# System Call

- SPIM is a MIPS processor simulator, and the name is a reversal of the letters “MIPS”
- SPIM provides a small set of operating-system-like services through the MIPS system call (syscall) instruction
  - The service could be: print something on the console window, read you input from the console, etc.
  - These syscalls are NOT functionalities provided by MIPS language!
  - They are functionalities provided by the simulator.
- To request a service, the procedure is:
  - Load the service number in register $v0
  - Load argument values (if any), in $a0, ..., $a3 (or $f12 for floating point values)
  - Issue the SYSCALL instruction
  - Return values (if any), and put the result in register $v0 (or $f0 for floating point results)
- May destroy $v0–$v1, $a0–$a3, $t0–$t9, $ra
- But always preserves $s0–$s7

# Some SPIM System Calls

- Complete list in Patterson and Hennessey's book, Appendix A-44

| Service | $v0 | Arguments | Result |
|---|---|---|---|
| print_int | 1 | $a0 | *none* |
| print_string | 4 | address in $a0 | *none* |
| read_int | 5 | *none* | $v0 |
| read_string | 8 | into $a0, $a1 length | *none* |
| sbrk | 9 | allocate $a0 bytes | starting at $v0 |
| exit | 10 | *none* | *never!* |

# Example: Input, Output, Arithmetic

```
        .data
nl:     .asciiz "\n"
        .text
        .globl main
main:   li $v0, 5
        syscall          # read_int
        add $a0, $v0, $v0
        li $v0, 1
        syscall          # print_int
        la $a0, nl
        li $v0, 4
        syscall          # print_string
        li $v0,10
        syscall          # exit
```