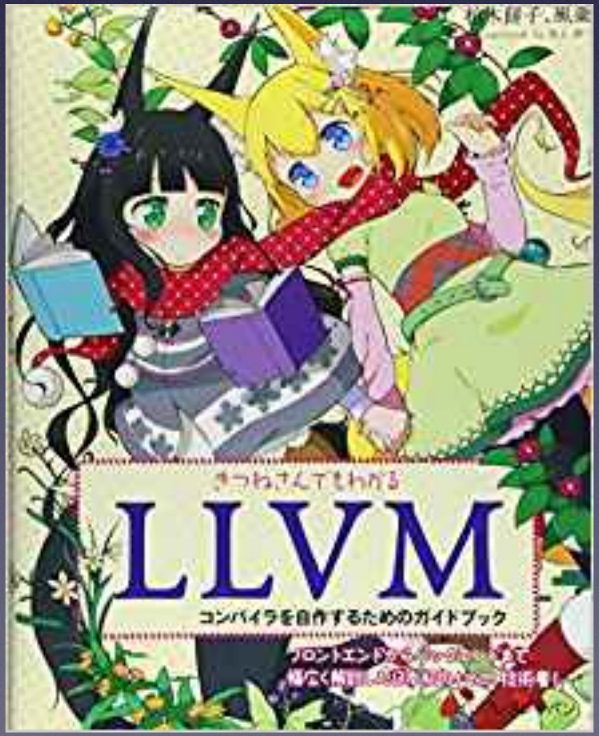
きつねさん以下からはじめた LLVM



Kohei Asano (@asakokok)

おしながき

はじめに

- whoami
  - 北大数学B3 (留年*1 + 休学*1)
  - Python, React.js, Firebase チョットデキル
  - サカダチ デキル, EDM スキ
- LLVMに興味を持ったきっかけ
  - Turing Complete FM (https://turingcomplete.fm/)
    - ポッドキャストの主がLLD (Linker)のMaintainer
    - 同い年の人(UT)がLLVMのDeveloper's Meetingでプレゼンしてる！
    - https://www.youtube.com/watch?v=zLPwPdZBpSY

LLVM?

- コンパイラ基盤オープンソースプロジェクトの総称

-

LLVM?

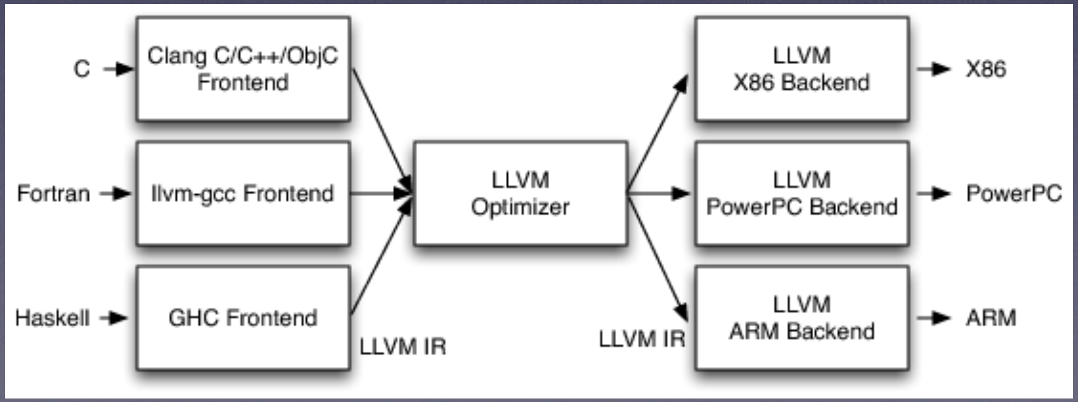
- コンパイラ基盤オープンソースプロジェクトの総称
    - Clang, LLVM-Core, LLD(Linker),… etc
    -

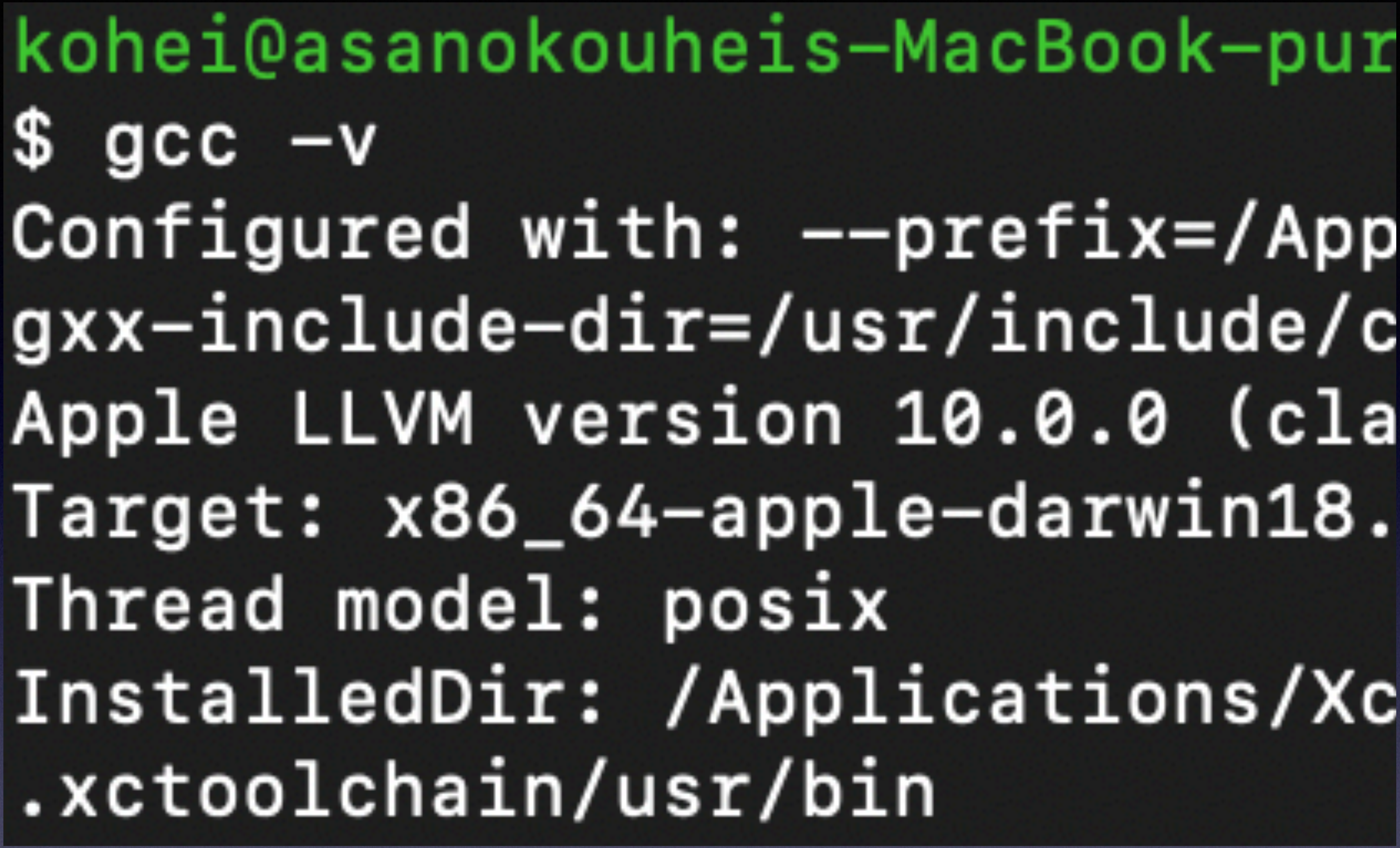
LLVM?

- コンパイラ基盤オープンソースプロジェクトの総称
    - Clang, LLVM-Core, LLD(Linker),… etc
- 多言語→多CPU
    -

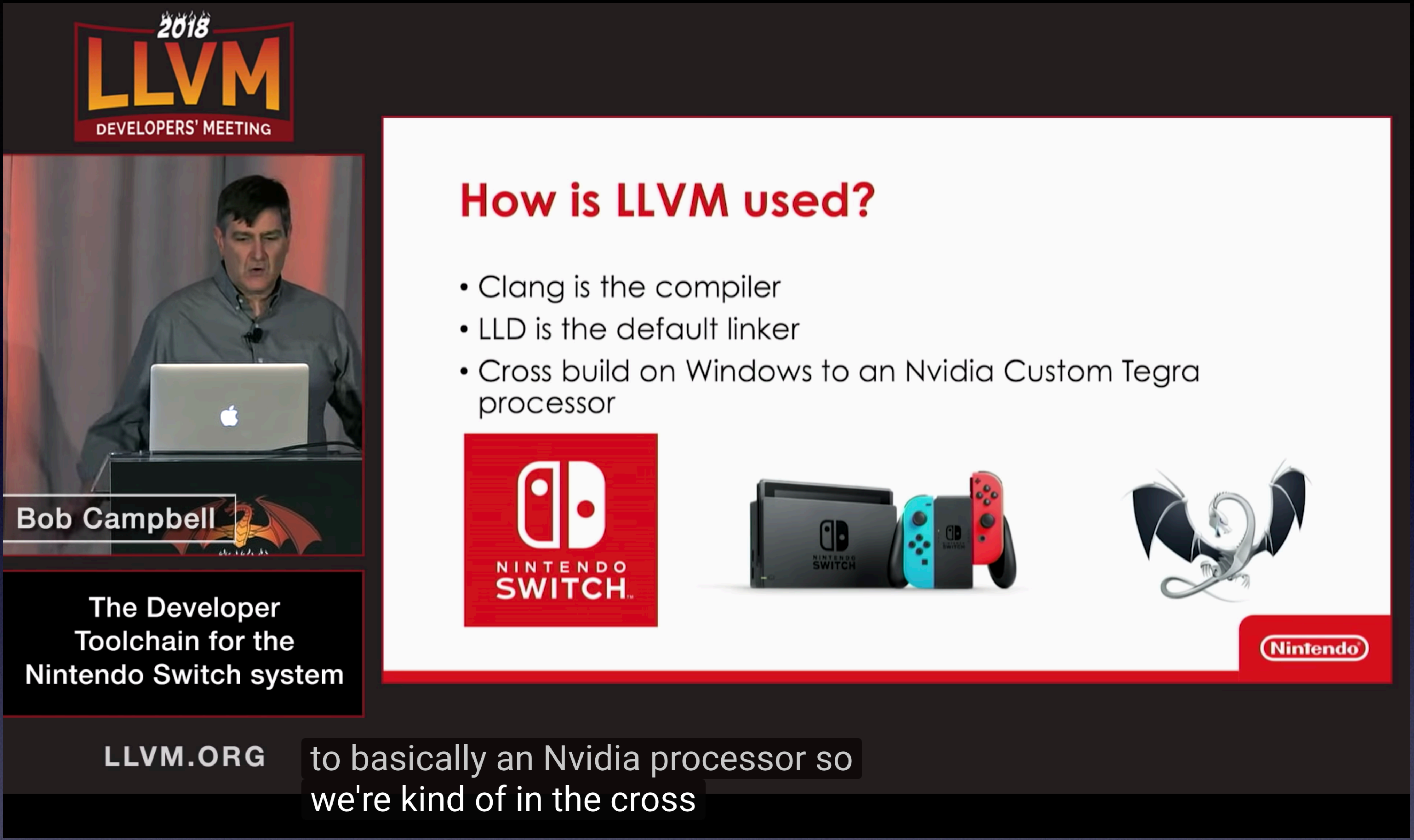
LLVM?

- コンパイラ基盤オープンソースプロジェクトの総称
  - Clang, LLVM-Core, LLD(Linker),… etc
- 多言語→多CPU
  - フロントエンド (言語→LLVM-IR)
  - ミドルエンド (最適化)
  - バックエンド(IR→object)





```
kohei@asanokouheis-MacBook-pur
$ gcc -v
Configured with: --prefix=/App
gxx-include-dir=/usr/include/c
Apple LLVM version 10.0.0 (cla
Target: x86_64-apple-darwin18.
Thread model: posix
InstalledDir: /Applications/Xc
.xctoolchain/usr/bin
```



How is LLVM used?

- Clang is the compiler
- LLD is the default linker
- Cross build on Windows to an Nvidia Custom Tegra processor

to basically an Nvidia processor so
we're kind of in the cross

https://youtu.be/9_7exO60EA8

LLVM-IR よみはじめ

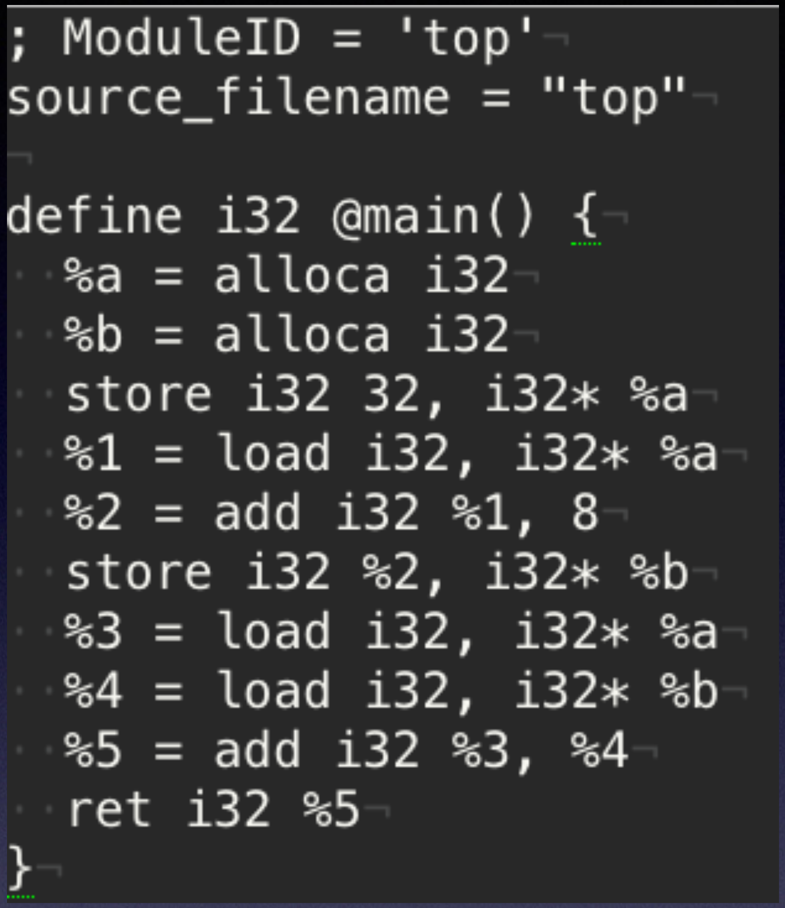
- LLVM-IR?

    -

LLVM-IR よみはじめ

- LLVM-IR?

    -

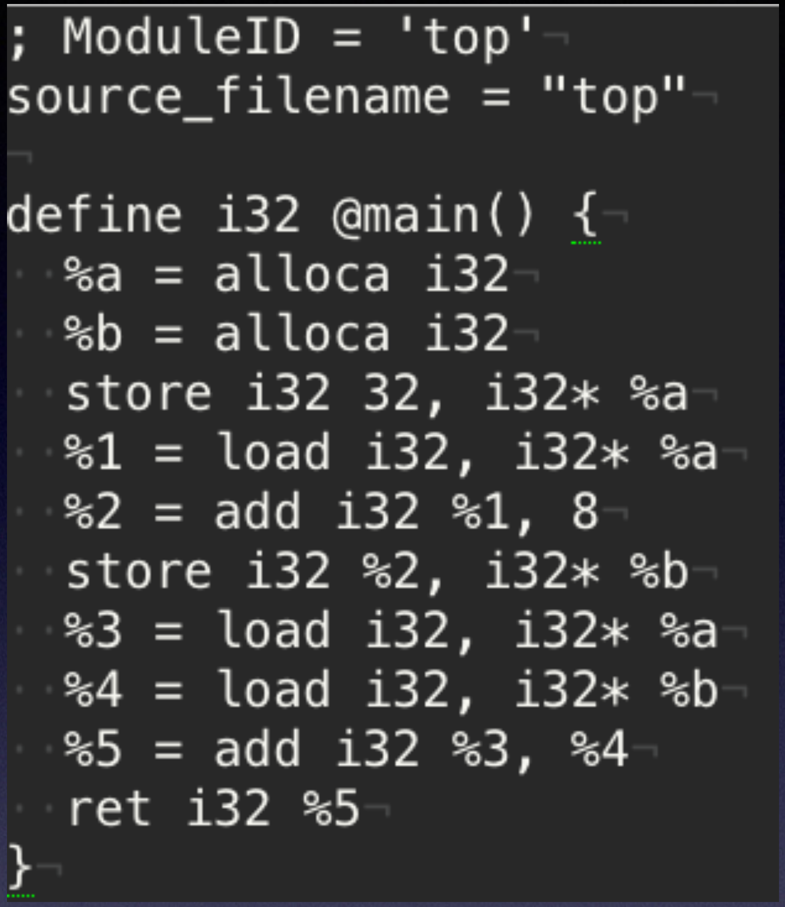


```
; ModuleID = 'top'
source_filename = "top"

define i32 @main() {
  %a = alloca i32
  %b = alloca i32
  store i32 32, i32* %a
  %1 = load i32, i32* %a
  %2 = add i32 %1, 8
  store i32 %2, i32* %b
  %3 = load i32, i32* %a
  %4 = load i32, i32* %b
  %5 = add i32 %3, %4
  ret i32 %5
}
```

LLVM-IR よみはじめ

- LLVM-IR?
  - 型付きAsm

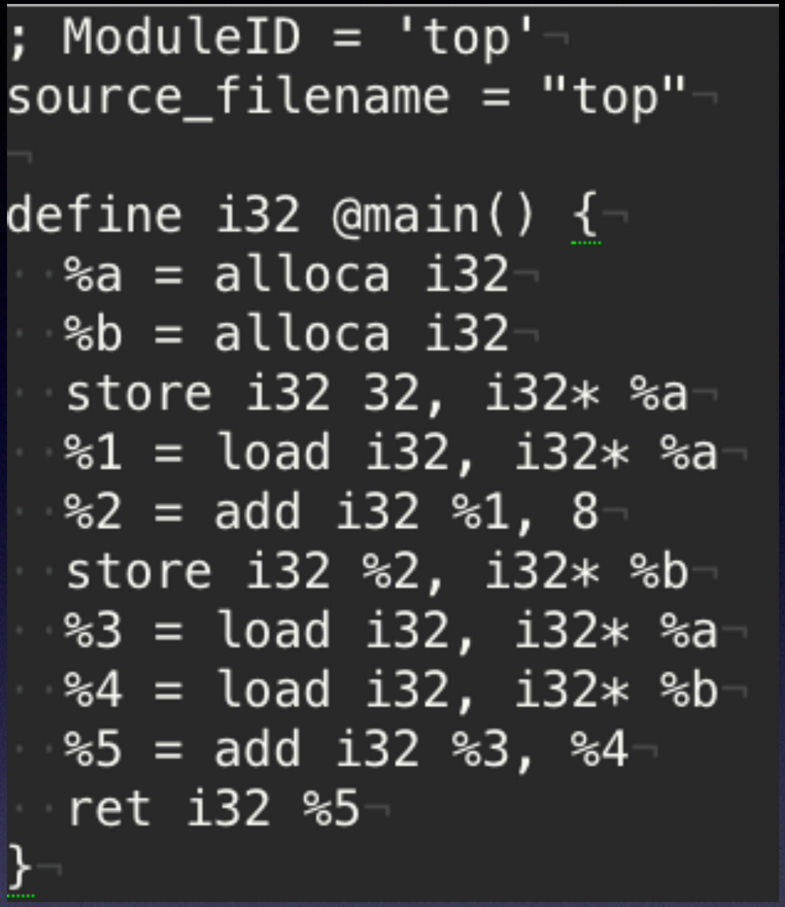
-



```
; ModuleID = 'top'
source_filename = "top"

define i32 @main() {
  %a = alloca i32
  %b = alloca i32
  store i32 32, i32* %a
  %1 = load i32, i32* %a
  %2 = add i32 %1, 8
  store i32 %2, i32* %b
  %3 = load i32, i32* %a
  %4 = load i32, i32* %b
  %5 = add i32 %3, %4
  ret i32 %5
}
```

LLVM-IR よみはじめ

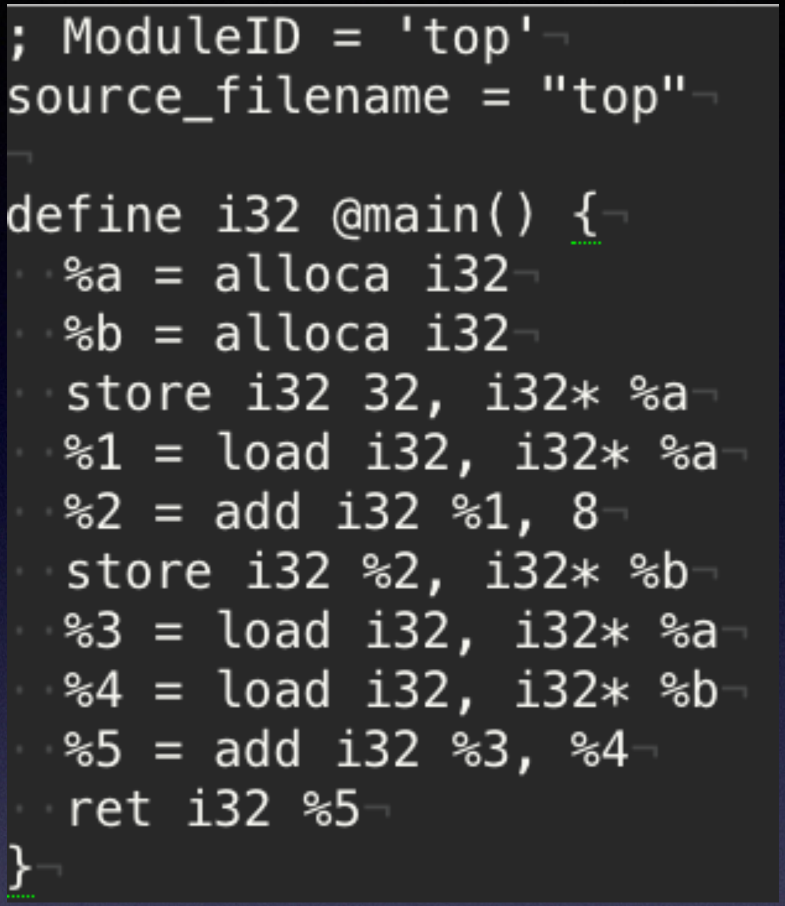
- LLVM-IR?
  - 型付きAsm
  - SSA



```
; ModuleID = 'top'
source_filename = "top"

define i32 @main() {
  %a = alloca i32
  %b = alloca i32
  store i32 32, i32* %a
  %1 = load i32, i32* %a
  %2 = add i32 %1, 8
  store i32 %2, i32* %b
  %3 = load i32, i32* %a
  %4 = load i32, i32* %b
  %5 = add i32 %3, %4
  ret i32 %5
}
```

LLVM-IR よみはじめ

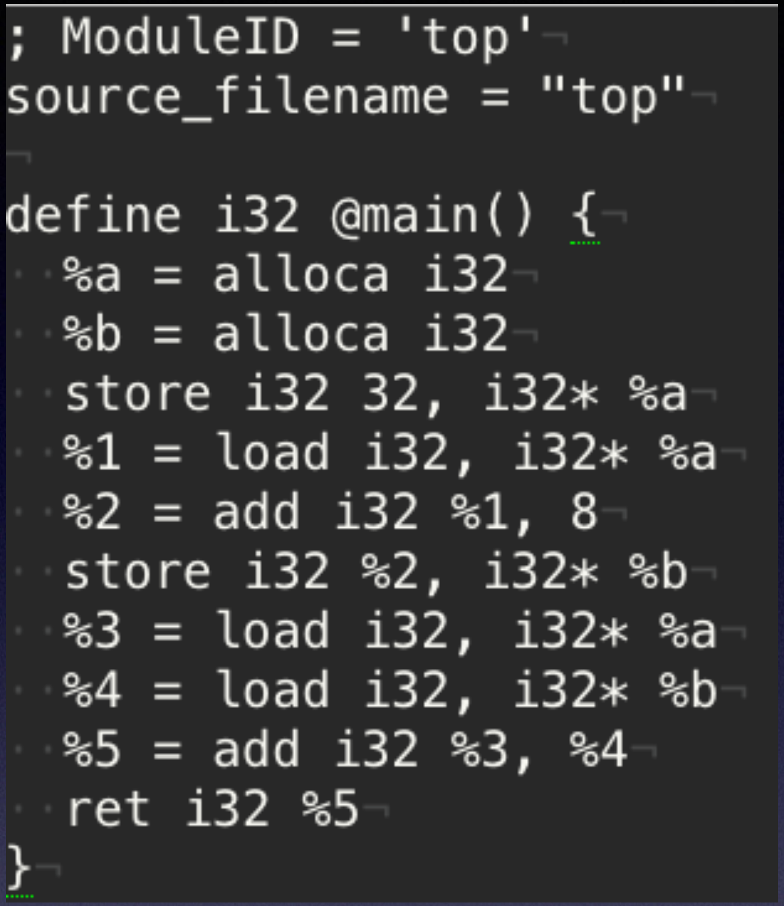
- LLVM-IR?
  - 型付きAsm
  - SSA
    - 各レジスタ(%x)への代入は一度だけ
    -



```
; ModuleID = 'top'
source_filename = "top"

define i32 @main() {
  %a = alloca i32
  %b = alloca i32
  store i32 32, i32* %a
  %1 = load i32, i32* %a
  %2 = add i32 %1, 8
  store i32 %2, i32* %b
  %3 = load i32, i32* %a
  %4 = load i32, i32* %b
  %5 = add i32 %3, %4
  ret i32 %5
}
```

LLVM-IR よみはじめ

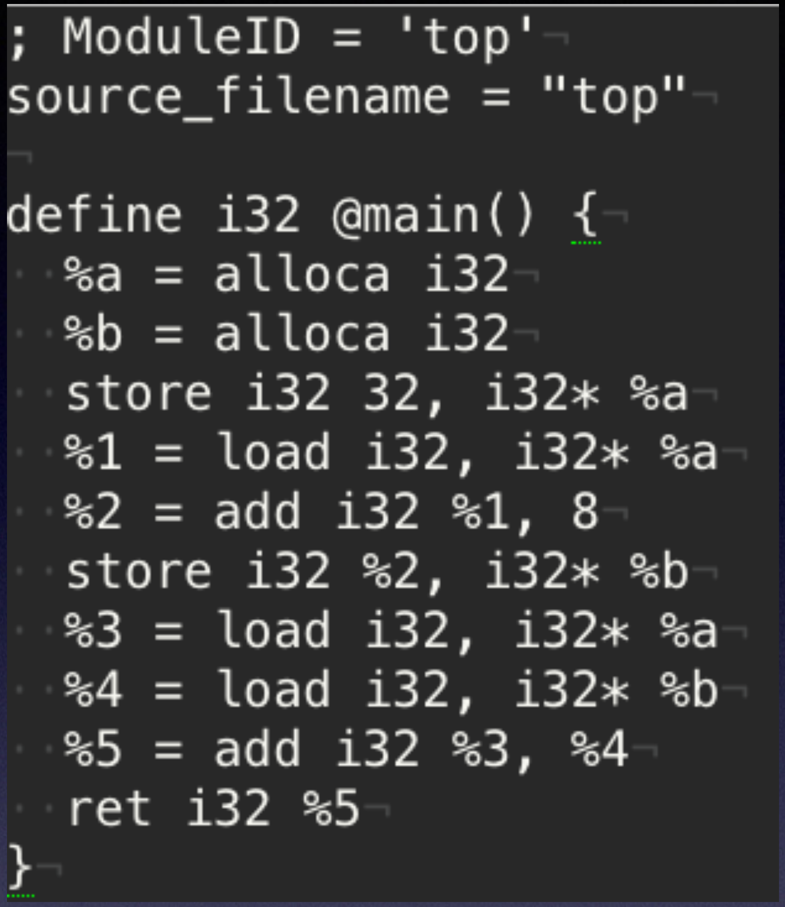
- LLVM-IR?
  - 型付きAsm
  - SSA
    - 各レジスタ(%x)への代入は一度だけ
  - 翻訳単位はModule
    -



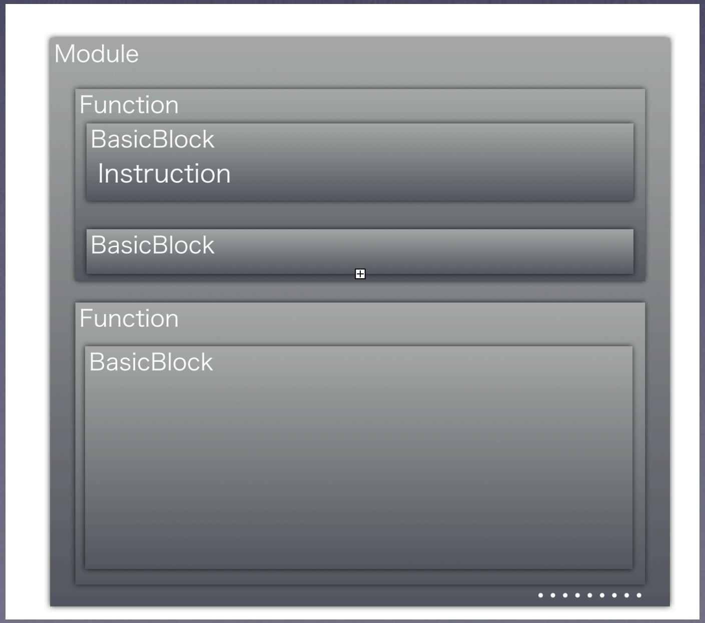
```
; ModuleID = 'top'
source_filename = "top"

define i32 @main() {
  %a = alloca i32
  %b = alloca i32
  store i32 32, i32* %a
  %1 = load i32, i32* %a
  %2 = add i32 %1, 8
  store i32 %2, i32* %b
  %3 = load i32, i32* %a
  %4 = load i32, i32* %b
  %5 = add i32 %3, %4
  ret i32 %5
}
```

LLVM-IR よみはじめ

- LLVM-IR?
  - 型付きAsm
  - SSA
    - 各レジスタ(%x)への代入は一度だけ
  - 翻訳単位はModule
    - Function, BasicBlock, Instruction

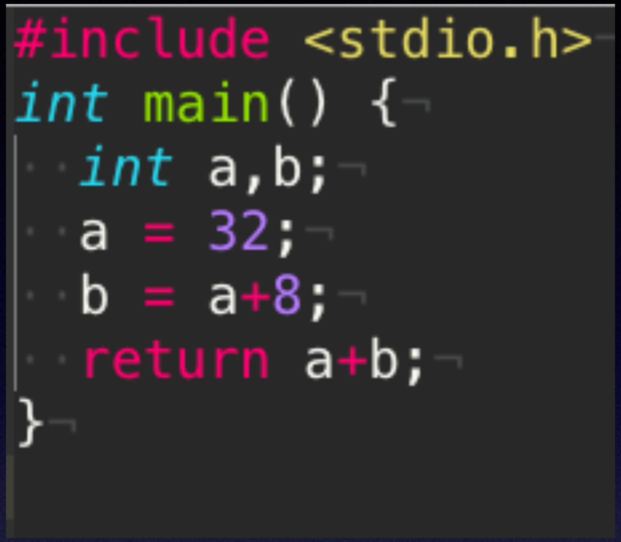


```
; ModuleID = 'top'
source_filename = "top"

define i32 @main() {
  %a = alloca i32
  %b = alloca i32
  store i32 32, i32* %a
  %1 = load i32, i32* %a
  %2 = add i32 %1, 8
  store i32 %2, i32* %b
  %3 = load i32, i32* %a
  %4 = load i32, i32* %b
  %5 = add i32 %3, %4
  ret i32 %5
}
```



LLVM-IR よみはじめ

C -> IR

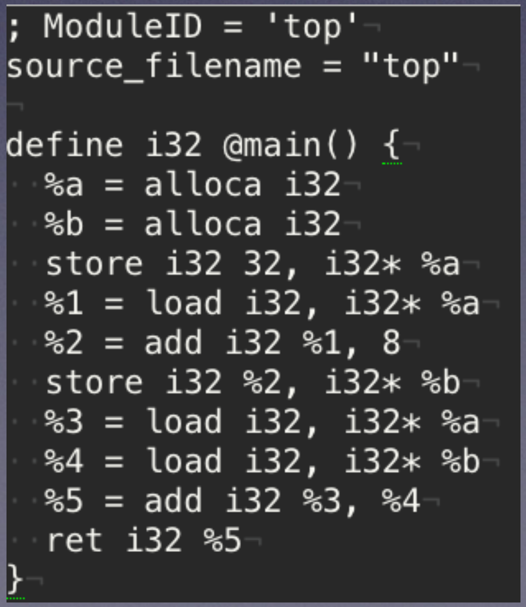


```c
#include <stdio.h>
int main() {
    int a,b;
    a = 32;
    b = a+8;
    return a+b;
}
```

clang

$ clang -emit-llvm -S -o test test.c

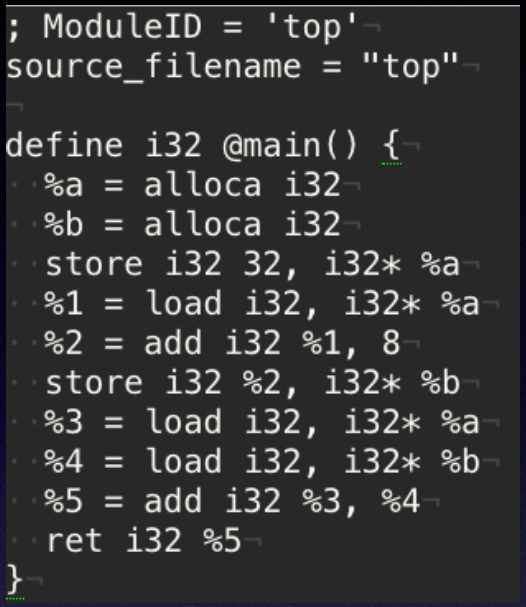
実行



```llvm
; ModuleID = 'top'
source_filename = "top"

define i32 @main() {
  %a = alloca i32
  %b = alloca i32
  store i32 32, i32* %a
  %1 = load i32, i32* %a
  %2 = add i32 %1, 8
  store i32 %2, i32* %b
  %3 = load i32, i32* %a
  %4 = load i32, i32* %b
  %5 = add i32 %3, %4
  ret i32 %5
}
```

LLVM-IR よみはじめ

IR ->



```
; ModuleID = 'top'
source_filename = "top"

define i32 @main() {
  %a = alloca i32
  %b = alloca i32
  store i32 32, i32* %a
  %1 = load i32, i32* %a
  %2 = add i32 %1, 8
  store i32 %2, i32* %b
  %3 = load i32, i32* %a
  %4 = load i32, i32* %b
  %5 = add i32 %3, %4
  ret i32 %5
}
```

lli

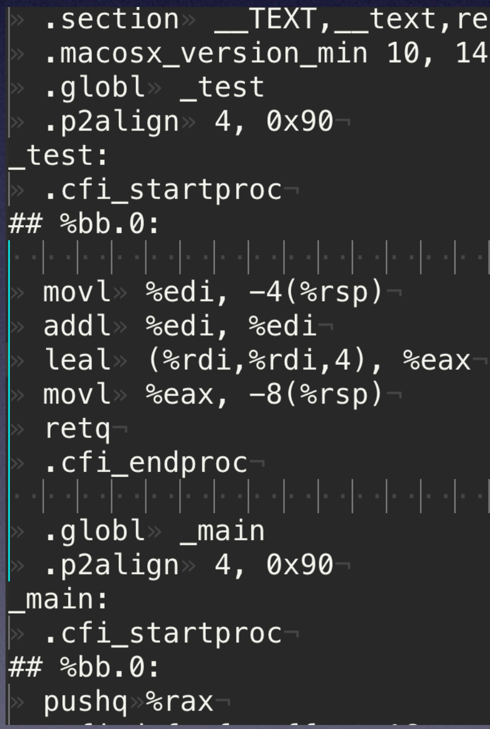
$ lli tmp.ll

実行

llc

$ llc tmp.ll

Asm

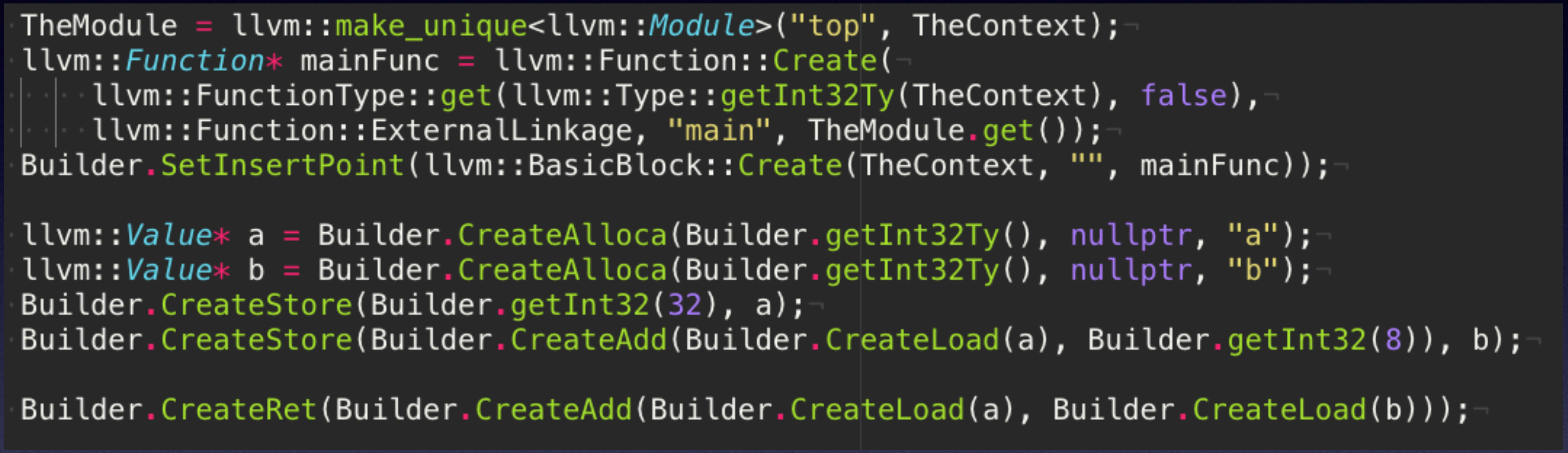


```
	.section	__TEXT,__text,re
	.macosx_version_min 10, 14
	.globl	_test
	.p2align	4, 0x90
_test:
	.cfi_startproc
## %bb.0:
	movl	%edi, -4(%rsp)
	addl	%edi, %edi
	leal	(%rdi,%rdi,4), %eax
	movl	%eax, -8(%rsp)
	retq
	.cfi_endproc

	.globl	_main
	.p2align	4, 0x90
_main:
	.cfi_startproc
## %bb.0:
	pushq	%rax
```

LLVM-IR よみはじめ

API

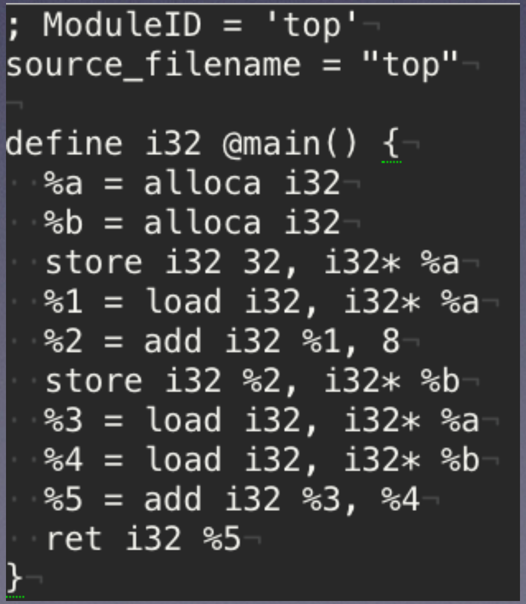
Cpp



```cpp
TheModule = llvm::make_unique<llvm::Module>("top", TheContext);
llvm::Function* mainFunc = llvm::Function::Create(
        llvm::FunctionType::get(llvm::Type::getInt32Ty(TheContext), false),
        llvm::Function::ExternalLinkage, "main", TheModule.get());
Builder.SetInsertPoint(llvm::BasicBlock::Create(TheContext, "", mainFunc));

llvm::Value* a = Builder.CreateAlloca(Builder.getInt32Ty(), nullptr, "a");
llvm::Value* b = Builder.CreateAlloca(Builder.getInt32Ty(), nullptr, "b");
Builder.CreateStore(Builder.getInt32(32), a);
Builder.CreateStore(Builder.CreateAdd(Builder.CreateLoad(a), Builder.getInt32(8)), b);

Builder.CreateRet(Builder.CreateAdd(Builder.CreateLoad(a), Builder.CreateLoad(b)));
```

```
$ g++ `llvm-config --cxxflags --ldflags --libs --system-libs` tmp.cpp -o tmp
$ ./tmp
```

IR



```llvm
; ModuleID = 'top'
source_filename = "top"

define i32 @main() {
  %a = alloca i32
  %b = alloca i32
  store i32 32, i32* %a
  %1 = load i32, i32* %a
  %2 = add i32 %1, 8
  store i32 %2, i32* %b
  %3 = load i32, i32* %a
  %4 = load i32, i32* %b
  %5 = add i32 %3, %4
  ret i32 %5
}
```

DummyCCompiler

https://github.com/Kmotiko/DummyCCompiler

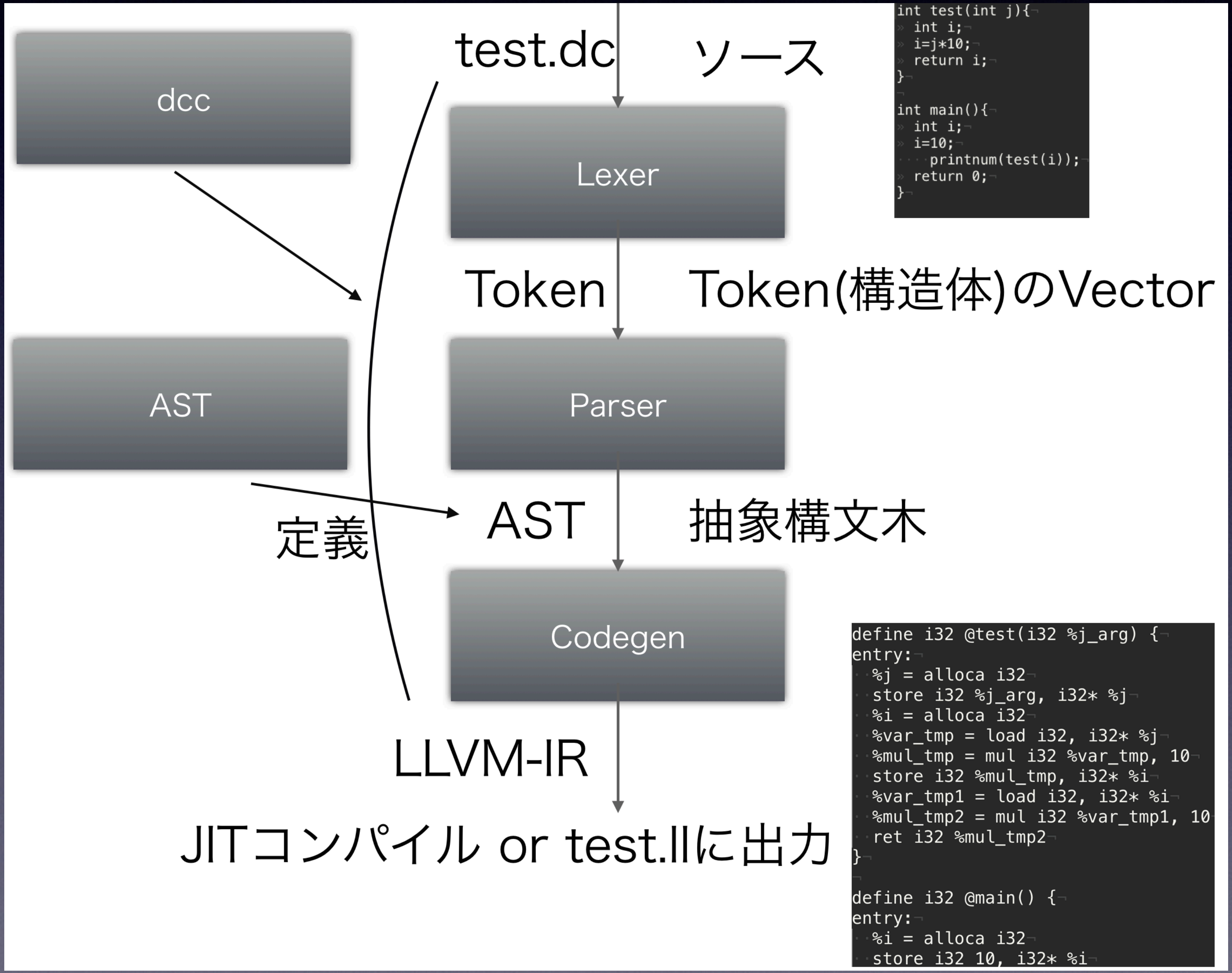
- C → LLVM IR

  -

DummyCCompiler

https://github.com/Kmotiko/DummyCCompiler

- C → LLVM IR



```c
int test(int j){
	int i;
	i=j*10;
	return i;
}

int main(){
	int i;
	i=10;
    printnum(test(i));
	return 0;
}
```

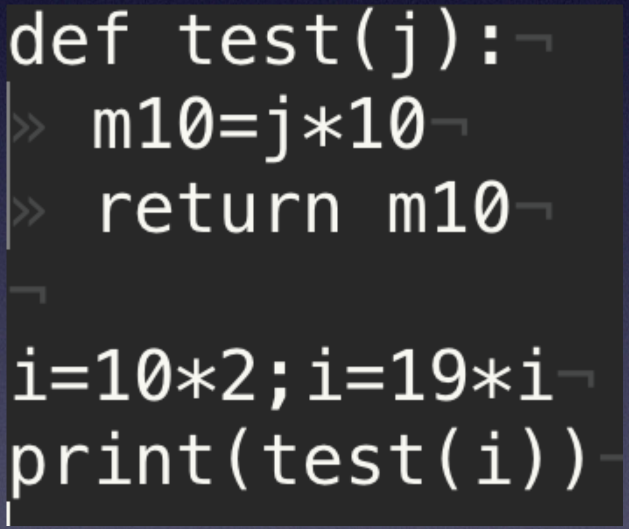
```llvm
define i32 @test(i32 %j_arg) {
entry:
  %j = alloca i32
  store i32 %j_arg, i32* %j
  %i = alloca i32
  %var_tmp = load i32, i32* %j
  %mul_tmp = mul i32 %var_tmp, 10
  store i32 %mul_tmp, i32* %i
  %var_tmp1 = load i32, i32* %i
  %mul_tmp2 = mul i32 %var_tmp1, 10
  ret i32 %mul_tmp2
}

define i32 @main() {
entry:
  %i = alloca i32
  store i32 10, i32* %i
```

-

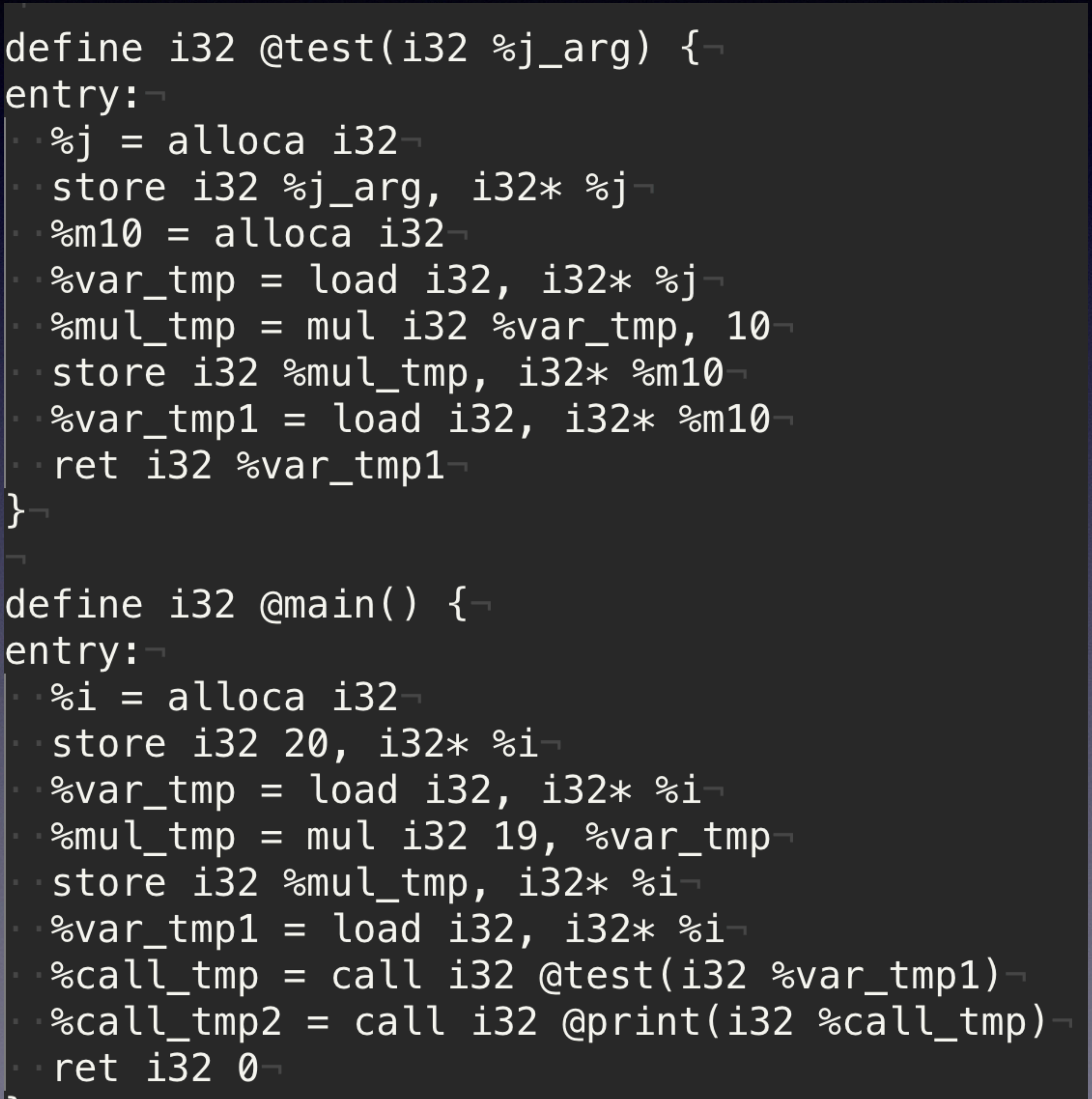
DummyPyCompiler

https://github.com/KoheiAsano/DummyPyCompiler

- Python → LLVM IR



```python
def test(j):
	m10=j*10
	return m10

i=10*2;i=19*i
print(test(i))
```



```llvm
define i32 @test(i32 %j_arg) {
entry:
  %j = alloca i32
  store i32 %j_arg, i32* %j
  %m10 = alloca i32
  %var_tmp = load i32, i32* %j
  %mul_tmp = mul i32 %var_tmp, 10
  store i32 %mul_tmp, i32* %m10
  %var_tmp1 = load i32, i32* %m10
  ret i32 %var_tmp1
}

define i32 @main() {
entry:
  %i = alloca i32
  store i32 20, i32* %i
  %var_tmp = load i32, i32* %i
  %mul_tmp = mul i32 19, %var_tmp
  store i32 %mul_tmp, i32* %i
  %var_tmp1 = load i32, i32* %i
  %call_tmp = call i32 @test(i32 %var_tmp1)
  %call_tmp2 = call i32 @print(i32 %call_tmp)
  ret i32 0
```

参考

- BrainFuckのコンパイラをLLVMで作る
https://itchyny.hatenablog.com/entry/2017/03/06/100000
- The Architecture of Open Source Applications
http://www.aosabook.org/en/llvm.html
- LLVM tut
https://llvm.org/docs/tutorial/
-

その他春休みやったこと

- 新田くんとHUIT勉強会ハッカソン(React, Redux, Firebaseで読書あぷり)

https://github.com/KoheiAsano/bookapp

- クックパッドSwiftCompiler1Dayインターン

https://github.com/KoheiAsano/MinSwift-workshop

- 技術書展(Pythonコンパイラ)

- AtCoder Problemsでちょっと精進(早解き芸人)

https://github.com/KoheiAsano/asako-atcoder-problems

- TCFM聴きながらヒッチハイク旅行！

https://turingcomplete.fm/
https://www.instagram.com/kohei_asaano/?hl=en

今年度前期がんばりたいこと

- HCPCアジア予選にいきたい！AtCoderで蒼くなりたい
- SecHackに応募する。その他インターンに多めに応募する
- バイトの給料をあげたい！
- 片手で逆立ち出来るようになりたい！
- 情エレの授業でちゃんと勉強したい！