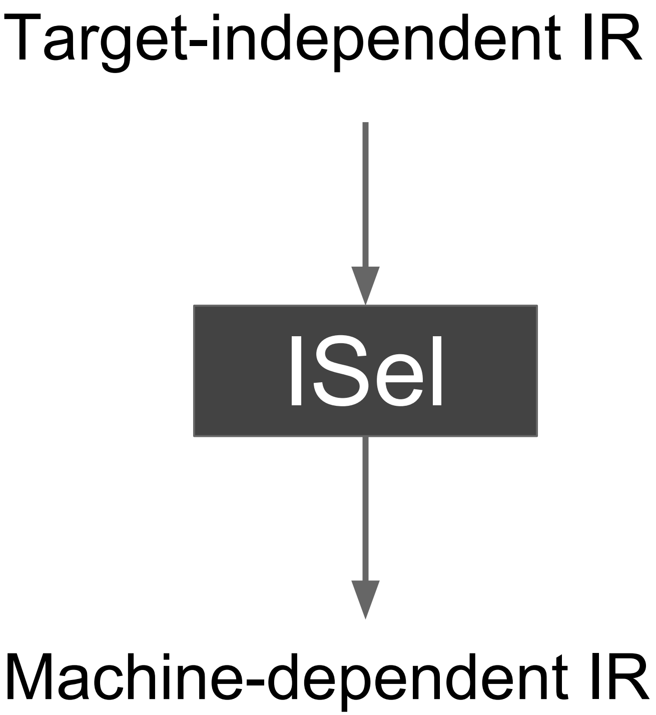
GlobalISel

LLVM’s Latest Instruction Selection Framework

Diana Picuş

Instruction Selection

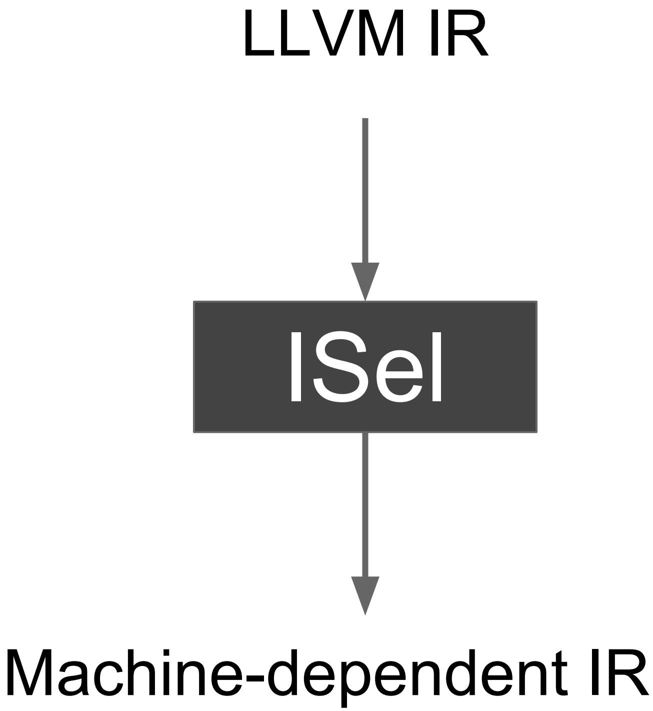


Target-independent IR

ISel

Machine-dependent IR

Instruction Selection



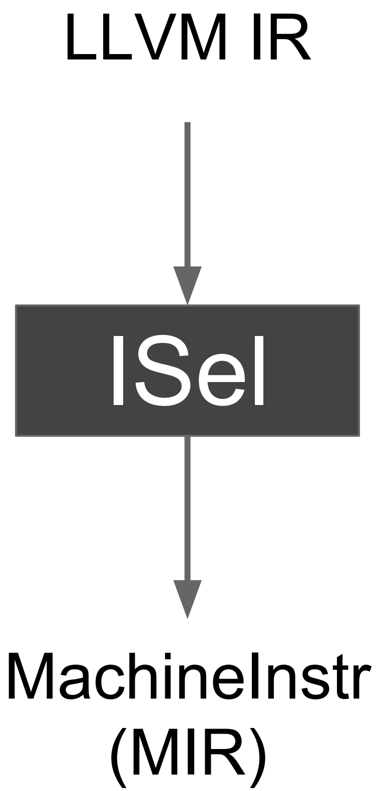
LLVM IR

ISel

Machine-dependent IR

```
define i32 @add(i32 %a, i32 %b) {
entry:
  %add = add nsw i32 %b, %a
  ret i32 %add
}
```

Instruction Selection



LLVM IR

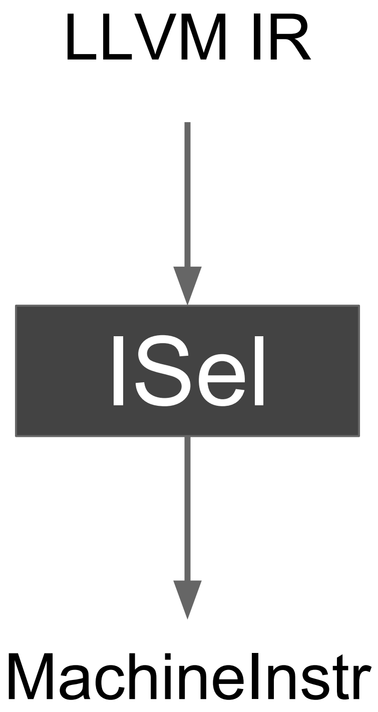
ISel

MachineInstr
(MIR)

```
define i32 @add(i32 %a, i32 %b) {
entry:
  %add = add nsw i32 %b, %a
  ret i32 %add
}
```

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:             |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

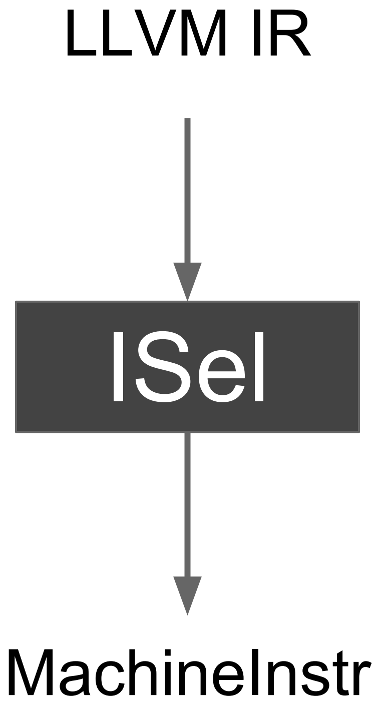
Instruction Selection



- Static Single Assignment

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:             |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

Instruction Selection



LLVM IR

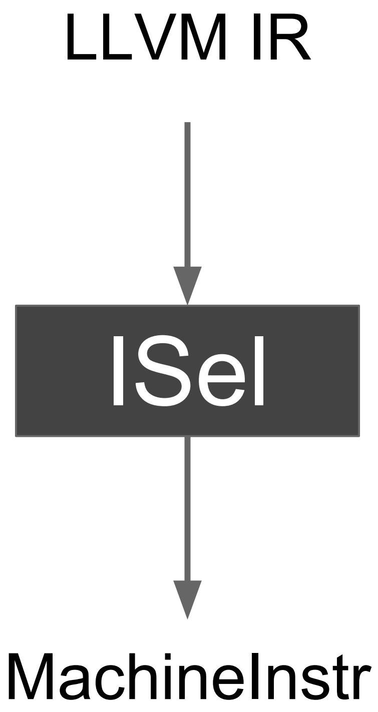
ISel

MachineInstr

- Static Single Assignment
- Virtual registers

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:             |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

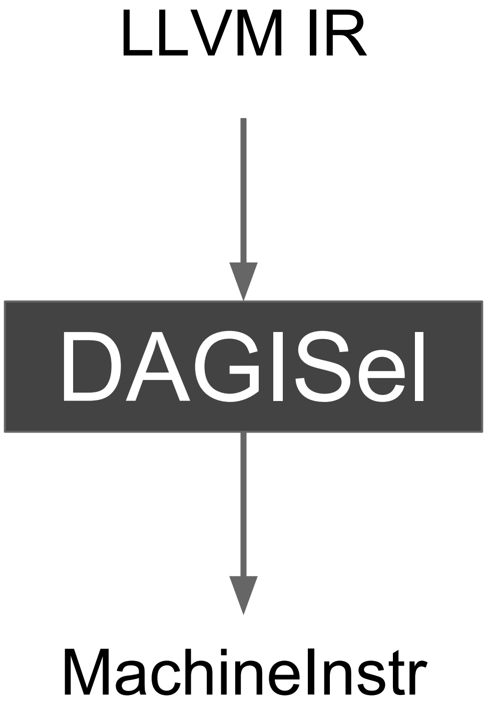
Instruction Selection



- Static Single Assignment
- Virtual registers
- Pseudoinstructions

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:             |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

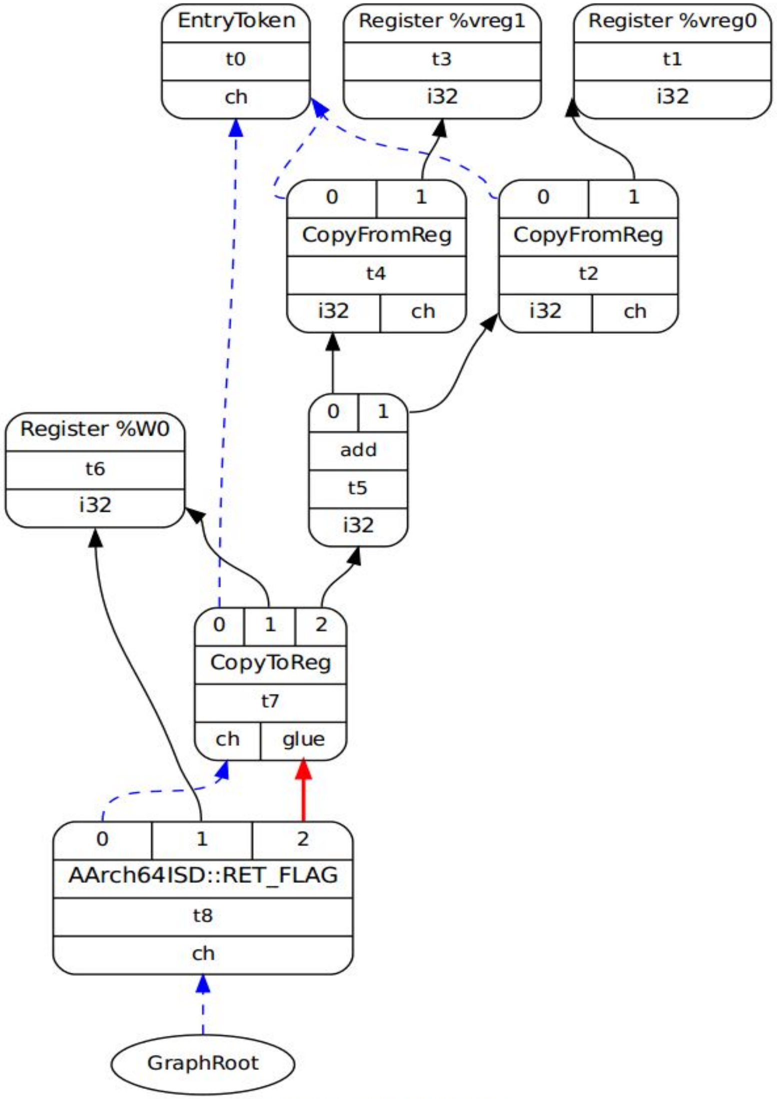
Currently in LLVM



LLVM IR

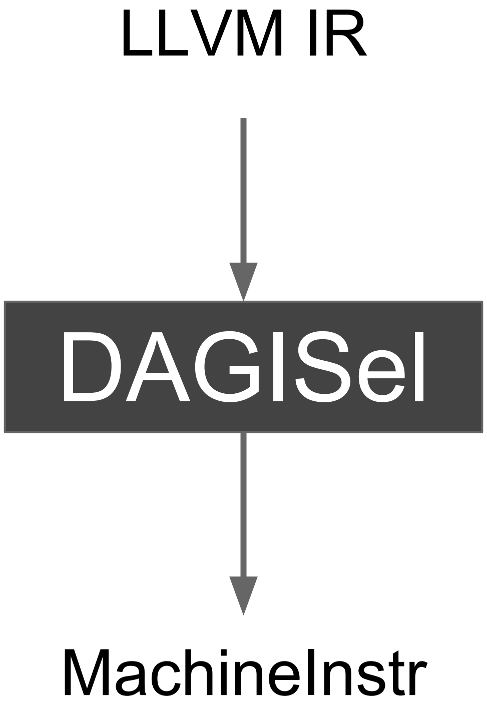
DAGISel

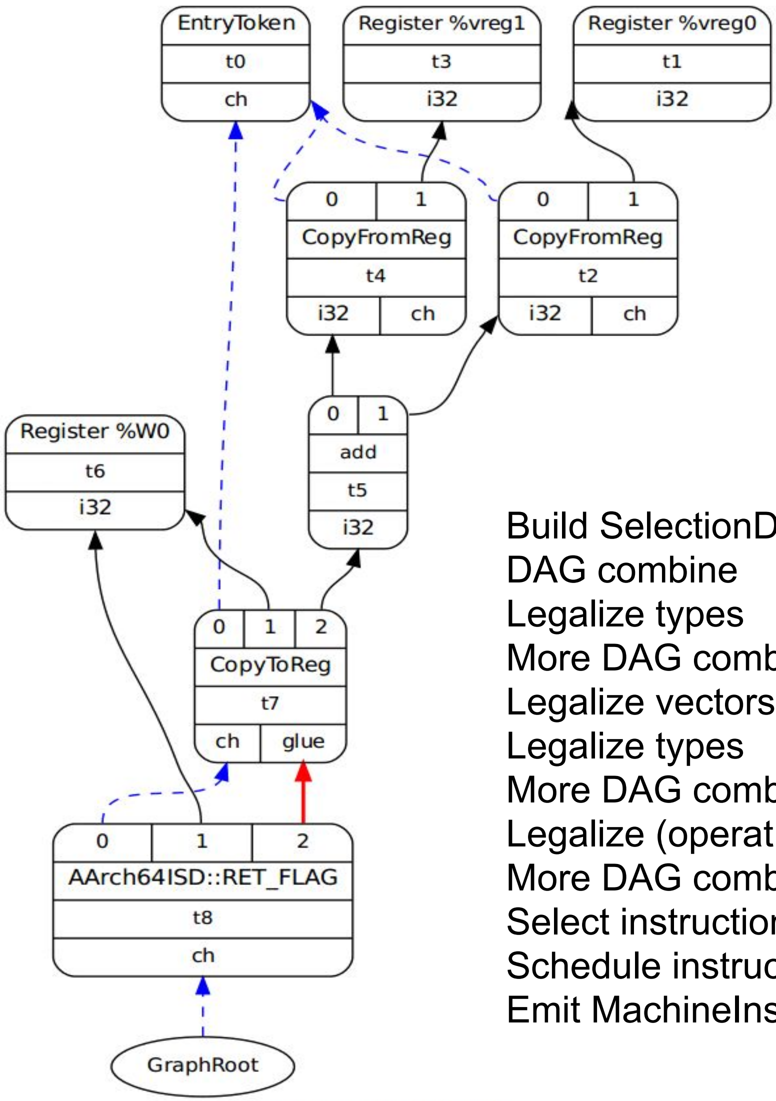
MachineInstr



isel input for add:entry

Currently in LLVM

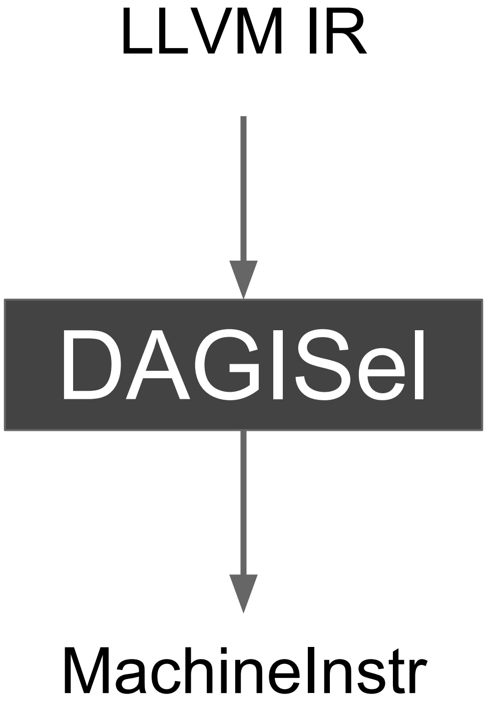




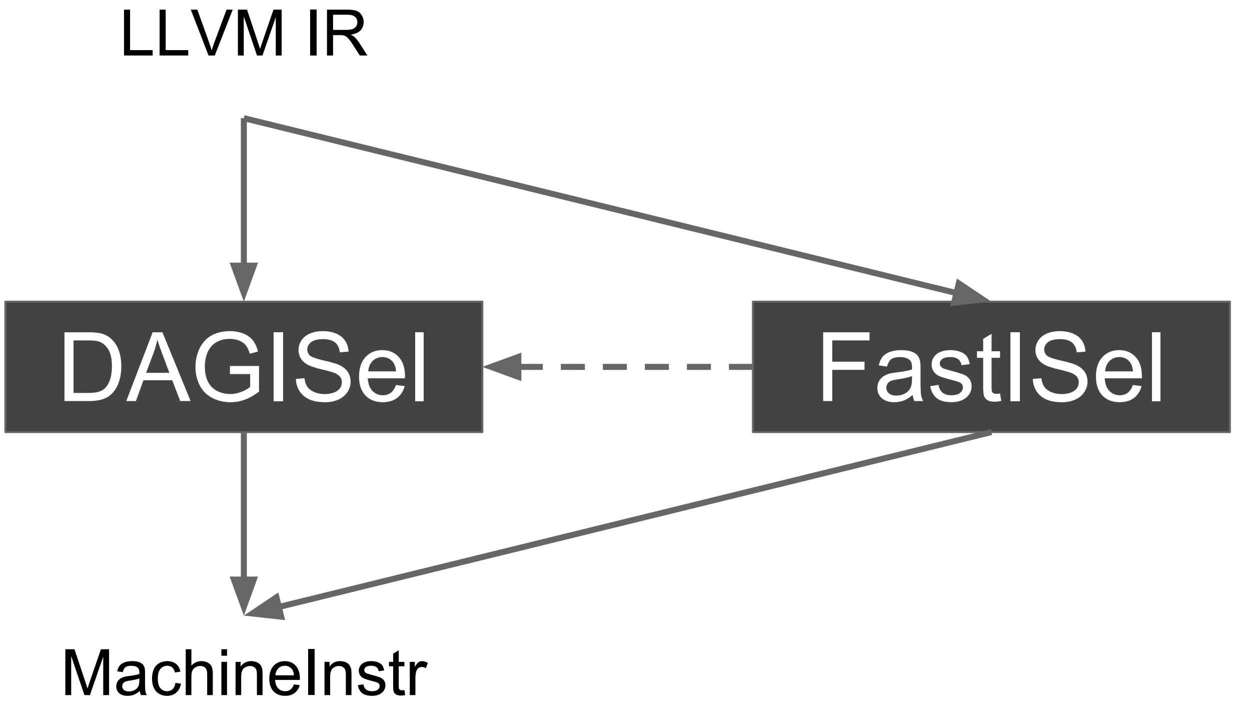
isel input for add:entry

Build SelectionDAG
DAG combine
Legalize types
More DAG combine
Legalize vectors
Legalize types
More DAG combine
Legalize (operations)
More DAG combine
Select instructions
Schedule instructions
Emit MachineInstr

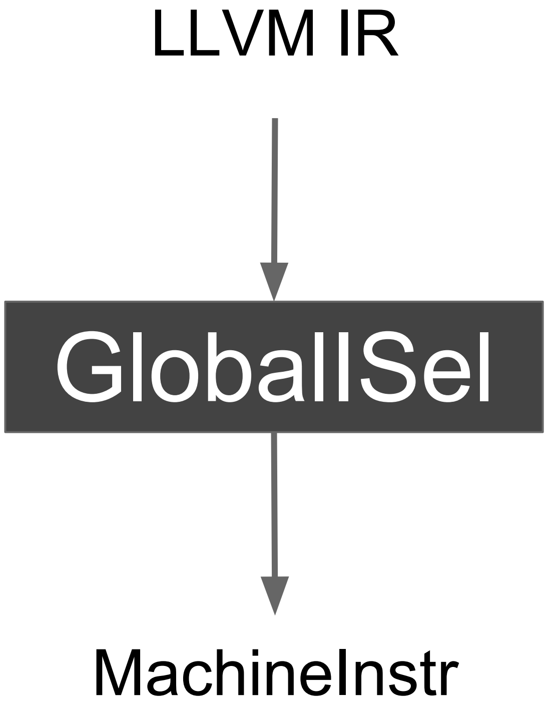
Currently in LLVM



Currently in LLVM



In the Making

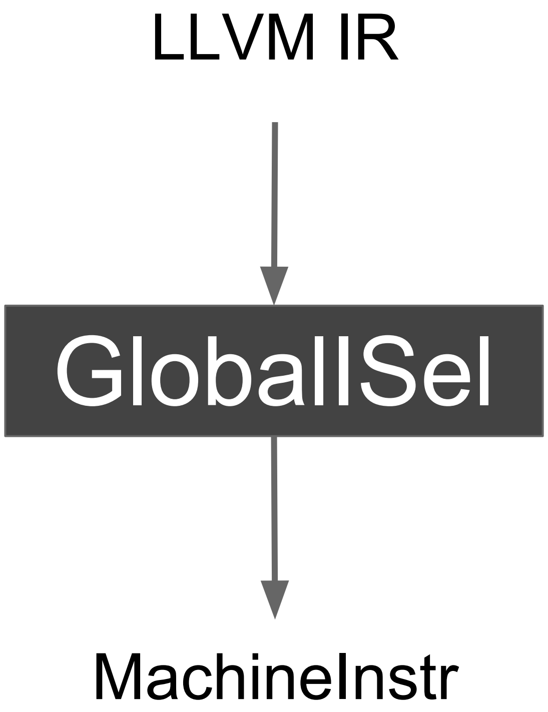


LLVM IR

GlobalISel

MachineInstr

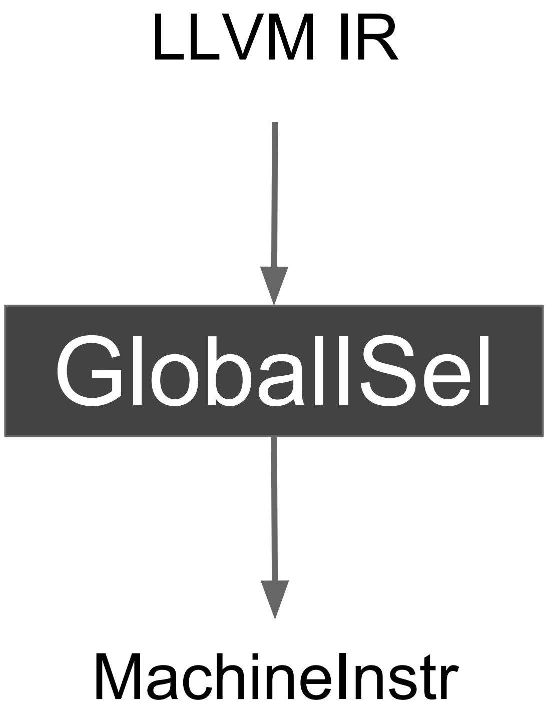
In the Making



MachineInstr

- + Register banks
- + Generic instructions

In the Making


LLVM IR

GlobalISel

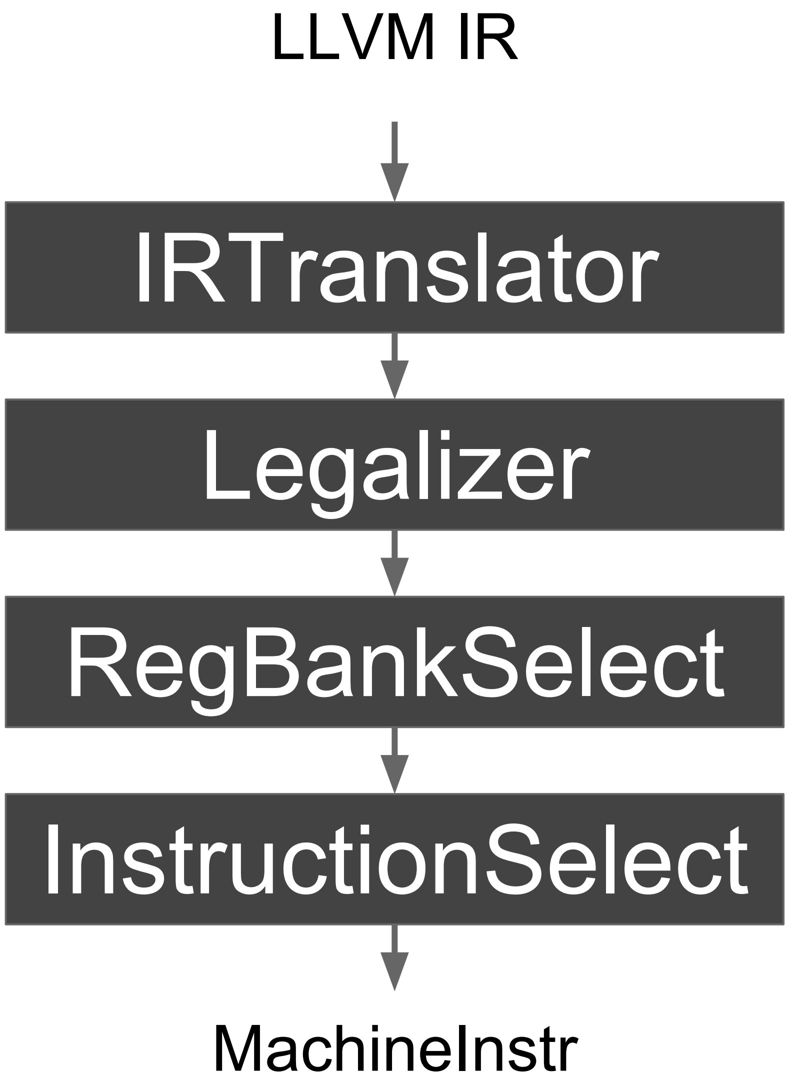
MachineInstr

MachineInstr

II
V

Machine Passes

The GlobalISel Pipeline



LLVM IR

IRTranslator

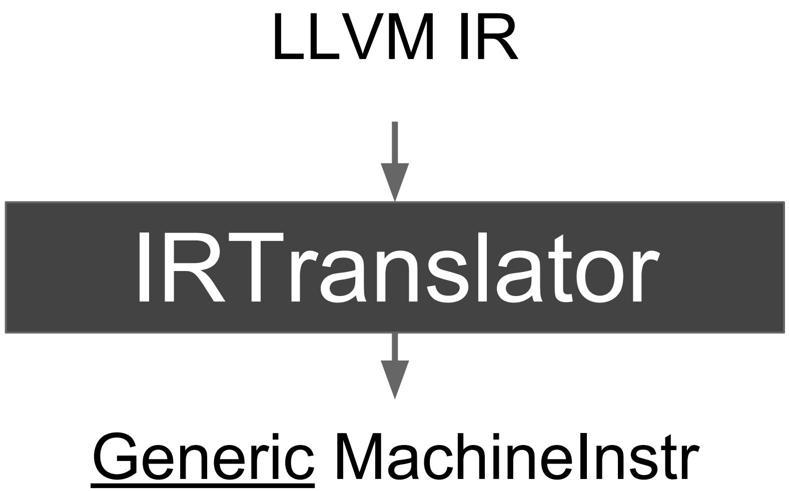
Legalizer

RegBankSelect

InstructionSelect

MachineInstr

IRTranslator



LLVM IR

IRTranslator

Generic MachineInstr

G_ADD
G_LOAD
G_ANYEXT
G_FRAME_INDEX
G_CONSTANT
G_BRCOND
G_INTRINSIC
G_FADD
...

IRTranslator

IRTranslator output (Generic MIR):

```
name:            add
registers:
  - { id: 0, class: _ }
  - { id: 1, class: _ }
  - { id: 2, class: _ }
body:             |
  bb.1.entry:
    liveins: %w0, %w1
    %0(s32) = COPY %w0
    %1(s32) = COPY %w1
    %2(s32) = G_ADD %1, %0
    %w0 = COPY %2(s32)
    RET_ReallyLR implicit %w0
```

Final goal of instruction selection (MIR):

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:             |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

IRTranslator

IRTranslator output (Generic MIR):

```
name:            add
registers:
  - { id: 0, class: _ }
  - { id: 1, class: _ }
  - { id: 2, class: _ }
body:             |
  bb.1.entry:
    liveins: %w0, %w1
    %0(s32) = COPY %w0
    %1(s32) = COPY %w1
    %2(s32) = G_ADD %1, %0
    %w0 = COPY %2(s32)
    RET_ReallyLR implicit %w0
```

Final goal of instruction selection (MIR):

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:             |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

IRTranslator

IRTranslator output (Generic MIR):

```
name:                  add
registers:
  - { id: 0, class: _ }
  - { id: 1, class: _ }
  - { id: 2, class: _ }
body:                  |
  bb.1.entry:
    liveins: %w0, %w1
    %0(s32) = COPY %w0
    %1(s32) = COPY %w1
    %2(s32) = G_ADD %1, %0
    %w0 = COPY %2(s32)
    RET_ReallyLR implicit %w0
```

Final goal of instruction selection (MIR):

```
name:                  add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:                  |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

```
Scalar        sN (number of bits)
Pointer       pN (address space)
Vector    M x sN (lanes x number of bits)
```

IRTranslator

IRTranslator output (Generic MIR):

```
name:            add
registers:
  - { id: 0, class: _ }
  - { id: 1, class: _ }
  - { id: 2, class: _ }
body:             |
  bb.1.entry:
    liveins: %w0, %w1
    %0(s32) = COPY %w0
    %1(s32) = COPY %w1
    %2(s32) = G_ADD %1, %0
    %w0 = COPY %2(s32)
    RET_ReallyLR implicit %w0
```

Final goal of instruction selection (MIR):

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:             |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

IRTranslator

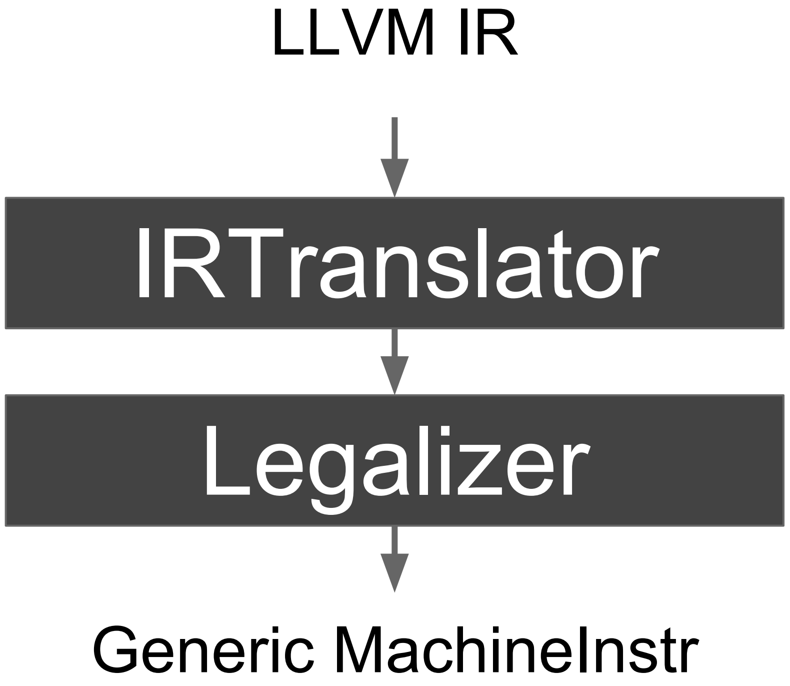
IRTranslator output (Generic MIR):

```
name:            add
registers:
  - { id: 0, class: _ }
  - { id: 1, class: _ }
  - { id: 2, class: _ }
body:            |
  bb.1.entry:
    liveins: %w0, %w1
    %0(s32) = COPY %w0
    %1(s32) = COPY %w1
    %2(s32) = G_ADD %1, %0
    %w0 = COPY %2(s32)
    RET_ReallyLR implicit %w0
```

Final goal of instruction selection (MIR):

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:            |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

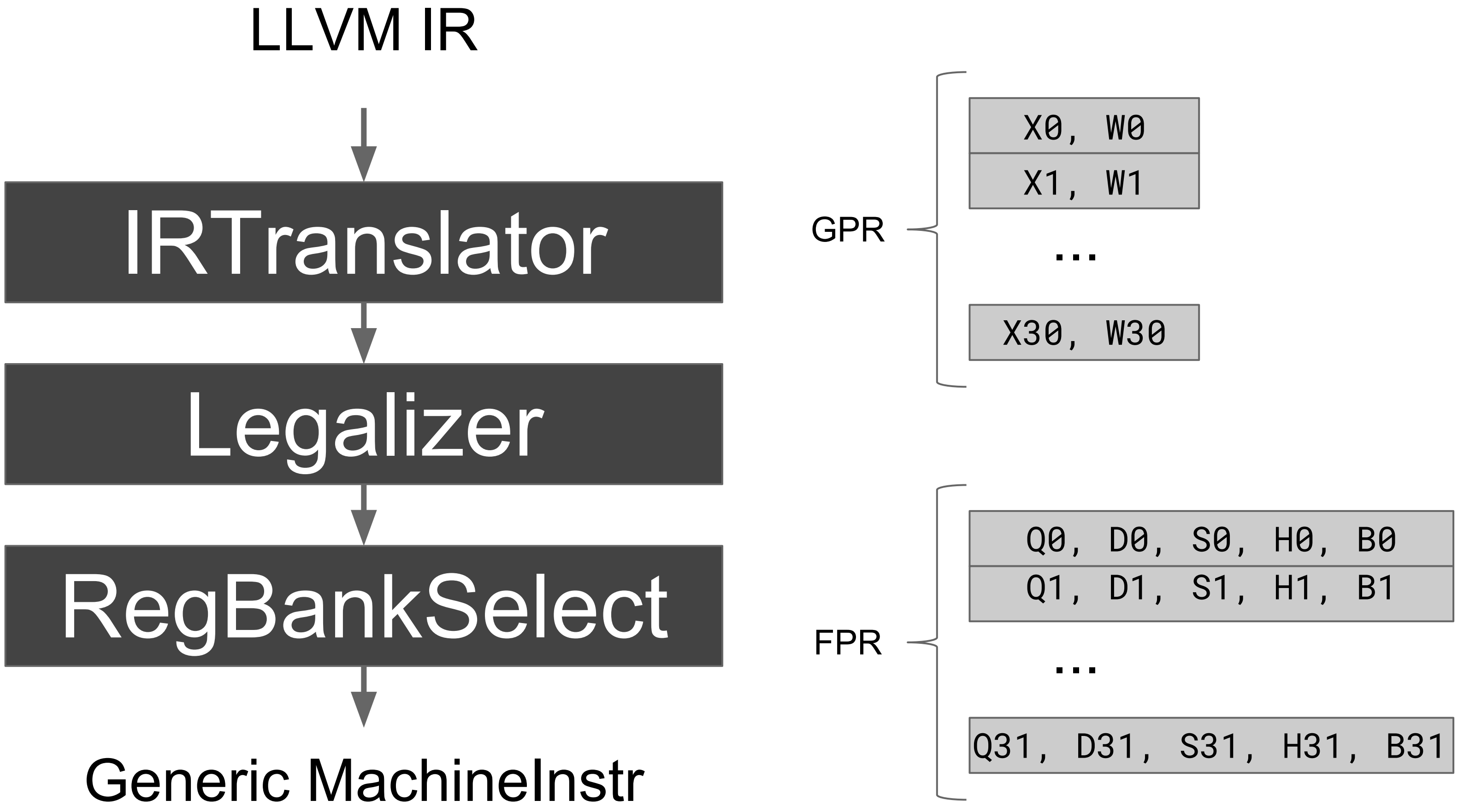
Legalizer



(operation, type)

- Legal
- NarrowScalar
- WidenScalar
- FewerElements
- MoreElements
- Lower
- Libcall
- Custom
- Unsupported

Register Bank Selection



LLVM IR

IRTranslator

Legalizer

RegBankSelect

Generic MachineInstr

GPR

X0, W0

X1, W1

...

X30, W30

FPR

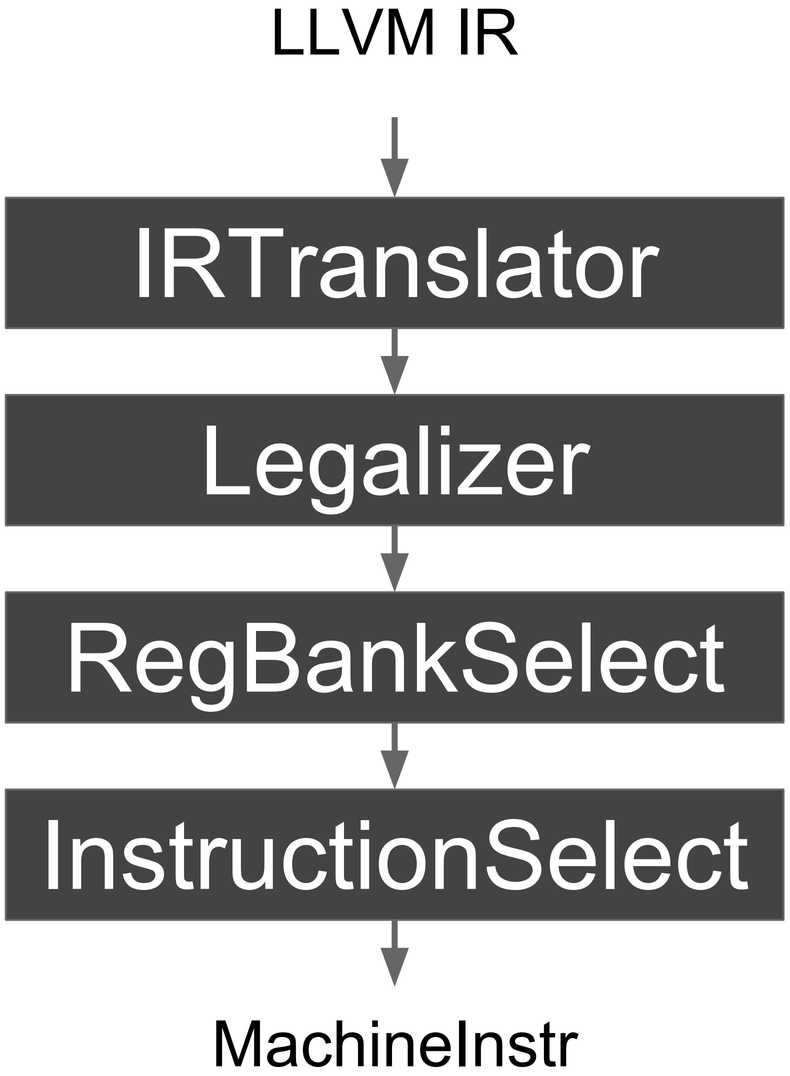
Q0, D0, S0, H0, B0

Q1, D1, S1, H1, B1

...

Q31, D31, S31, H31, B31

Instruction Selection



Instruction Selection

Before instruction selection:

```
name:            add
registers:
  - { id: 0, class: gpr }
  - { id: 1, class: gpr }
  - { id: 2, class: gpr }
body:             |
  bb.1.entry:
    liveins: %w0, %w1
    %0(s32) = COPY %w0
    %1(s32) = COPY %w1
    %2(s32) = G_ADD %1, %0
    %w0 = COPY %2(s32)
    RET_ReallyLR implicit %w0
```

After instruction selection:

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:             |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

Instruction Selection

Before instruction selection:

```
name:                 add
registers:
  - { id: 0, class: gpr }
  - { id: 1, class: gpr }
  - { id: 2, class: gpr }
body:                  |
  bb.1.entry:
    liveins: %w0, %w1
    %0(s32) = COPY %w0
    %1(s32) = COPY %w1
    %2(s32) = G_ADD %1, %0
    %w0 = COPY %2(s32)
    RET_ReallyLR implicit %w0
```

After instruction selection:

```
name:                 add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:                  |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

Instruction Selection

Before instruction selection:

```
name:            add
registers:
  - { id: 0, class: gpr }
  - { id: 1, class: gpr }
  - { id: 2, class: gpr }
body:             |
  bb.1.entry:
    liveins: %w0, %w1
    %0(s32) = COPY %w0
    %1(s32) = COPY %w1
    %2(s32) = G_ADD %1, %0
    %w0 = COPY %2(s32)
    RET_ReallyLR implicit %w0
```

After instruction selection:

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:             |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

Instruction Selection

```
name:            add
registers:
  - { id: 0, class: gpr32 }
  - { id: 1, class: gpr32 }
  - { id: 2, class: gpr32 }
body:             |
  bb.0.entry:
    liveins: %w0, %w1
    %0 = COPY %w0
    %1 = COPY %w1
    %2 = ADDWrr %1, %0
    %w0 = COPY %2
    RET_ReallyLR implicit %w0
```

Backend Passes

Current Status

- Prototype, disabled by default

```
llc -global-isel [...]
clang -mllvm -global-isel [...]


llc -global-isel -global-isel-abort=0 [...]
```

Current Status

- Work in progress: Improving the framework (e.g. TableGen)

```
/// General Purpose Registers: W, X.
def GPRRegBank : RegisterBank<"GPR", [GPR64all]>;

/// Floating Point/Vector Registers: B, H, S, D, Q.
def FPRRegBank : RegisterBank<"FPR", [QQQQ]>;

/// Conditional register: NZCV.
def CCRRegBank : RegisterBank<"CCR", [CCR]>;
```

Current Status

- Work in progress: Improving the framework (e.g. TableGen)

```
/// General Purpose Registers: W, X.
def GPRRegBank : RegisterBank<"GPR", [GPR64all]>;

/// Floating Point/Vector Registers: B, H, S, D, Q.
def FPRRegBank : RegisterBank<"FPR", [QQQQ]>;

/// Conditional register: NZCV.
def CCRRegBank : RegisterBank<"CCR", [CCR]>;
```

- Work in progress: Target adoption
  - AArch64
    - Passes > 63% of the test-suite
    - Plans to replace FastISel this year
    - Much faster than DAGISel (but worse code)
    - Hoping to get within 1.1x of FastISel
  - ARM
  - AMDGPU
  - x86

Summary + Q&A

- Status (20 January 2017):
  http://lists.llvm.org/pipermail/llvm-dev/2017-January/109366.html
- Docs: http://llvm.org/docs/GlobalISel.html
- In depth presentation from US LLVM:
  https://www.youtube.com/watch?v=6tfb344A7w8

?

Linaro

Legalizer

Input:

```
%0(p0) = COPY %x0
%1(s128) = G_LOAD %0(p0) :: (load 16 from %ir.x)
```

Output:

```
%0(p0) = COPY %x0
%13(s64) = G_CONSTANT i64 0
%12(p0) = G_GEP %0, %13(s64)
%11(s64) = G_LOAD %12(p0) :: (load 16 from %ir.x)
%16(s64) = G_CONSTANT i64 8
%15(p0) = G_GEP %0, %16(s64)
%14(s64) = G_LOAD %15(p0) :: (load 16 from %ir.x)
%9(s128) = G_SEQUENCE %11(s64), 0, %14(s64), 64
```

Register Bank Selection

```
%0(s64) = COPY %x0
%1(p0) = COPY %x1
%2(<2 x s32>) = G_BITCAST %0(s64)
%3(<2 x s32>) = G_LOAD %1(p0) :: (load 8 from %ir.addr)
%4(<2 x s32>) = G_OR %2, %3
%5(s64) = G_BITCAST %4(<2 x s32>)
%x0 = COPY %5(s64)
RET_ReallyLR implicit %x0
```

Fast:

```
registers:
  - { id: 0, class: gpr }
  - { id: 1, class: gpr }
  - { id: 2, class: fpr }
  - { id: 3, class: fpr }
  - { id: 4, class: fpr }
  - { id: 5, class: gpr }
```

Greedy:

```
registers:
  - { id: 0, class: gpr }
  - { id: 1, class: gpr }
  - { id: 2, class: gpr }
  - { id: 3, class: gpr }
  - { id: 4, class: gpr }
  - { id: 5, class: gpr }
```