Life of an instruction in Clang / LLVM

Francis Visoiu Mistrih - francis@lse.epita.fr

Clang

foo.c - C

```c
int foo(int a, int b)
{
   return a + b;
}
```

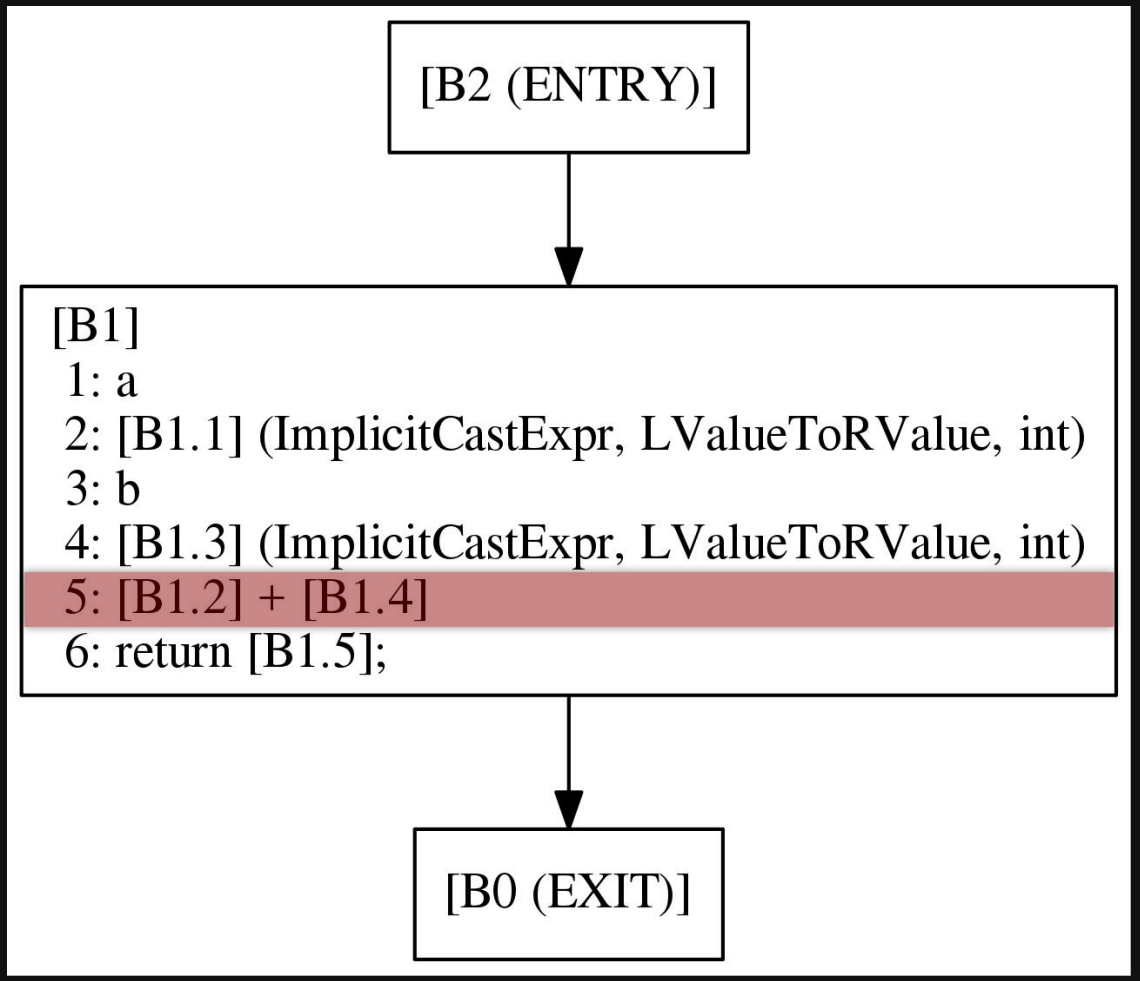
Clang AST

`clang-check -ast-dump foo.c`

```
TranslationUnitDecl 0x29eaa20
`-FunctionDecl 0x29eb460 <foo.c:1:1, line:4:1> line:1:5 foo 'int (int, int)'
  |-ParmVarDecl 0x29eb318 <col:9, col:13> col:13 used a 'int'
  |-ParmVarDecl 0x29eb388 <col:16, col:20> col:20 used b 'int'
  `-CompoundStmt 0x29eb618 <line:2:1, line:4:1>
    `-ReturnStmt 0x29eb600 <line:3:3, col:14>
      `-BinaryOperator 0x29eb5d8 <col:10, col:14> 'int' '+'
        |-ImplicitCastExpr 0x29eb5a8 <col:10> 'int' <LValueToRValue>
        | `-DeclRefExpr 0x.. <col:10> 'int' lvalue ParmVar 0x29eb318 'a' 'int'
        `-ImplicitCastExpr 0x29eb5c0 <col:14> 'int' <LValueToRValue>
          `-DeclRefExpr 0x.. <col:14> 'int' lvalue ParmVar 0x29eb388 'b' 'int'
```

Clang CFG

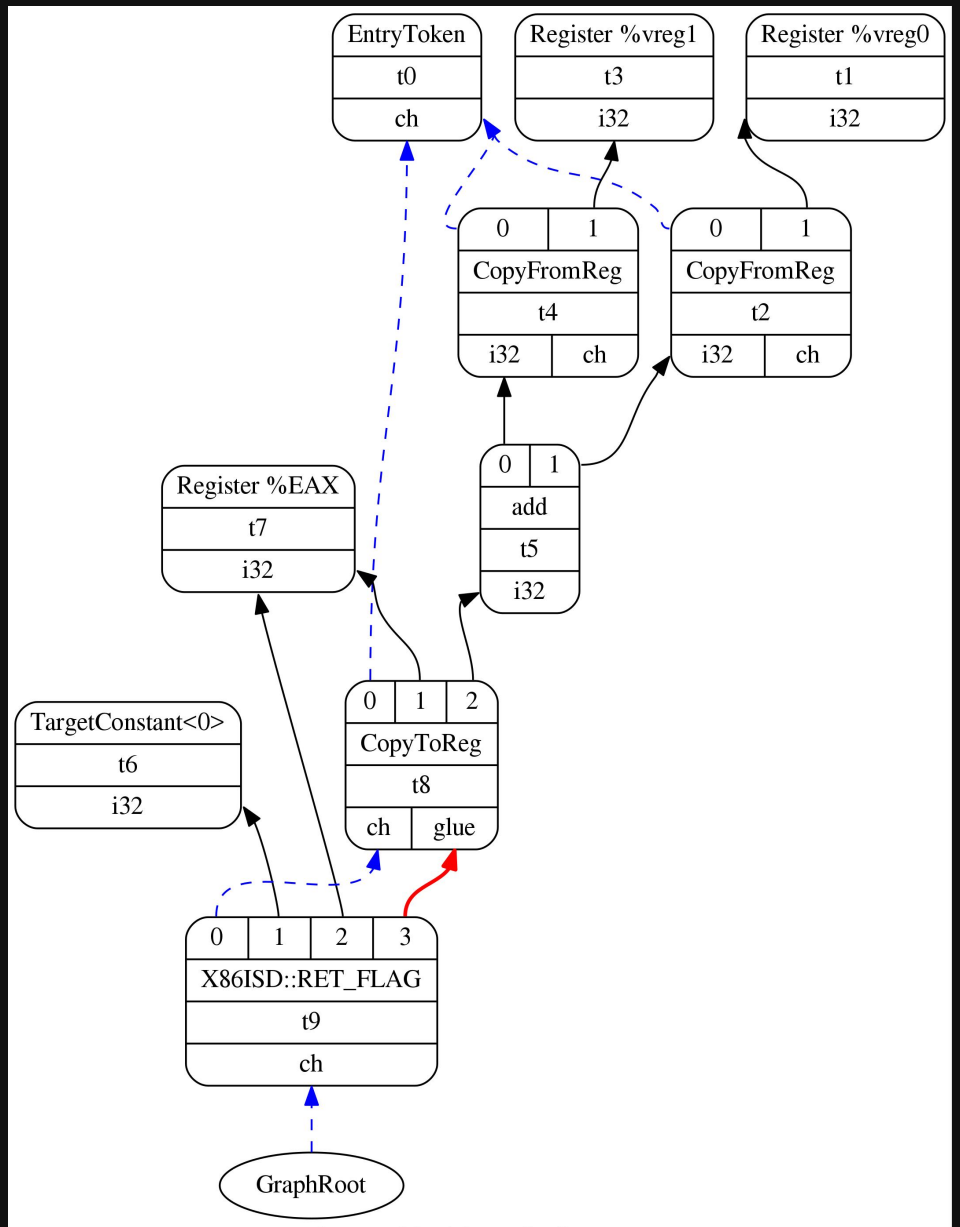
```
clang -cc1 -analyze -analyzer-checker=debug.ViewCFG foo.c
```



LLVM IR

`clang -O1 -S -emit-llvm foo.c -o foo.ll`

```
; Function Attrs: norecurse nounwind readnone uwtable
define i32 @foo(i32 %a, i32 %b) {
entry:
  %add = add nsw i32 %b, %a
  ret i32 %add
}
```

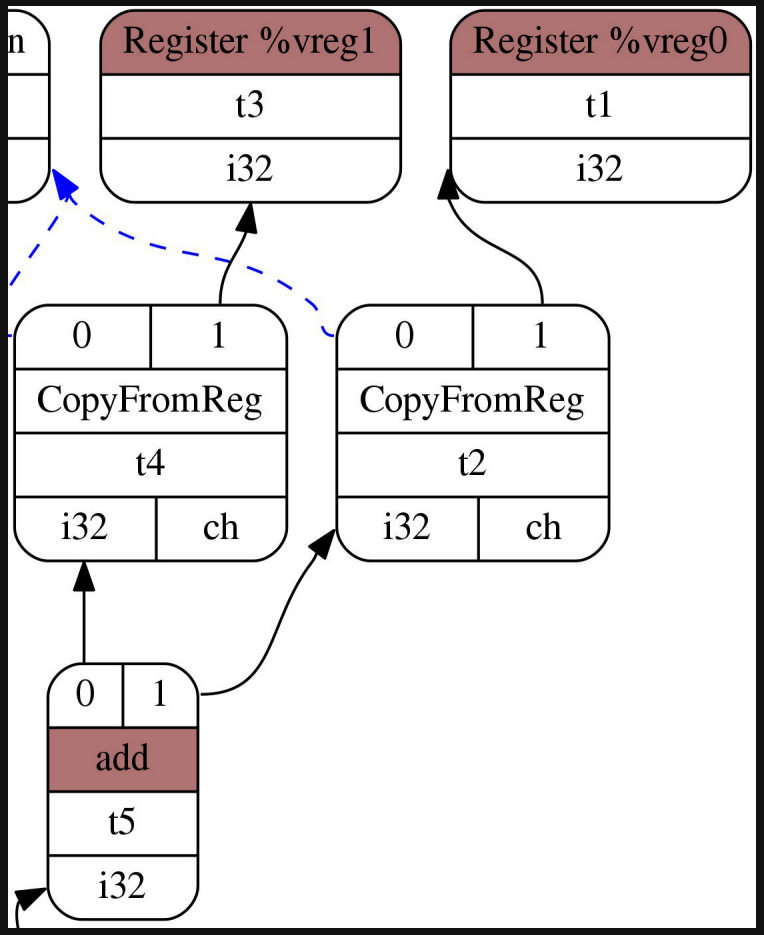


dag-combine1 input for foo:entry

TargetLowering - SelectionDAG

SDNode

`llc -O1 -view-dag-combine1-dags foo.ll`



SelectionDAG legalization

SDNode

`llc -O1 -view-isel-dags foo.ll`

- Remove illegal types

```
i1 -> i32
```

- Remove target-illegal instructions

```
mul i32 %a, 8
```

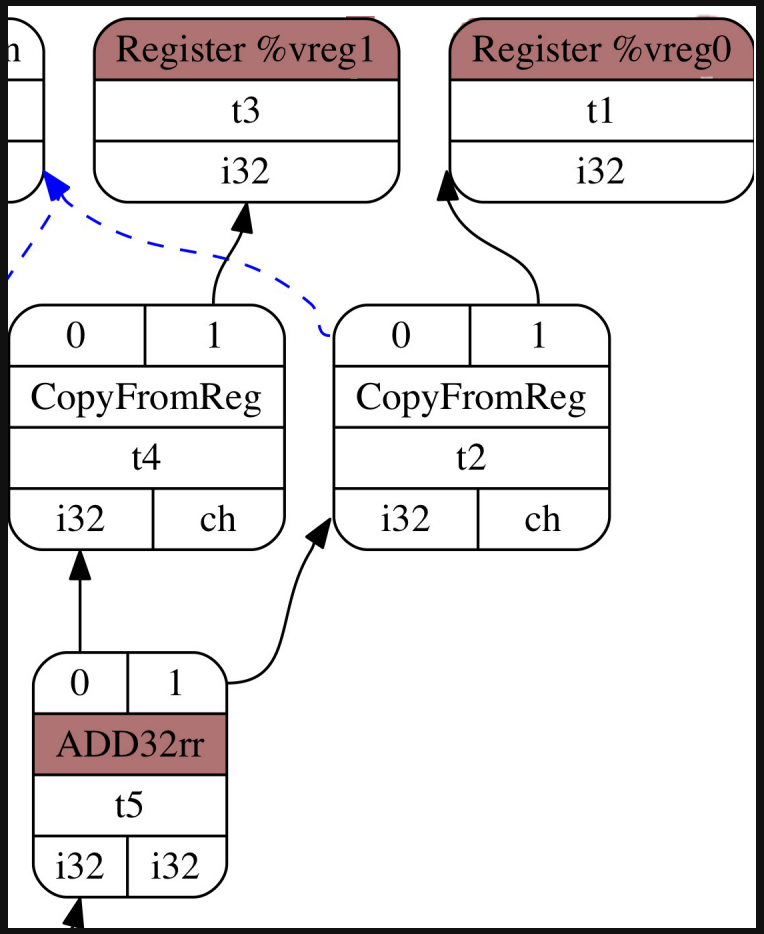
to

```
shl i32 %a, 3
```

SelectionDAGISel

MachineSDNode

```
llc -O1 -view-sched-dags foo.ll
```



SelectionDAGISel

MachineSDNode

llvm/lib/Target/X86/X86InstrCompiler.td

```
def : Pat<(add GR32:$src1, GR32:$src2), (ADD32rr GR32:$src1, GR32:$src2)>;
```

ScheduleDAG

MachineInstr

`llc -O1 -print-before=expand-isel-pseudos foo.ll`

```
Function Live Ins: %EDI in %vreg0, %ESI in %vreg1

BB#0: derived from LLVM BB %entry
  Live Ins: %EDI %ESI
    %vreg1<def> = COPY %ESI; GR32:%vreg1
    %vreg0<def> = COPY %EDI; GR32:%vreg0
    %vreg2<def,tied1> = ADD32rr %vreg0<tied0>, %vreg1, %EFLAGS<imp-def,dead>
    %EAX<def> = COPY %vreg2; GR32:%vreg2
    RET 0, %EAX
```

Register allocation

Non-SSA MachineInstr

llc -O1 -print-machineinstrs=virtregrewriter foo.ll

```
Function Live Ins: %EDI, %ESI

0B  BB#0: derived from LLVM BB %entry
      Live Ins: %EDI %ESI
80B     %EAX<def> = LEA64_32r %RDI<kill>, 1, %RSI<kill>, 0, %noreg
112B    RET 0, %EAX
```

MCInstLower

MCInst

`llc -O1 -asm-show-inst foo.ll -o foo.s`

```
 @foo
 %entry
<MCInst #1278 LEA64_32r
  <MCOperand Reg:19>
  <MCOperand Reg:39>
  <MCOperand Imm:1>
  <MCOperand Reg:43>
  <MCOperand Imm:0>
  <MCOperand Reg:0>>
<MCInst #2472 RETQ
  <MCOperand Reg:19>>
```

AsmPrinter

Assembly

`llc -O1 foo.ll -o foo.s / clang -O1 foo.c -S -o foo.s`

```
foo:                                    # @foo
# BB#0:                                 # %entry
        leal    (%rdi,%rsi), %eax       # <MCInst #1278 LEA64_32r
                                        #  <MCOperand Reg:19>
                                        #  <MCOperand Reg:39>
                                        #  <MCOperand Imm:1>
                                        #  <MCOperand Reg:43>
                                        #  <MCOperand Imm:0>
                                        #  <MCOperand Reg:0>>
        retq                            # <MCInst #2472 RETQ
```

AsmPrinter

Assembly

lib/Target/X86/X86InstrArithmetic.td

```
def LEA64_32r : I<0x8D, MRMSrcMem,
                  (outs GR32:$dst), (ins lea64_32mem:$src),
                  "lea{l}\t{$src|$dst}, {$dst|$src}",
                  [(set GR32:$dst, lea64_32addr:$src)], IIC_LEA>,
                  OpSize32, Requires<[In64BitMode]>;
```

AsmPrinter

Assembly

lib/Target/X86/X86InstrArithmetic.td

```
def LEA64_32r : I<0x8D, MRMSrcMem,
                  (outs GR32:$dst), (ins lea64_32mem:$src),
                  "lea{l}\t{$src|$dst}, {$dst|$src}",
                  [(set GR32:$dst, lea64_32addr:$src)], IIC_LEA>,
                  OpSize32, Requires<[In64BitMode]>;
```

AsmPrinter

Assembly

lib/Target/X86/X86InstrArithmetic.td

```
def LEA64_32r : I<0x8D, MRMSrcMem,
                  (outs GR32:$dst), (ins lea64_32mem:$src),
                  "lea{l}\t{$src|$dst}, {$dst|$src}",
                  [(set GR32:$dst, lea64_32addr:$src)], IIC_LEA>,
                  OpSize32, Requires<[In64BitMode]>;
```

AsmPrinter

Assembly

lib/Target/X86/X86InstrArithmetic.td

```
def LEA64_32r : I<0x8D, MRMSrcMem,
                  (outs GR32:$dst), (ins lea64_32mem:$src),
                  "lea{l}\t{$src|$dst}, {$dst|$src}",
                  [(set GR32:$dst, lea64_32addr:$src)], IIC_LEA>,
                  OpSize32, Requires<[In64BitMode]>;
```

MCCodeEmitter

ELF

`clang -c -fintegrated-as -O1 foo.c -o foo.o`

`llvm-objdump -disassemble foo.o`

```
foo:
       0:       8d 04 37        leal    (%rdi,%rsi), %eax
       3:       c3              retq
```

References

- http://eli.thegreenplace.net/2012/11/24/life-of-an-instruction-in-llvm
- http://llvm.org/git/llvm.git
- http://llvm.org/docs/CodeGenerator.html

Any questions?

francis@lse.epita.fr

@thegameg