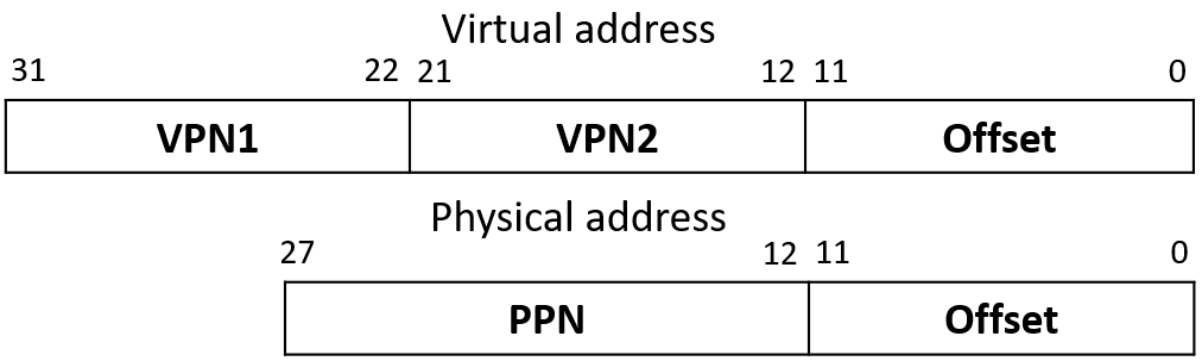
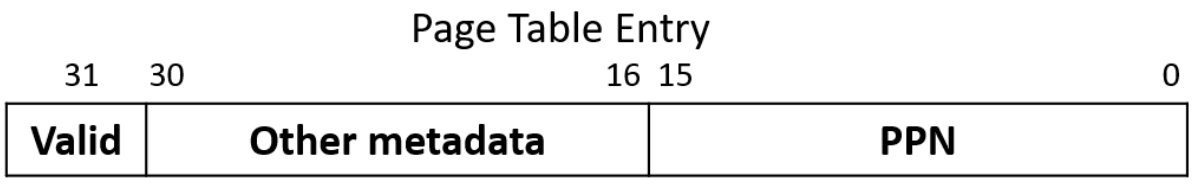
HW9.3. Address Translation and TLB (Randomized)

It's highly recommended that you finish and understand HW9.2 Page Table Walk before attempting this problem. That problem covers how to navigate the different page table hierarchies starting from the virtual address.

Consider a machine with a 2-level hierarchical page table that has 20 bits of VPN divided equally between L1 and L2 page tables. The page offset is 12 bits and the PPN is 16 bits. The 32-bit virtual and 28-bit physical addresses are partitioned as shown below:



A single page table entry (PTE) is 4 bytes, with the corresponding data partitioning as shown below:



The most significant bit is 1 when the entry is valid and the last 16 bits correspond to the PPN pointing to either the next level page table or the data page. Extra metadata bits are included in the PTE but are not relevant for this problem.

Now, suppose the physical memory and the page table have this initial state. Each arrow indicates the location of the next-level page table being pointed to.

Note that the Page Table Base Register (PTBR) is not included in the diagram anymore for simplicity. It is not relevant for this problem. The L1 page table that the PTBR is pointing to is already given in the diagram.

Homework 9

Assessment overview

Total points: 40/100

Score: 40%

Question

Value: 20

History:

Awarded points:

Report an error in this question

Previous question

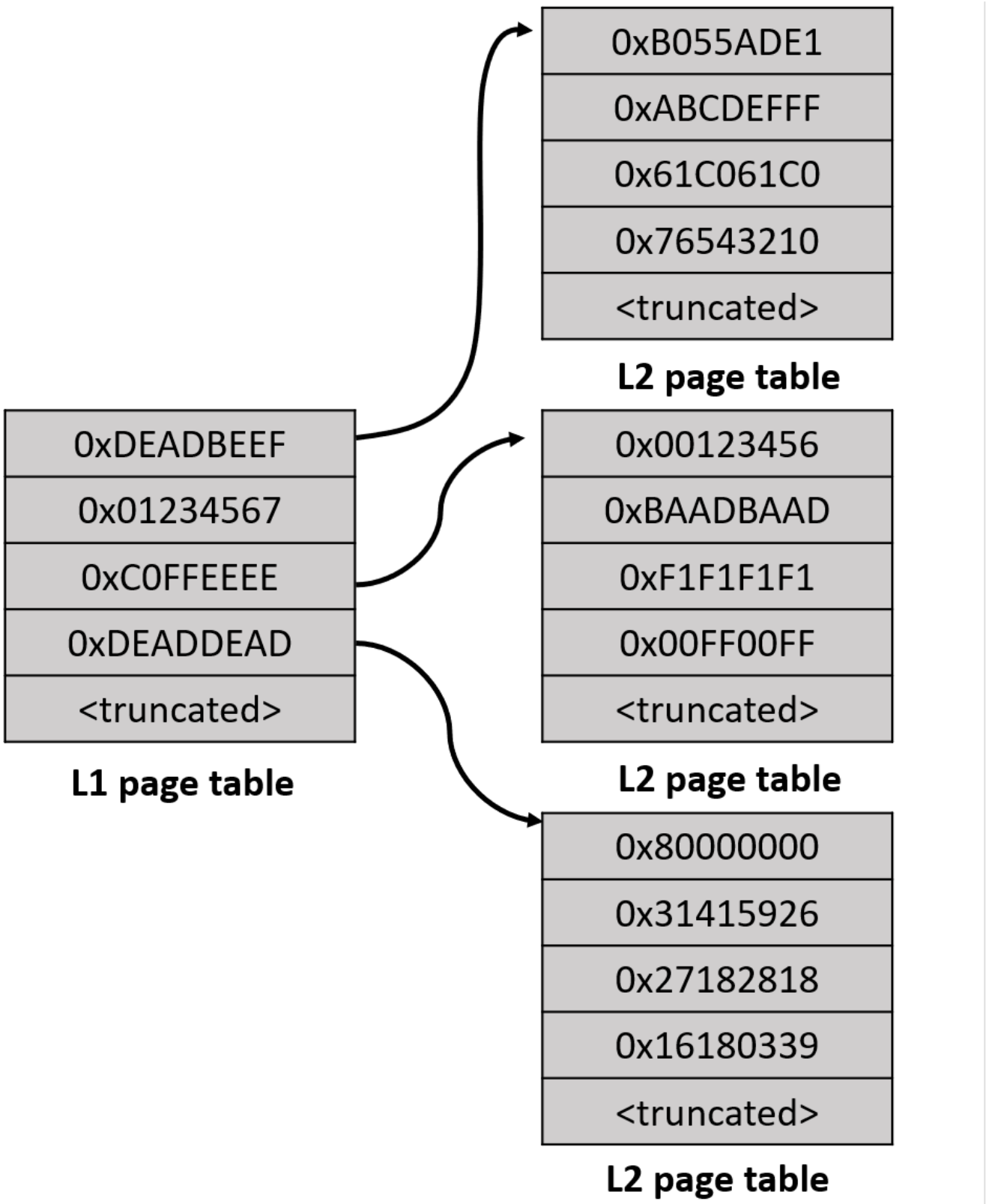
Next question

Attached files

No attached files

Attach a file

Attach text



The following virtual memory addresses are accessed in sequence, with the initial state in the above diagram. **The system uses a 2-entry fully-associative LRU TLB and starts out empty.** As a recap, the TLB is just the cache of the address translations. One entry of the TLB contains a translation of VPN to PPN. Given the presence of the TLB, mark the outcome of each access.

Moreover, translate each virtual address into the corresponding physical address that it maps to. Write your answer in hexadecimal, without the leading `0x`. Since the physical address is 28 bits, the answer should be in written in 7 digit hex. If a valid physical address does not exist, write `N/A` instead.

Tip: When doing the translation, make sure you are properly extracting the VPN1 and VPN2 bits from the virtual address. It might help writing the virtual address out in binary first, then count the number of bits needed for each VPN.

Additionally, keep track of the translations that have been loaded into the TLB. That determines whether you'll end up with a TLB hit or miss. The valid bit on the page table entry then determines whether it is a page table hit or a page fault.

Hint: Following the given L1 and L2 page tables, the virtual address `0x00C0061C` translates to the physical address `0x000061C`. However, the virtual address `0x00C0161C` does not have a valid translation and, therefore, corresponds to a page fault. Think about how to get this. Review the guided steps in HW9.2 if needed.

Q1.1) `0x00000840`

- ○ (a) TLB hit
- ○ (b) TLB miss but page table hit
- ○ (c) Page fault

0x

Q1.2) `0x0000149E`

- (a) TLB hit
- (b) TLB miss but page table hit
- (c) Page fault

0x

Q1.3) `0x008010A1`

- (a) TLB hit
- (b) TLB miss but page table hit
- (c) Page fault

0x

Q1.4) `0x00001405`

- (a) TLB hit
- (b) TLB miss but page table hit
- (c) Page fault

0x

Q1.5) `0x00000334`

- (a) TLB hit
- (b) TLB miss but page table hit
- (c) Page fault

0x

Q1.6) `0x000027B7`

- (a) TLB hit
- (b) TLB miss but page table hit
- (c) Page fault

0x

Save & Grade *20 attempts left* | Save only | *Additional attempts available with new variants*