

Comp 120 Computer Organization
Spring 2005

Problem Set #10

Issued Tuesday, 04/14/05; Due Thursday, 04/21/05

Homework Information: Some of the problems are probably too long to be done the night before the due date, so plan accordingly. Late homework will not be accepted. Feel free to get help from others, but the work you hand in should be your own.

Problem 1. Virtual Realities

A particular next-generation byte-addressable processor has been proposed with a 64-bit virtual address space and a 48-bit physical address space. Assume that each page table entry (PTE) requires 8 bytes.

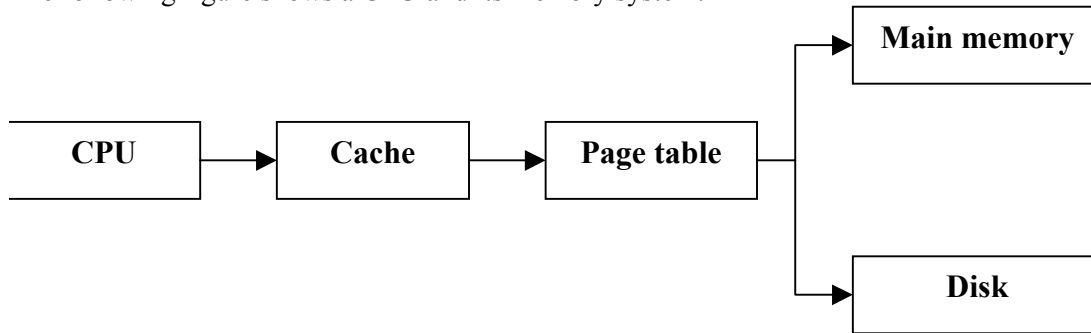
- (A) What is the smallest page-size that will allow the entire pagemap to be resident in physical memory, while preserving at least half of memory for program and data pages? How many PTE bits are needed to hold the physical page number (PPN)?
- (B) Assuming a price of \$100 per gigabyte of DRAM (roughly the going rate), how much would it cost to fully populate physical memory?

From above, it should be evident that a multilevel hierarchical pagemap is the only viable MMU alternative. Moreover, provisions for paging-out sections of the pagemap are also required. In the proposed architecture all intermediate page translation table entries (TTE) other than the first level, are memory resident and broken into page-sized sub-tables. The first level of the hierarchical page-translation is locked into a single memory page (i.e. it is not swapped out) and it contains the first-level translations for multiple contexts. All PTEs and TTEs occupy 8-bytes, and each level's lookup (other than the first) fits into a page-sized table.

- (C) If the desired page-size is 64Kbytes, how many levels of address translation are required? How many TTEs are in the first-level on-chip lookup? In the worse case (a fully instantiated page-map), how much overhead in terms of pages is associated with the multilevel translation?
- (D) Again assuming a 64Kbyte page-size, how many bits of the PPE are needed for the PPN? How many bits of the TTE are needed for the PPN?
- (E) On a TLB miss the multi-level table walk is implemented in a software memory-fault handler. How many additional page loads could be generated in the worse case on a TLB miss? How many different contexts can be supported?

Problem 2. Jogging Your Memory System

The following figure shows a CPU and its memory system:



The computer features a single process; 32-bit virtual addresses; a cache with a *total* of 2^{10} lines that is four-way set associative, has 8-byte data blocks and is write-through; a main memory of 2^{26} bytes; and a page size of 2^{12} bytes.

- (A) Does this system cache virtual or physical addresses?
- (B) How many bytes of memory *data* can the cache hold? Don't count tags.
- (C) Consider the contents of memory location 0x1000. In how many different locations in the cache *might* this data be stored? In how many different locations in the cache will the data *actually* be found at any given moment?
- (D) Each block of data in the cache must have a tag associated with it. How many bits wide are these tags?
- (E) How many comparators are needed to build this cache?
- (F) Explain how the 32-bit addresses produced by the CPU are processed by the cache. Indicate which bits are used to select data from the data block, which bits are used to select which cache line to access, and which bits are sent to the tag comparators to determine if there is a cache hit. Assume that the cache has an architecture like the one shown on slide 17 of L18 ("Cache Issues").
- (G) How many entries are there in the page table? How many bits does each entry have? Assume a one-level page table.
- (H) At any given time, what is the greatest number of page-table entries that can have their resident bit set to 1?
- (I) Explain how the 32-bit addresses produced by the cache after a "miss" are processed by the page table. Indicate which bits are used to form the page offset and which bits are used to form the virtual page number.
- (J) Given your answers to parts (G) and (H) would you recommend switching to a two-level page table? Why?

Problem 3. One Last Bus Stop

Suppose you have a system with the following characteristics:

1. A processor memory interface that accesses blocks of either 4 or 16, 32-bit words.
 2. A 64-bit synchronous bus clocked at 200 MHz. One cycle is required for the master assertion, which specifies the address and operation, and 1 clock cycle is required for the 64-bit data transfer, and acknowledgement. There must be at least one idle clock between the Master assertion cycle to allow the master to release the operation signals and possibly the address signals, on a read.
 3. One clock cycle is required between bus cycles, for deassertion of the slave acknowledge, and bus arbitration.
 4. A random memory access time of 200ns is needed for the first four words read from a slave. Subsequent blocks of 4 words can be read from the same slave device in 20 nS if the bus address sequentially follows the address of the previous bus cycle.
- (A) Find the sustained bandwidth and latency for a read of 256 words for transfers that use 4-word blocks and for transfers that use 16-word blocks.
- (B) Compute the effective number of bus transactions per second for each of the cases in part A. Each bus transaction consists of a Master assertion cycle.
- (C) Assume 2 bus masters share the synchronous bus discussed above. Assume that in a particular application alternate bus cycles are assigned each master. How does this affect the sustained bandwidth and transactions per second estimates from parts A and B? You should assume that it is very unlikely that the two masters will generate nearby memory addresses.