

01001110010000110101001101010101

Number Systems

4e435355

1,313,035,093

ECGR2181
Lecture Notes 1A

1.525 × 2²⁹

These are all different interpretations of the same bit string.

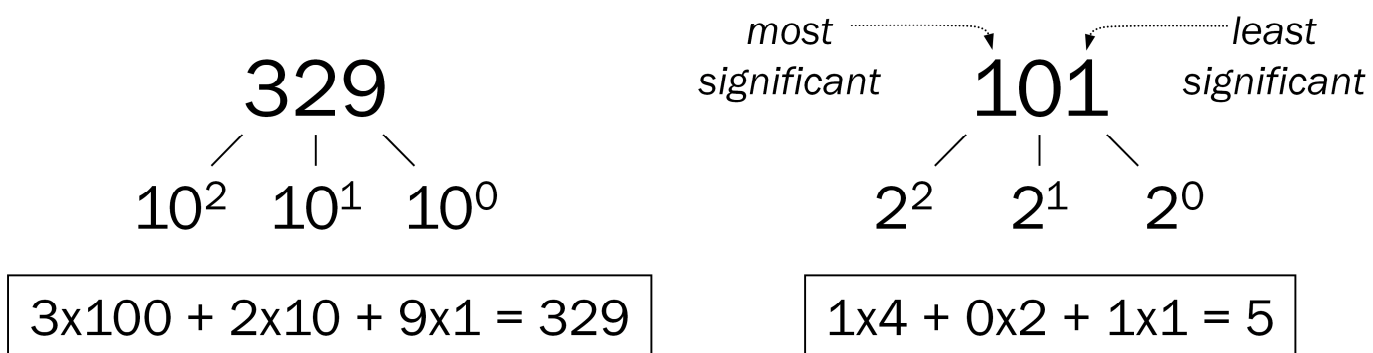
Unsigned Integers

Non-positional notation

- could represent a number (“5”) with a string of ones (“11111”)
- problems?

Weighted positional notation

- like decimal numbers: “329”
- “3” is worth 300, because of its position, while “9” is only worth 9



Problems with non-positional: (1) large numbers require lots of bits, (2) arithmetic is not easy.
Positional: compact, simple arithmetic.

Unsigned Binary Arithmetic

Base-2 addition – just like base-10!

– add from right to left, propagating carry

$$\begin{array}{r} 10010 \\ + 1001 \\ \hline 11011 \end{array}$$
$$\begin{array}{r} \text{carry} \\ \curvearrowright \\ 10010 \\ + 1011 \\ \hline 11101 \end{array}$$
$$\begin{array}{r} \curvearrowleft \curvearrowleft \curvearrowleft \curvearrowleft \\ 1111 \\ + 1 \\ \hline 10000 \end{array}$$

Unsigned Binary Arithmetic Practice

Base-2 addition – just like base-10!

- add from right to left, propagating carry

$$\begin{array}{r} 10111 \\ + \underline{1111} \end{array}$$

$$\begin{array}{r} 10111 \\ - \underline{1111} \end{array}$$

Subtraction, multiplication, division,...

Odometer numbers

Consider an odometer of a car at a location on a street:

0	0	1	3
---	---	---	---

 Go 3 miles in reverse and it reads:

0	0	1	0
---	---	---	---

- ◆ Same as “subtracting 3” or “adding -3”
- ◆ What happens when...

0	0	0	0
---	---	---	---

 Go 3 miles in reverse and it reads:

9	9	9	7
---	---	---	---

- ◆ As far as the odometer is concerned, $9997 = -3$
- ◆ Note that fixed-width binary is very similar to odometer numbers in its limitations
- ◆ Can the same representation be used?
 - ◇ $0000_2 - 0001_2 = “1111_2”$ or -1
 - ◇ $0000_2 - 0011_2 = “1101_2”$ or -3
- ◆ This is called “2’s complement”

Two's Complement

Two's complement representation developed to make circuits easy for arithmetic.

- for each positive number (X), assign value to its negative ($-X$), such that $X + (-X) = 0$ with “normal” addition, ignoring carry out

$$\begin{array}{r} 00101 \quad (5) \\ + 11011 \quad (-5) \\ \hline 00000 \quad (0) \end{array}$$

$$\begin{array}{r} 01001 \quad (9) \\ + \quad \quad \quad (-9) \\ \hline 00000 \quad (0) \end{array}$$

To add sign-magnitude numbers:

- (1) if signs are the same, just add magnitudes and preserve sign (ignoring overflow for now)
- (2) if signs are different, subtract smaller magnitude from larger and set sign according to larger

To add one's complement:

Add normally, then increment by carry-out.

Two's Complement Representation

If number is positive or zero,

- normal binary representation, zeroes in upper bit(s)

If number is negative,

- start with positive number
- flip every bit (i.e., take the one's complement)
- then add one

$$\begin{array}{r} \begin{array}{l} \text{00101} \quad (5) \\ \text{11010} \quad (1's \text{ comp}) \\ + \quad \quad \quad \underline{\quad 1} \\ \hline \text{11011} \quad (-5) \end{array} \qquad \begin{array}{l} \text{01001} \quad (9) \\ \text{10110} \quad (1's \text{ comp}) \\ + \quad \quad \quad \underline{\quad 1} \\ \hline \text{10111} \quad (-9) \end{array} \end{array}$$

Common mistake: I say, “What is the two’s complement representation of +5?”

Student takes the two’s complement of +5 (00101) and tells me “11011”.

Two's Complement Signed Integers

MS bit is sign bit – it has weight -2^{n-1} .

Range of an n-bit number: -2^{n-1} through $2^{n-1} - 1$.

- The most negative number (-2^{n-1}) has no positive counterpart.

-2^3	2^2	2^1	2^0		-2^3	2^2	2^1	2^0	
0	0	0	0	0	1	0	0	0	-8
0	0	0	1	1	1	0	0	1	-7
0	0	1	0	2	1	0	1	0	-6
0	0	1	1	3	1	0	1	1	-5
0	1	0	0	4	1	1	0	0	-4
0	1	0	1	5	1	1	0	1	-3
0	1	1	0	6	1	1	1	0	-2
0	1	1	1	7	1	1	1	1	-1

Two's Complement Practice

Show the two's complement representation of the decimal number -6.

Converting Binary (2's C) to Decimal

1. If leading bit is one, take two's complement to get a positive number.
2. Add powers of 2 that have "1" in the corresponding bit positions.
3. If original number was negative, add a minus sign.

$$\begin{aligned} X &= 01101000_{\text{two}} \\ &= 2^6 + 2^5 + 2^3 = 64 + 32 + 8 \\ &= 104_{\text{ten}} \end{aligned}$$

<i>n</i>	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
(K) 10	1024
(M) 20	1048576

Assuming 8-bit 2's complement numbers.

Memorize this table!

More Examples

$$\begin{aligned}
 X &= 00100111_{\text{two}} \\
 &= 2^5 + 2^2 + 2^1 + 2^0 = 32 + 4 + 2 + 1 \\
 &= 39_{\text{ten}}
 \end{aligned}$$

$$\begin{aligned}
 X &= 11100110_{\text{two}} \\
 -X &= 00011010 \\
 &= 2^4 + 2^3 + 2^1 = 16 + 8 + 2 \\
 &= 26_{\text{ten}} \\
 X &= -26_{\text{ten}}
 \end{aligned}$$

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
(K) 10	1024
(M) 20	1048576

Assuming 8-bit 2's complement numbers.

Converting Binary (2's C) to Decimal Practice

Convert binary 00011111 to decimal:

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
(K) 10	1024
(M) 20	1048576

Converting Decimal to Binary (2's C)

First Method: *Division*

1. Divide by two – remainder is least significant bit.
2. Keep dividing by two until answer is zero, writing remainders from right to left.
3. Append a zero as the MS bit;
if original number negative, take two's complement.

$X = 104_{\text{ten}}$	$104/2 = 52 \text{ r}0 \text{ bit } 0$
	$52/2 = 26 \text{ r}0 \text{ bit } 1$
	$26/2 = 13 \text{ r}0 \text{ bit } 2$
	$13/2 = 6 \text{ r}1 \text{ bit } 3$
	$6/2 = 3 \text{ r}0 \text{ bit } 4$
	$3/2 = 1 \text{ r}1 \text{ bit } 5$
$X = 01101000_{\text{two}}$	$1/2 = 0 \text{ r}1 \text{ bit } 6$

Converting Decimal to Binary (2's C)

Second Method: **Subtract Powers of Two**

1. Change to positive decimal number.
2. Subtract largest power of two less than or equal to number.
3. Put a one in the corresponding bit position.
4. Keep subtracting until result is zero.
5. Append a zero as MS bit; if original was negative, take two's complement.

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
(K) 10	1024
(M) 20	1048576

$X = 104_{\text{ten}}$	$104 - 64 = 40$	<i>bit 6</i>
	$40 - 32 = 8$	<i>bit 5</i>
	$8 - 8 = 0$	<i>bit 3</i>
$X = 01101000_{\text{two}}$		

Converting Decimal to Binary Practice

Convert decimal 270 to binary using both methods described above:

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
(K) 10	1024
(M) 20	1048576

More Converting Decimal to Binary Practice

Convert decimal 255 to binary using both methods described above:

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
(K) 10	1024
(M) 20	1048576

Operations: Arithmetic

Recall: a data type includes *representation* and *operations*.

We now have a good representation for signed integers, so let's look at some ***arithmetic*** operations:

- Addition
- Subtraction
- Sign Extension

Addition

As we've discussed, 2's comp. addition is just binary addition.

- assume all integers have the same number of bits
- ignore carry out
- for now, assume that sum fits in n-bit 2's comp. representation

$$\begin{array}{r} 01101000 \text{ (104)} \\ + 11110000 \text{ (-16)} \\ \hline 01011000 \text{ (88)} \end{array} \quad \begin{array}{r} 11110110 \text{ (-10)} \\ + \text{ (-9)} \\ \hline \text{ (-19)} \end{array}$$

Assuming 8-bit 2's complement numbers.

Subtraction

Negate subtrahend (2nd no.) and add.

- assume all integers have the same number of bits
- ignore carry out
- for now, assume that difference fits in n-bit 2's comp. representation

$$\begin{array}{r} 01101000 \text{ (104)} \\ - 00010000 \text{ (16)} \\ \hline \end{array} \qquad \begin{array}{r} 11110110 \text{ (-10)} \\ - \text{ (-9)} \\ \hline \end{array}$$

is just

$$\begin{array}{r} 01101000 \text{ (104)} \\ + 11110000 \text{ (-16)} \\ \hline 01011000 \text{ (88)} \end{array} \qquad \begin{array}{r} 11110110 \text{ (-10)} \\ + \text{ (9)} \\ \hline \text{ (-1)} \end{array}$$

Assuming 8-bit 2's complement numbers.

Could also subtract, with borrows, from left to right.

This way, they only have to learn addition and they're prepared for LC-2, which doesn't have a subtract instruction.

Practice

Perform the Two's Complement operation to the following decimal numbers: - 56 - 14

11001000

11110010 = -14

Sign Extension

To add two numbers, we must represent them with the same number of bits.

If we just pad with zeroes on the left:

<u>4-bit</u>	<u>8-bit</u>
0100 (4)	00000100 (still 4)
1100 (-4)	00001100 (12, not -4)

Instead, replicate the most significant bit -- the sign bit:

<u>4-bit</u>	<u>8-bit</u>
0100 (4)	00000100 (still 4)
1100 (-4)	11111100 (still -4)

Overflow

If operands are too big,
then sum cannot be represented as an n -bit 2's comp
number.

$$\begin{array}{r} 01000 \quad (8) \\ + 01001 \quad (9) \\ \hline 10001 \quad (-15) \end{array} \qquad \begin{array}{r} 11000 \quad (-8) \\ + 10111 \quad (-9) \\ \hline 01111 \quad (+15) \end{array}$$

We have overflow if:

- signs of both operands are the same, and
- sign of sum is different.

Another test -- easy for hardware:

- carry into MS bit does not equal carry out

Hexadecimal Notation

It is often convenient to write binary (base-2) numbers as hexadecimal (base-16) numbers instead.

- fewer digits -- four bits per hex digit
- less error prone -- easy to corrupt long string of 1's and 0's

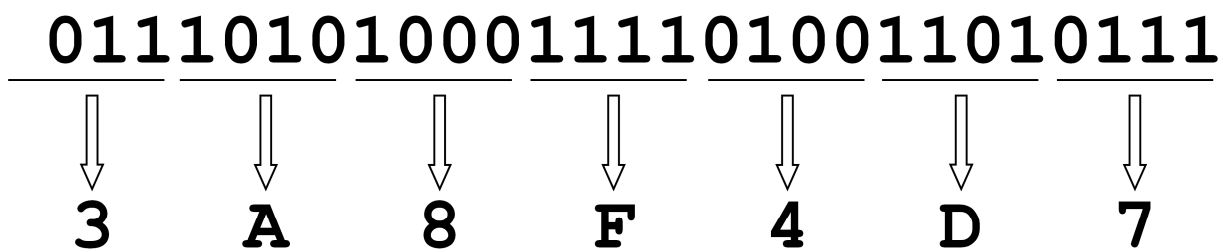
Binary	Hex	Decimal	Binary	Hex	Decimal
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	A	10
0011	3	3	1011	B	11
0100	4	4	1100	C	12
0101	5	5	1101	D	13
0110	6	6	1110	E	14
0111	7	7	1111	F	15

Memorize this table!!!!

Converting from Binary $\leftarrow \rightarrow$ Hexadecimal

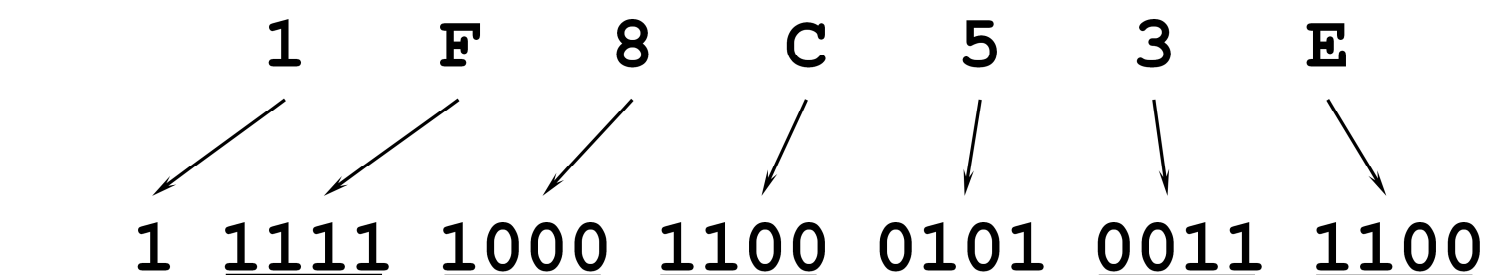
Every four bits is a hex digit.

- start grouping from right-hand side



Every hex digit is represented by 4-bits.

- start with 1st hex digit from right-hand side



This is not a new machine representation, just a convenient way to write the number.

Converting from Hexadecimal to Decimal

Every hex digit position has a base value

- multiply the value at the position by the base value

$$\begin{array}{cccc} \mathbf{8} & \mathbf{4} & \mathbf{D} & \mathbf{7} \\ \hline \Downarrow & \Downarrow & \Downarrow & \Downarrow \\ \mathbf{8 \times 16^3} & + & \mathbf{4 \times 16^2} & + & \mathbf{13 \times 16^1} & + & \mathbf{7 \times 16^0} = \\ \mathbf{8 \times 4096} & + & \mathbf{4 \times 256} & + & \mathbf{13 \times 16} & + & \mathbf{7 \times 1} = \\ \mathbf{32768} & + & \mathbf{1024} & + & \mathbf{208} & + & \mathbf{7} = \mathbf{34007} \end{array}$$

Another method is to convert to binary first (easy) then convert to decimal:

$$\begin{array}{l} \text{– } 84D7h = 1000\ 0100\ 1101\ 0111_2 = \\ \quad 1+2+4+16+64+128+1024+32768 = \mathbf{34007} \end{array}$$

Practice Converting from Hex to Decimal

