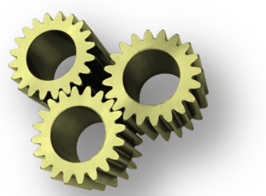
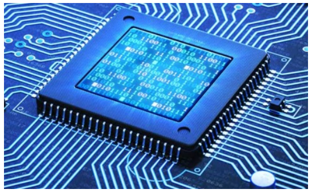
Algorithms and Architectures

Michael T. Goodrich
CS 165
Univ. of California, Irvine



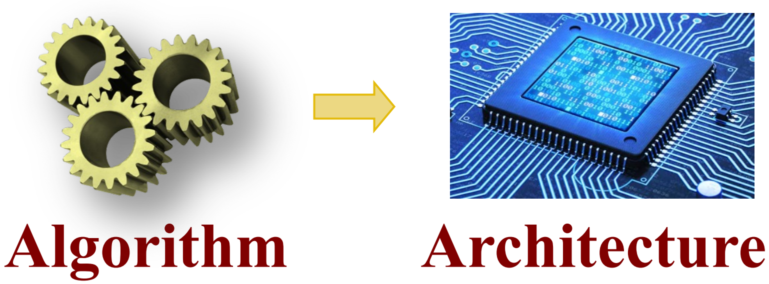
Algorithm



Architecture

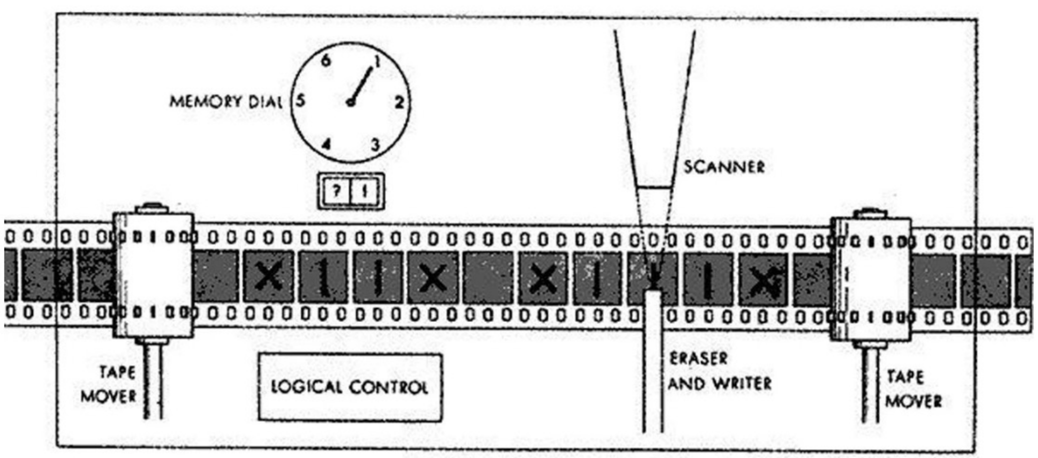
Computational Models

- We often skip this step, but an algorithm description requires a computational model.
- There are several computational models, which are based on various computational architectures.



Turing Machine

- Mathematical model of computation, 1936
- **Complexity-theoretic Church–Turing thesis:** Any polynomial-time computable function (for any computational model) has a polynomial-time algorithm on a Turing machine.



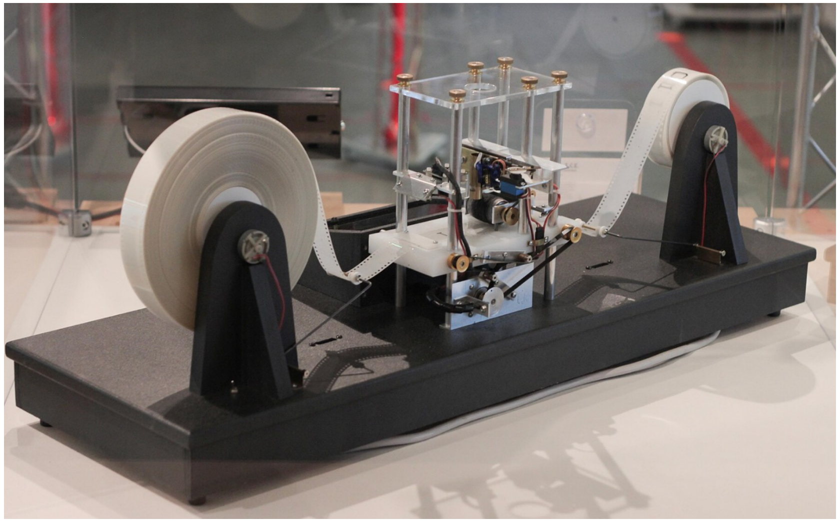
https://mediartinnovation.com/2014/05/26/alan-turing-turing-machine-1936/

Alan Turing

https://en.wikipedia.org/wiki/Alan_Turing

Turing Machine

- Not a real-world computer, but imitations exist.
- Any computational model/system that can simulate a Turing machine can compute any computable function: **Turing-complete**.

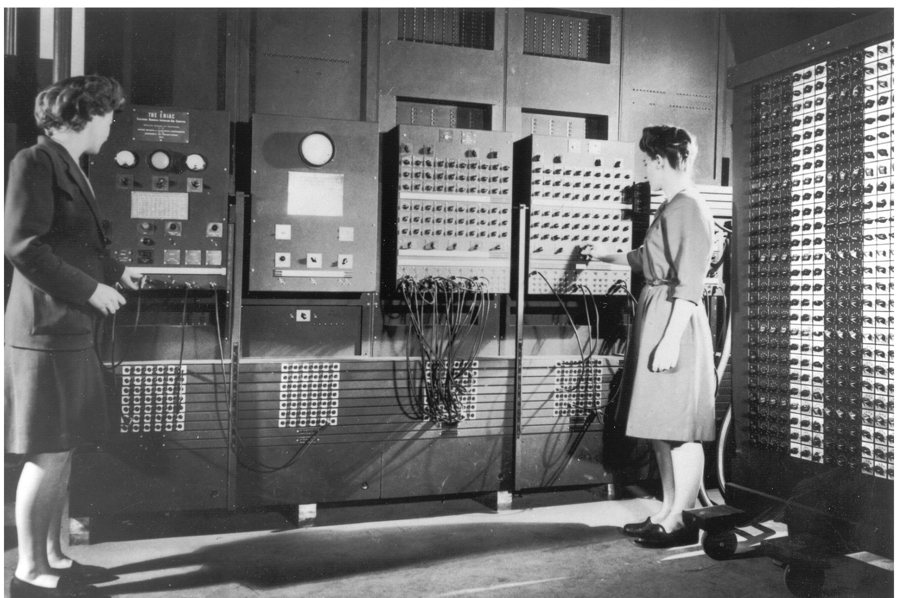


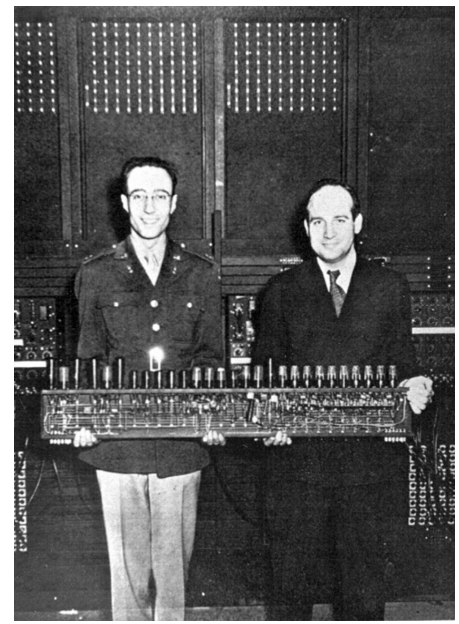
https://en.wikipedia.org/wiki/Turing_machine

not Alan Turing

ENIAC

- The first programmable, electronic, general-purpose digital computer, completed in 1945.
- It was Turing-complete



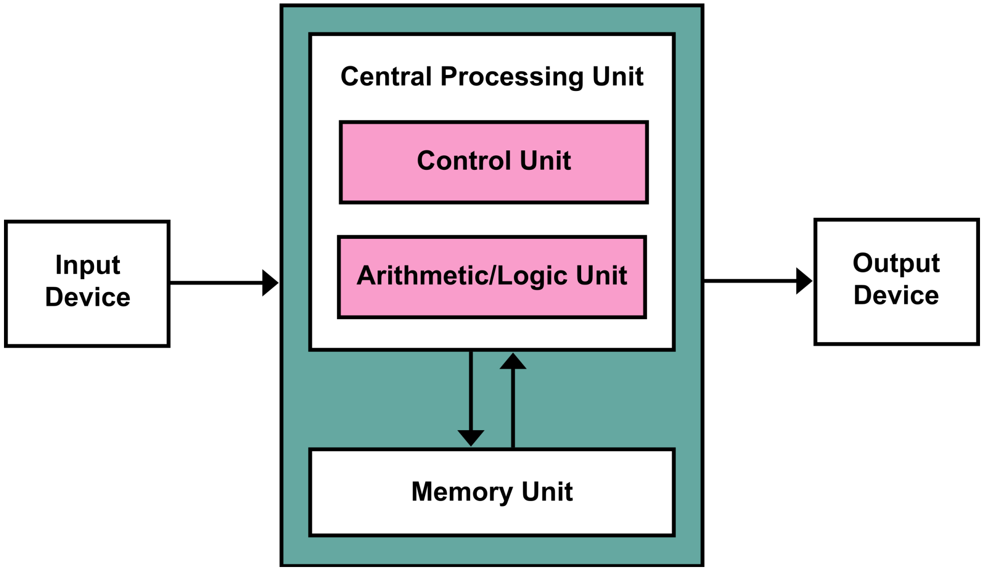


Presper Eckert and John Mauchly

https://en.wikipedia.org/wiki/ENIAC

The von Neumann Architecture

- ENIAC gave rise to a computational model called the von Neumann Architecture.



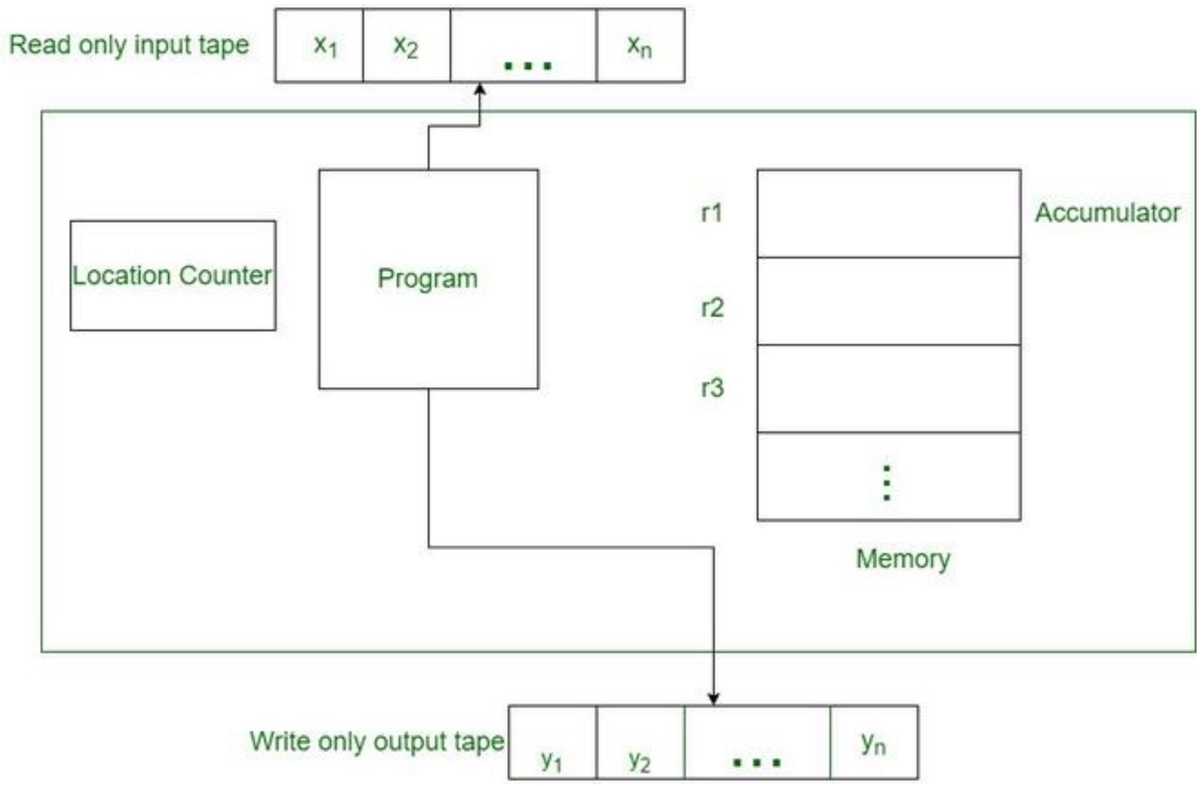
Memory stores both data and instructions

John von Neumann

https://en.wikipedia.org/wiki/Von_Neumann_architecture

Random Access Machine (RAM)

- A refinement of the von Neumann architecture.



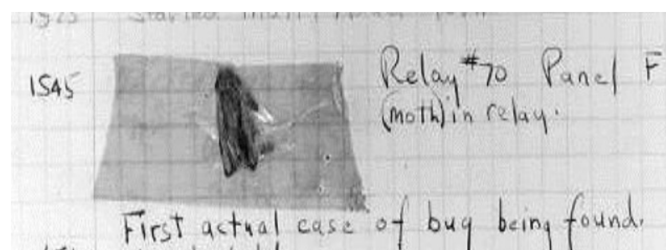
https://www.geeksforgeeks.org/what-is-random-access-machine/

Programming Languages

- Machine-independent languages written by humans and compiled into instructions for specific computer architectures.

Grace Hopper

The FLOW-MATIC programming language she created was later extended to create COBOL, a widely used high-level language for business applications.



RAM Programming Primitives

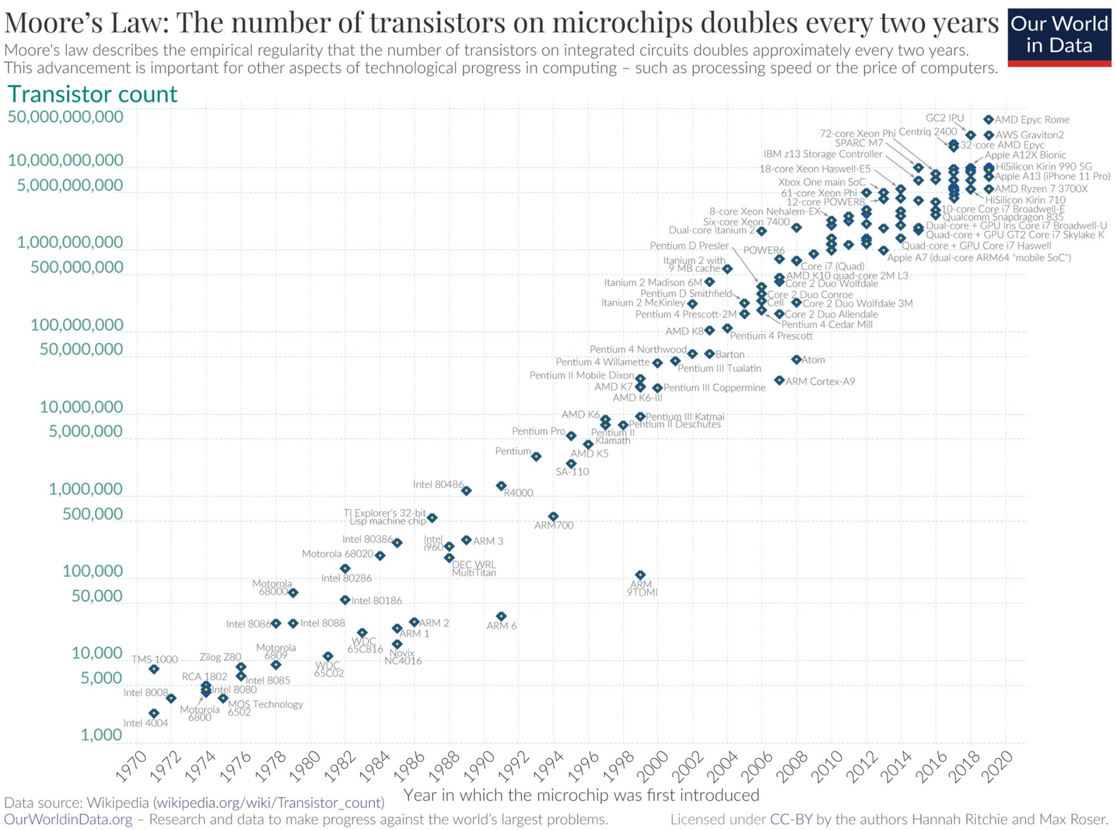
- Language primitives for the RAM model are based on high-level programming languages, which were defined for the von Neumann architecture.

Operations	Number of steps
Arithmetic operations: + - * /	1
Logical operations: AND, OR, NOT	1
Conditional: - Comparison: a < b - Conditional branching: if	1
Subroutine calls: call, return	1
Loops	Depends on the number of loop iterations and loop condition
Subprogram	Depends on the nature of the subprogram
Memory access: Read, Write	1

https://medium.com/@exploreintellect/what-is-the-ram-model-of-computation-a5e4a7ce22b4

Moore’s Law

- Moore’s “law” is the observation that the number of transistors in an integrated circuit doubles about every two years.

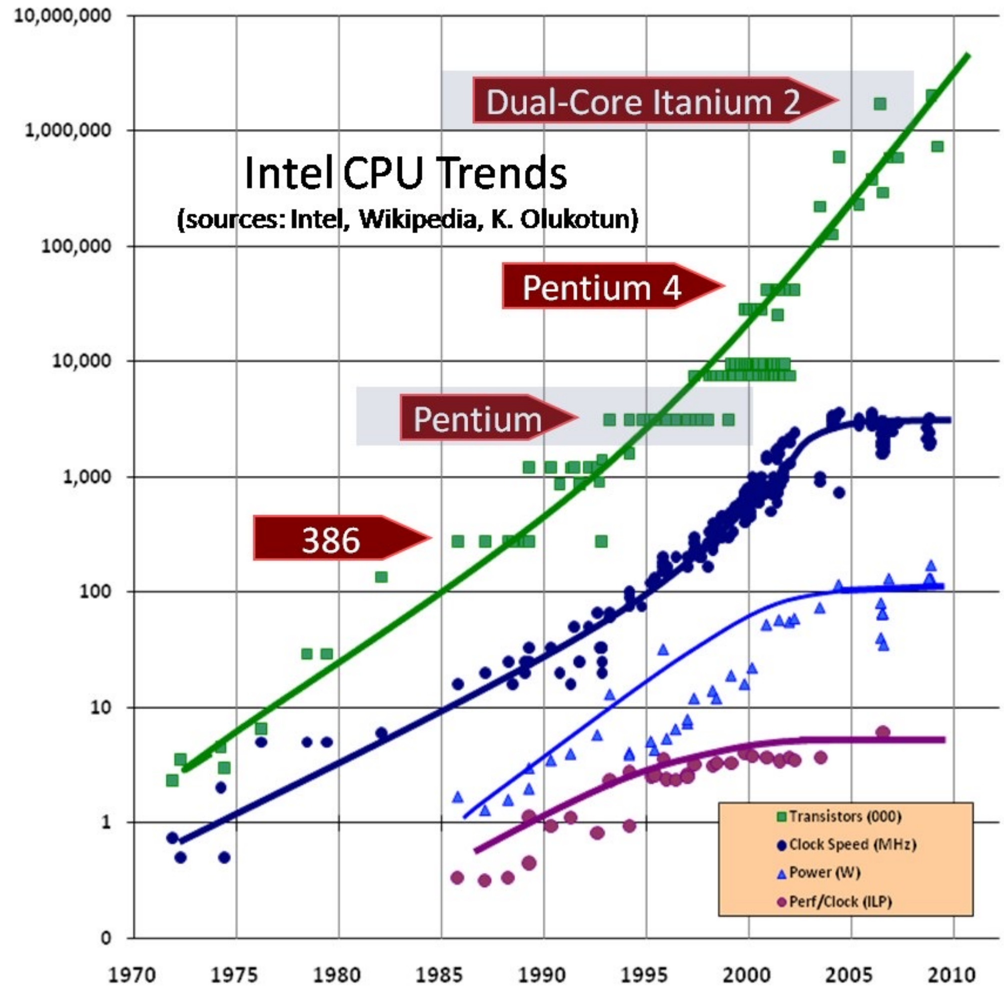


Gordon Moore
(Intel co-founder)

Moore's Law Misquote

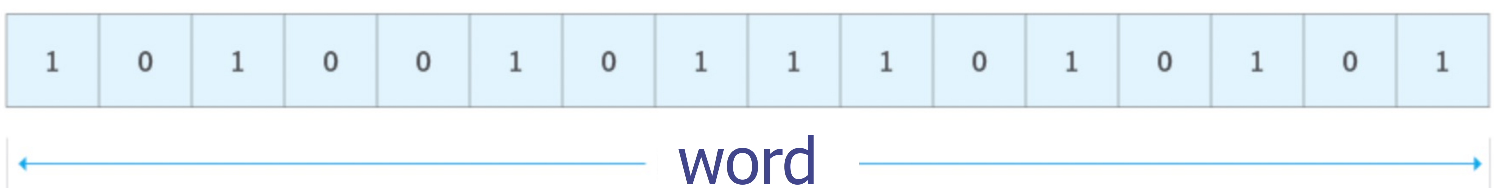
- Not clock speed

Grace Hopper gave out "nanoseconds" to prove this point.



Word RAM Model

- The **word RAM** (word random-access machine) model is a model of computation in which a random-access machine can do arithmetic and bitwise operations on a word of w bits in constant time.



- Examples: bitwise AND, OR, XOR, MSB

Example Word RAM Algorithm

- Given two arrays, A and B, of n bits, compute the first bit where A and B differ.
  1. Compute C = A XOR B. Time: O(n/w)
  2. Repeatedly test each word of MSB for equality with 0. Time: O(n/w).
  3. For the first word, c, in C that is not 0, compute MSB(c).
- Total running time: O(n/w).

Understanding the Orders of Magnitude

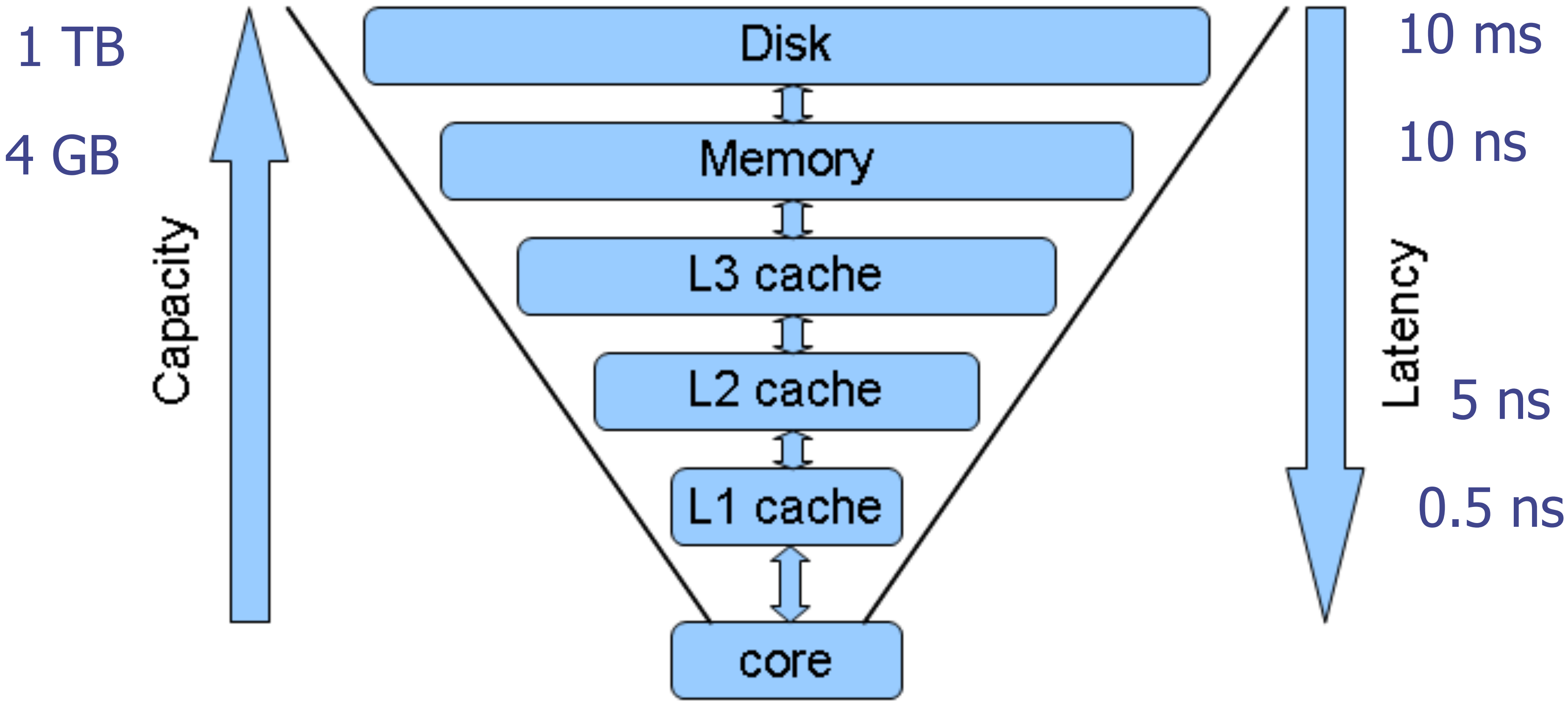
Internal memory: 10 ns

External memory: 10 ms

1 million times slower!

The Memory Hierarchy

- The trade-off of size and speed



Analogy: Cooking Eggs

- Suppose Anna is cooking eggs in in Irvine and wants to add salt and pepper

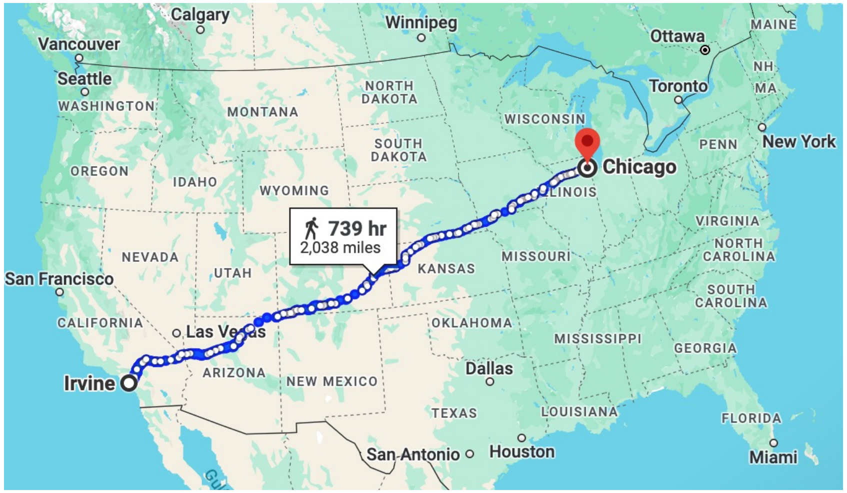
- Suppose it takes her 10 seconds to go to her pantry, get salt and pepper and add them to her eggs

Analogy: Cooking Eggs

- If going to her pantry is like a computer going to internal memory, then what would be analogous to going to external memory?

Analogy: Cooking Eggs

- If going to her pantry is like a computer going to internal memory, then what would be analogous to going to external memory?



Walking to Chicago, buying salt and pepper, and walking back

External Memory Model

- External memory identifies the frontier between the highest two layers in the memory hierarchy for a particular data set.

  B = block size

  M = "internal" memory size

  m = M/B (number of internal blocks)

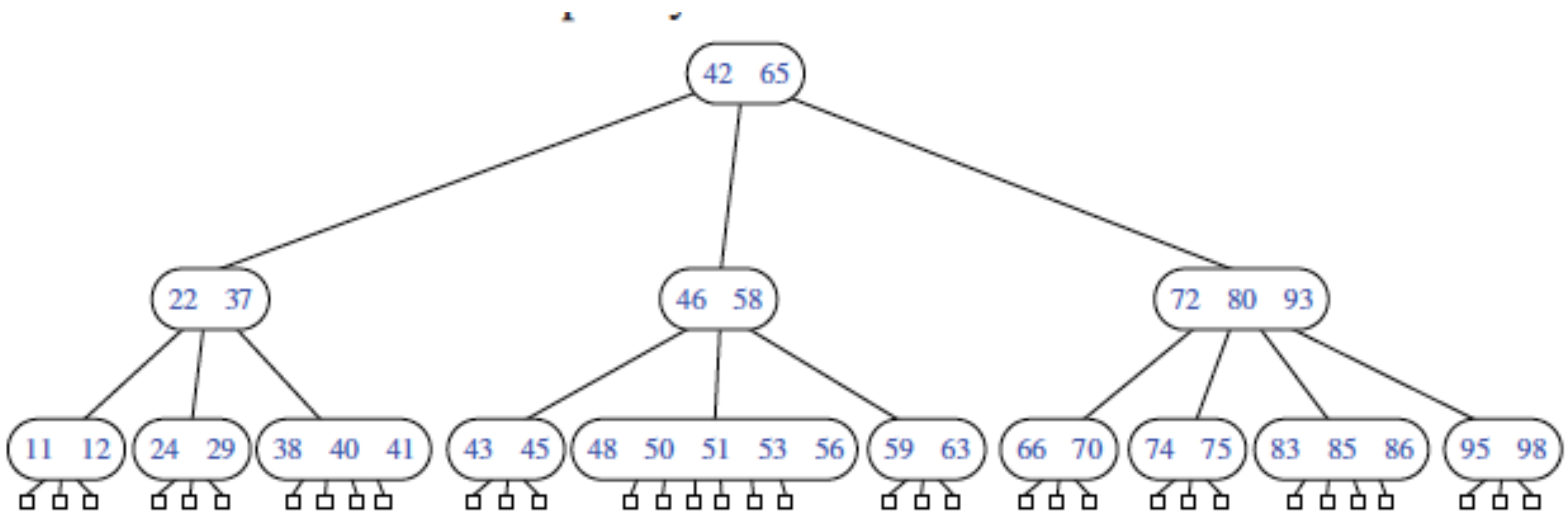
  N = "external" input size

  n = N/B (number of input blocks)

- The model only counts the number of reads and writes to external memory (I/Os). All other costs are ignored.

B-Trees

- A version of the **(a,b)** tree data structure, which is the best-known method for maintaining a map in external memory, is a "**B-tree.**"
- A **B-tree of order d** is an **(a,b)** tree with **a = d/**2 and **b = d**.



B-tree I/O Complexity

Proposition 15.2: *A B-tree with n entries has I/O complexity $O(\log_B n)$ for search or update operation, and uses $O(n/B)$ blocks, where B is the size of a block.*

- **Proof**:
  - Each time we access a node to perform a search or an update operation, we need only perform a single disk transfer.
  - Each search or update requires that we examine at most **O(**1**)** nodes for each level of the tree.

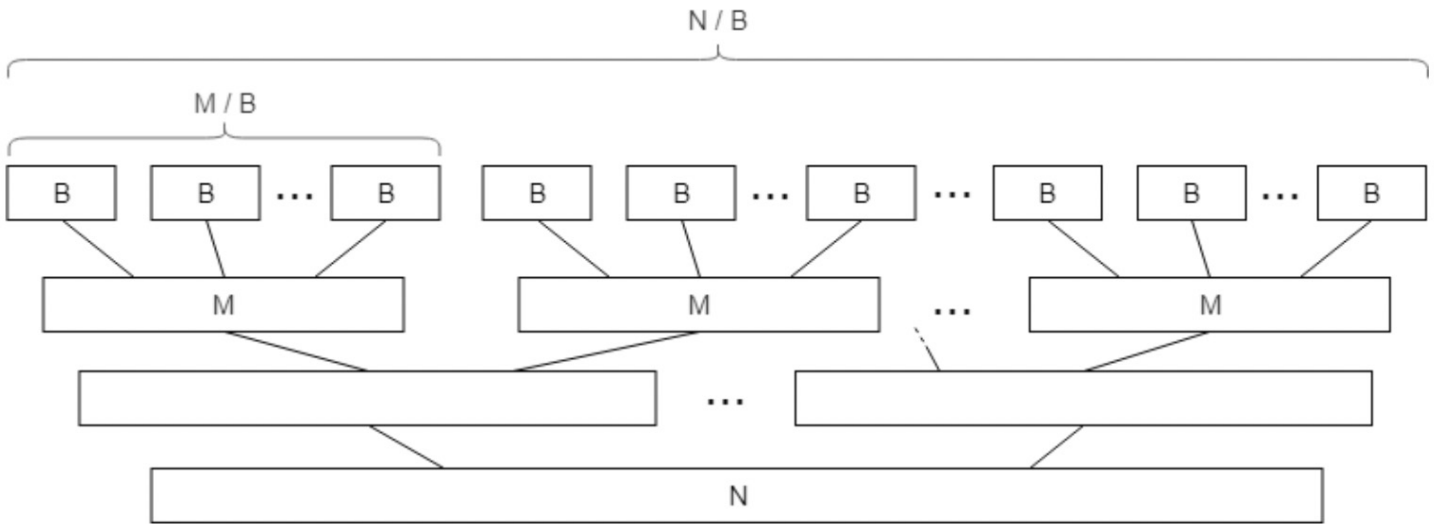
External-Memory Sorting

- Which of these sorting algorithms is good/bad in external memory?

- Insertion-sort
- Heapsort
- Shellsort
- Mergesort
- Quicksort

Better External-Memory Sorting

- **Multi-way Merge-sort:**
- Merge M/B sorted subarrays instead of 2.



- Number of I/Os: $O((N/B)\ \log_{M/B}(N/B))$.
- This is optimal.

Parallel RAM (PRAM)

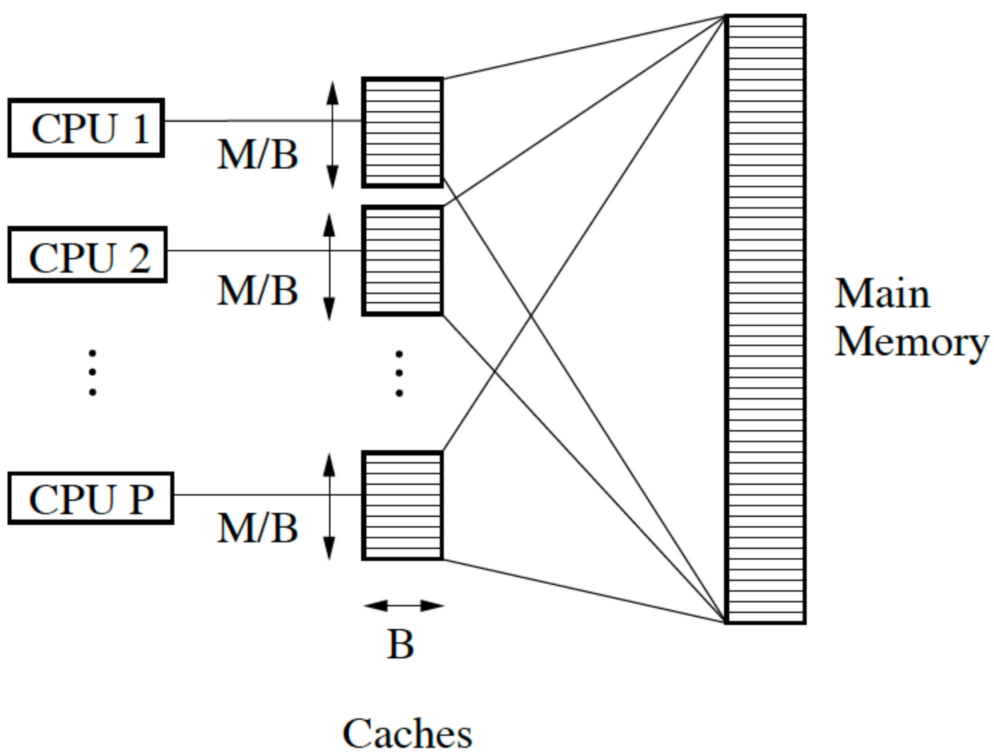
- Synchronous shared memory model.
  - Exclusive Read (ER): p processors can simultaneously read the content of p distinct memory locations.
  - Concurrent Read (CR): p processors can simultaneously read the content of p' memory locations, where $p' < p$.
  - Exclusive Write (EW): p processors can simultaneously write the content of p distinct memory locations.
  - Concurrent Write (CW): p processors can simultaneously write the content of p' memory locations, where $p' < p$.

PRAM Algorithms

- CRCW PRAM can compute OR of n bits in O(1) time with n processors.
- CRCW PRAM can compute Min of n numbers in O(1) time with n^2 processors.
- Merge two sorted lists of n elements in O(log n) time with n processors in CREW PRAM.
- Sort n numbers in $O(\log^2 n)$ time with n processors by parallel merge-sort in CREW PRAM.

Parallel External Memory (PEM)

- In joint work, we defined a parallel external memory model and designed efficient algorithms for it.



Fundamental Parallel Algorithms for Private-Cache Chip Multiprocessors

Lars Arge[*,†]
MADALGO
University of Aarhus
Aarhus, Denmark
large@madalgo.au.dk

Michael T. Goodrich
University of California, Irvine
Irvine, CA 92697, USA
goodrich@ics.uci.edu

Michael Nelson
University of California, Irvine
Irvine, CA 92697, USA
mjnelson@ics.uci.edu

Nodari Sitchinava
University of California, Irvine
Irvine, CA 92697, USA
nodari@ics.uci.edu

ABSTRACT

In this paper, we study parallel algorithms for private-cache chip multiprocessors (CMPs), focusing on methods for foundational problems that are scalable with the number of cores. By focusing on private-cache CMPs, we show that we can design efficient algorithms that need no additional assumptions about the way cores are interconnected, for we assume that all inter-processor communication occurs through the memory hierarchy. We study several fundamental problems, including prefix sums, selection, and sorting, which often form the building blocks of other parallel algorithms. Indeed, we present two sorting algorithms, a distribution sort and a mergesort. Our algorithms are asymptotically optimal in terms of parallel cache accesses and space complexity under reasonable assumptions about the relationships between the number of processors, the size of memory, and the size of cache blocks. In addition, we study sorting lower bounds in a computational model, which we call the parallel external-memory (PEM) model, that formalizes the essential properties of our algorithms for private-cache CMPs.

Categories and Subject Descriptors

F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—*Sorting and searching*

[*]Supported in part by the US Army Research Office through grant W911NF-04-01-0278, by an Ole Roemer Scholarship from the Danish National Science Research Council, a NABIIT grant from the Danish Strategic Research Council, and by the Danish National Research Foundation.

[†]Center for Massive Data Algorithmics – a Center of the Danish National Research Foundation

SPAA'08, June 14–16, 2008, Munich, Germany.

General Terms

Algorithms, Theory

Keywords

Parallel External Memory, PEM, private-cache CMP

1. INTRODUCTION

Advances in multi-core architectures are showing great promise at demonstrating the benefits of parallelism at the chip level. Current architectures have 2, 4, or 8 cores on a single die, but industry insiders are predicting orders of magnitude larger numbers of cores in the not too distance future [12, 20, 23]. Such advances naturally imply a number of paradigm shifts, not the least of which is the impact on algorithm design. That is, the coming multicore revolution implies a compelling need for algorithmic techniques that scale to hundreds or even thousands of cores. Parallelism extraction at the compiler level may be able to handle part of this load, but part of the load will also need to be carried by parallel algorithms. This paper is directed at this latter goal.

There is a sizable literature on algorithms for shared-memory parallel models, most notably for variations of the PRAM model (e.g., see [17, 18, 24]). Indeed, some researchers (e.g., see [26]) advocate that PRAM algorithms can be directly implemented in multicores, since separate cores share some levels of the memory hierarchy, e.g. the L2 cache or main memory. After all, an argument can be made that ignoring the memory hierarchy during algorithm design worked reasonably well for the single-processor architectures: in spite of recent developments in the cache-optimal models, most algorithms implemented and used by an average user are designed in the RAM model due to the small size of average input sets and relative simplicity of the RAM algorithms. However, we feel that to take advantage of the parallelism provided by the multicore architectures, problems will have to be partitioned across a large number of processors. Therefore, the latency of the shared memory will have a bigger impact on the overall speed of execution of the algorithms, even if the original problem fits into the memory of a single processor. The PRAM model contains

PEM Sorting Result

- Sorting with optimal parallel I/O complexity:

$$O\left(\frac{N}{PB}\log_{\frac{M}{B}}\frac{N}{B}\right)$$

  - N: Input size
  - P: # of processors
  - M: memory (cache) size
  - B: block (cache line) size