

CPS 104
Computer Organization
Lecture 7 :MIPS ISA and Assembler

Robert Wagner

Overview of Today's Lecture:

- **Review: MIPS Assembly Language Programming Conventions.**
- **System calls**

- **Stacks, Functions and Procedures.**

MIPS: Software conventions for Registers

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

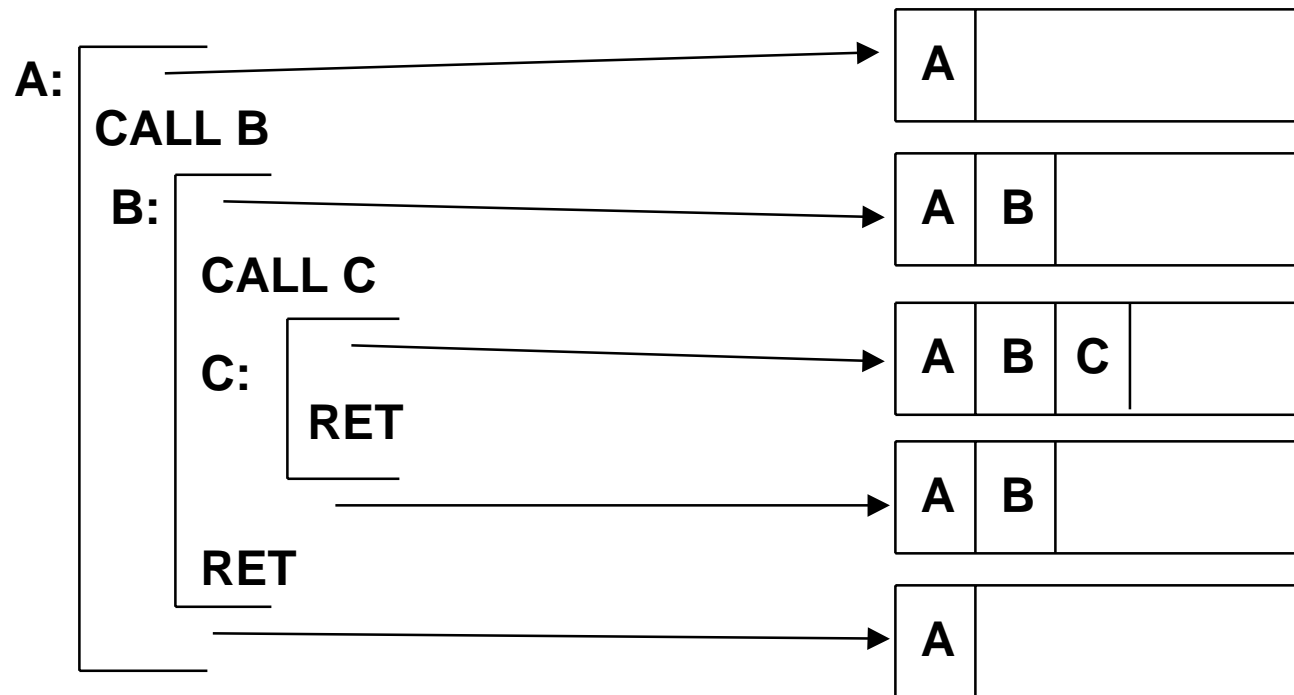
System call

- 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		(address)
2	print float	\$f12		
3	print double	\$f12		
4	print string	\$a0		
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			

Calls: Why Are Stacks So Great?

Stacking of Subroutine Calls & Returns and Environments:



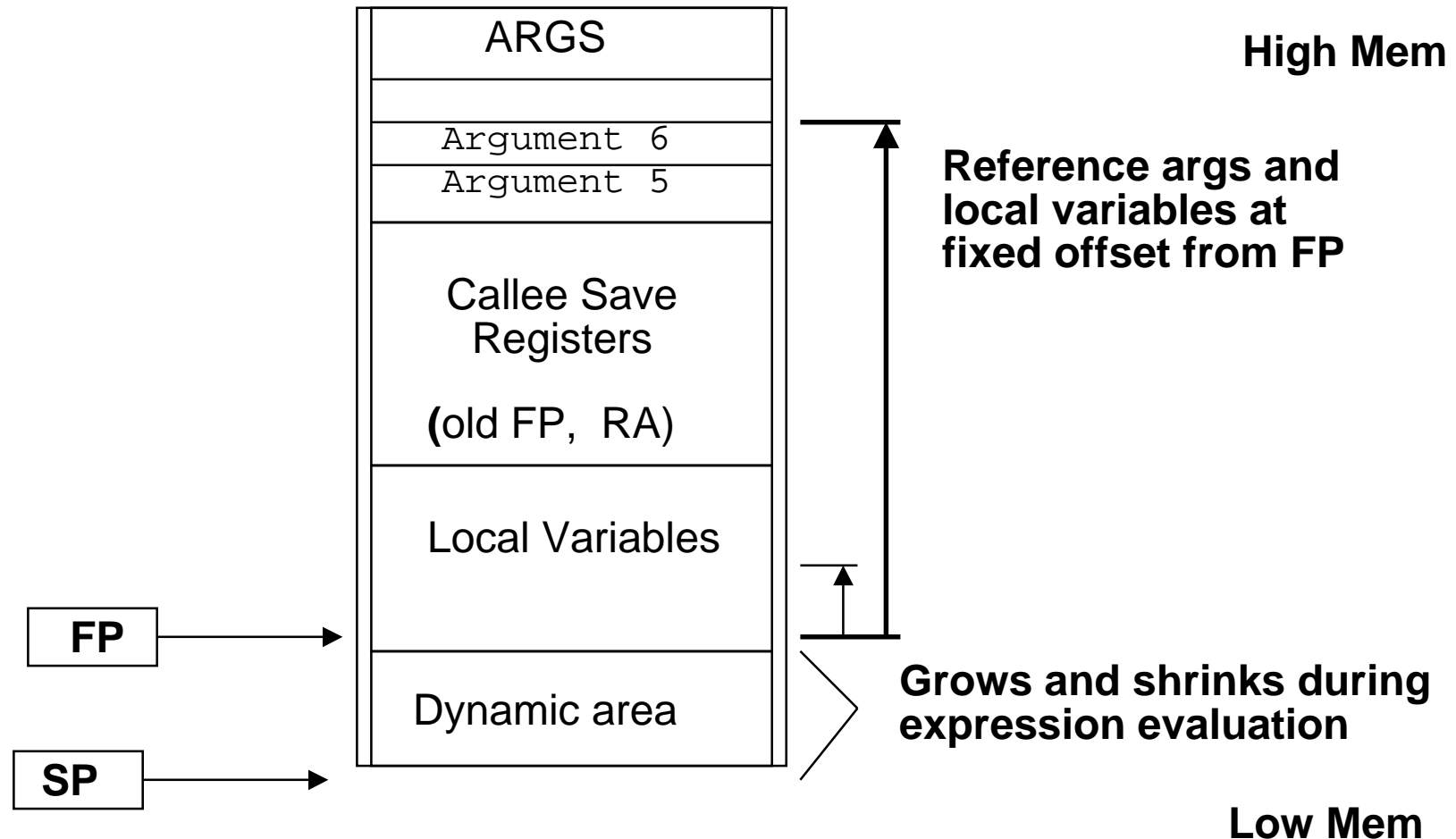
**Some machines provide a memory stack as part of the architecture
(e.g., VAX)**

**Sometimes stacks are implemented via software convention
(e.g., MIPS)**

Procedure Call (Stack) Frame

- **Procedures use a frame in the stack to:**
 - **Hold values passed to procedures as arguments.**
 - **Save registers that a procedure may modify, but which the procedure's caller does not want changed.**
 - **To provide space for local variables.
(variables with local scope)**
 - **To evaluate complex expressions.**

Call-Return Linkage: Stack Frames



- Many variations on stacks possible (up/down, last pushed / next)
- Block structured languages contain link to lexically enclosing frame
- Compilers normally keep scalar variables in registers, not memory!
- FP and SP coincide, if there is no dynamic area, as in SPIM

MIPS/GCC Procedure Calling Conventions

Calling Procedure:

- **Step-1: Pass the arguments:**
 - The first four arguments are passed in registers $\$a0-\$a3$
 - Remaining arguments are pushed into the stack
 - ➔ (in reversed order `arg5` is at the top of the stack).

- **Step-2: Save caller-saved registers**
 - Save registers $\$t0-\$t9$ if they contain live values at the call site.

- **Step-3: Execute a `jal` instruction.**

MIPS/GCC Procedure Calling Conventions (cont.)

Called Routine

- **Step-1: Establish stack frame.**
 - **Subtract the frame size from the stack pointer.**
`subiu $sp, $sp, <frame-size>`
- **Step-2: Save callee saved registers in the frame.**
 - **Register `$ra` is saved if routine makes a call.**
 - **Registers `$s0`–`$s7` are saved if they are used.**

MIPS/GCC Procedure Calling Conventions (cont.)

On return from a call

- **Step-1: Put returned values in registers $\$v0$, [$\$v1$].**
(if values are returned)
- **Step-2: Restore callee-saved registers.**
 - **Restore any saved registers [$\$ra$, $\$s0$ - $\$s7$]**

Step-3: Pop the stack

- **Add the frame size to $\$sp$.**

```
addiu $sp, $sp, <frame-size>
```

Step-4: Return

- **Jump to the address in $\$ra$.**

```
jr      $ra
```

MIPS Function Calling Conventions

fact:

```
addiu $sp, $sp, -32
```

```
sw    $ra, 20($sp)
```

```
...
```

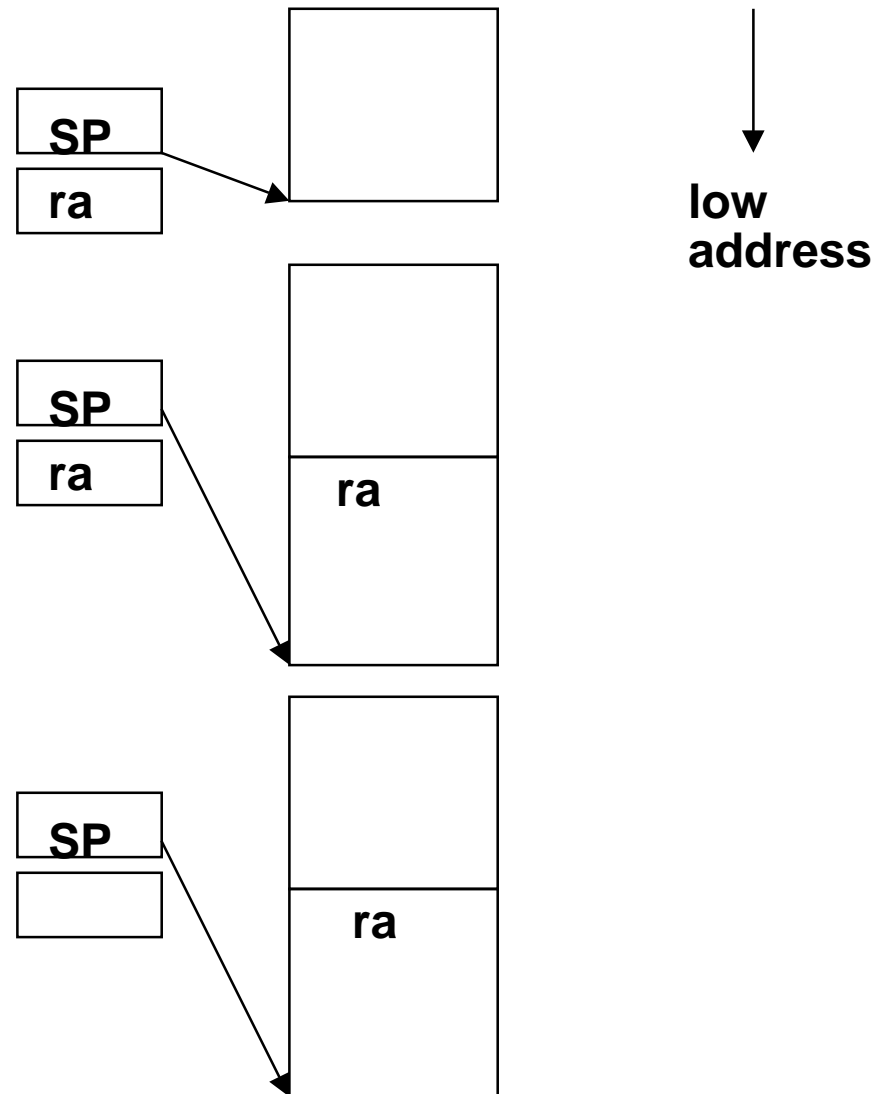
```
sw    $s0, 4($sp)
```

```
...
```

```
lw    $ra, 20($sp)
```

```
addiu $sp, $sp, 32
```

```
jr    $ra
```



First four arguments passed in registers.

Example: Factorial

```
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));  
}
```

```

.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"

```

```

        .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

```

```
.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	0xffffffff4
\$ra	0x10000018
\$a0	3
\$v0	

```
location
```

```
0xfffffffffc
```

```
0xffffffff4
```

```
0xffffffffec
```

```
0xffffffffe4
```

Memory contents
0x100000018 4

```

        .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	0xffffffff4
\$ra	0x10000120
\$a0	3
\$v0	

```

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

location	Memory contents
0xffffffffc	0x10000018 4
0xffffffff4	
0xffffffffec	
0xffffffffe4	

```
.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	0xffffffffec
\$ra	0x10000120
\$a0	2
\$v0	

```
location
```

```
0xfffffffffc
```

```
0xfffffffff4
```

```
0xffffffffec
```

```
0xffffffffe4
```

Memory contents
0x100000018
4
0x100000120
3

```

        .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

```

\$sp	0xffffffffec
\$ra	0x10000120
\$a0	2
\$v0	

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

location
0xfffffffffc
0xfffffffff4
0xffffffffec
0xffffffffe4

Memory contents
0x100000018 4
0x100000120 3

```

        .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	0xffffffffe4
\$ra	0x10000120
\$a0	1
\$v0	

location

0xfffffffffc

0xfffffffff4

0xffffffffec

0xffffffffe4

Memory
contents

0x100000018
4

0x100000120
3

0x100000120
2

```

        .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

```

\$sp	0xffffffffe4
\$ra	0x10000120
\$a0	1
\$v0	

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

location

0xfffffffffc

0xfffffffff4

0xffffffffec

0xffffffffe4

	Memory contents
	0x100000018 4
	0x100000120 3
	0x100000120 2

```

        .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

```

subu	\$sp,\$sp,8
sw	\$ra,8(\$sp)
sw	\$a0,4(\$sp)
subu	\$a0,\$a0,1
bgtz	\$a0,L2

\$sp	0xffffffffdc
\$ra	0x10000120
\$a0	0
\$v0	

LABELS	
Fact	0x10000100
L2	0x1000011c
L1	0x10000128

location	Memory contents
0xfffffffffc	0x10000018 4
0xfffffffff4	0x100000120 3
0xfffffffec	0x100000120 2
0xffffffe4	0x100000120 1

```

        .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

```

\$sp	0xffffffffdc
\$ra	0x10000120
\$a0	0
\$v0	1

```

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

location	Memory contents
0xfffffffffc	0x10000018 4
0xfffffffff4	0x100000120 3
0xfffffffec	0x100000120 2
0xffffffe4	0x100000120 1

```

    .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	0xffffffffe4
\$ra	0x10000120
\$a0	0
\$v0	1

location	Memory contents
0xfffffffffc	0x10000018 4
0xfffffffff4	0x100000120 3
0xffffffffec	0x100000120 2
0xffffffffe4	0x100000120 1

```

        .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	0xffffffffe4
\$ra	0x10000120
\$a0	2
\$v0	2

location	Memory contents
0xfffffffffc	0x10000018 4
0xfffffffff4	0x100000120 3
0xffffffffec	0x100000120 2
0xffffffffe4	0x100000120 1

```

        .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	0xffffffffec
\$ra	0x10000120
\$a0	2
\$v0	2

location	Memory contents
0xfffffffffc	0x10000018 4
0xfffffffff4	0x100000120 3
0xffffffffec	0x100000120 2
0xffffffffe4	0x100000120 1

```

        .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	0xffffffffec
\$ra	0x10000120
\$a0	3
\$v0	6

location	Memory contents
0xfffffffffc	0x10000018 4
0xfffffffff4	0x100000120 3
0xffffffffec	0x100000120 2
0xffffffffe4	0x100000120 1

```

    .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	0xffffffff4
\$ra	0x10000120
\$a0	3
\$v0	6

location	Memory contents
0xfffffffffc	0x10000018 4
0xffffffff4	0x100000120 3
0xfffffffec	0x100000120 2
0xffffffe4	0x100000120 1

```

        .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	0xffffffff4
\$ra	0x10000120
\$a0	4
\$v0	24

location	Memory contents
0xfffffffffc	0x10000018 4
0xffffffff4	0x100000120 3
0xfffffffec	0x100000120 2
0xffffffe4	0x100000120 1

```

    .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	0xffffffffc
\$ra	0x10000018
\$a0	4
\$v0	24

location	Memory contents
0xffffffffc	0x100000018 4
0xfffffffff4	0x100000120 3
0xfffffffec	0x100000120 2
0xffffffe4	0x100000120 1

Example: Factorial

