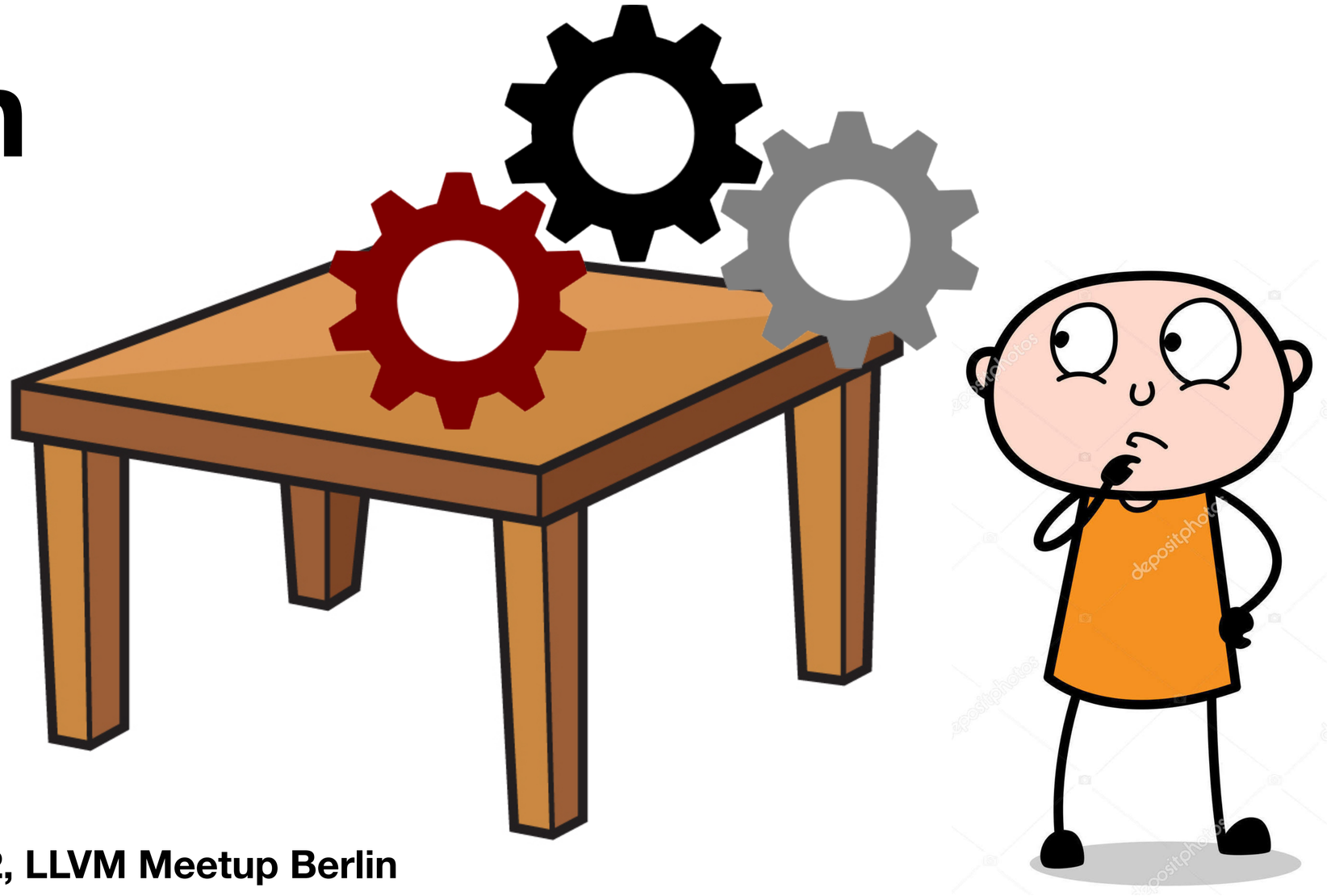
LLVM Essentials

TableGen



Stefan Gränitz, 26 Oct. 2022, LLVM Meetup Berlin

TableGen

DSL in the heart of LLVM’s target-independent code generator

Quick facts:

- Help a humans develop and maintain huge records information, i.e. instruction set definitions https://llvm.org/docs/TableGen
- Source-to-Source Transformation: target description → llvm-tblgen → C++
- People say it’s Turing complete.. 🙈
  https://www.slideshare.net/bekketmcclane/how-to-write-a-tablegen-backend

TableGen

Generate instruction tables for target-independent code generator:

- Operands
- Assembly Syntax
- ISel Rules
- …

```
// Section B.11 - Logical Instructions, p. 106
defm AND    : F3_12<"and", 0b000001, and, IntRegs, i32, simm13Op>;

def ANDNrr  : F3_1<2, 0b000101,
                   (outs IntRegs:$rd), (ins IntRegs:$rs1, IntRegs:$rs2),
                   "andn $rs1, $rs2, $rd",
                   [(set i32:$rd, (and i32:$rs1, (not i32:$rs2)))]>;
def ANDNri  : F3_2<2, 0b000101,
                   (outs IntRegs:$rd), (ins IntRegs:$rs1, simm13Op:$simm13),
                   "andn $rs1, $simm13, $rd", []>;

defm OR     : F3_12<"or", 0b000010, or, IntRegs, i32, simm13Op>;

def ORNrr   : F3_1<2, 0b000110,
                   (outs IntRegs:$rd), (ins IntRegs:$rs1, IntRegs:$rs2),
                   "orn $rs1, $rs2, $rd",
                   [(set i32:$rd, (or i32:$rs1, (not i32:$rs2)))]>;
def ORNri   : F3_2<2, 0b000110,
                   (outs IntRegs:$rd), (ins IntRegs:$rs1, simm13Op:$simm13),
                   "orn $rs1, $simm13, $rd", []>;
defm XOR    : F3_12<"xor", 0b000011, xor, IntRegs, i32, simm13Op>;

def XNORrr  : F3_1<2, 0b000111,
                   (outs IntRegs:$rd), (ins IntRegs:$rs1, IntRegs:$rs2),
                   "xnor $rs1, $rs2, $rd",
                   [(set i32:$rd, (not (xor i32:$rs1, i32:$rs2)))]>;
def XNORri  : F3_2<2, 0b000111,
                   (outs IntRegs:$rd), (ins IntRegs:$rs1, simm13Op:$simm13),
                   "xnor $rs1, $simm13, $rd", []>;
```

TableGen

Over time adopted for a variety of tasks like:

```
def fatal_too_many_errors
  : Error<"too many errors emitted, stopping now">, DefaultFatal;

def warn_stack_exhausted : Warning<
  "stack nearly exhausted; compilation time may suffer, and "
  "crashes due to stack overflow are likely">,
  InGroup<DiagGroup<"stack-exhausted">>, NoSFINAE;

def note_declared_at : Note<"declared here">;
def note_previous_definition : Note<"previous definition is here">;
def note_previous_declaration : Note<"previous declaration is here">;
def note_previous_implicit_declaration : Note<
  "previous implicit declaration is here">;
def note_previous_use : Note<"previous use is here">;
def note_duplicate_case_prev : Note<"previous case defined here">;
def note_forward_declaration : Note<"forward declaration of %0">;
def note_type_being_defined : Note<
  "definition of %0 is not complete until the closing '}'">;
/// note_matching - this is used as a continuation of a previous diagnostic,
/// e.g. to specify the '(' when we expected a ')'.
def note_matching : Note<"to match this %0">;

def note_using : Note<"using">;
def note_possibility : Note<"one possibility">;
def note_also_found : Note<"also found">;
```

TableGen

Over time adopted for a variety of tasks like:

Clang Diagnostics

clang/include/clang/Basic/DiagnosticCommonKinds.td

```
def fatal_too_many_errors
  : Error<"too many errors emitted, stopping now">, DefaultFatal;

def warn_stack_exhausted : Warning<
  "stack nearly exhausted; compilation time may suffer, and "
  "crashes due to stack overflow are likely">,
  InGroup<DiagGroup<"stack-exhausted">>, NoSFINAE;

def note_declared_at : Note<"declared here">;
def note_previous_definition : Note<"previous definition is here">;
def note_previous_declaration : Note<"previous declaration is here">;
def note_previous_implicit_declaration : Note<
  "previous implicit declaration is here">;
def note_previous_use : Note<"previous use is here">;
def note_duplicate_case_prev : Note<"previous case defined here">;
def note_forward_declaration : Note<"forward declaration of %0">;
def note_type_being_defined : Note<
  "definition of %0 is not complete until the closing '}'">;
/// note_matching - this is used as a continuation of a previous diagnostic,
/// e.g. to specify the '(' when we expected a ')'.
def note_matching : Note<"to match this %0">;

def note_using : Note<"using">;
def note_possibility : Note<"one possibility">;
def note_also_found : Note<"also found">;
```

TableGen

Over time adopted for a variety of tasks like:

```
let Command = "process attach" in {
  def process_attach_continue : Option<"continue", "c">,
    Desc<"Immediately continue the process once attached.">;
  def process_attach_plugin : Option<"plugin", "P">, Arg<"Plugin">,
    Desc<"Name of the process plugin you want to use.">;
  def process_attach_pid : Option<"pid", "p">, Group<1>, Arg<"Pid">,
    Desc<"The process ID of an existing process to attach to.">;
  def process_attach_name : Option<"name", "n">, Group<2>, Arg<"ProcessName">,
    Desc<"The name of the process to attach to.">;
  def process_attach_include_existing : Option<"include-existing", "i">,
    Group<2>, Desc<"Include existing processes when doing attach -w.">;
  def process_attach_waitfor : Option<"waitfor", "w">, Group<2>,
    Desc<"Wait for the process with <process-name> to launch.">;
}

let Command = "process continue" in {
  def process_continue_ignore_count : Option<"ignore-count", "i">, Group<1>,
    Arg<"UnsignedInteger">, Desc<"Ignore <N> crossings of the breakpoint (if it"
    " exists) for the currently selected thread.">;
  def process_continue_run_to_bkpt : Option<"continue-to-bkpt", "b">, Group<2>,
    Arg<"BreakpointIDRange">, Desc<"Specify a breakpoint to continue to, temporarily "
    "ignoring other breakpoints.  Can be specified more than once.  "
    "The continue action will be done synchronously if this option is specified.">;
}

let Command = "process detach" in {
  def process_detach_keep_stopped : Option<"keep-stopped", "s">, Group<1>,
    Arg<"Boolean">, Desc<"Whether or not the process should be kept stopped on"
    " detach (if possible).">;
}
```

TableGen

Over time adopted for a variety of tasks like:

LLDB Commands

lldb/source/Commands/Options.td

```
let Command = "process attach" in {
  def process_attach_continue : Option<"continue", "c">,
    Desc<"Immediately continue the process once attached.">;
  def process_attach_plugin : Option<"plugin", "P">, Arg<"Plugin">,
    Desc<"Name of the process plugin you want to use.">;
  def process_attach_pid : Option<"pid", "p">, Group<1>, Arg<"Pid">,
    Desc<"The process ID of an existing process to attach to.">;
  def process_attach_name : Option<"name", "n">, Group<2>, Arg<"ProcessName">,
    Desc<"The name of the process to attach to.">;
  def process_attach_include_existing : Option<"include-existing", "i">,
    Group<2>, Desc<"Include existing processes when doing attach -w.">;
  def process_attach_waitfor : Option<"waitfor", "w">, Group<2>,
    Desc<"Wait for the process with <process-name> to launch.">;
}

let Command = "process continue" in {
  def process_continue_ignore_count : Option<"ignore-count", "i">, Group<1>,
    Arg<"UnsignedInteger">, Desc<"Ignore <N> crossings of the breakpoint (if it"
    " exists) for the currently selected thread.">;
  def process_continue_run_to_bkpt : Option<"continue-to-bkpt", "b">, Group<2>,
    Arg<"BreakpointIDRange">, Desc<"Specify a breakpoint to continue to, temporarily "
    "ignoring other breakpoints.  Can be specified more than once.  "
    "The continue action will be done synchronously if this option is specified.">;
}

let Command = "process detach" in {
  def process_detach_keep_stopped : Option<"keep-stopped", "s">, Group<1>,
    Arg<"Boolean">, Desc<"Whether or not the process should be kept stopped on"
    " detach (if possible).">;
}
```

TableGen

Over time adopted for a variety of tasks like:

```
def version : Flag<["--"], "version">,
  HelpText<"Display the version of this program">;
def : Flag<["-"], "v">, Alias<version>, HelpText<"Alias for --version">;

def adjust_vma_EQ : Joined<["--"], "adjust-vma=">,
  MetaVarName<"offset">,
  HelpText<"Increase the displayed address by the specified offset">;

def all_headers : Flag<["--"], "all-headers">,
  HelpText<"Display all available header information, "
           "relocation entries and the symbol table">;
def : Flag<["-"], "x">, Alias<all_headers>, HelpText<"Alias for --all-headers">;

def arch_name_EQ : Joined<["--"], "arch-name=">,
  HelpText<"Target arch to disassemble for, "
           "see --version for available targets">;
def archive_headers : Flag<["--"], "archive-headers">,
  HelpText<"Display archive header information">;

def : Flag<["-"], "a">, Alias<archive_headers>,
  HelpText<"Alias for --archive-headers">;

def demangle : Flag<["--"], "demangle">, HelpText<"Demangle symbol names">;
def : Flag<["-"], "C">, Alias<demangle>, HelpText<"Alias for --demangle">;

def disassemble : Flag<["--"], "disassemble">,
  HelpText<"Disassemble all executable sections found in the input files">;
def : Flag<["-"], "d">, Alias<disassemble>, HelpText<"Alias for --disassemble">;
```

TableGen

Over time adopted for a variety of tasks like:

Option Parsing

llvm/tools/llvm-objdump/ObjdumpOpts.td

```
def version : Flag<["--"], "version">,
  HelpText<"Display the version of this program">;
def : Flag<["-"], "v">, Alias<version>, HelpText<"Alias for --version">;

def adjust_vma_EQ : Joined<["--"], "adjust-vma=">,
  MetaVarName<"offset">,
  HelpText<"Increase the displayed address by the specified offset">;

def all_headers : Flag<["--"], "all-headers">,
  HelpText<"Display all available header information, "
           "relocation entries and the symbol table">;
def : Flag<["-"], "x">, Alias<all_headers>, HelpText<"Alias for --all-headers">;

def arch_name_EQ : Joined<["--"], "arch-name=">,
  HelpText<"Target arch to disassemble for, "
           "see --version for available targets">;
def archive_headers : Flag<["--"], "archive-headers">,
  HelpText<"Display archive header information">;

def : Flag<["-"], "a">, Alias<archive_headers>,
  HelpText<"Alias for --archive-headers">;

def demangle : Flag<["--"], "demangle">, HelpText<"Demangle symbol names">;
def : Flag<["-"], "C">, Alias<demangle>, HelpText<"Alias for --demangle">;

def disassemble : Flag<["--"], "disassemble">,
  HelpText<"Disassemble all executable sections found in the input files">;
def : Flag<["-"], "d">, Alias<disassemble>, HelpText<"Alias for --disassemble">;
```

TableGen

Over time adopted for a variety of tasks like:

```
/// Function must be in a unwind table.
def UWTable : IntAttr<"uwtable", [FnAttr]>;

/// Minimum/Maximum vscale value for function.
def VScaleRange : IntAttr<"vscale_range", [FnAttr]>;

/// Function always comes back to callsite.
def WillReturn : EnumAttr<"willreturn", [FnAttr]>;

/// Function only writes to memory.
def WriteOnly : EnumAttr<"writeonly", [FnAttr, ParamAttr]>;

/// Zero extended before/after call.
def ZExt : EnumAttr<"zeroext", [ParamAttr, RetAttr]>;

/// Function is required to make Forward Progress.
def MustProgress : EnumAttr<"mustprogress", [FnAttr]>;

/// Function is a presplit coroutine.
def PresplitCoroutine : EnumAttr<"presplitcoroutine", [FnAttr]>;
```

TableGen

Over time adopted for a variety of tasks like:

IR Attributes

llvm/include/llvm/IR/Attributes.td

```
/// Function must be in a unwind table.
def UWTable : IntAttr<"uwtable", [FnAttr]>;

/// Minimum/Maximum vscale value for function.
def VScaleRange : IntAttr<"vscale_range", [FnAttr]>;

/// Function always comes back to callsite.
def WillReturn : EnumAttr<"willreturn", [FnAttr]>;

/// Function only writes to memory.
def WriteOnly : EnumAttr<"writeonly", [FnAttr, ParamAttr]>;

/// Zero extended before/after call.
def ZExt : EnumAttr<"zeroext", [ParamAttr, RetAttr]>;

/// Function is required to make Forward Progress.
def MustProgress : EnumAttr<"mustprogress", [FnAttr]>;

/// Function is a presplit coroutine.
def PresplitCoroutine : EnumAttr<"presplitcoroutine", [FnAttr]>;
```

TableGen

Source-to-Source Transformation: TD → llvm-tblgen → C++

clang/include/clang/Basic/DiagnosticCommonKinds.td

```
def fatal_too_many_errors
  : Error<"too many errors emitted, stopping now">, DefaultFatal;
```

build/tools/clang/include/clang/Basic/DiagnosticCommonKinds.inc

```
DIAG(fatal_too_many_errors, CLASS_ERROR, (unsigned)diag::Severity::Fatal,
"too many errors emitted, stopping now", 0, SFINAE_SubstitutionFailure,
false, true, true, false, 0)
```

TableGen

Source-to-Source Transformation: TD → llvm-tblgen → C++

Find build command:

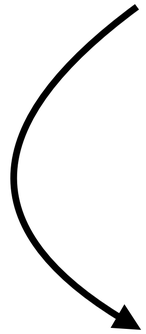
```
> cd llvm-project/build
> ninja -t commands clang | grep DiagnosticCommonKinds.inc
llvm-project/build/bin/clang-tblgen -gen-clang-diags-defs -clang-component=Common
    -I llvm-project/clang/include/clang/Basic -I llvm-project/clang/include
    -I llvm-project/build/tools/clang/include -I llvm-project/build/include
    -I llvm-project/llvm/include llvm-project/clang/include/clang/Basic/Diagnostic.td
    -o tools/clang/include/clang/Basic/DiagnosticCommonKinds.inc --write-if-changed
```

TableGen

C++ output is #included like a header

build/tools/clang/include/clang/Basic/DiagnosticCommonKinds.inc

```
DIAG(fatal_too_many_errors, CLASS_ERROR, (unsigned)diag::Severity::Fatal,
"too many errors emitted, stopping now", 0, SFINAE_SubstitutionFailure,
false, true, true, false, 0)
```



clang/include/clang/Basic/DiagnosticIDs.h

```
    // Get typedefs for common diagnostics.
    enum {
#define DIAG(ENUM, FLAGS, DEFAULT_MAPPING, DESC, GROUP, SFINAE, CATEGORY,      \
             NOWERROR, SHOWINSYSHEADER, SHOWINSYSMACRO, DEFFERABLE)            \
  ENUM,
#define COMMONSTART
#include "clang/Basic/DiagnosticCommonKinds.inc"
      NUM_BUILTIN_COMMON_DIAGNOSTICS
#undef DIAG
    };
```

TableGen

Inspect preprocessed output

Find and modify build command:

```
> cd llvm-project/build
> find tools/clang/lib | grep DiagnosticIDs.cpp
tools/clang/lib/Basic/CMakeFiles/obj.clangBasic.dir/DiagnosticIDs.cpp.o
> ninja -t commands
              tools/clang/lib/Basic/CMakeFiles/obj.clangBasic.dir/DiagnosticIDs.cpp.o | tail -1
/usr/bin/clang++ ...
    -o tools/clang/lib/Basic/CMakeFiles/obj.clangBasic.dir/DiagnosticIDs.cpp.o
    -c llvm-project/clang/lib/Basic/DiagnosticIDs.cpp
> /usr/bin/clang++ ...
    -o tools/clang/lib/Basic/CMakeFiles/obj.clangBasic.dir/DiagnosticIDs.ii
    -E llvm-project/clang/lib/Basic/DiagnosticIDs.cpp
```

TableGen

Preprocessed output:

```
enum {
  __COMMONSTART = DIAG_START_COMMON,
  err_arcmt_nsinvocation_ownership,
  err_asm_invalid_type,
  ...
  fatal_too_many_errors,
  note_also_found,
  ...
  warn_stack_exhausted,
  warn_target_unsupported_branch_protection_attribute,
  warn_unknown_attribute_ignored,
  NUM_BUILTIN_COMMON_DIAGNOSTICS
};
```

clang/include/clang/Basic/DiagnosticCommonKinds.td

```
def fatal_too_many_errors
  : Error<"too many errors emitted, stopping now">, DefaultFatal;

def warn_stack_exhausted : Warning<
  "stack nearly exhausted; compilation time may suffer, and "
  "crashes due to stack overflow are likely">,
  InGroup<DiagGroup<"stack-exhausted">>, NoSFINAE;

def note_declared_at : Note<"declared here">;
def note_previous_definition : Note<"previous definition is here">;
def note_previous_declaration : Note<"previous declaration is here">;
def note_previous_implicit_declaration : Note<
  "previous implicit declaration is here">;
def note_previous_use : Note<"previous use is here">;
def note_duplicate_case_prev : Note<"previous case defined here">;
def note_forward_declaration : Note<"forward declaration of %0">;
def note_type_being_defined : Note<
  "definition of %0 is not complete until the closing '}'">;
/// note_matching - this is used as a continuation of a previous diagnostic,
/// e.g. to specify the '(' when we expected a ')'.
def note_matching : Note<"to match this %0">;

def note_using : Note<"using">;
def note_possibility : Note<"one possibility">;
def note_also_found : Note<"also found">;
```

LLVM Backends

Main target definition file includes further TD files

llvm/lib/Target/RISCV/RISCV.td

```
//===-----------------------------------------------------------------===//
// Named operands for CSR instructions.
//===-----------------------------------------------------------------===//

include "RISCVSystemOperands.td"

//===-----------------------------------------------------------------===//
// Registers, calling conventions, instruction descriptions.
//===-----------------------------------------------------------------===//

include "RISCVSchedule.td"
include "RISCVRegisterInfo.td"
include "RISCVCallingConv.td"
include "RISCVInstrInfo.td"
include "RISCVRegisterBanks.td"
include "RISCVSchedRocket.td"
include "RISCVSchedSiFive7.td"
```

LLVM Backends

CMake takes main target definition file and defines TableGen actions

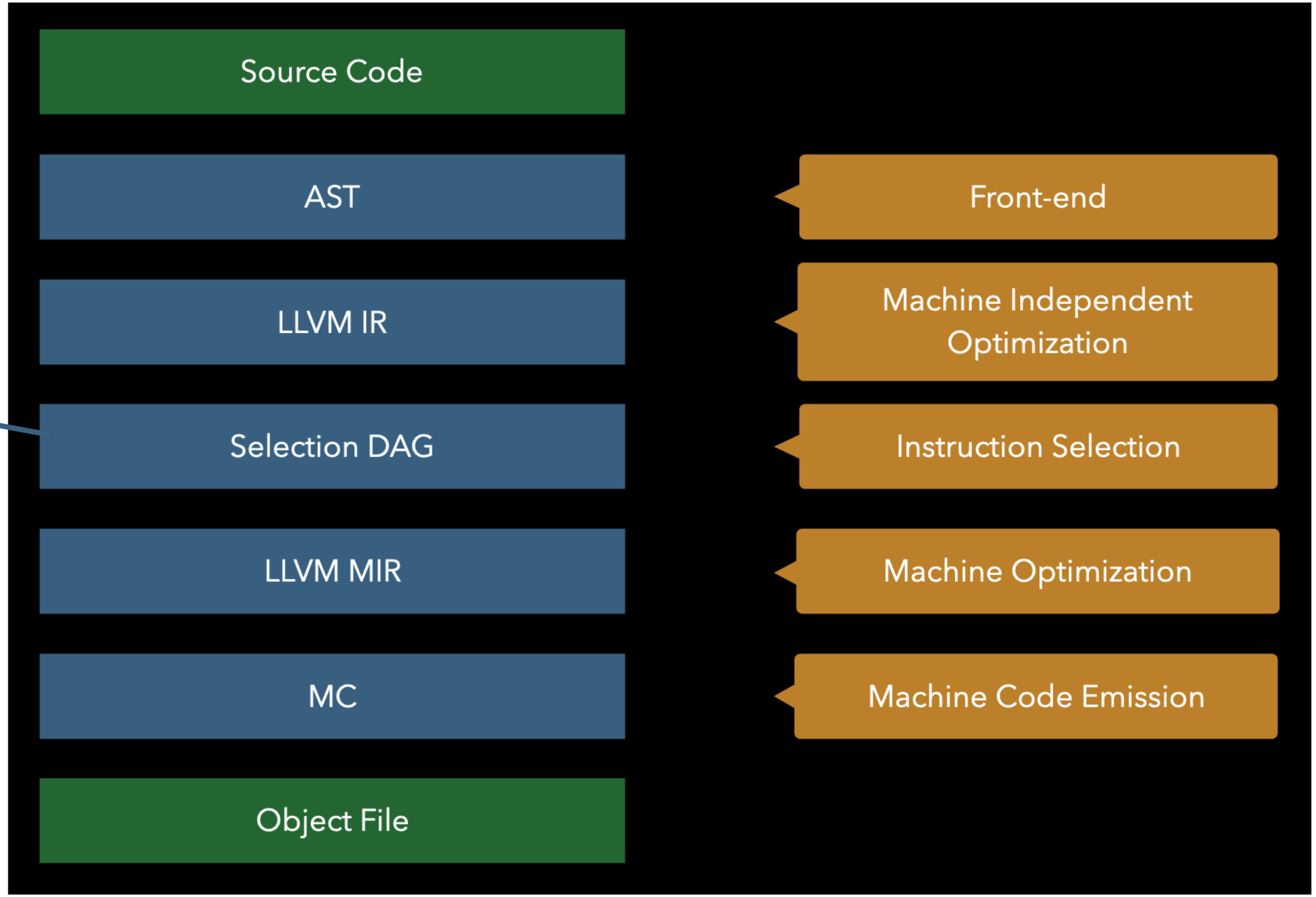
llvm/lib/Target/RISCV/CMakeLists.txt

```
set(LLVM_TARGET_DEFINITIONS RISCV.td)
tablegen(LLVM RISCVGenAsmMatcher.inc -gen-asm-matcher)
tablegen(LLVM RISCVGenAsmWriter.inc -gen-asm-writer)
tablegen(LLVM RISCVGenCompressInstEmitter.inc -gen-compress-inst-emitter)
tablegen(LLVM RISCVGenDAGISel.inc -gen-dag-isel)
tablegen(LLVM RISCVGenDisassemblerTables.inc -gen-disassembler)
tablegen(LLVM RISCVGenGlobalISel.inc -gen-global-isel)
tablegen(LLVM RISCVGenInstrInfo.inc -gen-instr-info)
tablegen(LLVM RISCVGenMCCodeEmitter.inc -gen-emitter)
tablegen(LLVM RISCVGenMCPseudoLowering.inc -gen-pseudo-lowering)
tablegen(LLVM RISCVGenRegisterBank.inc -gen-register-bank)
tablegen(LLVM RISCVGenRegisterInfo.inc -gen-register-info)
tablegen(LLVM RISCVGenSearchableTables.inc -gen-searchable-tables)
tablegen(LLVM RISCVGenSubtargetInfo.inc -gen-subtarget)
```

Program Representations

ISel: DAG to DAG

- Selects machine instructions (MIR) for SelectionDAG operations using target-independent pattern-matching algorithm (ISel)
- Part of transformation IR → MIR



https://llvm.org/devmtg/2017-10/slides/Braun-Welcome to the Back End.pdf

Glimpse into Target-independent Instruction Selection

SelectionDAGISel::CodeGenAndEmitDAG() → SelectionDAGISel::DoInstructionSelection() → RISCVDAGToDAGISel::Select() → SelectCode() → SelectionDAGISel::SelectCodeCommon()

build/lib/Target/RISCV/RISCVGenDAGISel.inc

```
/*1256101*/ /*SwitchOpcode*/ 70|128,4/*582*/, TARGET_VAL(RISCVISD::BR_CC),// ->1256687
/*1256105*/  OPC_RecordNode, // #0 = 'riscv_brcc' chained node
/*1256106*/  OPC_RecordChild1, // #1 = $rs1
/*1256107*/  OPC_Scope, 31|128,2/*287*/, /*->1256397*/ // 2 children in Scope
/*1256110*/   OPC_CheckChild1Type, MVT::i64,
/*1256112*/   OPC_Scope, 21|128,1/*149*/, /*->1256264*/ // 2 children in Scope
/*1256115*/    OPC_CheckChild2Integer, 0,
/*1256117*/    OPC_MoveChild3,
/*1256118*/    OPC_Scope, 23, /*->1256143*/ // 6 children in Scope
/*1256120*/     OPC_CheckCondCode, ISD::SETEQ,
/*1256122*/     OPC_MoveParent,
/*1256123*/     OPC_RecordChild4, // #2 = $imm12
/*1256124*/     OPC_MoveChild4,
/*1256125*/     OPC_CheckOpcode, TARGET_VAL(ISD::BasicBlock),
/*1256128*/     OPC_MoveParent,
/*1256129*/     OPC_CheckPatternPredicate, 12, // (MF->getSubtarget().checkFeatures("+64bit"))
/*1256131*/     OPC_EmitMergeInputChains1_0,
/*1256132*/     OPC_EmitRegister, MVT::i64, RISCV::X0,
/*1256135*/     OPC_MorphNodeTo0, TARGET_VAL(RISCV::BEQ), 0|OPFL_Chain,
                    3/*#Ops*/, 1, 3, 2,
   // Src: (riscv_brcc GPR:{ *:[i64] }:$rs1, 0:{ *:[i64] }, SETEQ:{ *:[Other] }, (bb:{ *:[Other] }):$imm12) - Complexity = 8
   // Dst: (BEQ GPR:{ *:[i64] }:$rs1, X0:{ *:[i64] }, simm13_lsb0:{ *:[Other] }:$imm12)
```

TableGen

Is it worth it?

- Extra language
- Extra build infrastructure
- No good tool support

…it depends!

llvm/lib/Target/Hexagon/MCTargetDesc/HexagonMCCodeEmitter.cpp

```
#define _ fixup_Invalid
#define P(x) Hexagon::fixup_Hexagon##x
static const std::map<unsigned, std::vector<unsigned>> ExtFixups = {
  { MCSymbolRefExpr::VK_DTPREL,
    { _,                 _,             _,                  _,
      _,                 _,             P(_DTPREL_16_X),    P(_DTPREL_11_X),
      P(_DTPREL_11_X),   P(_9_X),       _,                  P(_DTPREL_11_X),
      P(_DTPREL_16_X),   _,             _,                  _,
      P(_DTPREL_16_X),   _,             _,                  _,
      _,                 _,             _,                  _,
      _,                 _,             _,                  _,
      _,                 _,             _,                  _,
      P(_DTPREL_32_6_X) }},
  { MCSymbolRefExpr::VK_GOT,
    { _,                 _,             _,                  _,
      _,                 _,             P(_GOT_11_X),       _ /* [1] */,
      _ /* [1] */,       P(_9_X),       _,                  P(_GOT_11_X),
      P(_GOT_16_X),      _,             _,                  _,
      P(_GOT_16_X),      _,             _,                  _,
      _,                 _,             _,                  _,
      _,                 _,             _,                  _,
      _,                 _,             _,                  _,
      P(_GOT_32_6_X)     }},
  { MCSymbolRefExpr::VK_GOTREL,
    { _,                 _,             _,                  _,
      _,                 _,             P(_GOTREL_11_X),    P(_GOTREL_11_X),
      P(_GOTREL_11_X),   P(_9_X),       _,                  P(_GOTREL_11_X),
      P(_GOTREL_16_X),   _,             _,                  _,
      P(_GOTREL_16_X),   _,             _,                  _,
      _,                 _,             _,                  _,
      _,                 _,             _,                  _,
      _,                 _,             _,                  _,
      P(_GOTREL_32_6_X) }},
  { MCSymbolRefExpr::VK_TPREL,
    { _,                 _,             _,                  _,
      _,                 _,             P(_TPREL_16_X),     P(_TPREL_11_X),
      P(_TPREL_11_X),    P(_9_X),       _,                  P(_TPREL_11_X),
```