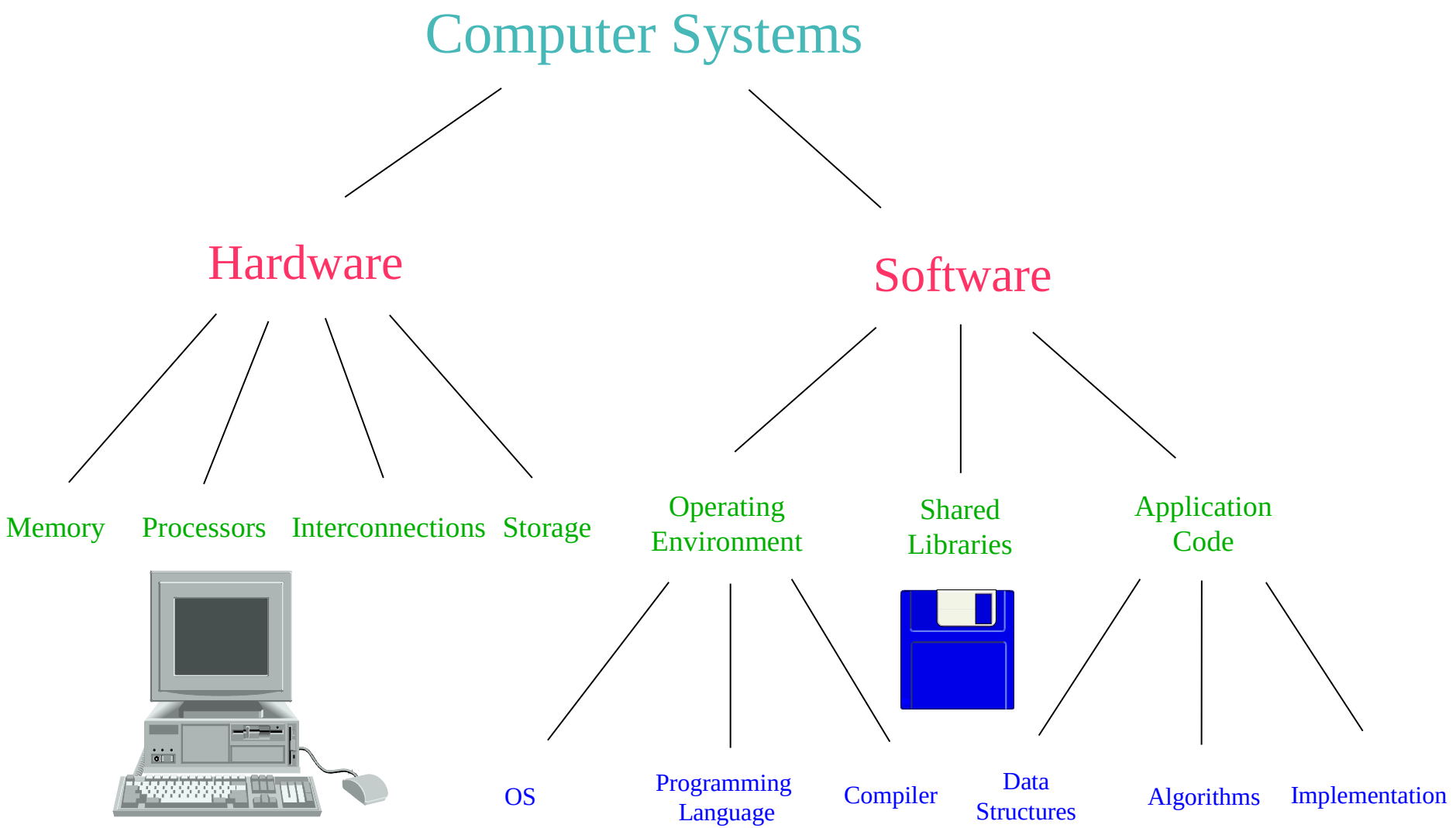
Models of Computation

IE 496 Lecture 1

Reading for this lecture

- Miller and Boxer, Chapters 1 and 5
- Aho, Hopcroft, and Ullman, Chapter 1

What is a Computer System?



What Are Computational Methods?

- Computational methods are
  - Algorithms based on logical procedures rooted in mathematics
  - Meant to be implemented on a computer
- Such methods are usually stated initially in the language of mathematics.
- This class is about how to translate the mathematics into a programming language.
- This step must be done (primarily) by a human.
- Doing this well requires an understanding of how computer systems work.

Pseudo-code notation

- We will often need to write pseudo-code, which uses notation from mathematics and programming languages.
- Our notation will be loosely based on C/Python with some parallel constructs.
- Declarations, etc. can be left out when the context makes it clear.
- Basic functions which are not the focus of the exercise can simply be called.

```
for (i = 0; i < 10; i++)
  parallel for (j = 0; j < 10; j++)
        find the minimum element of x[i][10*j, 10*(j+1)];
```

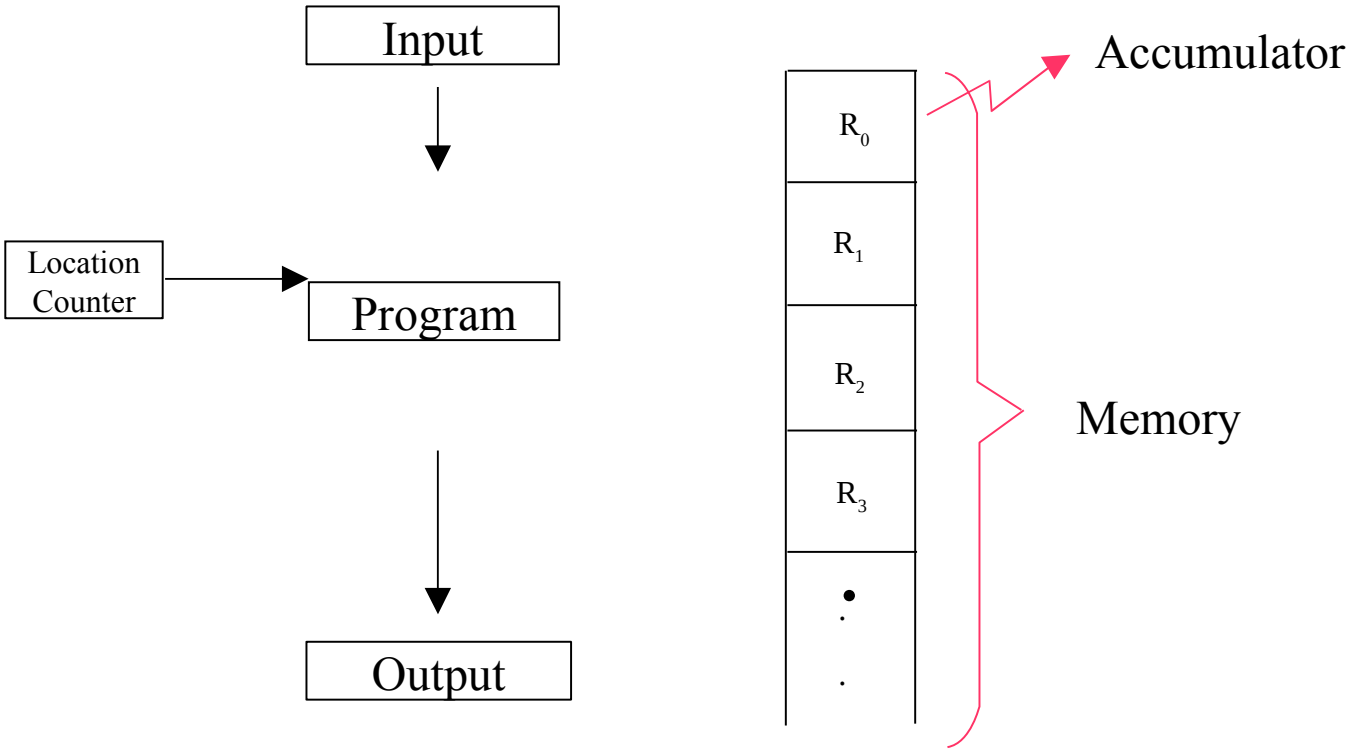
What is a Model of Computation?

- In order to do formal analysis of computational methods, we need a conceptual model of how a computer works.
- Established models do not take into account details of the hardware like the memory hierarchy.
- The analysis we can do with these models is primarily asymptotic.

Analysis of Algorithms

- We are interested in the time and space needed to perform an algorithm.
- There are several ways of approaching this analysis.
  - Worst case
  - Average case
  - Best case
- Worst case is the most common type of analysis (why?).
- Generally speaking, time is the most constraining resource.

Random Access Machine Model



A RAM Program

- At each time step, one elementary operation is completed.
- Sample list of elementary operations

- LOAD
- STORE
- ADD
- SUB
- MULT
- DIV
- READ
- WRITE
- JUMP
- JGTZ
- JZERO
- HALT

Execution of the Program

- In each time step, one instruction is executed.
- The instructions are similar to those in a machine language.
- Each instruction consists of an *operation* and an *address*.
- The “address” can be either a location in the code to jump to or a memory location or just a constant operand.
- The input is from a “tape” that is read sequentially and is read-only.
- The output is written sequentially to a “tape” that is write-only.

Languages

- The simplest kind of RAM program is one that reads an input string and outputs either 1 (TRUE) or 0 (FALSE).
- We say that a given input string is accepted by the program if the output for that string is TRUE.
- The characters that are allowed in forming the input string are called the *alphabet*.
- The set of strings that are accepted by a program P is called the *language accepted by P*.

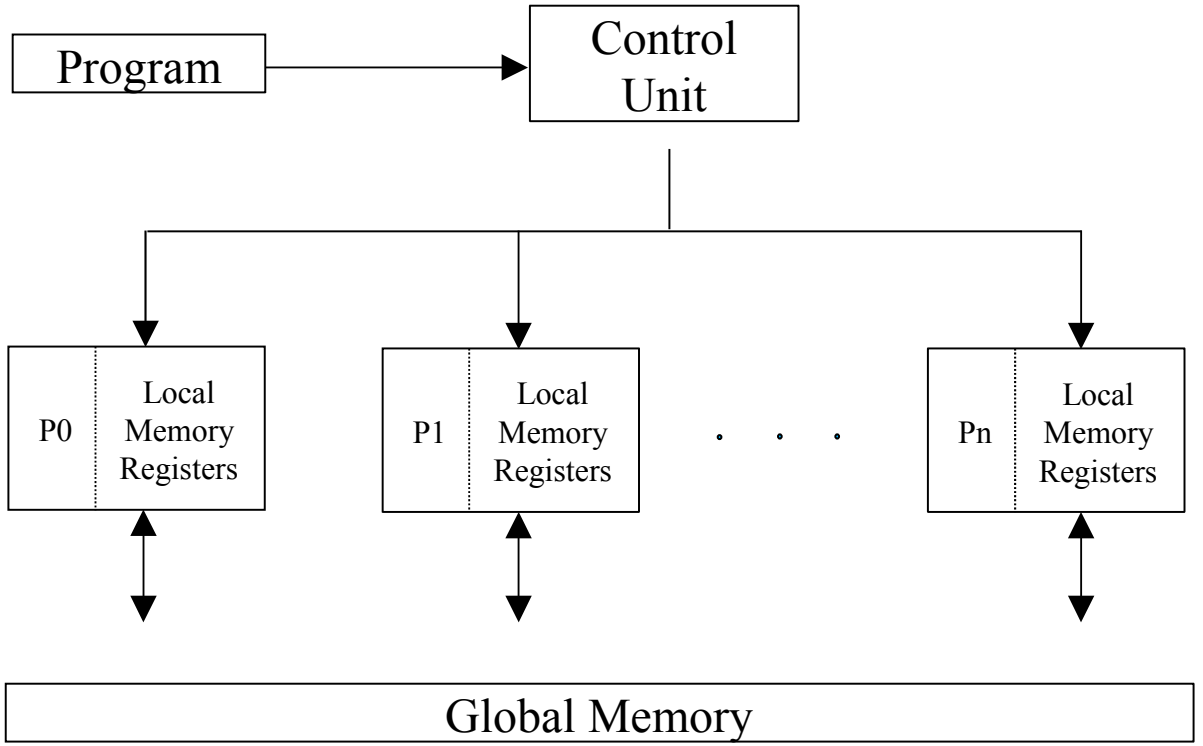
Assumptions of the RAM model

- The program is not stored in memory and hence cannot be modified.
- The problem is small enough to fit in the memory.
- Any size integer is allowed.
- Fundamental operations can be performed in one unit of time.
- Any memory location can be accessed in one unit of time.
- This is what is known as a "unit cost model".

Assessment of the model

- The details of the model are not especially important.
- Sequential Computation Thesis: All "reasonable" models are "polynomially equivalent".
- The assumptions of the model allow us to do rigorous asymptotic analysis.
- It is possible to abuse the assumptions of the model.
- *Log cost* model takes into account the size of the numbers.

The Basic PRAM model



Assumptions of the PRAM model

- This is a synchronous model with shared memory.
- There are a fixed number of processors (bounded).
- All processors execute the same program, but each one can be in a different place.
- At each time step, each processor performs one read, one elementary operation, and one write.
- Memory access is performed in constant time.
- Processors are not linked directly.
- Communication issues are not considered.

Concurrent Memory Access

- What if two processors try to read/write to/from the same memory location in the same time step?
- We have to resolve these conflicts.
- Four possible models:
  - CREW <--- we will use this one (most of the time)
  - CRCW
  - EREW
  - ERCW

Assessment of the PRAM Model(s)

- This model is not as "robust" as the RAM model.
- However, it allows us to do rigorous analysis.
- It is a reasonable model of a small parallel machine.
- It is not "scalable".
- It does not model distributed memory or interconnection networks.
- How do we fix it?

Distributed PRAM Model

- Attempt to model the interconnection network.
- Eliminate global memory.
- Each processor can read or write only from its neighbors' registers.
- This will likely increase the complexity of many algorithms, but is more realistic and scalable.

Computational Complexity

Computational Complexity

- The time complexity of an algorithm is the number of time steps needed to execute it.
  - Worst case
  - Average case
  - Best case
- The space complexity is the number of registers required to execute the algorithm.
- Complexity is usually expressed as a function *f* that maps the *size of the input* to the worst-case *running time*.
- Algorithms that executes in polynomial time and space are usually considered "good".

Asymptotic Analysis

- We are interested in how algorithms behave as the input size increases, i.e., asymptotically.
- Order relations help us group functions according to their approximate rate of growth.
- Definitions

All constants are positive in these definitions

  - $f \in O(g) \Leftrightarrow \exists\, c, n_0$ s.t. $f(n) \leq cg(n)\ \forall\, n \geq n_0$
  - $f \in \Omega(g) \Leftrightarrow \exists\, c, n_0$ s.t. $f(n) \geq cg(n)\ \forall\, n \geq n_0$
  - $f \in \Theta(g) \Leftrightarrow \exists\, c_1, c_2, n_0$ s.t. $c_1 g(n) \leq f(n) \leq c_2 g(n)\ \forall\, n \geq n_0$
  - $f \in o(g) \Leftrightarrow \forall\, C, \exists\, n_0$ s.t. $f(n) < Cg(n)\ \forall\, n \geq n_0$
  - $f \in \omega(g) \Leftrightarrow \forall\, C, \exists\, n_0$ s.t. $f(n) > Cg(n)\ \forall\, n \geq n_0$

Limitations of Asymptotic Analysis

- Ignores constant factors
  - These are nearly impossible to model
  - Example:

```
for (i = 0; i < 10; i++)
    write i;

for (i = 9; i >= 0; i++)
    write i;
```

- Generally speaking, these models do not take the minor details of implementation into account.
- Small problem sizes
- Worst case vs. average case

Cost of a Parallel Algorithm

- In the case of a RAM algorithm, the measure of performance was the time (number of steps).
- In the PRAM case, we may consider both time and cost.
  - *Running time* is the number of steps required in “real time”.
  - *Cost* is the product of running time and number of processors.
- An “optimal” parallel algorithm is one for which the cost is the same as the running time on a single processor.
- In some cases, the running time decreases with additional processors, but the cost increases.

Comparing PRAM models

Simple examples

- Broadcasting a unit of data
  - $O(1)$ under the shared-memory CREW model
  - $O(n)$ under the shared-memory EREW model
  - $O(\sqrt{n})$ under the distributed-memory CREW model on a mesh
  - $O(\log n)$ under the distributed-memory tree model
- Note: These models are architecture dependent
- This is the biggest difference between sequential and parallel complexity analysis

Semigroup operations

- Definition: A binary associative operation.
  - $\Rightarrow (x \otimes y) \otimes z = x \otimes (y \otimes z)$
- Typical semigroup operations.
  - maximum
  - minimum
  - sum
  - product
  - OR
- Can be used to compare parallel architectures.

Semigroup operations example

- RAM Algorithm
- Shared-memory PRAM Algorithm
  Assumptions: *n* processors, CREW
  Input: An array $X = [x_1, x_2, \ldots, x_{2n}]$
  Output: The smallest entry of X

```
for (i = 0; i < log2(n); i++){
    parallel for (j = 0; j < 2^(log(n)-i-1); j++){
        read x2j-1 and x2j;
        write min(x2j-1, x2j);
    }
}
```

t_1 is the desired minimum