


EECE7205: Fundamentals of Computer Engineering



Introduction

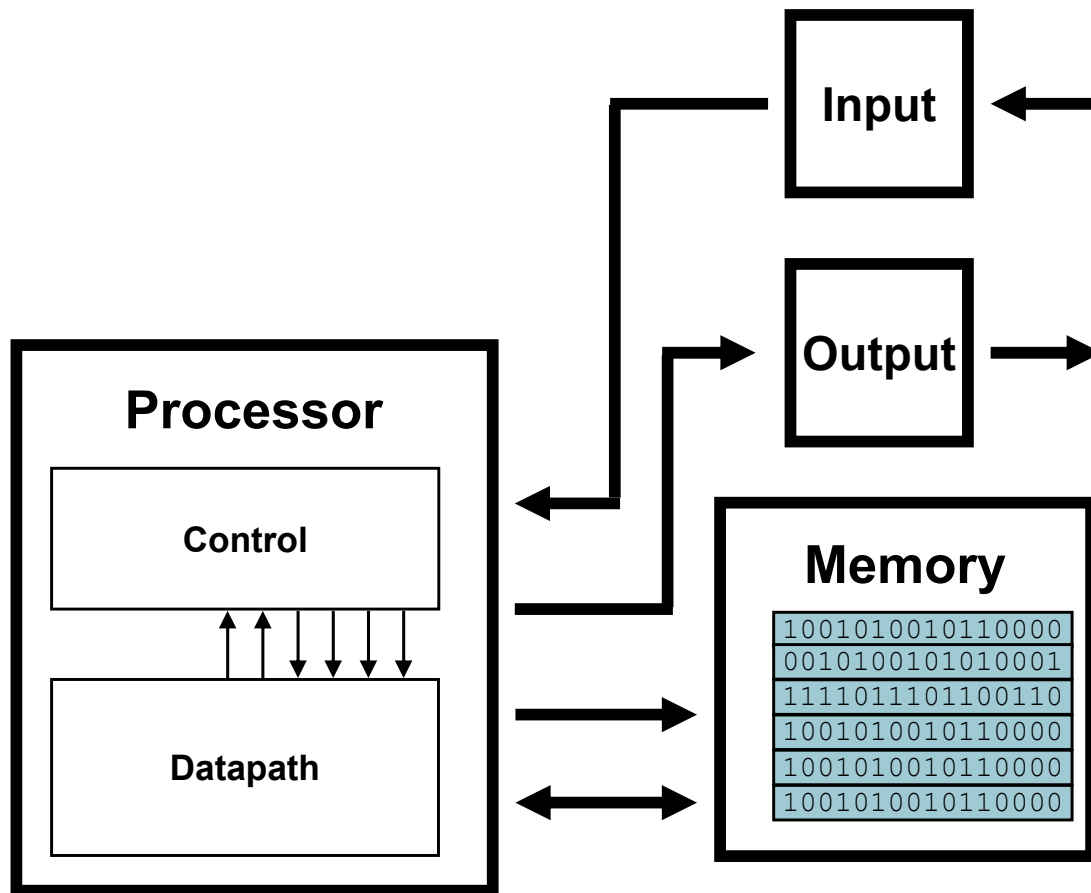


Computer Model

- For most of this course, we shall assume a generic one processor, random-access machine (RAM) model of computation as our implementation technology
- In this model, instructions are executed one after another, with no concurrent operations.



The Von Neumann Computer Model



- Same components for all kinds of computer
 - Desktop, server, embedded
- Input/output includes
 - User-interface devices
 - Display, keyboard, mouse
 - Storage devices
 - Hard disk, CD/DVD, flash
 - Network adapters
 - For communicating with other computers

John von Neumann (1903-1957) was a Hungarian-American mathematician, physicist, inventor, computer scientist

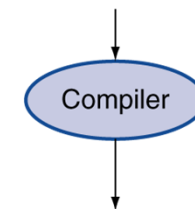


Levels of Program Code

- High-level language
 - Level of abstraction closer to problem domain
 - Provides for productivity and portability
- Assembly language
 - Textual representation of instructions
- Hardware representation
 - Binary digits (bits)
 - Encoded instructions and data

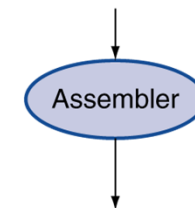
High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```



Assembly
language
program
(for MIPS)

```
swap:
  muli $2, $5, 4
  add  $2, $4, $2
  lw   $15, 0($2)
  lw   $16, 4($2)
  sw   $16, 0($2)
  sw   $15, 4($2)
  jr   $31
```



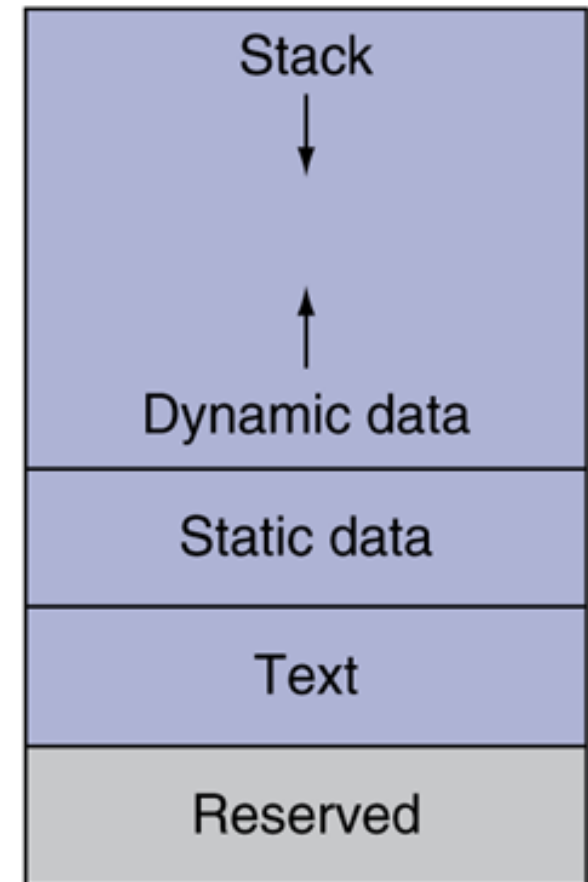
Binary machine
language
program
(for MIPS)

```
000000001010000100000000000011000
000000000000110000001100000100001
100011000110001000000000000000000
1000110011110010000000000000000100
101011001111001000000000000000000
1010110001100010000000000000000100
000000111110000000000000000001000
```



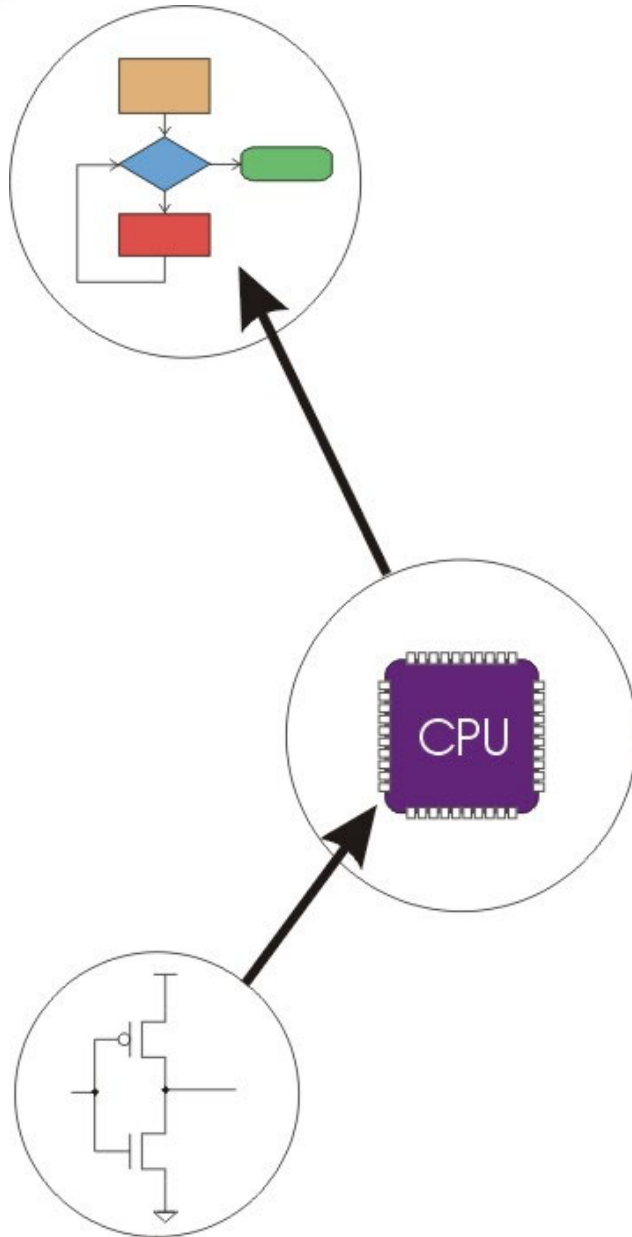
Memory Layout

- Text: program code
- Static data: global variables
 - Example: global and static variables in C.
- Dynamic data: heap
 - Examples: malloc in C, new in Java
- Stack: local variables of functions.





Layers of Abstraction



Problems

Algorithms

Program

Instruction Set Architecture

Microarchitecture

Circuits

Devices



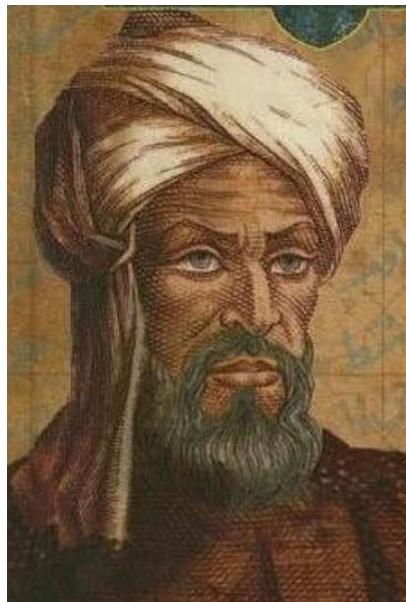
What is an Algorithm?

- An algorithm can be viewed as a tool for solving a well-specified ***computational problem***.
 - The statement of the problem specifies the desired input/output relationship.
 - The algorithm describes a specific computational procedure for achieving that input/output relationship.
- An algorithm is said to be ***correct*** if, for every input instance, it halts with the correct output.
- An algorithm can be specified in English or as a computer program.



The Origin of the Word Algorithm

- The English word "algorithm" derives from the Latin form of Al-Khwarizmi's name. He developed the concept of an algorithm in mathematics.
- Al-Khwarizmi was born around 800 CE in Khwarizm, now in Uzbekistan and lived in Baghdad (Iraq).

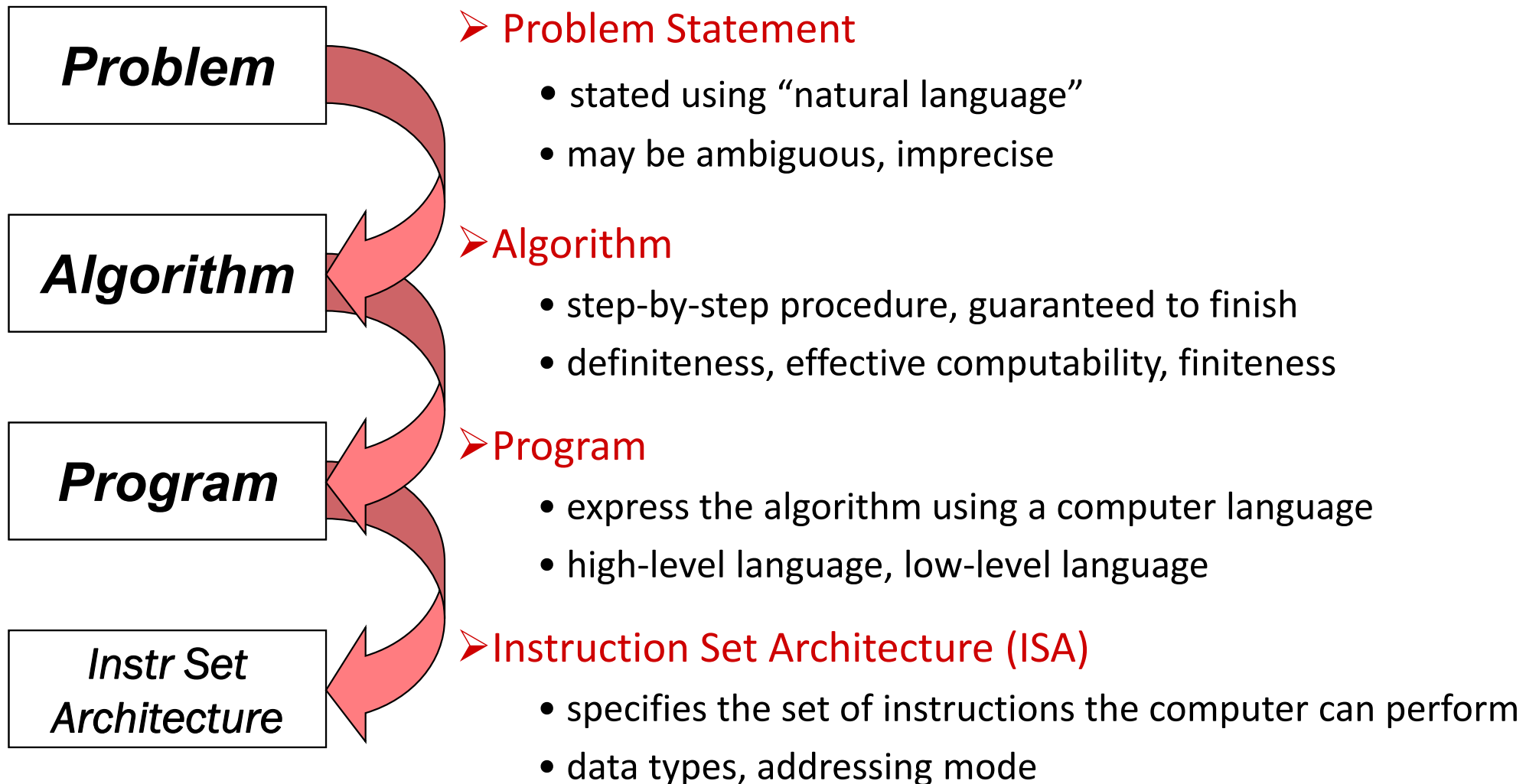




Levels of Transformation

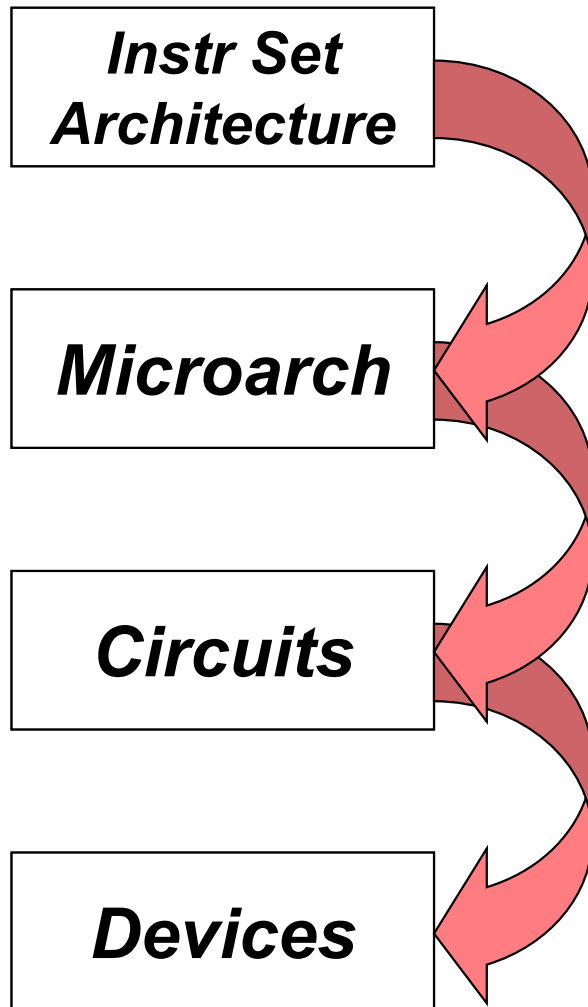
■ How do we solve a problem using a computer?

A systematic sequence of transformations between layers of abstraction.





Levels of Transformation (Cont'd)



➤ Microarchitecture

- detailed organization of a processor implementation
- different implementations of a single ISA

➤ Logic Circuits

- combine basic operations to realize microarchitecture
- many different ways to implement a single function (e.g., addition)

➤ Devices

- properties of materials, manufacturability



Analysis of Algorithms

- Analyzing an algorithm has come to mean predicting the resources that the algorithm requires.
 - Resources such as memory, communication bandwidth, or computer hardware.
 - Most often it is **computational time** that we want to measure.
- Less computational time results in better computer-program **performance**.
- Other features include locality (for cache memory) and simplicity (for programmers).



Performance Factors

- Algorithm
 - Determines number of operations executed
- Programming language, compiler, architecture
 - Determine number of machine instructions executed per operation
- Processor and memory system
 - Determine how fast instructions are executed
- I/O system (including OS)
 - Determines how fast I/O operations are executed



Data Structures

- A ***data structure*** is a way to store and organize data in order to facilitate access and modifications.
- No single data structure works well for all purposes, and so it is important to know the strengths and limitations of several of them.
- We will cover different data structures in this course.



Reasons to Study Algorithms

- Studying algorithms will allow you to:
 - Find an efficient solution to a new computational problem.
 - Compare different available solutions to a problem.
 - Understand different techniques used in many computing systems (e.g., operating systems, computer networks, computer architecture).
 - Finally studying algorithms will allow you to answer many interviews questions.