Computer Architecture and Engineering

CS152 Quiz #3

March 19th, 2009

Professor Krste Asanovic

Name:___Answer Key____

This is a closed book, closed notes exam.

80 Minutes

6 Pages

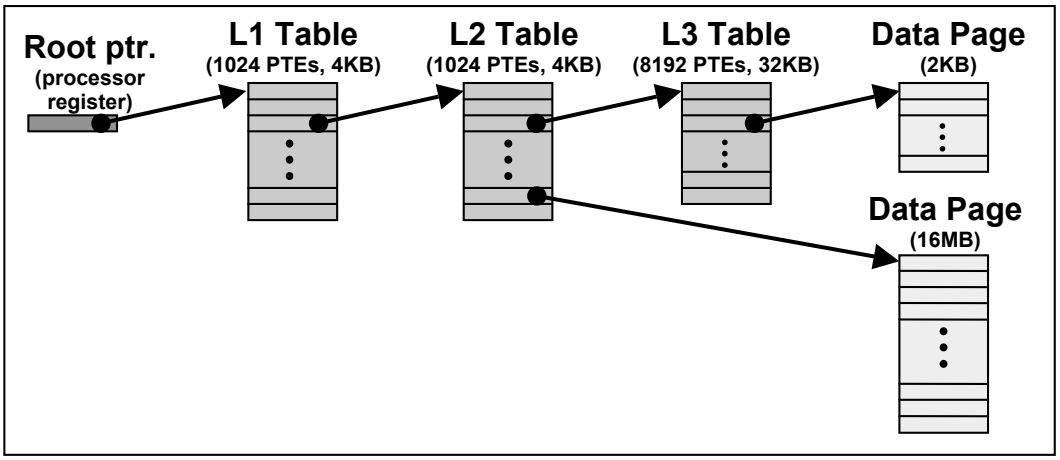
Notes:

- **Not all questions are of equal difficulty, so look over the entire exam and budget your time carefully.**
- **Please carefully state any assumptions you make.**
- **Please write your name on every page in the quiz.**
- **You must not discuss a quiz's contents with students who have not yet taken the quiz. If you have inadvertently been exposed to the quiz prior to taking it, you must tell the instructor or TA.**
- **You will get no credit for selecting multiple choice answers without giving explanations if the instructions ask you to explain your choice.**

Writing name on each sheet	**_____**	**1 Point**
Question 1	**_____**	**29 Points**
Question 2	**_____**	**16 Points**
Question 3	**_____**	**16 Points**
Question 4	**_____**	**18 Points**
TOTAL	**_____**	**80 Points**

Problem Q3.1: Page Size and TLBs 29 Points

In this problem we will be analyzing two systems, both with 36-bit physical addresses, 44-bit virtual addresses, and 4-byte PTEs. One configuration is a single-level page table with 2KB pages and the second configuration is a hierarchical page table which supports both 2KB and 16MB pages. The following figure summarizes the hierarchical page table structure and indicates the sizes of the page tables and data pages (not drawn to scale):



Problem Q3.1.A 8 Points

For each of the proposed systems, how much space is needed for the page table for one user process if one 2KB data page is allocated? (4 points each)

2KB page => 11 bit offset

$2^{44-11} = 2^{33}$ VPNs

Page Table Size = (# VPNs) × (PTE size) = $2^{33} \times 4 = 2^{35}$

Size of single level page table 2^{35}B = 32GB

Size of hierarchical page table 40KB

Page Table Size = (size of L1) + (size of 1 L2) + (size of 1 L3)
= 4KB + 4KB + 32KB = 40KB

Problem Q3.1.B 8 Points

Let's define the page table overhead (PTO) for a process to be (as in the problem set):

$$\textbf{PTO} = \frac{\text{Physical memory that is allocated to page tables}}{\text{Physical memory that is allocated to data pages}}$$

For the hierarchical page table, which usage of data pages by a user process will result in the smallest possible PTO and which will result in the largest PTO? (4 points each)

Smallest: Only 16MB pages used, consecutively allocated and enough to fill up memory
Largest: One 2 KB page

Note: The question did not ask you to compute the PTO.

The processor has a fully-associative data TLB with 128 entries, and each entry can map either a 2KB page or a 16MB page. After a TLB miss, a hardware engine walks the page table to reload the TLB. The TLB uses a first-in/first-out (FIFO) replacement policy.

We will evaluate the execution of the following program which adds the elements from three 4MB arrays and stores the results in a fourth 4MB array (note that, 1MB = 1,048,576 Bytes, the starting address of the arrays are given below):

```
double A[524288]; // 4MB array 0x00001800000
double B[524288]; // 4MB array 0x00001c00000
double C[524288]; // 4MB array 0x00002000000
double Y[524288]; // 4MB array 0x00002400000

for(int i=0; i<524288; i++)
  Y[i] = A[i] + B[i] + C[i];
```

Assume that the above program is the only process in the system, and ignore any instruction memory or operating system overheads. The data TLB is initially empty.

Problem Q3.1.C **5 Points**

How many TLB misses will the program incur with the single-level page table?

The program will access 16MB of consecutive data, and it never reuses the same data. The problem implicitly asked about 2KB pages, but if 16MB pages were used correctly credit was given. For this problem, it doesn't matter that the TLB has 128 entries, all that is important is that it has 4 or more entries.

misses = (total memory)/(page size) = 16MB/2KB = $2^{24-11} = 2^{13}$ = **8K misses**

Problem Q3.1.D **8 Points**

How many TLB misses will the program incur with the hierarchical page table using 2KB pages? What about if it uses 16MB pages? (4 points each)

Misses with 2KB pages **8k**

Same as Q3.1.C. Page table doesn't affect TLB miss rate.

Misses with 16MB pages **2**

Although all four arrays could fit in one page, the addresses are not correctly aligned, so the program's data will span two pages. 2/4 points were given for 1 as a response. Additional credit was given if alignment assumptions were stated.

Problem Q3.2: Virtual Address Aliasing 16 Points

A problem with virtually addressed caches is the aliasing problem. The aliasing problem happens when distinct virtual addresses refer to the same physical location ($VA_1 \rightarrow PA$ and $VA_2 \rightarrow PA$, but $VA_1 \neq VA_2$).

In lecture 14, a solution to this problem was presented for a direct-mapped cache. The solution was a software restriction that required if two virtual addresses want to address the same physical address, they must have the same virtual index bits. This prevents more than one of the aliases residing in the cache at a time. If VA_1 is in the cache and VA_2 is accessed, there will be cache miss because $VA_1 \neq VA_2$ and VA_1 will be evicted and replaced with VA_2, which happens to be mapped to the same physical address.

For this problem (3.2) there is only one level of cache. Please explain your reasoning in your answers below.

Problem Q3.2.A 6 Points

With a virtually addressed (virtual index and virtual tag) cache, will the above solution work if the cache is 2-way set associative? Explain.

The software solution will force VA_1 and VA_2 to have the same index, which implies they will be mapped to the same set. Since $VA_1 \neq VA_2$, their tags must differ, so they could each occupy a different way of the set. Having two copies in the cache is problematic.

Problem Q3.2.B 10 Points

Now consider a cache that is virtually indexed and physically tagged where:

$$\text{page size} \geq \frac{\text{cache capacity}}{\text{associativity}}$$

Insight: The above relation implies that the index will be contained within the page offset, which means an address will have the same index in its physical address as in its virtual address since the offset isn't translated. Since both addresses map to PA, it means both VA_1 and VA_2 will have the same index and map to the same set.

i) Do we still need the software restriction if the cache is direct mapped? Explain. (5 points)

No. Because of the above insight, they will have the same index without the software's help. Because they map to the same physical address, their physical tag will be the same, so both VA_1 and VA_2 will access the same line in the cache.

ii) Will the system work if the cache is 2-way set associative? Explain. (5 points)

Yes (the system without the SW restriction will work). The same reasoning as in part i applies. Since the physical tags are the same, both VA_1 and VA_2 will map to the same line the cache, even though there are now 2 ways.

Problem Q3.3: System Design 16 Points

In this problem we will investigate systems that have a subset of the three features: protection, translation, and virtual memory.

Problem Q3.3.A 4 Points

Would a system with only protection be useful (i.e., no translation and no virtual memory)? If so, describe how such a system might be used. If not, explain why.

Yes, protection is useful whenever you want to run more than one application at a time. The other features might not be needed in an embedded system. Protection improves fault tolerance since buggy code "shouldn't" be able to damage other code. Even if only one application is running, it might be made of multiple pieces of code that have bugs, and thus need protection from itself.

Problem Q3.3.B 6 Points

Would a system with only protection and translation (i.e., no virtual memory) be useful? If so, describe how such a system might be used. If not, explain why.

Yes, it just isn't able to swap out pages to offer the illusion of larger memory capacity. This might be useful with a supercomputer where your job should all fit in memory for peak performance.

Problem Q3.3.C 6 Points

Would a system with only protection and virtual memory (i.e., no translation) be useful? If so, describe how such a system might be used? If not, explain why.

Yes, but it would be very slow.

Without translation how will things get found? One way is to only allow one process to inhabit physical memory at a time, so on a context switch all of main memory must be swapped.

Without translation how could 2 programs inhabit memory at the same time with each program given the illusion that it has all of memory? Techniques to solve this will have some sort of implicit translation.

Problem Q3.4: Design Tradeoffs 18 points

Mark whether the following modifications will cause each of the categories to **increase, decrease**, or whether the modification will have **no effect**. You can assume the baseline system uses a two-level page table with a single page size. **Explain your reasoning** to receive credit. (2 points each)

	Page Table Size	TLB Hit Rate	Internal Fragmentation
Increase page size	Decrease Larger pages means less pages for the same virtual address space. Less pages means less PTEs which should make a smaller page table.	Increase (probably) The TLB's reach has been extended since each entry maps more memory. For this to improve hit rate there needs to be some spatial locality (there should be).	Increase With larger pages there is a greater chance that each page will not be completely filled.
Making the page table 3 level (page size and virtual address size constant)	Decrease if program has sparsely populated address space. Increase if program has more densely populated address space.	No effect Page size is the same so the same TLB entries will get the same amount of hits and misses.	No effect Page size is the same, so there shouldn't be a change in fragmentation.
Adding support for multiple page sizes	Decreases if the new page size is larger since we will need less PTEs. Could increase if we support a smaller page size and need more PTEs.	Increases with larger pages for same reason as above.	Should decrease as the smallest page to hold data is used to reduce fragmentation. Could increase if larger pages are used to increase TLB hit rate.

END OF QUIZ