GHC Reading Guide

- Exploring entrance gates and mental models -

Takenobu T.

Rev. 0.00.0
WIP

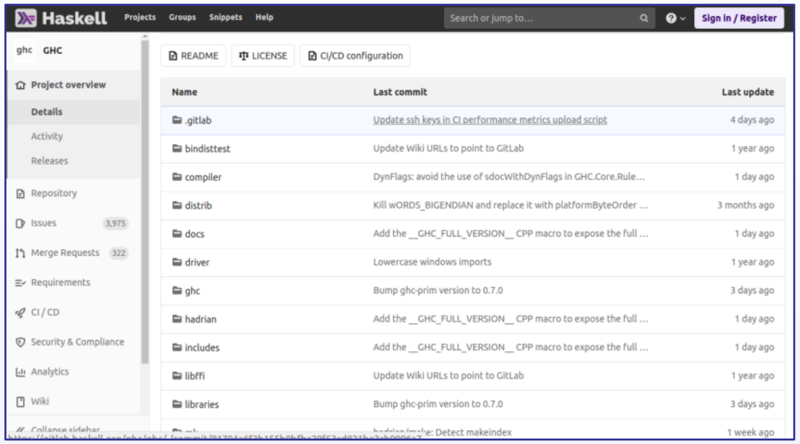
NOTE
- This is not an official document by the ghc development team.
- Please refer to the official documents in detail.
- Don't forget “semantics”. It's very important.
- This is written for ghc 8.12.

Contents

Introduction

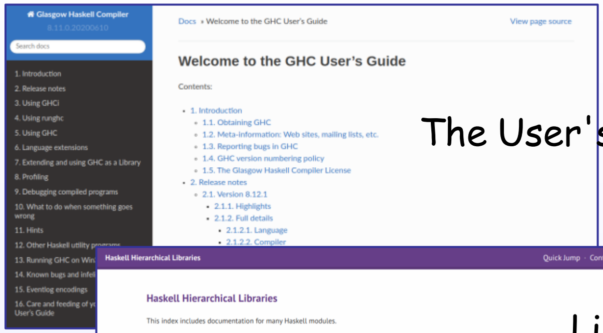
Official resources are here

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



Name	Last commit	Last update
.gitlab	Update ssh keys in CI performance metrics upload script	4 days ago
bindisttest	Update Wiki URLs to point to GitLab	1 year ago
compiler	DynFlags: avoid the use of sdocWithDynFlags in GHC.Core.Rule...	1 day ago
distrib	Kill wORDS_BIGENDIAN and replace it with platformByteOrder ...	3 months ago
docs	Add the __GHC_FULL_VERSION__ CPP macro to expose the full ...	1 day ago
driver	Lowercase windows imports	1 year ago
ghc	Bump ghc-prim version to 0.7.0	3 days ago
hadrian	Add the __GHC_FULL_VERSION__ CPP macro to expose the full ...	1 day ago
includes	Add the __GHC_FULL_VERSION__ CPP macro to expose the full ...	1 day ago
libffi	Update Wiki URLs to point to GitLab	1 year ago
libraries	Bump ghc-prim version to 0.7.0	3 days ago
[illegible]	hadrian/make: Detect makeindex	1 week ago

GHC Documentation
https://downloads.haskell.org/~ghc/latest/docs/html/

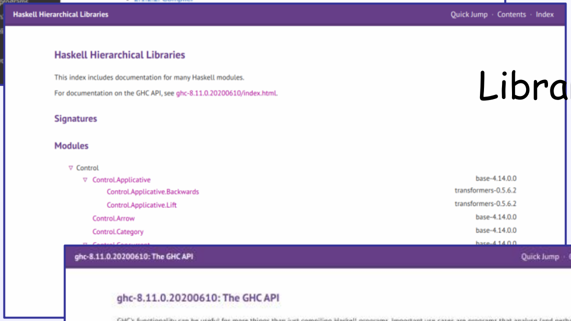


Welcome to the GHC User's Guide

Contents:

- 1. Introduction
  - 1.1. Obtaining GHC
  - 1.2. Meta-information: Web sites, mailing lists, etc.
  - 1.3. Reporting bugs in GHC
  - 1.4. GHC version numbering policy
  - 1.5. The Glasgow Haskell Compiler License
- 2. Release notes
  - 2.1. Version 8.12.1
    - 2.1.1. Highlights
    - 2.1.2. Full details
      - 2.1.2.1. Language
      - 2.1.2.2. Compiler

The User's Guide



Haskell Hierarchical Libraries

This index includes documentation for many Haskell modules.

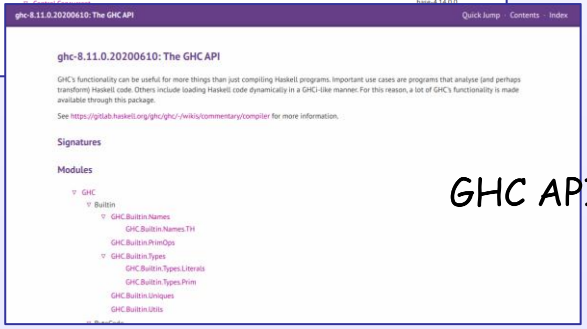
For documentation on the GHC API, see ghc-8.11.0.20200610/index.html.

Signatures

Modules

```
▽ Control
  ▽ Control.Applicative                         base-4.14.0.0
      Control.Applicative.Backwards             transformers-0.5.6.2
      Control.Applicative.Lift                  transformers-0.5.6.2
    Control.Arrow                               base-4.14.0.0
    Control.Category                            base-4.14.0.0
```

Libraries


ghc-8.11.0.20200610: The GHC API
Quick Jump · Contents · Index

ghc-8.11.0.20200610: The GHC API

GHC's functionality can be useful for more things than just compiling Haskell programs. Important use cases are programs that analyse (and perhaps transform) Haskell code. Others include loading Haskell code dynamically in a GHCi-like manner. For this reason, a lot of GHC's functionality is made available through this package.

See https://gitlab.haskell.org/ghc/ghc/-/wikis/commentary/compiler for more information.

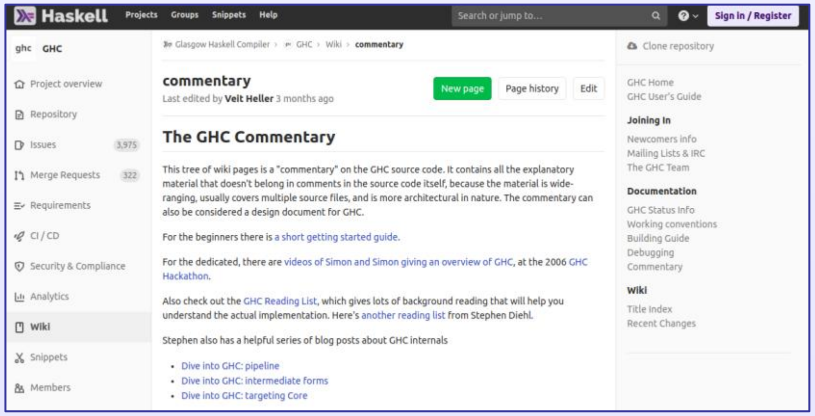
Signatures

Modules

```
▽ GHC
  ▽ Builtin
    ▽ GHC.Builtin.Names
        GHC.Builtin.Names.TH
      GHC.Builtin.PrimOps
    ▽ GHC.Builtin.Types
        GHC.Builtin.Types.Literals
        GHC.Builtin.Types.Prim
      GHC.Builtin.Uniques
      GHC.Builtin.Utils
```

GHC API

The GHC Commentary
https://gitlab.haskell.org/ghc/ghc/-/wikis/commentary



commentary
Last edited by Veit Heller 3 months ago

The GHC Commentary

This tree of wiki pages is a "commentary" on the GHC source code. It contains all the explanatory material that doesn't belong in comments in the source code itself, because the material is wide-ranging, usually covers multiple source files, and is more architectural in nature. The commentary can also be considered a design document for GHC.

For the beginners there is a short getting started guide.

For the dedicated, there are videos of Simon and Simon giving an overview of GHC, at the 2006 GHC Hackathon.

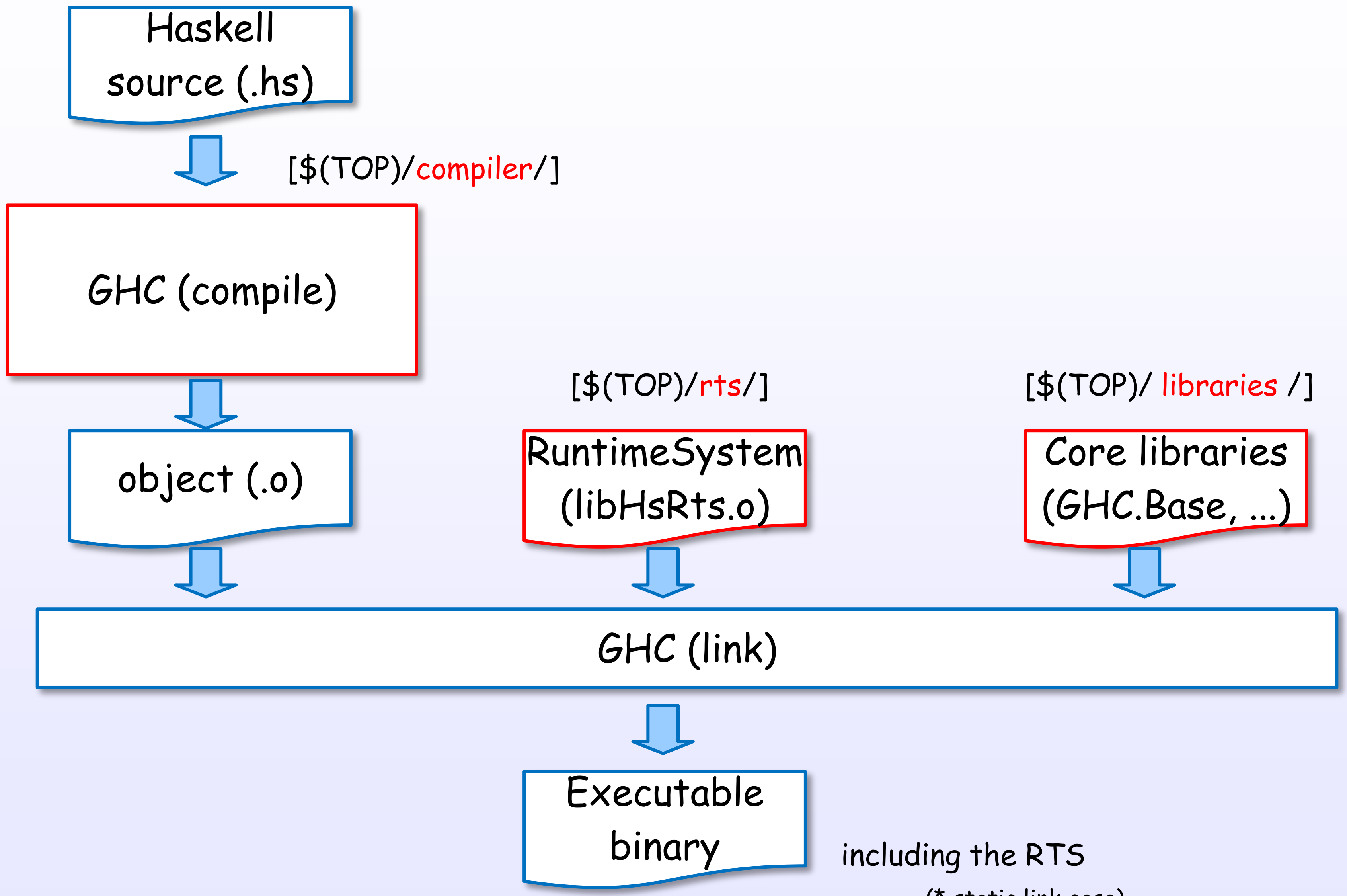
Also check out the GHC Reading List, which gives lots of background reading that will help you understand the actual implementation. Here's another reading list from Stephen Diehl.

Stephen also has a helpful series of blog posts about GHC internals

- Dive into GHC: pipeline
- Dive into GHC: intermediate forms
- Dive into GHC: targeting Core

References : [C2], [22]

The GHC = Compiler + Runtime System (RTS) + Core libraries



including the RTS

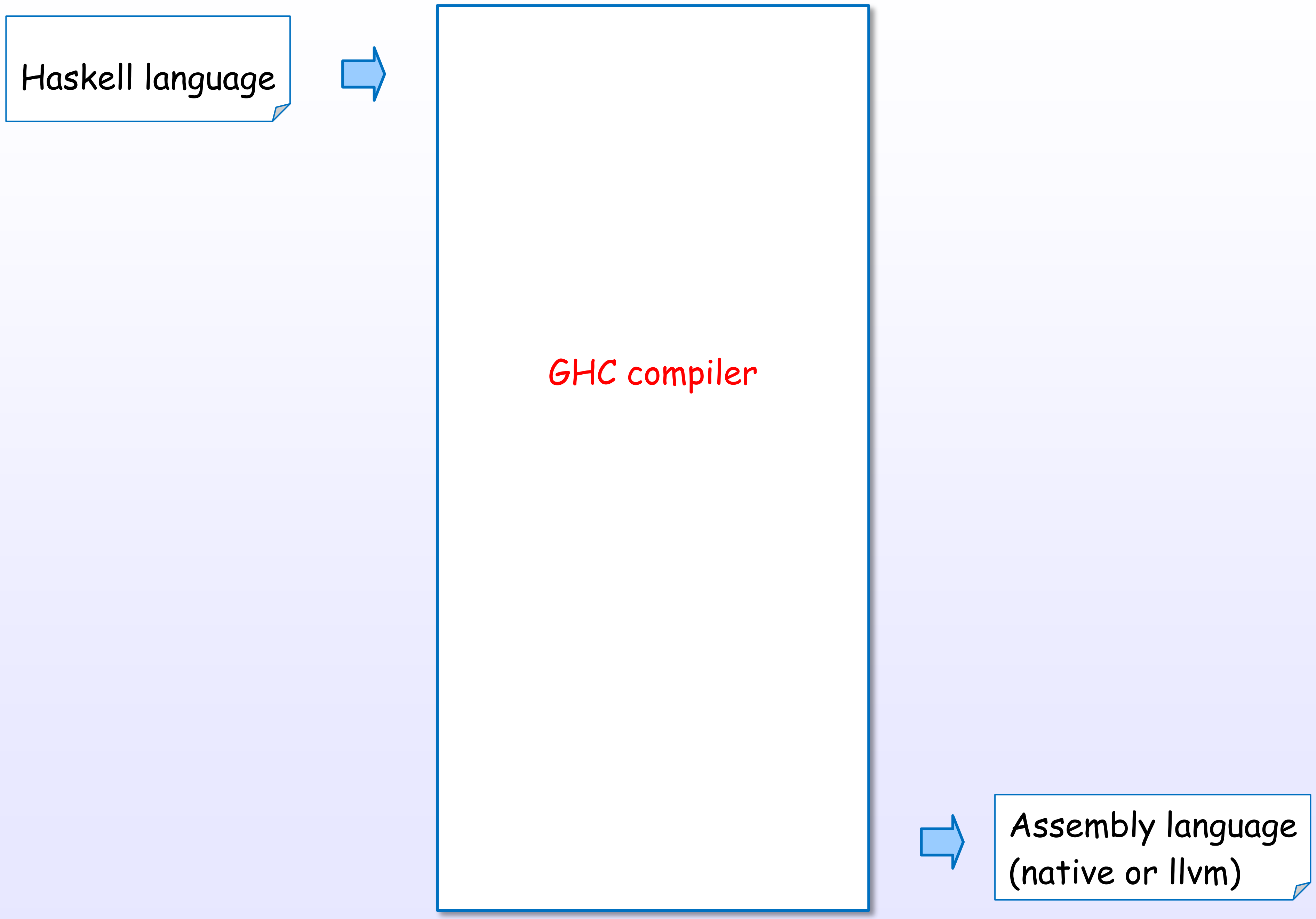
(* static link case)

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

1. Compiler

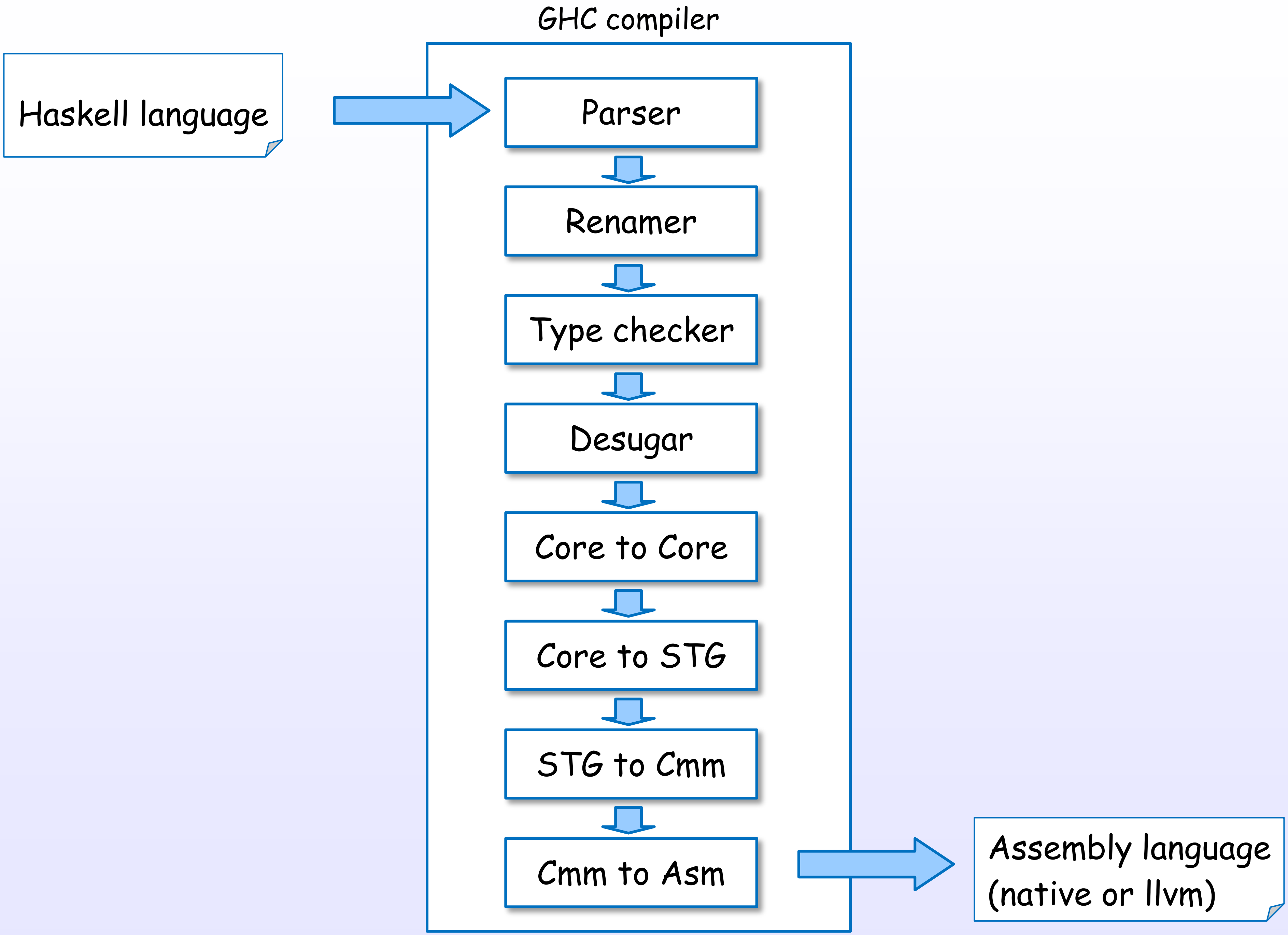
Compilation pipeline

The GHC compiler



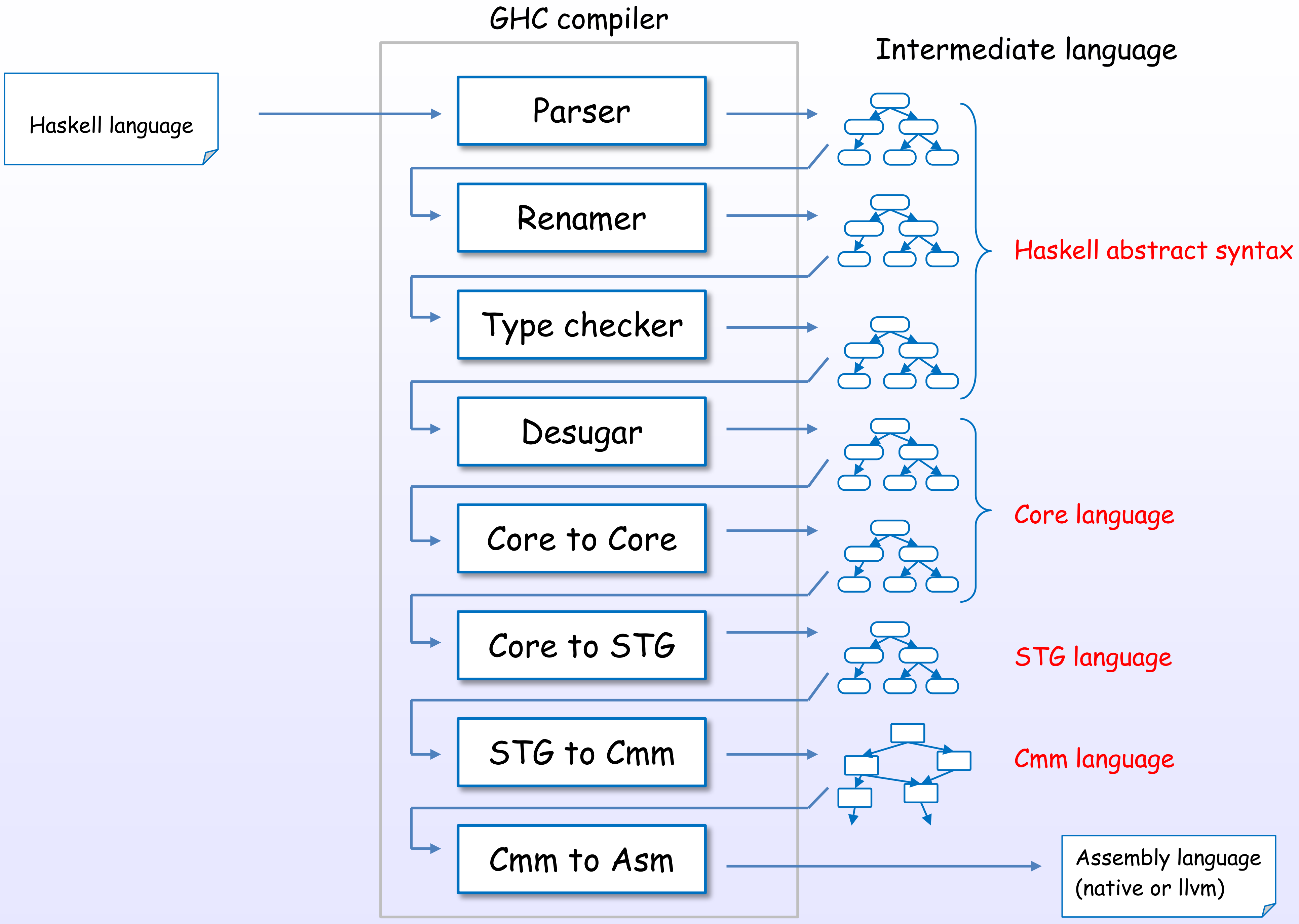
References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

GHC compilation pipeline



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

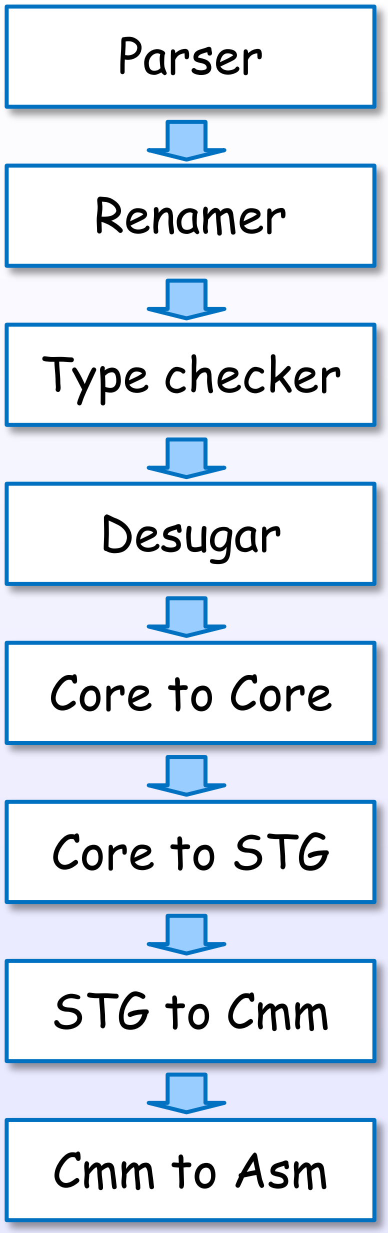
GHC compilation pipeline with intermediate languages



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

Corresponding to source files

GHC compiler



[compiler/GHC/Parser]

[compiler/GHC/Renamer]

[compiler/GHC/Tc]

[compiler/GHC/HsToCore]

[compiler/GHC/Core/Opt]

[compiler/GHC/CoreToStg]

[compiler/GHC/StgToCmm]

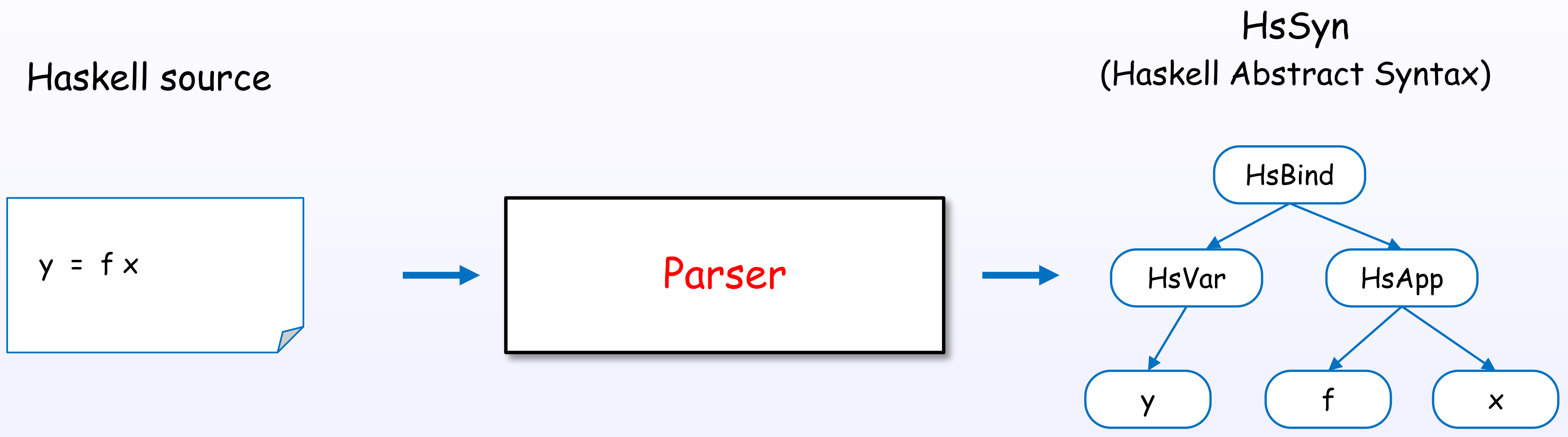
[compiler/GHC/CmmTo{Asm, Llvm}]

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

1. Compiler

Each pipeline stages

Parser

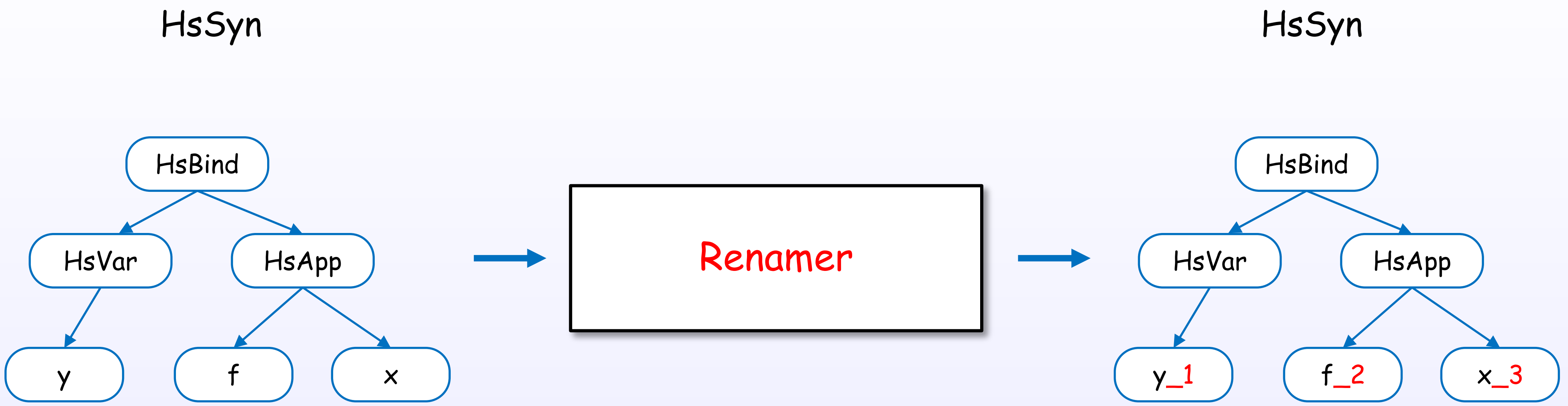


- Parse

Abstracted

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

Renamer

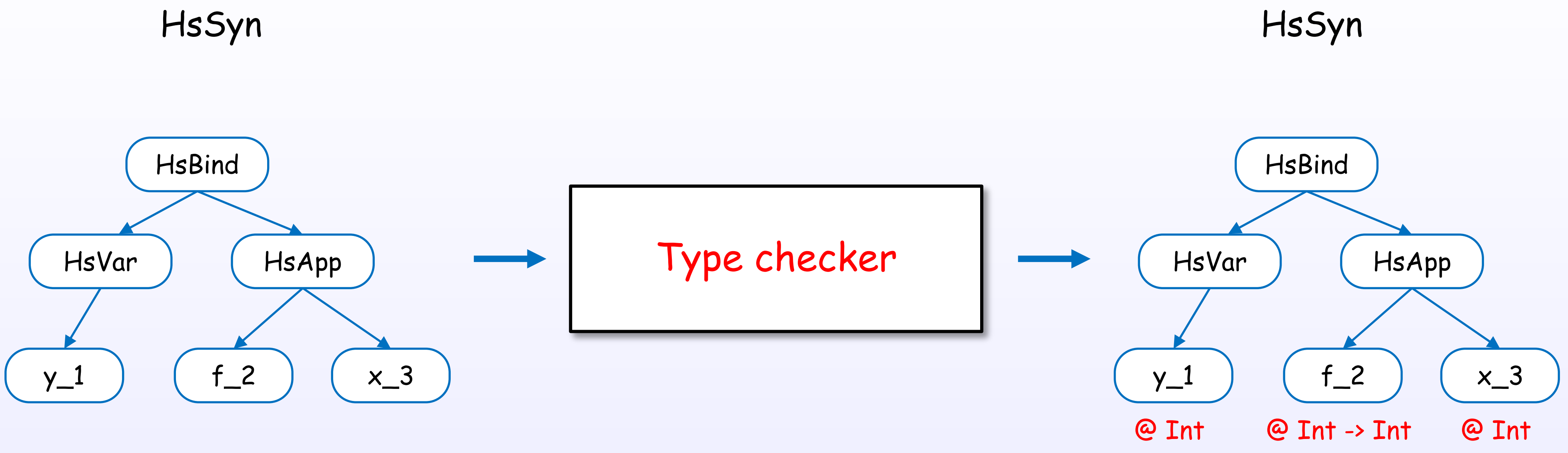


- Unify name
- Fixing
- Error check

Unique named

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

Type checker

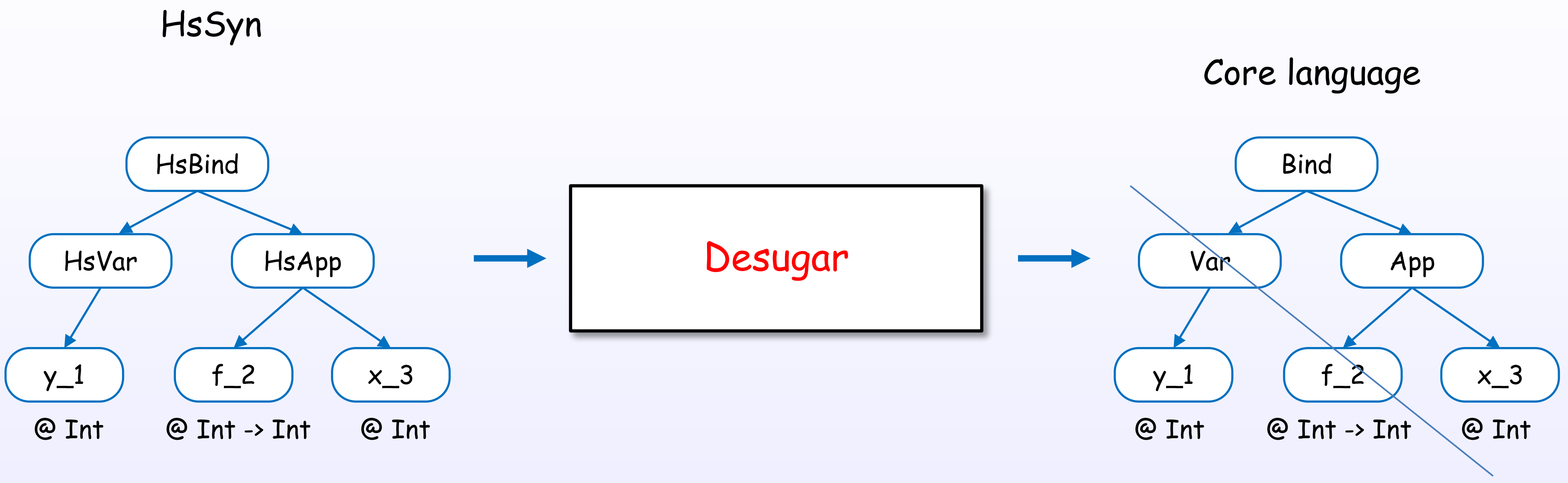


- Infer type
- Check type error

Fully typed

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

Desugar

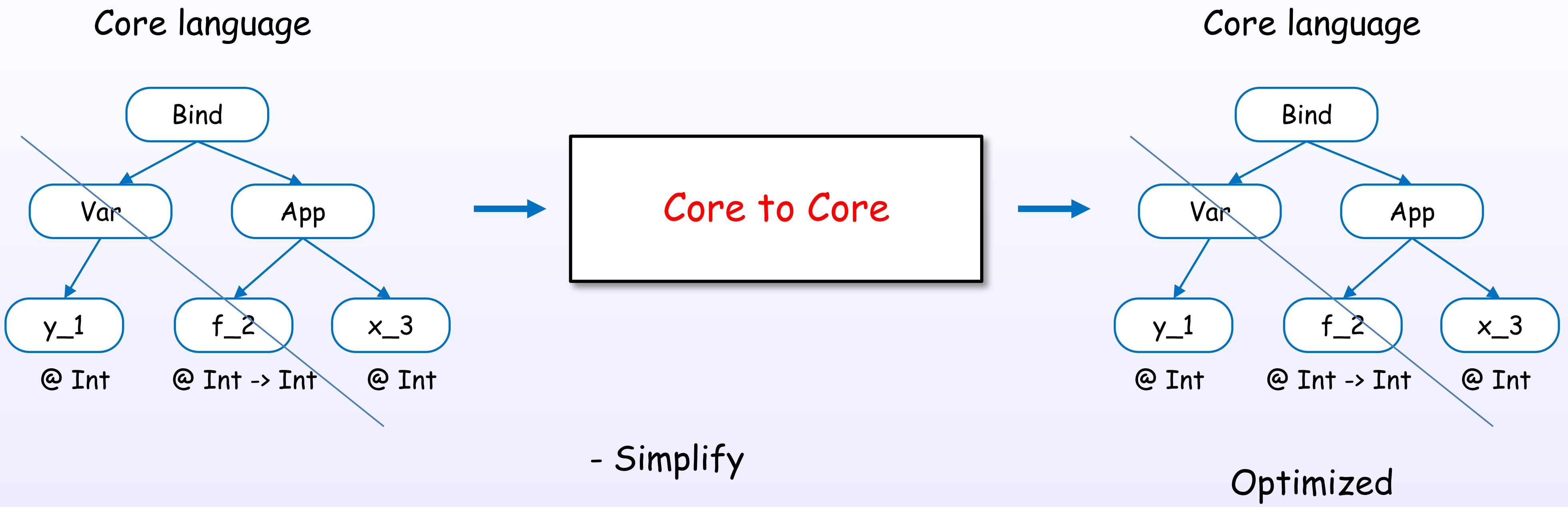


- Desugar to Core

Squeeze to typed IR

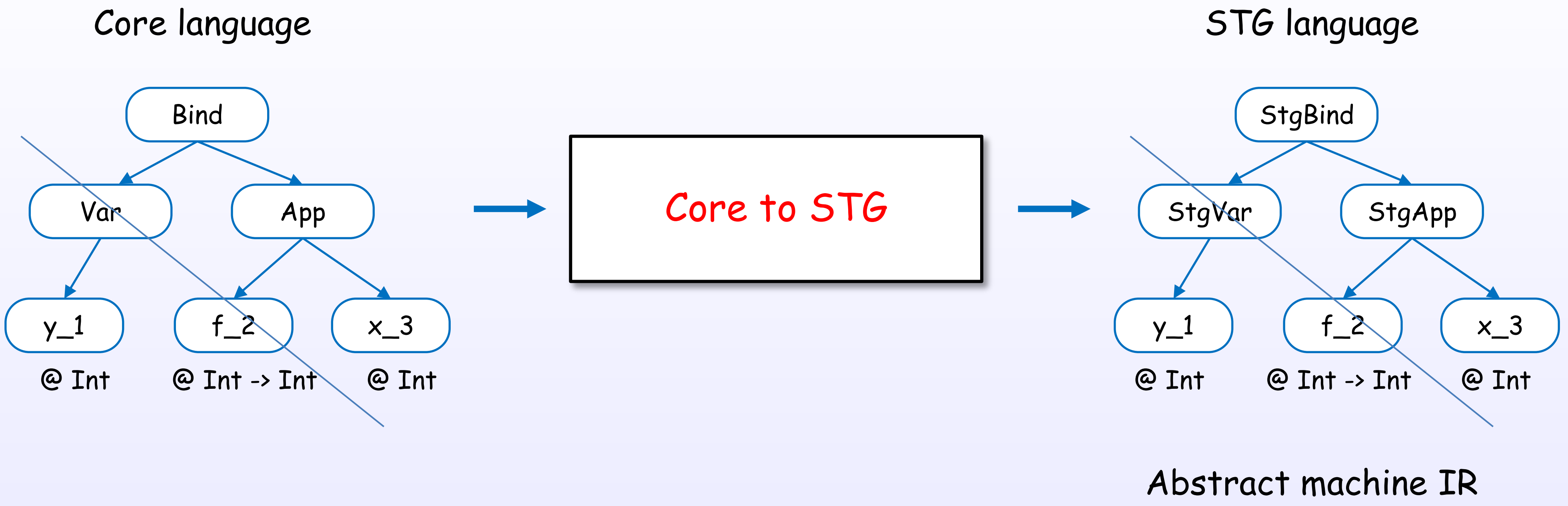
References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

Core to Core



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

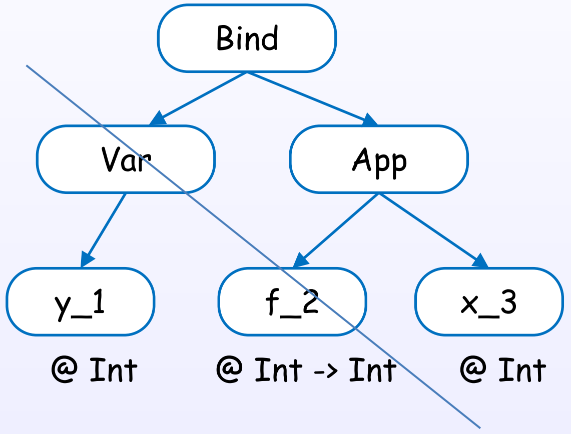
Core to Stg



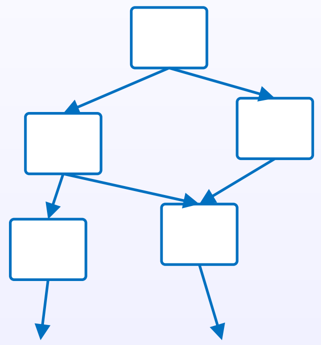
References : [1], [C3], [C4], [9], [C5], [C6], [C7], [C8], [S7], [S8], [21], [22]

STG to Cmm

STG language



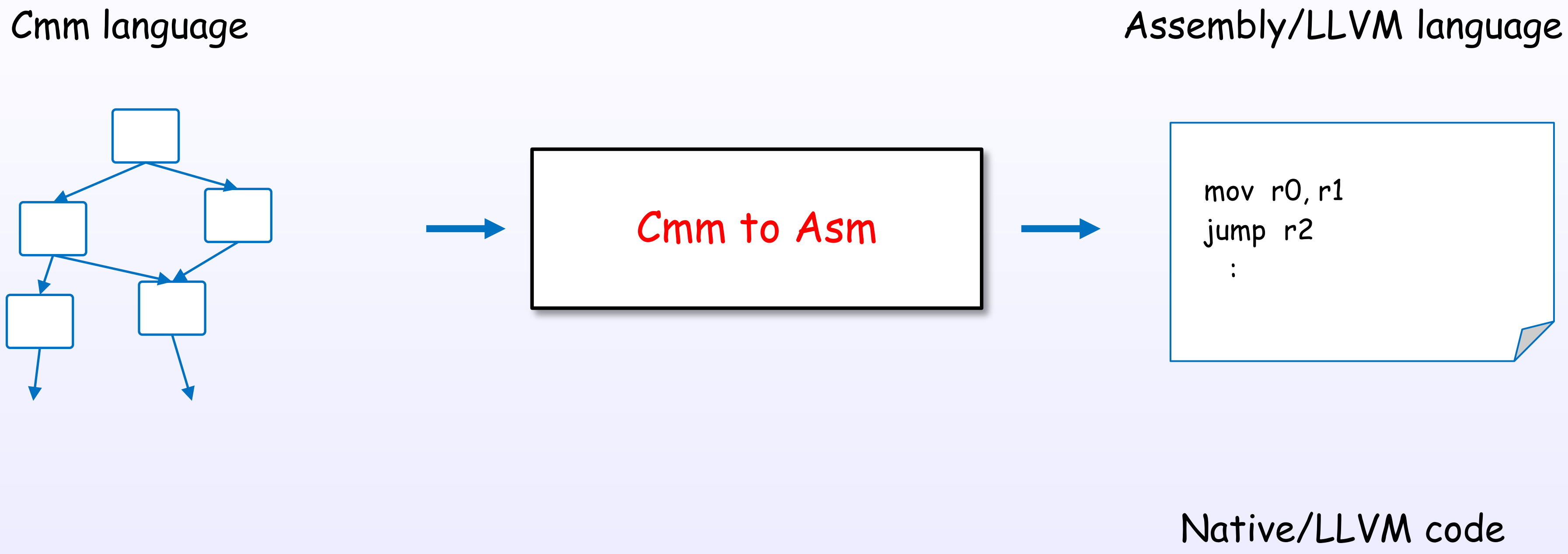
Cmm language



Portable Assembly

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

Cmm to Assembly



```
mov  r0, r1
jump  r2
  :
```

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

1. Compiler

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)                        -- 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
```

Minimul typed functional language.
Only ten data constractors 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 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
```

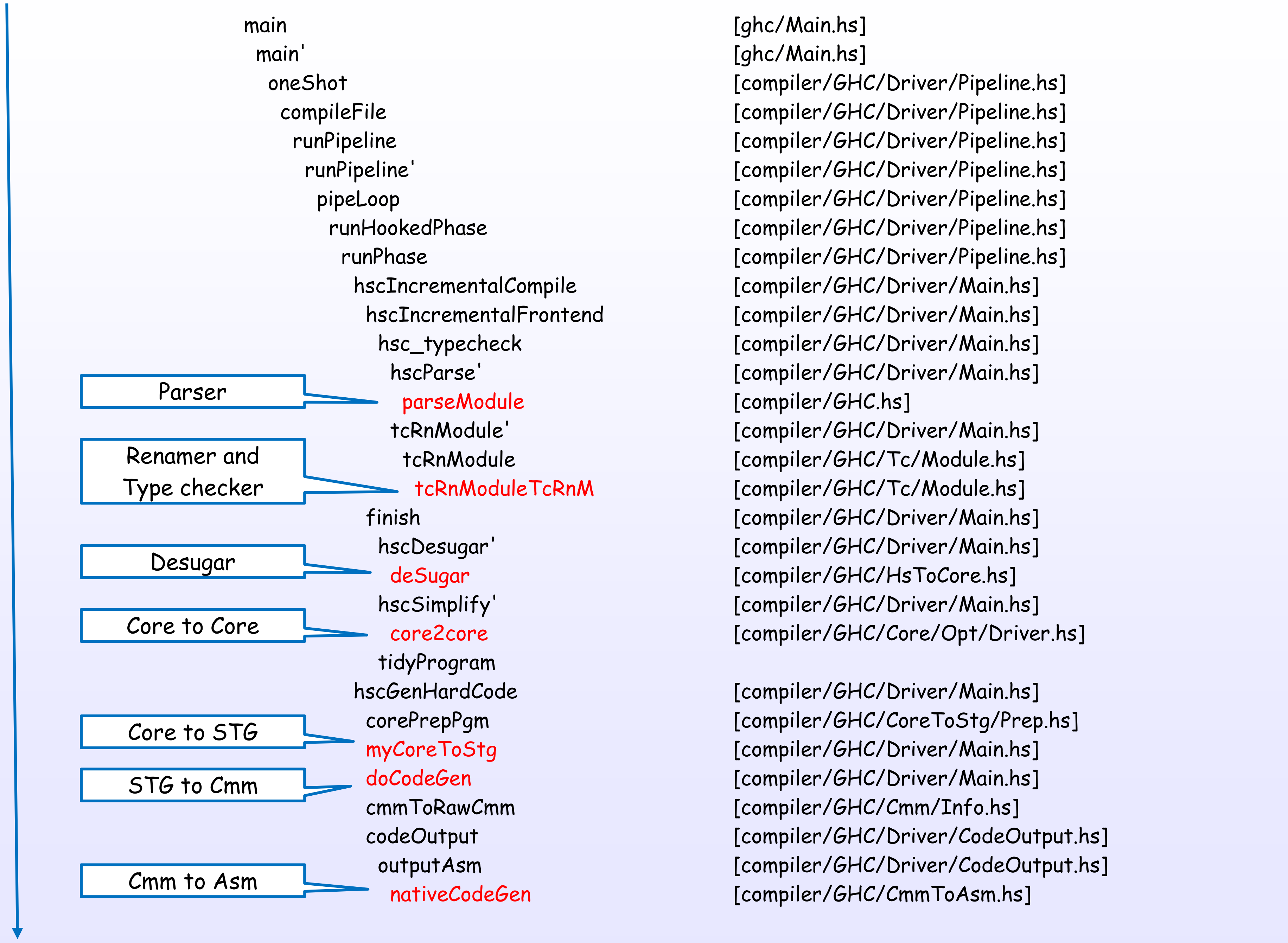
Portable assembly with imperative language.

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

1. Compiler

Call graph

Example of call graph



```
                main                                  [ghc/Main.hs]
                 main'                                [ghc/Main.hs]
                  oneShot                             [compiler/GHC/Driver/Pipeline.hs]
                   compileFile                        [compiler/GHC/Driver/Pipeline.hs]
                    runPipeline                       [compiler/GHC/Driver/Pipeline.hs]
                     runPipeline'                     [compiler/GHC/Driver/Pipeline.hs]
                      pipeLoop                        [compiler/GHC/Driver/Pipeline.hs]
                       runHookedPhase                 [compiler/GHC/Driver/Pipeline.hs]
                        runPhase                      [compiler/GHC/Driver/Pipeline.hs]
                         hscIncrementalCompile        [compiler/GHC/Driver/Main.hs]
                          hscIncrementalFrontend      [compiler/GHC/Driver/Main.hs]
                           hsc_typecheck              [compiler/GHC/Driver/Main.hs]
                            hscParse'                 [compiler/GHC/Driver/Main.hs]
Parser                       parseModule              [compiler/GHC.hs]
                            tcRnModule'               [compiler/GHC/Driver/Main.hs]
Renamer and                  tcRnModule               [compiler/GHC/Tc/Module.hs]
Type checker                  tcRnModuleTcRnM         [compiler/GHC/Tc/Module.hs]
                          finish                      [compiler/GHC/Driver/Main.hs]
                           hscDesugar'                [compiler/GHC/Driver/Main.hs]
Desugar                     deSugar                   [compiler/GHC/HsToCore.hs]
                           hscSimplify'               [compiler/GHC/Driver/Main.hs]
Core to Core                core2core                 [compiler/GHC/Core/Opt/Driver.hs]
                           tidyProgram
                         hscGenHardCode               [compiler/GHC/Driver/Main.hs]
                          corePrepPgm                 [compiler/GHC/CoreToStg/Prep.hs]
Core to STG               myCoreToStg                 [compiler/GHC/Driver/Main.hs]
STG to Cmm                doCodeGen                   [compiler/GHC/Driver/Main.hs]
                          cmmToRawCmm                 [compiler/GHC/Cmm/Info.hs]
                          codeOutput                  [compiler/GHC/Driver/CodeOutput.hs]
                           outputAsm                  [compiler/GHC/Driver/CodeOutput.hs]
Cmm to Asm                  nativeCodeGen             [compiler/GHC/CmmToAsm.hs]
```

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

References

References

aosabook
dive-into-core
cs
users guide

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