UNIVERSITY OF MASSACHUSETTS
Dept. of Electrical & Computer Engineering

Digital Computer Arithmetic
ECE 666

Part 1
Introduction

Israel 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

Administrative Details

- **Instructor**: Prof. Israel Koren
- **Office**: KEB 309E, Tel. 545-2643
- **Email**: koren@ecs.umass.edu
- **Office Hours**: TuTh 4:00-5:00
- **Course web page**: *http://www.ecs.umass.edu/ece/koren/ece666/*
- **Grading**:
  - OWL quizzes – 15%
  - Two Mid-term exams - 25% each
  - Final Take-home Exam or Project - 35%

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 $(x_{n-1}, x_{n-2}, \cdots, x_1, x_0)$ of binary digits
- Each digit x_i (bit) is 0 or 1
- The above sequence represents the integer value X

$$X = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_1 2 + x_0 = \sum_{i=0}^{n-1} x_i 2^i$$

- 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 then 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 mod 32
  - If X+Y exceeds X_{max} - the result is S = (X+Y) mod 32
- Example:

X	10001	17	
+Y	10010	18	
1	00011	3	= 35 mod 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 $W_{n-1},W_{n-2},\ldots,W_1,W_0$ determines the value of the n-tuple $X_{n-1},X_{n-2},\ldots,X_1,X_0$ by

$$X = \sum_{i=0}^{n-1} x_i w_i.$$

  - W_i - weight assigned to X_i - digit in ith 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 - $0 \le x_i \le r-1$
- $x_i \ge r$ introduces redundancy into the fixed-radix number system
- If $x_i \ge r$ is allowed -

$$x_i r^i = (x_i - r)r^i + 1 \cdot r^{i+1}$$

- two machine representations for the same value - $(\ldots, x_{i+1}, x_i, \ldots)$ and $(\ldots, x_{i+1}+1, x_i-r, \ldots)$

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

$$X = x_{k-1}r^{k-1} + x_{k-2}r^{k-2} + \cdots + x_1 r + x_0 + x_{-1}r^{-1} + \cdots + x_{-m}r^{-m} = \sum_{i=-m}^{k-1} x_i r^i$$

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
  - $aX \bullet aY = a^2 X \bullet Y$; $aX/aY = X/Y$
- 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

  $\bar{X} = (\bar{x}_{k-1},\bar{x}_{k-2},\ldots,\bar{x}_{-m})$ obtained by complementing every digit in the sequence corresponding to X

Selecting R in Radix-Complement Rep.

- $X+\bar{X}+ulp = r^k$
- Result stored into a register of length n(=k+m)

$$\begin{array}{lccccc} X & x_{k-1} & x_{k-2} & \cdots & x_{-m} & \\ +\bar{X} & \bar{x}_{k-1} & \bar{x}_{k-2} & \cdots & \bar{x}_{-m} & \\ \hline & (r-1) & (r-1) & \cdots & (r-1) & \\ +ulp & & & & 1 & \\ \hline 1 & 0 & 0 & \cdots & 0 & = r^k \end{array}$$

- Most significant digit discarded - final result is zero
- In general, storing the result of any arithmetic operation into a fixed-length register is equivalent to taking the remainder after dividing by r^k
- $r^k - X = \bar{X} + ulp$
- Selecting $R = r^k$: $R - X = r^k - X = \bar{X} + ulp$
- Calculation of R-X - simple and independent of k
- This is radix-complement representation
- $R=r^k$ discarded when calculating R+(X-Y) - no correction needed when X+(R-Y) is positive (X>Y)

Example - Two's Complement

- $r=2$, $k=n=4$, $m=0$, $ulp=2^0=1$
- Radix complement (called two's complement in the binary case) of a number $X = 2^4 - X$
- It can instead be calculated by $\bar{X}+1$
- 0000 to 0111 represent positive numbers 0_{10} to 7_{10}
  - The two's complement of 0111 is 1000+1=1001 - it represents the value $(-7)_{10}$
  - The two's complement of 0000 is $1111+1=10000=0 \bmod 2^4$ - single representation of zero
- Each positive number has a corresponding negative number that starts with a 1
- 1000 representing $(-8)_{10}$ has no corresponding positive number
- Range of representable numbers is $-8 \le X \le 7$

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

Example - Addition in Two's complement

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

```
   0011    3
+  1011   -5
   1110   -2
```

- 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)

```
   0101    5
+  1101   -3
 1 0010    2
```

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

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

- Selecting R as $R=r^k - ulp$
- This is the diminished-radix complement
- $R - X = (r^k - ulp) - X = \bar{X}$
- 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

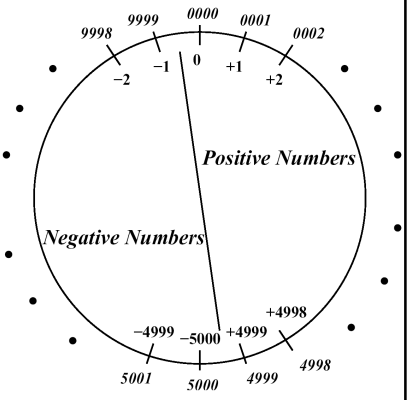
- r=2, k=n=4, m=0, $ulp=2^0=1$
- Diminished-radix complement (called one's complement in the binary case) of a number X = $(2^4 - 1) - X = \bar{X}$
- As before, the sequences 0000 to 0111 represent the positive numbers 0_{10} to 7_{10}
- The one's complement of 0111 is 1000, representing $(-7)_{10}$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

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

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 - $-5000 \le X \le 4999$
- Y=1234
- Representation of -Y=-1234 - radix complement R-Y with $R=10^4$
- $R-Y = \bar{Y} + ulp$
- Digit complement = 9 - digit ; ulp=1
- $\bar{Y}=8765$; $\bar{Y}+1 = 8766$ - representation of -Y
- $Y+(-Y)=1234+8766=10^4 = 0 \bmod 10^4$



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: $-2^{n-1} \le X \le 2^{n-1} - ulp$ ($ulp=2^0=1$)
- Slightly asymmetric - one more negative number
- -2^{n-1} (represented by 10…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 $(x_{n-1},x_{n-2},\ldots,x_0)$ 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)

```
  01001   9
  11001  -7
------------
1 00010   2
```

  - 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)

	0	0	1	0	1	5	
+	1	0	1	1	0	−10	
	1	1	0	1	1	−5	No carry-out
	0	1	0	1	0	10	
+	1	1	0	1	1	−5	
1	0	0	1	0	1	5	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)

```
  10111  -9
  10111  -9
------------------------
1 01110   14 = -18 mod 32
```

  - Carry-out and overflow

```
  01001   9
  00111   7
------------------------
0 10000  -16 = 16 mod 32
```

  - No carry-out but overflow

Addition/Subtraction - One's Complement

- Carry-out - indicates the need for a correction step
- Example - adding positive X and negative -Y
  $X+(2^n - ulp)-Y = (2^n - ulp)+(X-Y)$
- If X>Y - correct result is X-Y
- 2^n represents the carry-out bit - discarded in a register of length n
- Result is X-Y-ulp - corrected by adding ulp
- Example -

	01001	9
	+ 11000	-7
	1 00001	
Correction	1	ulp
	00010	2

- The generated carry-out is called end-around carry - it is an indication that a 1 should be added to the least significant position

Addition/Subtraction - One's Complement -Cont.

- If X<Y - the result X-Y=-(Y-X) is negative
- Should be represented by $(2^n - ulp) - (Y-X)$
- There is no carry-out - no correction is needed
- Example -

10110	-9
00111	7
11101	-2

- No end-around carry correction is necessary in two's complement addition

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 -

  $X-Y=X+(\bar{Y}+ulp)$
- 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 $x_{n-2},\ldots, x_0$ can be viewed as the infinite sequence
  $\ldots,0,0,\{x_{n-2},\ldots,x_0\},0,0,\ldots$
- Arithmetic operation resulting in a nonzero prefix - an overflow
- Radix-complement scheme - the infinite extension is
  $\ldots,x_{n-1},x_{n-1},\{x_{n-1},\ldots,x_0\},0,0,\ldots$ (x_{n-1} - the sign digit)
- Diminished-radix complement scheme - the sequence is
  $\ldots,x_{n-1},x_{n-1},\{x_{n-1},\ldots,x_0\},x_{n-1},x_{n-1},\ldots$

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	One's complement
Sh.L{00110=6}=01100=12	Sh.L{11001=-6}=10011=-12
Sh.R{00110=6}=00011=3	Sh.R{11001=-6}=11100=-3
Sh.L{11010=-6}=10100=-12	
Sh.R{11010=-6}=11101=-3	