Virtual Memory

Practice Exercises

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

Answer:
A page fault occurs when an access to a page that has not been brought into main memory takes place. The operating system verifies the memory access, aborting the program if it is invalid. If it is valid, a free frame is located and I/O is requested to read the needed page into the free frame. Upon completion of I/O, the process table and page table are updated, and the instruction is restarted.

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?

Answer:

a. n

b. p

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

Answer:

Rank	Algorithm	Suffer from Belady's anomaly
1	Optimal	no
2	LRU	no
3	Second-chance	yes
4	FIFO	yes

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.

Answer:

$$\begin{aligned} \text{effective access time} &= 0.99 \times (1\ \mu\text{sec} + 0.008 \times (2\ \mu\text{sec}) \\ &\quad + 0.002 \times (10{,}000\ \mu\text{sec} + 1{,}000\ \mu\text{sec}) \\ &\quad + 0.001 \times (10{,}000\ \mu\text{sec} + 1{,}000\ \mu\text{sec}) \\ &= (0.99 + 0.016 + 22.0 + 11.0)\ \mu\text{sec} \\ &= 34.0\ \mu\text{sec} \end{aligned}$$

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

Answer:

- $9EF \rightarrow 0EF$
- $111 \rightarrow 211$
- $700 \rightarrow D00$
- $0FF \rightarrow EFF$

10.6 Discuss the hardware functions required to support demand paging.

Answer:

For every memory-access operation, the page table must be consulted to check whether the corresponding page is resident and whether the program has read or write privileges for accessing the page. These checks must be performed in hardware. A TLB could serve as a cache and improve the performance of the lookup operation.

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;
```

Answer:

a. 5,000

b. 50

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

Answer:

Number of frames	LRU	FIFO	Optimal
1	20	20	20
2	18	18	15
3	15	16	11
4	10	14	8
5	8	10	7
6	7	10	7
7	7	7	7

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

Answer:

- 18
- 17
- 13

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.

Answer:
You can use the valid/invalid bit supported in hardware to simulate the reference bit. Initially set the bit to invalid. On first reference, a trap to the operating system is generated. The operating system will set a software bit to 1 and reset the valid/invalid bit to valid.

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.

Answer:
No. An optimal algorithm will not suffer from Belady's anomaly because—by definition—an optimal algorithm replaces the page that will not be used for the longest time. Belady's anomaly occurs when a page-replacement algorithm evicts a page that will be needed in the immediate future. An optimal algorithm would not have selected such a page.

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.

Answer:

a. **FIFO**. Find the first segment large enough to accommodate the incoming segment. If relocation is not possible and no one segment is large enough, select a combination of segments whose memories are contiguous, which are "closest to the first of the list,"

and which can accommodate the new segment. If relocation is possible, rearrange the memory so that the first *N* segments large enough for the incoming segment are contiguous in memory. Add any leftover space to the free-space list in both cases.

b. **LRU**. Select the segment that has not been used for the longest time and that is large enough, adding any leftover space to the free-space list. If no one segment is large enough, and if relocation is not available, select a combination of the "oldest" segments that are contiguous in memory and are large enough. If relocation is available, rearrange the oldest *N* segments to be contiguous in memory and replace those with the new segment.

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

Answer:

a. Thrashing is occurring.

b. CPU utilization is sufficiently high to leave things alone and increase the degree of multiprogramming.

c. Increase the degree of multiprogramming.

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?

Answer:
The page table can be set up to simulate base and limit registers provided that the memory is allocated in fixed-size segments. The base of a segment can be entered into the page table and the valid/invalid bit used to indicate that portion of the segment as resident in the memory. There will be some problem with internal fragmentation.

Exercises

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

Answer:

- TLB miss with no page fault: page has been brought into memory but has been removed from the TLB.
- TLB miss with page fault: page fault has occurred.
- TLB hit with no page fault: page is in memory and in the TLB (most likely a recent reference).
- TLB hit with page fault cannot occur. The TLB is a cache of the page table. If an entry is not in the page table, it will not be in the TLB.

10.16 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.

Answer:

a. The page-fault rate is initially quite high, as needed pages are not yet loaded into memory.

b. Once all necessary pages are loaded into memory, the page-fault rate should be quite low.

c. (1) Ignore it. (2) Get more physical memory. (3) Reclaim pages more aggressively to address the high page-fault rate.

10.17 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`

Answer:

- `0x2A1` → `0xAA1`
- `0x4E6` → `0x9E6`
- `0x94A` → `0x14A`
- `0x316` → `0xF16`

10.18 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?

Answer:

- `0x621C` → `0x821C`
- `0xF0A3` → `0x20A3`

- 0xBC1A → 0x4C1A
- 0x5BAA → 0xDBAA
- 0x0BA1 → 0x9BA1

- The only choices are pages 1, 9, and 14. Thus, example addresses include anything that begins with the hexadecimal sequence 0x1..., 0x9..., and 0xE....
- Any page table entries that have a reference bit of zero. This includes the following page frames: {10, 15, 6, 12, 7, 5, 1, 0}

10.19 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.

Answer:

- FIFO = 18, LRU = 17, OPT = 13
- FIFO = 16, LRU = 19, OPT = 13
- FIFO = 15, LRU = 16, OPT = 11
- FIFO = 20, LRU = 20, OPT = 16
- FIFO = 12, LRU = 11, OPT = 11

10.20 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.

Answer:
Consider the following sequence of memory accesses in a system that can hold four pages in memory: 1 1 2 3 4 5 1. When page 5 is accessed, the least frequently used page-replacement algorithm will replace a page other than 1 and therefore will not incur a page fault when page 1 is accessed again. For the sequence "1 2 3 4 5 2," the least recently used algorithm performs better.

10.21 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?

Answer:

a. When a page fault occurs and 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.

b. When a page fault occurs and 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.

c. When the number of resident pages is set to one, then the system degenerates to the page-replacement algorithm used in the free-frame pool, which is typically LRU.

d. When the number of pages in the free-frame pool is zero, then the system degenerates to a FIFO page-replacement algorithm.

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

Answer: Mobile devices do not support paging, and compressed memory is an alternative to paging. When memory runs low, rather than swap pages to swap space, a number of pages are compressed into one page. For example, three pages may be compressed into one page. If we do this three times, we free up six pages of memory.

10.23 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?

Answer:
Such an algorithm could be implemented through the use of a reference bit. After every examination, the bit is set to zero; it is set back to one if the page is referenced. The algorithm then selects an arbitrary page for replacement from the set of unused pages.

The advantage of this algorithm is its simplicity—nothing other than a reference bit need be maintained. The disadvantage is that it ignores locality by using only a short time frame for determining whether to evict a page. For example, a page may be part of the working set of a process but may be evicted because it has not been referenced since the last examination (that is, not all pages in the working set may be referenced between examinations).

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

Answer:
Yes. In fact, many processors provide two TLBs for this very reason. As an example, the code being accessed by a process may retain the same working set for a long time. However, the data the code accesses may change, thus reflecting a change in the working set for data accesses.

10.25 In a 1,024-KB segment, memory is allocated using the buddy system. Using Figure 10.26 as a guide, draw a tree illustrating how the following memory requests are allocated:

- Request 5-KB
- Request 135 KB.
- Request 14 KB.
- Request 3 KB.
- Request 12 KB.

Next, modify the tree for the following releases of memory. Perform coalescing whenever possible:

- Release 3 KB.
- Release 5 KB.
- Release 14 KB.
- Release 12 KB.

Answer:
The following allocations are made by the Buddy system: The 5-KB request is assigned an 8-KB segment. The 135-KB request is assigned a 256-KB segment. The 14-KB request is assigned a 16-KB segment. The 3-KB request is assigned a 4-KB segment. The 12-KB request is assigned a 16-KB segment. After the allocation, the following segment sizes are available: 512 KB, 128 KB, 64 KB, 16 KB, and 4 KB.

After the releases of memory, the only segment in use would be a 256-KB segment containing 135 KB. After coalescing, there would be a 256-KB segment and a 512-KB segment available.

10.26 The slab-allocation algorithm uses a separate cache for each different object type. Assuming there is one cache per object type, explain why this scheme doesn't scale well with multiple CPUs. What could be done to address this scalability issue?

Answer:
This has long been a problem with the slab allocator—poor scalability with multiple CPUs. The issue comes from having to lock the global cache when it is being accessed. This has the effect of serializing cache accesses on multiprocessor systems. Solaris has addressed this problem by introducing a per-CPU cache rather than a single global cache.