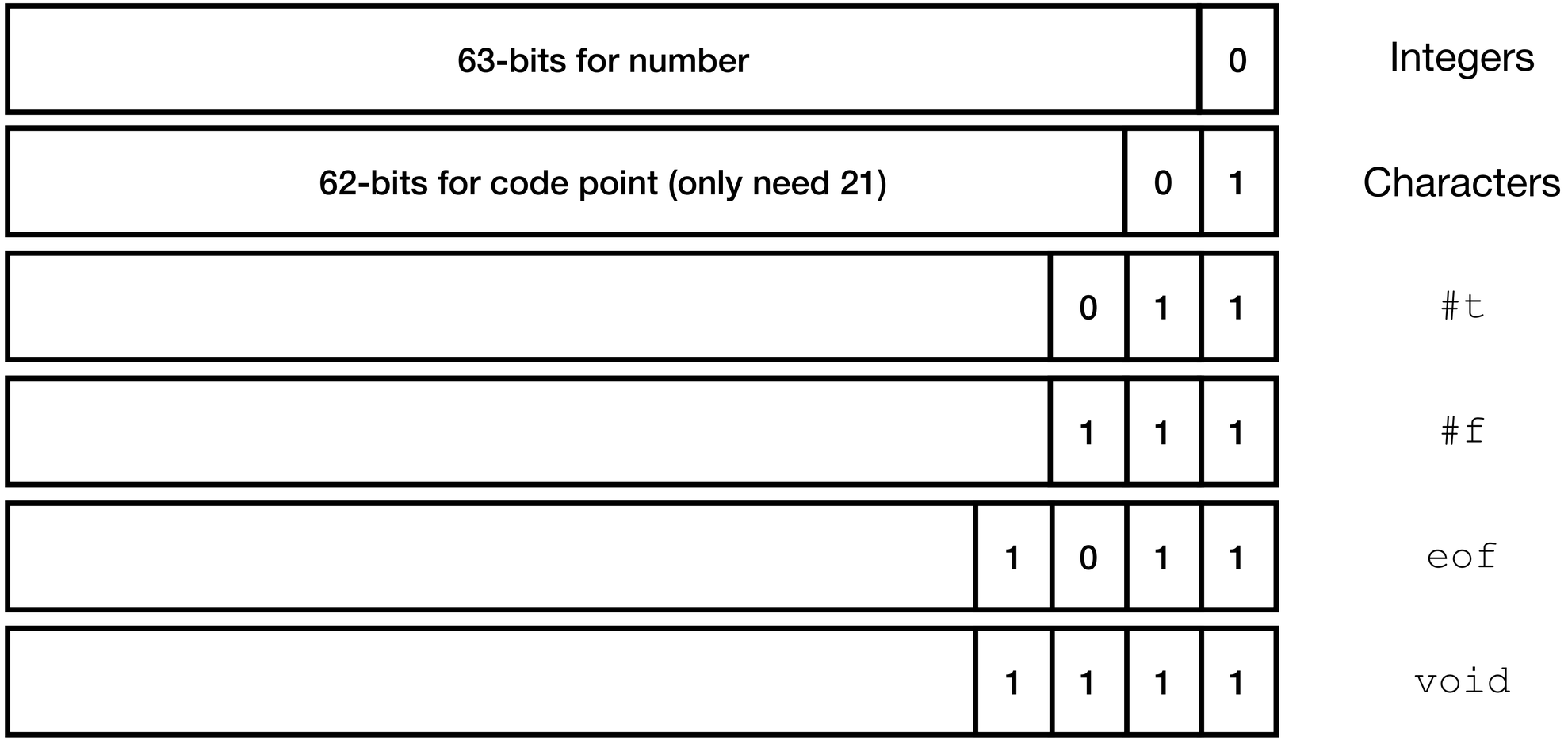
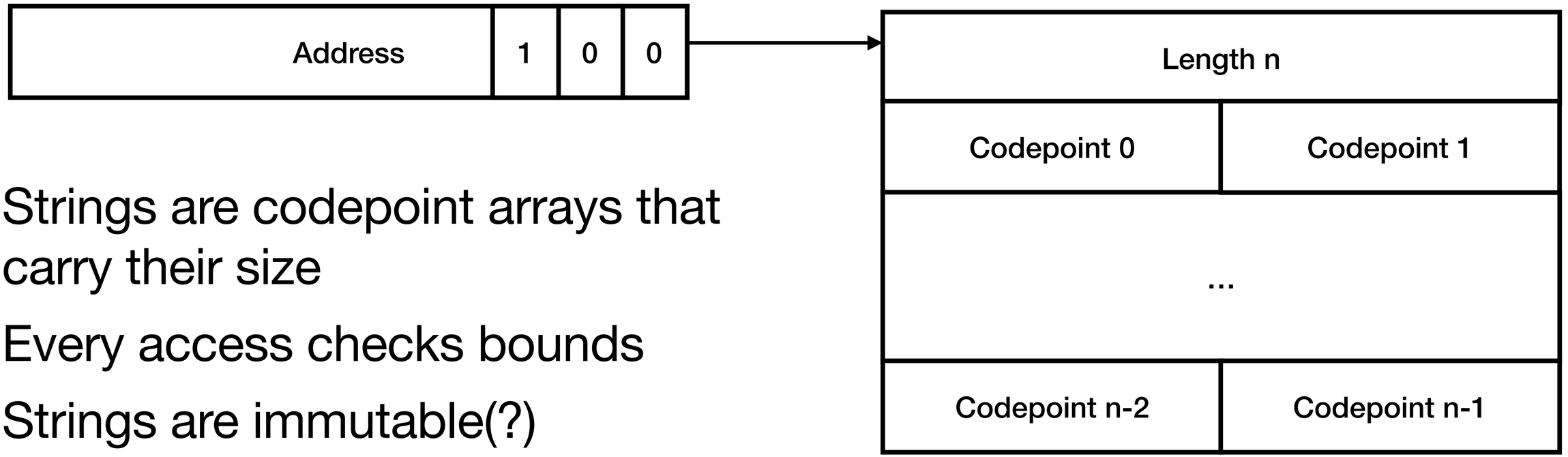
Encoding values so far in Evildoer

Type tag in least significant bits



How to represent strings?

Sized homogeneous arrays



Strings are codepoint arrays that carry their size

Every access checks bounds

Strings are immutable(?)

Designing our own calling convention

Function calls are like “`let` at a distance”

```
(f 3 4)
```

```
(define (f x y)
  (+ x y))
```

is like

```
(let ((x 3) (y 4))
  (+ x y))
```

Except the code for f is not part of the application expression

Designing our own calling convention

A first attempt (doesn't work)

```
(f 3 4)
```

```
(define (f x y)
  (+ x y))
```

Idea: arguments passed on the stack,

return point after arguments,

caller pushes and pops

(Push 3)

(Push 4)

(Call ‘f)

(Pop)

(Pop)

(Label ‘f)

(compile-e (parse ‘(+ x y)) ‘(y x))

(Ret)

Designing our own calling convention

Same thing without Call (still doesn't work)

```
(f 3 4)
```

(Push 3)

(Push 4)

(Lea ‘rax ‘r)

(Push ‘rax)

(Jmp ‘f)

(Label ‘r)

(Pop)

(Pop)

```
(define (f x y)
  (+ x y))
```

(Label ‘f)

(compile-e (parse ‘(+ x y)) ‘(y x))
(Ret)

Idea: arguments passed on the stack,

return point after arguments,

caller pushes and pops

Designing our own calling convention

Return point before arguments (still doesn't work)

```
(f 3 4)
```

(Lea ‘rax ‘r)

(Push ‘rax)

(Push 3)

(Push 4)

(Jmp ‘f)

(Label ‘r)

(Pop)

(Pop)

```
(define (f x y)
  (+ x y))
```

(Label ‘f)

(compile-e (parse ‘(+ x y)) ‘(y x))
(Ret)

Idea: arguments passed on the stack,

return point *before* arguments,

caller pushes and pops

Designing our own calling convention

Return point before arguments (works!)

```
(f 3 4)
```

(Lea ‘rax ‘r)

(Push ‘rax)

(Push 3)

(Push 4)

(Jmp ‘f)

(Label ‘r)

```
(define (f x y)
  (+ x y))
```

(Label ‘f)

(compile-e (parse ‘(+ x y)) ‘(y x))

(Pop)

(Pop)

(Ret)

Idea: arguments passed on the stack,

return point *before* arguments,

caller pushes, *callee pops*

CMSC 430 - 11 Dec 2023

Wrapping up!

Outlaw:

- Putting it all together
- Standard Library
- Added primitives
- Primitives as Functions
- More I/O
- Self compilation