

The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM), 2026

Volume 42, Pages 136-151

ICBASET 2026: International Conference on Basic Sciences, Engineering and Technology

Why Two's Complement Became the Standard: A Comparative Analysis of Unsigned, Sign-Magnitude, One's Complement, and Two's Complement Representations

Bojken Shehu

Polytechnic University of Tirana

Abstract: Binary number representation is a fundamental concept in computer architecture that directly influences arithmetic operations, data representation, processor design, and hardware implementation. Although several methods have been developed for representing signed integers, modern computer systems predominantly employ Two's Complement representation. This paper presents a comparative analysis of four binary number representation methods: Unsigned, Sign-Magnitude, One's Complement, and Two's Complement. The analysis examines their numerical ranges, number of distinct representable values, zero representation, binary encoding, bit-width expansion procedures, sign extension techniques, overflow and underflow behavior, and suitability for hardware arithmetic. A unified comparative framework, supported by mathematical formulas, illustrative examples, conversion cases, and comparative tables, is used to systematically examine the similarities, differences, and architectural trade-offs among the four representations. The results show that the four representations differ in how numerical values are encoded and how numerical meaning is preserved during bit-width expansion. Unsigned representation relies on zero padding, while Sign-Magnitude preserves the sign bit and inserts additional zeros between the sign bit and the magnitude bits. In contrast, One's Complement and Two's Complement employ classical sign extension by replicating the most significant bit. The comparison further demonstrates that Sign-Magnitude and One's Complement suffer from duplicate zero representations and additional arithmetic complexity. Two's Complement, however, provides a unique representation of zero, fully utilizes all available binary combinations, supports efficient sign extension, and enables addition and subtraction using simplified arithmetic circuitry. The analysis also highlights numerical limitations and overflow characteristics associated with fixed-width representations. These technical advantages explain why Two's Complement has become the standard representation for signed integers in modern computer systems. The proposed framework provides a structured reference for understanding design trade-offs and practical implications in computer architecture and digital systems.

Keywords: Binary number representation, Unsigned representation, Sign-magnitude, One's complement, Two's complement

Introduction

Binary number representation is a fundamental concept in digital computer systems and computer architecture. Since digital hardware operates using only two voltage levels, all numerical values must be represented in binary form. The choice of a binary number representation directly influences arithmetic operations, processor design, memory organization, compiler implementation, instruction execution, and the overall efficiency of computer systems. Consequently, understanding how computers represent both positive and negative integers is essential in computer architecture and digital systems (Stallings, 2022; Patterson & Hennessy, 2020; Mano & Ciletti, 2018). Representing unsigned integers in binary is straightforward because every bit contributes directly to the numerical value. However, representing signed integers introduces additional challenges, as the system must distinguish between positive and negative values while maintaining efficient arithmetic operations. Throughout the history of

- This is an Open Access article distributed under the terms of the Creative Commons Attribution-Noncommercial 4.0 Unported License, permitting all non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

- Selection and peer-review under responsibility of the Organizing Committee of the Conference

© 2026 Published by ISRES Publishing: www.isres.org

computer architecture, several techniques have been developed to address this problem, including Sign-Magnitude, One's Complement, and Two's Complement representations. Although these methods represent signed integers using the same fixed bit width, they differ significantly in terms of numerical range, zero representation, arithmetic complexity, bit-width expansion, sign extension, and hardware implementation (Stallings, 2022; Mano & Ciletti, 2018; Parhami, 2010).

Today, virtually all modern computer systems use Two's Complement for signed integer representation (Patterson & Hennessy, 2020; Intel Corporation, n.d.; RISC-V International, n.d.). This widespread adoption is primarily motivated by its ability to eliminate duplicate zero representations, fully utilize all available binary combinations, simplify arithmetic circuits, and support efficient sign extension (Stallings, 2022; Parhami, 2010; Bastien, 2018). Despite the widespread use of Two's Complement in modern computer systems, a unified comparison of the four principal binary number representation methods is rarely presented within a single analytical framework. Existing literature in computer architecture, digital design, and computer arithmetic commonly discusses these representation methods in separate contexts, which can make their relationships, implementation differences, and practical implications less apparent when considered as a unified system. This paper addresses this gap by providing a structured comparative analysis that examines all four representation methods using identical evaluation criteria.

This paper presents a comprehensive comparative analysis of four binary number representation methods: Unsigned, Sign-Magnitude, One's Complement, and Two's Complement. The comparison is performed using a unified framework that examines their numerical ranges, maximum number of representable values, zero representation, bit-width expansion mechanisms, sign extension techniques, and suitability for hardware arithmetic. To improve clarity, the paper also provides comparative tables and illustrative examples that highlight the similarities and differences among the four representations.

The main contributions of this paper are as follows:

- A unified comparative framework for analyzing Unsigned, Sign-Magnitude, One's Complement, and Two's Complement representations.
- A systematic comparison of numerical range, zero representation, bit-width expansion, sign extension, and hardware arithmetic characteristics.
- Comparative tables and illustrative binary examples that highlight the similarities and differences among the four representations.
- A technical explanation of why Two's Complement has become the standard representation for signed integers in modern computer systems.

To guide the analysis, the following research questions are addressed:

RQ1: What are the fundamental differences among Unsigned, Sign-Magnitude, One's Complement, and Two's Complement representations?

RQ2: How do these representations differ in terms of numerical range, zero representation, bit-width expansion, sign extension, and hardware arithmetic?

RQ3: Why has Two's Complement become the standard representation for signed integers in modern computer systems?

The contribution of this study is not the introduction of a new binary representation method, but the systematic comparison of the four principal representations using a unified set of technical criteria, including numerical range, zero representation, bit-width expansion, sign extension, overflow behavior, and hardware arithmetic. The remainder of this paper is organized as follows. Section II introduces the fundamental concepts of binary number representation and discusses the four representation methods. Section III presents a comparative analysis supported by comparative tables and illustrative examples. Section IV discusses the findings of the comparative analysis and explains why Two's Complement has become the standard representation for signed integers in modern computer systems. Finally, Section V concludes the paper by summarizing the main findings and outlining directions for future work.

Binary Number Representation Fundamentals

Binary numbers can be represented using different encoding methods depending on whether only non-negative integers or both positive and negative integers need to be represented. Each representation has distinct

characteristics regarding numerical range, zero representation, arithmetic operations, and bit-width expansion. Figure 1 provides an overview of the four binary number representation methods analyzed in this paper and illustrates their fundamental relationships (Stallings, 2022; Patterson & Hennessy, 2020; Mano & Ciletti, 2018; Tanenbaum & Austin, 2013).

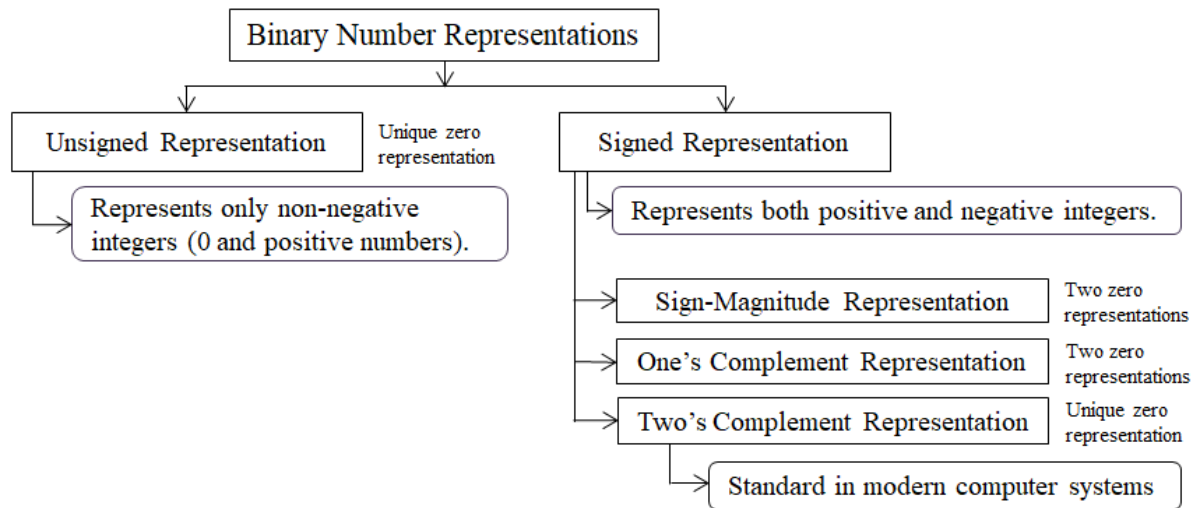


Figure 1. Classification of binary number representations methods

As illustrated in Figure 1, binary number representations can be broadly classified into unsigned and signed representations. Unsigned representation is suitable only for non-negative integers, whereas signed representations support both positive and negative values. Among the signed methods, Sign-Magnitude and One's Complement provide two representations of zero, while Two's Complement has a unique representation of zero and is therefore adopted as the standard representation in modern computer systems (Stallings, 2022; Patterson & Hennessy, 2020; Parhami, 2010; Intel Corporation, n.d.; RISC-V International, n.d.). The following subsections examine each representation in detail, including its numerical range, binary encoding, bit-width expansion, and main advantages and disadvantages.

Unsigned Representation

The Unsigned Representation is the simplest method for representing binary integers. In this representation, all bits are used to represent the magnitude of the number, and no bit is reserved to indicate the sign. Consequently, only non-negative integers (zero and positive values) can be represented (Stallings, 2022; Mano & Ciletti, 2018). The numerical range of an n -bit unsigned representation is :

$$0 \leq x \leq 2^n - 1 ,$$

providing a total of 2^n distinct numerical values. Since no sign bit exists, there is only one representation of zero, and all possible binary combinations correspond to unique decimal values. Consequently, any arithmetic result greater than $2^n - 1$ cannot be represented within the given bit width, resulting in positive overflow. Similarly, a negative result falls outside the representable range of the unsigned representation. Table 1 summarizes the numerical range for both the 3-bit example and the general n -bit case.

Table 1. Numerical range of unsigned representation		
Representation	Minimum Value	Maximum Value
Unsigned (3-bit)	0	7
n -bit	0	$2^n - 1$

Table 2 presents the complete 3-bit unsigned representation, illustrating all eight binary combinations and their corresponding decimal values from 0 to 7. Because every bit contributes directly to the numerical value, the Unsigned representation provides the largest range of non-negative integers among the four methods for the same bit width (Mano & Ciletti, 2018).

Table 2. Three-bit unsigned representation

Binary (3 Bits)	Decimal Value (Unsigned)
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

When the bit width is increased, unsigned numbers are expanded by inserting leading zeros to the left of the most significant bit. Unlike signed representations, unsigned numbers do not require sign extension because no sign bit exists. Examples of bit-width expansion are presented in Table 3 (Stallings, 2022; Hamacher et al., 2012).

Table 3. Examples of bit-width expansion in unsigned representation

Number	Original Unsigned Representation	Extended 8-Bit Unsigned Representation
1 ₍₁₀₎	0001 ₍₂₎ (n = 4 bits)	00000001 ₍₂₎
3 ₍₁₀₎	00011 ₍₂₎ (n = 5 bits)	00000011 ₍₂₎
5 ₍₁₀₎	0000101 ₍₂₎ (n = 7 bits)	00000101 ₍₂₎
0 ₍₁₀₎	00 ₍₂₎ (n = 2 bits)	00000000 ₍₂₎

Finally, the principal advantages and disadvantages of the unsigned representation are summarized in Table 4 (Patterson & Hennessy, 2020; Stallings, 2022).

Table 4. Advantages and disadvantages of unsigned representation

Aspect	Description
Advantages	Utilizes all 2^n available binary combinations to represent distinct numerical values.
	Provides a unique representation of zero, eliminating ambiguity.
	Simple and intuitive representation, making binary-to-decimal conversion straightforward.
	Enables efficient arithmetic operations for non-negative integers.
	Requires no sign bit, allowing the maximum possible positive range for a given bit-width.
Disadvantages	Cannot represent negative numbers.
	Unsuitable for applications requiring signed arithmetic.
	Arithmetic underflow may occur when subtracting a larger value from a smaller one. Additional mechanisms are required when both signed and unsigned values must be processed together.

Sign-Magnitude Representation

Sign-Magnitude Representation is one of the earliest methods developed for representing signed integers in binary form. In this representation, the most significant bit (MSB) is reserved as the sign bit, while the remaining $n-1$ bits represent the magnitude (absolute value) of the number. A sign bit of 0 indicates a positive number, whereas a sign bit of 1 indicates a negative number (Mano & Ciletti, 2018; Stallings, 2022).

The numerical range of an n -bit Sign-Magnitude representation is :

$$-(2^{n-1} - 1) \leq x \leq +(2^{n-1} - 1),$$

providing a total of $2^n - 1$ distinct numerical values because zero has two separate representations (+0 and -0). Therefore, arithmetic results exceeding $+(2^{n-1} - 1)$ or falling below $-(2^{n-1} - 1)$ cannot be represented within the given bit width, resulting in a range overflow condition. Table 5 summarizes the numerical range for both the 3-bit example and the general n -bit case (Mano & Ciletti, 2018; Stallings, 2022).

Table 5. Numerical range of sign-magnitude representation

Representation	Minimum Value	Maximum Value
Sign-Magnitude (3-bit)	-3	+3
n -bit	$-(2^{n-1} - 1)$	$+(2^{n-1} - 1)$

Table 6 presents the complete 3-bit Sign-Magnitude representation, illustrating the binary encoding of all representable values from -3 to +3. Unlike the Unsigned representation, Sign-Magnitude supports both positive and negative integers by reserving the most significant bit as the sign bit (Mano & Ciletti, 2018).

Table 6. Three-bit sign-magnitude representation	
Binary number(3 Bits)	Decimal Value (Sign-Magnitude)
000	+0
001	+1
010	+2
011	+3
100	-0
101	-1
110	-2
111	-3

When the bit width is increased, the sign bit remains unchanged, while additional bits are inserted between the sign bit and the magnitude bits. This differs from One's Complement and Two's Complement representations, which use classical sign extension by replicating the sign bit. Examples of bit-width expansion are presented in Table 7 (Hamacher et al., 2012; Stallings, 2022).

Table 7. Examples of bit-width expansion in sign-magnitude representation		
Number	Original Sign Magnitude Representation	Extended 8-Bit Sign-Magnitude Representation
-5 ₍₁₀₎	1101 ₍₂₎ (n=4 bit)	10000101 ₍₂₎
-8 ₍₁₀₎	11000 ₍₂₎ (n=5 bit)	10001000 ₍₂₎
-37 ₍₁₀₎	1100101 ₍₂₎ (n=7 bit)	10100101 ₍₂₎
-0 ₍₁₀₎	10 ₍₂₎ (n=2 bit)	10000000 ₍₂₎
+0 ₍₁₀₎	00 ₍₂₎ (n=2 bit)	00000000 ₍₂₎
+42 ₍₁₀₎	0101010 ₍₂₎ (n= 7 bit)	00101010 ₍₂₎

Finally, the main advantages and disadvantages of the Sign-Magnitude representation are summarized in Table 8 (Parhami, 2010; Stallings, 2022).

Table 8. Advantages and disadvantages of sign-magnitude representation	
Aspect	Description
Advantages	Represents both positive and negative numbers using an intuitive sign-and-magnitude format.
	Easy to understand because the sign and magnitude are stored separately, resembling the decimal representation of signed numbers.
	Positive and negative numbers have the same magnitude representation, differing only in the sign bit.
	Has two representations of zero (+0 and -0), leading to ambiguity.
Disadvantages	Equality checking requires special handling because both +0 and -0 must be recognized as zero.
	Addition and subtraction require separate processing depending on the signs of the operands, increasing arithmetic complexity.
	Requires more complex hardware implementation than Two's Complement representation.
	Rarely used for signed integer representation in modern computer systems.

One's Complement Representation

One's Complement Representation is a signed binary representation in which positive numbers are represented using standard binary notation, while negative numbers are obtained by inverting every bit of the corresponding positive number. In other words, every bit is inverted: each 0 becomes 1, and each 1 becomes 0 (Parhami, 2010; Stallings, 2022).

The numerical range of an n -bit One's Complement representation is:

$$-(2^{n-1} - 1) \leq x \leq (2^{n-1} - 1),$$

providing a total of $2^n - 1$ distinct numerical values. Similar to Sign-Magnitude representation, One's Complement contains two representations of zero: $+0$, represented by all zeros, and -0 , represented by all ones. Therefore, arithmetic results exceeding $+(2^{n-1} - 1)$ or falling below $-(2^{n-1} - 1)$ cannot be represented within the given bit width. Table 9 summarizes the numerical range for both the 3-bit example and the general n -bit case (Parhami, 2010; Stallings, 2022).

Table 9. Numerical range of one's complement representation

Representation	Minimum Value	Maximum Value
One's Complement (3-bit)	-3	+3
n -bit	$-(2^{n-1} - 1)$	$+(2^{n-1} - 1)$

Table 10 presents the complete 3-bit One's Complement representation, illustrating the binary encoding of all representable values. Unlike Sign-Magnitude representation, negative numbers are generated by complementing every bit of the corresponding positive value rather than storing the sign separately (Parhami, 2010).

Table 10. Three-bit one's complement representation

Binary Number (3 Bits)	Decimal Value (One's Complement)
000	0
001	1
010	2
011	3
100	-3
101	-2
110	-1
111	-0

When the bit width is increased, One's Complement uses classical sign extension by replicating the most significant bit (MSB). If the MSB is 0, additional leading zeros are inserted; if the MSB is 1, additional leading ones are inserted. This process preserves the numerical value of the represented integer. Examples of bit-width expansion are presented in Table 11 (Hamacher et al., 2012; Parhami, 2010).

Table 11. Examples of sign extension in one's complement representation

Number	One's Complement Representation	8-Bit Representation after Sign Extension
$-5_{(10)}$	$1010_{(2)}$ ($n = 4$ bits)	$11111010_{(2)}$
$-8_{(10)}$	$10111_{(2)}$ ($n = 5$ bits)	$11110111_{(2)}$
$-37_{(10)}$	$1011010_{(2)}$ ($n = 7$ bits)	$11011010_{(2)}$
$-0_{(10)}$	$11_{(2)}$ ($n = 2$ bits)	$11111111_{(2)}$
$+0_{(10)}$	$00_{(2)}$ ($n = 2$ bits)	$00000000_{(2)}$
$+42_{(10)}$	$0101010_{(2)}$ ($n = 7$ bits)	$00101010_{(2)}$

Finally, the principal advantages and disadvantages of the One's Complement representation are summarized in Table 12 (Parhami, 2010; Patterson & Hennessy, 2020).

Table 12. Advantages and disadvantages of one's complement representation

Aspect	Description
Advantages	Negative numbers can be generated by inverting all bits of the corresponding positive number.
	Sign extension is performed by replicating the most significant bit (MSB).
Disadvantages	Two representations of zero ($+0$ and -0) complicate comparisons and equality checking.
	Addition requires an end-around carry, increasing arithmetic and hardware complexity. Less efficient hardware implementation than Two's Complement representation.

Two's Complement Representation

Two's Complement Representation is the most widely used method for representing signed integers in modern computer systems. Positive numbers are represented using standard binary notation, whereas negative numbers are obtained by taking the One's Complement of the corresponding positive value and adding 1 to the resulting bit pattern (Parhami, 2010; Patterson & Hennessy, 2020; Stallings, 2022).

The numerical range of an n -bit Two's Complement representation is:

$$-2^{n-1} \leq x \leq (2^{n-1} - 1),$$

providing a total of 2^n distinct numerical values. Unlike Sign-Magnitude and One's Complement representations, Two's Complement has only one representation of zero, eliminating the ambiguity associated with positive and negative zero. Therefore, arithmetic results greater than $+(2^{n-1} - 1)$ or less than -2^{n-1} cannot be represented within the given bit width. Table 13 summarizes the numerical range for both the 3-bit example and the general n -bit case (Parhami, 2010; Stallings, 2022).

Table 13. Numerical range of two's complement representation

Representation	Minimum Value	Maximum Value
Two's Complement (3-bit)	-4	+3
n -bit	-2^{n-1}	$+(2^{n-1} - 1)$

Table 14 presents the complete 3-bit Two's Complement representation, illustrating the binary encoding of all representable values. The minimum value is represented by 100, while the maximum value is 011, resulting in an asymmetric numerical range with one additional negative value (Parhami, 2010).

Table 14. Three-bit two's complement representation

Binary Number (3 Bits)	Decimal Value (Two's Complement)
000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2
111	-1

When the bit width is increased, Two's Complement uses classical sign extension by replicating the most significant bit (MSB). If the MSB is 0, leading zeros are inserted; if the MSB is 1, leading ones are inserted. This operation preserves the numerical value of both positive and negative integers. Examples of bit-width expansion are presented in Table 15 (Hamacher et al., 2012; Parhami, 2010).

Table 15. Examples of sign extension in two's complement representation

Number	Two's Complement Representation	8-Bit Representation after Sign Extension
$-5_{(10)}$	$1011_{(2)}$ ($n = 4$ bits)	$11111011_{(2)}$
$-8_{(10)}$	$11000_{(2)}$ ($n = 5$ bits)	$11111000_{(2)}$
$-37_{(10)}$	$1011011_{(2)}$ ($n = 7$ bits)	$11011011_{(2)}$
$0_{(10)}$	$00_{(2)}$ ($n = 2$ bits)	$00000000_{(2)}$
$+42_{(10)}$	$0101010_{(2)}$ ($n = 7$ bits)	$00101010_{(2)}$

Table 16. Advantages and disadvantages of two's complement representation

Aspect	Description
Advantages	Provides a unique representation of zero.
	Simplifies arithmetic operations by eliminating the end-around carry required in One's Complement representation.
	The standard binary adder can be used for both addition and subtraction.
	Simple and efficient sign extension by replicating the most significant bit (MSB).
	Supports efficient hardware implementation, making it the standard representation for signed integers in modern computer systems.
Disadvantages	Negating the minimum representable value results in overflow.
	Less intuitive for beginners than Sign-Magnitude because the stored bit pattern does not directly represent the numerical magnitude.
	The asymmetric numerical range provides one additional negative value (e.g., -128 to +127 in 8-bit representation). Requires understanding of complement-based representation rather than direct sign-and-magnitude notation.

Finally, the principal advantages and disadvantages of the Two's Complement representation are summarized in Table 16. The combination of a unique zero representation, efficient hardware arithmetic, and straightforward sign extension has made Two's Complement the standard representation for signed integers in virtually all modern processors (Intel Corporation, n.d.; Patterson & Hennessy, 2020; RISC-V International, n.d.; ARM Ltd., n.d.).

Comparative Analysis

This section presents a comparative analysis of the four binary number representation methods discussed in the previous section. The comparison focuses on their numerical ranges, binary encoding, bit-width expansion mechanisms, conversion processes, and overall characteristics. Through a series of comparative tables, the similarities and differences among Unsigned, Sign-Magnitude, One's Complement, and Two's Complement representations are systematically analyzed.

Comparison of Numerical Ranges, Overflow, and Underflow

The numerical range is one of the most important characteristics of a binary number representation method, as it determines the minimum and maximum values that can be represented using a fixed number of bits. Although all four representations use the same bit width, they differ in the allocation of binary patterns, resulting in different numerical ranges and numbers of representable values. These numerical limits also determine the boundaries at which overflow or underflow may occur during arithmetic operations. When an arithmetic result exceeds the maximum or falls below the minimum value representable within a fixed bit width, the result cannot be represented correctly. Therefore, comparing numerical ranges also provides a basis for analyzing the overflow and underflow behavior of the four representation methods. Table 17 summarizes the numerical range formulas for Unsigned, Sign-Magnitude, One's Complement, and Two's Complement representations for an arbitrary n -bit word.

Table 17. Numerical range formulas for different number representation methods

Representation	Minimum Value	Maximum Value	Range
Unsigned	0	$2^n - 1$	0 to $2^n - 1$
Sign-Magnitude	$-(2^{n-1} - 1)$	$+(2^{n-1} - 1)$	$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$
One's Complement	$-(2^{n-1} - 1)$	$+(2^{n-1} - 1)$	$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$
Two's Complement	-2^{n-1}	$+(2^{n-1} - 1)$	-2^{n-1} to $+(2^{n-1} - 1)$

As shown in Table 17, the Unsigned representation allocates all available bit patterns to non-negative integers, resulting in a range from 0 to $2^n - 1$. In contrast, Sign-Magnitude and One's Complement reserve one bit for the sign, reducing the number of distinct representable values because both methods include separate representations for $+0$ and -0 . Consequently, their numerical range extends from $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$. Two's Complement differs by eliminating the duplicate representation of zero, thereby utilizing all 2^n binary combinations and extending the minimum value to -2^{n-1} , while the maximum value remains $+(2^{n-1} - 1)$.

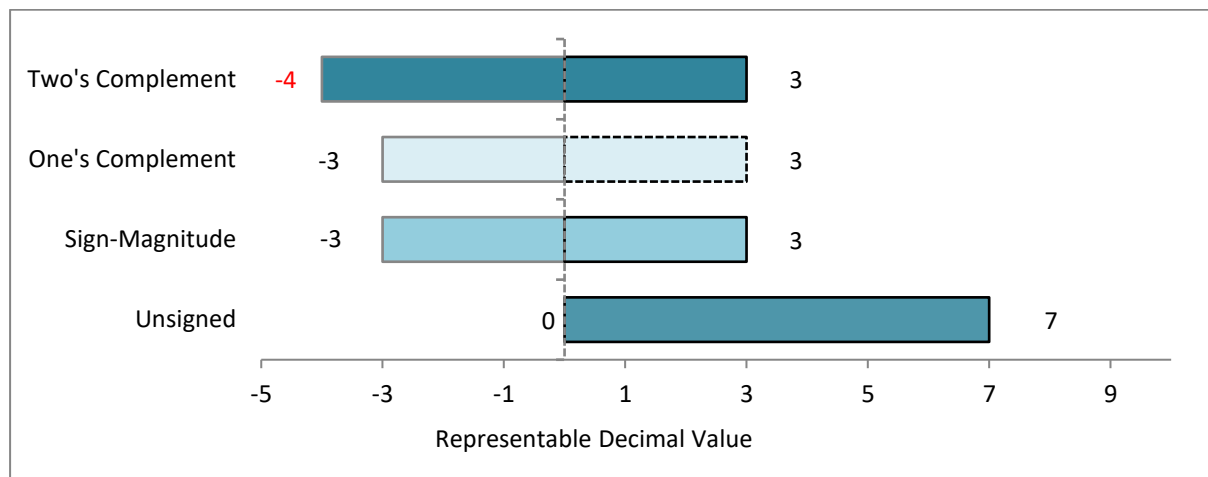


Figure 2. Numerical range comparison of 3-bit binary number representations

The differences in numerical ranges can also be visualized using a 3-bit representation. Figure 2 provides a graphical comparison of the representable decimal values for Unsigned, Sign-Magnitude, One's Complement, and Two's Complement representations. As illustrated in Figure 2, Unsigned representation provides the widest positive range, from 0 to 7, while Sign-Magnitude and One's Complement represent values from -3 to $+3$. Two's Complement provides an asymmetric range from -4 to $+3$, thereby representing one additional negative value compared with the other signed representations. The figure also demonstrates visually how the available numerical range depends on the representation method even when the same 3-bit word length is used.

The numerical limits identified in Table 17 and illustrated in Figure 2 also determine the overflow and underflow boundaries of each representation. When the result of an arithmetic operation exceeds the maximum or minimum value representable within a fixed bit width, the result can no longer be represented correctly. Although all representation methods are affected by fixed-width limitations, their overflow and underflow boundaries differ according to their respective numerical ranges. Table 18 summarizes the representable ranges and the corresponding positive and negative overflow boundaries for the four binary number representation methods.

Table 18. Representable numerical ranges and overflow boundaries for fixed-width binary representations

Representation	Representable Range	Positive Overflow	Negative Overflow / Underflow
Unsigned	0 to $2^n - 1$	Result $> 2^n - 1$	N/A
Sign-Magnitude	$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$	Result $> +(2^{n-1} - 1)$	Result $< -(2^{n-1} - 1)$
One's Complement	$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$	Result $> +(2^{n-1} - 1)$	Result $< -(2^{n-1} - 1)$
Two's Complement	-2^{n-1} to $+(2^{n-1} - 1)$	Result $> +(2^{n-1} - 1)$	Result $< -2^{n-1}$

As shown in Table 18, the overflow boundaries are directly determined by the numerical range available in each representation. Unsigned representation provides the largest positive range, but it cannot represent negative values. Sign-Magnitude and One's Complement have identical numerical ranges, resulting in the same positive and negative boundaries. Two's Complement provides an asymmetric range, extending one value further in the negative direction than in the positive direction. Consequently, its minimum representable value is -2^{n-1} , while its maximum value is $+(2^{n-1} - 1)$. These differences in numerical boundaries are important when evaluating the behavior of fixed-width arithmetic operations.

The practical implications of these numerical boundaries can be illustrated using specific boundary values and arithmetic operations in a 3-bit representation. Because each representation assigns different numerical meanings to the available binary combinations, the same binary pattern may represent different values depending on the encoding scheme. Table 19 presents boundary cases and arithmetic operations that reach or exceed the representable limits of the four representation methods.

Table 19. Boundary values and arithmetic overflow conditions in 3-bit binary representations

Case	Result	Unsigned	Sign-Magnitude	One's Complement	Two's Complement
Boundary value	+4	100,Representable	Not representable-Overflow	Not representable-Overflow	Not representable-Overflow
Boundary value	-4	N/A	Not representable-Underflow	Not representable-Underflow	100,Representable
Arithmetic operation	$3+1=4$	Representable	Overflow	Overflow	Overflow
Arithmetic operation	$-3+(-1)=-4$	N/A	Underflow	Underflow	Representable

As shown in Table 19, the representability of boundary values and the occurrence of overflow or underflow depend directly on the selected representation method. In particular, $+4$ is representable in 3-bit Unsigned representation but exceeds the positive range of the three signed representations, whereas -4 is representable only in 3-bit Two's Complement among the signed methods considered. The arithmetic examples further demonstrate how these boundaries are encountered during addition. These cases illustrate the direct relationship between numerical range, binary encoding, and fixed-width arithmetic behavior.

Comparison of Binary Representations

To illustrate these theoretical ranges, Table 20 presents the complete set of 3-bit binary combinations interpreted according to the four representation methods. This comparison demonstrates how the same binary pattern can represent different decimal values depending on the selected representation.

Table 20. Comparison of three-bit unsigned and signed number representations

Binary Number	Unsigned	Sign-Magnitude	One's Complement	Two's Complement
000	0	+0	0	0
001	1	+1	1	1
010	2	+2	2	2
011	3	+3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1

Table 20 highlights the fundamental differences in the interpretation of binary numbers across the four representation methods. The binary patterns with the most significant bit equal to 0 (000, 001, 010, and 011) represent the same positive decimal values in all four representations. However, significant differences arise when the most significant bit is 1. In the Unsigned representation, these binary patterns are interpreted as ordinary positive binary numbers because no sign bit exists. In contrast, the same binary patterns are interpreted as signed values in Sign-Magnitude, One's Complement, and Two's Complement representations. As a result, each binary pattern corresponds to three different negative decimal values depending on the selected representation method. Furthermore, Sign-Magnitude and One's Complement provide two representations of zero, whereas Two's Complement eliminates the duplicate zero and introduces one additional negative value (-4 in the 3-bit case). This comparison demonstrates that the decimal value of a binary number depends not only on its bit pattern but also on the representation method used for its interpretation.

While the previous subsection demonstrated how the same binary pattern may correspond to different decimal values depending on the selected representation method, this subsection examines the reverse process. Specifically, it compares how identical decimal values are encoded into binary using Unsigned, Sign-Magnitude, One's Complement, and Two's Complement representations. Table 21 summarizes the binary representations of decimal values using the four encoding methods.

Table 21. Binary representations of decimal values using unsigned, sign-magnitude, one's complement, and two's complement (3 bits)

Decimal	Unsigned	Sign-Magnitude	One's Complement	Two's Complement
3	011	011	011	011
2	010	010	010	010
1	001	001	001	001
0	000	000	000	000
-0	x	100	111	x
-1	x	101	110	111
-2	x	110	101	110
-3	x	111	100	101
-4	x	x	x	100

Table 21 shows that all positive decimal values within the common representable range have identical binary representations in the four methods. Differences arise only when negative numbers are encoded. Since the Unsigned representation supports only non-negative integers, negative decimal values cannot be represented. In contrast, the three signed representations use different encoding techniques to represent the same negative decimal values. Sign-Magnitude represents negative numbers by setting the most significant bit to 1 while preserving the magnitude. One's Complement obtains the negative representation by inverting every bit of the corresponding positive number. Two's Complement generates the negative representation by adding 1 to the One's Complement representation.

Furthermore, Table 21 clearly demonstrates that the same negative decimal value is represented by different binary patterns in the three signed representation methods. For example, the decimal value -2 is represented as 110 in Sign-Magnitude, 101 in One's Complement, and 110 in Two's Complement, while the decimal value -3 is represented as 111, 100, and 101, respectively. These differences highlight that the binary encoding of signed integers depends entirely on the selected representation method.

The comparison also illustrates the fundamental distinction between the three signed representations. Sign-Magnitude stores the sign and magnitude separately, One's Complement represents negative values through bit inversion, whereas Two's Complement represents negative values by combining bit inversion with the addition of one. These differences form the basis for the distinct arithmetic properties discussed in the following subsections.

Comparison of Bit-Width Expansion

Increasing the bit width while preserving the numerical value is an essential operation in digital systems, particularly during data type conversions and arithmetic operations. Although all binary number representation methods support bit-width expansion, the expansion procedure differs according to the encoding scheme used for representing signed and unsigned integers.

	<i>bit 3</i>	<i>bit 2</i>	<i>bit 1</i>	<i>bit 0</i>
Unsigned	bit 3	bit 2	bit 1	bit 0
Sign-Magnitude	Sign	bit 2	bit 1	bit 0
One's Complement	Sign	bit 2	bit 1	bit 0
Two's Complement	Sign	bit 2	bit 1	bit 0

Figure 3. 4-bit layout of each representation (shaded box = sign bit)

Figure 3 illustrates the structural organization of a 4-bit word for each representation method and highlights the position of the sign bit in the three signed representations. Table 22 compares the expansion of 4-bit binary numbers to 6-bit binary numbers using the four representation methods (Hamacher et al., 2012; Parhami, 2010).

Table 22. Expansion of 4-bit binary numbers to 6 bits for different number representations

Representation	Original (4 Bits, MSB = 1)	Expanded (6 Bits)	Original (4 Bits, MSB = 0)	Expanded (6 Bits)
Unsigned	$1011_2 = 11$	$001011_2 = 11$	$0101_2 = 5$	$000101_2 = 5$
Sign-Magnitude	$1011_2 = -3$	$100011_2 = -3$	$0101_2 = +5$	$000101_2 = +5$
One's Complement	$1011_2 = -4$	$111011_2 = -4$	$0101_2 = +5$	$000101_2 = +5$
Two's Complement	$1011_2 = -5$	$111011_2 = -5$	$0101_2 = +5$	$000101_2 = +5$

Table 22 demonstrates that the Unsigned representation increases the bit width by inserting leading zeros before the original binary number. Since unsigned numbers do not contain a sign bit, this operation preserves the numerical value without requiring any additional processing. For the Sign-Magnitude representation, the expansion procedure differs because the most significant bit is reserved for the sign. Instead of inserting leading zeros before the entire binary number, the sign bit remains unchanged, while zeros are inserted between the sign bit and the magnitude bits. Consequently, the sign and the numerical magnitude are both preserved after the expansion. In contrast, One's Complement and Two's Complement employ classical sign extension. During this process, the most significant bit (MSB) is replicated into the newly added higher-order bits. Therefore, if the MSB is 0, leading zeros are inserted, whereas if the MSB is 1, leading ones are inserted. This procedure guarantees that both positive and negative values retain their original numerical meaning after increasing the bit width.

The comparison clearly demonstrates that, although all four representations preserve the numerical value after bit-width expansion, they employ different mechanisms to achieve this objective. Unsigned representation relies on zero padding, Sign-Magnitude preserves the sign while inserting additional bits between the sign bit and the magnitude bits, whereas One's Complement and Two's Complement employ classical sign extension by replicating the most significant bit. This distinction is particularly important in computer architecture because sign extension is extensively used by modern processors during data type conversions between different word sizes (Hamacher et al., 2012; Parhami, 2010; Intel Corporation, n.d.; RISC-V International, n.d.)

Illustrative Binary Representation Examples

To further demonstrate the differences among the four binary number representation methods, this subsection presents a set of illustrative conversion examples. Unlike the previous comparisons, which were limited to 3-bit

representations, the following examples use binary numbers with different bit widths. These examples provide a practical demonstration of how decimal values are encoded into binary and how binary patterns are interpreted as decimal values under different representation methods.

Table 23 presents the conversion of decimal numbers into binary using Unsigned, Sign-Magnitude, One's Complement, and Two's Complement representations. The examples demonstrate that positive decimal values generally produce identical binary representations in all four methods, whereas negative values require different encoding techniques depending on the selected representation. As expected, negative decimal values cannot be represented using the Unsigned representation.

Table 23. Conversion of decimal numbers to binary using different binary representation methods

Decimal	Unsigned	Sign-Magnitude	One's Complement	Two's Complement
10 ₍₁₀₎	1010 ₍₂₎	01010 ₍₂₎	01010 ₍₂₎	01010 ₍₂₎
-10 ₍₁₀₎	X	11010 ₍₂₎	10101 ₍₂₎	10110 ₍₂₎
-25 ₍₁₀₎	X	111001 ₍₂₎	100110 ₍₂₎	100111 ₍₂₎
25 ₍₁₀₎	11001 ₍₂₎	011001 ₍₂₎	011001 ₍₂₎	011001 ₍₂₎

Table 24 illustrates the reverse conversion process by interpreting the same binary value according to each representation method. The examples clearly show that identical binary patterns may correspond to different decimal values depending on the representation used. This observation reinforces one of the central findings of this study: a binary pattern has no intrinsic numerical meaning unless the corresponding representation method is specified.

Table 24. Conversion of binary numbers to decimal by interpreting the same binary value using different binary representation methods

Binary	Unsigned	Sign-Magnitude	One's Complement	Two's Complement
1101 ₍₂₎	13 ₍₁₀₎	-5 ₍₁₀₎	-2 ₍₁₀₎	-3 ₍₁₀₎
10001 ₍₂₎	17 ₍₁₀₎	-1 ₍₁₀₎	-14 ₍₁₀₎	-15 ₍₁₀₎
1111 ₍₂₎	15 ₍₁₀₎	-7 ₍₁₀₎	-0 ₍₁₀₎	-1 ₍₁₀₎
101011 ₍₂₎	43 ₍₁₀₎	-11 ₍₁₀₎	-20 ₍₁₀₎	-21 ₍₁₀₎

Taken together, Tables 23 and 24 provide complementary examples of binary encoding and interpretation. While Table 23 focuses on converting decimal values into binary representations, Table 24 demonstrates how the same binary pattern may produce different decimal values when interpreted using different representation methods. These examples reinforce the theoretical comparisons presented in the previous subsections and facilitate a deeper understanding of binary number representations.

Comparative Advantages and Disadvantages

Table 25. Comparative advantages and disadvantages of binary number representation methods

Representation	Advantages	Disadvantages
Unsigned	- Utilizes all 2^n binary combinations. - Single zero representation. - Simple encoding and decoding. - Efficient for non-negative values.	- Cannot represent negative values. - Unsuitable for signed arithmetic.
Sign-Magnitude	- Simple representation of sign and magnitude. - Easy to determine the sign of a number.	- Two zero representations. - More complex arithmetic operations. - Does not use classical sign extension; the sign bit must be preserved separately during bit-width expansion.
One's Complement	- Simple generation of negative values by bit inversion. - Supports sign extension.	- Two zero representations. - End-around carry required during addition. - More complex hardware implementation.
Two's Complement	- Single zero representation. - Uses all 2^n binary combinations. - Supports classical sign extension. - Efficient hardware arithmetic. - Standard representation in modern systems.	- Asymmetric numerical range. - Slightly less intuitive for beginners.

Each binary number representation method offers specific advantages and limitations that influence its suitability for different applications. While some methods simplify the representation of signed numbers, others improve arithmetic efficiency or maximize the number of representable values. Table 25 provides a comparative summary of the principal advantages and disadvantages of Unsigned, Sign-Magnitude, One's Complement, and Two's Complement representations.

As shown in Table 25, each representation method involves a different set of design trade-offs. Unsigned representation is simple and efficiently utilizes all available binary combinations but cannot represent negative values. Sign-Magnitude provides an intuitive separation between the sign and magnitude; however, it introduces two representations of zero and complicates arithmetic operations. One's Complement simplifies the generation of negative numbers through bit inversion and supports classical sign extension, but it also suffers from duplicate zero representations and requires an end-around carry during addition. In contrast, Two's Complement combines a unique representation of zero, full utilization of the available binary combinations, efficient hardware arithmetic, and classical sign extension, making it the preferred representation for signed integers in modern computer systems (ARM Ltd., n.d.; Intel Corporation, n.d.; Parhami, 2010; Patterson & Hennessy, 2020; RISC-V International, n.d.).

Summary Comparison

The previous subsections examined the four binary number representation methods from different perspectives, including numerical range, binary encoding, bit-width expansion, conversion examples, and their respective advantages and disadvantages. To consolidate these findings, Table 26 summarizes the principal characteristics of Unsigned, Sign-Magnitude, One's Complement, and Two's Complement representations. The comparison focuses on the maximum number of representable values, zero representation, support for classical sign extension, and suitability for efficient hardware arithmetic.

Table 26. Summary comparison of unsigned, sign-magnitude, one's complement, and two's complement representations

Representation	Maximum Number of Representable Values	Zero Representation	Classical Sign Extension	Efficient Hardware Arithmetic
Unsigned	2^n	Single Zero	N/A	✓
Sign-Magnitude	$2^n - 1$	Positive and Negative Zero	x	x
One's Complement	$2^n - 1$	Positive and Negative Zero	✓	x
Two's Complement	2^n	Single Zero	✓	✓

As summarized in Table 26, each representation method has distinct characteristics and design trade-offs. Unsigned representation efficiently utilizes all available binary combinations but is limited to non-negative integers. Sign-Magnitude and One's Complement extend the representation to signed integers; however, both introduce two representations of zero, reducing the number of distinct representable values and increasing arithmetic complexity. In contrast, Two's Complement combines a unique representation of zero, full utilization of the available binary combinations, support for classical sign extension, and efficient hardware arithmetic. These characteristics explain why Two's Complement has become the standard representation for signed integers in modern computer systems (ARM Ltd., n.d.; Intel Corporation, n.d.; Parhami, 2010; Patterson & Hennessy, 2020; RISC-V International, n.d.).

The comparative analysis presented in this section demonstrates that, although all four representation methods encode binary integers, they differ significantly in their design, implementation, and practical applications. The unified comparison highlights the strengths and limitations of each method and provides a structured foundation for understanding their use in computer systems. These observations naturally lead to the discussion in the following section, which examines the reasons behind the widespread adoption of Two's Complement in modern computer architectures.

Discussion

The comparative analysis presented in the previous section demonstrates that binary number representation methods differ not only in their numerical ranges but also in their encoding mechanisms, bit-width expansion procedures, arithmetic properties, and hardware implementation. Although all four representations encode binary

integers, each method was developed to address specific limitations of its predecessors. The findings of this study provide clear answers to the research questions introduced in Section I (Parhami, 2010; Stallings, 2022).

Regarding RQ1, the analysis shows that the four representation methods differ fundamentally in the way they encode signed integers. Unsigned representation is limited to non-negative values because all available bits are allocated to representing numerical magnitude. In contrast, Sign-Magnitude, One's Complement, and Two's Complement extend binary representation to include negative values using different encoding strategies. Sign-Magnitude stores the sign and magnitude separately, One's Complement generates negative numbers through bit inversion, whereas Two's Complement combines bit inversion with the addition of one (Parhami, 2010; Stallings, 2022).

Regarding RQ2, the comparison demonstrates that these representations exhibit different characteristics in terms of numerical range, zero representation, bit-width expansion, and arithmetic implementation. Unsigned representation provides the largest range of non-negative integers by utilizing all available binary combinations. Sign-Magnitude and One's Complement, however, sacrifice one distinct representable value because both contain two representations of zero. Furthermore, the analysis highlights different approaches to bit-width expansion. Unsigned representation performs zero padding, Sign-Magnitude preserves the sign while inserting additional bits between the sign bit and the magnitude bits, whereas One's Complement and Two's Complement employ classical sign extension by replicating the most significant bit. These differences have a direct impact on arithmetic operations and processor implementation (Hamacher et al., 2012; Parhami, 2010; Stallings, 2022).

Regarding RQ3, the results clearly explain why Two's Complement has become the standard representation for signed integers in modern computer systems. Unlike Sign-Magnitude and One's Complement, Two's Complement eliminates the duplicate representation of zero and fully utilizes all available binary combinations (Parhami, 2010). Furthermore, Two's Complement allows addition and subtraction to be implemented using a common binary adder, with subtraction performed by complementing the subtrahend and adding one. Classical sign extension simplifies data type conversion between different word sizes, making Two's Complement particularly suitable for modern processor architectures (ARM Ltd., n.d.; Intel Corporation, n.d.; Parhami, 2010; Patterson & Hennessy, 2020; RISC-V International, n.d.).

Overall, the comparative analysis confirms that the adoption of Two's Complement is not arbitrary but is the result of several technical advantages. While the other representation methods remain valuable for understanding the historical development of signed binary arithmetic and the fundamental concepts of computer architecture, Two's Complement offers the best balance between representational efficiency, arithmetic simplicity, and hardware implementation. These characteristics explain why Two's Complement has remained the dominant representation for signed integers in virtually all modern computer systems (ARM Ltd., n.d.; Intel Corporation, n.d.; Parhami, 2010; Patterson & Hennessy, 2020; RISC-V International, n.d.).

Conclusion

This paper presented a comprehensive comparative analysis of four binary number representation methods: Unsigned, Sign-Magnitude, One's Complement, and Two's Complement. The comparison examined their numerical ranges, binary encoding mechanisms, bit-width expansion techniques, zero representation, sign extension, and suitability for hardware arithmetic. A unified comparative framework supported by formulas, illustrative examples, and comparative tables was developed to highlight the similarities and differences among these representation methods.

The analysis demonstrated that each representation method has distinct characteristics, advantages, and limitations. Unsigned representation efficiently utilizes all available binary combinations for non-negative integers, whereas Sign-Magnitude and One's Complement extend the representation to signed numbers but introduce additional complexity due to the existence of two representations of zero. In contrast, Two's Complement combines a unique zero representation, full utilization of the available binary combinations, classical sign extension, and efficient hardware arithmetic, making it the most practical and widely adopted representation for signed integers in modern computer systems.

The comparative framework presented in this paper provides a structured reference for understanding binary number representations and their practical implications in computer architecture and digital systems. By evaluating the four representation methods using consistent technical criteria, the analysis clarifies the design

trade-offs associated with numerical range, zero representation, bit-width expansion, overflow behavior, and arithmetic implementation.

Future work may extend this analysis by examining the implications of these representations for subtraction, multiplication, and division, as well as their hardware complexity and implementation in contemporary processor architectures. Further studies may also investigate how knowledge of binary number representations can be applied in computer architecture education.

Recommendations

Based on the comparative analysis presented in this study, several recommendations can be formulated regarding the selection and interpretation of binary number representation methods.

First, Two's Complement should be preferred for signed integer representation in modern digital systems because it provides a unique representation of zero, utilizes all available binary combinations, supports straightforward sign extension, and enables efficient implementation of arithmetic operations.

Second, the representation method should always be explicitly specified when binary patterns are interpreted or communicated. The same binary pattern may correspond to substantially different numerical values depending on whether Unsigned, Sign-Magnitude, One's Complement, or Two's Complement representation is assumed.

Third, particular attention should be given to the numerical boundaries of fixed-width representations when designing or analyzing arithmetic operations. Overflow conditions should be considered whenever an arithmetic result exceeds the maximum representable value, while negative boundary violations should be evaluated against the minimum representable value of the selected representation.

Finally, the comparative framework presented in this study can be extended in future work to examine additional aspects of computer arithmetic, including subtraction, multiplication, and division, hardware implementation complexity, and the behavior of fixed-width arithmetic in contemporary processor architectures.

Scientific Ethics Declaration

* The author declares that the scientific ethical and legal responsibility of this article published in EPSTEM Journal belongs to the author.

Conflict of Interest

* The author declares that there is no conflict of interest.

Funding

* There is no funding available for this study.

Acknowledgements or Notes

* This article was presented as an oral presentation at the International Conference on Basic Sciences, Engineering and Technology (www.icbaset.net) held in Istanbul/Türkiye on September 17-18, 2026.

References

ARM Ltd. (n.d.). *Arm® architecture reference manual*. <https://support.arm.com/documentation/ddi0487/mc/>
Bastien, J. F. (2018). *Signed integers are two's complement* (P0907R4). ISO/IEC JTC1/SC22/WG21.

- Hamacher, C., Vranesic, Z., Manjikian, N., & Zaky, S. (2012). *Computer organization and embedded systems* (6th ed.). McGraw-Hill.
- Intel Corporation. (n.d.). *Intel® 64 and IA-32 architectures software developer's manual*. <https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html>
- Mano, M. M., & Ciletti, M. D. (2018). *Digital design* (6th ed.). Pearson.
- Parhami, B. (2010). *Computer arithmetic: Algorithms and hardware designs* (2nd ed.). Oxford University Press.
- Patterson, D. A., & Hennessy, J. L. (2020). *Computer organization and design RISC-V edition* (2nd ed.) Morgan Kaufmann.
- RISC-V International. (n.d.). *The RISC-V instruction set manual: Volume I, unprivileged ISA*. <https://docs.riscv.org/reference/isa/v20260120/unpriv/unpriv-index.html>
- Stallings, W. (2022). *Computer organization and architecture: Designing for performance* (11th ed.). Pearson.
- Tanenbaum, A. S., & Austin, T. (2013). *Structured computer organization* (6th ed.). Pearson.

Author(s) Information

Bojken Shehu

Polytechnic University of Tirana

“Mother Tereza” Square. Tirana, 1001, Albania

ORCID iD: <https://orcid.org/0009-0003-2109-00666>

*Contact e-mail: bshehu@fti.edu.al

*Corresponding author(s) contact email

To cite this article:

Shehu, B. (2026). Why two's complement became the standard: A comparative analysis of unsigned, sign-magnitude, one's complement, and two's complement representations. *The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM)*, 42, 136-151.