Chapter 2

Number System

Decimal – 0-9
Hexadecimal 0-9, A,B, C, D, E, F
Octal – 0-7
Binary - 0,1

Decimal	0	1	2	3	4	5	6	7	8	9					
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
Binary	0	1													

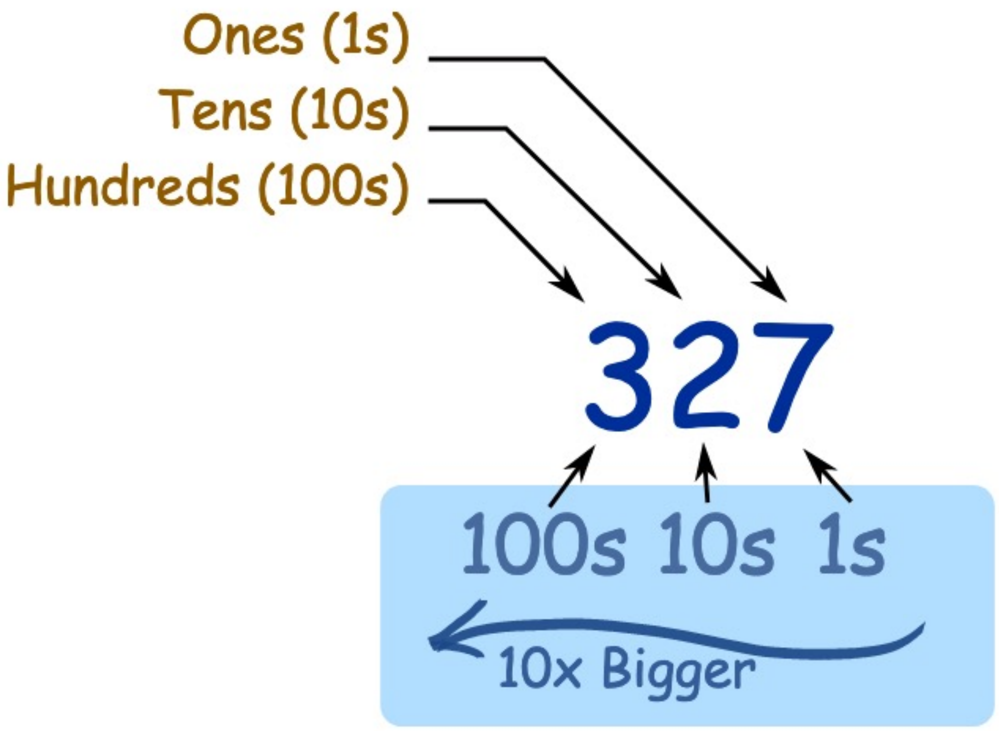
Definition of

Decimal Number System

more ...

The number system we use every day, based on 10 digits (0,1,2,3,4,5,6,7,8,9).

Position is important, with the first position being units, then next on the left being tens, then hundreds and so on.



Hexadecimal

16 Different Values

There are **16** Hexadecimal digits. They are the same as the decimal digits up to 9, but then there are the letters A, B, C, D, E and F in place of the decimal numbers 10 to 15:

Hexadecimal:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

So a single Hexadecimal digit can show 16 different values instead of the normal 10.

Octal

An Octal Number uses only these 8 digits: 0, 1, 2, 3, 4, 5, 6 and 7

Examples:

- **7** in Octal equals 7 in the Decimal Number System
- **10** in Octal equals 8 in the Decimal Number System
- **11** in Octal equals 9 in the Decimal Number System

...

- **167** in Octal equals 119 in the Decimal Number System

Binary Number

A Binary Number is made up of only **0**s and **1**s.

110100

Example of a Binary Number

There is no 2, 3, 4, 5, 6, 7, 8 or 9 in Binary!

Decimal to Binary

Conversion steps:

1. Divide the number by 2.
2. Get the integer quotient for the next iteration.
3. Get the remainder for the binary digit.
4. Repeat the steps until the quotient is equal to 0.

Example #1

Convert 13_{10} to binary:

Division by 2	Quotient	Remainder	Bit #
13/2	6	1	0
6/2	3	0	1
3/2	1	1	2
1/2	0	1	3

So $13_{10} = 1101_2$

Binary to Decimal

For binary number with n digits:

$$d_{n-1} \ldots d_3\, d_2\, d_1\, d_0$$

The decimal number is equal to the sum of binary digits (d_n) times their power of 2 (2^n):

$$\text{decimal} = d_0{\times}2^0 + d_1{\times}2^1 + d_2{\times}2^2 + \ldots$$

Example

Find the decimal value of 111001_2:

binary number:	1	1	1	0	0	1
power of 2:	2^5	2^4	2^3	2^2	2^1	2^0

$$111001_2 = 1\cdot2^5+1\cdot2^4+1\cdot2^3+0\cdot2^2+0\cdot2^1+1\cdot2^0 = 57_{10}$$

Decimal to Hexadecmial

Conversion steps:

1. Divide the number by 16.
2. Get the integer quotient for the next iteration.
3. Get the remainder for the hex digit.
4. Repeat the steps until the quotient is equal to 0.

Example #1

Convert 7562_{10} to hex:

Division by 16	Quotient (integer)	Remainder (decimal)	Remainder (hex)	Digit #
7562/16	472	10	A	0
472/16	29	8	8	1
29/16	1	13	D	2
1/16	0	1	1	3

So 7562_{10} = $1D8A_{16}$

Converting Decimal Fraction to Binary

To convert fraction to binary, start with the fraction in question and multiply it by **2** keeping notice of the resulting integer and fractional part. Continue multiplying by 2 until you get a resulting fractional part equal to zero. Then just write out the integer parts from the results of each multiplication.

Here is an example of such conversion using the fraction **0.375**.

$$0.375 \cdot 2 = 0 + 0.75$$
$$0.75 \cdot 2 = 1 + 0.5$$
$$0.5 \cdot 2 = 1 + 0$$

Now, let’s just write out the resulting integer part at each step — **0.011**. So, **0.375** in decimal system is represented as **0.011** in binary.

Converting Binary Fraction Integer to Decimal

To convert binary fraction to decimal, start from the right with the total of 0. Take your current total, add the current digit and divide the result by 2. Continue until there are no more digits left. Here is an example of such conversion using the fraction **0.1011**. I've simply replaced division by 2 with multiplication by **1/2**.

$$\frac{1}{2} \cdot (1 + 0) = 0.5$$
$$\frac{1}{2} \cdot (1 + 0.5) = 0.75$$
$$\frac{1}{2} \cdot (0 + 0.75) = 0.375$$
$$\frac{1}{2} \cdot (1 + 0.375) = 0.6875$$

Floating-Point Binary

IEEE Short Real: 32 bits	1 bit for the sign, 8 bits for the exponent, and 23 bits for the mantissa. Also called *single precision*.
IEEE Long Real: 64 bits	1 bit for the sign, 11 bits for the exponent, and 52 bits for the mantissa. Also called *double precision*.

Both formats use essentially the same method for storing floating-point binary numbers, so we will use the Short Real as an example in this tutorial.
The bits in an IEEE Short Real are arranged as follows, with the most significant bit (MSB) on the left:

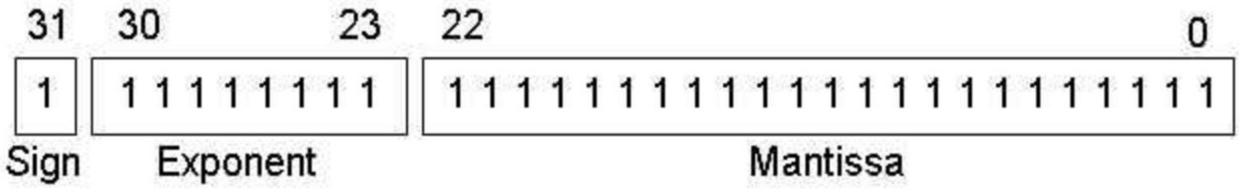


Fig. 1

http://cstl-csm.semo.edu/xzhang/Class%20Folder/CS280/Workbook_HTML/FLOATING_tut.htm

Floating-Point Binary

Floating-Point Binary

mantissa										exponent					
0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0

0 1 0 1 0 1 0 0 0 0

-32	16	8	4	2	1
0	0	0	0	1	0

= 2

0 1 0 . 1 0 1 0 0 0 0 $\times\ 2^{2}$

2	1	0.5	0.25	0.125
1	0 .	1	0	1

= 2.625

$0101010000000010_2 = 2.625_{10}$

Floating-Point Binary

mantissa										exponent					
0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0

										-32	16	8	4	2	1	
0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	= -2

0 . 0 0 1 0 0 0 0 0 0 0 0 $\times 2^{-2}$

1	0.5	0.25	0.125	
0 .	0	0	1	= 0.125

$0100000000111110_2 = 0.125_{10}$

Floating-Point Binary

mantissa										exponent					
1	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1

										-32	16	8	4	2	1	
1	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	= 3

1 1 1 0 . 1 0 0 0 0 0 $\times 2^3$

-8	4	2	1	0.5	
1	1	1	0 .	1	= -1.5

$1110100000000011_2 = -1.5_{10}$

Floating-Point Binary

mantissa										exponent					
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0

1 1 0 0 0 0 0 0 0 0

-32	16	8	4	2	1	
1	1	1	1	1	0	= -2

0 . 0 1 1 0 0 0 0 0 0 0 0 $\times 2^{-2}$

1	0.5	-0.25	0.125	
0 .	0	1	1	= -0.125

$1100000000111110_2 = -0.125_{10}$

Floating-Point Binary

Convert the following floating point binary numbers into denary. Assume 10 bits for the mantissa and 6 bits for the exponent, both in two's complement

0110110000000100

0101000000111111

Floating-Point Binary

Convert the following floating point binary numbers into denary. Assume 10 bits for the mantissa and 6 bits for the exponent, both in two's complement

0110110000000100

= 0.110110000 000100

= 0.110110000×2^{4}

= 01101.10000

= 1101.1

= 13.5

$0110110000000100_2 = 13.5_{10}$

0101000000111111

= 0.101000000 111111

= $0.101000000 \times 2^{-1}$

= .0101000000

= 0.0101

= 0.3125

$0101000000111111_2 = 0.3125_{10}$

Floating-Point Binary

Convert the following floating point binary numbers into denary. Assume 4 bits for the mantissa and 4 bits for the exponent, both in two's complement

01110011

01111110

Floating-Point Binary

Convert the following floating point binary numbers into denary. Assume 4 bits for the mantissa and 4 bits for the exponent, both in two's complement

01110011

= 0.111 0011

$= 0.111 \times 2^{3}$

= 0111.

= 0111.0

= 7

$01110011_2 = 7_{10}$

01111110

= 0.111 1110

$= 0.111 \times 2^{-2}$

= .00111

= 0.00111

= 0.21875

$01111110_2 = 0.21875_{10}$

Floating-Point Binary

Convert the following floating point binary numbers into denary. Assume 10 bits for the mantissa and 6 bits for the exponent, both in two's complement

1101000000 111111

1001101000000110

Floating-Point Binary

Convert the following floating point binary numbers into denary. Assume 10 bits for the mantissa and 6 bits for the exponent, both in two's complement

1101000000 111111

= 1.101000000   111111

= 1.101000000   x 2^{-1}

= .1101000000

= 0.1101

= -0.1875

1101000000111111_2 = -0.1875_{10}

1001101000000110

= 1001101000   000110

= 1.001101000   x 2^{6}

= 1001101.000

= 1001101

= -51

1001101000000110_2 = -51_{10}

Primitive Types and Expressions

1. Variables
2. Constants
3. Scope
4. Overflow
5. Operators

```
int number;

int Number = 1;

const float Pi = 3.14f;
```

Variables – A name given to storage location in the memory
Constants – An immutable value

Data Type and Identifer is required to declare a variable followed by a semicolan. For constants. Its compulsory to assign a value to it.

Identifiers

1. Cannot starts with a number
    1. 1route – illegal
    2. oneroute – legal

2. No Whitespaces
    1. First Name – illegal
    2. firstName - legal

3. Cannot be a keyword
    1. int – illegal
    2. @int - legal

4. Always use meaningful names

Code need to be

1. Readable
2. Maintainable
3. Cleaner

Naming Conventions – C Language Family

Camel Case – firstName
Pascal Case – FirstName
Hungarian Notation – strFirstName (Came from C/C++ background. However, not liked by C# developers.

- For local variables: Camel Case

```
int number;
```

- For constants: Pascal Case

```
const int MaxZoom = 5;
```

Primitive Data Types

	C# Type	.NET Type	Bytes	Range
Integral Numbers	**byte**	Byte	1	0 to 255
	short	Int16	2	-32,768 to 32,767
	int	Int32	4	-2.1B to 2.1B
	long	Int64	8	...
Real Numbers	**float**	Single	4	-3.4×10^{38} to 3.4×10^{38}
	double	Double	8	...
	decimal	Decimal	16	-7.9×10^{28} to 7.9×10^{28}
Character	**char**	Char	2	Unicode Characters
Boolean	**bool**	Boolean	1	True / False

Real Numbers

Real Numbers	C# Type	.NET Type	Bytes	Range
	float	Single	4	-3.4×10^{38} to 3.4×10^{38}
	double	Double	8	...
	decimal	Decimal	16	-7.9×10^{28} to 7.9×10^{28}

```
float number = 1.2f;

decimal number = 1.2m;
```

Non-Primitive Data Types

1. Strings
2. Arrays
3. Enum
4. Class

Overflowing

```
byte number = 255;

number = number + 1;  // 0
```

```
checked
{
    byte number = 255;

    number = number + 1;
}
```

Ariane 5 Explosion | A Very Costly Coding Error

https://www.youtube.com/watch?v=5tJPXYA0Nec

Scope

Scope – where a variable or constant have a meaning

```
{
    byte a = 1;

    {
        byte b = 2;

        {
            byte c = 3;
        }
    }
}
```

Type Conversions

Implicit Type Conversion
Explicit Type Conversion (Casting)
Conversion between non compatible types

Implicit Type Conversion

```
byte b = 1;                                    00000001

int i = b;     00000000 00000000 00000000 00000001
```

Explicit Types Conversion

```
int i = 1;

byte b = i;            // won't compile
```

```
int i = 1;

byte b = (byte)i;
```

```
float f = 1.0f;

int i = (int)f;
```

Non-Compatible Types

```
string s = "1";

int i = (int)s;      // won't compile
```

Use Convert class defined in System Namespace or Parse method

```
string s = "1";

int i = Convert.ToInt32(s);

int j = int.Parse(s);
```

Convert Class

- ToByte()
- ToInst16()
- ToInt32()
- ToInt64()

Primitive Types and Expressions

Follow Book