Department of Computer Science
Operating Systems Qualifying Exam
Fall, 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:

- Write your banner ID, but not your name, on every piece of paper you turn in.
- Diagrams used in your answers must be sufficiently clear that they can be unambiguously interpreted. This especially includes timing relationships.
- 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. (20 points) A virtual machine environment will typically represent its disk as a file in the host machine's filesystem. Suppose the guest machine initiates a disk write, and the host crashes before the write has been completed on the physical hardware.

   If the host machine uses a journalling filesystem that guarantees host filesystem metadata integrity but not host filesystem data integrity, can the guest filesystem suffer a loss of metadata integrity?

2. (15 points) The following message, including information and CRC check bits, arrives:

   ```
   101010101010
   ```

   The left-most bit shown in the line above arrives first, and is the most-significant bit. The CRC polynomial is $1+x+x^3$

   (a) Which bits are the information, and which are the check bits?

   (b) Has the message been corrupted?

3. (30 points) Assume the following code is executed on an Alewife using the LimitLESS protocol:

P1	P2	P3
`x=1;`		
	`y = x + 1;`	
		`y = x + 2;`
`x = y;`		

   The code is executed as shown in the table; *i.e.* first processor P1 executes its assignment, then processor P2 executes its assignment, and so forth. There are no additional synchronization operations; it "just happens" this way. Variables `x` and `y` are both homed on processor P1. Initially, their processor sets are empty, and their initial values are all 0.

   (a) Using the standard memory operation notation, express the operations that occur as a result of this code (a description of the notation follows this exam for reference).

   (b) Give all the messages that are needed for executing each of the memory operations in this code. Be sure to specify the source node, destination node, and message type for each message.

4. (20 points) Some computer system uses a uses a hashed page table for virtual memory, with the following characteristics (as usual for these questions, the sizes given are unrealistically small in order to make the problem tractable):

   - Total size of virtual memory: 256 bytes
   - Size of a page: 16 bytes
   - Number of entries in primary hashed page table: 8

   (a) How wide is a virtual address?

   (b) How wide is the offset field?

   (c) Suppose we use a standard modulo function to perform the hash for a lookup in the hashed page table. What is the hash function?

   (d) Suppose the primary hash table and the external collision resolution table have the following contents:

Primary Hash Table

Index	VPN	PFN	Next
0	a18		
1			
2	012	210	5
3	bf3	13c	0
4	ccc	900	6
5	135	abc	9
6			
7	ac7	fff	3

Collision Resolution Table

Index	VPN	PFN	Next
0	bfb	a79	
1	59d	000	
2			
3	33f	dd9	a
4			
5	89a	047	8
6	194	f90	b
7			
8	123	339	
9	475	179	1
a	b47	231	
b	abc	ca0	

In this table, invalid entries are shown as blank. All numbers in the table are expressed in hexadecimal.

What will be the result of trying to translate each of the following virtual addresses to a physical address?

   i. 0123

   ii. 3210

   iii. 89ab

   iv. fedc

5. (15 points) Computer schedulers being tuned for good interactive response typically use a smaller time quantum than schedulers being tuned for good computing throughput. Why? Would you expect a web server to show better performance with a shorter or a longer quantum?

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!