Convert WHIRL IR to LLVM IR

Yan Zhao

28/March/2016

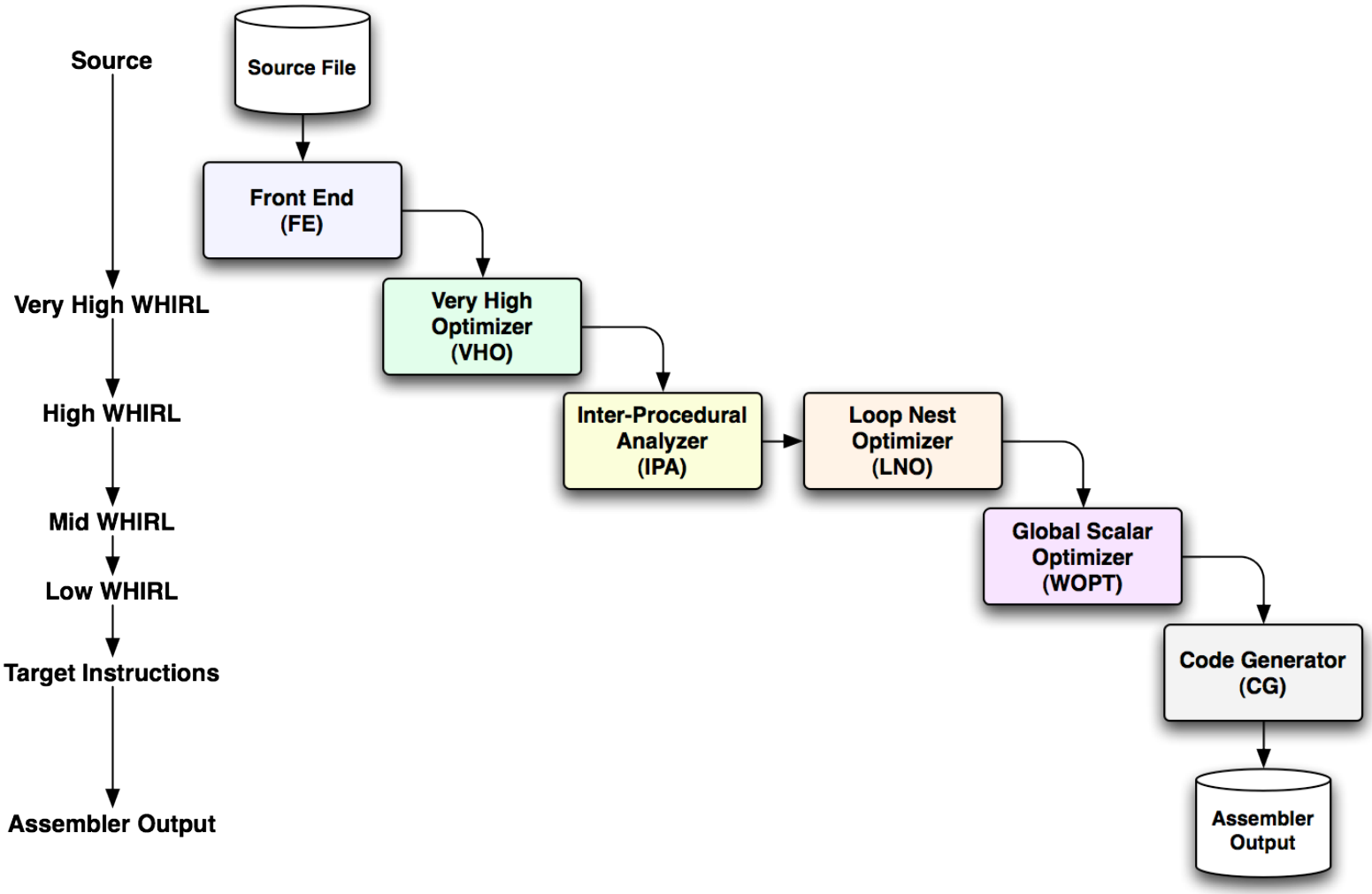
Contents

- Basic Idea
- Methodologies
- Difficulties and solutions
- Progress

Current Project-Basic

- OpenUH compiler and WHIRL IR
  - WHIRL is hierarchical intermediate representation (IR).
  - 5 main levels.
  - High-quality interprocedural analysis, data-flow analysis, data dependence analysis.
  - Mainly used as research platform.

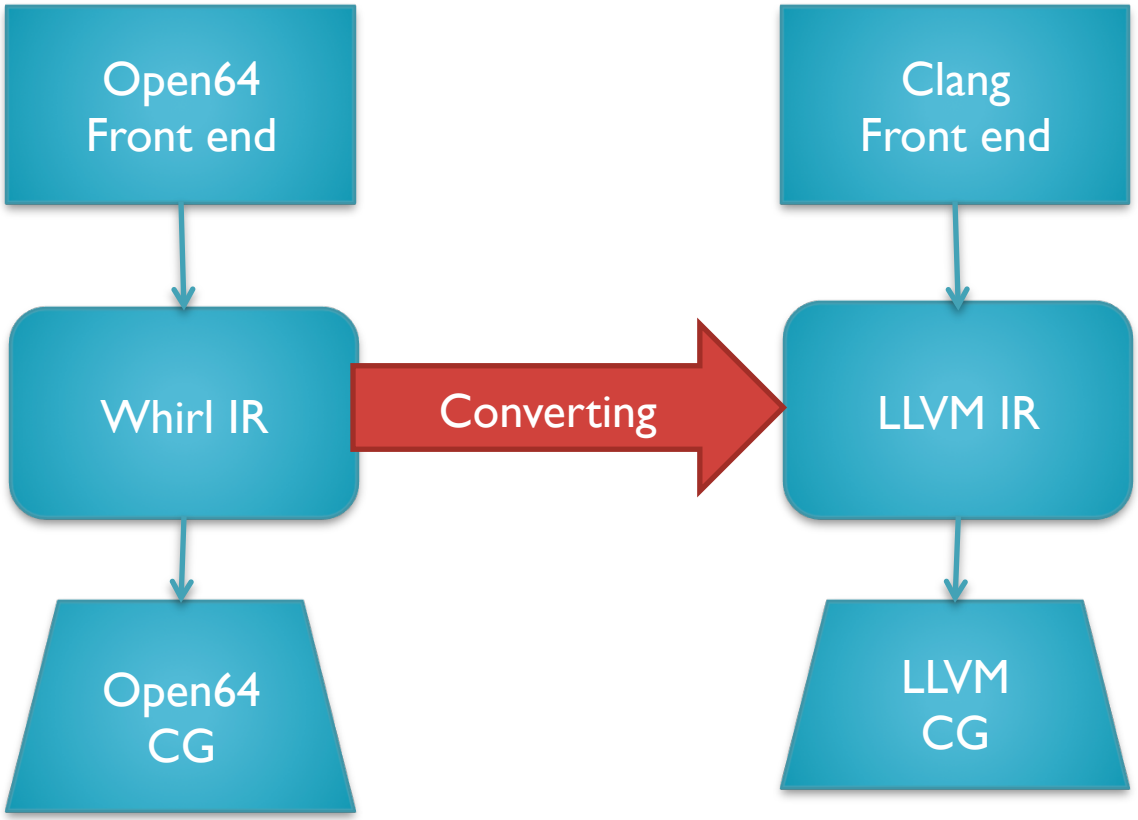
Current Project-Basic



Current Project-Basic

- Clang and LLVM IR
  - Modular design and more documents.
  - More commercial support.
  - More permissive License.
  - Nearly the same performance as GCC

Current Project-Basic



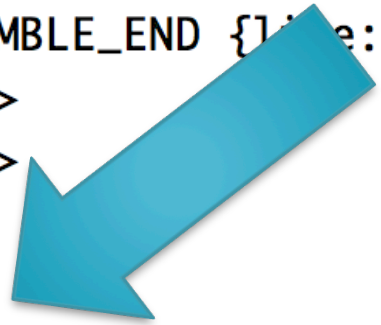
Current Project-Methodologies(1)

- Given an C++ source code
  - compile a.cpp by OpenUH
    - produce WHIRL IR file
  - compile a.cpp by clang
    - produce LLVM IR file

Current Project-Methodologies(2)

- Work on WHIRL IR side
  - Understand WHIRL IR language. (Whirl IR reference)

```
BLOCK {line: 1/5}
PRAGMA 0 119 <null-st> 0 (0x0) # PREAMBLE_END {line: 1/5}
  I4I4LDID 0 <2,4,x> T<4,.predef_I4,4>
  I4I4LDID 0 <2,5,y> T<4,.predef_I4,4>
 I4ADD
I4RETURN_VAL {line: 1/7}
END_BLOCK
```



Current Project-Methodologies(3)

- Work on LLVM IR side(1)
  - Understand LLVM IR(LLVM IR language reference)

```
%x.addr = alloca i32, align 4
%y.addr = alloca i32, align 4
store i32 %x, i32* %x.addr, align 4
store i32 %y, i32* %y.addr, align 4
%0 = load i32, i32* %x.addr, align 4
%1 = load i32, i32* %y.addr, align 4
%add = add i32 %0, %1
ret i32 %add
```

Current Project-Methodologies(4)

- Work on LLVM IR(2)
  - How to produce LLVM IR by C++ code?
  - llc –march=cpp llvm.bc
  - clang source code

```
StoreInst *st0 = new StoreInst(int32_a, ptrA, false, labelEntry);
st0->setAlignment(4);
StoreInst *st1 = new StoreInst(int32_b, ptrB, false, labelEntry);
st1->setAlignment(4);

LoadInst *ld0 = new LoadInst(ptrA, "", false, labelEntry);
ld0->setAlignment(4);
LoadInst *ld1 = new LoadInst(ptrB, "", false, labelEntry);
ld0->setAlignment(4);
BinaryOperator *addRes = BinaryOperator::Create(Instruction::Add, ld0, ld1,
add", labelEntry);
ReturnInst::Create(mod->getContext(), addRes, labelEntry);
```

Current Project-Methodologies(5)

- Work on WHIRL IR side
  - Modify WHIRL IR traversal code. (ir_b2a)
  - Inserting LLVM IR generating code into WHIRL IR traversal code
    - I4ADD → BinaryOperator * addRes = BinaryOperator::Create(Instruction::Add, ld0, ld1, labelentry);

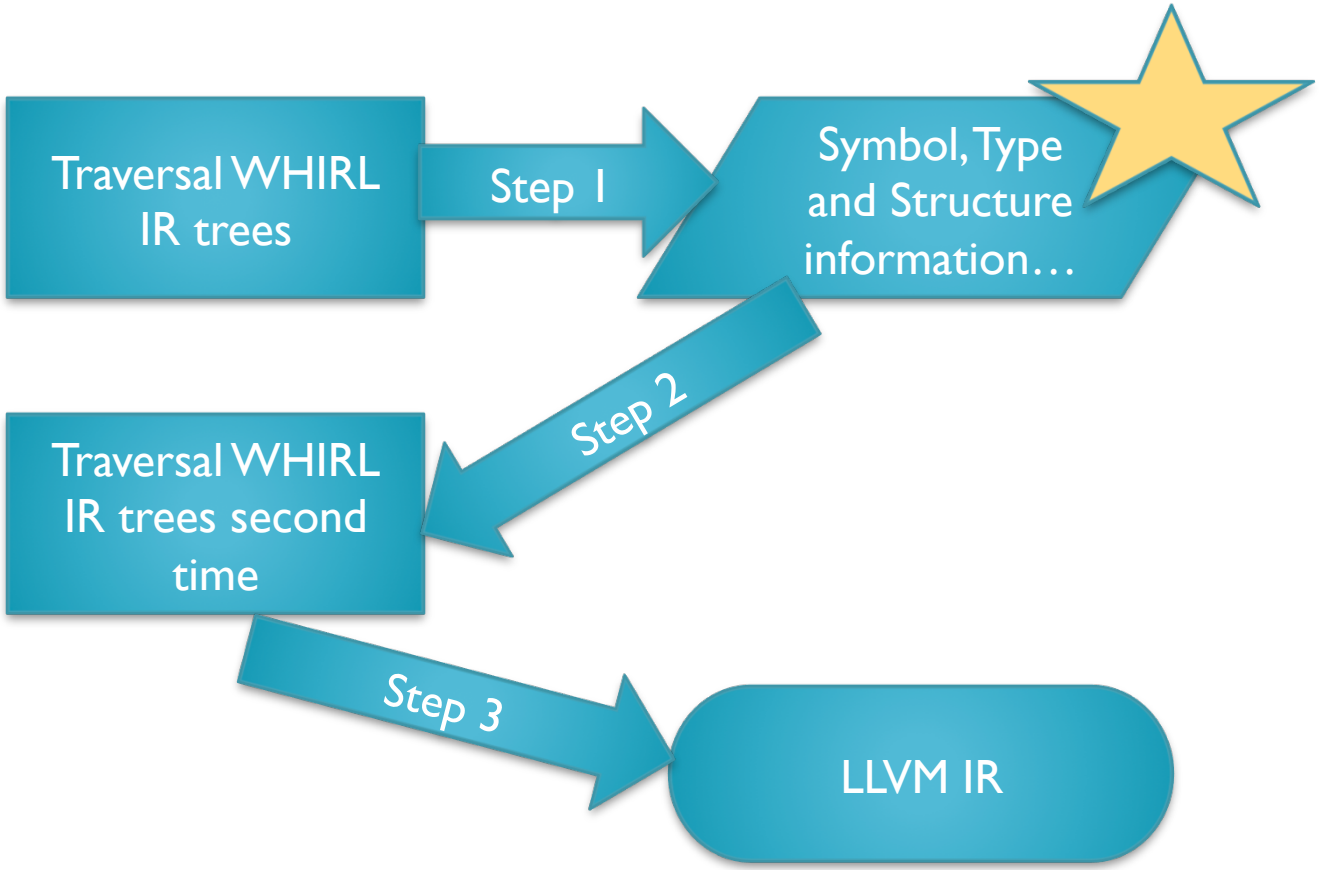
```
BLOCK {line: 1/5}
PRAGMA 0 119 <null-st> 0 (0x0) # PREAMBLE_END {line: 1/5}
  I4I4LDID 0 <2,4,x> T<4,.predef_I4,4>
  I4I4LDID 0 <2,5,y> T<4,.predef_I4,4>
 I4ADD
I4RETURN_VAL {line: 1/7}
END_BLOCK
```

Current Project-Methodologies(6)

- Verify your LLVM IR
  - lli llvm.bc
  - lli is a JIT engine, It will run LLVM IR directly

Current Product-Difficulties

- No one-to-one Mapping.



Current Product-Difficulties

- How to read, understand and modify large, complex software.(Compilers)
  - Comprehensive C/C++ language knowledge.
  - C11 and C14 standard
  - Basic Compiler knowledge.
  - Good tools
    - VIM is good, but not enough(Good IDE)
    - Visual debugger.

Current Product-progress

- Finished
  - Main Control flow
    - For, while, switch, function, invoke a function.
  - Data type
    - Pointer, Struct, Array
  - Developing Platform, Methodologies, and Tool
- Future
  - Practical C++ source code.
  - More test.
  - Maybe redesign!!!

The end

- At least for the people who send me mail about a new language that they're designing, the general advice is: do it to learn about how to write a compiler.

  --Dennis Ritchie (Father of C language)--

- **Thank you.**