be delayed even though the handling of process *A*’s page fault will bring the shared page into mem-ory? Explain.
   2. (b) What is the potential cost of delaying the page table update?
2. 38. Consider the following two-dimensional array:

int X[64][64];

Suppose that a system has four page frames and each frame is 128 words (an integer occupies one word). Programs that manipulate the *X* array fit into exactly one page and always occupy page 0. The data are swapped in and out of the other three frames. The *X* array is stored in row-major order (i.e., *X*[0][1] follows *X*[0][0] in memory). Which of the two code fragments shown below will generate the lowest number of page faults? Explain and compute the total number of page faults.

*Fragment A*

for (int j = 0; j < 64; j++)

for (int i = 0; i < 64; i++) X[i][j] = 0;

*Fragment B*

for (int i = 0; i < 64; i++)

for (int j = 0; j < 64; j++) X[i][j] = 0;

1. 39. You have been hired by a cloud computing company that deploys thousands of servers at each of its data centers. They have recently heard that it would be worthwhile to handle a page fault at server A by reading the page from the RAM memory of some other server rather than its local disk drive.
   * 1. (a) How could that be done?
     2. (b) Under what conditions would the approach be worthwhile? Be feasible?
2. 40. One of the first timesharing machines, the DEC PDP-1, had a (core) memory of 4K 18-bit words. It held one process at a time in its memory. When the scheduler decided to run another process, the process in memory was written to a paging drum, with 4K 18-bit words around the circumference of the drum. The drum could start writing (or reading) at any word, rather than only at word 0. Why do you suppose this drum was chosen?
3. 41. A computer provides each process with 65,536 bytes of address space divided into pages of 4096 bytes each. A particular program has a text size of 32,768 bytes, a data size of 16,386 bytes, and a stack size of 15,870 bytes. Will this program fit in the machine’s address space? Suppose that instead of 4096 bytes, the page size were 512 bytes, would it then fit? Each page must contain either text, data, or stack, not a mix-ture of two or three of them.
4. 42. It has been observed that the number of instructions executed between page faults is di-rectly proportional to the number of page frames allocated to a program. If the avail-able memory is doubled, the mean interval between page faults is also doubled. Sup-pose that a normal instruction takes 1 microsec, but if a page fault occurs, it takes 2001 sec (i.e., 2 msec) to handle the fault. If a program takes 60 sec to run, during which time it gets 15,000 page faults, how long would it take to run if twice as much memory were available?
5. 43. A group of operating system designers for the Frugal Computer Company are thinking about ways to reduce the amount of backing store needed in their new operating sys-tem. The head guru has just suggested not bothering to save the program text in the swap area at all, but just page it in directly from the binary file whenever it is needed. Under what conditions, if any, does this idea work for the program text? Under what conditions, if any, does it work for the data?
6. 44. A machine-language instruction to load a 32-bit word into a register contains the 32-bit address of the word to be loaded. What is the maximum number of page faults this in-struction can cause?
7. 45. Explain the difference between internal fragmentation and external fragmentation. Which one occurs in paging systems? Which one occurs in systems using pure seg-mentation?
8. 46. When segmentation and paging are both being used, as in MULTICS, first the segment descriptor must be looked up, then the page descriptor. Does the TLB also work this way, with two levels of lookup?
9. 47. We consider a program which has the two segments shown below consisting of instruc-tions in segment 0, and read/write data in segment 1. Segment 0 has read/execute pro-tection, and segment 1 has just read/write protection. The memory system is a demand-paged virtual memory system with virtual addresses that have a 4-bit page number, and a 10-bit offset. The page tables and protection are as follows (all numbers in the table are in decimal):

|  |  |  |  |
| --- | --- | --- | --- |
| **Segment 0** | | **Segment 1** | |
|  |  |  |  |
| **Read/Execute** | | **Read/Write** | |
|  |  |  |  |
| **Vir tual Page #** | **Pag e frame #** | **Vir tual Page #** | **Pag e frame #** |
|  |  |  |  |
| 0 | 2 | 0 | On Disk |
|  |  |  |  |
| 1 | On Disk | 1 | 14 |
|  |  |  |  |
| 2 | 11 | 2 | 9 |
|  |  |  |  |
| 3 | 5 | 3 | 6 |
|  |  |  |  |
| 4 | On Disk | 4 | On Disk |
|  |  |  |  |
| 5 | On Disk | 5 | 13 |
|  |  |  |  |
| 6 | 4 | 6 | 8 |
|  |  |  |  |
| 7 | 3 | 7 | 12 |
|  |  |  |  |

For each of the following cases, either give the real (actual) memory address which re-sults from dynamic address translation or identify the type of fault which occurs (either page or protection fault).

* 1. (a) Fetch from segment 1, page 1, offset 3
  2. (b) Store into segment 0, page 0, offset 16
  3. (c) Fetch from segment 1, page 4, offset 28
  4. (d) Jump to location in segment 1, page 3, offset 32

1. 48. Can you think of any situations where supporting virtual memory would be a bad idea, and what would be gained by not having to support virtual memory? Explain.
2. 49. Virtual memory provides a mechanism for isolating one process from another. What memory management difficulties would be involved in allowing two operating systems to run concurrently? How might these difficulties be addressed?
3. 50. Plot a histogram and calculate the mean and median of the sizes of executable binary files on a computer to which you have access. On a Windows system, look at all .exe and .dll files; on a UNIX system look at all executable files in */bin*, */usr/bin*, and */local/bin* that are not scripts (or use the *file* utility to find all executables). Determinethe optimal page size for this computer just considering the code (not data). Consider internal fragmentation and page table size, making some reasonable assumption about the size of a page table entry. Assume that all programs are equally likely to be run and thus should be weighted equally.
4. 51. Write a program that simulates a paging system using the aging algorithm. The number of page frames is a parameter. The sequence of page references should be read from a file. For a given input file, plot the number of page faults per 1000 memory references as a function of the number of page frames available.
5. 5. Write a program that simulates a toy paging system that uses the WSClock algorithm. The system is a toy in that we will assume there are no write references (not very realistic), and process termination and creation are ignored (eternal life). The inputs will be:

* The reclamation age threshhold
* The clock interrupt interval expressed as number of memory references
* A file containing the sequence of page references
  + 1. (a) Describe the basic data structures and algorithms in your implementation.
    2. (b) Show that your simulation behaves as expected for a simple (but nontrivial) input example.
    3. (c) Plot the number of page faults and working set size per 1000 memory references.
    4. (d) Explain what is needed to extend the program to handle a page reference stream that also includes writes.

1. 53. Write a program that demonstrates the effect of TLB misses on the effective memory access time by measuring the per-access time it takes to stride through a large array.
   * 1. (a) Explain the main concepts behind the program, and describe what you expect the output to show for some practical virtual memory architecture.
     2. (b) Run the program on some computer and explain how well the data fit your expecta-tions.
     3. (c) Repeat part (b) but for an older computer with a different architecture and explain any major differences in the output.
2. 54. Write a program that will demonstrate the difference between using a local page re-placement policy and a global one for the simple case of two processes. You will need a routine that can generate a page reference string based on a statistical model. This model has *N* states numbered from 0 to *N* − 1 representing each of the possible page references and a probability *pi* associated with each state *i* representing the chance that the next reference is to the same page. Otherwise, the next page reference will be one of the other pages with equal probability.
   * 1. (a) Demonstrate that the page reference string-generation routine behaves properly for some small *N*.
     2. (b) Compute the page fault rate for a small example in which there is one process and a fixed number of page frames. Explain why the behavior is correct.
     3. (c) Repeat part (b) with two processes with independent page reference sequences and twice as many page frames as in part (b).
     4. (d) Repeat part (c) but using a global policy instead of a local one. Also, contrast the per-process page fault rate with that of the local policy approach.
3. 55. Write a program that can be used to compare the effectiveness of adding a tag field to TLB entries when control is toggled between two programs. The tag field is used to ef-fectively label each entry with the process id. Note that a nontagged TLB can be simu-lated by requiring that all TLB entries have the same tag at any one time. The inputs will be:
   * + The number of TLB entries available
     + The clock interrupt interval expressed as number of memory references
     + A file containing a sequence of (process, page references) entries
     + The cost to update one TLB entry

(a) Describe the basic data structures and algorithms in your implementation.

(b) Show that your simulation behaves as expected for a simple (but nontrivial) input example.

(c) Plot the number of TLB updates per 1000 references.