Convert the following unsigned numbers as specified. For the conversions to binary, you can use either successive subtraction or successive division.

i. \((1001110111)_2\) to decimal

\[
1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 127 + 0 + 0 + 16 + 0 + 4 + 2 + 1 = 160
\]

ii. \((10011110110101)_2\) to hex

\[
(10011110110101)_2 = (9F6D)_{16}
\]

iii. \((631)_{10}\) to binary

\[
2^8 = 256 < 631 \leq 2^9 = 512 \\
\frac{631}{512} = 1 \text{ remainder } 119 \\
\frac{119}{64} = 1 \text{ remainder } 55 \\
\frac{55}{32} = 1 \text{ remainder } 23 \\
\frac{23}{16} = 1 \text{ remainder } 7 \\
\frac{7}{4} = 1 \text{ remainder } 3 \\
\frac{3}{2} = 1 \text{ remainder } 1 \\
\frac{1}{1} = 1 \text{ remainder } 0
\]

\[
(631)_{10} = (10011110111)_2
\]

iv. \((631)_{10}\) to hex

\[
2 \times 2^9 = 256 < 631 \leq 2 \times 2^{10} = 512 \\
\frac{631}{512} = 1 \text{ remainder } 119 \\
\frac{119}{64} = 1 \text{ remainder } 55 \\
\frac{55}{32} = 1 \text{ remainder } 23 \\
\frac{23}{16} = 1 \text{ remainder } 7 \\
\frac{7}{4} = 1 \text{ remainder } 3 \\
\frac{3}{2} = 1 \text{ remainder } 1 \\
\frac{1}{1} = 1 \text{ remainder } 0
\]

\[
(631)_{10} = (277)_{16}
\]
Perform the specified operation assuming the signed numbers are (a) sign magnitude and (b) twos complement. Identify if overflow occurs. For the case of subtraction, first take the negation of the number, then perform addition.

i. \((10111) - (11100)\)

ii. \((10101) + (01011)\)

iii. \((011011) + (000111)\)

iv. \((001110) - (011011)\)

\[
\begin{array}{c}
\text{S-M} \\
\hline
10111 + 0100 \\
\text{Different sign, smaller bigger} \\
01112 \times 00000 \\
\hline
001012 \\
\end{array}
\]

\[
\begin{array}{c}
\text{2's comp} \\
10111 + (00011 +1) \\
\text{No overflow} \\
10111 + 00100 \\
\hline
110112 \\
\end{array}
\]

\[
\begin{array}{c}
\text{Different sign, smaller mag} \\
10101 + 01011 \\
\text{Different signs, smaller mag} \\
01110 \\
\hline
001102 \\
\end{array}
\]

\[
\begin{array}{c}
\text{No overflow} \\
10111 + (000111 + 1) \\
0010111 + 010111 \\
\hline
000002 \\
\end{array}
\]

\[
\begin{array}{c}
\text{Same sign, different mag} \\
011011 + 000111 \\
\text{Overflow!} \\
000010_2 \\
\hline
\end{array}
\]

\[
\begin{array}{c}
\text{Different sign, smaller mag} \\
001110 + 110111 \\
\text{Different signs, larger mag} \\
011101 \\
\hline
101112 \\
\end{array}
\]

\[
\begin{array}{c}
101110 + (100100 +1) \\
\text{No overflow} \\
001110 + 100101 \\
\hline
110112 \\
\end{array}
\]
Convert the following decimal numbers to 8-bit binary (a) sign magnitude and (b) twos complement numbers. You must show all of your work in deriving your answer.

i. \(-34\)

\[ (-34) = -(32 + 2) = -(100010) = -(00100010) \]

**Sign Magnitude:**

\[ 10100010_2 \]

**2's Complement:**

\[ 11011110 + 1 = 11011111_2 \]

ii. \(-45\)

\[ (-45) = -(32 + 8 + 4 + 1) = -(101101) = -(00101101) \]

**Sign Magnitude:**

\[ 10101101_2 \]

**2's Complement:**

\[ 11010010 + 1 = 11010011_2 \]

iii. \(+23\)

\[ (+23) = +(16 + 4 + 2 + 1) = +(10111) = +(00101111) \]

**Sign Magnitude:**

\[ 00010111_2 \]

**2's Complement:**

\[ 00101111_2 \]

iv. \(-62\)

\[ (-62) = -(32 + 16 + 8 + 4 + 2) = -(111110) = -(0011110) \]

**Sign Magnitude:**

\[ 10111110_2 \]

**2's Complement:**

\[ 11000010 + 1 = 11000011_2 \]
Design a circuit that can tell whether a 4-bit 2's complement number \((A_3A_2A_1A_0)\) is greater than 0.

\[
A_3A_2A_1A_0 > 0 \quad \text{if} \quad A_3 = 0 \quad \text{and} \quad A_3A_2A_1A_0 \neq 0
\]

\[
\overline{A_3} \neq (A_2 + A_1 + A_0)
\]

![Circuit Diagram](image)
Design a circuit that can tell whether a 4-bit sign-magnitude number ($A_3A_2A_1A_0$) is greater than or equal to 0. Note: if you think this is just a wire connected to $A_3$ you're missing something!

<table>
<thead>
<tr>
<th>$A_3A_2A_1A_0$</th>
<th>Val</th>
<th>$GT_EQ_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>+0</td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>+2</td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>+3</td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td>+4</td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td>+5</td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td>+6</td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>+7</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>-0</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>-2</td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>-4</td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>-5</td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>-6</td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>-7</td>
<td></td>
</tr>
</tbody>
</table>

$G = \overline{A_3} + \overline{A_2} \overline{A_1} \overline{A_0}$
Design a circuit that can determine whether a 4-bit 2's complement number \((A_3A_2A_1A_0)\) is larger than another 4-bit 2's complement number \((B_3B_2B_1B_0)\). You may use premade gates such as full adders, basic gates like AND, OR, NOT. If you use full adders, be sure to make sure your circuit works even if an overflow occurs.

Hint: If \(A>B\), do you know anything about \(A-B\) or \(B-A\)?

Lots of approaches, but if \(A>B\) then \(0>A-B\).

But overflow: 2 fixes:
1. On overflow sign bit is flipped
2. Convert 4-bit \(2's\) comp, which cannot overflow

I do #2 here.

Compute: is \(0>B_3B_2B_1B_0-A_3A_2A_1A_0\)

\(0>B_3B_2B_1B_0+(A_3A_2A_1A_0+1)\)

Can check if result \(<0\) just by checking sign bit.

\[ \begin{array}{c}
A_3 \\
B_3 \\
B_2 \\
B_1 \\
B_0 \\
A_0 \\
\end{array} \]

\[ \begin{array}{c}
+ \\
+ \\
+ \\
+ \\
1 \\
\end{array} \]

Larger