1 Administrivia

Announcements

Assignment

Read 2.7.

From Last Time

Intro to Unix.

Outline

1. Using SPIM.

2. Lab exercise.

Coming Up

Procedure calls in assembly.
2 Using SPIM

Things to notice:

1. Structure of a MIPS assembly language program.

2. System calls: *syscall*. Exit from your program:

   ```
   li $v0, 10
   syscall
   ```

3. I/O:

   (a) Reading an integer.

   (b) Writing a string or integer.

4. Debugging:

   (a) Creating global labels with `.globl`.

   (b) Setting and hitting breakpoints. Continuing from a breakpoint.

   (c) Printing register values.

Example:

```plaintext
# addn.spim
# Input: A number of inputs, n, and n integers.
# Output: The sum of the n inputs.
# Demonstrates reading and writing integers.

# Register usage:
#   $t0: how many integers remain to be read.
#   $t1: sum of the integers read so far.

.data # Constants.
prmpt1: .asciiz "How many inputs? "
```
.text
.globl main
main: li $v0, 4 # Syscall to print prompt string.
la $a0, prmpt1
syscall
li $v0, 5 # Syscall to read an integer.
syscall # Result returned in $v0.
move $t0, $v0 # n stored in $t0.
li $t1, 0 # sum stored in $t1 -- clear it.
.globl while
while: blez $t0, endwhile # Read n integers.
li $v0, 4 # Prompt for next integer
la $a0, prmpt2
syscall
li $v0, 5 # Read next integer.
syscall
add $t1, $t1, $v0 # Increase sum by new input.
sub $t0, $t0, 1 # Decrement n.
b while
.endwhile: li $v0, 4 # Print result string.
la $a0, sum
syscall
move $a0, $t1 # Print sum.
li $v0, 1
syscall
li $v0, 4 # Print a newline character.
la $a0, nl
syscall
li $v0, 10 # Syscall to exit.
3 Lab Exercise

See lab handout.