Abstraction
Correctness

- **Example 1:** Is $x^2 \geq 0$?
  - **Floats:** Yes!
  - **Ints:**
    - $40000 \times 40000 \rightarrow 1600000000$
    - $50000 \times 50000 \rightarrow ??$

- **Example 2:** Is $(x + y) + z = x + (y + z)$?
  - **Ints:** Yes!
  - **Floats:**
    - $(2^{30} + -2^{30}) + 3.14 \rightarrow 3.14$
    - $2^{30} + (-2^{30} + 3.14) \rightarrow ??$
Performance

• Hierarchical memory organization

• Performance depends on access patterns
  • Including how step through multi-dimensional array

```c
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}

void copyij(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}
```

4.3ms 81.8ms
void admin_stuff(int authenticated) {
    if (authenticated) {
        // do admin stuff
    }
}

int dontTryThisAtHome(char * user_input, int size) {
    char data[size];
    int ret = memcpy(*user_input, data);
    return ret;
}
Bits

• a **bit** is a binary digit that can have two possible values

• can be physically represented with a two state device
For added security, after we encrypt the data stream, we send it through our Navajo Code Talker.

...is he just using Navajo words for "zero" and "one"?

Whoa, hey, keep your voice down!
Storing bits

• Static random access memory (SRAM): stores each bit of data in a flip-flop, a circuit with two stable states
• Dynamic Memory (DRAM): stores each bit of data in a capacitor, which stores energy in an electric field (or not)
• Magnetic Disk: regions of the platter are magnetized with either N-S polarity or S-N polarity
• Optical Disk: stores bits as tiny indentations (pits) or not (lands) that reflect light differently
• Flash Disk: electrons are stored in one of two gates separated by oxide layers
Bytes and Memory

• A **byte** is a unit of eight bits
• **Memory** is an array of bytes

• An index into the array is an **address**, **location**, or **pointer**
  • Often expressed in hexadecimal

• We speak of the **value** in memory at an address
  • The value may be a single byte …
  • … or a multi-byte quantity starting at that address
Binary Numbers

4211

\[= 4 \cdot 10^3 + 2 \cdot 10^2 + 1 \cdot 10^1 + 1 \cdot 10^0\]
\[= 4211\]

1011

\[= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0\]
\[= 11\]
Binary Numbers

On a scale of 1 to 10, how likely is it that this question is using binary?

What's a 4?
Hexidecimal Numbers

- Use digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Compute numbers base 16

1011

\[= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0\]
\[= 11\]

\[= 1 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 1 \cdot 10^0\]
\[= 1011\]

\[= 1 \cdot 16^3 + 0 \cdot 16^2 + 1 \cdot 16^1 + 1 \cdot 16^0\]
\[= 4113\]

- one byte is two digits in hex
## ASCII characters

<table>
<thead>
<tr>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>(sp)</td>
<td>32</td>
<td>0040</td>
<td>0x20</td>
<td>@</td>
<td>64</td>
<td>0100</td>
<td>0x40</td>
<td>`</td>
<td>96</td>
<td>0140</td>
<td>0x60</td>
</tr>
<tr>
<td>!</td>
<td>33</td>
<td>0041</td>
<td>0x21</td>
<td>A</td>
<td>65</td>
<td>0101</td>
<td>0x41</td>
<td>a</td>
<td>97</td>
<td>0141</td>
<td>0x61</td>
</tr>
<tr>
<td>&quot;</td>
<td>34</td>
<td>0042</td>
<td>0x22</td>
<td>B</td>
<td>66</td>
<td>0102</td>
<td>0x42</td>
<td>b</td>
<td>98</td>
<td>0142</td>
<td>0x62</td>
</tr>
<tr>
<td>#</td>
<td>35</td>
<td>0043</td>
<td>0x23</td>
<td>C</td>
<td>67</td>
<td>0103</td>
<td>0x43</td>
<td>c</td>
<td>99</td>
<td>0143</td>
<td>0x63</td>
</tr>
<tr>
<td>$</td>
<td>36</td>
<td>0044</td>
<td>0x24</td>
<td>D</td>
<td>68</td>
<td>0104</td>
<td>0x44</td>
<td>d</td>
<td>100</td>
<td>0144</td>
<td>0x64</td>
</tr>
<tr>
<td>%</td>
<td>37</td>
<td>0045</td>
<td>0x25</td>
<td>E</td>
<td>69</td>
<td>0105</td>
<td>0x45</td>
<td>e</td>
<td>101</td>
<td>0145</td>
<td>0x65</td>
</tr>
<tr>
<td>&amp;</td>
<td>38</td>
<td>0046</td>
<td>0x26</td>
<td>F</td>
<td>70</td>
<td>0106</td>
<td>0x46</td>
<td>f</td>
<td>102</td>
<td>0146</td>
<td>0x66</td>
</tr>
<tr>
<td>'</td>
<td>39</td>
<td>0047</td>
<td>0x27</td>
<td>G</td>
<td>71</td>
<td>0107</td>
<td>0x47</td>
<td>g</td>
<td>103</td>
<td>0147</td>
<td>0x67</td>
</tr>
<tr>
<td>(</td>
<td>40</td>
<td>0048</td>
<td>0x28</td>
<td>H</td>
<td>72</td>
<td>0110</td>
<td>0x48</td>
<td>h</td>
<td>104</td>
<td>0150</td>
<td>0x68</td>
</tr>
<tr>
<td>)</td>
<td>41</td>
<td>0051</td>
<td>0x29</td>
<td>I</td>
<td>73</td>
<td>0111</td>
<td>0x49</td>
<td>i</td>
<td>105</td>
<td>0151</td>
<td>0x69</td>
</tr>
<tr>
<td>*</td>
<td>42</td>
<td>0052</td>
<td>0x2a</td>
<td>J</td>
<td>74</td>
<td>0112</td>
<td>0x4a</td>
<td>j</td>
<td>106</td>
<td>0152</td>
<td>0x6a</td>
</tr>
<tr>
<td>+</td>
<td>43</td>
<td>0053</td>
<td>0x2b</td>
<td>K</td>
<td>75</td>
<td>0113</td>
<td>0x4b</td>
<td>k</td>
<td>107</td>
<td>0153</td>
<td>0x6b</td>
</tr>
<tr>
<td>,</td>
<td>44</td>
<td>0054</td>
<td>0x2c</td>
<td>L</td>
<td>76</td>
<td>0114</td>
<td>0x4c</td>
<td>l</td>
<td>108</td>
<td>0154</td>
<td>0x6c</td>
</tr>
<tr>
<td>-</td>
<td>45</td>
<td>0055</td>
<td>0x2d</td>
<td>M</td>
<td>77</td>
<td>0115</td>
<td>0x4d</td>
<td>m</td>
<td>109</td>
<td>0155</td>
<td>0x6d</td>
</tr>
<tr>
<td>.</td>
<td>46</td>
<td>0056</td>
<td>0x2e</td>
<td>N</td>
<td>78</td>
<td>0116</td>
<td>0x4e</td>
<td>n</td>
<td>110</td>
<td>0156</td>
<td>0x6e</td>
</tr>
<tr>
<td>/</td>
<td>47</td>
<td>0057</td>
<td>0x2f</td>
<td>O</td>
<td>79</td>
<td>0117</td>
<td>0x4f</td>
<td>o</td>
<td>111</td>
<td>0157</td>
<td>0x6f</td>
</tr>
<tr>
<td>0</td>
<td>48</td>
<td>0060</td>
<td>0x30</td>
<td>P</td>
<td>80</td>
<td>0120</td>
<td>0x50</td>
<td>p</td>
<td>112</td>
<td>0160</td>
<td>0x70</td>
</tr>
<tr>
<td>1</td>
<td>49</td>
<td>0061</td>
<td>0x31</td>
<td>Q</td>
<td>81</td>
<td>0121</td>
<td>0x51</td>
<td>q</td>
<td>113</td>
<td>0161</td>
<td>0x71</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>0062</td>
<td>0x32</td>
<td>R</td>
<td>82</td>
<td>0122</td>
<td>0x52</td>
<td>r</td>
<td>114</td>
<td>0162</td>
<td>0x72</td>
</tr>
<tr>
<td>3</td>
<td>51</td>
<td>0063</td>
<td>0x33</td>
<td>S</td>
<td>83</td>
<td>0123</td>
<td>0x53</td>
<td>s</td>
<td>115</td>
<td>0163</td>
<td>0x73</td>
</tr>
<tr>
<td>4</td>
<td>52</td>
<td>0064</td>
<td>0x34</td>
<td>T</td>
<td>84</td>
<td>0124</td>
<td>0x54</td>
<td>t</td>
<td>116</td>
<td>0164</td>
<td>0x74</td>
</tr>
<tr>
<td>5</td>
<td>53</td>
<td>0065</td>
<td>0x35</td>
<td>U</td>
<td>85</td>
<td>0125</td>
<td>0x55</td>
<td>u</td>
<td>117</td>
<td>0165</td>
<td>0x75</td>
</tr>
<tr>
<td>6</td>
<td>54</td>
<td>0070</td>
<td>0x3b</td>
<td>V</td>
<td>86</td>
<td>0126</td>
<td>0x56</td>
<td>v</td>
<td>118</td>
<td>0166</td>
<td>0x76</td>
</tr>
<tr>
<td>7</td>
<td>55</td>
<td>0073</td>
<td>0x3c</td>
<td>W</td>
<td>87</td>
<td>0127</td>
<td>0x57</td>
<td>w</td>
<td>119</td>
<td>0167</td>
<td>0x77</td>
</tr>
<tr>
<td>8</td>
<td>56</td>
<td>0074</td>
<td>0x3d</td>
<td>X</td>
<td>88</td>
<td>0130</td>
<td>0x58</td>
<td>x</td>
<td>120</td>
<td>0170</td>
<td>0x78</td>
</tr>
<tr>
<td>9</td>
<td>57</td>
<td>0077</td>
<td>0x3f</td>
<td>Y</td>
<td>89</td>
<td>0131</td>
<td>0x59</td>
<td>y</td>
<td>121</td>
<td>0171</td>
<td>0x79</td>
</tr>
<tr>
<td>:</td>
<td>58</td>
<td>0072</td>
<td>0x3a</td>
<td>Z</td>
<td>90</td>
<td>0132</td>
<td>0x5a</td>
<td>z</td>
<td>122</td>
<td>0172</td>
<td>0x7a</td>
</tr>
<tr>
<td>;</td>
<td>59</td>
<td>0073</td>
<td>0x3b</td>
<td>[</td>
<td>91</td>
<td>0133</td>
<td>0x5b</td>
<td>{</td>
<td>123</td>
<td>0173</td>
<td>0x7b</td>
</tr>
<tr>
<td>&lt;</td>
<td>60</td>
<td>0074</td>
<td>0x3c</td>
<td>\</td>
<td>92</td>
<td>0134</td>
<td>0x5c</td>
<td></td>
<td></td>
<td>124</td>
<td>0174</td>
</tr>
<tr>
<td>=</td>
<td>61</td>
<td>0075</td>
<td>0x3d</td>
<td>]</td>
<td>93</td>
<td>0135</td>
<td>0x5d</td>
<td>}</td>
<td>125</td>
<td>0175</td>
<td>0x7d</td>
</tr>
<tr>
<td>&gt;</td>
<td>62</td>
<td>0076</td>
<td>0x3e</td>
<td>^</td>
<td>94</td>
<td>0136</td>
<td>0x5e</td>
<td>_</td>
<td>126</td>
<td>0176</td>
<td>0x7e</td>
</tr>
<tr>
<td>?</td>
<td>63</td>
<td>0077</td>
<td>0x3f</td>
<td>_</td>
<td>95</td>
<td>0137</td>
<td>0x5f</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
x86 instructions

<table>
<thead>
<tr>
<th>Machine code bytes</th>
<th>Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>B8 22 11 00 FF</td>
<td>movl $0xFF001122, %eax</td>
</tr>
<tr>
<td>01 CA</td>
<td>addl %ecx, %edx</td>
</tr>
<tr>
<td>31 F6</td>
<td>xorl %es1, %es1</td>
</tr>
<tr>
<td>53</td>
<td>pushl %ebx</td>
</tr>
<tr>
<td>8B 5C 24 04</td>
<td>movl 4(%esp), %ebx</td>
</tr>
<tr>
<td>8D 34 48</td>
<td>leal (%eax,%ecx,2), %esi</td>
</tr>
<tr>
<td>39 C3</td>
<td>cmpl %eax, %ebx</td>
</tr>
<tr>
<td>72 EB</td>
<td>jnae foo</td>
</tr>
<tr>
<td>C3</td>
<td>retl</td>
</tr>
</tbody>
</table>
Bits and Bytes Require Interpretation

00000000 00110101 00110000 00110001 (or 0x00353031) might be interpreted as

- The integer $3,485,745_{10}$
- A floating point number close to $4.884569 \times 10^{-39}$
- The string “105”
- A portion of an image or video
- An address in memory
Information is Bits + Context
```c
#include<stdio.h>

int main(int argc, char** argv) {
    printf("Hello world!\n");
    return 0;
}
```
## Example Data Representations

<table>
<thead>
<tr>
<th>C Data Type</th>
<th>x86-64</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>1</td>
</tr>
<tr>
<td>short</td>
<td>2</td>
</tr>
<tr>
<td>int</td>
<td>4</td>
</tr>
<tr>
<td>long</td>
<td>8</td>
</tr>
<tr>
<td>float</td>
<td>4</td>
</tr>
<tr>
<td>double</td>
<td>8</td>
</tr>
<tr>
<td>pointer</td>
<td>8</td>
</tr>
</tbody>
</table>
int x;    // an integer
int *p;   // a pointer to an integer

// normal initialization:
x = 0;

// silly, but illustrative:
p = &x;   // & means “address of”
*p = 0;   // * means “memory at address”

• & and * are inverses of one another
• prefix vs infix operators
• x occupies 4 bytes in memory; p occupies 8
Arrays

• Contiguous block of memory
• Pointer to start, then indexed by element size
  • Indices start at zero

• \texttt{ary[k]} is the same as \texttt{* (ary+k)}
  • Location of \texttt{ary+k} depends on the type of array elements
Arrays and Pointers Combined

```c
int *p[47];
```

• Array of pointers ... or ... pointer to an array??

• It’s an array of 47 pointers
  • `p[3]` is the fourth pointer in the array `p`
  • `p[3]` is the base of an array
  • `p[3][6]` is the integer at position 6 in the array `p[3]`
What is printed?

```
int a[100];
int *p[47];
p[3] = a+12;
for (int i = 0; i < 100; i++){
    a[i] = i;
}
printf("%d\n", p[3][4]);
```
Structs

- Heterogeneous records, like objects
- Typical linked list declaration:

```c
typedef struct cell {
    int value;
    struct cell *next;
} cell_t;
```

- Usage:

```c
cell_t c;
c.value = 42;
c.next = NULL;
```

- Usage with pointers:

```c
cell_t *p;
p->value = 42;
p->next = NULL;
```

How many bytes are allocated for `c`? for `p`?

`p->next` is an abbreviation for `(*p).next`
Compilation

• gcc –o hello hello.c

```
#include <stdio.h>

int main(int argc, char ** argv) {
    printf("Hello world!\n");
    return 0;
}
```

```
... int printf(const char * restrict, ...
  __attribute__((__format__ (__printf__, 1, 2)));
... int main(int argc, char ** argv) {
    printf("Hello world!\n");
    return 0;
}
```
Running a Program

• ./hello
A Computer System

CPU

Register file
ALU

System bus
Main memory

Memory bus

Bus interface

I/O bridge

I/O bus

Expansion slots for other devices such as network adapters

USB controller

Graphics adapter

Disk controller

Mouse Keyboard

Display

Disk

hello executable stored on disk
LOGISTICS
Course staff

Prof. Eleanor Birrell
Edmunds 221

Research in security and privacy
OH: T 1:30-3:30pm, W 7-9pm

Jenna Brandt  Joe Brennan  Gabriel da Motta  Adam Lininger-White  Douglas Webster
The Course in a Nutshell

• Textbook

• Classes
  • Monday and Wednesday, 1:15-2:30 or 2:45-4pm in Edmunds 101

• Labs
  • Mondays 7-8:15 in Edmunds 229/219
  • **Start Monday!** Be sure to have an account and password

| Mentor Session Schedule (Edmunds 227) |
|-------------------------------|----------------|---------------|-------------|-------------|-------------|-------------|-------------|
| Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
| LAB   | 7-9pm   | 7-9pm     | 7-9pm     | 2-4pm   | 2-4pm     | 2-4pm     |
Grading

• Assignments
  • Introduced during labs, Due Fridays at 5pm
  • Tremendous fun, work in pairs
  • 45% of the grade
  • Seven late days

• Midterm exam
  • March 11
  • 20% of the grade each

• Final exam
  • Thursday, May 7 or Tuesday, May 12 or Friday, May 15 (2-5pm)
  • 30% of the grade

• Participation
  • 5% of the grade
Course website

https://www.cs.pomona.edu/classes/cs105

- All information is on the course website
- All course materials get posted on the course website
- Links from the course page:
  - Piazza, for questions and discussion
  - Lab assistants and mentors, schedule
  - Submission site
  - Grading site