More powerful GHC Plugins

Moritz Angermann, ICFP HiW 2016, Nara

Who

- Moritz Angermann
- mobile and dekstop apps

History

- ~ 2014
  Evaluating GHC for use in iOS
  oopth
- ~ 2015
  🇩🇪 → 🇸🇬
- ~ 2016
  👫 → 👨‍👩‍👧
  Evaluating GHC again.

Plugin Interface

- Frontend Plugins

```
data FrontendPlugin = FrontendPlugin {
    frontend :: [String] -> [(String, Maybe Phase)] -> Ghc ()
  }
```

Plugin Interface

- Core/Typechecker Plugins

```
data Plugin = Plugin {
    installCoreToDos :: [CommandLineOption] -> [CoreToDo]
                     -> CoreM [CoreToDo]
  , tcPlugin         :: [CommandLineOption] -> Maybe TcPlugin
  }
```

GHC Pipeline

```
   Parse   -> Rename   -> Typecheck
-> Desugar -> Simplify
-> Core    -> STG      -> Cmm
-> Codegen -> Assembly -> Linking
```

Plugin Interface

- Core/Typechecker Plugins

```
data Plugin = Plugin {
    installCoreToDos :: [CommandLineOption] -> [CoreToDo]
                     -> CoreM [CoreToDo]
  , tcPlugin         :: [CommandLineOption] -> Maybe TcPlugin
  }
```

Plugin Interface

- Core/Typechecker Plugins + DynFlags Plugin

```
data Plugin = Plugin {
    installCoreToDos :: [CommandLineOption] -> [CoreToDo]
                     -> CoreM [CoreToDo]
  , tcPlugin         :: [CommandLineOption] -> Maybe TcPlugin
  , updateDynFlags   :: [CommandLineOption] -> DynFlags
                     -> DynFlags
  }
```

Hooks

--

```
data Hooks = Hooks { -- [...]
  , runPhaseHook      :: Maybe (PhasePlus -> FilePath -> DynFlags
                                -> CompPipeline (PhasePlus, FilePath))
  }
lookupHook :: (Hooks -> Maybe a) -> a -> DynFlags -> a
lookupHook hook def = fromMaybe def . hook . hooks
getHooked :: (Functor f, HasDynFlags f)
          => (Hooks -> Maybe a) -> a -> f a
getHooked hook def = fmap (lookupHook hook def) getDynFlags
```

Hooks

```
type Hook a = Maybe (a -> a)                                                  --

data Hooks = Hooks { -- [...]
   , runPhaseHook      :: Hook  (PhasePlus -> FilePath -> DynFlags
                                 -> CompPipeline (PhasePlus, FilePath))
   }
lookupHook :: (Hooks -> Hook a) -> a -> DynFlags -> a
lookupHook hook def = fromMaybe def . fmap ($ def) . hook . hooks
getHooked :: (Functor f, HasDynFlags f)
          => (Hooks -> Hook a) -> a -> f a
getHooked hook def = fmap (lookupHook hook def) getDynFlags
```

Hooks

```haskell
type Hook a = Maybe (a -> a)                                                   --

data Hooks = Hooks { -- [...]
   , runPhaseHook       :: Hook  (PhasePlus -> FilePath -> DynFlags
                                  -> CompPipeline (PhasePlus, FilePath))
   , codeOutputHook     :: Hook  (DynFlags -> Module -> ModLocation -> FilePath
                                  -> Stream IO RawCmmGroup () -> [UnitId] -> IO ())
   , nextPhaseHook      :: Hook  (DynFlags -> Phase -> Phase)
   , startPhaseHook     :: Hook  (String -> Phase)
   , phaseInputExtHook  :: Hook  (Phase -> String)
   }
```

Plugin example

```haskell
module Plugin where

import CodeGen (codeOutput, phaseInputExt, phaseHook)

installHooks :: [CommandLineOption] -> DynFlags -> DynFlags
installHooks _ dflags = dflags { hooks = addHooks (hooks dflags) }
  where
    addHook h = h { codeOutputHook    = Just codeOutput
                  , phaseInputExtHook = Just phaseInputExt
                  , runPhaseHook      = Just phaseHook
                  }
```

Plugin example

```haskell
module CodeGen where

codeOutput :: Hook (DynFlags -> Module -> ModLocation
                   -> FilePath -> Stream IO RawCmmGroup () -> [UnitId] -> IO ())
codeOutput super dflags mod mloc fp cmm_stream pkg_deps = genCode ...

phaseInputExt :: Hook (Phase -> String)
phaseHook :: Hook (PhasePlus -> FilePath -> DynFlags
                  -> CompPipeline (PhasePlus, FilePath))
```

Plugin example

```haskell
module CodeGen where

codeOutput :: Hook (DynFlags -> Module -> ModLocation
                 -> FilePath -> Stream IO RawCmmGroup () -> [UnitId] -> IO ())
phaseInputExt :: Hook (Phase -> String)
phaseInputExt super phase = case phase of
  (As _)     -> "g"
  _ -> super phase

phaseHook :: Hook (PhasePlus -> FilePath -> DynFlags
                  -> CompPipeline (PhasePlus, FilePath))
```

Plugin example

```haskell
module CodeGen where

codeOutput :: Hook (DynFlags -> Module -> ModLocation
                   -> FilePath -> Stream IO RawCmmGroup () -> [UnitId] -> IO ())
phaseInputExt :: Hook (Phase -> String)
phaseHook :: Hook (PhasePlus -> FilePath -> DynFlags
                  -> CompPipeline (PhasePlus, FilePath))
phaseHook super phase input_fn dflags = case phase of
  (RealPhase (As with_cpp)) -> do                             -- Assembly phase
    let next_phase = StopLn
    output_fn <- phaseOutputFilename next_phase
    ...
    return (RealPhase nextPhase, output_fn)
  _ -> super phase input_fn dflags
```

Open issues

- Reenable Core / TC plugins
- Multiple plugins
- Allow stage1 to load plugins
- ghc compiles plugin before using it
- Plugin fragility
  - accessing dynflags
  - profiled ghc + profiled plugin

Thanks!

Questions?

FIN

phabricator.haskell.org/D535