# GHC Reading Guide

*- Exploring entrances and mental models to source code -*

Takenobu T.

Rev. 0.00.0
WIP

NOTE

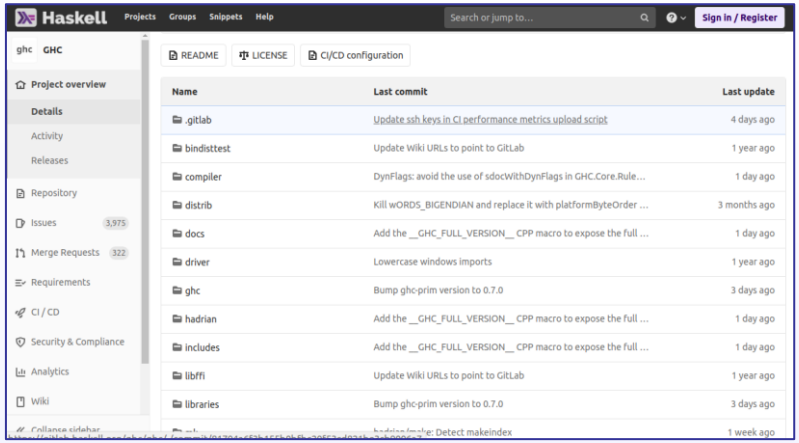
- This is not an official document by the ghc development team.
- Please refer to the official documents in detail.
- Don't forget “semantics”.  It's very important.
- This is written for ghc 8.12.

# Contents

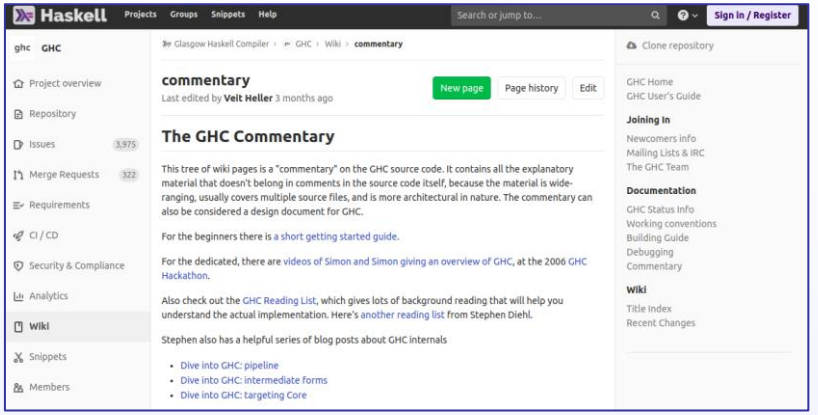
# Introduction

# Official resources are here

GHC source repository
https://gitlab.haskell.org/ghc/ghc

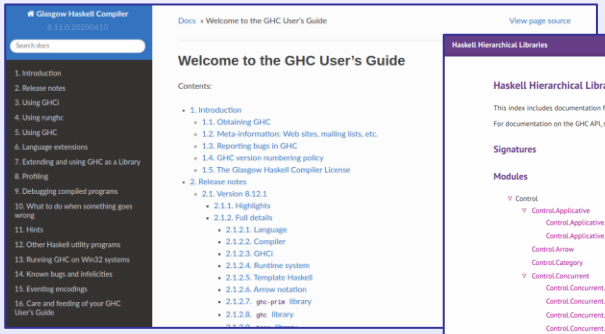



The GHC Commentary (for developers)
https://gitlab.haskell.org/ghc/ghc/-/wikis/commentary

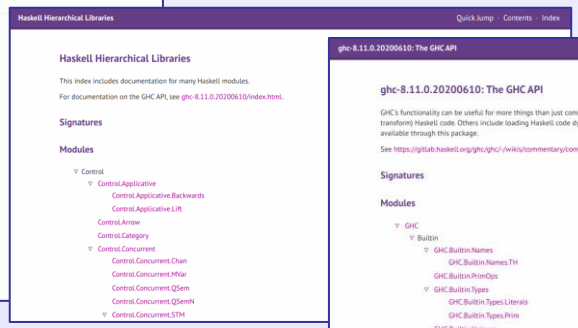



GHC Documentation (for users)

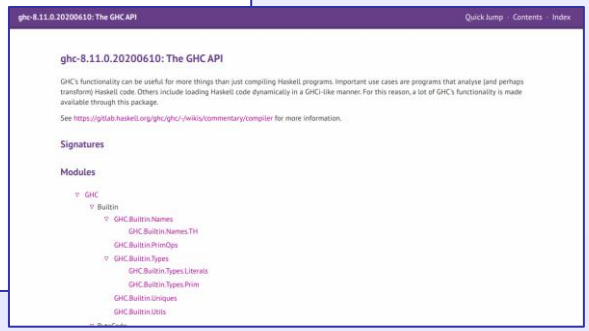
* master HEAD https://ghc.gitlab.haskell.org/ghc/doc/
* latest major release https://downloads.haskell.org/~ghc/latest/docs/html/
* version specified https://downloads.haskell.org/~ghc/8.12.1/docs/html/




The User's Guide



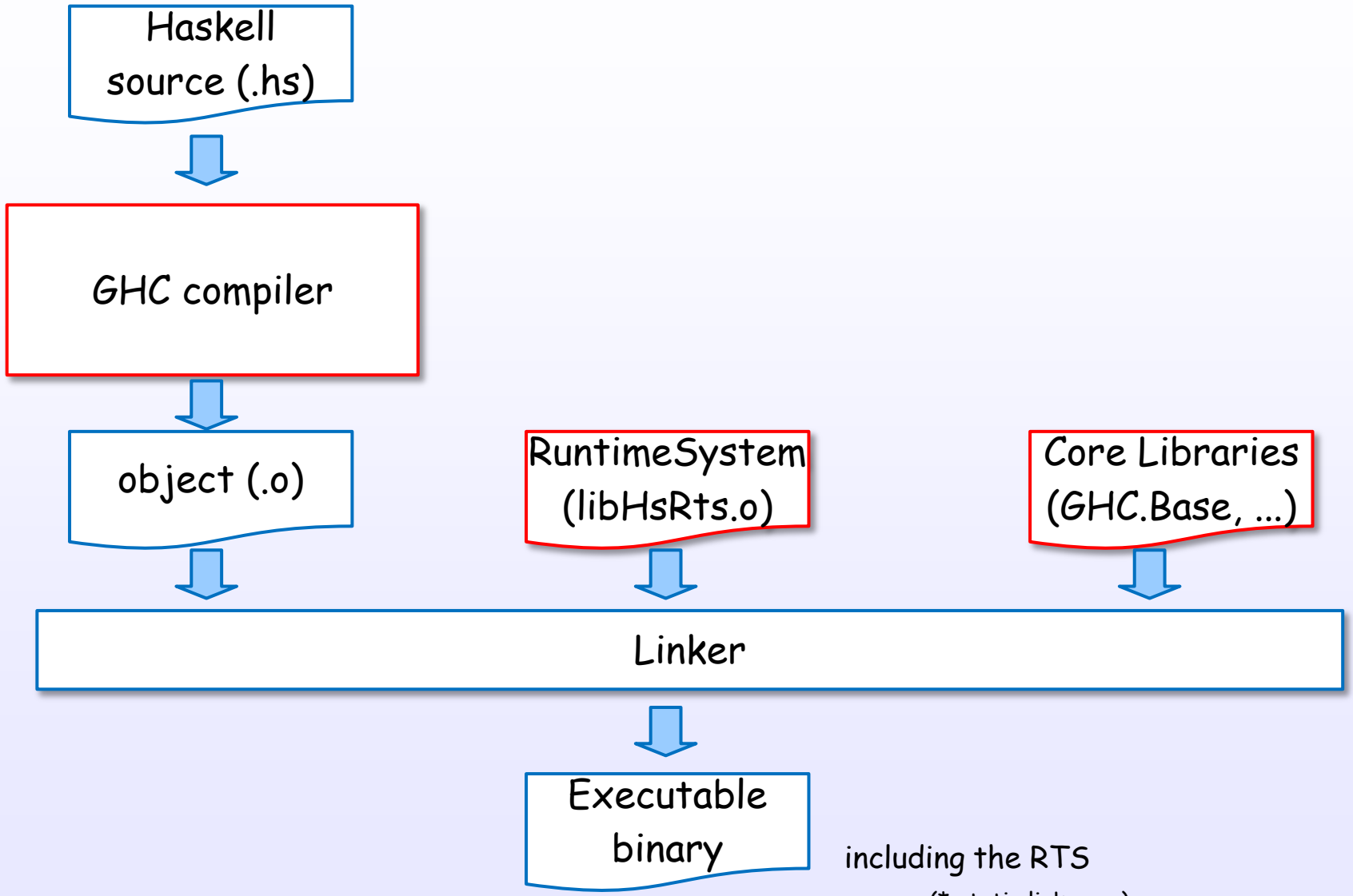

Core Libraries




GHC API

References : [C2], [22]

# The GHC = Compiler + Runtime System (RTS) + Core Libraries




References : [1], [C1], [C3], [C12], [C21], [S7], [21], [22]

# Source code organization

GHC official repository
https://gitlab.haskell.org/ghc/ghc

| | |
|---|---|
| compiler/ | ... compiler sources |
| rts/ | ... runtime system sources |
| libraries/ | ... core library sources |
| ghc/ | ... compiler main |
| includes/ | ... include files |
| testsuite/ | ... test suites |
| nofib/ | ... performance tests |
| mk/ | ... build system |
| hadrian/ | ... hadrian build system |
| docs/ | ... documents |
| : | : |

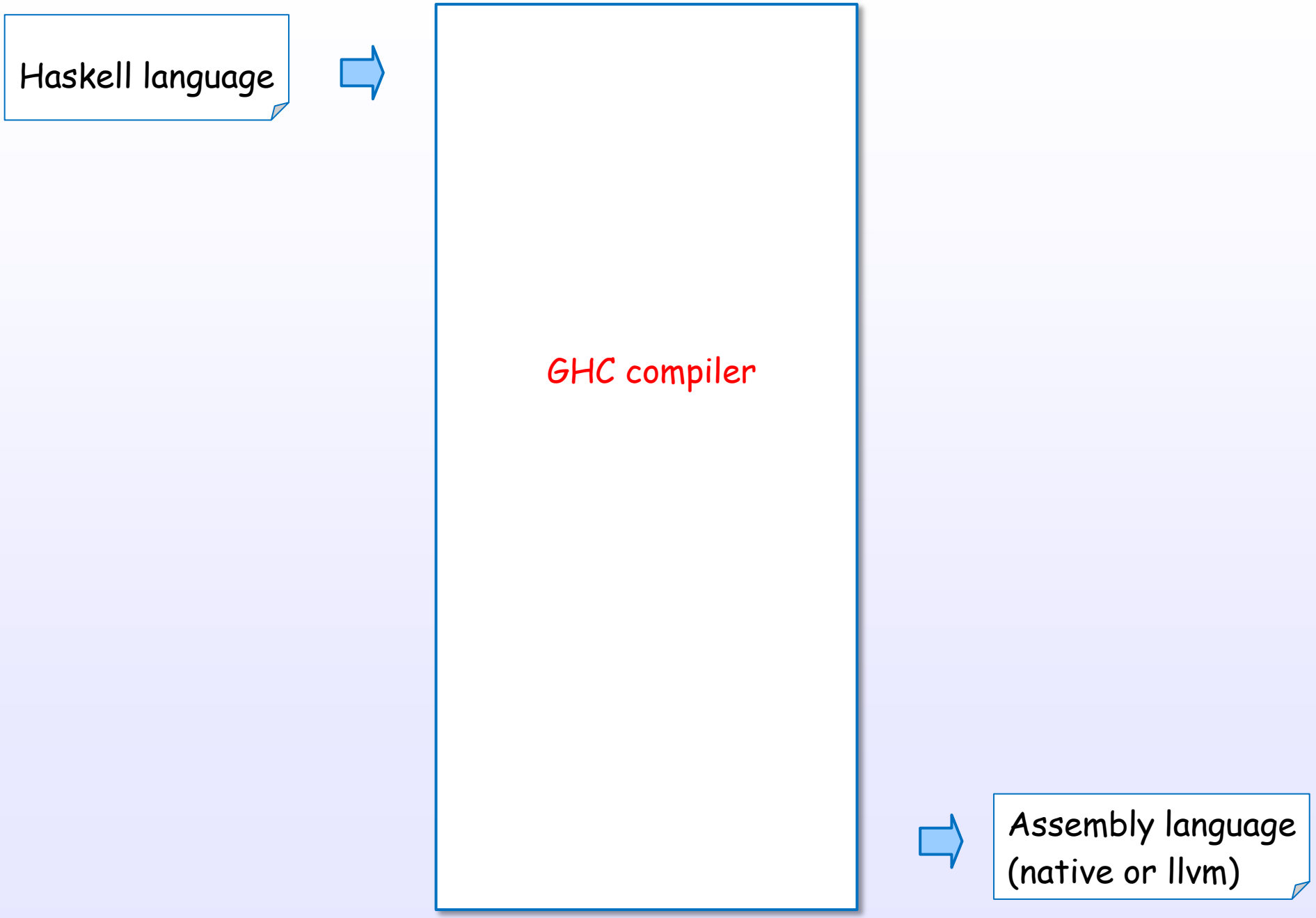
References : [C2], [22]

# 1. Compiler

# 1. Compiler

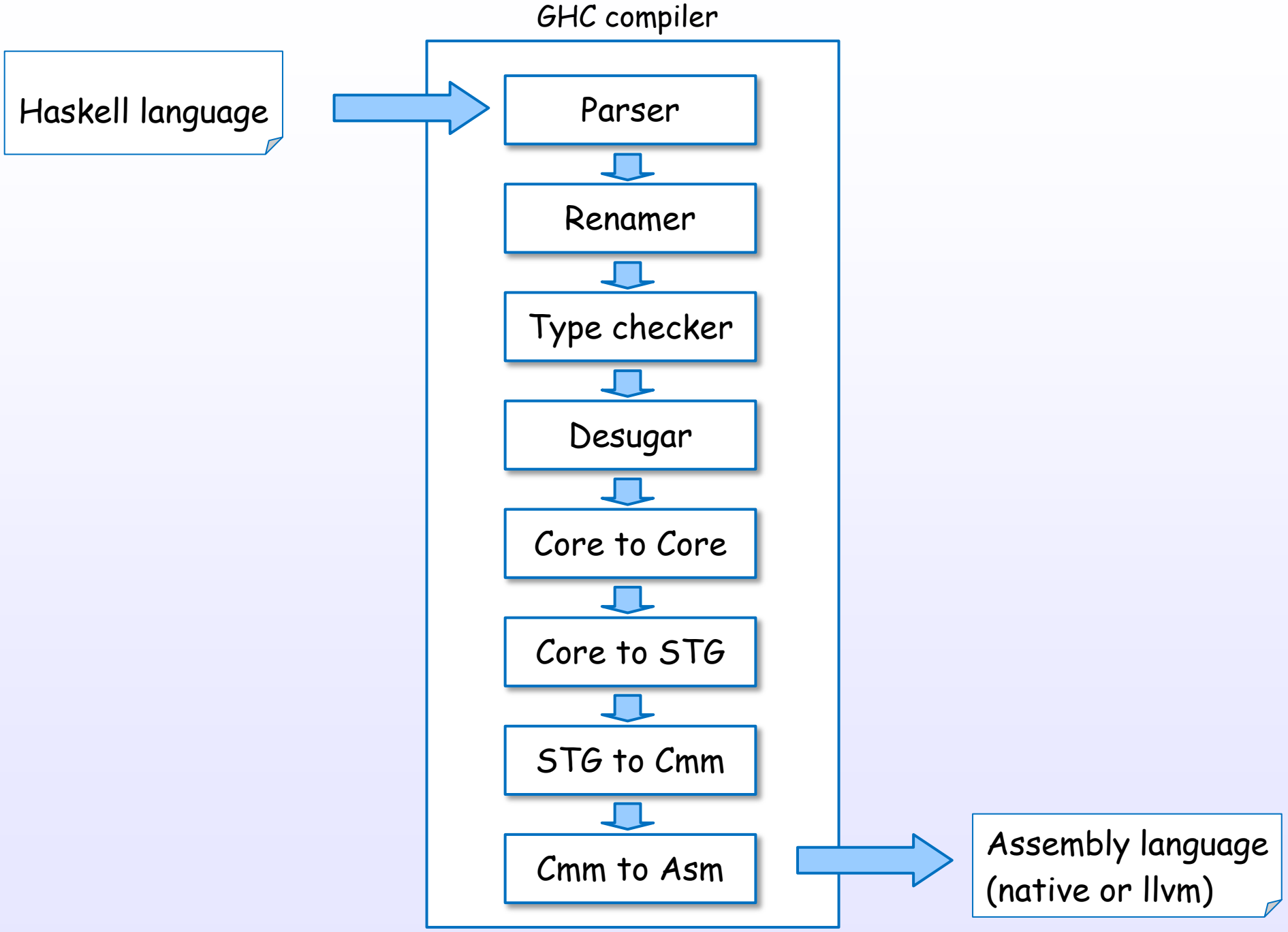
## Compilation pipeline

# The GHC compiler



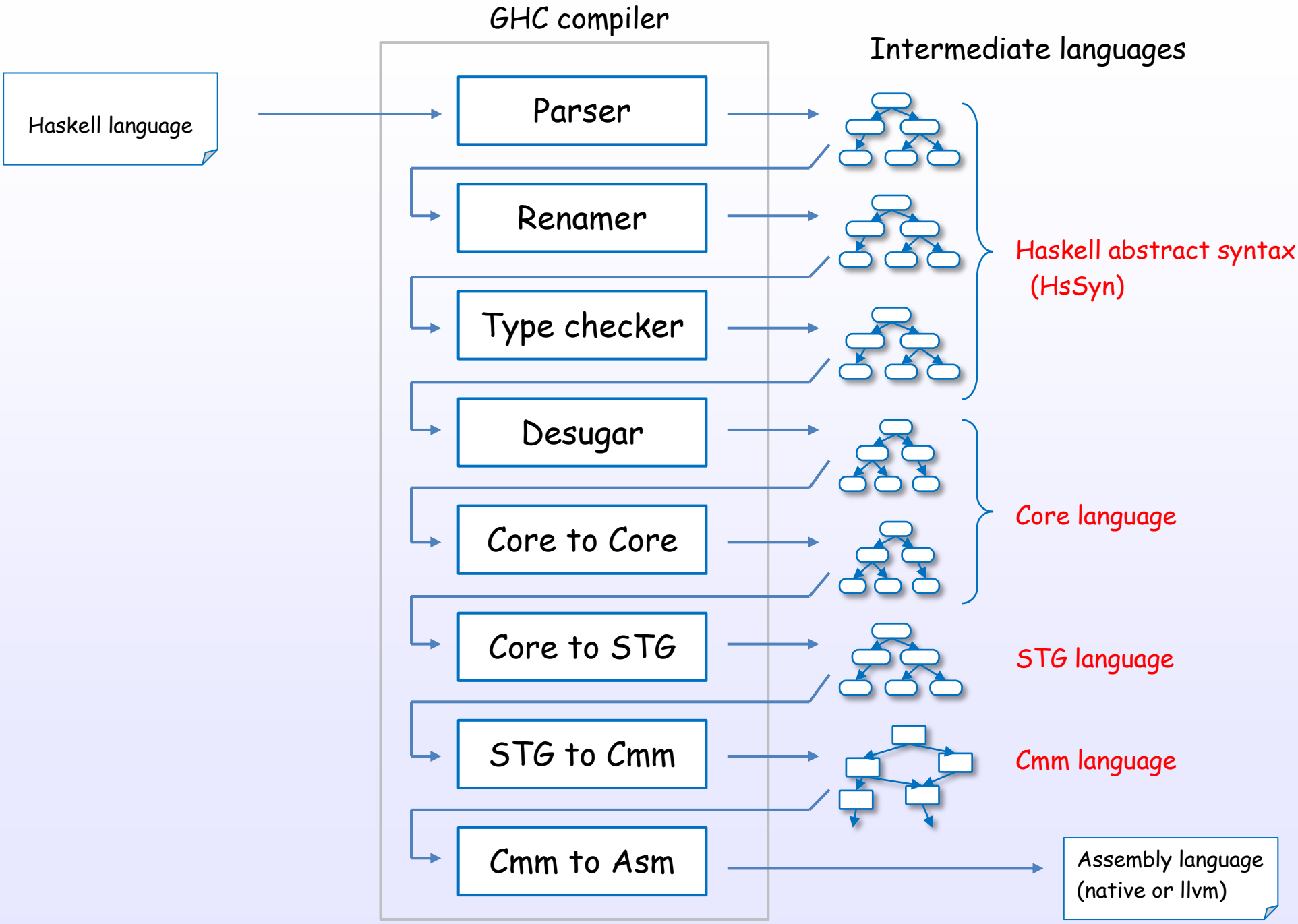

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# GHC compilation pipeline



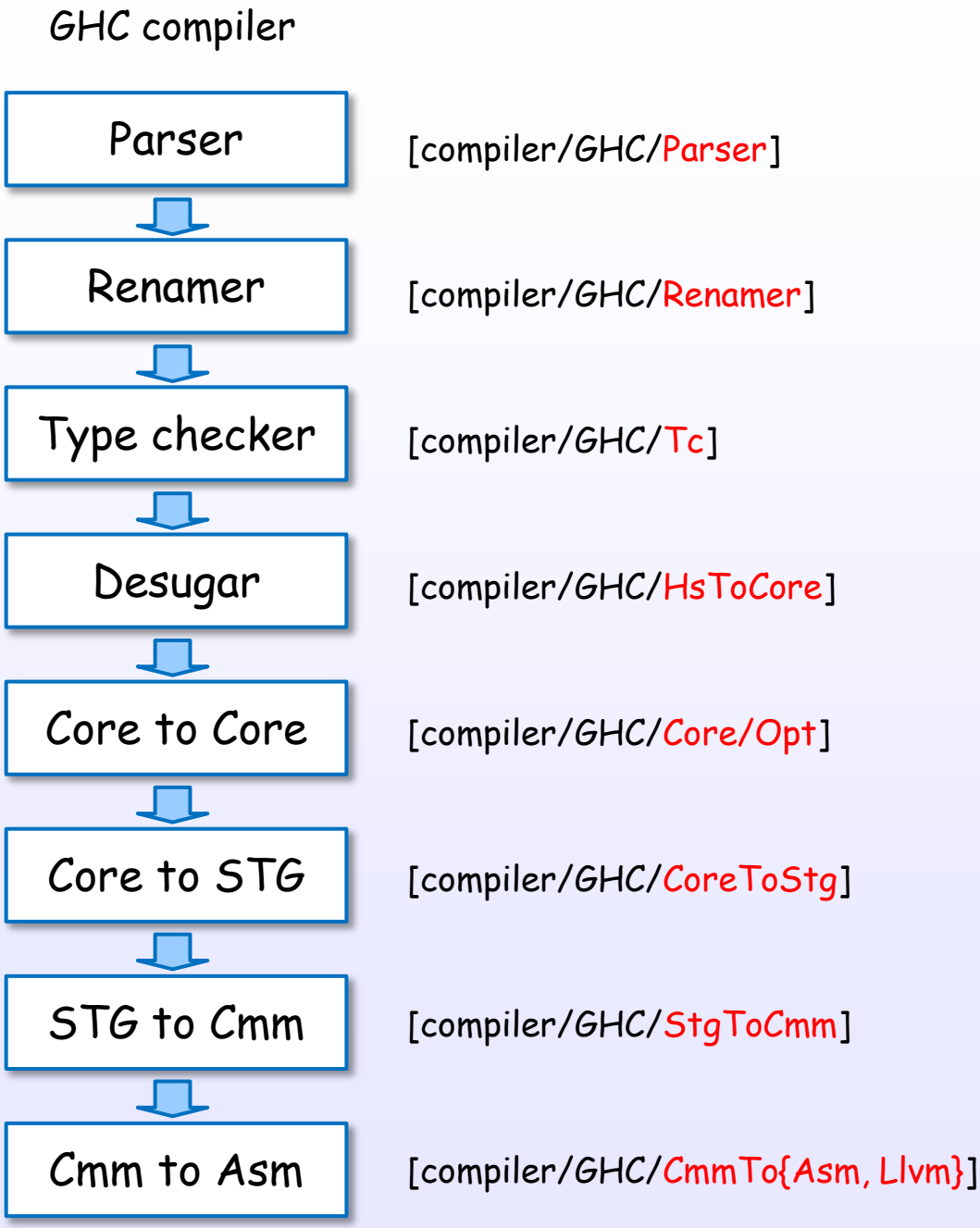

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# GHC compilation pipeline with intermediate languages




References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# Corresponding to source files



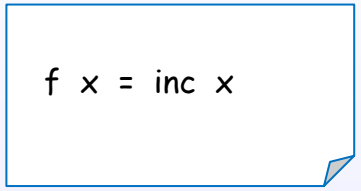

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

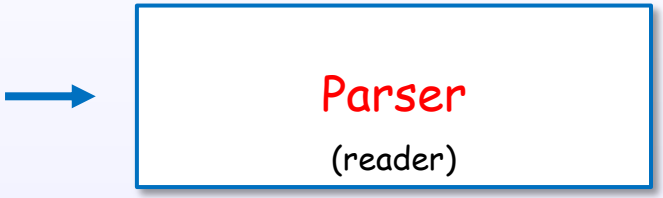
# 1. Compiler

Each pipeline stages

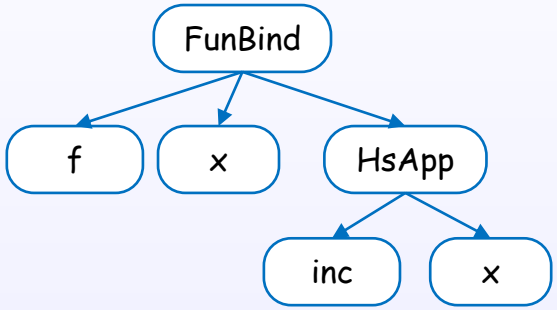
# Parser

Haskell source




HsSyn
(Haskell Abstract Syntax)

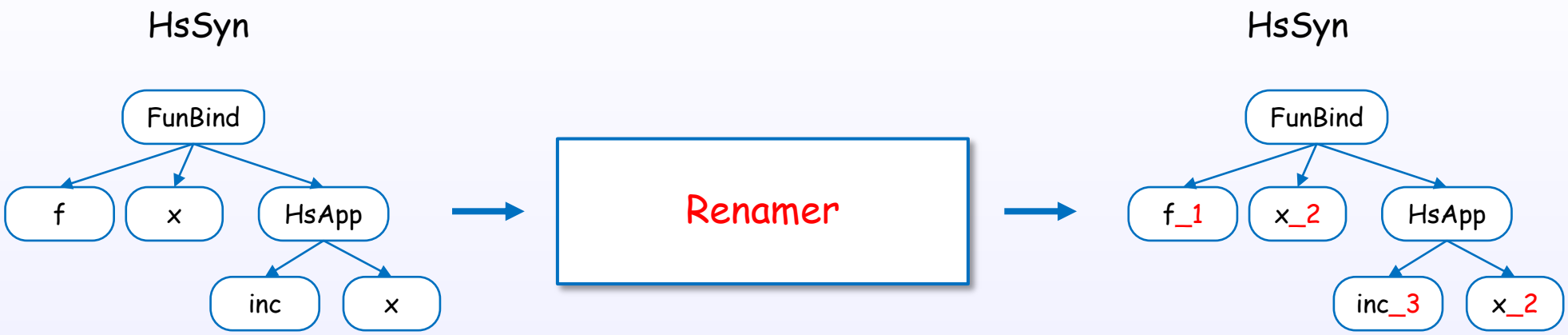



- Parse

Abstracted

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# Renamer

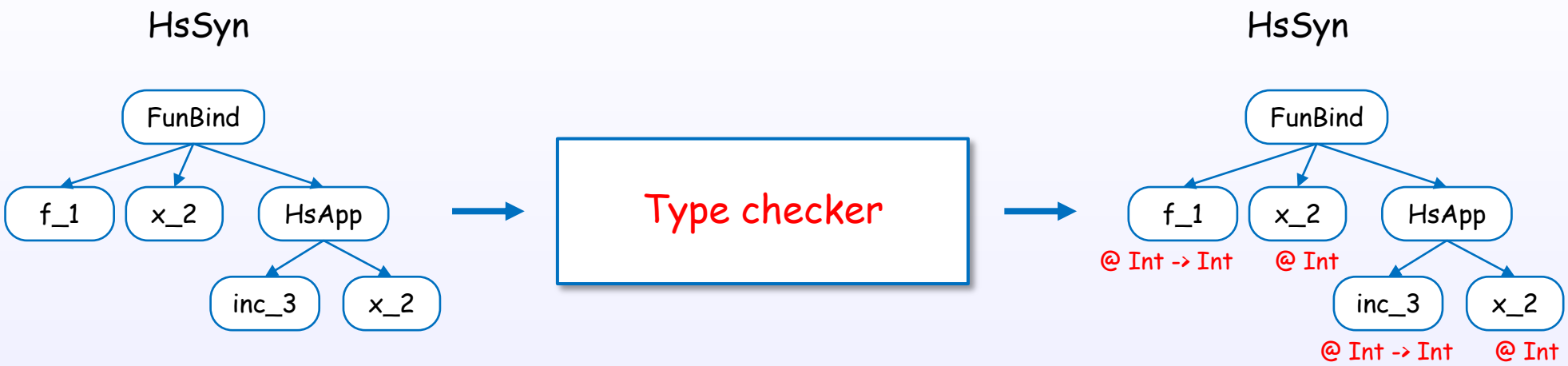



- Unify name
- Fixing
- Error check

Unique named

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# Type checker

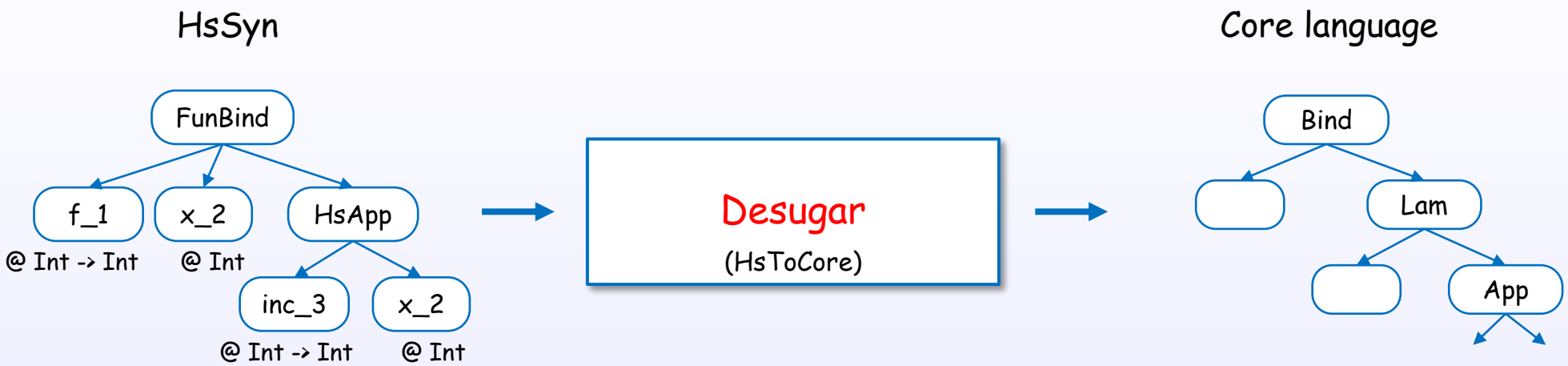



- Infer type
- Check type error

Fully typed

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# Desugar



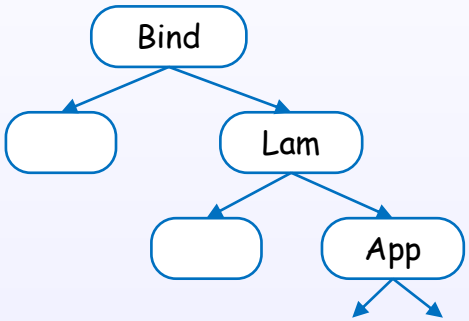

- Desugar to Core

Squeezed to λ calculus

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

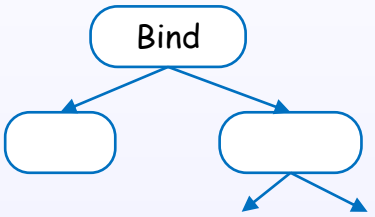
# Core to Core

Core language





Core language



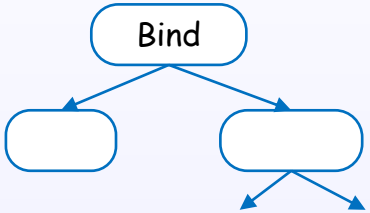

- Simplify

Optimized

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

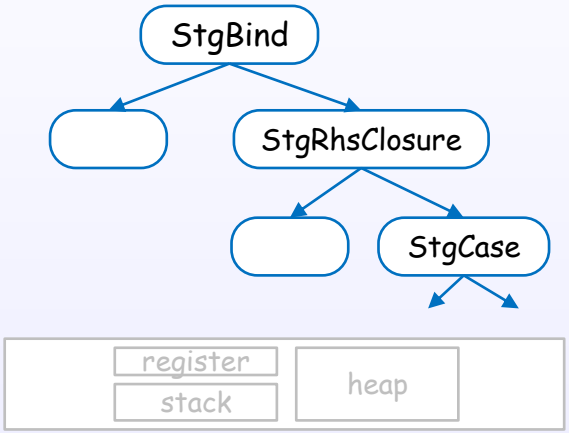
# Core to Stg

Core language





STG language

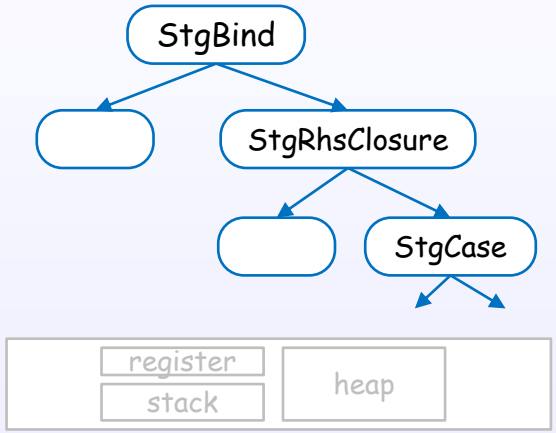



Operationally mapped

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# STG to Cmm

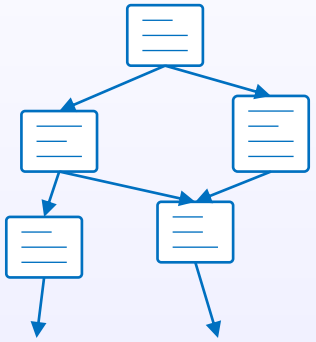
STG language




STG to Cmm

(code gen)

Cmm language

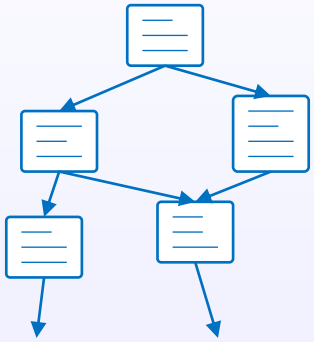


Instruction blocked

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# Cmm to Assembly

Cmm language



Cmm to Asm

(native code gen)

Assembly/LLVM language

```
add   $0x2, %rbx
jmpq  *0x0(%rbp)
  :
```

Machine coded

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# Intermediate language syntax

# HsSyn (Haskell abstract syntax)

[compiler/GHC/Hs/Decls.hs]

```
data HsDecl p
  = TyClD ...          -- Type or Class Declaration
  | InstD ...          -- Instance declaration
  | DerivD ...         -- Deriving declaration
  | ValD ...           -- Value declaration
  | SigD ...           -- Signature declaration
  | KindSigD ...       -- Standalone kind signature
  | DefD ...           -- 'default' declaration
  | ForD ...           -- Foreign declaration
  | WarningD ...       -- Warning declaration
  | AnnD ...           -- Annotation declaration
  | RuleD ...          -- Rule declaration
  | SpliceD ...        -- Splice declaration
  | DocD ...           -- Documentation comment declaration
  | RoleAnnotD ...     -- Role annotation declaration
  | XHsDecl ...
```

[compiler/GHC/Hs/Binds.hs]

```
data HsBindLR idL idR
  = FunBind ...        -- Function-like Binding
  | PatBind ...        -- Pattern Binding
  | VarBind ...        -- Variable Binding
  | AbsBinds ...       -- Abstraction Bindings
  | PatSynBind ...     -- Patterns Synonym Binding
  | XHsBindsLR ...
```

[compiler/GHC/Hs/Expr.hs]

```
data HsExpr p
  = HsVar ...
  | HsUnboundVar ...
  | HsConLikeOut ...
  | HsRecFld ...
  | HsOverLabel ...
  | HsIPVar ...
  | HsOverLit ...
  | HsLit ...
  | HsLam ...
  | HsLamCase ...
  | HsApp ...
  | HsAppType ...
  | OpApp ...
  | NegApp ...
  | HsPar ...
  | SectionL ...
  | SectionR ...
  | ExplicitTuple
  | ExplicitSum
  | HsCase ...
  | HsIf ...
  | HsMultiIf ...
  | HsLet ...
  | HsDo ...
  | ExplicitList
  | RecordCon
  | RecordUpd
  | ExprWithTySig
  | ArithSeq
      :
```

Abstract syntax corresponding to Haskell user source.

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# Core language

[compiler/GHC/Core.hs]

```
type CoreProgram = [CoreBind]
type CoreBndr = Var
type CoreExpr = Expr CoreBndr
type CoreArg  = Arg  CoreBndr
type CoreBind = Bind CoreBndr
type CoreAlt  = Alt  CoreBndr

data Expr b
  = Var   Id                          -- Variable
  | Lit   Literal                     -- Literal
  | App   (Expr b)  (Arg b)           -- Application
  | Lam   b  (Expr b)                 -- Lambda abstraction
  | Let   (Bind b)  (Expr b)          -- Variable binding
  | Case  (Expr b)  b  Type  [Alt b]  -- Pattern match
  | Cast  (Expr b)  Coercion          -- Cast
  | Tick  (Tickish Id)  (Expr b)      -- Internal note
  | Type  Type                        -- Type
  | Coercion Coercion                 -- Coercion
```

Minimal typed functional language.
Only ten data constructors based on System FC.

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# STG language

[compiler/GHC/Stg/Syntax.hs]

```
data GenStgTopBinding pass
  = StgTopLifted (GenStgBinding pass)  |  StgTopStringLit Id ByteString

data GenStgBinding pass
  = StgNonRec (BinderP pass) (GenStgRhs pass)  |  StgRec  [(BinderP pass, GenStgRhs pass)]

data GenStgRhs pass
  = StgRhsClosure  (XRhsClosure pass)  CostCentreStack  !UpdateFlag  [BinderP pass]  (GenStgExpr pass)
  | StgRhsCon        CostCentreStack DataCon [StgArg]

data GenStgExpr pass
  = StgApp           Id  [StgArg]
  | StgLit           Literal
  | StgConApp        DataCon  [StgArg]  [Type]
  | StgOpApp         StgOp  [StgArg]  Type
  | StgLam           (NonEmpty  (BinderP pass))  StgExpr
  | StgCase          (GenStgExpr pass)  (BinderP pass)  AltType  [GenStgAlt  pass]
  | StgLet           (XLet pass)  (GenStgBinding pass)  (GenStgExprpass)
  | StgLetNoEscape   (XLetNoEscape pass)  (GenStgBinding pass)  (GenStgExpr pass)
  | StgTick          (Tickish Id)  (GenStgExpr pass)
```

Tiny functional language for abstract machine (STG-machine) semantics.

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# Cmm language

[compiler/GHC/Cmm.hs]

```
type CmmProgram = [CmmGroup]
type CmmGroup   = GenCmmGroup CmmStatics CmmTopInfo CmmGraph
type CmmGraph = GenCmmGraph CmmNode
```

[compiler/GHC/Cmm/Node.hs]

```
data CmmNode e x where
  CmmEntry ...                                          -- Entry
  CmmComment ...                                        -- Comment
  CmmTick ...                                           -- Tick annotation
  CmmUnwind ...                                         -- Unwind pseudo-instruction
  CmmAssign :: !CmmReg -> !CmmExpr -> CmmNode O O       -- Assign to register
  CmmStore ...                                          -- Assign to memory location
  CmmUnsafeForeignCall ...                              -- An unsafe foreign call
  CmmBranch ...                                         -- Goto another block
  CmmCondBranch ...                                     -- Conditional branch
  CmmSwitch ...                                         -- Switch
  CmmCall ...                                           -- A native call or tail call
  CmmForeignCall ...                                    -- A safe foreign call
```

[compiler/GHC/Cmm/Expr.hs]

```
data CmmExpr
  = CmmLit        CmmLit                                -- Literal
  | CmmLoad       !CmmExpr  !CmmType                    -- Read memory location
  | CmmReg        !CmmReg                               -- Contents of register
  | CmmMachOp     MachOp  [CmmExpr]                     -- Machine operation (+, -, *, etc.)
  | CmmStackSlot  Area  {-# UNPACK #-} !Int
  | CmmRegOff     !CmmReg  Int
```

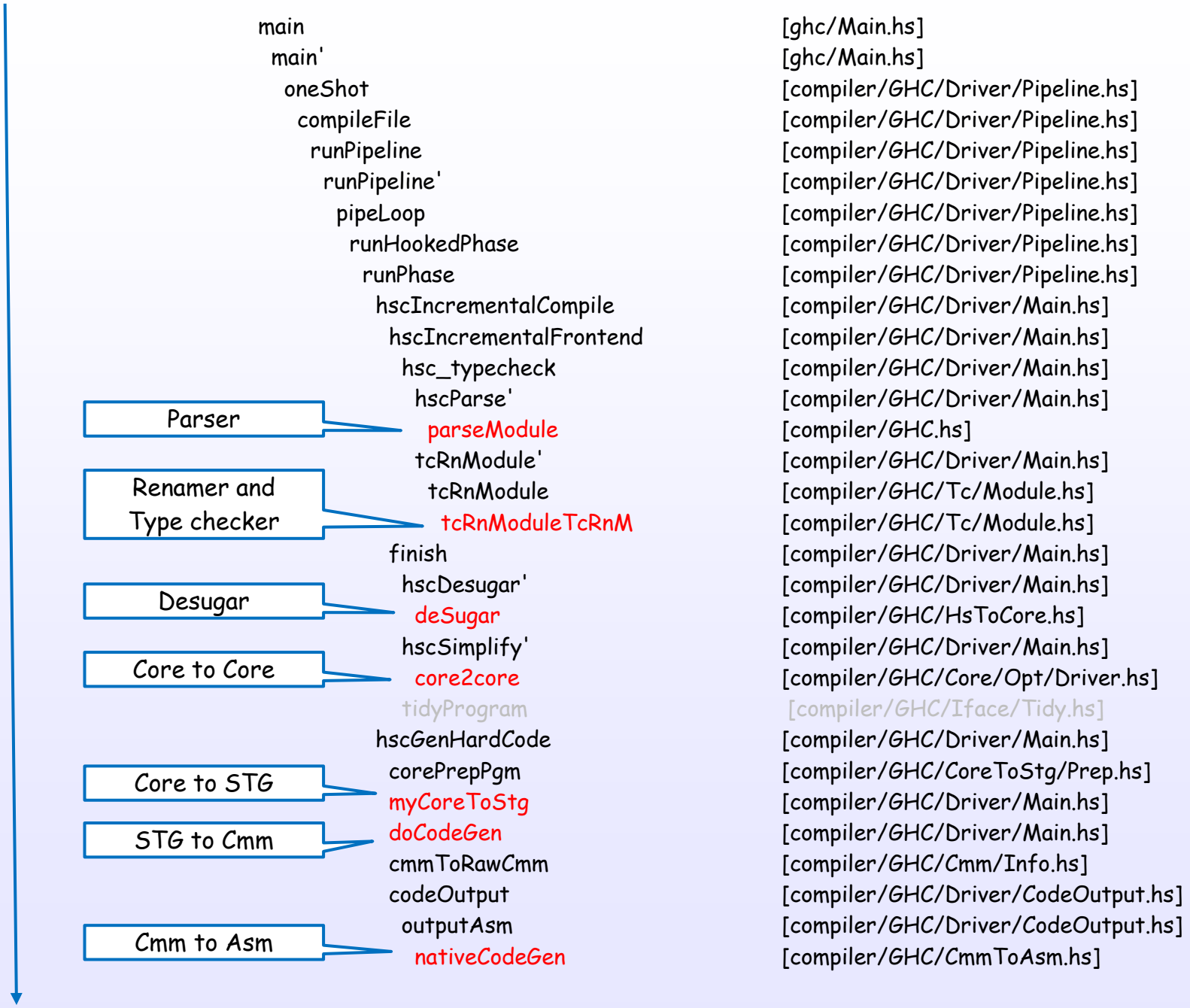
Portable assembly with imperative language.

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# 1. Compiler

## Call graph

# Example of call graph




References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# Appendix

# Dump intermediate languages

*Each intermediate code can be dumped by :*

```
$ ghc -ddump-parsed
$ ghc -ddump-rn
$ ghc -ddump-tc
```

```
$ ghc -ddump-parsed-ast
$ ghc -ddump-rn-ast
$ ghc -ddump-tc-ast
```

```
$ ghc -ddump-ds
$ ghc -ddump-simpl
$ ghc -ddump-prep
```

```
$ ghc -ddump-stg
```

```
$ ghc -ddump-cmm
$ ghc -ddump-opt-cmm
```

```
$ ghc -ddump-asm
$ ghc -ddump-llvm
```

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# tmp

* Top driver
* .hi file path
* GHCi path
* GHC API

References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

# References

# References

aosabook
dive-into-core
cs
users guide

Parser

Renamer

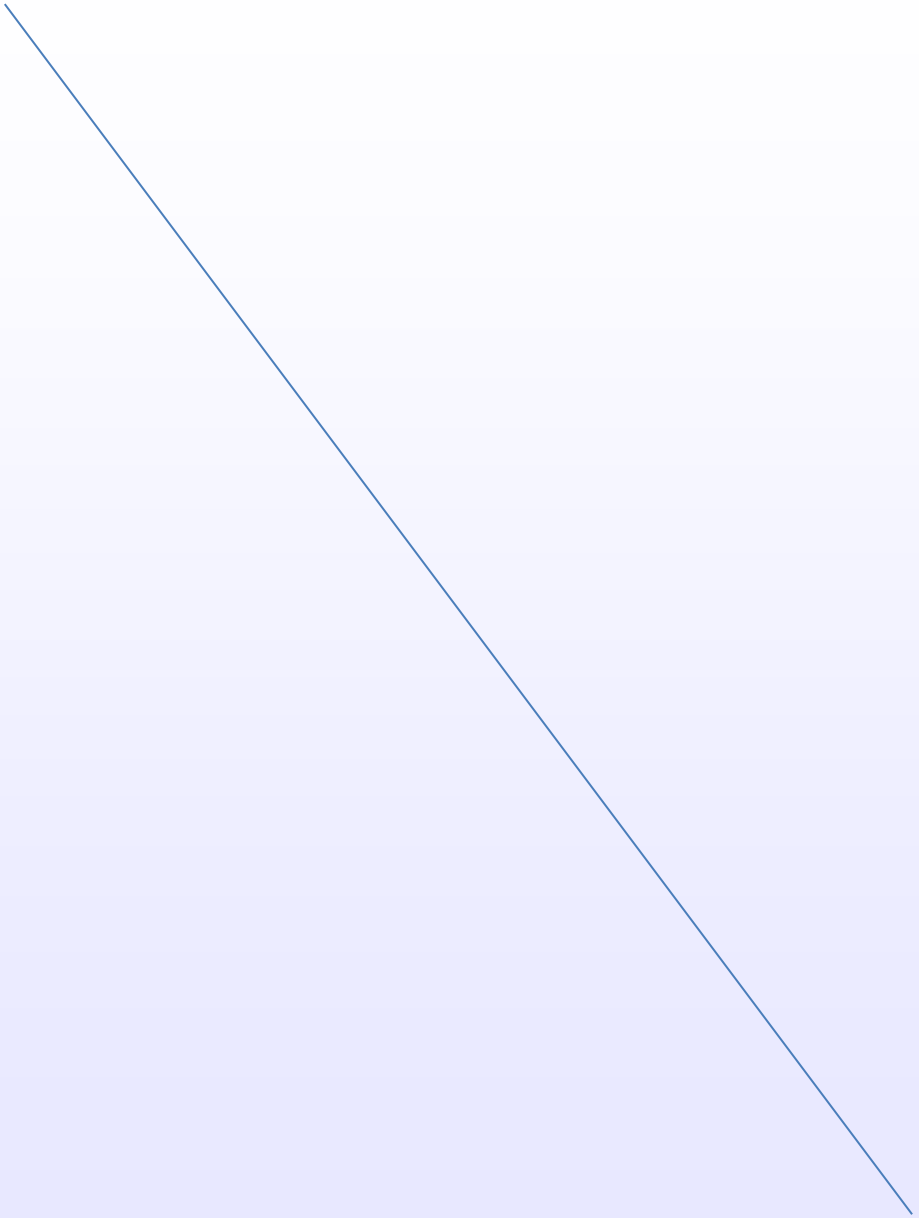
HsSyn

Source code

[S1] compiler/GHC

The GHC Commentary

[C1] https://gitlab.haskell.org/ghc/ghc/-/wikis/commentary



Happy haskelling!

Here is the slide: https://github.com/takenobu-hs/haskell-ghc-reading-guide