Machine Representation of Numbers

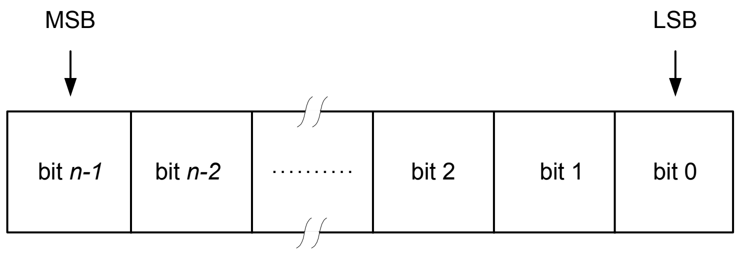
Objectives

- In this lesson, you will learn how signed numbers (positive or negative) are represented in digital computers.
- You will learn the 2 main methods for signed number representation:
    a. The signed-magnitude method, and
    b. The complement method.

Registers

- Digital computers store numbers in special digital electronic devices called **Registers**
- **Registers** consist of a fixed number *n* of *storage elements.*
- Each storage element is capable of storing one bit of data (either 0 or 1}.
- The register ***size*** is the number of storage bits in this register (*n*).
- Thus, registers are capable of holding *n*-bit binary numbers
- Register size (*n*) is typically a power of 2, e.g. 8, 16, 32, 64, etc.
- An *n-bit* register can represent (store) one of 2^n *Distinct Values*.
- Numbers stored in registers may be either *unsigned* or *signed* numbers. For example, **13** is an unsigned number but **+13** and **–13** are signed numbers.

Unsigned Number Representation



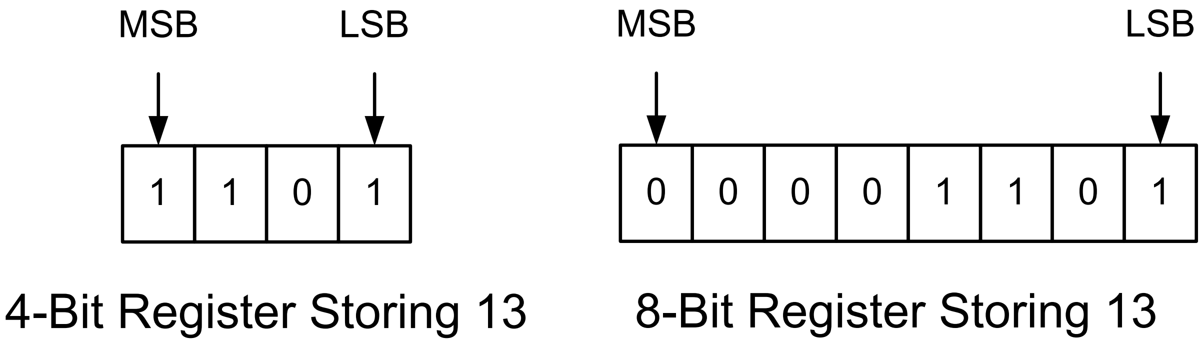
N-Bit Register holding an n-Bit Unsigned Number

- A register of n-bits, can store any unsigned number that has n-bits or less.

- Typically, the *rightmost* bit of the register is designated to be the least significant bit (LSB), while the *leftmost* bit is designated to be the most-significant bit (MSB).
- When representing an integer number , this *n-bit* register can hold values from 0 up to ($2^n - 1$).

Example

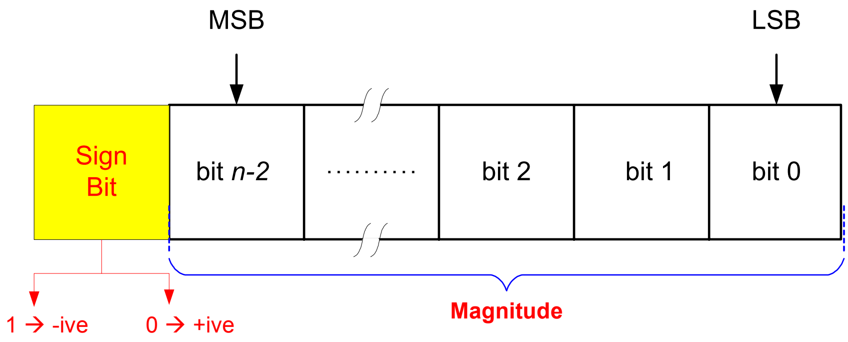
Show how the value $(13)_{10}$ (or **D** in Hexadecimal) is stored in a 4-bit register and in an 8-bit register



Signed Number Representation

- The n-bits of the register holding an <u>*unsigned number*</u> need only represent the value (magnitude) of the number. No sign information needs to be represented in this case.
- In the case of a signed number, the *n-bits* of the register should represent both the magnitude of the number and its sign as well.
- Two major techniques are used to represent signed numbers:
  1. Signed Magnitude Representation
  2. Complement method
     - Radix (R's) Complement (2's Complement)
     - Diminished Radix (R-1's) Complement (1's Complement)

Signed Magnitude Number Representation



Signed-Magnitude Number Representation in *n*-Bit Register

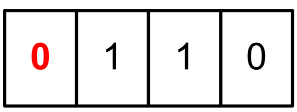
- Independent Representation of The Sign and The Magnitude
- The leftmost bit is used as a *Sign Bit*.
- The *Sign Bit* :
  - = 0 → +ive number
  - = 1 → -ive number.
- The remaining (*n-1*) bits are used to represent the magnitude of the number.
- Thus, the *largest* representable *magnitude*, in this method, is $(2^{n-1}-1)$

Example

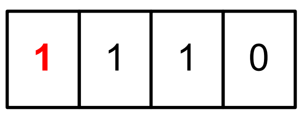
Show the signed-magnitude representations of +6, -6, +13 and –13 using a 4-Bit register and an 8-Bit register

Solution

- **For a 4-bit register**, the leftmost bit is a sign bit, which leaves 3 bits only to represent the magnitude.
- The largest magnitude representable in 3-bits is 7. Accordingly, we cannot use a 4-bit register to represent +13 or –13.



Signed-Magnitude Representation of +6



Signed-Magnitude Representation of -6

- **For an 8-bit register**, the leftmost bit is a sign bit, which leaves 7 bits to represent the magnitude.
- The largest magnitude representable in 7-bits is 127 (= 2^7-1).

0	0	0	0	0	1	1	0

Signed-Magnitude Representation of +6

1	0	0	0	0	1	1	0

Signed-Magnitude Representation of -6

0	0	0	0	1	1	0	1

Signed-Magnitude Representation of +13

1	0	0	0	1	1	0	1

Signed-Magnitude Representation of -13

Notes

1. Signed magnitude method has Two representations for 0 → {+0 , -0} → nuisance for implementation.

2. Signed magnitude method has a symmetric range of representation {-(2^{n-1} -1) : +(2^{n-1}- 1)}
3. Harder to implement addition/subtraction.
   a) *The sign and magnitude parts have to be processed independently.*
   b) *Sign bits of the operands have to be examined to determine the actual operation (addition or subtraction).*
   c) *Separate circuits are required to perform the addition and subtraction operations.*
4. Multiplication & division are less problematic.

Complement Representation

- Positive numbers (+N) are represented in ***exactly*** the same way as in signed magnitude system
- Negative numbers (-N) are represented by the *complement* of N (**N'**)

Define the Complement N' of some number N as:

$$N' = M - N \qquad \text{where, } M = \text{Some Constant}$$

- Applying a negative sign to a number (N→ **-N**) is equivalent to *Complementing* that number (N → **N'**)
- Thus, given the representation of some number N, the representation of –N is equivalent to the representation of the complement N'.

Important Property:

- The Complement of the Complement of some number *N* is the original number *N*.

$$N' = M - N$$

$$(N')' = M - (M - N) = N$$

- This is a required property to match the negation process since a number negated twice must yield the original number {- (-**N**) = N}

Why Use the Complement Method ?

Through the proper choice of the constant **M**, the complement operation can be fairly *simple* and quite *fast*. A simple complement process allows:

i. Simplified arithmetic operations since. subtraction can be totally replaced by addition and complementing.
ii. Lower cost, since no subtractor circuitry will be required and only an adder is needed.

Complement Arithmetic

Basic Rules

1. Negation is replaced by complementing (**- N** → **N'**)
2. Subtraction is replaced by addition to the complement.
   - *Thus,* (X – Y) *is replaced by* (X + Y')

Choice of M

The value of M should be chosen such that:

1. It simplifies the computation of the complement of a number.
2. It results in simplified arithmetic operations.

- Consider the operation

$$Z = X – Y,$$

where *both X and Y are positive* numbers

- In complement arithmetic, Z is computed by *adding* X to the *complement* of Y

$$Z = X + Y'$$

Consider the following two possible cases:

First case Y > X → **(Negative Result)**

- The result **Z** is **–ive**, *where*

$$Z = – (Y-X) →$$

- Being –ive**, Z** should be represented in the *complement form* as **M-(Y-X)**
- Using the complement method:

$$Z = X - Y$$

$$Z = X + Y'$$

$$= X + (M-Y)$$

$$= M - (Y-X)$$

= *Correct Answer in the Complement Form*

- Thus, in the case of a *negative result*, any value of M may be used.

Second case Y < X → (Positive Result)

The result **Z** is **+ive** *where,*

$$Z = +(X-Y).$$

Using complement arithmetic we get:

$$Z = X-Y$$

$$Z = X + Y'$$

$$= X + (M-Y)$$

$$Z = M + (X-Y)$$

- which is *different* from the *expected correct result* **of +(X-Y)**

- In this case, a *correction step* is required for the final result.
- The choice of the value of M affects the complexity of this correction step.

To summarize,

There are two constraints on the choice of M

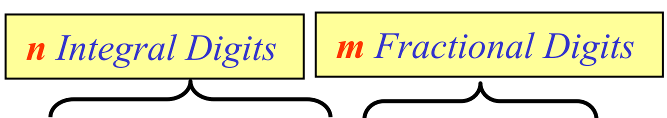
1. Simple and fast complement operation.
2. Elimination or simplification of the correction step.

R's and (R-1)'s Complements

- Two complement methods have generally been used.
- The two methods differ in the choice of the value of M.

  1. The diminished radix complement method {(R-1)'s Complement }, and
  2. The radix complement method (R's Complement).

- Consider the number **X**, with *n* integral digits and *m* fractional digits, where



$$\mathbf{X} = X_{n-1}\, X_{n-2}\ \ldots\ X_1\, X_0 \,.\, X_{-1}\, X_{-2} \ldots\ X_{-m}$$

- Next, we will show how to compute the (R-1)'s and the R's complements of X

The Diminished Radix Complement (R-1)'s Complement:

$$\mathbf{M}_{R-1} = r^{n} - r^{-m}$$

where; $r^{n} = 1\underbrace{000\ldots00}_{n\ digits}.\underbrace{00\ldots000}_{m\ digits}$ (the 1 is in the $(n+1)^{th}$ Position)

and; $r^{-m} = \underbrace{000\ldots00}_{n\ Positions}.\underbrace{00\ldots001}_{m\ Positions}$

- *Note that, if X is integer, then* $m=0$ *and* $r^{-m} = 1$.

Thus; $r^{-m} = 000\ldots00 . 00\ldots001$

$= $ *Unit (one) in Least Position (ulp)*

OR $\mathbf{M}_{R-1} = r^{n} - ulp$

where; $ulp = $ *Unit (one) in Least Position* $= r^{-m}$

Important Notes:

- The (R-1)'s complement of X will be denoted by X'_{r-1}.
- $(r^{n} - r^{-m})$ **is the** *largest number representable* in *n integral* digits and *m fractional* digits.
- $X'_{r-1} = L - X$, *where* L is *largest number representable* in *n integral* digits and *m fractional* digits

The shown table shows how to compute the (r-1)'s complement of X for various number systems

Number System	(R-1)'s Complement	Complement of X (X'_{r-1})
Decimal	**9's** Complement	$X'_9 = (10^n - 10^{-m}) - X$ $= 99...9.99...9 - X$
Binary	**1's** Complement	$X'_1 = (2^n - 2^{-m}) - X$ $= 11...1.111...1 - X$
Octal	**7's** Complement	$X'_7 = (8^n - 8^{-m}) - X$ $= 77...7.77...7 - X$
Hexadecimal	**F's** Complement	$X'_F = (16^n - 16^{-m}) - X$ $= FF...F.FF...F - X$

n-integral digits

m-fractional digits

Radix Complement (R's Complement):

$$M_R = r^n$$

Note that $r^n = 1000...00.000$

(n+1)

$(n+1)^{th}$ Position

n Positions

Notes:

1. The R's complement of X will be denoted by X'_r.
2. M_R depends only on the number of integral digits (n), but is independent of the number of fractional digits (m).
3. $X'_r = r^n - X$
4. $X'_{r-1} = (r^n - ulp) - X$

5. **Thus,** $X'_r = X'_{r-1} + ulp$, *i.e R's complement* =(*R-1*)*'s complement + ulp*

The shown table summarizes the radix complement computation of X for various number systems

Number System	R's Complement	Complement of X (X'_r)
Decimal	**10's** Complement	$X'_{10} = 10^n - X$
Binary	**2's** Complement	$X'_2 = 2^n - X$
Octal	**8's** Complement	$X'_8 = 8^n - X$
Hexa-decimal	**16's** Complement	$X'_{16} = 16^n - X$

Examples

Find the 9's and the 10's complement of the following decimal numbers:

a- 2357

b- 2895.786

Solution:

a- X = 2357→ $n=4$,

- $X'_9 = (10^4 - ulp) - 2357$
  $= 9999 - 2357 = 7642$
- $X'_{10} = 10^4 - 2357 = 7643$;
- *Alternatively*, $X'_{10} = X'_9 + 0001 = 7643$

b- X = 2895.786→ $n=4$, $m=3$

- $X'_9 = (10^4 - ulp) - 2895.786$
  $= 9999.999 - 2895.786 = 7104.213$

- $X'_{10} = 10^4 - 2895.786 = 7104.214$;
- *Alternatively*, $X'_{10} = X'_9 + 0000.001 = 7104.214$

Example

Find the 1's and the 2's complement of the following binary numbers:

a- 110101010

b- 1010011011

c- 1010.001

Solution:

a- X = 110101010→ *n=9,*

- $X'_1 = (2^9 - ulp) - 110101010 = 111111111 - 110101010$
  $= 001010101$
- $X'_2 = 2^9 - 110101010 = 1000000000 - 110101010$
  $= 001010110$
- *Alternatively*, $X'_2 = X'_1 + ulp = 001010101 + 000000001$
  $= 001010110$

b- X = 1010011011 → *n=10,*

- $X'_1 = (2^{10} - ULP) - 101001101 = 1111111111 - 101001101$
  $= 010110010$
- $X'_2 = 2^{10} - 101001101 = 10000000000 - 101001101 = 010110011$
- *Alternatively*, $X'_2 = X'_1 + ulp = 010110010 + 0000000001$
  $= 010110011$

c- X = 1010.001→ *n=4, m=3*

- $X'_1 = (2^4 - ULP) - 1010.001 = 1111.111 - 1010.001$
  $= 0101.110$
- $X'_2 = 2^4 - 1010.001 = 10000 - 1010.001$
  $= 0101.111$

- *Alternatively*, $X'_2 = X'_1 + ulp$ = 0101.110+ 0000.001

  = 0101.111

Important Notes:

1. The 1's complement of a number can be directly obtained by bitwise complementing of each bit, i.e. each 1 is replaced by a 0 and each 0 is replaced by a 1.
   - Example: X = 1 1 0 0 1 0 1 0 0 1
   - X_1' = 0 0 1 1 0 1 0 1 1 0

2. The 2's complement of a number can be visually obtained as follows:
   - Scan the binary number from right to left.
   - 0's are replaced by 0's till the first 1 is encountered.
   - The first encountered 1 is replaced by a 1 but from this point onwards each bit is complemented replacing each 1 by a 0 and each 0 by a 1
     - Example: X = 1 1 0 0 1 0 1 0 0
     - X_2' = 0 0 1 1 0 1 1 0 0

Example

Find the 7's and the 8's complement of the following octal numbers:

a- 6770

b- 541.736

Solution:

a- X = 6770→ *n=4,*

- $X'_7 = (8^4 - ULP) - 6770$ = 7777 – 6770

  = 1007

- $X'_8 = 8^4 - 6770$ = 10000 – 6770 = 1010
- *Alternatively*, $X'_8 = X'_7 + ulp$ = 1007+ 0001= 1010

b- X = 541.736→ *n=3,* → *m=4*

- $X'_7 = (8^3 - ULP) - 541.736$ = 777.7777 – 541.736= 236.041

- $X'_8 = 8^3 - 541.736$ = 1000 – 541.736 = 236.042
- *Alternatively*, $X'_8 = X'_7 + ulp$ = 236.041+ 0.001 = 236.042

Example

Find the F's and the 16's complement of the following HEX numbers:

a- 3FA9

b- 9B1.C70

Solution:

a- X = 3FA9→ *n=4,*

- $X'_F = (16^4 - ULP) - 3FA9$ = FFFF – 3FA9 = C056
- $X'_{16} = 16^4 - 3FA9$ = 10000 – 3FA9 = C057
- *Alternatively*, $X'_{16} = X'_F + ulp$ = C056+ 0001 = C057

b- X = 9B1.C70→ *n=3,* → *m=3*

- $X'_F = (16^3 - ULP) - 9B1.C70$ = FFF.FFF – 9B1.C70= 64E.38F
- $X'_{16} = 16^3 - 9B1.C70$ = 1000 – 9B1.C70 = 64E.390
- *Alternatively*, $X'_{16} = X'_F + ulp$ = 64E.38F + 000.001= 64E.390

Example

Show how the numbers +53 and -53 are represented in 8-bit registers using signed-magnitude, 1's complement and 2's complement representations.

	+53	-53
Signed Magnitude	00110101	10110101
1's Complement	00110101	11001010
2's Complement	00110101	11001011

Important Notes:

1. In ***all*** signed number representation methods, the leftmost bit indicates the sign of the number, i.e. it is considered as a *sign bit*
2. If the *sign bit* (leftmost) is 1, then the number is negative and if it is 0 the number is positive.

Comparison:

	Signed Magnitude	**1's Complement**	**2's Complement**
No. of 0's	2 **(± 0)**	2 **(± 0)**	1 **(+ 0)**
Symmetric	yes	yes	no
Largest +ive value	$+(2^{n-1}-1)$	$+(2^{n-1}-1)$	$+(2^{n-1}-1)$
Smallest –ive Value	$-(2^{n-1}-1)$	$-(2^{n-1}-1)$	-2^{n-1}

Quiz:

For the shown 4-bit numbers, write the corresponding decimal values in the indicated representation.

X	Un-signed	Signed Magnitude	1's Comp (X_1')	2's Comp (X_2')
0000				
0001				
0010				
0011				
0100				
0101				
0110				
0111				
1000				
1001				
1010				
1011				
1100				
1101				
1110				
1111				

End of Lessons Exercises

1. Find the binary representation in signed magnitude, 1's complement, and 2's complement for the following decimal numbers: +13, -13, +39, -39, +1, -1, +73 and -73. For all numbers, show the required representation for 6-bit and 8-bit registers
2. Indicate the decimal value corresponding to all 5-bit binary patterns if the binary pattern is interpreted as a number in the signed magnitude, 1's complement, and 2's complement representations.