# CSE216 Programming Abstractions

## Overview

YoungMin Kwon

SUNY Korea

# Major Topics to Cover

- Functional Programming
  - Procedural abstraction, Data abstraction, Modular abstraction
  - Advanced techniques: CPS, Stream
  - Lambda calculus
- Imperative Programming
  - C, memory management
  - Event driven programming

SUNY Korea

# Programming Language Paradigms

- Imperative programming
  - Relying on assignments
  - E.g. rand()

```
int seed = 1;
int rand() {
    seed = (seed * 16807) % 0x7fffffff;
    return seed;
}
```

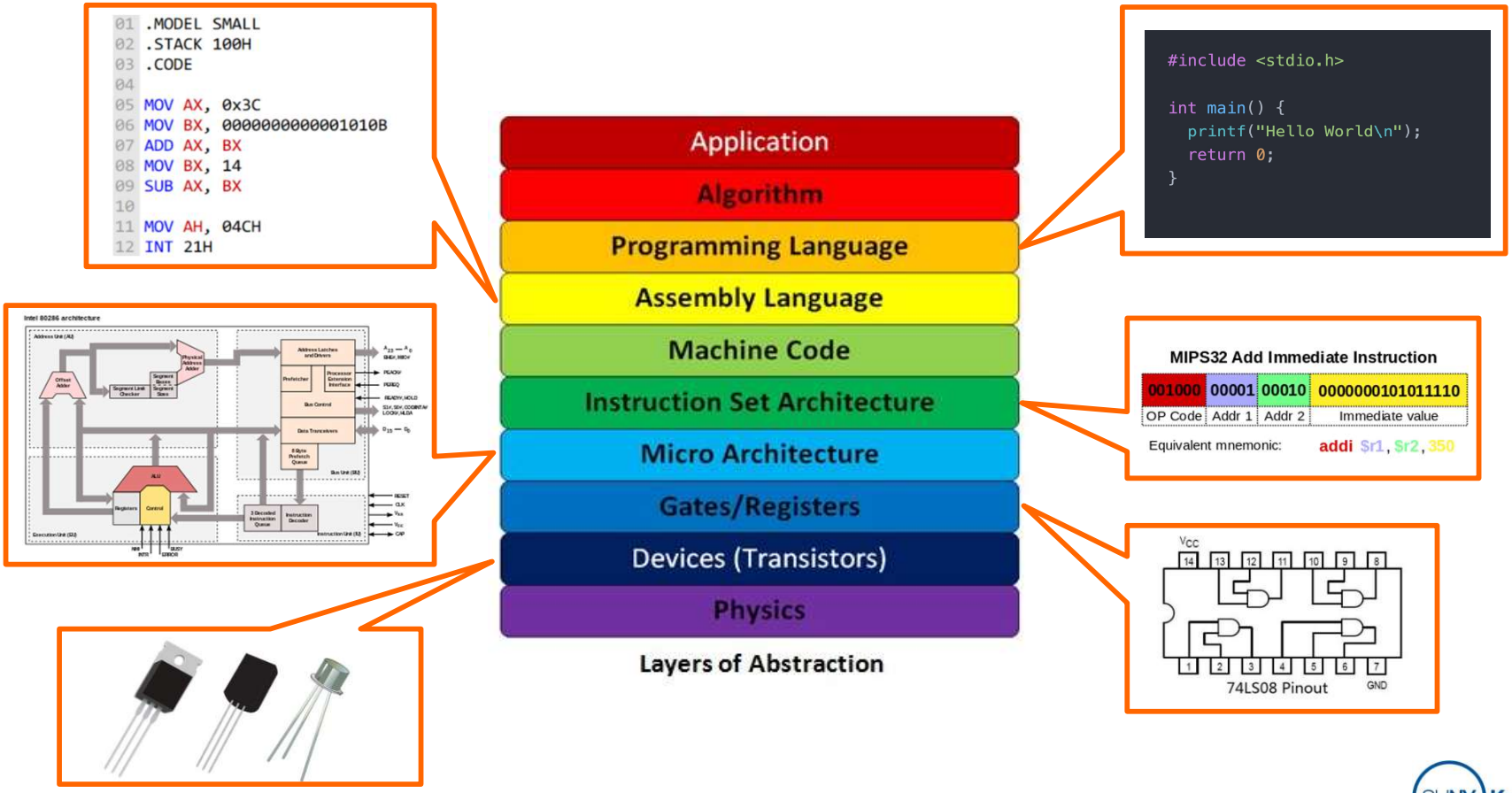
- Functional programming
  - No side effects: functions are mathematical functions
  - E.g.  sin(x)

- Logic programming
  - Search goals through logical rules and axioms

SUNY Korea

# Abstraction

- Abstraction: hide unnecessary details and provide the most essential information



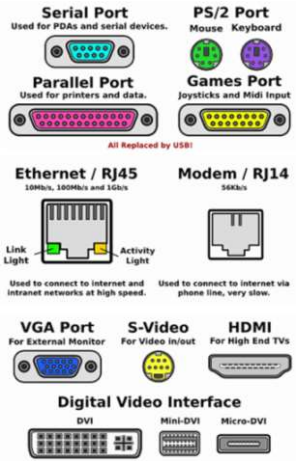
# Procedural Abstractions

- Elements of programming
  - Primitive expressions
  - Means of combination
  - Means of abstraction
- Topics on procedures
  - Variable binding and scoping
  - High-order functions
    - integrator, differentiator, map, filter, …

SUNY Korea

# Data Abstractions

- Elements of data
  - Primitive data
  - Compound data (tuple, list, record, …)
  - Data abstraction
    - Isolating data representation from data usage




- What is meant by data
  - Constructor, selector, and conditions that they meet

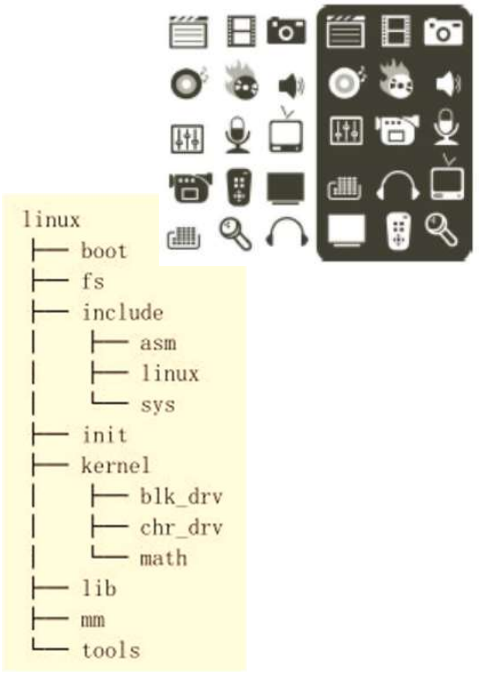
```
first( pair(1, 2) ) = 1, second( pair(1, 2) ) = 2
```

- Abstraction barriers

SUNY Korea

# Modula Abstraction

- To build a large system
  - Needs an organizational principle
    - Structure large systems
    - Divide naturally into coherent parts




- Objects
  - Viewing a large system as a collection of distinct objects
  - Local states, state variables, assignments

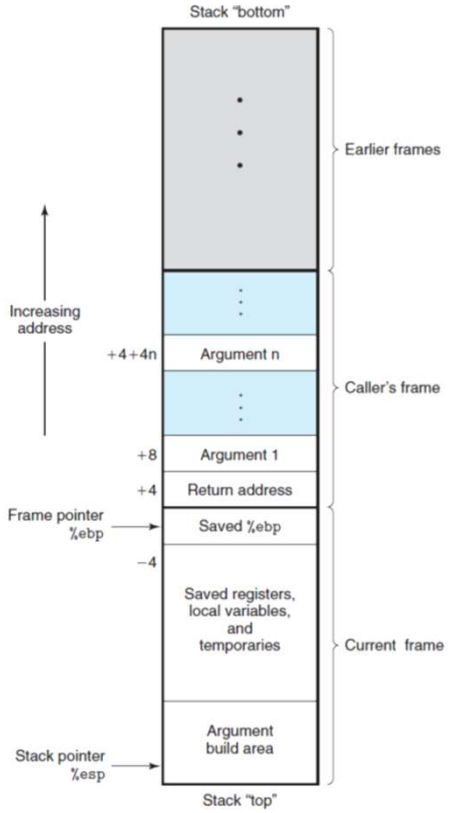
SUNY Korea

# Useful Techniques for Functional Programming

- Make recursive calls 1M times

```
int sum(int n)  {
    if(n == 0)
        return 0;
    else
        return n + sum(n-1);
}

sum(1000,000,000);
```




  - OCaml program crashes. Why?
  - Continuation Passing Style (CPS)

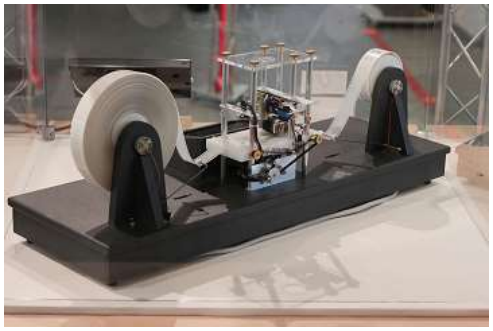
- rand() in functional programming?
  - Streams

SUNY Korea

# Lambda Calculus

- Lambda Calculus ≡ Turing Machine

```
expr = name
     | λ name . expr
     | expr1 expr2
```



- Lambda expressions
  - Numbers, Arithmetic Opr: Church numeral
  - Boolean, Boolean Opr: Church Boolean
  - Recursion: Y-combinator


SUNY Korea

# Type System

- Type checking

```
x = 1 + 2

x = 1 + False

if "Hello World": …
```

- Type inference
  - What is the type of f?

```
f = lambda x: x + 1
```

SUNY Korea

# Imperative Programming

- C Programming language
  - Procedures
  - Parameter passing modes
  - C data types
    - Primitive types, arrays, pointers, structures
  - Variable scope and lifetime

SUNY Korea

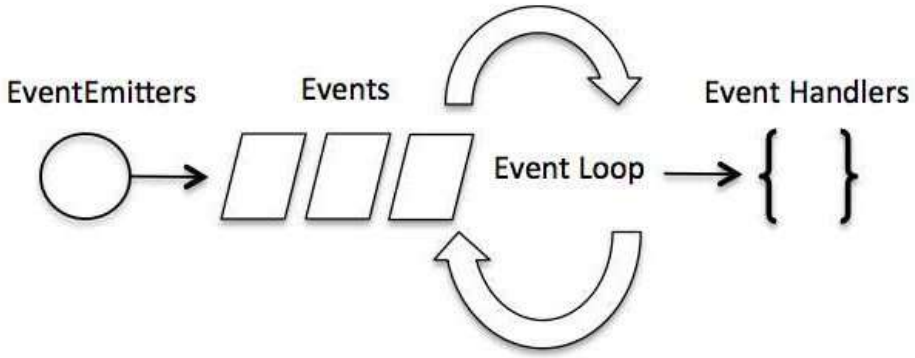
# Imperative Programming

- Dynamic memory allocation
- Memory management
  - Automatic garbage collection
  - Reference counting

SUNY Korea

# Event Driven Programming

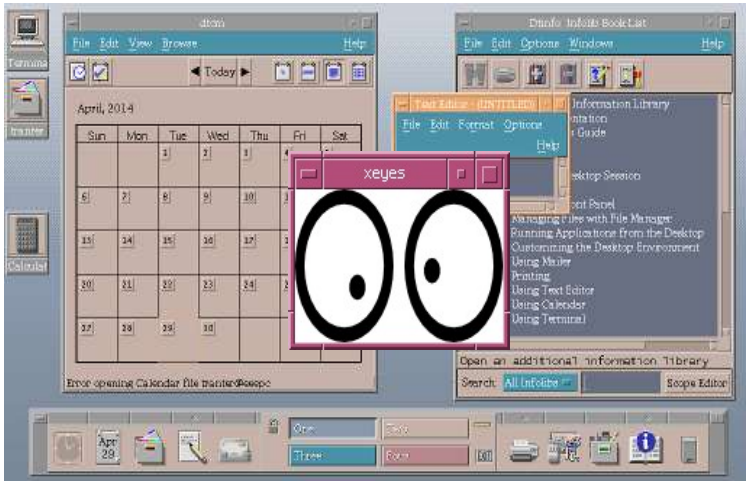
- Event driven programming
  - You do not call me; we will call you
  - Applications register callback functions (signal handlers)
  - Event loop will call callbacks later

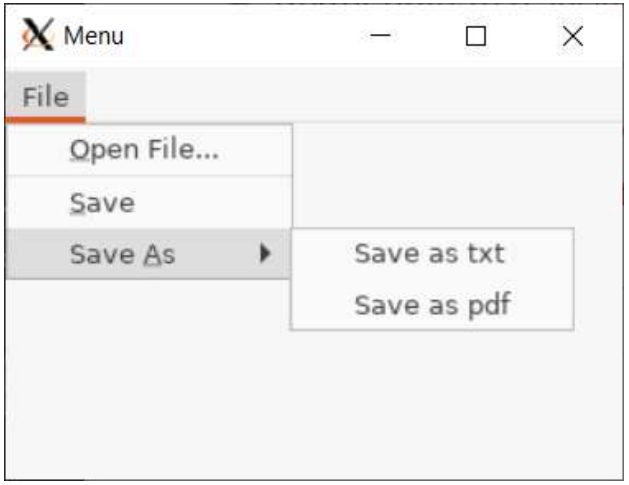



SUNY Korea

# Event Driven Programming

- X Window Programming




SUNY Korea