CIS450/ECE478, Solutions for homework #2

1. Answer:

$2^{10} = 1024$, $2^5 = 32$, $2^7 = 128$

a. Logical address: 10 + 5 = 15 bits

b. Physical address: 10 + 7 = 17 bits

2. Answer:

Address	4KB Page		8KB Page	
	Page#	Offset	Page#	Offset
20000	4	3616	2	3616
32768	8	0	4	0
60000	14	2656	7	2656
90000	21	3984	10	8080

3. Answer:

a. One Level Paging

Needs 12 bits for page offset and 20 bits for page number.

The virtual address space for the process is very sparse. However, since the program text is in the bottom 1024 pages and the stack is in the top 1024 pages, we need to have a full page table with 2^{20} entries (though all the page entries in the middle are invalid).

b. Two- Level Paging

We need to have one page table in the first level and three page tables in the second level. The number of entries in each page table is $2^{10} = 1024$. So, we need 4 * 1024 = 4096 page entries.

4. Answer:

a. 219 + 430 = 649

b. 2300 + 10 = 2310

c. 500 > 100 (length), illegal reference (segmentation fault), trap to operating system

d. 1327 + 400 = 1727

e. 112 > 96 (length), illegal reference (segmentation fault), trap to operating system

5. Answer:

The system obviously is spending most of its time paging, indicating over allocation of memory. If the level of multiprogramming is reduced, resident processes would page fault less frequently and the CPU utilization would improve. Another way to improve performance would be to get more physical memory or a faster paging disk.

a. Get a faster CPU—No.
b. Get a bigger paging disk—No.
c. Increase the degree of multiprogramming—No.
d. Decrease the degree of multiprogramming—Yes.
e. Install more main memory—Likely to improve CPU utilization as more pages can remain resident and not require paging to or from the disks.
f. Install a faster hard disk—Also an improvement, for as the disk bottleneck is removed by faster response and more throughput to the disks, the CPU will get more data more quickly.
g. Add prepaging to the page fetch algorithms—Again, the CPU will get more data faster, so it will be more in use. This is only the case if the paging action is amenable to prefetching (i.e., some of the access is sequential).
h. Increase the page size—Increasing the page size will result in fewer page faults if data is being accessed sequentially. If data access is more or less random, more paging action could ensue because fewer pages can be kept in memory and more data is transferred per page fault. So this change is as likely to decrease CPU utilization as it is to increase it.

6. 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

7. Answer:

a) What is the effective memory access time

```
EAT = Hit Time + Miss Rate * Miss Penalty
EAT = 100ns + 0.001 * 10ms = 100ns + 0.001 * 10000000ns = 10100ns
```

b) In order to achieve no more than 5 percent overhead due to demand paging for this system, what should the maximum page fault rate be?

```
105ns = 100ns + p * 10000000ns
5ns = p * 10000000ns
p = 1/2000000
```

8. Answer:

Contiguous Allocation

- Sequential access- Works very well as the file is stored contiguously. Sequential access simply involves traversing the contiguous disk blocks.
- Random access -Works very well as you can easily determine the adjacent disk block containing the position you wish to seek to.

Linked Allocation

- Sequential access - Satisfactory as you are simply following the links from one block to the next.
- Random access - Poor as it may require following the links to several disk blocks until you arrive at the intended seek point of the file.

Indexed Allocation

- Sequential access - Works well as sequential access simply involves sequentially accessing each index.
- Random access - Works well as it is easy to determine the index associated with the disk block containing the position you wish to seek to.

9. Answer:

block #	Number of disk access
1	1
10	1
200	2
500	3
1,000	3
2,200	3
10,000	3
100,000	4
1,000,000	4
1,400,000	4
5,000,000	4