Positional number systems

- Decimal: base 10
- Binary: base 2
- Hexadecimal: base 16

Decimal

- Ten symbols: 0, 1, 2, …, 9
- Weights change by a factor of 10 from one position to the next.
- Shifting the decimal point changes the value by a factor of 10 for each position shifted.
- Each symbol is called a “digit”

25.4 decimal

$$\frac{2 \quad 5 \quad . \quad 4}{10 \quad 1 \quad \quad \frac{1}{10}} \quad \text{weights}$$

$$2 \times 10 + 5 \times 1 + 4 \times \frac{1}{10}$$

Binary

- Two symbols: 0 and 1
- Weights change by a factor of 2 from one position to the next.
- Shifting the binary point changes the value by a factor of 2 for each position shifted.
- Each symbol is called a “bit”.

1011.1 binary

$$\begin{array}{ccccccc} 1 & 0 & 1 & 1 & \cdot & 1 & \\ \hline 8 & 4 & 2 & 1 & & \frac{1}{2} & \text{weights (in decimal)} \end{array}$$

and its value in decimal is

$$1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 + 1 \times \frac{1}{2} = 11.5$$

Most and least significant bits

0111010110010111

MSB LSB

Hexadecimal

- Sixteen symbols: 0, 1, 2, …, 9, A, B, C, D, E, F
- Weights change by a factor of 16 from one position to the next.
- Shifting the hexadecimal point changes the value by a factor of 16 for each position shifted.
- Each symbol is called a “hex digit”
- A = 10 decimal B = 11 decimal
- C = 12 decimal D = 13 decimal
- E = 14 decimal   F= 15 decimal

1CB.8 hex

1	C	B	.	8	
256	16	1		$\frac{1}{16}$	weights (in decimal)

and its value is

$$1 \times 256 + C \times 16 + B \times 1 + 8 \times \frac{1}{16}$$

Substituting the decimal equivalents for the symbols B (11 decimal) and C (12 decimal), we get an all-decimal expression from which we can compute its decimal value:

$$1 \times 256 + 12 \times 16 + 11 \times 1 + 8 \times \frac{1}{16} = 459.5$$

Memorize this table

FIGURE 1.1

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

Byte

- Sequence of 8 bits
- Each bit has two possibilities: 0 or 1
- Thus, there are 2x2x2x2x2x2x2x2 = 2^8 distinct patterns

FIGURE 1.2

	Distinct Binary Numbers			
Number of Bits		Decimal		Hex
4	2^4 =		16 =	10
7	2^7 =		128 =	80
8	2^8 =		256 =	100
10	2^{10} =	1K =	1,024 =	400
12	2^{12} =	4K =	4,096 =	1000
15	2^{15} =	32K =	32,768 =	8000
16	2^{16} =	64K =	65,536 =	10,000
20	2^{20} =	1M =	1,048,576 =	100,000
30	2^{30} =	1G =	1,073,741,824 =	40,000,000

Arithmetic in decimal

- In addition, whenever a column sum is greater than or equal to the number base, a carry is added to the next column. The column result is the rightmost symbol of the column sum; the carry is the left symbol(s) of the column sum. For example, in the decimal addition,

$$\begin{array}{r} 1 \longleftarrow \text{carry} \\ 28 \\ +\ 39 \\ \hline 67 \end{array}$$

  the right column sum is 17. Thus, the result digit in the right column is 7 with a carry of 1 into the next column.
- In subtraction, a borrow into a column is equal to the number base. For example, in the decimal subtraction

$$\begin{array}{r} 34 \\ -\ 8 \\ \hline 26 \end{array}$$

  we borrow 1 from the 3 in the left column. This borrow is worth 10 in the right column. Thus, we subtract 8 from the sum of 10 (the borrow) and 4 to get 6. We borrow whenever the bottom symbol in a column is greater than the effective top symbol (the top symbol less any borrows from it).
- A borrow from 0 propagates to the left. For example, in the subtraction

$$\begin{array}{r} 3000 \\ -\quad 2 \\ \hline 2998 \end{array}$$

Adding in binary

```
 111     ← carries
  01100
+ 11110
 101010
```

Subtraction in binary

```
  2 decimal ← borrow into second column
  101
- 011
  010
```

Addition in hex

```
  1  ← carry
  A9
+ 19
  C2
```

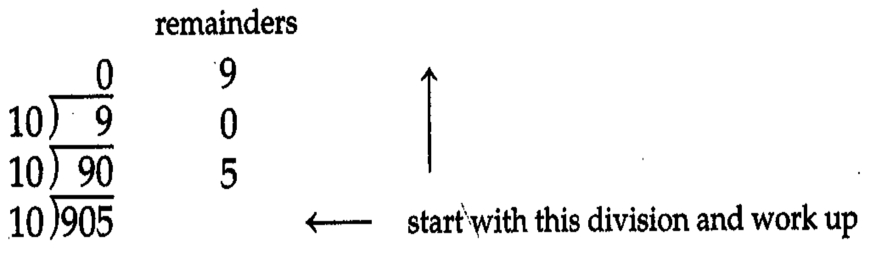
Subtraction in hex

```
16 decimal ← borrow into right column
  A5
- 2B
  7A
```

Converting to base n

- Integer part: Repeatedly divide by n. Remainders are the digits of the base n number.
- Fractional part: Repeatedly multiply by n. Whole parts are the digits of the base n number.

Each remainder is a digit



Convert 11 decimal to binary

	remainders	
0	1	↑
2)1	0	
2)2	1	
2)5	1	
2)11		← start with this division and work up

11 decimal = 1011 binary

Convert 30 decimal to hex

	remainders	
0	1	↑
16)1	14 = E	
16)30		← start with this division and work up

30 decimal = 1E hex

Each whole part is a digit (strip whole part of product after each multiplication by 10)

$$
\begin{array}{rl}
 & .43 \\
 & \times\, 10 \\ \hline
1^{st}\text{ digit} \longrightarrow & 4.\;\; 3 \\
 & \times\, 10 \\ \hline
2^{nd}\text{ digit} \longrightarrow & 3.\;\; 0
\end{array}
$$

Convert .6875 decimal to binary

whole parts		
	.6875	⟵ start with this multiplication and work down
	× 2	
1	.3750	
	× 2	
0	.7500	
	× 2	
1	.5000	
	× 2	
1	.0000	

.6875 decimal = .1011 binary

Converting between binary and hex

- Very easy to do
- The ease of conversion is the reason why hex is often used as a shorthand representation of binary.
- To do conversion, you need to know the binary-hex equivalents for the numbers 0 to 15.

Binary to hex

0111 1010 1100 · 1110

7 A C · E

Hex to binary

2 A D · B E

0010 1010 1101 · 1011 1110

Two’s complement

The two’s complement of the number M is the number which yields 0 when added to M.

Note this behavior

carry discarded

```
  1111111111111111   16 bit number containing all 1's
+                1   add 1
  ----------------
  0000000000000000   get all zeros
```

Simply flipping the bits does not yield the two's complement.

```
   0000000000000101 = +5
+  1111111111111010 = +5 flipped
   ----------------
   1111111111111111       should be zero
```

Getting the two's complement

- Simply flipping the bits does not yield the two's complement—it yields all 1's.
- But adding 1 to all 1's yields all 0's.
- So flipping the bits **and** adding 1 should yield the two's complement.

Flip bits and add 1

```
  1111111111111010  = +5 flipped
+                1
  1111111111111011  = +5 flipped + 1
```

Now let's see if this new value, +5 flipped + 1, yields zero when added to +5:

```
carry  <----
            0000000000000101 = +5
          + 1111111111111011 = +5 flipped + 1
            0000000000000000
```

Two's complement of 1

We can find the two's complement of any number using the procedure that we used on +5: Flip all the bits and then add one. For example, the computation of the two's complement of

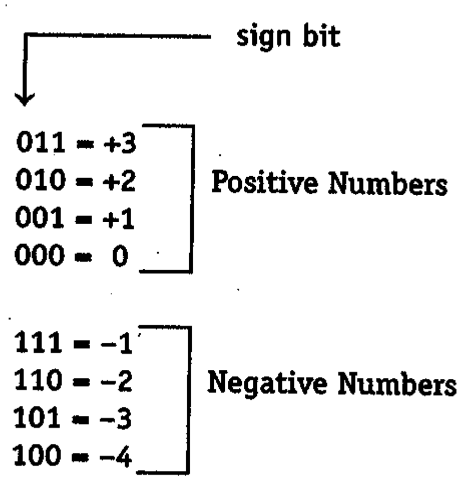
```
  0000000000000001  = +1
```

is:

```
  1111111111111110  = +1 flipped
+                1
  1111111111111111  = −1
```

3-bit two's complement numbers

FIGURE 1.3



Terminology

1. To *complement a bit* means to flip the bit.
2. To *complement a number* means to take its two's complement (i.e., flip its bits and add one).
3. To *bitwise complement a number* means to flip each of its bits but *not* add one. The bitwise complement of a number is sometimes called the *one's complement* of the number (see the next section).