File I/O

CSCI 201
Principles of Software Development

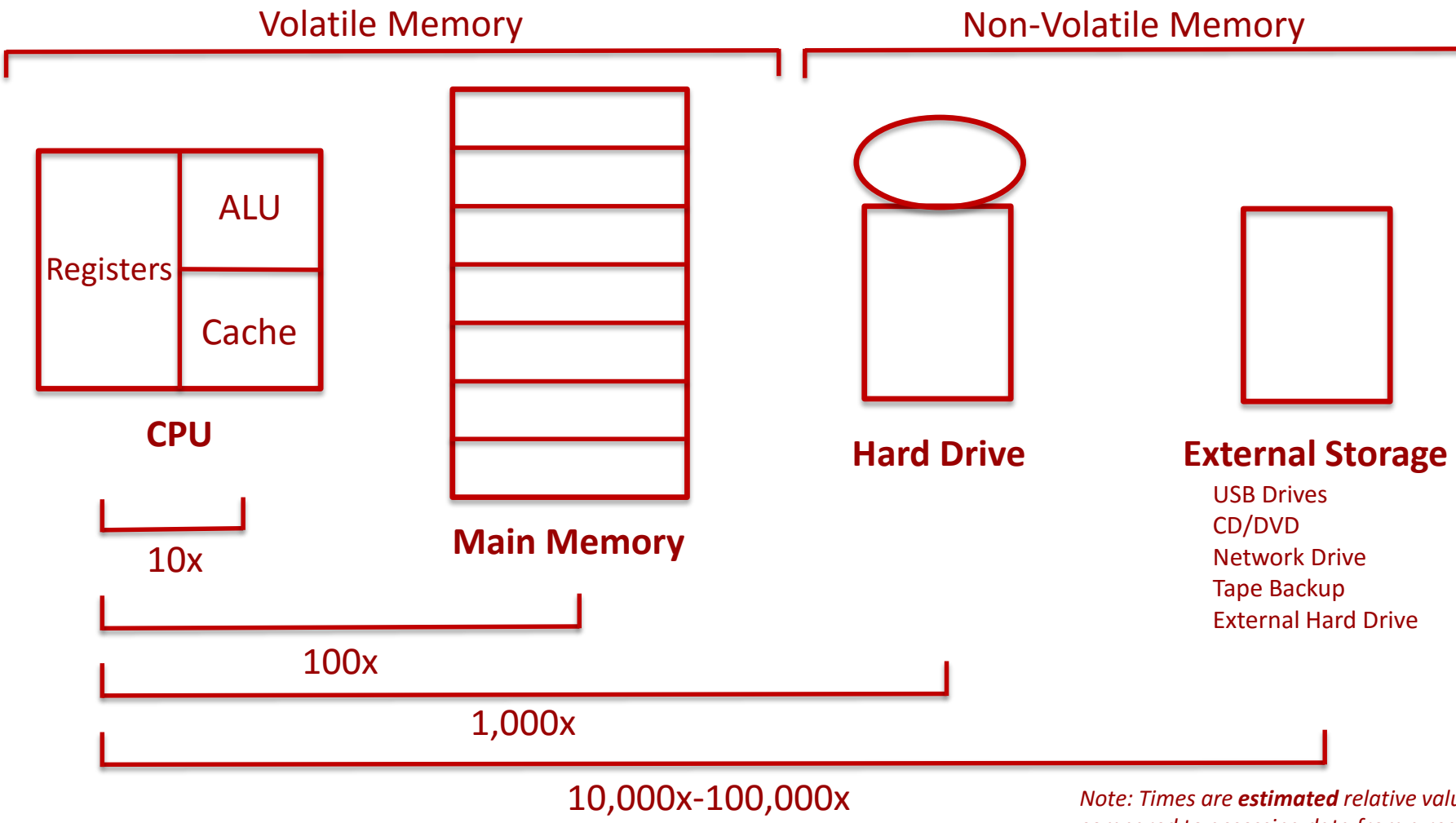
Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu

USC Viterbi
School of Engineering

Outline

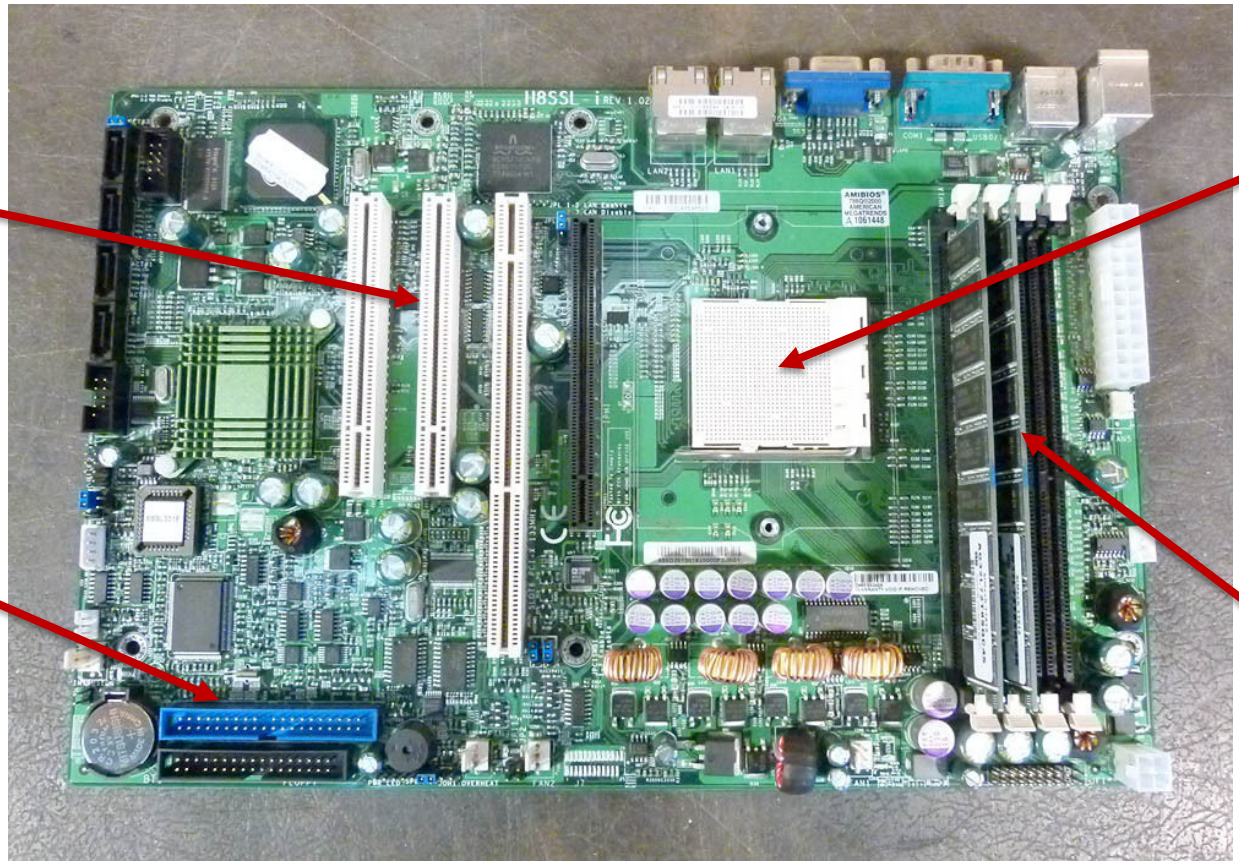
- File I/O
- Program

Storage Speeds



*Note: Times are **estimated** relative values compared to accessing data from a register*

Actual Computer Diagram



File Input

```
1  import java.io.BufferedReader;
2  import java.io.FileReader;
3  import java.io.FileNotFoundException;
4  import java.io.IOException;
5  public class Test {
6   public static void main(String [] args) {
7    FileReader fr;
8    BufferedReader br;
9    try {
10    fr = new FileReader("Test.java");
11    br = new BufferedReader(fr);
12    String line = br.readLine();
13    while (line != null) {
14     System.out.println(line);
15     line = br.readLine();
16    }
17   } catch (FileNotFoundException fnfe) {
18     System.out.println(fnfe.getMessage());
19   } catch (IOException ioe) {
20     System.out.println(ioe.getMessage());
21   } finally {
22     if (br != null) {
23       try {
24         br.close();
25       } catch (IOException ioe) {
26         System.out.print(ioe.getMessage());
27       }
28     }
29     if (fr != null) {
30       try {
31         fr.close();
32       } catch (IOException ioe) {
33         System.out.print(ioe.getMessage());
34       }
35     }
36   } // ends finally
37  } // ends main
38 } // ends Test
```

Relative vs Absolute Filenames

- A relative filename does not contain the full path
  - Example
    - `test.txt`
    - `desktop/test.txt`
    - `../../text.txt`
  - A relative filename will be relative to the directory from which your program is being run
- An absolute filename contains the full path to the file
  - Example
    - `c:\users\jmiller\Desktop\text.txt`
    - `/usr/jmiller/desktop/text.txt`
  - The location of the program is irrelevant
  - If the program is run on a different computer, it will need the same absolute path if it is hard-coded

File Output

```
1  import java.io.FileWriter;
2  import java.io.PrintWriter;
3  import java.io.IOException;
4
5  public class Test {
6   public static void main(String [] args) {
7    FileWriter fw;
8    PrintWriter pw;
9    try {
10    fw = new FileWriter("Test.java");
11    pw = new PrintWriter(fw);
12    pw.println("First line");
13    pw.println("Second line");
14    pw.flush();
15   } catch (IOException ioe) {
16    System.out.println(ioe.getMessage());
17   } finally {
18     if (pw != null) {
19      try {
20       pw.close();
21      } catch (IOException ioe) {
22       System.out.println(ioe.getMessage());
23      }
24     }
25     if (fw != null) {
26      try {
27       fw.close();
28      } catch (IOException ioe) {
29       System.out.println(ioe.getMessage());
30      }
31    } // ends finally
32  } // ends main
33 } // ends Test
```

To improve speed for writing, the OS will buffer writes to disk.
If the buffer is not full, the data written into the stream has not been written to disk.
To flush the current contents of the buffer, we can call `flush()` on the stream.

Outline

- File I/O
- Program

Program

- Write a program that prompts the user to enter a number for how many copies of a file to make. It will then prompt the user for the input file and the number of output files based on the user input. Create a custom exception if the value entered was outside of the desired range. No exceptions should be able to be thrown.

```
c:\>java csci201.FileCopy
Enter the number of copies (between 1 and 10): 13
OutOfRangeException: 13 is not between 1 and 10
Enter the number of copies (between 1 and 10): -3
OutOfRangeException: -3 is not between 1 and 10
Enter the number of copies (between 1 and 10): 2
Enter the input file: input.txt
Enter output file #1: output1.txt
Enter output file #2: output2.txt
The file was copied.
c:\>
```