1.实验说明

实验五采用了hby同学的框架代码，在理解的基础上进行一定的填写

2.活跃变量

```
LiveVariableAnalysis_isForward (LiveVariableAnalysis *t) {
    // TODO: return isForward?;
    //TODO();
    return false;
}
```

后向的分析

```
LiveVariableAnalysis_meetInto (LiveVariableAnalysis *t,
                               Set_IR_var *fact,
                               Set_IR_var *target) {
    /* TODO:
     * meet: union/intersect?
     * return VCALL(*target, union_with/intersect_with?, fact);
     */
    //TODO();
    return VCALL(*target, union_with, fact);
}
```

、

在进行IN的处理时，需要使用联合起来

```
  //TODO();
  if(def != IR_VAR_NONE) { // 生成新def
        VCALL(*fact, delete, def); // kill/gen ?
  }
  for(unsigned i = 0; i < use.use_cnt; i ++) {
        IR_val use_val = use.use_vec[i];
        if(!use_val.is_const) {
            IR_var use = use_val.var;
            VCALL(*fact, insert, use); // kill/gen ?
        }
  }
```

对于每一句stmt的翻译：采用先生成新def再kill掉这个def，之后操作use再加入进来即可

```
if(VCALL(*new_out_fact, exist, def) == false) {
    stmt->dead = true;
    updated = true;
}
```

对于死代码的消除，我们再现存的找不到这个def即可判断已死，update更新即可

3.其他的函数

基本根据hby同学的todo来做

主要是按照手册的三种类型进行对应的处理即可，很容易

典型的todo

常量传播的meet与calu

```
static CPValue meetValue(CPValue v1, CPValue v2) {
    /* TODO
     * 计算不同数据流数据汇入后变量的CPValue的meet值
     * 要考虑 UNDEF/CONST/NAC 的不同情况
     */
    //TODO();
    switch (v1.kind)
    {
    case UNDEF: return v2;
    case CONST:{
        if(v2.kind == UNDEF) return v1;
        else if(v2.kind == CONST){
            if(v2.const_val == v1.const_val) return v2;
            else return get_NAC();
        }
        else return get_NAC();
    }
    case NAC:{
        if(v2.kind == UNDEF) return v1;
        else return get_NAC();
    }
    }
}
```

```
static CPValue calculateValue(IR_OP_TYPE IR_op_type, CPValue v1, CPValue v2) {
    /* TODO
     * 计算二元运算结果的CPValue值
     * 要考虑 UNDEF/CONST/NAC 的不同情况
     * if(v1.kind == CONST && v2.kind == CONST) {
     *      int v1_const = v1.const_val, v2_const = v2.const_val;
     *      int res_const;
     *      switch (IR_op_type) {
     *          case IR_OP_ADD: res_const = v1_const + v2_const; break;
     *          case IR_OP_SUB: res_const = v1_const - v2_const; break;
     *          case IR_OP_MUL: res_const = v1_const * v2_const; break;
     *          case IR_OP_DIV:
     *              if(v2_const == 0) return get_UNDEF();
     *              res_const = v1_const / v2_const; break;
     *          default: assert(0);
     *      }
     *      return get_CONST(res_const);
     *  } ... 其他情况
     */
    //TODO();
    if(v1.kind == CONST && v2.kind == CONST) {
        int v1_const = v1.const_val, v2_const = v2.const_val;
        int res_const;
        switch (IR_op_type) {
            case IR_OP_ADD: res_const = v1_const + v2_const; break;
            case IR_OP_SUB: res_const = v1_const - v2_const; break;
            case IR_OP_MUL: res_const = v1_const * v2_const; break;
            case IR_OP_DIV:
                if(v2_const == 0) return get_UNDEF();
                res_const = v1_const / v2_const; break;
            default: assert(0);
        }
        return get_CONST(res_const);
    }
    if(v1.kind == UNDEF||v2.kind == UNDEF){
        return get_UNDEF();
    }
    return get_NAC();
}
```

4.问题

我认为自己的复制传播很有问题，但是确实无法理解hby同学的网站信息

总之很感谢hby同学，让我在减少代码量情况下也非常能理解