

## C1 扩展

PB13011054 陈远昭

1. 可以支持 int, float 类型, 它们都会有常量控制. 我所理解的常量控制的含义是, 对于赋值语句  $a = b$  而言, 若  $a$  是 const 变量, 则  $b$  必须也是 const 变量或者其他常量表达式; 而如果  $a$  不是 const 变量, 那么  $b$  就没有限制. 对于 const 变量还需要有一些其他的控制, 比如不能在后续中被赋值. 在 llvm 的 IR 中并没有体现局部的 const 变量. 全局的 const 变量有 constant 修饰. 这样就需要单独进行一些控制, 比如记录下所有的 const 变量, 在赋值语句中查看左值是否为 const.

```
→ P8 git:(finalcopy) X cat test/t1.c1
int main(){
    const float ap = 9.0;
    const float fp = ap;
    return 0;
}
→ P8 git:(finalcopy) X ./run.sh
rsrse done
; ModuleID = 'my cool jit'

define i32 @main() {
entry:
    %ap = alloca double
    store double 9.000000e+00, double* %ap
    %ap1 = load double* %ap
    %fp = alloca double
    store double %ap1, double* %fp
    ret i32 0
}
```

2. 可以有数组，其中数组的长度可以由常量决定，也可以由 const 变量决定，如果该项未明确，则有后面的初始化列表长度来决定。

```
→ P8 git:(finalcopy) X cat test/t1.c1
int main(){
    const int a = 4;
    float f[a];
    float g[3] = {1, 2, 3};
    return 0;
}
→ P8 git:(finalcopy) X ./run.sh
psrse done
; ModuleID = 'my cool jit'

define i32 @main() {
entry:
    %a = alloca i32
    store i32 4, i32* %a
    %f = alloca [4 x double]
    %g = alloca [3 x double]
    %0 = getelementptr [3 x double]* %g, i32 0, i32 0
    store double 1.000000e+00, double* %0
    %1 = getelementptr double* %0, i32 1
    store double 2.000000e+00, double* %1
    %2 = getelementptr double* %1, i32 1
    store double 3.000000e+00, double* %2
    ret i32 0
}
```

3. If 条件判断和 while 循环是 P7 的要求，其中加入了 break/continue 语句，允许从深层跳出来。

```
entry:
    %i = alloca i32
    store i32 0, i32* %i
    br label %whilecond

whilecond:                                ; preds = %ifcont, %entry
    %i1 = load i32* %i
    %ltmp = icmp slt i32 %i1, 5
    br i1 %ltmp, label %whileloop, label %whilet

whileloop:                                ; preds = %whilecond
    %i2 = load i32* %i
    %eqtmp = icmp eq i32 %i2, 2
    br i1 %eqtmp, label %then, label %else

then:                                     ; preds = %whileloop
    br label %whilet

after_break:                              ; No predecessors!
    br label %ifcont

else:                                     ; preds = %whileloop
    br label %ifcont

ifcont:                                   ; preds = %else, %after_break
    br label %whilecond

whilet:                                   ; preds = %then, %whilecond
    ret i32 0
}
→ P8 git:(finalcopy) X cat test/t1.c1
int main(){
    int i = 0;
    while(i < 5){
        if(i == 2)
            break;
    }
    return 0;
}
```

4 . extern 变量及函数，最开始是用于静态库的打印函数

```
→ P8 git:(finalcopy) X cat test/t1.c1
extern int globalint;
extern void printint();

int main(){
    return 0;
}

→ P8 git:(finalcopy) X ./run.sh
borsse done
; ModuleID = 'my cool jit'

@globalint = external global i32

declare void @printint()

define i32 @main() {
entry:
    ret i32 0
}
```

5. 支持带参数的函数调用和返回值，返回值可以是 int，float 类型

```
extern int globalint;
extern void printinta(int num);

int lmax(int a, int b){
    if(a > b)
        return a;
    return b;
}
→ P8 git:(finalcopy) X ./run.sh
psrse done
; ModuleID = 'my cool jit'

@globalint = external global i32

declare void @printinta(i32)

define i32 @lmax(i32 %a, i32 %b) {
entry:
    %a1 = alloca i32
    store i32 %a, i32* %a1
    %b2 = alloca i32
    store i32 %b, i32* %b2
    %a3 = load i32* %a1
    %b4 = load i32* %b2
    %gttmp = icmp sgt i32 %a3, %b4
    br i1 %gttmp, label %then, label %else

then:                                     ; preds = %entry
    %a5 = load i32* %a1
    ret i32 %a5
    br label %ifcont

else:                                     ; preds = %entry
    br label %ifcont

ifcont:                                   ; preds = %else, %then
    %b6 = load i32* %b2
    ret i32 %b6
}
```

6. 支持指针类型,即可以使用 \* 作为指针声明的标志,可以使用 & 取变量的地址,同时也允许指针之间的赋值.能够取数组某一元素的地址给指针变量赋值.当 \* 作用于指针变量能够取其所指向的值.

```
int main(){
    int a = 0;
    int b[5];
    int *p;
    p = &a;
    p = &b[3];
}

→ P8 git:(finalcopy) X ./run.sh
psrse done
; ModuleID = 'my cool jit'

define i32 @main() {
entry:
    %a = alloca i32
    store i32 0, i32* %a
    %b = alloca [5 x i32]
    %p = alloca i32*
    store i32* %a, i32** %p
    %0 = getelementptr [5 x i32]* %b, i32 0, i32 3
    store i32* %0, i32** %p
}
```

```
→ P8 git:(finalcopy) X ./run.sh
psrse done
; ModuleID = 'my cool jit'

define i32 @main() {
entry:
    %p = alloca i32*
    %q = alloca i32*
    %q1 = load i32** %q
    store i32* %q1, i32** %p
    %0 = load i32** %p
    %1 = load i32* %0
    %a = alloca i32
    store i32 %1, i32* %a
}

→ P8 git:(finalcopy) X cat test/t1.c1
int main(){
    int *p, *q;
    p = q;
    int a =*p;
}
```



7 . 指针可以作为函数的参数 ,也能作为返回值( 好像 llvm 中 PointerType 和 ArrayType

并不能在某种程度上统一 ,因此我一直希望能通过数组的 GEP 方式访问指针的方法失败了 ,

那就把这里的指针当成仅仅作为能指向某个类型元素地址的类型吧 )

```
define double* @f(i32* %p) {  
entry:  
    %p1 = alloca i32*  
    store i32* %p, i32** %p1  
    %0 = load i32** %p1  
    %1 = load i32* %0  
    %a = alloca i32  
    store i32 %1, i32* %a  
    %q = alloca double*  
    %q2 = load double** %q  
    ret double* %q2  
}  
→ P8 git:(finalcopy) X cat test/t1.c1  
float *f(int *p){  
    int a = *p;  
    float *q ;  
    return q;  
}
```

8. 支持结构体类型，这里的结构体类型需要在函数外声明作为一个全局类型。结构体元素支持了 int, float, int \*, float \*, int [], float [], 还有 struct \* ( 这样能够构成类似于链表之类的东西)。在结构体中可以用 '.' 来获取某一个元素。对结构体的赋值也可以直接对整个结构体赋值，也可以为为结构体的某一项赋值。( 为了赋值的方便，在这里允许给指针赋值为 0，表明它是一个空指针)

```
%A = type { i32, double* }
@0 = private constant %A { i32 2, double* null }

define i32 @main() {
entry:
    %lp = alloca %A
    %lq = alloca %A
    %cast = bitcast %A* %lp to i8*
    call void @llvmmemcpy(i8* %cast, i8* bitcast (%A* @0 to i8*), i64 12, i32 8, i1 false)
    %lp1 = load %A* %lp
    store %A %lp1, %A* %lq
}
→ P8 git:(finalcopy) Xcat test/t1.c1
struct A{
    int num;
    float *fp;
};

int main(){
    struct A lp;
    struct A lq;
    lp = { 2, 0 };
    lq = lp;
}
```



9 . 指针能够指向结构体，和其他类型一样都能使用 \* , & 操作符，但是由于不能支持 -> 操作，因此有很多东西无法实现。结构体也能做函数的参数。

```
%A = type { i32, %A* }

define i32 @main() {
entry:
    %lp = alloca %A*
    %lq = alloca %A
    store %A* %lq, %A** %lp
}
→ P8 git:(finalcopy) Xcat test/t1.c
struct A{
    int num;
    struct A *fp;
};

int main(){
    struct A *lp;
    struct A lq;
    lp = &lq;
}
```

```
%A = type { i32, %A* }

define void @f(%A* %hk) {
entry:
    %hk1 = alloca %A*
    store %A* %hk, %A** %hk1
    ret void
}
→ P8 git:(finalcopy) Xcat test/t1.c1
struct A{
    int num;
    struct A *fp;
};

void f(struct A *hk){
}
```

10. 可以使用结构体数组，但是没有考虑嵌套情况，即结构体中没有考虑其他结构体作为其中的一项，数组中没有考虑高维数组。

```
%A = type { i32, %A* }

define void @f(%A* %hk) {
entry:
    %hk1 = alloca %A*
    store %A* %hk, %A** %hk1
    %apq = alloca [10 x %A]
    %sr = alloca %A
    %sr2 = load %A* %sr
    %0 = getelementptr [10 x %A]* %apq, i64 0, i32
    store %A %sr2, %A* %0
    ret void
}

→ P8 git:(finalcopy) X cat test/t1.c1
struct A{
    int num;
    struct A *fp;
};

void f(struct A *hk){
    struct A apq[10];
    struct A sr;

    apq[3] = sr;
}

→ P8 git:(finalcopy) X scs
```

两个 Example

1.