SESSION 5 – INTEGER REPRESENTATION

Reading: Section 2.4

Objectives

• Understand the fundamentals of integer data representation and manipulation in digital computers
• Gain experience in working with binary numbers
• Understand the difference between a number and the representation of that number in a computer
• Understand how errors can occur in computations because of overflow and truncation
Number Representation

• Understanding the computer representation (in 0s and 1s) helps you:
  • Understand the limitations of a type (e.g. maximum value, precision)
  • Debug problems by reading the actual representation of the data
  • Understand implications of computer architecture choices
  • Better read hardware specs
  • Understand issues in porting code to a different computer

E.g., an 8-bit integer can only represent 256 integer values

Background

• Computers are limited in how they can represent data
  • Example – 1/3 can be represented mathematically, but in a computer is it .3 or .33 or .333 or .3333 or …
  • Some numbers can be represented exactly, others cannot
Signed Integer Representation

- The high-order bit usually indicates the sign of an integer
  - The high-order bit is the leftmost bit (also called the most significant bit or sign bit)
  - 0 is used to indicate a positive number; 1 indicates a negative number
  - The remaining bits contain the value of the number (but this can be interpreted in different ways)

You will see machine instructions that examine the “sign bit” of an integer

Integer Representations

- There are three ways in which signed binary integers may be expressed:
  - Signed magnitude
  - One’s complement
  - Two’s complement (most common)
- In an 8-bit integer, signed magnitude representation places the absolute value of the number in the 7 bits to the right of the sign bit

We mainly cover 2’s complement representation (text covers all 3)
Signed Magnitude

- For example, in 8-bit signed magnitude representation:
  - +3 is: 00000011
  - -3 is: 10000011

This is an intuitive representation for humans, but not ideal for computers

Complement - Terminology

- General term that means something that completes
  - Color – a color, which when combined with its complement forms black (or white)
  - Music – an interval, which when combined with its complement forms an octave
  - Geometry – an angle, which when combined with its complement, forms a right angle

Computer circuits can form a complement very easily
Complement Systems

- In arithmetic, we just represent a negative number with a minus sign in front of the number.
- In computers, we represent a negative number with a different set of bits so that we can use an adder to add a positive and a negative number.
- We still would like to access a sign bit to easily determine if a number is negative.

Complement Systems

- More general representation of binary numbers.
- Subtraction can be thought of as the addition of a positive number with the complement of the subtrahend.
- Negative values are represented by some difference between a number and its base.

E.g., 48 - 19 is the same as 48 + (-19)

minuend - subtrahend = difference

If we represent a negative number as a complement, the addition circuit can be used for both addition and subtraction.
Computing the Complement

- Computing – a bit sequence whose complement contains the opposite bit at each position
- Sometimes referred to as “flipping the bits”

One’s Complement

- One’s complement of a binary number – represents the negation of that number
- Behaves like the negative of the original number in most arithmetic operations
- It amounts to little more than flipping the bits of a binary number

What do you get if you add these two binary numbers?

0000 0001 represents 1
1111 1110 represents -1

Definition - Wikipedia
One's Complement

• For example, using one's complement in an 8-bit representation:
  - +3 is: 0000 0011
  - -3 is: 1111 1100

• In one's complement representation, as with signed magnitude, negative values are indicated by a 1 in the high order bit

If we add the binary representation of +3 and -3 we get 1111 1111 or negative zero

Why do we refer to 1111 1111 as negative zero?

One's Complement Subtraction

• The difference of two values is found by adding the minuend to the complement of the subtrahend

48 is the minuend and 19 is the subtrahend

E.g., 48 - 19 is the same as 48 + (-19)

• First form the complement on the number to be subtracted, then add the minuend to the complement

Complement circuitry allows the adder to be used for subtraction
Use of Carry in Addition

- Decimal
  
  When a pair of digits equals or exceeds the base number, you mark a carry.

- Binary

One's Complement Addition

- The final carry bit is “carried around” and added to the sum.

- Example: Using one's complement binary arithmetic, find the sum of 48 and -19

19 in binary is: 0001 0011
-19 in one's complement is: 1110 1100

Can you verify that this is correct?

48
-19

\[
\begin{array}{c}
11101100 \\
00110000 \\
\hline
11001100 \\
00011100 \\
+ 1 \\
\hline
00011101
\end{array}
\]
Transformation to Integer

- The result of your addition was a one’s-complement number
- In the computer this is OK since integers are stored as one’s-complement
- To verify the value in a hand calculation, you can transform to decimal integer

Are We on Track?

- Perform one’s complement subtraction (53-23)
  - Express 53 and 23 in binary
  - Form the one’s complement of 23
  - Add the two binary numbers
  - Express the result in decimal
  - Check with your decimal subtraction

Use 8 bits to represent each number
Were We on Track?

• Using one’s complement binary arithmetic, find the sum of -23 and 53

53 in binary is: 00110101,
23 in binary is: 00010111
-23 in one’s complement is: 11101000

\[
\begin{array}{c}
00110101 \quad \text{(53)} \\
11101000 \quad \text{(-23)} \\
00011101 \quad \text{(plus carry)} \\
\hline
00011110 \quad \text{(30)}
\end{array}
\]

Comments

• One’s complement:
  • is simpler to implement than signed magnitude
  • has the disadvantage of having two different representations for zero: positive zero and negative zero
Multiple Representations of Zero

- In one’s complement, zero is represented as
  - 1111 1111 and
  - 0000 0000
- Can cause problems or complications in some computer operations

Two’s Complement Representation

- Defined as the complement with respect to $2^N$ (the result of subtracting the number from $2^N$)
- To express a value in two’s complement representation:
  - If the number is positive, just convert it to binary
  - If the number is negative, find the one’s complement of the number and then add 1
- Example:
  - In 8-bit binary, 3 is: 0000 0011
  - -3 using one’s complement representation is 1111 1100
  - For 2’s complement, add 1: 1111 1100 + 1 = 1111 1101
Two’s Complement Arithmetic

- Add our two binary numbers
- Discard any carry emitting from the high order bit

Example: Using one’s complement binary arithmetic, find the sum of 48 and -19

\[
\begin{array}{c}
00110000 \\
+11101101 \\
\hline
11001101
\end{array}
\]

19 in binary is 00010011,
-19 using one’s complement is 11101100,
and -19 using two’s complement is 11101101

Transformation to Integer

- The result of your addition was a two’s-complement number
- In the computer this is OK since integers are stored as two’s-complement
- To verify the value using a hand calculation, you can transform to integer
  - If the number has 0 in the sign bit, no change is needed
  - If the number has 1 in the sign bit, subtract 1 from the result, then reverse the bits

Think of this as just doing the reverse of what you did to form the 2’s complement
Example

- Using two's complement binary arithmetic, find the sum of -48 and 19

19 in binary is: 00010011,
48 in binary is: 00110000
-48 in two's complement is:
  11001111
  +00000001
  11010000

The result is negative, so we transform the result to integer

Addition in two's complement

00010011   (19)
11010000   (-48)
11100011

Flipping the bits in the result above gives us 00011101 or 29, hence the result is -29

Convert result to 1’s complement

11100011
-00000001 (subtract 1)
11100000

Are We on Track?

- Perform two’s complement subtraction (23-53)
  - Express 53 and 23 in binary
  - Form the two’s complement of 53
  - Add the two binary numbers
  - Express the result in decimal
  - Check with your decimal subtraction

Use 8 bits to represent each number

Use 8 bits to represent each number
Were We on Track?

- Using two's complement binary arithmetic, find the sum of 23 and -53

<table>
<thead>
<tr>
<th>53 in binary is: 00110101,</th>
<th>23 in binary is: 00010111</th>
</tr>
</thead>
<tbody>
<tr>
<td>-53 in one's complement is:</td>
<td>11001010</td>
</tr>
<tr>
<td>so two's complement is</td>
<td>11001010</td>
</tr>
<tr>
<td>11001010 + 1 = 11001011</td>
<td>-00000001 (subtract 1)</td>
</tr>
</tbody>
</table>

Addition in two's complement

00010111 (23)
11001011 (-53)
11000010

flip the bits of the result above to find the result
00011110 (30)
Since the original result was negative the answer is -30

Overflow

- Calculating a number that is too large or too small to be represented correctly in a computer
- Overflow is always possible when we use any finite number of bits to represent a number
- While we can’t always prevent overflow, we can always detect overflow
- In complement arithmetic, an overflow condition is easy to detect
Overflow in Java

- Example causes an overflow of a short
- Notice that the overflow causes a wrap

```
package lectures;

import java.io.*;

public class IntOverflow {
    public static void main(String[] args) {
        short s = 32767;
        for (short i = 0; i < 10; i++) {
            s++;
        }
        System.out.println(s);
    }
}
```

Output: Code5E2M9P1U

```
-4
32764
32763
32762
32761
-32764
-32765
-32766
```

What is the maximum number of distinct integers you can represent with 8 bits?

What is the range of a 8-bit integer in two’s complement?

Integer Range

- Example:
  - Using two’s complement binary arithmetic, find the sum of 107 and 46
  - We see that the nonzero carry from the seventh bit overflows into the sign bit, giving us the erroneous result: 107 + 46 = -103.
- Ranges
  - One’s: $-(2^{8-1} - 1)$ to $+(2^{8-1} - 1)$
  - Two’s: $-(2^{8-1})$ to $+(2^{8-1} - 1)$
Detecting Overflow

- Example:
  - Using two’s complement binary arithmetic, find the sum of 23 and -9
  - We see that there is carry into the sign bit and carry out. The final result is correct: $23 + (-9) = 14$.

Overflow into the sign bit does not always mean that we have an error.

Rule for detecting signed two’s complement overflow:
When the “carry in” and the “carry out” of the sign bit differ, overflow has occurred. If the carry into the sign bit equals the carry out of the sign bit, no overflow has occurred.

Did You Satisfy the Objectives?

- Understand the fundamentals of integer data representation and manipulation in digital computers
- Gain experience in working with binary numbers
- Understand the difference between a number and the representation of that number in a computer
- Understand how errors can occur in computations because of overflow and truncation