Great Ideas in Computing

University of Toronto CSC196
Fall 2020

Week 1: September 9-September 14 (2020)

Course Organization

Course Instructor: Allan Borodin

- Email: bor@cs.toronto.edu
- Instructor course email: 196instr@cs.toronto.edu

Teaching Assistant : TBA
TA Course Email : 196ta@cs.toronto.edu

Note: These course emails may take a couple of days to set up

Course web site: http://www.cs.toronto.edu/~196f20/

Preliminaries

Getting to know each other

- A little about myself
- Your plans at the University?

What is this course about?

- Calendar introduction (read general statement for all first-year foundational seminar courses in the Faculty of Arts and Sciences)
- Brief theme of this CSC196 course: Great Ideas in Computing. What constitutes a "great idea"?.

Preliminaries

Getting to know each other

- A little about myself
- Your plans at the University?

What is this course about?

- Calendar introduction (read general statement for all first-year foundational seminar courses in the Faculty of Arts and Sciences)
- Brief theme of this CSC196 course: Great Ideas in Computing. What constitutes a "great idea"?.

  Impact, surprise, elegance. Of course, it is easier to agree on great ideas in retrospect rather than as ideas are being introduced. And of course ideas rarely occur in a "vacuum"; usually there are similar ideas known and often the timing of when an idea becomes exposed is very critical. It is also the case that credit for an idea is not always completely fair to all those involved.

Grading scheme, syllabus and possible topics

- The grading scheme will be based on 4 assignments (15% each), two quizzes (10% each), and class participation (20%). Students are expected to attend all classes regularly and participate actively. There will be no final exam.
- Assignments will be submitted on Markus https://markus.teach.cs.toronto.edu/csc196-2020-09/
- The syllabus (listed on the course web page) contains other organizational information.
- There is also an ambitious list of possible topics on the web page. What have we missed?

Grading scheme, syllabus and possible topics

- The grading scheme will be based on 4 assignments (15% each), two quizzes (10% each), and class participation (20%). Students are expected to attend all classes regularly and participate actively. There will be no final exam.
- Assignments will be submitted on Markus https://markus.teach.cs.toronto.edu/csc196-2020-09/
- The syllabus (listed on the course web page) contains other organizational information.
- There is also an ambitious list of possible topics on the web page. What have we missed?

  I have also lined up some guest speakers to lead discussions on some recent "great ideas". So far, in alphabetical order we have guests for the following: deep learning, differential privacy, quantum computing, and virtualization.
- **NOTE:** My slides will only be an outline of our discussions.

Our first great idea

Our first topic/great idea for discussion: the von Neumann architecture

- By the mid 40s the first computers were being built. (I am not going to talk say about Babbage and the recent implementation of Babbages 1850 machine.)
- The earliest computers had fixed programs and were not general purpose machines.
- The stored program computer. The conceptual idea is associated with von Neumann who first described the model in a paper "First Draft of a Report on the EDVAC" dated June 30, 1945.
- It is clear that the stored program idea is present in Turing's 1936 paper which we will later discuss.

The von Neumann architecture and digitilization

- The basic organization consists of 4 units: memory, I/O, ALU, control. (Some would say that a "bus" to cary signals between units is a fifth component but most people say that the basic architecture is 4 units.) The memory is organized into a list or array of "words", each with its own address.
- Each word w_i (with say address i) is some fixed length string of bits bits ; i.e., $w = b_{n-1}b_{n-2}, \ldots, b_0$ which as an integer represents $\sum_{j=0}^{n-1} 2^i$. (The choice of 2 is not essential mathematically.). It is also possible to think of a word as a string of "bytes" in which each byte can also addressable but we will ignore that. Why not unary, decimal, or some other representation?

The von Neumann architecture and digitilization

- The basic organization consists of 4 units: memory, I/O, ALU, control. (Some would say that a "bus" to cary signals between units is a fifth component but most people say that the basic architecture is 4 units.) The memory is organized into a list or array of "words", each with its own address.
- Each word w_i (with say address i) is some fixed length string of bits bits ; i.e., $w = b_{n-1}b_{n-2}, \ldots, b_0$ which as an integer represents $\sum_{j=0}^{n-1} 2^i$. (The choice of 2 is not essential mathematically.). It is also possible to think of a word as a string of "bytes" in which each byte can also addressable but we will ignore that. Why not unary, decimal, or some other representation?
- But before we go any further we need to step back and talk about *digitalization and encoding*. That is, what are we storing in these words?

The von Neumann architecture and digitilization

- The basic organization consists of 4 units: memory, I/O, ALU, control. (Some would say that a "bus" to cary signals between units is a fifth component but most people say that the basic architecture is 4 units.) The memory is organized into a list or array of "words", each with its own address.
- Each word w_i (with say address i) is some fixed length string of bits bits ; i.e., $w = b_{n-1}b_{n-2}, \ldots, b_0$ which as an integer represents $\sum_{j=0}^{n-1} 2^i$. (The choice of 2 is not essential mathematically.). It is also possible to think of a word as a string of "bytes" in which each byte can also addressable but we will ignore that. Why not unary, decimal, or some other representation?
- But before we go any further we need to step back and talk about *digitalization and encoding*. That is, what are we storing in these words? Data and instructions (as part of a program)

The von Neumann architecture and digitilization

- The basic organization consists of 4 units: memory, I/O, ALU, control. (Some would say that a "bus" to cary signals between units is a fifth component but most people say that the basic architecture is 4 units.) The memory is organized into a list or array of "words", each with its own address.
- Each word w_i (with say address i) is some fixed length string of bits bits ; i.e., $w = b_{n-1}b_{n-2}, \ldots, b_0$ which as an integer represents $\sum_{j=0}^{n-1} 2^i$. (The choice of 2 is not essential mathematically.). It is also possible to think of a word as a string of "bytes" in which each byte can also addressable but we will ignore that. Why not unary, decimal, or some other representation?
- But before we go any further we need to step back and talk about *digitalization and encoding*. That is, what are we storing in these words? Data and instructions (as part of a program)
- The concept of digitalization is one of the most profound concepts that underlies science today.

The September 9 class ended at slide 6

We will continue with some further discussion of the von Neumann model and digitilization. First some announcements.

Announcements

- Assignment 0 has been reposted. I have now set the due date to be Tuesday, September 15, 11PM (instead of Wednesday, 11AM) as initially stated.
- I have posted on quercus the link to the recording of the meeting on 9-9-20
- Piazza is now available for CSC196. I prefer general comments to be posted on piazza. More private comments can be sent to the email 196instr@cs.toronto.edu . I really appreciate students answering other students. It not only saves me time but more important indicates when I do and don't have to clarify things. http://piazza.com/utoronto.ca/fall2020/csc196h1flec01019101?token=g

More announcements

- Still waiting to be assigned a TA for our course. As I explained, the problem is that our initial TA is an international student who cannot currently enter Canada. That means that he cannot open a bank account which means thaty he can't get paid which means that he can't work.
- I am not available on Monday, September 28. When I am assigned a TA for the course and if that person is available we might use that time slot for a tutorial. Otherwise the class will be cancelled.
- Our Wednesday, September 30, the class will be led by Professor Henry Yuen, an expert in quantum computing. I will soon schedule more guests.

Comment about the end of our last class

Related to our discussion last class as to "are we responsible for the tools we build", you might want to watch the following:
The Social Dilemma: A Powerful New Netflix Documentary On How Social Media Is Changing Our Lives

I watched it last night and found it interesting although sensationalized.

We could spend the entire course on the impact of online social media (but we won't). But if you have about 1.5 hours to spare you might want to watch this documentary/drama on netflix.

Lots of interesting comments in the documentary.
. Like all new technolgies one has to ask can we adapt to the challenges? At the start of the "computer revolution", the question was "will computers, automation, robots" eliminate work for most indiviudal and then "will computers/robots turn on us".

Digital computing vs the continuous real world

- In the "real world", space and time are continous. A typical computing application might wish to study the trajectory or movement of an object (person, ball, missle, etc). An object is moving through continous 3 dimensional space in continuous time.
- A real value x can be approximated by a rational number q
- Every integer has a finite representation and every rational can be represnted by a pair of integers and hence can also have a finite reperesentation.
- Fixed length words and floating point numbers
- The early debate: relative benefits of digital computation vs that of analogue computation. (In the past, thermostats and other control devices were essentially simple analogue computers.) Why did digitlilization win the debate (so far)?

Memory and the von Neumann architecture

The von Neumann architecture is a *random access stored program model*.

- The von Neuman model assumes a random access memory in which each word is "addressable" and can be accessed at some unit (of time) cost.
  Is this how your lap top memory is organized?

Memory and the von Neumann architecture

The von Neumann architecture is a *random access stored program model*.

- The von Neuman model assumes a random access memory in which each word is "addressable" and can be accessed at some unit (of time) cost.
  Is this how your lap top memory is organized?
  The"von Neumann bottleneck" addressed by caching and the memory hierarchy

Memory and the von Neumann architecture

The von Neumann architecture is a *random access stored program model*.

- The von Neuman model assumes a random access memory in which each word is "addressable" and can be accessed at some unit (of time) cost.
  Is this how your lap top memory is organized?
  The"von Neumann bottleneck" addressed by caching and the memory hierarchy
- Algorithms consist of individual instructions that say what "basic operations" to perform on data and also to indicate what instruction to do next.
- Now here is a great idea (relating to digitalization): Instructions can also be represented by strings of symbols (indeed by strings of bits)! So instructions can also be stored in the memory, say for example one instruction per word!

Memory and the von Neumann architecture

The von Neumann architecture is a *random access stored program model*.

- The von Neuman model assumes a random access memory in which each word is "addressable" and can be accessed at some unit (of time) cost.
  Is this how your lap top memory is organized?
  The"von Neumann bottleneck" addressed by caching and the memory hierarchy
- Algorithms consist of individual instructions that say what "basic operations" to perform on data and also to indicate what instruction to do next.
- Now here is a great idea (relating to digitalization): Instructions can also be represented by strings of symbols (indeed by strings of bits)! So instructions can also be stored in the memory, say for example one instruction per word!
  Why is this such a great idea?

Memory and the von Neumann architecture

The von Neumann architecture is a *random access stored program model*.

- The von Neuman model assumes a random access memory in which each word is "addressable" and can be accessed at some unit (of time) cost.
  Is this how your lap top memory is organized?
  The"von Neumann bottleneck" addressed by caching and the memory hierarchy
- Algorithms consist of individual instructions that say what "basic operations" to perform on data and also to indicate what instruction to do next.
- Now here is a great idea (relating to digitalization): Instructions can also be represented by strings of symbols (indeed by strings of bits)! So instructions can also be stored in the memory, say for example one instruction per word!
  Why is this such a great idea?
  Why is the von Neumann model such a great idea?
  Are we stuck in a "von Neumann tarpit?"

Dataflow architecture

Direct from Wikipedia:

Dataflow architecture is a computer architecture that directly contrasts the traditional von Neumann architecture or control flow architecture. Dataflow architectures do not have a program counter (in concept): the executability and execution of instructions is solely determined based on the availability of input arguments to the instructions,[1] so that the order of instruction execution is unpredictable, i.e. behavior is nondeterministic. Although no commercially successful general-purpose computer hardware has used a dataflow architecture, it has been successfully implemented in specialized hardware such as in digital signal processing, network routing, graphics processing, telemetry, and more recently in data warehousing.[citation needed] It is also very relevant in many software architectures today including database engine designs and parallel computing frameworks.[citation needed]

More on data flow architecture

From J. Paul Morrison's Flow-Based Programming text

The von Neumann machine is perfectly adapted to the kind of mathematical or algorithmic needs for which it was developed: tide tables, ballistics calculations, etc., but business applications are rather different in nature. . . .
Business programming works with data and concentrates on how this data is transformed, combined, and separated. . . . Broadly speaking, whereas the conventional approaches to programming (referred to as "control flow") start with process and view data as secondary, business applications are usually designed starting with data and viewing processes as secondary—processes are just the way data is created, manipulated, and destroyed. We often call this approach "data flow." (21)

Multicore and Parallel Computation

- As you probably already know, computers today are often multicore machines meaning that some "small" constant number of processes can be running simultaneously.
- When people refer to large scale parrallism they have in mind that the number of processes running in parallel can depend (at least conceptually) on the computation
- The von Neumann architecture is an abstract model for *sequential computation*.
- In contrast to the well accepted von Neumann model for sequential computation, the situation for *parallel computation* is more nuanced.
  - There is the issue of a constant number of parallel processes vs a n umber of processes that depends on the size of the data and/or the computation as it evolves.
  - Do the processes run syncronously (i.e. according to some golbal clock) or asynchronously?
  - Do the processes communicate mainly through a shared memory or via some communication bus?
  - How do we maintain consistency of the information being shared?

The benefits of a well agreed upon abstract model of computation

One of the reasons to consider the von Neumann model a great idea is that by being a well agreed upon model, coordination amongst different people is minimized. That is,

- A computer architect doesn't need to know which programming languages will be run on their specific architecture. (The von Neumann model doesn't specify the instruction set, the memory management, how interupts are handled, etc.)
- A compiler writer for a programming language $\mathcal{L}$ doesn't have to know what algorithms will be implemented using the language $\mathcal{L}$.
- Without complete knowledge of the architecture, and the compiler, the algorithm designer can make a rough approximation for the memory and time requirements of their algorithm.

The progress in parallel computation had been relatively slow but there is now some common approaches (e.g., MapReduce for large scale parallel computation).

Start of Monday, January 14 meeting

- We ended the Friday meeting on slide 15
- I plan to provide anonymously the top 5 lists for each student. These lists will be part refered to in Assignment 1.
- Did anyone look at the Netflix documentary The Social Dilemma?
- We will start today discussing floating point representation.
- Then we will discuss algorithmic ideas for a basic problem we we should all be familiar with, namely looking up information associated with for example a person, product, etc. This will also let us be a little more familiar with the von Neumann architecture and algorithm analysis.
- **Important comment on nature of the course:** For some students, how I proceed may seem too slow and already known while for others it may seem that I am going too fast. If it seems that I am going too fast, then STOP ME and I will try to elaborate.

Floating point representation

- As mentioned we have to approximate real non rational numbers by fractions. It is easy to see that say every real number can be approximated to arbitrary precision. In particular, every fraction in the interval $[0, 1]$ can be approximated by a fraction $.b_0 b_1 b_2 \ldots b_n$ where $b_i \in \{0, 1\}$. The more bits better the approximation.
- Of course some fractions can be represented exactly in such a binary representation (e.g. $1/2$, $1/4$, $3/4$, etc.) while other numbers (say in decimal) like $1/10$ and is $1/3$ can only be approximated.
- We may need very small or very large numbers but the numbers of bits in a computer word is fixed (say 32 bits) so this limits how big or how small numbers can be IF we just use pure binary representation. This is not an artifact of the binary representation. The same limitations would apply to any base.

Floating point numbers continued

- The most common solution to this issue is provided by *floating point numbers*. We will stick to binary but again any base can be used. A floating point number uses the following representation (where I am using # just for clarity) as the bits would all be consecutive :

  $s\#e_{k-1}e_{k-2}\dots e_0\#b_0b_1\dots b_{\ell-1}$

  Here the bit s represents the sign (i.e. $+$ or $-$) of the number.

  The e_i bits represent the exponent E with value $E \in [-2^{k-1}, 2^{k-1}]$

  and the b_i bits represent the significand (i.e, the significant bits)
- The number being represented is $\{+,-\}2^E \times b_0.b_1\dots b_{\ell-1}$.

Floating point numbers continued

- The IEEE standard for a 32 bit *single precision* number uses 7 bits for the exponent and 24 bits for the significand.
- There are also double (and multiple) precision numbers where a double precision number would occupy two 32 bit words.
- History: According to Wikipedia, Leonardo Torres y Quevedo used floating point numbers in his design of Babbage's Analytical Engine. See also the reference to Konrad Zuse who designed a computer in 1938 and later versions in 1941 who used floating point numbers.
- It is interesting to note that von Neumann argued for fixed point nunbers (and not floating point) in the design for an Institute of Advanced Study machine.
- Once again, it is interesting to note that an algorithm designer (usually) doesn't need to know the specifics of the floating point representation but just needs to know the commands for the arithmetic opertaions.

Wikipedia

Is Wikipedia a great idea?

Wikipedia

Is Wikipedia a great idea?

In the Netflix documentary that I mentioned, the following question/comment was made: What if everyone was given their own Wikipedia page when they made a query using Wikipedia?
The commentary notes that when we are on social media we are often getting personalized news-feeds.

Wikipedia

Is Wikipedia a great idea?

In the Netflix documentary that I mentioned, the following question/comment was made: What if everyone was given their own Wikipedia page when they made a query using Wikipedia?
The commentary notes that when we are on social media we are often getting personalized news-feeds.

Confession: I didn't think Wikipedia would work. More specifically, I didn't think that enough knowledgeable people would be willing to spend their time to help create reassonably authoratative articles without getting any credit.

What is your experience with Wikipedia? Do you always believe what you read is accurate? How does it compare with other sources of information?

Septmeber 24: End of first week of meetings

- We will begin Wednesday (the second week of classes) with the slide on Wikipedia.
- I am speaking to a potential TA and hopefully we will have a tutorial on Friday.
- I am going to decide about whether to use Quercus or Markus for submitting aassignments. This is my first course using Quercus so I have to ask about the relative merits. I will post something by tomorrow AM.