Help Session
SPIM Processor Simulator

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.**

SPIM Versions

- **SPIM is the command line version.**
- **XSPIM is x-windows version (Unix workstations). (This is the version you will be using in the lab.).**
- **There is also a windows version. You can use this at home and it can be downloaded from:**

  http://www.cs.wisc.edu/~larus/spim.html.

- **Note that evaluation of your projects will take place using Unix; You can prepare your code at home OR in the lab, but it must be able to run on LINUX.**

Resources On the Web

- **There's a very good SPIM tutorial at**

  http://chortle.ccsu.edu/AssemblyTutorial/Chapter-09/ass09_1.html

- **In fact, there's a tutorial for a good chunk of the ISA portion of this course at:**

  http://chortle.ccsu.edu/AssemblyTutorial/tutorialContents.html

- **Here are a couple of other good references you can look at:**

  Patterson_Hennessy_AppendixA.pdf

And

http://babbage.clarku.edu/~jbreecher/comp_org/labs/Introduction_To_SPIM.pdf

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”.**

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”.**

Simple Example

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

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

main:
        lw      $a0,foo
```

Data Definitions

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

Data Examples

```
                .data
prompt:         .asciiz “Hello World\n”
msg:            .asciiz  “The answer is ”
x:              .space  4
y:              .word    4
str:            .space 100
```

MIPS: Software Conventions For Registers

0	zero	constant 0
1	at	reserved for assembler
2	v0	expression evaluation &
3	v1	function results
4	a0	arguments
5	a1	
6	a2	
7	a3	
8	t0	temporary: caller saves
. . .		
15	t7	
16	s0	callee saves
. . .		
23	s7	
24	t8	temporary (cont'd)
25	t9	
26	k0	reserved for OS kernel
27	k1	
28	gp	Pointer to global area
29	sp	Stack pointer
30	fp	frame pointer
31	ra	Return Address (HW)

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.**

I/O Functions

System call is used to communicate with the system and do simple I/O.
Load system call code into Register $v0
Load arguments (if any) into registers $a0, $a1 or $f12 (for floating point).
do: syscall
Results returned in registers $v0 or $f0.

code	service	Arguments	Result	comments
1	print int	$a0		
2	print float	$f12		
3	print double	$f12		
4	print string	$a0		(address)
5	read integer		integer in $v0	
6	read float		float in $f0	
7	read double		double in $f0	
8	read string	$a0=buffer, $a1=length		
9	sbrk	$a0=amount	address in $v0	
10	exit			

Example: Reading an int

```
    li      $v0,5           # Indicate we want function 5
    syscall

# Upon return from the syscall, $v0 has the integer typed by
# a human in the SPIM console

# Now print that same integer
    move  $a0,$v0       # Get the number to be printed into register
    li        $v0,1         # Indicate we're doing a write-integer
    syscall
```

Printing A String

```
                .data
msg:            .asciiz  “SPIM IS FUN”
                .text
                .globl
main:           li $v0,4
                la $a0,msg
                syscall
                jr       $ra
```

pseudoinstruction: load immediate

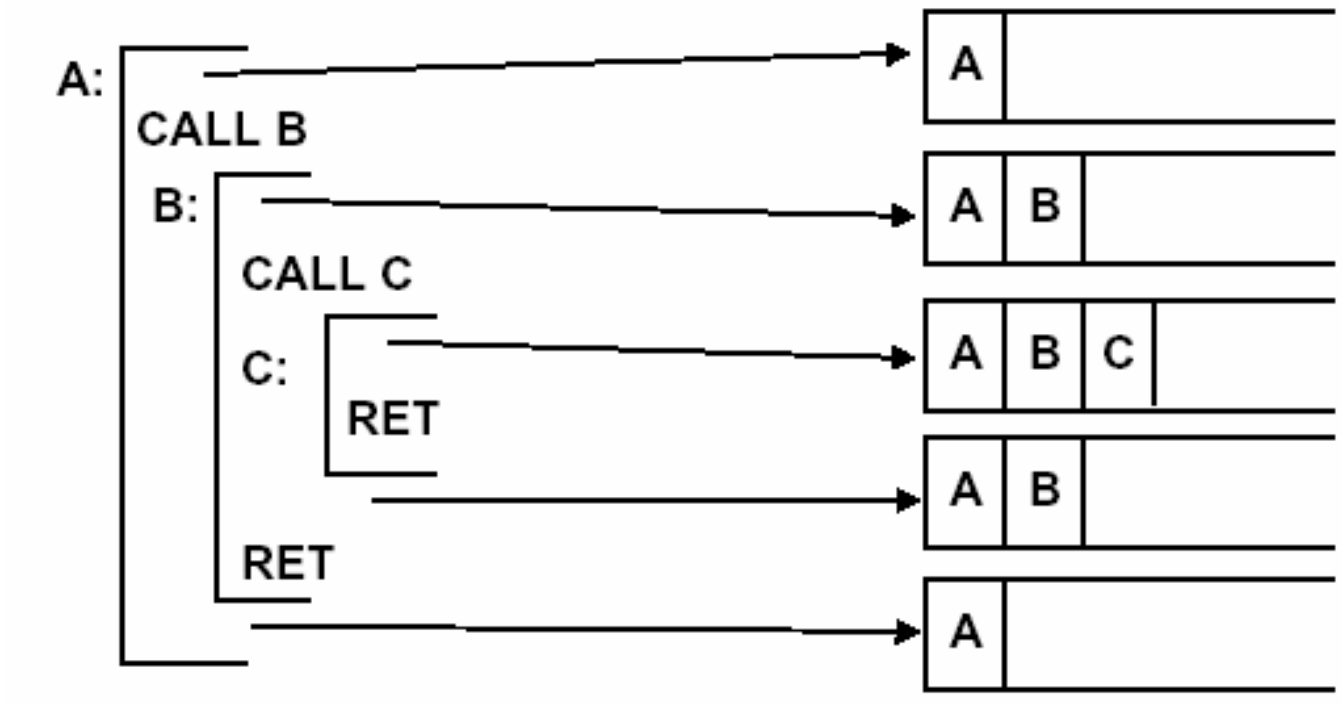
pseudoinstruction: load address

SPIM Subroutines

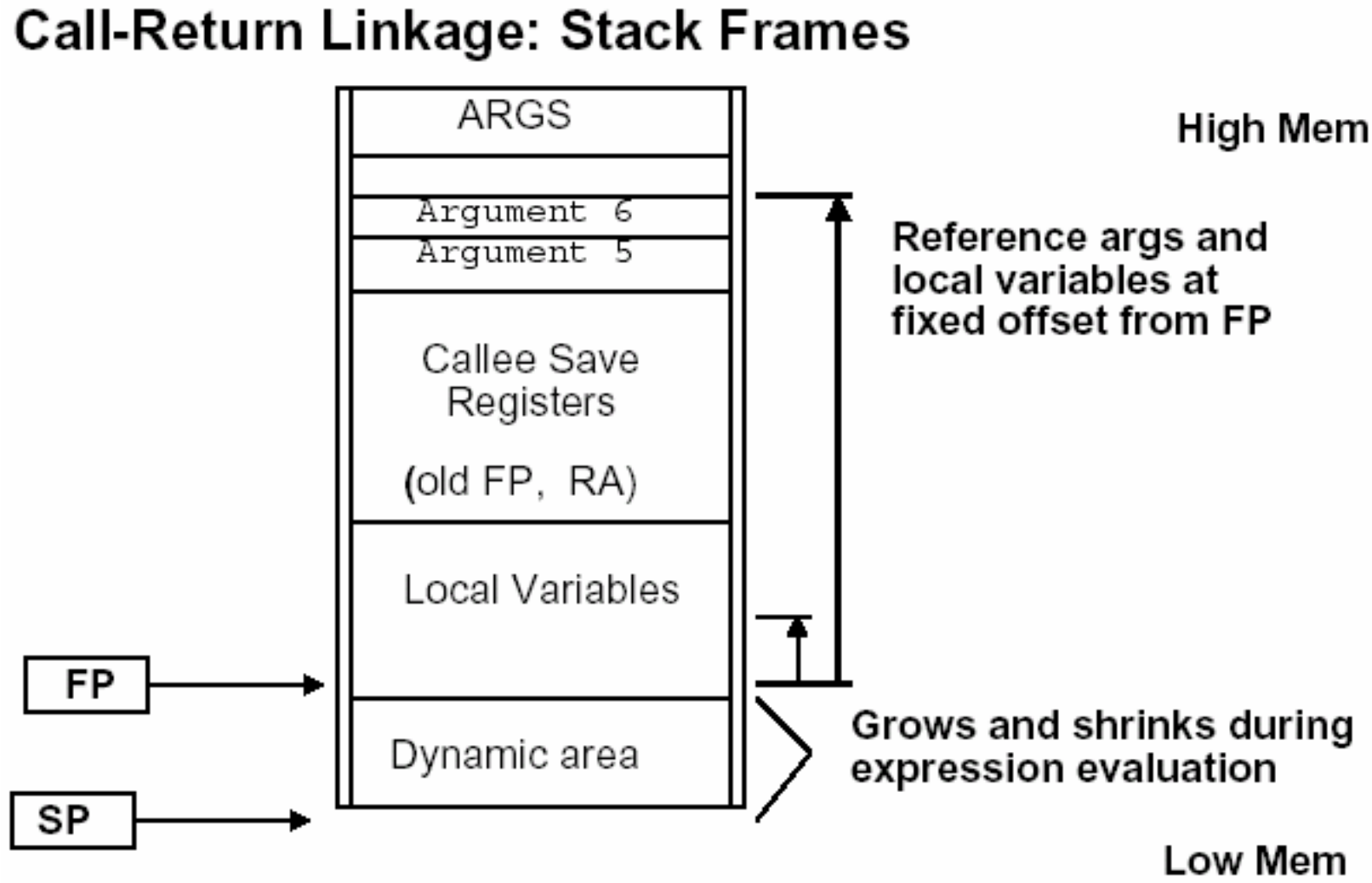
- **The stack is set up for you – just use $sp.**
- **You can view the stack in the data window.**
- **main is called as a subroutine (have it return using jr $ra).**
- **For now, don't worry about details. But the next few pages do some excellent example of how stacks all work.**

Why Are Stacks So Great?

- Some machines provide a memory stack as part of the architecture (e.g., VAX)
- Sometimes stacks are implemented via software convention (e.g., MIPS)

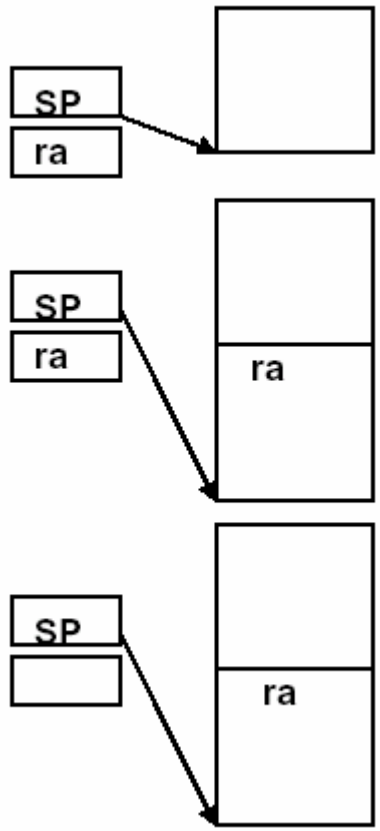


Why Are Stacks So Great?



MIPS Function Calling Conventions

```
SP fact:
    addiu $sp, $sp, -32
    sw $ra, 20($sp)
    . . .
    sw $s0, 4($sp)
    ...
    lw $ra, 20($sp)
    addiu $sp, $sp, 32
     jr $ra
```



MIPS Function Calling Conventions

```
main()   {
    printf("The factorial of 10 is %d\n", fact(10));
}
int fact (int n)  {
    if (n <= 1) return(1);
    return (n * fact (n-1));
}
```

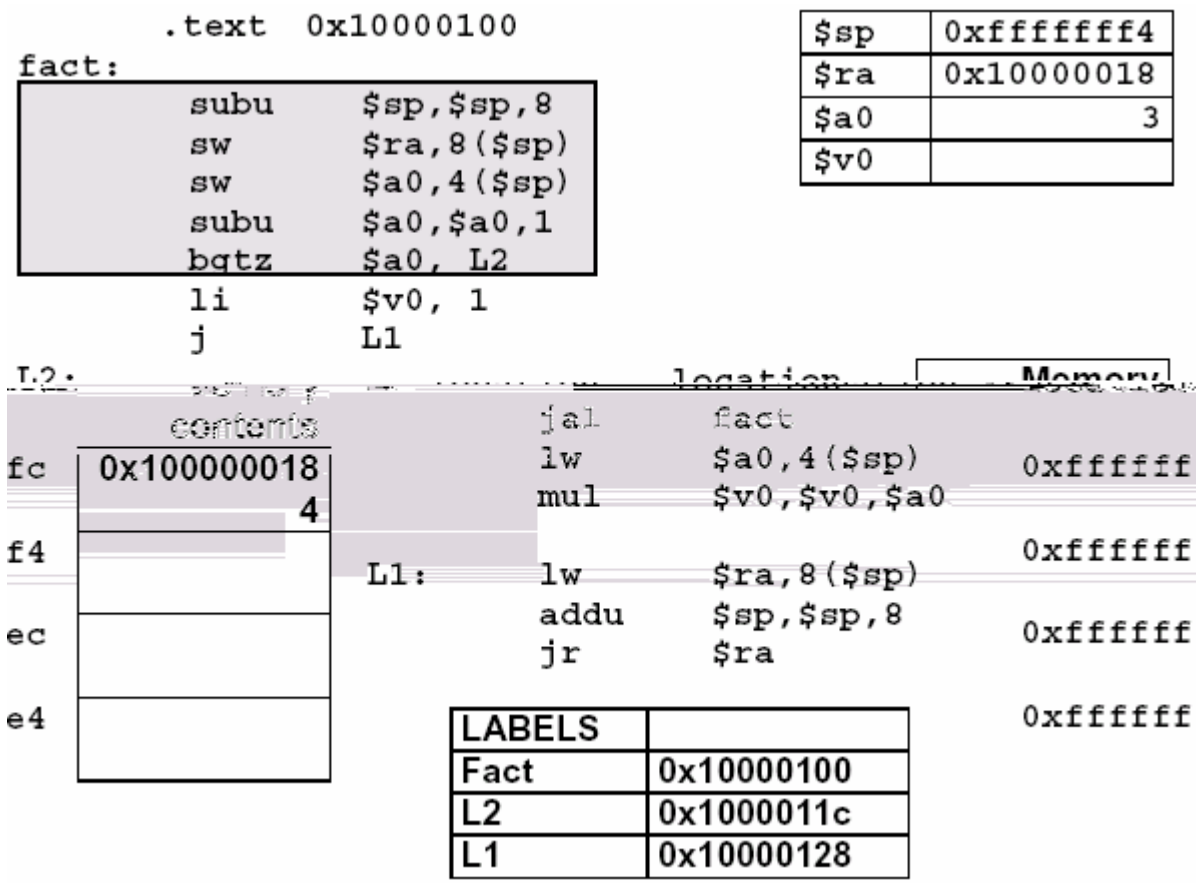
MIPS Function Calling Conventions

```
    .text
    .global main
main:
    subu    $sp, $sp, 32        #stack frame size is 32 bytes
    sw      $ra,20($sp)         #save return address
    li      $a0,10              # load argument (10) in $a0
    jal     fact                #call fact
    la      $a0 LC              #load string address in $a0
    move    $a1,$v0             #load fact result in $a1
    jal     printf              # call printf
    lw      $ra,20($sp)         # restore $sp
    addu    $sp, $sp,32         # pop the stack
    jr      $ra                 # exit()
    .data
LC:
    .asciiz "The factorial of 10 is %d\n"
```

MIPS Function Calling Conventions

```
    .text
fact:   subu    $sp,$sp,8          # stack frame is 8 bytes
        sw      $ra,8($sp)         #save return address
        sw      $a0,4($sp)         # save argument(n)
        subu    $a0,$a0,1          # compute n-1
        bgtz    $a0, L2  # if n-1>0 (ie n>1) go to L2
        li      $v0, 1             #
        j       L1                 # return(1)
        L2:                # new argument (n-1) is already in $a0
        jal     fact               # call fact
        lw      $a0,4($sp)         # load n
        mul     $v0,$v0,$a0        # fact(n-1)*n
L1:     lw      $ra,8($sp)          # restore $ra
        addu    $sp,$sp,8          # pop the stack
        jr      $ra                # return, result in $v0
```

MIPS Function Calling Conventions



MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0

L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xfffffff4
$ra	0x10000120
$a0	3
$v0	

location	Memory contents
0xfffffffc	0x100000184
0xfffffff4	
0xffffffec	
0xffffffe4	

MIPS Function Calling Conventions

```
        .text   0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0
L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

$sp	0xffffffec
$ra	0x10000120
$a0	2
$v0	

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

location	Memory contents
0xfffffffc	0x100000184
0xfffffff4	0x100001203
0xffffffec	
0xffffffe4	

MIPS Function Calling Conventions

```
        .text   0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0

L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xffffffec
$ra	0x10000120
$a0	2
$v0	

location	Memory contents
0xfffffffc	0x1000000184
0xfffffff4	0x1000001203
0xffffffec	
0xffffffe4	

MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0
L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

$sp	0xfffffffe4
$ra	0x10000120
$a0	1
$v0	

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

location	Memory contents
0xfffffffc	0x100000184
0xffffffff4	0x1000001203
0xffffffec	0x1000001202
0xfffffffe4	

MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0

L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

$sp	0xfffffffe4
$ra	0x10000120
$a0	1
$v0	

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

location	Memory contents
0xfffffffc	0x100000184
0xffffffff4	0x100001203
0xffffffec	0x100001202
0xfffffffe4	

MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0

L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xfffffffdc
$ra	0x10000120
$a0	0
$v0	

location	Memory contents
0xffffffffc	0x100000018 4
0xffffffff4	0x100000120 3
0xfffffffec	0x100000120 2
0xfffffffe4	0x100000120 1

MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0

L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xffffffdc
$ra	0x10000120
$a0	0
$v0	1

location	Memory contents
0xfffffffc	0x100000184
0xfffffff4	0x100000120 3
0xffffffec	0x100000120 2
0xffffffe4	0x100000120 1

MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0
L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xfffffffe4
$ra	0x10000120
$a0	0
$v0	1

location	Memory contents
0xfffffffc	0x100000184
0xffffffff4	0x100000120 3
0xffffffec	0x100000120 2
0xffffffe4	0x100000120 1

MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0
L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xfffffffe4
$ra	0x10000120
$a0	2
$v0	2

location	Memory contents
0xffffffffc	0x100000018 4
0xffffffff4	0x100000120 3
0xffffffec	0x100000120 2
0xffffffe4	0x100000120 1

MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0
L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xffffffec
$ra	0x10000120
$a0	2
$v0	2

location	Memory contents
0xfffffffc	0x100000018 4
0xffffffff4	0x100000120 3
0xffffffec	0x100000120 2
0xffffffe4	0x100000120 1

MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0
L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xffffffec
$ra	0x10000120
$a0	3
$v0	6

location	Memory contents
0xffffffc	0x100000184
0xffffff4	0x100000120 3
0xffffffec	0x100000120 2
0xffffffe4	0x100000120 1

MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0
L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xfffffff4
$ra	0x10000120
$a0	3
$v0	6

location	Memory contents
0xfffffffc	0x100000184
0xfffffff4	0x100001203
0xffffffec	0x100001202
0xffffffe4	0x100001201

MIPS Function Calling Conventions

```
        .text   0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0
L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xfffffff4
$ra	0x10000120
$a0	4
$v0	24

location	Memory contents
0xfffffffc	0x100000184
0xfffffff4	0x100001203
0xffffffec	0x100001202
0xffffffe4	0x100001201

MIPS Function Calling Conventions

```
        .text  0x10000100
fact:
        subu    $sp,$sp,8
        sw      $ra,8($sp)
        sw      $a0,4($sp)
        subu    $a0,$a0,1
        bgtz    $a0, L2
        li      $v0, 1
        j       L1
L2:
        jal     fact
        lw      $a0,4($sp)
        mul     $v0,$v0,$a0
L1:     lw      $ra,8($sp)
        addu    $sp,$sp,8
        jr      $ra
```

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

$sp	0xfffffffc
$ra	0x10000018
$a0	4
$v0	24

location	Memory contents
0xfffffffc	0x100000184
0xfffffff4	0x100001203
0xffffffec	0x100001202
0xffffffe4	0x100001201

Sample SPIM Programs (on the web)

multiply.s: multiplication subroutine based on repeated addition and a test program that calls it.

http://babbage.clarku.edu/~jbreecher/comp_org/labs/multiply.s

fact.s: computes factorials using the multiply subroutine.

http://babbage.clarku.edu/~jbreecher/comp_org/labs/fact.s

sort.s: the sorting program from the text.

http://babbage.clarku.edu/~jbreecher/comp_org/labs/sort.s

strcpy.s: the strcpy subroutine and test code.

http://babbage.clarku.edu/~jbreecher/comp_org/labs/strcpy.s