Homework 3 for CSE153 (Winter 2022)

Due: Friday, Wednesday, March 2

Instructions:

* Be brief in your answers. You will be graded for correctness, not on the length of your answers.

* Make sure to write legibly. Incomprehensible writing will be assumed to be incorrect.

I. Consider the following portion of a page table on a system with page size of 1024 bytes (2^{10}):

Page	Physical Page	S	W
0	0x10	1	1
1	0x31	1	0
2	0x02	1	0
3	0x24	0	1
4	0x33	0	0

In the table above, the physical page refers to the physical frame in memory where the page is stored. The S bit indicates whether the page is supervisor page (kernel only) or not (user page). The W bit indicates whether the page is writeable (e.g., code pages or constant data pages are often marked as not writeable). A hex digit (0-f) represents 4 binary digits, eg. (0x10f2 = 0001 0000 1111 0010). Be careful that the page number and offset are not multiples of 4.

Explain what happens when the following memory references are executed from a program (i.e., in user mode). If possible, show the translation then explain what happens.

(a) A store to address 0x00ee? (2 points)

(b) A read from 0x0f01 (2 points)

(c) A store to 0x0415 (2 points)

II. (a) An OS is using two-level paging to implement a 28-bit virtual address space per process. The page size is 256-bytes, and the machine does not have a TLB. Explain the steps involved in looking up the virtual address 0x03bf04d, when all pages are present in memory. (2 points)

(b) For the system above, what is the maximum number of page faults that could be generated in response to a memory access? (2 points)

III. Consider a virtual address system with the following parameters.

• The memory is byte addressable.

• Virtual addresses are 18 bits wide.

• Physical addresses are 16 bits wide.

• The page size is 1K bytes (note that offset is more than 2 hex digits, but less than 3).

• The TLB is fully associative with 16 total entries.

Recall that a fully associative cache has just one set of entries—The tag field is simply the VPN and we need to search the full TLB to check if the VPN we are seeking is in the TLB. In the following tables, all numbers are given in hexadecimal. The contents of the TLB and the page table for the first 16 virtual pages are as follows. If a VPN is not listed in the page table, assume it generates a page fault.

TLB

Tag	PPN	Valid
03	1B	1
06	06	0
28	23	1
01	18	0
31	01	1
12	00	0
07	3D	1
0B	11	1
2A	2C	0
11	1C	0
1F	03	1
08	14	1
09	2A	1
3F	30	0
10	0D	0
32	11	0

Page Table

VPN	PPN	Valid
00	27	1
01	0F	1
02	19	1
03	1B	1
04	06	0
05	03	0
06	06	0
07	3D	0
08	14	1
09	2A	1
0A	21	1
0B	11	1
0C	1C	1
0D	2D	0
0E	0E	0
0F	04	1

(a) Which bits of an address hold the VPO and which hold the VPN? (1 point)

(b) For the virtual address 0x02022, indicate the physical address. Indicate whether or not there is a TLB miss. If there is a page fault, enter "—" for the PPN and Physical Address. All answers should be given in hexadecimal, but *be careful since the offset and page number are not multiples of 4, there will be one digit that contains some bits from each. For example, given the address 0x12345, the digit 3 contains 2 bits from the offset and 2 bits from the page number. First convert to binary (01 0010 0011 0100 0101): the offset will be the bottom 10 bits (1101000101 or 0x345) and the page number the top 8 bits (01001000 or 0x48)* (1 points)

(c) Repeat for address 0x004F1 (1 points)

(d) Given an physical address (i.e., after translation) of 0x460e, is it possible to tell what the virtual address was from the given page table? If so, show the address. (1+1 bonus). Hint: identify the physical page number then check if there is a TLB entry or Page table entry that translates to this page number.

IV. Consider a process that has been allocated 5 pages of memory: P1, P2, P3, P4, and P5. The process accesses these pages in the following order:

P1 P2 P3 P4 P1 P2 P5 P1 P2 P3 P4 P5

(i) Illustrate Belady's anomaly by precisely describing the execution of the FIFO page eviction algorithm in two cases: a) where the machine has 3 pages of physical memory, and b) where the machine has 4 pages of physical memory, and by comparing the number of page faults incurred in these two cases. (When the process begins executing, none of its pages are present in memory.) (2 points)

(ii) Show how the LRU page eviction algorithm would work in the same scenarios a) and b) described above. (2 points)