



A PROGRAMMING LANGUAGE FOR
COMPILERS COURSES

ROHAN PADHYE, KOUSHIK SEN, PAUL N. HILFINGER
(UC BERKELEY)

1.

THE SET-UP

CS164 at UC Berkeley



COURSE CONTENT

Formal
Language
Design

Compiler
Project

Language
Implementation
Strategies



12 WEEKS

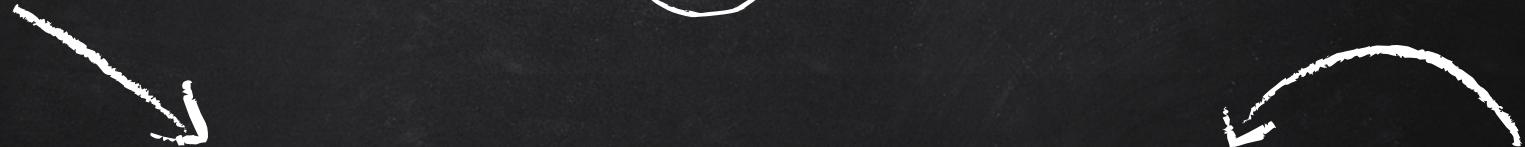


Build a full Compiler

Translates valid <source language> programs into <assembly language>

(working in teams)





CHOCOPY

A language that screams “Compile Me!”



WHY DID WE DEVELOP CHOCOPY?



Familiarity



Specification



Artifacts



Modern
Target

2.

THE LANGUAGE

What does a ChocoPy program look like?



CHOCOPY PROGRAMS

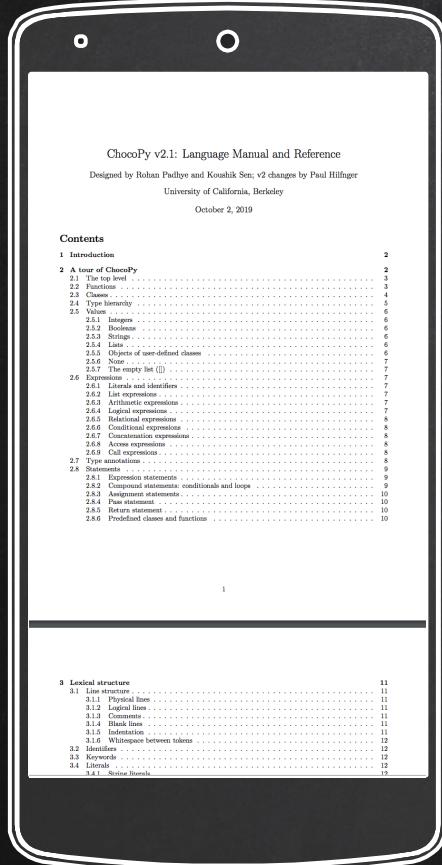
```
def contains(items:[int], x:int) -> bool:  
    i:int = 0  
  
    while i < len(items):  
        if items[i] == x:  
            return True  
        i = i + 1  
  
    return False  
  
if contains([4, 8, 15, 16, 23], 15):  
    print("Item found!")
```

```
class Animal(object):  
    makes_noise:bool = False  
  
    def make_noise(self: "Animal"):  
        if self.makes_noise:  
            print(self.sound())  
  
    def sound(self: "Animal") -> str:  
        return "???"  
  
class Cow(Animal):  
    def __init__(self: "Cow"):  
        self.makes_noise = True  
  
    def sound(self: "Cow") -> str:  
        return "moo"
```



CHOCOPY C PYTHON

Every valid ChocoPy program
can be executed in a Python interpreter
(to get same result)



LANGUAGE REFERENCE MANUAL

Comprehensive (36 page) specification of the ChocoPy language.

- Lexical structure of programs
- Formal grammar of syntax
- Typing rules
- Formal operational semantics



FORMAL TYPING RULES & OPERATIONAL SEMANTICS

VAR-INIT

$$\frac{O(id) = T \quad O, M, C, R \vdash e_1 : T_1 \quad T_1 \leq_a T}{O, M, C, R \vdash id:T = e_1}$$

ATTR-READ

$$\frac{O, M, C, R \vdash e_0 : T_0 \quad M(T_0, id) = T}{O, M, C, R \vdash e_0.id : T}$$

RETURN-E

$$\frac{O, M, C, R \vdash e : T \quad T \leq_a R}{O, M, C, R \vdash \text{return } e}$$

ATTR-INIT

$$\frac{M(C, id) = T \quad O, M, C, R \vdash e_1 : T_1 \quad T_1 \leq_a T}{O, M, C, R \vdash id:T = e_1}$$

LIST-SELECT

$$\frac{O, M, C, R \vdash e_1 : [T] \quad O, M, C, R \vdash e_2 : \text{int}}{O, M, C, R \vdash e_1[e_2] : T}$$

RETURN

$$\frac{\langle\text{None}\rangle \leq_a R}{O, M, C, R \vdash \text{return}}$$

VAR-READ

$$\frac{E(id) = l_{id} \quad S(l_{id}) = v}{G, E, S \vdash id : v, S, _}$$

LIST-SELECT

$$\frac{G, E, S_0 \vdash e_1 : v_1, S_1, _ \quad G, E, S_1 \vdash e_2 : \text{int}(i), S_2, _ \quad v_1 = [l_1, l_2, \dots, l_n] \quad 0 \leq i < n \quad v_2 = S_2(l_{i+1})}{G, E, S_0 \vdash e_1[e_2] : v_2, S_2, _}$$

VAR-ASSIGN-STM

$$\frac{G, E, S \vdash e : v, S_1, _ \quad E(id) = l_{id} \quad S_2 = S_1[v/l_{id}]}{G, E, S \vdash id = e : _, S_2, _}$$

RETURN-E

$$\frac{G, E, S \vdash e : v, S_1, _}{G, E, S \vdash \text{return } e : _, S_1, v}$$



CHOCOPY: LANGUAGE FEATURES



- Static typing with nominal subtyping
- Primitive types, objects, lists, None
- Top-level and nested functions
- Global, local, nonlocal variables
- Classes, attributes, methods



- Native dictionaries
- List comprehension
- Exceptions
- Default arguments
- Lambdas, closures

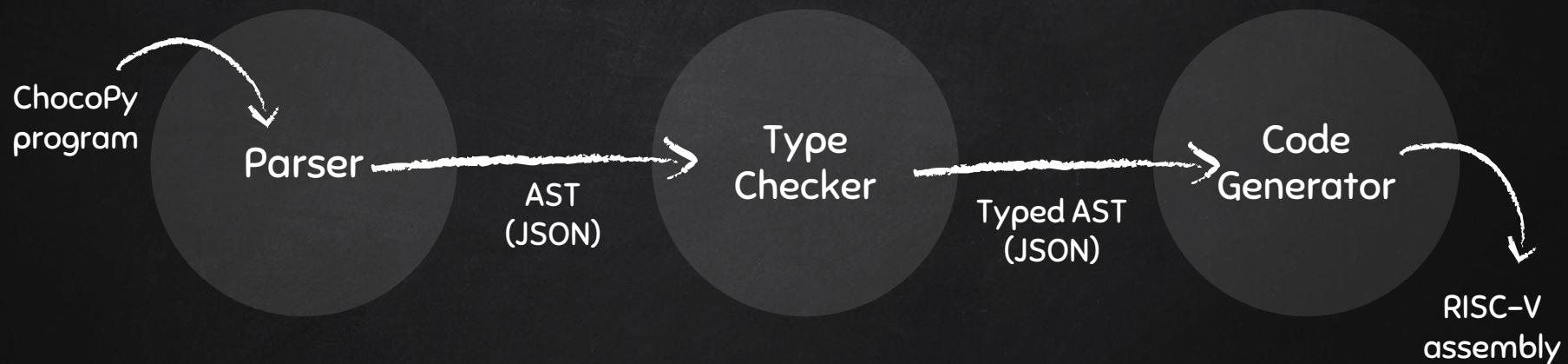
4.

THE PROJECT

What do students work with?

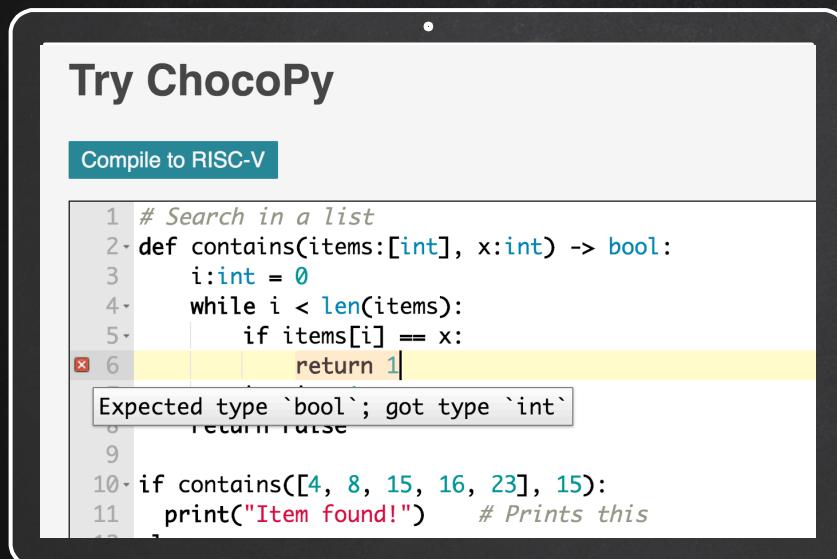


A COMPILER IN 3 PARTS (=ASSIGNMENTS)





WEB IDE



The image shows a web-based IDE interface titled "Try ChocoPy". It features a code editor with the following Python code:

```
1 # Search in a list
2 def contains(items:[int], x:int) -> bool:
3     i:int = 0
4     while i < len(items):
5         if items[i] == x:
6             return 1
7
8 Expected type `bool`; got type `int`
9
10 if contains([4, 8, 15, 16, 23], 15):
11     print("Item found!") # Prints this
12
```

A yellow highlight covers the line of code "return 1". A tooltip below it displays the error message: "Expected type `bool`; got type `int`".

Below the code editor is a teal button labeled "Compile to RISC-V".

- POWERED BY STUDENT OR REFERENCE COMPILER
- SELF-DOCUMENTING ASSEMBLY
- STEP-THROUGH DEBUGGING IN BROWSER



WEB IDE

```
275 li a0, 0          # Load integer literal 0
276 sw a0, -12(fp)    # local variable i
277 j label_6         # Jump to loop test
278 label_5:
279 lw a0, 4(fp)      # Top of while loop
280 sw a0, -16(fp)    # Load var: contains.items
281 lw a0, -12(fp)    # Push on stack slot 4
282 lw a1, -16(fp)    # Load var: contains.i
283 bnez a1, label_8 # Pop stack slot 4
284 j error.None     # Ensure not None
285 label_8:
286 lw t0, 12(a1)     # Go to error handler
287 bltu a0, t0, label_9 # Not None
288 j error.OOB       # Load attribute: __len__
289 label_9:
290 addi a0, a0, 4    # Ensure 0 <= index < len
291 li t0, 4          # Go to error handler
292 mul a0, a0, t0    # Index within bounds
293 add a0, a1, a0     # Compute list element offset
294 lw a0, 0(a0)      # Word size in bytes
295 add a0, a1, a0     # Compute list element offset
296 add a0, a1, a0     # Pointer to list element
297 lw a0, 0(a0)      # Get list element
```

- POWERED BY STUDENT OR REFERENCE COMPILER
- SELF-DOCUMENTING ASSEMBLY
- STEP-THROUGH DEBUGGING IN BROWSER



WEB IDE

The screenshot shows the Web IDE interface with the following components:

- Toolbar:** Run, Step, Prev, Reset, Dump.
- Editor:** Displays assembly code. A specific line, `lw x10 4(x8)`, is highlighted with a green background.
- Registers:** A table showing the current state of registers.

Registers	
zero	0x00000000
ra (x1)	0x000002dc
sp (x2)	0x7fffffc0
gp (x3)	0x10008024
tp (x4)	0x00000000
t0 (x5)	0x00000000
t1 (x6)	0xffffffff
t2 (x7)	0x10008024
s0 (x8)	0x7fffffe0
s1 (x9)	0x00000000
a0 (x10)	0x00000005
a1	0x00000009

- POWERED BY STUDENT OR REFERENCE COMPILER
- SELF-DOCUMENTING ASSEMBLY
- STEP-THROUGH DEBUGGING IN BROWSER



ASSIGNMENT RESOURCES



Language reference manual



Java-based starter code



Reference compiler



Web IDE



Auto-grader



RISC-V implementation guide

5.

EXPERIENCE

How do it go?



TAKEAWAYS FROM 2 ½ SEMESTERS



- Near-zero learning curve
- Language extensions intuitive
- Web-based IDE works well
- Student compilers beat CPython



- Lots of text to read
- Project is quite large
- Auto-grading error cases is tricky

Hacker News (Front Page #4)

The screenshot shows a mobile view of the Hacker News front page. The top navigation bar includes links for 'new', 'post', 'comments', 'ask', 'show', 'jobs', and 'submit'. Below the navigation, a post by 'mett_d' titled 'ChocoPy: A Programming Language for Compilers Courses' has received 189 points. The post discusses ChocoPy, a Python-like language designed for compiler creation, and includes a link to its GitHub repository. Several comments from users like 'userbrinster', 'xurte', 'tom_moller', and 'johnnepoed' follow, providing feedback and asking questions. At the bottom of the screen, there is a snippet of assembly code and a note from 'johnnepoed' explaining the meaning of certain labels.



TechRepublic.com

The screenshot shows a mobile view of a TechRepublic article titled 'How ChocoPy uses Python and RISC-V to teach compiler creation'. The article is written by James Sanders and published on September 13, 2019. It discusses the ChocoPy project, which is a restricted subset of Python 3 designed for compiler education. The article highlights how ChocoPy uses Python and RISC-V to simplify programming and help programmers understand compilers. It includes a bio for James Sanders and a snippet of ChocoPy code.



CHOCOPY.ORG

Running your own course?
instructors@chocopy.org

