Universal Computing Devices

Turing Machine: Any computable computation can be implemented by some machine

- Read/write "one" symbol at a time
- Select one action from fixed set of rules

Universal Turing Machine: One that can "simulate" any other Turing Machines by inputting a decription within others' rules

☆☆ This is just an idea, not physical machines

Ex. [A computer is a UTM (minus infinite memory)]

☆☆ Theory: A computer can compute anything given enough memory and time

Issues Raised : Computation is limited to constraints

Time / Cost / Power

"Speed" of Computers given enough time :

1、 Faster "clock"
2、 More parallel structures
3、 More complicated hardware
4、 Faster / closer / Larger memory unit

☆☆☆ Time + Cost + Power are 3 major constraints affecting computing

↓

Design trade-offs

Trends and Increasing Complexity

1965 Moore's Law: Doubling capacity every 2 years
(Held True)

Corollary to the law: Cost halves every 2 years

Example Computing system:

Processor: Computes the instructions
DRAM: Temporary information during a computation
Flash storage: Retains information (even power off)

Processor: Executes applications that have been broken down into sequences of simple instructions

E.g: Add value in location 1 to location 2, put the result in location 3.

Transistors : Think of "switches"

↓

(Electricity flows in different ways)

→ They are really "small" !!!

Question : How to create programs that controls these many transistors ?

↓

Abstraction

Abstraction Layers

Computers can be broadly categorized as "Hardware" and "software"

Description of required behaviors in UTM theory

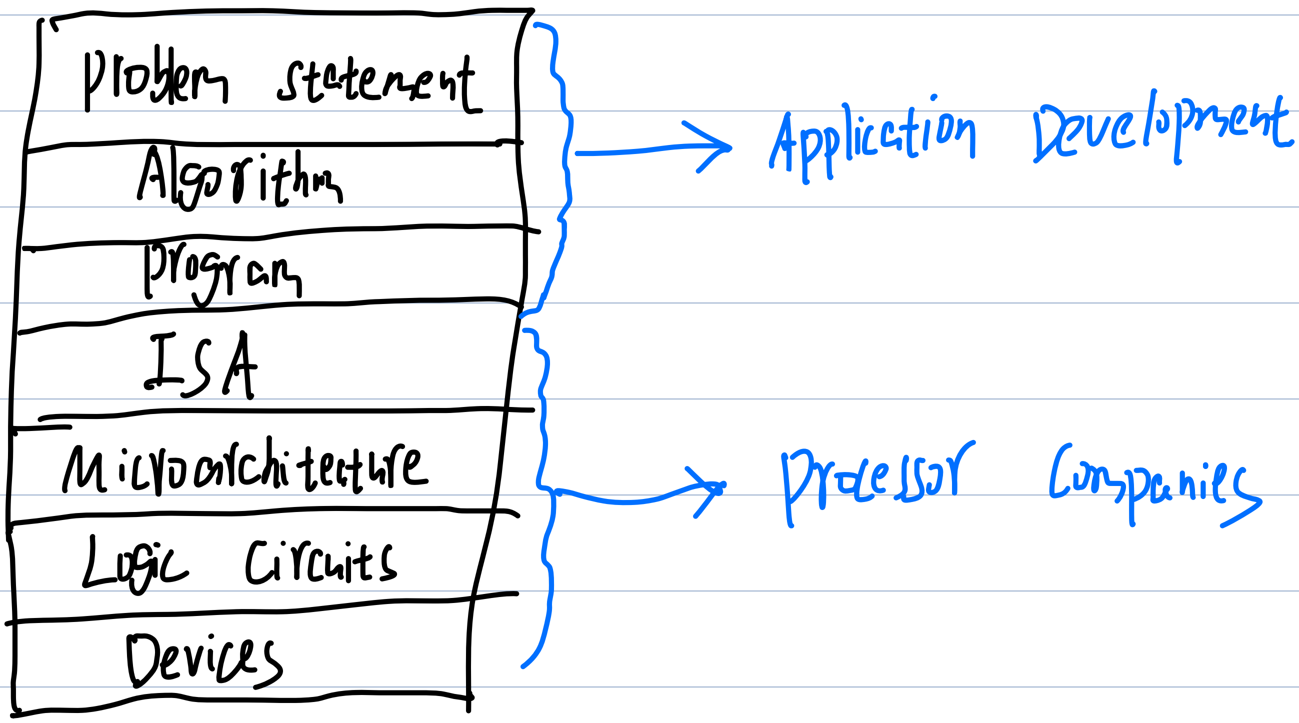
Transistors / machines in UTM theory

The Instruction Set Architecture (ISA):

- Bridge between hardware and software
- A specification of how software controls hardware

☆ We use layers of abstraction to focus on a piece of it at a time

Main Abstraction Layers:



Problem statement Layer: Stated using "human language"

Algorithm Layer: Procedure to finish the task

Program Layer: Express the algorithm using a computer language

ISA Layer: Specifies sets of instructions computer can perform

Microarchitecture Layer: Organization of a processor

(Different implementations of a single ISA)

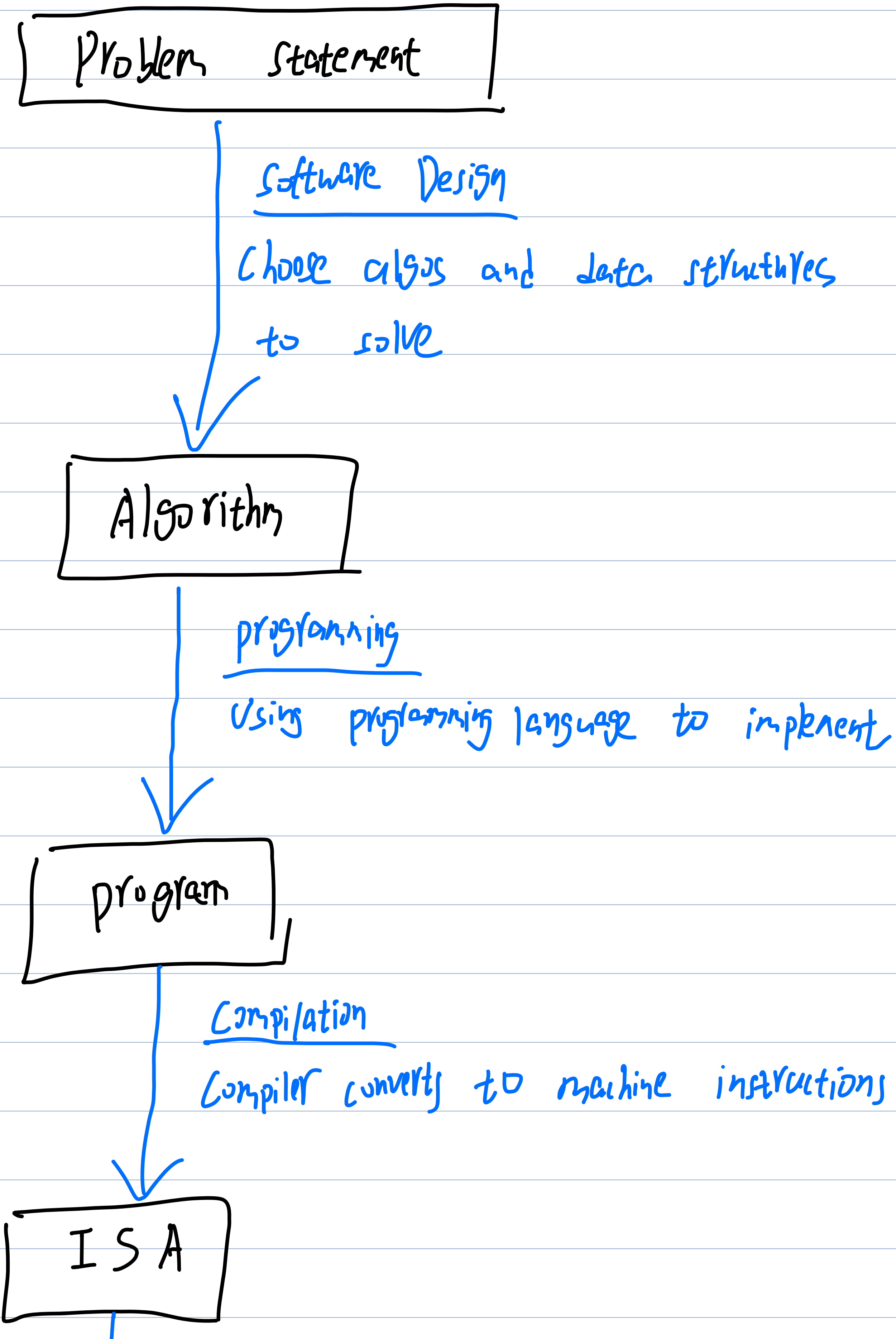
Logic Circuits: Combine basic operations to realize microarchitecture

↓ (zeros and ones)

Devices Layer: Transistor-based implementation of logic circuits

— Properties of materials, need to deal with voltage / current / power ......

Idea : Transformation Between Layers



Software Design

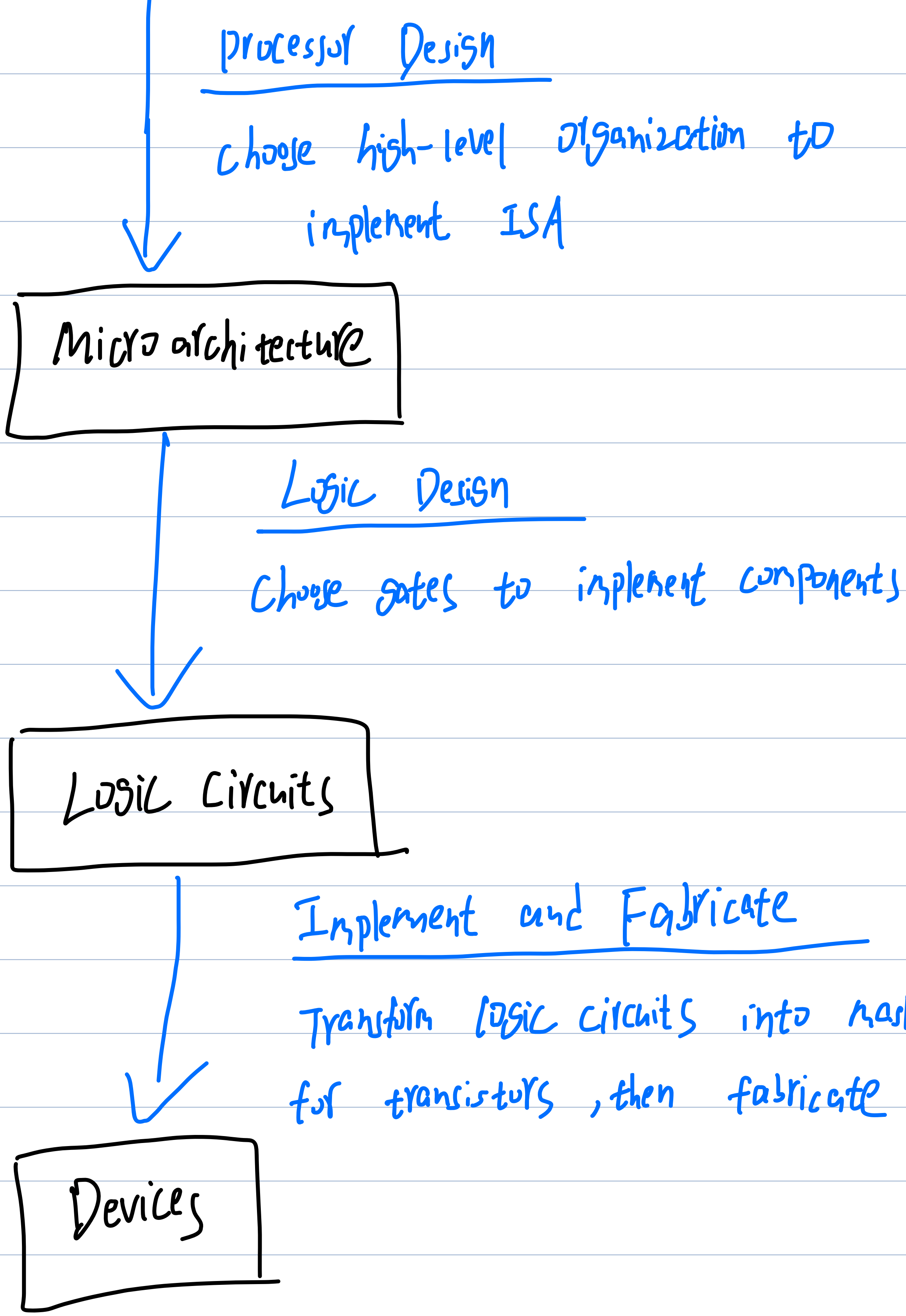
Choose algos and data structures to solve

programming

Using programming language to implement

Compilation

Compiler converts to machine instructions



Processor Design

Choose high-level organization to implement ISA

Microarchitecture

Logic Design

Choose gates to implement components

Logic Circuits

Implement and Fabricate

Transform logic circuits into masks for transistors, then fabricate

Devices

Electrical Information

Analog vs. Digital :

☆☆ Almost all computing systems are digital

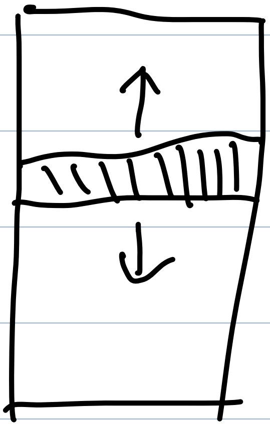
☆ Difference between "Analog" and "Digital"

"Analog" ↓ Continuous range of values

"Digital" ↓ Discrete set of values

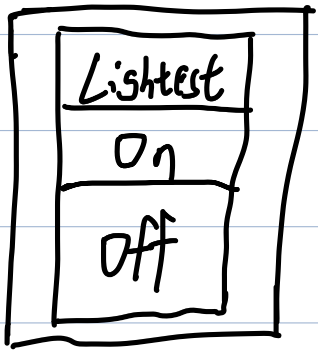
Ex. Light switch :

Analog →



(move upside and down)

Digital →



(buttons to press for control, limited options)

Digital Information:

"Voltage" is used to process and store information inside computers

↓

Different voltages represent different values

Binary Digital Information:

— 2 voltage levels represent 1 and 0

↓ range of voltages

(☆ Digital is the abstraction of Analog)

A wire transmits a voltage representing 0/1 at a time.

↓

(one bit)

☆ (Multiple wires used to transmit multiple bits)

Binary Information:

All information in computer is represented by binary numbers

Additional Concepts in Educational Objectives :

- Requirement of a good algorithm : ① Definitness
② Effective computability
③ Finiteness

Steps need to be specific and clear

Each step must be possible

The algorithm must be able to finish and stop