**CSCE 3613 Operating Systems Homework #5, ver. 4.3 Memory Management and Virtual Memory**

* Show all your work. Without proper justification and details of steps, correct answers alone may not carry full credit
* Minus 2 points if you present the worked problems out of order. In other words, please present the problems in the order assigned, 1, 2, 3, ...

1. Compare the main memory organization schemes of contiguous-memory allocation, pure segmentation, and pure paging with respect to the following issues:

Contiguous memory allocation Segmentation Pure Paging

|  |  |  |
| --- | --- | --- |
| Resident Operating System | Empty | Page 4 |
| Process 1 | Segment 0 | Empty |
| Process 2 | Empty | Page 0 |
| Process 3 | Segment 3 | Empty |
| Empty | Segment 2 | Page 3 |
|  | Segment 4 | Page 1 |
|  | Free | Empty |
|  |  | Empty |

a. (2 pts.) External fragmentation

Contiguous -Yes

Segmentation – Yes

Paging - No

b. (2 pts.) Internal fragmentation

Contiguous – No

Segmentation – No

Paging - Yes

c. (2 pts.) Ability to share code across processes

Contiguous – No

Segmentation – Yes

Paging - Yes

2. Consider a paging system with the page table stored in memory.  
a. (2 pts.) If a memory reference takes 250 nanoseconds, how long does a paged memory reference take?

Time to look in table = 250 ns

Time to get page = 250 ns

Total time = 500 ns

b. (2 pts.) If we add TLBs, and 80 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 10 nanoseconds, if the entry is there.

EAT = 0.8(10+250)+(0.2)(10+250+250) = 310 ns

3. 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?

1. (1 pt.) 0, 530 219+530 = 749
2. (1 pt.) 1, 15 2300+15 => Trap because larger than max length
3. (1 pt.) 2, 50 90+50 = 140
4. (1 pt.) 3, 600 1327+600 => Trap, 600 larger than 580
5. (1 pt.) 4, 12 1952+12 = 1964

4. Consider a logical address space of 32 pages with 1,024 words per page, mapped onto a physical memory of 16 frames.

a. (2 pts.) How many bits are required in the logical address?

To label 32 pages (2^5) need 5 bits

To label 1024 words (2^10) need 10 bits

5+10 = 15 bits

b. (2 pts.) How many bits are required in the physical address?

To label 16 frames (2^4) need 4 bits

To label 1024 words (2^10) need 10 bits

1 frame contains 1 page

4+10 = 14 bits

5. 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. In addition, in this system arrays are stored in memory in the following order: A[0][0], A[1][0], ... 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, using LRU replacement and assuming that page frame 1 contains the process and the other two are initially empty?

a. (2 pts.)

for(int j=0; j<100; j++){

for(int i=0; i<100; i++){

A[i][j]=0; } }

One page fault every 200 assignments

10,000 assignments total

50 Page Faults

b. (2 pts.)

for(int i=0; i<100; i++){

for(int j=0; j<100; j++){

A[i][j]=0; } }

One page fault every 2 assignments

10,000 assignments total

5,000 Page Faults

6. A certain computer provides its users with a virtual memory space of 2^32 bytes. The computer has 2^18 bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4,096 bytes. A user process generates the virtual address 11123456 hexadecimal.

a. (2 pts.) How many entries are there in the page table?

(2^32 bytes of virtual memory) / (2^12 Bytes/Page) = 2^20 Pages

b. (2 pts.) Explain how the system establishes the corresponding physical location.

Convert to binary

0001 0001 0001 0010 0011 0100 0101 0110

20 bits to ID Page 12 Bits for offset

Use 20 bits to point to correct page

See if in memory, if not load it

Use remaining 12 bits to locate memory location (offset) in page