6.823 Computer Architecture	DUE: 3/13/2002
Timothy Danford, Kilian Pohl, Bryan Russell	
Problem Set 3	*Instructors: Krste Asanovic, Srinivas Devadas*

Problem 4

(a) Since each page is 4KB (2^{12} bytes) and there are 2^{64} bytes of memory, there are 2^{52} PTEs. Since each PTE is 4 bytes, the page table is 2^{54} bytes in size.

(b) Again, since each page is 4KB (2^{12} bytes) and there are now 2^{42} bytes of memory, there are 2^{30} PTEs. Since each PTE is 4 bytes, the page table is 2^{32} bytes in size ($\sim$ 4295 MB).

(c) For the minimum page table overhead, we need to maximize the amount of physical memory used by the user code, stack, and heap and minimize the physical memory used by the page tables for a user process. Imagine that we allocate only one page and we use all of the space in that page with user code, stack, and heap data. Hence, we would have only used three page table pages. Now imagine that we allocate one more page and again use all of the space in that page for user code, stack, and heap data. Hence, we would have used at most five page table pages and used (for this problem) 8 KB for code, stack, and heap thus far. If we continue to add pages in this manner and filling them to capacity, we decrease the ratio. So, the minimum ratio is a full hierarchical page table with all allocated pages full of code, stack, and heap data. This minimum ratio is:

$$\frac{4(1024^3 + 1024^2 + 1024)}{2^{42}} = 2^{-30} + 2^{-20} + 2^{-10} = 0.00098$$

The maximum ratio is the case when we allocate only one page (we see that by adding more pages, the ratio decreases). We assume that the entire allocated page is useful, so the maximum ratio is:

$$\frac{4(3 \cdot 1024)}{2^{12}} = 3$$

(d) Since there are 4 KB pages, 64 slots, and 8 PTEs per slot, we can address 2^{21} bytes. So, we can address the entire array that we are looping over. To calculate the average number of memory accesses, first let's assume that the array is in memory and we are at the beginning of one of the loops. For each byte, we make 1 memory access to fetch the byte itself. Furthermore, we must search for the PTE within the slot that we hashed to. So, for each memory translation we must make 1 to 8 memory accesses to get the desired PTE. So, if we add all of this up over the entire array and divide by the total number of bytes, we get:

$$\frac{2^{21} + 2^{18}(\sum_{n=1}^{8} n)}{2^{21}} = 1 + \frac{36}{8} = 5.5$$

So, we have to do on average 5.5 memory accesses for every byte we fetch.

(e) As above, for every byte we fetch we must access the memory once to fetch the byte itself. Since we fetch from the array in sequential order, for every 4 KB we must hash to a different slot and use a different PTE. However, we do not look outside of the page while we access the 4 KB of array data in sequential order. So, with the one-entry TLB we only have to do the address translation lookup when we hash to a different PTE. As above, when we hash we must search through the slot to find the desired PTE, implying that we make 1 to 8 memory accesses. So, if we add all of this up over the entire array and divide by the total number of bytes, we get:

$$\frac{2^{21} + 2^{6}(\sum_{n=1}^{8} n)}{2^{21}} = 1 + \frac{36}{2^{15}} = 1.0011$$

So, with the TLB we have to do on average 1.0011 memory accesses for every byte we fetch.