Implement your own programming language with Rust in one hour

Wojciech Polak
Wrocław Rust Meetup #8

Wrocław, February 2019

(Implement your own programming language with Rust) in one hour

Wojciech Polak
Wrocław Rust Meetup #8

Wrocław, February 2019

Who am I?

Rust Developer for the last 4 months.

Previously .NET

Amateur PL enthusiast

Check your requirements

- How does your language look like? Design it.
- Do you want native (compiled), or interpreted?
- Static types or maybe dynamic?
- Embedded (like LUA)?
  Foreign Function Call (C libraries)?
- OOP/Functional/Procedural style?
- Multithreading?

Popular knowledge resources

- /r/ProgrammingLanguages and /r/Compilers
- Crafting Interpreters - online book
- “An Incremental Approach to Compiler Construction” - Ghuloum
- “Modern compiler implementation in C” - Appel
- “Types and Programming Languages” - Pierce

Popular tools

- **LLVM** - Most popular compiler backend - Rust uses it.
  Nice tutorial in C / Ocaml: https://llvm.org/docs/tutorial
  Rust bindings are still WIP.

- **Cranelift** - compiler backend **written in Rust.**
  Designed for WASM codes generation.
  https://github.com/CraneStation/simplejit-demo

- **C language** - as a compilation target
  JS language - as a compilation target

- **Pest** - parsing library for PEG (Parsing Expression Grammars) for Rust.
  https://github.com/pest-parser/pest

Language for this talk

- LISP(ish) syntax - to avoid operator precedence
- Functional
- Dynamic types (no typechecker)
  Integers, booleans, functions, lists
- With stack based byte code interpreter and compiler
- Primitive error handling

“Onion” approach

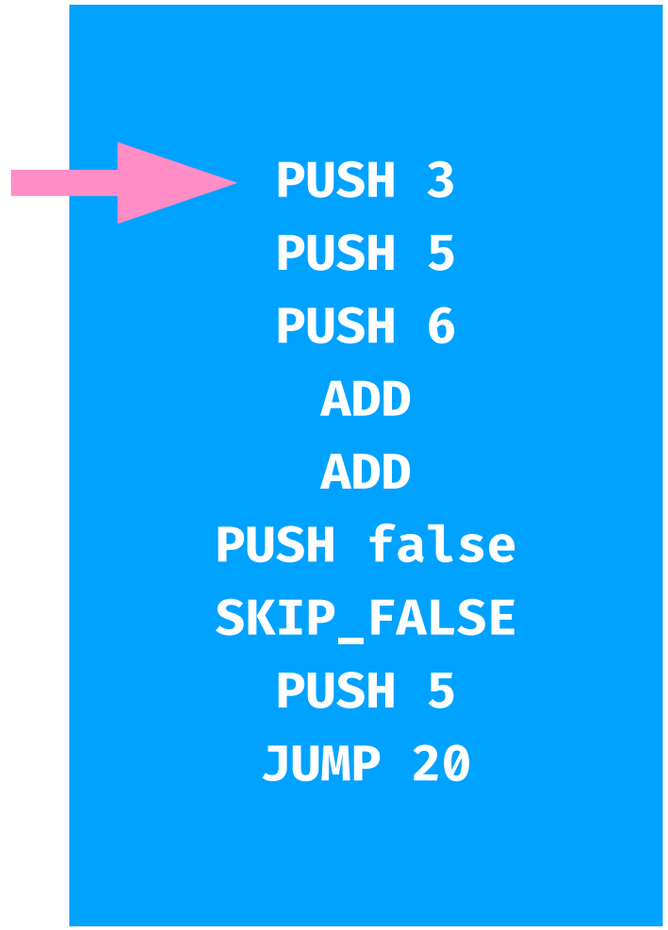
- We start with very simple subset of language
- Write some integration/unit tests.
- “Make it compile and run”
- Add new, more advanced feature, repeat process
- Each cycle takes from 10 minutes to maximum 1 week
- Nice, strong feedback loop

1. Integers and basics of VM.

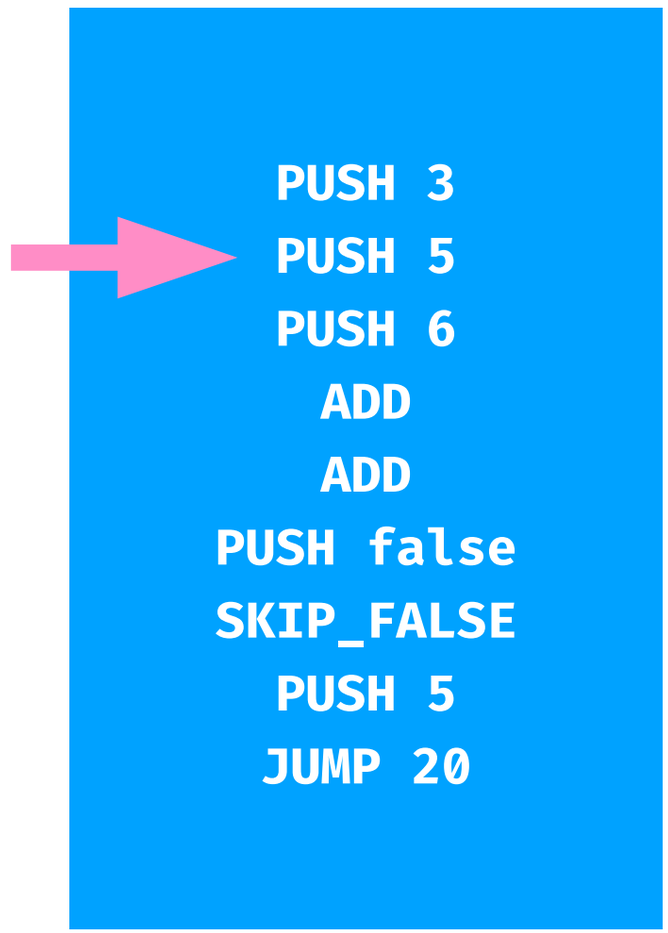
```
(+ 3 5 6)
(if false 5 6)
(* 124 (+ 3 34)
8)
```

```
PUSH 3
PUSH 5
PUSH 6
ADD
ADD
PUSH false
SKIP_FALSE
PUSH 5
JUMP 20
```

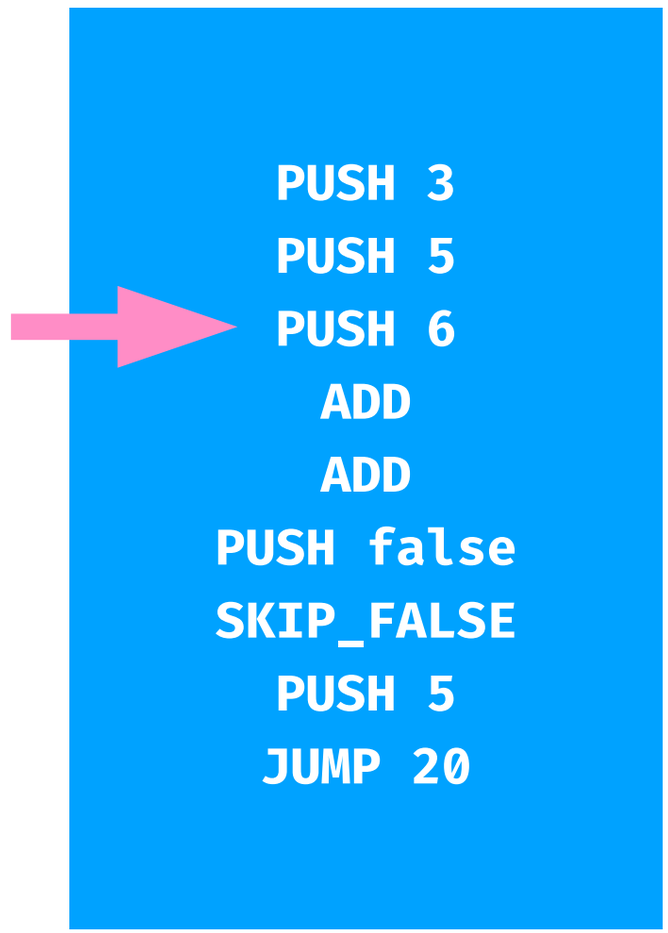
Virtual Machine



Virtual Machine



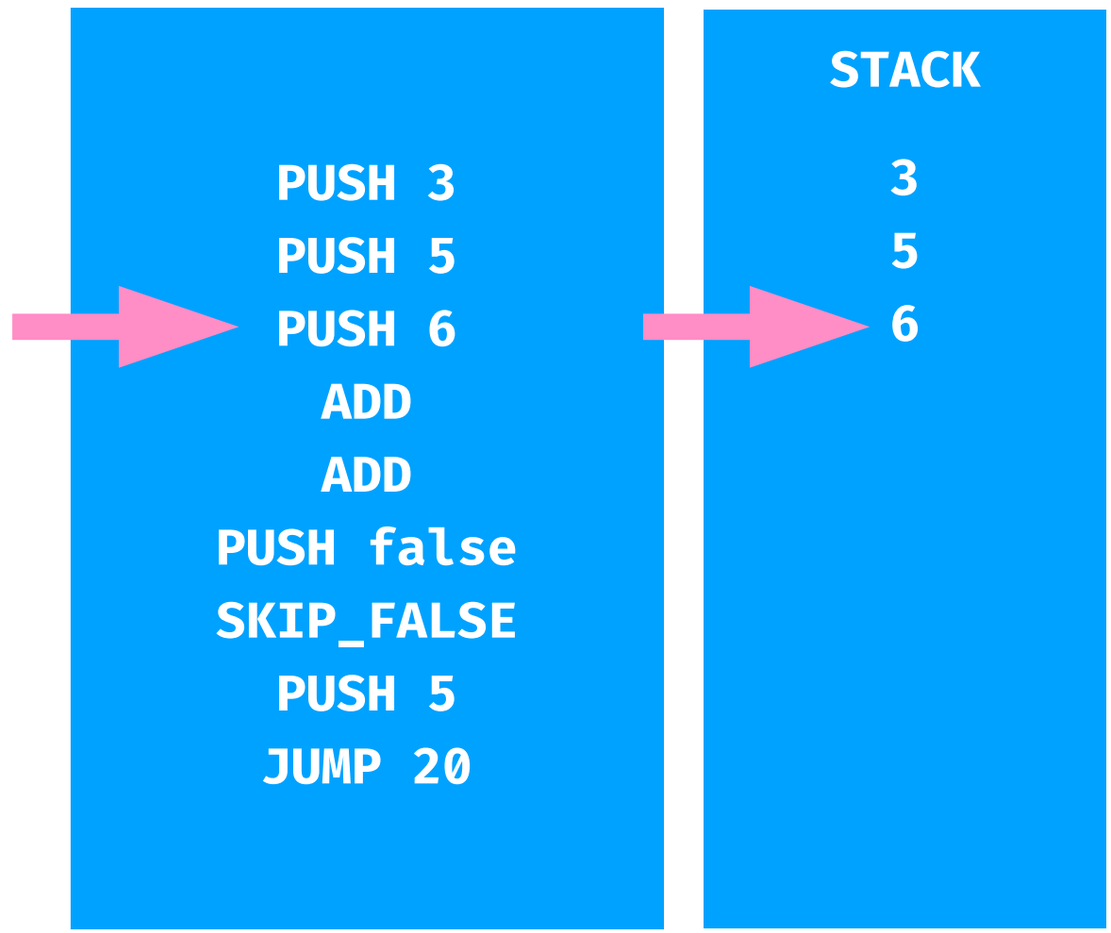
Virtual Machine



1. Implementation

- Dependencies:
  - Pest - parsing library
  - Test Case Derive - for `#[test_case()]` macro attribute https://crates.io/crates/test-case-derive

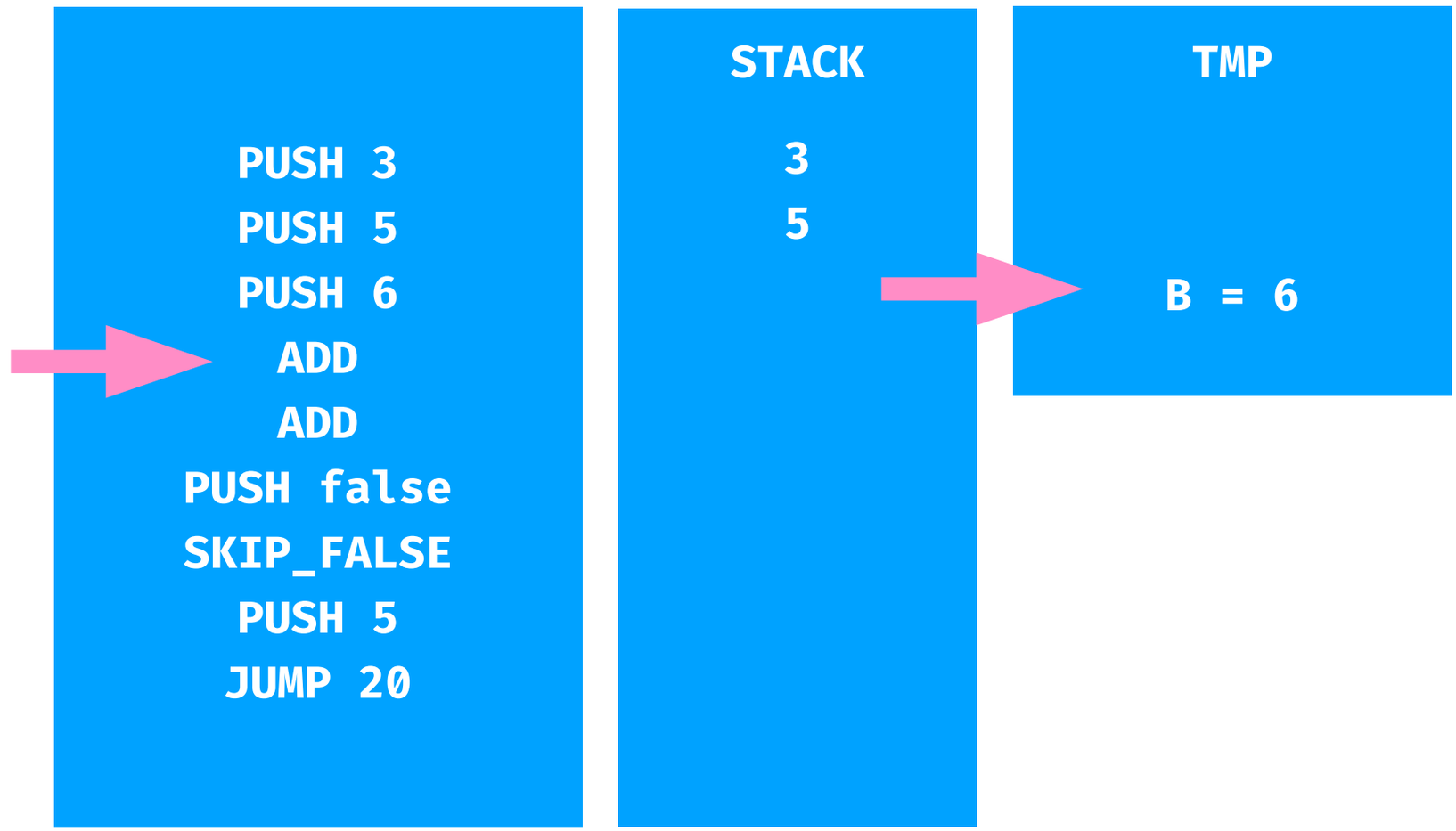
Virtual Machine



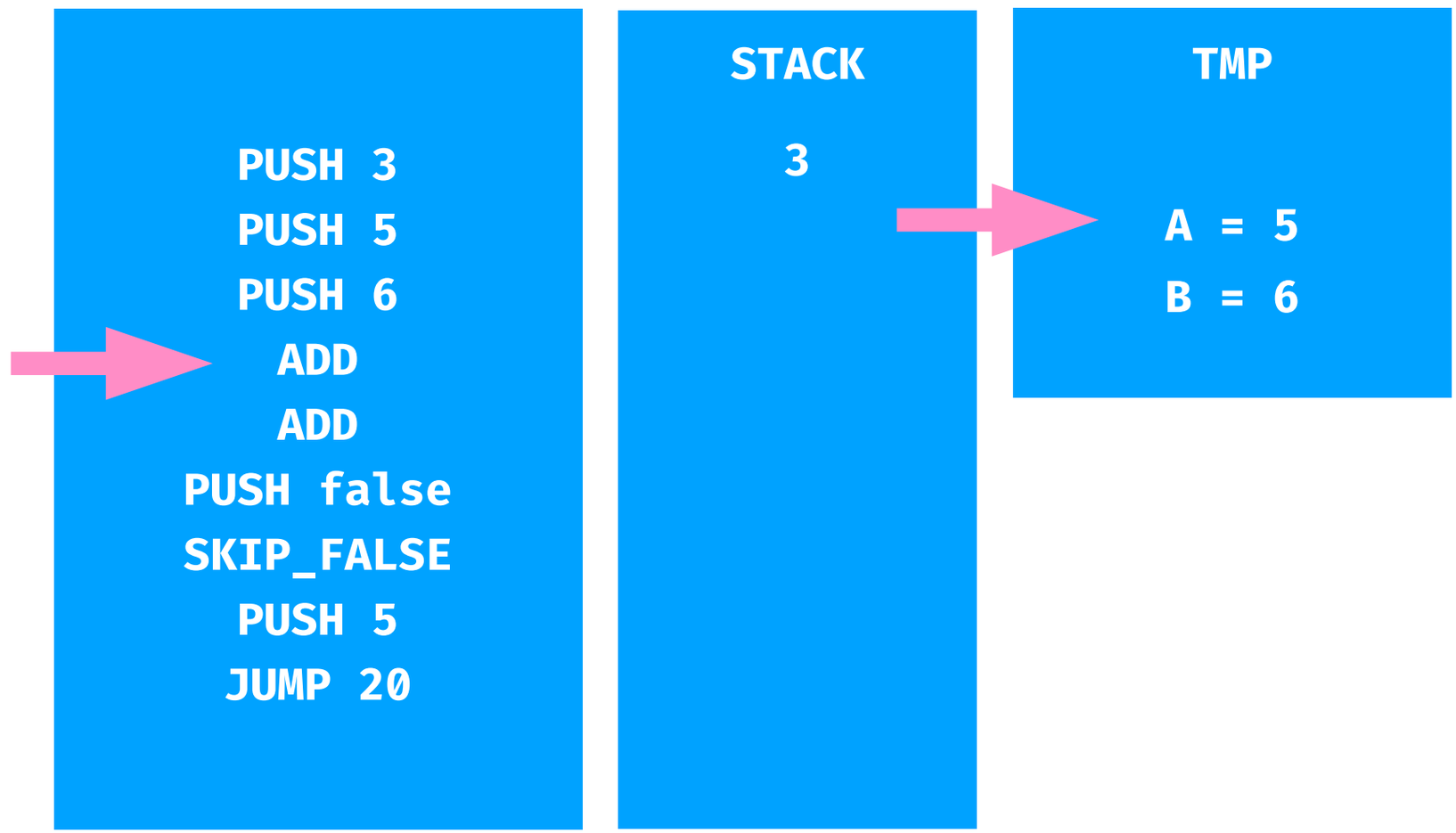
2. Binary operations

Let’s add some things!

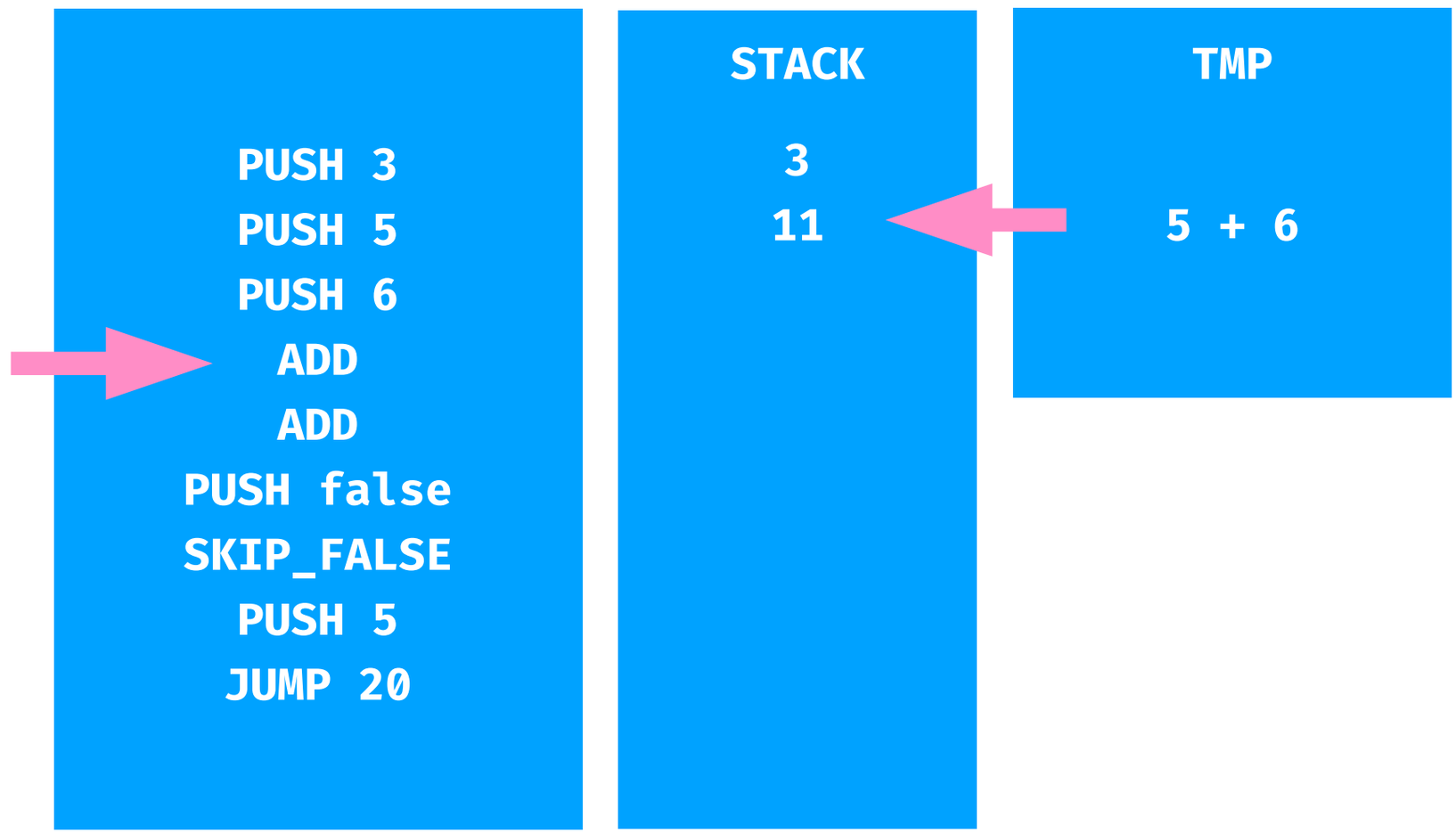
Virtual Machine



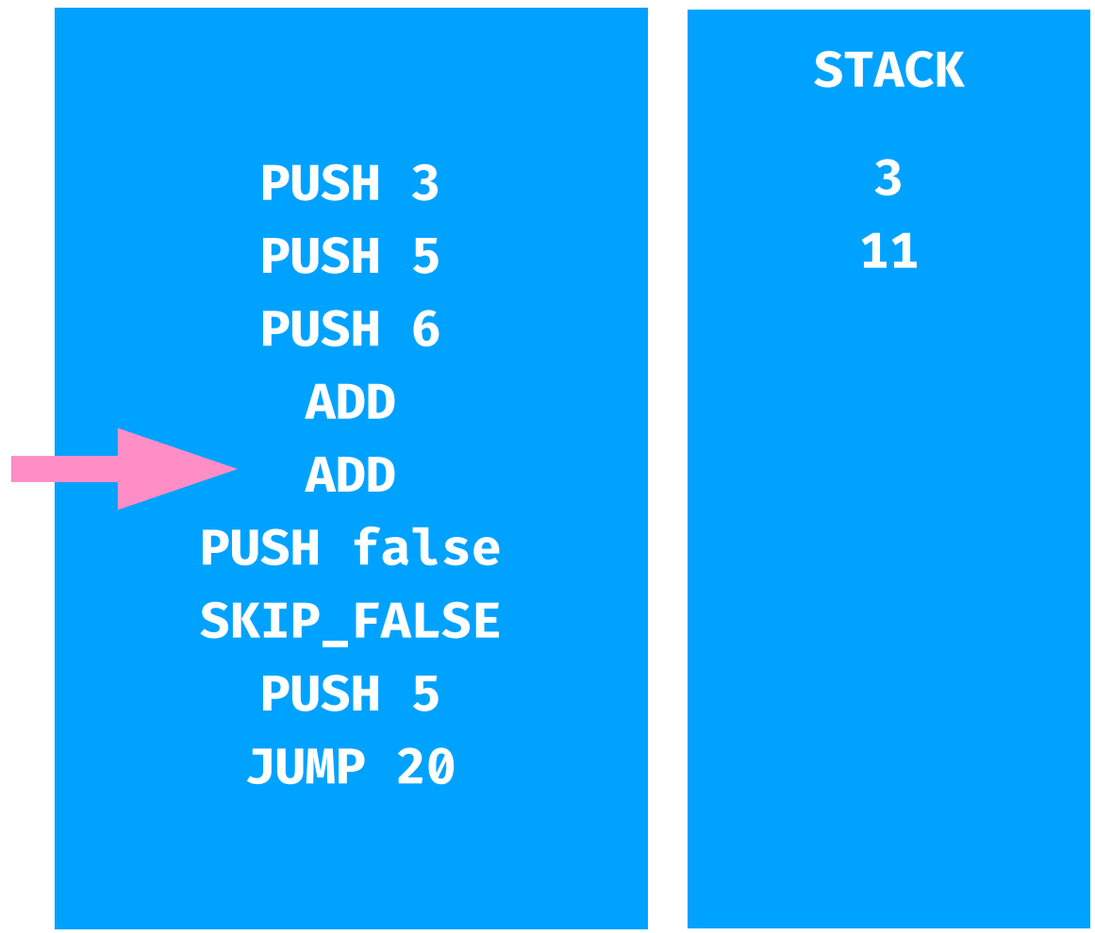
Virtual Machine



Virtual Machine

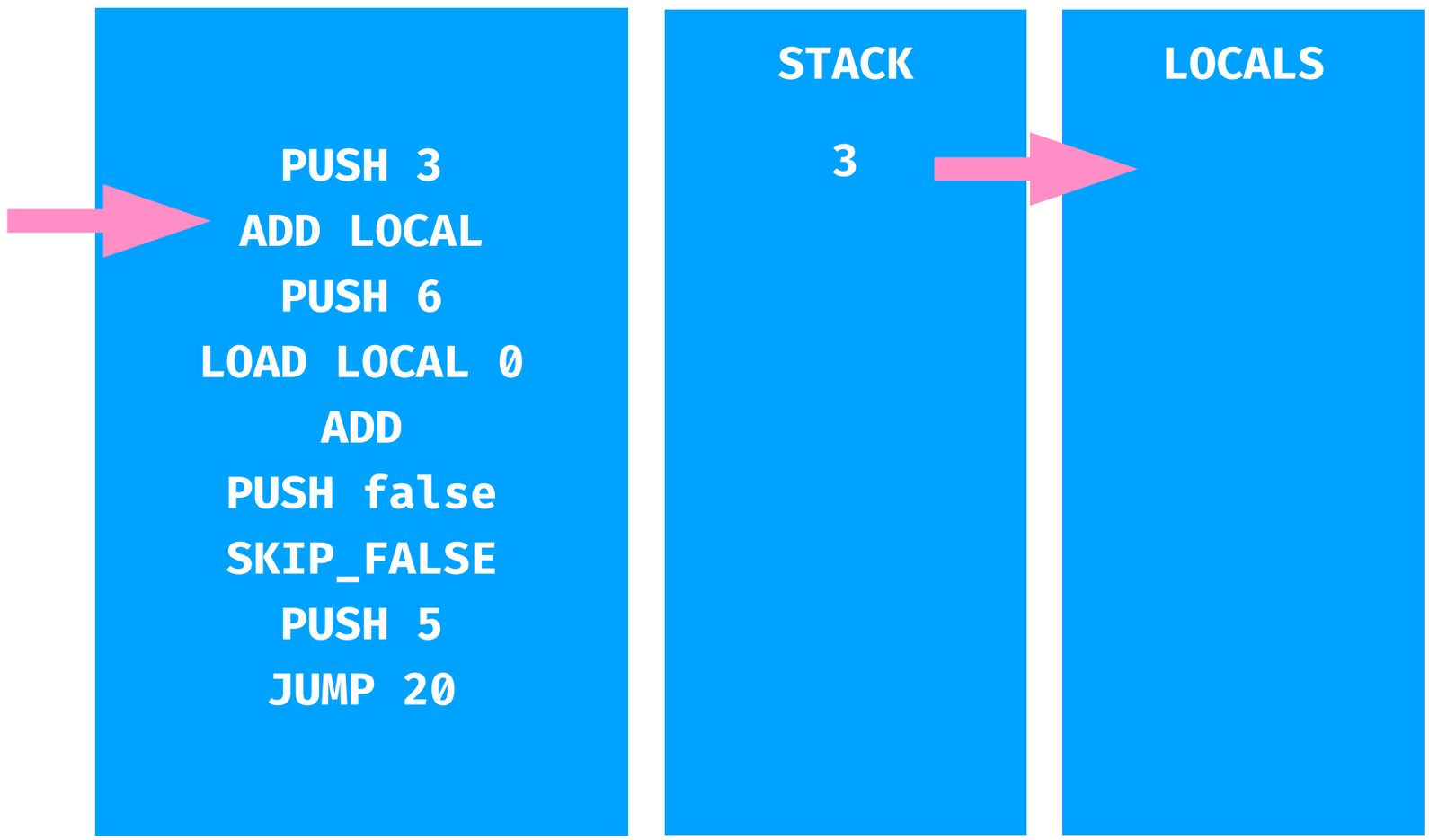


Virtual Machine

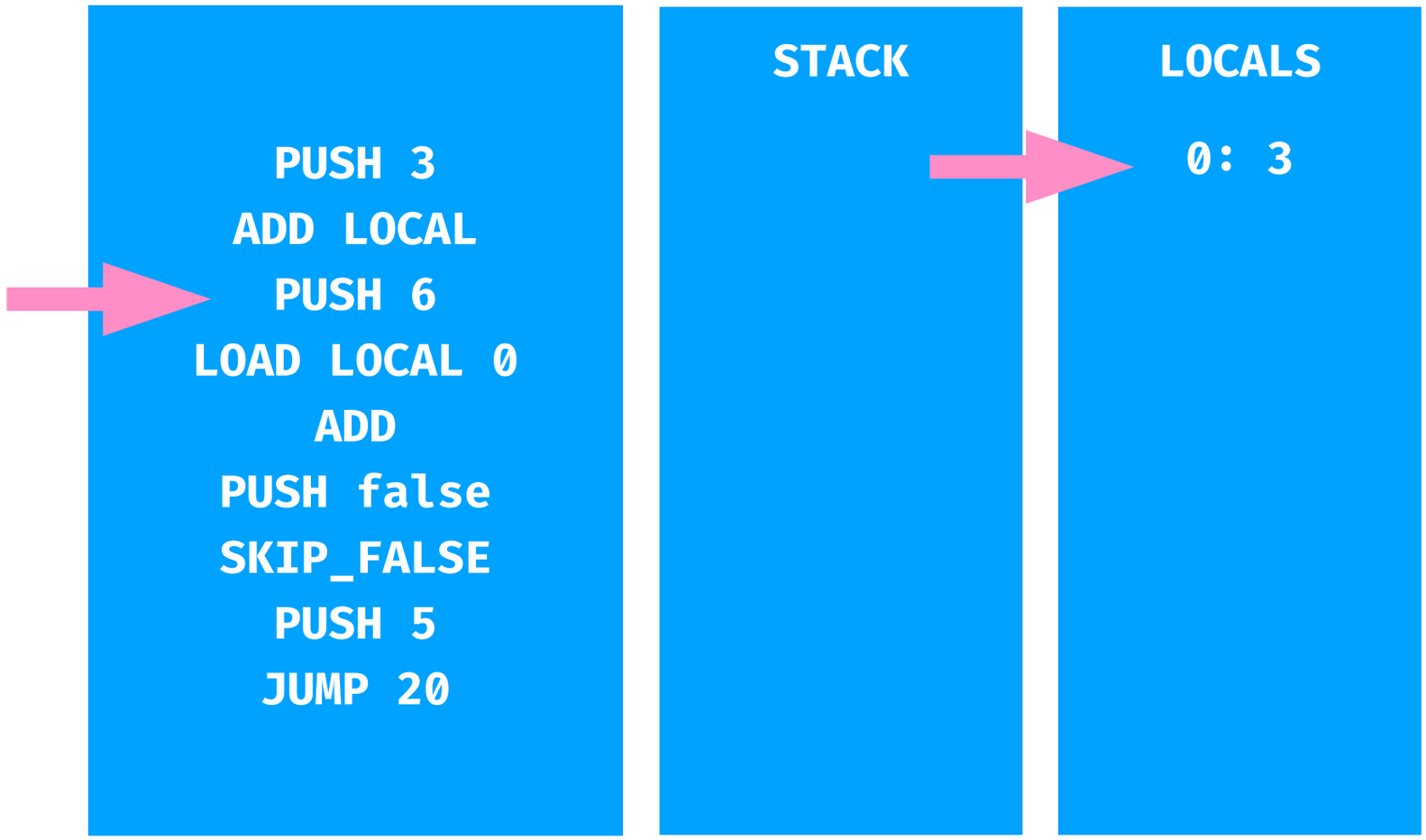


3. Local variables

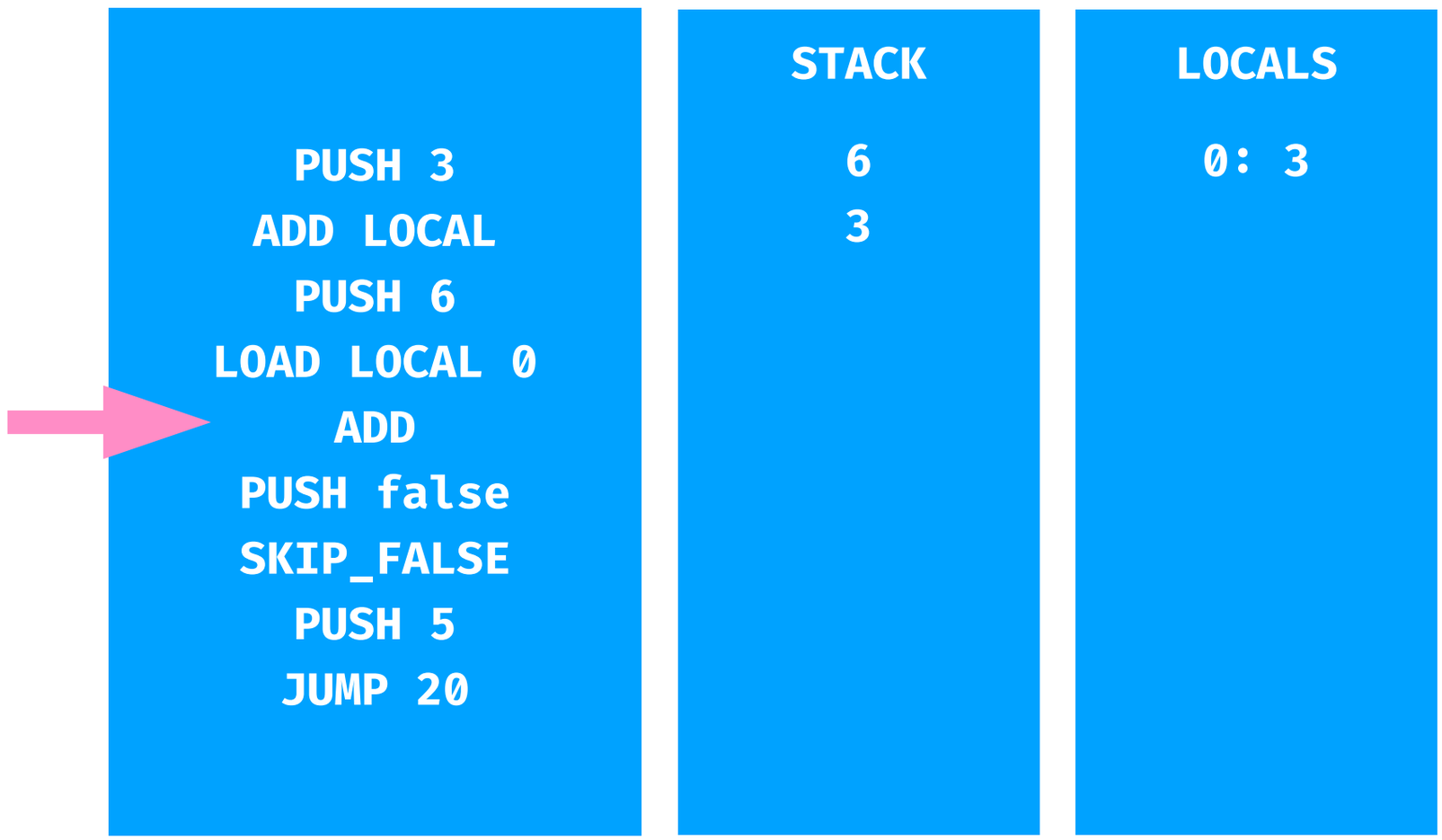
Virtual Machine



Virtual Machine



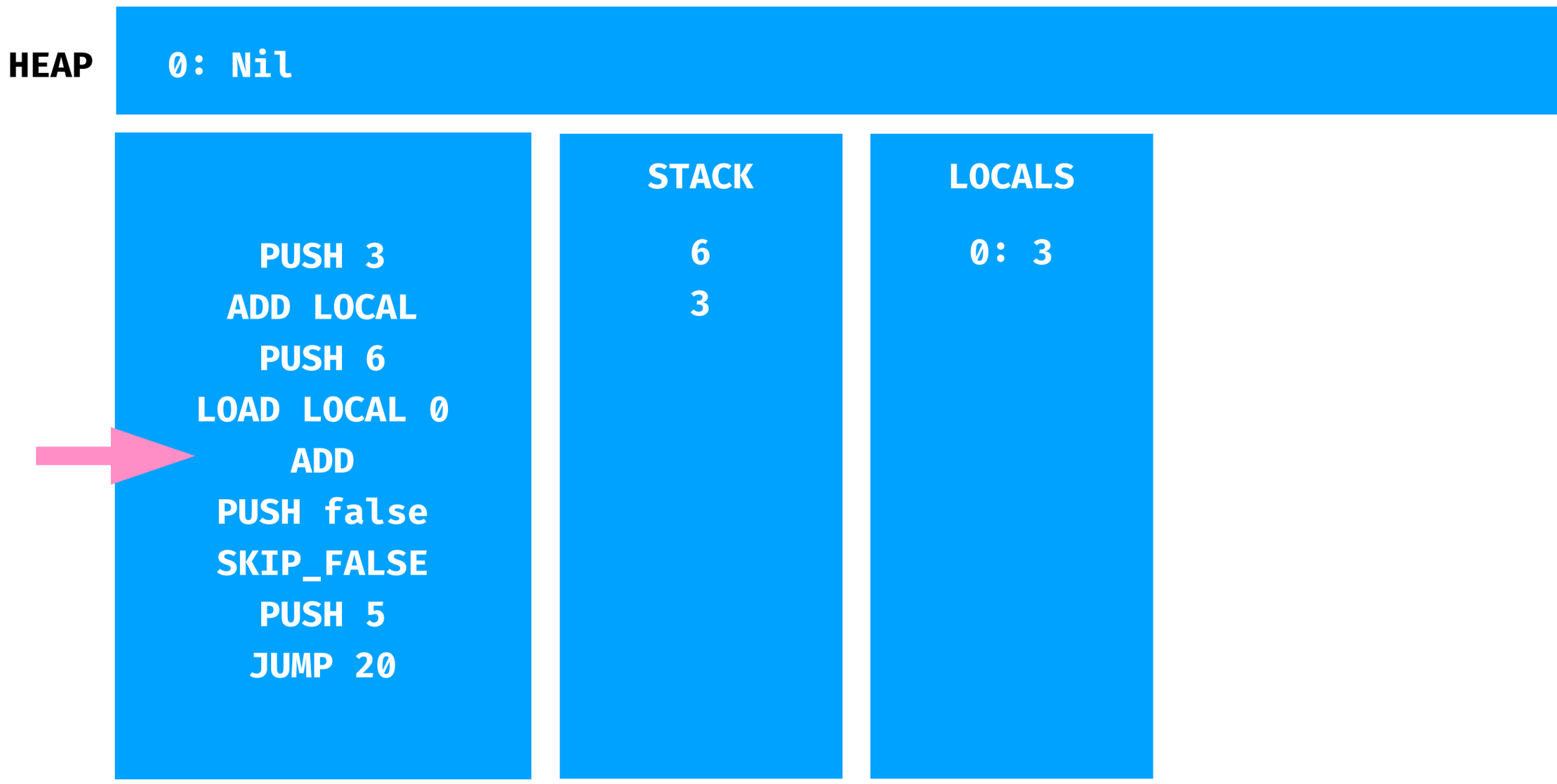
Virtual Machine



4. Conditional Expressions

5. Lists

Virtual Machine



Virtual Machine

HEAP

```
0: Nil   1: Cons(5, ptr(0))
```

```
PUSH 3
ADD LOCAL
PUSH 6
LOAD LOCAL 0
ADD
PUSH false
SKIP_FALSE
PUSH 5
JUMP 20
```

STACK

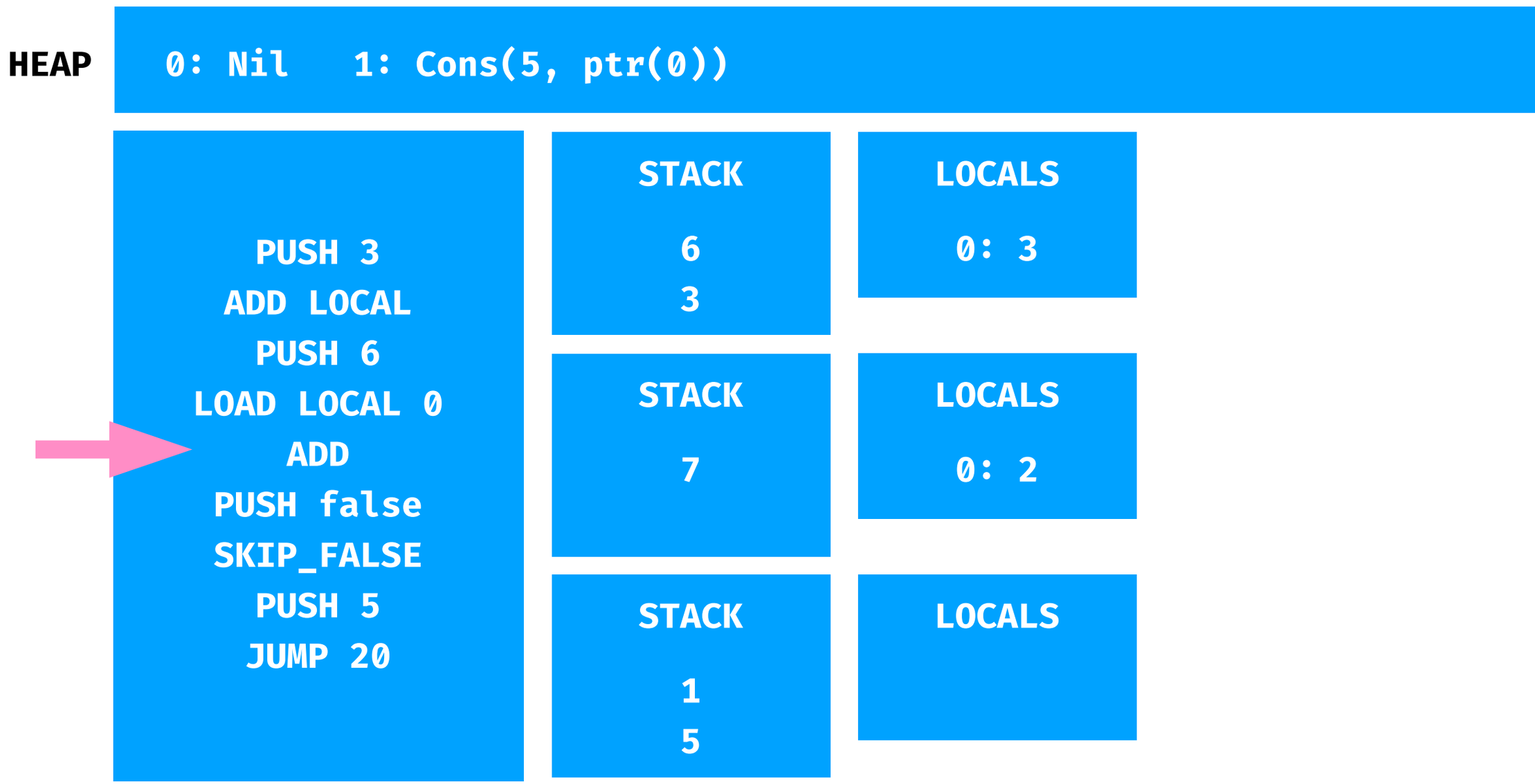
```
6
3
```

LOCALS

```
0: 3
```

6. Functions

Virtual Machine



What’s next?

Endless possibilities

Thank you!

Q&A

“How to implement type theory in an hour”

https://vimeo.com/286652934

“(How to implement type theory) in an hour”

https://vimeo.com/286652934