

Binary

- base 2: 0,1 (true/false) (on/off)
- ... 2^3 2^2 2^1 2^0 . 2^{-1} 2^{-2} 2^{-3} ...
... 8 4 2 1 . 1/2 1/4 1/8 ...
- binary → decimal:
 - $1001_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
 $= 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1$
 $= 9_{10}$
 - $1101_2 = x_{10}$?
- decimal → binary
 - $19_{10} = ?_2$
 $19/2 = 9$ remainder 1
 $9/2 = 4$ remainder 1
 $4/2 = 2$ remainder 0
 $2/2 = 1$ remainder 0
 $1/2 = 0$ remainder 1
 $\rightarrow 19_{10} = 10011_2$
 - $42_{10} = x_2$?

Negative Integers

- positive numbers (one byte):
 - 0000 0000 0
 - 0000 0001 1
 - 0000 0010 2
 - ...
 - 1111 1111 255_{10}
- how can we represent negative numbers? e.g. -42_{10} ?
 $42_{10} = 0001\ 0101_2$, but the negative?
 - reserve one bit for sign (highest bit, most significant bit)
 - positive numbers: as above but largest is:
 $0\ 111\ 1111_2\ 127_{10}$
 - negative numbers:
 $1\ 000\ 0000\ -0$
 $1\ 000\ 0001\ -1$
 ...
 $1\ 111\ 1111_2\ -127_{10}$
 - problems: two zeros (!), addition is not simple: $4 + (-1) = -5$

Binary Integer Addition and Subtraction

- as per decimal from right to left, propagating 'carries'
- addition:

11010_2	+	26_{10}	+
<u>01011_2</u>	=	<u>11_{10}</u>	=
100101_2		37_{10}	
- subtraction:

10101_2	-	21_{10}	-
<u>01011_2</u>	=	<u>11_{10}</u>	=
01010_2		10_{10}	

Two's Complement Representation of Signed Integers

- most common system
 - only one zero; left-most bit indicates sign
 - normal binary addition gives correct result
- rule: to negate a number, flip the bits to the left of the right-most 1
 - equivalently: flip all the bits and add 1 to the result
- examples:

$0000\ 0101$	5	$1111\ 1000$	-8
$1111\ 1011$	-5	$0000\ 1000$	8
- addition and subtraction (for 4-bit arithmetic: discard 5th bit, if any):

0100	4 +	1101	-3 +
<u>1111</u>	-1 =	<u>1110</u>	-2 =
10011	3	11011	-5

0111	7 +	1010	-6 +
<u>0010</u>	2 =	<u>1010</u>	-6 =
1001	-7 (or 9?)	10100	4 (or -12?)

