FACHHOCHSCHULE KIEL
University of Applied Sciences

Digital Design

Introduction to Number Systems

Overview

- The design of computers
  - It all starts with numbers
  - Building circuits
  - Building computing machines
- Digital systems
- Understanding decimal numbers
- Binary and octal numbers
  - The basis of computers!
- Conversion between different number systems

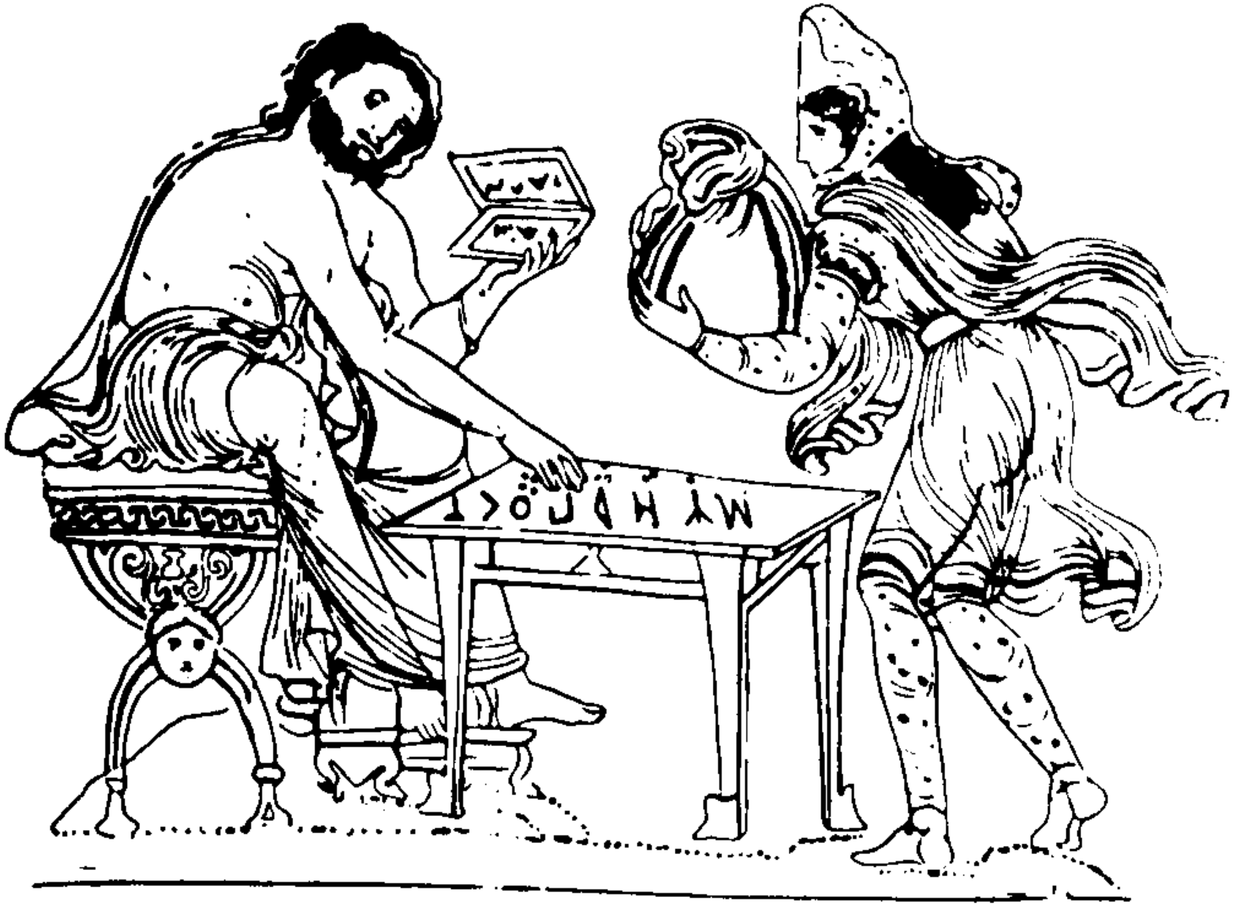
Digital Computer Systems

- Digital systems consider *discrete* amounts of data.
- Examples
  - 26 letters in the alphabet
  - 10 decimal digits
- Larger quantities can be built from discrete values:
  - Words made of letters
  - Numbers made of decimal digits (e.g. 239875.32)
- Computers operate on *binary* values (0 and 1)
- Easy to represent binary values electrically
  - Voltages and currents.
  - Can be implemented using circuits
  - Create the building blocks of modern computers

Digital Design

Understanding Number Systems

Understanding Numbers



Using counting boards in the past

Information units

The smallest binary information unit is called a "**Bit**", derived from the English term"*__bi__nary digi__t__*".

A bit describes the logical state of a two-valued system.

The fact that it describes a minimal system is expressed in the original description, introduced by **Claude Shannon:**

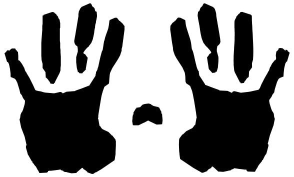
__B__asic __I__ndissoluble Uni__t__.

Claude Shannon
(1916 - 2001)

Understanding Decimal Numbers

- Decimal numbers are made of decimal digits:
  - (0,1,2,3,4,5,6,7,8,9)
- But how many items does a decimal number represent?
  - $8653 = 8x10^3 + 6x10^2 + 5x10^1 + 3x10^0$
- What about fractions?
  - $97654.35 =$
    $9x10^4 + 7x10^3 + 6x10^2 + 5x10^1 + 4x10^0 + 3x10^{-1} + 5x10^{-2}$
  - In formal notation -> $(97654.35)_{10}$
- Why do we use 10 digits, anyway?

digit from
latin "digitus" - finger



Understanding Octal Numbers

- Octal numbers are made of octal digits:
  - (0,1,2,3,4,5,6,7)
- How many items does an octal number represent?
  - $(4536)_8 = 4x8^3 + 5x8^2 + 3x8^1 + 6x8^0 = (1362)_{10}$
- What about fractions?
  - $(465.27)_8 = 4x8^2 + 6x8^1 + 5x8^0 + 2x8^{-1} + 7x8^{-2}$
- Octal numbers don’t use digits 8 or 9
- Who would use octal number, anyway?

Understanding Binary Numbers

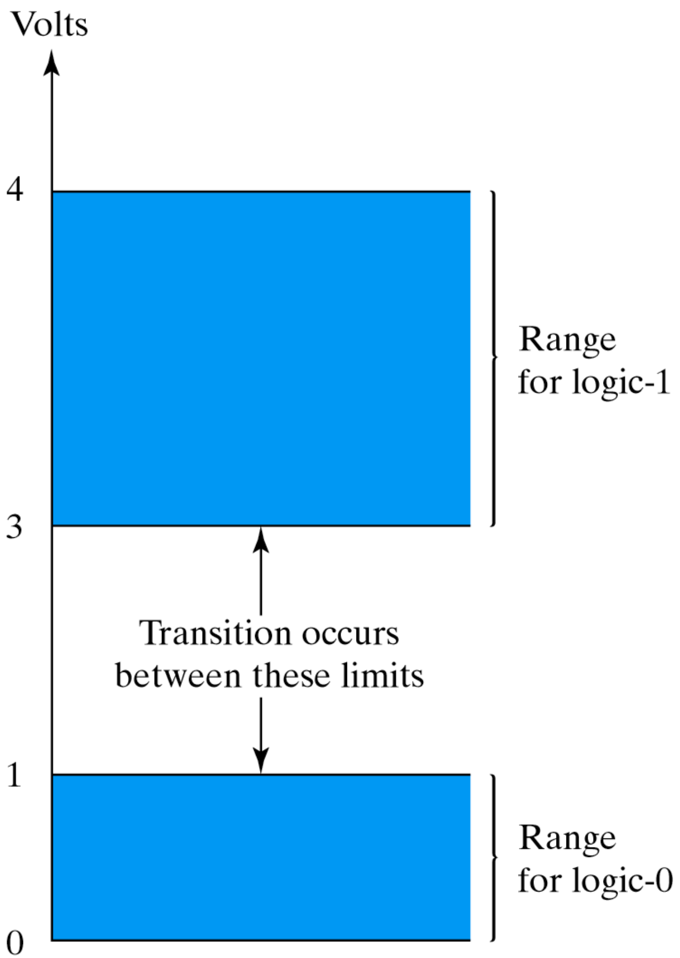
- Binary numbers are made of <u>b</u>inary dig<u>it</u>s (bits):
  - 0 and 1
- How many items does an binary number represent?
  - $(1011)_2 = 1x2^3 + 0x2^{2} + 1x2^{1} + 1x2^{0} = (11)_{10}$
- What about fractions?
  - $(110.10)_2 = 1x2^{2} + 1x2^{1} + 0x2^{0} + 1x2^{-1} + 0x2^{-2}$
- Groups of eight bits are called a ***byte***
  - $(11001001)_2$
- Groups of four bits are called a ***nibble***
  - $(1101)_2$

Binary Numbers

Presentation of information in binary form:

Area	Definition
Digital Design	**"0" and "1"**
Physics	**"*low*" and "*high*"**
Logic	**"*true*" and "*false*"**

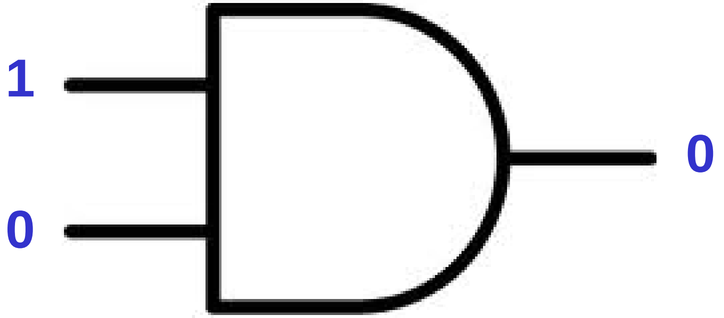
Why use Binary Numbers?



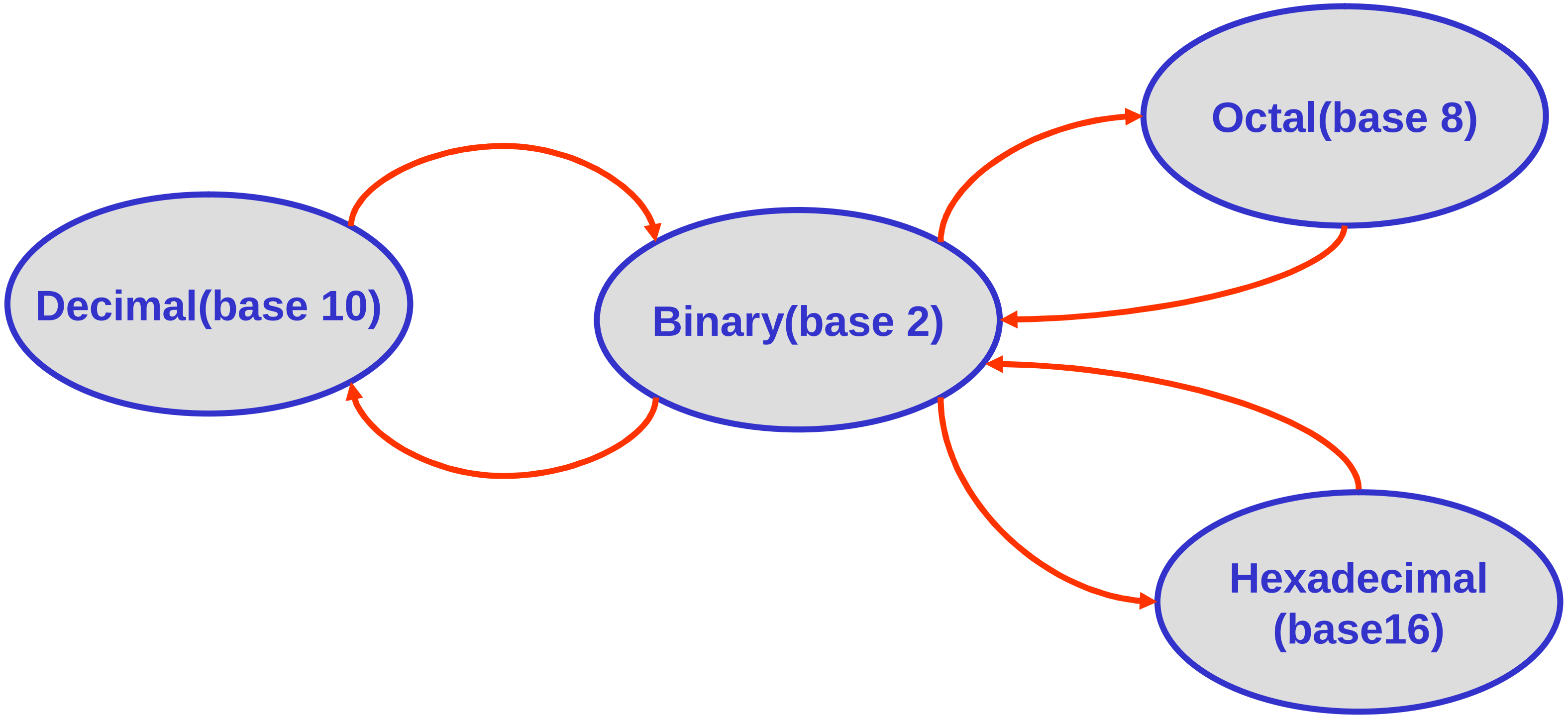
Example of binary signals

- Easy to represent 0 and 1 using electrical values.
- Possible to tolerate noise.
- Easy to transmit data
- Easy to build binary circuits.

AND Gate



Conversion Between Number Bases



- Learn how to convert between bases
- Convert from binary to decimal
- Hexadecimal comes later

Convert from Decimal

Convert an Integer from Decimal to Another Base:

For each digit position:

- Divide decimal number by the base (e.g. 2)
- The *remainder* is the lowest-order digit
- Repeat first two steps until no *divisor* remains.

Example for $(13)_{10}$:

		Integer Quotient		Remainder	Coefficient
13/2	=	6	+	1	$a_0 = 1$
6/2	=	3	+	0	$a_1 = 0$
3/2	=	1	+	1	$a_2 = 1$
1/2	=	0	+	1	$a_3 = 1$

Answer $(13)_{10} = (a_3\, a_2\, a_1\, a_0)_2 = (1101)_2$

Convert from Decimal

Convert a Fraction from Decimal to Another Base:

For each digit position:

1. Multiply decimal number by the base (e.g. 2)
2. The *integer* is the highest-order digit
3. Repeat first two steps until fraction becomes zero.

Example for $(0.625)_{10}$:

	Integer		Fraction	Coefficient
0.625 x 2 =	1	+	0.25	$a_{-1} = 1$
0.250 x 2 =	0	+	0.50	$a_{-2} = 0$
0.500 x 2 =	1	+	0	$a_{-3} = 1$

Answer $(0.625)_{10} = (0.a_{-1}\,a_{-2}\,a_{-3})_2 = (0.101)_2$

The Growth of Binary Numbers

n	2^n
0	$2^0 = 1$
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
5	$2^5 = 32$
6	$2^6 = 64$
7	$2^7 = 128$

n	2^n	
8	$2^8 = 256$	
9	$2^9 = 512$	
10	$2^{10} = 1024$	
11	$2^{11} = 2048$	
12	$2^{12} = 4096$	
20	$2^{20} = 1M$	Mega
30	$2^{30} = 1G$	Giga
40	$2^{40} = 1T$	Tera

And then: peta, exa, zetta, yotta, xona, weka, vunda, uda, treda, sorta, rinta, quexa, pepta, ocha, nena, minga, luma., etc.

Binary Addition

- Binary addition is simple.
- Example of adding two binary numbers:

		1	1	1	1	0	1
+			1	0	1	1	1
Carry bits		*1*	*1*	*1*	*1*	*1*	*1*
	---	---	---	---	---	---	---
Sum	1	0	1	0	1	0	0

Binary Subtraction

- Subtraction can also be performed (with borrows in place of carries).
- Example: subtract $(10111)_2$ from $(1001101)_2$

$$
\begin{array}{cccccccccc}
 & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & 1 & & \\
- & & & 1 & 0 & 1 & 1 & 1 & & \\
 & & 1 & & & 10 & & & & \\
 & 0 & \cancel{10} & 10 & 0 & \cancel{0} & 10 & & \longleftarrow & \text{borrows} \\
\hline
 & & 1 & 1 & 0 & 1 & 1 & 0 & &
\end{array}
$$

Binary Multiplication

- Binary multiplication is just like decimal multiplication, except that the multiplication operations are much simpler:

```
              1   0   1   1   1
x                 1   0   1   0
-----------------------
              0   0   0   0   0
          1   0   1   1   1
      0   0   0   0   0
  1   0   1   1   1
-----------------------
  1   1   1   0   0   1   1   0
```

Convert from Decimal

Convert an Integer from Decimal to Octal:

For each digit position:

1. Divide decimal number by the base (8)
2. The *remainder* is the lowest-order digit
3. Repeat first two steps until no *divisor* remains.

Example for $(175)_{10}$:

	Integer Quotient		Remainder	Coefficient
175 / 8 =	21	+	7/8	$a_0 = 7$
21 / 8 =	2	+	5/8	$a_1 = 5$
2 / 8 =	0	+	2/8	$a_2 = 2$

Answer $(175)_{10} = (a_2\, a_1\, a_0)_2 = (257)_8$

Convert from Decimal

Convert a Fraction from Decimal to Octal:

For each digit position:

1. Multiply decimal number by the base (e.g. 8)
2. The *integer* is the highest-order digit
3. Repeat first two steps until fraction becomes zero.

Example for $(0.3125)_{10:}$

	Integer		Fraction	Coefficient
0.3125 x 8 =	2	+	5	$a_{-1} = 2$
0.5000 x 8 =	4	+	0	$a_{-2} = 4$

Answer $(0.3125)_{10} = (0.24)_8$

Summary

- Binary numbers are made of binary digits (bits)
- Binary and octal number systems
- Conversion between number systems
- Addition, subtraction, and multiplication in binary

More Number Systems

Overview

- Hexadecimal numbers
  - Related to binary and octal numbers
- Conversion between hexadecimal, octal and binary
- Value ranges of numbers
- Representing positive and negative numbers
- Creating the complement of a number
  - Make a positive number negative (and vice versa)
- Why binary?

Rep.: Understanding Binary Numbers

- Binary numbers are made of binary digits (bits):
  - 0 and 1
- How many items does an binary number represent?
  - $(1011)_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = (11)_{10}$
- What about fractions?
  - $(110.10)_2 = 1x2^2 + 1x2^1 + 0x2^0 + 1x2^{-1} + 0x2^{-2}$
- Groups of eight bits are called a *byte*
  - $(11001001)_2$
- Groups of four bits are called a *nibble.*
  - $(1101)_2$

Understanding Hexadecimal Numbers

Transistion from the octal to the hexadecimal number system.

Example:

octal: 010 000 010 100 001 001 000 011

reorganizing:

hexadecimal: 0100 0001 0100 0010 0100 0011

Hexdecimal Value	Decimal Value
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Symbols to define all hexadecimal numbers:

0 1 2 3 4 5 6 7 8 9 A B C D E F

Understanding Hexadecimal Numbers

- Hexadecimal numbers are made of 16 digits:
  - (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)
- How many items does a hex number represent?
  - $(3A9F)_{16} = 3x16^3 + 10x16^2 + 9x16^1 + 15x16^0 = 14999_{10}$
- What about fractions?
  - $(2D3.5)_{16} = 2x16^2 + 13x16^1 + 3x16^0 + 5x16^{-1} = 723.3125_{10}$
- Note that *each* hexadecimal digit can be represented with four bits.
  - $(1110)_2 = (E)_{16}$
- Groups of four bits are called a *nibble.*
  - $(1110)_2$

Putting it all together

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

- Binary, octal, and hexadecimal similar
- Easy to build circuits to operate on these representations
- Possible to convert between the three formats

Converting between base 16 and base 2

- Conversion is easy!
  - Determine 4-bit value for each hex digit
- Note that there are $2^4 = 16$ different values of four bits
- Easier to read and write in hexadecimal.
- Representations are equivalent!

Example:

$$3A9F_{16} = \underset{3}{\underline{0011}}\ \ \underset{A}{\underline{1010}}\ \ \underset{9}{\underline{1001}}\ \ \underset{F}{\underline{1111}}_2$$

Converting between base 16 and base 8

1. Convert from Base 16 to Base 2
2. Regroup bits into groups of three starting from right
3. Ignore leading zeros
4. Each group of three bits forms an octal digit.

Example:

$$3A9F_{16} = \underset{3}{\underline{0011}}\ \ \underset{A}{\underline{1010}}\ \ \underset{9}{\underline{1001}}\ \ \underset{F}{\underline{1111}}{}_2$$

$$\downarrow$$

$$35237_8 = \underset{3}{\underline{011}}\ \ \underset{5}{\underline{101}}\ \ \underset{2}{\underline{010}}\ \ \underset{3}{\underline{011}}\ \ \underset{7}{\underline{111}}{}_2$$

How to represent signed numbers

- Plus and minus sign used for decimal numbers:
  25 (or +25), -16, etc.
- For computers, desirable to represent everything as *bits.*
- Three types of signed binary number representations:
  - signed magnitude
  - 1’s complement (One's complement)
  - 2’s complement (Two's complement)
- In each case: left-most bit indicates sign:
  positive (0) or negative (1).

Signed magnitude:

$0\underline{0001100}_2 = 12_{10}$

Sign bit

Magnitude

$1\underline{0001100}_2 = -12_{10}$

Sign bit

Magnitude

One's Complement Representation

- The one’s complement of a binary number involves inverting all bits.
    - 1’s comp of 00110011 is 11001100
    - 1’s comp of 10101010 is 01010101
- For an n bit number N the 1’s complement is $(2^n-1) – N$.
- Also called diminished radix complement.
- To find negative of 1’s complement number take the 1’s complement.

One's Complement:

$00001100_2 = 12_{10}$

$11110011_2 = -12_{10}$

Two's Complement Representation

- The two’s complement of a binary number involves inverting all bits and adding 1.
  - 2’s comp of 00110011 is 11001101
  - 2’s comp of 10101010 is 01010110
- For an n bit number N the 2’s complement is $(2^n-1) - N + 1$.
- Also called radix complement.
- To find negative of 2’s complement number take the 2’s complement.

Two's Complement:

$00001100_2 = 12_{10}$

Sign bit

Magnitude

$11110100_2 = -12_{10}$

Sign bit

Magnitude

Two's Complement Shortcuts

Algorithm 1:

Simply complement each bit and then add 1 to the result.

Finding the 2’s complement of $(01100101)_2$ and of its 2’s complement:

```
N    = 01100101        [N] = 10011011
       10011010              01100100
     +        1            +        1
       ---------           ----------
       10011011              01100101
```

Algorithm 2:

Starting with the least significant bit, copy all of the bits up to and including the first 1 bit and then complementing the remaining bits.

```
N   = 01100101         [N] = 10011011
```

Finite Number Representation

- Machines that use 2's complement arithmetic can represent integers in the range

$$-2^{n-1} <= N <= 2^{n-1}-1$$

where n is the number of bits available for representing N.

Note that $2^{n-1}-1 = (011...11)_2$

and $-2^{n-1} = (100...00)_2$

- For 2's complement more negative numbers than positive.
- For 1's complement two representations for zero.
- For an n bit number in base (radix) z there are z^n different unsigned values.

$$(0, 1, \ldots , z^{n-1})$$

One's Complement Addition

- Using 1’s complement numbers, adding numbers is easy.
- For example, suppose we wish to add $+(1100)_2$ and $+(0001)_2$.
- Let’s compute $(12)_{10} + (1)_{10}$.

$(12)_{10} = +(1100)_2 = 01100_2$ in 1’s comp.

$(1)_{10} = +(0001)_2 = 00001_2$ in 1’s comp.

Step 1: Add binary numbers
Step 2: Add carry to low-order bit

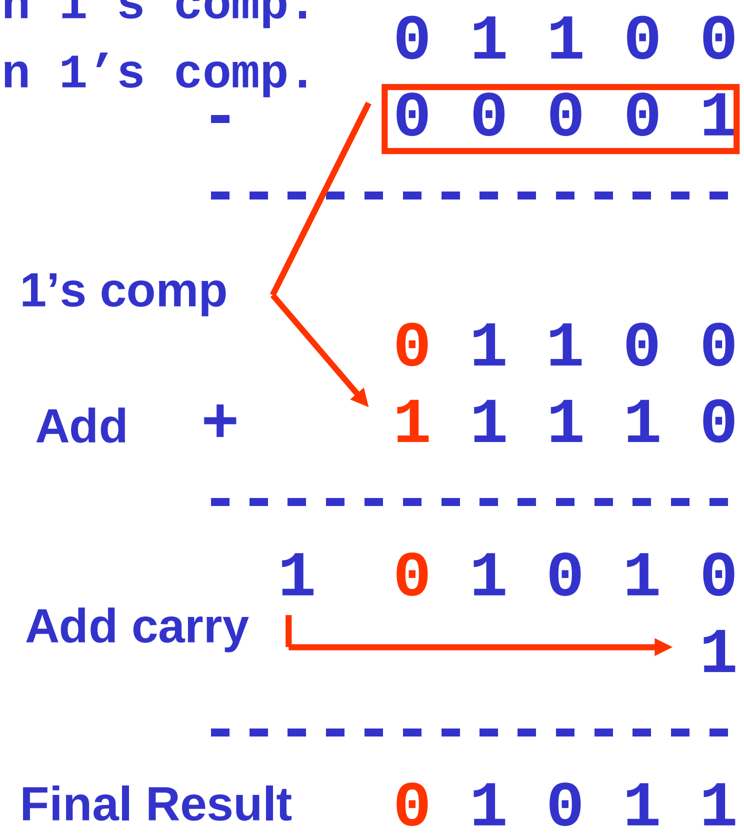
			0 1 1 0 0
Add	+		0 0 0 0 1
	- - - - - - - - - - - - - -		
		0	0 1 1 0 1
Add carry			0
	- - - - - - - - - - - - - -		
Final Result			0 1 1 0 1

One's Complement Subtraction

- Using 1’s complement numbers, subtracting numbers is also easy.
- For example, to subtract $+(0001)_2$ from $+(1100)_2$.
- Let’s compute $(12)_{10} - (1)_{10}$.

$(12)_{10} = +(1100)_2 = 01100_2$ in 1’s comp.

$(-1)_{10} = -(0001)_2 = 11110_2$ in 1’s comp.



0 1 1 0 0

\- 0 0 0 0 1

\-\-\-\-\-\-\-\-\-\-\-\-\-\-

1’s comp

0 1 1 0 0

Add + 1 1 1 1 0

\-\-\-\-\-\-\-\-\-\-\-\-\-\-

1 0 1 0 1 0

Add carry 1

\-\-\-\-\-\-\-\-\-\-\-\-\-\-

Final Result 0 1 0 1 1

Step 1: Take 1’s complement of 2nd operand

Step 2: Add binary numbers

Step 3: Add carry to low order bit

Two's Complement Addition

- **Using 2’s complement numbers, adding numbers is easy.**
- **For example, suppose we wish to add $+(1100)_2$ and $+(0001)_2$.**
- **Let’s compute $(12)_{10} + (1)_{10}$.**

$(12)_{10} = +(1100)_2 = 01100_2$ **in 2’s comp.**

$(1)_{10} = +(0001)_2 = 00001_2$ **in 2’s comp.**

Step 1: Add binary numbers
Step 2: Ignore carry bit

$$\begin{array}{lcc} & & 0\;\;1\;\;1\;\;0\;\;0 \\ \textbf{Add} \qquad + & & 0\;\;0\;\;0\;\;0\;\;1 \\ \hline \textbf{Final Result} & \underset{\substack{\uparrow \\ \textbf{Ignore}}}{0} & \boxed{0\;\;1\;\;1\;\;0\;\;1} \end{array}$$

Two's Complement Subtraction

- Using 2’s complement numbers, follow steps for subtraction
- For example, suppose we wish to subtract +$(0001)_2$ from +$(1100)_2$.
- Let’s compute $(12)_{10}$ - $(1)_{10}$.

$(12)_{10} = +(1100)_2 = 01100_2$ in 2’s comp.

$(-1)_{10} = -(0001)_2 = 11111_2$ in 2’s comp.

```
    0 1 1 0 0
-   0 0 0 0 1
-------------
```

Step 1: Take 2’s complement of 2nd operand

Step 2: Add binary numbers

Step 3: Ignore carry bit

```
2's comp          0 1 1 0 0
Add           +   1 1 1 1 1
               -------------
Final Result    1 0 1 0 1 1
```

Ignore Carry

Two's Complement Subtraction: Example 2

- Let’s compute $(13)_{10} - (5)_{10}$.

$$\begin{array}{lll} (13)_{10} & = +(1101)_2 & = (01101)_2 \\ (-5)_{10} & = -(0101)_2 & = (11011)_2 \end{array}$$

- Adding these two 5-bit codes:

$$\begin{array}{rcccccc} & & 0 & 1 & 1 & 0 & 1 \\ + & & 1 & 1 & 0 & 1 & 1 \\ \hline \text{ignore carry} \longrightarrow & 1 & 0 & 1 & 0 & 0 & 0 \end{array}$$

- Discarding the carry bit, the sign bit is seen to be zero, indicating a correct result. Indeed,

$$(01000)_2 = +(1000)_2 = +(8)_{10}.$$

Two's Complement Subtraction: Example 3

- Let’s compute $(5)_{10} - (12)_{10}$.

$$(-12)_{10} = -(1100)_2 = (10100)_2$$

$$(5)_{10} = +(0101)_2 = (00101)_2$$

- Adding these two 5-bit codes…

$$\begin{array}{rccccc} & 0 & 0 & 1 & 0 & 1 \\ + & 1 & 0 & 1 & 0 & 0 \\ \hline \text{no carry} \longrightarrow & 1 & 1 & 0 & 0 & 1 \end{array}$$

- Here, there is no carry bit and the sign bit is 1.
  This indicates a negative result, which is what we expect.

$$(11001)_2 = -(7)_{10}.$$

Summary

- Binary numbers can also be represented in octal and hexadecimal
- Easy to convert between binary, octal, and hexadecimal
- Signed numbers represented in signed magnitude, 1’s complement, and 2’s complement
- 2’s complement most important (only 1 representation for zero).
- Important to understand treatment of sign bit for 1’s and 2’s complement.

Digital Design

Polyadic Number Systems

Polyadic Systems

All number systems introduced so far belong to the so-called polyadic number systems. A polyadic number system to the base B (also called B-adic number system), is a system in which a number is formed as a sum of powers of B.

A natural number n can be represented as the following sum of powers:

$$n = \sum_{i=0}^{N} b_i \cdot B^i$$

With B as base of the number system: $B \in N, \quad B \geq 2$

and $b_i \in N \cup \{0\}, \quad 0 \leq b_i < B$

Polyadic Systems

Thus the number systems can be defined by chosing:

Base	Number System
2	Binary
8	Octal
10	Decimal
16	Hexadecimal

This leads to the system-independent writing:

$$n = (b_N b_{N-1} b_{N-2} \cdots b_1 b_0)_2$$

with:

- leading zeros can be omitted
- when the base is known it can be omitted

Polyadic Systems

Example:

$$(01000001010000100100000 11)_2$$

$$=(1000001010000100100000 11)_2$$

$$=(20241103)_8$$

$$=(414243)_{16}$$

End

Thank you!