指令选择

指令选择的实现方式

- 基于 SelectionDAG 的指令选择
- 快速指令选择
- 全局指令选择

Select()函数功能

- 基于 SelectionDAG 的指令选择通过节点模式匹配完成DAG到DAG的转换，将SelectionDAG节点转换成表示目标指令的节点。

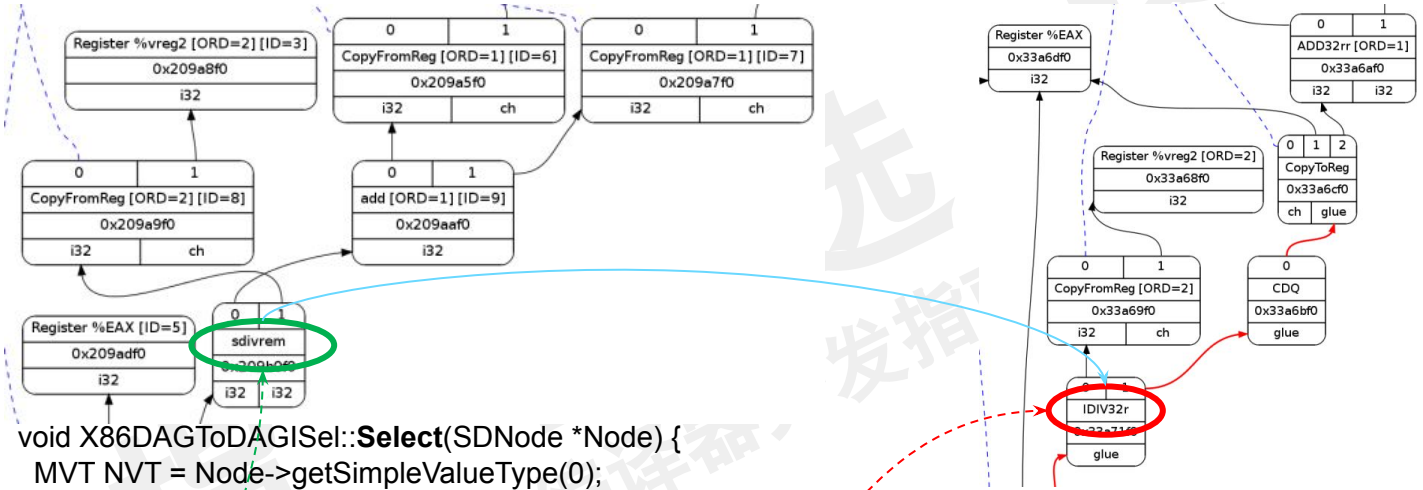
- 基于 SelectionDAG 的指令选择过程的主要逻辑在目标后端的SelectionDAGISel子类成员函数Select()中实现。

AMDGPUISelDAGToDAG.cpp

```
void AMDGPUDAGToDAGISel::Select(SDNode *N) {
  unsigned int Opc = N->getOpcode();
  ...
  switch (Opc) {
  ...
  case ISD::ADDC:
  case ISD::ADDE:
  case ISD::SUBC:
  case ISD::SUBE: {
    if (N->getValueType(0) != MVT::i64)
      break;

    SelectADD_SUB_I64(N);
    return;
  }
  ...
  SelectCode(N);
}
```

X86后端的Select()函数功能




```
void X86DAGToDAGISel::Select(SDNode *Node) {
  MVT NVT = Node->getSimpleValueType(0);
  unsigned Opcode = Node->getOpcode();
  switch (Opcode) {
  ...
  case ISD::SDIVREM:
  case ISD::UDIVREM: {
  ...
     case MVT::i32: ROpc = X86::IDIV32r; MOpc = X86::IDIV32m; break;
  ...
  SelectCode(Node);
}
```

Select()函数调用

```
void SelectionDAGISel::CodeGenAndEmitDAG() {
...
// Run the DAG combiner in pre-legalize mode.
CurDAG->Combine(BeforeLegalizeTypes, AA, OptLevel);
CurDAG->LegalizeTypes();

// Run the DAG combiner in post-type-legalize mode.
CurDAG->Combine(AfterLegalizeTypes, AA, OptLevel);
CurDAG->LegalizeVectors();

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

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

DoInstructionSelection();
...
ScheduleDAGSDNodes *Scheduler = CreateScheduler();
...
}
```

```
void SelectionDAGISel::DoInstructionSelection() {
...
  PreprocessISelDAG();
...
    while (ISelPosition != CurDAG->allnodes_begin()) {
      SDNode *Node = &*--ISelPosition;
      ...
      Select(Node);
    }

    CurDAG->setRoot(Dummy.getValue());
  }

  PostprocessISelDAG();
}
```

SelectCode()函数

- 其他指令通过 TableGen 生成的匹配表(MatcherTable)和SelectCode()函数，由 LLVM 默认指令选择过程完成ISD和<target>ISD到机器指令节点的映射。

AMDGPUGenDAGISel.inc

```
void DAGISEL_CLASS_COLONCOLON SelectCode(SDNode *N)
{
  #define TARGET_VAL(X) X & 255, unsigned(X) >> 8
  static const unsigned char MatcherTable[] = {
/*     0*/ OPC_SwitchOpcode /*212 cases */, 70|128,30|128,19/*315206*/, TARGET_VAL(ISD::ADD),// ->315212
  …
/*315212*/ /*SwitchOpcode*/ 100|128,92/*11876*/, TARGET_VAL(ISD::LOAD),// ->327092
…
/*451638*/ 0, // EndSwitchOpcode
    0
  }; // Total Array size is 451640 bytes
  …
  SelectCodeCommon(N, MatcherTable, sizeof(MatcherTable));
}
```

SelectCodeCommon()函数

```
void SelectionDAGISel::SelectCodeCommon(SDNode *NodeToMatch,
                                        const unsigned char *MatcherTable,
                                        unsigned TableSize) {
  ...
  while (true) {
    BuiltinOpcodes Opcode = (BuiltinOpcodes)MatcherTable[MatcherIndex++];
    switch (Opcode) {
    case OPC_Scope: {
      ...
      continue;
    }
    case OPC_RecordNode: {
      ...
      continue;
    }

    case OPC_RecordChild0: case OPC_RecordChild1:
    case OPC_RecordChild2: case OPC_RecordChild3:
    case OPC_RecordChild4: case OPC_RecordChild5:
    case OPC_RecordChild6: case OPC_RecordChild7: {
      ...
      continue;
    }
    ...
  }
}
```

指令选择举例

```
llc -march=amdgcn not.ll -show-mc-encoding -o - -debug-only=isel

define amdgpu_kernel void @scalar_not_i64(i64 addrspace(1)* %out, i64
%a) {
  %result = xor i64 %a, -1
  store i64 %result, i64 addrspace(1)* %out
  ret void
}

ISEL: Starting selection on root node: t25: i64 = xor t42, Constant:i64<-1>
ISEL: Starting pattern match
  Initial Opcode index to 400089
  Match failed at index 400093
  Continuing at 400224
  Match failed at index 400227
  Continuing at 400325
  Match failed at index 400327
  Continuing at 400416
  Match failed at index 400420
  Continuing at 400900
  Match failed at index 400903
  Continuing at 401374
  TypeSwitch[i64] from 401387 to 401400
  Morphed node: t25: i64,i1 = S_NOT_B64 t42
ISEL: Match complete!
```

```
/*400085*/ /*SwitchOpcode*/ 40|128,12/*1576*/, TARGET_VAL(ISD::XOR),// ->401665
/*400089*/  OPC_Scope, 4|128,1/*132*/, /*->400224*/ // 8 children in Scope
/*400092*/   OPC_MoveChild0,
/*400093*/   OPC_SwitchOpcode /*3 cases */, 40, TARGET_VAL(ISD::XOR),// ->400137
...
/*400224*/  /*Scope*/ 100, /*->400325*/
/*400225*/   OPC_RecordChild0, // #0 = $z
/*400226*/   OPC_MoveChild1,
/*400227*/   OPC_CheckOpcode, TARGET_VAL(ISD::AND),
...

...
/*401374*/  /*Scope*/ 34, /*->401409*/
/*401375*/   OPC_RecordChild0, // #0 = $src0
/*401376*/   OPC_CheckChild1Integer,
127|128,127|128,127|128,127|128,127|128,127|128,127|128,127|128,127|128,127|128,1/*184467
44073709551615*/,
/*401387*/   OPC_SwitchType /*2 cases */, 8, MVT::i32,// ->401398
/*401390*/    OPC_MorphNodeTo2, TARGET_VAL(AMDGPU::S_NOT_B32), 0,
                MVT::i32, MVT::i1, 1/*#Ops*/, 0,
              // Src: (xor:{ *:[i32] } i32:{ *:[i32] }:$src0, -1:{ *:[i32] }) - Complexity = 8
              // Dst: (S_NOT_B32:{ *:[i32] }:{ *:[i1] } i32:{ *:[i32] }:$src0)
/*401400*/    OPC_MorphNodeTo2, TARGET_VAL(AMDGPU::S_NOT_B64), 0,
                MVT::i64, MVT::i1, 1/*#Ops*/, 0,
              // Src: (xor:{ *:[i64] } i64:{ *:[i64] }:$src0, -1:{ *:[i64] }) - Complexity = 8
              // Dst: (S_NOT_B64:{ *:[i64] }:{ *:[i1] } i64:{ *:[i64] }:$src0)
/*401408*/   0, // EndSwitchType
```