编译原理 PA4 实验报告

计62 徐晟 2016011253

1. 本阶段DU链的求解

首先按照要求，只修改了FlowGraph.java和BasicBlock.java两个文件。

在每个liveUse, liveIn和liveOut之后都加入了usePair, inPair和outPair来保存当前in和out的DUchain信息。

在FlowGraph.java中修改比较简单，每次对于liveUse, liveIn和liveOut的修改同时对于usePair, inPair和outPair同样的修改进行Tac语句调用的副本保存。

在BasicBlock.java中，首先和FlowGraph.java中类似，每次liveUse, liveIn和liveOut进行add操作的时候，usePair, inPair和outPair同样的进行add操作，同时在analyzeLiveness()函数后再重新对于tac运算符进行判定一次，这次判定的目的是为了计算DUchain，只要把原来修改操作对应的改成analyzeLiveness\_get\_DUChain()函数，也就是把当前TAC信息记录进DU链即可。

简单的来说DU链的求解只需要在原有的框架上加上几行简单的代码就可以了，只要把use, in, out信息记录下来，在tac访问的时候对应的记录输出即可。

1. T0.decaf分析

T0.deacf的DOUT文件如下：

FUNCTION \_Main\_New :

BASIC BLOCK 0 :

Def = [ \_T0 \_T1 \_T2 ]

liveUse = [ ]

liveIn = [ ]

liveOut = [ ]

\_T0 = 4 [ \_T0 ]

parm \_T0 [ ]

\_T1 = call \_Alloc [ \_T1 ]

\_T2 = VTBL <\_Main> [ \_T1 \_T2 ]

\*(\_T1 + 0) = \_T2 [ \_T1 ]

END BY RETURN, result = \_T1

FUNCTION main :

BASIC BLOCK 0 :

Def = [ ]

liveUse = [ ]

liveIn = [ ]

liveOut = [ ]

call \_Main.f [ ]

END BY RETURN, void result

FUNCTION \_Main.f :

BASIC BLOCK 0 :

Def = [ \_T3 \_T4 \_T5 \_T6 \_T7 \_T8 \_T9 \_T10 \_T11 \_T12 \_T13 ]

liveUse = [ ]

liveIn = [ ]

liveOut = [ \_T4 \_T9 ]

\_T7 = 0 [ \_T7 ]

\_T5 = \_T7 [ ]

\_T8 = 1 [ \_T8 ]

\_T6 = \_T8 [ ]

\_T10 = 0 [ \_T10 ]

\_T9 = \_T10 [ \_T9 ]

\_T11 = 2 [ \_T9 \_T11 ]

\_T3 = \_T11 [ \_T3 \_T9 ]

\_T12 = 1 [ \_T3 \_T9 \_T12 ]

\_T13 = (\_T3 + \_T12) [ \_T9 \_T13 ]

\_T4 = \_T13 [ \_T4 \_T9 ]

END BY BRANCH, goto 1

BASIC BLOCK 1 :

Def = [ ]

liveUse = [ \_T9 ]

liveIn = [ \_T4 \_T9 ]

liveOut = [ \_T4 \_T9 ]

END BY BEQZ, if \_T9 =

0 : goto 7; 1 : goto 2

BASIC BLOCK 2 :

Def = [ \_T3 \_T14 ]

liveUse = [ \_T9 ]

liveIn = [ \_T4 \_T9 ]

liveOut = [ \_T3 \_T4 \_T9 ]

\_T14 = 1 [ \_T4 \_T9 \_T14 ]

\_T3 = \_T14 [ \_T3 \_T4 \_T9 ]

END BY BEQZ, if \_T9 =

0 : goto 4; 1 : goto 3

BASIC BLOCK 3 :

Def = [ ]

liveUse = [ ]

liveIn = [ \_T3 \_T4 \_T9 ]

liveOut = [ \_T3 \_T4 \_T9 ]

call \_Main.f [ \_T3 \_T4 \_T9 ]

END BY BRANCH, goto 4

BASIC BLOCK 4 :

Def = [ \_T15 \_T16 ]

liveUse = [ \_T4 \_T9 ]

liveIn = [ \_T3 \_T4 \_T9 ]

liveOut = [ \_T3 \_T4 \_T9 ]

\_T15 = 1 [ \_T3 \_T4 \_T9 \_T15 ]

\_T16 = (\_T4 + \_T15) [ \_T16 \_T3 \_T9 ]

\_T4 = \_T16 [ \_T3 \_T4 \_T9 ]

END BY BEQZ, if \_T9 =

0 : goto 6; 1 : goto 5

BASIC BLOCK 5 :

Def = [ \_T17 \_T18 ]

liveUse = [ \_T4 ]

liveIn = [ \_T3 \_T4 \_T9 ]

liveOut = [ \_T3 \_T4 \_T9 ]

\_T17 = 4 [ \_T17 \_T3 \_T4 \_T9 ]

\_T18 = (\_T4 - \_T17) [ \_T18 \_T3 \_T9 ]

\_T4 = \_T18 [ \_T3 \_T4 \_T9 ]

END BY BRANCH, goto 6

BASIC BLOCK 6 :

Def = [ \_T5 \_T6 ]

liveUse = [ \_T3 \_T4 ]

liveIn = [ \_T3 \_T4 \_T9 ]

liveOut = [ \_T4 \_T9 ]

\_T5 = \_T3 [ \_T4 \_T9 ]

\_T6 = \_T4 [ \_T4 \_T9 ]

END BY BRANCH, goto 1

BASIC BLOCK 7 :

Def = [ ]

liveUse = [ ]

liveIn = [ ]

liveOut = [ ]

END BY RETURN, void result

验证与2.4.2中图5是一样的：

显然，这个程序中BLOCK0是Main的Block，Block1是寄存调用信息Block，Block7也是跳转Block，而MIPS中的Block23456对应着图5中的Block12345，而例如Block2中的goto4和3等也说明了这一点。