Computer Architecture

Introduction

Debadatta Mishra, CSE, IITK

Abstraction

- What is abstraction?
- What are the different perspectives of abstraction?
- Is abstraction good or bad?
- This is a computer systems course; why are we bothered about abstraction?

Agenda: Motivate this course (CS422) by answering the above questions

Abstraction in day-to-day life (utility)

- Abstraction plays a crucial role in human life and progress

To grow crop, we need not know Geology, Botany, Meteorology and Chemistry.

Abstraction in day-to-day life (utility)

- Abstraction plays a crucial role in human life and progress

To grow crop, we need not know Geology, Botany, Meteorology and Chemistry.

To cook a delicious meal, the chef need not know the details of farming

Abstraction in day-to-day life (utility)

- Abstraction plays a crucial role in human life and progress

To grow crop, we need not know Geology, Botany, Meteorology and Chemistry.

To cook a delicious meal, the chef need not know the details of farming

You and me are not required to know the art of cooking to enjoy a good meal

Abstraction (science and philosophy)

Definition (cognitive process) [1]

"The cognitive process of isolating, or abstracting, a common feature or relationship observed in a number of things, or the product of such a process."[1]

- Examples:
  - Algebraic systems (Mathematics)
  - Force (Physics)
  - And many more …

1. https://www.britannica.com/science/abstraction

Abstraction (computer science)

- In CS, abstraction can be simplified as “ a process of hiding the details” while “exposing an interface” for the user (next layer)
- Examples:
  - Abstract Data Types
  - OS
  - ISA

Abstraction is good!

- Abstraction makes our life and interactions simple, and productive
- Allows us not to be bogged down with underlying details
  - A chef can concentrate on creating new recipes (not on farming!)
  - Design efficient algorithms (without worrying about its implementation)

- When can dealing with abstraction be an issue?
- Alternatively, are there any need for “understanding the underlying details”?

Understanding: Interfaces and underlying details

- Understanding the interfaces of an abstraction is fine if the abstraction is flawless and does not require any enhancements
- Understanding of the underlying details of an abstraction improves quality
- Example
  - A chef can select ingredients based on their farming method!
  - Better algorithms can be designed if we know the capabilities of underlying hardware

Example program and lessons

- Performance of different actions of demo program can be explained (and enhanced) if we understand the details of the underlying abstraction
  - OS details: usage of mmap( ) and malloc( )
  - Architectural details?