Andrew login ID:______________________________________

Full Name:______________________________________

Recitation Section:______________________________________

CS 15-213, Spring 2009
Exam 2

Tues., April 7th, 2009

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name, Andrew login ID, and recitation section (A–J) on the front.
- **Do not write any part of your answers outside of the space given below each question. Write clearly and at a reasonable size. If we have trouble reading your handwriting you will receive no credit on that problem.**
- The exam has a maximum score of 100 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is OPEN BOOK. You may use any books or notes you like. No calculators or other electronic devices are allowed.
- Good luck!

1 (20):
2 (15):
3 (15):
4 (15):
5 (10):
6 (25):
TOTAL (100):

Problem 1. (20 points):

We consider the following program:

```
typedef double matrix[2][8]

double comp(matrix A) {
  int i;
  double t = 1000.0;

  for (i = 0; i < 7; i++) { // note: 7 not 8 because of the boundary
    t = t/(A[0][i] + A[0][i+1]);
    t = t/(A[1][i] + A[1][i+1]);
  }

  return t;
}
```

We assume that a double requires 8 bytes, and that the array is cache aligned (that is $A[0][0]$ is mapped to the first set and the first position in a cache block. Further, A has been initialized to contain only positive numbers.

We assume a cold cache and ignore i and t in the cache analysis (they are held in registers). Recall that the miss rate is defined as $\frac{\#\text{misses}}{\#\text{accesses}}$.

Hint: It helps to draw the cache and the array.

1. How many times is A accessed in this program?
2. The cache is direct mapped, has a size of 64 bytes, and 4 sets.
   (a) How many doubles fit into one cache block?
   (b) What is the miss rate of the above program?
3. The cache is 2-way set associative, has a size of 128 bytes, and 4 sets.
   (a) How many doubles fit into one cache block?
   (b) What is the miss rate of the above program?

Problem 2. (15 points):

1. What is the purpose of the TLB?
   (a) Offers read/write protection to critical parts of memory
   (b) Speeds up the process of swapping pages to and from disk storage
   (c) Acts as a cache in the virtual to physical address translation
   (d) Manages the two-level page table system
2. You have a 32-bit virtual memory system with 4KB page frames, with a TLB with 4 sets, each of which is 8-way set associative. How many bits of the virtual address form the TLBi (TLB index)?
   (a) 2
   (b) 4
   (c) 8
   (d) 12
3. Which of these features in a system **best** justifies the use of a two level page table structure, as opposed to a one level page table structure?
   (a) Small page sizes
   (b) Frequent memory accesses
   (c) High degree of spatial locality in programs
   (d) Sparse memory usage patterns
4. Which section of an ELF file is initialized to zeros upon program startup?
   (a) .data
   (b) .rodata
   (c) .text
   (d) .bss
5. Which of the following is true?
   (a) Every signal that is sent is also received
   (b) Signals are always received immediately since they cause an interrupt
   (c) Signals can only be received after a context switch
   (d) Signals can only be received upon return from system mode

6. Consider a 32-bit x86 computer with 2-level address translation; page directory, page tables, and pages are all 4KB in size. How much total memory will be used by the page directory and page table(s) of a process that has the minimum amount of address space mapped for the following addresses to be valid: 0x01234000, 0x01235678, 0x01234567, 0xdeadbeef.

   (a) 8KB
   (b) 12KB
   (c) 16KB
   (d) 20KB

7. Consider a theoretical computer architecture with 50-bit virtual addresses and 16kb pages. What is the maximum number of levels of page tables that could be used in the virtual memory system?

   (a) 16
   (b) 36
   (c) 2
   (d) 50

8. If a process contains if(!fork()) execve(...), what aspects of virtual memory do the child and parent share after execve()?

   (a) stack space
   (b) heap space
   (c) read only space
   (d) nothing

9. Which of the following caches (all having the same size) will have the least number of conflict misses:

   (a) A direct-mapped cache
   (b) A fully associative cache
   (c) The associativity does not matter for conflict misses

10. Give two functions that can return twice.

11. A handler will not stop which of the following signals

   (a) SIGALRM
   (b) SIGKILL
   (c) SIGTERM
   (d) SIGABRT

Problem 3. (15 points):

Suppose we have the following two .c files:

alarm.c

```
int counter;

void sigalrm_handler (int num) {
  counter += 3;
}

int main (void) {
  signal(SIGALRM, &sigalrm_handler);
  counter = 4;
  alarm(1);
  sleep(1);
  counter -= 2;
  exit(counter);
  return 0;
}
```

fork.c

```
int counter;

void sigchld_handler(int num) {
  int i;
  wait(&i);
  counter += WEXITSTATUS(i);
}

int main (void) {
  signal(SIGCHLD, &sigchld_handler);
  counter = 1;
  if (!fork()) {
    counter++;
    execl("alarm", "alarm", NULL);
  }
  sleep(2);
  counter *= 2;
  printf("%d\n", counter);
  exit(0);
}
```

Assume that all system calls succeed and that all C arithmetic statements are atomic.

The files are compiled as follows:

```
gcc -o alarm alarm.c
gcc -o fork fork.c
```

Suppose we run `./fork` at the terminal. What are the possible outputs to the terminal?

Problem 4. (15 points):

Harry Q. Bovik builds and runs a C program from the following two files:

```
----------------------------------------
main.c:

#include <stdio.h>

long a = 1;
const long b = 2;
long c;
long d = -1;

int main(int argc, char *argv[]) {
    printf("a: %p\nb: %p\nc: %p\nd: %p\n", &a, &b, &c, &d);
    printf("%ld\n", c);
    return 0;
}
----------------------------------------
data.c:

unsigned int c[2] = {...};
----------------------------------------
```

And sees this output:

```
a: 0x601020
b: 0x400650
c: 0x601030
d: 0x601028
4294967297
```

Harry was expecting the variables to be in order, one after another. Obviously, he was very wrong. Help him figure out what's happening using your knowledge of linking and executable layouts. Be specific but concise with your answers.

(a) How many symbols does `main.c` generate in the executable program's symbol table?

(b) What are the strong symbols from `main.c`, and what are the weak symbols from `main.c`?

(c) Note the address of b. Why is it far removed from the addresses of the other variables?

(d) Why is c located after d in memory, even though it's before d in Harry's program?

(e) Note the output given by the final printf. Was Harry compiling and running the code on x86 or x86-64? How do you know?

(f) Given that $4294967297 = 2^{32} + 1$, what would be output by

```
printf("{%d, %d}", c[0], c[1]);
```

if it were executed in data.c?

Problem 5. (10 points):

Assume a System that has

1. A two way set associative TLB
2. A TLB with 8 total entries
3. 2^8 byte page size
4. 2^{16} bytes of virtual memory
5. one (or more) boats

TLB			
Index	Tag	Frame Number	Valid
0	0x27	0xC6	1
	0x29	0x73	1
1	0x11	0xFF	0
	0x0A	0xEC	1
2	0x29	0xCD	1
	0x3A	0xAB	1
3	0x32	0xFB	0
	0x23	0x46	0

Use the TLB to fill in the table. Strike out anything that you don't have enough information to fill in.

Virtual Address	Physical Address
0x8F0F	
	0x4690
	0x7300
0x2933	
0x2839	

Problem 6. (25 points):

Suppose a processor can support up to 64-bits of physical memory, but for binary backwards-compatibility uses 32-bit virtual addresses. You have been assigned the task of designing a virtual memory scheme for this processor.

You must meet the following requirements:

1. The scheme must be able to map any 32-bit virtual address to any 64-bit physical address.
2. The scheme must use a 16KB page size.

You must meet the following performance constraints in addition to the above requirements:

1. The scheme must minimize the number of physical memory accesses needed to resolve a virtual address (that is, the scheme must minimize the number of levels of page tables).
2. The size of a page table at any level must not exceed 512 bytes.

This is your design. You may specify anything not already defined, including the size of page tables, the size of page table entries, etc. Design your system, and then answer the following questions:

How many levels of page tables are there?

What is the size of the table at each level?

Show the mapping of the bits of a virtual address to each level. To illustrate the notation we expect, please follow the convention shown in the example below (the mapping for standard two-level Intel x86-32 virtual memory). Notice that the first set of bits maps into the top level page table, the second set maps into the second level page table, etc.

```
 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                             |                             |                                   |
|           VPN 1             |           VPN 2             |              OFFSET               |
|                             |                             |                                   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Draw your mapping here:

```
 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                                                                               |
|                                                                                               |
|                                                                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

What is the size of a page table entry at each level?