System Calls

SPIM provides a small set of operating-system-like services through the system call (`syscall`) instruction. To request a service, a program loads the system call code (see Figure A.9.1) into register `$v0` and arguments into registers `$a0`–`$a3` (or `$f12` for floating-point values). System calls that return values put their results in register `$v0` (or `$f0` for floating-point results). For example, the following code prints "`the answer = 5`":

```
        .data
str:
        .asciiz  "the answer = "
        .text

        li       $v0, 4   # system call code for print_str
        la       $a0, str # address of string to print
        syscall           # print the string

        li       $v0, 1   # system call code for print_int
        li       $a0, 5   # integer to print
        syscall           # print it
```

The `print_int` system call is passed an integer and prints it on the console. `print_float` prints a single floating-point number; `print_double` prints a double precision number; and `print_string` is passed a pointer to a null-terminated string, which it writes to the console.

The system calls `read_int`, `read_float`, and `read_double` read an entire line of input up to and including the newline. Characters following the number are ignored. `read_string` has the same semantics as the UNIX library routine `fgets`. It reads up to $n - 1$ characters into a buffer and terminates the string with a null byte. If fewer than $n - 1$ characters are on the current line, `read_string` reads up to and including the newline and again null-terminates the string.

Warning: Programs that use these syscalls to read from the terminal should not use memory-mapped I/O (see Section A.8).

`sbrk` returns a pointer to a block of memory containing n additional bytes. `exit` stops the program SPIM is running. `exit2` terminates the SPIM program, and the argument to `exit2` becomes the value returned when the SPIM simulator itself terminates.

`print_char` and `read_char` write and read a single character. `open`, `read`, `write`, and `close` are the standard UNIX library calls.