



University of
Massachusetts
Amherst

Engin112 – Lectures 2-4

Number Systems

Maciej Ciesielski
Department of Electrical and Computer Engineering
09/09-14/2011

Digital Systems

- Digital systems operate on discrete elements of information
 - Numbers (e.g., pocket calculator) -> “digits” -> “digital”
 - Letters (e.g., word processor)
 - Pictures (e.g., digital cameras)

- For a digital systems to operate on a continuous data, it needs to quantize (digitize) that data first
 - Covert data into digital representation

- Topics:
 - How are numbers represented in digital systems
 - How computer performs basic arithmetic operations

Numbers

- Common numbering system is “base10”

- Why?



- Numbers in base 10

- Ten different digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Number is represented by a sequence of digits: $a_n a_{n-1} \dots a_1 a_0$
- Value of number is: $a_n \times 10^n + a_{n-1} \times 10^{n-1} + \dots + a_1 \times 10^1 + a_0 \times 10^0$

- Positional notation

- General equation: $\sum_i a_i \times 10^i$
- May contain a decimal point
- Negative index for digits after decimal point

- Examples

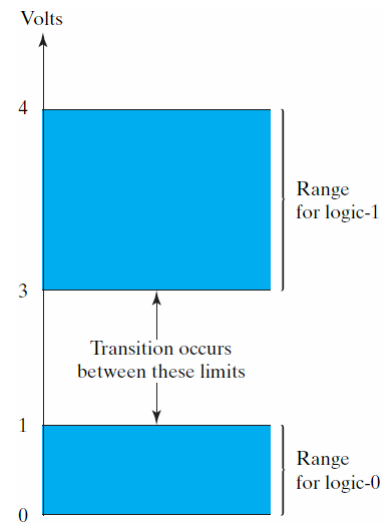
- 1234.56 – if ambiguous, write $(1234.56)_{10}$
- Leading zeros cause no problems: 00001234.56

Number Systems

- General form, with base r : $\sum_i a_i \times r^i$
- Base r is also called *radix*
 - In decimal system $r = 10$; in binary $r = 2$
- Coefficients in positional notation are: $0, 1, \dots, r-1$.
- What is the range of values of an n -bit number in radix r ?
 - Minimum value: 0
 - Maximum value: $r^n - 1$
 - Number of different values: r^n

Binary Numbers

- Base 2 number use only two digits: 0, 1
 - Why?
- Digits need to be represented in a system
 - Electronic systems typically use voltage levels
 - Representing 10 different voltages reliably is difficult
 - Binary decision is much easier (On, Off)
- Binary representation is ideal
 - Minimal number of digits
 - Easily represented in voltages
- Catch: humans require training 😊



Examples for Binary Numbers

- What value is represented by $(01001)_2$?
 - Leading zero makes no difference
 - $(1001)_2$ translates into $1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 0 + 0 + 1 = (9)_{10}$
- Same process for numbers with decimal point
 - What is the value of $(1001.1001)_2$?
 - $(1001.1001)_2 =$
 $1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} =$
 $8 + 0 + 0 + 1 + 1/2 + 0 + 0 + 1/16 = (9.5625)_{10}$
 - Important: it's NOT $(9.9)_{10}$!
- Can you count binary?
 - How far can you count with 10 fingers?

Binary Number Terminology

- Base is also called “radix”
- Binary numbers are made of binary digits (*bits*)
- Groups of four bits are called “nibbles”
 - E.g., $(1101)_2$
- Groups of eight bits are called “bytes”
 - E.g., $(01001101)_2$
- What is the range of values of an n -bit binary number?
 - Minimum value: 0
 - Maximum value: $2^n - 1$
 - Number of different values: 2^n
- Powers of 2 are important in Computer Engineering

Powers of 2

- You must memorize all powers of 2 up to 2^{16} !

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536

- Other important powers of 2:

8	16	24	32	64
256	65536	16777216	4294967296	18446744073709551616

- Trick to simplify estimation:
 - $2^{10} = 1024 \approx 1000 = 10^3$
 - Example: $2^{32} = 4 \times 2^{30} \approx 4 \times 10^9 = 4$ billion
- Prefixes:
 - kilo ($10^3 \approx 2^{10}$), Mega ($10^6 \approx 2^{20}$), Giga ($10^9 \approx 2^{30}$), Tera ($10^{12} \approx 2^{40}$), ...
 - In computer systems typically based on powers of 2


Memory Size

- Definition of *Mega* and *Giga* can make a difference!
- Disk drive manufacturers define it “conveniently”
- How many more MP3 songs could you store on this 80GB disk if “GB” was based on the real computer systems definition?

- Difference is $80 \times 2^{30} - 80 \times 10^9 = 5,899,345,920$ bytes
- 128kbit MP3 coding: 16kB per second
- $5,899,345,920 / 16,384 \approx 360000$ s. = 100 hours of music

Products Hard Disk Drives

Super Slimline
80.0 GB:
MK8025GAS



Download HDD Fact Sheet PDF

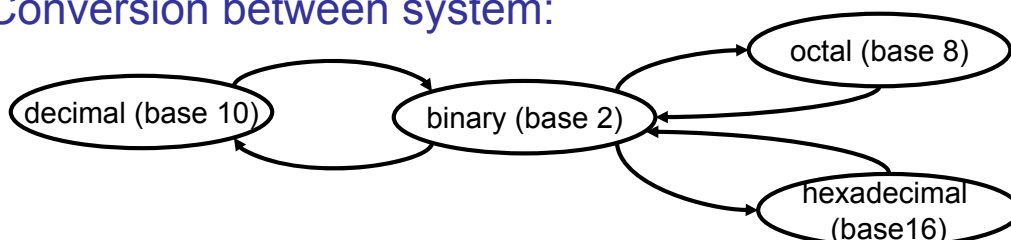
Providing capacious storage in a slim form factor, Toshiba's 80GB 2.5-inch hard disk drive is an ideal solution for mobile computing and consumer electronics. With an areal density of 64.8 gigabits per square inch and a super slim package measuring 9.5mm high, Toshiba's MK8025GAS hard disk drive accommodates the storage needs of multi-functional mobile PCs and other applications, such as telecommunications products, printers, copiers, GPS systems and MP3 players.

- 80GB capacity
- Fluid Dynamic Bearing (FDB) motor drive
- 9.5mm High
- 12ms Average Seek Time
- ATA-6 Interface
- 100MB/sec Transfer Rate
- 8MB Buffer
- 300,000 MTTF Hours
- Click for Warranty Information

*Toshiba defines a megabyte (MB) as 1,000,000 bytes and a gigabyte (GB) as 1,000,000,000 bytes.

Other Number Systems

- Octal number system
 - Digits 0, 1, ... 7
 - Aggregates 3 digits of binary system (*i.e.*, 3-bit number)
 - E.g., $(57)_8 = (47)_{10}$
- Hexadecimal number system
 - 16 digits require 6 new digit symbols: 0, ... 9, A, B, C, D, E, F
 - Aggregates 4 digits of binary system (1 nibble or half byte)
 - E.g., $(1F)_{16} = (31)_{10}$
- Conversion between system:



Number Base Conversions

- Conversion from base r to decimal
 - Expansion to power series $\sum_i a_i \times r^i$ and addition of terms
- Conversion from decimal to base r
 - Divide number and successive quotients by r
 - Sequence of remainders is base r number
- Example: convert $(41)_{10}$ to binary ($r=2$)
 - $41 = 20 \times 2 + 1$ $a_0 = 1$
 - $20 = 10 \times 2 + 0$ $a_1 = 0$
 - $10 = 5 \times 2 + 0$ $a_2 = 0$
 - $5 = 2 \times 2 + 1$ $a_3 = 1$
 - $2 = 1 \times 2 + 0$ $a_4 = 0$
 - $1 = 0 \times 2 + 1$ $a_5 = 1$

$(101001)_2$

Number Base Conversions

- Conversion to/from octal and hexadecimal
 - Easier if done via binary
 - 3 or 4 bit sequences correspond to digit
- Example:

$$(2414)_{10} = \underbrace{(100101101110)}_2 = \underbrace{(9 \ 6 \ E)}_{16} = \underbrace{(4 \ 5 \ 5 \ 6)}_8$$
- Direct conversion possible, too:
 - $2414 = 301 \times 8 + 6$
 - $301 = 37 \times 8 + 5$
 - $37 = 4 \times 8 + 5$
 - $4 = 0 \times 8 + 4 \Rightarrow (4 \ 5 \ 5 \ 6)_8$

Number Systems - Summary

- Base r numbers
 - Binary
 - Octal
 - Hexadecimal
- Conversion between number systems
 - Summation of power series
 - Division with remainders
- Powers of 2
 - Computer systems based on powers of 2
 - kilo = $2^{10} = 1024$
 - Mega = $2^{20} = (1024)^2 = 1,048,576$
 - Giga = $2^{30} = (1024)^3 = 1,073,741,824$
- Next:
 - Computer arithmetic
 - Signs, complements

Friendly Tips

- Today: professional communication
- You are preparing to become a professional
 - Instant Messenger is NOT professional communication

I think that writing appropriately phrased and formatted emails is important. Using slang and Instant Messenger abbreviations looks unprofessional.



I THINK TAHT R1TNG APROPRI8LY PHRAESD + 4MATED UMALLZ IZ IMPORTANT!! OMG WTF US1NG SLANG + INSTANT MESENGUR ABR3VIASHUNZ LOKZ UNPROF3S1ONAL!!!! LOL

- Use UMass email addresses
 - I will not respond to e1_duderino284@schoolsucks.com, biteme44@hotmail.com, or loverboy@ineedtogrowup.org
- Information lives a long time on the Internet
 - Google caches are amazing repositories of interesting stuff
- Eventually, you want a job. Practice proper communication now!

Computer Arithmetic

- Basic operations
 - Addition, subtraction
 - Multiplication
- Number systems
 - Signed numbers
 - Complements

Binary Addition

- Binary addition works like “normal” addition
 - Stay within {0,1}
 - Carries as usual

- Example:

$$\begin{array}{r} \text{Carry: } 111111 \\ 111101 \\ + 10111 \\ \hline 1010100 \end{array}$$

- Addition of multiple numbers possible
 - Carry gets a bit more difficult

Binary Multiplication

- Binary multiplication
 - Same as “normal” multiplication
 - Multiplication a lot easier in binary domain
- Example:

$$\begin{array}{r} 111101 \\ \times 1010 \\ \hline 000000 \\ 111101 \\ 000000 \\ 111101 \\ \hline 100110010 \end{array}$$

Binary Subtraction

- Subtraction same as “normal” subtraction
 - Borrows as usual
- Example:

$$\begin{array}{r} 10 \\ 0 \oplus 10 \\ 111101 \\ - 10111 \\ \hline 100110 \end{array}$$

But subtraction is NOT done this way !

Signed Numbers

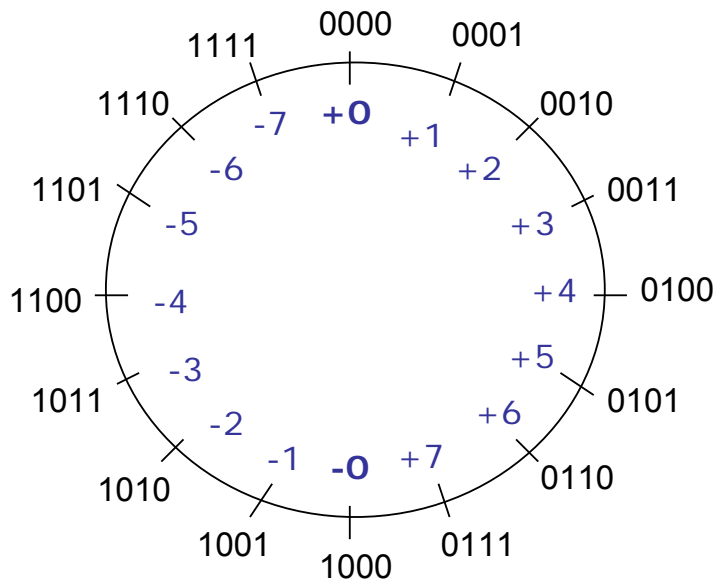
- How are signed numbers handled in base 10?
 - Plus or minus sign placed in front of number
- Can we do that for binary numbers?
 - Sign needs to be represented in digital system
 - Only choice are '0' and '1'
 - » '0' indicates '+'
 - » '1' indicates '-'
- Examples on five-bit numbers:

• $\underbrace{01101}_{+13}$	$\underbrace{11101}_{-13}$	$\underbrace{00000}_{+0}$	$\underbrace{10000}_{-0}$
------------------------------	----------------------------	---------------------------	---------------------------

- Signed Magnitude representation

Signed Magnitude Representation

- Arithmetic with signed magnitude is difficult
 - Two representations of zero
 - Different cases for addition and subtraction



Sign bit



0101 = +5

1101 = -5

Arithmetic with Signed Magnitude

Addition example 1:

- Plus signs are leading zeros -> no problem

$$\begin{array}{r} 01001 \\ + 00010 \\ \hline 01011 \end{array} \quad \begin{array}{r} (9)_{10} \\ + (2)_{10} \\ \hline (11)_{10} \end{array}$$

Addition example 2:

- What happens with negative numbers?

$$\begin{array}{r} 01001 \\ + 10010 \\ \hline 11011 \end{array} \quad \begin{array}{r} (9)_{10} \\ + (-2)_{10} \\ \hline (7)_{10} \end{array} \quad \text{???$$

Problem: negative numbers

- Sign can turn addition into subtraction

Signed Magnitude - Problems

- Need to consider many cases to implement addition correctly
- Decision process for $(\pm A) + (\pm B)$:

1. If $A \geq B$ then:

		A		
		+	-	magnitude calculation
B	+	(A+B), +	(A-B), -	
	-	(A-B), +	(A+B), -	sign of result

2. If $A < B$ then:

		A	
		+	-
B	+	(A+B), +	(B-A), +
	-	(B-A), -	(A+B), -

Complements

- “Complements” allow easier arithmetic
 - Representation of negative numbers a bit more involved
- Two types of complements
 - Radix complement: r 's complement
 - » Decimal: 10's complement
 - » Binary: 2's complement
 - Diminished radix complement: $(r-1)$'s complement
 - » Decimal: 9's complement
 - » Binary: 1's complement

Diminished Radix Complement

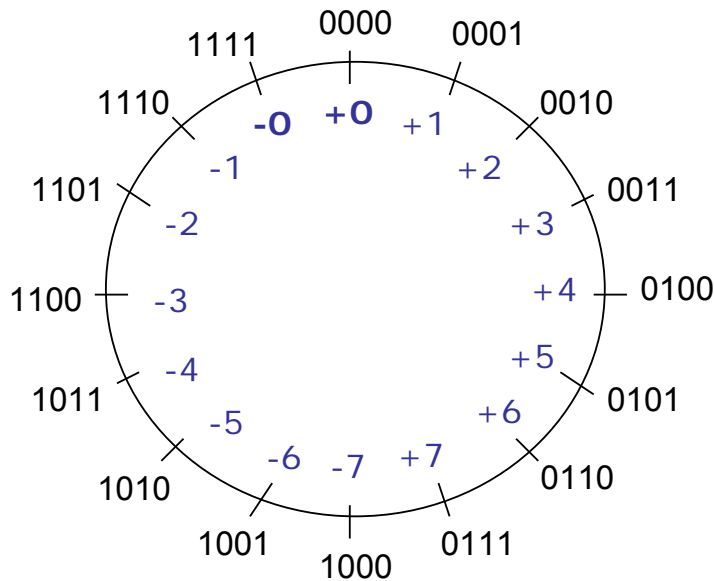
- Given a number N in base r having n digits, the $(r-1)$'s complement of N is defined as:

$$(r^n - 1) - N$$

- Example for 6-digit decimal numbers:
 - 9's complement is $(r^n - 1) - N = (10^6 - 1) - N = 999999 - N$
 - 9's complement of 546700 is $999999 - 546700 = 453299$
- Example for 7-digit binary numbers:
 - 1's complement is $(r^n - 1) - N = (2^7 - 1) - N = 1111111 - N$
 - 1's complement of 1011000 is $1111111 - 1011000 = 0100111$
- Observation:
 - Subtraction from $(r^n - 1)$ will never require a borrow
 - Diminished radix complement can be computed digit-by-digit
 - For binary: $1 - 0 = 1$ and $1 - 1 = 0$
 - » Flips 0's to 1's and 1's to 0' (bit complementation)

Ones Complement Representation

- 1's complement, simple to compute: $N = (2^n - 1) - N$
 - Complement bit by bit
 - Still, two representations of zero



$$N = 0101 = +5$$

Subtract from $2^n - 1$

$$\begin{array}{r} 1111 \\ - 0101 \\ \hline 1010 = -5 \end{array}$$

Or just flip the bits!

Radix Complement

- The r 's complement of an n -digit number N in base r is defined as

$$r^n - N \text{ for } N \neq 0 \text{ and } 0 \text{ for } N = 0$$

- Radix complement is diminished radix complement + 1:

$$(r^n - 1) - N + 1 = r^n - N$$

- Example for 6-digit decimal numbers:

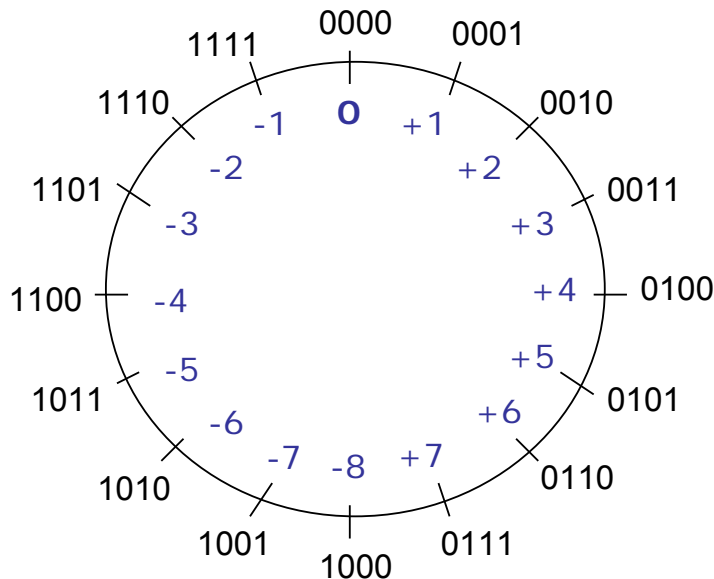
- 10's complement is $r^n - N = 10^6 - N = 1000000 - N$
- 10's complement of 546700 is $1000000 - 546700 = 453300$
- Rule: Leave least significant 0's unchanged, subtract first nonzero least significant digit from 10, subtract all higher significant digits from 9.

- Example for 7-digit binary numbers:

- 2's complement is $r^n - N = 2^7 - N = 10000000 - N$
- 2's complement of 1011000 is $10000000 - 1011000 = 0101000$
- Rule: "Leave least significant 0's and first 1 unchanged, replace 1's with 0's and 0's with 1's in all higher significant digits."

Two's Complement Representation

- Arithmetic with 2's complement: $N = 2^n - N = \underbrace{(2^n - 1) - N + 1}_{1\text{'s complement}}$
 - Bit complementation + 1
 - Single representation of zero!



$$N = 0101 = +5$$

Subtract from 2^n

$$\begin{array}{r} 10000 \\ - 0101 \\ \hline 1011 = -5 \end{array}$$

Or add 1 to 1's complement of N

$$\begin{array}{r} 1010 \\ + 1 \\ \hline 1011 = -5 \end{array}$$

Subtraction with 2's Complements

- Subtraction $M - N$
 - Add minuend M to 2's complement of subtrahend N
 - $M - N = M + (r^n - N) = r^n + M - N$
- What about the additional r^n ?
- If $M \geq N$ (result is positive)
 - $r^n + M - N \geq r^n$
 - With n bits we can only express numbers $< r^n$
 - $n+1^{\text{st}}$ digit is ignored (ignore the carry out bit)
- If $M < N$ (result is negative)
 - $r^n + M - N = r^n - (N - M) = r$'s complement of $(N - M)$
 - r 's complement signifies negative number
 - There is no carry out (the number fits in n bits)
- If both M and N are negative
 - $r^n - M + r^n - N = r^n + (r^n - (M + N)) > r^n$
 - ignore carry

Twos Complement Calculations

- Adding two positive numbers

4	0100	
+3	0011	
---	-----	
7	0111	MSB = 1 indicates overflow!

- Subtracting two numbers, $M \geq N$

4	0100	
- 3	1101	
---	-----	
1	10001	Ignore carry

- Subtracting two numbers, $M < N$

3	0011	
- 4	1100	
---	-----	
-1	1111	No carry

- Adding two negative numbers

-4	1100	
-3	1101	
---	-----	
-7	11001	Ignore carry

Complements - Summary

- Complement of complement is original number

 - Diminished radix complement:

$$(r^{n-1}) - ((r^{n-1}) - N) = r^{n-1} - r^{n-1} + N = N$$
 - Radix complement:

$$r^n - (r^n - N) = r^n - r^n + N = N$$

- Representation of zero

 - Radix complement: 0
 - Diminished radix complement: $r^n - 1$ and its complement 0

- Radix points

 - Remove radix point
 - Compute complement
 - Put radix point back at same relative position

Signed Numbers with Complements

- 3-bit numbers:

Decimal	Signed 2's complement	Signed 1's complement	Signed Magnitude
+3	011	011	011
+2	010	010	010
+1	001	001	001
0	000	000	000
-0	----	111	100
-1	111	110	101
-2	110	101	110
-3	101	100	111
-4	100	----	----

- Bit sequence does not indicate coding
 - Can be interpreted as signed or unsigned number!

Arithmetic with Radix Complement

- Subtraction works as addition of complement
- Addition/subtraction of signed numbers
 - Negative numbers are expressed as r 's complement
 - Simple addition yields correct result
 - No need to distinguish different cases
- Addition/subtraction of unsigned numbers
 - Can be performed on same hardware as signed numbers
- Most modern digital systems use 2's complement