Introduction to Computer Science

CSCI 109

Andrew Goodney

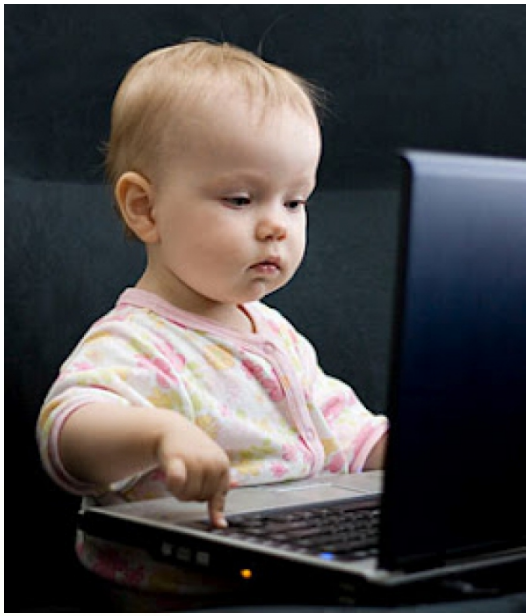
Fall 2019

Lecture 1: Introduction *August 26, 2019*

Purpose of this Course

- Introduce computer science as a discipline, a body of knowledge, and a domain of science/engineering
  - The focus is on ideas and concepts
  - Significant amounts of reading but no programming (see CSCI 103L)
- What is computing, a computer, science (and engineering)?
- How do computers work?
  - Computers, architectures, data structures and algorithms, programming, operating systems, networks, abstract machines and theory, artificial intelligence, robotics, human computer interaction, …
- What is comprised within the domain of computing?
  - Comprehending its content and structure
  - Appreciating its past, present and future
- Provide a basis upon which you can build throughout the remainder of your computing education

Purpose of this Course



Course Outline

Date	Topic		Assigned	Due	Quizzes/Midterm/Final	Slide Deck
26-Aug	Introduction	What is computing, how did computers come to be?				1
2-Sep	Labor day					
9-Sep	Computer architecture	How is a modern computer built? Basic architecture and assembly	HW1			2
16-Sep	Data structures	Why organize data? Basic structures for organizing data			**Quiz 1** on material taught in class 8/20-8/27	3
23-Sep	Data structures	Trees, Graphs and Traversals	HW2	HW1		4
30-Sep	More Algorithms/Data Structures	Recursion and run-time				5
7-Oct	Complexity and combinatorics	How "long" does it take to run an algorithm. P vs NP			**Quiz 2** on material taught in class 9/10-9/24	5
14-Oct	Algorithms and programming	Programming, languages and compilers		HW2	**Quiz 3** on material taught in class 10/1	7
21-Oct	Operating systems	What is an OS? Why do you need one?	HW3		**Quiz 4** on material taught in class 10/8	8
28-Oct	Midterm	Midterm			**Midterm** on all material taught so far.	
4-Nov	Computer networks	How are networks organized? How is the Internet organized?		HW3		9
11-Nov	Artificial intelligence	What is AI? Search, plannning and a quick introduction to machine learning			**Quiz 5** on material taught in class 10/29	10
18-Nov	The limits of computation	What can (and can't) be computed?	HW4		**Quiz 6** on material taught in class 11/5	11
25-Nov	Robotics	Robotics: background and modern systems (e.g., self-driving cars)			**Quiz 7** on material taught in class 11/12	12
2-Dec	Summary, recap, review	Summary, recap, review for final		HW4	**Quiz 8** on material taught in class 11/19	13
13-Dec	Final exam 11 am - 1 pm in SGM 123				**Final** on all material covered in the semester	

1. How many college level computer classes have you completed?

a. 0 b. 1 c. 2 d. 3 e. 4 or more

2. How many years of computer programming have you done?

a. 0 b. 1 c. 2 d. 3 e. 4 or more

3. How many programming languages do you know?

a. 0 b. 1 c. 2 d. 3 e. 4 or more

4. Are you taking CS 103 concurrently with this class?

a. Yes b. No

5. Your reason for taking CS 109 is

a. Required for your current major or minor.

b. Not required for your current major or minor but is required for a major or minor you want to add (or move to).

c. Not required for your current major or minor nor for a major or minor you are considering but you are interested in learning about Computer Science.

Survey:
https://tinyurl.com/y2cot2r5
Password: CS109Fall2019

Logistics

Instructor and TAs

- **Instructor:** Andrew Goodney
- **Office:** PHE 406
- **Office Hour:** See course website
- **Contact Info:** goodney@usc.edu

Teaching Assistants (TAs)

Max Pflueger
pflueger@usc.edu

Office: TBD
Office Hour: TBD

Artem Molchanov
molchano@usc.edu

Office: TBD
Office Hour: TBD

Note: Office hours start next week! See course website for details.

Important Info

- **Class**
  - **Location:** SGM 123 **Days & Time:** M 12:00-13:50
- **There are no discussion or quiz sections**
- **Co-requisite : CSCI 103L**
  - There is no prerequisite
- **Required Textbook**
  - *Computing for Ordinary Mortals*, St. Amant, R. Oxford University Press, 2013
- Syllabus is on http://bytes.usc.edu/cs109/
- Slides will be posted on “bytes”
- Other reading material will be made available there

Homework (30%)

- Four homeworks (7.5% each)
- Collaboration is welcome on the homework
  - But copying is not permitted
- You are allowed a total of two late days on the homework
  - One homework may be 2 days late, or two may be 1 day late, with no penalty
  - Once late days are used, one day late reduces the score by 25%, two days late reduces the score by 50%, no credit is given for three or more days late
  - All 4 homeworks must be submitted to earn a passing grade
- All homework submissions must be typed

Quizzes (5%), Midterm (30%),Final (35%)

- ~8 in-class quizzes
- No collaboration is permitted on the quizzes
- Best five scores will be retained so quizzes are worth 5% of your grade

- 1 midterm: worth 30% of your overall grade

- 1 final exam (cumulative): worth 35% of your overall grade

Quiz policy

There are **absolutely no make up quizzes**. If you need to be away from class to see a doctor, or to play on a sports team, the missed quizzes need to come from your quota of the two 'allowed misses.' Please plan on this.

If you miss quizzes earlier in the semester because you don't come to class for no good reason and then are faced with a situation later in the semester where you need to see the doctor, please do not request a medical exemption. **You should marshal the quota of 'allowed misses' carefully**.

The quizzes will be administered in class but **it is impossible to predict exactly when during the lecture they will occur**. If you come to class after the quiz for that day has been administered (or leave before it is administered), you are not entitled to a make up or to have the quiz re-administered for you.

How is the final grade assigned?

- Each homework, quiz and exam receives a raw numeric score
- Best five quiz scores are retained
- Weighted combination of raw numeric scores produces total raw score (out of 100)
- The total raw score is normalized – i.e. each total raw score is divided by the 95th percentile raw score in the class. Scores are NOT rounded.
- Recent semesters have seen the 95th percentile score be ~95%
  - This means to calculate your normalized score you divide your raw score by .95
- Grade boundaries drawn to group similar normalized scores in same final grade
  - Starting point for boundaries is: >93:A, >90:A-, >87:B+, >83:B, >80:B-, >68:C, >65:C-, >63:D+, >60:D, >55:D-
- If you “need” a particular grade for this course, the time to worry about that is now!

Other Misc. Items

- Grading disputes/reviews
  - When you homework is graded, please review it in a timely manner. If you would like clarification or review of a graded item, you may do so in one of two ways: see the TAs in office hours (preferred) or make an “instructors only” post on Piazza. Either way, you must make any requests within one week of the homework being returned.
- DSP students
  - If you have an accommodation letter from DSP, please e-mail it to me as soon as possible. Specific accommodations will be discussed in advance of the exams.
- CECS -> CS
  - If you were a CECS student, and you are switching to pure CS, please let me know (email)
- Piazza
  - https://piazza.com/usc/fall2019/csci109/
  - We use Piazza for a discussion board, please use it to ask for help with the homework or other course related questions
  - This lets all students see the responses = more efficient than e-mail!

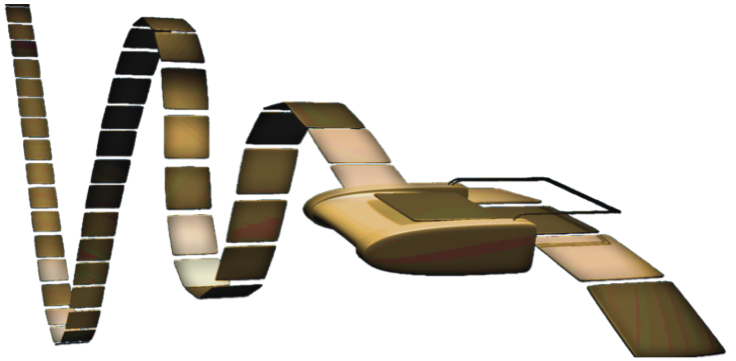
What is a Computer?

Computer or Not?

Standard Definitions (dictionary.com)

- An electronic device designed to accept data, perform prescribed mathematical and logical operations at high speed, and display the results of these operations
- A programmable machine that performs high-speed processing of numbers, as well as of text, graphics, symbols, and sound
  - All computers contain a central processing unit that interprets and executes instructions; input devices, such as a keyboard and a mouse, through which data and commands enter the computer; memory that enables the computer to store programs and data; and output devices, such as printers and display screens, that show the results after the computer has processed data
- An electronic device that stores and manipulates information
  - Unlike a calculator, it is able to store a program and retrieve information from its memory
- A machine that can be programmed to manipulate symbols
- A person who computes; computist.
  - 1640s: “one who calculates”

An information transformer



Types of Information

- Bits: 0/1, T/F, True/False, Yes/No
  - And strings of bits, such as 010110
- Numbers: 5, 101, -3, 3.14159, i, π
  - And numeric expressions, such as (3 + 2)
- Statements in logic: $\forall x\ \text{At}(x,\text{USC}) \wedge \text{Person}(x) \Rightarrow \text{Smart}(x)$
- Letters, words, sentences, paragraphs, articles, books
- Audio, image and video files
- URLs (such as http://www/google.com) and web pages
- Data bases
- …

Binary

- Modern computers use binary arithmetic
- Examples:
  - $24_{10} = 16 + 8 = 2^4 + 2^3$

    $= \mathbf{1} * 2^4 + \mathbf{1} * 2^3 + \mathbf{0} * 2^2 + \mathbf{0} * 2^1 + \mathbf{0} * 2^0$

    $= 11000_2$
  - $90_{10} = 64 + 16 + 8 + 2$

    $= \mathbf{1} * 2^6 + \mathbf{0} * 2^5 + \mathbf{1} * 2^4 + \mathbf{1} * 2^3 + \mathbf{0} * 2^2 + \mathbf{1} * 2^1 + \mathbf{0} * 2^0$

    $= 1011010_2$
  - $10111_2 = \mathbf{1} * 2^4 + \mathbf{0} * 2^3 + \mathbf{1} * 2^2 + \mathbf{1} * 2^1 + \mathbf{1} * 2^0$

    $= 16 + 4 + 2 + 1$

    $= 23_{10}$

Information Transformation

- Convert one body of information to another
  - That is, **compute**
- Example: Boolean algebra
  - Information expressed in bits: 0/1 (or F/T)
  - Operations transform input bits to yield output bits
    - AND, OR, NOT, …

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

NOT	0	1
	1	0

XOR	0	1
0	0	1
1	1	0

AND(0, 1) → 0 **OR**(0, 1) → 1

AND(1, 1) → 1 **OR**(0, 0) → 0

Information Transformation

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

NOT	0	1
	1	0

What is the truth table for f(x,y) = **AND**(**OR**(*x*, *y*), **NOT**(**AND**(*x*, *y*)))?

x	y	OR(*x*, *y*)	AND(*x*, *y*)	NOT(AND(*x*, *y*))	AND(OR(*x*, *y*), NOT(AND(*x*, *y*)))
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

f	0	1
0	0	1
1	1	0

CS Topic: representing numbers with binary

- Here is our first “real” CS topic!
- Get comfortable with looking at binary numbers!
- No hard (easier?) than base 10
- Why binary?
  - We use electronic computers (99.99999999% of us anyway)
  - Circuits can be on or off: two states -> binary representation
- Boolean operations and algebra is one way of computing with binary numbers
- First homework (and quiz) has you look at binary logic and transforming numbers base 10 <-> base 2

More on Information Transformation

- Other examples
  - Mathematical calculations – (10+2)/2=6 – and logical proofs
  - Solving puzzles
  - Sorting lists: 4, 2, 1, 3, 6, 5
  - Computational thinking

  - Transforming data into insights (*big data* or *analytics*)
  - Transforming knowledge into decisions about what actions to perform
  - Literary, musical and artistic composition
- *Hardware* enables implementing transformations
- *Software* (*programs*) control(s) transformations
- *Algorithms* are abstract descriptions of transformations

Computational Thinking

- "thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Cuny, Snyder, Wing)
  - way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science
    - To flourish in today's world, computational thinking has to be a fundamental part of the way people think and understand the world
  - creating and making use of different levels of **abstraction**, to understand and solve problems more effectively
  - thinking **algorithmically** and with the ability to apply mathematical concepts such as **induction** to develop more efficient, fair, and secure solutions
  - understanding the consequences of **scale**, not only for reasons of efficiency but also for economic and social reasons

Humans thinking (i.e., transforming information) to devise procedures for execution by information transformers (human and/or machine)

Computer or Not?

Implications

- Defining computers in terms of their functionality…
  - Strips away ancillary attributes previously thought essential
    - Machine, electronic, speed, explicit programmability, …
  - Enables appreciating the full scope of computers and computing
- Facilitates recognition of "natural" computers
  - *Brain*: Thought is preeminently information transformation
  - *Embryonic development*: Based on instructions written in DNA
  - *Evolution*: Combines and modifies information in DNA
  - *Immune system*: Includes pattern recognizers, memory, …

David Baltimore: "How biology became an information science"
Richard Dawkins: "The difference between life and non-life is a matter not of substance but of information"

Computer History

Looms, the discrete machine abstraction, and the first computer programs

A History of Human-Built Computers

- Although computers can (and have been) built using all kinds of hardware, modern computing really took off with the invention of electronic computers which were preceded by mechanical computers
- A history of computing and the Jacquard Loom
- Mechanical Computers
  - The Difference and Analytical Engines
  - The Hollerith Machine
- Electronic Computers
  - From EDSAC to the Macbook

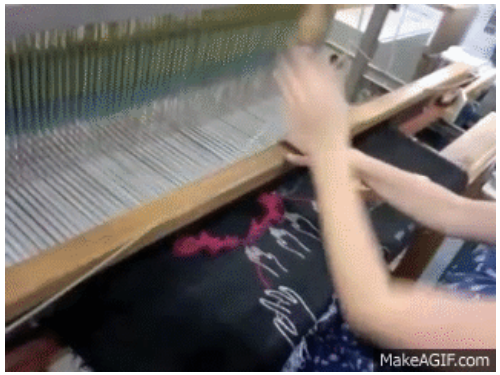
Reading:
St. Amant: Introduction, Ch. 1 and Ch. 2

Before Mechanical Computers

Electronic computers were preceded by mechanical computers and mechanical computers were preceded by…

A Simple Mechanical Loom

- Pressing treadles causes harnesses to lift threads
- The shuttle slides a cross thread under the lifted threads
- Then the threads are lowered
- Pressing treadles causes harnesses to lift threads
- The shuttle slides a cross thread under the lifted threads
- Then the threads are lowered
- Pressing treadles causes harnesses to lift threads
- The shuttle slides a cross thread under the lifted threads
- Then the threads are lowered
- …
- **What kind of patterns does this produce?**



Discrete Machines: State

- How does the loom behave as a function of time?
- At any given time a set of threads is raised and the rest are lowered
- Writing down the sequence of raised (and lowered) threads tells us the steps the machine went through to produce the cloth/tapestry/whatever
- The pattern of raised (and lowered) threads is called the *state* of the machine

CS Topic: State

- State is a very common CS concept
- Here we have the state of a physical machine
- In CS we talk about the "state" of an object
  - Of a database
  - Of a robot
  - Of a "state-machine" (finite, Turing, etc…)
  - Of a system (physical or virtual)
  - …
- Then we need a way to describe the state
  - Gives us the notion of an encoding

CS Topic: Discrete Machines, State and Encoding

- Choosing a state representation takes skill. The state should be
  - Parsimonious: it should be a “small” descriptor of what the machine is doing at any given time
  - Adequate: it should be “big enough” to capture everything “interesting” about the machine
- These are sometimes contradictory. They are also qualitative and depend on what behavior of the machine we want to describe
- Usually you need a vocabulary (*encoding*) to describe state. In the case of a loom, state can be expressed as a binary pattern (1 for raised, 0 for lowered)

Discrete Machines: Abstraction

- The loom is a *discrete machine*
  - *State is binary pattern – i.e. discrete*
  - *The notion of time is discrete – i.e. time is modeled as proceeding in steps or finite chunks*
- More precisely, *the loom can be usefully modeled as a discrete machine*
  - *Because of course being a physical device there is variation, nothing is exactly precise*
  - *But modeling the machine as discreet is good enough and works for this purpose*
- This is an example of an *abstraction* – a key concept in Computer Science

CS topic: Abstraction

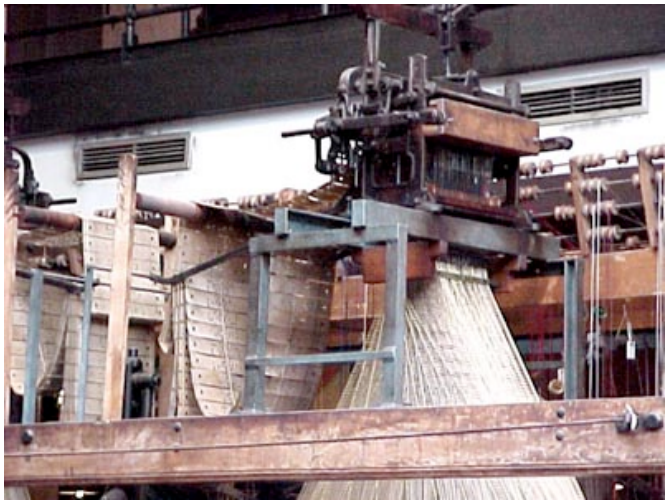
- One of the fundamental “things” we do in CS
- Reducing or distilling a problem or concept to the essential qualities
  - Simple set of characteristics that are most relevant to the problem
- Many (most, all) of what we do in engineering and computer science involves abstractions
- Here the abstraction is modelling the loom as a simple discreet state machine
  - Makes it possible to understand
  - And makes it possible to “program” the loom

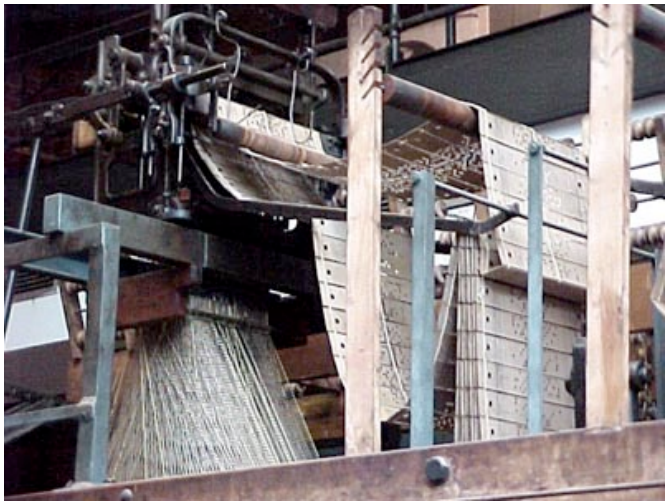
Weaving Complex Patterns

- How to produce more complex patterns?
- Early solution was human – the draw loom
  - Master weaver calls out which threads to lift
  - Drawboy lifts threads
  - Master weaver threads the shuttle
  - Master weaver calls out which threads to lift
  - Drawboy lifts threads
  - Master weaver threads the shuttle
  - Master weaver calls out which threads to lift
  - Drawboy lifts threads
  - Master weaver threads the shuttle
  - …

The Jacquard Loom (1801): Mechanism

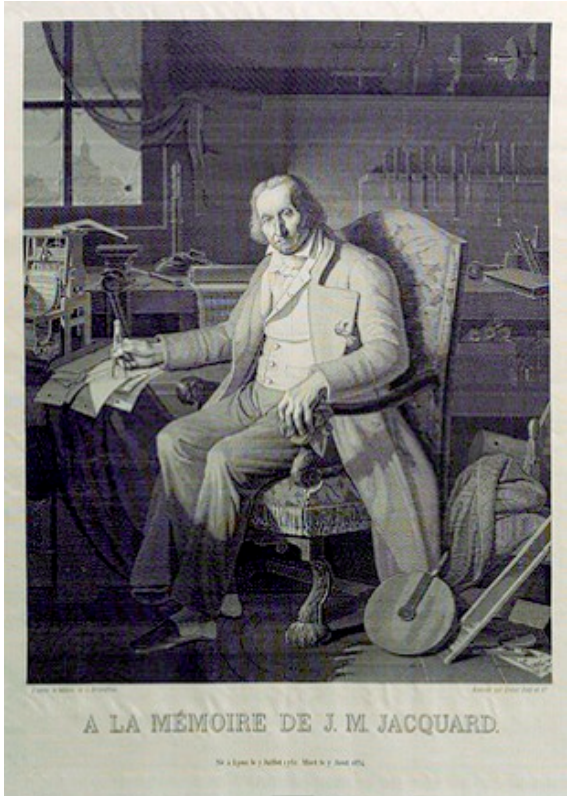
- Mechanism:
  - Threads attached to spring-loaded rods
  - Springs make all threads want to lift unless stopped somehow
  - A metal ‘card’ with holes is inserted into the path of the threads
  - A hole in the corresponding place allows a thread to lift. No hole arrests the thread motion and stops it from lifting
- The Jacquard Loom (from the Teaching Palette via YouTube)





The Jacquard Loom: Programming

- Weaving becomes the process of
  - Creating cards with holes in them (punched cards)
  - Sequencing the cards in the right order
- Each card is an *instruction* to the machine to do precisely one thing (i.e., put itself into one particular *state*)
- A sequence of cards (i.e., a sequence of instructions) causes the machine to step through a sequence of states. The card sequence is a *program*
- The weaver as a *programmer*

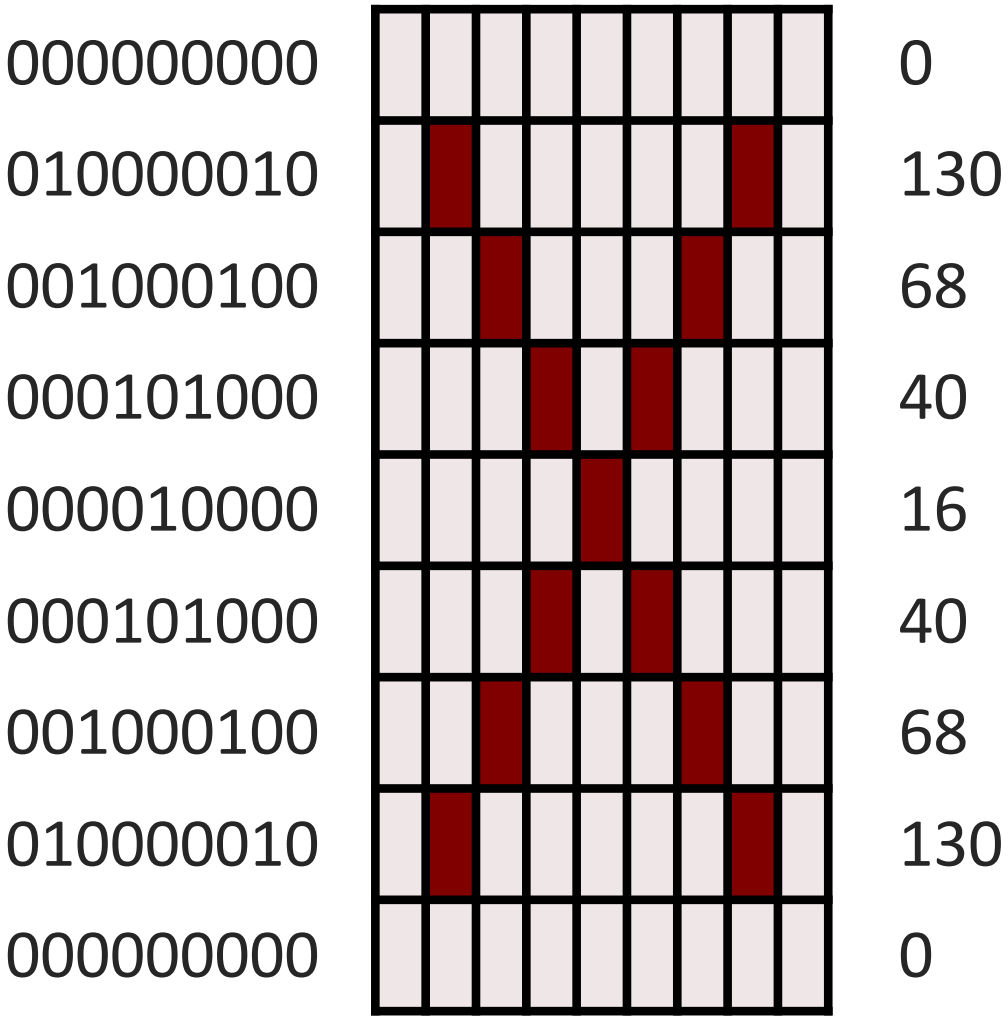


Joseph Marie Jacquard (as woven by his loom via a program of 24,000 instructions). Image courtesy of Wikipedia.

The Jacquard Loom: Programming

- A discrete, ‘automatic’ machine
- Since we have chosen a binary encoding the machine *state* is a binary number
- Each *instruction* is also a binary number since each instruction is (literally) the state the programmer wants the machine to be in when that instruction is executed
- The *program* for the machine is a sequence of instructions. Each program is (literally) a sequence of states the programmer wants the machine to step through
- The *program* is thus a sequence of binary numbers

A Loom Program



- This machine's state is captured in a 9-bit word
- Each instruction in the program is also a 9-bit word
- The state and each instruction is thus 9 bits wide
- This program is 9 instructions long

Loom Program Limitations

- Does not scale:
  - Large instructions: To program a ‘big’ machine you need ‘big’ words (large state implies large instruction widths)
  - No counting: No repeats or loops to do things over a certain number of times (No “do..while” or “repeat..until”)
  - No modularity: No logical chunks for sub-patterns that can be reused without replication (No “functions, methods, subroutines…”)
- No decision making on the fly:
  - No branching to decide to do one thing instead of another based on a condition (No “if-then-else”. No jumps or “goto”)
- So is the Jacquard loom a computer?

How Big/Fast is a Modern Computer?

- Typical Macbook Pro laptop has ~1 billion transistors
- The state of the machine is a binary number with ~1 billion bits (a binary word of width ~1 billion)
  - Not possible to program a Macbook Pro by writing a sequence of instructions each ~1 Billion bits wide
- A Macbook Pro executes ~5 billion instructions per second
  - Possible to have programs that are billions of instructions long and yet have them finish operating in a reasonable time

Modern Computers?

- If modern computers are so big, how do we program them?
- We model (abstract) the computer as something more simple
- Program to that model
- Then let the hardware and OS sort out the difference between reality and our abstraction
- This method of problem solving is very, very common in CS
- More later in semester

Computer History

Mechanical computers: The difference and analytical engines, the Hollerith machine

The Difference Engine (1822)

Video from YouTube.

- *Charles Babbage*
- Mechanical calculator to compute mathematical tables
  - Loom: program transforms threads to patterns on cloth
  - Difference engine: program transforms numbers into other numbers
- Polynomial function computation using differences
- Output was via a 'printer' – a device that produced printer's plates so they could be stamped onto paper
- No branching or looping, limited in what it could compute

The Analytical Engine (1837)

- *Charles Babbage*

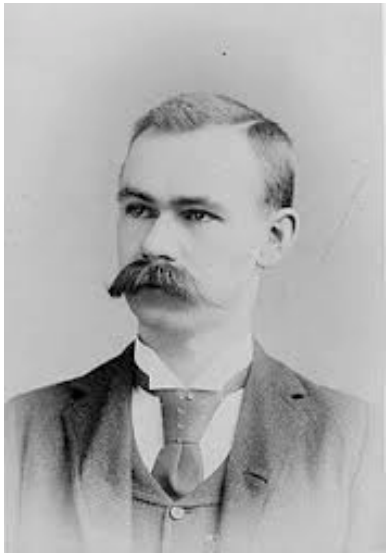
Video from YouTube.

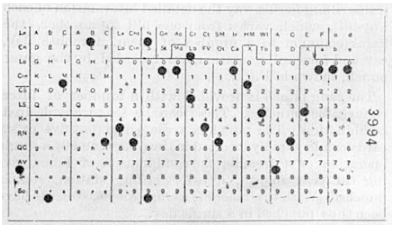
- World's first general purpose mechanical calculator
  - Memory
  - Arithmetic unit
  - Branching
  - Looping
- Programmed by punched cards like a loom
- Output was via a 'printer' – a device that produced printer's plates so they could be stamped onto paper

The Hollerith Tabulator (1890)

- *Hermann Hollerith*
- First device to read **data** into a machine: an electromechanical system based on punched cards
- Built to tabulate the results of US census
- Hollerith's contributions to modern computing are... "incalculable"
  - He did not stop at his original 1890 tabulating machine and sorter, but produced many other innovative new models. He also invented the first automatic card-feed mechanism, the first key punch, and took what was perhaps the **first step towards programmin**g by introducing a wiring panel in his 1906 Type I Tabulator, allowing it to do different jobs without having to be rebuilt! (The 1890 Tabulator was hardwired to operate only on 1890 Census cards.) These inventions were the foundation of the modern information processing industry.
- Hollerith went on to form the Tabulating Machine Company. Merged with others to form the Computing Tabulating Recording Company (CTR). Renamed in 1924 to

International Business Machines (IBM)

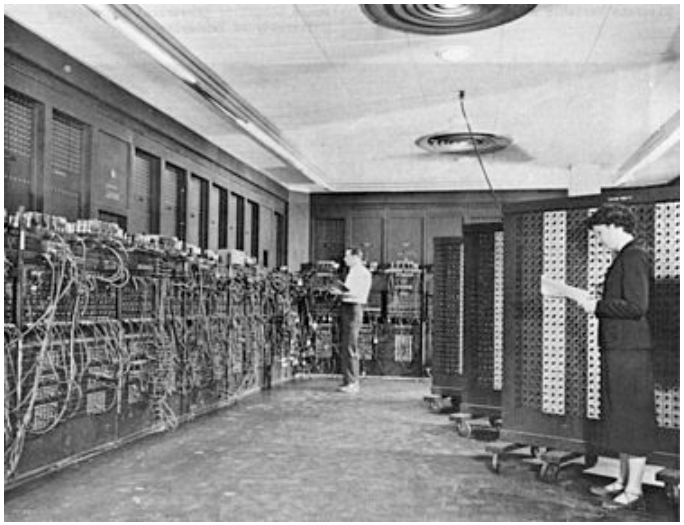


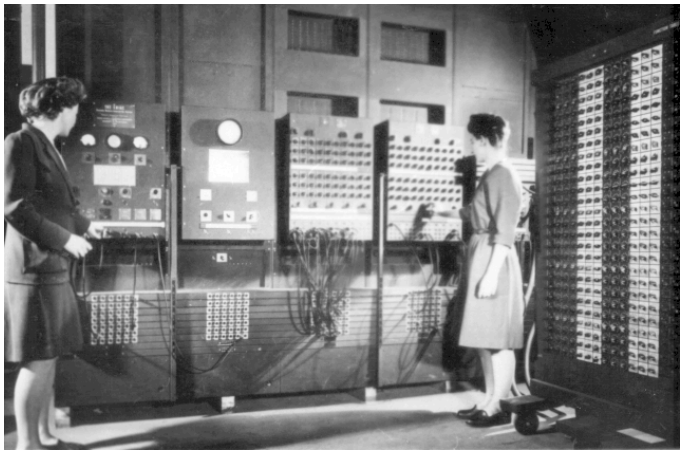


Hermann Hollerith and one of his punched cards. Images courtesy of Wikipedia.

The ENIAC (1943-46)

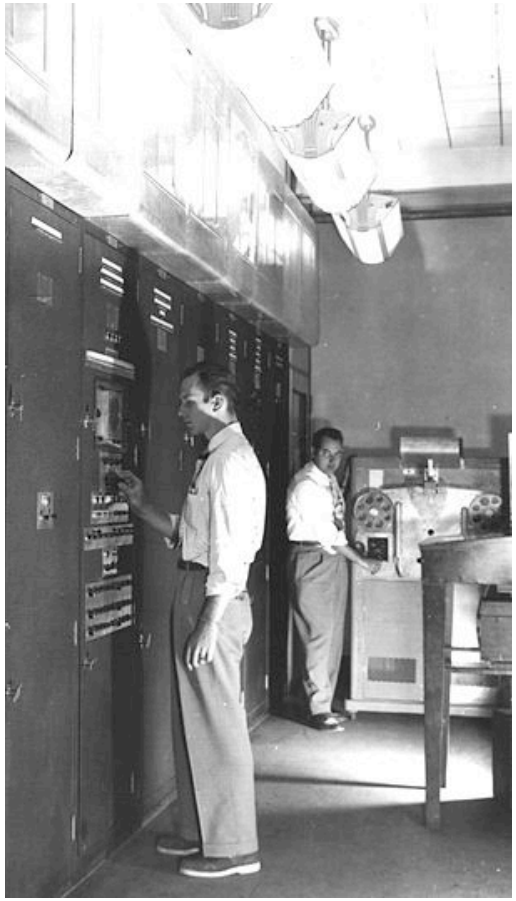
- Electronic Numerical Integrator And Computer
- *Eckert and Mauchly (University of Pennsylvania)*
- First electronic, general purpose computer. Turing-complete, digital, reprogrammable (cumbersome)
- Vacuum tube and diode-based





The EDVAC (1944-49)

- Electronic Discrete Variable Automatic Computer (*Eckert and Mauchly)*
- First stored program computer, binary (ENIAC was decimal). Operational 1951.
- Popularized by *von Neumann (*First Draft of a Report on the EDVAC) – first report on a modern computer architecture



Computer History Summary

- Although computers can (and have been) built using all kinds of hardware, modern computing really took off with the invention of electronic computers which were preceded by mechanical computers
- A history of computing and the Jacquard Loom
- Mechanical Computers
  - The Difference and Analytical Engines
  - The Hollerith Machine
- Electronic Computers
  - From EDSAC to the Macbook

Fundamental Concepts

- State and discrete machines
- Abstraction and models
- Encoding data and instructions
- Programming
- To be general, programs need to access a memory, and to be able to control the order of the instructions to execute based on the results of computation

Review of terms

- State: The condition of a system at a point in time
- Encoding: Symbolic expression used to represent information
- Discrete: Proceeding in finite steps, individually separate and distinct
- Binary: Numerical notation that uses base 2
- Abstraction: Simplified (“higher-level”) description

Next time: Computer Architecture

How are computers built?