

# Procedures

Functions in MIPS

# Simple Procedure

Refer  
fun1.asm

- Beginning of *.text* is MAIN (where program starts executing)
- End of main / any program (similar to exit):

```
li $v0, 10
```

```
syscall
```

- Function

- begins with a label (name of the function)
- ends with:

```
jr $ra
```

Return to the  
instruction  
following  
function call

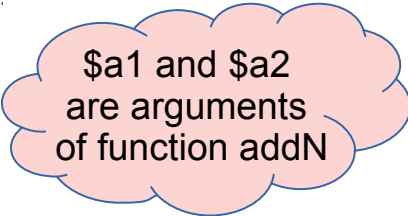
- call:

```
jal <label>
```

# Arguments and Return Values

- Arguments in \$a registers

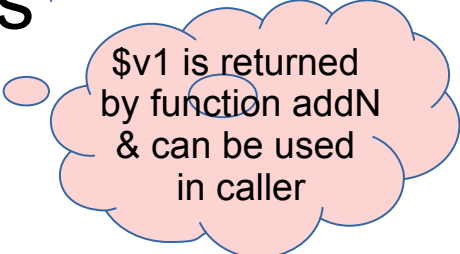
```
addi $a1, $zero, 0  
addi $a2, $zero, 100  
jal addN
```



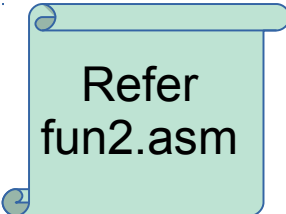
\$a1 and \$a2  
are arguments  
of function addN

- Return values in \$v registers

```
addN:  
add $v1, $a1, $a2  
jr $ra
```

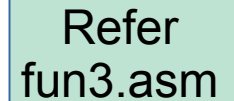


\$v1 is returned  
by function addN  
& can be used  
in caller



Refer  
fun2.asm

# Nested Procedures

A light green rectangular box with rounded corners and a blue border, containing the text "Refer fun3.asm".

- Remember!!! \$ra stores return address when there is a function call.
- There is only 1 register (\$ra) to store return address
- Then, how to call a function from a function?
  - Store \$ra before a nested function call in a STACK

Before nested call:

store in stack

```
addi $sp, $sp, -4
```

```
sw $ra, 0($sp)
```

After nested call:

restore from stack

```
lw $ra, 0($sp)
```

```
addi $sp, $sp, 4
```

# Assignment

- String argument
- Array argument
- Return String
- Return array
- Retain values in \$s registers after procedure calls