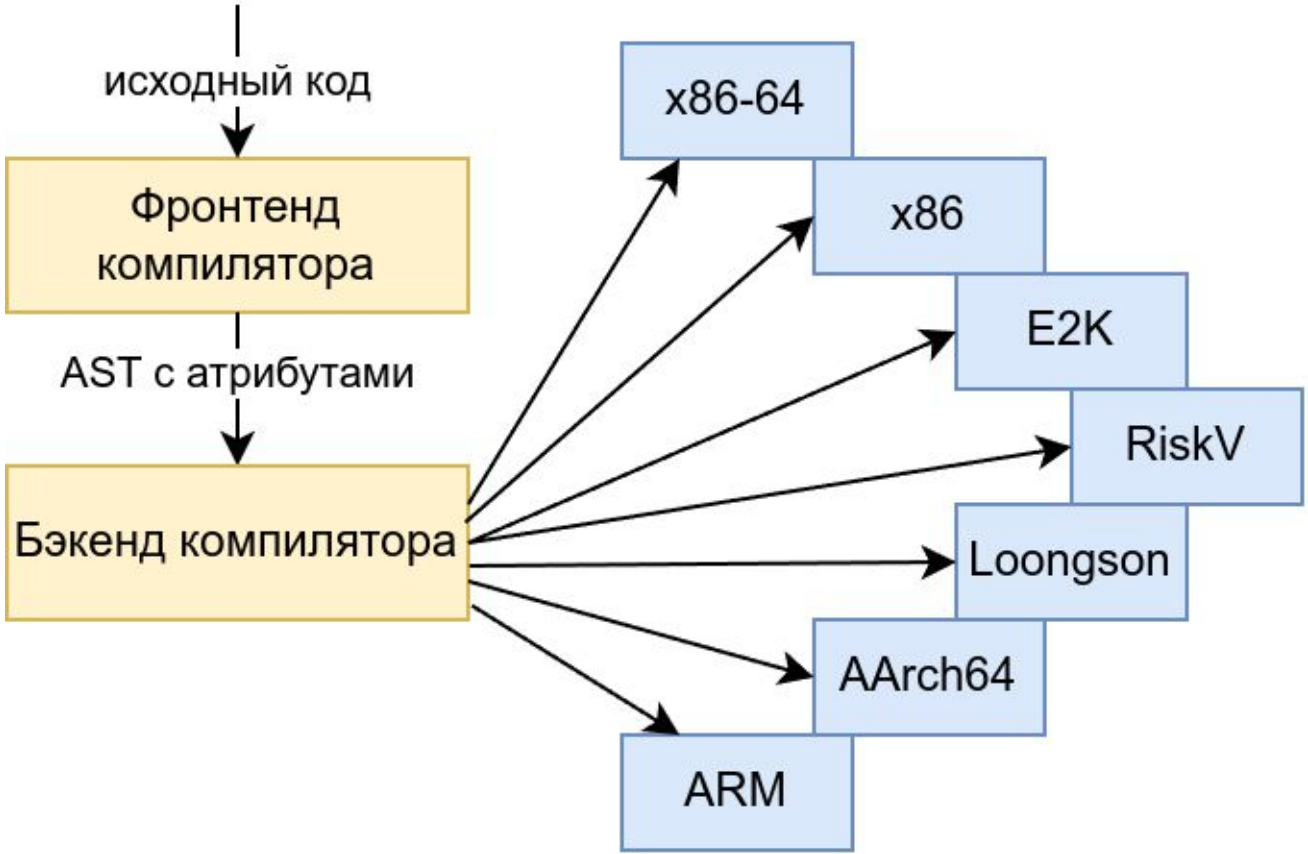
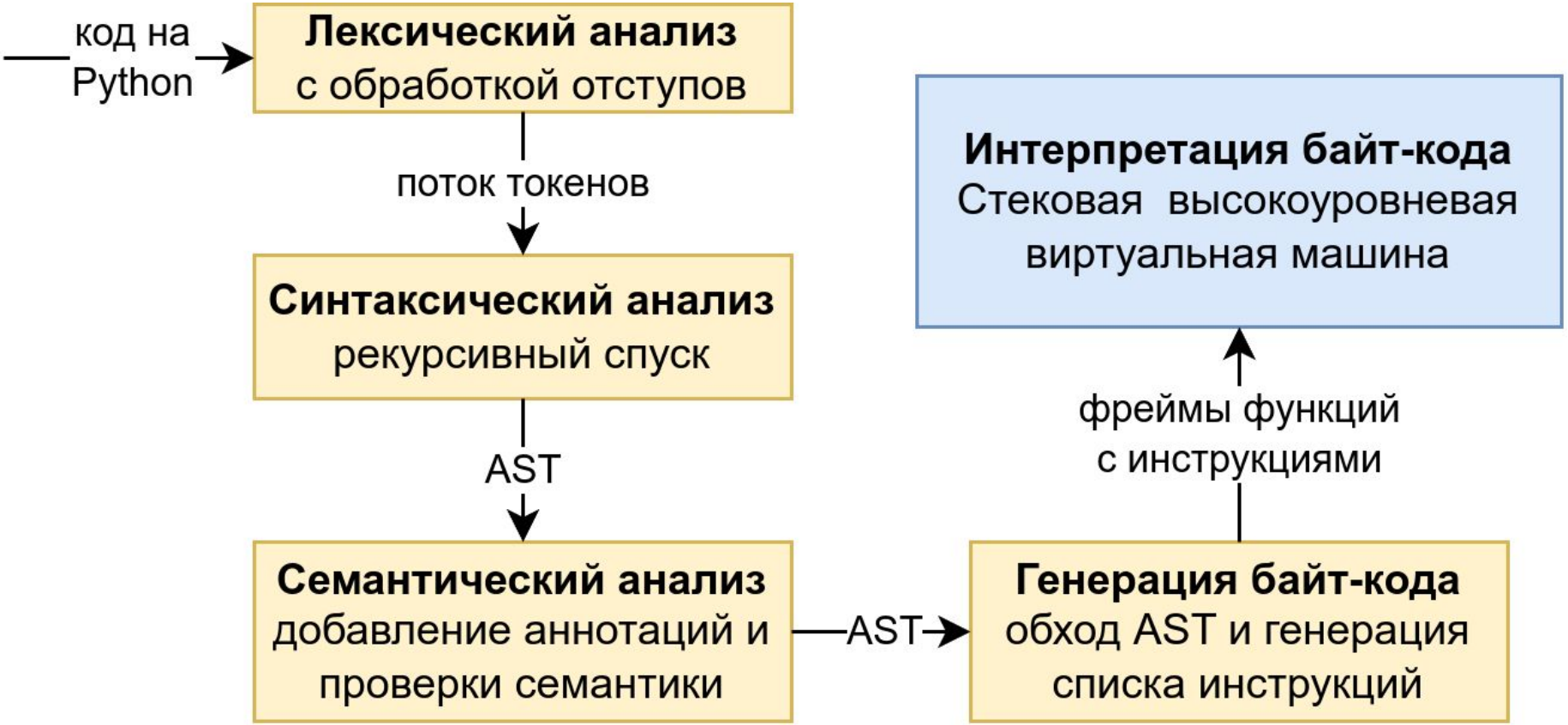
Виртуальные машины

Лекция №02
Теория языков программирования

Проблема: сколько нужно бэкендов компилятору?



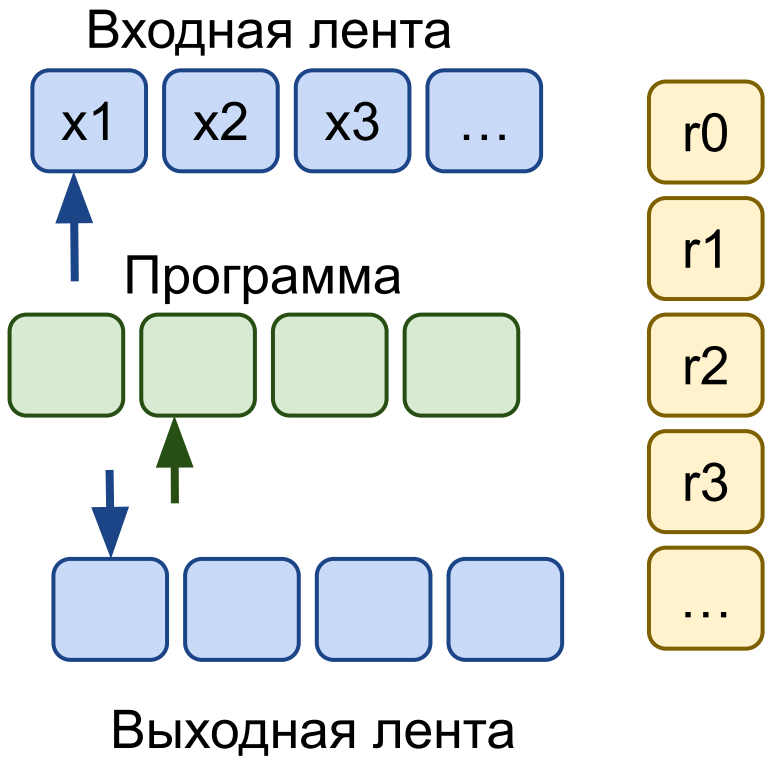
Интерпретатор CPython



Проектируем ВМ

Элементы RAM-машины (Random Access Machine)

1. Внутренняя память — конечный набор регистров с целыми числами
2. Входная лента целых чисел
3. Выходная лента целых чисел
4. Программа — список инструкций
5. Указатель на текущую инструкцию
6. Набор инструкций

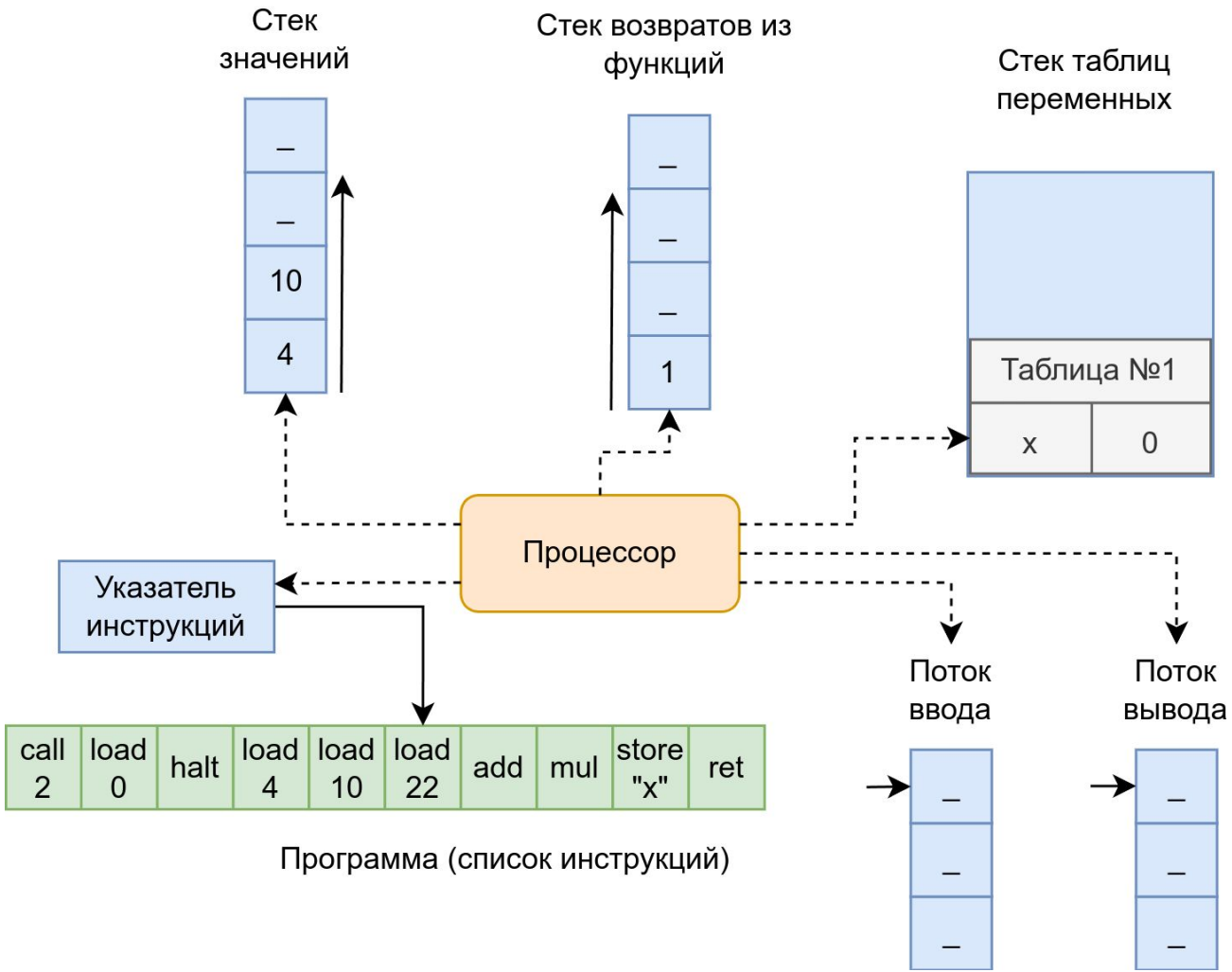


Процесс проектирования

Цель — спроектировать ВМ для своего языка

1. Рисуем схему машины
2. Составляем набор инструкций
    - Учитываем все возможности языка
    - Оцениваем способ реализации

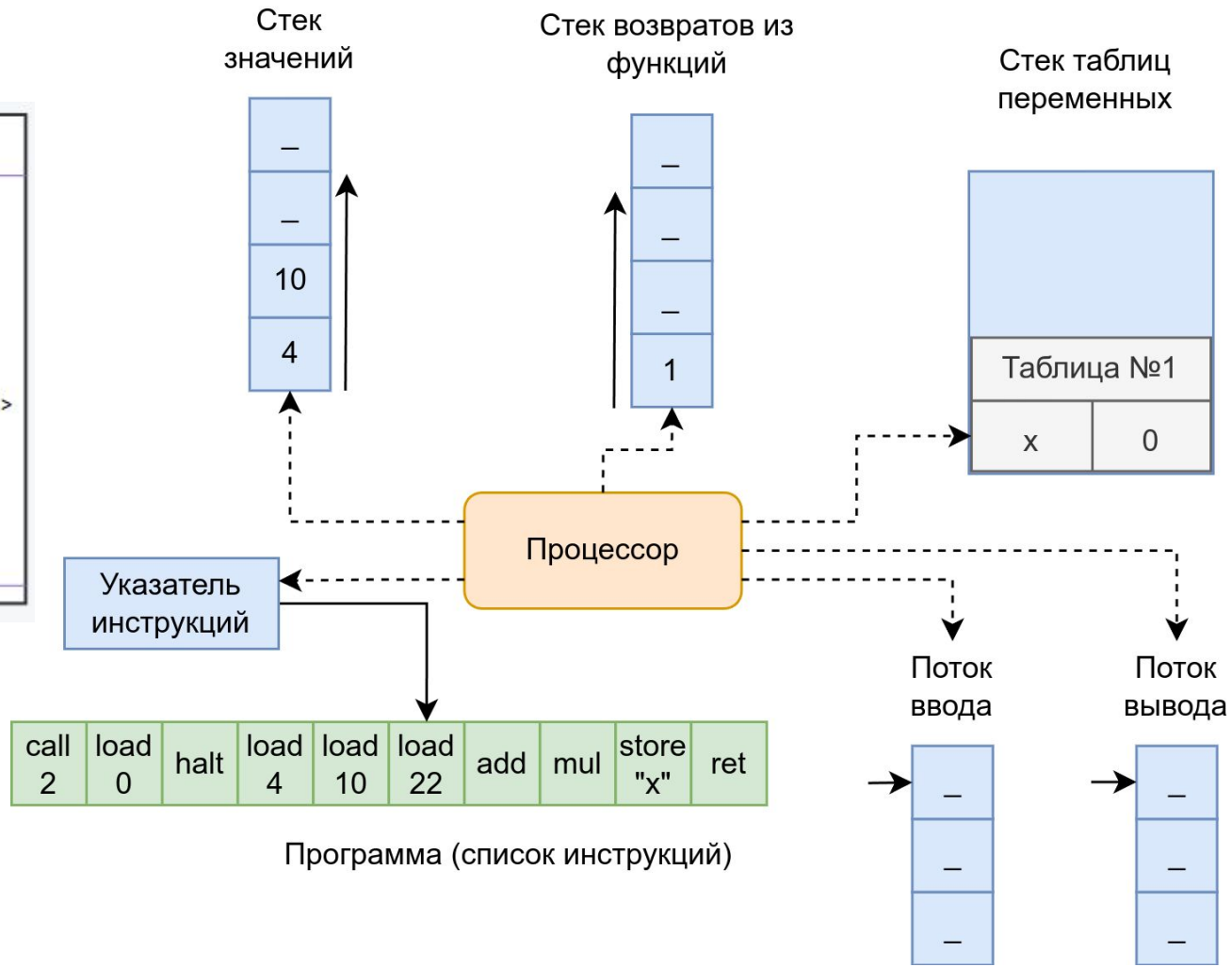
TigerVM


Стек значений
Стек возвратов из функций
Стек таблиц переменных
10
4
1
Таблица №1
x
0
Процессор
Указатель инструкций
call 2
load 0
halt
load 4
load 10
load 22
add
mul
store "x"
ret
Программа (список инструкций)
Поток ввода
Поток вывода

TigerVM

TigerVm

- _evaluationStack: Stack<Value>
- _returnStack: Stack<ReturnContext>
- _variables: VariablesTable?
- _instructionPointer: int
- _instructions: IReadOnlyList<Instruction>
- _builtinFunctions: BuiltinFunctions
- _exitCode: int
- _result: Value



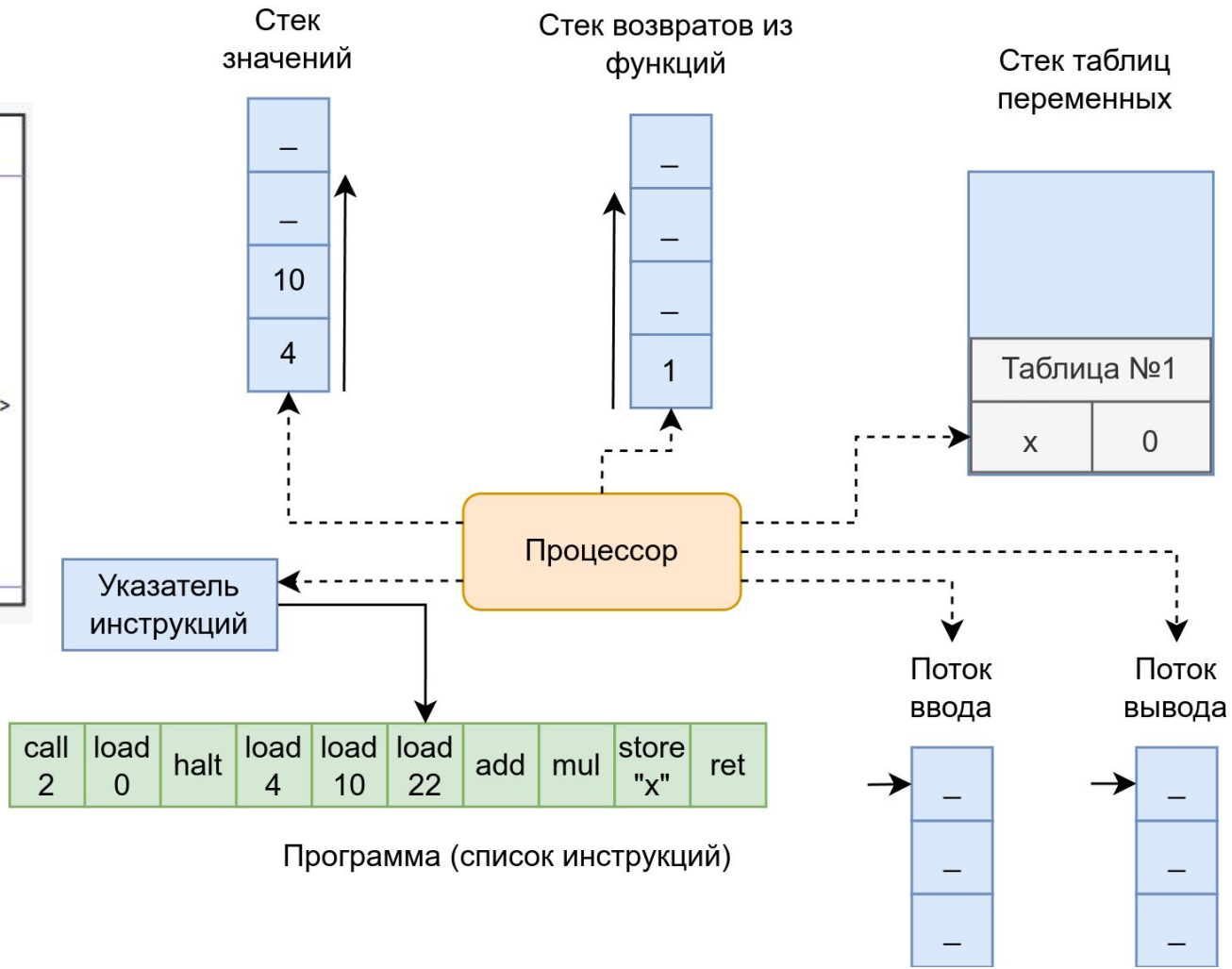
TigerVM

TigerVm

- _evaluationStack: Stack<Value>
- _returnStack: Stack<ReturnContext>
- _variables: VariablesTable?
- _instructionPointer: int
- _instructions: IReadOnlyList<Instruction>
- _builtinFunctions: BuiltinFunctions
- _exitCode: int
- _result: Value

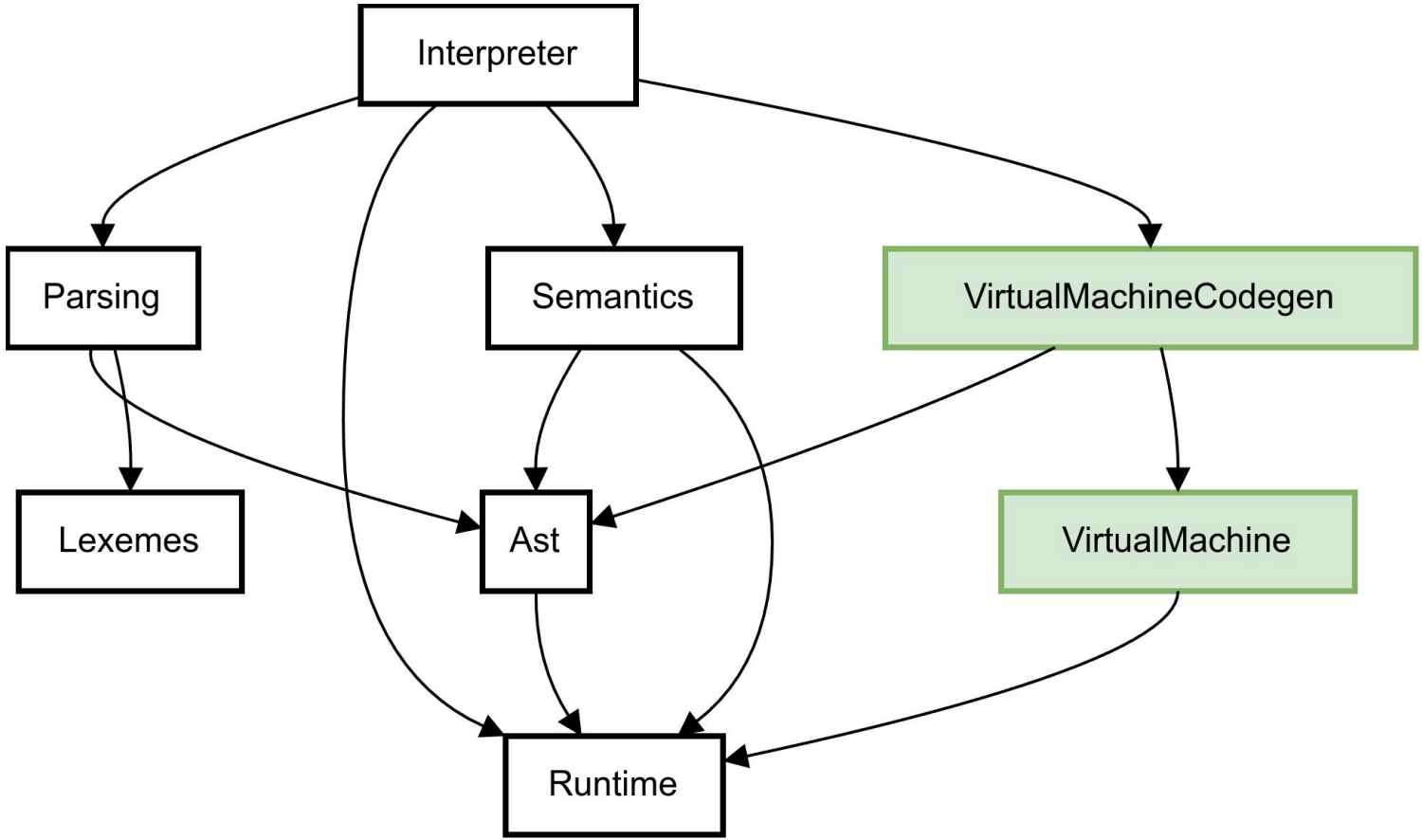
VariablesTable

- _parent: VariablesTable
- _variables: Dictionary<string, Value>
- _depth: int

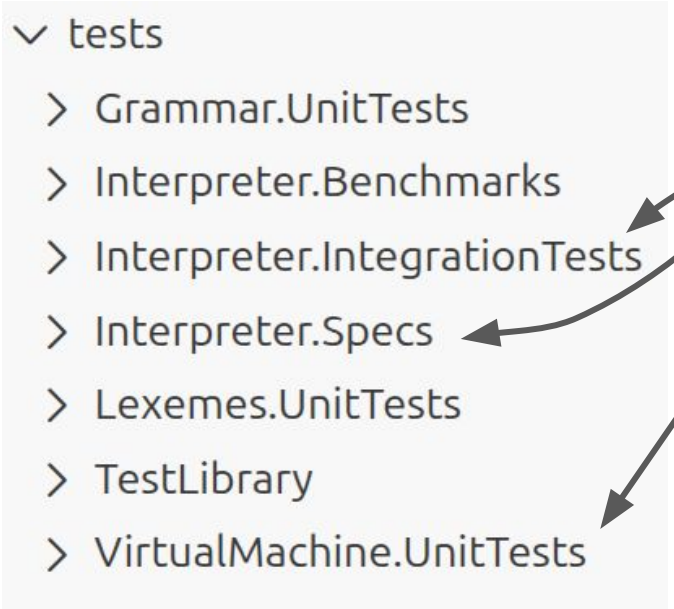


Архитектура проекта

Новые модули в проекте



Стратегия тестирования



1. Фокус на приёмочные тесты
2. Модульные тесты на обработку инструкций VM

Тесты для ВМ

Модульный тест для VM

```
[Theory]
[MemberData(nameof(GetEvaluateExpressionData))]
public void Can_evaluate_expression(
    List<Instruction> instructions, Value expected
)
{
    FakeEnvironment environment = new();
    TigerVm vm = new(environment, instructions);
    Value result = vm.RunProgram();

    Assert.Equal(0, vm.ExitCode);
    Assert.Equal(expected, result);
    Assert.Empty(environment.BufferedOutput);
    Assert.Empty(environment.FlushedOutput);
}
```

Модульный тест для VM

```
// Возврат одного значения со стека
{
    [
        new Instruction(InstructionCode.Push, 67),
        new Instruction(InstructionCode.StoreResult),
        new Instruction(InstructionCode.Push, 0),
        new Instruction(InstructionCode.Halt),
    ],
    new Value(67)
},
```

Реализуем ВМ

Инструкции

```
public enum InstructionCode
{
    // Добавляет значение в стек
вычислений.
    Push,

    // Забирает значение из стека
вычислений, записывает его в новую
переменную с указанным именем.
    DefineVar,

    // ...
}
```

```
public class Instruction
{
    public Instruction(
      InstructionCode code, Value value
    )
    {
        Code = code;
        Operand = value;
    }

    public InstructionCode Code { get; set; }

    public Value Operand { get; set; }
}
```

Главный цикл виртуальной машины

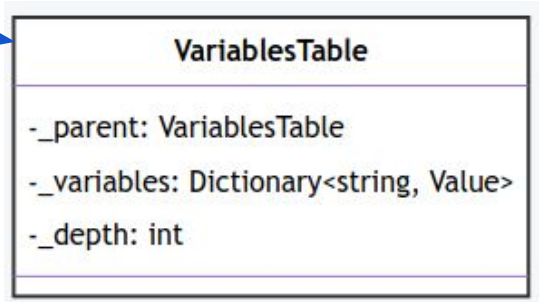
```
public Value RunProgram() {
  while (true) {
    Instruction instruction = _instructions[_instructionPointer++];
    switch (instruction.Code) {
      case InstructionCode.Push:
        _evaluationStack.Push(instruction.Operand);
        break;

      case InstructionCode.Halt:
        // Получаем код возврата программы.
        _exitCode = _evaluationStack.Pop().AsInt();
        return _result;

      default:
        throw new NotImplementedException(
          $"Unsupported instruction code: {instruction.Code}"
        );
    }
```

Ключевые идеи в реализации

1. Вычисления на стеке `Stack<Value>`
2. Словари переменных `VariablesTable` — стек хэш-массивов
3. Условные и безусловные переходы
4. Вызовы функций с помощью стека возвратов



Вычисления на стеке

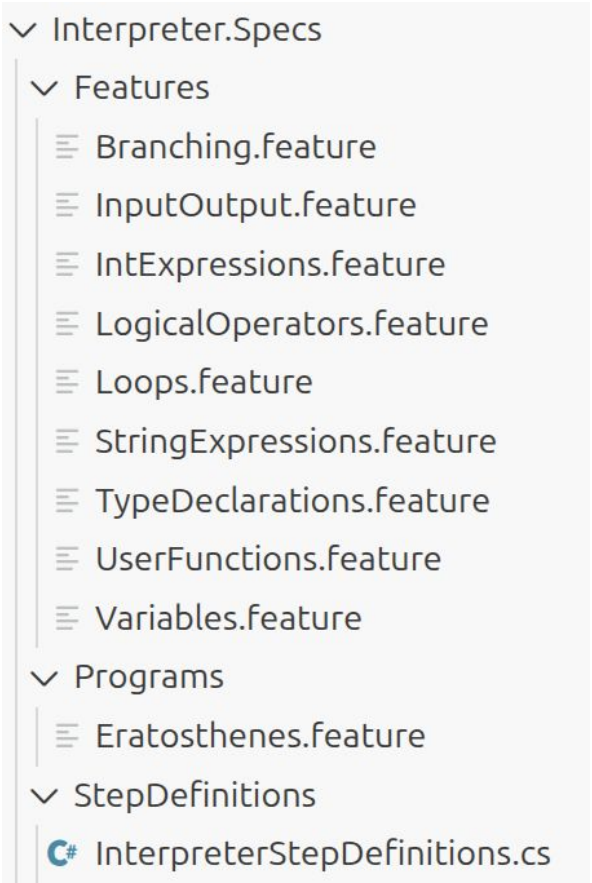
```
[  // (20 + 50) - 3 = 67
   new Instruction(InstructionCode.Push, 20),
   new Instruction(InstructionCode.Push, 50),
   new Instruction(InstructionCode.Add),
   new Instruction(InstructionCode.Push, 3),
   new Instruction(InstructionCode.Subtract),
   new Instruction(InstructionCode.StoreResult),
   new Instruction(InstructionCode.Push, 0),
   new Instruction(InstructionCode.Halt),
],
```

Вызов пользовательской функции

```
case InstructionCode.Call: {
        _returnStack.Push(new ReturnContext(_instructionPointer, _variables));
        _instructionPointer = instruction.Operand.AsInt();
    }
    break;
case InstructionCode.Return: {
        ReturnContext context = _returnStack.Pop();
        _instructionPointer = context.InstructionPointer;
        _variables = context.Variables;
    }
    break;
```

Приёмочные тесты Interpreter.Specs

Приёмочные тесты



1. Два слоя тестов
   - Features — возможности языка
   - Programs — полноценные программы

2. Тесты запускают весь интерпретатор

Пример приёмочного теста

```
#language: ru
Функциональность: ветвления

   Сценарий: ветвление if...then
       Пусть я загрузил программу "features/branching/if_then.tig"

       Когда я выполняю программу

       Тогда я увижу вывод:
       """
       2 * 2 = 4
       """
```

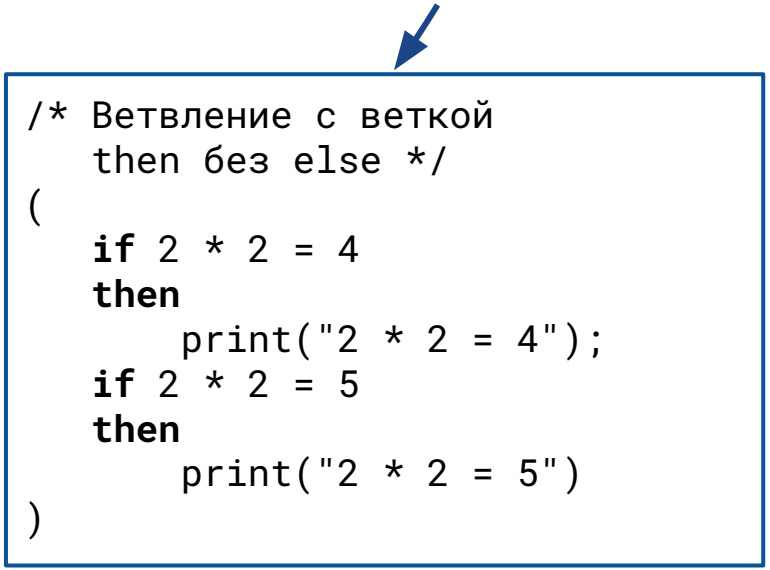
Пример приёмочного теста

```
#language: ru
Функциональность: ветвления

    Сценарий: ветвление if...then
        Пусть я загрузил программу "features/branching/if_then.tig"

        Когда я выполняю программу

        Тогда я увижу вывод:
        """
        2 * 2 = 4
        """
```



```
/* Ветвление с веткой
   then без else */
(
   if 2 * 2 = 4
   then
       print("2 * 2 = 4");
   if 2 * 2 = 5
   then
       print("2 * 2 = 5")
)
```

Как найти файл из теста в C#?

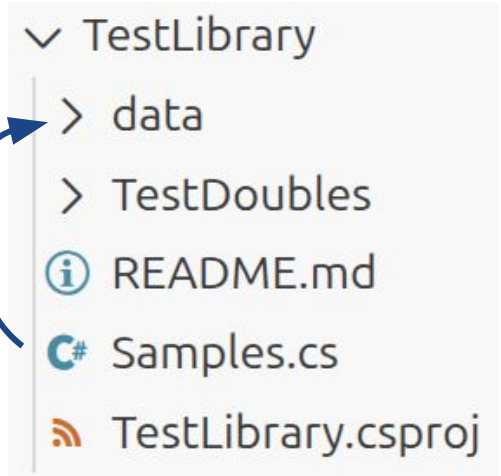
```
public static class Samples
{
    public static string GetSampleProgram(string filename)
    {
        // ...
    }

    private static string GetClassDirectory(
        [CallerFilePath] string path = ""
    )
    {
        return Path.GetDirectoryName(path) ?? throw new ArgumentException(
            $"Could not get directory path from {path}"
        );
    }
}
```

Как найти файл из теста в C#?

```
public static class Samples
{
    public static string GetSampleProgram(string filename)
    {
        // ...
    }

    private static string GetClassDirectory(
        [CallerFilePath] string path = ""
    )
    {
        return Path.GetDirectoryName(path) ?? throw new ArgumentException(
            $"Could not get directory path from {path}"
        );
    }
}
```



Конец