- 8-bit Numbers

- The Negative of N, written as $-N_2$, is computed from

$$2^8 - N_2 = 1\,0000\,0000_2 - N_2$$

- Example:

$$\begin{aligned} N_2 &= 0000\,0011_2 \\ -N_2 &= \quad 1\,0000\,0000_2 \\ & \quad -0000\,0011_2 \\ \hline -N_2 &= \quad 1111\,1101_2 \end{aligned}$$

- **Definition:** The negative of N is called the **two's complement of N** and is denoted by **2comp(N)**

- $\boxed{\text{Math Tip:}}$
- When calculating $2comp(N_2)$ use this "short-cut"
- 8-Bits

$$\begin{aligned} 2comp(N_2) &= 2^8 - N_2 \\ &= ((2^8 - 1) - N_2) + 1 \\ &= (1111\,1111 - N_2) + 1 \end{aligned}$$

- This is a shortcut because

$$1111\,1111 - N_2$$

can be calculated by simply replacing each 0 by 1, and each 1 with 0 in N_2. (You don't have to worry so much about borrows).

- Two Steps in Calculating 2comp(N)

1. Complement N_2 ; also called $1comp(N_2)$
2. Add 1

N-bit Two's Complement Numbers

1. Two's complement n-bit numbers are binary numbers of length n. Bits are indexed from 0 to $n-1$. For each bit d_i, the subscript corresponds to the power of 2 in the binary expansion.
2. **Range:** $-2^{n-1} \ldots (2^{n-1} - 1)$
3. **Sign:**MSB = Sign bit; $1 \rightarrow -ive$, $0 \rightarrow +ive$
4. For an n-bit number M, Negative of M is 2comp(M)
   (a) 2comp(M) = $2^n - M$
   (b) Shortcut: 2comp(M) = 1Comp(M) + 1
5. $\forall$ values of M, $M + 2comp(M) = 0$
   - Except for -2^{n-1}, $2comp(-2^{n-1}) = -2^{n-1}$!!!!!!

6. **Magnitude:** The magnitude of a negative number in two's complement representation is hidden. You know it is negative because the MSB = 1, but what is its magnitude?

   - Magnitude of a negative number is calculated by finding its two's complement:
     - example: $M = 10001011_2$
     - M is negative, because MSB = 1
     - What is M's magnitude; M is -ive what?
     - $2comp(M) = 01110100_2 + 1 = 01110101_2 = 117_{10}$
     - Therefore : M is -117_{10}

$\boxed{\text{2's complement Hex numbers}}$

- We can compactly write binary numbers in HEX

Size = 8 bits

2's complement	Decimal value	HEX
0111 1111	127	7F
0111 1110	126	7E
0111 1101	125	7D
.....	.....	.....
0000 0011	3	03
0000 0010	2	02
0000 0001	1	01
0000 0000	0	00
1111 1111	-1	FF
1111 1110	-2	FE
1111 1101	-3	FD
.....	.....	.....
1000 0010	-126	82
1000 0001	-127	81
1000 0000	-128	80

2's complement Hex numbers ...cont'd

16-bit numbers

- example 1: 2comp(06FD)

$2^{16} - 06FD$		SHORTCUT
10000 $-06FD$	OR	$FFFF$ $-06FD$
		$F902$ $+0001$
$F903$		$F903$

- example 2: 2comp(903B)

$2^{16} - 903B$		SHORTCUT
10000 $-903B$	OR	$FFFF$ $-903B$
		$6FC4$ $+0001$
$6FC5$		$6FC5$

$\boxed{\text{Answers you should know}}$

- What's the two's complement representation of $-9C8$?
- How many integers (positive and negative) can we represent in n bits using the 2's complement representation?
- Conversely, if we want to represent the integers between $-N\ to\ N$, what's the minimum number of bits that we need?

SIGN EXTENSION

Decimal	3-Bit	4-Bit	8-Bit
0	000	0000	0000 0000
1	001	0001	0000 0001
2	010	0010	0000 0010
3	011	0011	0000 0011
-1	111	1111	1111 1111
-2	110	1110	1111 1110
-3	101	1101	1111 1101
-4	100	1100	1111 1100

- **Definition:** Let X be a 2's complement number with fixed length M. The sign extension of X with fixed length N ($N > M$) replicates the higher-order bit of X in its high-order (N - M) bits.

- $\boxed{\text{Examples}}$
- **Binary:** 8-bit number to 16-bit number
  1. 1010 1100 $\rightarrow$ 1111 1111 1010 1100
  2. 0001 1100 $\rightarrow$ 0000 0000 0001 1100
- **Hex:** 8-bit to 16-bit
  1. $9B \rightarrow FF\ 9B$

     $9B$ = 1001 1011; MSB is 1; sign extension 1111 1111 10011011 = $FF\ 9B$
  2. $5F \rightarrow 00\ 5F$
  3. $7A \rightarrow 00\ 7A$
  4. $87 \rightarrow FF\ 87$
  5. $A5 \rightarrow FF\ A5$
  6. $D6 \rightarrow FF\ D6$
  7. $36 \rightarrow 00\ 36$
  8. $6C \rightarrow 00\ 6C$

$\boxed{\text{Addition and Subtraction of 2's complement}}$

- **Addition Rule:** Add two terms using straight binary addition. DISCARD any final carry.
- **Subtraction Rule:** $N - M = N + 2comp(M)$

- Addition

$$
\begin{array}{rr}
 & -2 \\
+ & 3 \\
\hline
 & 1
\end{array}
\qquad
\begin{array}{rr}
 & 1111\,1110 \\
+ & 0000\,0011 \\
\hline
carry = 1 & 0000\,0001 \\
 & \\
result & 0000\,0001
\end{array}
$$

$$
\begin{array}{rr}
 & -1 \\
+ & -1 \\
\hline
 & -2
\end{array}
\qquad
\begin{array}{rr}
 & 1111\,1111 \\
+ & 1111\,1111 \\
\hline
carry = 1 & 1111\,1110 \\
 & \\
result & 1111\,1110
\end{array}
$$

- Subtraction

$$\begin{array}{rrrr} & 1 & & 1 \\ - & 2 & + & -2 \\ \hline & -1 & & -1 \end{array} \qquad \begin{array}{rr} & 0000\,0001 \\ + & 1111\,1110 \\ \hline carry = 0 & 1111\,1111 \\ \\ result & 1111\,1111 \end{array}$$

$$\begin{array}{rrrr} & 8 & & 8 \\ - & 7 & + & -7 \\ \hline & 1 & & 1 \end{array} \qquad \begin{array}{rr} & 0000\,1000 \\ + & 1111\,1001 \\ \hline carry = 1 & 0000\,0001 \\ \\ result & 0000\,0001 \end{array}$$

Two's complement Hex operations

- A5B6 + 3A94 = ?
- 803E + A74B = ?
- D67A + 80D7 = ?

- 30FC - 0011 = ?
- 5F3C - A0F4 = ?

OVERFLOW

- Signed Overflow
- $3000_h + 6000_h = 9000_h$; (+ive + +ive = -ive) ???

  The sum of these +ive numbers exceeds the upper limit of 16bit 2's complement positive numbers $7FFF_h$ → **Signed Overflow**

- Unsigned Overflow
- $6000_h + D000_h = 3000_h$ Adding two +ive numbers results in a smaller number ???

  Sum exceeds $FFFF_h$ (largest 16bit unsigned number) → **Unsigned Overflow**

- Checking for Signed Overflow
  - Addition:
    1. OCCURS When sum of two numbers of like sign has a result with the Opposite sign.
    2. Never occurs when adding two numbers of opposite sign
  - Subtraction: M - N = M + 2comp(N)
    1. -ive - +ive = +ive
    2. +ive - -ive = -ive
    3. Never Occurs when subtracting two numbers of the same sign.

  50A3 + 6A38 =

  3826 - 7000 =

  38A3 + A330 =

  C839 - 7000 =

- NOTE: The Microprocessor checks for signed overflow and flags it. You must be aware of this and insert code to respond to the possibility of overflow!

- Signed overflow carry test
- Two's complement numbers: M, N

  M + N produces signed overflow iff the carry **into** the high-order bit is **not equal** to the carry **out** of the high-order bit

$\boxed{\text{Representing Characters}}$

- **ASCII:** American Standard Code for Information Interchange
- EBCDIC: Extended Binary Coded Decimal Interchange Code (IBM)
- ASCII: Fit in One byte but is a 7-bit code
- $2^7 = 128_{10}$ characters
- 95 printable characters
- 33 control characters for communications and peripheral interface controls
- A $= 41_h$; Z $= 5A_h$; a $= 61_h$; z $= 7A_h$
- Adding $32_{10} = 20_h$ converts from Upper case to Lower case for the letters of the alphabet