What You Have Done So Far

Part I
- A title for the website page
- The bogus logo inserted somewhere near the title
- The unmodified image
- Paragraph 1: source of the image and your argument as to why you can alter it
- Paragraph 2: true context of the image (what it really represents, unmodified)
- Paragraph 3: “fictional” context of the image. Explain how you plan to alter it and use it so support the “storyline” of your website

To do for Part II:
Putting it all together and User Testing
- Modify the image
- Include text to support your modified image: create your “story”
- Use other formatting elements to make your site “look” credible: font size and color, graphics, background color, etc.
- Add an additional link to your email address and a link to your disclaimer page
- Create a copy of the page above (user testing page)
  - Remove the bogus logo
  - Show it to two friends, have them evaluate it based on the Assignment 3 criteria
- Create a second page (a disclaimer page) in which you:
  - Provide a disclaimer about quality of information presented on the main page
  - Reflect on the ethical issues surrounding accuracy, authority, credibility, etc of information on the Web
  - Write up the results of your user testing
  - Add a link to your main Misinformation page and to your user testing page

Web Site of Misinformation:
A Plausible Title to match your content

Seattle, WA: In a press conference yesterday,

..................

Sources close to the ....

..................

If you see ....

..................

Contact Info (email) 
Disclaimer
Your reflections on the ethical issues surrounding accuracy, credibility, authority, etc (all the criteria we discussed) when dealing with information on The Web: .................................................................

The results of your user testing: .................................

Sources close to the ....

If you see: ..................................................................

Contact Info (email):  

Disclaimer about the quality of any information found on this site.

Disclaimer Page

Seattle, WA: In a press conference yesterday, Seattle Police Chief: 

Sources close to the ....

If you see: .........................................................

User Testing

- Show this web site to two individuals
- Using the criteria from Assignment 2, have each one evaluate your site
- Write a report of your results and add it to your disclaimer page

Computer Basics

Regardless of how much computers have changed over the last 50 years (think of our first lecture), they are still characterized by the same basic principles.
Abstractly, A Computer Is…

- Computers process information by deterministically following instructions, called executing instructions.
- Unlike humans, computers follow instructions exactly:
  - Computers have no imagination or creativity
  - Computers have no intuition
  - Computers are literal: they have no sense of irony, subtlety, proportion...
  - Computers don't joke, they're not vindictive or cruel
  - Computers are not purposeful (they don't have their own changing agenda!)

…Computers execute instructions. Nothing more.

Remember this when you feel like screaming at your monitor…!

If a computer has any useful characteristics, it's because someone has programmed it—in other words, given it the instructions— to behave usefully.

Interpreting the Instructions

- To perform instructions, a computer's hardware implement a process called the fetch/execute cycle.

  Fetch/Execute Cycle:
  - Instruction Fetch (IF)
  - Instruction Decode (ID)
  - Data Fetch (DF)
  - Instruction Execution (EX)
  - Return Result (RR)

- The F/E Cycle is an unending process.

Anatomy Of A Computer

- A computer is essentially made up of 5 components:
  - Arithmetic/Logic Unit (ALU) – the part doing computations
  - Control – the part that follows the Fetch/Execute Cycle of the program and tells the ALU what to compute
  - Memory – where data, programs are kept while computing
  - Input – ports to peripheral devices that allow/bring data in
  - Output – ports to peripheral devices that allow/send data out
A simple example

- Suppose you have
  - A set of envelopes, each with a card in it
  - A number or an instruction can be written on each card
  - There are three kinds of instructions:
      - ADD env# env# env#
      - ASK env#
      - SAY env#
      - NEXT env#

- Envelope 1: ASK 15
- Envelope 2: ASK 13
- Envelope 3: ADD 15 13 10
- Envelope 4: SAY 10
- Envelope 5: NEXT 1
- Envelope 10: ??
- Envelope 13: ??
- Envelope 15: ??

Memory

- The memory component is passive, storing programs and data

<table>
<thead>
<tr>
<th>address</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>M</td>
</tr>
<tr>
<td>1</td>
<td>J</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

- Memory is like a series of “byte-size” boxes – each has an address and some contents called its value
- Memory is called RAM for “random access memory” because the control can access any random location in the memory
- RAM is volatile memory – it disappears when the power does
There always needs to be something in

Control: Control Rules!

- The control follows through the instructions, executing them by telling other parts what to do.
- The instructions come from the program stored in the memory.

The instructions are in the end expressed in a machine language, which the control can understand. A typical machine instruction is:

```
add 124, 1005, 6215
```
Which means "add the number in memory location 124 to the number in memory location 1005 and put the result in memory location 6215."

Just to be clear…

- The instruction add 124, 1005, 6215 does not add 124, 1005 and 6215 together. We can do that in our heads or with a calculator.
- It simply adds whatever has been stored at those memory locations.
- Different numbers in those locations produce different results:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>124</td>
<td>23</td>
</tr>
<tr>
<td>124</td>
<td>25</td>
</tr>
<tr>
<td>124</td>
<td>35</td>
</tr>
<tr>
<td>699</td>
<td>697</td>
</tr>
</tbody>
</table>

Following Instructions

- The control maintains the correct place in the program by using a program counter, or PC. A better name might be "instruction pointer."
- The control also prepares for data-fetches from and result returns to the memory.

The Fetch/Execute Process

- Just before the Instruction Fetch….
**Instruction Fetch**

- Get instruction at the memory location PC

**Instruction Decode**

- Analyze instruction and set up later steps
  - Specify the ALU operation (add)
  - Specify addresses to fetch (124, 1005) and to store (6215)

**Data Fetch**

- Move values stored at fetch-addresses to ALU for processing

**Execute**

- The operation of the instruction (add) is performed
The result is returned to memory to the address specified in the instruction.

The result is returned to memory to the address specified in the instruction.

The PC's PC

After the instruction has been fetched and executed, the next instruction in sequence is fetched at PC +1.

This scheme should cause the computer to run through memory executing all instructions once and then "fall off the end of memory".

Computers have machine instructions to branch and jump, i.e. go to some instruction other than the next.

Jump and Branch change the PC after increment.

Programs generally repeat many instructions.

What's in a Number?

A memory location can store one byte of information, enough for a keyboard character.

A "normal" whole number (integer) uses 4 bytes.

A machine instruction uses 4 bytes.

Units of memory size are ...

- KB, kilobyte, 1024 bytes ...
- MB, megabyte, 1,048,576 bytes ...
- GB, gigabyte, 1,073,741,824 bytes ...
- TB, terabyte, 1,099,511,627,776 bytes ...

Free Memory!

Why do computers use such weird amounts to indicate 1000, 1 Million, etc? These numbers are powers of 2:

- \(2^{10} = 1,024\) call it a thousand
- \(2^{20} = 1,048,576\) call it a million
- \(2^{30} = 1,073,741,824\) call it a billion
- \(2^{40} = 1,099,511,627,776\) call it a trillion

When you buy a megabyte of member, it's as if you get \(48,576\) bytes for free!
Computational Time: The Pace of Computing

- Computers use electronic clocks to pace the Fetch/Execute Cycle.
- If the computer goes around the F/E cycle once per tick, then the rate of the clock ("ticks/second") gives the number of instructions executed per second.
- Hertz measures "cycles per second".
- 500MHz, specifies "500 million cycles per second".
- The reality is that the "one instruction per clock cycle" rule is only an approximation... modern computers are MUCH more complicated.

Summary

- Computers deterministically execute instructions to process information.
- Computers have five parts: ALU, Control, Memory, Input and Output.
- The control implements a process called the Fetch/Execute Cycle.
- The F/E cycles is a fundamental method of performing operations EXACTLY the same way specified, every time. This idea is used in many places in computation.

For Monday

- Assignment 2 is due in your Monday/Tuesday lab.
- Read Chapter 10 of FIT Course pack.
- Lab 7 is the Introduction to Visual Basic.
  - Read through the Lab.
  - Read the Chapters suggested there.