Beyond Base 10: Non-decimal Based Number Systems

- **What is the decimal based number system?**
- **How do other number systems work (binary, octal and hex)**
- **How to convert to and from non-decimal number systems to decimal**
- **Binary math**

What Is Decimal?

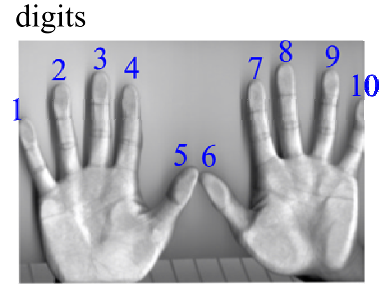
Base 10

- 10 unique symbols are used to represent values

0
1
2
3
4
5
6
7
8
9
10
:

The largest decimal value that can be represented by a single decimal digit is 9 = base(10) - 1

The number of digits is based on…the number of digits



Binary

Base two

Employs two unique symbols (0 and 1)

Largest decimal value that can be represented by 1 binary digit = 1 = base(2) - 1

Table Of Binary Values

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

Why Bother With Binary?

Representing information

- ASCII (American Standard Code for Information Interchange)
- Unicode

It's the language of the computer

Representing Information: ASCII

Decimal	Binary	ASCII
0 – 31	00000000 – 00011111	Invisible (control characters)
32 – 47	00100000 – 00101111	Punctuation, mathematical operations
48 - 57	00110000 – 00111001	Characters 0 - 9
58 – 64	00111010 – 01000000	Comparators and other miscellaneous characters : ; ? @
65 - 90	01000001 – 01011010	Alphabetic (upper case A - Z)
91 – 96	01011011 – 01100000	More miscellaneous characters [\ ] ^ _ '
97 – 122	01100001 – 01111010	Alphabetic (lower case a - z)
123 – 127	01111011 - 01111111	More miscellaneous characters { | } ~ DEL

Representing Information: ASCII (2)

Uses 7 bits to represent characters

Max number of possibilities = 2^7 = 128 characters that can be represented

e.g., 'A' is 65 in decimal or 01000001in binary. In memory it looks like this:

0	1	0	0	0	0	0	1

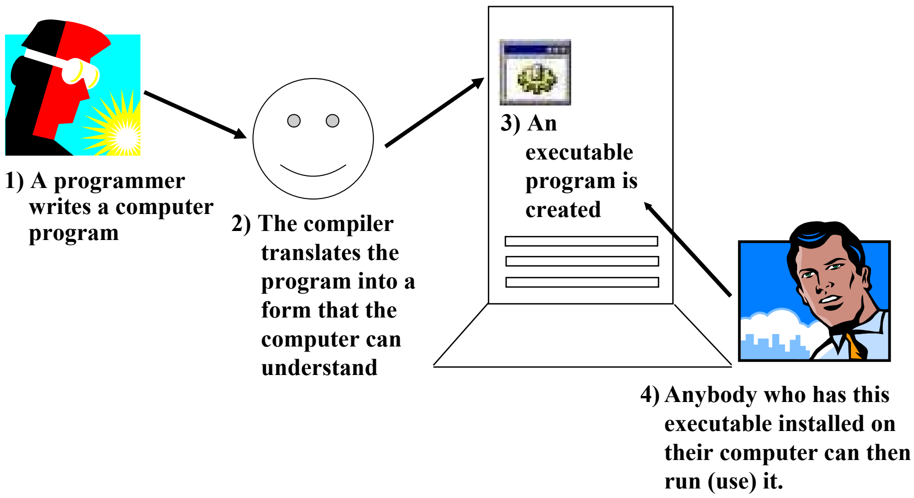
Representing Information: Unicode

Uses 16 bits (or more) to represent information

Max number of possibilities = 2^{16} = 65536 characters that can be represented (more if more bits are used)

Computer Programs

Binary is the language of the computer



Octal

Base eight

Employs eight unique symbols (0 - 7)

Largest decimal value that can be represented by 1 octal digit = 7 = base(8) - 1

Table Of Octal Values

Decimal value	Octal value	Decimal value	Octal value
0	0	8	10
1	1	9	11
2	2	10	12
3	3	11	13
4	4	12	14
5	5	13	15
6	6	14	16
7	7	15	17

Why Octal?

1001 0100 1100 1100?

1001 0100 1100 0100?

1001 0100 1100 0011?

Why Octal? (2)

Machine language	**Octal value**
1010111000000	012700
1001010000101	011205

Hexadecimal (Hex)

Base sixteen

Employs sixteen unique symbols (0 – 9, followed by A - F)

Largest decimal value that can be represented by 1 hex digit = 15

Table of Hex Values

Decimal value	Hexadecimal value	Decimal value	Hexadecimal value
0	0	9	9
1	1	10	A
2	2	11	B
3	3	12	C
4	4	13	D
5	5	14	E
6	6	15	F
7	7	16	10
8	8	17	11

Why Hexadecimal?

1001 0100 1000 0000 1100 0100 0110 1010?

Or

1001 0100 1000 0000 1100 0100 0110 1011?

James Tam

Why Hexadecimal? (2)

Machine language	**Hexadecimal value**
1010011000001	14C1
110000011100000	60E0

Example from 68000 Family Assembly Language by Clements A.

Summary (Decimal, Binary, Octal, Hex)

Decimal	Binary	Octal	Hex	Decimal	Binary	Octal	Hex
0	0000	0	0	8	1000	10	8
1	0001	1	1	9	1001	11	9
2	0010	2	2	10	1010	12	A
3	0011	3	3	11	1011	13	B
4	0100	4	4	12	1100	14	C
5	0101	5	5	13	1101	15	D
6	0110	6	6	14	1110	16	E
7	0111	7	7	15	1111	17	F

High Vs. Low Level

Human languages — E.g., English, French, Spanish, Chinese, German etc.

High level programming language — E.g., Pascal, Java, C++

Low level programming language — Assembly

Machine language — Binary

Computer hardware

High level

Low level

Overflow: A Real World Example

You can only represent a finite number of values

Overflow: Binary

Occurs when you don't have enough bits to represent a value ("wraps around" to zero)

Binary (1 bit)	Value
0	0
1	1

0	0
1	1
:	:

Binary (2 bits)	Value
00	0
01	1
10	2
11	3

00	0
01	1
10	2
11	3
:	:

Binary (3 bits)	Value
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

000	0
001	1
:	:

Arbitrary Number Bases

Base N

Employs N unique symbols

Largest decimal value that can be represented by 1 digit = Base (N) - 1

Converting Between Different Number Systems

Binary to/from octal

Binary to/from hexadecimal

Octal to/from hexadecimal

Decimal to any base

Any base to decimal

Binary To Octal

3 binary digits equals one octal digit (remember $2^3=8$)

Form groups of three starting at the decimal

- For the integer portion start grouping at the decimal and go left
- For the fractional portion start grouping at the decimal and go right

e.g. $101\,100_2 = ???_8$

$5\ 4_8$.

Octal To Binary

1 octal digit equals = 3 binary digits

Split into groups of three starting at the decimal

•For the integer portion start splitting at the decimal and go left

•For the fractional portion start splitting at the decimal and go right

e.g. $12.5_8 = ???_2$

$001\ 010\ .\ 101_2$

Binary To Hexadecimal

4 binary digits equals one hexadecimal digit (remember $2^4=16$)

Form groups of four at the decimal

•For the integer portion start grouping at the decimal and go left

•For the fractional portion start grouping at the decimal and go right

e.g., $1000.0100_2 = ???_{16}$

$8\ .\ 4_{16}$

Hexadecimal To Binary

1 hex digit equals = 4 binary digits

Split into groups of four starting at the decimal

- For the integer portion start splitting at the decimal and go left
- For the fractional portion start splitting at the decimal and go right

e.g., $A.3_{16} = ???_2$

1010.0011_2

Octal To Hexadecimal

Convert to binary first!

e.g., 25_8 to $???_{16}$

010101_2.

Octal To Hexadecimal

Convert to binary first!

e.g., 25_8 to $???_{16}$

Add any leading zeros that are needed (in this case two).

Regroup in groups of 4

0001 0101_2.

1 5_{16}.

Hexadecimal To Octal

e.g., 15_{16} to $???_8$

0001 0101_2.

Hexadecimal To Octal

e.g., 15_{16} to $???_8$

Add any leading zeros that are needed (in this case one).

Regroup in groups of 3

$\underline{000}\ \underline{010}\ \underline{101}_2$.

$0\ \ 2\ \ 5_8$.

Decimal To Any Base

Split up the integer and the fractional portions

1) For the integer portion:

a. Divide the integer portion of the decimal number by the target base.
b. The remainder becomes the first integer digit of the number (immediately left of the decimal) in the target base.
c. The quotient becomes the new integer value.
d. Divide the new integer value by the target base.
e. The new remainder becomes the second integer digit of the converted number (second digit to the left of the decimal).
f. Continue dividing until the quotient is less than the target base and this quotient becomes the last integer digit of the converted number.

Decimal To Any Base (2)

2) For the fractional portion:

a. Multiply by the target base.
b. The integer portion (if any) of the product becomes the first rational digit of the converted number (first digit to the right of the decimal).
c. The non-rational portion of the product is then multiplied by the target base.
d. The integer portion (if any) of the new product becomes the second rational digit of the converted number (second digit to the right of the decimal).
e. Keep multiplying by the target base until either the resulting product equals zero or you have the desired number of places of precision.

Decimal To Any Base (2)

e.g., 9_{10} to $???_2$

$1\ 0\ 0\ 1\ ._2$

9 / 2: q = 4 r = 1

4 / 2: q = 2 r = 0

2/2: q = 1 r = 0

1

Stop dividing! (quotient less than target base)

Any Base To Decimal

Multiply each digit by the base raised to some exponent[1] and sum the resulting products.

	3	2	1	0	-1	-2	-3	← Position of digits
i.e.	d7	d6	d5	d4.	d3	d2	$d1_b$	← Number to be converted

Base = b

Value in decimal = (digit7*b^3) + (digit6*b^2) + (digit5*b^1) + (digit4*b^0) + (digit3*b^{-1}) + (digit2*b^{-2}) + (digit1*b^{-3})

1 The value of this exponent will be determined by the position of the digit.

Any Base To Decimal (2)

e.g., 12_8 to $???_{10}$

1	0	← Position of the digits
1	2.	← Number to be converted

Base = 8

Value in decimal = $(1*8^1) + (2*8^0)$

$= (1*8) + (2*1)$

$= 8 + 2$

$= 10_{10}$

Addition In Binary: Five Cases

Case 1: sum = 0, no carry out

$$\begin{array}{r} 0 \\ +\ \underline{0} \\ 0 \end{array}$$

Case 2: sum = 1, no carry out

$$\begin{array}{r} 0 \\ +\ \underline{1} \\ 1 \end{array}$$

Case 3: sum = 1, no carry out

$$\begin{array}{r} 1 \\ +\ \underline{0} \\ 1 \end{array}$$

Case 4: sum 0, carry out = 1

$$\begin{array}{rr} {}^{1} & \\ & 1 \\ + & \underline{1} \\ 1 & 0 \end{array}$$

1 + 1 = 2 (in decimal) = 10 (in binary)

Addition In Binary: Five Cases (2)

Case 5: Sum = 1, Carry out = 1

$$\begin{array}{rr} {}^{1} & \\ & 1 \\ & 1 \\ + & \underline{1} \\ 1 & 1 \end{array}$$

1 + 1 + 1 = 3 (in decimal) = 11 (in binary)

Subtraction In Binary (4 cases)

Case 1:

$$\begin{array}{r} 0 \\ -\ \underline{0} \\ 0 \end{array}$$

Case 2:

$$\begin{array}{r} 1 \\ -\ \underline{1} \\ 0 \end{array}$$

Case 3:

$$\begin{array}{r} 1 \\ -\ \underline{0} \\ 1 \end{array}$$

Case 4:

$$\begin{array}{r} \overset{0}{\cancel{1}}\ \overset{2}{\cancel{0}} \\ -\ \underline{1} \\ 1 \end{array}$$

The amount that you borrow equals the base

- Decimal: Borrow 10
- Binary: Borrow 2

You Should Now Know

- What is meant by a number base.
- How binary, octal and hex based number systems work and what role they play in the computer.
- What is overflow, why does it occur and when does it occur.
- How to/from convert between non-decimal based number systems and decimal.
- How to perform simple binary math (addition and subtraction).