Section 5: Address Translation & Caches and TLBs

Short Answer

1. If a computer has a 32 bit address space, and 1K (i.e. 2^10 bytes) sized pages, how many page table entries does it have? (fine to express answer base-2).

2^32/2^10 = 2^22 = 2^2*2^20 = 4M = ~4Million

2. If you have a very large virtual address space, what are the benefits of using an inverted page table? What are the drawbacks?

   Benefit: Table's size is directly proportional to the size of physical memory, as opposed to the size of virtual memory, which is typically much larger than physical memory.
   Drawback: Requires maintaining a hash table in hardware, which is hard

Long Answer

Caching: Assume a computer system employing a cache, where the access time to the main memory is 100 ns, and the access time to the cache is 20ns.

1. Assume the cache hit rate is 95%. What is the average access time?
   Average Access Time = Hit*cache_access_time + (1Hit)*memory_access_time
   = 0.95*20 ns + 0.05*100 ns = 24 ns
   Alternatively, we accepted solutions that included the cache time in the memory access time: AAT = 0.95 * 20 ns + 0.05 * (20 ns + 100 ns) = 25 ns.

2. Assume the system implements virtual memory using a two--level page table with no TLB, and assume the CPU loads a word X from main memory. Assume the cache hit rate for the page entries as well as for the data in memory is 95%. What is the average time it takes to load X?
   The Average Memory Access Time for X (AMAT) requires three memory accesses, two for each page entry, and one for reading X: 3*24 = 72 ns.
   The alternate solution from (a) yields 3*25 = 75 ns. We only accepted the alternate solution for (b) if you derived the same value for (a)

Longer Answer (From Fall 2012 Midterm)

Assume a system with a two level page table. The virtual memory address space is 32 bits and the physical memory address space is 16 bits.

1. Make sure that each translation table fits in a page.
    a. What is the optimal page size? 4 KB
    b. Specify the length of each field in the virtual address.

(bit 31) (bit 0)

Virtual P1 Index	Virtual P2 Index	Offset
10	10	12

2. Assume you add to your system a 4- way set--associative data cache with 16 total cache blocks. Each block in the cache holds 8 bytes of data. In order to address a specific byte of data, you will have to split the address into the cache tag, cache index and byte select. Assume there are no modifiers bits in the table.

(bit 31) (bit 0)

tag	index	byte select
16-3-2 = 11	16 blocks/4 (2^2) = 2	8 bytes (2^3) = 3

5. (18 points) Paging:
Suppose you have a system with 32-bit pointers and 4 megabytes of physical memory that is partitioned into 8192-byte pages. The system uses an Inverted Page Table (IPT). Assume that there is no page sharing between processes.

a. Describe what page table entries should look like. Specifically, how many bits should be in each page table entry, and what are they for? Also, how many page table entries should there be in the page table?

Virtual addresses are 32 bits, and split into two parts. The page number is the first 19 bits, and the offset within the page is the last 13 bits (2^{13} = 8,192).

Virtual Page Number	*Offset*
19 bits	*13 bits*

The inverted page table is a mapping of physical addresses to virtual addresses. Memory is 4 megabytes, partitioned into 512 pages. Therefore the inverted page table will consist of 512 entries. Each of these entries must have:

19 bits for the virtual page number of the physical page.
Some number of bits (16 in Unix) for the processs ID of the process that owns the page.
Protection bits (r/w/x)

We awarded two points each for: the correct number of IPT entries, storing the virtual page number in the IPT, storing the process ID in the IPT, and storing the protection bits(any reasonable set was accepted) in the IPT.
We subtracted one point for each extraneous item that you stored in the IPT, and subtracted one point for storing the physical page number instead of the virtual page number in the IPT.

b. Describe how an IPT is used to translate a virtual address into a physical address.

A virtual address is translated to a physical address hashing virtual addresses (worth two points). Thus, in the normal case, the translation may be found in a few memory lookups rather than an entire table traversal. If it is found, and the owning process is equal to the current running process (owning PID check is worth 2 points), then its index in the table is the frame number of the physical page. If it is not found, then a memory fault must occur (not found fault is worth 1 point).
We also accepted a general search of the IPT, as described in the book and midterm review session.