Andrew Candelaresi

OS Problem Set

1.
compaction:
The goal is to shuffle the memory contents so as to place all free memory together in one large block. Compaction is not always possible, however. If relocation is static and is done at assembly or load time, compaction cannot be done.

Segmentation:
divide a process into variably sized segments that are organized according to some logical criteria. A process has code, data, stack, and heap. Each of these could be a separate segment. A process could subdivide its cod into functional segments, e.g. a web server could divide its functional components into a networking segment. Need a segmentation table to keep track of where each segment is mapped in main memory. Segments are variable in size; each entry of the segment table needs a base and a limit field to place in the base and limit registers.

Paging:
Segmentation permits the physical address space of a process to be noncontiguous. Paging is another memory-management scheme that offers this advantage. However, paging avoids external fragmentation and the need for compaction, whereas segmentation does not. It also solves the considerable problem of fitting memory chunks of varying sizes onto the backing store.

2.
a. 2200
b.1000
c. 2200
d. 2200
e. 2200
f. 2200

3. P1 = 7 pages, P2 = 4 pages
Page Table P1

page #:	frame #:
0	6
1	4
2	11
3	5
4	13
5	15
6	9

Page Table P2

page #:	frame #:
0	6
1	4
2	0
3	10

frames in Mem	process#: Page #:
f0	p2:2
f1	
f2	
f3	
f4	p1:1 & p2:1
f5	p1:5
f6	p1:0 & p2:0
f7	
f8	
f9	p1:6
f10	p2:3
f11	p1:2
f12	
f13	p1:4
f14	
f15	p1:5

4.
Avg access time = (p_TLB * T) + ((1-p+TLB)*p * D)+((1-p_PTLB)*(1-p)xM)
Avg acces time = (.9 × 1ns) + (.001 * .1* 10ms) + (.1 *.999*10ns)= 1001.899

5. LRU does not suffer from Belady's anomaly because it replaces pages based on their frequency of use. So that it will not pull out a page that is being used often, just because it has been in memory for along time.

Ex. calculating page fault with LRU

frame allocation of 3

0 2 3 4 6 4 8 4 6 5 4 4 5 8 7
0!2!3!4!6!48!4 6 5!4 4 5 8!7!
0 2 3 4 6 4 8 8 6 5 5 4 4 8
0 2 3 3 6 6 4 4 6 6 6 5 5

frame allocation 6

0	2	3	4	6	4	8	4	6	5	4	4	5	8	7
0!	2!	3!	4!	6!	6	8!	8	8	5!	5	5	5	5	7!
	0	2	3	4	4	6	6	6	8	8	8	8	8	5
		0	2	3	3	4	4	4	6	6	6	6	6	8
			0	2	2	3	3	3	4	4	4	4	4	6
				0	0	2	2	2	3	3	3	3	3	4

						0	0	0	2	2	2	2	2	3

there is one less page fault in the larger frame allocation

6.
3,2,4,3,4,2,2,3,4,5,6,7,7,6,5,4,5,6,7,2,1
FIFO
12 page faults

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

LRU
10 page faults

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

OPT
3,2,4,3,4,2,2,3,4,5,6,7,7,6,5,4,5,6,7,2,1
10 page faults

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

7.

{3245}
{3 *2* 4* 3 {4 {2 2} 3 4 5*} 6 7 7 6 5 4 5 6 7 2 1
{324}
{324567}
{3 2 4 3 4 {2 {2 3 4 5 6*} 7*} 7 6 5 4 5 6 7 2 1}
{32456}
{324567}
{3 2 4 3 4 2 2 3 4 5 6 7 7 6 {5 {4 5 6 7 2 }1}}
{324567}
6 page faults