Numerical Representations On The Computer: Negative And Rational Numbers

- How are negative and rational numbers represented on the computer?

James Tam

Representing Negative Numbers

Real world

•Positive numbers – just use the appropriate type and number of digits e.g., 12345.

•Negative numbers – same as the case of positive numbers but precede the number with a negative sign "-" e.g., -123456.

Computer world

Positive numbers – convert the number to binary e.g., 7 becomes 111

Negative numbers – employ signed representations.

Unsigned Binary

All of the digits (bits) are used to represent the number

e.g. $1010_2 = 10_{10}$

The sign must be represented explicitly (with a minus sign "-").

e.g. $-1010_2 = -10_{10}$

Signed Binary

Positive

One bit (called the sign bit or the most significant bit/MSB) is used to indicate the sign of the number

If the MSB equals 0 then number is positive

Negative

•e.g. 0 bbb is a positive number (bbb stands for a binary number)

If the MSB equals 1 then the number is negative

•e.g. 1 bbb is a negative number (bbb stands for a binary number)

Types of signed representations

- One's complement
- Two's complement

Converting From Unsigned Binary to 1's Complement

For positive values there is no difference

e.g., positive seven

0111 (unsigned)

0111 (1's complement)

For negative values reverse (flip) the bits (i.e., a 0 becomes 1 and 1 becomes 0).

e.g., minus six

-0110 (unsigned)

1001 (1's complement)

Converting From Unsigned Binary to 2's Complement

For positive values there is no difference

e.g., positive seven

0111 (unsigned)

0111 (2's complement)

For negative values reverse (flip) the bits (i.e., a 0 becomes 1 and 1 becomes 0) *and add one to the result*.

e.g., minus six

-0110 (unsigned)

1010 (2's complement)

Interpreting The Pattern Of Bits

Bit pattern	Unsigned binary	1's complement	2's complement
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	-7	-8
1001	9	-6	-7
1010	10	-5	-6
1011	11	-4	-5
1100	12	-3	-4
1101	13	-2	-3
1110	14	-1	-2
1111	15	-0	-1

Overflow: Unsigned

Occurs when you don't have enough bits to represent a value

Binary (1 bit)	Value
0	0
1	1

Binary (2 bits)	Value
00	0
01	1
10	2
11	3

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

Overflow: Signed

In all cases it occurs do to a “shortage of bits”

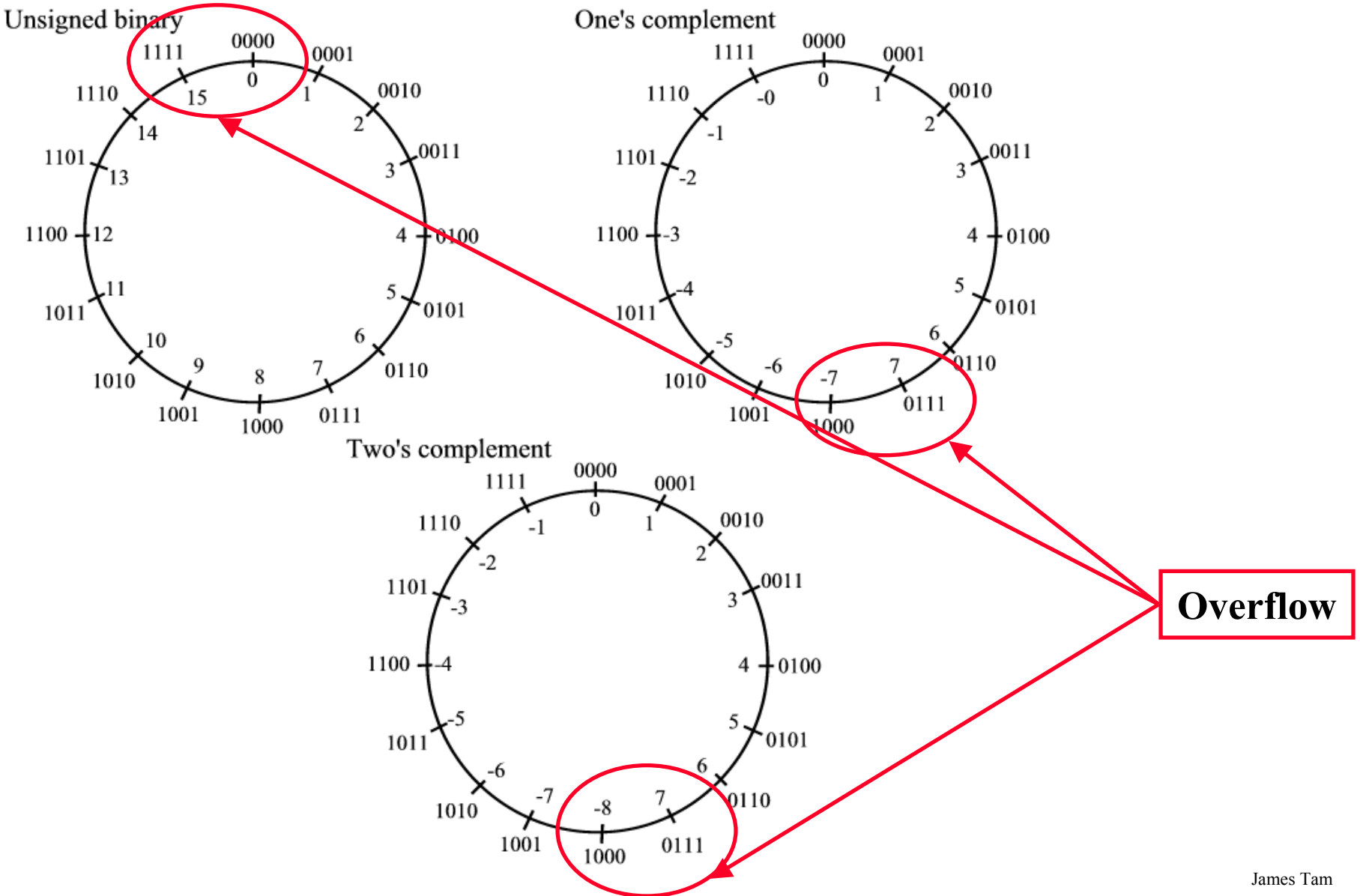
Subtraction – subtracting two negative numbers results in a positive number.

e.g. - 7
- 1
\+ 7

Addition – adding two positive numbers results in a negative number.

e.g. 7
\+ 1
\- 8

Summary Diagram Of The 3 Binary Representations



James Tam

Binary Subtraction

Unsigned binary subtraction

e.g., 1000_2

$- \underline{0010_2}$

0110_2

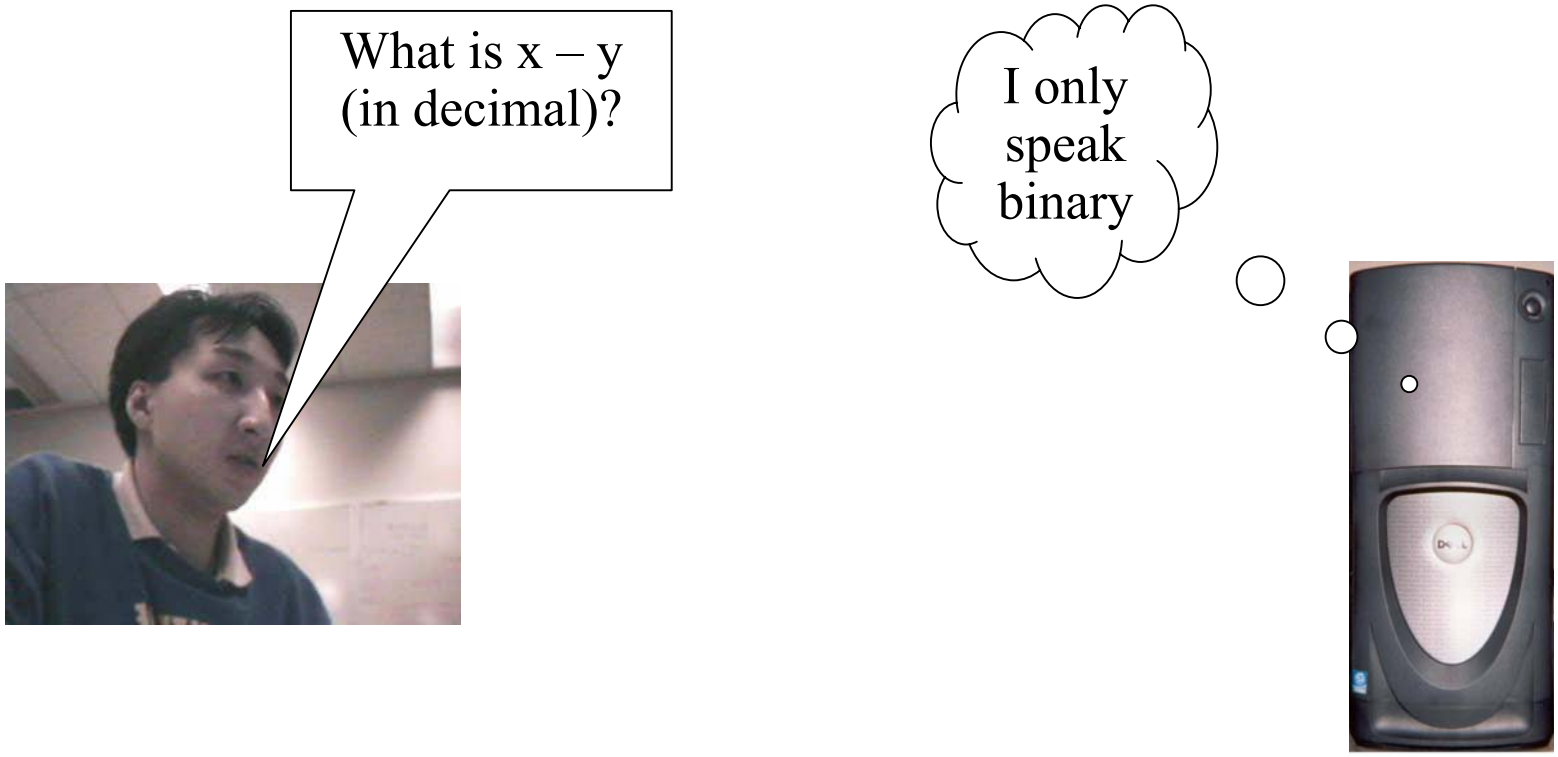
Subtraction via complements (negate and add)

A - B

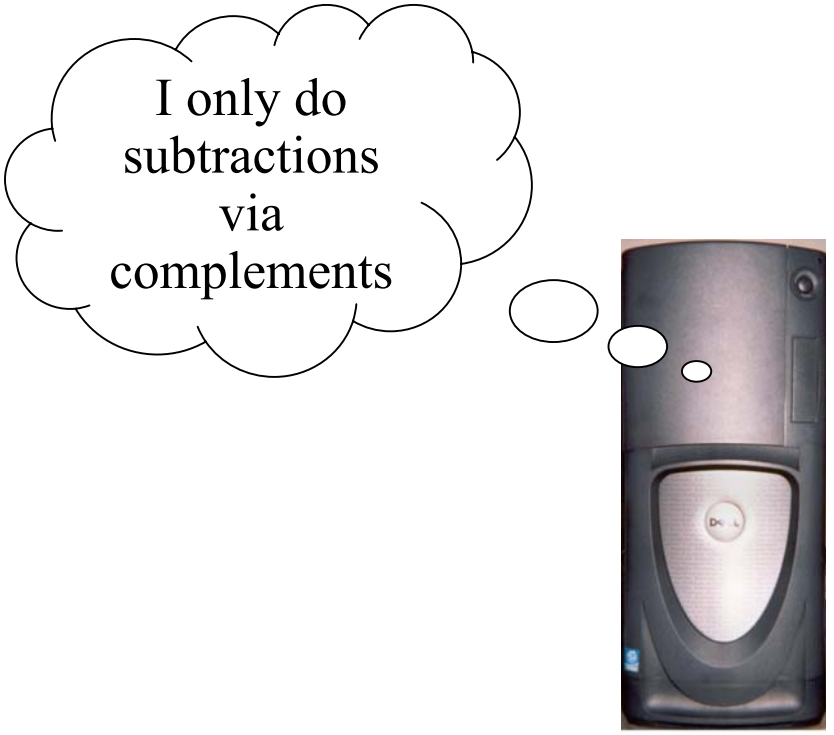
Becomes

A + (-B)

Binary Subtraction Via Negate And Add: A High-Level View



Binary Subtraction Via Negate And Add: A High-Level View



Binary Subtraction Via Negate And Add: A High-Level View

1) Convert the decimal values to unsigned binary

2) Convert the unsigned binary values to complements

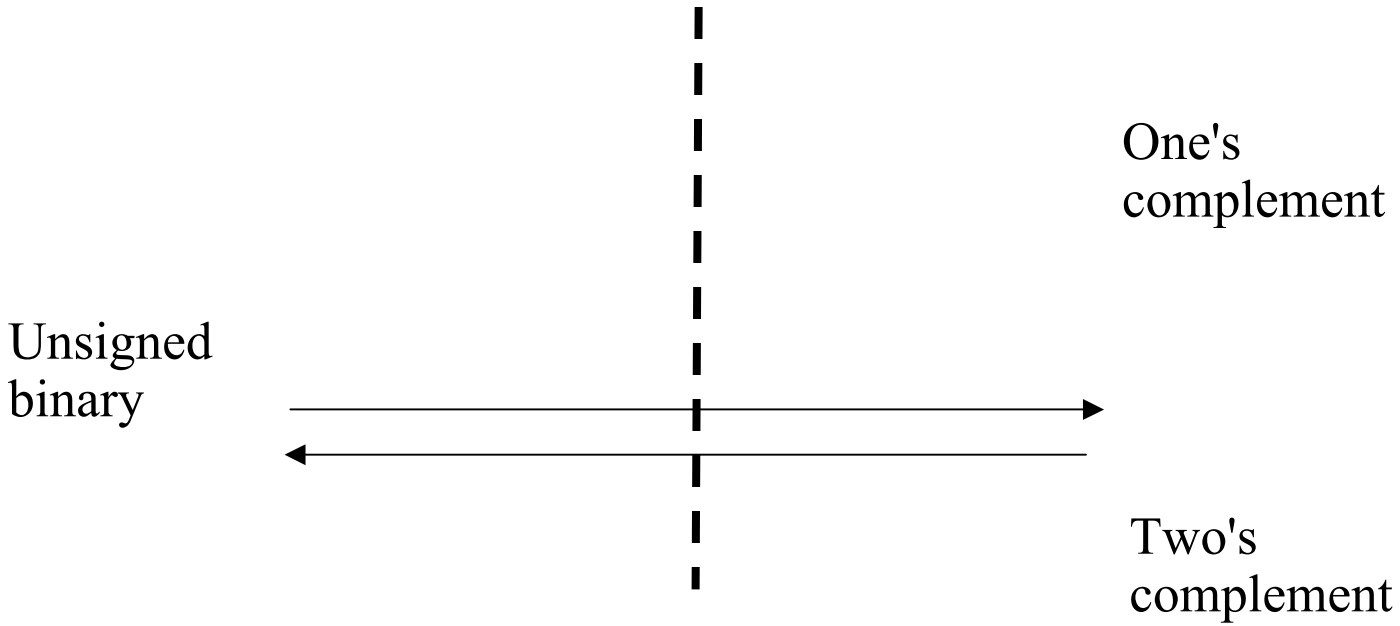
3) Perform the subtraction via negate and add

4) Convert the complements to decimal values

5) Convert the unsigned binary values to decimal

This section

Crossing The Boundary Between Unsigned And Signed



Each time that this boundary is crossed (steps 2 & 4) apply the rule:

1) Positive numbers pass unchanged

2) Negative numbers must be converted

Binary Subtraction Through 1's Complements

1) Negate any negative numbers (flip the bits) to *convert to 1's complement*

2) Add the two binary numbers

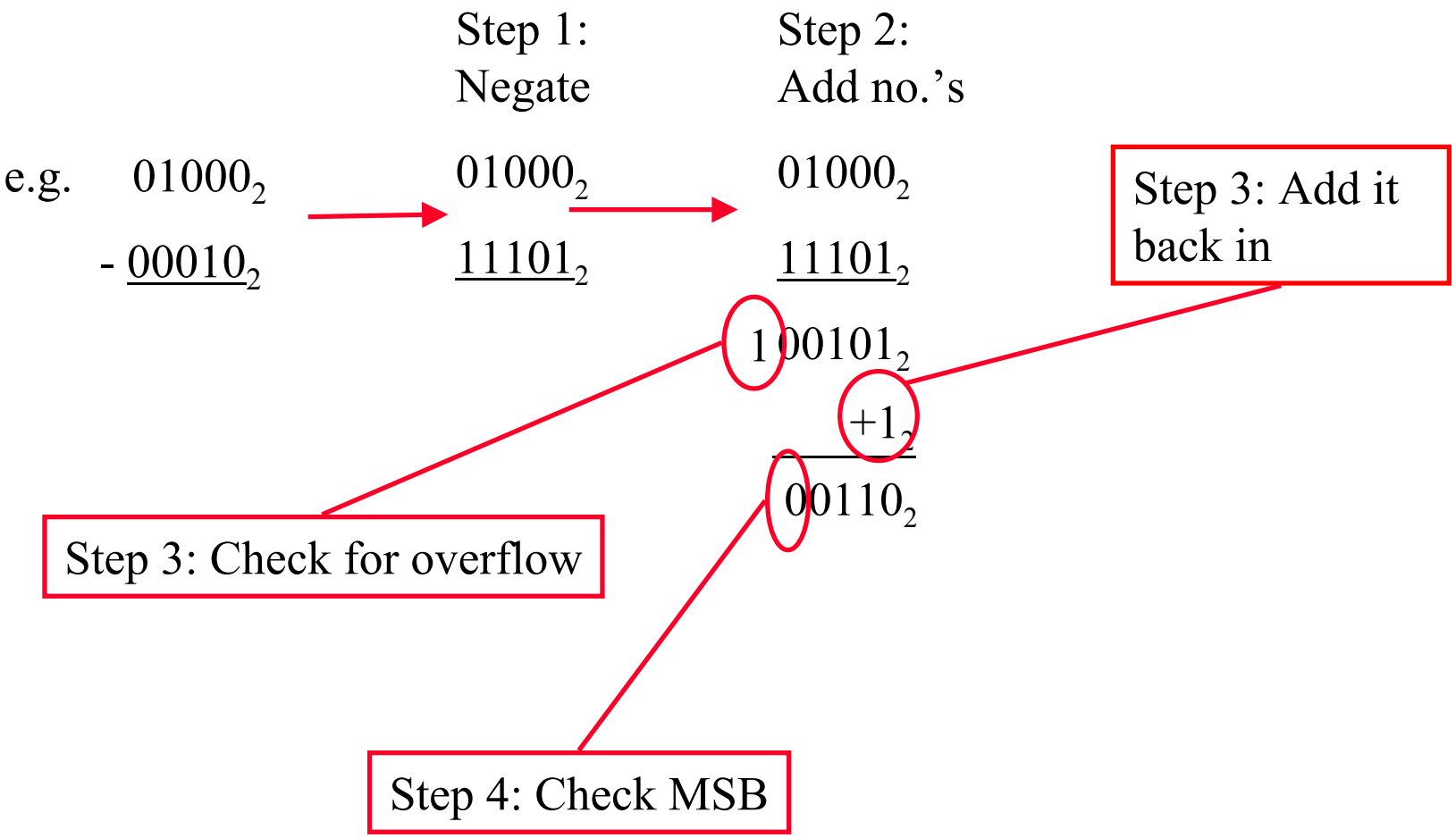
3) Check if there is overflow (a bit is carried out) and if so add it back.

4) Convert the 1's complement value back to unsigned binary (check the value of the MSB)

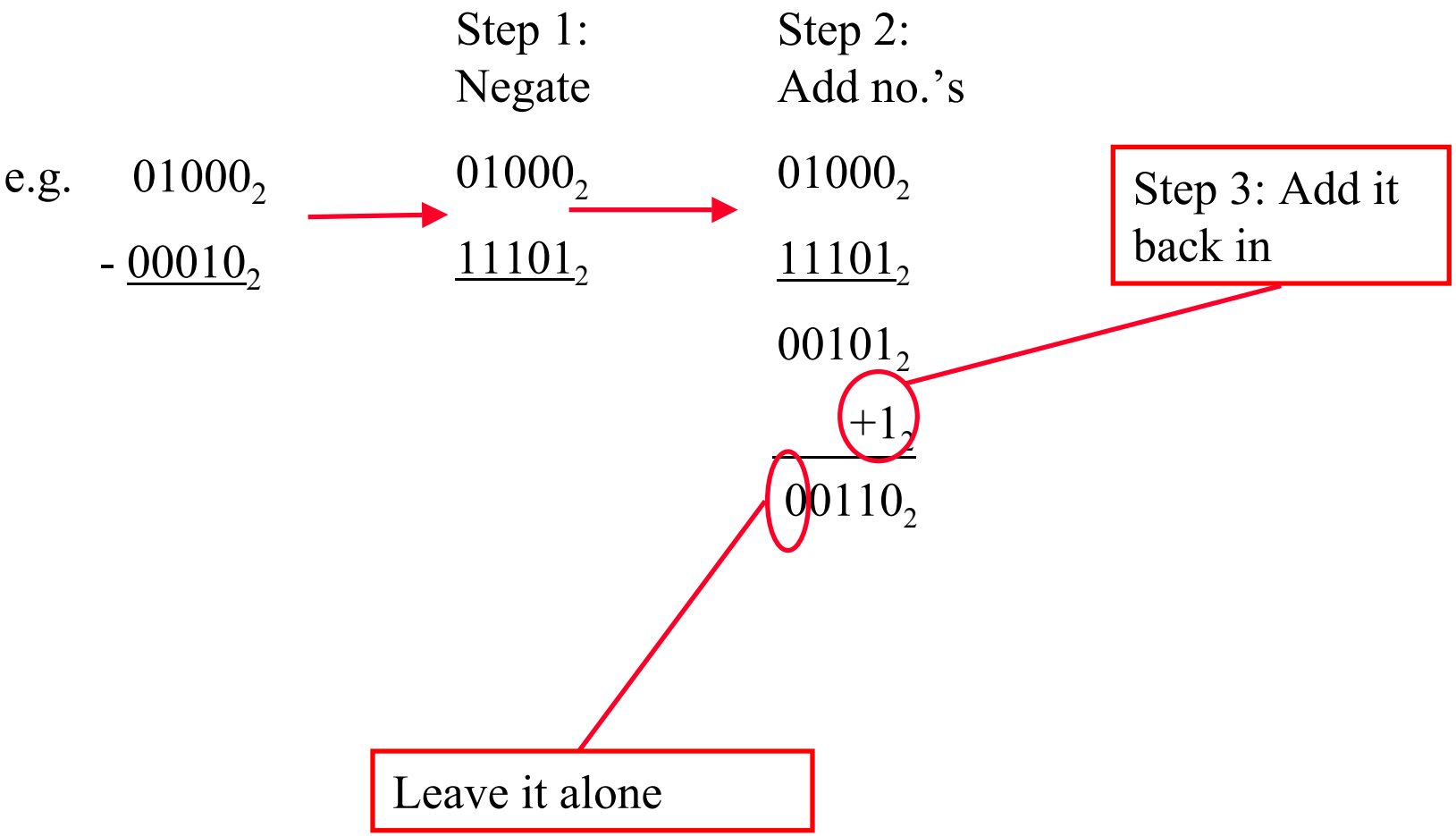
a) If the MSB = 0, the number is positive (leave it alone)

b) If the MSB = 1, the number is negative (flip the bits) and precede the number with a negative sign

Binary subtraction through 1's complements



Binary subtraction through 1's complements



Binary subtraction through 2's complements

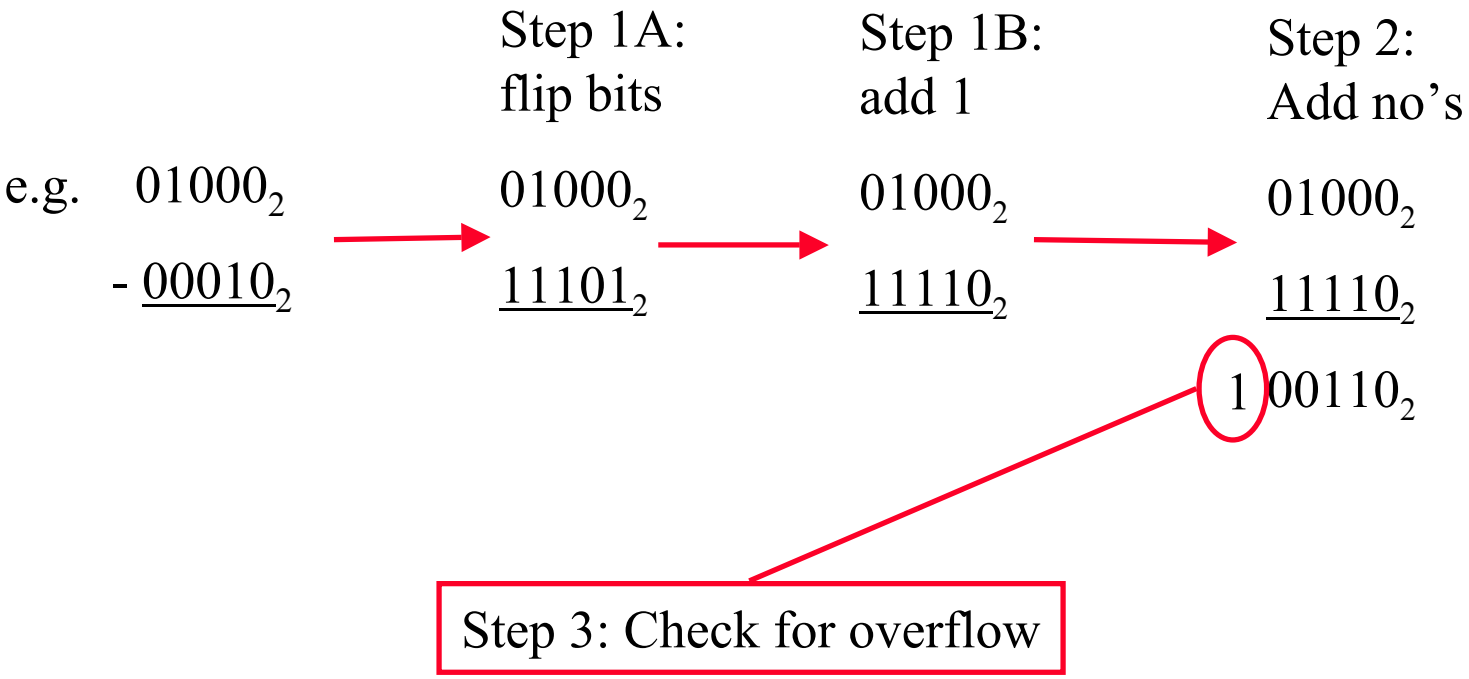
1) Negate any negative numbers to *convert from unsigned binary to 2's complement values*
   a) Flip the bits.
   b) Add one to the result.

2) Add the two binary numbers

3) Check if there is overflow (a bit is carried out) and if so discard it.

4) *Convert the 2's complement value back to unsigned binary* (check the value of the MSB)
   a) If the MSB = 0, the number is positive (leave it alone)
   b) If the MSB = 1, the number is negative (flip the bits and add one) and precede the number with a negative sign

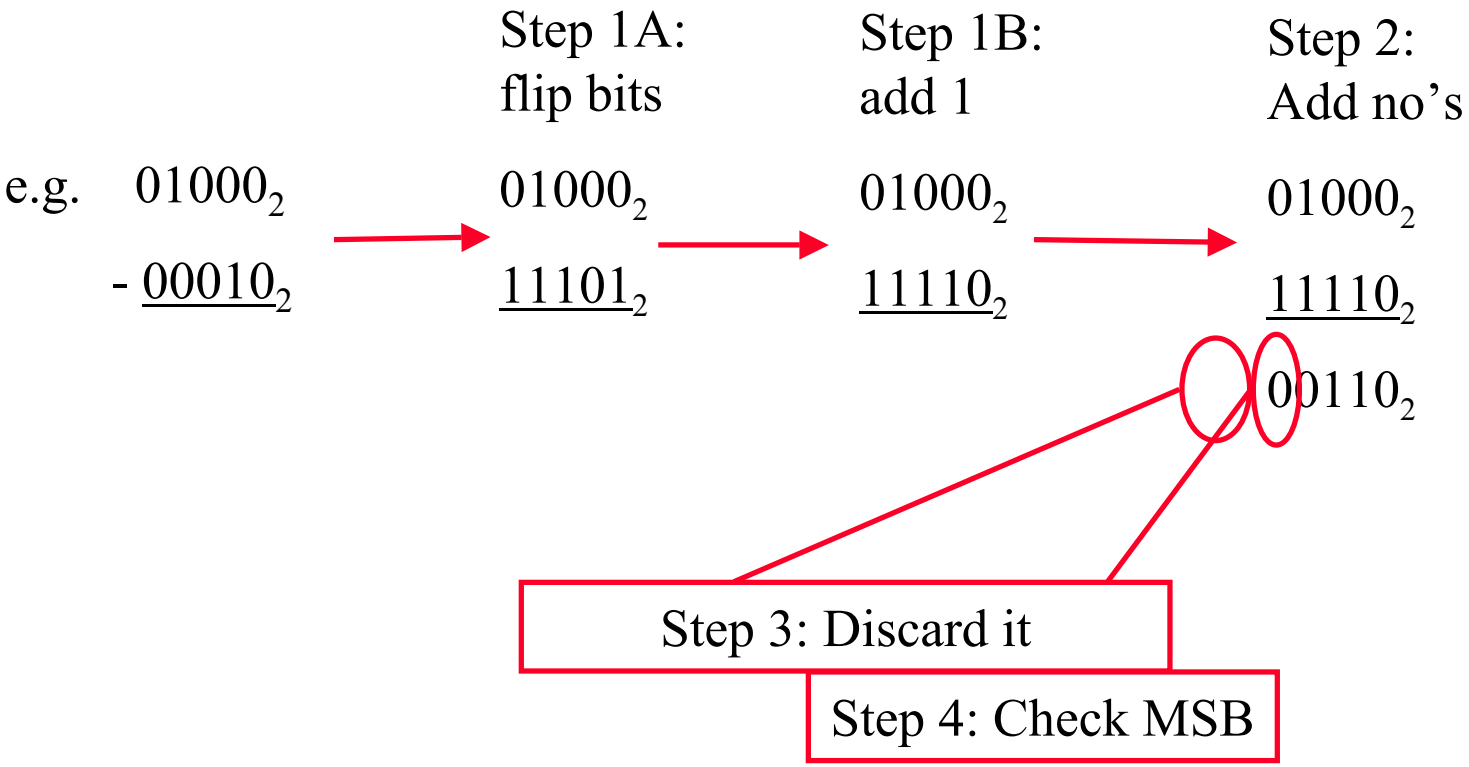
Binary subtraction through 2's complements



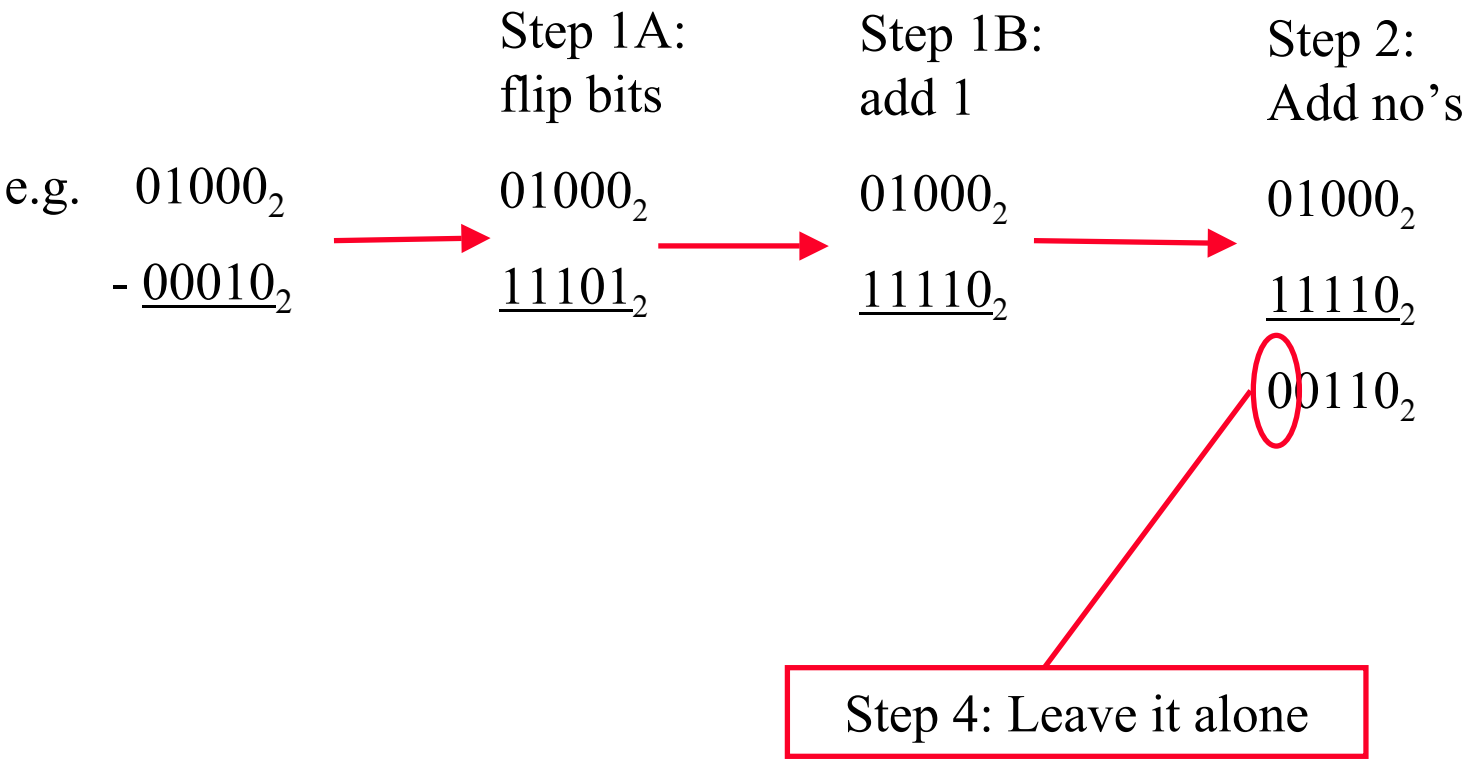
Binary subtraction through 2's complements

		Step 1A: flip bits	Step 1B: add 1	Step 2: Add no's
e.g.	01000_2	01000_2	01000_2	01000_2
	$-\ \underline{00010_2}$	$\underline{11101_2}$	$\underline{11110_2}$	$\underline{11110_2}$
				$1\ 00110_2$

Binary subtraction through 2's complements

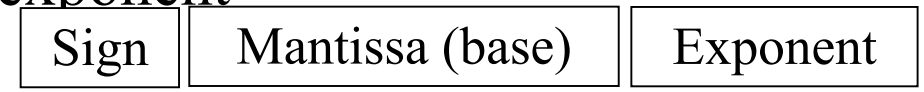


Binary subtraction through 2's complements



Representing Real Numbers Via Floating Point

Numbers are represented through a sign bit, a base and an exponent



Examples with 5 digits used to represent the mantissa:

- e.g. One: 123.45 is represented as $12345 * 10^{-2}$
- e.g. Two: 0.12 is represented as $12000 * 10^{-5}$
- e.g. Three: 123456 is represented as $12345 * 10^{1}$

Floating point numbers may result in a loss of accuracy!

Summary

How negative numbers are represented using 1's and 2's complements

How to convert unsigned values to values into their 1's or 2's complement equivalent

What is meant by overflow

How to perform binary subtractions via the negate and add technique.

How are real numbers represented through floating point representations