LLVM No.2 Instruction Selection

https://github.com/xiaoyu1998

Resolution:2560×1440(2K)
https://github.com/llvm/llvm-project/tree/llvmorg-16.0.6

Entries

runOnMachineFunction
llvm/lib/CodeGen/SelectionDAG/SelectionDAGiSei.cpp

```
SelectAllBasicBlocks(Fn);
```

SelectAllBasicBlocks
llvm/lib/CodeGen/SelectionDAG/SelectionDAGiSei.cpp

```
LowerArguments(Fn);
for (const BasicBlock *LLVMBB : RPOT) {
    BasicBlock::const_iterator const Begin =LLVMBB->getFirstNonPHI()->getIterator();
    BasicBlock::const_iterator const End = LLVMBB->end();
    BasicBlock::const_iterator BI = End;
    for (; BI != Begin; --BI) {
        SelectBasicBlock(Inst->getIterator(), BI, HadTailCall);
```

SelectBasicBlock
llvm/lib/CodeGen/SelectionDAG/SelectionDAGiSei.cpp

```
for (BasicBlock::const_iterator I = Begin; I != End && !SDB->HasTailCall; ++I) {
    SDB->visit(*I);
CodeGenAndEmitDAG();
```

CodeGenAndEmitDAG
llvm/lib/CodeGen/SelectionDAG/SelectionDAGiSei.cpp

```
CurDAG->Combine(BeforeLegalizeTypes, AA, OptLevel);
Changed = CurDAG->LegalizeTypes();
CurDAG->Combine(AfterLegalizeTypes, AA, OptLevel);
Changed = CurDAG->LegalizeVectors();

CurDAG->LegalizeTypes();
CurDAG->Combine(AfterLegalizeVectorOps, AA, OptLevel);
CurDAG->Legalize();
CurDAG->Combine(AfterLegalizeDAG, AA, OptLevel);

DoInstructionSelection();
Scheduler->Run(CurDAG, FuncInfo->MBB);
```

RISCVSeiDAGToDAG

DoInstructionSelection
llvm/lib/CodeGen/SelectionDAG/SelectionDAGiSEI.cpp

```
while (ISelPosition != CurDAG->allnodes_begin()) {
    Select(Node);
```

Select
lvm/lib/Target/RISCV/RISCISeiDAGToDAG.cpp

```
switch (Opcode) {
    case ISD::Constant: {

SelectCode(Node);
```

generated by tablegen

DAGCombiner

Combine
llvm/lib/CodeGen/SelectionDAG/DAGCombiner.cpp

```
DAGCombiner(*this, AA, OptLevel).Run(Level);
```

Run
llvm/lib/CodeGen/SelectionDAG/DAGCombiner.cpp

```
while (SDNode *N = getNextWorklistEntry()) {
    SDValue RV = combine(N);
```

combine
llvm/lib/CodeGen/SelectionDAG/DAGCombiner.cpp

```
if (!DisableGenericCombines)
    RV = visit(N);

if (!RV.getNode()) {
    if (N->getOpcode() >= ISD::BUILTIN_OP_END ||
    TLI.hasTargetDAGCombine((ISD::NodeType)N->getOpcode())) {
        TargetLowering::DAGCombinerInfo DagCombineInfo(DAG, Level, false, this);
        RV = TLI.PerformDAGCombine(N, DagCombineInfo);
```

PerformDAGCombine
llvm/lib/Target/RISCV/RISCVSeiLowering.cpp

```
case ISD::SELECT:
    return performSELECTCombine(N, DAG, DCI, Subtarget);
```

SelectionDAGBuilder

LowerArguments
llvm/lib/CodeGen/SelectionDAG/SelectionDAGBuilder.cpp

```
SDValue NewRoot = TLI->LowerFormalArguments(
    DAG.getRoot(), F.getCallingConv(), F.isVarArg(), Ins, dl, DAG, InVals);
```

LowerFormalArguments
llvm/lib/Target/RISCV/RISCVSeiLowering.cpp

```
RVFI->setVarArgsFrameIndex(FI);
```

visit
llvm/lib/CodeGen/SelectionDAG/SelectionDAGBuilder.cpp

```
visit(I.getOpcode(), I);
```

visit
llvm/lib/CodeGen/SelectionDAG/SelectionDAGBuilder.cpp

```
switch (Opcode) {
    case Instruction::OPCODE: visit##OPCODE((const CLASS&)I); break;
```

visitSDiv
llvm/lib/CodeGen/SelectionDAG/SelectionDAGBuilder.cpp

```
setValue(&I, DAG.getNode(ISD::SDIV, getCurSDLoc(), Op1.getValueType(), Op1,Op2, Flags));
```

visitRet
llvm/lib/CodeGen/SelectionDAG/SelectionDAGBuilder.cpp

```
CallingConv::ID CallConv =
    DAG.getMachineFunction().getFunction().getCallingConv();
Chain = DAG.getTargetLoweringInfo().LowerReturn(
    Chain, CallConv, isVarArg, Outs, OutVals, getCurSDLoc(), DAG);
```

LowerReturn
llvm/lib/Target/RISCV/RISCVSeiLowering.cpp

```
return DAG.getNode(RetOpc, DL, MVT::Other, RetOps);
```

target classes handle the Instruction Selection(and lowering, combine, legalizeTypes, LegalizeVectors, Legalize, schedule), if the llvm can't handle.