Systems Qualifying Exam, January 1998
Computer Science Dept., Indiana University, Bloomington

Operating Systems

Problem 1.

a. What is a trap? What are the major uses of traps in a multi-user multi-tasking operating system?

b. What is an interrupt? What are the major uses of interrupts in a multi-user multi-tasking operating system?

Problem 2. Design a system in which threads $T_0, T_1, T_2, \ldots, T_N$ take turns performing actions. Specifically, the sequence of actions that should occur is:

$$\texttt{action}(0), \texttt{action}(1), \ldots, \texttt{action}(N), \texttt{action}(0), \texttt{action}(1), \ldots, \texttt{action}(N), \ldots$$

Thread T_i is responsible for repeatedly performing $\texttt{action}(i)$. The code for T_i is:

```
while (TRUE) {
  WaitForTurn(i);
  action(i);
  FinishedTurn(i)
};
```

Give declarations (and initializations, if necessary) for your global variables, and give code for `WaitForTurn(i)` and `FinishedTurn(i)`. Your code should use semaphores (not locks, condition variables, monitors, etc.). It should not use busy-waiting.

Problem 3. This problem is about computer systems with a memory mangement unit (MMU) and operating system (OS) that support paged virtual memory. Although the page size is constrained by the MMU, the OS has the freedom to treat each chunk of (say) 4 consecutive pages as a single page, and each chunk of 4 consecutive page frames as a single page frame. This gives some but not all of the benefits of having a truly larger page size.

a. Describe at least two benefits of truly larger pages that this scheme does provide.

b. Describe at least one benefit of truly larger pages that this scheme does not provide.

Problem 4. Recall that in standard UNIX filesystems, i-nodes are stored in a special area (or areas) on disk, and that each directory entry contains a name and an i-node number. Suppose, instead, that i-nodes are stored directly in directories; thus, each directory entry contains a name and an entire i-node. Note: With this proposal, the root directory must be handled specially, because it is not contained in any directory; this is not a significant disadvantage.

a. Give at least one advantage of this variant.

b. Give at least two disadvantages of this variant.

Problem 5. Recall that standard centralized UNIX file systems keep i-nodes for open files in memory, in the "i-node table". Each entry in the open file table contains an index into the i-node table, so when the file system is processing a file system call, it can easily find the appropriate i-node. Recall that NFS servers are stateless, so they can't keep track of which files are open, so they can't use the obvious policy of keeping in memory i-nodes for files that are open by some client.

a. Suggest a method for NFS servers to decide which i-nodes to keep in memory.

b. Suggest a method for NFS servers to find the necessary i-node when a request arrives.

c. Compare the methods proposed in a. and b. to what a local UNIX file system does. What are the advantages and disadvantages?

RISC and CISC Architecture Efficiency

The purpose of this question is to examine the bus traffic, memory efficency, and effect on hazards and pipelining of five styles of architecture:

```
        Accumulator

        Memory-Memory (all three instruction operands are in memory)

        Stack (all operands are on the top of the stack.  only push and pop
           access memory.  all other instructions remove their operands
           from the top of the stack, and return the result to the stack.
           the two top entries in the stack are implemented as registers.
           all other stack entries are kept in memory.)

        Load/Store (all operations occur in registers.  register-to-register
                instructions have three register operands. there are 16
                registers, with register specifiers four bits long.)

        Stream (instructions and operands are fetched in 32-bit packets
            from memory. results are returned in 32-bit packets to
                memory. instructions, operands, and results may span packets.
                instructions are a variable number of BITS long:  they are
                Huffman encoded -- most frequent instructions use fewest bits.)
```

Constraints:

All memory addresses are two bytes long (16 bits).

All data operands are four bytes long (32 bits).

Excepting the Stream architecture, the opcode is one byte long (8 bits).

Excepting the Stream architecture, the instruction is always an integral number of bytes long.

There are no optimizations to reduce memory traffic (most importantly, there is no cache or prefetching). The variables A, B, C, and D are initially in memory.

Invent your own assembly language mnemonics and write the most efficient assembly language code you can for the high-level language fragments given.

1. 25% Write the five code sequences for:

$$A = B + C$$

For each sequence compute the number of instruction BITS fetched and memory data BYTES transferred. Which architecture is most efficient measured by code size? By total bandwidth (code + data)?

2. 45% Write the five code sequences for:

```
A = B + C
B = A + C
D = A - B
```

For each sequence compute the number of instruction BITS fetched and memory data BYTES transferred. Which architecture is most efficient measured by code size? By total bandwidth (code + data)? If the answers differ from part (1), explain briefly.

3. 30% Assume that each of the five architectures for which you wrote code sequences is pipelined. Analyze each of the five code sequences from part (2) for hazards. For each architecture describe how the hazards could be removed by adding hardware or by optimizing software. Pay careful attention to the Stream architecture, which fetches and writes 32-bit packets that can be spanned by variable-length instructions, fixed-length operands, and fixed-length results.

Use your determination of code fetches, total bandwidth, and ease of pipelining to pick the most RISC-like and the most CISC-like architecture. Briefly justify your choice.