CS 537: Intro to Operating Systems (Fall 2017)

Worksheet 6 - Memory Virtualization III

Due: Oct 18th 2017 (Wed) in-class OR email Simmi before 11:59 pm

1. **Segmentation & Paging**

   In this question, we consider address translation in a system which uses a **hybrid of segmentation and paging** for memory management. There are **three segments** namely, code, heap, and stack. The **two higher-order bits** (MSBs) in the virtual address are used to identify the segment. 00 for code, 01 for heap, and 11 for stack.

   Parameters and Assumptions:

   - Size of virtual address space = 1 KB
   - Page size = 32 bytes
   - Size of physical memory = 4 KB
   - Size of one Page Table Entry (PTE) = 4 bytes
   - The virtual pages with VPNs 6, 7, 8, 9, 10, and 31 are the ONLY valid pages.

   Answer the following questions based on the parameters and assumptions described above.

   (a) Number of **bits** needed for the Virtual Page Number (**VPN**): ————————

   (b) Number of **valid virtual pages** in the **code** segment: ————————

   (c) Value of the **bounds register** for the **heap** segment: ————————

   (d) Number of **PTEs** in the **stack** segment's **page table**: ————————

   (e) **Total size** of **ALL** the **page tables** used in this system: ————————

2. **Swapping Mechanisms**

In this question you will examine virtual memory reference traces. An access can be a TLB Hit or a TLB Miss; if it is a TLB miss, the reference can be a page hit (page present in physical memory) or a page fault (page not present in physical memory).

Assume a **TLB with 2 entries** and a **memory that can hold 4 pages**. Assume the TLB and memory are initially **empty**. Finally, assume **LRU replacement** is used for both TLB and memory.

Below each virtual memory reference, **mark** if the reference is a:

- TLB Hit (**H**), or
- TLB Miss followed by a page hit (**M**), or
- TLB Miss followed by a page fault (**F**)

Also, write the **contents** of the TLB and the Memory at the **end** of each virtual memory trace.

Virtual Memory Reference	**TLB**	**Memory**
0, 1, 2, 3, 4, 5, 6, 7		
0, 1, 2, 3, 0, 1, 2, 3		
0, 1, 2, 3, 4, 0, 1, 2		
3, 7, 3, 7, 1, 3, 1, 7		

3. **Swapping Policies**

(a) Consider the following request sequence of virtual pages

0, 1, 2, 3, 0, 1, 4, 0, 1, 2, 3, 4

For each replacement policy below, give the number of hits and the virtual page numbers remaining in physical memory at the end of this request sequence.

The number of physical frames = 3

Policy	**# of Hits**	**VPNs remaining at end**
OPT		
FIFO		
LRU		

(b) Write a **sequence of 10 virtual page requests** that has a **hit rate of zero** with **LRU** page replacement policy.

Virtual Pages available: 0, 1, 2, 3, 4
The number of physical frames = 4

(c) Consider the following page requests:

3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4.

The page replacement policy used is **FIFO**.

i. What is the number of **page faults** with a cache size of **three**?

ii. What is the number of **page faults** with a cache size of **four**?

iii. What is the name of this anomaly?

4. Consider the **clock algorithm** for approximating LRU with the following assumptions:

   - The number of physical frames in the cache is **four**.
   - The physical frames are numbered 0, 1, 2, and 3 in the clockwise direction.
   - The cache is initially empty.
   - The reference/use bit for all the frames is initially 0.
   - When a page is accessed, the hardware sets the use bit of the physical frame corresponding to the page to 1.
   - The clock hand initially points to the physical frame 0.
   - The clock hand always moves in the **clock-wise direction**.
   - When a page fault occurs, the operating system checks the physical frame pointed to by the clock hand.
     - If the use **bit = 0**, the OS uses this physical frame to place the page.
     - If the use **bit = 1**, the OS clears the use bit to 0 for this physical frame and advances the clock hand to point to the next frame.
   - After a page is placed in a physical frame due to a page fault, the use bit of the frame is set to 1 and the clock hand advances to the next physical frame.
   - When the hardware references a page that is already found in the cache (i.e., page hit), the clock hand doesn't move.
   - On each page fault, the clock hand resumes from it's **previous position**.

   Consider the following sequence of page requests:

   0, 1, 3, 6, 2, 4, 5, 2, 5, 0, 3, 1, 2, 5, 4, 1, 0

   (a) What are the contents (page numbers and use bits) corresponding to the 4 physical frames at the end after processing all these page requests?

Physical frame #	**Page #**	**Use bit**
0		
1		
2		
3		

   (b) What is the total number of page faults that occur while accessing these pages?