Question 1. (18 points) Translate the following high-level language code segment to MIPS assembly language. Use the registers indicated in the code.

a) for $4 = 0 to 100 by steps of size 10 do
   if ($3 < $4) OR ($2 >= 50) then
      $2 = $2 + $3
   end if
end for

for:
  li $4, 0
for_compare:
  bgt $4, 100, end_for
if:
  bit $3, $4, then
  bit $2, 50, end_if
then:
  add $2, $2, $3
end_if:
  addi $4, $4, 10
  j for_compare
end_for:

b) while ($8 > 20) do
   if ($8 > 100) AND ($8 < 200) then
      $7 = $8
      $8 = $8 - 10
   else
      $8 = $8 - $7
   end if
   $7 = $6 + 4
end while

while:
  ble $8, 20, end_while
if:
  ble $8, 100, else
  bge $8, 200, else
  move $7, $8
  sub $8, $8, 10
  j end_if
else:
  sub $8, $8, 7
end_if:

  addi $7, $6, 4
  j while
end_while:
Question 2. (12 points) Suppose you have the following .data area in MIPS assembly language:

```
data
array: .word  10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
```

For each of the following assembly language segments, what value is loaded into register $t2$?

<table>
<thead>
<tr>
<th>a) li $t0, 5</th>
<th>b) la $t1, array</th>
<th>c) li $t0, 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>la $t1, array</td>
<td>lw $t2, 8($t1)</td>
<td>la $t1, array</td>
</tr>
<tr>
<td>mul $t3, $t0, 4</td>
<td>add $t1, $t1, $t3</td>
<td>sll $t3, $t0, 2  # shift logical</td>
</tr>
<tr>
<td>lw $t2, 0($t1)</td>
<td></td>
<td>add $t1, $t1, $t3</td>
</tr>
<tr>
<td>answer: 15</td>
<td></td>
<td>lw $t2, 0($t1)</td>
</tr>
</tbody>
</table>

Question 3. (8 points) For the .data area in question 2, complete the translation of the following high-level code segment to MIPS assembly language.

```assembly
def:  
for i = 0 to 14 do  
    array[i+1] = array[i]
end for
```

```assembly
for i = 0 to 14 do  
    li $t0, 0  
    li $t2, 14
    for_compare:  
        bgt $t0, $t2, end_for
        mul $t3, $t0, 4  
        add $t3, $t3, $t1  
        lw $t4, 0($t3)  
        sw $t4, 4($t3)
    end_for:  
```
Question 5. Consider the following selection sort subprogram that utilizes a function Max to search for the largest element in the unsorted part of the array.

```plaintext
procedure selectionSort(numbers - array of integers, count - integer)
    local integer variables: lastUnsortedIndex, maxIndex, temp
    for lastUnsortedIndex = (count-1) downto 1 do
        maxIndex = Max(numbers, 0, lastUnsortedIndex)
        temp = numbers[lastUnsortedIndex]
        numbers[lastUnsortedIndex] = numbers[maxIndex]
        numbers[maxIndex] = temp
    end for
end selectionSort
```

a) (6 points) Using the MIPS register conventions ($a0-$a3, $t0-$t9, $s0-$s7, $v0-$v1, $sp, $ra), what registers would be used to pass each of the following parameters to selectionSort:

<table>
<thead>
<tr>
<th>base address of &quot;numbers&quot; array</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a0</td>
<td>$a1</td>
</tr>
</tbody>
</table>

b) (6 points) Using the MIPS register conventions, which of these parameters ("numbers", "count", or both of them) should be moved into $s-registers?

- numbers should get moved into $s0, but count is just used to initialize lastUnsortedIndex

c) (6 points) Using the MIPS register conventions, what registers should be used for each of the local variables:

<table>
<thead>
<tr>
<th>lastUnsortedIndex</th>
<th>maxIndex</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s1</td>
<td>$v0 (is best, but $t0 is okay)</td>
<td>$t1</td>
</tr>
</tbody>
</table>

d) (4 points) In addition to the above registers, the value of "numbers[maxIndex]" will need to be stored into a register. Using the MIPS register conventions, what register should be used to hold this value?

- some $t register like $t2

e) (6 points) For the registers indicated above, write the MIPS instructions to set up the selectionSort call-frame on the run-time stack (i.e., move the stack pointer and save registers, etc.).

```
selectionSort:
    sub $sp, $sp, 12
    sw $ra, 4($sp)
    sw $s0, 8($sp)
    sw $s1, 12($sp)
```
f) (8 points) For the registers indicated above, write the assemble language code to call the Max function (\(\text{maxIndex} = \text{Max(numbers, 0, lastUnsortedIndex)}\)). Include the MIPS instructions to setup the parameters to Max and assigning "maxIndex" the value returned. (You do not need to write the Max function code just the code to call it)

\[
\begin{align*}
\text{move} & \quad $a0, $s0 \\
\text{li} & \quad $a1, 0 \\
\text{move} & \quad $a2, $s1 \\
\text{jal} & \quad \text{Max}
\end{align*}
\]

g) (8 points) Using the registers you indicated, write the MIPS assembly language statements to perform the statements:

\[
\begin{align*}
\text{temp} &= \text{numbers[lastUnsortedIndex]} \\
\text{numbers[lastUnsortedIndex]} &= \text{numbers[maxIndex]} \\
\text{numbers[maxIndex]} &= \text{temp}
\end{align*}
\]

\[
\begin{align*}
\text{mul} & \quad $t3, $s1, 4 \\
\text{add} & \quad $t3, $t3, $s0 \\
\text{lw} & \quad $t1, 0($t3) \\
\text{mul} & \quad $t4, $v0, 4 \\
\text{add} & \quad $t4, $t4, $s0 \\
\text{lw} & \quad $t2, 0($t4) \\
\text{sw} & \quad $t2, 0($t3) \\
\text{sw} & \quad $t1, 0($t4)
\end{align*}
\]