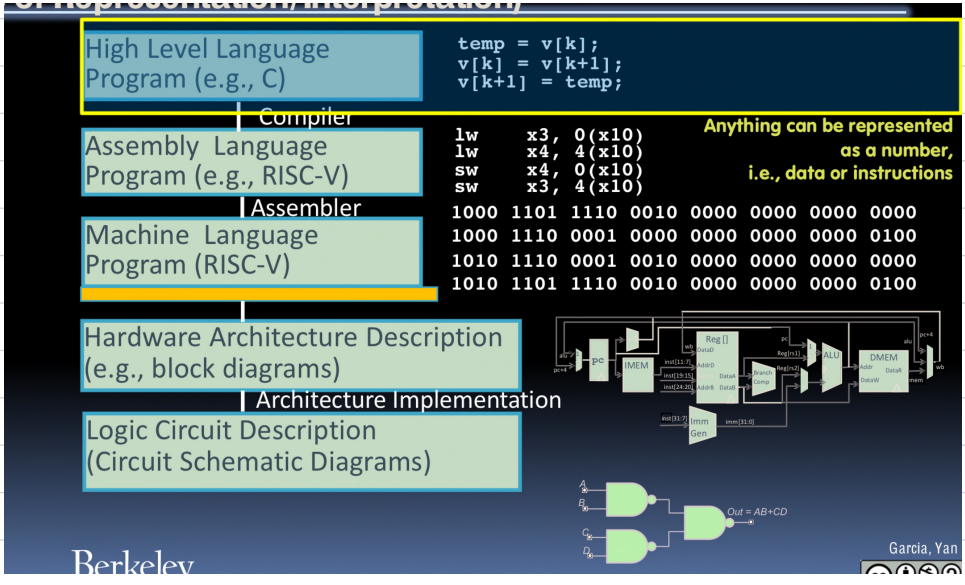
Introduction to the C programming Language

Abstraction



Introduction to C

C enabled first operating system not written in assembly language.

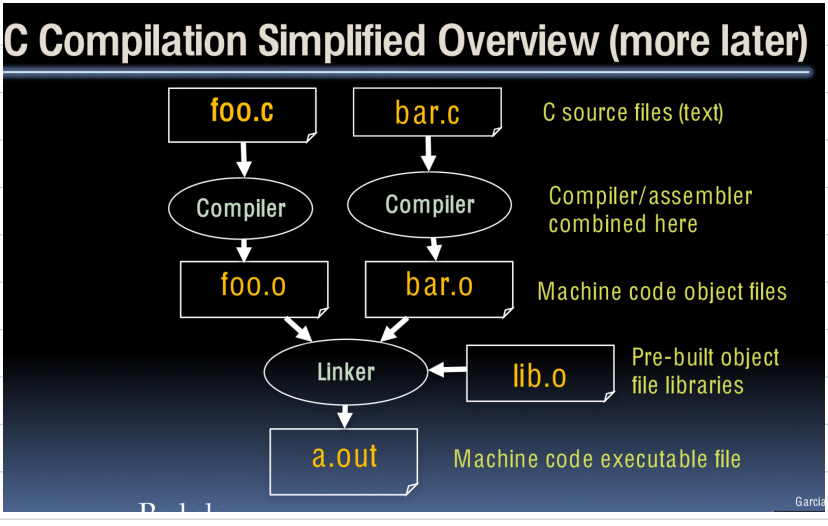
We can write programs that allow us to exploit underlying features of the architecture.

- Rust, "C-but-Safe": By the time your C is (theoretically) correct w/all necessary checks it should be no safer than Rust
- Go, "Concurrency": Practical concurrent programming to take advantage of modern multi-core microprocessors.

Compile v. Interpret

C compilers map C programs directly into architecture-specific machine code (string of 1s and 0s)

For C, generally a two part process of compiling: .c files to .o files, then linking the .o files into executables



Compilation: Advantages

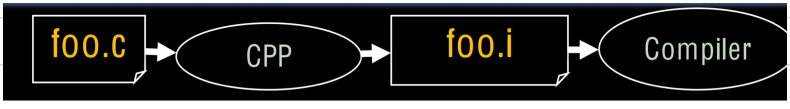
- Reasonable compilation time: enhancements in compilation procedure (Makefiles) allow only modified files to be recompiled
- Excellent run-time performance: generally much faster than Scheme or Java for comparable code (because it optimizes for a given architecture)

Compilation: Disadvantages

- Compiled files, including the executable, are architecture-specific, depending on processor type (e.g., MIPS vs. x86 vs. RISC-V) and the

operating system (e.g., Windows vs. Linux vs. MacOS)
- Executable must be rebuilt on each new system
- "Change → Compile → Run [repeat]" iteration cycle can be slow during development

C Pre-Processor (CPP)



- C source files first pass through macro processor, CPP, before compiler sees code
- CPP replaces comments with a single space
- CPP commands begin with "#"
  - `#include "file.h"` /* Inserts `file.h` into output */
  - `#include <stdio.h>` /* Looks for file in standard location, but no actual difference! */
  - `#define PI (3.14159)` /* Define constant */
  - `#if/#endif` /* Conditionally include text */
- Use `-save-temps` option to gcc to see result of preprocessing
  - Full documentation at: `http://gcc.gnu.org/onlinedocs/cpp/`

Garcia

C Syntax

- What evaluates to FALSE in C?
  - 0 (integer)
  - Null (pointer)
  - Boolean types provided by C99's Std bool.h
- What evaluates to TRUE in C
  - ... everything else ...

- Same idea as in Scheme
  - Only #f is false, everything else is true!

- Types can't change. E.g, `int var = 2;`

Type	Description	Example
`int`	Integer Numbers (including negatives) At least 16 bits, can be larger	`0, 78, -217, 0x7337`
`unsigned int`	Unsigned Integers	`0, 6, 35102`
`float`	Floating point decimal	`0.0, 3.14159, 6.02e23`
`double`	Equal or higher precision floating point	`0.0, 3.14159, 6.02e23`
`char`	Single character	`'a', 'D', '\n'`
`long`	Longer `int`, Size >= `sizeof(int)`, at least 32b	`0, 78, -217, 301720971`
`long long`	Even longer `int`, size >= `sizeof(long)`, at least 64b	`31705192721092512`

Integers: Python vs. Java vs. C

- C: `int` should be integer type that target processor works with most efficiently
- Only guarantee:
  - `sizeof(long long) ≥ sizeof(long) ≥ sizeof(int) ≥ sizeof(short)`
  - Also, `short` >= 16 bits, `long` >= 32 bits
  - All could be 64 bits
  - This is why we encourage you to use `intN_t` and `uintN_t`!!

Language	`sizeof(int)`
Python	>=32 bits (plain `int`s), infinite (long `int`s)
Java	32 bits
C	Depends on computer; 16 or 32 or 64

Consts and Enums in C

Constant is assigned a typed value once in the declaration; value can't change during entire execution of program

const int/double/float/... name = ...;

Enums: a group of related integer constants

enum cardsuit {CLUBS, DIAMONDS, HEARTS, SPADES}

enum color {RED, GREEN, BLUE};

Typed Functions in C

- You have to declare the type of data you plan to return from a function
- Return type can be any C variable type, and is placed to the left of the function name
- You can also specify the return type as void
- Also need to declare types for values passed into a function
- Variables and functions MUST be declared before used

```
int number_of_people() {return 3;}
float dollars_and_cents() {return 10.33;}
```

Structs in C

- Typedef allows you to define new types.

```
typedef uint8_t BYTE;
BYTE b1, b2;
```

- Structs are structured groups of variables

```
typedef struct {
    int length_in_seconds;
    int year_recorded;
} SONG;

SONG song1;
song1.length_in_seconds = 213;
song1.year_recorded = 1994;

SONG song2;
```

```
song2.length_in_seconds = 248;
song2.year_recorded = 1988;
```

C Syntax: Variable Declaration

- All variable declarations must appear before they are used
- All must be at the beginning of a block
- A variable may be initialized in its declaration
  - if not, it holds garbage!