Check-In

Review - Parameters

Give an example of a program that would compile under both a pass-by-value and pass-by-reference scheme but gives different output under both.

Announcements

Administrivia

University of Kansas | Drew Davidson

EECS 665 COMPILER CONSTRUCTION

Runtimes

Previous Lecture

Review - Parameters

Vocabulary:

- lval/rval
- Memory references
- Arguments

Parameter Passing

- Call by value
- Call by reference
- Call by value-result
- Call by name

You Should Know

- What the vocab terms are, how they'd appear in error messages
- The difference between *formal* arguments and *actual* arguments
- The semantic effect of call-by-value and call-by-reference parameter passing schemes

Semantics

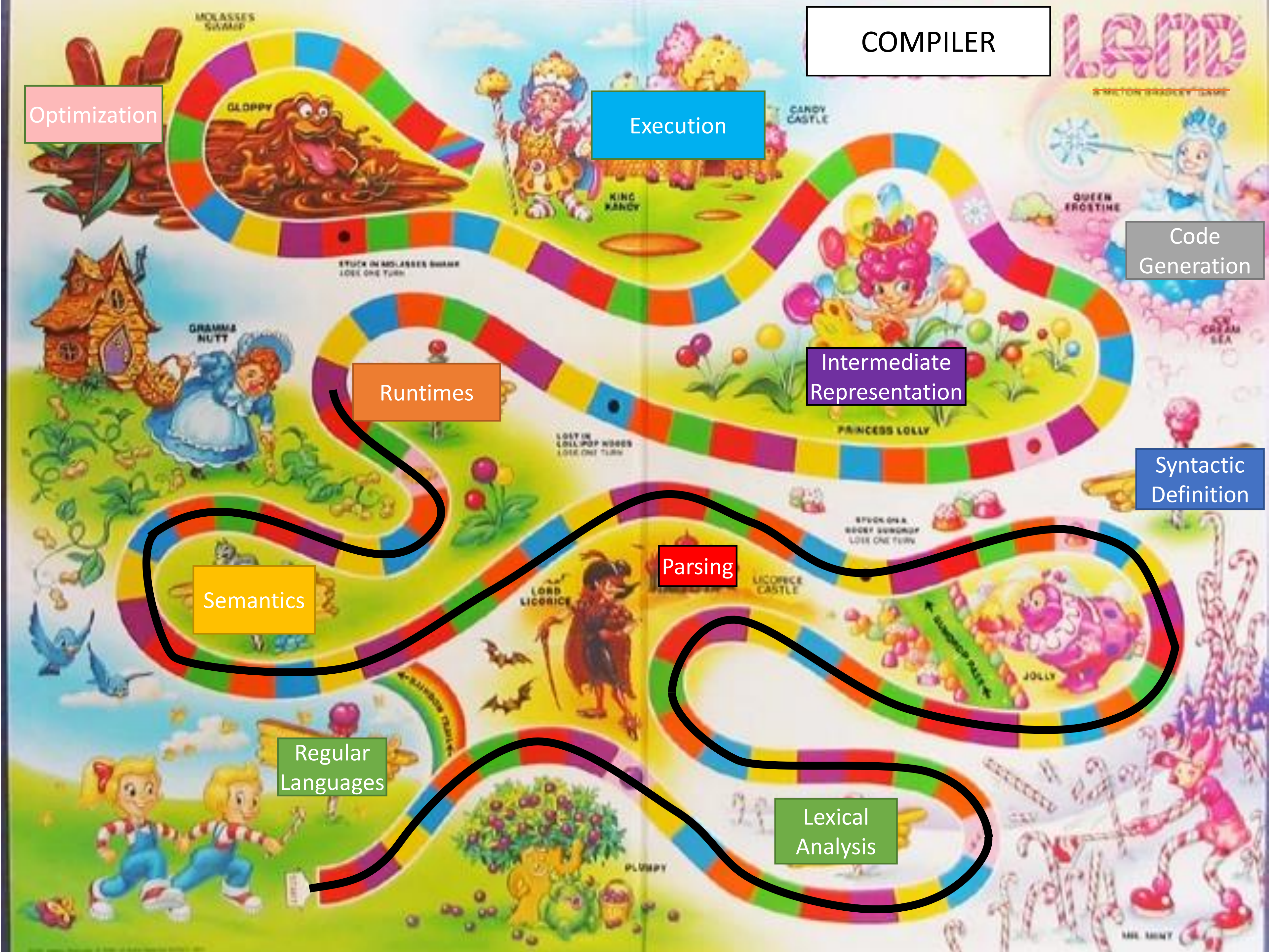
Lecture Outline

Runtimes

Runtimes

- Runtime Environments
- The semantic gap (again)
- Interpreters

Semantics



COMPILER LAND

Optimization

Execution

Code Generation

Intermediate Representation

Runtimes

Syntactic Definition

Parsing

Semantics

Regular Languages

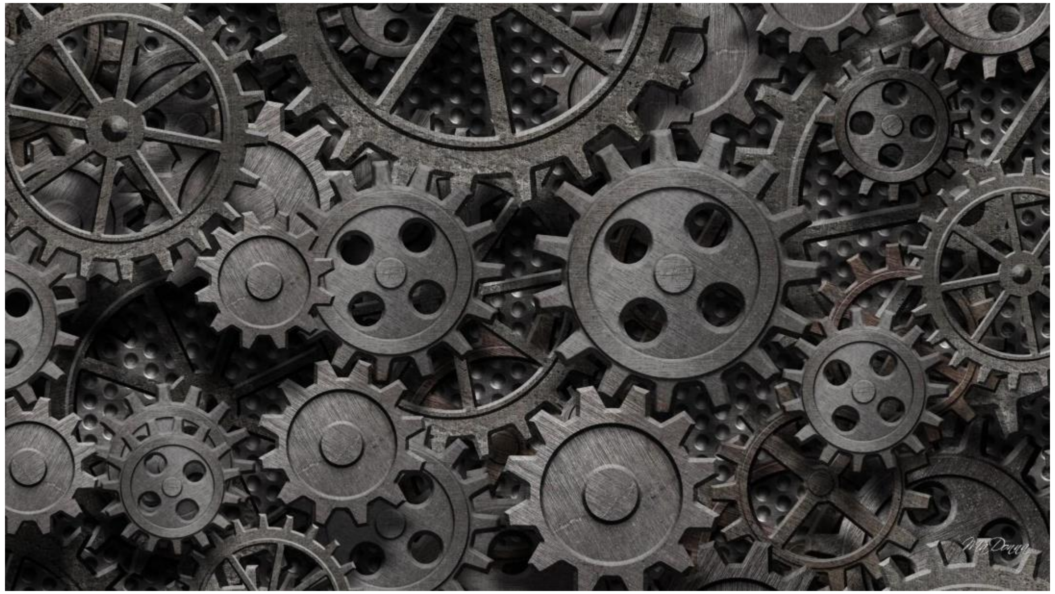
Lexical Analysis

Switching Gears: Targets

Runtime Environments – Setup

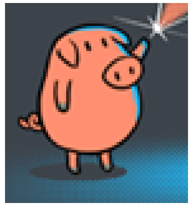
Time to look at how code is actually run

- For this we'll need to understand execution systems (runtimes)



Compilers: A Tasty Mix of Disciplines

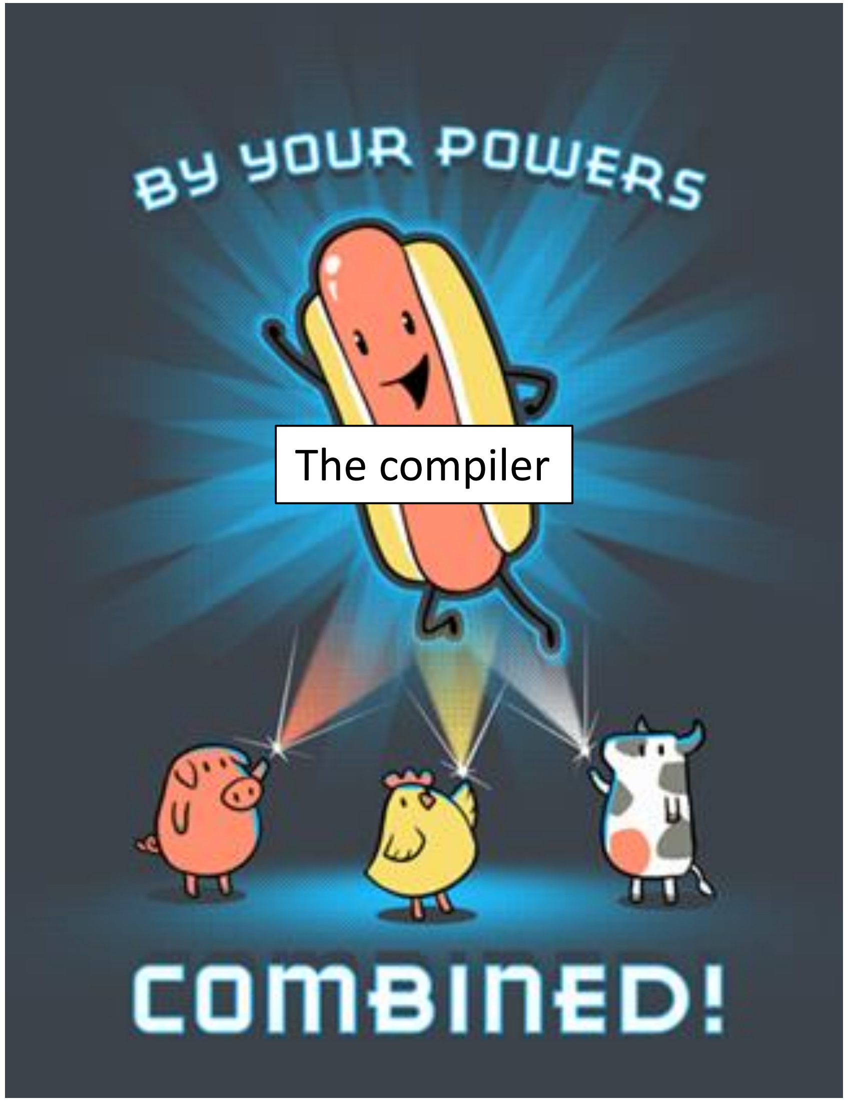
Runtime Environments - Setup



Front-end:
- Automata theory
- Algorithms

Middle-end:
- Software engineering
- Program semantics

Back-end:
- Emulation
- Architecture

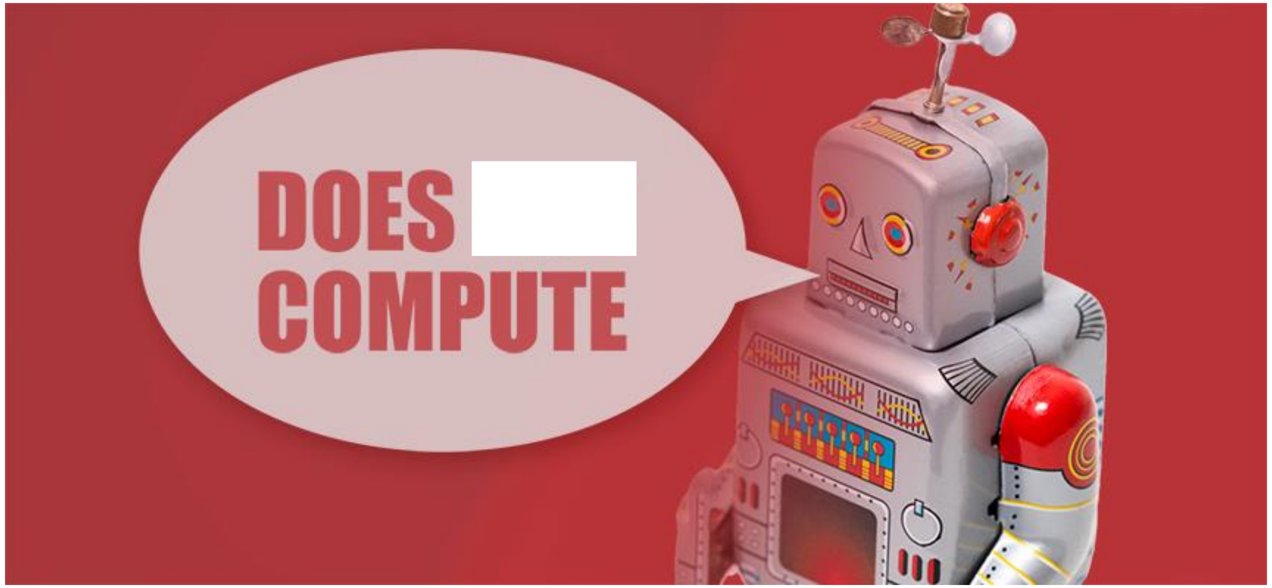


Relation to Compilers

Overview

Compilers job (roughly):
turn something from a non-executable format into that same thing in an executable format

Hard to pin down!

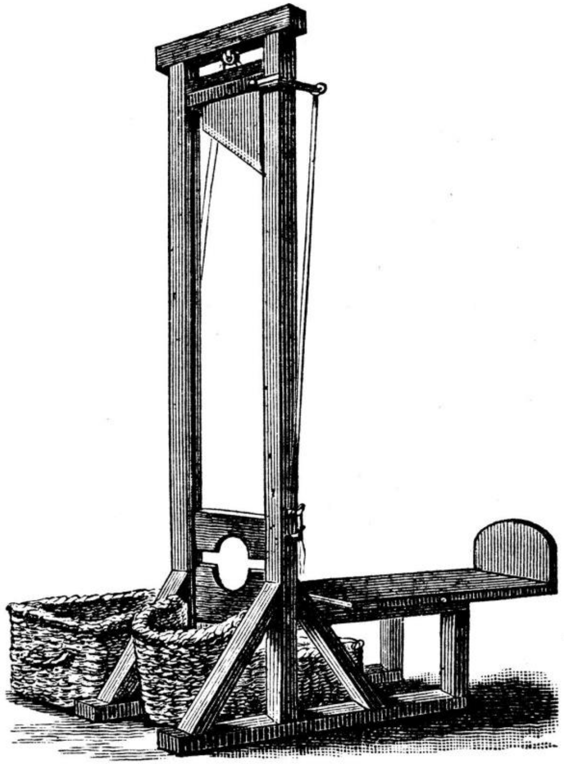


The Tools of Execution

Overview

Stepping back from compilers

What do we need for execute code?



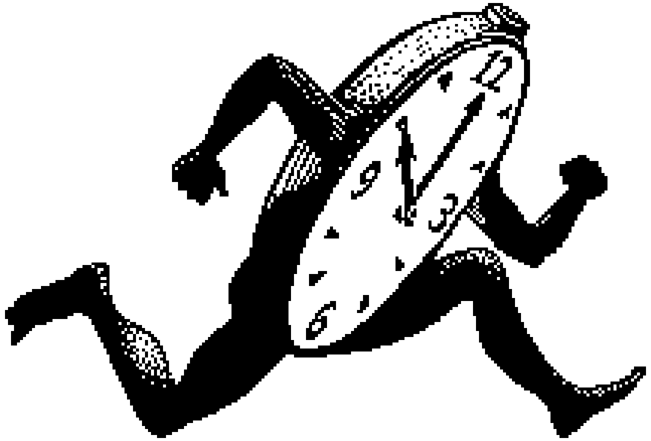
Not this kind of execution!

Runtime Environment Working Defn.

Underlying software and hardware configuration assumed by the program

- May include an OS (may not!)
- May include a virtual machine

May be co-designed with the programming language



Get it? "Run time"

Some Example Runtime Environments

Audience Participation: What are some example languages / runtime environments they provide?

Wait, why DO we need a Compiler?

“Obvious” Answer

- To implement a programming language
- To avoid dealing with target language directly

But is compilation the only option?

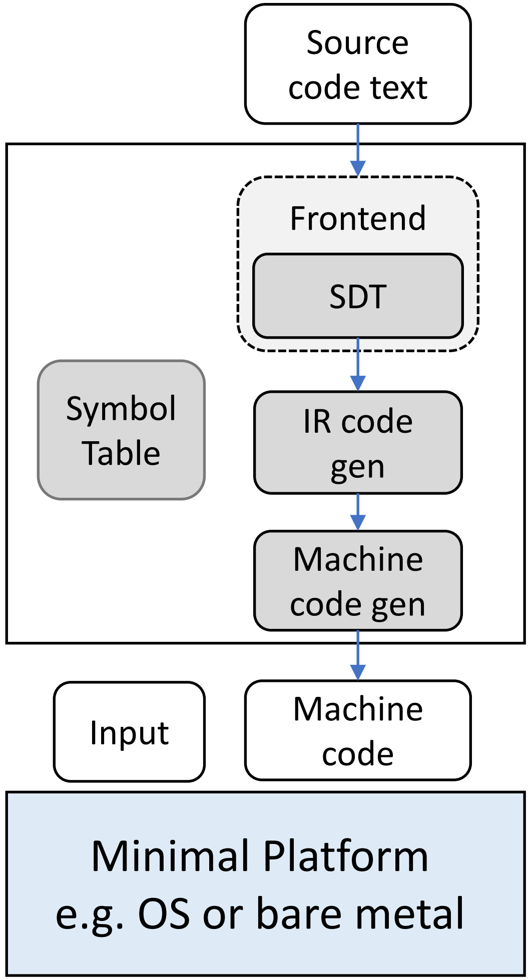
- Depends on your definition

A strawman

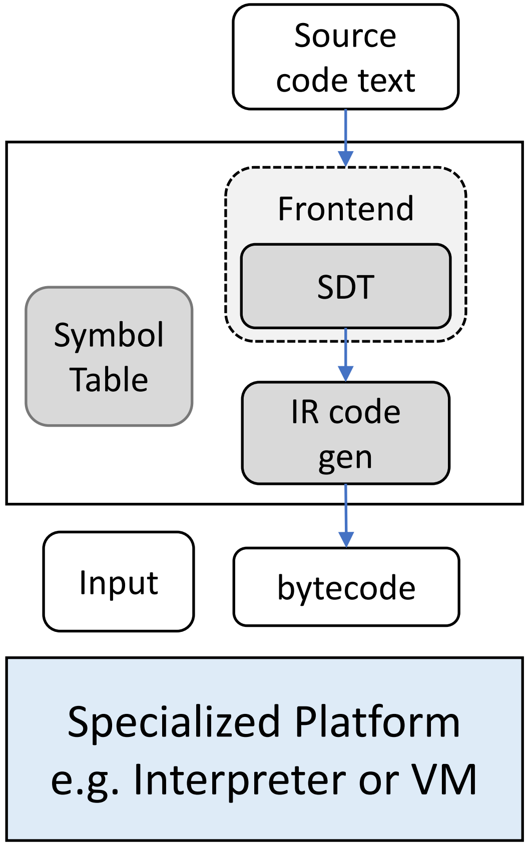
“Alternatives” to “Compilation”

Runtime Environments

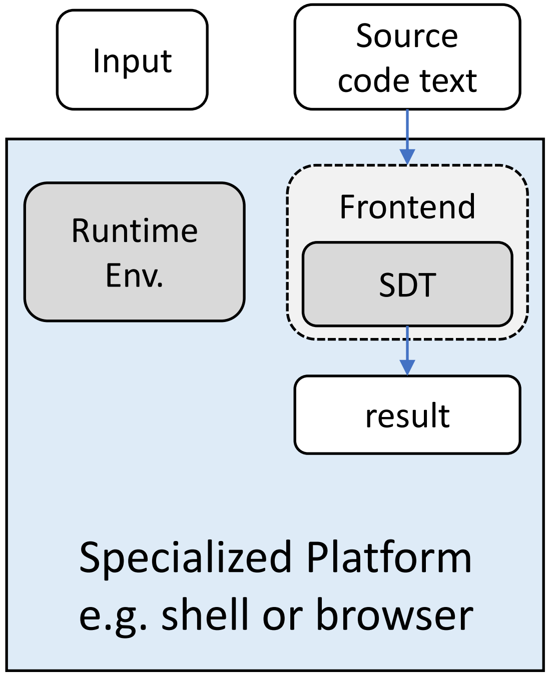
Compiling



Interpreting



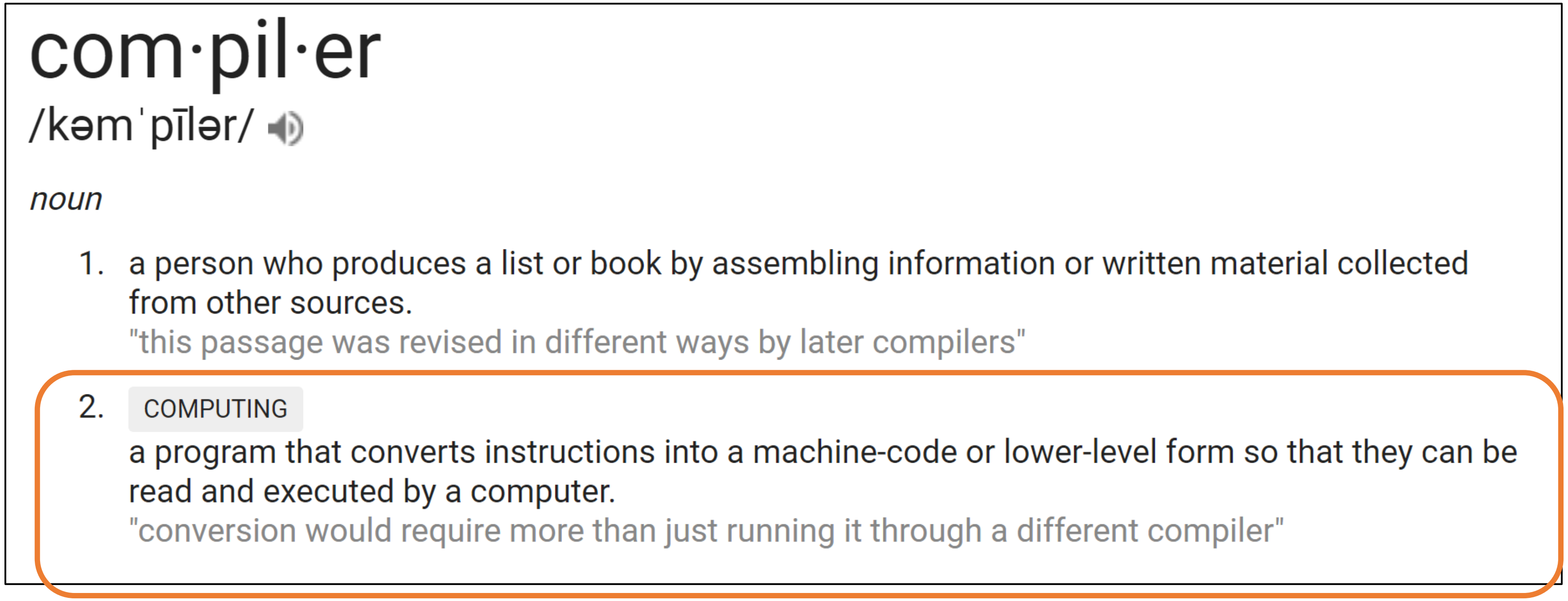
Scripting



Defining Compilers

Introduce IRs

Oxford languages dictionary



com·pil·er

/kəmˈpīlər/

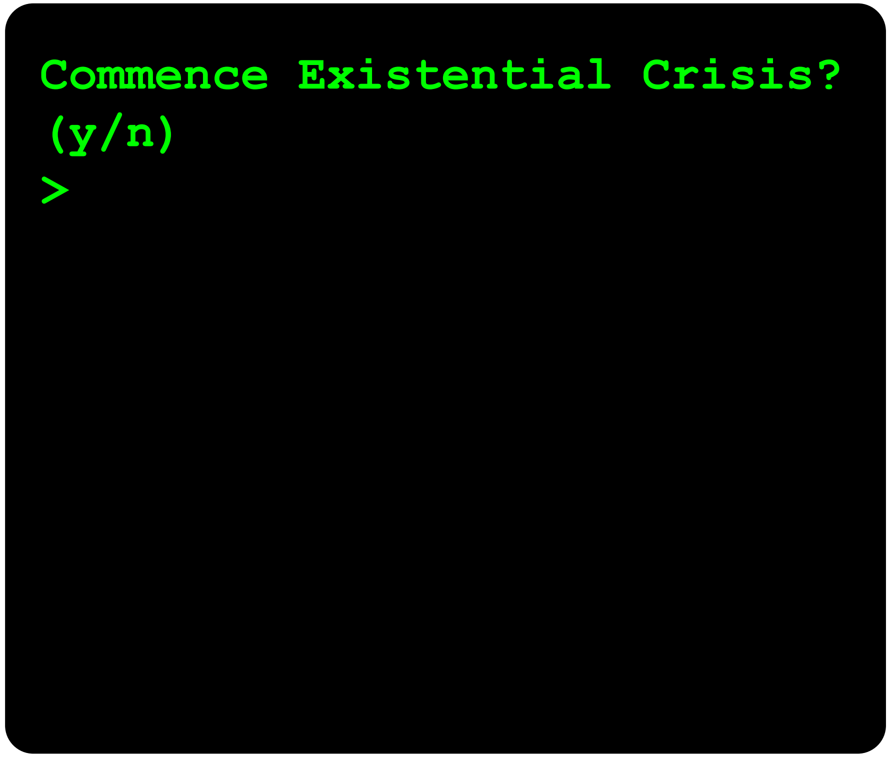
noun

1. a person who produces a list or book by assembling information or written material collected from other sources.
   "this passage was revised in different ways by later compilers"
2. COMPUTING
   a program that converts instructions into a machine-code or lower-level form so that they can be read and executed by a computer.
   "conversion would require more than just running it through a different compiler"

Rely on
scripting,
skip compilation

Then Why Compile at All?!?!?!?

Introduce IRs



```
Commence Existential Crisis?
(y/n)
>
```

Then Why Compile at All?!?!?!?

Introduce IRs

Rely on scripting, skip compilation

Analysis

- Error checking: predict bugs before they strike
- Optimization: generate better code statically

Write target code directly

Abstraction

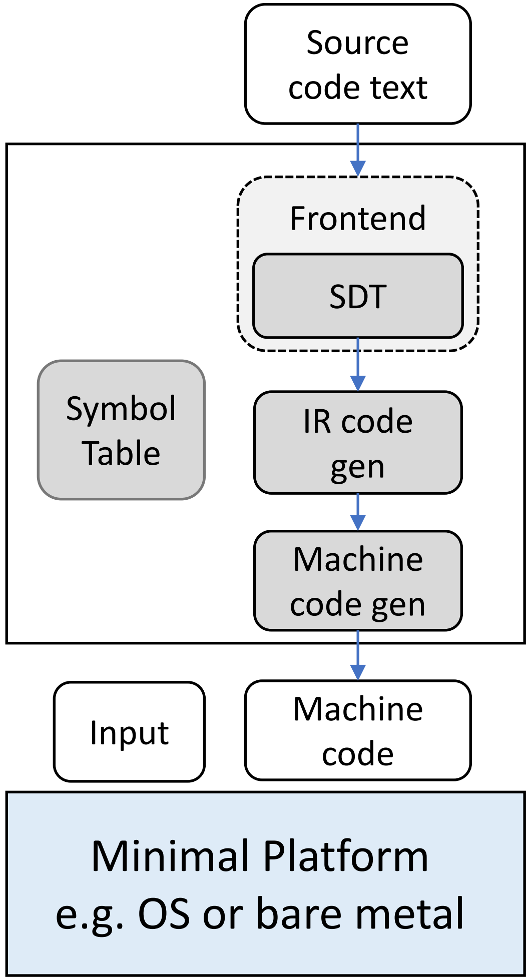
- Allow some distance from the target language

```
Commence Existential Crisis?
(y/n)
>
```

“Alternatives” to “Compilation”

Runtime Environments

Compiling



Source code text

Frontend

SDT

Symbol Table

IR code gen

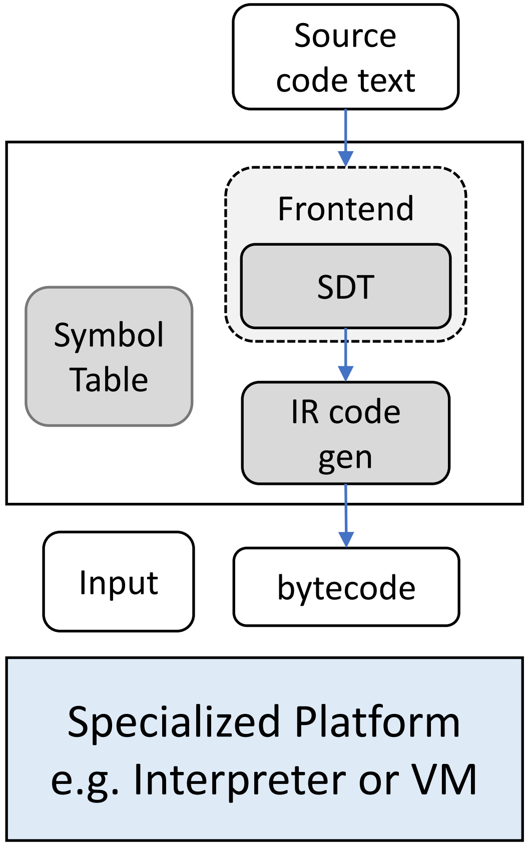
Machine code gen

Input

Machine code

Minimal Platform
e.g. OS or bare metal

Interpreting



Source code text

Frontend

SDT

Symbol Table

IR code gen

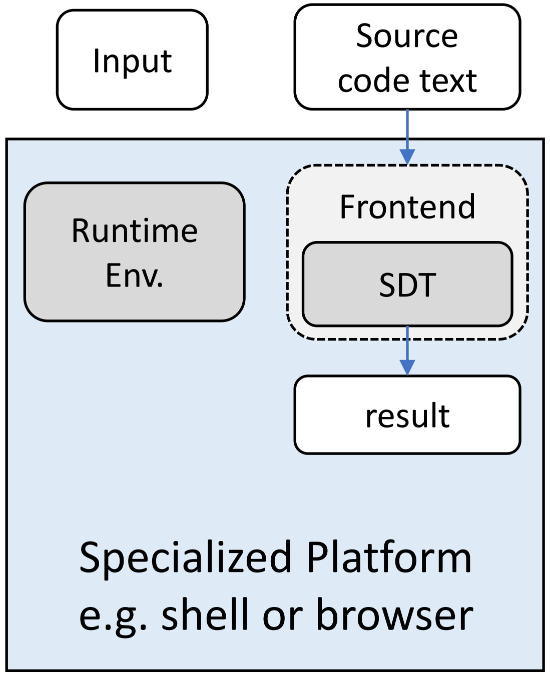
Input

bytecode

Specialized Platform
e.g. Interpreter or VM

Not really an alternative

Scripting



Input

Source code text

Runtime Env.

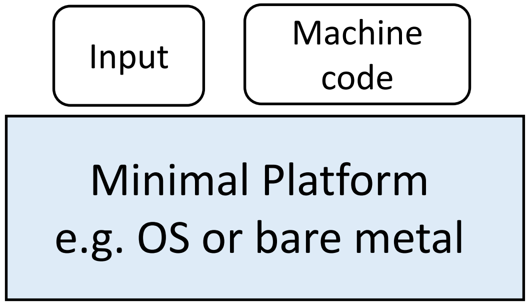
Frontend

SDT

result

Specialized Platform
e.g. shell or browser

Writing target code



Input

Machine code

Minimal Platform
e.g. OS or bare metal

Limitations that make large system building impractical

A Wider View of Compilation

Runtime Environments

Our definition

“A translator from source code to target code”

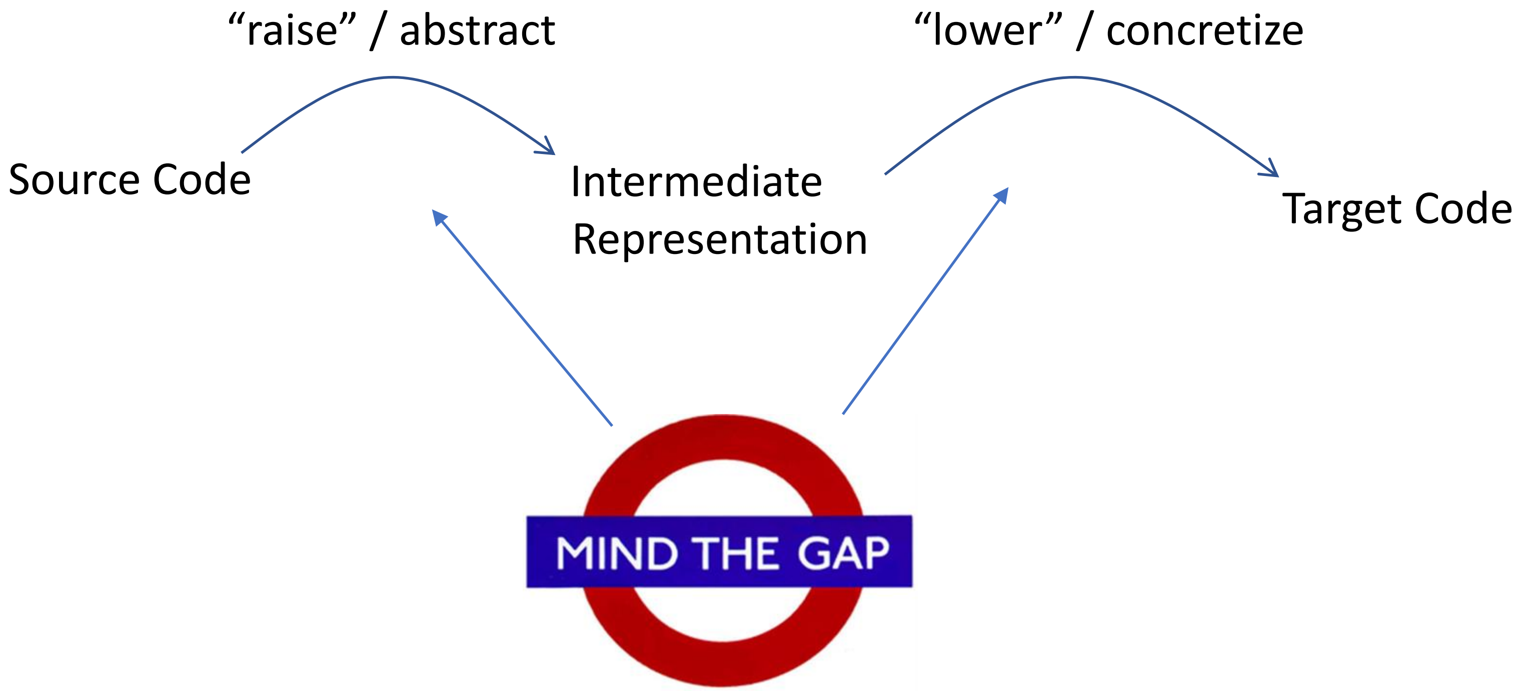
- May alter the source language for tractability
- May (or may not!) manipulate the target runtime for a variety of purposes

Another Semantic Gap

Runtime Environments

Difference between the specification in IR and executable

- Usually means shedding abstractions to concretize runnable code



"raise" / abstract

"lower" / concretize

Source Code

Intermediate Representation

Target Code

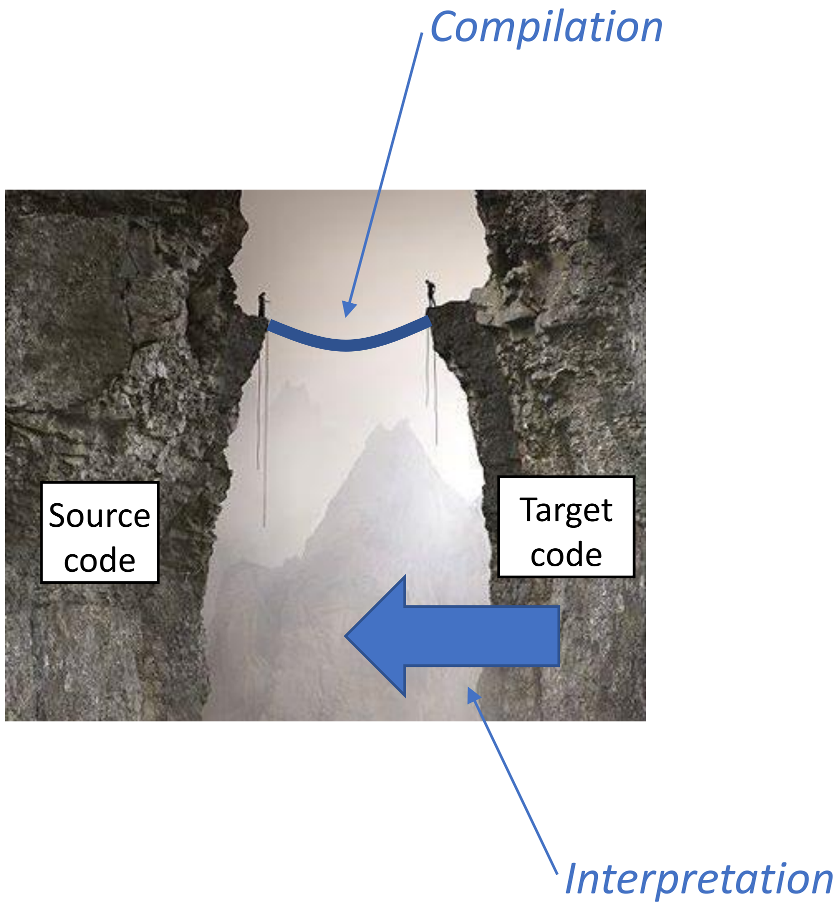
Bridging the Semantic Gap

We need code that is…

- Easy for humans to understand
- Easy for computers to run

There are various approaches to span this divide

- Build a translator (compiler)
- Move the target (interpreter)



Target Platforms

Runtime Environments

Static workload depends on the platform we target

- Real hardware
- Virtual hardware
- Shell

It's a platform!

Heavyweight Runtimes

Runtime Environments

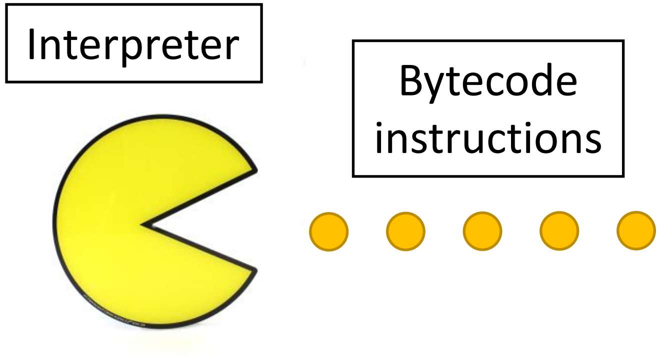
Interpreted languages often relegate a lot of work to their runtime

- Why?

Bytecode

Runtime Environments

An executable format that doesn't target hardware!



Interpreter

Bytecode instructions

Mediation Means Checking

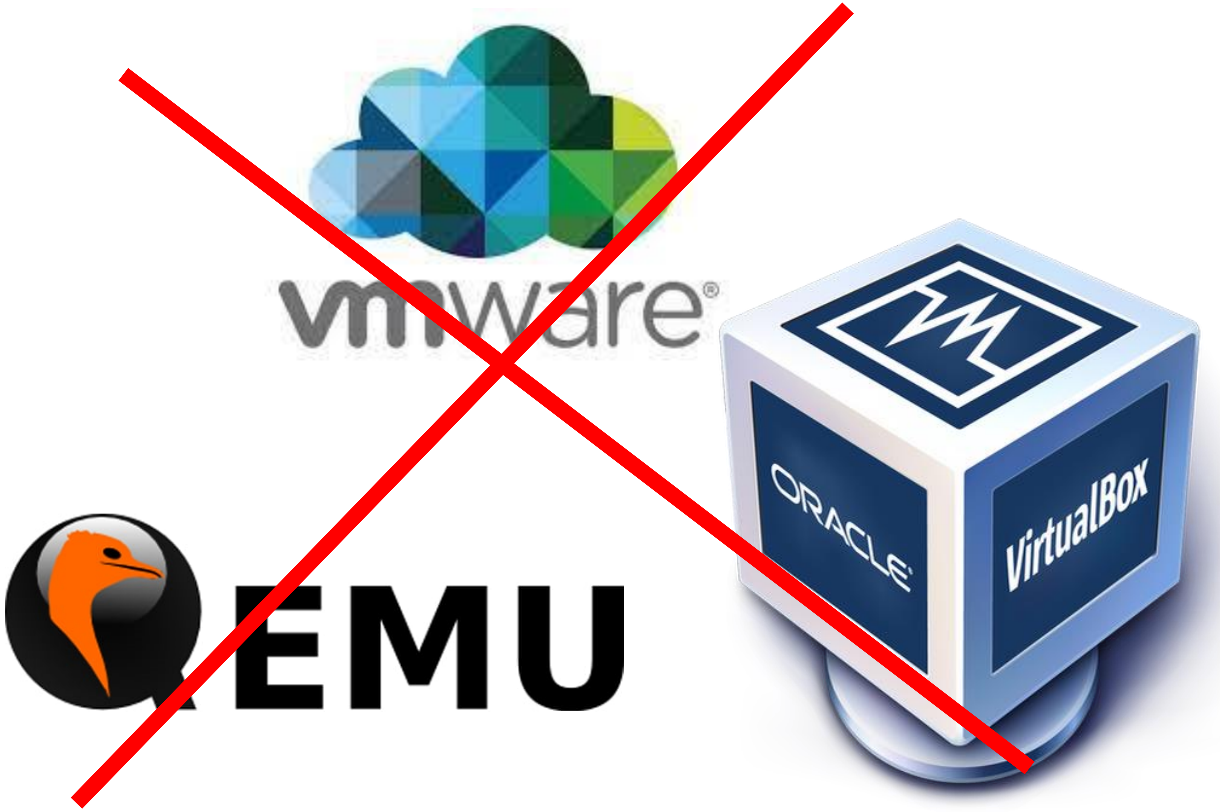
Runtime Environments

Many safety checks cannot be done until runtime

Virtual Machines

Runtime Environments

Provide a runtime environment for the abstract instruction set!



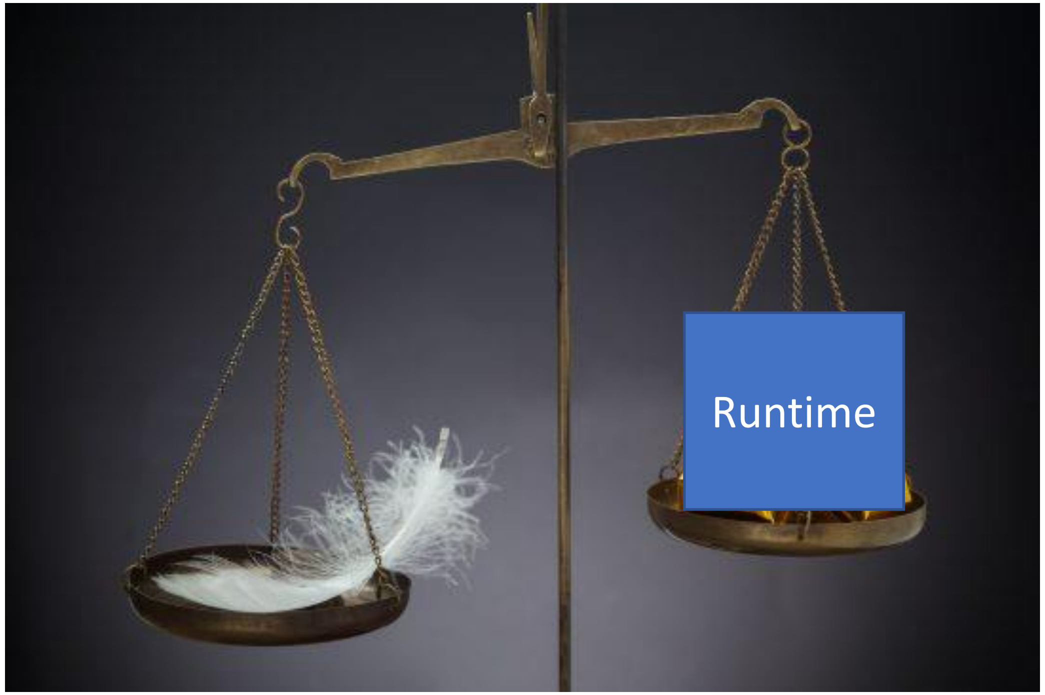
Less ambitious than whole-system virtualization

Lightweight Runtimes

Runtime Environments

Compiled languages often minimize their runtime

- Why?

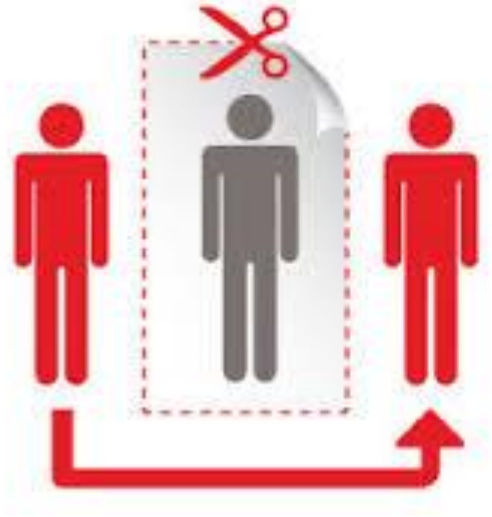


Lighter than a feather!

Mediation is Slow

Runtime Environments

- For the most part, OS does not control program
- Compiler’s job to use the environment as best as possible
  - This often means interfacing with the hardware architecture



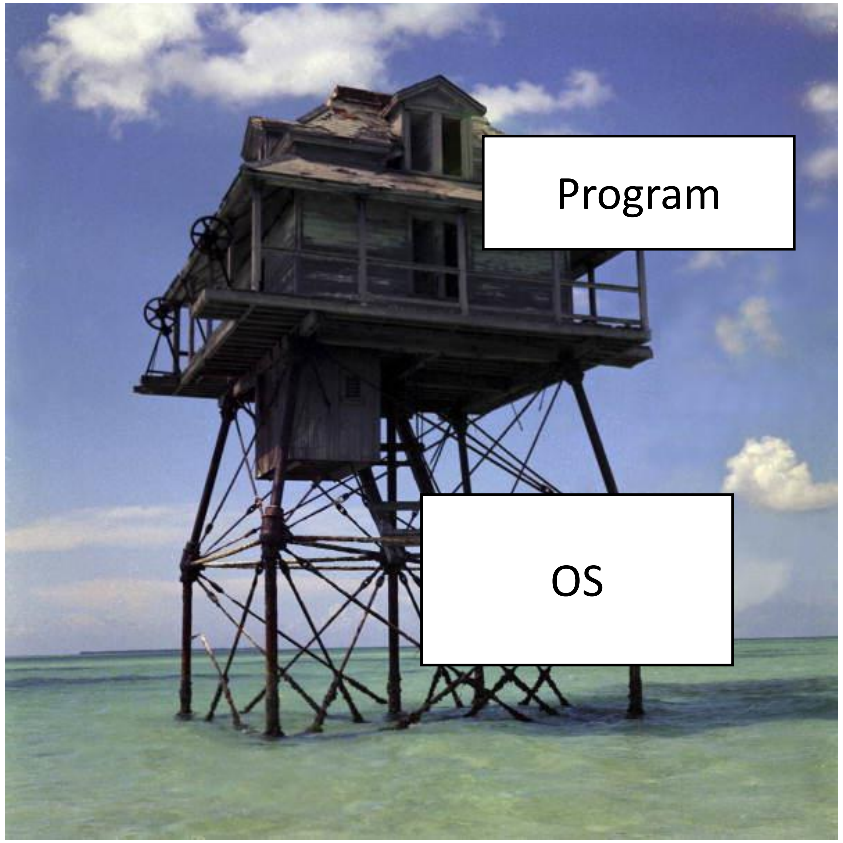
Cuttin’ out the middle-man

The Role of the OS/VM

Runtime Environments

Provides a platform for program

- System calls to access hardware
- “Illusion of uniqueness”
- Protects processes and system from each other



Our Language

Runtime Environments

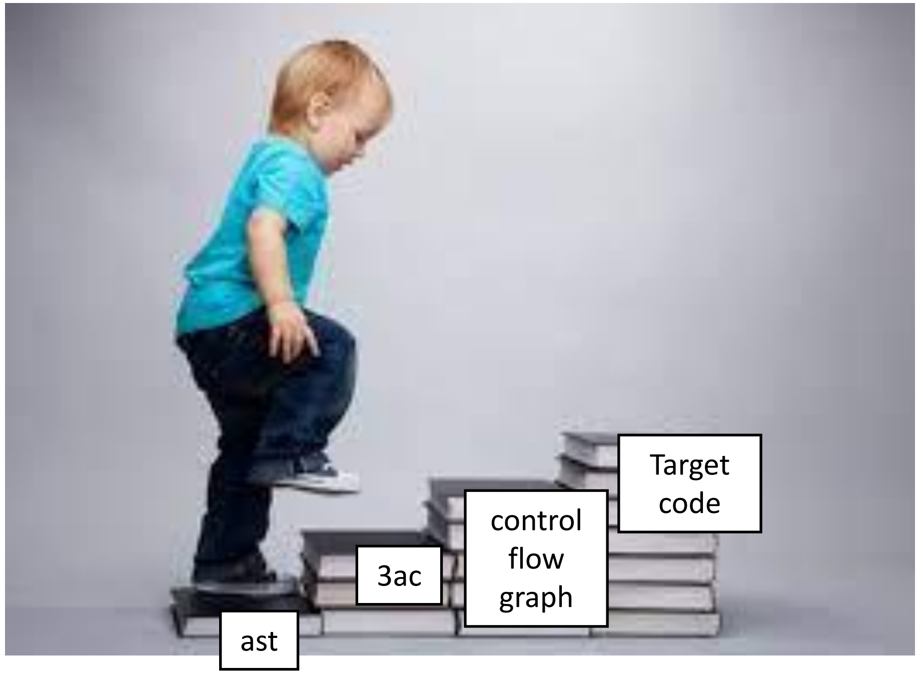
We target machine code for two reasons (beyond the classic reasons)

1) Discharge the obligation of writing a virtual machine
2) Get to learn how X64 code works

Many Steps Towards Target Code

Rather than bridging the semantic gap in one step, transform the code in many baby steps

- Encourages modularity
- Accommodate analysis goals



Summary

Runtime Environments – Wrap-up

- Defined runtime environments
  - The implicit dependencies of a program
  - May not be real hardware
- The compilers job is to support program abstractions in the runtime
  - For hardware platforms, these abstractions need to be simulated from memory, registers, and instruction sets
  - For software platforms, the abstractions of the software may be designed to support the language

Next Time

Runtime Environments – Wrap-up

- Talk about intermediate representations more generally
- Begin discussing our next intermediate representation, three-address code