# 5.18 jal, jr: Subroutine instructions

**A brief note from your instructor:**

Sections 5.18-5.21 provide a good conceptual description of how MIPS subroutines work, using temporary registers ($t) and parameters. However, practically speaking their translation is not accurate because temporary registers by definitio saved across procedure calls, and MIPS provides $a registers for parameters.

Therefore I recommend skimming this section but will not require completing the activities. I will be demonstrating in cl way of translating subroutines to MIPS, using these concepts.

## Subroutines

A program often needs to perform the same operation for different data values. Ex: Determining the maximum of two valu temperature from Fahrenheit to Celsius, etc. Instead of duplicating the instruction sequence for an operation multiple time can use a subroutine. A ***subroutine*** is a sequence of instructions that performs a specific operation that can be called from a program. A subroutine call causes the subroutine's statements to execute.

**PARTICIPATION ACTIVITY** 5.18.1: Subroutine for computing maximum of two values.

Start 2x speed

```
# Compute max of DM[5000] & DM[5004]
addi $t6, $zero, 5000
lw $t0, 0($t6)
addi $t6, $zero, 5004
lw $t1, 0($t6)
```


```
# Compute max of DM[5000] & DM[5004]
addi $t6, $zero, 5000
lw $t0, 0($t6)
addi $t6, $zero, 5004
lw $t1, 0($t6)
```

```
slt $t3, $t0, $t1
bne $t3, $zero, Max1IsT1
add $t2, $zero, $t0
j Max1End
Max1IsT1: add $t2, $zero, $t1
Max1End: addi $t6, $zero, 5008
sw $t2, 0($t6)

# Compute max of DM[5012] & DM[5016]
addi $t6, $zero, 5012
lw $t0, 0($t6)
addi $t6, $zero, 5016
lw $t1, 0($t6)
slt $t3, $t0, $t1
bne $t3, $zero, Max2IsT1
add $t2, $zero, $t0
j Max2End
Max2IsT1: add $t2, $zero, $t1
Max2End: addi $t6, $zero, 5020
sw $t2, 0($t6)
```

No subroutine with redundant code

```
Call CompMax subroutine (not shown)
addi $t6, $zero, 5008
sw $t2, 0($t6)

# Compute max of DM[5012] & DM[5016]
addi $t6, $zero, 5012
lw $t0, 0($t6)
addi $t6, $zero, 5016
lw $t1, 0($t6)
Call CompMax subroutine (not shown)
addi $t6, $zero, 5020
sw $t2, 0($t6)

CompMax:
   slt $t3, $t0, $t1
   bne $t3, $zero, MaxIsT1
   add $t2, $zero, $t0
   j MaxEnd
   MaxIsT1: add $t2, $zero, $t1
   MaxEnd: Return from subroutine
```

CompMax subroutine computes max of $t0 and $t1, writing max to $t2.

**PARTICIPATION ACTIVITY** 5.18.2: Subroutines.

Refer to the animation above.

1) What label indicates the first instruction of the subroutine for computing the maximum value?

   - CompMax
   - MaxEnd

2) How many redundant instructions in the original code were moved to the subroutine.
   - ○ 5
   - ○ 10

3) A subroutine's instructions must be duplicated each time the subroutine is called.
   - ○ True
   - ○ False

4) A subroutine may have up to 1024 instructions.
   - ○ True
   - ○ False

## Jump and link and jump register instructions

The ***jump and link*** (***jal***) instruction stores the address of the next instruction in register $ra, and then jumps to the instructio specified location. Ex: `jal CalcCube` stores the address of the instruction after the jal instruction in $ra, and continues exe instruction at CalcCube; CalcCube is the label for the first instruction of the subroutine. The ***$ra*** register (or ***return address*** the instruction address to which a subroutine returns after executing. The ***jump register*** (***jr***) instruction jumps to the instruc address held in a register. Ex: `jr $ra` jumps to the instruction at the address held in register $ra. A programmer uses jal to subroutine, and jr to return from a subroutine.

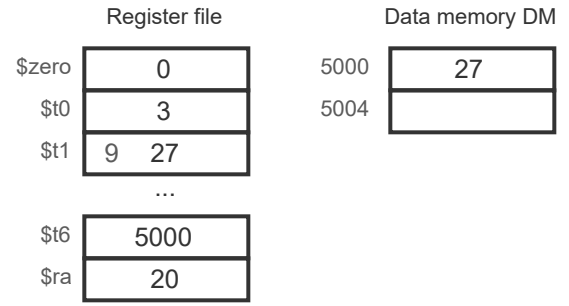
**PARTICIPATION ACTIVITY** 5.18.3: Subroutine call using jal and jr instructions.

Start ☐ 2x speed

```
   # Compute cube of 3
12 addi $t0, $zero, 3
16 jal CalcCube    # Call CalcCube
20 sw $t1, 0($t6)

   ...

   # CalcCube subroutine
   CalcCube:
60    mul $t1, $t0, $t0
64    mul $t1, $t1, $t0
68    jr $ra
```




**PARTICIPATION ACTIVITY** 5.18.4: jal and jr instructions.

1) Write a jump and link instruction to call a subroutine named CalcTip.

Check Show answer

2) If the jal instruction below is located in instruction memory at address 200, what value is written to register $ra?

```
jal DetSpeed
```

Check Show answer

3) Using the $ra register, write an instruction to return from a subroutine named CalcTip.

Check Show answer

4) Assume $ra holds 116. If the jr instruction below is located in instruction memory at address 200, what is the address of the instruction executed after `jr $ra`?

Check Show answer

## Arguments and return values

An ***argument*** is a value passed to a subroutine, that influences the subroutine's operations. A ***return value*** is a value returne subroutine. A simple subroutine may use specific registers for the argument and return value. Ex: The CalcCube subroutine for the subroutine's argument and $t1 for the return value.

The assembly program below passes arguments to the CalcCube subroutine using $t0. The CalcCube subroutine returns t $t1. The program first passes 3 to the subroutine by writing 3 to register $t0. After executing the subroutine, $t1 holds the v stored in data memory at address 5000. The program then passes 17 to the subroutine by writing 17 to $t0. The result of 4 stored to data memory at address 5004.

Figure 5.18.1: Passing arguments to multiple CalcCube subroutine calls.

```
# Initialize registers for DM addresses
addi $t5, $zero, 5000
addi $t6, $zero, 5004

# Compute cube of 3
addi $t0, $zero, 3  # Pass argument of 3
jal CalcCube        # Call CalcCube
sw $t1, 0($t5)      # Store result to DM[5000]

# Compute cube of 17
addi $t0, $zero, 17  # Pass argument of 17
jal CalcCube         # Call CalcCube
sw $t1, 0($t6)       # Store result to DM[5004]
j Done

# CalcCube subroutine.
#    $t0 is subroutine argument
#    $t1 is subroutine return value
CalcCube:
   mul $t1, $t0, $t0
   mul $t1, $t1, $t0
   jr $ra            # Return from subroutine

Done:
```

**PARTICIPATION ACTIVITY** 5.18.5: Subroutine arguments and return values.

The CalcEq subroutine below evaluates the equation: x * (y - z). Values for x, y, and z are passed to the subroutine as arguments.

```
CalcEq:
   sub $t5, $t1, $t2
   mul $t3, $t5, $t0
   jr $ra
```

1) Which register is used for x?

- $t0
- $t1
- $t2

2) Which register is used for the argument y?

- $t0
- $t1
- $t2

3) Which register is used for the argument z?

- $t2
- $t3

4) Which register is used for the return value?

- $ra
- $t3

**PARTICIPATION ACTIVITY** 5.18.6: Create a subroutine.

Using the CompMax subroutine, complete the assembly program to compute the maximum of the three values in DM[5000], DM[5004], and DM[5008], storing the result in DM[5020].

1. Load $t0 and $t1 with DM[5000] and DM[5004], and call the CompMax subroutine.
2. Copy the result, which is held in $t2, into $t0. Load $t1 with DM[5008]. Call the CompMax subroutine.
3. Store the result, which is held in $t2, to DM[5020]

Assembly

```
 Line 1  # FIXME: Compute maximum of DM[5000],
 Line 2  # DM[5004], and DM[5008]
 Line 3  j Done
 Line 4
 Line 5  CompMax:
 Line 6     slt $t3, $t0, $t1
 Line 7     bne $t3, $zero, MaxIsT1
 Line 8     add $t2, $zero, $t0
 Line 9     j MaxEnd
Line 10     MaxIsT1: add $t2, $zero, $t1
Line 11     MaxEnd: jr $ra
Line 12
Line 13  Done:
```

Registers

| | |
|---|---|
| $zero | 0 |
| $t0 | 0 |
| $t1 | 0 |
| $t2 | 0 |
| $t3 | 0 |
| $t4 | 0 |
| $t5 | 0 |
| $t6 | 0 |

| | |
|---|---|
| 5000 | |
| 5004 | |
| 5008 | |
| 5020 | |

ENTER SIMULATION STEP RUN

More options

## Table 5.18.1: Instruction summary: jal, jr.

| Instruction | Format | Description | Example |
|---|---|---|---|
| jal | `jal JLabel` | Jump and link: Stores the address of the next instruction in register $ra, and continues execution with the instruction at JLabel. | `jal CalcTip` |
| jr | `jr $a` | Jump register: Causes execution to continue with the instruction at address $a. | `jr $t3` |

CHALLENGE

**ACTIVITY** 5.18.1: Call and create subroutines.

Start

Pass DM[$t3] to the YearlySalary subroutine, and store the return value to DM[$t4].
$t1 is the subroutine argument.
$t2 is the subroutine return value.

```
        lw   $t1, 0($t1)
        lw   $t1, 0($t1)
        lw   $t1, 0($t1)
        lw   $t1, 0($t1)
YearlySalary: addi $t2, $zero, 2000
        mul  $t2, $t1, $t2
        jr   $ra
Done:
```

Registers

| | |
|---|---|
| $t1 | 0 |
| $t2 | 0 |
| $t3 | 5000 |
| $t4 | 5004 |

1 2

1 2

Check Next

Provide feedback on this section