SPIM simulator

CSE 410, Spring 2009
Computer Systems

http://www.cs.washington.edu/410
Reading and References

• See the Resources section on the SPIM web page for documentation on SPIM
  » http://www.cs.wisc.edu/~larus/spim.html
  » (shortcut: google “spim” – that should be the first hit)

• In particular, Appendix B (from the textbook) and Getting Started with PCSpim are useful reading from that site
  » (Appendix A in older editions of the textbook)
SPIM simulator

- SPIM lets you write MIPS assembly language code and run it on a PC
- PCSpim should be installed on the machines in the A&S computer lab
- You can download versions for Windows and all varieties of *nix (including MacOS X) from the web site
  » Trade hints on the discussion list if you have troubles building/installing it
Spim display

• Register panel
  » register names and numbers

• Text segment (code) panel
  » note jump and link to “main” at [0x00400014]
  » your code defines the label “main”

• Data and Stack segment panel

• Message panel
### CPU Registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0 (t0)</td>
<td>00000000</td>
</tr>
<tr>
<td>R1 (t1)</td>
<td>00000000</td>
</tr>
<tr>
<td>R2 (t2)</td>
<td>00000000</td>
</tr>
<tr>
<td>R3 (t3)</td>
<td>00000000</td>
</tr>
<tr>
<td>R4 (t4)</td>
<td>00000000</td>
</tr>
<tr>
<td>R5 (t5)</td>
<td>00000000</td>
</tr>
<tr>
<td>R6 (t6)</td>
<td>00000000</td>
</tr>
<tr>
<td>R7 (t7)</td>
<td>00000000</td>
</tr>
</tbody>
</table>

### General Registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R16 (a0)</td>
<td>00000000</td>
</tr>
<tr>
<td>R17 (a1)</td>
<td>00000000</td>
</tr>
<tr>
<td>R18 (a2)</td>
<td>00000000</td>
</tr>
<tr>
<td>R19 (a3)</td>
<td>00000000</td>
</tr>
<tr>
<td>R20 (a4)</td>
<td>00000000</td>
</tr>
<tr>
<td>R21 (a5)</td>
<td>00000000</td>
</tr>
<tr>
<td>R22 (a6)</td>
<td>00000000</td>
</tr>
<tr>
<td>R23 (a7)</td>
<td>00000000</td>
</tr>
</tbody>
</table>

### Memory

- **Data Memory**
  - Initial values:
    - [0x00000000]...[0x10000000] 0x00000000
    - [0x10000000]...
    - [0x10000000]...

- **Stack Memory**
  - [0x7ffeefc] 0x00000000

- **Kernel Data Memory**
  - Initial values:
    - [0x90000000]...[0x90000000] 0x00000000
    - [0x90000001]...
    - [0x90000002]...

### Code Execution

Here is a segment of code being executed:

```assembly
[0x00400000] 0x81a40000 lw $4, 0($29) ; 175: lw $a0 0($sp) # argc
[0x00400004] 0x27a50004 addiu $5, $29, 4 ; 176: addiu $a1 $sp 4 # argv
[0x00400008] 0x24a60004 addiu $6, $5, 4 ; 177: addiu $a2 $a1 4 # envp
[0x0040000c] 0x3e040030 li $2, $4, 2 ; 178: sw $v0 $a0 2
[0x00400010] 0x3e023021 addu $6, $6, $2 ; 179: or $a2 $a2 $v0
[0x00400014] 0x3e010008 jal 0x03000024 [main] ; 180: jal main
[0x00400018] 0x00000000 nop ; 181: nop
[0x0040001c] 0x3d03000a ori $2, $0, $10 ; 183: li $v0 10
[0x00400020] 0x0000000c syscall ; 164: syscall # syscall 10 (exit)
[0x00400024] 0x3e000004 ori $2, $0, 4 ; 9: li $v0,4 # print_string code
[0x00400028] 0x3e041001 lui $4, 4067 [str] ; 10: la $s0,str # addr[str]
[0x0040002c] 0x0000000c syscall ; 11: syscall # print it
```

### Notes

- **SFIM Version** Version 7.2 of August 7, 2005
- **Copyright** 1990-2004 by James R. Lorus [lorus@cs.wisc.edu].
- **All Rights Reserved.**
Editing SPIM programs

- You can use any (plain) text editor you like to write the source code
  » Not Microsoft Word
- Textpad, notepad++, etc., on PC’s
- jEdit also provides a MIPS highlighter
- emacs can do anything including asm – but has a huge learning curve
hello.s

# This MIPS program uses a system call to print a string
.data
str:
    .asciiz "Hello World\n"
.text
main:
    li $v0,4  # print_string code
    la $a0,str  # addr(str)
    syscall  # print it
    jr $ra  # return
Assembly language basics

• SPIM reads a program written in MIPS assembly language, translates them to machine code (001011001100…), then executes them

• Programs have two sections
  » .data – storage for constants and variables
  » .text – program code

• Code must contain a label main:
  » Execution begins here; SPIM “calls” main
  » main should return when done

• Much, much more in the book
# load two numbers from memory into registers, add them,
# and store their sum

.data
one:   .word   1
two:   .word   2
sum:   .word   -1

.text
main:  lw       $t0,one
       lw       $t1,two
       add      $t2,$t0,$t1
       sw       $t2,sum
       jr       $ra # return
# load two numbers into registers and add them.
# this time the numbers are loaded directly
# from the instructions, not from memory

.text
main:    li      $t0,1
         li      $t1,2
         add    $t2,$t0,$t1
         jr      $ra  # return