The RAM Computation Model

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

This is not a programming course.

Main take-away message from this course

Computer science is a branch of mathematics with its art reflected in the beauty of algorithms.

- Programming knowledge is not necessary to study algorithms.

Many people believe that this branch holds the future of mankind.

In mathematics (and hence, computer science) everything — including every term and symbol — must be rigorous.

Computer science is a subject where we

1. first define a **computation model**, which is a **simple** yet **accurate** abstraction of a computing machine;
2. then slowly build up a theory for this model from scratch.

The Random Access Machine (RAM) model

A machine has a **memory** and a **CPU**.

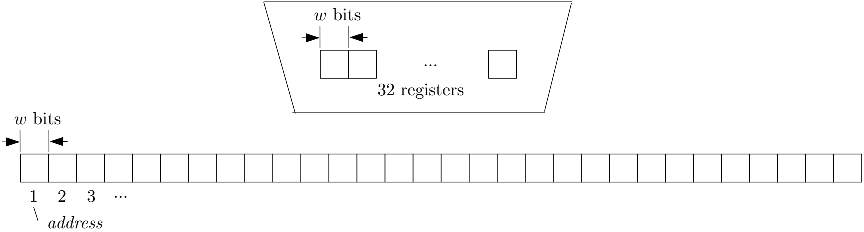
Memory

- An infinite sequence of **cells**, each of which contains the same number w of bits.
- Every cell has an **address**: the first cell of memory has address 1, the second cell 2, and so on.

The Random Access Machine (RAM) model

CPU

- Contains a fixed number — 32 in this course — of **registers**, each of which has w bits (i.e., same as a memory cell).



The Random Access Machine (RAM) model

CPU

- Can do the following **atomic operations**:

  1. **(Register (Re-)Initialization)**
     Set a register to a fixed value (e.g., 0, -1, 100, etc.), or to the content of another register.

The Random Access Machine (RAM) model

CPU

- Can do the following **atomic operations**:

  2. **(Arithmetic)**
     Take the integers a, b stored in two registers, calculate one of the following, and store the result in a register:
     - $a + b$, $a - b$, $a \cdot b$, and a/b.

> Note: a/b is "integer division", which returns an integer. For example, $6/3 = 2$ and $5/3 = 1$.

The Random Access Machine (RAM) model

CPU

- Can do the following **atomic operations**:

  3. **(Comparison/Branching)**
     Take the integers a, b stored in two registers, compare them, and learn which of the following is true:
     - $a < b$, $a = b$, $a > b$.

The Random Access Machine (RAM) model

CPU

- Can do the following **atomic operations**:

  4. **(Memory Access)**
     Take a memory address A currently stored in a register. Do one of the following:
     - Read the content of the memory cell with address A into another register (overwriting the bits there).
     - Write the content of another register into the memory cell with address A (overwriting the bits there).

The Random Access Machine (RAM) model

An **execution** is a sequence of atomic operations.

Its **cost** (also called **running time**, or simply, **time**) is the length of the sequence, namely, the number of atomic operations.

The Random Access Machine (RAM) model

A **word** is a sequence of w bits, where w is called the **word length**.

- In other words, each memory cell and CPU register store a word.

> Unless otherwise stated, you do not need to pay attention to the value of w in this course.

Algorithm

- An **input** refers to the initial state of the registers and the memory before an execution starts.
- An **algorithm** is a piece of description that, given an input, can be utilized to produce a sequence of atomic operations, namely, the algorithm's execution.
- The **cost** of an algorithm on an input is the length of the algorithm's execution on that input (i.e., the number of atomic operations required).
- The **space** of an algorithm on an input is the **largest** memory address accessed by the algorithm's execution on that input.

Example 1

Problem: Suppose that an integer of $n \geq 1$ has already been stored in a register. We want to calculate $1 + 2 + ... + n$ and store the result in a register.

Example 1

Suppose that n is stored in register a.
Set register b to 0, c to 1, and d to 1.

Repeat the following until $c > a$:

- Calculate $b + c$ and store the result in b.
- Calculate $c + d$ and store the result in c (effectively increasing c by 1).

Cost of the execution of our algorithm $= 3n + 3$

Think: which atomic operations are performed?

We have described the algorithm in English. The next slide shows a different way to describe the same.

Example 1

```
/* n in register a */
register b ← 0, c ← 1, d ← 1
repeat
    b ← b + c
    c ← c + d
until c > a
return b
```

The above is called **pseudocode**. As you can see, we do not restrict ourselves to any particular programming languages. The description is in a "free form", mixing English words and programming-like statements as we wish, as long as it serves the purpose of clarifying —without ambiguity — our strategy.

Example 2

Same problem as before. But this time we will give a faster algorithm.

Example 2

Set register a to 1, and b to 2. Let c be the register storing n. Then:

- Set a to $a + c$ (note: now a equals $n + 1$).
- Set a to $a * c$ (now a equals $n(n + 1)$).
- Set a to a/b (now a equals $n(n + 1)/2$).

Cost of the execution $= 5$

This is significantly faster than the previous algorithm when n is large. In particular, the time of the previous algorithm increases **linearly** with n, while the time of the above one remains **constant**.

Although we have not talked about how to implement our algorithms into **actual** programs, it should be straightforward for you (who must have passed a programming course) to do so.

> Computer scientists **rarely** have a programming language in mind when attacking a problem. They, instead, focus on the algorithm design because once an algorithm is clear, turning it into a program is merely a matter of translation and experience.