IE 495 Lecture 3

Septermber 5, 2000

Reading for this lecture

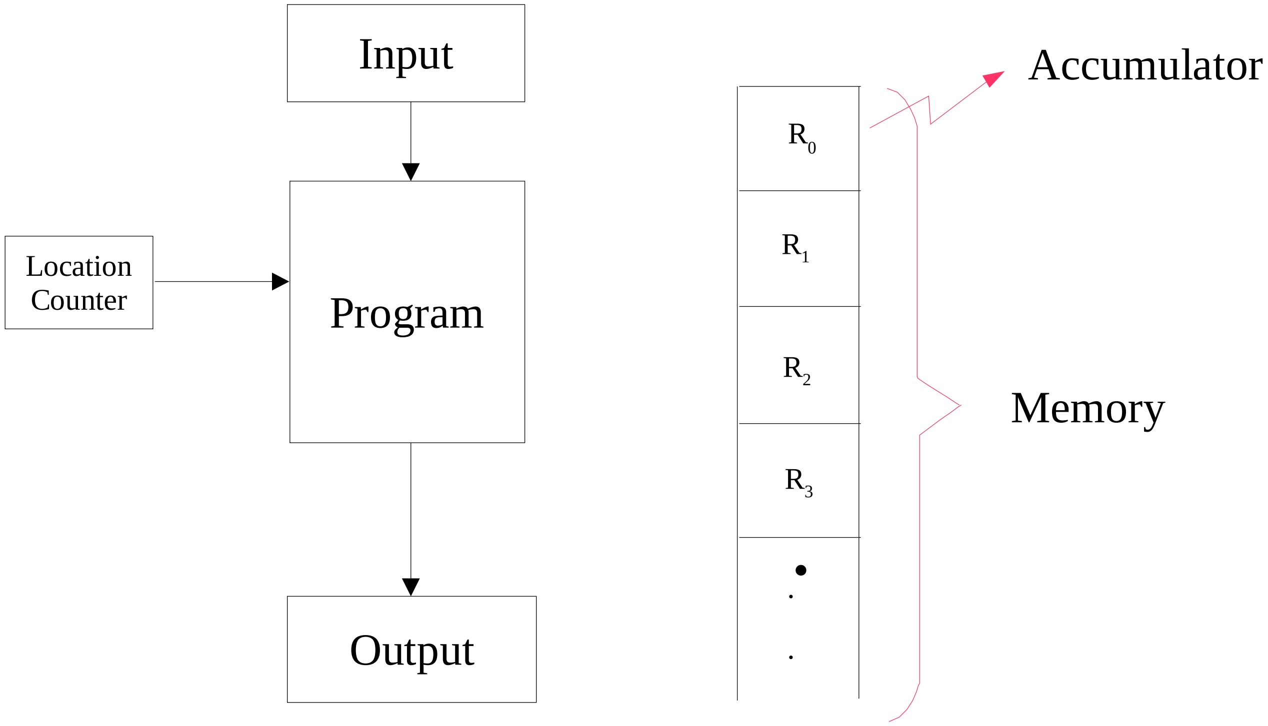
- Primary
  - Miller and Boxer, Chapter 1
  - Aho, Hopcroft, and Ullman, Chapter 1
- Secondary
  - Parberry, Chapters 3 and 4
  - Cosnard and Trystram, Chapter 5
  - Chaudhuri, Chapters 2 and 3

Models of Computation

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



Input

Location Counter

Program

Output

R_0

R_1

R_2

R_3

Accumulator

Memory

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

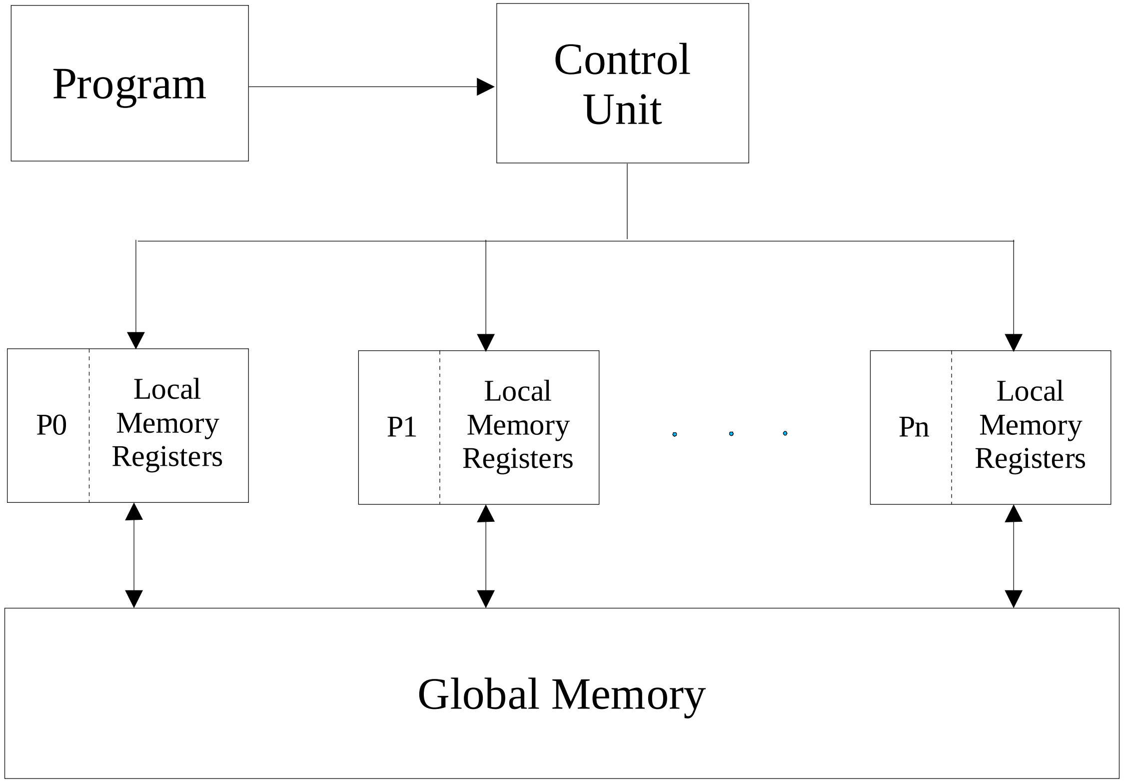
Assumptions of the RAM model

- 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



Program

Control Unit

P0 Local Memory Registers

P1 Local Memory Registers

. . .

Pn Local Memory Registers

Global Memory

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 elementary operation.
- Memory access is performed in constant time.
- Processors are not linked directly.
- Communication issues are not considered.
- What are some problems with this model?

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.

Algorithmic Complexity

Algorithmic 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 $f(n)$, where n is the size of the input.
- Algorithms that execute 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

  - $f(n) \in \mathrm{O}(g(n)) \Leftrightarrow \exists\, c, n_0$ s.t. $f(n) \le cg(n)\ \forall\, n \ge n_0$
  - $f(n) \in \Omega(g(n)) \Leftrightarrow \exists\, c, n_0$ s.t. $f(n) \ge cg(n)\ \forall\, n \ge n_0$
  - $f(n) \in \Theta(g(n)) \Leftrightarrow \exists\, c_1, c_2, n_0 \textit{ s.t. } c_1 g(n) \le f(n) \le c_2 g(n)\ \forall\, n \ge n_0$
  - $f(n) \in o(g(n)) \Leftrightarrow \forall\, C, \exists\, n_0$ s.t. $f(n) < \mathrm{C}g(n)\ \forall\, n \ge n_0$
  - $f(n) \in \omega(g(n)) \Leftrightarrow \forall\, C, \exists\, n_0$ s.t. $f(n) > \mathrm{C}g(n)\ \forall\, n \ge n_0$

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;
```

- Small problem sizes
- Worst case vs. average case

Comparing the 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*(√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 $X = [x_1, x_2, . . . , x_{2n}]$

Output: The smallest entry of *X*

```
for (i = 0; i < $\log_2(n)$; i++){
    parallel for (j = 0; j < $2^{\log(n)-i-1}$; j++){
        read $x_{2j-1}$ and $x_{2j}$;
        write min($x_{2j-1}$, $x_{2j}$);
    }
}
```

t_1 is the desired minimum

Example: Insertion Sort