Reminder:

• we will review Homework 2 in class on Tuesday
• due Monday (by midnight) to pllee@email.arizona.edu
Today's Topic

• Let's continue talking about the computer representation of data (numbers and characters) ...

interesting use of binary notation: 6 bit Leica camera lens encoding

The 6-bit code shown on the right is 101011 (1 = black, 0 = white), read starting from the philips screw head
Introduction

• Storage:
  • based on digital logic
  • binary (base 2) – everything is a power of 2
  • Byte: 8 bits
    • 01011011
    • = $2^6+2^4+2^3+2^1+2^0$
    • = $64 + 16 + 8 + 2 + 1$
    • = 91 (in decimal)
  • Hexadecimal (base 16)
    • 0-9,A,B,C,D,E,F (need 4 bits)
    • 5B (= 1 byte)
    • = $5*16^1 + 11$
    • = $80 + 11$
    • = 91
Introduction: data types

• Integers
  • In one byte (= 8 bits), what’s the largest and smallest number, we can represent?
  • **Answer:** -128 .. 0 .. 127
  • Why? -2^{8-1} .. 0 .. 2^{8-1} – 1
  • 00000000 = 0
  • 01111111 = 127
  • 10000000 = -128
  • 11111111 = -1

\[
\begin{array}{cccccccc}
2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0\]

\[2^8 - 1 = 255\]

\[
\begin{array}{cccccccc}
2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[2^7 - 1 = 127\]

\[
\begin{array}{cccccccc}
2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Introduction: data types

• Integers
  • In one byte (= 8 bits), what’s the largest and smallest number, we can represent?
  • 00000000 = 0
  • 01111111 = 127
  • 10000000 = -128
  • 11111111 = -1

00000000       11111111
  0 ...       127 -128 -127       -1

2’s complement representation
Introduction: data types

• Integers
  • In one byte, what’s the largest and smallest number, we can represent?
  • **Answer**: -128 .. 0 .. 127 using the *2’s complement representation*
  • Why? super-convenient for arithmetic operations
  • “to convert a positive integer X to its negative counterpart, flip all the bits, and add 1”
  • **Example**:
    • 00001010 = $2^3 + 2^1 = 10$ (decimal)
    • 11110101 + 1 = 11110110 = -10 (decimal)
    • 11110110 flip + 1 = 00001001 + 1 = 00001010

**Addition:**
-10 + 10
= 11110110
+ 00001010 = 0 (ignore overflow)
Introduction: data types

- Typically 32 bits (4 bytes) are used to store an integer
  - range: -2,147,483,648 ($2^{31} - 1$) to 2,147,483,647 ($2^{32} - 1$)

```
  2^{31}  2^{30}  2^{29}  2^{28}  2^{27}  2^{26}  2^{25}  2^{24}  ...  ...  2^{7}  2^{6}  2^{5}  2^{4}  2^{3}  2^{2}  2^{1}  2^{0}
```

```
byte 3 | byte 2 | byte 1 | byte 0
```

- what if you want to store even larger numbers?
  - Binary Coded Decimal (BCD)
  - code each decimal digit separately, use a string (sequence) of decimal digits ...
Introduction: data types

• what if you want to store even larger numbers?
  • Binary Coded Decimal (BCD)
  • 1 byte can code two digits (0-9 requires 4 bits)
  • 1 nibble (4 bits) codes the sign (+/-), e.g. hex C/D

<table>
<thead>
<tr>
<th>2^3</th>
<th>2^2</th>
<th>2^1</th>
<th>2^0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

0

1

9

credit (+)

2 bytes (= 4 nibbles)

2.5 bytes (= 5 nibbles)

debit (-)
Introduction: data types

• Typically, 64 bits (8 bytes) are used to represent floating point numbers (double precision)
  • \( c = 2.99792458 \times 10^8 \) (m/s)
  • coefficient: 52 bits (implied 1, therefore treat as 53)
  • exponent: 11 bits (not 2’s complement, unsigned with bias)
  • sign: 1 bit (+/-)

C:
float
double

x86 CPUs have a built-in floating point coprocessor (x87)
80 bit long registers
Introduction: data types

• How about letters, punctuation, etc.?

• ASCII
  • American Standard Code for Information Interchange
  • Based on English alphabet (upper and lower case) + space + digits + punctuation + control (Teletype Model 33)

• Question: how many bits do we need?
  • 7 bits + 1 bit parity
  • Remember everything is in binary ...

ASCII TABLE

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>002</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>07F</td>
<td>?</td>
</tr>
</tbody>
</table>

Teletype Model 33 ASR
Teleprinter (Wikipedia)
Introduction: data types

Order is important in sorting!

0-9: there's a connection with BCD. **Notice**: code 30 (hex) through 39 (hex)

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>[NUL]</td>
<td>32</td>
<td>20</td>
<td>[SPACE]</td>
<td>64</td>
<td>40</td>
<td>@</td>
<td>96</td>
<td>60</td>
<td>`</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>[START OF HEADING]</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>[START OF TEXT]</td>
<td>34</td>
<td>22</td>
<td>*</td>
<td>66</td>
<td>42</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>[END OF TEXT]</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>[END OF TRANSMISSION]</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>[ENQUIRY]</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>[ACKNOWLEDGE]</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>f</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>[BEL]</td>
<td>39</td>
<td>27</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>g</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>[VERNIER]</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>h</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>[VERTICAL TAB]</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>i</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>[EN QUOTE]</td>
<td>42</td>
<td>2A</td>
<td>+</td>
<td>74</td>
<td>4A</td>
<td>J</td>
<td>106</td>
<td>6A</td>
<td>j</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>[EN SPACE]</td>
<td>43</td>
<td>2B</td>
<td>,</td>
<td>75</td>
<td>4B</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>k</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>[FORM FEED]</td>
<td>44</td>
<td>2C</td>
<td>.</td>
<td>76</td>
<td>4C</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>l</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>[CARRIAGE RETURN]</td>
<td>45</td>
<td>2D</td>
<td>;</td>
<td>77</td>
<td>4D</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>m</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>[SHIFT OUT]</td>
<td>46</td>
<td>2E</td>
<td>&lt;</td>
<td>78</td>
<td>4E</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>n</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>[SHIFT IN]</td>
<td>47</td>
<td>2F</td>
<td>=</td>
<td>79</td>
<td>4F</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>o</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>[DATA LINK ESCAPE]</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>p</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>[DEVICE CONTROL 1]</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>[DEVICE CONTROL 2]</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>[DEVICE CONTROL 3]</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>s</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>[DEVICE CONTROL 4]</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>t</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>[NEGATIVE ACKNOWLEDGE]</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>u</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>[SYNCHRONOUS CR]</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>V</td>
<td>118</td>
<td>76</td>
<td>v</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>[END OF Trans. BLOCK]</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>W</td>
<td>119</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>[CANCEL]</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>X</td>
<td>120</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>[END OF MEDIUM]</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>Y</td>
<td>121</td>
<td>79</td>
<td>y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>[SUBSTITUTE]</td>
<td>58</td>
<td>3A</td>
<td>:</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
<td>122</td>
<td>7A</td>
<td>z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>[ESCAPE]</td>
<td>59</td>
<td>3B</td>
<td>;</td>
<td>91</td>
<td>5B</td>
<td>]</td>
<td>123</td>
<td>7B</td>
<td>] (</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>[FILE SEPARATOR]</td>
<td>60</td>
<td>3C</td>
<td>&lt;</td>
<td>92</td>
<td>5C</td>
<td>\</td>
<td>124</td>
<td>7C</td>
<td>\ [</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>[GROUP SEPARATOR]</td>
<td>61</td>
<td>3D</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>)</td>
<td>125</td>
<td>7D</td>
<td>) )</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>[RECORD SEPARATOR]</td>
<td>62</td>
<td>3E</td>
<td>&gt;</td>
<td>94</td>
<td>5E</td>
<td>^</td>
<td>126</td>
<td>7E</td>
<td>~</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>[UNIT SEPARATOR]</td>
<td>63</td>
<td>3F</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>_</td>
<td>127</td>
<td>7F</td>
<td>[DEL]</td>
</tr>
</tbody>
</table>

**Notice**: code 30 (hex) through 39 (hex)
Introduction: data types

• Parity bit:
  • transmission can be noisy
  • parity bit can be added to ASCII code
  • can spot single bit transmission errors
  • even/odd parity:
    • receiver understands each byte should be even/odd
  • Example:
    • 0 (zero) is ASCII 30 (hex) = 011000
    • even parity: 0110000, odd parity: 0110001
• Checking parity:
  • Exclusive or (XOR): basic machine instruction
    • \( A \text{ xor } B \) true if either \( A \) or \( B \) true but not both
• Example:
  • (even parity 0) 0110000 xor bit by bit
    • 0 xor 1 = 1 xor 1 = 0 xor 0 = 0 xor 0 = 0 xor 0 = 0 xor 0 = 0

x86 assembly language:
1. PF: even parity flag set by arithmetic ops.
2. TEST: AND (don’t store result), sets PF
3. JP: jump if PF set

Example:
MOV al,<char>
TEST al, al
JP <location if even>
<go here if odd>
Introduction: data types

- UTF-8
  - standard in the post-ASCII world
  - backwards compatible with ASCII
  - (previously, different languages had multi-byte character sets that clashed)
  - Universal Character Set (UCS) Transformation Format 8-bits

<table>
<thead>
<tr>
<th>Bits of code point</th>
<th>First code point</th>
<th>Last code point</th>
<th>Bytes in sequence</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>U+0000</td>
<td>U+007F</td>
<td>1</td>
<td>0xxxxxxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>U+0080</td>
<td>U+07FF</td>
<td>2</td>
<td>110xxxxx</td>
<td>10xxxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>U+0800</td>
<td>U+FFFF</td>
<td>3</td>
<td>1110xxxxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>U+10000</td>
<td>U+1FFFFF</td>
<td>4</td>
<td>11110xxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
</tr>
</tbody>
</table>

(Wikipedia)
Introduction: data types

<table>
<thead>
<tr>
<th>Bits of code point</th>
<th>First code point</th>
<th>Last code point</th>
<th>Bytes in sequence</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>U+0000</td>
<td>U+007F</td>
<td>1</td>
<td>0xxxxxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>U+0080</td>
<td>U+07FF</td>
<td>2</td>
<td>110xxxx</td>
<td>10xxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>U+0800</td>
<td>U+FFFF</td>
<td>3</td>
<td>1110xxxx</td>
<td>10xxxxx</td>
<td>10xxxxx</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>U+10000</td>
<td>U+1FFFF</td>
<td>4</td>
<td>11110xxx</td>
<td>10xxxxx</td>
<td>10xxxxx</td>
<td>10xxxxx</td>
</tr>
</tbody>
</table>

Example:

- わ Hiragana letter A: UTF-8: E38182
  - Byte 1: E = 1110, 3 = 0011
  - Byte 2: 8 = 1000, 1 = 0001
  - Byte 3: 8 = 1000, 2 = 0010
  - い Hiragana letter I: UTF-8: E38184

Many Windows programs (including Windows Notepad) add the bytes 0xEF, 0xBB, 0xBF at the start of any document saved as UTF-8. This is the UTF-8 encoding of the Unicode byte order mark (BOM), and is commonly referred to as a UTF-8 BOM, even though it is not relevant to byte order. A BOM can also appear if another encoding with a BOM is translated to UTF-8 without stripping it. Software that is not aware of multibyte encodings will display the BOM as three strange characters (e.g. "è«ž" in software interpreting the document as ISO 8859-1 or Windows-1252) at the start of the document.
Introduction: data types

• How can you tell what encoding your file is using?
• Detecting UTF-8
  • Microsoft:
    • 1st three bytes in the file is EF BB BF
    • (not all software understands this; not everybody uses it)
  • HTML:
    • `<meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
    • (not always present)
  • Analyze the file:
    • Find non-valid UTF-8 sequences: if found, not UTF-8...
    • Interesting paper:
Introduction: data types

• Text files:
  • text files have lines: *how do we mark the end of a line?*
  • End of line (EOL) control character(s):
    • LF 0x0A (Mac/Linux),
    • CR 0x0D (Old Macs),
    • CR+LF 0x0D0A (Windows)
• End of file (EOF) control character:
  • (EOT) 0x04 (*aka* Control-D)

<table>
<thead>
<tr>
<th>ASCII Code Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>NUL</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>!</td>
</tr>
<tr>
<td>@</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>a</td>
</tr>
<tr>
<td>p</td>
</tr>
</tbody>
</table>

programming languages:
NUL used to mark
the end of a string