CS 351 Exam 3

Mon. 5/11/2015

Name: ____________________________________

Rules and Hints

- You may use one handwritten 8.5 × 11" cheat sheet (front and back). This is the only additional resource you may consult during this exam. *No calculators.*
- Where applicable, you may write your answers in the form *[mathematical expression][units]*. There is no need to actually do the arithmetic.
- Include step-by-step explanations and comments in your answers, and show as much of your work as possible, in order to maximize your partial credit.
- Answer these questions on the provided scratch paper. Clearly label each answer.

Grade

	Your Score	**Max Score**
Problem 1: Cache tracing		20
Problem 2: Address translation and page table sizing		20
Problem 3: Reliability metrics		8
Problem 4: Disks and RAID		22
Problem 5: Flash		8
Problem 6: Parallelizing code		10
Problem 7: GPU vs. CPU		12
Total		**100**

Problem 1: Cache tracing (20 points)

You have a hilariously tiny 2-way set-associative, write-back cache that stores 64B of data in 8-byte blocks. Memory addresses are 12 bits.

Part A (4 points)

How many blocks does this cache have? How many sets does it have?

Part B (10 points)

Draw the final state of the cache (data and metadata) after the following sequence of references (which are all writes). Assume that all blocks in the cache are initially invalid.
4, 5, 6, 7, 20, 21, 22, 23, 64, 65, 128, 129, 4, 5, 6

Part C (3 points)

How many bytes will be written back to the next level of the memory hierarchy during this sequence of operations? If there are 2 write-back operations to a particular address, you should count this as 2 bytes. Explain.

Part D (3 points)

If this were a write-through cache, how many bytes would be written to the next level of the memory hierarchy? Explain.

Problem 2: Address translation and page table sizing (20 points)

Your memory system has hilariously tiny virtual and physical address spaces, as follows:

- 64B pages
- 12-bit virtual addresses
- 16-bit physical addresses

Part A (4 points)

How many virtual pages does each process have, and how many physical pages does the system have?

Part B (4 points)

If each page has 3 protection bits, how large is a page table entry? Show your work.

Part C (6 points)

Draw Process X's page table if all of its pages are currently mapped to physical memory, in the following pattern: virtual page 0 is mapped to the largest physical page, virtual page 1 is mapped to the next largest, etc.

Part D (6 points)

Based on your page table from Part D, translate each of these virtual addresses from Process X to the corresponding physical address:

- 0x97f
- 0x953
- 0x800

Problem 3: Reliability (8 points)

For each scenario below, explain whether it will *increase*, *decrease* or have *no effect* on each metric for the servers in a datacenter. Consider the scenarios independently.

Part A (4 points)

A flaky network switch that used to crash every 3 weeks now crashes every 4 weeks instead.

- Effect on MTTF:
- Effect on MTTR:
- Effect on MTBF:
- Effect on availability:

Part B (4 points)

The disks in a storage array are replaced with disks that are identical in all respects except that their capacity is larger, so they take longer to reconstruct after a failure.

- Effect on MTTF:
- Effect on MTTR:
- Effect on MTBF:
- Effect on availability:

Problem 4: Disks and RAID (22 points)

You have 8 TB of unique, non-redundant data. You are planning to store it on an array of disks, where each disk has the following characteristics:

- Average seek time of 4 ms
- Rotation speed of 10,000 rpm
- Transfer rate of 100 MB/s
- 1 TB capacity

Part A: Disk performance (8 points)

How long will it take you to do a 512B read on a single disk? How long will it take you to do a 1 MB read (using 10^6 for MB)?

Part B: RAID 1 (7 points)

Answer the following questions about constructing a RAID 1 array for your data from these disks:

- How many *total* disks will you need?
- If small accesses are evenly distributed across your data, how many small reads can you do in parallel on this array?
- If small accesses are evenly distributed across your data, how many small writes can you do in parallel on this array?
- Based on your answers to Part A, how long will it take to do a single 1 MB read on this array?
- Based on your answers to Part A, how long will it take to do a single 1 MB write on this array?

Part C: RAID 5 (7 points)

Answer the following questions about constructing a RAID 5 array for your data from these same disks, with a chunk size of significantly less than 1 MB:

- How many *total* disks will you need?
- If small accesses are evenly distributed across your data, how many small reads can you do in parallel on this array?
- If small accesses are evenly distributed across your data, how many small writes can you do in parallel on this array?
- Based on your answers to Part A, how long will it take to do a single 1 MB read on this array?
- Based on your answers to Part A, how long will it take to do a single 1 MB write on this array?

Problem 5: Flash (8 points)

List and briefly explain the three basic operations on a solid-state disk. State the minimum granularity of each operation (e.g. byte, page, block...). Indicate which operation takes the longest.

Problem 6: Parallelizing code (10 points)

Consider the following code. Each function in this code takes a long time to run, and the functions *f1* through *f5* have no side effects.

```
A = f1(X, Y);     // Line 1
B = f2(A, X);     // Line 2
C = f3(A);        // Line 3
D = f4(Y);        // Line 4
E = f5(B, C);     // Line 5
```

Part A (4 points)

Draw a dependency graph with one node for each line.

Part B (3 points)

At most, how many threads can this code take advantage of?

Part C (3 points)

Label the nodes of your dependency graph with a set of times that will ensure the maximum speedup when the code is parallelized. (For example: Line 1 takes 3 units of time, Line 2 takes 4 units, etc.).

Problem 7: GPU vs. CPU (12 points)

For this problem, consider a loop of the form

```
for (unsigned long i = 0; i < N; i++) {
    // Loop contents
}
```

Assume that this loop operates on an array A of size N.

Part A: Mapping to threads (6 points)

Assume that the loop contents are simply:

```
        A[i]++;
```

- If you parallelize this loop on a 4-core **CPU** that supports 2 hardware threads per core, how many threads would you create? Which values of i would you assign to each thread, and why? State your answers in terms of N, if applicable.
- If you parallelize this loop on an nVidia **GPU** that supports 1024 threads per block, how many threads (total) would you create, and how would you arrange them in terms of blocks and threads per block? Which values of i would you assign to each thread, and why? State your answers in terms of N, if applicable.

Part B: Suitability for GPU (6 points)

- Give or describe an example of loop contents that would be *less suited* to a GPU than `A[i]++;`. Explain.
- Give or describe an example of loop contents that would be *more suited* to a GPU than `A[i]++;`. Explain.