<table>
<thead>
<tr>
<th>Type of Instruction</th>
<th>MIPS Assembly Language</th>
<th>Register Transfer Language Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Access (Load and Store)</td>
<td>lw $4, Mem</td>
<td>$4← [Mem]</td>
</tr>
<tr>
<td></td>
<td>sw $4, Mem</td>
<td>Mem←$4</td>
</tr>
<tr>
<td></td>
<td>lw $4, 16($3)</td>
<td>$4← [Mem at address in $3 + 16]</td>
</tr>
<tr>
<td></td>
<td>sw $4, Mem</td>
<td>[Mem at address in $3 + 16]← $4</td>
</tr>
<tr>
<td>Move</td>
<td>move $4, $2</td>
<td>$4← $2</td>
</tr>
<tr>
<td></td>
<td>li $4, 100</td>
<td>$4← 100</td>
</tr>
<tr>
<td>Load Address</td>
<td>la $5, mem</td>
<td>$4← load address of mem</td>
</tr>
<tr>
<td>Arithmetic Instruction (reg. operands only)</td>
<td>add $4, $2, $3</td>
<td>$4← $2 + $3</td>
</tr>
<tr>
<td></td>
<td>mul $10, $12, $8</td>
<td>$10← $12 * $8 (32-bit product)</td>
</tr>
<tr>
<td></td>
<td>sub $4, $2, $3</td>
<td>$4← $2 - $3</td>
</tr>
<tr>
<td>Arithmetic with Immediates (last operand must be an integer)</td>
<td>addi $4, $2, 100</td>
<td>$4← $2 + 100</td>
</tr>
<tr>
<td></td>
<td>mul $4, $2, 100</td>
<td>$4← $2 * 100 (32-bit product)</td>
</tr>
<tr>
<td>Conditional Branch</td>
<td>bgt $4, $2, LABEL (bge, blt, ble, beq, bne)</td>
<td>Branch to LABEL if $4 &gt; $2</td>
</tr>
<tr>
<td>Unconditional Branch</td>
<td>j LABEL</td>
<td>Always Branch to LABEL</td>
</tr>
</tbody>
</table>

1. Translate the following high-level language code segment to MIPS assembly language. Use the registers indicated in the code.

   a) for $1 = 0 to 100 by steps of size 10 do
      if ($3 < $1) AND ($2 >= 50) then
         $2 = $2 + $3
      end if
   end for

   b) while ($8 > 20) do
      if ($8 < 100) OR ($8 > 200) then
         $7 = $8
         $8 = $8 - 10
      else
         $8 = $8 - R7
      end if
      $7 = $6 + 4
   end while
   sum = 0;
for i = 0 to length-1 do
    sum = sum + numbers[i]
end for

2. Write MIPS Assembly Language code for the above algorithm that sums the array's elements.

.data
numbers: .word 20, 30, 10, 40, 50, 60, 30, 25, 10, 5
length: .word 10

.text
.globl main
main:

    li $v0, 10
    syscall  # system call to exit
integer firstUnsortedIndex, testIndex, elementToInsert;
for firstUnsortedIndex = 1 to (length-1) do
    testIndex = firstUnsortedIndex-1;
    elementToInsert = numbers[firstUnsortedIndex];
    while (testIndex >=0) AND (numbers[testIndex] > elementToInsert ) do
        numbers[ testIndex + 1 ] = numbers[ testIndex ];
        testIndex = testIndex - 1;
    end while
    numbers[ testIndex + 1 ] = elementToInsert;
end for

3. Write MIPS Assembly Language code for the above insertion sort algorithm

.data
numbers:      .word 20, 30, 10, 40, 50, 60, 30, 25, 10, 5
length:       .word 10

.text
.globl main
main:

li $v0, 10 # system call to exit
syscall