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

Problem → Information → Compute → Information → Solution

encode → compute → decode

Problem:

Information:
0110110
0101010
0101...

Information:
0110110
0101010
0101...

Solution:
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

encode

compute

decode

Problem

0110110
01010100101...

encode

compute

decode

Problem

0110110
01010100101...

0110110
01010100101...

Solution
Computability

All Integer Functions (IF) ≟ Computable Integer Functions (CF)

IF ≠ CF

Halting Problem
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.

Want to describe the set of computable functions?

Program P → Compute → P Halts on I?
Input I → Cannot Build This
Mathematical Computers: The Turing Machine (1936)

Alan Turing

Tape Head
Infinite I/O TAPE
FSM

Input Symbol
Next State
Write Symbol
Head Movement

0/{1,0,R}
1/{0,1,L}
Mathematical Computers: The Turing Machine (1936)

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: INTERPRETATION!!!
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

All have embedded PL’s

Build Many

Build One

Excel
MatLab
PSpice
Pascal
C++
Fortran
Assembler 1
Assembler 2
Assembler 3
ISA
FSM
Gate
CMOS

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 lower layer

Because we know of no other way of doing anything
Church’s Thesis

“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)

I/O devices  
Allow communication with outside world

Central Processing Unit (CPU)  
Interprets instructions

Memory  
Stores both programs and data

After 60 years … most processors still look like this!
Practical Universal Computers
(John) Von Neumann Architecture (1945)

CPU is a universal TM
An interpreter of some programming language (PL)

This looks just like a TM Tape