SPIM S20: Syscalls

James R. Larus
Edited by Tom Kelliher

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 into register `$v0` and the arguments into registers `$a0`...`$a3` (or `$f12` for floating point values). System calls that return values put their result in register `$v0` (or `$f0` for floating point results). For example, to print "`the answer = 5`", use the commands:

```
        .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
```

Service	System Call Code	Arguments	Result
print_int	1	`$a0` = integer	
print_float	2	`$f12` = float	
print_double	3	`$f12` = double	
print_string	4	`$a0` = string	
read_int	5		integer (in `$v0`)
read_float	6		float (in `$f0`)
read_double	7		double (in `$f0`)
read_string	8	`$a0` = buffer, `$a1` = length	
sbrk	9	`$a0` = amount	address (in `$v0`)
exit	10		
print_character	11	`$a0` = character	
read_character	12		character (in `$v0`)
open	13	`$a0` = filename, `$a1` = flags, `$a2` = mode	file descriptor (in `$v0`)
read	14	`$a0` = file descriptor, `$a1` = buffer, `$a2` = count	bytes read (in `$v0`)
write	15	`$a0` = file descriptor, `$a1` = buffer, `$a2` = count	bytes written (in `$v0`)
close	16	`$a0` = file descriptor	0 (in `$v0`)
exit2	17	`$a0` = value	