CMPE 322/327 - Theory of Computation

Week 0: History of Computing – An Overview

Burak Ekici

February 14–18, 2022

About Me: Parcours/Carrier

Undergrad	IZTECH	CE	2004-2009, İzmir
Master's	Yaşar Üni	CE	2009-2012, İzmir
Traineeship	EC JRC	CE	2010-2011, Varese-Italy
PhD	U Joseph Fourier	CS & Math	2013-2015, Grenoble-France
PostDoc	U of Iowa	CS	2016-2017, IA-USA
PostDoc	U of Innsbruck	CS	2018-2019, Innsbruck-Austria
Assist. Prof. Dr.	Kültür Üni	CE	2019-2020, İstanbul
Assist. Prof. Dr.	TED Üni	CE	2021-now, Ankara

Logistics

lecturer	Burak Ekici (burak.ekici@tedu.edu.tr)

Logistics

lecturer	Burak Ekici (burak.ekici@tedu.edu.tr)
teaching assistants	Merve Işıl Peten (misil.peten@tedu.edu.tr)
	Ali Egemen Taşören (egemen.tasoren@tedu.edu.tr)

Logistics

lecturer	Burak Ekici (burak.ekici@tedu.edu.tr)
teaching assistants	Merve Işıl Peten (misil.peten@tedu.edu.tr)
	Ali Egemen Taşören (egemen.tasoren@tedu.edu.tr)
consultation	Thursday 13h30 – 16h30 at A224

About the Course

- Prerequisites:
  - Strong motivation

About the Course

- Prerequisites:
  - Strong motivation
- Text Book:
  - Automata and Computability, Dexter C Kozen, Springer-Verlag, 1997

About the Course

- Prerequisites:
  - Strong motivation
- Text Book:
  - Automata and Computability, Dexter C Kozen, Springer-Verlag, 1997
- Additional References:
  - Introduction to Automata Theory, Languages, and Computation, 3rd Edition, Hopcroft, Motwani, Ullman
  - Introduction to the Theory of Computation, Michael Sipser
  - An Introduction to Formal Languages and Automata, Peter Linz

About the Course

- Prerequisites:
  - Strong motivation
- Text Book:
  - Automata and Computability, Dexter C Kozen, Springer-Verlag, 1997
- Additional References:
  - Introduction to Automata Theory, Languages, and Computation, 3rd Edition, Hopcroft, Motwani, Ullman
  - Introduction to the Theory of Computation, Michael Sipser
  - An Introduction to Formal Languages and Automata, Peter Linz
- Tentative Grading:

~~Attendance~~	Quizzes & Homeworks	Midterm	Final
~~5%~~	35%	30%	35%

About the Course (cont'd: Goals – Roughly)

- Provide computation models, and analyze their power in terms of computability

About the Course (cont'd: Goals – Roughly)

- Provide computation models, and analyze their power in terms of computability
  - What computational problems can a model solve?

About the Course (cont'd: Goals – Roughly)

- Provide computation models, and analyze their power in terms of computability
  - What computational problems can a model solve?
- Answer complexity questions:
  - What can a computer do efficiently? How much time we need to solve the problems?

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\Longleftrightarrow$ NFAs

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\iff$ NFAs
Week 5	DFA State Minimization & Myhill-Nerode Relations

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\Longleftrightarrow$ NFAs
Week 5	DFA State Minimization & Myhill-Nerode Relations
Week 6	Derivatives & Kleene Algebra & Equivalence of REGEXPs

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\Longleftrightarrow$ NFAs
Week 5	DFA State Minimization & Myhill-Nerode Relations
Week 6	Derivatives & Kleene Algebra & Equivalence of REGEXPs
Week 7	Midterm

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\iff$ NFAs
Week 5	DFA State Minimization & Myhill-Nerode Relations
Week 6	Derivatives & Kleene Algebra & Equivalence of REGEXPs
Week 7	Midterm
Week 8	NFA/DFA Equivalence Checking & Context Free Grammars & Context Free Languages & Ambiguity

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\iff$ NFAs
Week 5	DFA State Minimization & Myhill-Nerode Relations
Week 6	Derivatives & Kleene Algebra & Equivalence of REGEXPs
Week 7	Midterm
Week 8	NFA/DFA Equivalence Checking & Context Free Grammars & Context Free Languages & Ambiguity
Week 9	Chomsky Normal Form & Pumping Lemma & CKY Algorithm

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\iff$ NFAs
Week 5	DFA State Minimization & Myhill-Nerode Relations
Week 6	Derivatives & Kleene Algebra & Equivalence of REGEXPs
Week 7	Midterm
Week 8	NFA/DFA Equivalence Checking & Context Free Grammars & Context Free Languages & Ambiguity
Week 9	Chomsky Normal Form & Pumping Lemma & CKY Algorithm
Week 10	Ogden's Lemma & Push Down Automata

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\iff$ NFAs
Week 5	DFA State Minimization & Myhill-Nerode Relations
Week 6	Derivatives & Kleene Algebra & Equivalence of REGEXPs
Week 7	Midterm
Week 8	NFA/DFA Equivalence Checking & Context Free Grammars & Context Free Languages & Ambiguity
Week 9	Chomsky Normal Form & Pumping Lemma & CKY Algorithm
Week 10	Ogden's Lemma & Push Down Automata
Week 11	Turing Machines & Recursive(ly Enumerable) Languages & Decision Problems

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\iff$ NFAs
Week 5	DFA State Minimization & Myhill-Nerode Relations
Week 6	Derivatives & Kleene Algebra & Equivalence of REGEXPs
Week 7	Midterm
Week 8	NFA/DFA Equivalence Checking & Context Free Grammars & Context Free Languages & Ambiguity
Week 9	Chomsky Normal Form & Pumping Lemma & CKY Algorithm
Week 10	Ogden's Lemma & Push Down Automata
Week 11	Turing Machines & Recursive(ly Enumerable) Languages & Decision Problems
Week 12	Halting Problem & Reduction

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\iff$ NFAs
Week 5	DFA State Minimization & Myhill-Nerode Relations
Week 6	Derivatives & Kleene Algebra & Equivalence of REGEXPs
Week 7	Midterm
Week 8	NFA/DFA Equivalence Checking & Context Free Grammars & Context Free Languages & Ambiguity
Week 9	Chomsky Normal Form & Pumping Lemma & CKY Algorithm
Week 10	Ogden's Lemma & Push Down Automata
Week 11	Turing Machines & Recursive(ly Enumerable) Languages & Decision Problems
Week 12	Halting Problem & Reduction
Week 13	Rice's Theorem & Unrestricted Grammars

Syllabus & Tentative Schedule

Week 0	History of Computing – An Overview
Week 1	Central Concepts and Mathematical Preliminaries
Week 2	Deterministic Finite Automata (DFA) & Regular Languages & Closure Properties
Week 3	Non-Deterministic Finite Automata (NFA) & ε-transitions
Week 4	Pattern Matching & Regular Expressions (REGEXP) & REGEXP $\iff$ NFAs
Week 5	DFA State Minimization & Myhill-Nerode Relations
Week 6	Derivatives & Kleene Algebra & Equivalence of REGEXPs
Week 7	Midterm
Week 8	NFA/DFA Equivalence Checking & Context Free Grammars & Context Free Languages & Ambiguity
Week 9	Chomsky Normal Form & Pumping Lemma & CKY Algorithm
Week 10	Ogden's Lemma & Push Down Automata
Week 11	Turing Machines & Recursive(ly Enumerable) Languages & Decision Problems
Week 12	Halting Problem & Reduction
Week 13	Rice's Theorem & Unrestricted Grammars
Week 14	Final

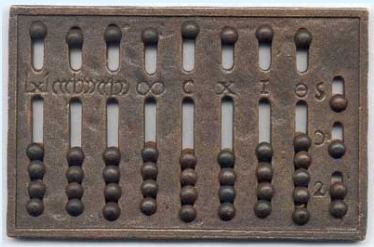
The idea of computing

... is not new; dates to 500 BC. Could be studied in mainly three categories in terms of time:

- Mechanical Era
- Electronic Era
- Microprocessors Era

Mechanical Era

- The era of computing devices before the invention of electricity.
- Babylonians invented the abacus to keep track of their vast storehouses of grain.
- It has been used extensively and is still in use today.



Pascal's Calculator

- constructed of gears and wheels
- has 10 teeth that, when moved one complete revolution, advanced a second gear one place.
- used to add and subtract two numbers directly and perform multiplication and division via repeated addition and subtractions

Pascal's Calculator

- constructed of gears and wheels
- has 10 teeth that, when moved one complete revolution, advanced a second gear one place.
- used to add and subtract two numbers directly and perform multiplication and division via repeated addition and subtractions

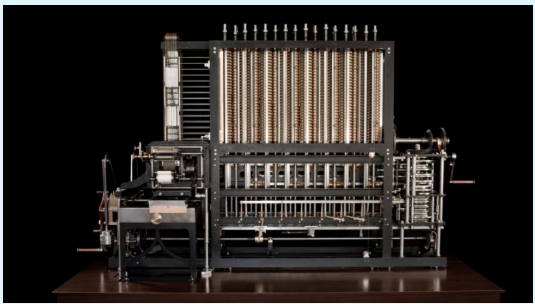
Incidentally, the PASCAL programming language is named in honor of Blaise Pascal for his pioneering work in mathematics and with the mechanical calculator

The Difference Engine

Designed by Charles Babbage at around 1820 to tabulate polynomial functions. I.e., helps build trigonometric tables (sin, cosin, etc.), logarithmic tables, etc.

The Difference Engine

Designed by Charles Babbage at around 1820 to tabulate polynomial functions. I.e., helps build trigonometric tables (sin, cosin, etc.), logarithmic tables, etc.



The Difference Method

x	$p(x) = 2x^2 - 3x + 2$	$\text{diff1}(x) = (p(x + 1) - p(x))$	$\text{diff2}(x) = (\text{diff1}(x + 1) - \text{diff1}(x))$
0	2	−1	4
1	1	3	4
2	4	7	4
3	11	11	
4	22		

The idea:

- Compute once the values at the first row
  - which necessitates $p(0), p(1)$ and $p(2)$ to be calculated.
- Observe that the value $diff2(x)$ remains unchanged for any x.
- To compute the $n + 1^{th}$ column at row 2, one needs to add the values stored in n^{th} column rows 2 and 3.
  - This could be performed ad infinitum.

Mechanical vs Electronic devices

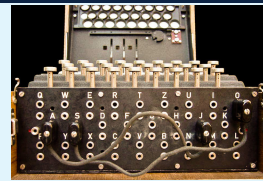
Mechanical devices	Electronic Devices
• use mechanical energy in the form of wheels, gears, levers, etc., to work	• use the energy of electricity to do work
• function involving physical movements of some parts or components	• can function without involving any moving components (motionless, so to speak)
• output is the motion itself	• output comes in different forms such as sound, heat, light, etc

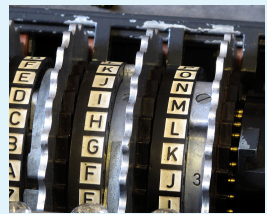
Enigma: an Electro-mechanical Device

- An enciphering machine used by the German armed forces to exchange messages securely during the Word War I (only by the end) and the Word War II.
- Works under some initial configuration to be shared between parties.

Enigma: an Electro-mechanical Device (cont'd)

- Two stages of shuffling:
  - includes ten different wires connecting pairs of letters so as to be swapped over on an occurrence. I.e, if A is connected to L the input text APPLE would be considered as LPPAE
  - each time a letter is typed the outermost rotor turns once, and outputs an encrypted letter. After several turns, it hits the second innermost rotor and forces it to turn once as well
  - the same scenario applies to the relation between the second innermost and the innermost rotors

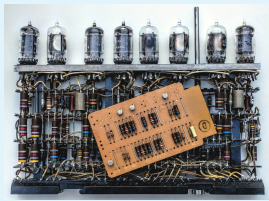




All these complications sum up to 158,962,555,217,826,360,000 possibilities when it comes to break a cipher.

Electronic Era

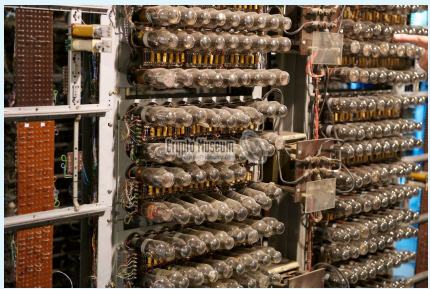
- After the invention of the electric motor (due to Faraday), dozens of motor-driven difference/adding machines came up.
- The principle of these machines is similar to that of Pascal's calculator but wheels and gears are now driven by electric motors and controlled by vacuum tubes
- Vacuum tubes are used to to amplify or switch the electronic signals. This conceptually is the same job done by transistors appear in modern computers
  - a two element electronic tube (diode): (1) anode, (2) cathode insulated by a vacuum
  - powering on the cathode would make electrons flow in the vacuum and absorbed by the anode - the closed circuit mode
  - powering off the cathode - the open circuit mode



- They are much bigger and get hot pretty sooner in comparison with transistors

Colossus

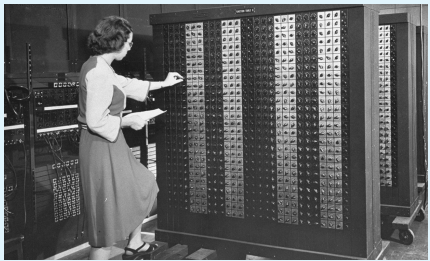
- The world's first electronic computer
- the sole purpose is to help decipher the Lorenz-encrypted messages between German generals during World War II.
- The cipher text was input via paper tape and the 2500 thermionic valve (vacuum tubes) of Colossus would find the corresponding plain text.



NB: the Enigma ciphers were not broken with not Colossus but with the machine called `Bombe`.

Electronic Numerical Integrator and Computer (ENIAC)

- The first general purpose, boot up in 1945, programmable electronic computer.
- It is able to solve "a large class of numerical problems".
  - Could be programmed to perform operations including loops, branches, and subroutines.



- However, instead of the stored-program computers that exist today, ENIAC was just a large collection of arithmetic machines, which originally had programs set up into the machine by a combination of plugboard wiring and three portable function tables (containing 1200 ten-way switches each).

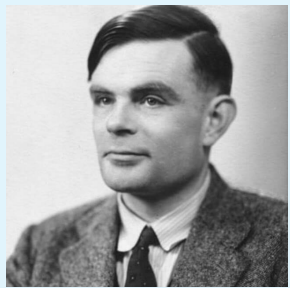
Alan Turing: the father of computing

Neither Colossus nor ENIAC stored programs in memory.

Alan Turing: the father of computing

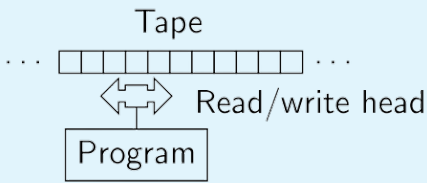
Neither Colossus nor ENIAC stored programs in memory.

- The basic principle of the modern computer the idea of controlling the machine's operations by means of a program of coded instructions stored in the computer's memory was conceived by **Alan Turing**.



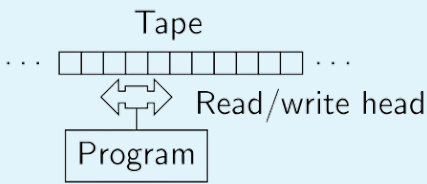
Alan Turing: the father of computing (cont'd)

He came up with a mathematical model of computation that defines an abstract machine, later named Turing Machine, which manipulates symbols on a strip of tape according to a program (a table of rules) stored in memory.



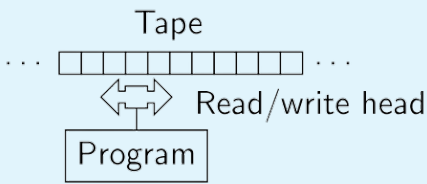
Alan Turing: the father of computing (cont'd)

He came up with a mathematical model of computation that defines an abstract machine, later named Turing Machine, which manipulates symbols on a strip of tape according to a program (a table of rules) stored in memory.



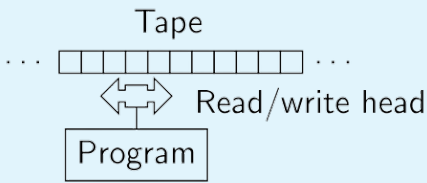
Alan Turing: the father of computing (cont'd)

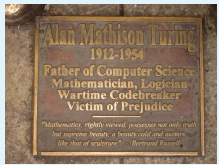
He came up with a mathematical model of computation that defines an abstract machine, later named Turing Machine, which manipulates symbols on a strip of tape according to a program (a table of rules) stored in memory.



Alan Turing: the father of computing (cont'd)

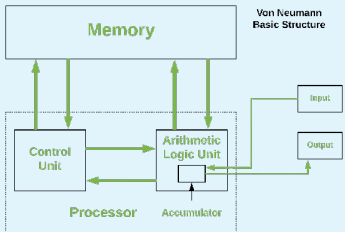
He came up with a mathematical model of computation that defines an abstract machine, later named Turing Machine, which manipulates symbols on a strip of tape according to a program (a table of rules) stored in memory.





John von Neumann: the (older) uncle of computing

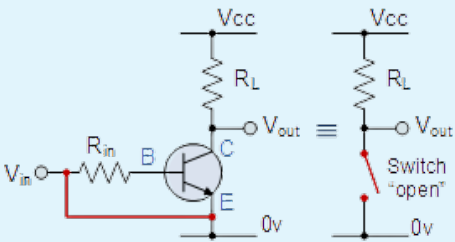
- von Neumann had a similar idea.
- His *EDVAC (Electronic Discrete Variable Computer)* has been designed in a way that programs could electronically be stored in memory.



Microprocessors Era

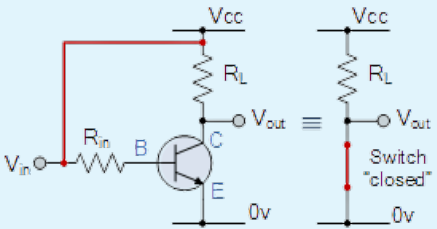
- Vacuum tubes are replaced with transistors.
- A transistor is a semiconductor device mainly used to switch electronic signals and electrical power.
- Austro-Hungarian physicist Julius Edgar Lilienfeld proposed the concept of a field-effect transistor in 1926, but he could not make it work.
- In 1947, American physicists John Bardeen and Walter Brattain invented the first working transistor, and shared the Nobel Prize in 1956.

NPN Transistor: Cut-Off Characteristics



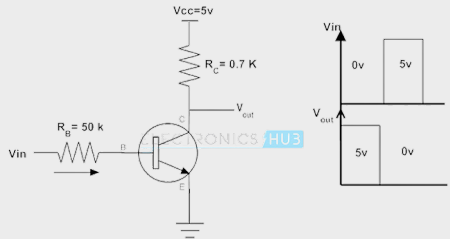
- The input and Base are grounded (0V)
- Base-Emitter voltage $V_{BE} < 0.7V$
- Transistor is “fully-OFF”
- No Collector current flows ($I_C = 0$)
- Transistor operates as an “open switch”
- $V_{out} = V_{CE} = V_{CC}$. Logical “1”...

NPN Transistor: Saturation Characteristics



- The input and Base are connected to V_{CC}
- Base-Emitter voltage $V_{BE} > 0.7V$
- Transistor is “fully-ON”
- Maximum Collector current flows ($I_C = V_{CC}/R_L$)
- Transistor operates as a “closed switch”
- $V_{out} = V_{CE} = 0V$. Logical “0”...

Example (NPN Transistor as a Switch)



$V_{in} = 0V$

$I_C = V_{CC}/R_C = 5V/0.7K$

$= 7.1mA$

$I_B = I_C/\beta = 7.1mA/100$

$= 71\mu A$ (peak value of the collector current)

$V_{in} = 5V$

$I_B = V_{in} - V_{BE}/R_B = (5V - 0.7V)/50K$

$= 86\mu A$

β is the current gain in the *grounded emitter* configuration, and V_{BE} is 0.7V for silicon transistors.

Microprocessors

A microprocessor is a programmable device that performs arithmetic and logical operations on some input data and outputs a result.

Microprocessors

A microprocessor is a programmable device that performs arithmetic and logical operations on some input data and outputs a result.

- programmable device: it can perform different behavior, on the data, provided by a set of instructions (or a program).

Microprocessors

A microprocessor is a programmable device that performs arithmetic and logical operations on some input data and outputs a result.

- programmable device: it can perform different behavior, on the data, provided by a set of instructions (or a program).
- arithmetic and logical operations: involves instructions such as multiplication, division, AND, OR, XOR, etc.

Microprocessors

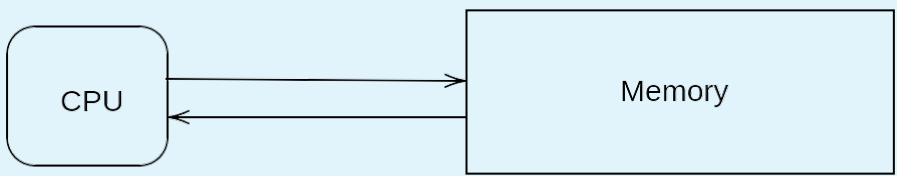
A microprocessor is a programmable device that performs arithmetic and logical operations on some input data and outputs a result.

- programmable device: it can perform different behavior, on the data, provided by a set of instructions (or a program).
- arithmetic and logical operations: involves instructions such as multiplication, division, AND, OR, XOR, etc.
- input data: is in binary form.

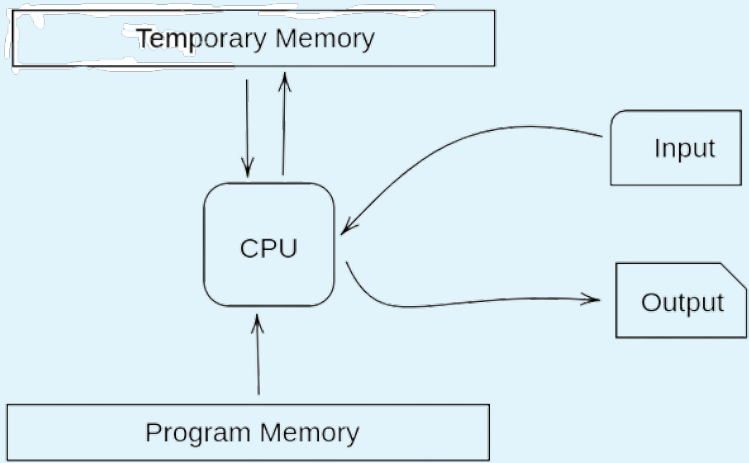
It contains a number of transistors connected by wires.

Computing In General

- A widely accepted model of computation

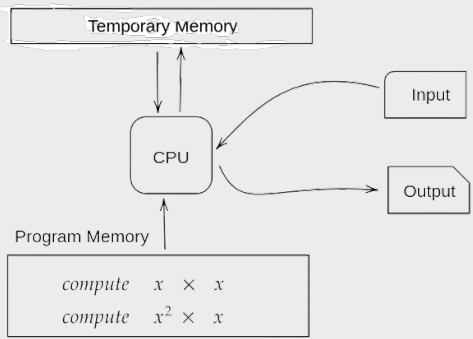


Different Components of Computer Memory



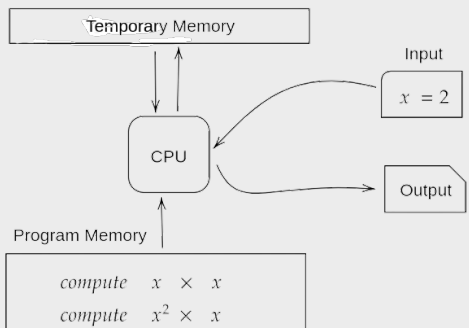
Example (Computation)

$f : \mathbb{Z} \to \mathbb{Z}$
$f := x^3$



Example (Computation)

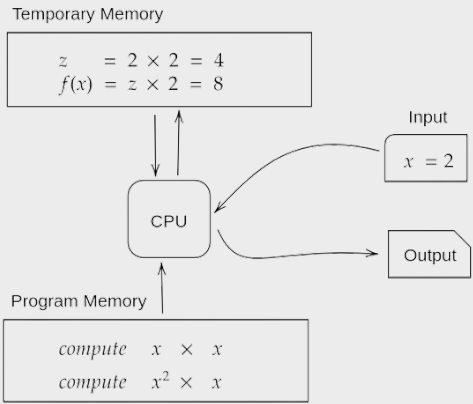
$f \quad : \quad \mathbb{Z} \to \mathbb{Z}$
$f \quad := \quad x^3$



Example (Computation)

$f : \mathbb{Z} \to \mathbb{Z}$

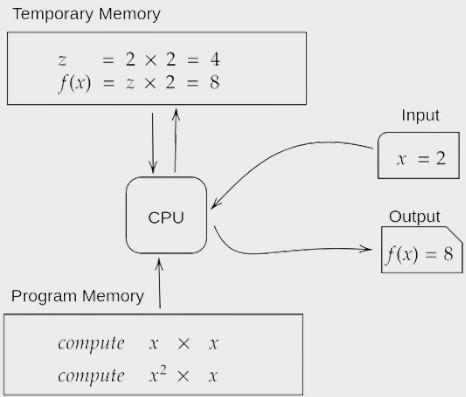
$f := x^3$



Example (Computation)

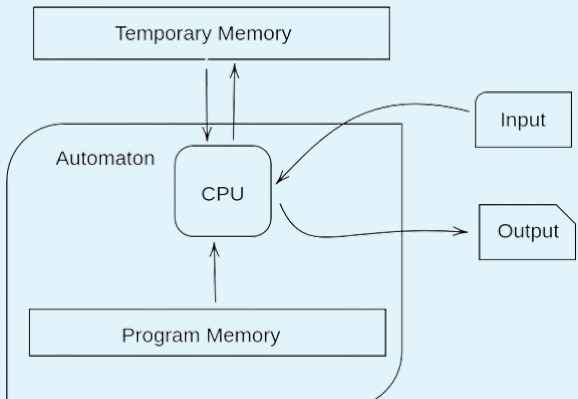
$f \quad : \quad \mathbb{Z} \to \mathbb{Z}$

$f \quad := \quad x^3$



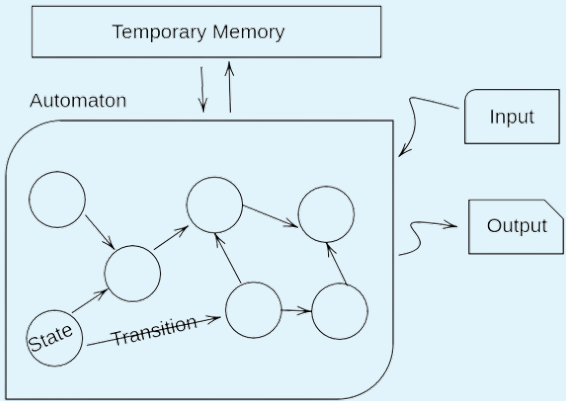
Automaton

- CPU + Program Memory = States + Transitions



Automaton

- CPU + Program Memory = States + Transitions



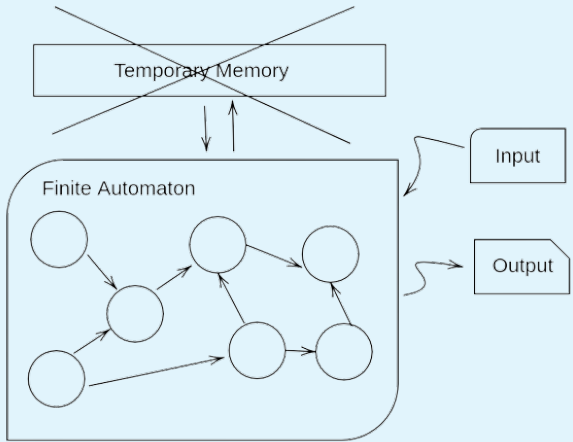
Different Kinds of Automata

- Automata are distinguished by the temporary memory they employ:

Finite Automata	no temporary memory
Pushdown Automata	stack
Turing Machines	random access memory

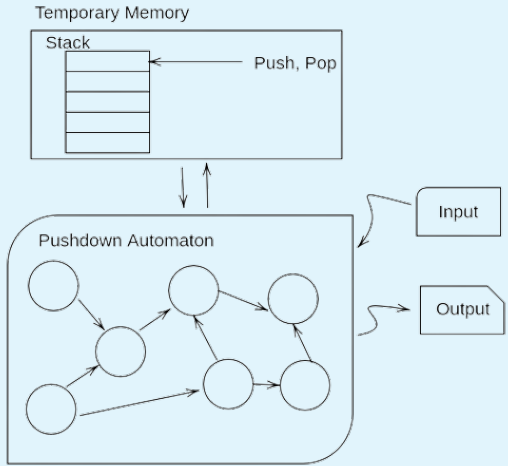
- Memory use designates computational power: the use of more flexible memory brings out the ability to solve more (powerful) computational problems

Finite Automaton



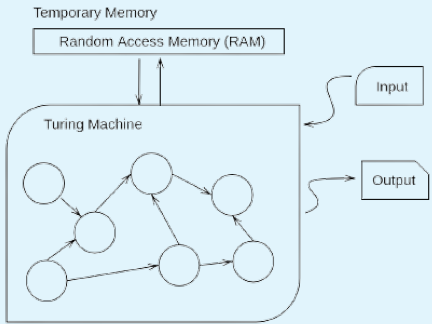
- Examples: Vending Machines, Lexical Analyzers (small computing power)

Pushdown Automaton



- Examples: Parsers for Programming Languages (medium computing power)

Turing Machine



- Examples: Any Algorithm (highest known computing power)

Power of Automata

Simple Problems		More Complex Problems		Hardest Problems
Finite Automata	$\langle$	Pushdown Automata	$\langle$	Turing Machines
Less Power		$\longrightarrow$		More Power

Thanks! & Questions?