50.002 COMPUTATIONAL STRUCTURES

INFORMATION SYSTEMS TECHNOLOGY AND DESIGN

Problem Set 12

1 Problem 1

Consider a virtual memory system that uses a single-level page map to translate virtual addresses into physical addresses. Each of the questions below asks you to consider what happens when one of the design parameters of the original system is changed.

1. If the physical memory size (in bytes) is doubled, how does the number of bits in each entry of the page table change?
   Solution:
   Increases by 1 bit. Assuming the page size remains the same, there are now twice as many physical pages, so the physical page number needs to expand by 1 bit.

2. If the physical memory size (in bytes) is doubled, how does the number of entries in the page map change?
   Solution:
   No change. The number of entries in the page table is determined by the size of the virtual address and the size of a page – it's not affected by the size of physical memory.

3. If the virtual memory size (in bytes) is doubled, how does the number of bits in each entry of the page table change?
   Solution:
   No change. The number of bits in a page table entry is determined by the number of control bits (usually 2: dirty and resident) and the number of physical pages – the size of each entry is not affected by the size of virtual memory.

4. If the virtual memory size (in bytes) is doubled, how does the number of entries in the page map change?
   Solution:
   The number of entries doubles. Assuming the page size remains the same, there are now twice as many virtual pages and so there needs to be twice as many entries in the page map.

Virtual page	Valid bit	Physical page
0	0	7
1	1	9
2	0	3
3	1	2
4	1	5
5	0	5
6	0	4
7	1	1

Table 1

5. If the page size (in bytes) is doubled, how does the number of bits in each entry of the page table change?
   Solution:
   Each entry is one bit smaller. Doubling the page size while maintaining the size of physical memory means there are half as many physical pages as before. So the size of the physical page number field decreases by one bit.

6. If the page size (in bytes) is doubled, how does the number of entries in the page map change?
   Solution:
   There are half as many entries. Doubling the page size while maintaining the size of virtual memory means there are half as many virtual pages as before. So the number of page table entries is also cut in half.

7. The following table shows the first 8 entries in the page map. Recall that the valid bit is 1 if the page is resident in physical memory and 0 if the page is on disk or hasn't been allocated.

   If there are 1024 (2^{10}) bytes per page, what is the physical address corresponding to the decimal virtual address 3956?

   Solution:
   3956 = 0xF74. So the virtual page number is 3 with a page offset of 0x374. Looking up page table entry for virtual page 3, we see that the page is resident in memory (valid bit = 1) and lives in physical page 2. So the corresponding physical address is (2<<10)+0x374 = 0xB74 = 2932.

2 Problem 2

Consider two possible page-replacement strategies: LRU (the least recently used page is replaced) and FIFO (the page that has been in the memory longest is replaced). The merit of a page-replacement strategy is judged by its hit ratio.

Assume that, after space has been reserved for the page table, the interrupt service routines, and the operating-system kernel, there is only sufficient room left in the main memory for *four* user-program pages. Assume also that initially virtual pages 1, 2, 3, and 4 of the user program are brought into physical memory in that order.

1. For each of the two strategies, what pages will be in the memory at the end of the following sequence of virtual page accesses? Read the sequence from left to right: (6, 3, 2, 8, 4).
   Solution:

```
LRU:
start: 4 3 2 1
access 6: replace 1 => 6 4 3 2
access 3: reorder list => 3 6 4 2
access 2: reorder list => 2 3 6 4
access 8: replace 4 => 8 2 3 6
access 4: replace 6 => 4 8 2 3

FIFO:
start: 4 3 2 1
access 6: replace 1 => 6 4 3 2
access 3: no change => 6 4 3 2
access 2: no change => 6 4 3 2
access 8: replace 2 => 8 6 4 3
access 4: no change => 8 6 4 3
```

2. Which (if either) replacement strategy will work best when the machine accesses pages in the following (stack) order: (3, 4, 5, 6, 7, 6, 5, 4, 3, 4, 5, 6, 7, 6, ...)?
   Solution:
   LRU misses on pages 3 & 7 $\rightarrow$ 2/8 miss rate. FIFO doesn't work well on stack accesses $\rightarrow$ 5/8 miss rate.

3. Which (if either) replacement strategy will work best when the machine accesses pages in the following (repeated sequence) order: (3, 4, 5, 6, 7, 3, 4, 5, 6, 7, ...).
   Solution:
   Both strategies have a 100% miss rate in the steady state.

4. Which (if either) replacement strategy will work best when the machine accesses pages in a randomly selected order, such as (3, 4, 2, 8, 7, 2, 5, 6, 3, 4, 8, ...).
   Solution:
   Neither FIFO nor LRU is guaranteed to be the better strategy in dealing with random accesses since there is no locality to the reference stream.