Chris Banci
CS433
5/4/17

Written Homework 5

8.11) Given six memory partitions of 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)? Rank the algorithms in terms of how efficiently they use memory.

Best-fit algorithm:

115 KB = (300 KB, 600 KB, 350 KB, 200 KB, 750 KB, 10 KB)
500 KB = (300 KB, 100 KB, 350 KB, 200 KB, 750 KB, 10 KB)
358 KB = (300 KB, 100 KB, 350 KB, 200 KB, 392 KB, 10 KB)
200 KB = (300 KB, 100 KB, 350 KB, 0 KB, 392 KB, 10 KB)
375 KB = (300 KB, 100 KB, 350 KB, 0 KB, 17 KB, 10 KB)

First-fit algorithm:

115 KB = (185 KB, 600 KB, 350 KB, 200 KB, 750 KB, 125 KB)
500 KB = (185 KB, 100 KB, 350 KB, 200 KB, 750 KB, 125 KB)
358 KB = (185 KB, 100 KB, 350 KB, 200 KB, 392 KB, 125 KB)
200 KB = (185 KB, 100 KB, 150 KB, 200 KB, 392 KB, 125 KB)
375 KB = (185 KB, 100 KB, 150 KB, 200 KB, 17 KB, 125 KB)

Worst-fit algorithm:

115 KB = (300 KB, 600 KB, 350 KB, 200 KB, 635 KB, 125 KB)
500 KB = (300 KB, 600 KB, 350 KB, 200 KB, 135 KB, 125 KB)
358 KB = (300 KB, 242 KB, 350 KB, 200 KB, 135 KB, 125 KB)
200 KB = (300 KB, 242 KB, 150 KB, 200 KB, 135 KB, 125 KB)
375 KB = process must wait

Ranking:

1. Best-Fit algorithm
2. First-Fit algorithm
3. Worst-fit algorithm

8.13) Compare the memory organization schemes of contiguous memory allocation, pure segmentation, and pure paging with respect to the following issues:

a. External fragmentation
b. Internal fragmentation
c. Ability to share code across processes

Continuous memory allocation

- A. Affected by external fragmentation since space is continuously allocated.
- B. Not affected by internal fragmentation.
- C. Does not allow processes to share code.

Pure segmentation

- A. Affected by external fragmentation.
- B. Not affected by internal fragmentation.
- C. Allows processes to share code.

Pure paging

- A. Not affected by external fragmentation.
- B. Affected by internal fragmentation.
- C. Allows processes to share code.

8.20) Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

a. 3085 =

Page # (3085 / 1024) = 3.14
Offset (3085 % 1024) = 13

b. 42095

Page # (42095 / 1024) = 41.11
Offset (42095 % 1024) = 111

c. 215201

Page # (215201 / 1024) = 210.16
Offset (215201 % 1024) = 161

d. 650000

Page # (650000 / 1024) = 634.77
Offset (650000 % 1024) = 784

e. 2000001

Page # (2000001 / 1024) = 1953.13
Offset (2000001 % 1024) = 129

8.23) Consider a logical address space of 256 pages with a 4-KB page size, mapped onto a physical memory of 64 frames.

a. How many bits are required in the logical address?

256 pages * 4096 bytes = 1048576 bytes of logical memory.

- 1048576 = 2^20.
- 20 bits are required in the logical address.

b. How many bits are required in the physical address?

64 frames * 4096 bytes = 262144 bytes of physical memory

- 262144 = 2^18

- 18 bits are required in the physical address.

8.25) Consider a paging system with the page table stored in memory.

a. If a memory reference takes 50 nanoseconds, how long does a paged memory reference take?

- It will take 100 nanoseconds, because it takes 50 nanoseconds to access the page table and another 50 nanoseconds to access the data in memory.

b. If we add TLBs, and 75 percent of all page-table references are found in the TLBs, what is the effective memory reference time? (Assume that finding a page-table entry in the TLBs takes 2 nanoseconds, if the entry is present.)

75% of the time its 52 nanoseconds, 25% of the time its 102 nanoseconds.

- (.75% * (50 + 2)) + *(25% * (100 + 2)) = 64.5ns
- The effective memory reference time is 64.5 nanoseconds

8.28) Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a. 0, 430

219 + 430 = 649

b. 1, 10

2300 + 10 + 2310

c. 2, 500

90 + 500 = 590

illegal reference.

d. 3, 400

1327 + 400 = 1727

e. 4, 112

1952 + 112 = 2064

illegal reference.

9.19) 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?

200ns = ((1-p) * 100ns) + (P(100ns + .30 * 8ms)) + (P(.70 * 20ms))

200ns = 100ns + P(16.4ns)

P = .000006

The maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds is .000006.

9.21) 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

  7, 2, 3, 1, **2**, 5, 3, 4, 6, 7, **7**, 1, 0, 5, 4, 6, 2, 3, 0, 1

  18 page faults will occur

- FIFO replacement

  7, 2, 3, 1, **2**, 5, **3**, 4, 6, 7, **7**, 1, 0, 5, 4, 6, 2, 3, 0, 1

  17 page faults will occur

- Optimal replacement

  7, 2, 3, 1, **2**, 5, **3**, 4, 6, 7, **7**, **1**, 0, **5**, 4, 6, 2, 3, **0**, 1

  13 page faults will occur

9.22) The page table shown in Figure 9.32 is for a system with 16-bit virtual and physical addresses and with 4,096-byte pages. The reference bit 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 the page is not in memory. The page-replacement algorithm is localized LRU, and all numbers are provided in decimal.

Page	Page Frame	Reference Bit
0	9	0
1	1	0
2	14	0
3	10	0
4	–	0
5	13	0
6	8	0
7	15	0
8	–	0
9	0	0
10	5	0
11	4	0
12	–	0
13	–	0
14	3	0
15	2	0

Figure 9.32 Page table for Exercise 9.22.

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.

- 0xE12C
  - Binary = 1110 0001 0010 1100
    - Page = 14
    - Frame = 2
    - Physical Address = 0x212C
- 0x3A9D
  - Binary = 0011 1010 1001 1101
    - Page = 3
    - Frame = 10
    - Physical Address = 0xAA9D
- 0xA9D9
  - Binary = 1010 1001 1101 1001
    - Page = 10
    - Frame = 5
    - Physical Address = 0x59D9
- 0x7001
  - Binary = 0111 0000 0000 0001
    - Page = 7
    - Frame = 15
    - Physical Address = 0xF001
- 0xACA1
  - Binary = 1010 1100 1010 0001
    - Page = 10
    - Frame = 5
    - Physical Address = 0x5CA1

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

Any logical address that starts with 0x4, 0x8, 0x12, 0x13 will result in a page fault because the pages are not in virtual memory.

- Example: 0x49D9

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

Any page entry that has a referenced bit set to 0 such as 0, 1, 4, 8, 9,13,14 can be used to resolve a page fault.

9.27) 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.

No it will not improve cpu utilization because cpu is underutilized.

b. Install a bigger paging disk.

No it will not improve cpu utilization because increasing paging disk increases access time.

c. Increase the degree of multiprogramming.

No it will not improve cpu utilization because less memory will be available for programs and more page faults can occur.

d. Decrease the degree of multiprogramming.

Yes, it is likely to improve cpu utilization because there less number of pages, leading to more execution.

e. Install more main memory.

Yes, it is likely to improve cpu utilization because more frames leads to less excessive paging.

f. Install a faster hard disk or multiple controllers with multiple hard disks.

Yes, it is likely to improve cpu utilization because cpu will able to get data more quickly.

g. Add prepaging to the page-fetch algorithms.

No, it will not improve utilization because bottleneck is between disk and memory.

h. Increase the page size.

No, it will not improve cpu utilization because each process will have fewer frames, leading to more page faults.