Basics

Chapter 3

Data And Program Representation

Topics

- Introduction
- Digital Logic And Abstraction
- Bits, Bytes
- Byte Size And Possible Values
- Binary Arithmetic
- Hexadecimal Notation
- Notation For Constants
- Character Sets
- Unsigned Integers, Overflow, And Underflow

Topics

- The Little Big Endian
- Signed Integers
- Unsigned And Equivalent Two's Complement
- Sign Extension
- Floating Point
- IEEE Standard 754 Specification For Single And Double Precision
- Other Representation Issues
- Class Exercises
- Vocabulary

Introduction

- How digital systems encode programs and data
  - i.e. Data Program representation
- Representation understanding important for
  - hardware engineers
  - software programmers

Digital Logic And Abstraction

- Abstraction
  - hide underlying details and use high level representations.
  - example False is 0V.
  - allows complex systems like processors and memories to be built without worrying about low level details like individual transistors
- Programmer concerned about representation used for data and programs

Bits, Bytes

- *Bit*
  - binary digit that describes a digital entity
  - two permissible values 0, 1.
- *Byte*
  - multiple bits to represent complex data
  - smallest data item larger than a bit that hardware can manipulate
- So how big is a byte ?
  - 6, 8, 10 bits ?
  - depends on the computer
  - 8 is default

Byte Size And Possible Values

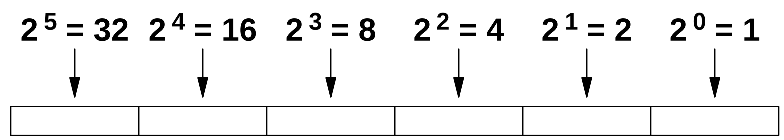
- Programs use bytes to store values.
- Byte size important to programmers.
- Byte size determines maximum value that can be stored.
  - k bits, byte can represent 2^k values
  - maximum value is 2^k-1
- Example with 2 bits
  - 00, 01, 10, 11
  - above 4 values can be represented, maximum value is 3
- Bits do not have intrinsic meaning, they are interpreted by hardware or software

Binary Arithmetic

- Decimal 123
  - $1 \text{ X } 100 + 2 \text{ X } 10 + 3 \text{ X } 1$
  - i.e. $1 \text{ X } 10^2 + 2 \text{ X } 10^1 + 3 \text{ X } 10^0$
- Binary 010101
  - $0 \text{ X } 2^5 + 1 \text{ X } 2^4 + 0 \text{ X } 2^3 + 1 \text{ X } 2^2 + 0 \text{ X } 2^1 + 1 \text{ X } 2^0 = 21$

Binary Arithmetic

- Note, k bits can represent 2^k values between 0 and $2^{(k-1)}$



- Value associated with the first six positions in binary number system.

Hexadecimal Notation

- Programmers find
  - decimal difficult to understand
  - binary may be unwieldy (long strings)
- Compromise
  - positional numbering system with larger base
  - base multiple of two, easy to translate to binary

Hexadecimal Notation

- Solution: Hexadecimal = base 16
  - strings are shorter.
  - converting to binary is trivial
- Encode 4 bits to single hex between 0-15
  - Hex 0-F
  - Decimal 0-15
  - Binary 0000-1111

Notation For Constants

- How to tell if the number is binary, decimal, or hex?
  - subscripts for non-decimal, example 132_{16}.
  - prefi x 0x for hexadecimal, 0b for binary.
  - example 0x132.

Character Sets

- Defi ned by the computer system, encoding decided by architect
- Need representation for upper/lower case letters, digits, punctuation.
- Byte size-character set relationship: 8 bit byte allows a 256 characters set.

Character Sets

- Encoding standards
  - EBCDIC
    - *Extended Binary Coded Decimal interchange Code*
  - ASCII (ANSI)
    - *American Standard Code for Information Interchange*
    - *American National Standards Institute*
    - 7-bit bytes
  - Unicode
    - proposal for 16+ bit set to accommodate all languages

Unsigned Integers, Overflow, And Underflow

- Situations arising with unsigned integer operations
  - overflow during addition
  - negative result from subtraction
- Solution
  - wraparound, and set carry bit to overflow/underflow

$$
\begin{array}{r}
100 \\
+\ 110 \\
\hline
\underset{\textit{overflow}}{1}\underbrace{010}_{\textit{result}}
\end{array}
$$

- Illustration of addition with unsigned producing overflow.

The Little Big Endian

- Should set of bits be numbered from left or right?
  - from left if viewed as a string.
  - from right (least signifi cant bit, LSB) if viewed as a binary number
- What happens when transferring 32 bits, byte at time?

The Little Big Endian

- Little Endian: number from LSB to MSB
- Big endian: number from MSB to LSB
- Bit little endian: number bits within byte from LSB to MSB

0	1	2	3
0x00	0x00	0x00	0x01

Big Endian

3	2	1	0
0x00	0x00	0x00	0x01

Little Endian

- Illustration of big endian and little endian byte numbering for a 32 bit integer equal to 1.

Signed Integers

- Positional representation can’t deal with negative numbers
- Alternatives
  - signed magnitude: a separate bit (leftmost i.e.e MSB) stores the sign
  - 1’s complement: reverse each bit for negative
  - 2’s complement: subtract 1, then reverse each bit

Signed Integers

- Quirks in signed integers
  - sign magnitude: Negative 0
  - 1's complement: two values for 0
  - 2's complement: one extra negative number than positive numbers

- Best interpretation: sign magnitude, 1's or 2's complement?
  - depends !!
  - computer architects usually choose 2's complement

Signed Integers

- Example of 2's complement
  - K bits now represents $2^{(k-1)}-1$ numbers
  - MSB is 1 for negative numbers
- Computer can use single piece of hardware to provide unsigned or 2's complement integer arithmetic
- Software can choose the interpretation

Unsigned And Equivalent Two's Complement

Binary Value	Unsigned Equivalent	Two's Complement Equivalent
1111	15	-1
1110	14	-2
1101	13	-3
1100	12	-4
1011	11	-5
1010	10	-6
1001	9	-7
1000	8	-8
0111	7	7
0110	6	6
0101	5	5
0100	4	4
0011	3	3
0010	2	2
0001	1	1
0000	0	0

Sign Extension

- What happens if smaller size integer is copied to larger size integer in 2's complement ?
  - Positive: fi ll MSB's with 0.
  - Negative: fi ll MSB's with 1's
- Because 2's complement hardware performs sign extension; copying an unsigned integer to a larger unsigned integer changes the value.

Floating Point

- Mantissa and exponent are encoded separately in binary
- Advantage, optimizes space
  - normalize value
    * remove leading 0’s from mantissa
    * leading bit is implicit as 1 and can be dropped, saving a bit
  - exponent can be biased to allow representation of very small or very large numbers

IEEE Standard 754 Specification For Single And Double Precision

Other Representation Issues

- Special values
  - positive and negative infinity
    * exponent all 1; mantissa all 0
    * Helps software determine overflows
- Range
  - 2^{-126} to 2^{127}
  - 10^{-38} to 10^{38}
  - double precision 10^{-308} to 10^{308}

Other Representation Issues

- Data Aggregates
  - arrays, records, structures stored in contiguous bytes
  - note: some memories disallow (i)
- Programs stored in memory
  - instructions and data

Class Exercises

- Choosing data representation (number of bits) for designing a keyboard
- Decimal to binary, decimal to octal, decimal to hex conversion
- Binary to decimal, octal to decimal, hex to decimal conversion
- Converting decimal fraction to binary equivalent
- Converting binary fraction to decimal equivalent

Class Exercises

- Converting sign magnitude numbers to equivalent 1’s and 2’st complement
- Addition and subtraction using 1’s and 2’s complement
- Computing suitable bias constant to improve accuracy of very small numbers

Vocabulary

- Bits, bytes
- Binary, hex, decimal
- Ascii character set
- Overfbw, underfbw
- Big little endian
- Sign-magnitude, 1’s, 2’s complement
- Floating point
- and that´s it ..