MIPS Assembly Language Syntax

[label:] Op-Code [operand], [operand], [operand] [#comment]

Label	***Op-Code***	***Dest.***	***S1,***	***S2***	***Comment***
chico:	**add**	**$a0,**	**$t0,**	**$t1**	**# a0 = t0 + t1**

Translation an Arithmetic Expression
$s0 = srt ($a0 * $a0 + $a1 * $a1)

Hypotenuse:

```
        mult  $a0, $a0     # Square $a0
        mflo  $t0          # t0 = Lower 32-bits of product
        mult  $a1, $a1     # Square $a1
        mflo  $t1          # t1 = Lower 32-bits of product
        add   $a0, $t0, $t1        # a0 = t0 + t1
        jal   srt          # Call the square root function
        move $s0, $v0      # By convention, the result of sqr
                           # is returned in $v0
```

Area of a Circle
$s0 = π * $t8 * $t8

Area:

```
        li    $t0, 314156        # Load immediate Pi scaled up  100,000
        mult  $t8, $t8           # Radius squared
        mflo  $t1          # Move lower 32-bits of product in
                           # Low register to $t1
        mult  $t1, $t0           # Multiply by scaled Pi
        mflo  $s0          # Move lower 32-bits of product in
```

```
                        # Low register to $s0
        li      $t1, 100000     # Load immediate scale factor of 100,000
        div     $s0, $t1        # Divide by scale factor
        mflo    $s0         # Truncated integer result left in $s0
```

Translation of an "if … then … else …" Control Structure

if ($t8 < 0) **then**
{$s0 = 0 - $t8; $t1 = $t1 +1}
else
{$s0 = **$**t8; $t2 = $t2 + 1}

```
        bgez $t8, else     # if ($t8 is greater than or
        # equal to zero) branch to else
        sub   $s0, $zero, $t8 # $s0 gets the negative of $t8
        addi  $t1, $t1, 1  # increment $t1 by 1
        b     next         # branch around the else code
else:
        move $s0, $t8      # $s0 gets a copy of $t8
        addi  $t2, $t2, 1  # increment $t2 by 1
next:
```

Translation of a "while" Control Structure

while ($a1 < $a2) **do**
{ $a1 = $a1 + 1
$a2 = $a2 - 1}

```
while:
        bgeu $a1, $a2, done    # If( $a1 >= $a2) Branch to done
        addi  $a1, $a1, 1   # $a1 = $a1 + 1
        addi  $a2, $a2, -1          # $a2 = $a2 - 1
        b     while           # Branch to while
done:
```

Translation of a "for" Loop Control Structure

```
$a0 = 0;
for ( $t0 =10; $t0 > 0; $t0 = $t0 -1) do
{$a0 = $a0 + $t0}
```

```
li      $a0, 0              #  $a0 = 0
        li      $t0, 10             # Initialize loop counter to 10
loop:
        add     $a0, $a0, $t0
        addi    $t0, $t0, -1 # Decrement loop counter
        bgtz    $t0, loop    # If ($t0 > 0) Branch to loop
```

Now for an example, let us suppose that we want to write an assembly language program to find the sum of the integers from 1 to N.

In other words do the following: 1 + 2 + 3 + 4 + 5 + 6 + 7 + ...+ N,

where "N" is an input value.

On the next slide you will see a pseudocode description of the algorithm and following that the corresponding assembly language program, where processor register $t0 is used to accumulate the sum, and processor register $v0 is used as a loop counter.

Use a word processor to create the following program file.

Be sure to save as text only.

Next, load the program into SPIM. Run the program and experiment with the different features of the MIPS simulator. (For example: Single Step)

Read the help file for a description of how to use the simulator.

An Example MIPS Program

```
#  Program #1 : (descriptive name)          Programmer: YOUR NAME
#  Due Date  : Sep. 26,2022                 Course: CSIS-2810
#  Last Modified: Sep. 13, 2022
# Functional Description:  Find the sum of the integers from 1 to N where
#  N is a value input from the keyboard.
#############################################################
# Algorithmic Description in Pseudocode:
# main:    v0 <<  value read from the keyboard (syscall 4)
#               if (v0 < = 0 ) stop
#              t0 = 0;              # t0 is used to accumulate the sum
#              While (v0 > 0)  { t0 = t0 + v0;  v0 = v0 - 1}
#              Output to monitor syscall(1) << t0;     goto main
#############################################################
# Register Usage: $t0 is used to accumulate the sum
#                    $v0 the loop counter, counts down to zero
#############################################################
.data
prompt:          .asciiz       "\n\n   Please Input a value for N = "
result:          .asciiz        " The sum of the integers from 1 to N is "
bye:        .asciiz        "\n  **** Adios Amigo - Have a good day **** "
            .globl main
            .text
main:       li      $v0, 4              # system call code for print_str
            la      $a0, prompt         # load address of prompt into a0
            syscall                     # print the prompt message
            li      $v0, 5              # system call code for read_int
            syscall                     # reads a value of N into v0
```

```
        blez  $v0,  done  # if ( v0  < = 0 ) go to done
        li    $t0, 0      # clear $t0 to zero

loop:   add   $t0, $t0, $v0     # sum of integers in register $t0
        addi  $v0, $v0, -1      # summing  in reverse order
        bnez  $v0, loop   # branch to loop if $v0 is != zero

        li    $v0, 4            # system call code for print_str
        la    $a0, result # load address of message into $a0
        syscall                 # print the string

        li    $v0, 1            # system call code for print_int
        move  $a0, $t0          # a0 = $t0
        syscall                 # prints the value  in register $a0
        b     main

done:   li    $v0, 4            # system call code for print_str
        la    $a0, bye          # load address of msg. into $a0
        syscall                 # print the string

        li    $v0, 10           # terminate program
        syscall           # return control to  system
```