Department of Computer Science
Operating Systems Qualifying Exam
Spring, 2009

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed. Please note the following instructions:

- show your work whenever appropriate. There can be no partial credit unless you show how you derived your answers
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

1. (25 points) Suppose some number of processors are cooperating to solve a problem. To obtain a unit of work, a processor calls the function

```
int getworkunit() {
    myworkunit = nextunit;
    nextunit = nextunit + 1;
    return myworkunit;
}
```

   (a) No variable declarations are shown in this code. Of the two variables (`myworkunit` and `nextunit`) which needs to be local? Which needs to be shared among the processors?

   (b) Using the standard memory transaction notation (a refresher is on the last page of this exam), show the memory transactions on the shared variable(s) when P1 executes this code. Assume the initial value of any shared variable is `0`.

   (c) Again using the same notation, show how processors P1 and P2 can attempt to obtain work simultaneously, and end up having the same value returned, under the serial (or strict) memory consistency model.

   (d) Add binary semaphore operations to this code to ensure that it works correctly (*i.e.* as processors call it, successive integers are returned).

2. (25 points) Suppose processes P1, P2, and P3 execute the following memory transactions on an Alewife machine:

P1	r(x)0	w(x)1			w(x)1
P2			r(x)1		
P3				r(x)1	

   Assume `x` is homed on a fourth processor P4, and initially all the caches are empty. Show all the messages that need to be exchanged among the processors, and the state changes to `x` as a result.

3. (20 points) Some computer system uses 32 bit virtual and physical addresses. A page is 4K. The level two cache has a 2MB capacity, uses a 16 byte line, and is two-way set-associative.

   If we use page coloring, how many colors do we have?

4. (15 points) The following string includes a message and CRC bits, computed using a generating polynomial of $1+X+X^3$. Does the message contain any errors? The message is given with its highest-order bit on the left.

```
1000000010101
```

5. (15 points) The Happy Go Lucky filesystem (HFS) creates a new file by:[1]

   1: allocating an inode from its free list, cached in memory.

   2: inserting a directory entry in the directory's data block (also cached in memory).

   3: initializing the new file's metadata (also cached in memory).

   4: returning to the user.

   5: At some later point HFS writes the cached copies of the modified directory block, the inode, and the free list to disk. The order in which these writes take place is an artifact of the disk scheduling algorithm, so they may happen in any order.

   Recall that disks can only atomically write a single sector at a time.

   (a) Suppose the system crashes during file creation. Give two possible file system inconsistencies that can result depending on the order of writes and the point in the process in which the crash occurs (there are actually more than just two possible inconsistencies here; you're only being asked to describe two of them).

   (b) Describe how a check of the filesystem can be made when the computer reboots, which can identify and repair these inconsistencies. "Repair" can mean either complete the operation (so the new, empty file exists in the recovered file system) or revert (so the state of the filesystem is the same as if the file creation had never been attempted).

[1] inspired by a question I found on an old Stanford University qualifying exam

A Memory Operation Notation

This appendix provides a brief review of the memory operation notation being used in this exam.

A.1 Operations

In this notation, a memory operation is represented with the characters `op(var)val` where

- `op` is the operation: `R` is used for a read operation, and `W` is used for a write operation.
- `var` is the variable to be read or written.
- `val` is the value that is read or written.

So, using this notation, `W(x)1` means "write the value 1 to the variable `x`", and `R(y)2` means "read the value of `y` from memory. The value memory returned is 2."

A.2 Sequences of Operations

A sequence of operations performed by a single processor is written horizontally on a line:

```
R(x)0  W(x)1
```

states that a processor first reads the value of `x`, and obtains a 0. It then writes `x` to memory again; the value it writes is a 1.

An example of a line of C code which might result in this sequence of memory operations is

```
x = x + 1;
```

As we can see, the addition performed by the processor has been abstracted away: all that is shown in the memory operation notation is the actual memory operations performed.

A.3 Multiprocessor Memory Operations

Finally, if we have several processors performing memory operations, we represent each processor's operations on a line, like this:

P1	`R(x)0`	`W(x)1`	
P2	`R(x)0`		`W(x)2`

In this example, P1 and P2 simultaneously read `x`, and both obtain a value of 0. In the second time unit P1 writes a new value to `x` (a 1), and on the third time unit P2 writes a new value to `x` (a 2). The final value of `x` in this example is 2.

It's important to keep in mind that there is nothing saying that this is the only possible sequence of operations – from the information provided, P2 could equally well have written its result first, or it might not have even read `x` until after P1 had written it. The notation merely specifies what did happen, it *doesn't* say there's any reason it needed to happen that way.

A.4 Extensions

There are a variety of extensions to this notation, to either insert synchronization or to explicitly represent communication. Those are beyond the scope of what we need for this!