CIS501– Computer Architecture	**Midterm Exam**
Prof. Martin	**Thursday, Oct. 29, 2009**

This exam is an individual-work exam. Write your answers on these pages. Additional pages may be attached (with staple) if necessary. Please ensure that your answers are concise and legible. Read and follow the directions of each question carefully. Please attempt to answer all the questions (don't allow yourself to get stuck on a single question). You have 80 minutes to complete the exam (approximately one point per minute).

Write only as much as necessary. Be brief!

Name: ____

Problem	Page	Possible	Score
1	2	11	
2	3	11	
3	4	8	
4	5	7	
5	6	9	
6	7	8	
7	8	11	
8	9	10	
Total		75	

1. **[11 Points]** **True or False**. If a statement is "false," briefly explain how so by describing how the statement may most simply be made true. Unjustified (or poorly justified) "false" answers will be marked wrong. Simply stating the negation of the false statement is *not* sufficient justification. *Please be specific!*

   (a) *Computer architecture* involves using computers to design buildings.

   (b) A typical high-end chip today has approximately 100 million transistors.

   (c) Decreasing the gate "length" and increasing the "width" of transistor both reduce resistance.

   (d) Each process generation transition (say, 45nm to 32nm) doubles transitor density.

   (e) A 1Ghz processor takes longer to execute a program than a 2Ghz processor.

   (f) The x86 ISA is an example of a CISC ISA.

   (g) Both the CPI and the clock frequency of a processor vary from program to program.

   (h) Performance metrics based on instructions per second (for example, MIPS), is a reasonable metric to compare the performance of two processors with the **same ISA** and the **same compiler**.

   (i) With optimal buffer insertion, the delay of a wire is quadratic in its length.

   (j) A victim cache is accessed in parallel with the main data cache, and thus increases the hit latency.

   (k) The primary advantage of a multi-level page table is faster address translation.

2. **[11 Points] Processor Performance**

(a) Assume a typical program has the following instruction type breakdown:
- 35% loads
- 10% stores
- 50% adds
- 3% multiplies
- 2% divides

Assume the current-generation processor has the following instruction latencies:
- loads: 4 cycles
- stores: 4 cycles
- adds: 2 cycles
- multiplies: 16 cycles
- divides: 50 cycles

If for the next-generation design you could pick one type of instruction to make twice as fast (half the latency), which instruction type would you pick? Why?

(b) Assume that a program executes one branch ever 4 instructions for the first 1M instructions and a branch every 8 instructions for the next 2M instructions. What is the average number instructions per branch for the entire program?

(c) You are the architect of a new line of processors, and you have the choice to add new instructions to the ISA if you wish. You consider adding a fused multiply/add instruction to the ISA (which is helpful in many signal processing and image processing applications). The advantage of this new instruction is that it can reduce the overall number of instructions in the program. The disadvantage is that its implementation requires extending the clock period by 10% and increases the CPI by 15%. Calculate under what conditions would adding this instruction lead to an overall performance increase.

(d) What qualitative impact would this modification have on the overall MIPS (millions of instructions per second) of the processor? What does this say about using MIPS as a performance metric?

3. **[8 Points] Short Answer**

   (a) What are two important hardware features present in today's microprocessors that are *unrelated* to performance, cost, or energy consumption.

   1.

   2.

   (b) Consider two different computing applications: a mobile wireless internet device and a server for processing transactions for a large bank. What are three design considerations that differentiates the design of a computer for these two application domains.

   1.

   2.

   3.

   (c) Give two different reasons why increasing the die (chip) size of a microprocessor increases the cost of the microprocessor. Recall that die (chips) are tiled on a wafer for fabrication.

   1.

   2.

   (d) What is the largest speedup that can be achieved by optimizing something that represents 80% of a program's execution time.

4. **[7 Points]   Energy**

(a) What are three hardware approaches for reducing the **dynamic energy** to perform a given computation:

1.

2.

3.

(b) What are three hardware approaches for reducing the **static energy** to perform a given computation:

1.

2.

3.

(c) What can be done in software to reduce energy consumption?

5. **[9 Points] Stream Buffers**

(a) What specific performance problem do stream buffers target?

(b) Why does each stream buffer have multiple entries?

(c) Would stream buffers in today's systems likely have more entries or fewer entries than when they were first proposed? Why?

(d) What is the benefit of multi-way stream buffers?

(e) As proposed, stream buffers do not place data directly into the cache. What is one advantage and two disadvantages of such a choice?
Advantage:

Disadvantage #1:

Disadvantage #2:

(f) What is the most significant potential *negative* performance impact caused by stream buffers?

(g) How might the presence of stream buffers impact the choice of cache block size?

6. **[8 Points] Cache Traffic**

   (a) How does increasing a cache's block size typically impact (increase, decrease, or unchanged) its fill traffic (measured in bytes)?

   (b) How does increasing a **write-back** cache's block size typically impact (increase, decrease, or unchanged) its write traffic (measured in bytes)?

   (c) How does increasing a **write-through** cache's block size typically impact (increase, decrease, or unchanged) its write traffic (measured in bytes)?

In the cache simulation assignment, a block in a write-back cache had a single dirty bit. Consider a new design for a write-back cache that has a dirty bit for each byte in the cache. The corresponding dirty bit is set whenever that byte is written. Upon replacement, only the dirty bytes are written back to the second-level cache.

   (a) What is the main advantage of such a cache? Specifically, how does it compare to traditional write-back and write-through caches?

   (b) How might such a cache effect the choice of block size?

   (c) Give two disadvantages of such a cache design:
   1.

   2.

7. **[11 Points] Virtual Memory Mechanics**. Consider a system with a 28-bit virtual address space, a 20-bit physical address space, and a 4KB (4096 byte) page size.

   (a) Number of bits in the page offset:

   (b) Number of bits in the virtual page number:

   (c) Number of bits in the physical page number:

   (d) Assuming a simple single-level page table, how large is the page table (in bytes)? (Ignore any valid bits, permission bits, etc.)

   (e) If the processor incorporated a 256-entry **fully-associative** TLB, how large (in bytes of storage) would it be?

   (f) If the processor incorporated a 256-entry **direct-mapped** TLB, how large (in bytes of storage) would it be?

   (g) Consider a two-level page table. Each level is indexed using an equal number of bits. How large is each second-level page table allocation?

   (h) Consider a program that accesses the virtual addresses listed below (in binary).

```
0000 0000 0000 0000 0000 0000 0001
0000 0000 0000 0000 0000 0001 0000
0000 0000 0000 0000 0001 0000 0000
0000 0000 0000 0001 0000 0000 0000
0000 0000 0001 0000 0000 0000 0000
0000 0001 0000 0000 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000
```

   How many physical pages would the OS allocate to the program?

   How much memory would the OS allocate for the second-level of the program's page table?

8. **[10 Points]** **Cache Conflicts**. This problem explores different ways to eliminate cache conflicts. Consider a processor with 16-bit addresses executing a program that repeatedly accesses the addresses given below (say, in a loop). The addresses are in binary with the most significant bit first, and the list is the same for all parts of the question.

   (a) **Larger Capacity.** For a direct-mapped cache with 256B cache blocks, what is the smallest cache size sufficient to avoid repeated conflict misses for the above accesses?

   ```
   0100 1000 0100 0000
   0001 0110 0011 0000
   0010 0110 0010 0000
   0001 0110 0001 0000
   ```

   (b) **Smaller Block Size.** For direct-mapped 1KB cache, what is the largest block size that is sufficient to avoid conflicts?

   ```
   0100 1000 0100 0000
   0001 0110 0011 0000
   0010 0110 0010 0000
   0001 0110 0001 0000
   ```

   (c) **Higher Associativity.** For a 1KB cache with 128B block size, what is the lowest associativity sufficient to avoid conflicts?

   ```
   0100 1000 0100 0000
   0001 0110 0011 0000
   0010 0110 0010 0000
   0001 0110 0001 0000
   ```

   (d) **Higher Associativity with Way Prediction.** For a 4-way set-associative 1KB cache with a 64B block size, what is the smallest way predictor (in bytes) that is guaranteed to avoid repeated way mis-predictions?

   ```
   0100 1000 0100 0000
   0001 0110 0011 0000
   0010 0110 0010 0000
   0001 0110 0001 0000
   ```

   (e) **Virtual to Physical Page Mapping.** Consider a system with 16-bit physical addresses, 16-bit virtual addresses, 256-byte pages, and a 0.5KB (512-byte) direct-mapped cache with 64-byte blocks. The given addresses are virtual addresses that are translated to physical addresses before each cache access. The mapping chosen by the OS is given below (right). Complete the missing mapping to avoid cache conflicts.

   ```
   0100 1000 0100 0000             /- VPN -\     /- PPN -\
   0001 0110 0011 0000             01001000  -> 00001001
   0010 0110 0010 0000             00010111  -> 00000101
   0001 0110 0001 0000             00100111  ->
   ```

End of exam

For reference:

Hex	Binary	Decimal
0x0	0000	0
0x1	0001	1
0x2	0010	2
0x3	0011	3
0x4	0100	4
0x5	0101	5
0x6	0110	6
0x7	0111	7
0x8	1000	8
0x9	1001	9
0xA	1010	10
0xB	1011	11
0xC	1100	12
0xD	1101	13
0xE	1110	14
0xF	1111	15

Power	Bytes	Kilobytes
2^{1}	2	
2^{2}	4	
2^{3}	8	
2^{4}	16	
2^{5}	32	
2^{6}	64	
2^{7}	128	
2^{8}	256	0.25 KB
2^{9}	512	0.5 KB
2^{10}	1,024	1 KB
2^{11}	2,048	2 KB
2^{12}	4,096	4 KB
2^{13}	8,192	8 KB
2^{14}	16,384	16 KB
2^{15}	32,768	32 KB
2^{16}	65,536	64 KB
2^{17}	131,072	128 KB
2^{18}	262,144	256 KB
2^{19}	524,288	512 KB
2^{20}	1,048,576	1024 KB

[Scratch space]