



LLVM IR SDK - an LLVM IR Eclipse* Plugin

alon.mishne@intel.com, Intel® Software and Services Group - Israel



LLVM IR SDK is an Eclipse* plugin which provides LLVM developers with a .ll file editor, helping when writing tests or analyzing pass output.

The plugin implements a wide range of validation checks, including type checking and comination checking, enabling a quick modify-and-run cycle for IR files without having to run LLVM module validation in-between.

In addition, the plugin provides content-assistance and exposes a variety of “quick-fix” options for common code actions, to speed up code authoring.

```
test.ll
%mytype = type i32
global float 3.0
declare void @f()
define i32 @g(i32* %x) {
    load i32* %x
    store i32 sitofp (float @0 to i32),
        i32* %x
    br label %ret
ret:
    ret i32 %1
}
```

Outline

- test
 - typedef %mytype
 - global @0
 - declare @f
 - define @g
 - %0 (3 instructions)
 - %ret (1 instructions)

- Free to use (and open source).
- Does not depend on any LLVM tool.
- Supports IR from versions 3.2 and 3.3 (including attribute groups).
- Still in beta!

Automatic Declaration Creation

```
Couldn't resolve reference to GlobalValueDef '@missing'.
1 quick fix available:
Create function declaration
21
22 declare double @missing(i1)
23
```

Type Checking & Auto-Conversion

```
Expected i7, found float
2 quick fixes available:
Insert fptosi conversion for %1
Insert fptoui conversion for %1
11 %converted.1 = fptoui float %1 to i7
12 %x = add i7 3155, %converted.1
```

“Unnamed” Name Inferring & Fixing

```
Incorrect number in sequence: expected %4, got %5
2 quick fixes available:
Rename %5 to %4
Update all names in current sequence
17 phi i7 [ 0, %bb1 ], [%x, %bb2]
18 %4 = add i7 1, 1
19 %5 = add i7 %4, %x1
```

```
*test.ll
1 %mytype = type { %mytype* }
2
3 ; My function!
4 ; CHECK: mov
5
6 define double (i8, { %mytype})* @test(i1 %cond) {
7   add float 3.0, 0xabc
8   %y = call double @missing(i1 %cond)
9   br i1 %cond, label %bb1, label %bb2
10 bb1:
11   %x = add i7 3155, %1
12   br label %join
13 bb2: No predecessors!
14   br label %join
15 join:
16   phi i7 [ %x, %bb1 ]
17   phi i7 [ 0, %bb1 ], [%x, %bb2]
18   %5 = add i7 1, 1
19   %6 = add i7 %5, %x1
20 }
21 no viable alternative at input ')' (did you forget a terminator instruction for basic block %join?)
```

Outline

- test
 - typedef %mytype
 - define @test
 - %0 (3 instructions)
 - %bb1 (2 instructions)
 - %bb2 (1 instructions)
 - %join (4 instructions)

Definitions on Hover

```
bb1:
%x = add i7 3155, %1
br label %join
```

Control-Flow Awareness

```
The basic block %bb2 is missing from this phi node
The value %x is neither defined in %bb2 nor is dominating it
```

