Number Representation and Arithmetic Operations
1.4.1 Integers
Consider an n-bit vector : B = bn−1 . . . b1b0 where bi = 0 or 1 for 0 ≤ i ≤ n − 1.

$$V(B) = b_{n-1} \times 2^{n-1} + \cdots + b_1 \times 2^1 + b_0 \times 2^0$$

We need to represent both positive and negative numbers. Three systems are used for representing such numbers:
• Sign-and-magnitude
• 1's-complement
• 2's-complement

In all three systems, the leftmost bit is 0 for positive numbers and 1 for negative numbers.

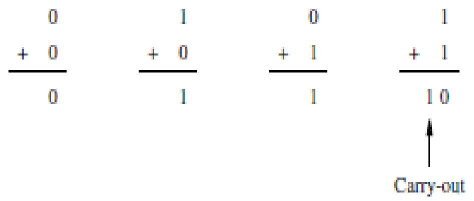
B	Values represented		
$b_3 b_2 b_1 b_0$	Sign and magnitude	1's complement	2's complement
0 1 1 1	+7	+7	+7
0 1 1 0	+6	+6	+6
0 1 0 1	+5	+5	+5
0 1 0 0	+4	+4	+4
0 0 1 1	+3	+3	+3
0 0 1 0	+2	+2	+2
0 0 0 1	+1	+1	+1
0 0 0 0	+0	+0	+0
1 0 0 0	−0	−7	−8
1 0 0 1	−1	−6	−7
1 0 1 0	−2	−5	−6
1 0 1 1	−3	−4	−5
1 1 0 0	−4	−3	−4
1 1 0 1	−5	−2	−3
1 1 1 0	−6	−1	−2
1 1 1 1	−7	−0	−1

In *1's-complement* representation, negative values are obtained by complementing each bit of the corresponding positive number. Thus, the representation for −3 is obtained by complementing each bit in the vector 0011 to yield 1100.

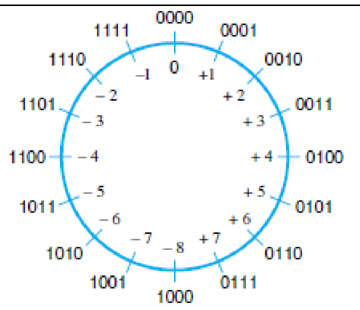
In the *2's-complement* system, forming the 2's-complement of an *n*-bit number is done by subtracting the number from 2^n. Hence, the 2's-complement of a number is obtained by adding 1 to the 1's-complement of that number.

There are distinct representations for +0 and −0 in both the sign-and magnitude and 1's-complement systems, but the 2's-complement system has only one representation for 0.

Addition of Unsigned Integers
The sum of 1 and 1 is the 2-bit vector 10, which represents the value 2. We say that the *sum* is 0 and the *carry-out* is 1. We add bit pairs starting from the low-order (right) end of the bit vectors, propagating carries toward the high-order (left) end. The carry-out from a bit pair becomes the *carry-in* to the next bit pair to the left.



Addition and Subtraction of Signed Integers
The 2's-complement system is the most efficient method for performing addition and subtraction operations.
Unsigned integers mod *N* is a circle with the values 0 through *N* − 1. The decimal values 0 through 15 are represented by their 4-bit binary values 0000 through 1111.



(b) Mod 16 system for 2's-complement numbers

The operation (7 + 5) mod 16 yields the value 12. To perform this operation graphically, locate 7 (0111) on the outside of the circle and then move 5 units in the clockwise direction to arrive at the answer 12 (1100).
Similarly, (9 + 14) mod 16 = 7; this is modeled on the circle by locating 9 (1001) and moving 14 units in the clockwise direction past the zero position to arrive at the answer 7 (0111).
Apply the mod 16 addition technique to the example of adding +7 to −3. The 2's-complement representation for these numbers is 0111 and 1101, respectively.

To *add* two numbers, add their *n*-bit representations, ignoring the carry-out bit from the most significant bit (MSB) position. The sum will be the algebraically correct value in 2's-complement representation if the actual result is in the range -2^{n-1} through $+2^{n-1} - 1$.
To *subtract* two numbers X and Y, that is, to perform $X - Y$, form the 2's-complement of Y, then add it to X using the *add* rule. Again, the result will be the algebraically correct value in 2's-complement representation if the actual result is in the range -2^{n-1} through $+2^{n-1} - 1$.

0010	(+2)	(b)	0100	(+4)
+ 0011	(+3)		+ 1010	(−6)
0101	(+5)		1110	(−2)
1011	(−5)	(d)	0111	(+7)
+ 1110	(−2)		+ 1101	(−3)
1001	(−7)		0100	(+4)

Floating-Point Numbers

If we use a full word in a 32-bit word length computer to represent a signed integer in 2's-complement representation, the range of values that can be represented is -2^{31} to $+2^{31} - 1$.
Since the position of the binary point in a floating-point number varies, it must be indicated explicitly in the representation. For example, in the familiar decimal scientific notation, numbers may be written as 6.0247×10^{23}, 3.7291×10^{-27}, -1.0341×10^{2}, -7.3000×10^{-14}. these numbers have been given to 5 *significant digits* of precision.

A binary floating-point number can be represented by:
· a sign for the number
· some significant bits
· a signed scale factor exponent for an implied base of 2

Character Representation

Bit positions 3210	Bit positions 654: 000	001	010	011	100	101	110	111
0000	NUL	DLE	SPACE	0	@	P	´	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	/	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	—	o	DEL

The most common encoding scheme for characters is ASCII (American Standard Code for Information Interchange). Alphanumeric characters, operators, punctuation symbols, and control characters are represented by 7-bit codes. It is convenient to use an 8-bit *byte* to represent and store a character.
The code occupies the low-order seven bits. The high-order bit is usually set to 0. This facilitates sorting operations on alphabetic and numeric data.
The low-order four bits of the ASCII codes for the decimal digits 0 to 9 are the first ten values of the binary number system.
This 4-bit encoding is referred to as the *binary-coded decimal* (BCD) code.