伪SSA形式

所有的变量都通过alloca分配到栈上，当变量在等式被使用时通过load参与后续的计算，当变量被赋值时通过store来赋值。

基本块内IR的生成

基本块内IR的生成较为简单，许多代码都有直接对应的LLVMIR。

变量的定义可以翻译为alloca指令，再通过store指令来赋初值。

```
let t:int = 0;

%t = alloca i32
store i32 0, i32* %t
```

基本的四则运算可以直接翻译成对应的LLVMIR。

```
t = a + b + c;

%a_t = load i32,i32* %a
%b_t = load i32,i32* %b
%temp = add i32 %a_t, %b_t
%c_t = load i32,i32* %c
%t_t = add i32 %temp, %c_t
store i32 %t_t,i32* %t
```

函数调用可以先将所有函数形参求值翻译为对应的LLVMIR，再把函数调用翻译为对应的call指令。

```
fn sum(a:int,b:int)->int;
t = sum(a,b+1);

%a_t = load i32,i32* %a
%b_t = load i32,i32* %b
%temp = add i32 %b_t, 1
%t_t = call i32 @sum(%a_t,%temp)
store i32 %t_t,i32* %t
```

控制流的翻译

if-else可以翻译为下面的LLVMIR模式：先通过icmp指令计算if-else中条件表达式的值，在根据该值跳转到对应的label。if-else翻译为LLVMIR要生成3个label，分别为if_true,if_false,if_end。当条件表达式为真时跳转到if-true，为假时跳转到if-false，if_true,if_false两个基本块最后要跳转到if-end。

```
%t = icmp ......
br i1 %t, label %if_true,label %if_false
if_true:
......
br label if_end
if_false:
......
br label if_end
if_end:
```

while的翻译和if-else的翻译类似，先通过icmp指令计算条件表达式的值，在根据该值跳转到对应的label。while翻译为LLVMIR要生成3个label，分别为while_true,while_false,while_test。当条件表达式为真时跳转到while-true，为假时跳转到while-false，while_true最后要跳转到while-test，while_test在计算条件表达式的值的前面。

```
while_test:
%t = icmp ......
br i1 %t, label %while_true,label %while_false
while_true:
......
br label while_test
while_false:
```

!(not)的翻译：LLVMIR中并没有直接的取反指令，翻译!可以通过xor指令。值得注意的是!的参数类型都是bool类型，而使用bool类型时都可以生成两个对应跳转的label来完成翻译，所以交换这两个label也可以完成!的翻译。

```
if(!(a==0))

%t = icmp eq i32 %a_t,0
%t_not = xor i1 %t,1
br i1 t_not, label %if_true, label if_false

%t = icmp eq i32 %a_t,0
br i1 t, label %if_false, label if_true
```

&&和||的翻译：LLVMIR中只有bitwise的and和or，而没有逻辑and和or，所以&&和||不能直接翻译为对应的LLVMIR。&&和||的翻译可以通过先计算&&和||左边的表达式，在根据左边的值跳转到计算右边的值的label，或者if-else和while生成的label。

```
if((a==0)||(b==1))

%t_l = icmp eq %a_t,0
br i1 %t_l,label if_true,label or_false
or_false:
%t_r = icmp eq %b_t,1
br i1 %t_r,label if_true,label if_false
```

```
while((a==0)&&(b==1))

while_test:
%t_l = icmp eq %a_t,0
br i1 %t_l,label and_true,label while_false
and_true:
%t_r = icmp eq %b_t,1
br i1 %t_r,label while_true,label while_false
```