Name _____ Matriculation no. _____ Tutorial no. _____

Karlsruhe Institute of Technology

Operating Systems 2013/14
Assignment 5

Prof. Dr. Frank Bellosa
Dipl.-Inform. Marius Hillenbrand
Dipl.-Inform. Marc Rittinghaus

Submission Deadline: Monday, January 27th, 2014 – 9:30 a.m.

T-Question 5.1: Caches

a. Which of the following statements are correct, which are incorrect? (correctly marked: 0.5P, not marked: 0P, incorrectly marked: -0.5P) **2 T-pt**

correct	incorrect	
☐	☐	Virtually indexed, virtually tagged caches must be flushed when the address space is switched.
☐	☐	Virtually indexed, physically tagged caches must be flushed when the address space is switched.
☐	☐	Virtually indexed, physically tagged caches do not suffer from the alias problem.
☐	☐	Modified cache lines of a physically indexed, physically tagged cache must be written back to main memory immediately after a context switch.

b. Explain, and point out the differences between, the write-allocate and write-to-memory policies! **3 T-pt**

c. What kinds of cache-misses do exist? What can you do to reduce the number of cache misses of each type? **3 T-pt**

Name _____ Matriculation no. _____ Tutorial no. _____

T-Question 5.2: Paging

a. Where does the MMU know the page directory's base address from in IA-32 systems? Where does the OS kernel know the faulting address from when handling a pagefault? (+1 bonus for the latter) **1 T-pt**

b. How many page tables exist when using inverted page tables and single level forward page tables in the operating system, respectively? Briefly explain how you derived your answer. **2 T-pt**

c. Consider a system with 32-bit virtual addresses, a page/frame size of 4kB, and a single-level page table where each entry is 4 bytes in size. Calculate the total memory consumption of the page table. **1 T-pt**

d. Now assume that a 2-level page table hierarchy is used, where each table consists of 1024 4-byte-wide entries. Calculate the maximum memory consumption of the page table hierarchy. **1 T-pt**

Name _____ Matriculation no. _____ Tutorial no. _____

e. Consider a system with three-level page tables and a hardware-controlled TLB. Assume that a TLB access takes 2 nanoseconds, an access to main memory takes 80 nanoseconds. What is the effective memory access time if we assume a TLB hit ratio of 97%? (You may ignore all caches apart from the TLB.) **1 T-pt**

T-Question 5.3: Page Replacement

a. Explain Belady's anomaly. Give an example (must be different from the one presented in the lecture). **2 T-pt**

P-Question 5.1: Page Replacement Algorithms

In this assignment you are going to write a simulator for two page replacement algorithms. Your program will run two simulations side by side: Two systems otherwise being equal, one using the page replacement algorithm least-recently-used (LRU), the other using clock. Your simulator will read a trace of memory accesses from stdin and feed those accesses into the two simulations. Then, it will output the resulting behavior to `stdout`. Using this setup, you will be able to directly compare both page replacement algorithms, as both will handle the same accesses.
Assume main memory has the amount of frames defined by the macro `FRAMES` (in `access.h`). Page numbers range from 0 to 255 (see macro `PAGES`).
At program start, no pages are mapped. Both systems must start allocating page frames from the lowermost frame 0 up. Once all page frames are occupied, pages have to be evicted to provide frames for unmapped pages.
Report the following events and use the provided functions upon page accesses:

- `reportMiss( a, x );` to indicate a pagefault for page x; parameter a specifies from which of the simulations (`LRU` or `CLOCK`) the event gets reported.
- `reportHit( a, x, y );` to indicate that page x is in page frame y
- `reportEviction( a, x, y );` to indicate that page x has been evicted from page frame y

Don't print directly or modify the format of the provided reporting output! Make sure you report a "miss" before reporting an "eviction" upon a pagefault!

a. Implement the least-recently-used (LRU) page replacement algorithm. Choose an implementation based on counters or with a stack, as presented in the lecture. Complete the function `void access_lru( int pn )`. **4 P-pt**

b. Implement the clock algorithm in `access_clock( int pn )`. The `main()` function will run the simulators for both algorithms side by side, so make sure that you design your datastructures properly: Keep in mind that LRU and clock can result in different mappings of pages to frames.
Clock (aka "second chance") is a popular page replacement algorithm that works as follows:
"[...] keep all the page frames on a circular list in the form of a clock, [...]. A hand points to the oldest page. When a page fault occurs, the page being pointed to by the hand is inspected. If its R bit is 0, the page is evicted, the new page is inserted into the clock in its place, and the hand is advanced one position. If R is 1, it is cleared and the hand is advanced to the next page. This process is repeated until a page is found with R=0. Not surprisingly, this algorithm is called clock." [1]

[1] Andrew S. Tanenbaum, Modern Operating Systems
Do not forget to set R to 1 on a reference! The clock hand should initially point to frame 0. **4 P-pt**

P-Question 5.2: Memory-Mapped Files

Write a program that performs an in-place encryption of lower-case letters in a file (ignore all other characters). Use memory-mapped file I/O to access the file. Read `man 2 mmap` to prepare for the assignment. The direction (encrypt/decrypt), the filename, the cipher, and the encryption key are supplied by commandline parameters (in `argv[1]`, ... `argv[4]`, in the order given here).

a. Write code that parses the parameters and memory-maps a file. Then implement the `caesar` encryption method. If you place the letters a-z in a circle, the Caesar cipher transposes each letter by rotating it to another letter. A single letter serves as the key and determines how far each letter is rotated during encryption. The key 'a' means that characters are not changed; the key 'b' means that 'a' is rotated to 'b', 'b' to 'c', ..., and 'z' to 'a'; the key 'c' means that 'a' is rotated to 'c', 'b' is rotated to 'd', ..., and 'y' to 'a', and so on. Your code must support both encryption and decyption. **6 P-pt**

b. Implement the Vigenere encryption method. The Vigenere method employs the Caesar cipher for each letter. Instead of a single letter, it uses a string of letters (e.g., a word) as the key. The first letter is encrypted with the Caesar method according to the first letter in the key, the second letter according to the second letter in the key, and so on. When you encrypt texts that are longer than the key, you wrap around and use the key repeatedly. Skip over bytes that are no lowercase letters without moving along in the key string. **2 P-pt**

Total: 16 T-pt 16 P-pt