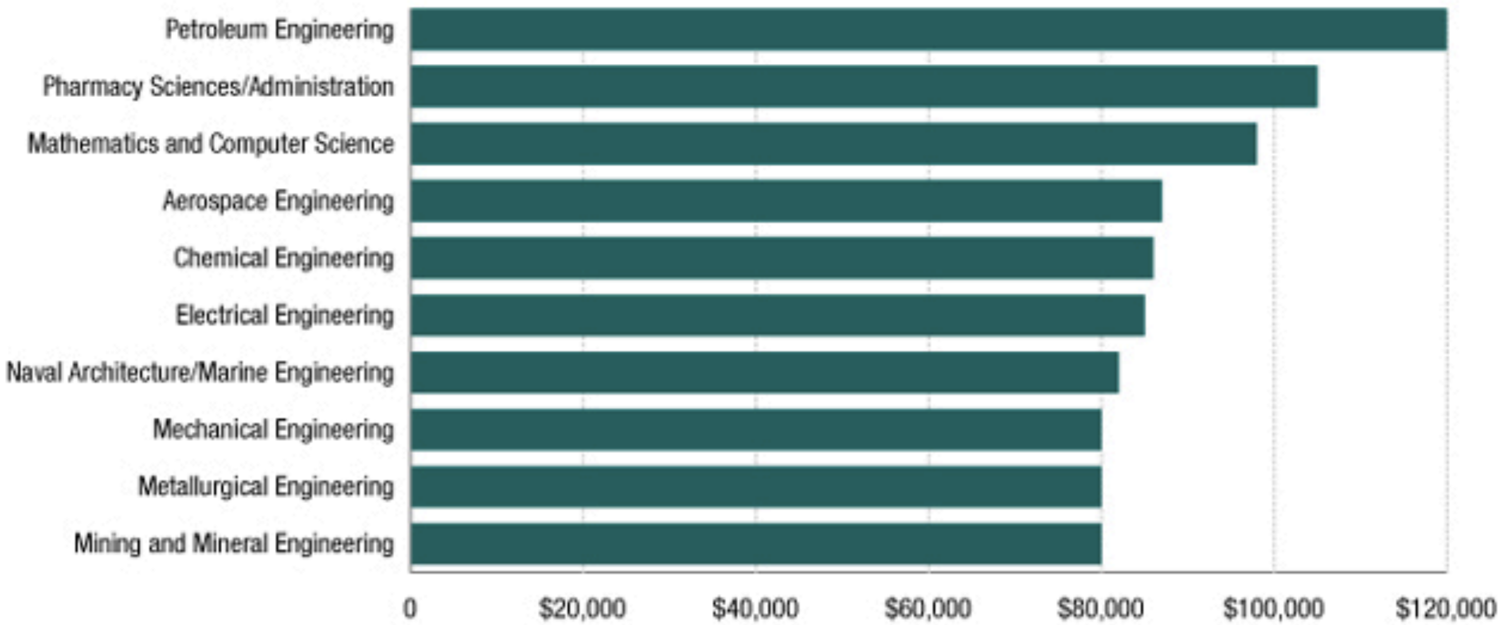
What is Computer Science? Part I

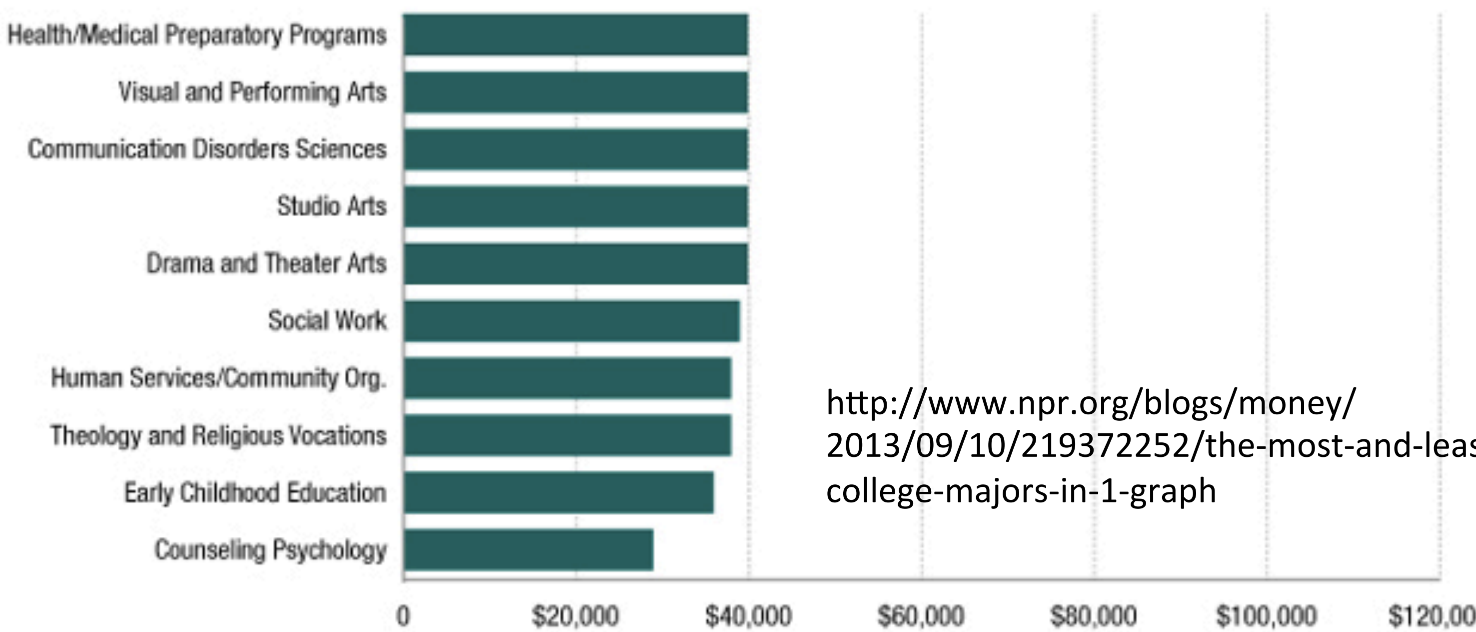
CS 100

Fall 2015

Majors With The Highest Earnings



Majors With The Lowest Earnings

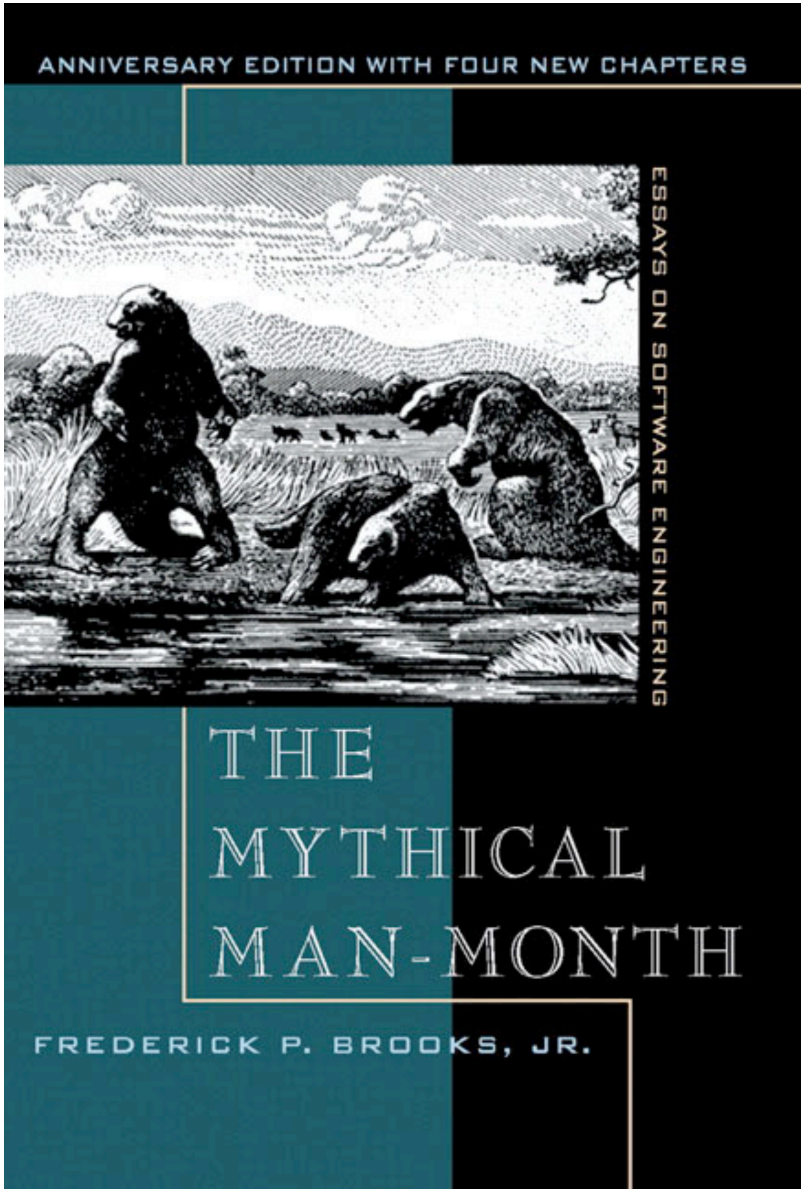


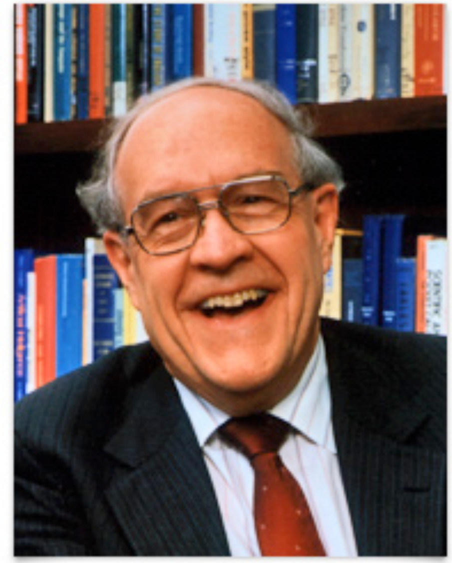
http://www.npr.org/blogs/money/2013/09/10/219372252/the-most-and-least-lucrative-college-majors-in-1-graph

"Computers represent a radical novelty ... Coming to grips with a radical novelty amounts to creating and learning a new foreign language that can not be translated into one's mother tongue"

- Edsger W. Dijkstra
("EWD")
1930-2002

Edsger W. Dijkstra, 1972 Turing Award Winner





"…The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are […] so readily capable of realizing grand conceptual structures…"

Fred Brooks, 1999 Turing Award Winner. "For landmark contributions to computer architecture, operating systems, and software engineering."


3 6 '9

The Name Game

- "Computer Science" is an unfortunate name

  "Computer Science is no more about computers than Astronomy is about telescopes." – EWD

  "Any discipline that has 'science' in its name isn't."

  - unknown

- Better (?) possibilities:
  - Computing Science
  - Abstraction Engineering

my favorite

Computing Science

Three basic parts:

- Foundations/Theory:
  - What can/can't be computed?
  - How can we compare different ways of solving a problem?
  - What is the optimal way to solve problems of a given type?
- Systems Design/Engineering:
  - How can we organize systems so that they compute faster, using less energy, in less space, ... ?
  - What can be done in hardware? ... software?
  - What are the right abstractions: what to hide, what to expose?
- Applications:
  - How can computation be used to make our lives better?

Example Sub-Fields

- Theory:
  - Numerical analysis, algorithms, complexity
- Systems Design/Engineering:
  - Networking, Operating Systems, Software Engineering, Visualization and Graphics
- Applications
  - Artificial Intelligence, Scientific Computing, Databases, Machine Learning, Data mining, Entertainment (Games), Biomedical Informatics, ...
    - Plus 400,000+ different things at the "App Store"!

What is Computation?

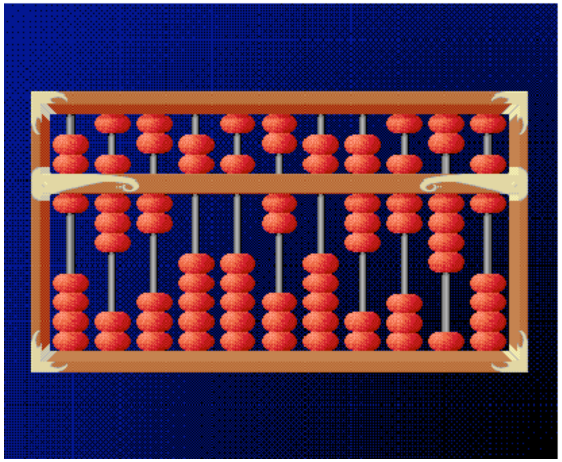
Definition 0: Computation what computers do.

- Then the question becomes:

What is a Computer?

- Something that performs, or aids in performing, some kind of calculation, usually involving symbol manipulation.

Jacquard loom
ca. 1800 →



← Abacus
ca. 2500 B.C.

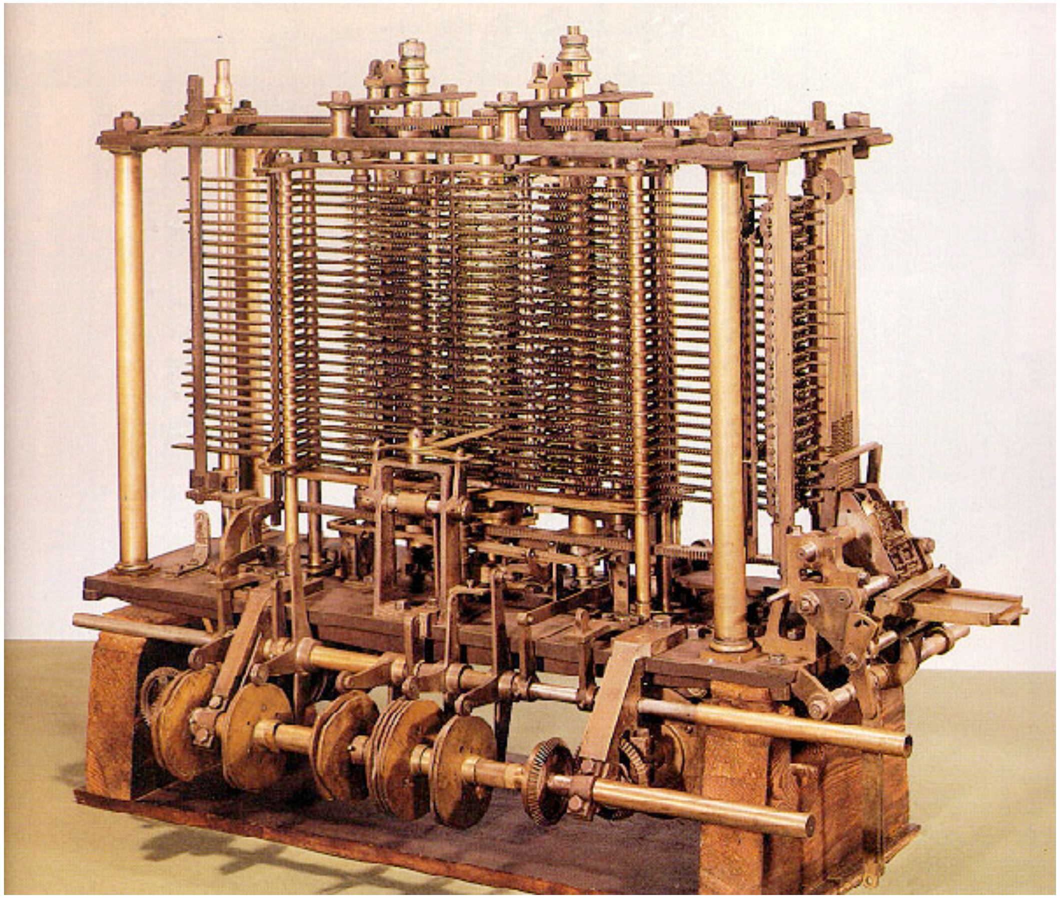
What is a “Computer”?

Some of the gals- circa 1944, Still from *Top Secret Rosies: The Female ‘Computers’ of WWII*

“Some of the gals – circa 1944” – taken from
Top Secret Rosies: The Female ‘Computers’ of WWII

A WWII Application of Computers: Ballistics Tables for Big Guns

Computing is Older Than You Think



A model of
Charles Babbage's
"Analytical Engine"
described circa 1837,
but (likely) never built.

It was the first
programmable digital
computer, and resembled
modern computer
architecture in many
ways. (!)

Texas Instruments SR-10

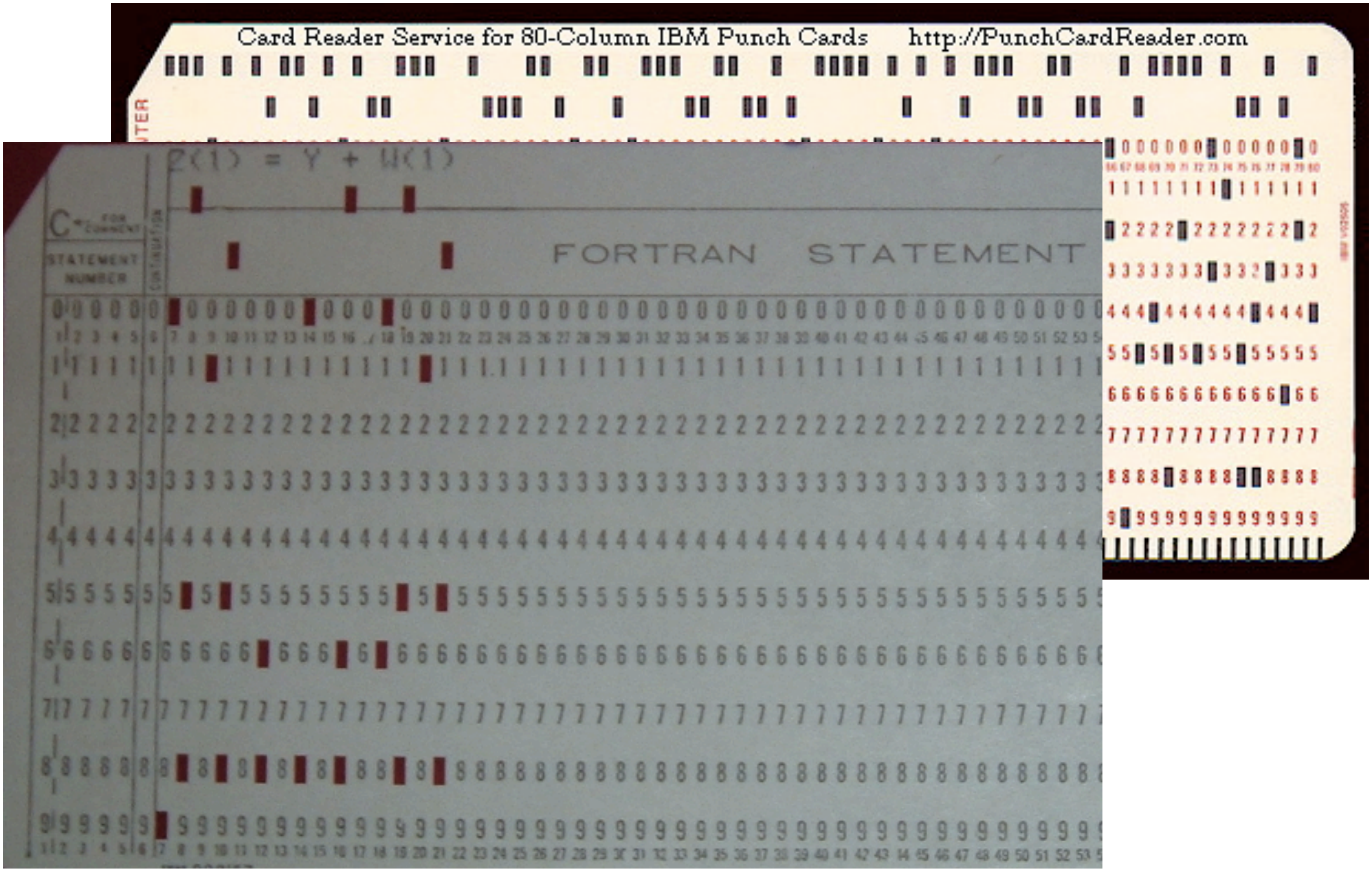
Introduced in November 1972, with a direct mail price of US$149.95

That is $859 in today's dollars…

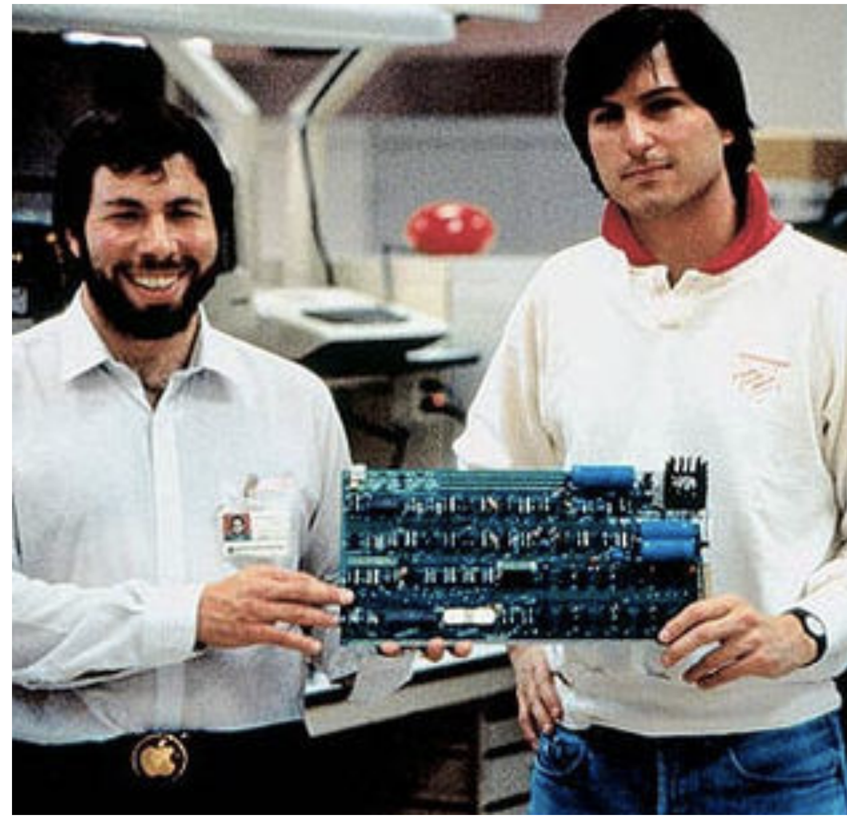
Fortran

C ← FOR COMMENT / STATEMENT NUMBER (1–5)	CONTINUATION (6)	FORTRAN STATEMENT (7–72)	IDENTI-FICATION (73–80)
C		PROGRAM FOR FINDING THE LARGEST VALUE	
C	X	ATTAINED BY A SET OF NUMBERS	
		DIMENSION A(999)	
		FREQUENCY 30(2,1,10), 5(100)	
		READ 1, N, (A(I), I = 1,N)	
1		FORMAT (I3/(12F6.2))	
		BIGA = A(1)	
5		DO 20 I = 2,N	
30		IF (BIGA-A(I)) 10,20,20	
10		BIGA = A(I)	
20		CONTINUE	
		PRINT 2, N, BIGA	
2		FORMAT (22H1THE LARGEST OF THESE I3, 12H NUMBERS IS F7.2)	
		STOP 77777	

Punch Cards

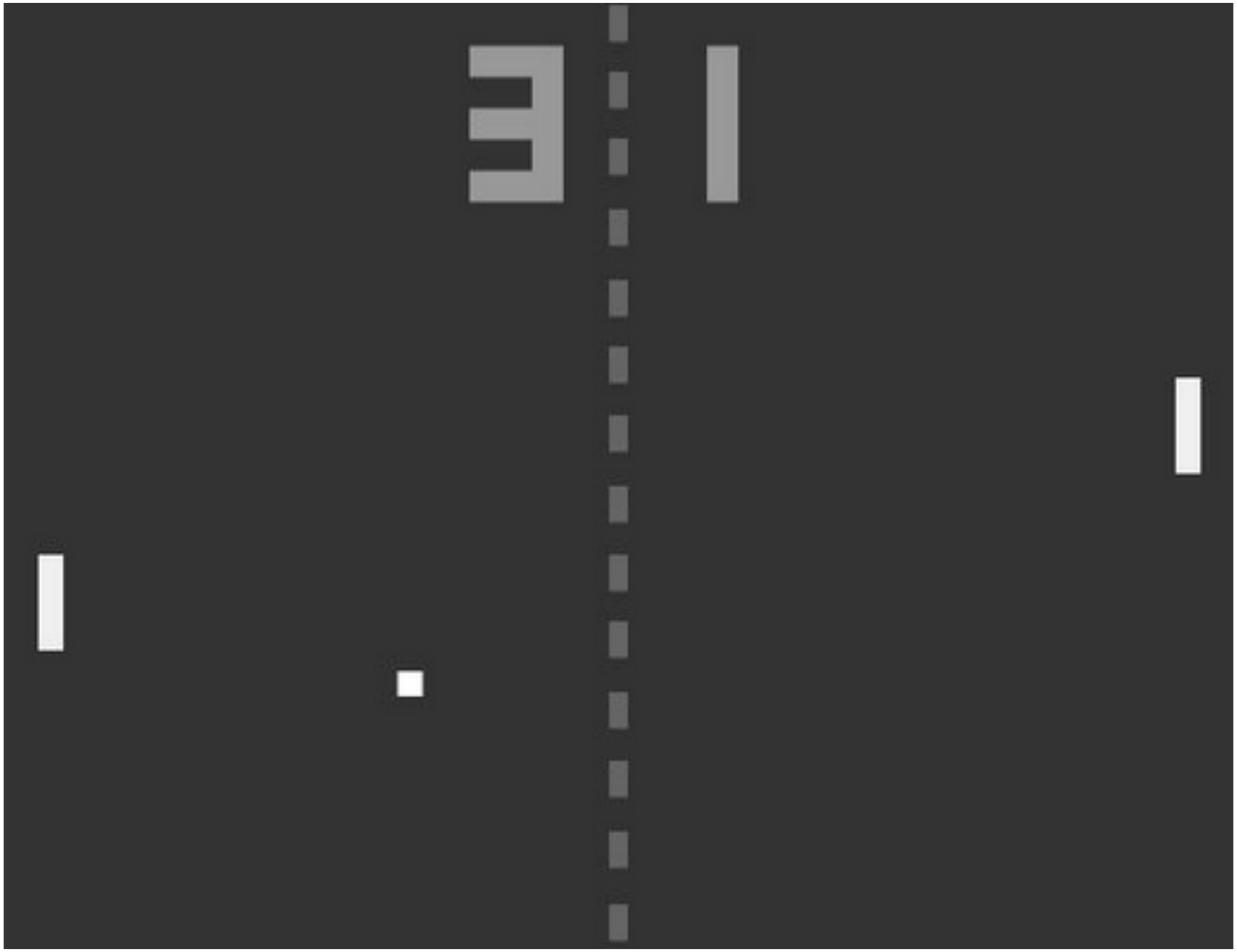

Card Reader Service for 80-Column IBM Punch Cards
http://PunchCardReader.com
Z(1) = Y + W(1)
STATEMENT NUMBER
FORTRAN STATEMENT

High Schools in 1979



US $1298 with 4K RAM   ($4,176 in 2013)
US $2638 with 48K RAM ($8,488 in 2013)


TELE-GAMES
PONG
ATARI
ON
PUSH
START GAME



US $100 from Sears   ($434 in 2013)

Computer: Key Characteristics

- A method of providing input
  - Encoded in some way (always!)
- Ability to be in one of a number of different states
  - Configuration of wheels, or contents of memory
- Ability to change state autonomously
  - Power must be applied
  - New state derived from old state
  - Computation defined by sequence of states
- A method of producing output

"Does it matter what hardware I get?"

- Fact: Once they attain a certain level of complexity, all computers have the same computational power.

Anything that <u>can be computed</u> on a Mac

can be computed on a Wintel machine

or a PDP11

or an iPhone

or a ...

What is Computation?

- Proposed Definition 1:

  Automatic symbol manipulation to accomplish some purpose.

  - "Automatic" = according to fixed rules
  - "symbol" = an abstract representation of something
  - "manipulation" = comparing, modifying

The Most General Computer

Alan Turing (1912-1954)

Came up with an idealized device as a model for reasoning about computation.

Today it is known as the "Turing Machine" model.

Because of its simplicity, it is (relatively) easy to reason about what it can/can't do.

Turing Machine Components

- (Semi-infinite) Tape, divided into squares
  - Input is written on the first part of the tape
  - Rest of the tape is blank
- Read-write head
  - Scans the square under the head
  - Can write 1, 0 or blank into the square
- Fixed control program
  - Keeps track of current program "state"
  - Next actions based on current state and tape contents
    - write 0 or 1 <u>and</u> move head L or R <u>and</u> change to next state

The Church-Turing Thesis

- Around the same time Alan Turing came up with the Turing Machine model, a number of other scientists were proposing other very basic models of computation
  - Alonzo Church was one
- Anything that can be computed, can be computed with a Turing Machine
    - Or with any of the other very simple models of computation: lambda calculus, combinators

The Universal Machine

Key idea:

- Write a Turing Machine Program that takes other programs as input
- Given a program as input, and the input data to run it on, simulate the computation that would occur if the given program were the TM's control program

The Universal Machine can simulate any other machine.

The Universal Machine

Church-Turing Thesis:

The Turing Machine – or any machine powerful enough to run a Universal Machine program – is as powerful as any thinkable computing device

- in the sense that anything that can be computed by one can be computed by the other
- "powerful enough:"
  - Conditionals ("if" statements)
  - "While" loops

What's the point? **It's all about the software!**

Combinators: Turing-Complete Computing Framework

- Function application:
  - Suppose f is a function (e.g., addition)
  - Denote the application of f to arguments by juxtaposition: f x y means "f(x,y)"
    - x and y here represent "any expression"
  - Except we assume association happens to the left: so f x y is really ((f x) y), or "take the result of applying f to x, and apply it to y"

Combinators

- Define two basic functions, call them S and K
  - K takes two arguments, throws the second away:

    K x y => x

    "=>" means "reduces to" – the symbol manipulation part!
  - S takes 3 arguments, applies 1st to 3rd, then applies that to result of applying 2nd to 3rd:

    S x y z => x z (y z) = (x z) (y z)
  - Theorem: Any function that can be computed can be defined in terms of these basic functions.
    - Idea: Build up other functions in terms of these:

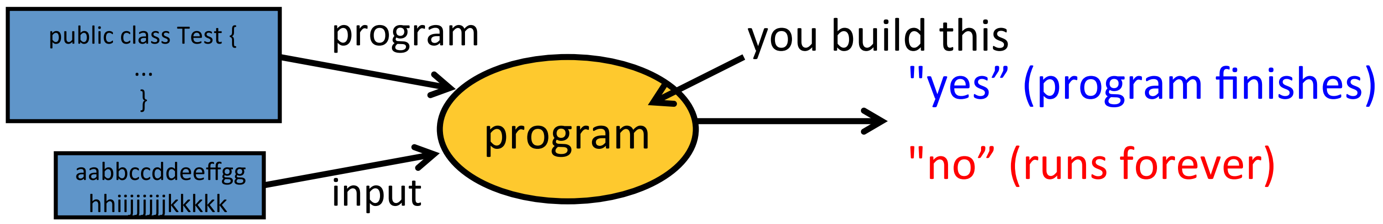
      Define B as S(KS)K. Then show B x y z => ... => x (y z)

      Define C as S(BBS)(KK). Then show C x y z => ... => x z y

A Hard Problem

Write a program (any language) that does the following:

- Inputs:
  - a Java program (stored in a text file)
  - another file, given as input to the Java program (System.in)
- Output: one word
  - print "yes" if the given program, if compiled and run with the given file as input, eventually terminates
  - "no" if the program never terminates with that file as input
- The program must always give an answer in finite time
- The program must work for any legal Java program and input



- Theorem: You can't do it. No program satisfies this specification!
  - Known as the "Halting Problem"
  - Alan Turing proved the impossibility of solving the Halting Problem before any electronic computer existed!

Another Hard One: Traveling Sales Rep

From \ To	ATL	KC	SAN	DEN	MSP	DFW
ATL	0	$350	$600	$575	$490	$375
KC	$350	0	$450	$200	$420	$300
SAN	$600	$450	0	$440	$550	$300
DEN	$575	$200	$440	0	$175	$275
MSP	$490	$420	$550	$175	0	$425
DFW	$375	$300	$300	$275	$425	0

- This table shows airfares to fly from city to city
  - Simple pricing model:
    - All flights one-way, nonstop
    - Price between two cities is the same in both directions
- You are a technical sales rep, and you have to visit all these cities each month

Traveling Sales Rep

To

From		ATL	KC	SAN	DEN	MSP	DFW
	ATL	0	$350	$600	$575	$490	$375
	KC	$350	0	$450	$200	$420	$300
	SAN	$600	$450	0	$440	$550	$300
	DEN	$575	$200	$440	0	$175	$275
	MSP	$490	$420	$550	$175	0	$425
	DFW	$375	$300	$300	$275	$425	0

- Your boss is a stickler, won't pay for you to fly to any city twice
- Your airfare budget is only $1600/month!
- Can you visit all cities once (& get home) for ≤ $1600?
- You suspect it's not possible, but want to be sure before asking your boss for more money

Traveling Sales Rep

To

From		ATL	KC	SAN	DEN	MSP	DFW
	ATL	0	$350	$600	$575	$490	$375
	KC	$350	0	$450	$200	$420	$300
	SAN	$600	$450	0	$440	$550	$300
	DEN	$575	$200	$440	0	$175	$275
	MSP	$490	$420	$550	$175	0	$425
	DFW	$375	$300	$300	$275	$425	0

- Generalize: write a program (algorithm) to answer this question:
  - Given a table of N cities, is there a tour that visits each city exactly once and ends up where it started, that costs less than $X?
  - Must work for any size table (N) and any bound X

Traveling Sales Rep

To

From		ATL	KC	SAN	DEN	MSP	DFW
	ATL	0	\$350	\$600	\$575	\$490	\$375
	KC	\$350	0	\$450	\$200	\$420	\$300
From	SAN	\$600	\$450	0	\$440	\$550	\$300
	DEN	\$575	\$200	\$440	0	\$175	\$275
	MSP	\$490	\$420	\$550	\$175	0	\$425
	DFW	\$375	\$300	\$300	\$275	\$425	0

- There is no known solution that works significantly better than trying all the possible tours!
- How many possible tours are there? (See CS 275)
  - For six cities: 60 (not counting rotations and reversals)
  - For twelve cities: 19,958,400
  - For 25 cities: 3.1 x 10^{23} (≈ half a mole)

Traveling Sales Rep

To

From		ATL	KC	SAN	DEN	MSP	DFW
	ATL	0	$350	$600	$575	$490	$375
	KC	$350	0	$450	$200	$420	$300
	SAN	$600	$450	0	$440	$550	$300
	DEN	$575	$200	$440	0	$175	$275
	MSP	$490	$420	$550	$175	0	$425
	DFW	$375	$300	$300	$275	$425	0

- But: Nobody has been able to prove that's the best you can do!
- So: in spite of decades of trying:
  - Nobody has found a better-than-brute-force solution
  - Nobody has proved there is no better solution
- Note: this is called the "P vs. NP" problem, and it is the most famous open problem in Computer Science

Theoretical Computer Science

- Deals with finding quantitative ways to answer the question “How hard is this problem?”
- Looks for lower bounds
  - “Any solution for this problem takes at least time exponential in the size of the input”
- Looks for better algorithms for all kinds of problems
  - Algorithm ≈ Turing Machine Program
  - A step-by-step procedure for computing output from input

Engineering Across Scale

- Take locomotion as an example
  - Crawl
  - Walk
  - Run
  - Ride a bike
  - Drive a car
  - Fly a plane
  - Rocket?

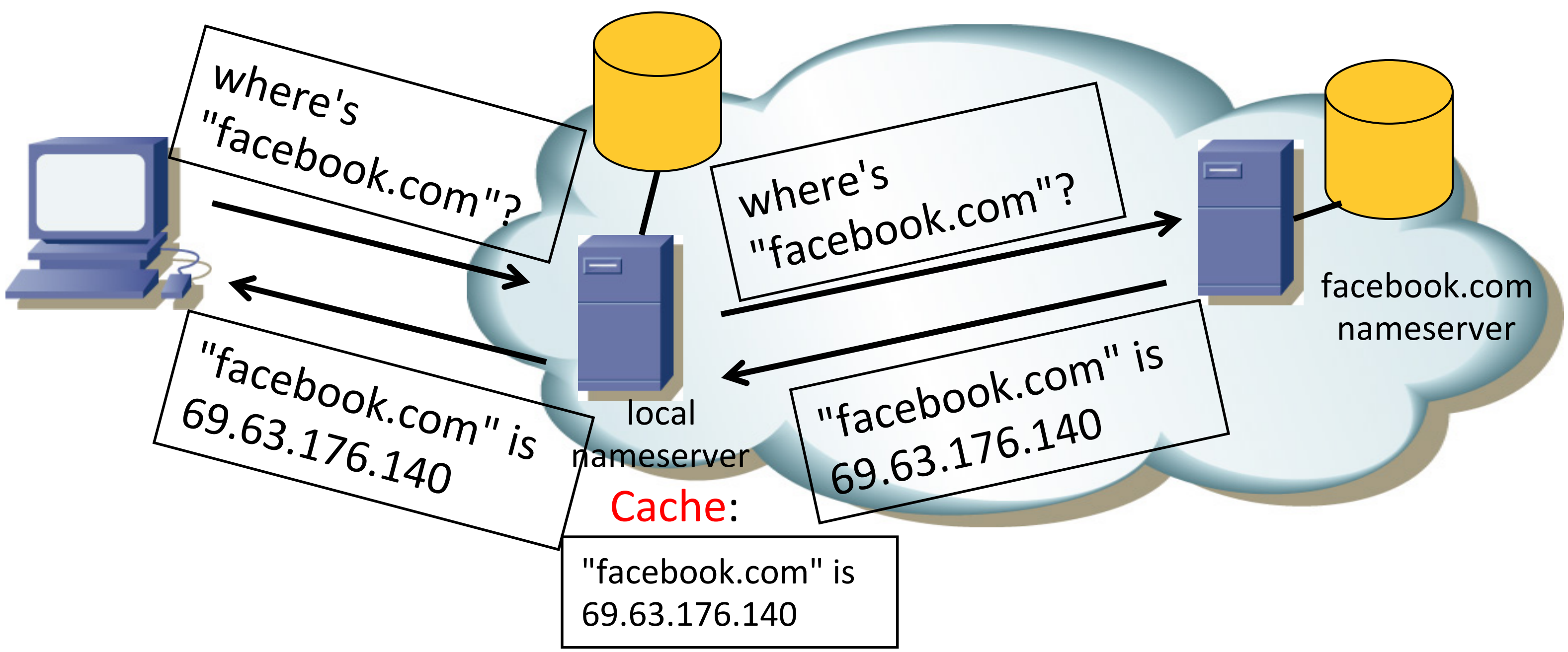
Modern Wonder: the Domain Name System

- Internet protocols deal with 32-bit addresses
  - 10000000101000111000110011011101, or
    0x80A38CDD, or 128.163.140.221
- Names are easier for humans
- Problem: design a system for resolving names to (Internet) numbers and other information

  Require the following characteristics:
    - Scalable to billions of name-number pairs
    - Distributed control: UK gets to decide what names end in "uky.edu"; CS dept gets to decide what names are under "cs.uky.edu" (but not pa.uky.edu or delta.com or...)
    - Robust: no single point of failure

Domain Name System

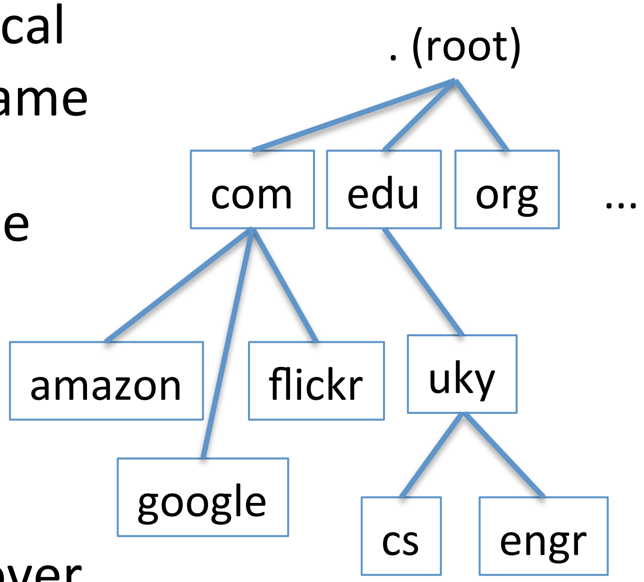
- Every URL contains a DNS name
  - http://video.google.com
  - http://protocols.netlab.uky.edu/~calvert/
- Every time you click, your computer sends a message to its local DNS server asking to resolve that DNS name to an IP address
- This happens at least millions of times per second worldwide



Scaling the DNS

The DNS depends on two standard CS techniques:

- Hierarchy
  - The namespace authority is hierarchical
  - Names in the same group have the same suffix (e.g., ".org.")
  - Each group only concerned with those below it
- Caching
  - Remember what you've learned
  - Exploit locality of reference
  - Amortize the cost of one resolution over many queries

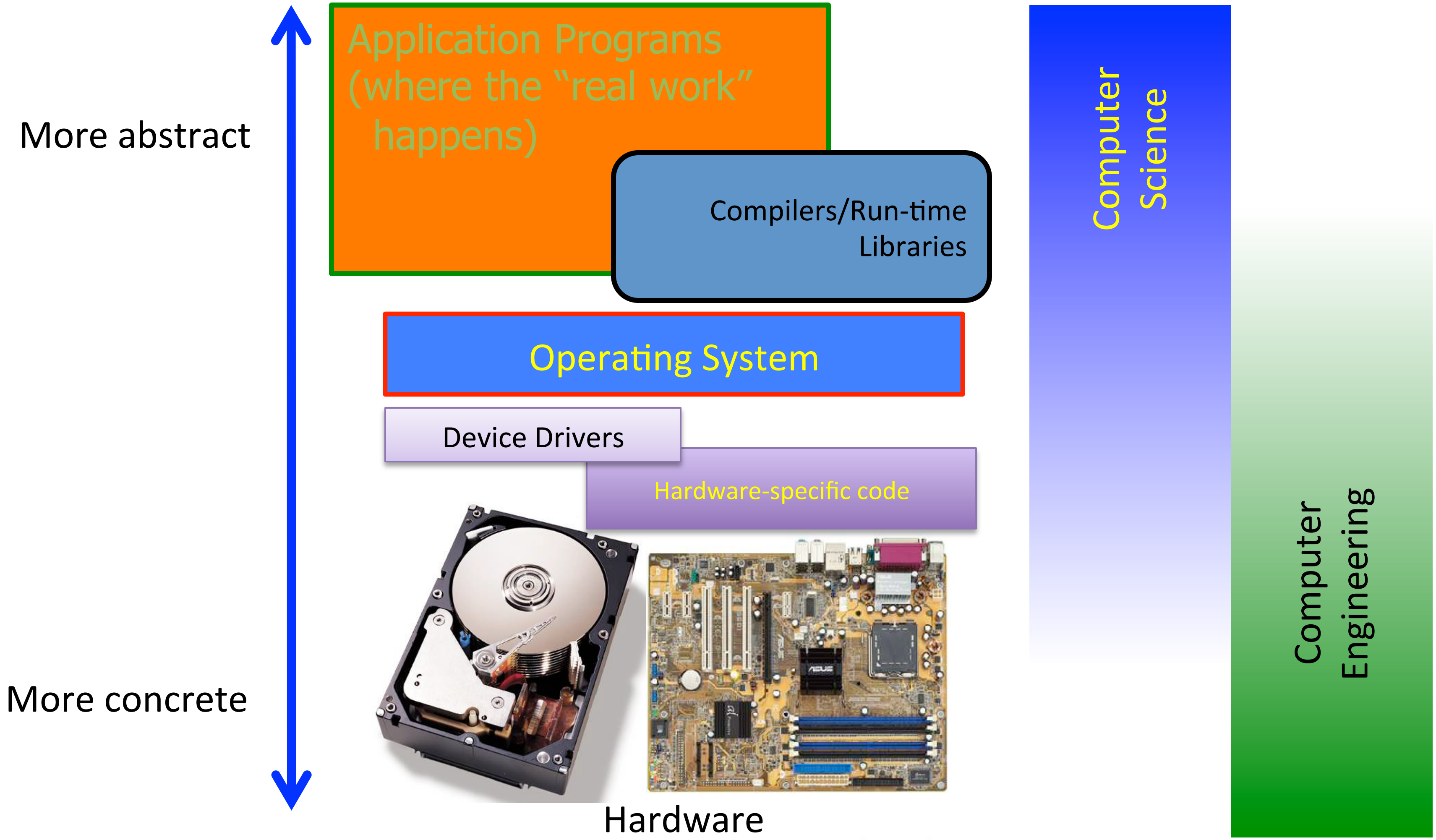


The Central Intellectual Challenge of Computing

How to keep the complexity of our systems from overwhelming us?

SCALE

Computer Science vs. Computer Engineering



Take-Aways – 1

- Computers are a "radical novelty" – unlike anything else humanity has invented.
- Computation is about symbol manipulation.
  - The Turing Machine is a simple model of a very powerful computing device.
  - Some surprisingly simple systems are capable of computing anything that can be computed
- The Church-Turing Thesis says all sufficiently-powerful computing devices can compute the same set of functions (although some may do it faster than others)

Take-Aways – 2

- There are problems that can be clearly and precisely stated, but that cannot be solved with an algorithm.

  That is: some things can't be computed.

  Halting Problem: Given a program and its input, determine whether the program ever stops when run on the input.

- Some problems are believed to not be efficiently solvable; we can only solve instances of limited size.

  "Traveling Sales Rep" is one.

Take-Aways – 3

- A key problem in CS is how to build systems that scale: grow large and still function efficiently.
- Two commonly used techniques for building scalable systems:
  - Hierarchical structure: focused responsibility and abstraction
  - Caching: saving results of a computation for later re-use, taking advantage of locality of reference
- The Central Challenge of Computing is how to manage complexity.

CS100 Action Items

- Prepare for Job Fair
- Read first chapter of “Team Geek”
- Resolve any remaining start-up issues
  - Clickers
  - Submission of assignments to portal
  - Roster / email
- Consider meeting other CS100 students for the purpose of grouping up for the project

What is Computer Science *really* about?