Chapter 1:

Integer representations in computer systems (Review)

Topics:

Unsigned representations
Signed representations
Signed binary and hexadecimal arithmetic
Internal Representation of Integers

Unsigned integers (non-negative)
signed integers
Fixed number of digits

Decimal number system

* base (or radix) = 10
* digits: 0, 1, 2, ... , 9

Example:

\[(739)_{10}\]

\[= 7 \times 10^2 + 3 \times 10^1 + 9 \times 10^0\]

Rules:
1) number digit positions 0, 1, 2..., right to left
2) multiply by radix to (digit position) power
**Binary number system**

* base 2
* digits: 0, 1

* binary digit: **bit**

**Example:**

Convert this binary number to decimal:

\[(1011)_2\]

\[
= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
= 8 + 0 + 2 + 1 \\
= 11
\]

**Useful information:** powers of 2 are...

\[2^4 = 16, \ 2^5 = 32, \ 2^6 = 64, \ 2^7 = 128\]

\[2^8 = 256, \ 2^9 = 512, \ 2^{10} = 1024\]
Table 1: All possible unsigned 4-bit binary numbers

<table>
<thead>
<tr>
<th>binary</th>
<th>decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
</tr>
</tbody>
</table>
Example: 8-bit unsigned binary number

Convert to decimal:

\[(10010101)_2\]

\[= 27 + 24 + 22 + 20\]
\[= 128 + 16 + 4 + 1\]
\[= 149\]

Computer hardware is based on digital logic circuits; data is represented using binary system.
Convert decimal numbers to binary numbers

Method 1 (slow but intuitive):

break up into sum of powers of 2

Example:

\[(249)_{10}\]

\[= 128 + 64 + 32 + 16 + 8 + 1\]
\[= 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^0\]
\[= (11111001)_{2}\]
Convert decimal numbers to binary numbers

Method 2 (shortcut):

1) divide by 2
2) write down remainder
3) repeat until quotient = 0
4) combine remainders in reverse order (bottom to top)

Example:

\( (249)_{10} \)

\[
\begin{align*}
249/2 &= 124 \text{ rem } 1 \\
124/2 &= 62 \text{ rem } 0 \\
62/2 &= 31 \text{ rem } 0 \\
31/2 &= 15 \text{ rem } 1 \\
15/2 &= 7 \text{ rem } 1 \\
7/2 &= 3 \text{ rem } 1 \\
3/2 &= 1 \text{ rem } 1 \\
1/2 &= 0 \text{ rem } 1 \\
249 &= (11111001)_{2}
\end{align*}
\]
Hexadecimal (hex) number system

* base 16
* digits: 0, 1, 2, ..., 9, A, B, C, D, E, F

(shorthand for binary)

Example: (0x means base 16)

Convert 3-digit hex int to decimal:

$$ \text{0x38F} $$

$$ = 3 \times 16^2 + 8 \times 16^1 + 15 \times 16^0 $$
$$ = 3 \times 256 + 8 \times 16 + 15 $$
$$ = 911 $$
Convert unsigned decimal int to hex

Method 1: break into sums of powers of 16

\[ 305 \]

\[ = 1 \times 256 + 3 \times 16 + 1 \]
\[ = 1 \times 16^2 + 3 \times 16^1 + 1 \times 16^0 \]
\[ = 0x131 \]

Method 2: divide repeatedly by 16

\[ 305 \]

\[ 305 / 16 = 19, \text{ rem } = 1 \]
\[ 19 / 16 = 1, \text{ rem } = 3 \]
\[ 1 / 16 = 0, \text{ rem } = 1 \]

\[ 305 = 0x131 \]
Convert hexadecimal numbers to binary numbers

one hexadecimal digit = 4 bits
start at least significant (rightmost) digit

Example:

\[ 0x5BE3 \]

\[ 5 = 0101, B = 1011, 3 = 0011 \]

\[ 0x5B3 = (0101 1011 0011)_2 \]

Convert binary ints to hex (reverse):

group bits into groups of 4
start at right most bit
Octal number system

* base 8
* digits: 0, 1, 2, ..., 7

Convert octal to decimal:

1) number digit positions 0, 1, 2... right to left
2) multiply each digit by radix to (digit position) power

Convert octal to binary:

each octal digit = 3 bits
Unsigned integer addition

Binary:

\[
\begin{array}{c}
1 \\
0110 \\
+ 0011 \\
1001
\end{array}
\]

Hexadecimal:

\[
\begin{array}{c}
1 \text{3de} \\
+ 782 \\
b60
\end{array}
\]
Unsigned integer subtraction

Binary:

```
  1 1010
-  0100
  0110
```

```
  1 1 1011
  0100 1011
-  0011 0100
  0001 0111
```
3 systems for representing signed integers:
1. sign-magnitude (SM)
2. one’s complement (OC)
3. two’s complement (TC)

Binary Sign-magnitude

Rules:
1) most significant (leftmost) bit is sign bit
   non-negative if sign bit = 0,
   negative if sign bit = 1

2) non-negative numbers same as unsigned

3) rest of bits represents magnitude of integer

Convert 8-bit binary SM to decimal:

0010 0101 = + 010 0101 = 32 + 4 + 1 = 37
Convert binary SM to decimal:

1110 0101 = - 0110 0101
= - (64 + 32 + 4 + 1)
= - 101

Convert decimal to 8-bit binary SM:

X = -13 = ??

-X = 13 = 0000 1101
X = 1000 1101

Dirty zero problem in SM:
(0000)₂ = 0
(1000)₂ = -0 = 0

Two different representations for zero
Not suitable for fast hardware implementations
One’s Complement (OC):

Rules:
1) most significant (leftmost) bit is sign bit
   non-negative if sign bit = 0,
   negative if sign bit = 1

2) non-negative numbers same as unsigned

3) to negate a OC binary int:
   flip/complement each bit
Convert 8-bit OC binary to decimal:

\[(1100 0101)_2 = - (0011 1010)_2\]
\[= -(32+16+8+2) = - 58\]

Convert decimal to 8-bit OC binary:

\[-58 = ??\]
Let \(X = -58\)
\[-X = 58 = (0011 1010)_2\]
\(X = (1100 0101)_2\)

Dirty zero problem in OC:

\[(0000)_2 = 0\]
\[(1111)_2 = -0 = 0\]
Two’s complement

Rules:
1) most significant (leftmost) bit is sign bit
   non-negative if sign bit = 0,
   negative if sign bit = 1

2) non-negative numbers same as unsigned

3) to negate a two’s complement binary number,
   i. complement (or flip) each bit
   ii. add 1 (discard carry out)
Example:

Convert 8-bit TC binary int to decimal:

\[(00100110)_2\]

\[= 32 + 4 + 2 = 38\]

Example:

Negate TC binary int:

\[(00100110)_2\]

\[X = (00100110)_2\]

\[-X = (11010011) + 1 = (11011010)_2\]
Example:

Convert -29 to 8-bit TC binary:

Let X = -29
-X = 29 = 0001 1101
X = 1110 0010 + 1 = 1110 0011

Shortcut for negating binary TC int:

1) look for rightmost bit == 1
2) complement each bit to the left
3) (all other bits stay the same)

Example:

X = (1011 1000)₂ [rightmost 1 is underlined]
-X = (01001000)₂
Another way to convert TC binary int to decimal:
* for n-bit ints, digit position n-1 is \(-2^{(n-1)}\)

Examples:
X = (1110 0011)\(_2\)
= -128 + 64 + 32 + 2 + 1 = -29
No dirty zeros in TC:

\[ (0000)_2 = 0 \]

Try to construct negative zero:
\(- (0000)_2 = (1111)_2 + 1\]
\[ = 10000 \]

Discard carryout: \(-0 = (0000)_2 \]

Sign-extension in TC:
(writing the same integer, but with more bits)

4-bit to 8-bit: \((0101)_2 = (0000\ 0101)_2\]
4-bit to 8-bit: \((1101)_2 = (1111\ 1101)_2\]

Extend (or duplicate the sign bit).
<table>
<thead>
<tr>
<th>Unsigned</th>
<th>SM</th>
<th>OC</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>0</td>
<td>-7</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>-1</td>
<td>-6</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>-2</td>
<td>-5</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>-3</td>
<td>-4</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
<td>-4</td>
<td>-3</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
<td>-5</td>
<td>-2</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>-6</td>
<td>-1</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>-7</td>
<td>0</td>
</tr>
</tbody>
</table>
Range of binary integer representations:

Unsigned:
4-bit: 0 to 15 = 0 to \((2^4 - 1)\)

N-bit: 0 to \((2^N - 1)\)

SM:
4-bit: -7 to 7 = \((-2^3 - 1)\) to \((2^3 - 1)\)

N-bit: \((-2^{N-1} - 1)\) to \((2^{N-1} - 1)\)

OC:
4-bit: -7 to 7 = \((-2^3 - 1)\) to \((2^3 - 1)\)

N-bit: \((-2^{N-1} - 1)\) to \((2^{N-1} - 1)\)

TC:
4-bit: -8 to 7 = \((-2^3)\) to \((2^3 - 1)\)

N-bit: \((-2^{N-1})\) to \((2^{N-1} - 1)\)
Range of integer data types:

short (usually 16 bits)

\((-2^{16}-1)\) to \((2^{16}-1 - 1)\), or

-65536 to 65535

int (usually 32 bits)

\((-2^{32}-1)\) to \((2^{32}-1 - 1)\), or

-4,294,967,296 to 4,294,967,295

long (usually 64 bits)

\((-2^{64}-1)\) to \((2^{64}-1 - 1)\)
Two’s complement arithmetic

similar to unsigned

1) treat sign bit as numeric bit
2) if carry out is produced, discard it
3) check for overflow

Examples:

\[
\begin{array}{c}
1001 0110 \\
+ 0011 0110 \\
\hline
1100 1100 \\
\end{array}
\]
check: \(-196 + 54 = -52\)

\[
\begin{array}{c}
1101 0110 \\
+ 1110 0001 \\
\hline
1 1011 0111 \\
\end{array}
\]
discard carryout;
ans = \(1011 0111\)
\[
\begin{align*}
0111 \ 1111 \\
+ \ 0000 \ 0001 \\
\hline
1000 \ 0000
\end{align*}
\]

sum of 2 positive integers cannot be negative: overflow occurred

\[
\begin{align*}
1000 \ 0000 \\
+ \ 1111 \ 1111 \\
\hline
10111 \ 1111
\end{align*}
\]

sum of 2 negative integers cannot be positive: overflow occurred
Definition:
Overflow is an error condition in which the result of a computation does not fit into the available number of bits.

In TC arithmetic:
1) if we add 2 positive integers and get a negative result, or
2) if we add 2 negative integers and get a positive result,

we have overflow.
Two’s complement subtraction

X - Y = X + (-Y)

Example:

X = 1001 0010
Y = 0101 1111

-Y = 1010 0001

X - Y = 1001 0010 + 1010 0001
= 1 00110011

Discard carryout; result = 0011 0011

overflow occurred

[Note: for subtraction, check X + (-Y), using the overflow check for addition!]
Two’s complement hexadecimal

Rules:
1) most significant (leftmost) digit is sign digit
   non-negative if sign digit =
   negative if sign digit =

2) non-negative integers same as unsigned

3) to negate a two’s complement hex int,
   i. subtract each digit from f
   ii. add 1 (discard carry out)

Example: convert 2-digit TC hex int to decimal

\[ X = 0xab \]
\[- X = 0xff - 0xab + 1 = 0x55 \]
\[ = 5 \times 16 + 5 = 85 \]
\[ X = -85 \]
Convert decimal int to 2-digit TC hex:

\[ X = -73 \]

\[ -X = 73 = 4 \times 16 + 9 = 0x49 \]
\[ X = 0xff - 0x49 + 1 = 0xb6 + 1 = 0xb7 \]

**TC hex addition/subtraction:**

\[ 0x5678 + 0x432b \]

\[ 0x5678 \]
\[ 0x432b \]
\[ 0x99a3, \text{ overflow} \]

\[ 0xdcba + 0xe2f3 \]

\[ 0xdcba \]
\[ 0xe2f3 \]
\[ 0x1bfad \]
\[ \text{discard carryout; result} = 0xbfad \]
$0x1cba - 0xbd0d$

$= 0x1cba + - (0xbd0d)$

$= 0x1cba + 0xffff - 0xbd0d + 1$

$= 0x1cba + 0x42f3$

$0x1cba$

$0x42f3$

$0x5fad$
CSc 310 Practice Exercise #1 on arithmetic

For problems 1-9, solutions can be found in the file ~whsu/310/EXERCISES/ex1.soln.
To make a copy of this file in your directory, type

    cp ~whsu/310/EXERCISES/ex1.soln ex1.soln

A copy of the solutions file called ex1.soln will be created in your directory.

1) Consider the 16-bit binary integer $X = 1001\ 0000\ 0000\ 0011$.
   
   Convert $X$ to decimal if $X$ is
   
   a. unsigned  b. in sign-magnitude notation
   
   c. in one's complement notation  d. in two's complement notation

For problems 2-6, assume all integers are in binary and in two's complement notation. Remember to indicate overflow if necessary.

2) $0110\ 1010 + 1001\ 1110 = ?$

3) $1001\ 1111 + 1001\ 0001 = ?$

4) $1000\ 1111 - 0001\ 0000 = ?$

5) $0001\ 0010 - 0010\ 1111 = ?$

6) $1111\ 1010 - 1110\ 1110 = ?$

For problems 7-9, assume all integers are in hexadecimal and two's complement notation. Remember to indicate overflow if necessary.

7) $0x2AF6 + 0x7017 = ?$

8) $0x345E + 0xFFAB = ?$

9) $0x966A - 0x6996 = ?$