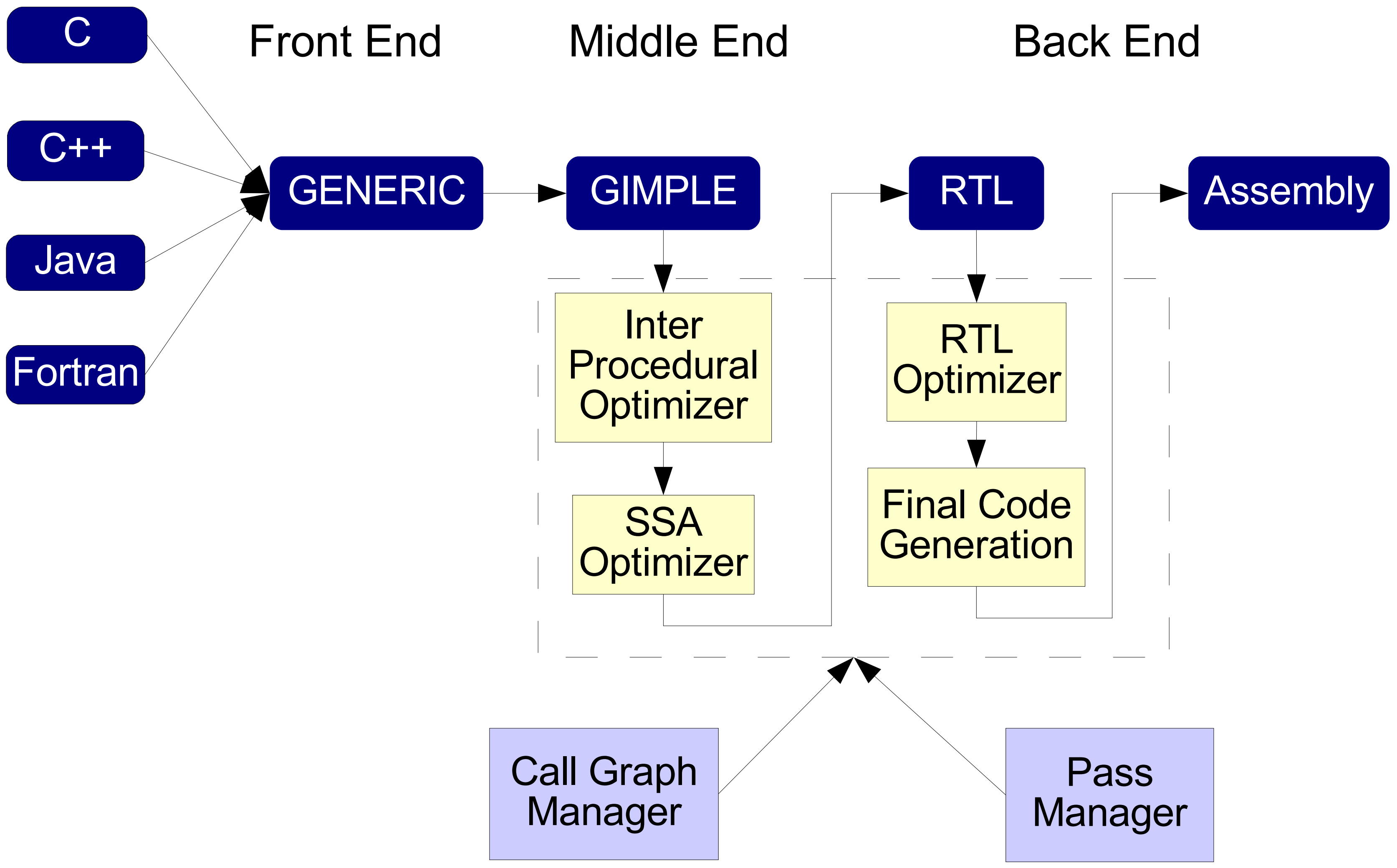
GCC Internals
Compiler Pipeline

Diego Novillo
`dnovillo@google.com`

November 2007

Compiler pipeline



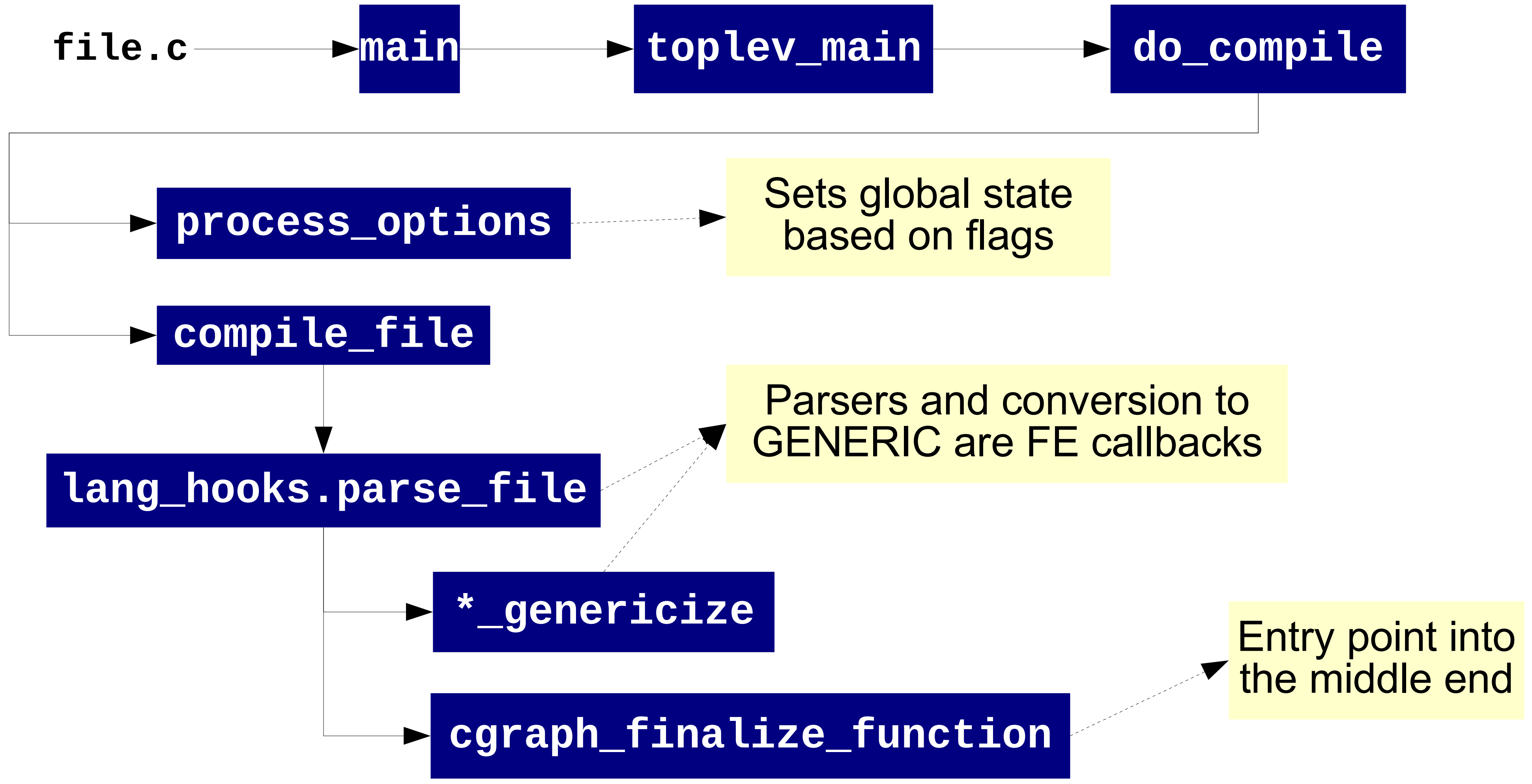
SSA Optimizers

- Operate on GIMPLE
- Around 100 passes
  - Vectorization
  - Various loop optimizations
  - Traditional scalar optimizations: CCP, DCE, DSE, FRE, PRE, VRP, SRA, jump threading, forward propagation
  - Field-sensitive, points-to alias analysis
  - Pointer checking instrumentation for C/C++
  - Interprocedural analysis and optimizations: CCP, inlining, points-to analysis, pure/const and type escape analysis

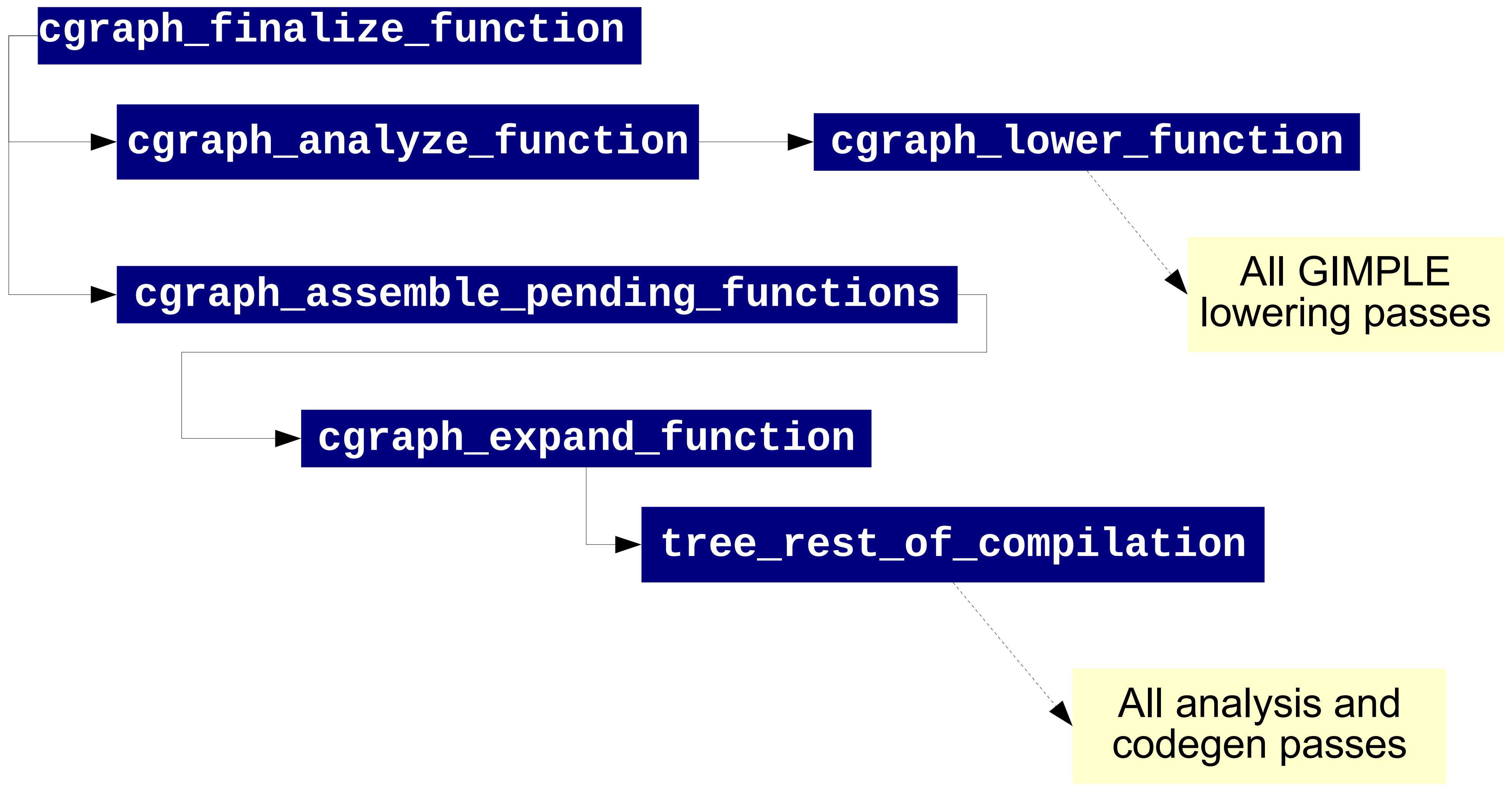
RTL Optimizers

- Around 70 passes
- Operate closer to the target
  - Register allocation
  - Scheduling
  - Software pipelining
  - Common subexpression elimination
  - Instruction recombination
  - Mode switching reduction
  - Peephole optimizations
  - Machine specific reorganization

Simplified compilation flow (O0)



Simplified compilation flow (O0)



`toplev.c:do_compile()`

- Drives compilation process
- Initializes the compiler:
  - Timers in `timevar.def`
  - Machine modes for all target data types
  - Back end data (`backend_init`): RTL, hash tables, pools, target-specific initialization, etc.
- Calls `compile_file()`
- Finishes off with call to `finalize()` to shut everything down.

toplev.c:compile_file()

- Initializes cgraph and gcov data
- Calls `lang_hooks.parse_file` (`c_common_parse_file`)
  - Parses the entire file
  - Calls `finish_function` after parsing each function body (`c_parser_declaration_or_fndef`)
  - Function bodies are registered in call graph (or emitted)
- Calls `lang_hooks.decls.final_write_globals` (`c_write_global_declarations`)
  - Emits all symbols and functions with file and external scope
  - Calls `cgraph_optimize`

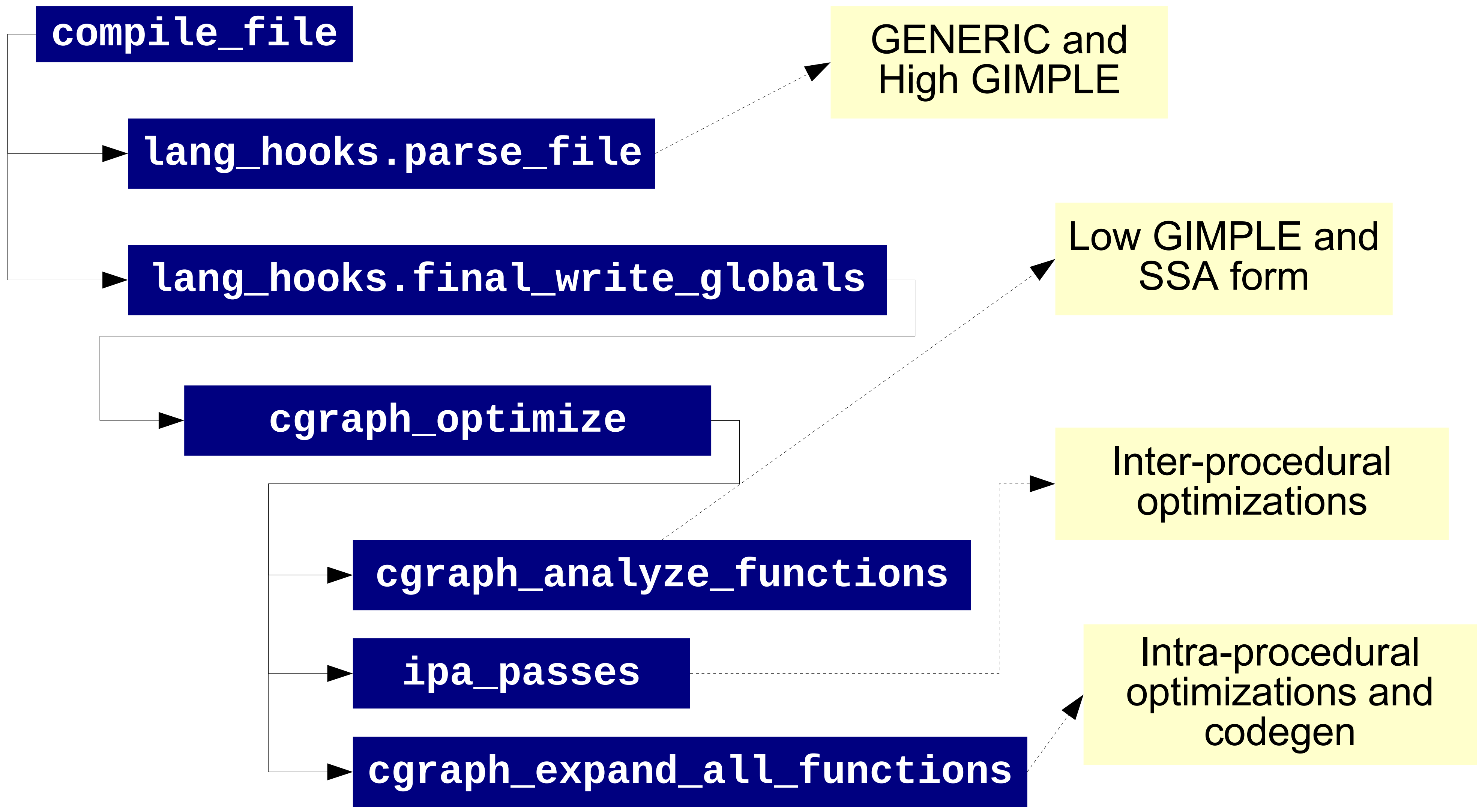
`c-decl.c:finish_function`

- Massage arguments and return value as per ABI
- Calls `c_genericize`
  - Converts to GENERIC
  - C goes straight to GIMPLE
  - C++ goes straight to GIMPLE (for now)
- Calls `cgraph_finalize_function`
  - Only on non-nested functions
  - If function is nested, it only creates a new call graph node

`cgraphunit.c:cgraph_finalize_function`

- Calls `lower_nested_functions`
- At `-O0`:
  - Calls `cgraph_analyze_function`
    - Lowers GIMPLE, EH, OpenMP, mudflap and builds CFG (`all_lowering_passes`)
    - Expands OpenMP, builds profile data (`pass_early_local_passes`)
  - Calls `cgraph_assemble_pending_functions` → `cgraph_expand_function` → `tree_rest_of_compilation`
    - Expands into RTL (`pass_expand` in `all_passes`)
- At `-O1+` decides if node is intrinsically needed and/or reachable

Simplified compilation flow (O1+)


compile_file
lang_hooks.parse_file
lang_hooks.final_write_globals
cgraph_optimize
cgraph_analyze_functions
ipa_passes
cgraph_expand_all_functions
GENERIC and High GIMPLE
Low GIMPLE and SSA form
Inter-procedural optimizations
Intra-procedural optimizations and codegen

`cgraphunit.c:cgraph_finalize_function`

- At `-O1` and up
  - Lowers GIMPLE, EH, OpenMP, mudflap and builds CFG (`all_lowering_passes`)
  - Determines if the call graph node is intrinsically needed
  - Determines if the call graph node is intrinsically reachable

`c-decl.c:c_write_global_declarations`

- Called via `lang_hooks.final_write_globals`
- The whole file has been parsed and converted to GENERIC
- Emits all symbols in the global scope, ultimately calling `decl_rest_of_compilation` for each one
- Calls `cgraph_optimize` to get into ME/BE
- Emits debug information for all surviving globals

`cgraphunit.c:cgraph_optimize`

- Main driver for inter and intra procedural optimization
- Computes reachability and lowers every function body (`cgraph_analyze_functions`)
- Performs inter-procedural optimization (`ipa_passes`)
- Decides what functions to emit (`cgraph_mark_functions_to_output`)
- Performs intra-procedural optimization and final code generation (`cgraph_expand_all_functions`)

`cgraphunit.c:cgraph_analyze_functions`

- Computes reachability for the whole call graph
- For every reachable node
  - Lowers GIMPLE, EH, OpenMP, mudflap and builds CFG (`all_lowering_passes`)
  - Creates callgraph edges at call sites
  - Expands OpenMP constructs
  - Builds SSA form
  - Early optimizations that do not require aliasing information (`pass_early_local_passes`)

To be fixed

`cgraphunit.c:cgraph_expand_all_functions`

- Sorts call graph in reverse topological order to output a function after its callees
- Calls `cgraph_expand_function` on each node
  - Performs all intra-procedural optimizations, RTL expansion and code generation via `tree_rest_of_compilation`
- Processes new functions added during optimization (`cgraph_process_new_functions`)