King Fahd University of Petroleum and Minerals
College of Computer Sciences and Engineering
Computer Engineering Department
COE 301: Computer Architecture

LAB 07:
MIPS Functions and Stack Segment

Saleh AlSaleh

Agenda

- Caller vs. Callee
- Functions: Declaration, Execute (Call), Return Back
- Registers Use
- Stack Segment
- Recursive Function Example
- Live Examples
- Tasks

Caller vs. Callee

- The function that initiates the call to another function is known as **Caller.**
- The function that receives and executes the call is known as the **Callee.**
- To execute a function, the program must follow these steps:
  - The **caller** must put the parameters in a place where the **callee** function can access them
  - Transfer control to the **callee** function
  - Execute the **callee** function
  - The **callee** function must put the results in a place where the **caller** can access them
  - Return control to the **caller** (point of origin) next to where the call was made

Functions: Declaration, Execution(Call), Return Back

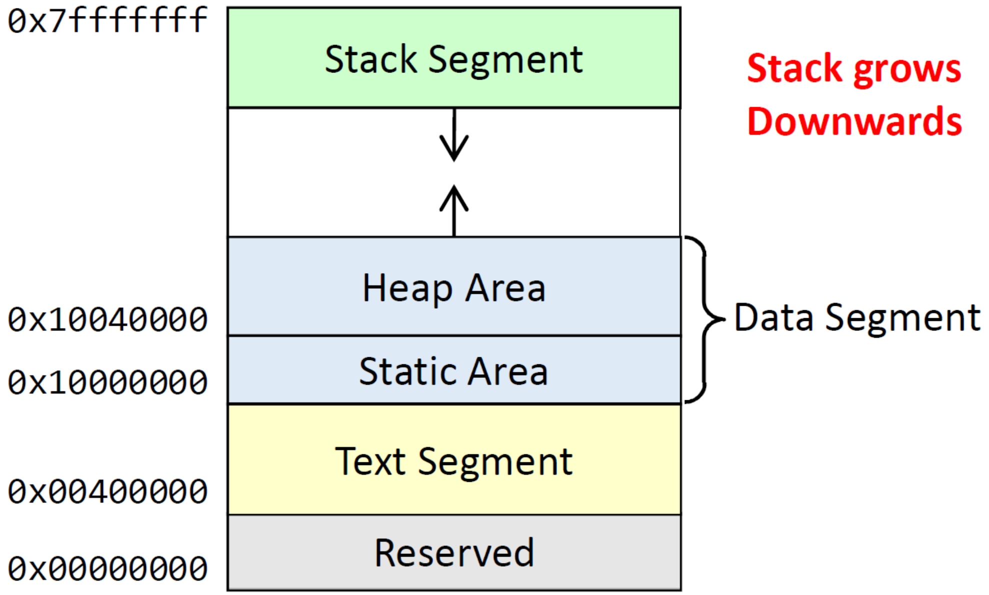
- Declaration:
  - Define a label similar to if statements and loops
  - Write the body of the function after the label
- Execution:
  - Prepare the arguments in $a0-$a3 registers
  - Call the function using the jal instruction (e.g. jal function)
- Return Back
  - Prepare the results if any in $v0-$v1 registers
  - Return to the caller using jr instruction (jr $ra)

Registers Use

Register Name	Register Number	Register Usage
$zero	$0	Always zero, forced by hardware
$at	$1	Assembler Temporary register, reserved for assembler use
$v0 - $v1	$2 - $3	Results of a function
$a0 - $a3	$4 - $7	Arguments of a function
$t0 - $t7	$8 - $15	Registers for storing temporary values
$s0 - $s7	$16 - $23	Registers that should be saved across function calls
$t8 - $t9	$24 - $25	Registers for storing more temporary values
$k0 - $k1	$26 - $27	Registers reserved for the OS kernel use
$gp	$28	Global Pointer register that points to global data
$sp	$29	Stack Pointer register that points to top of stack
$fp	$30	Frame Pointer register that points to stack frame
$ra	$31	Return Address register used to return from a function call

Stack Segment

- Stack Segment provides an area that can be allocated and freed by functions. The programmer has no control over where these segments are located in memory.
- The stack segment can be used by functions for passing many parameters, for allocating space for local variables, and for saving and preserving registers across calls.
- Without the stack segment in memory, it would be impossible to write recursive functions, or pure functions that have no side effects.



Recursive Function Example

```
int fact (int n) {
    if (n<2) return 1;
    else return (n*fact(n-1));
}
```

```
fact:
    bge $a0, 2, else  # branch if (n >= 2) to else
    li $v0, 1         # $v0 = 1
    jr $ra            # return to caller
else:
    addi $sp, $sp, -8 # allocate a stack frame of 8 bytes
    sw $a0, 0($sp)    # save the argument n
    sw $ra, 4($sp)    # save the return address
    addi $a0, $a0, -1 # argument $a0 = n-1
    jal fact          # call fact(n-1)
    lw $a0, 0($sp)    # restore $a0 = n
    lw $ra, 4($sp)    # restore return address
    mul $v0, $a0, $v0 # $v0 = n * fact(n-1)
    addi $sp, $sp, 8  # free stack frame
    jr $ra            # return to the caller
```

Live Examples