The number system

Number system and Binary numbers
What is the number system?

The number system is a writing system for expressing numbers. It is composed of a collection of symbols used to represent small numbers, together with a system of rules for representing larger numbers.

<table>
<thead>
<tr>
<th>Number Systems</th>
<th>Base</th>
<th>Digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>2</td>
<td>01</td>
</tr>
<tr>
<td>Octal</td>
<td>8</td>
<td>01234567</td>
</tr>
<tr>
<td>Decimal</td>
<td>10</td>
<td>0123456789</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>16</td>
<td>0123456789ABCDEF</td>
</tr>
</tbody>
</table>

Example of a 32-bit instruction

```
LDR <Rd>, [<Rn>, #<immed_5> * 4]
```

32 bit ARM LDR
Example of a 32-bit instruction

Instruction format (machine language)

- **Machine Language**
  - Computers do not understand “add R8, R17, R18”
  - Instructions are translated to machine language (1s and 0s)

- **Example:**
  
  ```
  add R8, R17, R18 →
  00000010 00110010 01000000 00100000
  ```

Number system: unary

What is the first numeral system we learn?

What is the most natural numeral system?

The unary uses one symbol.
This is the simplest way of counting
Number system: Roman

Numbers in the Roman system are represented by combinations of letters from the Latin alphabet.

Roman numerals used today, have seven symbols, each with a fixed integer value:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>I</th>
<th>V</th>
<th>X</th>
<th>L</th>
<th>C</th>
<th>D</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>50</td>
<td>100</td>
<td>500</td>
<td>1,000</td>
</tr>
</tbody>
</table>

Number system: base

A radix, or base, is the number of unique digits, including zero, used to represent numbers in a positional numeral system.

image credit: http://thepictutor.com/decimal.html
Number system: base

Generalized form of positional systems in Base B:
Commonly used numeral systems include:

**Binary** Base 2: 0, 1

**Octal** Base 8: 0, 1, 2, 3, 4, 5, 6, 7

**Decimal** Base 10: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

**Hexadecimal** Base 16: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Consider a number with n+1 digits:

\[ N_B = a_n a_{n-1} a_{n-2} \ldots a_0 \]

\[ N_B = a_n \times B^n + a_{n-1} \times B^{n-1} + a_{n-2} \times B^{n-2} + \ldots + a_1 \times B^1 + a_0 \times B^0 \]
Number system: base

The Egyptians had a base-10 system of hieroglyphs for numerals. They separate symbols for one unit, one ten, one hundred, one thousand, one ten thousand, one hundred thousand, and one million.

e.g. The number 276 in hieroglyphs, requires fifteen symbols:
Number system: base

<table>
<thead>
<tr>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
<th>100000</th>
<th>10⁵</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>100</td>
<td>1000</td>
<td>10000</td>
<td>100000</td>
<td>10⁵</td>
</tr>
</tbody>
</table>

Egyptian numeral hieroglyphs

e.g. The number 4622 in hieroglyphs, requires these symbols:

Number system: Binary numbers

Conversion of a number from binary to decimal:

\[ N_2 = 1 \times 2^9 + 0 \times 2^8 + 0 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \]

\[ = 537 \]
Number system: Binary numbers

Conversion of a number from **decimal** to **binary**:  
To convert a decimal number to binary:  
1. First, subtract the largest possible power of two  
2. Keep subtracting the next largest possible power of 2 from the remainder, marking 1s in each column where this is possible and 0s where it is not.

Converting the decimal (base 10) number 86 to binary (base 2)  

64 is the largest power of 2 that goes into 86. The result is:

```
 64  32  16   8   4   2   1
```

86 - 64 is 22. 32 is larger than 22, so a 0 is placed in the bit for the value 32. 16 is less than 22, so a 1 is placed in the bit for the value 16.

```
  1    0    1    ?    ?    ?    ?
 64  32  16   8   4   2   1
```

22 - 16 is 6. 8 is larger than 6. The bits 4+2 equal 6 so each of those bits become 1

```
  1    0    1    0    1    1    0
 64  32  16   8   4   2   1
```

This 86 in decimal is 1010110 in binary. \(86_{10} = 1010110_2\)
Number system: conversion int()

Conversion using Python functions.

We can use the int() built-in function to convert any base k number to decimal.

The syntax of int() method is: `int(x, base)`

```python
# binary conversion to integer
print(f"1110 is integer: {int('1110', 2)}")

# hexadecimal conversion to integer
print(f"AB is integer: {int('ABC', 16)}")

1110 is integer: 14
AB is integer: 2748
```

The syntax of int() method is: `int(x, base)`

- `x`: number or string to convert
- `base`: base of the number in `x`

```python
> int('241103')
241103
> int('241103', 10)
241103
> int('100101', 2)
37
> int('9401BA', 16)
9699770
```
The `bin()` method takes one parameter:

**input**: the integer number to convert to binary  
**output**: a binary string

If the input is not an integer, `bin()` raises a `TypeError` exception.

```python
numbers = [25, 16, '128']
for n in numbers:
    try:
        print(bin(n))
    except TypeError:
        print("Incorrect type for binary conversion")
```

0b11001  
0b10000  
Incorrect type for binary conversion

Conversion using Python functions.

You can use the `bin()` built-in function to convert a decimal number to binary. `bin()` outputs the binary number as a string.

```python
# The decimal number 50 in binary
# '0b' is the prefix for a binary number
print(bin(50))
'0b110010'
print(bin(357))
'0b101100101'
# ignoring the prefix '0b' in the output of bin()
print(bin(357)[2:])
'101100101'
```
Number system: hexadecimal numbers

Conversion using Python functions.

The \texttt{hex()} function converts an integer to the corresponding hexadecimal number and returns it.

```python
numbers = [25, 16, 128]
for n in numbers:
    try:
        print(hex(n))
    except TypeError:
        print("Incorrect type for binary conversion")
```

Number system: Binary addition

Rules for binary addition

\[
\begin{align*}
0 + 0 &= 0 \\
0 + 1 &= 1 \\
1 + 0 &= 1 \\
1 + 1 &= 10
\end{align*}
\]

for 1 + 1, we write down a zero in the right-most column and carry over a one to the next column.
Number system: Binary addition

```
# Convert decimal 19 to binary number '0b10011'
print(bin(19))

# 0b on the previous line means the result is binary
print(int("10011", 2))

# the 4 binary addition rules
rule1 = bin(int("0", 2) + int("0", 2)) # 0 + 0 = 0
rule2 = bin(int("0", 2) + int("1", 2)) # 0 + 1 = 1
rule3 = bin(int("1", 2) + int("0", 2)) # 1 + 0 = 1
rule4 = bin(int("1", 2) + int("1", 2)) # 1 + 1 = 10

# Adding binary numbers 10101 and 110 results in 11011
number = bin(int("10101", 2) + int("110", 2))
print(number)
```

Number system: bitwise operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>Complement</td>
</tr>
<tr>
<td>&amp;</td>
<td>Binary AND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>Binary XOR</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>Binary Left Shift</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>Binary Right Shift</td>
</tr>
</tbody>
</table>

- Each bit of the output is 1 if the bit of x is 0, otherwise it is 0.
- Each bit of the output is 1 if the bit of x AND the bit of y is 1, otherwise it's 0.
- Each bit of the output is 0 if the bit of x AND the bit of y is 0, otherwise it's 1.
- Each bit of the output is 1 if the bit of x is the one complement of the bit of y, otherwise it's 0.
- Returns x with the bits shifted to the left by y places.
- Returns x with the bits shifted to the right by y places.
### Number system: bitwise operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>Complement</td>
</tr>
<tr>
<td>&amp;</td>
<td>Binary AND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>Binary XOR</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>Binary Left Shift</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>Binary Right Shift</td>
</tr>
</tbody>
</table>

#### Truth Tables

- **Complement (~)**
  - $a$: 0001
  - $\sim a$: 1110

- **Binary AND (&)**
  - $a$: 0001
  - $b$: 0110
  - $a \& b$: 0000

- **Binary OR (|)**
  - $a$: 0001
  - $b$: 0110
  - $a | b$: 0111

- **Binary XOR (^)**
  - $a$: 0001
  - $b$: 0110
  - $a ^ b$: 0110

- **Binary Left Shift (<<)**
  - $a$: 0001
  - $a << 1$: 0010

- **Binary Right Shift (>>)**
  - $a$: 0001
  - $a >> 1$: 0000
Activity 6

a = 40
b = 18

1. Convert a to binary
2. Convert b to binary
3. Calculate ~a
4. Calculate a & b
5. Calculate a \& b
6. Calculate (a \& b) << 3
7. Calculate a + b (binary addition)

Activity 6: solution

a = 40, b = 18

1. Convert a to binary: 40 = 0b101000
2. Convert b to binary: 18 = 0b10010
3. Calculate ~a = 0b10111
4. Calculate a & b = 0b0
5. Calculate a \& b = 0b111010
6. Calculate (a \& b) << 3 = 0b111010000
7. Calculate a + b (binary addition) = 0b00111010