Question: Compute the result of the operation $A + B$ with $A = +5$ and $B = -15$, representing all values in two's complement over 5 bits, specifying whether the operation results in an overflow or not.

Solution:
$A = +5_{10}$ is positive, thus we can convert it to binary directly. Using 5 bits: $+5 = 00101_2 \implies A = 00101_{2C}$

$B = -15$ is negative, thus we start converting its absolute value to binary, and then we compute the two's complement. We have $+15 = 01111_2$. The two's complement corresponds to $10000 + 1 = 10001$, thus $B = 10001_{2C}$

To compute the sum, we simply use binary addition:

$$\begin{array}{ll} 00101 & + \\ 10001 & = \\ \hline 10110 & \end{array}$$

Since the operands have different sign, there is no overflow.

We can verify that $C = 10110_{2C}$ corresponds to -10. Taking the two's complement of C, we have $01001 + 1 = 01010$, and $01010_2 = +10_{10}$, thus $C = -10$

Question: Perform the following operations in two's complement over 5 bits. Specify whether the operations generate an overflow.

1. 11011 + 01111
2. 10111 - 10011

Solution:
To compute the sum, we simply use binary addition.

$$\begin{array}{ll} 11011 & + \\ \underline{01111} & = \\ 01010 & \end{array}$$

Since the operands have different sign, there is no overflow.

For the subtraction, we add the first term to the two's complement of the second term. The two's complement of 10011 is $01100 + 1 = 01101$. The result is

$$\begin{array}{ll} 10111 & + \\ \underline{01101} & = \\ 00100 & \end{array}$$

Since the operands have different sign, there is no overflow.

Question: Perform the following operations in Sign and Magnitude over 5 bits. Specify whether the operations generate an overflow.

1. 11011 - 01111
2. 00101 + 00011

Solution:
The first operation is the difference of a negative and a positive number. We convert it to a sum by taking the negative of the second value: $11011 - 01111 = 11011 + 11111$. We can then compute the sum of the respective absolute values (on 4 bits) and add a negative sign bit to the result. The absolute values are pure numbers on 4 bits. Their sum is

$$\begin{array}{rl} 1011 & + \\ 1111 & = \\ \hline 11010 & \end{array}$$

Since the result cannot be represented on 4 bits (we have a carry), the operation results in an overflow. Notice that absolute value of the number (i.e. the result of the sum) must be representable on 4 bits, since the 5-th bit would then be required to represent the sign (1 in this case).
The result on 5 bits, including the sign bit (1) would be 11110 (sign bit followed by the 4 least significant bits of the result), but does not correspond to the difference of the two original values because we are in presence of overflow.

The second operation is simply the sum of two positive numbers. Again, we sum their absolute values (on 4 bits), we check for overflow and then we add the sign bit (0)

$$\begin{array}{rl} 0101 & + \\ 0011 & = \\ \hline 1000 & \end{array}$$

In this case there is no overflow. Adding the sign bit (0), we obtain the result: 01000

Question: Given the number 34_7 in base 7, write its representation in base 2 (binary), base 8 and base 16

Solution:
We first convert the value to base 10: $34_7 = 3 \times 7 + 4 = 25_{10}$
We now convert the value to binary:

$$\begin{gathered}
25\%2 = \mathbf{1};\ 25//2 = 12 \\
12\%2 = \mathbf{0};\ 12//2 = 6 \\
6\%2 = \mathbf{0};\ 6//2 = 3 \\
3\%2 = \mathbf{1};\ 3//2 = 1 \\
1\%2 = \mathbf{1};\ 1//2 = 0
\end{gathered}$$

Thus $34_7 = 11001_2$

To convert to base 8:

$$\begin{gathered}
25\%8 = \mathbf{1};\ 25//8 = 3 \\
3\%8 = \mathbf{3};\ 3//8 = 0
\end{gathered}$$

Thus $34_7 = 31_8$

To convert to base 16:

$$\begin{gathered}
25\%16 = \mathbf{9};\ 25//16 = 1 \\
1\%16 = \mathbf{1};\ 1//16 = 0
\end{gathered}$$

Thus $34_7 = 19_{16}$

Question: Given the number 31_{10} in base 10, write its representation in base 2 (binary) and base 16

Solution:
We convert the value to binary:

$$\begin{gathered}31\%2 = \mathbf{1};\ 31//2 = 15\\ 15\%2 = \mathbf{1};\ 15//2 = 7\\ 7\%2 = \mathbf{1};\ 7//2 = 3\\ 3\%2 = \mathbf{1};\ 3//2 = 1\\ 1\%2 = \mathbf{1};\ 1//2 = 0\end{gathered}$$

Thus $31_{10} = 11111_2$

To convert to base 16:

$$\begin{gathered}31\%16 = \mathbf{15};\ 31//16 = 1\\ 1\%16 = \mathbf{1};\ 1//16 = 0\end{gathered}$$

The value 15 must be encoded with the corresponding base-16 digit, i.e. F. Thus $34_7 = 1F_{16}$

Question: Convert the following values to two's complement and sign & magnitude over 6 bits:

- 6_{10}
- -8_{10}
- 17_{10}
- -23_{10}
- 32_{10}
- -32_{10}

Solutions:

- 000110_{2C}; $000110_{S\&M}$
- 111000_{2C}; $101000_{S\&M}$
- 010001_{2C}; $010001_{S\&M}$
- 101001_{2C}; $110111_{S\&M}$
- Overflow – cannot represent 32_{10} in two's complement over 6 bits; Overflow – cannot represent 32_{10} in S&M over 6 bits
- 100000_{2C}; Overflow – cannot represent -32_{10} in S&M over 6 bits

Question: Explain in brief the different phases of the execution of an instruction in a microprocessor

Question: Explain the differences between a volatile and non-volatile memory. Classify the different memories of a computer according to this property.

Question: Which, among the following, is a correctly declared tuple?

1. `bob = [10, "30", 4.25, 'bye']`
2. `bob = 10, "30", 4.25, 'bye'`
3. `bob == (10, "30", 4.25, 'bye')`
4. `bob = {10, "30", 4.25, 'bye'}`

Question: Explain the differences in the way data is accessed in lists and dictionaries, respectively

Question: What is the role of the ALU?

Question: What is the difference between function arguments and return values? How many arguments can be passed to a function? How many values can be returned by a function?

Question: Explain the mechanism for passing parameters to functions in Python

Question: Explain briefly the similarities and differences between sets and lists in Python

Question: Explain the role and characteristics of a bus.

Question: Given a bus composed of a 18-bit address bus and a 4 byte data bus, what is the maximum size of a memory that can be accessed by the bus?