Procedure Calls

COMP 411
Feb 24, 2017

MIPS Registers

Registers have designated functions by convention

Name	Register number	Usage
$zero	0	the constant value 0
$at	1	assembler temporary
$v0-$v1	2-3	procedure return values
$a0-$a3	4-7	procedure arguments
$t0-$t7	8-15	temporaries
$s0-$s7	16-23	saved by callee
$t8-$t9	24-25	more temporaries
$k0-$k1	26-27	reserved for operating system
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

Simple Procedure Call

Call a procedure without arguments or return values

C code

```c
#include <stdio.h>

void hello() {
    printf("Hello\n");
}

int main() {
    hello();
}
```

Hello

MIPS code

```
.data 0x0
string:   .asciiz "Hello\n"

.text 0x3000
main:
     jal hello
     ori $v0, $0, 10  # system call code for exit
     syscall          # exit the program

hello:
    li $v0, 4       # system call code for string
    la $a0, string  # address of string is in $a0
    syscall         # print the string
    jr $ra
```

Simple Procedure Call

We only used $ra to store the return address

Name	Register number	Usage
$zero	0	the constant value 0
$at	1	assembler temporary
$v0-$v1	2-3	procedure return values
$a0-$a3	4-7	procedure arguments
$t0-$t7	8-15	temporaries
$s0-$s7	16-23	saved by callee
$t8-$t9	24-25	more temporaries
$k0-$k1	26-27	reserved for operating system
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

Nested Simple Procedure Call

How about two nested procedures?

C code

```c
#include <stdio.h>

void world() {
    printf(" world\n");
}

void hello() {
    printf("Hello");
    world();
}

int main() {
    hello();
}
```

```
Hello world
```

MIPS code

```
1   .data 0x0
2   string1:    .asciiz "Hello"
3   string2:    .asciiz " world\n"
4
5   .text 0x3000
6   main:
7       jal hello
8       ori $v0, $0, 10  # system call code for exit
9       syscall          # exit the program
10
11   hello:
12      li $v0, 4        # system call code for string
13      la $a0, string1  # address of string is in $a0
14      syscall          # print the string
15      jal world
16      jr $ra
17
18   world:
19      li $v0, 4
20      la $a0, string2
21      syscall
22      jr $ra
```

Let's see what happens in MARS

About to call the first procedure

```
1  .data 0x0
2  string1:   .asciiz "Hello"
3  string2:   .asciiz " world\n"
4
5  .text 0x3000
6  main:
7       jal hello
8       ori $v0, $0, 10  # system call code for exit
9       syscall          # exit the program
10
11 hello:
12      li $v0, 4        # system call code for string
13      la $a0, string1  # address of string is in $a0
14      syscall          # print the string
15      jal world
16      jr $ra
17
18 world:
19      li $v0, 4
20      la $a0, string2
21      syscall
22      jr $ra
```

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00003000	0x0c000c03	jal 0x0000300c	7: jal hello
	0x00003004	0x3402000a	ori $2,$0,0x0000000a	8: ori $v0, $0, 10 # system call code for exit
	0x00003008	0x0000000c	syscall	9: syscall # exit the program
	0x0000300c	0x24020004	addiu $2,$0,0x00000004	12: li $v0, 4 # system call code for string
	0x00003010	0x20040000	addi $4,$0,0x00000000	13: la $a0, string1 # address of string is in $a0
	0x00003014	0x0000000c	syscall	14: syscall # print the string
	0x00003018	0x0c000c08	jal 0x00003020	15: jal world
	0x0000301c	0x03e00008	jr $31	16: jr $ra
	0x00003020	0x24020004	addiu $2,$0,0x00000004	19: li $v0, 4
	0x00003024	0x20040006	addi $4,$0,0x00000006	20: la $a0, string2
	0x00003028	0x0000000c	syscall	21: syscall
	0x0000302c	0x03e00008	jr $31	22: jr $ra

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00000000	l l e H	w \0 o	d l r o	\0 \0 \0 \n	\0 \0 \0 \0	\0 \0 \0 \
0x00000020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00002ffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00003000
hi		0x00000000
lo		0x00000000

Let's see what happens in MARS

```
1  .data 0x0
2  string1:   .asciiz "Hello"
3  string2:   .asciiz " world\n"
4
5  .text 0x3000
6  main:
7       jal hello
8       ori $v0, $0, 10  # system call code for exit
9       syscall          # exit the program
10
11 hello:
12      li $v0, 4        # system call code for string
13      la $a0, string1  # address of string is in $a0
14      syscall          # print the string
15      jal world
16      jr $ra
17
18 world:
19      li $v0, 4
20      la $a0, string2
21      syscall
22      jr $ra
```

The jump and link instruction saved the return address 0x3004 to $ra

Text Segment

Bkpt	Address	Code	Basic	Source
	0x0000300	0x0c000c03	jal 0x0000300c	7: jal hello
	0x00003004	0x3402000a	ori $2,$0,0x0000000a	8: ori $v0, $0, 10 # system call code for exit
	0x00003008	0x0000000c	syscall	9: syscall # exit the program
	0x0000300c	0x24020004	addiu $2,$0,0x00000004	12: li $v0, 4 # system call code for string
	0x00003010	0x20040000	addi $4,$0,0x00000000	13: la $a0, string1 # address of string is in $a0
	0x00003014	0x0000000c	syscall	14: syscall # print the string
	0x00003018	0x0c000c08	jal 0x00003020	15: jal world
	0x0000301c	0x03e00008	jr $31	16: jr $ra
	0x00003020	0x24020004	addiu $2,$0,0x00000004	19: li $v0, 4
	0x00003024	0x20040006	addi $4,$0,0x00000006	20: la $a0, string2
	0x00003028	0x0000000c	syscall	21: syscall
	0x0000302c	0x03e00008	jr $31	22: jr $ra

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00000000	l l e H	w \0 o	d l r o	\0 \0 \0 \n	\0 \0 \0 \0	\0 \0 \0 \
0x00000020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00002ffc
$fp	30	0x00000000
$ra	31	0x00003004
pc		0x0000300c
hi		0x00000000
lo		0x00000000

Let's see what happens in MARS

```
1  .data 0x0
2  string1:   .asciiz "Hello"
3  string2:   .asciiz " world\n"
4
5  .text 0x3000
6  main:
7       jal hello
8       ori $v0, $0, 10  # system call code for exit
9       syscall          # exit the program
10
11  hello:
12      li $v0, 4        # system call code for string
13      la $a0, string1  # address of string is in $a0
14      syscall          # print the string
15      jal world
16      jr $ra
17
18  world:
19      li $v0, 4
20      la $a0, string2
21      syscall
22      jr $ra
```

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00003000	0x0c000c03	jal 0x0000300c	7: jal hello
	0x00003004	0x3402000a	ori $2,$0,0x0000000a	8: ori $v0, $0, 10 # system call code for exit
	0x00003008	0x0000000c	syscall	9: syscall # exit the program
	0x0000300c	0x24020004	addiu $2,$0,0x00000004	12: li $v0, 4 # system call code for string
	0x00003010	0x20040000	addi $4,$0,0x00000000	13: la $a0, string1 # address of string is in $a0
	0x00003014	0x0000000c	syscall	14: syscall # print the string
	0x00003018	0x0c000c08	jal 0x00003020	15: jal world
	0x0000301c	0x03e00008	jr $31	16: jr $ra
	0x00003020	0x24020004	addiu $2,$0,0x00000004	19: li $v0, 4
	0x00003024	0x20040006	addi $4,$0,0x00000006	20: la $a0, string2
	0x00003028	0x0000000c	syscall	21: syscall
	0x0000302c	0x03e00008	jr $31	22: jr $ra

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00000000	l l e H	w \0 o	d l r o	\0 \0 \0 \n	\0 \0 \0 \0	\0 \0 \0 \
0x00000020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000004
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00002ffc
$fp	30	0x00000000
$ra	31	0x00003004
pc		0x00003018
hi		0x00000000
lo		0x00000000

Let's see what happens in MARS

The jump and link instruction saved the return address 0x301c to $ra

```
1  .data 0x0
2  string1:   .asciiz "Hello"
3  string2:   .asciiz " world\n"
4
5  .text 0x3000
6  main:
7       jal hello
8       ori $v0, $0, 10  # system call code for exit
9       syscall          # exit the program
10
11 hello:
12      li $v0, 4        # system call code for string
13      la $a0, string1  # address of string is in $a0
14      syscall          # print the string
15      jal world
16      jr $ra
17
18 world:
19      li $v0, 4
20      la $a0, string2
21      syscall
22      jr $ra
```

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00003000	0x0c000c03	jal 0x0000300c	7: jal hello
	0x00003004	0x3402000a	ori $2,$0,0x0000000a	8: ori $v0, $0, 10 # system call code for exit
	0x00003008	0x0000000c	syscall	9: syscall # exit the program
	0x0000300c	0x24020004	addiu $2,$0,0x00000004	12: li $v0, 4 # system call code for string
	0x00003010	0x20040000	addi $4,$0,0x00000000	13: la $a0, string1 # address of string is in $a0
	0x00003014	0x0000000c	syscall	14: syscall # print the string
	0x00003018	0x0c000c08	jal 0x00003020	15: jal world
	0x0000301c	0x03e00008	jr $31	16: jr $ra
	0x00003020	0x24020004	addiu $2,$0,0x00000004	19: li $v0, 4
	0x00003024	0x20040006	addi $4,$0,0x00000006	20: la $a0, string2
	0x00003028	0x0000000c	syscall	21: syscall
	0x0000302c	0x03e00008	jr $31	22: jr $ra

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00000000	l l e H	w \0 o	d l r o	\0 \0 \0 \n	\0 \0 \0 \0	\0 \0 \0 \
0x00000020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000004
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00002ffc
$fp	30	0x00000000
$ra	31	0x0000301c
pc		0x00003020
hi		0x00000000
lo		0x00000000

Let's see what happens in MARS

```
1  .data 0x0
2  string1:    .asciiz "Hello"
3  string2:    .asciiz " world\n"
4
5  .text 0x3000
6  main:
7       jal hello
8       ori $v0, $0, 10  # system call code for exit
9       syscall          # exit the program
10
11  hello:
12      li $v0, 4        # system call code for string
13      la $a0, string1  # address of string is in $a0
14      syscall          # print the string
15      jal world
16      jr $ra
17
18  world:
19      li $v0, 4
20      la $a0, string2
21      syscall
22      jr $ra
```

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00003000	0x0c000c03	jal 0x0000300c	7: jal hello
	0x00003004	0x3402000a	ori $2,$0,0x0000000a	8: ori $v0, $0, 10 # system call code for exit
	0x00003008	0x0000000c	syscall	9: syscall # exit the program
	0x0000300c	0x24020004	addiu $2,$0,0x00000004	12: li $v0, 4 # system call code for string
	0x00003010	0x20040000	addi $4,$0,0x00000000	13: la $a0, string1 # address of string is in $a0
	0x00003014	0x0000000c	syscall	14: syscall # print the string
	0x00003018	0x0c000c08	jal 0x00003020	15: jal world
	0x0000301c	0x03e00008	jr $31	16: jr $ra
	0x00003020	0x24020004	addiu $2,$0,0x00000004	19: li $v0, 4
	0x00003024	0x20040006	addi $4,$0,0x00000006	20: la $a0, string2
	0x00003028	0x0000000c	syscall	21: syscall
	0x0000302c	0x03e00008	jr $31	22: jr $ra

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00000000	l l e H	w \0 o	d l r o	\0 \0 \0 \n	\0 \0 \0 \0	\0 \0 \0 \
0x00000020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000004
$v1	3	0x00000000
$a0	4	0x00000006
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00002ffc
$fp	30	0x00000000
$ra	31	0x0000301c
pc		0x0000302c
hi		0x00000000
lo		0x00000000

The second procedure overwrites $ra, which is the same register the first procedure uses

The return address saved in $ra is 0x301c (the address of the current jump instruction), therefore we enter an infinite loop

```
1  .data 0x0
2  string1:    .asciiz "Hello"
3  string2:    .asciiz " world\n"
4
5  .text 0x3000
6  main:
7       jal hello
8       ori $v0, $0, 10  # system call code for exit
9       syscall          # exit the program
10
11 hello:
12      li $v0, 4        # system call code for string
13      la $a0, string1  # address of string is in $a0
14      syscall          # print the string
15      jal world
16      jr $ra
17
18 world:
19      li $v0, 4
20      la $a0, string2
21      syscall
22      jr $ra
```

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00003000	0x0c000c03	jal 0x0000300c	7: jal hello
	0x00003004	0x3402000a	ori $2,$0,0x0000000a	8: ori $v0, $0, 10 # system call code for exit
	0x00003008	0x0000000c	syscall	9: syscall # exit the program
	0x0000300c	0x24020004	addiu $2,$0,0x00000004	12: li $v0, 4 # system call code for string
	0x00003010	0x20040000	addi $4,$0,0x00000000	13: la $a0, string1 # address of string is in $a0
	0x00003014	0x0000000c	syscall	14: syscall # print the string
	[illegible]	0x0c000c08	jal 0x00003020	15: jal world
	0x0000301c	0x03e00008	jr $31	16: jr $ra
	[illegible]	0x24020004	addiu $2,$0,0x00000004	19: li $v0, 4
	0x00003024	0x20040006	addi $4,$0,0x00000006	20: la $a0, string2
	0x00003028	0x0000000c	syscall	21: syscall
	0x0000302c	0x03e00008	jr $31	22: jr $ra

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00000000	l l e H	w \0 o	d l r o	\0 \0 \0 \n	\0 \0 \0 \0	\0 \0 \0 \
0x00000020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x000000e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
0x00000160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \
[illegible]	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000004
$v1	3	0x00000000
$a0	4	0x00000006
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00002ffc
$fp	30	[illegible]
$ra	31	0x0000301c
pc		[illegible]
hi		0x00000000
lo		0x00000000

Nested Simple Procedure Call

Call two nested procedures without arguments or return values.

C code

```c
#include <stdio.h>

void world() {
    printf(" world\n");
}

void hello() {
    printf("Hello");
    world();
}

int main() {
    hello();
}
```

```
Hello world
```

MIPS code

```
1   .data 0x0
2   string1:    .asciiz "Hello"
3   string2:    .asciiz " world\n"
4
5   .text 0x3000
6   main:
7       jal hello
8       ori $v0, $0, 10  # system call code for exit
9       syscall          # exit the program
10
11   hello:
12      li $v0, 4        # system call code for string
13      la $a0, string1  # address of string is in $a0
14      syscall          # print the string
15      jal world
16      jr $ra
17
18   world:
19      li $v0, 4
20      la $a0, string2
21      syscall
22      jr $ra
```

How Can We Avoid Overwriting Registers?

Use different registers for each call?

- We quickly run out of registers
- Each procedure call is different, hard to keep track
- Code is not reusable
- Doesn’t work for recursive calls!

Save and restore registers that should be preserved after the procedure call

- We can use the same convention across all procedure calls

How to Fix Our Code?

C code

```c
#include <stdio.h>

void world() {
    printf(" world\n");
}

void hello() {
    printf("Hello");
    world();
}

int main() {
    hello();
}
```

```
Hello world
```

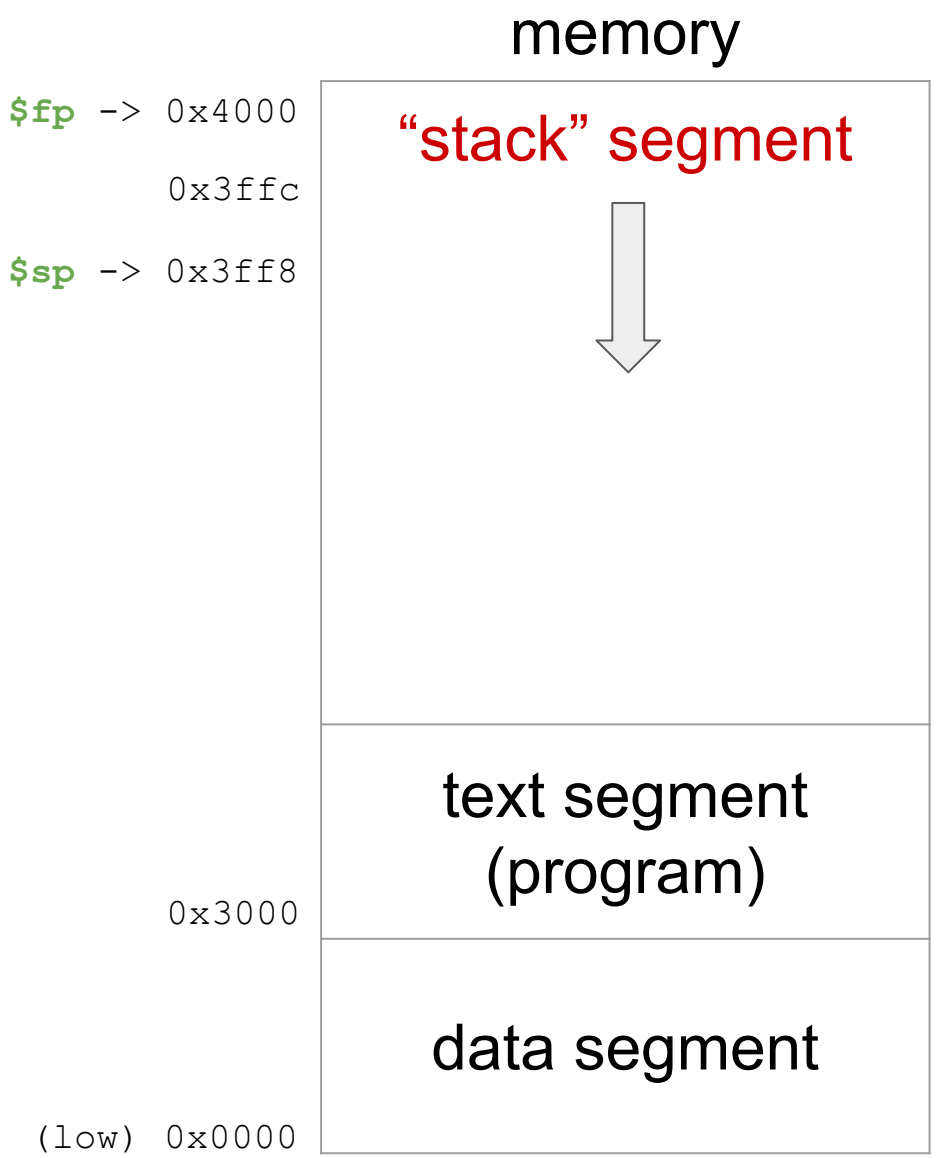
MIPS code

```
1   .data 0x0
2   string1:    .asciiz "Hello"
3   string2:    .asciiz " world\n"
4
5   .text 0x3000
6   main:
7       jal hello
8       ori $v0, $0, 10
9       syscall
10
11   hello:
12      save $ra to memory
13      li $v0, 4
14      la $a0, string1
15      syscall
16      jal world
17      restore $ra from memory
18      jr $ra
19
20   world:
21      li $v0, 4
22      la $a0, string2
23      syscall
24      jr $ra
```

Where to Save Registers?

The STACK!

- $sp points to the last USED stack memory location
- Use $fp to keep track of the top of the current stack frame
- The stack grows DOWN (towards lower addresses)



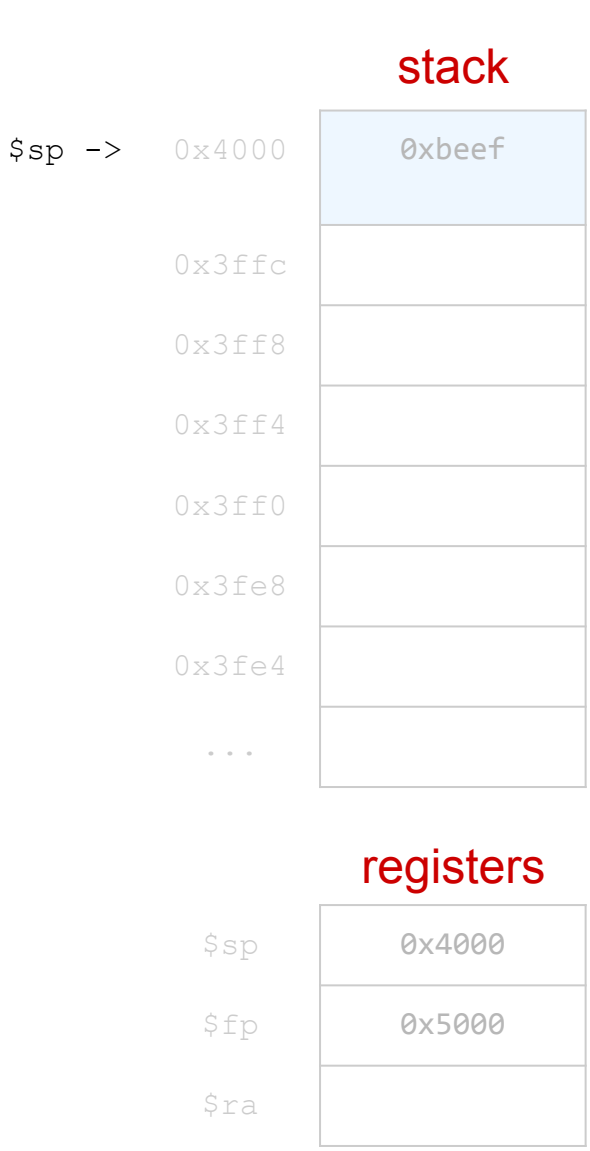
We Also Need to Save $fp

Let's assume we have a new version of world() that also uses the stack

MIPS code

```
1           .data 0x0
2           string1:    .asciiz "Hello"
3           string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7             jal hello
8             ori $v0, $0, 10
9             syscall
10
11          hello:
12            save $ra to memory
13            save $fp to memory
14            set $fp to top of current stack frame
15            li $v0, 4
16            la $a0, string1
17            syscall
18           jal world
19           restore $ra from memory
20           restore $fp from memory
21           jr $ra
22
23           world:
24             ...
```

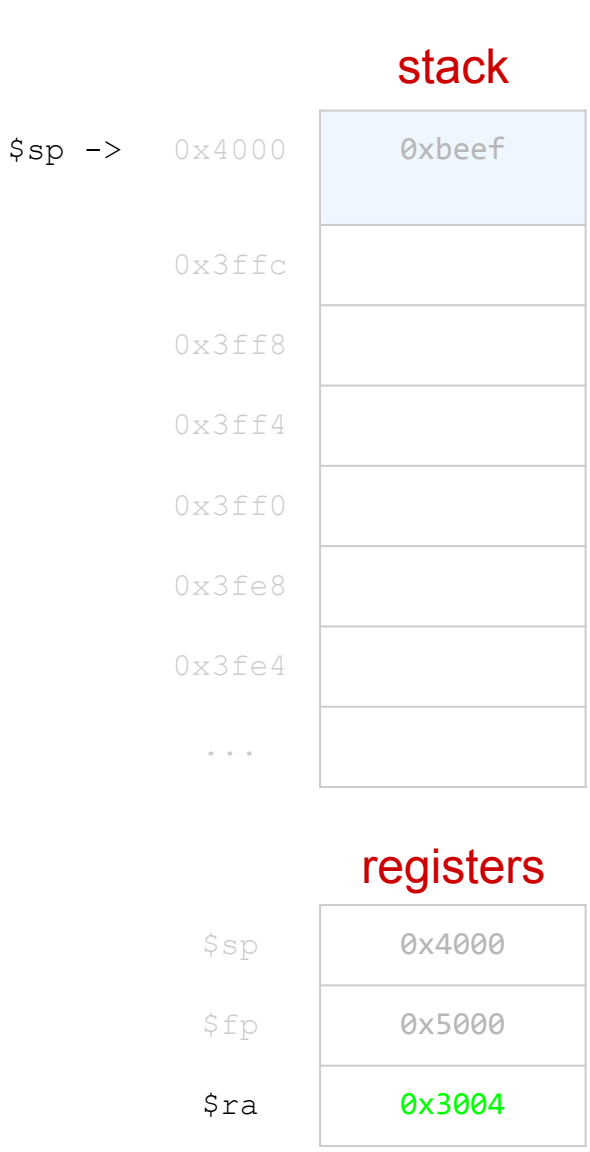
Does It Work?



MIPS code

```
1           .data 0x0
2  0x0000   string1:    .asciiz "Hello"
3  0x0007   string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7  0x3000     jal hello
8  0x3004     ori $v0, $0, 10
9  0x3008     syscall
10
11          hello:
12 0x300c     addi $sp, $sp, -8    # make room on stack
13 0x3010     sw $ra, 4($sp)       # save $ra
14 0x3014     sw $fp, 0($sp)       # save $fp
15 0x3018     addi $fp, $sp, 4     # set $fp
16 0x301c     li $v0, 4
17 0x3020     la $a0, string1
18 0x3024     syscall
19 0x3028     jal world
20 0x302c     lw  $ra, 0($fp)      # restore $ra
21 0x3030     addi $sp, $fp, 4     # restore $sp
22 0x3034     lw  $fp, -4($fp)     # restore $fp
23 0x3038     jr $ra
24
25          world:
...           ...
```

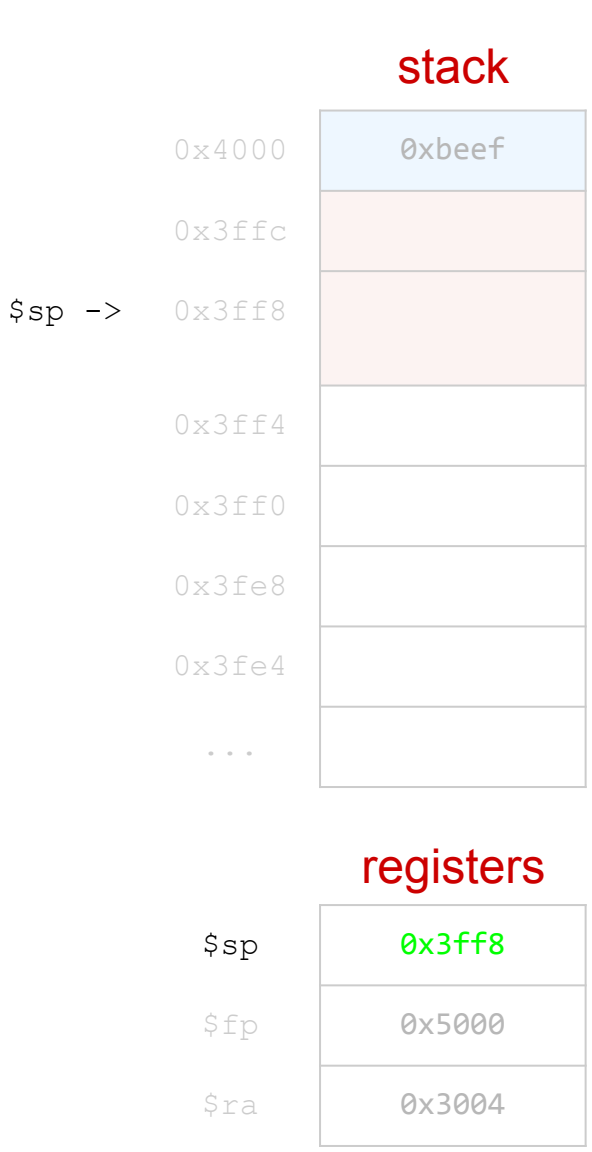
Does It Work?



MIPS code

```
1           .data 0x0
2   0x0000  string1:    .asciiz "Hello"
3   0x0007  string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7   0x3000    jal hello
8   0x3004    ori $v0, $0, 10
9   0x3008    syscall
10
11          hello:
12  0x300c    addi $sp, $sp, -8    # make room on stack
13  0x3010    sw $ra, 4($sp)       # save $ra
14  0x3014    sw $fp, 0($sp)       # save $fp
15  0x3018    addi $fp, $sp, 4     # set $fp
16  0x301c    li $v0, 4
17  0x3020    la $a0, string1
18  0x3024    syscall
19  0x3028    jal world
20  0x302c    lw  $ra, 0($fp)      # restore $ra
21  0x3030    addi $sp, $fp, 4     # restore $sp
22  0x3034    lw  $fp, -4($fp)     # restore $fp
23  0x3038    jr $ra
24
25          world:
...           ...
```

Does It Work?



MIPS code

```
1           .data 0x0
2   0x0000  string1:    .asciiz "Hello"
3   0x0007  string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7   0x3000    jal hello
8   0x3004    ori $v0, $0, 10
9   0x3008    syscall
10
11          hello:
12  0x300c    addi $sp, $sp, -8    # make room on stack
13  0x3010    sw $ra, 4($sp)       # save $ra
14  0x3014    sw $fp, 0($sp)       # save $fp
15  0x3018    addi $fp, $sp, 4     # set $fp
16  0x301c    li $v0, 4
17  0x3020    la $a0, string1
18  0x3024    syscall
19  0x3028    jal world
20  0x302c    lw  $ra, 0($fp)      # restore $ra
21  0x3030    addi $sp, $fp, 4     # restore $sp
22  0x3034    lw  $fp, -4($fp)     # restore $fp
23  0x3038    jr $ra
24
25          world:
...           ...
```

Does It Work?

MIPS code

```
1           .data 0x0
2   0x0000  string1:    .asciiz "Hello"
3   0x0007  string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7   0x3000    jal hello
8   0x3004    ori $v0, $0, 10
9   0x3008    syscall
10
11          hello:
12  0x300c    addi $sp, $sp, -8    # make room on stack
13  0x3010    sw $ra, 4($sp)       # save $ra
14  0x3014    sw $fp, 0($sp)       # save $fp
15  0x3018    addi $fp, $sp, 4     # set $fp
16  0x301c    li $v0, 4
17  0x3020    la $a0, string1
18  0x3024    syscall
19  0x3028    jal world
20  0x302c    lw  $ra, 0($fp)      # restore $ra
21  0x3030    addi $sp, $fp, 4     # restore $sp
22  0x3034    lw  $fp, -4($fp)     # restore $fp
23  0x3038    jr $ra
24
25          world:
...           ...
```

Does It Work?

stack

	0x4000	0xbeef
	0x3ffc	0x3004
$sp ->	0x3ff8	0x5000
	0x3ff4	
	0x3ff0	
	0x3fe8	
	0x3fe4	
	...	

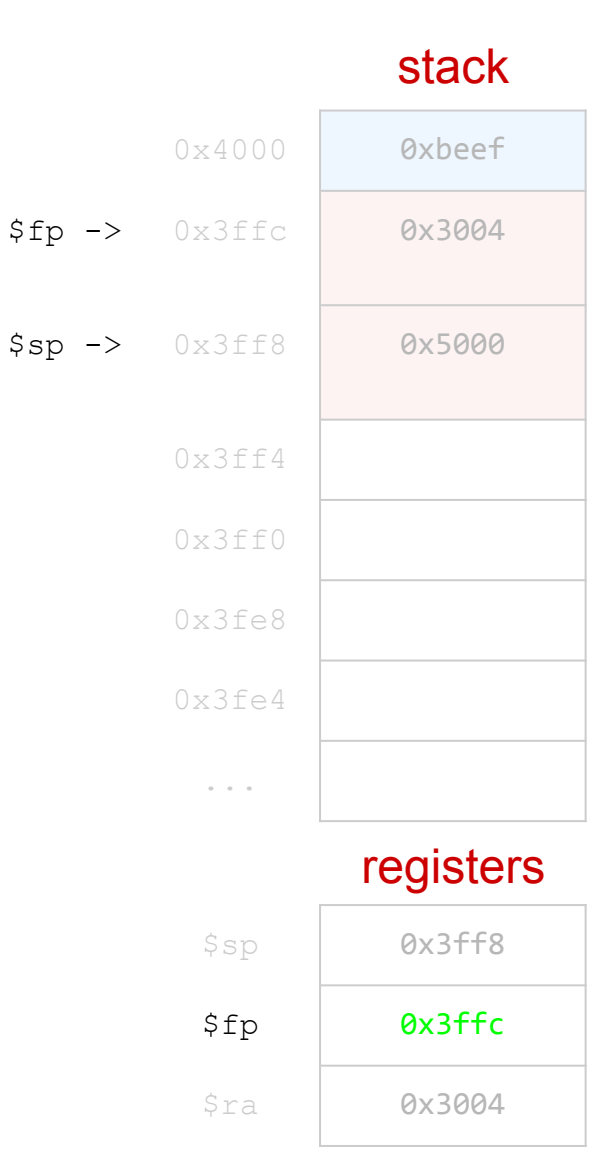
registers

$sp	0x3ff8
$fp	0x5000
$ra	0x3004

MIPS code

```
1           .data 0x0
2  0x0000   string1:   .asciiz "Hello"
3  0x0007   string2:   .asciiz " world\n"
4
5           .text 0x3000
6           main:
7  0x3000     jal hello
8  0x3004     ori $v0, $0, 10
9  0x3008     syscall
10
11          hello:
12 0x300c     addi $sp, $sp, -8    # make room on stack
13 0x3010     sw $ra, 4($sp)       # save $ra
14 0x3014     sw $fp, 0($sp)       # save $fp
15 0x3018     addi $fp, $sp, 4     # set $fp
16 0x301c     li $v0, 4
17 0x3020     la $a0, string1
18 0x3024     syscall
19 0x3028     jal world
20 0x302c     lw  $ra, 0($fp)      # restore $ra
21 0x3030     addi $sp, $fp, 4     # restore $sp
22 0x3034     lw  $fp, -4($fp)     # restore $fp
23 0x3038     jr $ra
24
25          world:
...           ...
```

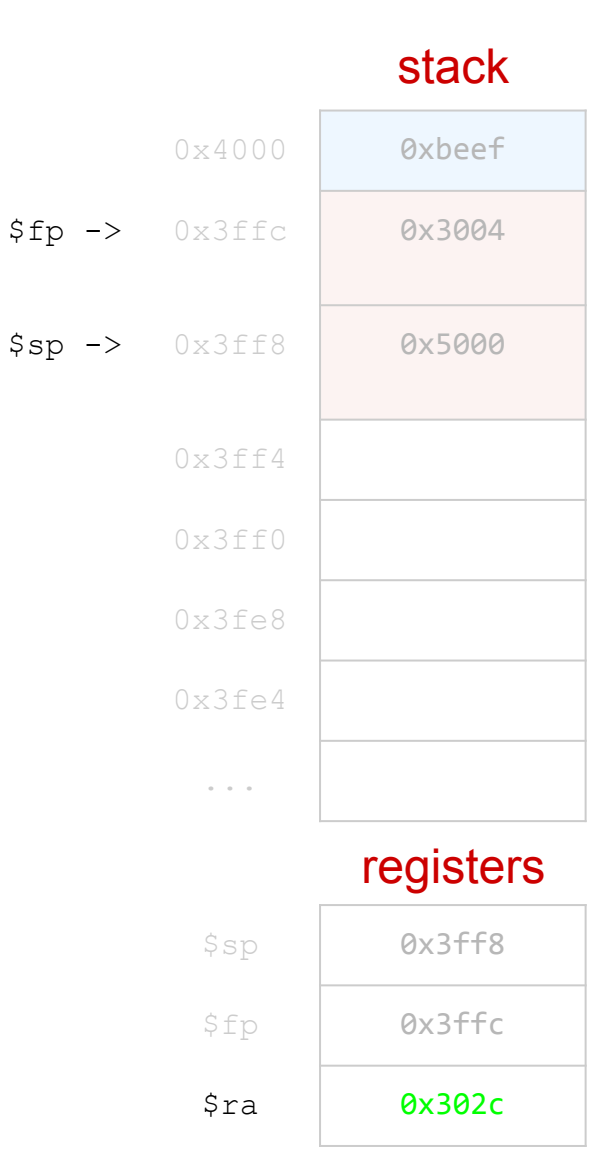
Does It Work?



MIPS code

```
1           .data 0x0
2   0x0000  string1:    .asciiz "Hello"
3   0x0007  string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7   0x3000    jal hello
8   0x3004    ori $v0, $0, 10
9   0x3008    syscall
10
11          hello:
12  0x300c    addi $sp, $sp, -8    # make room on stack
13  0x3010    sw $ra, 4($sp)       # save $ra
14  0x3014    sw $fp, 0($sp)       # save $fp
15  0x3018    addi $fp, $sp, 4     # set $fp
16  0x301c    li $v0, 4
17  0x3020    la $a0, string1
18  0x3024    syscall
19  0x3028    jal world
20  0x302c    lw  $ra, 0($fp)      # restore $ra
21  0x3030    addi $sp, $fp, 4     # restore $sp
22  0x3034    lw  $fp, -4($fp)     # restore $fp
23  0x3038    jr $ra
24
25          world:
...           ...
```

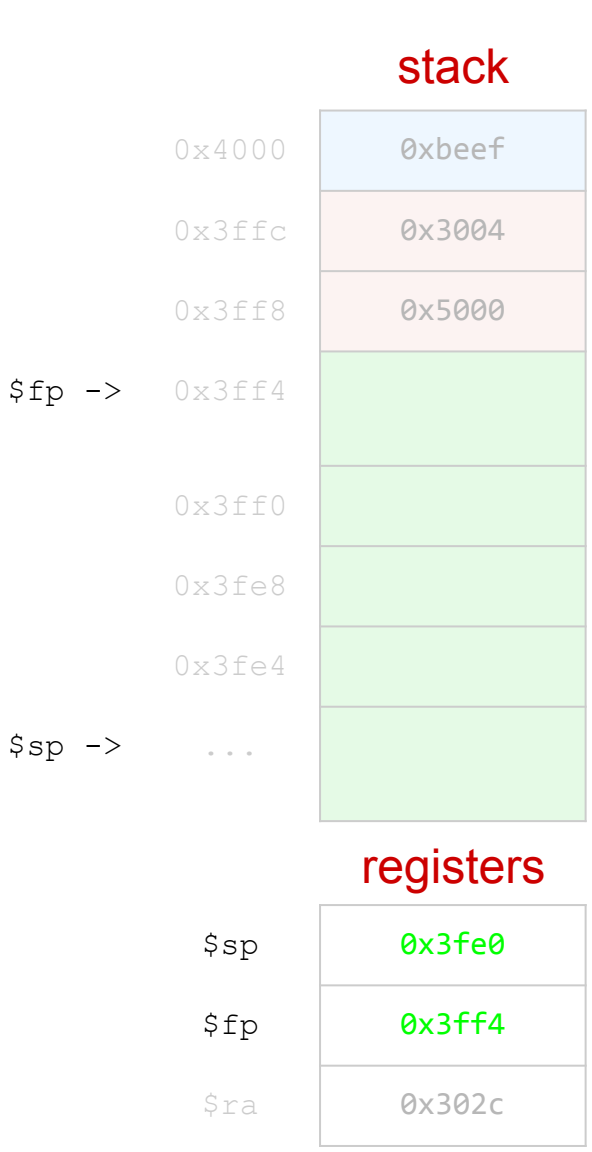
Does It Work?



MIPS code

```
1           .data 0x0
2   0x0000  string1:    .asciiz "Hello"
3   0x0007  string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7   0x3000    jal hello
8   0x3004    ori $v0, $0, 10
9   0x3008    syscall
10
11          hello:
12  0x300c    addi $sp, $sp, -8    # make room on stack
13  0x3010    sw $ra, 4($sp)       # save $ra
14  0x3014    sw $fp, 0($sp)       # save $fp
15  0x3018    addi $fp, $sp, 4     # set $fp
16  0x301c    li $v0, 4
17  0x3020    la $a0, string1
18  0x3024    syscall
19  0x3028    jal world
20  0x302c    lw  $ra, 0($fp)      # restore $ra
21  0x3030    addi $sp, $fp, 4     # restore $sp
22  0x3034    lw  $fp, -4($fp)     # restore $fp
23  0x3038    jr $ra
24
25          world:
...           ...
```

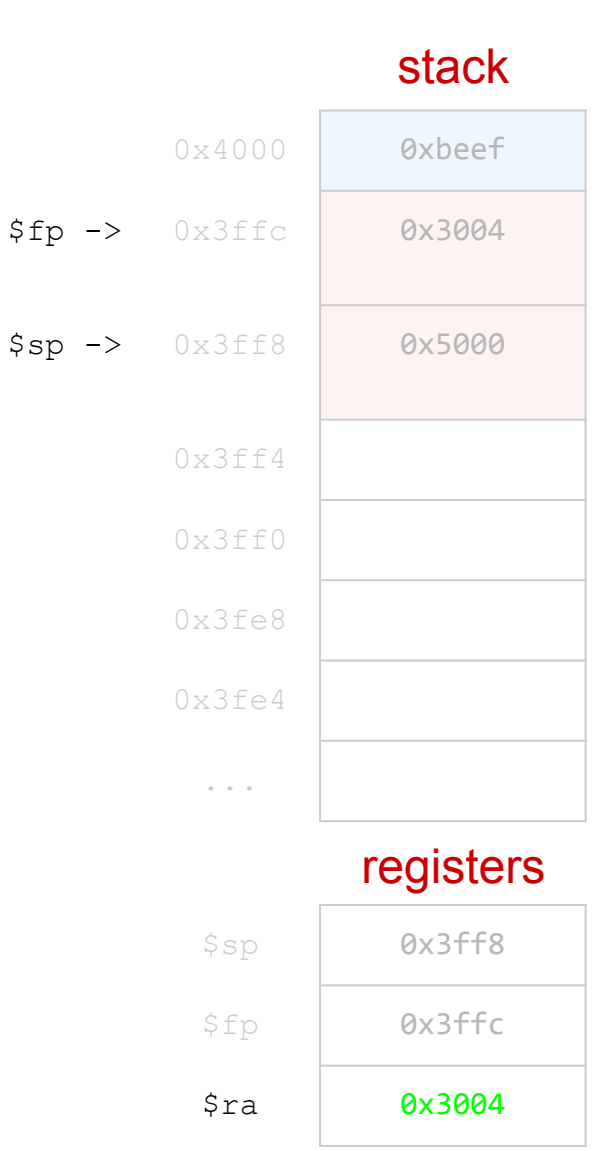
Does It Work?



MIPS code

```
1           .data 0x0
2   0x0000  string1:    .asciiz "Hello"
3   0x0007  string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7   0x3000    jal hello
8   0x3004    ori $v0, $0, 10
9   0x3008    syscall
10
11          hello:
12  0x300c    addi $sp, $sp, -8    # make room on stack
13  0x3010    sw $ra, 4($sp)       # save $ra
14  0x3014    sw $fp, 0($sp)       # save $fp
15  0x3018    addi $fp, $sp, 4     # set $fp
16  0x301c    li $v0, 4
17  0x3020    la $a0, string1
18  0x3024    syscall
19  0x3028    jal world
20  0x302c    lw  $ra, 0($fp)      # restore $ra
21  0x3030    addi $sp, $fp, 4     # restore $sp
22  0x3034    lw  $fp, -4($fp)     # restore $fp
23  0x3038    jr $ra
24
25          world:
...           ...
```

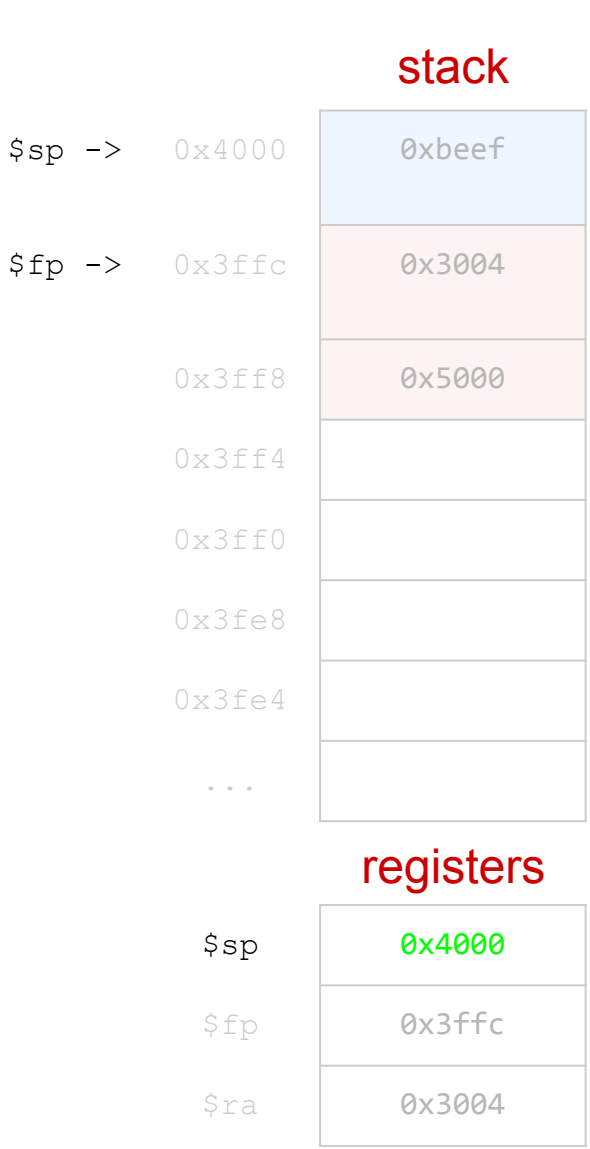
Does It Work?



MIPS code

```
1           .data 0x0
2  0x0000   string1:    .asciiz "Hello"
3  0x0007   string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7  0x3000     jal hello
8  0x3004     ori $v0, $0, 10
9  0x3008     syscall
10
11          hello:
12 0x300c     addi $sp, $sp, -8    # make room on stack
13 0x3010     sw $ra, 4($sp)       # save $ra
14 0x3014     sw $fp, 0($sp)       # save $fp
15 0x3018     addi $fp, $sp, 4     # set $fp
16 0x301c     li $v0, 4
17 0x3020     la $a0, string1
18 0x3024     syscall
19 0x3028     jal world
20 0x302c     lw  $ra, 0($fp)      # restore $ra
21 0x3030     addi $sp, $fp, 4     # restore $sp
22 0x3034     lw  $fp, -4($fp)     # restore $fp
23 0x3038     jr $ra
24
25          world:
...           ...
```

Does It Work?

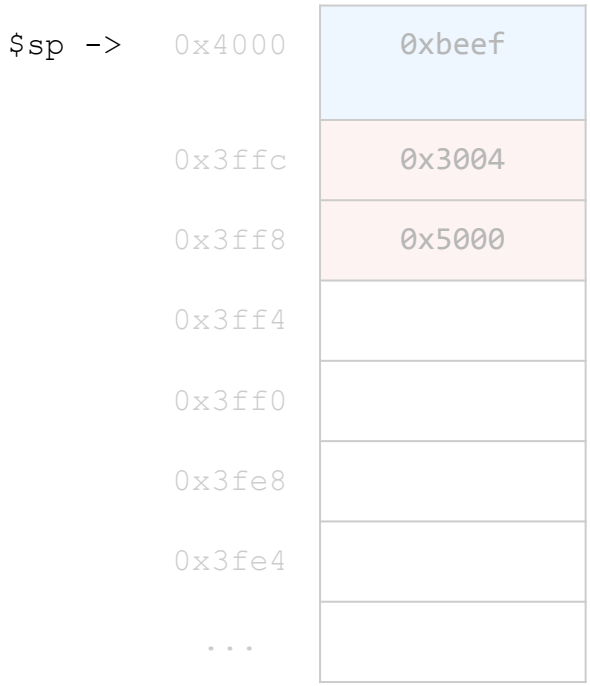


MIPS code

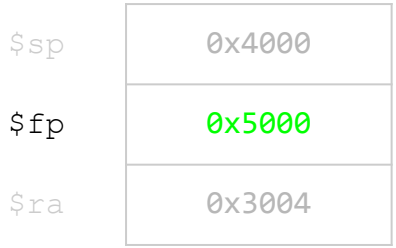
```
1           .data 0x0
2   0x0000  string1:    .asciiz "Hello"
3   0x0007  string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7   0x3000    jal hello
8   0x3004    ori $v0, $0, 10
9   0x3008    syscall
10
11          hello:
12  0x300c    addi $sp, $sp, -8    # make room on stack
13  0x3010    sw $ra, 4($sp)       # save $ra
14  0x3014    sw $fp, 0($sp)       # save $fp
15  0x3018    addi $fp, $sp, 4     # set $fp
16  0x301c    li $v0, 4
17  0x3020    la $a0, string1
18  0x3024    syscall
19  0x3028    jal world
20  0x302c    lw  $ra, 0($fp)      # restore $ra
21  0x3030    addi $sp, $fp, 4     # restore $sp
22  0x3034    lw  $fp, -4($fp)     # restore $fp
23  0x3038    jr $ra
24
25          world:
...           ...
```

Does It Work?

stack



registers



MIPS code

```
1           .data 0x0
2  0x0000   string1:   .asciiz "Hello"
3  0x0007   string2:   .asciiz " world\n"
4
5           .text 0x3000
6           main:
7  0x3000     jal hello
8  0x3004     ori $v0, $0, 10
9  0x3008     syscall
10
11          hello:
12 0x300c     addi $sp, $sp, -8    # make room on stack
13 0x3010     sw $ra, 4($sp)       # save $ra
14 0x3014     sw $fp, 0($sp)       # save $fp
15 0x3018     addi $fp, $sp, 4     # set $fp
16 0x301c     li $v0, 4
17 0x3020     la $a0, string1
18 0x3024     syscall
19 0x3028     jal world
20 0x302c     lw  $ra, 0($fp)      # restore $ra
21 0x3030     addi $sp, $fp, 4     # restore $sp
22 0x3034     lw  $fp, -4($fp)     # restore $fp
23 0x3038     jr $ra
24
25          world:
...           ...
```

Does It Work?

stack

$sp ->	0x4000	0xbeef
	0x3ffc	0x3004
	0x3ff8	0x5000
	0x3ff4	
	0x3ff0	
	0x3fe8	
	0x3fe4	
	...	

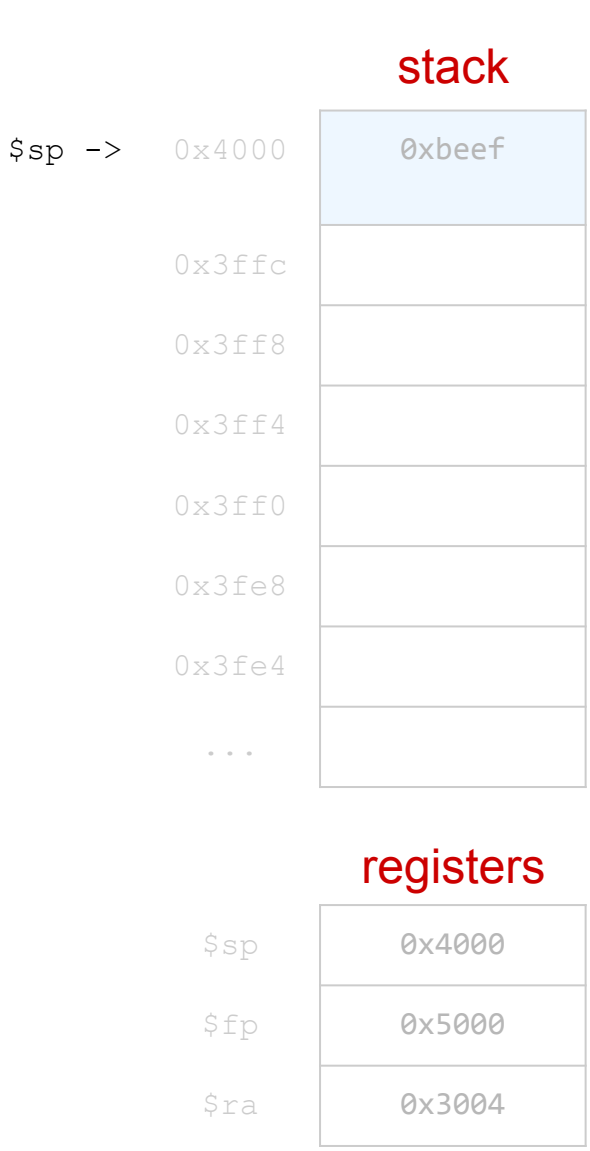
registers

$sp	0x4000
$fp	0x5000
$ra	0x3004

MIPS code

```
1           .data 0x0
2   0x0000  string1:    .asciiz "Hello"
3   0x0007  string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7   0x3000    jal hello
8   0x3004    ori $v0, $0, 10
9   0x3008    syscall
10
11          hello:
12  0x300c    addi $sp, $sp, -8    # make room on stack
13  0x3010    sw $ra, 4($sp)       # save $ra
14  0x3014    sw $fp, 0($sp)       # save $fp
15  0x3018    addi $fp, $sp, 4     # set $fp
16  0x301c    li $v0, 4
17  0x3020    la $a0, string1
18  0x3024    syscall
19  0x3028    jal world
20  0x302c    lw  $ra, 0($fp)      # restore $ra
21  0x3030    addi $sp, $fp, 4     # restore $sp
22  0x3034    lw  $fp, -4($fp)     # restore $fp
23  0x3038    jr $ra
24
25          world:
...           ...
```

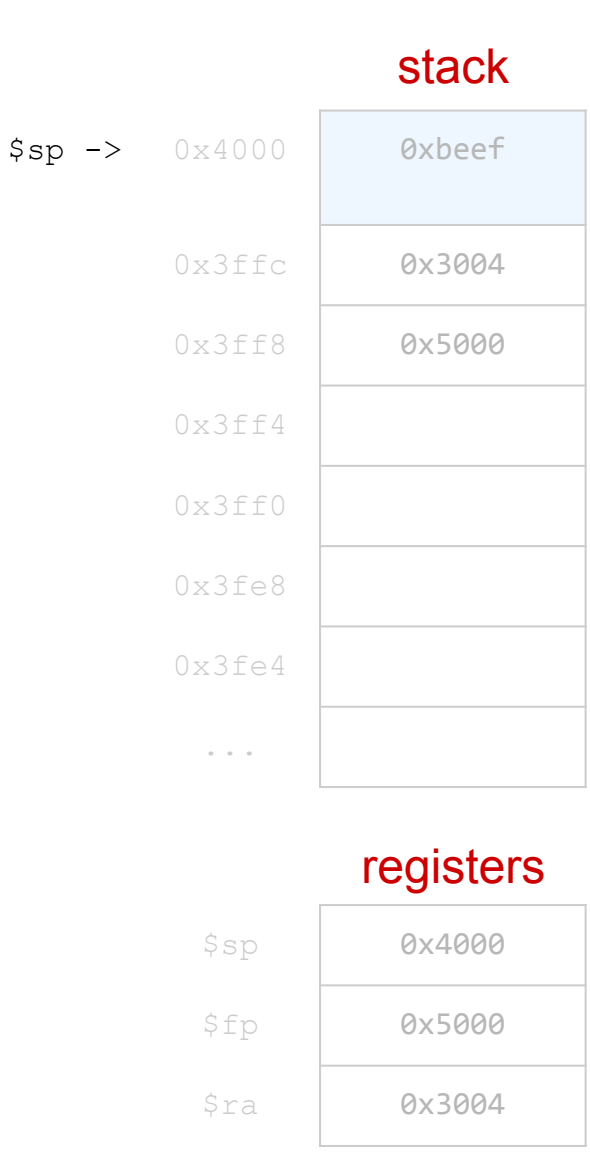
Does It Work?



MIPS code

```
1           .data 0x0
2   0x0000  string1:    .asciiz "Hello"
3   0x0007  string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7   0x3000    jal hello
8   0x3004    ori $v0, $0, 10
9   0x3008    syscall
10
11          hello:
12  0x300c    addi $sp, $sp, -8    # make room on stack
13  0x3010    sw $ra, 4($sp)       # save $ra
14  0x3014    sw $fp, 0($sp)       # save $fp
15  0x3018    addi $fp, $sp, 4     # set $fp
16  0x301c    li $v0, 4
17  0x3020    la $a0, string1
18  0x3024    syscall
19  0x3028    jal world
20  0x302c    lw  $ra, 0($fp)      # restore $ra
21  0x3030    addi $sp, $fp, 4     # restore $sp
22  0x3034    lw  $fp, -4($fp)     # restore $fp
23  0x3038    jr $ra
24
25          world:
...           ...
```

Does It Work?



MIPS code

```
1           .data 0x0
2  0x0000   string1:    .asciiz "Hello"
3  0x0007   string2:    .asciiz " world\n"
4
5           .text 0x3000
6           main:
7  0x3000     jal hello
8  0x3004     ori $v0, $0, 10
9  0x3008     syscall
10
11          hello:
12 0x300c     addi $sp, $sp, -8    # make room on stack
13 0x3010     sw $ra, 4($sp)       # save $ra
14 0x3014     sw $fp, 0($sp)       # save $fp
15 0x3018     addi $fp, $sp, 4     # set $fp
16 0x301c     li $v0, 4
17 0x3020     la $a0, string1
18 0x3024     syscall
19 0x3028     jal world
20 0x302c     lw  $ra, 0($fp)      # restore $ra
21 0x3030     addi $sp, $fp, 4     # restore $sp
22 0x3034     lw  $fp, -4($fp)     # restore $fp
23 0x3038     jr $ra
24
25          world:
...           ...
```

What If We Use More Registers?

The callee should preserve (save and restore if used) the highlighted registers

Name	Register number	Usage
$zero	0	the constant value 0
$at	1	assembler temporary
$v0-$v1	2-3	procedure return values
$a0-$a3	4-7	procedure arguments
$t0-$t7	8-15	temporaries
$s0-$s7	16-23	saved by callee
$t8-$t9	24-25	more temporaries
$k0-$k1	26-27	reserved for operating system
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

What If We Use More Registers?

MIPS code

```
1           .data 0x0
2
3           .text 0x3000
4           main:
5   0x3000    jal proc1
6   0x3004    ori $v0, $0, 10
7   0x3008    syscall
8
9           proc1:
10  0x300c    addi $sp, $sp, -8
11  0x3010    sw $ra, 4($sp)
12  0x3014    sw $fp, 0($sp)
13  0x3018    addi $fp, $sp, 4
14
15  0x301c    addi $sp, $sp, -16
16  0x3020    sw $s0, 12($sp)
17  0x3024    sw $s1, 8($sp)
18  0x3028    sw $s2, 4($sp)
19  0x302c    sw $s3, 0($sp)
20
21  0x3030    addi $sp, $sp, -4
22  0x3034    sw $t0, 0($sp)
23
24  0x3038    jal proc2
...
```

```
...
25
26  0x303c    lw $t0, -24($fp)
27
28            # use $s0-$s3, $t0
29
30  0x3040    lw $s0, -8($fp)
31  0x3044    lw $s1, -12($fp)
32  0x3048    lw $s2, -16($fp)
33  0x304c    lw $s3, -20($fp)
34
35  0x3050    lw  $ra, 0($fp)
36  0x3054    addi $sp, $fp, 4
37  0x3058    lw  $fp, -4($fp)
38  0x305c    jr $ra
39
40          proc2:
...             ...
```

Save and restore $ra, $fp

What If We Use More Registers?

MIPS code

```
1           .data 0x0
2
3           .text 0x3000
4           main:
5   0x3000    jal proc1
6   0x3004    ori $v0, $0, 10
7   0x3008    syscall
8
9           proc1:
10  0x300c    addi $sp, $sp, -8
11  0x3010    sw $ra, 4($sp)
12  0x3014    sw $fp, 0($sp)
13  0x3018    addi $fp, $sp, 4
14
15  0x301c    addi $sp, $sp, -16
16  0x3020    sw $s0, 12($sp)
17  0x3024    sw $s1, 8($sp)
18  0x3028    sw $s2, 4($sp)
19  0x302c    sw $s3, 0($sp)
20
21  0x3030    addi $sp, $sp, -4
22  0x3034    sw $t0, 0($sp)
23
24  0x3038    jal proc2
...
```

```
...
25
26  0x303c    lw $t0, -24($fp)
27
28            # use $s0-$s3, $t0
29
30  0x3040    lw $s0, -8($fp)
31  0x3044    lw $s1, -12($fp)
32  0x3048    lw $s2, -16($fp)
33  0x304c    lw $s3, -20($fp)
34
35  0x3050    lw  $ra, 0($fp)
36  0x3054    addi $sp, $fp, 4
37  0x3058    lw  $fp, -4($fp)
38  0x305c    jr $ra
39
40          proc2:
...           ...
```

Save and restore $s0-$s3

What If We Use More Registers?

MIPS code

```
1           .data 0x0
2
3           .text 0x3000
4           main:
5   0x3000    jal proc1
6   0x3004    ori $v0, $0, 10
7   0x3008    syscall
8
9           proc1:
10  0x300c    addi $sp, $sp, -8
11  0x3010    sw $ra, 4($sp)
12  0x3014    sw $fp, 0($sp)
13  0x3018    addi $fp, $sp, 4
14
15  0x301c    addi $sp, $sp, -16
16  0x3020    sw $s0, 12($sp)
17  0x3024    sw $s1, 8($sp)
18  0x3028    sw $s2, 4($sp)
19  0x302c    sw $s3, 0($sp)
20
21  0x3030    addi $sp, $sp, -4
22  0x3034    sw $t0, 0($sp)
23
24  0x3038    jal proc2
...
```

```
...
25
26  0x303c    lw $t0, -24($fp)
27
28            # use $s0-$s3, $t0
29
30  0x3040    lw $s0, -8($fp)
31  0x3044    lw $s1, -12($fp)
32  0x3048    lw $s2, -16($fp)
33  0x304c    lw $s3, -20($fp)
34
35  0x3050    lw  $ra, 0($fp)
36  0x3054    addi $sp, $fp, 4
37  0x3058    lw  $fp, -4($fp)
38  0x305c    jr $ra
39
40          proc2:
...             ...
```

Save and restore $t0

What If We Use More Registers?

MIPS code

```
1           .data 0x0
2
3           .text 0x3000
4           main:
5   0x3000    jal proc1
6   0x3004    ori $v0, $0, 10
7   0x3008    syscall
8
9           proc1:
10  0x300c    addi $sp, $sp, -8
11  0x3010    sw $ra, 4($sp)
12  0x3014    sw $fp, 0($sp)
13  0x3018    addi $fp, $sp, 4
14
15  0x301c    addi $sp, $sp, -16
16  0x3020    sw $s0, 12($sp)
17  0x3024    sw $s1, 8($sp)
18  0x3028    sw $s2, 4($sp)
19  0x302c    sw $s3, 0($sp)
20
21  0x3030    addi $sp, $sp, -4
22  0x3034    sw $t0, 0($sp)
23
24  0x3038    jal proc2
...
```

```
...
25
26  0x303c    lw $t0, -24($fp)
27
28            # use $s0-$s3, $t0
29
30  0x3040    lw $s0, -8($fp)
31  0x3044    lw $s1, -12($fp)
32  0x3048    lw $s2, -16($fp)
33  0x304c    lw $s3, -20($fp)
34
35  0x3050    lw  $ra, 0($fp)
36  0x3054    addi $sp, $fp, 4
37  0x3058    lw  $fp, -4($fp)
38  0x305c    jr $ra
39
40          proc2:
...           ...
```

Call proc2