



**UNIVERSITY OF MASSACHUSETTS**  
**Dept. of Electrical & Computer Engineering**

**Digital Computer Arithmetic**

**ECE 666**

**Part 1**  
**Introduction**

**Israel Koren**  
**Spring 2008**

ECE666/Koren Part.1 .1

Copyright 2008 Koren

**Prerequisites and textbook**

- ◆ Prerequisites: courses in
  - \* Digital Design
  - \* Computer Organization/Architecture
- ◆ Recommended book: *Computer Arithmetic Algorithms*, I. Koren, 2nd Edition, A.K. Peters, Natick, MA, 2002
- ◆ Textbook web page:  
<http://www.ecs.umass.edu/ece/koren/arith>
- ◆ Recommended Reading:
  - \* **B. Parhami**, *Computer Arithmetic: Algorithms and Hardware Design*, Oxford University Press, 2000
  - \* **M. Ercegovac and T. Lang**, *Digital Arithmetic*, Morgan Kaufman, 2003

ECE666/Koren Part.1 .2

Copyright 2008 Koren

## Administrative Details

- ◆ **Instructor:** Prof. Israel Koren
- ◆ **Office:** KEB 309E, Tel. 545-2643
- ◆ **Email:** koren@ecs.umass.edu
- ◆ **Office Hours:** TuTh 2:30-3:30
- ◆ **Course web page:**  
<http://www.ecs.umass.edu/ece/koren/ece666/>
- ◆ **Grading:**
  - Homework - No credit
  - Two Mid-term exams - 25% each
  - Final Exam or Project - 50%

## Course Outline

- ◆ **Introduction: Number systems and basic arithmetic operations**
- ◆ **Unconventional fixed-point number systems**
- ◆ **Sequential algorithms for multiplication and division**
- ◆ **Floating-point arithmetic**
- ◆ **Algorithms for fast addition**
- ◆ **High-speed multiplication**
- ◆ **Fast division and division through multiplication**
- ◆ **Efficient algorithms for evaluation of elementary functions**
- ◆ **Logarithmic number systems**
- ◆ **Residue number system; error correction and detection in arithmetic operations**

## The Binary Number System

- ◆ In conventional digital computers - integers represented as binary numbers of fixed length  $n$
- ◆ An ordered sequence  $(x_{n-1}, x_{n-2}, \dots, x_1, x_0)$  of binary digits
- ◆ Each digit  $x_i$  (**bit**) is **0** or **1**
- ◆ The above sequence represents the integer value  $X$

$$X = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \dots + x_12 + x_0 = \sum_{i=0}^{n-1} x_i2^i$$

- ◆ Upper case letters represent numerical values or sequences of digits
- ◆ Lower case letters, usually indexed, represent individual digits

## Radix of a Number System

- ◆ The weight of the digit  $x_i$  is the  $i$  th power of **2**
- ◆ **2** is the **radix** of the **binary number system**
- ◆ Binary numbers are **radix-2** numbers - allowed digits are **0,1**
- ◆ Decimal numbers are **radix-10** numbers - allowed digits are **0,1,2,...,9**
- ◆ Radix indicated in subscript as a **decimal number**
- ◆ **Example:**
  - \*  $(101)_{10}$  - decimal value **101**
  - \*  $(101)_2$  - decimal value **5**

## Range of Representations

- ◆ Operands and results are stored in registers of fixed length  $n$  - finite number of distinct values that can be represented within an arithmetic unit
- ◆  $X_{min}$  ;  $X_{max}$  - smallest and largest representable values
- ◆  $[X_{min}, X_{max}]$  - range of the representable numbers
- ◆ A result larger than  $X_{max}$  or smaller than  $X_{min}$  - incorrectly represented
- ◆ The arithmetic unit should indicate that the generated result is in error - an **overflow indication**

## Example - Overflow in Binary System

- ◆ Unsigned integers with 5 binary digits (bits)
  - \*  $X_{max} = (31)_{10}$  - represented by  $(11111)_2$
  - \*  $X_{min} = (0)_{10}$  - represented by  $(00000)_2$
  - \* Increasing  $X_{max}$  by 1 =  $(32)_{10} = (100000)_2$
  - \* 5-bit representation - only the last five digits retained - yielding  $(00000)_2 = (0)_{10}$
- ◆ In general -
  - \* A number  $X$  not in the range  $[X_{min}, X_{max}] = [0, 31]$  is represented by  $X \bmod 32$
  - \* If  $X+Y$  exceeds  $X_{max}$  - the result is  $S = (X+Y) \bmod 32$
- ◆ Example:
 

$X$	$10001$	$17$
$+Y$	$10010$	$18$
	$100011$	$35$
		$= 3 \pmod{32}$
- \* Result has to be stored in a 5-bit register - the most significant bit (with weight  $2^5 = 32$ ) is discarded

## Machine Representations of Numbers

- ◆ **Binary system** - one example of a number system that can be used to represent numerical values in an arithmetic unit
- ◆ A **number system** is defined by the set of values that each digit can assume and by an interpretation rule that defines the mapping between the sequences of digits and their numerical values
- ◆ **Types of number systems** -
  - ◆ **conventional** (e.g., binary, decimal)
  - ◆ **unconventional** (e.g., signed-digit number system)

## Conventional Number Systems

- ◆ **Properties of conventional number systems:**
- ◆ **Nonredundant** -
  - \* Every number has a unique representation, thus
  - \* No two sequences have the same numerical value
- ◆ **Weighted** -
  - \* A sequence of weights  $W_{n-1}, W_{n-2}, \dots, W_1, W_0$  determines the value of the  $n$ -tuple  $X_{n-1}, X_{n-2}, \dots, X_1, X_0$  by
$$X = \sum_{i=0}^{n-1} x_i w_i.$$
  - \*  $W_i$  - weight assigned to  $X_i$  - digit in  $i$ th position
- ◆ **Positional** -
  - \* The weight  $W_i$  depends only on the position  $i$  of digit  $X_i$
  - \*  $W_i = r^i$

## Fixed Radix Systems

- ◆  $r$  - the radix of the number system
- ◆ **Conventional** number systems are also called **fixed-radix** systems
- ◆ With no redundancy -  $0 \leq x_i \leq r-1$
- ◆  $x_i \geq r$  introduces redundancy into the fixed-radix number system
- ◆ If  $x_i \geq r$  is allowed -
 
$$x_i r^i = (x_i - r)r^i + 1 \cdot r^{i+1}$$
- ◆ two machine representations for the same value -  
 $(\dots, x_{i+1}, x_i, \dots)$  and  $(\dots, x_{i+1}+1, x_i-r, \dots)$

## Representation of Mixed Numbers

- ◆ A sequence of  $n$  digits in a register - not necessarily representing an integer
- ◆ Can represent a mixed number with a fractional part and an integral part
- ◆ The  $n$  digits are partitioned into two -  $k$  in the integral part and  $m$  in the fractional part ( $k+m=n$ )
- ◆ The value of an  $n$ -tuple with a radix point between the  $k$  most significant digits and the  $m$  least significant digits

is 
$$\underbrace{(x_{k-1}x_{k-2}\cdots x_1x_0)}_{\text{integral part}} \cdot \underbrace{(x_{-1}x_{-2}\cdots x_{-m})}_{\text{fractional part}}_r$$

$$X = x_{k-1}r^{k-1} + x_{k-2}r^{k-2} + \cdots + x_1r + x_0 + x_{-1}r^{-1} + \cdots + x_{-m}r^{-m} = \sum_{i=-m}^{k-1} x_i r^i$$

## Fixed Point Representations

- ◆ Radix point not stored in register - understood to be in a fixed position between the **k** most significant digits and the **m** least significant digits
  - \* These are called **fixed-point** representations
- ◆ Programmer not restricted to the predetermined position of the radix point
  - \* Operands can be scaled - same scaling for all operands
- ◆ Add and subtract operations are correct -
  - \*  $aX \pm aY = a(X \pm Y)$  ( $a$  - scaling factor)
- ◆ Corrections required for multiplication and division
  - \*  $aX \cdot aY = a^2 X \cdot Y$  ;  $aX/aY = X/Y$
- ◆ Commonly used positions for the radix point -
  - \* rightmost side of the number (pure integers -  $m=0$ )
  - \* leftmost side of the number (pure fractions -  $k=0$ )

## ULP - Unit in Last Position

- ◆ Given the length **n** of the operands, the weight  $r^{-m}$  of the least significant digit indicates the position of the radix point
- ◆ **Unit in the last position (ulp)** - the weight of the least significant digit
- ◆  $ulp = r^{-m}$
- ◆ This notation simplifies the discussion
- ◆ No need to distinguish between the different partitions of numbers into fractional and integral parts
  
- ◆ Radix conversion - see textbook p. 4-6.

## Representation of Negative Numbers

- ◆ Fixed-point numbers in a radix  $r$  system
- ◆ Two ways of representing negative numbers:
- ◆ Sign and magnitude representation (or signed-magnitude representation)
- ◆ Complement representation with two alternatives
  - \* Radix complement (two's complement in the binary system)
  - \* Diminished-radix complement (one's complement in the binary system)

## Signed-Magnitude Representation

- ◆ Sign and magnitude are represented separately
- ◆ First digit is the sign digit, remaining  $n-1$  digits represent the magnitude
- ◆ Binary case - sign bit is 0 for positive, 1 for negative numbers
- ◆ Non-binary case - 0 and  $r-1$  indicate positive and negative numbers
- ◆ Only  $2r^{n-1}$  out of the  $r^n$  possible sequences are utilized
- ◆ Two representations for zero - positive and negative
  - \* Inconvenient when implementing an arithmetic unit - when testing for zero, the two different representations must be checked

## Disadvantage of the Signed-Magnitude Representation

- ◆ Operation may depend on the signs of the operands
- ◆ **Example** - adding a positive number  $X$  and a negative number  $-Y$  :  $X+(-Y)$
- ◆ If  $Y>X$ , final result is  $-(Y-X)$
- ◆ **Calculation** -
  - \* switch order of operands
  - \* perform subtraction rather than addition
  - \* attach the minus sign
- ◆ A sequence of decisions must be made, costing excess control logic and execution time
- ◆ This is avoided in the complement representation methods

## Complement Representations of Negative Numbers

- ◆ Two alternatives -
  - \* **Radix complement** (called **two's complement** in the binary system)
  - \* **Diminished-radix complement** (called **one's complement** in the binary system)
- ◆ In both complement methods - positive numbers represented as in the signed-magnitude method
- ◆ A negative number  $-Y$  is represented by  $R-Y$  where  $R$  is a constant
- ◆ This representation satisfies  $-(-Y)=Y$  since  $R-(R-Y)=Y$

## Advantage of Complement Representation

- ◆ No decisions made before executing addition or subtraction
- ◆ **Example:**  $X - Y = X + (-Y)$
- ◆  $-Y$  is represented by  $R - Y$
- ◆ Addition is performed by  $X + (R - Y) = R - (Y - X)$
- ◆ If  $Y > X$ ,  $-(Y - X)$  is already represented as  $R - (Y - X)$
- ◆ No need to interchange the order of the two operands

## Requirements for Selecting R

- ◆ If  $X > Y$  - the result is  $X + (R - Y) = R + (X - Y)$  instead of  $X - Y$  - additional  $R$  must be discarded
- ◆  $R$  selected to simplify or eliminate this correction
- ◆ Another requirement - calculation of the complement  $R - Y$  should be simple and done at high speed
- ◆ **Definitions:**
- ◆ Complement of a single digit  $x_i$   
$$\bar{x}_i = (r - 1) - x_i$$
- ◆ Complement of an  $n$ -tuple  $X$   
$$\bar{X} = (\bar{x}_{k-1}, \bar{x}_{k-2}, \dots, \bar{x}_m)$$
 obtained by complementing every digit in the sequence corresponding to  $X$



## The Two's Complement Representation

Sequence	Two's complement	One's complement	Signed-magnitude
0111	7	7	7
0110	6	6	6
0101	5	5	5
0100	4	4	4
0011	3	3	3
0010	2	2	2
0001	1	1	1
0000	0	0	0
1111	-1	-0	-7
1110	-2	-1	-6
1101	-3	-2	-5
1100	-4	-3	-4
1011	-5	-4	-3
1010	-6	-5	-2
1001	-7	-6	-1
1000	-8	-7	-0

ECE666/Koren Part.1 .23

Copyright 2008 Koren

### Example - Addition in Two's complement

- ◆ Calculating  $X+(-Y)$  with  $Y>X$  -  $3+(-5)$

$$\begin{array}{r} 0011 \quad 3 \\ + 1011 \quad -5 \\ \hline 1110 \quad -2 \end{array}$$

- ◆ Correct result represented in the two's complement method - no need for preliminary decisions or post corrections

- ◆ Calculating  $X+(-Y)$  with  $X>Y$  -  $5+(-3)$

$$\begin{array}{r} 0101 \quad 5 \\ + 1101 \quad -3 \\ \hline 1 \ 0010 \quad 2 \end{array}$$

- ◆ Only the last four least significant digits are retained, yielding **0010**

ECE666/Koren Part.1 .24

Copyright 2008 Koren

## A 2nd Alternative for R : Diminished-Radix Complement Representation

- ◆ Selecting R as  $R=r^k - \text{ulp}$
- ◆ This is the **diminished-radix** complement
- ◆  $R - X = (r^k - \text{ulp}) - X = \bar{X}$
- ◆ Derivation of the complement is simpler than the radix complement
- ◆ All the digit-complements  $\bar{x}_i$  can be calculated in parallel - fast computation of  $\bar{X}$
- ◆ A correction step is needed when  $R+(X-Y)$  is obtained and  $X-Y$  is positive

## Example - One's Complement in Binary System

- ◆  $r=2, k=n=4, m=0, \text{ulp}=2^0=1$
- ◆ Diminished-radix complement (called **one's complement** in the binary case) of a number  $X =$   
 $(2^4 - 1) - X = \bar{X}$
- ◆ As before, the sequences **0000** to **0111** represent the positive numbers  $0_{10}$  to  $7_{10}$
- ◆ The one's complement of **0111** is **1000**, representing  $(-7)_{10}$
- ◆ The one's complement of **zero** is **1111** - two representations of zero
- ◆ Range of representable numbers is  $-7 \leq X \leq 7$

## Comparing the Three Representations in a Binary System

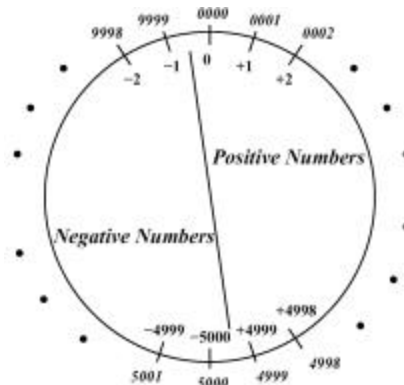
Sequence	Two's complement	One's complement	Signed-magnitude
0111	7	7	7
0110	6	6	6
0101	5	5	5
0100	4	4	4
0011	3	3	3
0010	2	2	2
0001	1	1	1
0000	0	0	0
1111	-1	-0	-7
1110	-2	-1	-6
1101	-3	-2	-5
1100	-4	-3	-4
1011	-5	-4	-3
1010	-6	-5	-2
1001	-7	-6	-1
1000	-8	-7	-0

ECE666/Koren Part.1 .27

Copyright 2008 Koren

## Example: Radix-Complement Decimal System

- ◆ Leading digit 0,1,2,3,4 - positive
- ◆ Leading digit 5,6,7,8,9 - negative
- ◆ **Example** -  $n=4$
- ◆ 0000 to 4999 - positive
- ◆ 5000 to 9999 - negative - (-5000) to -1
- ◆ Range -  $-5000 \leq X \leq 4999$
- ◆  $Y=1234$
- ◆ Representation of  $-Y=-1234$  - radix complement  $R-Y$  with  $R=10^4$
- ◆  $R-Y = \bar{Y} + \text{ulp}$
- ◆ Digit complement =  $9 - \text{digit}$  ;  $\text{ulp}=1$
- ◆  $\bar{Y}=8765$  ;  $\bar{Y}+1=8766$  - representation of  $-Y$
- ◆  $Y+(-Y)=1234+8766=10^4 \equiv 0 \pmod{10^4}$



ECE666/Koren Part.1 .28

Copyright 2008 Koren

## The Two's Complement Representation

From now on, the system is:  $r=2$ ,  $k=n$ ,  $ulp=1$

- ◆ Range of numbers in two's complement method:  
 $-2^{n-1} \leq X \leq 2^{n-1} - ulp$  ( $ulp=2^0=1$ )
- ◆ Slightly asymmetric - one more negative number
- ◆  $-2^{n-1}$  (represented by  $10 \dots 0$ ) does not have a positive equivalent
- ◆ A complement operation for this number will result in an overflow indication
- ◆ On the other hand, there is a unique representation for 0

## Numerical Value of a Two's Complement Representation

- ◆ Numerical value  $X$  of representation  $(x_{n-1}, x_{n-2}, \dots, x_0)$  in two's complement -
- ◆ If  $x_{n-1}=0$  -  $X = \sum_{i=0}^{n-1} x_i 2^i$
- ◆ If  $x_{n-1}=1$  - negative number - absolute value obtained by complementing the sequence (i.e., complementing each bit and adding 1) and adding a minus sign
- ◆ **Example** - Given the 4-tuple 1010 - negative - complementing -  $0101+1=0110$  - value is 6 - original sequence is -6

## Different Calculation of Numerical Value

$$X = -x_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

- ◆ **Example** - 1010 -  $X = -8 + 2 = -6$
- ◆ Calculating the numerical value of one's complement

$$X = -x_{n-1}(2^{n-1} - ulp) + \sum_{i=0}^{n-2} x_i 2^i$$

- ◆ **Example** - 4-tuple 1001 -  $X = -7 + 1 = -6$

## Addition and Subtraction

- ◆ **In signed-magnitude representation** -
  - \* Only magnitude bits participate in adding/subtracting - sign bits are treated separately
  - \* Carry-out (or borrow-out) indicates overflow

◆ **Example** -

0	1001	+9	
0	+ 0111	+7	_____
0	1 0000	0 = 16 mod 16	

- ◆ Final result positive (sum of two positive numbers) but wrong
- ◆ **In both complement representations** -
  - \* All digits, including the sign digit, participate in the add or subtract operation
  - \* A carry-out is not necessarily an indication of an overflow

## Addition/Subtraction in Complement Methods

◆ **Example - (two's complement)**

$$\begin{array}{r} 01001 \quad 9 \\ \underline{11001 \quad -7} \\ 1 \quad 00010 \quad 2 \end{array}$$

\* Carry-out discarded - does not indicate overflow

◆ In general, if **X** and **Y** have opposite signs - no overflow can occur regardless of whether there is a carry-out or not

◆ **Examples - (two's complement)**

$$\begin{array}{r} 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 5 \\ + \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad -10 \\ \hline 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad -5 \end{array} \quad \text{No carry-out}$$

$$\begin{array}{r} 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 10 \\ + \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad -5 \\ \hline 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 5 \end{array} \quad \text{Carry-out}$$

## Addition/Subtraction - Complement - Cont.

◆ If **X** and **Y** have the same sign and result has different sign - overflow occurs

◆ **Examples - (two's complement)**

$$\begin{array}{r} 10111 \quad -9 \\ \underline{10111 \quad -9} \\ 1 \quad 01110 \quad 14 = -18 \text{ mod } 32 \end{array}$$

\* Carry-out and overflow

$$\begin{array}{r} 01001 \quad 9 \\ \underline{00111 \quad 7} \\ 0 \quad 10000 \quad -16 = 16 \text{ mod } 32 \end{array}$$

\* No carry-out but overflow



## Subtraction

- ◆ In both complement systems - subtract operation,  $X - Y$ , is performed by adding the complement of  $Y$  to  $X$
- ◆ In the one's complement system -  
$$X - Y = X + \bar{Y}$$
- ◆ In the two's complement system -  
$$X - Y = X + (\bar{Y} + \text{ulp})$$
- ◆ This still requires only a single adder operation, since **ulp** is added through the forced carry input to the binary adder

## Arithmetic Shift Operations

- ◆ Another way of distinguishing among the three representations of negative numbers - the infinite extensions to the right and left of a given number
- ◆ **Signed-magnitude method** - the magnitude  $X_{n-2}, \dots, X_0$  can be viewed as the infinite sequence  
$$\dots, 0, 0, \{X_{n-2}, \dots, X_0\}, 0, 0, \dots$$
- ◆ Arithmetic operation resulting in a nonzero prefix - an overflow
- ◆ **Radix-complement scheme** - the infinite extension is  
$$\dots, X_{n-1}, X_{n-1}, \{X_{n-1}, \dots, X_0\}, 0, 0, \dots$$
 ( $X_{n-1}$  - the sign digit)
- ◆ **Diminished-radix complement scheme** - the sequence is  
$$\dots, X_{n-1}, X_{n-1}, \{X_{n-1}, \dots, X_0\}, X_{n-1}, X_{n-1}, \dots$$

## Arithmetic Shift Operations - Examples

- ◆ **1010.**, **11010.0**, **111010.00** - all represent **-6** in two's complement
- ◆ **1001.**, **11001.1**, **111001.11** - all represent **-6** in one's complement
- ◆ Useful when adding operands with different numbers of bits - shorter extended to longer
- ◆ Rules for arithmetic shift operations: left and right shift are equivalent to multiply and divide by **2**, respectively

### Two's complement

Sh.L{00110=6}=01100=12

Sh.R{00110=6}=00011=3

Sh.L{11010=-6}=10100=-12

Sh.R{11010=-6}=11101=-3

### One's complement

Sh.L{11001=-6}=10011=-12

Sh.R{11001=-6}=11100=-3