

SPIM

Ref: Appendix A, Web Links

CompOrg Fall 2001 - SPIM

1

MIPS Simulation

- SPIM is a simulator
 - reads a MIPS assembly language program.
 - simulates each instruction.
 - displays values of registers and memory
 - supports breakpoints and single stepping
 - provides simple I/O for interacting with user.

CompOrg Fall 2001 - SPIM

2

SPIM Versions

- SPIM is the command line version.
- XSPIM is X-Windows version (Unix workstations).
- There is also a Windows version.

CompOrg Fall 2001 - SPIM

3

SPIM Program

- MIPS assembly language.
- Must include a label “main” – this will be called by the SPIM startup code (allows you to have command line arguments).
- Can include named memory locations, constants and string literals in a “data segment”.

CompOrg Fall 2001 - SPIM

4

General Layout

- Data definitions start with **.data** directive
- Code definition starts with **.text** directive
 - “text” is the traditional name for the memory that holds a program.
- Usually have a bunch of subroutine definitions and a “main”.

CompOrg Fall 2001 - SPIM

5

Simple Example

```
.data          # data memory
foo .word 0    # 32 bit variable

.text         # program memory
.align 2      # word alignment
.globl main   # main is global

main:
```

CompOrg Fall 2001 - SPIM

6

Data definitions

- You can define variables/constants with:
 - word : defines 32 bit quantities.
 - byte: defines 8 bit quantities
 - ascii: zero-delimited ascii strings
 - .space: allocate some bytes

CompOrg Fall 2001 - SPIM

7

Data Examples

```
.data
prompt: .ascii "Hi Dave"
msg:    .ascii "The answer is "
x:      .word 0
y:      .word 0
str:    .space 100
```

CompOrg Fall 2001 - SPIM

8

Simple I/O

- SPIM provides some simple I/O using the "syscall" instruction. The specific I/O done depends on some registers.
 - You set **\$v0** to indicate the operation.
 - Parameters in **\$a0**, **\$a1**

CompOrg Fall 2001 - SPIM

9

I/O Functions

\$v0	Function	Parameter
1	<code>print_int</code>	<code>\$a0</code> is int
4	<code>print_string</code>	<code>\$a0</code> is address of string
5	<code>read_int</code>	returned in <code>\$v0</code>
8	<code>read_string</code>	<code>\$a0</code> is address of buffer, <code>\$a1</code> is length

CompOrg Fall 2001 - SPIM

10

Example: Reading an int

```
addi $v0,$zero,5
syscall
```

```
# now $v0 has the integer typed by
# a human in the SPIM console
```

CompOrg Fall 2001 - SPIM

11

Printing a string

```
.data
msg: .asciiz "SPIM IS FUN"
      pseudoinstruction: load immediate
main: li $v0,4 ←
      la $a0,msg ←
      syscall
      pseudoinstruction: load address
```

CompOrg Fall 2001 - SPIM

12

SPIM subroutines

- The stack is set up for you – just use `$SP`
- You can view the stack in the data window.
- `main` is called as a subroutine (have it return using `jr $ra`).

CompOrg Fall 2001 - SPIM

13

Sample SPIM programs (on the web)

- `multiply.asm`: multiplication subroutine based on repeated addition and a test program that calls it.
- `fact.asm`: computes factorials using the multiply subroutine.
- `sort.asm`: the sorting program from the text.
- `strcpy.asm`: the strcpy subroutine and test code.

CompOrg Fall 2001 - SPIM

14
