The Two's Complement System

Introduction

The *two's complement* system is a special binary system in which any number, positive or negative, has a unique binary code representation. In addition, the representations for negative numbers are <u>not</u> just positive number representations with minus signs in front; each negative number is represented by a specially derived binary code. Because of this special property, addition and subtraction can be performed solely by addition. This makes the 2's complement system very common in modern computers and calculators.

Signed Magnitude

Before the 2's complement system is introduced, a different system called signed magnitude will be discussed to emphasize the concept of sign representation in a computer. In the signed magnitude system, if a binary number is positive, a "0" is placed in the Most Significant Bit (MSB) of the binary number. The MSB is the most left hand bit of a binary number. If a number is negative, a "1" is placed in the MSB. For example,

Base 10	Base 2	Signed Magnitude	
+5	+101	0101	
		↖	Denotes positive number
-5	-101	1101	
		↖	Denotes negative number

This system appears to be a convenient method of handling signs (and is quite similar to our understanding of the base 10 system), but for a simple computation as:

$$\begin{array}{lr} +5 & 0101 \\ \underline{-6} & \underline{+\ 1010} \end{array}$$

a computer (or a human) must:

1. Identify that a negative and positive number are being added
2. Identify the larger number (absolute value)
3. Put this larger number on top
4. Subtract
5. If the larger number (absolute value) is negative, make the answer negative. If the larger number (absolute value) is positive, make the answer positive.

Is this not how you would do it? There is an easier way – 2's complement!

The Complement Idea

To understand how negative numbers may be represented in a computer, let's look at an automobile odometer. Traveling forward adds miles to the odometer (addition), while traveling backwards subtracts miles from the odometer (subtraction). If a vehicle is driven backwards, the following might happen:

```
00004     - Starting value
00003
00002
00001
00000
99999
99998
99997
```

It seems that 99997 represents –3. And if we add any two numbers using this representation, such as (+4) and (-3), we get:

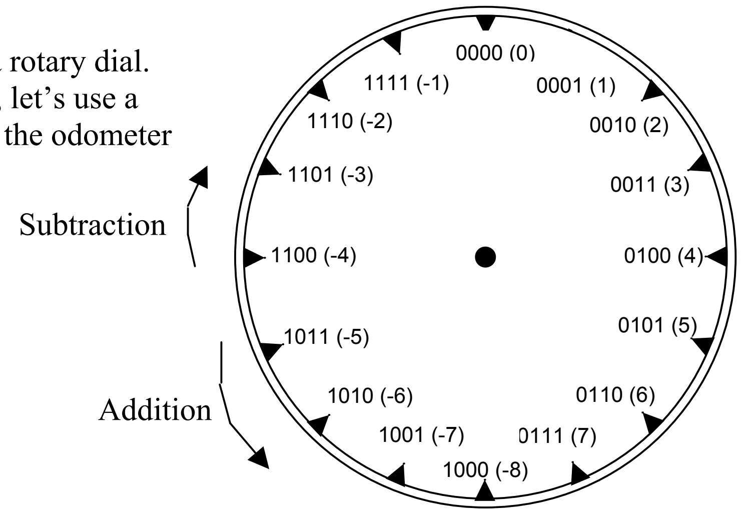
$$\begin{array}{rr} (+4) & 00004 \\ + \; (-\;3) & \underline{+99997} \\ & \underset{\nwarrow \text{Carry}}{1}\,00001 \end{array}$$

Ignoring the "carry", we have <u>subtracted</u> 3 from 4 by <u>adding</u> the "complement of 3" to 4. This is the idea behind complement arithmetic.

The Binary Complement Idea

Instead of the odometer, let's use a rotary dial. And, since we are interested in computers, let's use a binary system. A rotary dial analogous to the odometer might look as seen here.

On the dial, twisting the dial counterclockwise performs addition. Likewise, subtraction is performed by twisting the dial clockwise. Starting at 0, we may add 4 (twist clockwise 4 steps), then subtract 3 (counterclockwise 3 steps) and read the answer of +1 at the top of the dial. Or beginning at zero again, if we add 5 then subtract 6 the answer at the top of the dial is –1. In this manner, we can add and subtract any 4 bit number on the dial.



These computations may also be performed using binary addition. Adding +4 (equivalent to 0100 on the dial) to –3 (1101 on the dial) the result is:

$$\begin{array}{lrl}
+4 & 0100 & +4 \text{ on our dial} \\
(-3) & \underline{+1101} & -3 \text{ on our dial} \\
 & 1\ 0001 & +1 \text{ on our dial} \\
 & \nwarrow\!\!\longleftarrow\!\!\!-\!\!\!-\!\!\!- & \text{Carry (ignore)}
\end{array}$$

If we ignore the carry, our answer is correct.

Another example,

$$\begin{array}{lrl}
+5 & 0101 & +5 \text{ on our dial} \\
(-6) & \underline{+1010} & -6 \text{ on our dial} \\
 & 1111 & -1 \text{ on our dial}
\end{array}$$

It would thus appear that we can perform subtraction by adding. The appearance is, in fact, correct.

The Two's Complement System

The system on the dial is a 4-bit two's complement system. Of the 16 combinations (2^4), we have used half for positive numbers (0 to +7) and half for negative numbers (-1 to –8). Notice that an unsigned binary system would use all 16 combinations as positive numbers (0 to 15). Be aware that zero is treated as a positive number in the twos complement system. The table below shows the unique reorganization:

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

Notice from the table that it is the negative numbers in the 2's complement system that have a peculiar representation — the positive numbers are not affected.

This reorganization was not done in a haphazard manner. Similar to Signed Magnitude Numbers, all 2's complement positive numbers have a "0" in the MSB position AND all 2's complement negative numbers have a "1" in the MSB. But, unlike signed magnitude numbers, you may NOT form a negative 2's complement number by simply placing a "1" in the MSB position. The proper conversion method will be shown shortly.

Range and Overflow

The 4 bit system in Table 2 is limited to 16 combinations (2^4) with eight positive numbers (0 to +7) and eight negative numbers (-1 to –8). We may increase the number of bits to encompass more numbers. If we use 5 bits, then 32 combinations (2^5) are possible. Since half will be positive and half negative, we end up with 16 positive numbers (0 to 15) and 16 negative numbers (-16 to –1). A formal equation for the range of a 2's complement system with n bits is:

$$-\left(\frac{2^n}{2}\right),\ldots,-1,0,+1,\ldots,+\left(\frac{2^n}{2}-1\right)$$

One possible problem encountered using the 2's complement system is an answer that is "out of range." For example, using our 4 bit dial system,

+6	0110	+6 in our 2's complement system
+6	$\underline{+0110}$	+6 in our 2's complement system
	1100	- 4 in our 2's complement system

The answer should be +12, but +12 is not within the range of our system. We (or a circuit in a computer) can quickly recognize this problem by noting that a positive number plus a positive number should not equal a negative number (a "1" appeared in the MSB position of our answer). This problem is called "overflow" and can only be corrected by re-performing the computation with 5 bits instead of 4 (try it).

Overflow has also occurred if two negative numbers are added and produce a positive result. For example,

- 7	1001	- 7 in our 2's complement system
- 7	$\underline{+1001}$	- 7 in our 2's complement system
	1 0010	+2 in our 2's complement system
	$\nwarrow$——— Carry (ignore)	

Two negative numbers added together should not equal a positive number. Notice the addition of a positive number and a negative number will never produce overflow – can you explain why not?

Converting To and From the 2's Complement System

The 4 bit rotary dial has been used thus far to find positive and negative numbers. Notice that positive 2's complement numbers are just base 2 numbers with a "0" in the MSB position. But to find the negative number representation, the following steps must be performed:

1. Start with a negative number (positive numbers don't need to follow this procedure)
2. Find the base 2 representation of the absolute value of the number
3. Complement all bits. That is, all "0"s become "1"s and all "1"s become "0"s.
4. Add 1 to the result

For example, find the 2's complement representation of -4_{10} in a 3, 4, 5 and 6 bit system.

	Number of bits in the system			
	3 bits	4 bits	5 bits	6 bits
1. Start with negative number	-4	-4	-4	-4
2. Find base 2 of absolute value	Out of Range	0100	00100	000100
3. Complement all bits		1011	11011	111011
4. Add 1		1100	11100	111100

Verify the -4_{10} is equal to 1100 using the rotary dial for a 4 bit system. Also notice that extra leading "0"s may be added to a positive number to increase the number of bits. These extra bits translate into leading "1"s in the resultant negative number.

To convert the other way (from a 2's complement negative number to its decimal equivalent), follow a similar procedure. Given a negative 2's complement number, complement (flip) all the bits and add 1. The result is the absolute value of the negative decimal number.

For example, given a 2's complement number (1001), find the decimal equivalent.

1) Complement all bits of the negative number.

$1001 \rightarrow 0110$

2) Add 1

$$\begin{array}{r} 0110 \\ +\underline{\quad\ 1} \\ 0111 \end{array} \leftarrow \text{Equals } +7\text{, so our answer is } -7$$

Review of the 2's Complement Mechanics

1. Positive numbers always have a "0" in the MSB position.
2. Negative numbers always have a "1" in the MSB position.
3. Carry out of MSB is always discarded
4. Overflow occurs in only 2 situations
   (Positive number) + (Positive number) results in (Negative number).
   (Negative number) + (Negative number) results in (Positive number).
5. The range of a 2's complement system with n bits is: $-\left(\frac{2^n}{2}\right),\ldots,-1,0,+1,\ldots,+\left(\frac{2^n}{2}-1\right)$
6. You cannot tell that a number is 2's complement merely by inspection. You must be told.
7. To find the 2's complement representation of a number:
   a. If the number is positive, convert it to base 2 and stop.
   b. Find the base 2 representation of the absolute value
   c. Complement (flip) all bits.
   d. Add 1
8. To find the decimal equivalent of a 2's complement number:
   a. If the number is positive, convert to base 10 and stop.
   b. Complement (flip) all bits.
   c. Add 1.
   d. Convert to base 10. Put a minus sign in front.

Examples Using 2's Complement

1. Add 7 and –3 using 4 bits:

```
      +0111   =     0111
  +  - 0011   =    +1101
                   1 0100   =   +4
                   ^-------- Carry (ignore)
```

2. Add –7 and +3 using 4 bits:

```
      - 0111  =     1001
  +  +0011    =    +0011
                    1100    =   -(0100)   =   -4
```

3 a. Add 15 and 2 using 5 bits:

```
     01111
  +  00010
     10001
     ^-------- Overflow!!
```

b. Add 15 and 2 using 6 bits:

$$\begin{array}{r} 001111 \\ \underline{+\ 000010} \\ 010001 \end{array}$$

$\nwarrow$ No Overflow

4 a. Add –2 and –3 using 3 bits:

$$\begin{array}{rcr} -\ 010 & = & 110 \\ +\ -\ 011 & = & \underline{+\,101} \\ & & 1\ 011 \end{array}$$

Carry $\nearrow$ $\nwarrow$ Overflow!!

b. Add –2 and –3 using 4 bits:

$$\begin{array}{rcrcccc} -\ 0010 & = & 1110 & & & & \\ +\ -\ 0011 & = & \underline{+\,1101} & & & & \\ & & 1\ 1011 & = & -(0101) & = & -5 \end{array}$$

Carry $\nearrow$

Conclusion

Because the 2's complement system has a special representation for negative numbers, both addition and subtraction can be performed merely with addition. This property, which eliminates the need for special subtraction hardware, makes the use of 2's complement arithmetic ubiquitous on modern computers and calculators.

Reward

For those who have actually read this far, the following simple technique for finding the 2's complement of a number is offered as a reward. Use EITHER this procedure or the one where you invert all the bits and add 1. DO NOT CONFUSE THE TWO PROCEDURES

To find the 2's complement of a number, complement all bits <u>after</u> the first "1" bit when moving from right to left. Leave all other bits unchanged.

Example: $\underbrace{10110}_{\text{Complement each bit}}\underset{\uparrow \text{ First "1" bit}}{1}0$ 2's complement = $01001\underbrace{10}_{\text{Unchanged bits}}$