POLYTECHNIC UNIVERSITY
Department of Computer and Information Science

Representing Integers in Digital Computers

K. Ming Leung

Abstract: We explain how integers are represented and stored in modern digital computers.

Directory

Last Revision Date: January 21, 2004

The smallest and most basic data item in a digital computer is called a bit. Physically a bit is a switch that can be either open or closed. We can associate that with a value of 0 or 1. A bit by itself has limited usefulness. Usually 8 of them are grouped together to from a byte. A single byte consisting of 8 bits, each of which can be a 0 or 1, can therefore have any one of $2^8 = 256$ distinct patterns. In C++ for example, a single byte is used to represent a single ASCII character.

1. Representing and Storing Integers

The patterns used to store numbers are called number codes. The most common number code for storing integer values in a computer is called the two's complement representation. For convenience we will first assume that each integer is stored in a single byte. We will then extend it to larger bit-size patterns.

To determine the integer represented by a given bit pattern, we start by constructing the so-called value box. Starting from the right we have $2^0 = 1, 2^1 = 2, 2^2 = 4, \ldots, 2^6 = 64$. The last one (the left-most one) is $2^7 = 128$, but we reverse its sign to give -128. These

values are placed on the top of the value box.

```
-128| 64 | 32 | 16 |  8 |  4 |  2 |  1
----|----|----|----|----|----|----|----
    |    |    |    |    |    |    |
```

Conversion of any binary number, for example `10001101`, simply requires inserting the bit pattern into the value box and adding the values having ones under them.

```
-128 | 64 | 32 | 16 |  8 |  4 |  2 |  1
 ----|----|----|----|----|----|----|----
   1 |  0 |  0 |  0 |  1 |  1 |  0 |  1
-128 +  0 +  0 +  0 +  8 +  4 +  0 +  1  = -115
```

Thus the bit pattern `10001101` represents the integer `-115`.

In reviewing the value box, it is evident that any two's complement binary number with a leading 1 represents a negative number, and any bit pattern with a leading 0 represents a positive number. The most negative number that can be stored is the decimal number `-128`, which has the bit pattern `10000000`. It is clear that a positive number must have a 0 as it left-most bit. From this you can see that the largest

positive 8-bit two's complement number is `127` which has a bit pattern `01111111`.

Thus using a single byte all integers from `-128` to `127` can be represented. But how about integers outside this range? Notice that if we add 1 to the bit pattern `01111111` which represent the integer`127`, we have the bit pattern `10000000` which represents the integer `-128`. Similarly if you subtract `1` from `-128` you will get `127`. Thus the numbers are wrapped around in a cyclic fashion. The bit pattern for 0 is `00000000`. Adding 1 to the bit pattern gives the bit pattern `00000001` for integer 1. Continuing this way we finally get to the bit pattern `01111111` which represent the integer `127`. After that we get the bit pattern `10000000` which represents the integer `-128`. Continuing this way we get the bit pattern `10000001` which represents the integer `-127`, etc., until finally we have the bit pattern `11111111` which represents the integer `-1`.

Of course 127 or 128 are not very large integers. In order to be able to represent integers outside of `-128` to `127`, two or more bytes are grouped together into larger units, called words, which facilitate faster and more extensive data access. The number of bytes in a word

determines the maximum and minimum integer values that can be represented. The table lists these values for 1, 2 and 4 byte words, each of the values listed can be derived using 8-, 16-, and 32-bit value boxes, respectively.

Word Size	Minimum Integer Value	Maximum Integer Value
1 Byte (8-bits)	-128	127
2 Bytes (16-bits)	-32,768	32,767
4 Bytes (32-bits)	-2,147,483,648	2,147,483,647

In C++ running on modern 32-bit environment, the minimum and maximum integer values are `-2,147,483,648` and `2,147,483,647` and are stored in pre-defined global constants `INT_MIN` and `INT_MIN`, respectively.

MATLAB has no integers (except in storage). Integers are treated as floating point numbers (more precisely as an array of doubles).