How to make an A in this class

- Do the homework
  - Some homework will be taken up, some not
    - Sometimes it will be graded
    - Sometimes you get full credit for turning it in (i.e., we may be too lazy to grade it)
- Ask questions
  - Or get someone else to
  - Almost always, if you don't understand something, many others in the class don't either. Don't let the instructor (me) get away with this.
  - Talk to other class members! Often they can help.
- Go over your exams carefully after they're returned
  - Possible grading errors. Don't let me get away with this!
  - Lazy instructors (me) often reuse questions on finals
- More the bad news:
  - Come to class
  - Read the book

Class Overview

- Class web site: www.cs.odu.edu/~cmo
  - Syllabus, tentative schedule, etc.
  - Old exams, Old assignments, more.
  - Copies of slides
- Will only cover parts of text
- Programming is part of the class
  - C++
    - Some people think programming is fun, some people hate it!
    - In this class, you found out which kind you are (if you don't already know).
First Quiz

Send it to me e-mail so I will have your e-mail address
Answer questions on next slide; your answers only
Grade the correctness of your answers, include the number of correct & the number of incorrect answers
Due before class, Jan. 16

Background questions

1. Do you own a computer? If, so which OS?
   a. Some version of Windows
   b. Linux
   c. Mac (some version of X)
   d. Palm OS
2. How do you connect to the internet
   a. Work
   b. ODU
   c. Home—modem
   d. Home—isdn
   e. Home—cable
   f. Home—other
3. Major?
4. If you are employed, how many hr/week?

Why study computers?

They're everywhere, only more so
How many computers in this room?
When I came to ODU, the university had two, one for administrative use, one for academic use
You may be able to use some of what we do in this class in your profession
New experimentation methodology is computation
Many basic sciences have significant computation component
Knowledgeable citizens need to understand what computers can do to us!!
Is electronic voting a good idea?
Homework 1

- Check out the class web site
- Check out the text web site
- Read text, chapter 1 by Monday

What’s Google Earth?

- Over the weekend, I was checking to see if it is available for Macs yet
- See very, very long discussion between Mac users and Windows users.
  - Each group knew for certain that people in the other group were:
    - Idiots; they wouldn’t know a good computer if it bit them!
    - All good computer stuff was invented by Apple
    - All good computer stuff was invented by Microsoft
- Who’s right?
- Don’t forget: what’s Google Earth?
Chapter 1: Data Storage

- 1.1 Bits and Their Storage
- 1.2 Main Memory
- 1.3 Mass Storage
- 1.4 Representing Information as Bit Patterns
- 1.5 The Binary System
- 1.6 Storing Integers

Chapter 1: Data Storage (continued)

- 1.7 Storing Fractions
- 1.8 Data Compression
- 1.9 Communications Errors

Short history – and “bad” names

- First important applications of these machines were for doing complex, messy calculations
- So they were called “computers”
- They are really “pattern manipulators”
- They add and multiply by having wires that start with patterns and produce other patterns
  - Input: patterns representing 2 numbers
  - Output: pattern representing their sum (or product)
Bits and their meaning

- **Bit** = Binary Digit = a symbol whose meaning depends on the application at hand.
- Some possible meanings for a single bit:
  - Numeric value (1 or 0)
  - Boolean value (true or false)
  - Voltage (high or low)
  - Light or no light
  - Reflection or no reflection
  - etc., etc.

  - Somebody will come up with some new gadget that stores bits faster, cheaper, more reliably
  - (unrelated): saw an article yesterday that asserted that burned CDs and DVDs have a life of 2 to 5 years!

Bit patterns

- All data stored in a computer are represented by patterns of bits:
  - Numbers
  - Text characters
  - Images
  - Sound
  - Anything else...

Boolean operations

- **Boolean operation** = any operation that manipulates one or more true/false values
  - Can be used to operate on bits
- Specific operations:
  - AND
  - OR
  - XOR
  - NOT
Figure 1.1 The Boolean operations AND, OR, and XOR (exclusive or)

<table>
<thead>
<tr>
<th>Operation</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>OR</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XOR</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Gates

- **Gates** = devices that produce the outputs of Boolean operations when given the operations’ input patterns
  - Often implemented as electronic circuits
  - Provide the building blocks from which computers are constructed

Figure 1.2 A pictorial representation of AND, OR, XOR, and NOT gates as well as their input and output values
Flip-flops

- **Flip-flop** = a circuit built from gates that can store one bit of data.
- Has an input line which sets its stored value to 1
- Has an input line which sets its stored value to 0
- While both input lines are 0, the most recently stored value is preserved

**Figure 1.3** A simple flip-flop circuit

![Flip-flop circuit diagram]

**Figure 1.4** Setting the output of a flip-flop to 1

- 1 is placed on the upper input.
**Figure 1.4** Setting the output of a flip-flop to 1 (cont’d)

b. This causes the output of the OR gate to be 1 and, in turn, the output of the AND gate to be 1.

![Diagram showing the logic of setting output to 1](image)

**Figure 1.4** Setting the output of a flip-flop to 1 (cont’d)

c. The 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.

![Diagram showing further logic](image)

**Figure 1.5** Another way of constructing a flip-flop

![Diagram showing another flip-flop construction](image)
Other storage techniques

- Dynamic memory – must be replenished periodically – Example: capacitors
- Volatile memory – holds its value until the power is turned off – Example: flip-flops
- Non-volatile memory – holds its value after the power is off – Example: magnetic storage
- Read-only memory (ROM) – never changes – Examples: flash memory, compact disks

Hexadecimal notation

- **Hexadecimal notation** = a shorthand notation for streams of bits.
  - **Stream** = a long string of bits.
  - Long bit streams are difficult to make sense of.
  - The lengths of most bit streams used in a machine are multiples of four.
  - Hexadecimal notation is more compact.
    - Less error-prone to manually read, copy, or write

**Figure 1.6** The hexadecimal coding system

<table>
<thead>
<tr>
<th>Bit pattern</th>
<th>Hexadecimal representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>A</td>
</tr>
<tr>
<td>1011</td>
<td>B</td>
</tr>
<tr>
<td>1100</td>
<td>C</td>
</tr>
<tr>
<td>1101</td>
<td>D</td>
</tr>
<tr>
<td>1110</td>
<td>E</td>
</tr>
<tr>
<td>1111</td>
<td>F</td>
</tr>
</tbody>
</table>
Main memory: cells

- **Cells** = manageable units (typically 8 bits) into which a computer’s main memory is arranged.
- **Byte** = a string of 8 bits.
- **High-order end** = the left end of the conceptual row in which the contents of a cell are laid out.
- **Low-order end** = the right end of the conceptual row in which the contents of a cell are laid out.
- **Least significant bit** = the last bit at the low-order end.

**Figure 1.7** The organization of a byte-size memory cell

<table>
<thead>
<tr>
<th>High-order end</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>Low-order end</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Most significant bit</td>
<td></td>
<td></td>
<td>Least significant bit</td>
<td></td>
</tr>
</tbody>
</table>

Main memory addresses

- **Address** = a “name” to uniquely identify one cell in the computer’s main memory
- The names for cells in a computer are consecutive numbers, usually starting at zero
- Cells have an order: “previous cell” and “next cell” have reasonable meanings
- **Random Access Memory** = memory where any cell can be accessed independently
Figure 1.8 Memory cells arranged by address

Measuring memory capacity: Not quite like the metric system

- “Kilo-” normally means 1,000; but Kilobyte = $2^{10} = 1024$
- “Mega-” normally means 1,000,000; Megabyte = $2^{20} = 1,048,576$
- “Giga-” normally means 1,000,000,000; Gigabyte = $2^{30} = 1,073,741,824$

Mass Storage Systems

- Non-volatile; data remains when computer is off
- Usually much bigger than main memory
- Usually rotating disks
  - Hard disk, floppy disk, CD-ROM
  - Much slower than main memory
    - Data access must wait for seek time (head positioning)
    - Data access must wait for rotational latency.
Figure 1.9  A disk storage system

Figure 1.10  CD storage format

Figure 1.11  A magnetic tape storage mechanism
Files

- **File** = the unit of data stored on a mass storage system.
- **Logical record** and **Field** = natural groups of data within a file.
- **Physical record** = a block of data conforming to the physical characteristics of the storage device.
- **Buffer** = main memory area sometimes set aside for assembling logical records or fields of a file.

**Figure 1.12** Logical records versus physical records on a disk

**Figure 1.13** The message “Hello.” in ASCII
Representing text

- Each printable character (letter, punctuation, etc.) is assigned a unique bit pattern.
  - ASCII = 7-bit values for most symbols used in written English text
  - Unicode = 16-bit values for most symbols used in most world languages today
  - ISO proposed standard = 32-bit values

---

Representing numeric values

- Binary notation – uses bits to represent a number in base two
- Limitations of computer representations of numeric values
  - Overflow – happens when a number is too big to be represented
  - Truncation – happens when the correct pattern to represent a number is too big

---

**Figure 1.14** A possible sound wave represented by the sequence 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0
Figure 1.15 The base ten and binary systems

Figure 1.16 Decoding the binary representation 100101

Figure 1.17 An algorithm for finding the binary representation of a positive integer

**Step 1.** Divide the value by two and record the remainder.

**Step 2.** As long as the quotient obtained is not zero, continue to divide the newest quotient by two and record the remainder.

**Step 3.** Now that a quotient of zero has been obtained, the binary representation of the original value consists of the remainders listed from right to left in the order they were recorded.
Figure 1.18 Applying the algorithm in Figure 1.15 to obtain the binary representation of thirteen

Figure 1.19 The binary addition facts

|   | + |   | + |   |   |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |

Figure 1.20 Decoding the binary representation 101.101
Back to patterns

- Kinda’ arbitrary choices
- But when computers were first being designed and built, they were very expensive
- So patterns were chosen to make them a little less expensive
- Today, many of the choices seem poor, but we’re stuck with them
  - They work OK.
  - No one wants to spend the effort (or money) to make better choices

Representing Integers

- Unsigned integers can be represented in base two
  - Why not base 10? Ans. Because it’s a little cheaper to build the hardware for base 2
    - Due to basic physics
  - Good for torturing students
- Signed integers = numbers that can be positive or negative
  - Two’s complement notation = the most popular representation
  - Excess notation = another less popular representation

Figure 1.21 Two’s complement notation systems
Figure 1.22  Coding the value “-6” in two’s complement notation using four bits

Figure 1.23  Addition problems converted to two’s complement notation

<table>
<thead>
<tr>
<th>Problem in base ten</th>
<th>Problem in two’s complement</th>
<th>Answer in base ten</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 + 2</td>
<td>0011 + 0010 = 0101</td>
<td>5</td>
</tr>
<tr>
<td>-3 + -2</td>
<td>1111 + 1110 = 1011</td>
<td>-5</td>
</tr>
<tr>
<td>7 + -5</td>
<td>0111 + 1011 = 0000</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 1.24  An excess eight conversion table

<table>
<thead>
<tr>
<th>Bit pattern</th>
<th>Value represented</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>7</td>
</tr>
<tr>
<td>1110</td>
<td>6</td>
</tr>
<tr>
<td>1101</td>
<td>5</td>
</tr>
<tr>
<td>1100</td>
<td>4</td>
</tr>
<tr>
<td>1011</td>
<td>3</td>
</tr>
<tr>
<td>1010</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>0111</td>
<td>-1</td>
</tr>
<tr>
<td>0110</td>
<td>-2</td>
</tr>
<tr>
<td>0101</td>
<td>-3</td>
</tr>
<tr>
<td>0100</td>
<td>-4</td>
</tr>
<tr>
<td>0011</td>
<td>-5</td>
</tr>
<tr>
<td>0010</td>
<td>-6</td>
</tr>
<tr>
<td>0001</td>
<td>-7</td>
</tr>
<tr>
<td>0000</td>
<td>-8</td>
</tr>
</tbody>
</table>
**Figure 1.25** An excess notation system using bit patterns of length three

<table>
<thead>
<tr>
<th>Bit pattern</th>
<th>Value represented</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>3</td>
</tr>
<tr>
<td>110</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>-1</td>
</tr>
<tr>
<td>010</td>
<td>-2</td>
</tr>
<tr>
<td>001</td>
<td>-3</td>
</tr>
<tr>
<td>000</td>
<td>-4</td>
</tr>
</tbody>
</table>

**Figure 1.26** Floating-point notation components

- Bit positions
- Mantissa
- Exponent
- Sign bit

**Figure 1.27** Coding the value $2 \times \frac{5}{8}$

- Original representation
- Base two representation
- Raw bit pattern
- Lost bit
**Figure 1.28** Decompressing xyxyxy (5, 4, x)

- **a.** Count backward 5 symbols.
  
  \[
  \text{xyxyxy}
  \]

- **b.** Identify the four-bit segment to be appended to the end of the string.
  
  \[
  \text{xyxy}
  \]

- **c.** Copy the four-bit segment onto the end of the message.
  
  \[
  \text{xyxyxyxyxy}
  \]

- **d.** Add the symbol identified in the triple to the end of the message.
  
  \[
  \text{xyxyxyxyxyx}
  \]

---

**Figure 1.29** ASCII codes for “A” and “F” adjusted for odd parity

- Parity bit: ASCII A containing an even number of 1s
  
  \[
  \begin{array}{cccc}
  0 & 1 & 0 & 0
  \end{array}
  \]

  Total pattern has an even number of 1s

- Parity bit: ASCII F containing an odd number of 1s
  
  \[
  \begin{array}{cccc}
  0 & 1 & 0 & 0
  \end{array}
  \]

  Total pattern has an odd number of 1s

---

**Figure 1.30** An error-correcting code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>000000</td>
</tr>
<tr>
<td>B</td>
<td>001111</td>
</tr>
<tr>
<td>C</td>
<td>010011</td>
</tr>
<tr>
<td>D</td>
<td>011100</td>
</tr>
<tr>
<td>E</td>
<td>100110</td>
</tr>
<tr>
<td>F</td>
<td>101001</td>
</tr>
<tr>
<td>G</td>
<td>110101</td>
</tr>
<tr>
<td>H</td>
<td>111010</td>
</tr>
</tbody>
</table>
Figure 1.31  Decoding the pattern 010100 using the code in Figure 1.30

<table>
<thead>
<tr>
<th>Character</th>
<th>Distance between the received pattern and the character being considered</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>H</td>
<td>4</td>
</tr>
</tbody>
</table>

Smallest distance

Homework (list also on web)

- Due in class Jan. 25.
  - Pg. 24: 1, 4, 5, 6, 7; Pg. 34: 2, 3, 4
- Due in class Jan. 30.
  - Pg. 40: 1, 5, 6, 10
- Due Feb. 1
  - Pg. 45: 3, 4, 5; Pg. 51: 1, 2, 5
- Due Feb. 6
  - Pg 56: 1, 2, 3; Pg. 65: 1, 5

Diversion: Lab 1

- Alice chat bot
  - What's a "bot"?
  - Assignment: find Alice on the web and have a conversation with her
    - Mail me your impressions after talking with her
    - See class web site for questions to answer.
    - Figure out if she remembers earlier parts of a conversation.
    - Check out the Chatterbox Challenge. Who won last year?
  - Computer science frustration for teachers
    - Stuff in movies often much better than what is possible
    - Remember 2001: A Space Odyssey (1968)?
      - HAL figured out how to read lips
      - He had to WANT to do this
    - Alice can’t read lips. Alice really does not exhibition volition.
- Due in recitation Jan. 30. Handholding provided.