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_char	11	$a0 = char	
read_char	12		char (in $a0)
open	13	$a0 = filename (string), $a1 = flags, $a2 = mode	file descriptor (in $a0)
read	14	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars read (in $a0)
write	15	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars written (in $a0)
close	16	$a0 = file descriptor	
exit2	17	$a0 = result	

FIGURE A.9.1 System services.

```
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.