Lecture 12: Number Representation

Integers and Computer Arithmetic
Numbers: positional notation

• Number Base $B \Rightarrow B$ symbols per digit:
  • Base 10 (Decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  • Base 2 (Binary): 0, 1

• Number representation:
  • $d_{31}d_{30} \ldots d_{1}d_{0}$ is a 32 digit number
  • value = $d_{31} \times B^{31} + d_{30} \times B^{30} + \ldots + d_{1} \times B^{1} + d_{0} \times B^{0}$

• Binary: 0,1 (In binary digits called “bits”)
  • $0b11010 = 1 \times 2^{4} + 1 \times 2^{3} + 0 \times 2^{2} + 1 \times 2^{1} + 0 \times 2^{0}$
  • = 16 + 8 + 2
  • = 26

#s often written

0b...
Hexadecimal Numbers: Base 16

• Hexadecimal:
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  • Normal digits + 6 more from the alphabet
  • In C and SPIM, written as 0x... (e.g., 0xFAB5)

• Conversion: Binary ⇔ Hex
  • 1 hex digit represents 16 decimal values
  • 4 binary digits represent 16 decimal values
  ⇒ 1 hex digit replaces 4 binary digits

• One hex digit is a “nibble”. Two is a “byte”

• Example:
  • 1010 1100 0011 (binary) = 0x______ ?
Decimal vs. Hexadecimal vs. Binary

Examples:

1010 1100 0011 (binary) = 0xAC3

10111 (binary) = 0001 0111 (binary) = 0x17

0x3F9 = 11 1111 1001 (binary)

How do we convert between hex and Decimal?

MEMORIZE!
BIG IDEA: Bits can represent anything!!

• Characters?
  • 26 letters $\Rightarrow$ 5 bits ($2^5 = 32$)
  • upper/lower case + punctuation $\Rightarrow$ 7 bits (in 8) (“ASCII”)
  • standard code to cover all the world’s languages $\Rightarrow$ 8,16,32 bits (“Unicode”)
    www.unicode.com

• Logical values?
  • 0 $\Rightarrow$ False, 1 $\Rightarrow$ True

• Colors? Ex: Red (00) Green (01) Blue (11)

• Locations / addresses? MIPS instructions?

• MEMORIZE: N bits $\Rightarrow$ at most $2^N$ things
Till now, we have only considered how unsigned numbers can be represented. There are three common ways of representing signed numbers:

- **Sign-And-Magnitude**
- **1s complement**
- **2s complement**
Negative Numbers: Sign-and-Magnitude (I)

- Negative numbers are usually written by pre-pending a minus sign in front.
  - Example:
    - \(- (12)_{10} = - (1100)_{2}\)

- In computer memory of fixed width, this sign is usually represented by a bit:
  - 0 for +
  - 1 for −
Negative Numbers: Sign-and-Magnitude (II)

- Example: an 8-bit number can have 1-bit sign and 7-bits magnitude.
Negative Numbers: Sign-and-Magnitude (III)

- Largest Positive Number: $0\ 1111111\quad +\ (127)_{10}$
- Largest Negative Number: $1\ 1111111\quad -\ (127)_{10}$
- Zeroes:
  - $0\ 0000000\quad +\ (0)_{10}$
  - $1\ 0000000\quad -\ (0)_{10}$
- Range: $-(127)_{10}$ to $+(127)_{10}$
Negative Numbers: Sign-and-Magnitude (IV)

- To negate a number, just invert the sign bit.
- Examples:
  - $(0 \ 0100001)_{sm} = (1 \ 0100001)_{sm}$
  - $(1 \ 0000101)_{sm} = (0 \ 0000101)_{sm}$
Shortcomings of sign and magnitude?

- Arithmetic circuit complicated
  - Special steps depending whether signs are the same or not

- Also, **two** zeros
  - $0x00000000 = +0_{\text{ten}}$
  - $0x80000000 = -0_{\text{ten}}$
  - What would two 0s mean for programming?

- Therefore sign and magnitude abandoned
1s and 2s complement notations

• In these notations, a positive number is represented as it is (like an unsigned positive number)

• A negative number, however, is represented by taking the complement of unsigned number
1s Complement (I)

- 1s complement of an unsigned number is obtained by inverting all the bits of the number.

Examples:

1s complement of $(00000001)_2$ is $(11111110)_{1s}$
1s complement of $(01111111)_2 = (10000000)_{1s}$

- 1s complement representation of the number in 8-bits

Example:

1. $(+14)_{10} = (00001110)_2 = (00001110)_{1s}$
2. $(-14)_{10} = -(00001110)_2 = (11110001)_{1s}$
3. $(-80)_{10} = (?)_2 = (?)_{1s}$
1s Complement (II)

For 8-bits number system:

- Largest Positive Number: \( 0 \, 1111111 \) \((127)_{10}\)
- Largest Negative Number: \( 1 \, 0000000 \) \(-(127)_{10}\)
- Zeroes: \( 0 \, 0000000 \) \( 1 \, 1111111 \)
- Range: \(-(127)_{10}\) to \((127)_{10}\)
- The most significant bit still represents the sign:
  
  \[ 0 = +, \quad 1 = - \]
1s Complement (III)

• Given a number \( x \) which can be expressed as an \( n \)-bit binary number, its negative value can be obtained in 1s-complement representation using:

\[
-x = 2^n - x - 1
\]

Example: With an 8-bit number 00001100, its negative value, expressed in 1s complement, is obtained as follows:

\[
-(00001100)_{2} = -(12)_{10}
\]

\[
= (2^8 - 12 - 1)_{10}
\]

\[
= (243)_{10}
\]

\[
= (1110011)_{1s}
\]
Shortcomings of One’s complement

• Arithmetic still a somewhat complicated.

• Still two zeros
  • \(0x00000000 = +0_{\text{ten}}\)
  • \(0xFFFFFFFF = -0_{\text{ten}}\)

• Although used for a while on some computer products, one’s complement was eventually abandoned because another solution was better.
2s Complement (I)

- 2s complement of an unsigned number is obtained by inverting all the bits and adding 1.

Examples:

1. 2s complement of \(00000001\) sub 2 = \(11111110\) sub 1s (invert)
   
   \[= (11111111)\] sub 2s (add 1)

2. 2s complement of \(01111110\) sub 2 = \(10000001\) sub 1s (invert)
   
   \[= (10000010)\] sub 2s (add 1)
2s Complement (II)

- 2s complement representation for 8 bit numbers:

Example:

1. \((+14)_{10} = (00001110)_2 = (00001110)_{2s}\)
2. \((-14)_{10} = -(00001110)_2 = (11110010)_{2s}\)
3. \((-80)_{10} = (?)_2 = (?)_{2s}\)
2s Complement (III)

- Given a number $x$ which can be expressed as an $n$-bit binary number, its negative number can be obtained in 2s-complement representation using:

$$-x = 2^n - x$$

Example: With an 8-bit number 00001100, its negative value in 2s complement is thus:

$$-(00001100)_2 = -(12)_{10}$$

$$= (2^8 - 12)_{10}$$

$$= (244)_{10}$$

$$= (11110100)_2$$
2s Complement (IV)

• Largest Positive Number: \( 0 \ 1111111 \) \(+\( 127 \)\(_{10}\)\)
• Largest Negative Number: \( 1 \ 0000000 \) \(-\( 128 \)\(_{10}\)\)
• Zero: \( 0 \ 0000000 \)
• Range: \(-\( 128 \)\(_{10}\) to \(+\( 127 \)\(_{10}\)\)
• The most significant bit still represents the sign:

\[
0 = +, \quad 1 = -
\]
2’s Complement Number “line”: N = 5

• $2^{N-1}$ non-negatives
• $2^{N-1}$ negatives
• one zero
• how many positives?

00000 00001 ... 01111
10000 ... 11110 11111

Number Representation (21)
### Two’s Complement for N=32

<table>
<thead>
<tr>
<th></th>
<th>Two's Complement</th>
<th>Ten's Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 ... 0000</td>
<td>0\textsubscript{two}</td>
<td>0\textsubscript{ten}</td>
</tr>
<tr>
<td>0000 ... 0000</td>
<td>1\textsubscript{two}</td>
<td>1\textsubscript{ten}</td>
</tr>
<tr>
<td>0000 ... 0000</td>
<td>2\textsubscript{two}</td>
<td>2\textsubscript{ten}</td>
</tr>
<tr>
<td>0111 ... 1111</td>
<td>2,147,483,645\textsubscript{two}</td>
<td>2,147,483,645\textsubscript{ten}</td>
</tr>
<tr>
<td>0111 ... 1111</td>
<td>2,147,483,646\textsubscript{two}</td>
<td>2,147,483,646\textsubscript{ten}</td>
</tr>
<tr>
<td>0111 ... 1111</td>
<td>2,147,483,647\textsubscript{two}</td>
<td>2,147,483,647\textsubscript{ten}</td>
</tr>
<tr>
<td>1000 ... 0000</td>
<td>-2,147,483,648\textsubscript{two}</td>
<td>-2,147,483,648\textsubscript{ten}</td>
</tr>
<tr>
<td>1000 ... 0000</td>
<td>-2,147,483,647\textsubscript{two}</td>
<td>-2,147,483,647\textsubscript{ten}</td>
</tr>
<tr>
<td>1000 ... 0000</td>
<td>-2,147,483,646\textsubscript{two}</td>
<td>-2,147,483,646\textsubscript{ten}</td>
</tr>
<tr>
<td>1111 ... 1111</td>
<td>-3\textsubscript{two}</td>
<td>-3\textsubscript{ten}</td>
</tr>
<tr>
<td>1111 ... 1111</td>
<td>-2\textsubscript{two}</td>
<td>-2\textsubscript{ten}</td>
</tr>
<tr>
<td>1111 ... 1111</td>
<td>-1\textsubscript{two}</td>
<td>-1\textsubscript{ten}</td>
</tr>
</tbody>
</table>

- One zero; 1st bit called **sign bit**
- 1 “extra” negative: no positive 2,147,483,648\textsubscript{ten}
Two’s Complement Formula

• Can represent positive and negative numbers in terms of the bit value times a power of 2:
  \[ d_{31} \times -(2^{31}) + d_{30} \times 2^{30} + ... + d_{2} \times 2^{2} + d_{1} \times 2^{1} + d_{0} \times 2^{0} \]

• Example: \(1101_{\text{two}}\)
  \[ = 1 \times -(2^{3}) + 1 \times 2^{2} + 0 \times 2^{1} + 1 \times 2^{0} \]
  \[ = -2^{3} + 2^{2} + 0 + 2^{0} \]
  \[ = -8 + 4 + 0 + 1 \]
  \[ = -8 + 5 \]
  \[ = -3_{\text{ten}} \]
Arithmetic

- Complement numbers can help perform subtraction.

With complements, subtraction can be performed by addition.
Use of complements

• Complement number system is used to minimize the amount of circuitry needed to perform integer arithmetic.

• For example, A-B can be performed by computing A + (-B), where (-B) is represented in 2s complement of B.

• Hence, the computer needs only binary adder and complementing circuit to handle both addition and subtraction
Overflow

• Signed binary numbers are of a fixed range.

• If the result of addition/subtraction goes beyond this range, overflow occurs.

• Two conditions under which overflow can occur are:

  (i) positive add positive gives negative

  (ii) negative add negative gives positive
2s complement addition

Algorithm:

1. Perform binary addition on the two numbers.
2. Ignore the carry out of the MSB.
3. Check for overflow: Overflow occurs if the carrier into and out of the MSB are different.
Algorithm for performing $A - B$:

$A - B = A + (-B)$

1. Take 2s complement of $B$ by inverting all the bits and adding 1
2. Add the 2s complement of $B$ to $A$
Examples: 2s addition/Subtraction

4-bits system

+3  0011
+ +4  + 0100
----  ------
+7    0111
----  ------

-2  1110
+ -6  + 1010
----  ------
-8    11000
----  ------

+6  0110
+ -3  + 1101
----  ------
+3    10011
----  ------

+4  0100
+ -7  + 1001
----  ------
-3    1101
----  ------
Examples: Overflow in 2s addition/Subtraction
4-bits system

\[ \begin{array}{c c}
-3 & 1101 \\
+ -6 & + 1010 \\
\hline
-9 & 1011 \\
\end{array} \]

\[ \begin{array}{c c}
+5 & 0101 \\
+ +6 & + 0110 \\
\hline
+11 & 1011 \\
\end{array} \]
Algorithm C=A+B:

1. Perform binary addition on the two numbers
2. If there is a carry out of the MSB, add 1 to the result (to get C)
3. Check for overflow: if carried into and out of MSB are different and C is opposite sign of A and B

Algorithm A-B

1. Complement all bits of B
2. Proceed as addition
Examples: 1s addition/subtraction

+3 0011
+  +4 + 0100
---- -------
+7 0111
---- -------

+5 0101
+ -5 + 1010
---- -------
-0 1111
---- -------

-2 1101
+ -5 + 1010
---- -------
-7 10111
---- + 1
---- 1000

-3 1100
+ -7 + 1000
---- -------
-10 10100
---- + 1
---- 0101
Quickie Quiz

1. In a 6-bit 2’s complement binary number system, what is the decimal value represented by (100100)$_{2s}$?
   a. -4    b. 36    c. -36    d. -27    e. -28

2. In a 6-bit 1’s complement binary number system, what is the decimal value represented by (010100)$_{1s}$?
   a. -11    b. 43    c. -43    d. 20    e. -20

3. For 2’s complement binary numbers, the range of values for 5-bit numbers is
   a. 0 to 31    b. -8 to +7    c. -8 to +8    d. -15 to +15
   e. -16 to +15
4. In a 4-bit twos-complement scheme, what is the result of this operation: (1011)\_2s + (1001)\_2s?
   a. 0100   b. 0010   c. 1100   d. 1001   +e. overflow

Q5. Perform subtraction with the following unsigned binary numbers by taking first the 1's complement, and then, the 2's complement, of the subtrahend: (this is a 6 bit system)
   (a) 11010 - 10000 (26-16)
   (b) 11010 - 1101  (26-13)
   (c) 100 - 110000  (4-48)
   (d) 1010100 - 1010100 (84-84)