# Exercises

**10.1** Under what circumstances do page faults occur? Describe the actions taken by the operating system when a page fault occurs.

**10.2** Assume that you have a page-reference string for a process with $m$ frames (initially all empty). The page-reference string has length $p$, and $n$ distinct page numbers occur in it. Answer these questions for any page-replacement algorithms:
**a.** What is a lower bound on the number of page faults?

**b.** What is an upper bound on the number of page faults?

**10.3** Consider the following page-replacement algorithms. Rank these algorithms on a five-point scale from "bad" to "perfect" according to their page-fault rate. Separate those algorithms that suffer from Belady's anomaly from those that do not.

- a. LRU replacement
- b. FIFO replacement
- c. Optimal replacement
- d. Second-chance replacement

**10.4** An operating system supports a paged virtual memory. The central processor has a cycle time of 1 microsecond. It costs an additional 1 microsecond to access a page other than the current one. Pages have 1,000 words, and the paging device is a drum that rotates at 3,000 revolutions per minute and transfers 1 million words per second. The following statistical measurements were obtained from the system:

- One percent of all instructions executed accessed a page other than the current page.
- Of the instructions that accessed another page, 80 percent accessed a page already in memory.
- When a new page was required, the replaced page was modified 50 percent of the time.

Calculate the effective instruction time on this system, assuming that the system is running one process only and that the processor is idle during drum transfers.

**10.5** Consider the page table for a system with 12-bit virtual and physical addresses and 256-byte pages.

| Page | Page Frame |
|------|------------|
| 0 | – |
| 1 | 2 |
| 2 | C |
| 3 | A |
| 4 | – |
| 5 | 4 |
| 6 | 3 |
| 7 | – |
| 8 | B |
| 9 | 0 |

The list of free page frames is *D, E, F* (that is, *D* is at the head of the list, *E* is second, and *F* is last). A dash for a page frame indicates that the page is not in memory.

Convert the following virtual addresses to their equivalent physical addresses in hexadecimal. All numbers are given in hexadecimal.

- 9EF
- 111
- 700
- 0FF

**10.6** Discuss the hardware functions required to support demand paging.

**10.7** Consider the two-dimensional array A:

```
int A[][] = new int[100][100];
```

where `A[0][0]` is at location 200 in a paged memory system with pages of size 200. A small process that manipulates the matrix resides in page 0 (locations 0 to 199). Thus, every instruction fetch will be from page 0.

For three page frames, how many page faults are generated by the following array-initialization loops? Use LRU replacement, and assume that page frame 1 contains the process and the other two are initially empty.

**a.**
```
for (int j = 0; j < 100; j++)
    for (int i = 0; i < 100; i++)
        A[i][j] = 0;
```
**b.**
```
for (int i = 0; i < 100; i++)
    for (int j = 0; j < 100; j++)
        A[i][j] = 0;
```

**10.8** Consider the following page reference string:

```
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.
```

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, and seven frames? Remember that all frames are initially empty, so your first unique pages will cost one fault each.

- LRU replacement
- FIFO replacement
- Optimal replacement

**10.9** Consider the following page reference string:

```
7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0, 1.
```

Assuming demand paging with three frames, how many page faults would occur for the following replacement algorithms?

- LRU replacement
- FIFO replacement
- Optimal replacement

**10.10** Suppose that you want to use a paging algorithm that requires a reference bit (such as second-chance replacement or working-set model), but the hardware does not provide one. Sketch how you could simulate a reference bit even if one were not provided by the hardware, or explain why it is not possible to do so. If it is possible, calculate what the cost would be.

**10.11** You have devised a new page-replacement algorithm that you think may be optimal. In some contorted test cases, Belady's anomaly occurs. Is the new algorithm optimal? Explain your answer.

**10.12** Segmentation is similar to paging but uses variable-sized "pages." Define two segment-replacement algorithms, one based on the FIFO page-replacement scheme and the other on the LRU page-replacement scheme. Remember that since segments are not the same size, the segment that is chosen for replacement may be too small to leave enough consecutive locations for the needed segment. Consider strategies for systems where segments cannot be relocated and strategies for systems where they can.

**10.13** Consider a demand-paged computer system where the degree of multiprogramming is currently fixed at four. The system was recently measured to determine utilization of the CPU and the paging disk. Three alternative results are shown below. For each case, what is happening? Can the degree of multiprogramming be increased to increase the CPU utilization? Is the paging helping?

**a.** CPU utilization 13 percent; disk utilization 97 percent

**b.** CPU utilization 87 percent; disk utilization 3 percent

**c.** CPU utilization 13 percent; disk utilization 3 percent

**10.14** We have an operating system for a machine that uses base and limit registers, but we have modified the machine to provide a page table. Can the page table be set up to simulate base and limit registers? How can it be, or why can it not be?

**10.15** Assume that a program has just referenced an address in virtual memory. Describe a scenario in which each of the following can occur. (If no such scenario can occur, explain why.)

- TLB miss with no page fault
- TLB miss with page fault
- TLB hit with no page fault
- TLB hit with page fault

**10.16** A simplified view of thread states is *ready*, *running*, and *blocked*, where a thread is either ready and waiting to be scheduled, is running on the processor, or is blocked (for example, waiting for I/O).
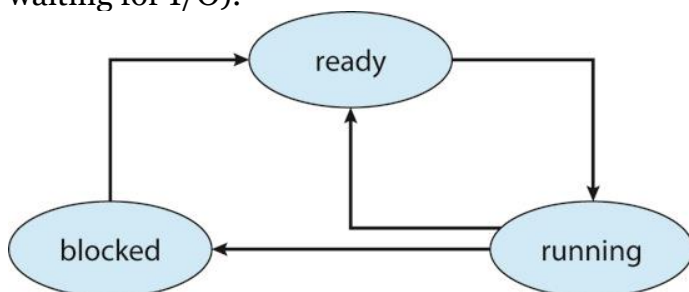


Assuming a thread is in the *running* state, answer the following questions, and explain your answers:

a. Will the thread change state if it incurs a page fault? If so, to what state will it change?

b. Will the thread change state if it generates a TLB miss that is resolved in the page table? If so, to what state will it change?

c. Will the thread change state if an address reference is resolved in the page table? If so, to what state will it change?

**10.17** Consider a system that uses pure demand paging.

a. When a process first starts execution, how would you characterize the page-fault rate?

b. Once the working set for a process is loaded into memory, how would you characterize the page-fault rate?

c. Assume that a process changes its locality and the size of the new working set is too large to be stored in available free memory. Identify some options system designers could choose from to handle this situation.

**10.18** The following is a page table for a system with 12-bit virtual and physical addresses and 256-byte pages. Free page frames are to be allocated in the order `9`, `F`, `D`. A dash for a page frame indicates that the page is not in memory.

| Page | Page Frame |
|---|---|
| 0 | 0 x 4 |
| 1 | 0 x B |
| 2 | 0 x A |
| 3 | – |
| 4 | – |
| 5 | 0 x 2 |
| 6 | – |
| 7 | 0 x 0 |
| 8 | 0 x C |
| 9 | 0 x 1 |

Convert the following virtual addresses to their equivalent physical addresses in hexadecimal. All numbers are given in hexadecimal. In the case of a page fault, you must use one of the free frames to update the page table and resolve the logical address to its corresponding physical address.

- `0x2A1`
- `0x4E6`
- `0x94A`
- `0x316`

**10.19** What is the copy-on-write feature, and under what circumstances is its use beneficial? What hardware support is required to implement this feature?

**10.20** A certain computer provides its users with a virtual memory space of $2^{32}$ bytes. The computer has $\$2^{22}$ bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4,096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations.

**10.21** Assume that 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 frame is available or if 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?

**10.22** Consider the page table for a system with 16-bit virtual and physical addresses and 4,096-byte pages.

| Page | Page Frame | Reference Bit |
|---|---|---|
| 0 | 9 | 0 |
| 1 | – | 0 |
| 2 | 10 | 0 |
| 3 | 15 | 0 |
| 4 | 6 | 0 |
| 5 | 13 | 0 |
| 6 | 8 | 0 |
| 7 | 12 | 0 |
| 8 | 7 | 0 |
| 9 | – | 0 |
| 10 | 5 | 0 |

| 11 | 4 | 0 |
|----|---|---|
| 12 | 1 | 0 |
| 13 | 0 | 0 |
| 14 | – | 0 |
| 15 | 2 | 0 |

The reference bit for a page is set to 1 when the page has been referenced. Periodically, a thread zeroes out all values of the reference bit. A dash for a page frame indicates that the page is not in memory. The page-replacement algorithm is localized LRU, and all numbers are provided in decimal.

a. Convert the following virtual addresses (in hexadecimal) to the equivalent physical addresses. You may provide answers in either hexadecimal or decimal. Also set the reference bit for the appropriate entry in the page table.

- 0x621C
- 0xF0A3
- 0xBC1A
- 0x5BAA
- 0x0BA1

b. Using the above addresses as a guide, provide an example of a logical address (in hexadecimal) that results in a page fault.

c. From what set of page frames will the LRU page-replacement algorithm choose in resolving a page fault?

**10.23** When a page fault occurs, the process requesting the page must block while waiting for the page to be brought from disk into physical memory. Assume that there exists a process with five user-level threads and that the mapping of user threads to kernel threads is many to one. If one user thread incurs a page fault while accessing its stack, would the other user threads belonging to the same process also be affected by the page fault—that is, would they also have to wait for the faulting page to be brought into memory? Explain.

**10.24** Apply the (1) FIFO, (2) LRU, and (3) optimal (OPT) replacement algorithms for the following page-reference strings:

- 2,6,9,2,4,2,1,7,3,0,5,2,1,2,9,5,7,3,8,5
- 0,6,3,0,2,6,3,5,2,4,1,3,0,6,1,4,2,3,5,7
- 3,1,4,2,5,4,1,3,5,2,0,1,1,0,2,3,4,5,0,1
- 4,2,1,7,9,8,3,5,2,6,8,1,0,7,2,4,1,3,5,8
- 0,1,2,3,4,4,3,2,1,0,0,1,2,3,4,4,3,2,1,0

Indicate the number of page faults for each algorithm assuming demand paging with three frames.

**10.25** Assume that you are monitoring the rate at which the pointer in the clock algorithm moves. (The pointer indicates the candidate page for replacement.) What can you say about the system if you notice the following behavior:

a. Pointer is moving fast.

b. Pointer is moving slow.

**10.26** Discuss situations in which the least frequently used (LFU) page-replacement algorithm generates fewer page faults than the least recently used (LRU) page-replacement algorithm. Also discuss under what circumstances the opposite holds.

**10.27** Discuss situations in which the most frequently used (MFU) page-replacement algorithm generates fewer page faults than the least recently used (LRU) page-replacement algorithm. Also discuss under what circumstances the opposite holds.

**10.28** The KHIE (pronounced "k-hi") operating 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 LRU replacement policy. Answer the following questions:

a. If a page fault occurs and 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 the page exists in the free-frame pool, how are the resident page set and the free-frame pool managed to make space for the requested page?

c. To what does the system degenerate if the number of resident pages is set to one?

d. To what does the system degenerate if the number of pages in the free-frame pool is zero?

**10.29** Consider a demand-paging system with the following time-measured utilizations:

| CPU utilization | 20% |
|---|---|
| Paging disk | 97.7% |
| Other I/O devices | 5% |

For each of the following, indicate whether it will (or is likely to) improve CPU utilization. Explain your answers.

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.

**10.30** Explain why minor page faults take less time to resolve than major page faults.

**10.31** Explain why compressed memory is used in operating systems for mobile devices.

**10.32** Suppose that a machine provides instructions that can access memory locations using the one-level indirect addressing scheme. What sequence of page faults is incurred when all of the pages of a program are currently nonresident and the first instruction of the program is an indirect memory-load operation? What happens when the operating system is using a per-process frame allocation technique and only two pages are allocated to this process?

**10.33** Consider the page references:
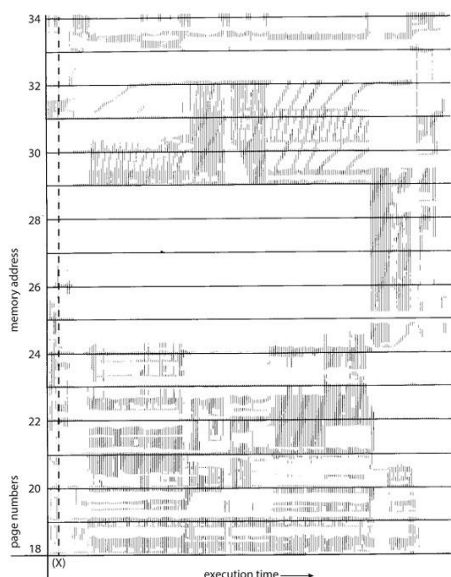
What pages represent the locality at time ($X$)?

**10.34** Suppose that your replacement policy (in a paged system) is to examine each page regularly and to discard that page if it has not been used since the last examination. What would you gain and what would you lose by using this policy rather than LRU or second-chance replacement?

**10.35** A page-replacement algorithm should minimize the number of page faults. We can achieve this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages associated with that frame. Then, to replace a page, we can search for the page frame with the smallest counter.

a. Define a page-replacement algorithm using this basic idea. Specifically address these problems:

- What is the initial value of the counters?

- When are counters increased?

- When are counters decreased?

- How is the page to be replaced selected?

b. How many page faults occur for your algorithm for the following reference string with four page frames?

    1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2.

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?

**10.36** Consider a demand-paging system with a paging disk that has an average access and transfer time of 20 milliseconds. Addresses are translated through a page table in main memory, with an access time of 1 microsecond 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 those remaining, 10 percent (or 2 percent of the total) cause page faults. What is the effective memory access time?

**10.37** What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?

**10.38** Is it possible for a process to have two working sets, one representing data and another representing code? Explain.

**10.39** Consider the parameter Δ used to define the working-set window in the working-set model. When Δ is set to a low value, what is the effect on the page-fault frequency and the number of active (nonsuspended) processes currently executing in the system? What is the effect when Δ is set to a very high value?