Jeffrey Everett
CSCI 3753
4/12/2017

Problem Set 4

Question 1:

Three approaches to solving the external fragmentation problem are:

1. Periodically defragment/compact the data. This has the disadvantage of being time-consuming, and the processor can do no other useful tasks while this task is being undertaken.
2. Processes divide themselves into logical components called segments. These segments need not be contiguous, so the chunks are smaller on average; this reduces the fragmentation problem slightly.
3. Partition memory into chunks of equal sizes, called pages. Then allocate the necessary pages or swap in the necessary pages when the process must run. This solves the external fragmentation problem because there are clear-cut allocation sizes and pages may be stored in non-contiguous locations. This solution has internal fragmentation, however.

Question 2:

A. 2200 byte block
B. 1000 byte block
C. 2200 byte block
D. 2200 byte block
E. 2200 byte block
F. 2200 byte block

Question 3:

Page table for P1:

Logical Address	Physical Address
0	0
1	1
2	2
3	3
4	4
5	5

6	6

Page table for P2:

Logical Address	Physical Address
0	7
1	8
2	4
3	5

X is stored in physical address 4 (P1 logical address 4 and P2 logical address 2).
Y is stored in physical address 5 (P1 logical address 5 and P2 logical address 3).
The page table for P1 is stored in physical addresses 9 and 10.
The page table for P2 is stored in physical address 11.
Frames 12, 13, 14, and 15 are left unused.

Question 4:
in general:
average memory access time = p_TLB*T + (1-p_TLB)*(1-p)*M + (1-p_TLB)*p*D

with substituted-in values:
average memory access time = 0.90*1ns + 0.10*0.999*10ns + 0.10*0.001*10ms = 1.00 μs

Question 5:
LRU cannot suffer from Belady's Anomaly because increasing the number of page frames only increases the number of least-recently used pages that are stored. Thus, adding page frames allows additional pages to be stored, but the pages that would have been stored if there were fewer frames are always present in the set of pages stored by the larger frame.

Consider the reference string "1 2 3 4 3 2 1". If we have two page frames, then there will be faults on 1, then 2, then 3, then 4, then 2, then 1. If we have three page frames, there will be faults on 1, then 2, then 3, then 4, then 1. If we have four page frames, there will be faults on 1, then 2, then 3, then 4. Clearly, adding page frames does not increase the number of page faults in this example.

Question 6:
FIFO: (bold denotes next page to be replaced upon page fault)

- 3 - [**3**, -, -] - page fault
- 2 - [**3**, 2, -] - page fault
- 4 - [**3**, 2, 4] - page fault
- 3 - [**3**, 2, 4]

- 4 - [**3**, 2, 4]
- 2 - [**3**, 2, 4]
- 2 - [**3**, 2, 4]
- 3 - [**3**, 2, 4]
- 4 - [**3**, 2, 4]
- 5 - [5, **2**, 4] - page fault
- 6 - [5, 6, **4**] - page fault
- 7 - [**5**, 6, 7] - page fault
- 7 - [**5**, 6, 7]
- 6 - [**5**, 6, 7]
- 5 - [**5**, 6, 7]
- 4 - [4, **6**, 7] - page fault
- 5 - [4, 5, **7**] - page fault
- 6 - [**4**, 5, 6] - page fault
- 7 - [7, **5**, 6] - page fault
- 2 - [7, 2, **6**] - page fault
- 1 - [**7**, 2, 1] - page fault

12 page faults total

OPT:

- 3 - [3, -, -] - page fault
- 2 - [3, 2, -] - page fault
- 4 - [3, 2, 4] - page fault
- 3 - [3, 2, 4]
- 4 - [3, 2, 4]
- 2 - [3, 2, 4]
- 2 - [3, 2, 4]
- 3 - [3, 2, 4]
- 4 - [3, 2, 4]
- 5 - [5, 2, 4] - page fault
- 6 - [5, 6, 4] - page fault
- 7 - [5, 6, 7] - page fault
- 7 - [5, 6, 7]
- 6 - [5, 6, 7]
- 5 - [5, 6, 7]
- 4 - [5, 6, 4] - page fault
- 5 - [5, 6, 4]
- 6 - [5, 6, 4]
- 7 - [7, 6, 4] - page fault
- 2 - [2, 6, 4] - page fault
- 1 - [1, 6, 4] - page fault

10 page faults total

LRU: (bold denotes next page to be replaced upon page fault)

- 3 - [**3**, -, -] - page fault

- 2 - [**3**, 2, -] - page fault
- 4 - [**3**, 2, 4] - page fault
- 3 - [3, **2**, 4]
- 4 - [3, **2**, 4]
- 2 - [**3**, 2, 4]
- 2 - [**3**, 2, 4]
- 3 - [3, 2, **4**]
- 4 - [3, **2**, 4]
- 5 - [**3**, 5, 4] - page fault
- 6 - [6, 5, **4**] - page fault
- 7 - [6, **5**, 7] - page fault
- 7 - [6, **5**, 7]
- 6 - [6, **5**, 7]
- 5 - [6, 5, **7**]
- 4 - [**6**, 5, 4] - page fault
- 5 - [**6**, 5, 4]
- 6 - [6, 5, **4**]
- 7 - [6, **5**, 7] - page fault
- 2 - [**6**, 2, 7] - page fault
- 1 - [1, 2, **7**] - page fault

10 page faults total

For this reference string, OPT and LRU are tied for having the fewest page faults (10).

Question 7:

Working set algorithm: (assume there are still three frames, evict first encountered page not in set if such a page exists, and evict first encountered page in set otherwise)

- 3 - Working set: {3}, Frame allocation: [3, -, -], page fault
- 2 - Working set: {2,3}, Frame allocation: [3, 2, -], page fault
- 4 - Working set: {2,3,4}, Frame allocation: [3, 2, 4], page fault
- 3 - Working set: {2,3,4}, Frame allocation: [3, 2, 4]
- 4 - Working set: {2,3,4}, Frame allocation: [3, 2, 4]
- 2 - Working set: {2,3,4}, Frame allocation: [3, 2, 4]
- 2 - Working set: {2,3,4}, Frame allocation: [3, 2, 4]
- 3 - Working set: {2,3,4}, Frame allocation: [3, 2, 4]
- 4 - Working set: {2,3,4}, Frame allocation: [3, 2, 4]
- 5 - Working set: {2,3,4,5}, Frame allocation: [5, 2, 4], page fault
- 6 - Working set: {2,3,4,5,6}, Frame allocation: [6, 2, 4], page fault
- 7 - Working set: {2,3,4,5,6,7}, Frame allocation: [7, 2, 4], page fault
- 7 - Working set: {3,4,5,6,7}, Frame allocation: [7, 2, 4]
- 6 - Working set: {4,5,6,7}, Frame allocation: [7, 6, 4], page fault
- 5 - Working set: {5,6,7}, Frame allocation: [7, 6, 5], page fault
- 4 - Working set: {4,5,6,7}, Frame allocation: [4, 6, 5], page fault
- 5 - Working set: {4,5,6,7}, Frame allocation: [4, 6, 5]
- 6 - Working set: {4,5,6,7}, Frame allocation: [4, 6, 5]

- 7 - Working set: {4,5,6,7}, Frame allocation: [7, 6, 5], page fault
- 2 - Working set: {2,4,5,6,7}, Frame allocation: [2, 6, 5], page fault
- 1 - Working set: {1,2,4,5,6,7}, Frame allocation: [1, 6, 5], page fault

12 page faults - same as FIFO, more than OPT and LRU