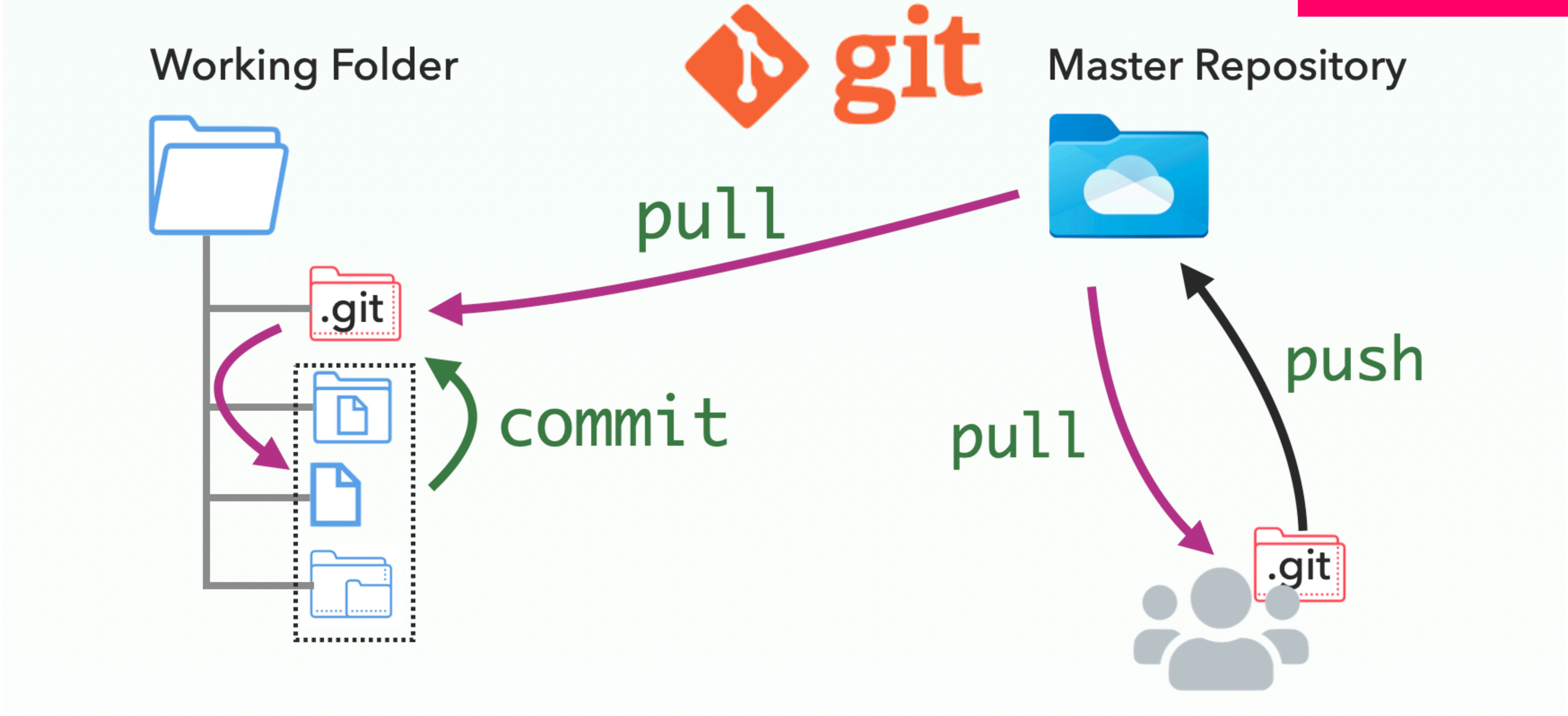
CS62 Lab7: Version Control

Basic SWE skills



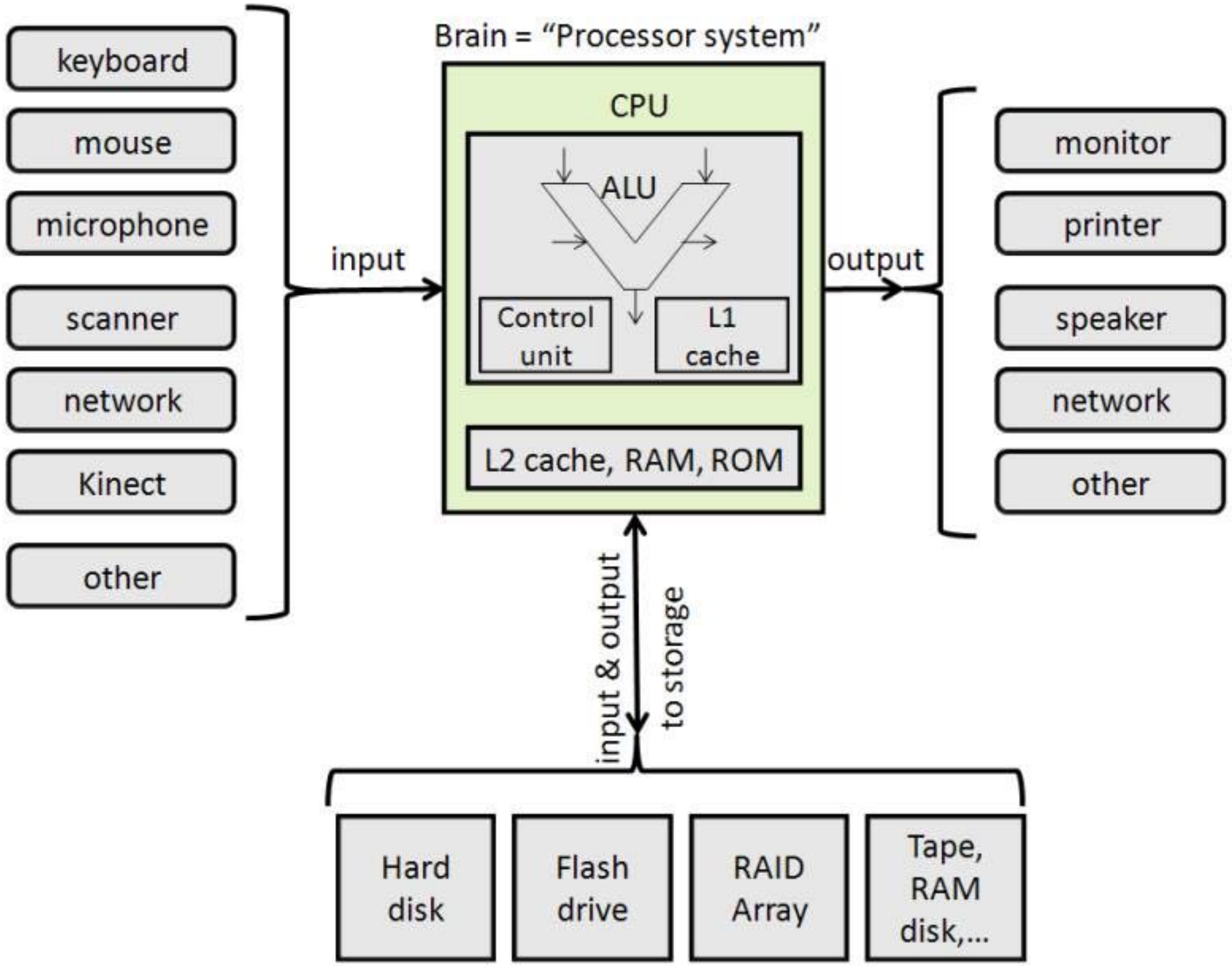
Agenda

- Please pass me your checkpoint regrades
- Quiz (please start reading version control lab if you’re done early)
- Short motivation behind HW6: On Disk Sort
- Lab

Announcements

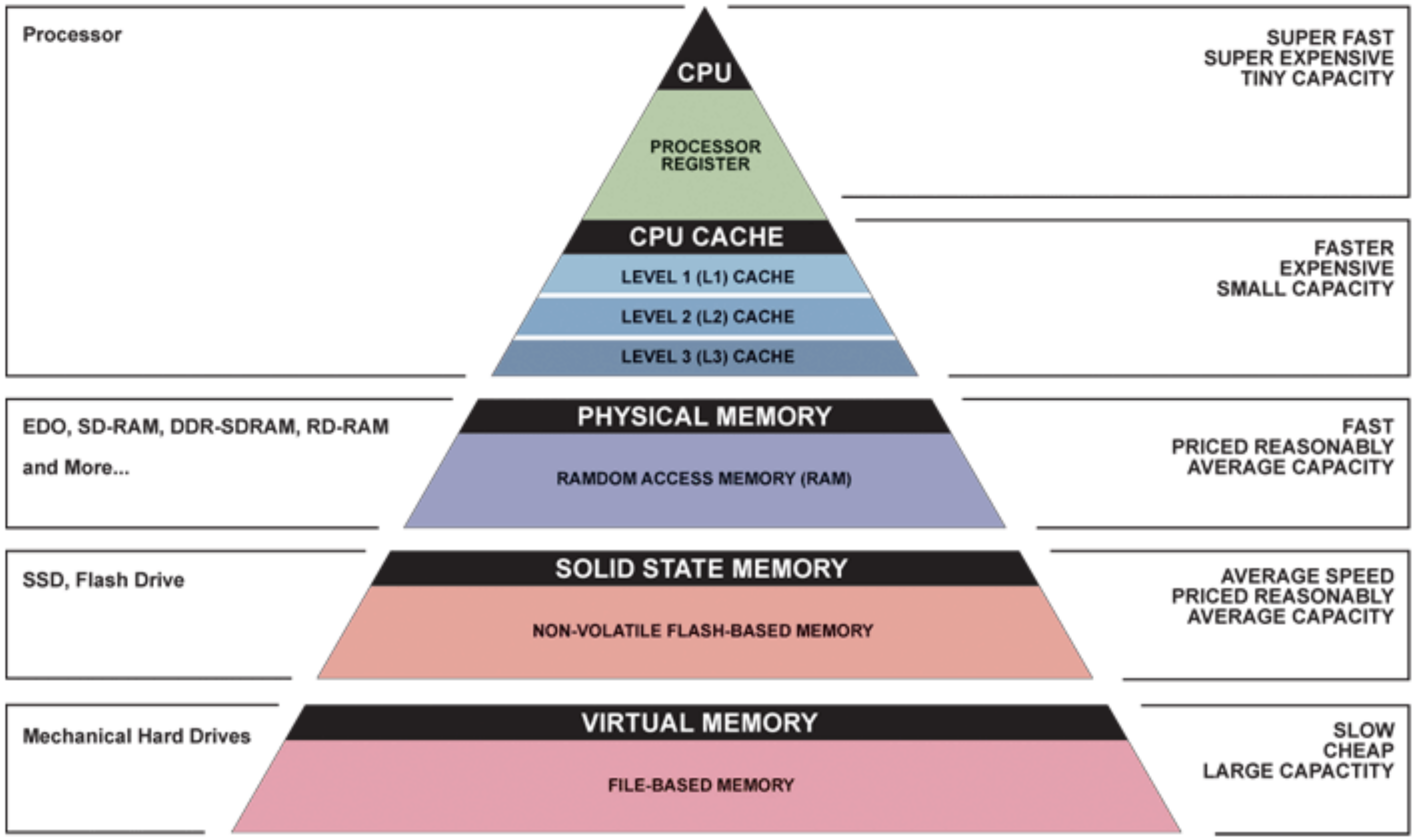
- Everyone gets a 24 hour extension for Compression Pt 2
- We don’t release the autograder tests, but one is definitely testing the table in the README file. Try making that test yourself and make sure you pass it! (E.g., initially there should be 9 nodes in your linked list.)

Basic Von Neumann computer architecture



http://cs.sru.edu/~mullins/cpsc100book/module03_internalHardware/module03-05_internalHardware.html

Memory Hierarchy (a simplified summarized story)



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

Registers

- Fastest, most expensive, tiny capacity.
- Run at same speed with CPU’s clock
  ~3GHz today (3 billion operations/sec).
- Typically, 16-32 of them per core.
- Can hold 32 or 64 bits based on architecture that correspond to data or memory locations.

Cache

- Faster, expensive, small capacity.
- A bit slower than CPU’s speed but faster than main memory.
- Typically, 2-3 levels (L1, L2, L3, etc.).
- 32-64 KB for L1, 128-512 KB for L2.

Main (physical) memory (RAM)

- Fast, reasonably priced, average capacity.
- Much slower than CPU but significantly faster than disk.
- 8-32 GB.
- All programs and data must fit in memory.
  - If not, virtual memory to the rescue while accessing the disk.

External (secondary or auxiliary) memory (disk)

- Slower speed, reasonably priced, large capacity.
- Most computers have now solid state drives (SSDs) which are faster (but typically more expensive) than mechanical hard disk drives (HDDs).
- Hundreds of GB to a few TB.

Assignment 6: On-disk merge sort

- All sorting algorithms we have seen assume that the data to be sorted can fit in main memory (RAM). In the era of big data, this is not always the case.
- For assignment 5, you will work on an external (on-disk) mergesort.
- Data are read from the disk in chunks of maximum size and are individually sorted using regular merge sort and stored back on disk in temporary “chunk" files.
- Temporary files are merged into an increasingly larger temporary file till the entire original data have been sorted and saved back on disk.
  - This is accomplished by iteratively merging the temporary file with a sorted temporary “chunk” file. Two buffered readers allow you to read a line/file at a time, compare them, and choose the “smallest” to be saved into the temporary file that contains all of merged data so far. Essentially, merge method of mergesort!

BufferedReader

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class ReadFile {


 public static void main(String[] args) {
  String fileName = "somePath/File.txt";

  try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {

   String strCurrentLine = br.readLine();

   while (strCurrentLine != null) {
        //do something
        strCurrentLine = br.readLine();
   }

  } catch (IOException e) {
   e.printStackTrace();
  }
 }
}
```