CSC 230

Numeration

Tom Arjannikov

tarjan@uvic.ca

University of Victoria

Fall 2019

Outline

- Number systems
- Conversion between
- Binary arithmetic
- Computer logic
- Negative numbers
- Horner's Algorithm
- Endianness

Numbers

- Numeration - The action or process of calculating or assigning a number to something (Oxford dictionary).
- Arabic numbers (digits)
- Abacus (oldest calculator)
- Binary numbers (bits)

Integer Number Systems

Binary

- Base: 2
- Digits: 0,1

Octal

- Base: 8
- Digits: 0,1,2,3,4,5,6,7

Decimal

- Base: 10
- Digits: 0,1,2,3,4,5,6,7,8,9

Hexadecimal

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

Positional Representation

Base 10

$$
\begin{array}{cr}
+ & 7 \\
+ & 80 \\
+ & 500 \\
+ & 3000 \\
\hline
= & 3587
\end{array}
$$

Weighted Positional Representation

$$Integer\ \ Value = \sum_{i=0}^{n-1} d_i * b_i$$

$$Decimal\ \ Value = \sum_{i=-m}^{n-1} d_i * b_i$$

e.g. $312.98 = 3 * 10^2 + 1 * 10^1 + 2 * 10^0 + 9 * 10^{-1} + 8 * 10^{-2}$

Polynomial representation

Base 10

$3 * 1000 + 5 * 100 + 8 * 10 + 7 =$
$3 * 10^3 + 5 * 10^2 + 8 * 10^1 + 7 * 10^0$

General Form

$d_n * b^n + d_{n-1} * b^{n-1} + ... + d_2 * b^2 + d_1 * b^1 + d_0 * b^0$

where d_n is the digit in n^{th} position starting from the right and b is the base.

Notation

Binary: 0b or % prefix

e.g. 0b11011010 or %11011010

Octal: 0o (zero-oh) or 0 prefix

e.g 0o617 or 0617

Hexadecimal (hex): 0x (zero-oh) or $ prefix

e.g 0x1F or $1F

General: use a subscript with base number

e.g. $101_2 \neq 101_8 \neq 101_{10} \neq 101_{16}$

Memorize this table! (not)

Decimal	Binary	Hex	Octal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	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	

Conversion shortcut: binary $\rightarrow$ hex or octal

Group the bits:

Binary:

$$(\ 11010110 \ 10110101 \ )_2 = 54965_{10}$$

Octal:

$$(\ 1 \ 101 \ 011 \ 010 \ 110 \ 101)_2 = 153265_8$$

Hex:

$$(\ 1101 \ 0110 \ 1011 \ 0101 \ )_2 = D6B5_{16}$$

Example: convert 119_{10} to octal.

Repeated division by base:

$$\begin{array}{rcll}
119/8 & = & 14 * 8 + 7 & r = 7 \quad \text{(right most digit)} \\
14/8 & = & 1 * 8 + 6 & r = 6 \quad \text{(second last)} \\
1/8 & = & 0 * 8 + 1 & r = 1 \quad ()
\end{array}$$

stop at 0

$119_{10} = 167_8$

Conversion review

Decimal $\rightarrow$ binary, octal, hex

Binary $\rightarrow$ decimal, octal, hex

Octal $\rightarrow$ decimal, binary, hex

Hex $\rightarrow$ decimal, binary, octal

Binary Arithmetic

	1	1	1	1	1	1	1		$\leftarrow$ carry
		1	0	1	1	1	1	1	$\leftarrow$ input 1
$+$		0	1	0	1	0	1	0	$\leftarrow$ input 2
	1	0	0	0	1	0	0	1	$\leftarrow$ result

Traditional vs long carry methods

Binary Subtraction

	*							← borrow
	1	0	1	1	1	1	1	← input 1
-	0	1	0	1	0	1	0	← input 2
	0	1	1	0	1	0	1	← result

Binary Multiplication

$$
\begin{array}{ccccccccccl}
 & & & & & 1 & 0 & 1 & 0 & \leftarrow \text{input 1} \\
* & & & & & 1 & 1 & 0 & 1 & \leftarrow \text{input 2} \\
\hline
 & & & & & 1 & 0 & 1 & 0 & \\
+ & & & & 0 & 0 & 0 & 0 & & \\
+ & & & 1 & 0 & 1 & 0 & & & \\
+ & & 1 & 0 & 1 & 0 & & & & \\
\hline
 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \leftarrow \text{result}
\end{array}
$$

Binary Division

							1	1	1	$\leftarrow$ quotient
1	1	0	)	1	0	1	1	0	1	$\leftarrow$ divisor) dividend
-					1	1	0			
					1	0	1	0		
-						1	1	0		
						1	0	0	1	
-							1	1	0	
								1	1	$\leftarrow$ remainder

Computer (Boolean) Logic

Logic Gates Notation

NOT

$(\bar{A})$ or $(\neg A)$ or $(\sim A)$ or $(!A)$ or (A')

AND

$(A \cdot B)$ or $(A \wedge B)$ or $(A\&B)$

OR

$(A + B)$ or $(A \vee B)$ or $(A \parallel B)$

XOR

$(A \oplus B)$ or $(A \veebar B)$

Truth Table

A	B	NOT		AND	OR	XOR	NAND	NOR
0	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0
1	0	0	1	0	1	1	1	0
1	1	0	0	1	1	0	0	0

AND → 1 when both are one, 0 otherwise
OR → 0 when both are 0, 1 otherwise
XOR → 0 when both are the same, 1 otherwise

Bitwise Operations

Rotating

Shifting

$<<$ left shift

- multiply by 2
- overflow can occur

$>>$ right shift

- divide by 2
- underflow may occur

Masking

input: string of bits
mask: string of bits (same size as input)
binary Boolean operator:

- AND - 1 preserves, 0 clears
- OR - 0 preserves, 1 sets
- XOR - 0 preserves, 1 toggle

Negative Numbers

Signed Magnitude

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

1's Compliment

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	-0

2's Compliment

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

2's Compliment Algorithm

Convert negative value to binary:

1. Convert the absolute value to binary
2. Compliment (flip all the bits)
3. Add 1 (in binary)

Convert a negative binary number to decimal:

1. Subtract 1 (in binary)
2. Compliment (flip all the bits)
3. Convert as positive binary integer
4. Prepend a minus

2's Compliment Advantages

- efficient
- uniform
- avoids two zeros
- only need one adder (no subtraction unit)

In Addition...

Endianness

Byte-addressable memory:

Byte order	Big-endian	23-bit integer	Little-endian
		0A1B2C3D	
1	0A		3D
2	1B		2C
3	2C		1B
4	3D		0A

Word-addressable memory:

Byte order	Big-endian	23-bit integer	Little-endian
		0A1B2C3D	
1	0A1B		2C3D
2	2C3D		0A1B

Horner's Method

In 1819, William George Horner developed
a method for evaluation of a polynomial of degree n
using at most n multiplications and n additions.

$$d_n * b^n + d_{n-1} * b^{n-1} + ... + d_2 * b^2 + d_1 * b^1 + d_0 * b^0$$

$$= ((...(d_n * b + d_{n-1}) * b + ... + d_2) * b + d_1) * b + d_0$$

Note: evaluation starts from inner-most parentheses.

Allows for fast multiplication and division w/o hardware multiplier.