Binary representation:

1. What is the 5-bit binary representation of the decimal number 21?

   \[10101\]

2. What is the hexadecimal representation for decimal 219 encoded as an 8-bit binary number?

   \[1101\_1011 \rightarrow 0xDB\]

3. What is the hexadecimal representation for decimal 51 encoded as a 6-bit binary number?

   \[110011 \rightarrow 0x33\]

4. The hexadecimal representation for an 8-bit binary number is 0x9E. What is its decimal representation?

   \[158\]

5. What is the range of integers that can be represented with a single 8-bit quantity?

   \[0 - 255\]

6. Since the start of official pitching statistics in 1988, the highest number of pitches in a single game has been 172. Assuming that remains the upper bound on pitch count, how many bits would we need to record the pitch count for each game as a binary number?

   \[\text{ceil}(\log_2(172)) = 8\]
7. Compute the sum of these two 4-bit binary numbers. Express the result in hexadecimal.

\[
\begin{array}{c}
1101 \\
+0110 \\
\hline
10011
\end{array} == 0x13
\]
Assembly Language:

<table>
<thead>
<tr>
<th></th>
<th>lw rd, offset(rs1)</th>
<th>Load Word</th>
<th>reg[rd] &lt;= mem[reg[rs1] + offset]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW</td>
<td>sw rs2, offset(rs1)</td>
<td>Store Word</td>
<td>mem[reg[rs1] + offset] &lt;= reg[rs2]</td>
</tr>
<tr>
<td>ADDI</td>
<td>addi rd, rs1, constant</td>
<td>Add Immediate</td>
<td>reg[rd] &lt;= reg[rs1] + constant</td>
</tr>
<tr>
<td>BEQ</td>
<td>beq rs1, rs2, label</td>
<td>Branch if =</td>
<td>pc &lt;= (reg[rs1] == reg[rs2]) ? label : pc + 4</td>
</tr>
<tr>
<td>BNE</td>
<td>bne rs1, rs2, label</td>
<td>Branch if ≠</td>
<td>pc &lt;= (reg[rs1] != reg[rs2]) ? label : pc + 4</td>
</tr>
<tr>
<td>BLT</td>
<td>blt rs1, rs2, label</td>
<td>Branch if &lt;</td>
<td>pc &lt;= (reg[rs1] &lt; reg[rs2]) ? label : pc + 4</td>
</tr>
<tr>
<td>BGE</td>
<td>bge rs1, rs2, label</td>
<td>Branch if ≥</td>
<td>pc &lt;= (reg[rs1] &gt;= reg[rs2]) ? label : pc + 4</td>
</tr>
</tbody>
</table>

li rd, constant | Load Immediate | reg[rd] <= constant

Compile the following expressions to RISCV assembly using the instructions above. Assume a is stored at address 0x1000, b is stored at 0x1004, and c is stored at 0x1008.

1.  \( a = b + c; \)

   \[
   \begin{align*}
   &\text{li } a1, 0x1000 \quad // \text{actually lui } a1, 1 \\
   &\text{lw } a2, 4(a1) \quad // a2 = c \\
   &\text{lw } a3, 8(a1) \quad // a3 = b \\
   &\text{add } a3, a3, a2 \quad // a3 = b + c \\
   &\text{sw } a3, 0(a1) \quad // \text{store } a3 \text{ into } a
   \end{align*}
   \]

2.  if \((a > b)\) \(c = 17;\)

   \[
   \begin{align*}
   &\text{li } a1, 0x1000 \quad // \text{actually lui } a1, 1 \\
   &\text{lw } a2, 0(a1) \quad // a2 = a \\
   &\text{lw } a3, 4(a1) \quad // a3 = b \\
   &\text{bge } a3, a2, \text{end} \quad // \text{branch to end if } a \leq b \text{ (or } b \geq a) \\
   &\text{li } a4, 17 \quad // \text{actually just addi } a4, x0, 17 \\
   &\text{sw } a4, 8(a1) \quad // c = 17 \\
   &\text{end:}
   \end{align*}
   \]

3.  \( \text{sum} = 0; \)

   for \((i = 0; i < 10; i = i+1)\)

   \[
   \begin{align*}
   &\text{sum += } i; \\
   &\text{li } a1, 0 \quad // a1 = 0 \text{ (sum) or addi } a1, x0, 0, \text{ since } x0 \text{ is hardwired} \\
   &\text{to } 0 \\
   &\text{li } a2, 0 \quad // a2 = 0 \text{ (i) or addi } a2, x0, 0 \\
   &\text{li } a3, 10 \quad // a3 = 10 \quad \text{or addi } a3, x0, 10 \\
   &\text{loop:}
   \end{align*}
   \]

   \[
   \begin{align*}
   &\text{add } a1, a1, a2 \quad // a1 = a1 + a2 \text{ or sum = sum } + i \\
   &\text{addi } a2, a2, 1 \quad // i = i+1 \\
   &\text{blt } a2, a3, \text{loop} \quad // \text{if } i < 10, \text{ branch to beginning of loop body}
   \end{align*}
   \]