The Nature of Computing

INEL 4206 – Microprocessors

Lecture 3

Bienvenido Vélez Ph. D.

School of Engineering

University of Puerto Rico - Mayagüez

Some Inaccurate Yet Popular Perceptions of Computing

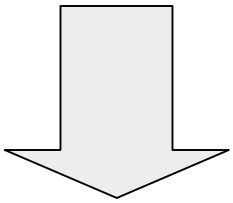
Computing = Computers

Computing = Programming

Computing = Software

Computing = Computers

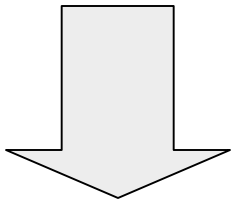
Computing is about solving problems using computers



A.K.A. The Computing Device View of Computing

Computing = Programming

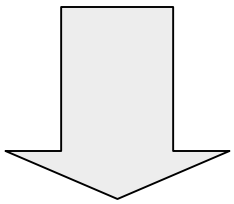
Computing is about writing
programs for computers



A.K.A. The Programming Language view of Computing

Computing = Software

Computing is not concerned with hardware design



A.K.A. The “Floppy Disk” view of Computing

Outline

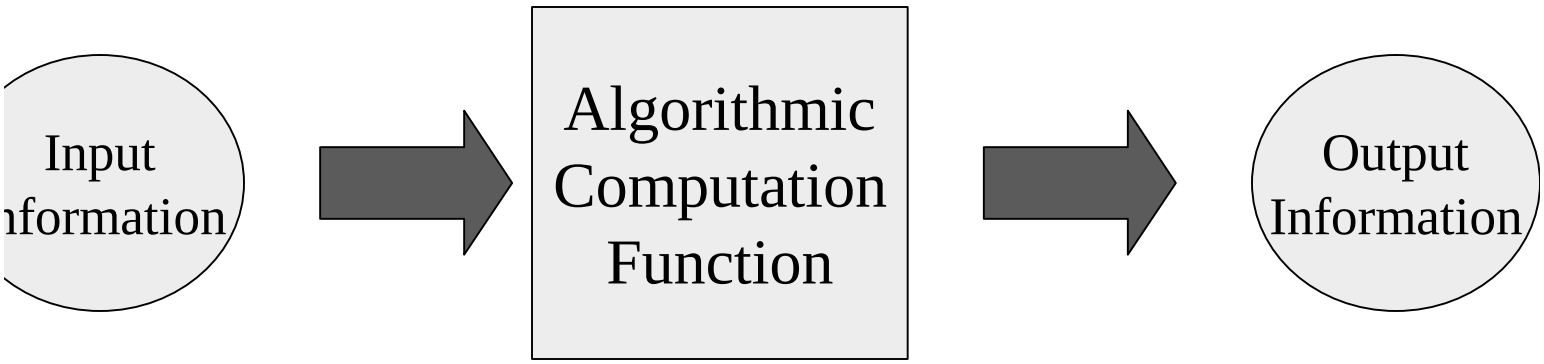
What is Computing?

Computing Models and Computability

Intepretation and Universal Computers

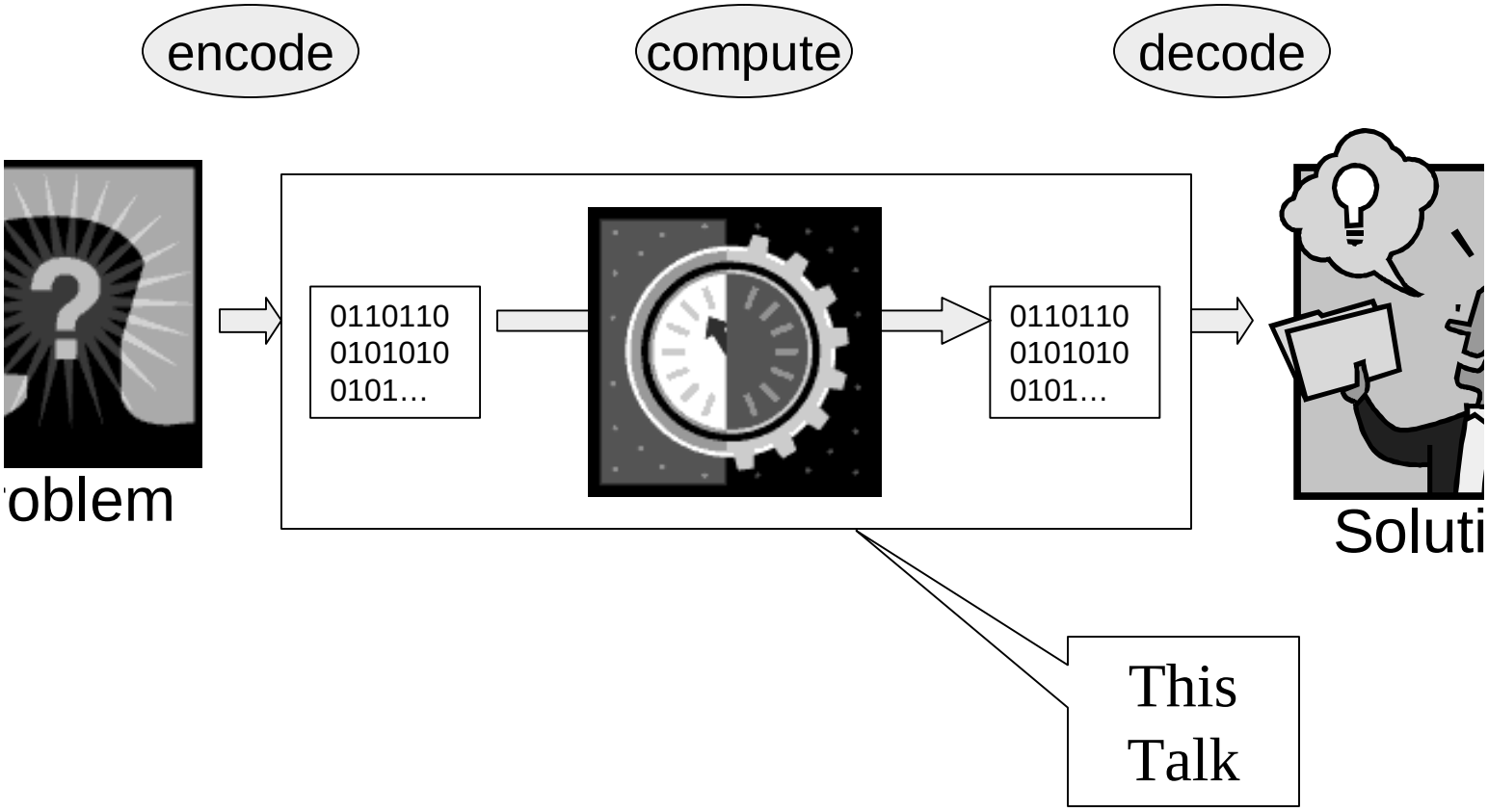
Abstraction and Building Blocks

What is computing then?



Computing is the study of Computation:
the process of transforming information

The Computation Process



Fundamental Questions Addressed by the Discipline of Computing

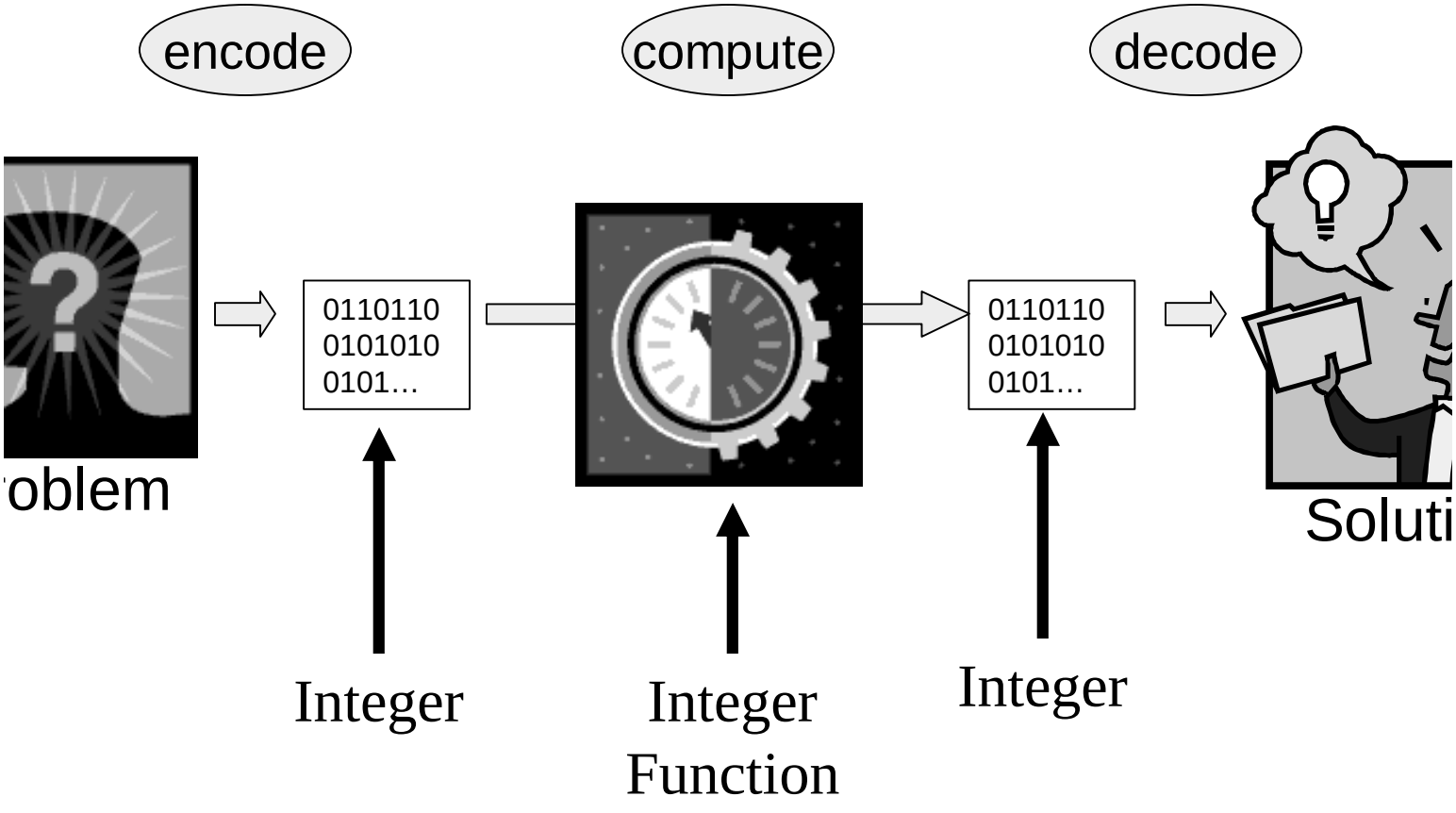
What is the nature of computation?

What can be computed?

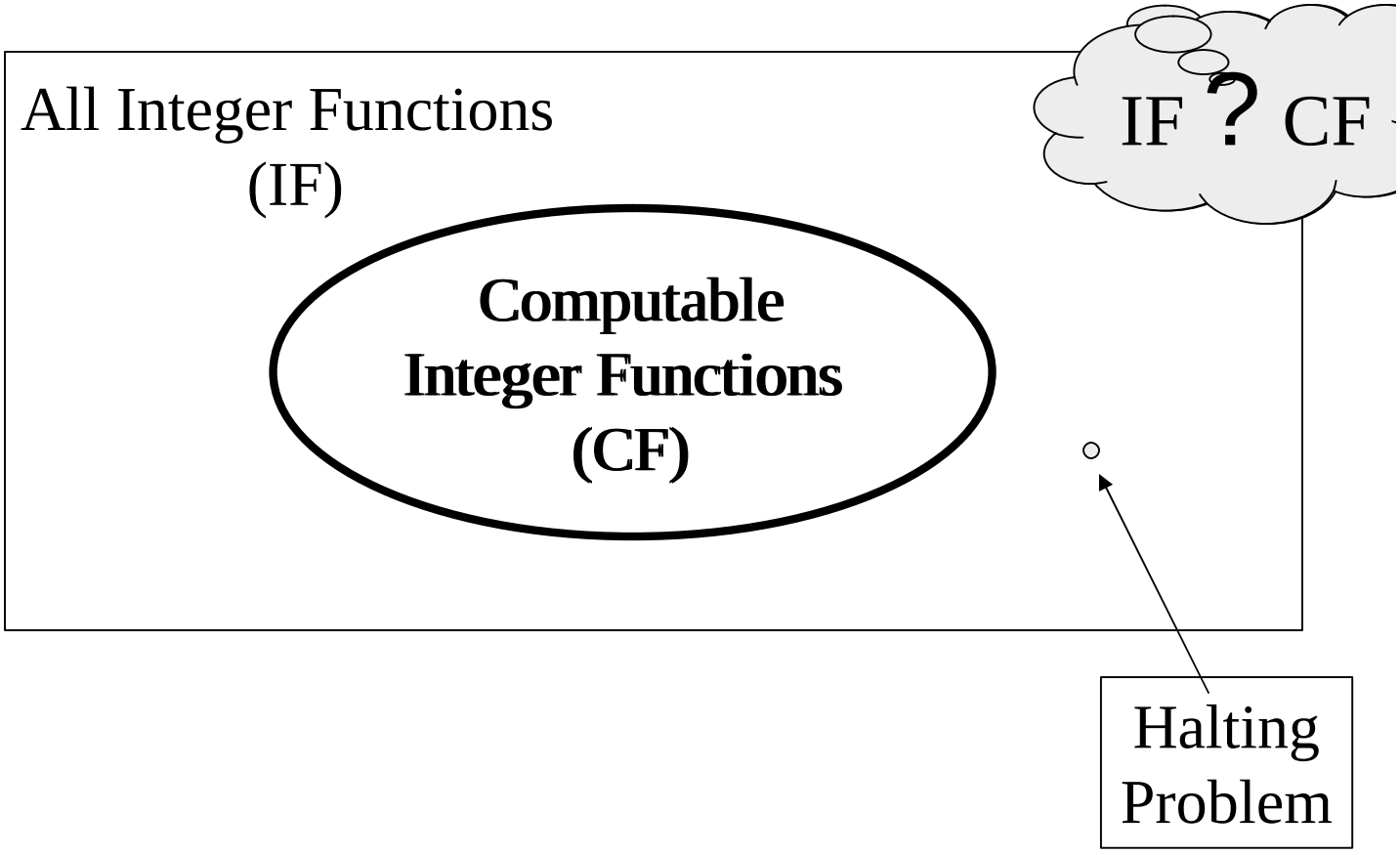
What can be computed efficiently?

How to build practical computing devices?

The Computation Process

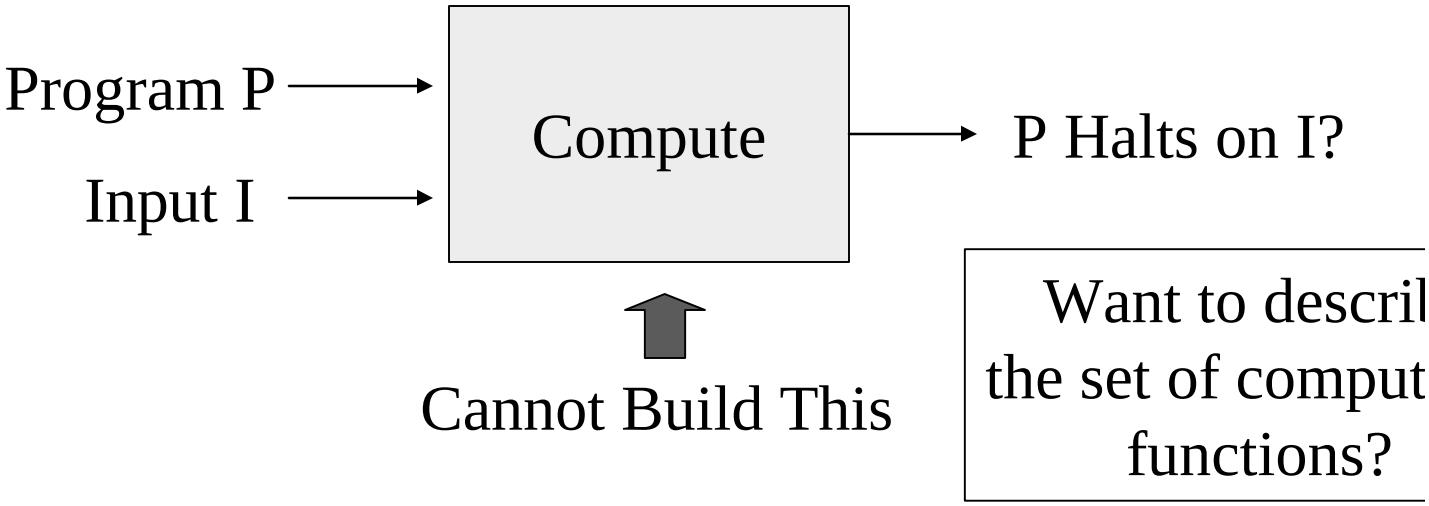


Computability



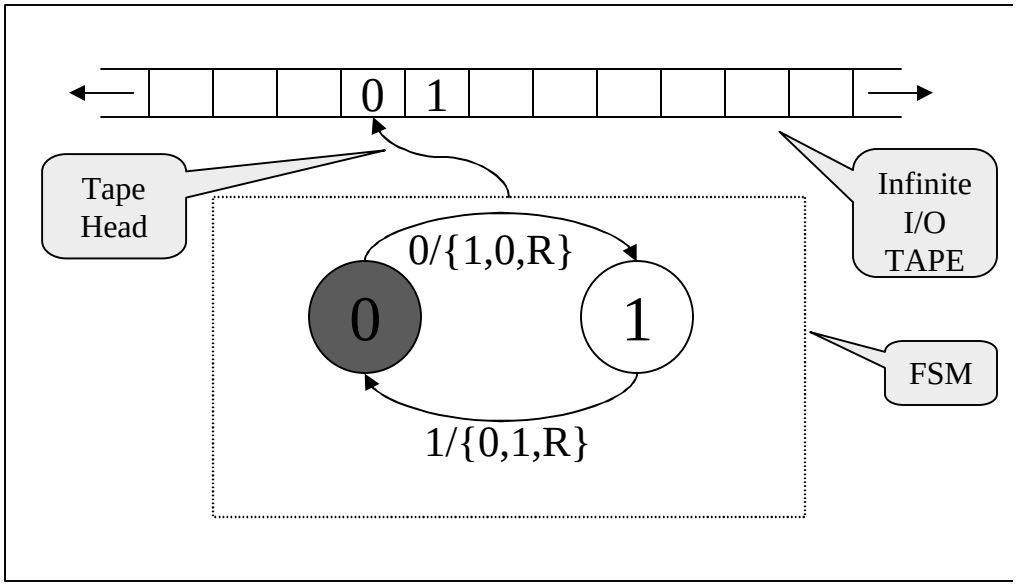
The Halting Problem
(Alan Turing 1936)

Given a program and an input to the program, determine if the program will eventually stop when it is given that input.



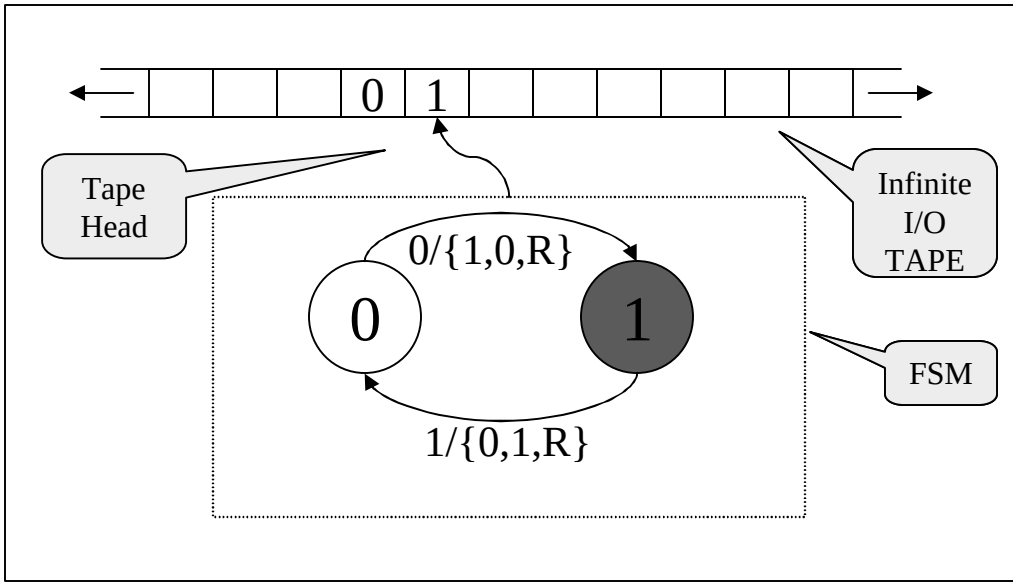
Mathematical Computers: The Turing Machine (1936)

Alan Turing



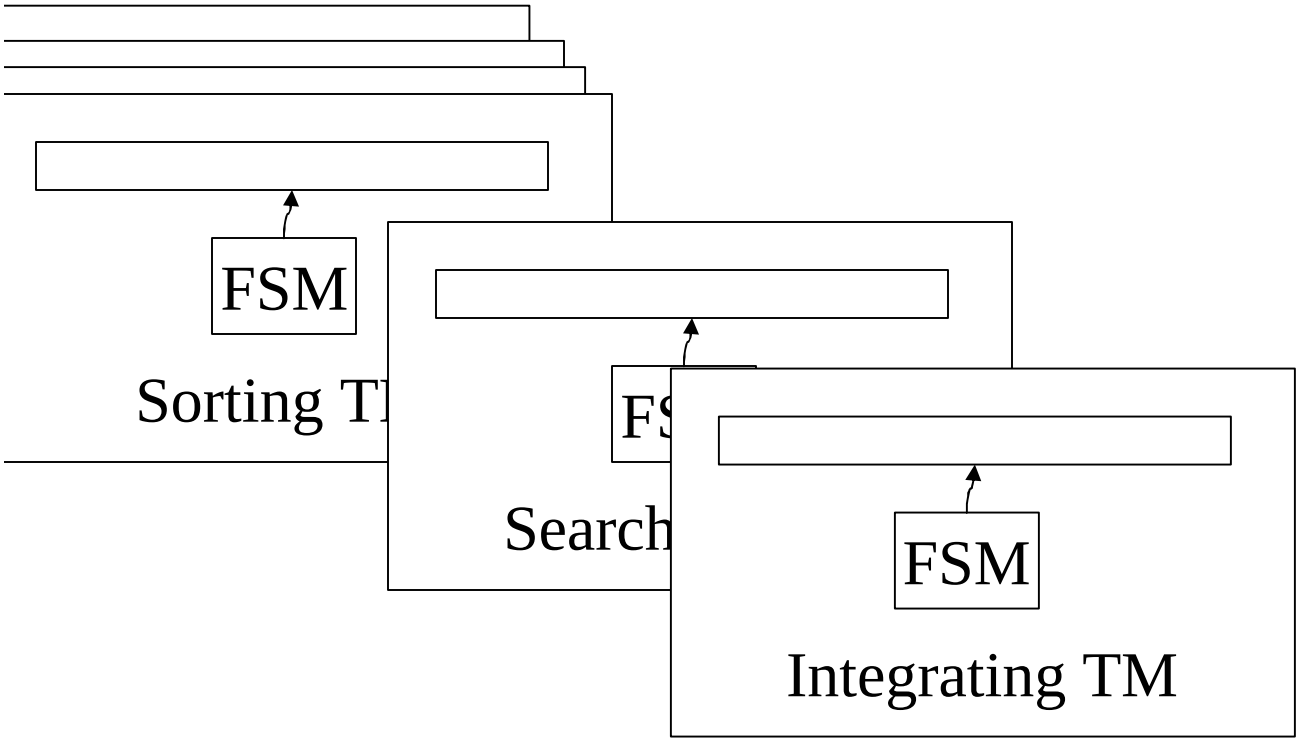
Mathematical Computers: The Turing Machine (1936)

Alan Turing



Turing demonstrated how to solve sever problems using his computing model

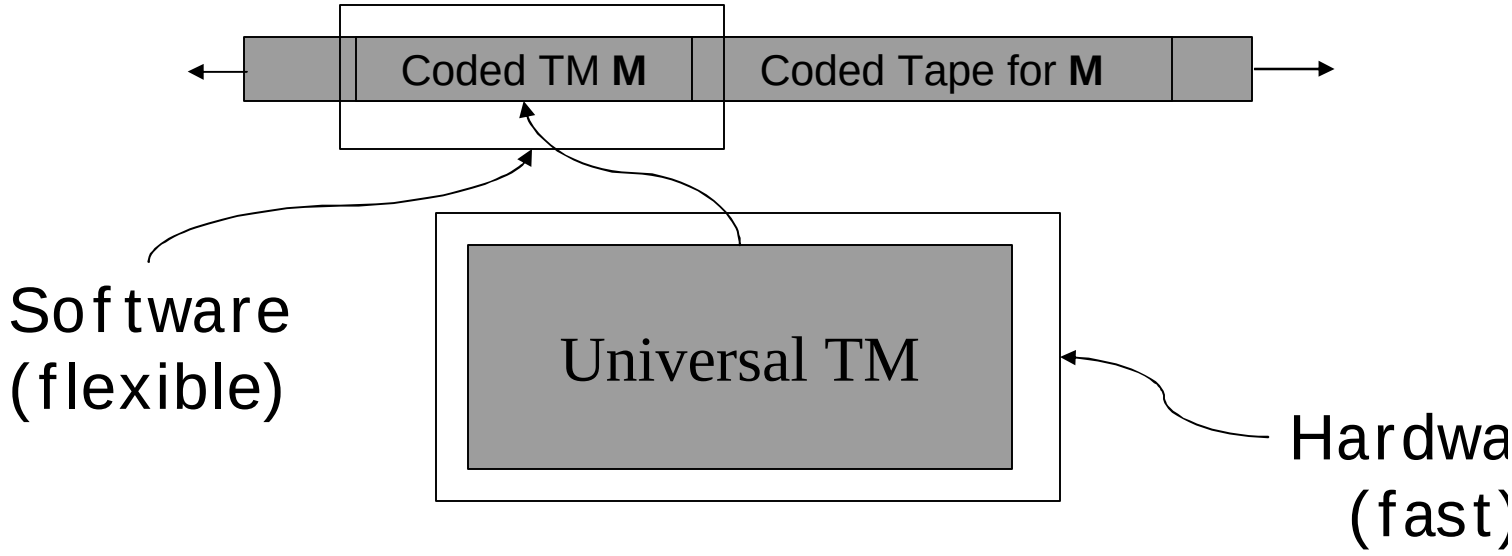
Ad-hoc Turing Machines



Can we build a general purpose TM?

The Universal Turing Machine (UTM)

The Paradigm for Modern General Purpose Computers



- Capable of Emulating Every other TM
- Shown possible by Alan Turing (1936)
- BIG IDEA: INTEPRETATION!!!

Other Familiar Models of Computation

Combinational Circuits
Sequential Circuits (FSM's)
Pentium Instruction Set Architectures
Lambda Calculus
Recursive Functions
C++

Can you tell which ones are Turing Universal?
That is, which ones can emulate any other Turing Machine?

Church's Thesis

Alonso Church

"Any realizable computing device can b
simulated by a Turing machine"

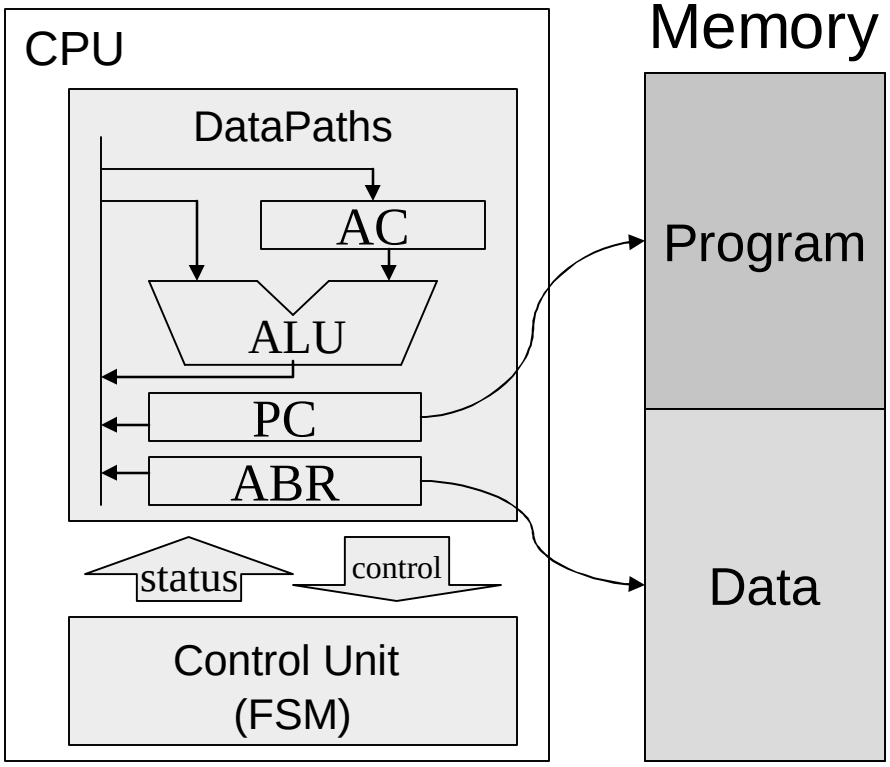
"All the models of computation yet developed, and all those that may be developed in the future, are equivalent i power."

Issues not considered: Size, Programmability, Performance
But they must be considered if one is to build …

Practical Universal Computers

(John) Von Neumann Architecture (1945)

ohn von Neumann, 1950's

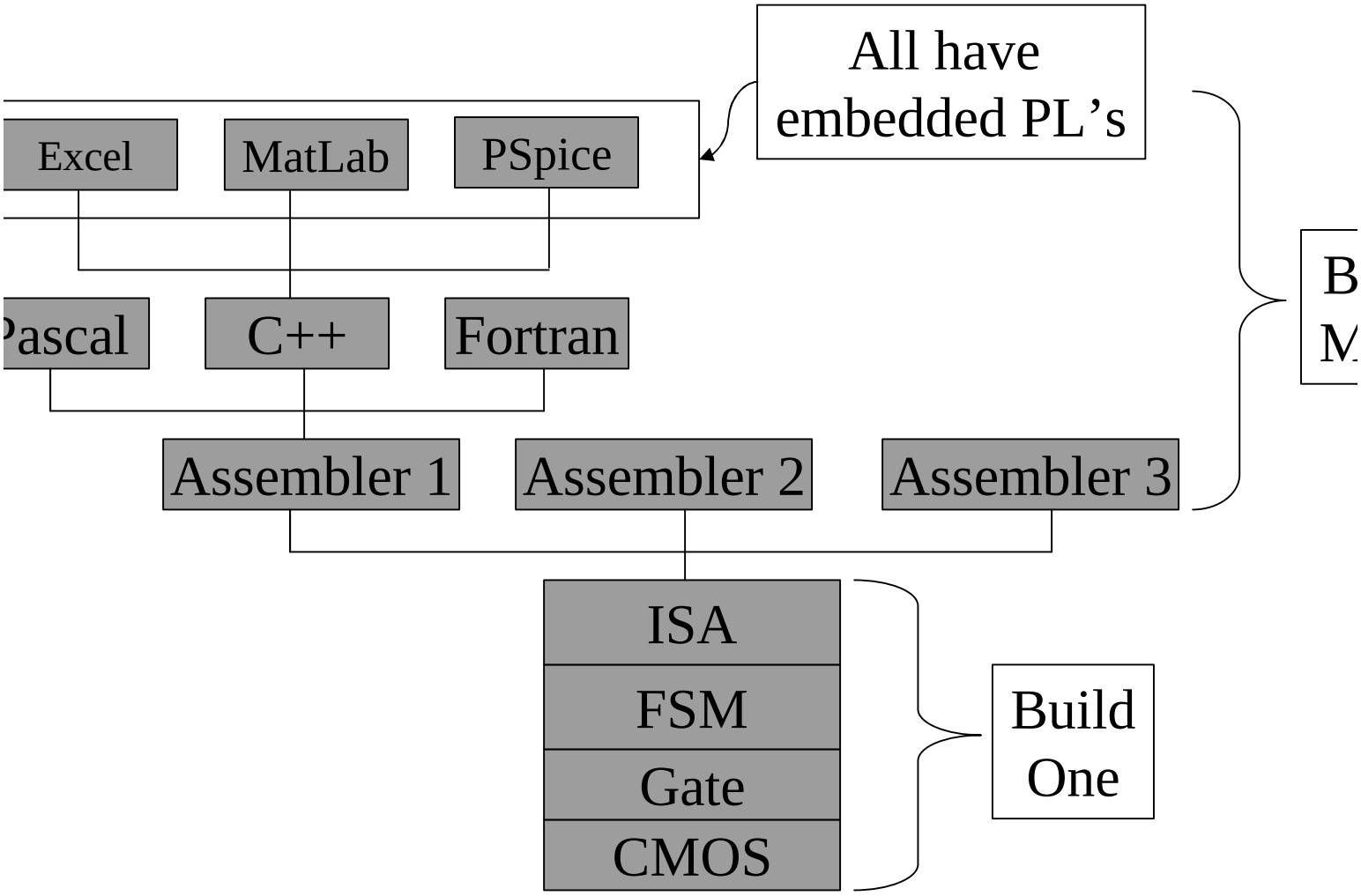


This
just
TM

CPU is a universal TM

An interpreter of some programming language (PL)

Computing in Perspective



Interpreter Design Demands Programming Language Desig

Computing in Perspective

People

Application Programs

High-Level Language

Assembly Language

Machine Language (ISA)

computers

Computer Human Interaction, User Interfac

CS1/CS2, Programming, Data Structures

Programming Languages, Compilers

Computer Architecture

Each layer implements an INTERPRETER for some programming languag

Why Abstraction Layers?

Resilience to change:

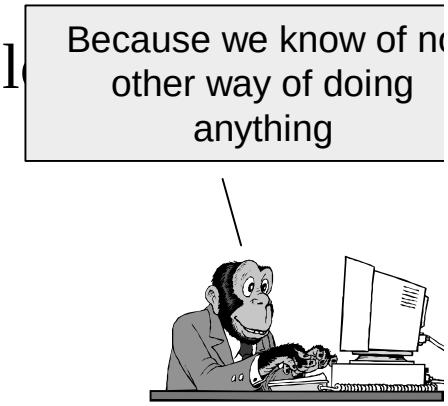
- Each layer provides a level of indirection

Divide and Conquer Approach:

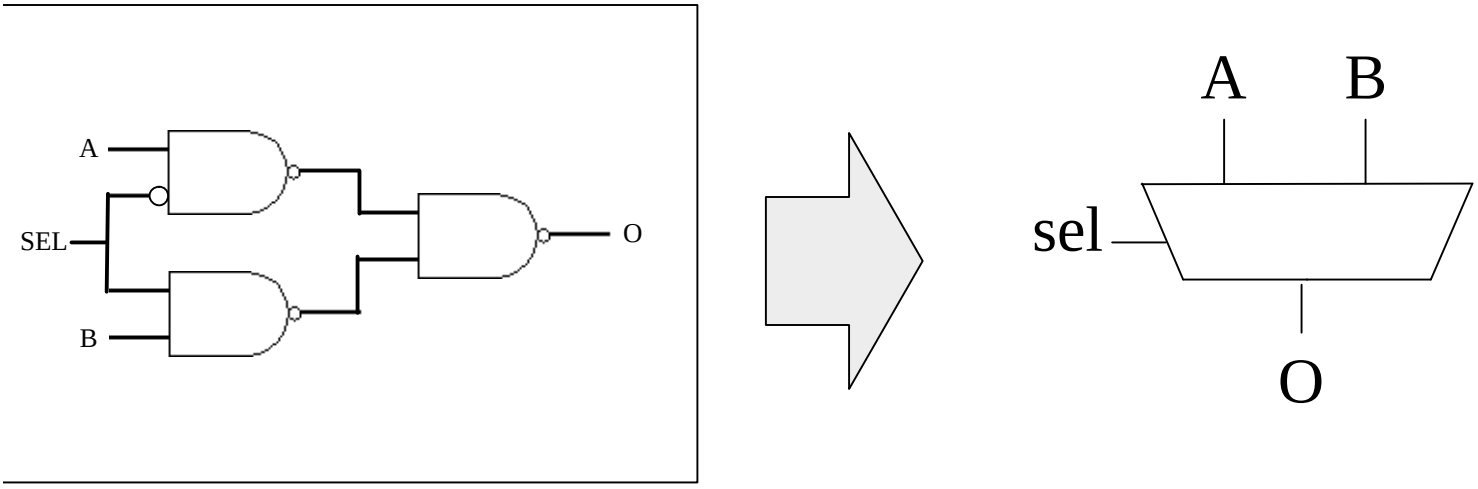
- Can work on one small semantic gap at a time

Building Block Approach:

- Can build many higher layer on same l



Hardware Building Blocks



Gate-Level Logic Provides a Computing Model

Software Building Blocks

The function is one of the most ubiquitous abstraction tools

```
// MUX – Implements a 2-1 binary multiplexer
bool MUX(bool a, bool b, bool sel) {
  switch(sel) {
    case 0: return a; break;
    case 1: return b; break;
  }
}
```

Other abstraction tools include:
structures, classes, modules

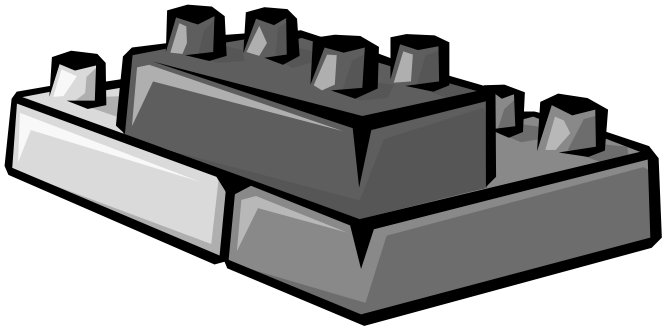
What Makes a Good Building Block

Provides a clear and simple contract

The contract hides irrelevant detail

The contract is general and orthogonal

The contract is easy to remember



Some Properties of a Good Programming Language

- Does not hide expressive power of lower layers
- Can be efficiently interpreted
- Provides adequate higher level abstractions
- Provides a variety of constructs for creating new abstractions, layers and modules
- Achieves all of the above with minimal complexity

Summary

"Computer Science is no more about computers than Astronomy is about telescopes"

E. Dijkstra

Summary

Computing = Information Transformation

Information Transformation = Integer Functions

Some integer functions are not computable

Turing Machine computations = All computations

Universal Computer = Universal TM

Interpretation => Programmability => Flexibility

Building blocks are abstract contracts

Outline of The Talk

- Introduccion a la disciplina
  - Informacion, modelos computacionales, computabilidad
- Interpretacion
  - Universalidad, expresividad, PL
- Modularidad & Building Blocks
  - Software example vs hardware example
- Criterios HW vs. SW
  - flexibilidad vs. performance