Topics

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

Dr. Rajesh Subramaniam, 2005
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

The Little Big Endian
Introduction

- How digital systems encode programs and data
- i.e., Program and Data Representation
  - How do digital systems encode programs and data
  - Representation understanding is important for
    - hardware engineers
    - software programmers
Digital Logic and Abstraction

- Abstraction
  - hides underlying details and uses 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.

• **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
  - 6, 8, 10 bits?
  - depends on the computer
  - 8 is default

So how big is a byte?

**Bits, Bytes**
Bytes do not have intrinsic meaning; they are interpreted by hardware or software

- Bits do not have intrinsic meaning; they are interpreted by hardware or software

Example with 2 bits

-maximum value is $2^2 - 1$
  - maximum value is 7

- 00, 01, 10, 11
  - above 4 values can be represented, maximum value is 3

- K bits, byte can represent $2^k$ values

Byte size determines maximum value that can be stored.

Byte size important to programmers.

Programs use bytes to store values.

Byte Size And Possible Values
Binary Arithmetic

- \( 123 \) in decimal is equal to \( 2 \times 10^2 + 1 \times 10^1 + 3 \times 10^0 \).
- \( 010101 \) in binary is equal to \( 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 21 \) in decimal.

- Binary Arithmetic

*Binary 010101 - Decimal 123*
• 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.

```
  2^5 = 32  2^4 = 16  2^3 = 8  2^2 = 4  2^1 = 2  2^0 = 1
```

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

Binary Arithmetic
Hexadecimal Notation

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

CS250 -- Part 1
Dr. Rajesh Subramanyan, 2005
Hexadecimal Notation

- Solution: Hexadecimal = base 16
  - Strings are shorter.
  - Converting to binary is trivial
  - Encode 4 bits to single hex between 0-15

Hexadecimal Notation
Notation For Constants

- How to tell if the number is binary, decimal, or hexadecimal?

  - Subscripts for non-decimal, example, example \(132_{16}\).
  - Prefix x 0x for hexadecimal, 0b for binary.
  - Example 0x132.

How to tell if the number is binary, decimal, or hex?

Notation For Constants
Character Sets

- Byte size-character set relationship: 8 bit byte allows a 256
  characters set.

- Need representation for upper/lower case letters, digits,
  punctuation.

- Architected by the computer system, encoding decided by
  architect.

Character Sets
Character Sets

- Encoding standards
  - EBCDIC
  - ASCII (ANSI)
  - Extended Binary Coded Decimal Interchange Code

Proposal for 16+ bit set to accommodate all languages

- Unicode
- 7-bit bytes

American National Standards Institute
American Standard Code for Information Interchange

Encoding standards
Unsigned Integers, Overflow, And Underflow

- Situations arising with unsigned producing overflow.

- Illustration of addition with unsigned producing overflow.

\[
\begin{array}{c}
\text{overflow result}
\end{array}
\]
\[
\begin{array}{c}
1010
\end{array}
\]
\[
\begin{array}{c}
1000
\end{array}
\]
\[
\begin{array}{c}
100
\end{array}
\]

- Wraparound, and set carry bit to overflow/underflow.

Solution

- Negative result from subtraction.

- Overflow during addition.

- Situations arising with unsigned integer operations.

Unsigned Integers, Overflow, And Underflow
The Little Big Endian

• Should set of bits be numbered from left or right?

− from left if viewed as a string
− from right (least significant cant bit, LSB) if viewed as a binary number

What happens when transferring 32 bits, byte at a time?
Little Endian: number from LSB to MSB
Big Endian: number from MSB to LSB

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

<table>
<thead>
<tr>
<th>Little Endian</th>
<th>0x00 0x00 0x00 0x01</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Big Endian</th>
<th>0x01 0x00 0x00 0x00</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

The Little Big Endian
Signed Integers

- Alternatives

Positional representation can't deal with negative numbers

Signed Integers
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

Best interpretation: sign magnitude, 1's or 2's complement?

- Depends!!
- Computer architects usually choose 2's complement

Signed Integers
Signed Integers

• Example of 2's complement

− K bits now represents \(2^{k-1}\) numbers

− MSB is 1 for negative numbers

\[\text{Computer can use single piece of hardware to provide unsigned or 2's complement integer arithmetic}\]

Software can choose the interpretation

Signed Integers
<table>
<thead>
<tr>
<th>Binary</th>
<th>Unsigned</th>
<th>Equivalent Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0100</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1110</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1010</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1110</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0000</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>1000</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0100</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>1110</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>0100</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>1110</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>0100</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>1110</td>
</tr>
</tbody>
</table>

Unsigned And Equivalent Two's Complement
• What happens if smaller size integer is copied to larger size integer in 2's complement?

  - Positive: fill MSB's with 0.
  - Negative: fill MSB's with 1.

Because 2's complement hardware performs sign extension:

- Negative: if II MSB's with 1.
- Positive: if II MSB's with 0.

What happens if smaller size integer is copied to larger size integer?
Floating Point

- Mantissa and exponent are encoded separately in binary
- Advantage, optimizes space

- Normalize value
- Advantage, optimizes space

- Normalized value
  + Leading bit is implicit as 1 and can be dropped, saving 1 bit
  + Remove leading 0s from mantissa

- Exponent can be biased to allow representation of very small or very large numbers
Double Precision
IEEE Standard 754 Specification For Single And
Other Representation Issues

- Special values
  - positive and negative infinity
  - exponent all 1; mantissa all 0
  - Helps software determine overflows

- Range
  - double precision: 10^-308 to 10^308
  - 10^-38 to 10^38
  - 2^-126 to 2^127
Other Representation Issues

- Data Aggregates
  - Arrays, records, structures stored in contiguous bytes
  - Note: some memories disallow (i)
  - Programs stored in memory
  - Instructions and data

Data Aggregates
Class Exercises

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

Keyboard

Choosing data representation (number of bits) for designing a
Class Exercises

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

• Bits, bytes
• Binary, hexadecimal
• ASCII character set
• Binary, hex, decimal
• Floating point
• Sign-magnitude, 1's, 2's complement
• Overflow, underflow
• Big little endian
• and that's it