LLVM first steps

Kouhei Ueno (id: nyaxt)
ueno@nyaxtstep.com
http://nyaxtstep.com

Introducing myself

- nytr renderer
  - http://nyaxtstep.com/projects/nytr
- libpolatsk: Task-based distributed computing library
- cagra: distributed storage system
  - http://cagra.org

Agenda

- Playing around w/ LLVM asm
- How to write an LLVM bitcode driver

Part 1: Playing around w/ LLVM asm

Why LLVM asm?

- LLVM C++ API sucks!
  - too complicated
  - many differences between versions
- Output LLVM asm and compile using llvm-as!

- Time to write our first LLVM code!

Step1: A function that returns 1.0

```
          return type
define double @func()
                    function name
{
  ret double 1.0
instruction
}       return type
                    arg1, arg2, ...
```

Step 2: using registers and performing basic ops.

```
define double @func()
{
  %x = add double 1.0, 2.0
```

register name: `%x` — instruction: `add` — return type: `double` — arg1, arg2, ...

```
  ret double %x
}
```

Step 3: Calling functions

```
define double @addwrap(double %a, double %b)
{
    %res = add double %a, %b
    ret double %res
}

define double @func()
{
    %x = call double @addwrap(double 1.0, double 2.0)
    ret double %x
}
```

Step 3': Calling external functions

```
declare void @putdoubled(double %x)

define double @func()
{
    %x = call double @addwrap(double 1.0, double 2.0)
    call void @putdoubled(double %x)
    ret double %x
}
```

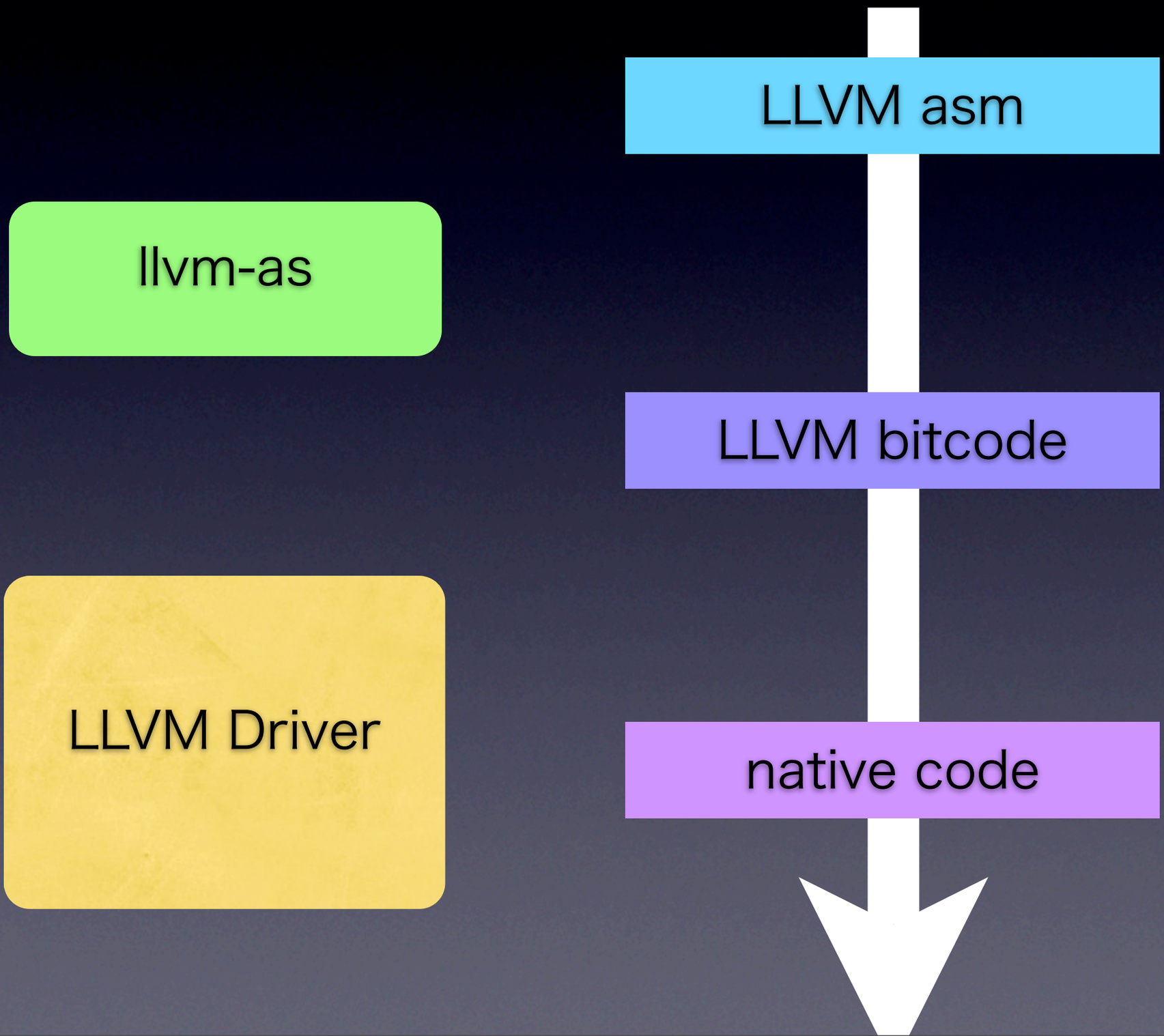
Step 4: Using Pointers

```
define double @func()
{
    %x = call double @addwrap(double 1.0, double 2.0)

    %ptr = alloca double
    store double %x, double* %ptr

    %y = load double* %ptr
    ret double %y
}
```

Part 2: LLVM driver

- Very basic LLVM driver
  - Load llvm bitcode file
  - Setup Execution Engine
  - JIT
  - RUN!!!

What we are going to do



Let’s start coding...

Loading Bitcode

```
if(argc < 1) return 1;

// create module from bit-code file
llvm::Module* pmodule;
{
    std::string strErr;

    boost::scoped_ptr<llvm::MemoryBuffer>
            pbuf(llvm::MemoryBuffer::getFile(argv[1], &strErr));
    pmodule = llvm::ParseBitcodeFile(pbuf.get(), &strErr);
}
```

Setup Execution Engine

```
// setup execution engine
llvm::ExecutionEngine* pee =
  llvm::ExecutionEngine::create(pmodule);

// find function to run
llvm::Function* pfunc =
  pmodule->getFunction("func");
```

JIT & RUN!

```
// jit compile and execute pfunc
{
    double (*pfuncnative)() =
        (double (*)())pee->getPointerToFunction(pfunc);
    std::cout << "evaled to " << pfuncnative();
}
```

(optional)

Perform Optimization

```
#ifdef OPTIMIZER
    // setup optimizer
    llvm::ExistingModuleProvider mp(pmodule);
    llvm::FunctionPassManager fpm(&mp);
    {
        fpm.add(new llvm::TargetData(*pee->getTargetData()));
        fpm.add(llvm::createInstructionCombiningPass());
        fpm.add(llvm::createReassociatePass());
        fpm.add(llvm::createGVNPass());
        fpm.add(llvm::createCFGSimplificationPass());
    }

    // run optimizer
    fpm.run(*pfunc);
#endif
```

Thank you for listening!

Slide pdf and source codes
will be made available @ google group

宣伝：　低レベルプログラミングIRC
#lowhacks @ irc.freenode.net