An LLVM backend for OCaml

Colin Benner
benner.colin@gmail.com

University of Siegen

September 14, 2012
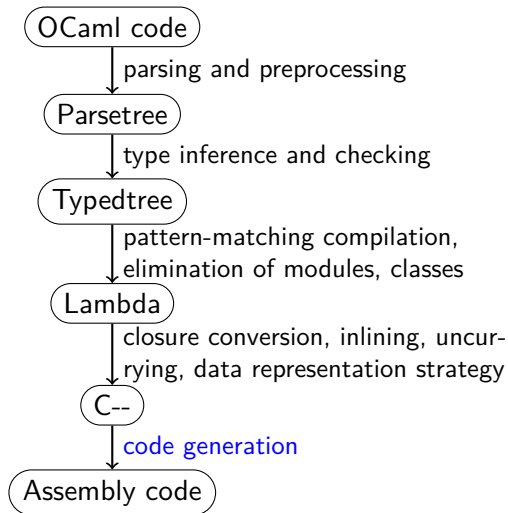
Why a new backend?

- ocamlopt works just fine
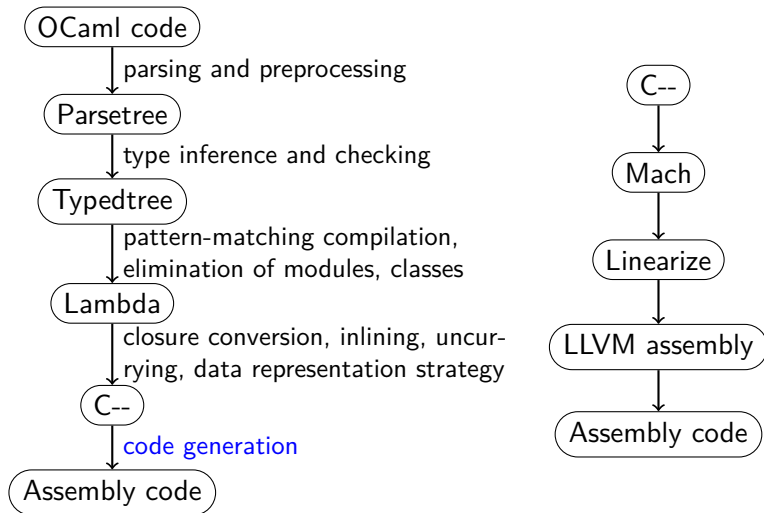- quite a lot of platform-specific code
- use LLVM instead

LLVM

- optimising compiler framework
- high level assembler
- single static assignment form (SSA)
- portable
- typed
- handles the generation of native assembly

The new backend

OCaml code

↓ parsing and preprocessing

Parsetree

↓ type inference and checking

Typedtree

↓ pattern-matching compilation, elimination of modules, classes

Lambda

↓ closure conversion, inlining, uncurrying, data representation strategy

C--

↓ code generation

Assembly code

The new backend

OCaml code
↓ parsing and preprocessing
Parsetree
↓ type inference and checking
Typedtree
↓ pattern-matching compilation, elimination of modules, classes
Lambda
↓ closure conversion, inlining, uncurrying, data representation strategy
C--
↓ code generation
Assembly code

C--
↓
Mach
↓
Linearize
↓
LLVM assembly
↓
Assembly code

The new backend

- AMD64 only
- incompatible with ocamlopt
- cannot use OCaml-style exceptions, instead use setjmp/longjmp
- problems with OCaml calling convention, use System V ABI calling convention
- garbage collection buggy
- rather slow

Conclusions

- LLVM lacks some features for handling OCaml nicely
- LLVM's garbage collection interface is not well suited for functional languages
- *much* lower complexity (almost no architecture dependent code in the backend)

Thank you for listening!

Contact: benner.colin@gmail.com

Code: http://github.com/colinbenner/ocamlllvm