# The Nature of Computing

ICOM 4036
Lecture 2

Prof. Bienvenido Velez

# Some Inaccurate Yet Popular Perceptions of Computing

- Computing = Computers
- Computing = Programming
- Computing = Software

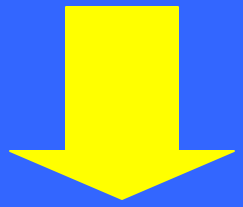
# Computing = Computers

Computing is about solving problems using computers

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

# Computing = Programming

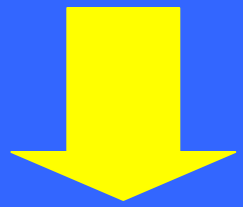
Computing is about writing
programs for computers



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

# Computing = Software

Computing is not concerned with hardware design

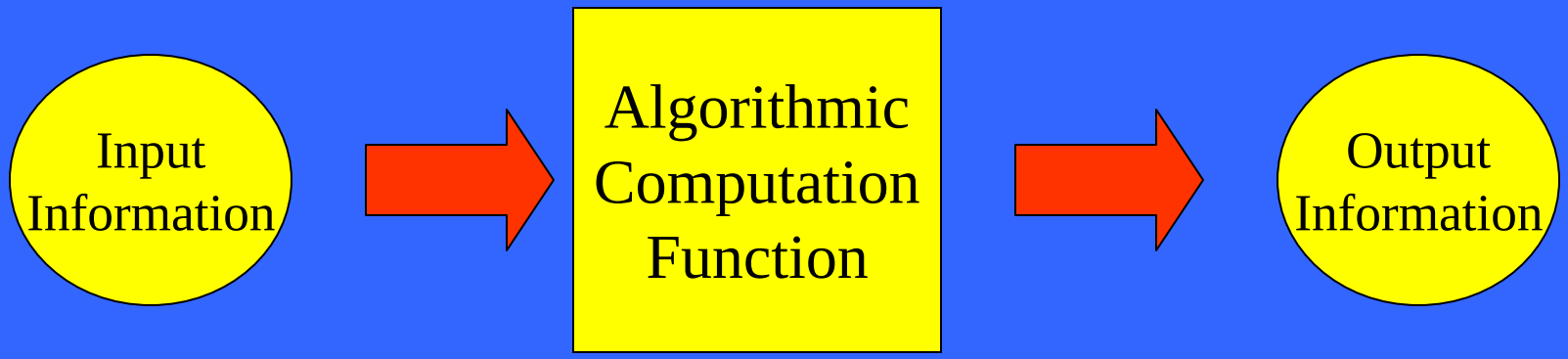


A.K.A. The “Floppy Disk” view of Computing

# Part I - Outline

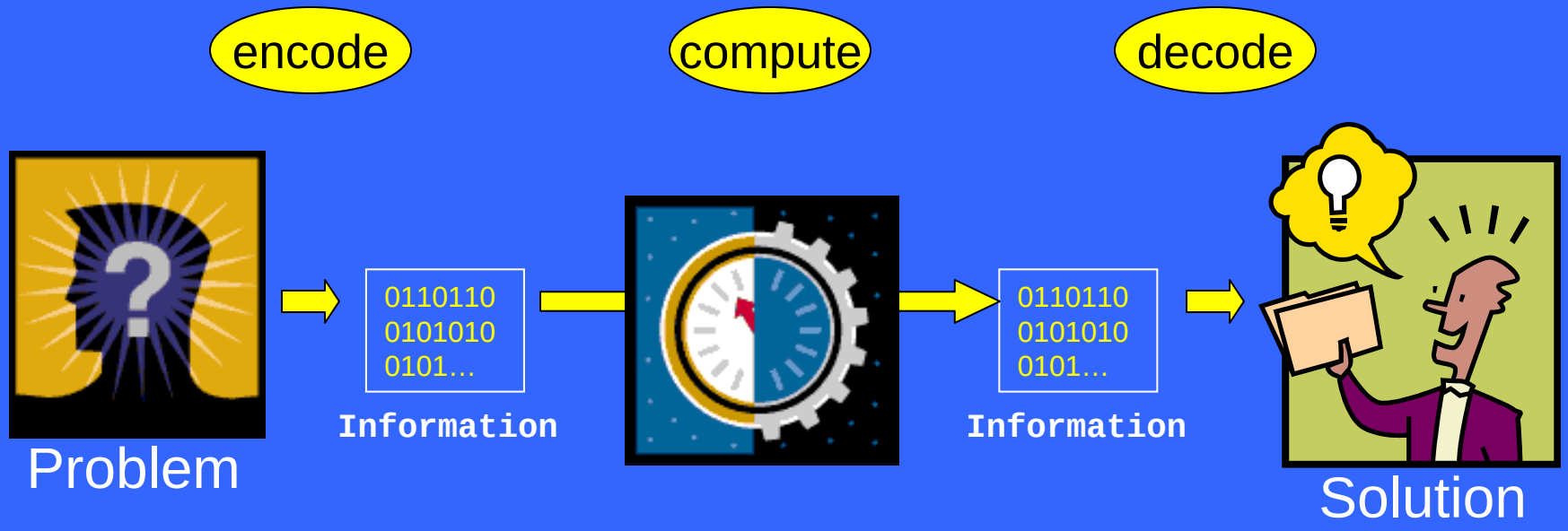
- What is Computing?
- Computing Models and Computability
- Interpretation and Universal Computers
- Church's Thesis

# What is computing then?




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

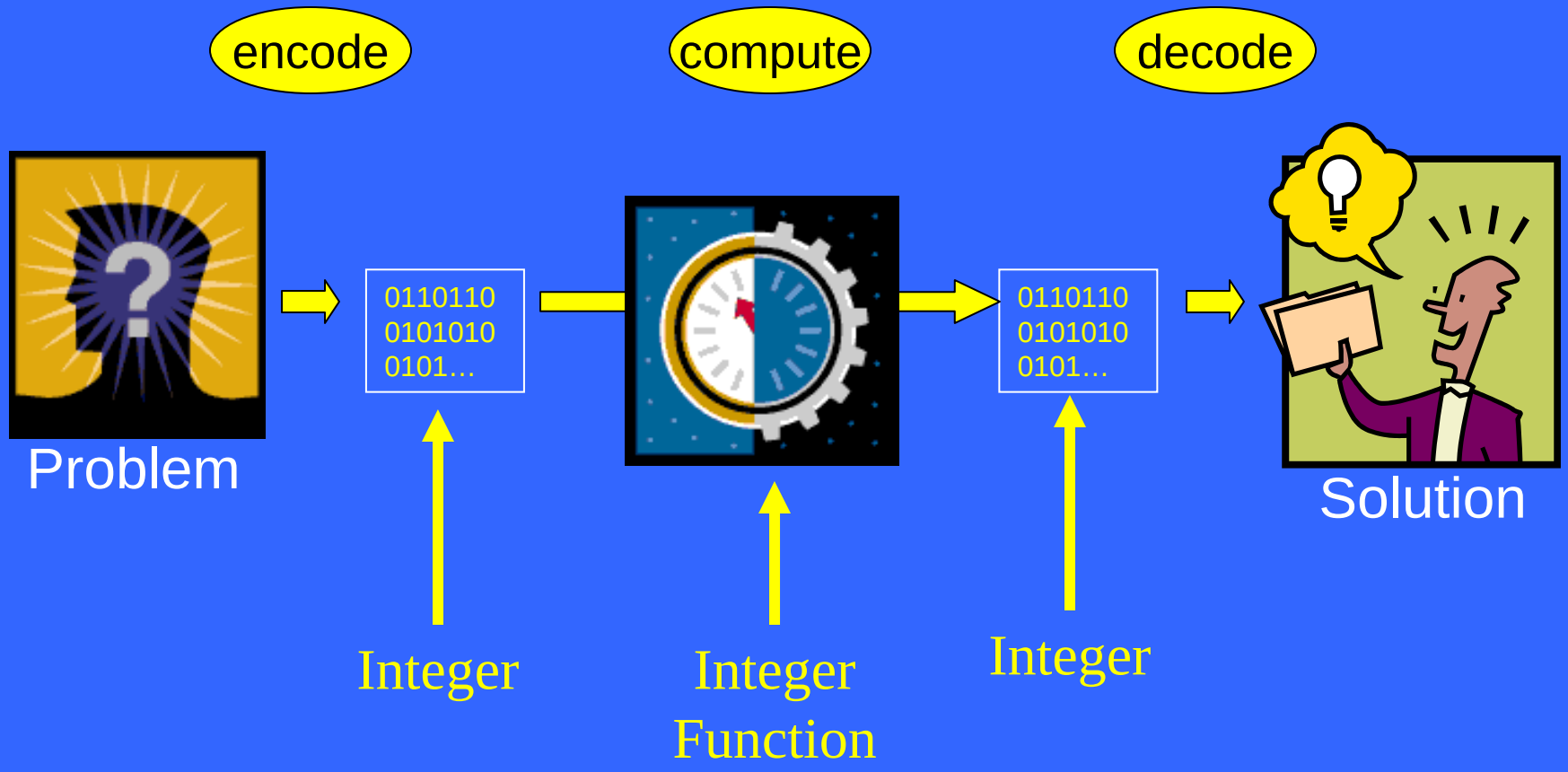
# The Computation Process



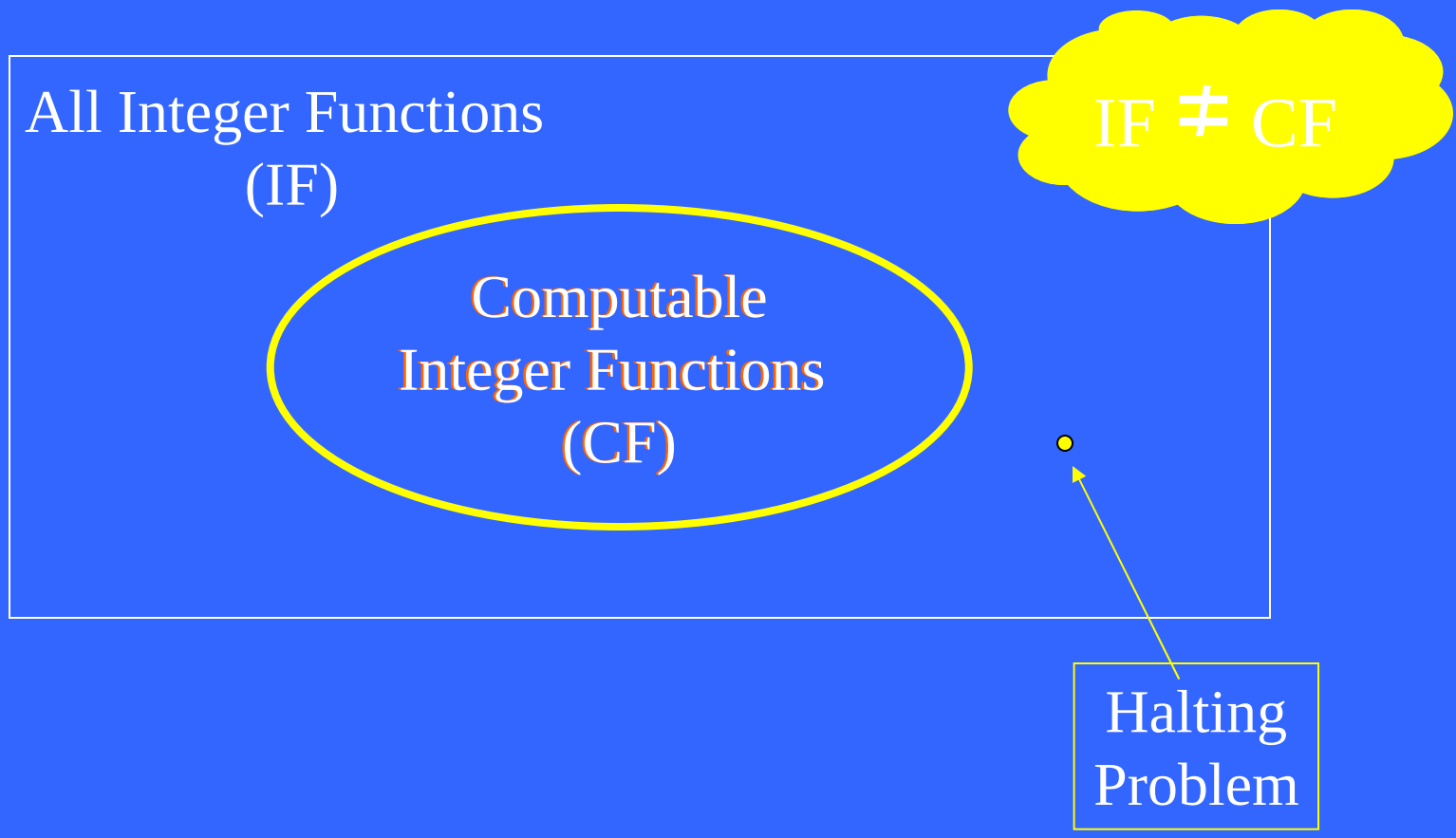
# Fundamental Questions Addressed by the Discipline of Computing

- What is the nature of computation?
- What can be computed?
- What can be computed efficiently?
- How can we build computing devices?

# The Computation Process

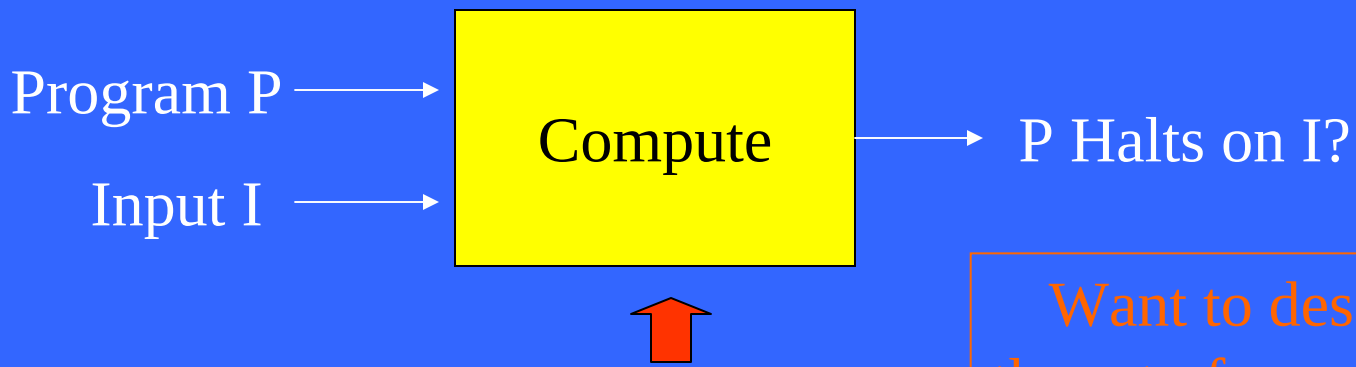


# Computability



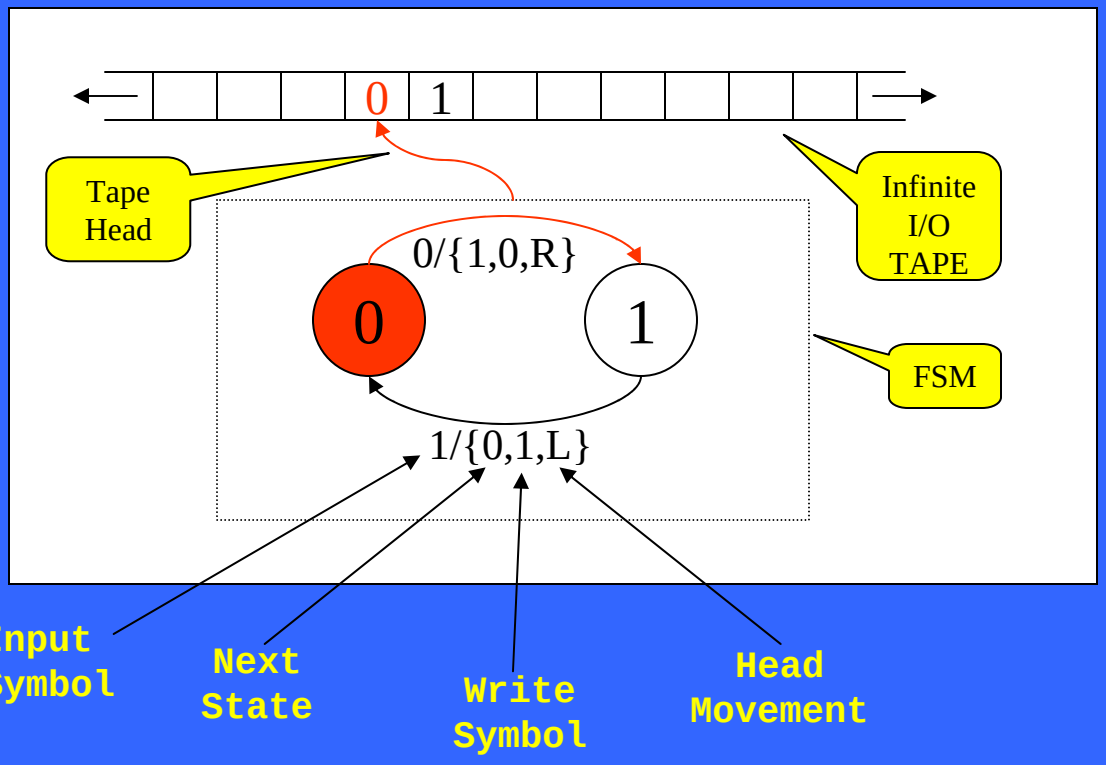
# The Halting Problem
## (Alan Turing 1936)

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



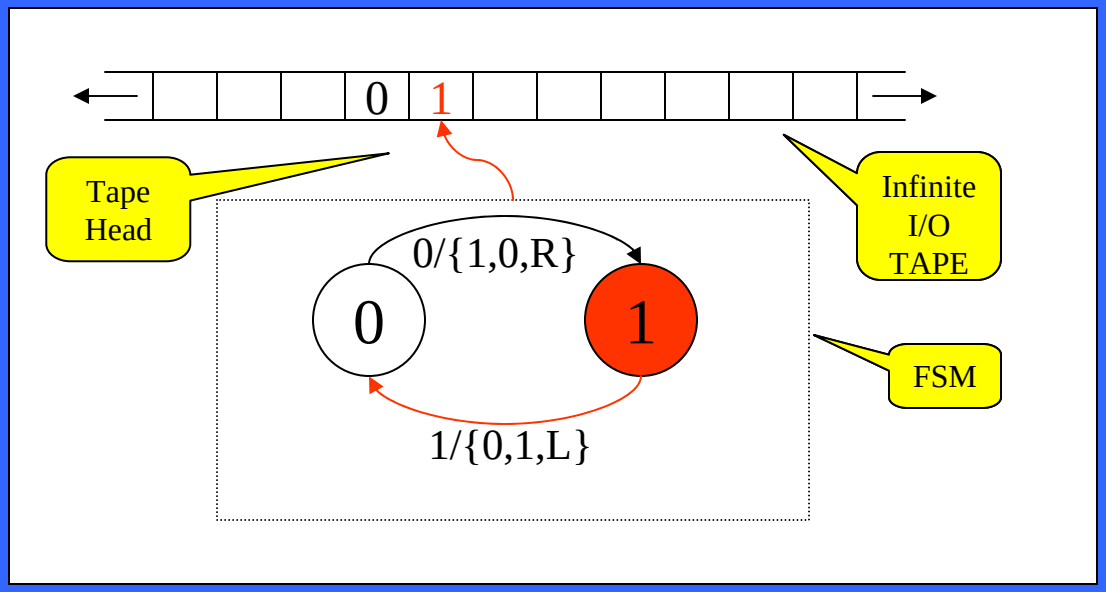
# Mathematical Computers: The Turing Machine (1936)

Alan Turing



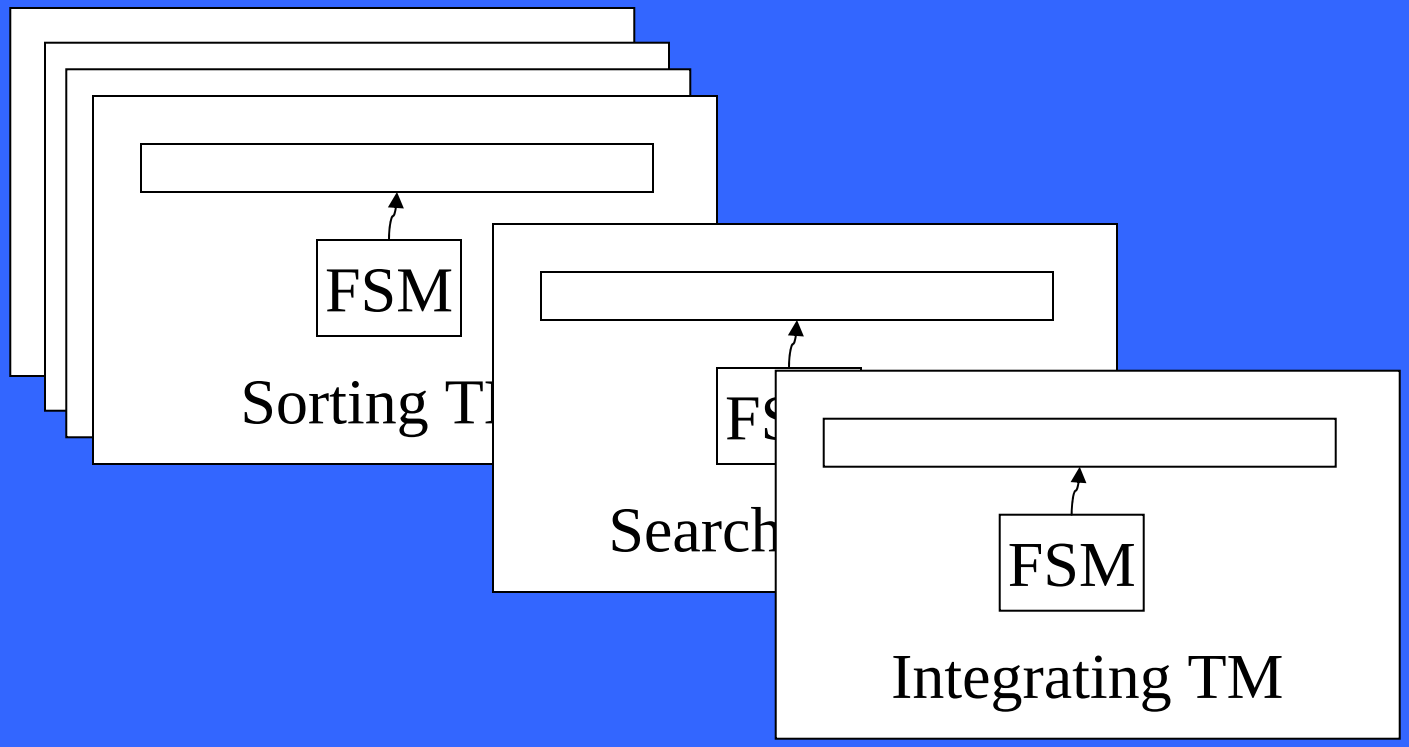
# Mathematical Computers: The Turing Machine (1936)

**Alan Turing**




Turing demonstrated how to solve several problems using his computing model

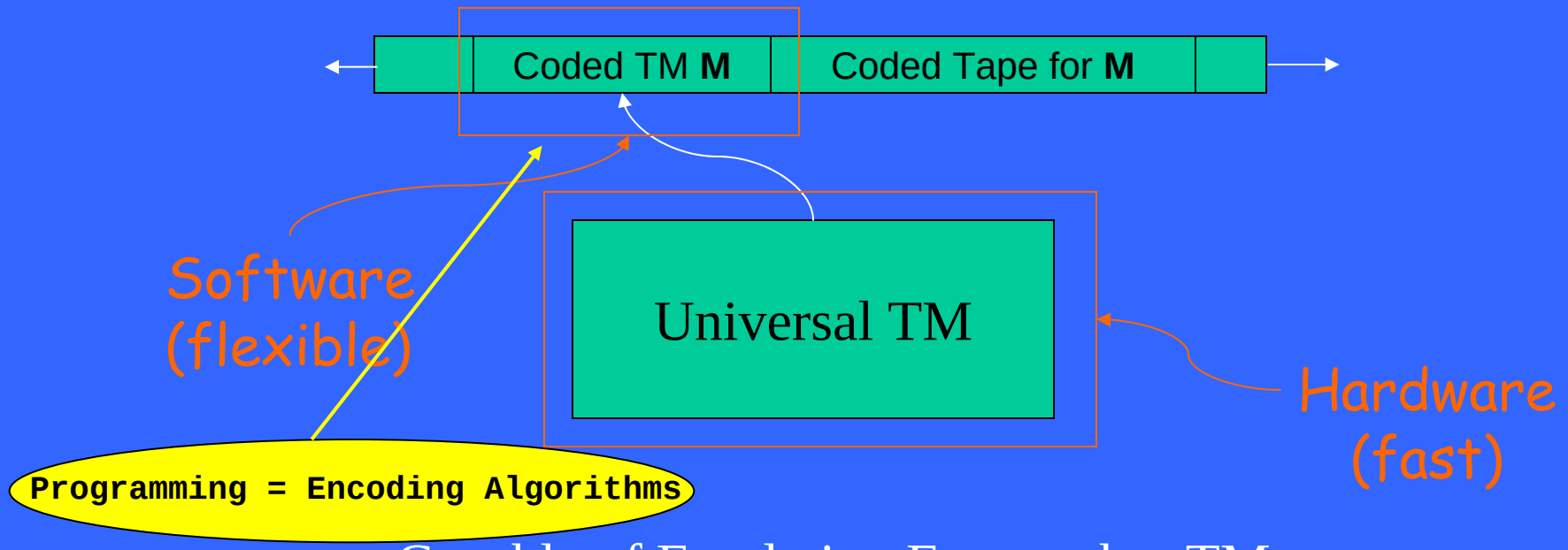
# Ad-hoc Turing Machines




Can we build a general purpose TM?

# The Universal Turing Machine (UTM)

## The Paradigm for Modern General Purpose Computers



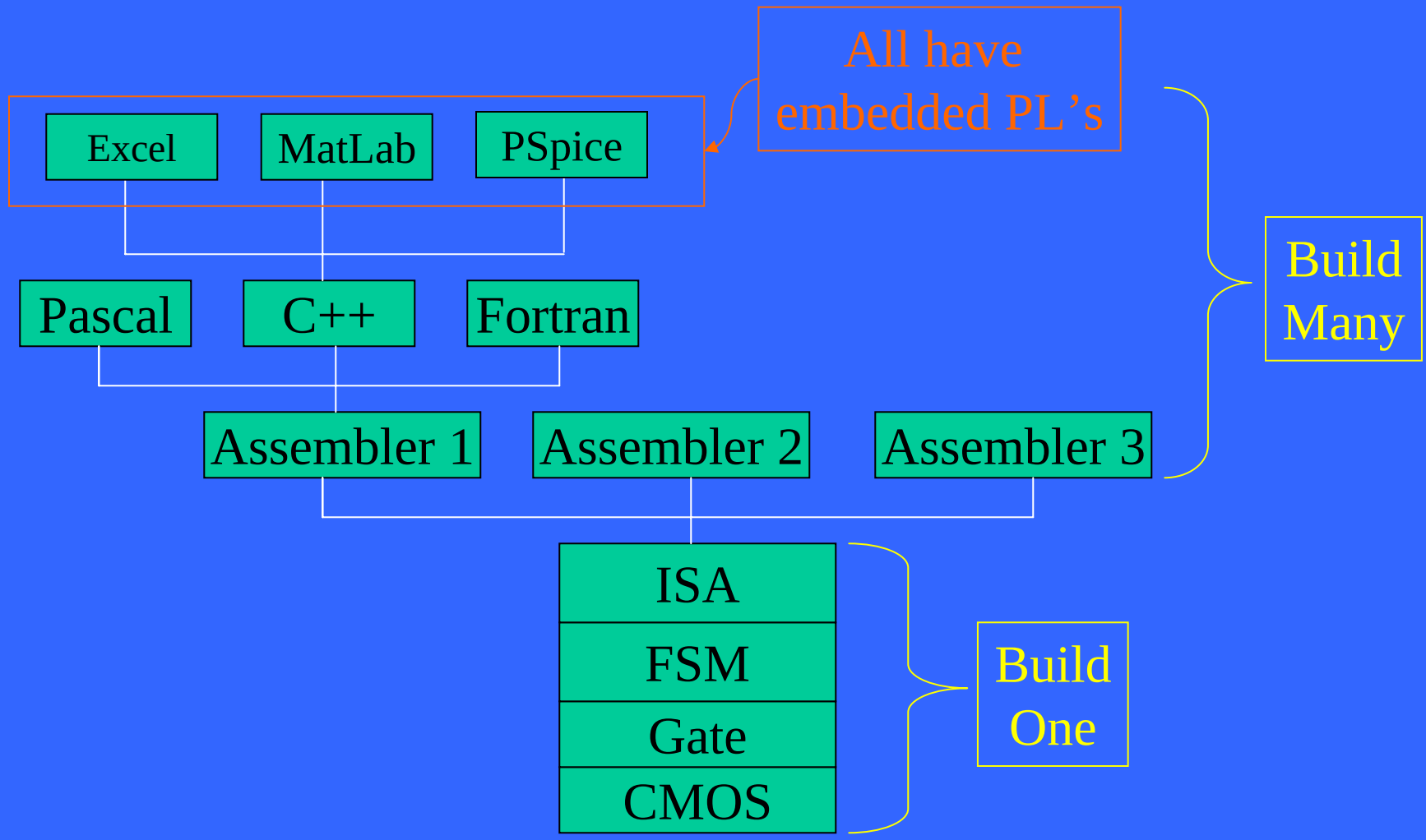

- Capable of Emulating Every other TM
- Shown possible by Alan Turing (1936)
- BIG IDEA: INTEPRETATION!!!

# 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?

# Computing in Perspective

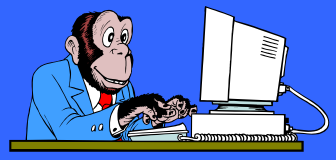



Interpreter Design Demands Programming Language Design

# 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

Because we know of no other way of doing anything



# Church's Thesis

**Alonso Church**

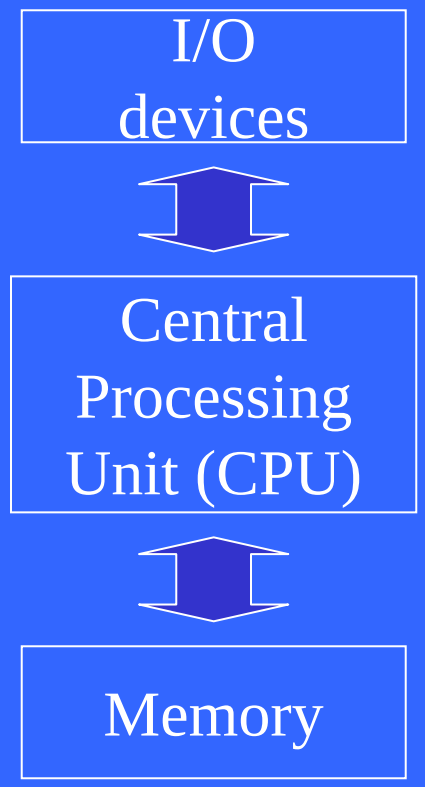
"Any realizable computing device can be simulated by a Turing machine"

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

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

# The (John) Von Neumann Architecture
## (late 40's)


John von Neumann, 1950's




Allow communication with outside world

Interprets instructions

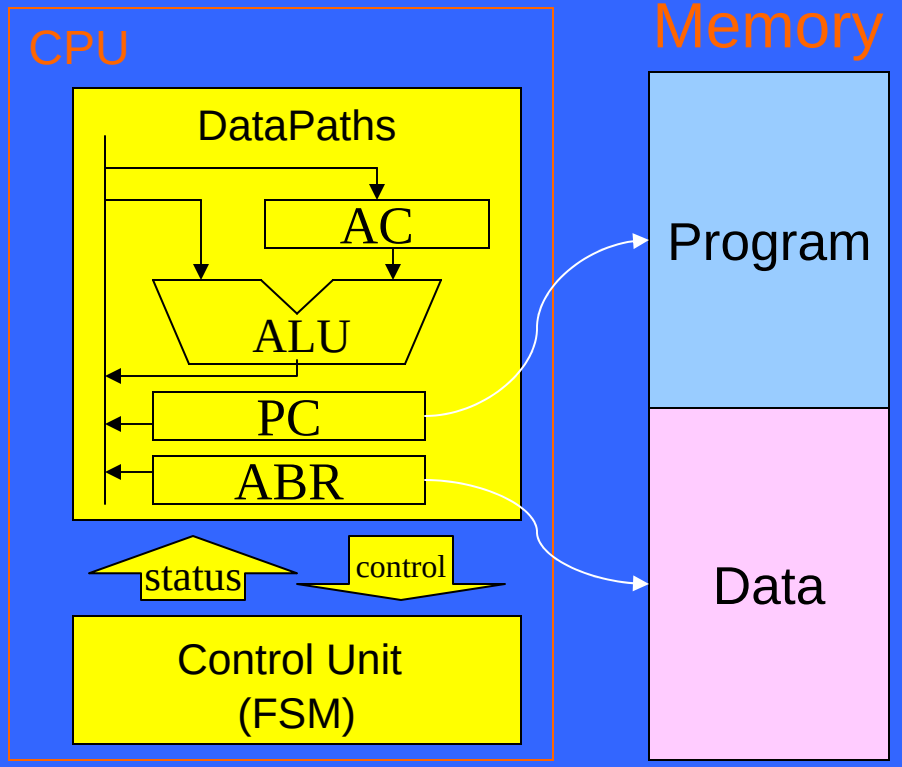
Stores both programs and data

**After 60 years … most processors still look like this!**

# Practical Universal Computers

## (John) Von Neumann Architecture (1945)


John von Neumann, 1950's




This looks just like a TM Tape

CPU is a universal TM

An interpreter of some programming language (PL)