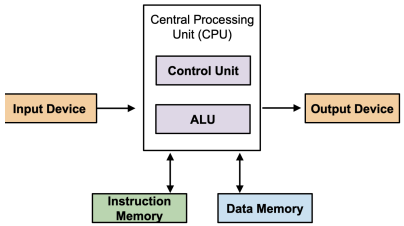
CS2106
AY22/23 Sem 2
github.com/securespider

01. Introduction

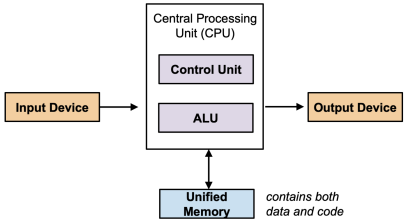
OS - Program that acts as an intermediary between user and hardware

Different architectures

Harvard architecture



Von Neumann architecture



Difference Separate vs common storage pathway for code and data

Why do we need OS?

Mainframe

Old analog "computers" using physical cards for programming

Improvements

- Problem: Batch processing inefficient
- Solution: Multiprogramming
  - Loading multiple jobs that runs while other jobs using I/O
  - Overlapping computation with I/O
- Problem: Only one user
- Solution: Time sharing OS
  - Multiple concurrent users using terminals
  - User job scheduling
  - Memory management
  - **Hardware virtualization** - Each program executes as if it had all resources

Motivation

1. Abstraction
   - Hide low level details and present common, high-level functionality to users
2. Resource allocation
   - Allow concurrent usage of resource and execute programs simultaneously
   - Arbitrate conflicting request fairly and efficiently
3. Control programs
   - Restrict resource allocation
   - Security, protection and error prevention
   - Ensure proper use of device

Advantage

- Portable and flexible
- Use computer resources efficiently

Disadvantage

- Significant overhead

OS vs User Program

Similarities

- Both softwares

Difference

- OS runs in **kernel mode** - Access to all hardware resources
- User programs run in **User mode** - Limited access
- User programs use syscalls to communicate with OS for hardware processes

Why OS dont occupy entire hardware layer

- Slow to have all operations pass through intermediary
- User programs can have direct interaction with hardware (eg. Arithmetic) during low risk operations

OS structure

Monolithic OS

- One big kernel program
- Well understood and has good performance
- Highly **coupled** - internal structure interconnected that unintentionally affect each other

Microkernel

- Small clean
- Basic and essential facilities
- IPC communication OR run external programs outside OS
- Robust and more **modular** - Extendible and maintainable
- Better isolation btw kernel and services
- Lower performance