MIPS Stack

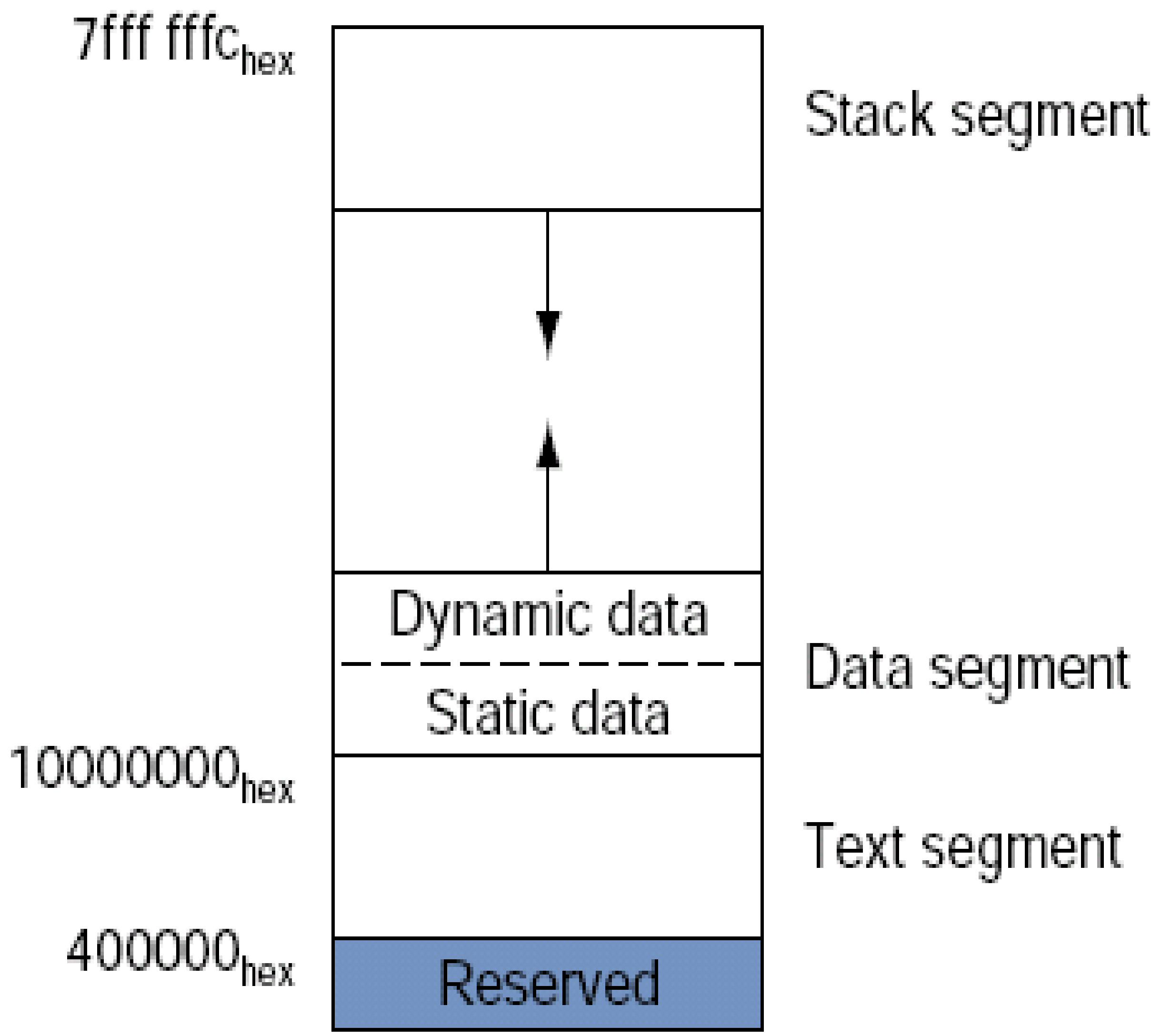
Giovani Gracioli

giovani@lisha.ufsc.br

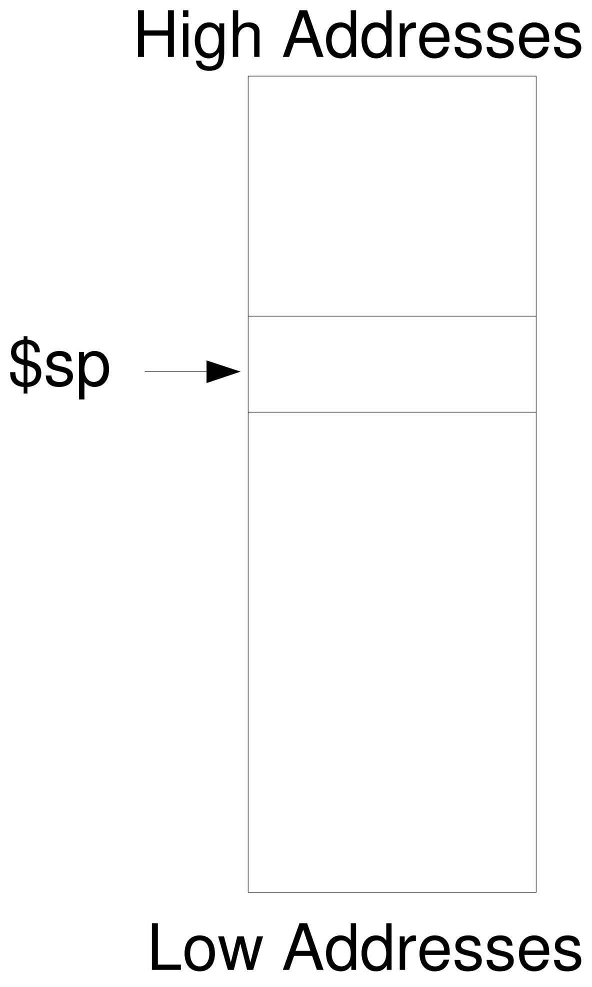
http://www.lisha.ufsc.br/~giovani

Set 2007

Memory Usage



The Stack



- The stack grows from High Addresses to Low Addresses
- Stack pointer ($sp)

Registers and Instructions for Procedures Calls

- Registers $a0-$a3 are used to pass arguments to procedures
- Register $v0-$v1 are used to return values from procedures
- $ra – return address register
- jal procedure (Jump And Link)
- jr $ra (Jump Register)

Calling a Procedure

- Six steps are executed during the procedure call:
  1. to put the parameters in a place that can be accessed by procedure;
  2. to transfer the control to the procedure;
  3. to guarantee the memory resources;
  4. to execute the task;
  5. to put the return value in a place that can be accessed by the program;
  6. to return the control to the source point.

Saving the Return Address

push $ra =

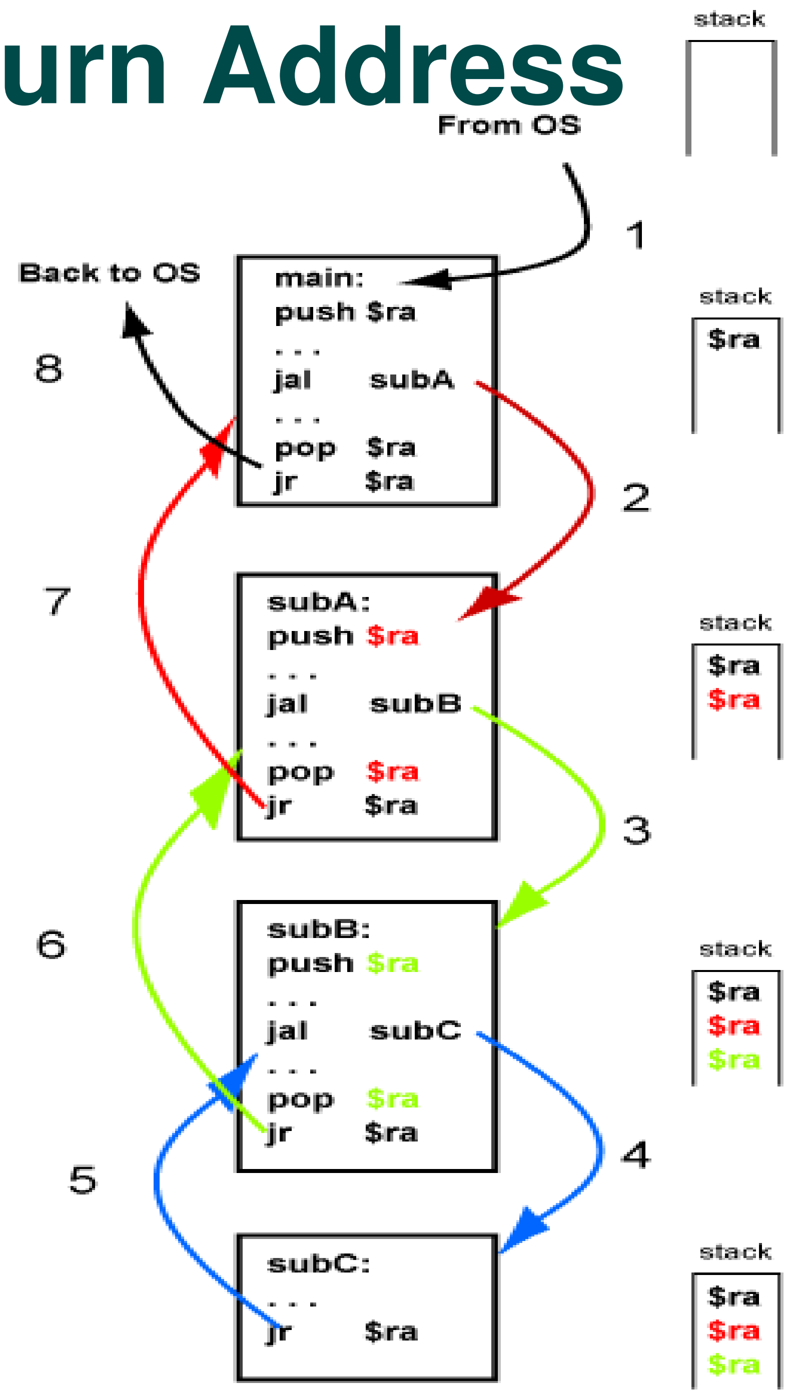
sub $sp, $sp – 4

sw $ra, 0($sp)

pop $ra =

lw $ra, 0($sp)

addi $sp, $sp, 4



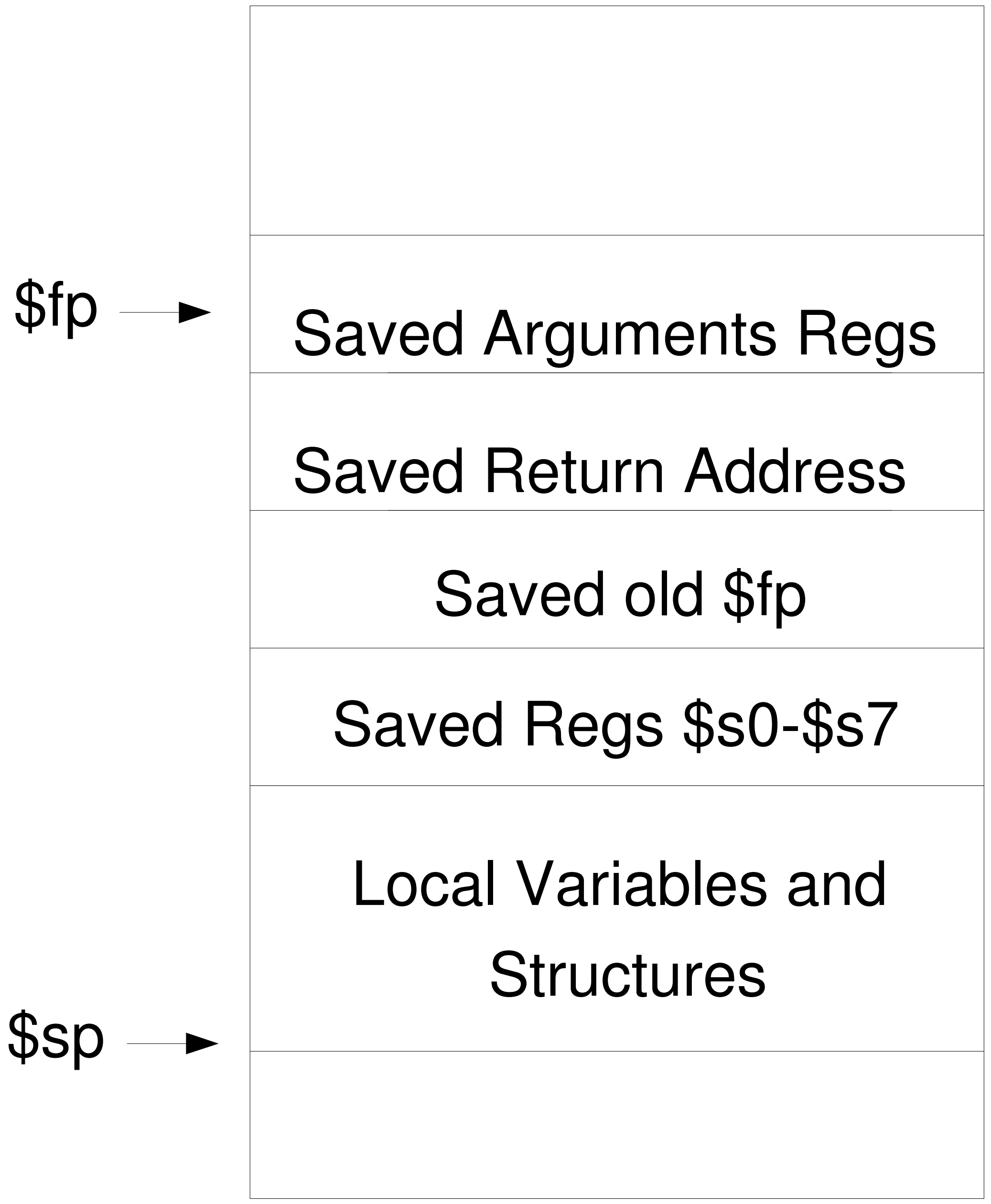
Returning a value

```
int function(int a, int b) {
    return = a + b;
}

.globl function
function:
    add $v0, $a0, $a1
    jr $ra
```

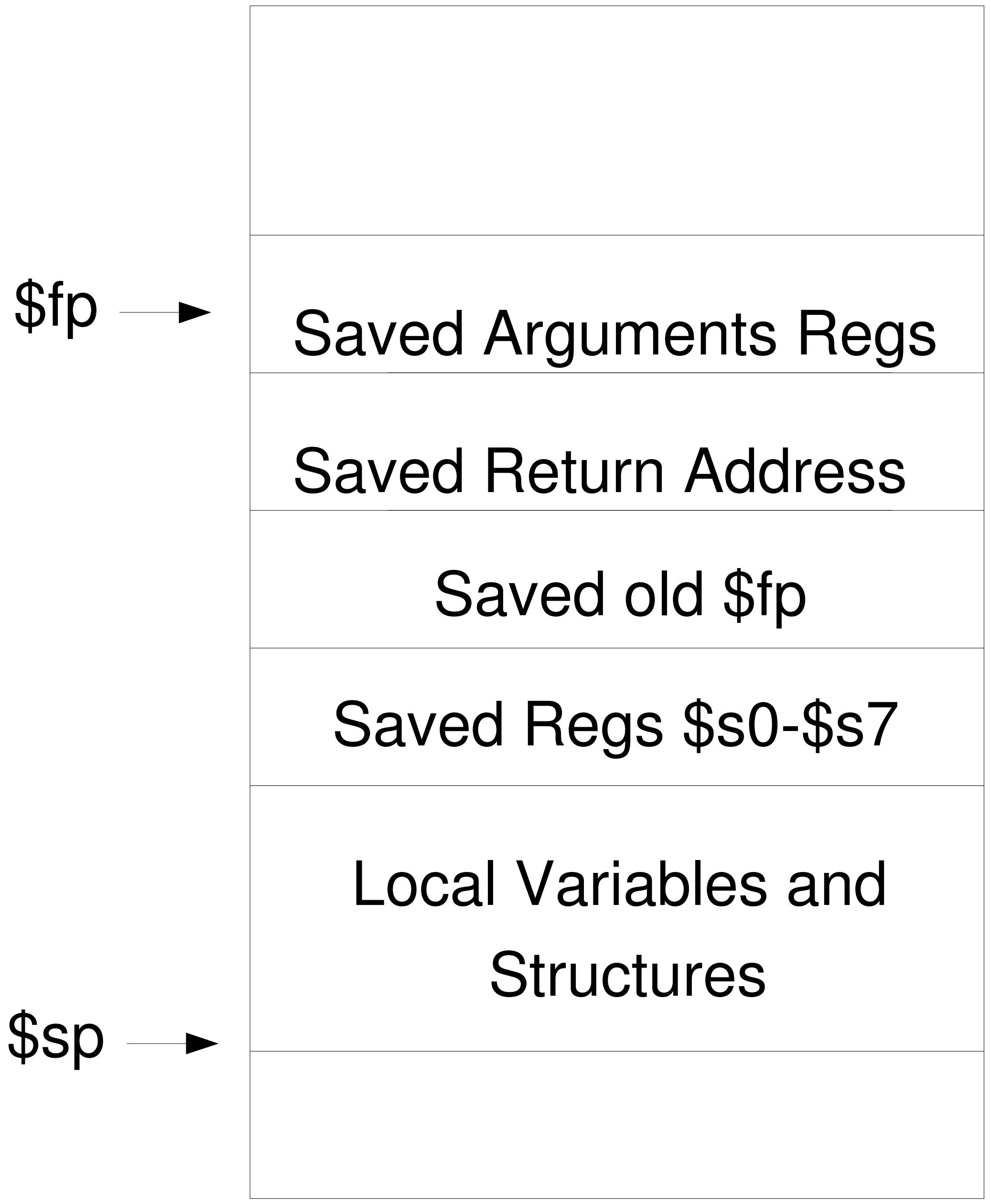
MIPS Stack Functionalities

- Save arguments regs (if necessary)
- Save the return address register ($ra)
- Save the old value of $fp
- Save regs $s0-$s7 (if necessary)
- Pass more than 4 arguments
- Declare Local Variables and Stuctures (if exists)



MIPS Stack Functionalities

- $fp points to the first word of frame
- $sp points to the last word of frame



Example: local variables

- Save the local variables or structures on the stack

```
void func_a(void) {
        int a = 10;
        int b = 20;
        return;
}
```

```
.globl func_a
func_a:
sub $sp, $sp, 12
sw $ra, 8($sp)
li $t8, 10
sw $t8, 4($sp)
li $t8, 20
sw $t8, 0($sp)
lw $ra, 8($sp)
addi $sp, $sp, 12
jr $ra
```

Example: saving the $s* register

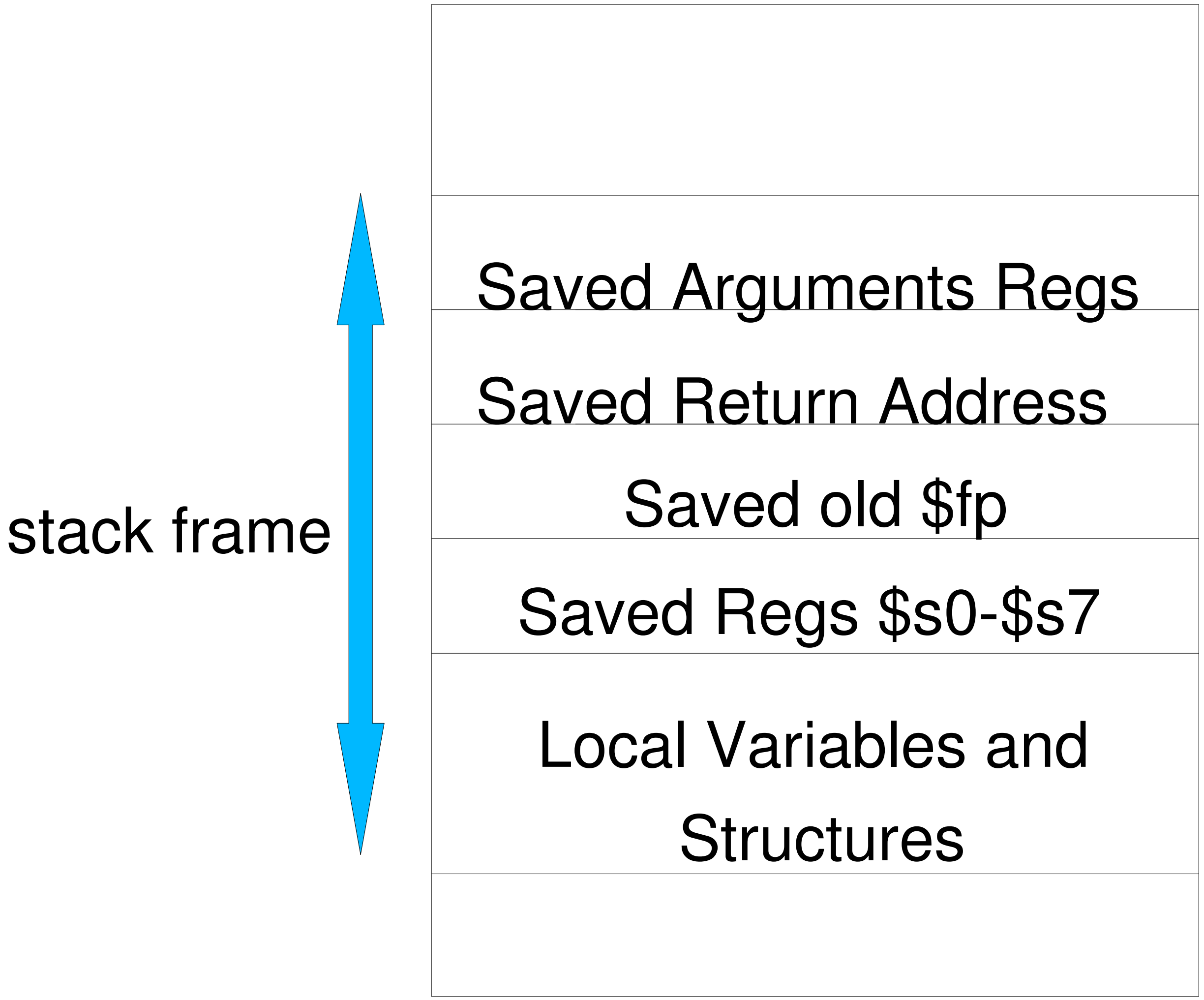
- The registers $s0-$s7 must be saved inside the procedure stack if they are used by the program

```
void func_a(void) {
    int a = 10;
    int b = 20;
    ...
    return;
}
```

```
.globl func_a
func_a:
    sub $sp, $sp, 20
    sw $ra, 16($sp)
    sw $s0, 12($sp)
    sw $s5, 8($sp)
    li $t0, 10
    sw $t0, 4($sp)
    li $t0, 20
    sw $t0, 0($sp)
    ....
    lw $ra, 16($sp)
    lw $s0, 12($sp)
    lw $s5, 8($sp)
    addi $sp, $sp, 20
    jr $ra
```

Example: frame pointer

- stack frames or activation records are the stack segments that have the saved registers and local variables



Example: frame pointer

```
int func_a(int a, int b) {
    int var_local;
    var_local = func_b();
    ...
    return var_local;
}
int func_b(void) {
    int var = 20;
    ...
    return var;
}
```

Example: frame pointer

```
.globl func_a
func_a:
    sub $sp, $sp, 12
    sw $ra, 8($sp)
    sw $fp, 4($sp)
    move $fp, $sp # $fp = $sp
    jal func_b
    sw $vo, 0($fp)
    ....
    lw $v0, 0($fp)
    move $sp, $fp # $sp = $fp
    lw $fp, 4($sp)
    lw $ra, 8($sp)
    addi $sp, $sp 12
    jr $ra
```

```
.globl func_b
func_b:
    sub $sp, $sp, 12
    sw $ra, 8($sp)
    sw $fp, 4($sp)
    move $fp, $sp
    li $t0, 20
    sw $t0, 0($fp)
    ....
    lw $v0, 0($fp)
    move $sp, $fp
    lw $fp, 4($sp)
    lw $ra, 8($sp)
    addi $sp, $sp, 12
    jr $ra
```