CS3411 Test #3, 30 October 2007. You have 1h15m (one class period). Answer all
questions. The number of points out of 100 are shown in parentheses for each. Be as
concise as you can while still answering the question.

1. (24) Mark each statement as "true" or "false" (write the word, not "T" or "F", some of
you have handwriting almost as bad as mine).

*The miss rate is equal to 1 minus the hit rate. True

*The miss penalty of the L1 cache is completely independent of the miss rate of the L2
cache. False

*A direct-mapped cache cannot have any conflict misses. False

*A direct-mapped cache requires no tags. False

*In terms of performance only, a bigger block size is always better. False

*By simply adding overlapped miss latency to our memory stall equation it is
straightforward to calculate the performance impact of memory on an out-of-order,
superscalar processor. False

*The addresses that a Linux process uses as pointers are physical addresses. False

*The size of an inverted page table scales with the size of physical memory, but can be
independent of the size of the virtual space. True

*A disadvantage of segments over paged memory is that pointers have two parts. True

*A cache coherence protocol defines what memory ordering is agreed upon by the
programmer and the system (e.g. sequential consistency), while a consistency model is a
device that enforces this ordering (e.g. write-once). False

*Aliasing is not possible in virtual memory, that is, it never occurs that two virtual
addresses refer to the same physical memory. False
A cache coherence protocol defines what memory ordering is agreed upon by the programmer and the system (e.g. sequential consistency), while a consistency model is a device that enforces this ordering (e.g. write-once). False

2. (36) Multiple choice. **Mark all answers that apply** (i.e. if no answers apply, circle none of them, if all apply circle all of them, if 1, 2, or 3 apply, circle those).

*The types of locality that we try to exploit with a memory hierarchy are...
  a. *Spatial Locality*
  b. Virtual Locality
  c. Tag Locality
  d. *Temporal Locality*

*Depending on where in the memory hierarchy you are talking about, a block may be called a...
  a. *Cache line*
  b. *Page*
  c. TLB

*In a typical, practical implementation, levels of the memory hierarchy that can be write-through include:
  a. In virtual memory the DRAM writes through to the hard drive
  b. *The L2 cache writes through to the DRAM*
  c. *The L1 cache writes through to the L2 cache*

*Which of these might work (to at least some degree--mark all that may apply) if I am trying to cause lots of cache misses in a 64KB fully-associative cache?*
  a. Find 2 cache lines in the same set and read them alternately over and over
  b. Find 9 cache lines in the same set and read them alternately over and over
  c. *Cause capacity misses by accessing >> 64KB of stuff*
  d. Access cache lines in different sets with matching tags
*Which of these might work (to at least some degree--mark all that may apply) if I am trying to cause lots of cache misses in a 64KB direct mapped cache?
 a. Find 2 cache lines in the same set and read them alternately over and over
 b. Find 9 cache lines in the same set and read them alternately over and over
 c. Cause capacity misses by accessing >> 64KB of stuff
 d. Access cache lines in different sets with matching tags

*Which of these might work (to at least some degree--mark all that may apply) if I am trying to cause lots of cache misses in a 64KB 4-way set-associative cache?
 a. Find 2 cache lines in the same set and read them alternately over and over
 b. Find 9 cache lines in the same set and read them alternately over and over
 c. Cause capacity misses by accessing >> 64KB of stuff
 d. Access cache lines in different sets with matching tags

*Virtual memory can provide:
 a. Memory protection for multiple processes
 b. A reduction in latency for accesses to DRAM
 c. The illusion of a larger amount of available physical memory in the system
 d. For each process to assume its own flat view of memory

*Which of these page table entry bits would get set if your user-space program wrote to the page:
 a. Reference a.k.a use bit
 b. Dirty bit
 c. Valid bit

*Which of these structures are indexed with virtual addresses?
 a. The page table
 b. A virtually addressable cache
 c. The TLB
*The "three C's" of cache misses are:
   a. Compulsory
   b. Capacity
   c. Context
   d. Conflict

3. (5) Suppose I have a 256 KB cache with 1KB cache lines that is 4-way set associative (assume the simple case of byte-addressable and physically addressed) for a 32-bit CPU. Which of these is the correct "number of sets, number of lines in each set, size of the tag". Mark only one answer.
   a. 32 sets, 8 lines per set, 32-10-10 = 12 bits for the tag
   b. 64 sets, 4 lines per set, 32-10-6 = 16 bits for the tag
   b. 128 sets, 2 lines per set, 32-10-6 = 16 bits for the tag
   b. 128 sets, 4 lines per set, 32-10-2 = 20 bits for the tag

4. (8) Label 3 of these as "impossible" and 2 as "unnecessary", the rest "okay."
   Impossible means that you can't have that sequence if the rules are followed that any page table entry in the TLB must be in the memory and that any data in the cache must belong to a valid page. Unnecessary means that you wouldn't need to check one if you hit in the earlier one. The elements correspond to hits or misses in that structure. Note that these are not ordered the same way as the figure in the book so you have to think about them.

<table>
<thead>
<tr>
<th>TLB</th>
<th>Page Table</th>
<th>Cache</th>
<th>Impossible, unnecessary, or okay?</th>
</tr>
</thead>
<tbody>
<tr>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>Okay</td>
</tr>
<tr>
<td>miss</td>
<td>miss</td>
<td>hit</td>
<td>Impossible</td>
</tr>
<tr>
<td>miss</td>
<td>hit</td>
<td>miss</td>
<td>Okay</td>
</tr>
<tr>
<td>miss</td>
<td>hit</td>
<td>hit</td>
<td>Okay</td>
</tr>
<tr>
<td>hit</td>
<td>miss</td>
<td>miss</td>
<td>Impossible</td>
</tr>
<tr>
<td>hit</td>
<td>miss</td>
<td>hit</td>
<td>Impossible</td>
</tr>
<tr>
<td>hit</td>
<td>hit</td>
<td>miss</td>
<td>Unnecessary</td>
</tr>
<tr>
<td>hit</td>
<td>hit</td>
<td>hit</td>
<td>Unnecessary</td>
</tr>
</tbody>
</table>
5. (5) Suppose I fill my entire virtual address space in a process on a Pentium (assume I can fill 0x00000000 through 0xFFFFFFFF because of Linux 2.6 segmentation tricks, i.e. no need to save room for the kernel). How much memory am I using for just the page tables (total for both levels).
   a. $1024 \times 4\text{KB} = 4\text{MB}$
   b. $1025 \times 4\text{KB} = 4\text{MB} + 4\text{KB}$
   c. $32 \times 4\text{KB} = 0.125\text{MB}$
   d. $1024 \times (1024 \times 4\text{KB}) = 4\text{GB}$

5. (12) For each of the following paging schemes in "a + b + c ..." notation, write the size of each page, the number of entries in each second-level page table, and the number of paging levels, and the size of the virtually addressable address space. No calculator necessary, write a a power of 2 if you must. The first (the Pentium) is done for you.

<table>
<thead>
<tr>
<th>Page size</th>
<th># entries in 2nd level</th>
<th># Levels</th>
<th>Virtually addressable address space</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 + 10 + 12</td>
<td>4KB</td>
<td>1024</td>
<td>2</td>
</tr>
<tr>
<td>10 + 10 + 10 + 13</td>
<td>8KB</td>
<td>1024</td>
<td>3</td>
</tr>
<tr>
<td>10 + 10 + 9 + 12</td>
<td>4KB</td>
<td>1024</td>
<td>3</td>
</tr>
<tr>
<td>9 + 9 + 9 + 9 + 12</td>
<td>4KB</td>
<td>512</td>
<td>4</td>
</tr>
</tbody>
</table>

6. (5) Between a sequential consistency model and a relaxed consistency model, which one typically has better performance? *Relaxed consistency model*

7. (5) As an alternative to flushing the TLB every time we do a context switch between two processes with different address spaces, we can add something to the TLB entry. What is this called? (Hint: the initials are P.I. or T.I.)

*Process Identifier or Task Identifier*