Übungsblatt 5

Kontroll- und Datenfluß – Gruppenarbeit (max. 3 Personen)

Abgabe: 13.6.00 vor 9:30h

1. (1 Punkt) Im Gnu C Compiler (GCC) wird die Reduktion von Operationskosten vor der Codegenerierung (d.h. auf dem von der syntaktischen und semantischen Analyse aufgebauten attributeriten Syntaxbaum) durchgeführt, während in VPO diese Optimierung auf dem Zwischencode (RTLs) durchgeführt wird. Erörtern Sie die Vor- und Nachteile der beiden Ansätze.

2. (6 Punkte) Gegen sei folgende C Funktion in RTLs. Konstruieren Sie den Kontrollflußgraphen und wenden Sie folgende Optimierungen schrittweise an, d.h. RTLs, Blöcke und Kontrollfluß sollen nach jeden einzelnen Optimierungen wiedergegeben werden, sofern sie sich verändert haben. Erklären Sie kurz, warum sich die Optimierung anwenden läßt. (vereinfachende Annahme: Sprungbefehle haben keine Delay-Slots.)

   (a) Eliminieren von Sprungketten
   (b) Eliminieren leerer Grundblöcke
   (c) Eliminieren unnötiger Sprünge
   (d) Eliminieren von totem Code
   (e) Umkehrung von Sprüngen
   (f) Fusion von Grundblöcken

```
+1      r[14]=SV[r[14]+-104];
+2      R[r[30]+.3_str]=r[24];
+3      R[r[30]+.3_n]=r[25];
+4      r[8]=R[r[30]+.3_str];
+5      R[r[14]+.3_s]=r[8];      r[8]
+6      r[8]=R[r[30]+.3_n];
+7      R[r[14]+.3_i]=r[8];      r[8]
    L84
+8      r[8]=R[r[14]+.3_i];
+9      r[9]=r[8]-1;
+10     R[r[14]+.3_i]=r[9];      r[9]
+11     IC=r[8]?0;       r[8]
+12     PC=IC==0,L85;
+13     r[8]=R[r[14]+.3_s];
+14     r[9]=r[8]+1;
+15     R[r[14]+.3_s]=r[9];      r[9]
+16     r[8]=(B[r[8]]{24)}24;
+17     IC=r[8]?0;       r[8]
+18     PC=IC!=0,L86;
+19     b[8]=32;
+20     r[9]=R[r[14]+.3_s];
+21     B[r[9]-1]=b[8]; b[8]r[9]
    L86
+22     PC=L84;
    L85
+23     r[8]=R[r[30]+.3_n];
+24     r[8]=r[8]+1;
+25     R[r[14]+.3_i]=r[8];      r[8]
    L87
+26     r[8]=R[r[14]+.3_i];
+27     r[9]=r[8]-1;
+28     R[r[14]+.3_i]=r[9];      r[9]
+29     IC=r[8]?0;       r[8]
+30     PC=IC!=0,L90;
+31     PC=L88
    L90
+32     r[8]=R[r[14]+.3_s];
+33     r[8]=r[8]-1;
+34     R[r[14]+.3_s]=r[8];
+35     r[8]=(B[r[8]]{24)}24;
+36     IC=r[8]?32;      r[8]
+37     PC=IC==0,L89;
+38     PC=L88;
    L89
+39     PC=L87;
+40     r[8]=R[r[14]+.3_s];
    L88
+41     r[8]=R[r[14]+.3_s];
+42     B[r[8]+1]=0;     r[8]
+43     r[8]=HI[L90];
+44     r[8]=r[8]+LO[L90];
+45     r[9]=R[r[30]+.3_str];
+46     ST=HI[_printf]+LO[_printf],76,2;
+47             r[8]r[9]
+48     PC=RT;
+49     NL=RS[];
```

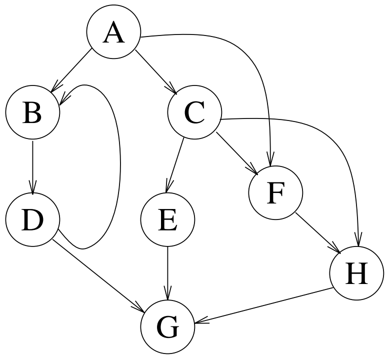
3. (1 Punkt) Konstruieren Sie den Dominatorbaum für den folgenden Kontrollflußgraphen einer Funktion. Gehen Sie dabei schrittweise vor: Geben Sie (für jeden Block)

   (a) die Nummer der Tiefensuche,

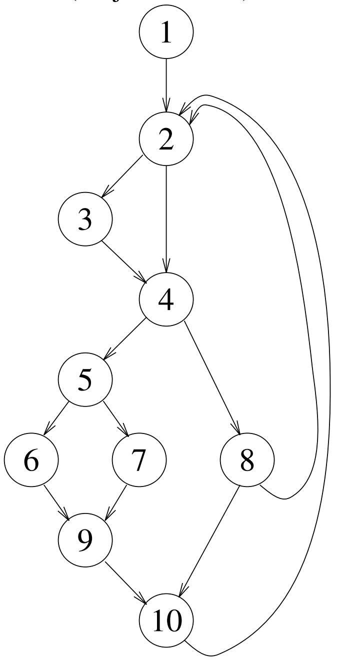
   (b) alle zur Auswahl des Semi-Dominators erwogenen Pfade,

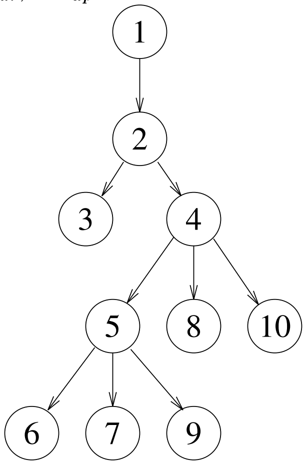
   (c) den Semi-Dominator und

   (d) den unmittelbaren Dominator an.

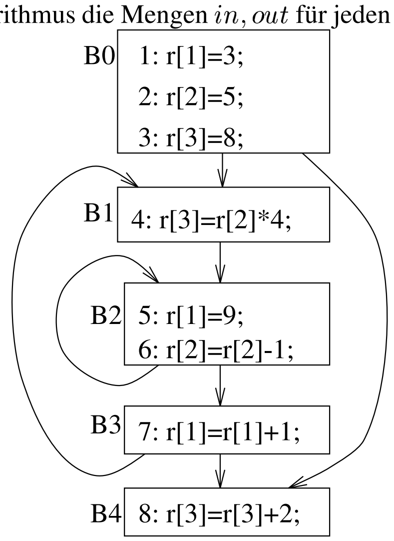


4. (1 Punkt) Geben sei folgender Kontrollfluß eines Programms und der Dominatorbaum. Bestimmen Sie die Dominatorgrenzen schrittweise: Geben Sie (für jeden Block) die Mengen DF_{local}, DF_{up} und DF an.





5. (1 Punkt) Bestimmen Sie die aktiven Variablen für den folgenden Code mit Flußgraphen, d.h. geben Sie die Mengen def, use sowie für jeden Paß des Algorithmus die Mengen in, out für jeden Block an.



Dominatorbaum

- Block X *dominiert* Y := X kommt auf jedem Pfad vom Eingangsblock nach Y vor
- X ist ein *strikter Dominator* von Y := X dominiert Y und $X \neq Y$
- *Semi-Dominator* von Block $Y \neq Root$, $sdom(Y) := min_{dfs(X)}\{X | X = B_0, B_1, ..., B_n = Y$ ist ein Pfad im Kontrollfluß und $dfs(B_i) > dfs(Y)$ für $0 < i < n\}$
- *unmittelbarer* Dominator von Y (Vater im Dominatorbaum), der von allen anderen Dominatoren von Y dominiert wird: I = idom(Y) falls I stdom Y und es gibt kein X mit I stdom X stdom Y.

  Oder durch sdom ausgedrückt (und für den Algorithmus benutzt):

  Satz: Sei $Y \neq Root$, X Knoten mit $sdom(X) = min_{dfs(X)}\{X | sdom(Y) \overset{+}{\rightarrow} X \overset{*}{\rightarrow} Y\}$. Dann gilt:

  $$idom(Y) \quad := \quad \begin{cases} sdom(Y)\ falls\ sdom(Y) = sdom(X) \\ idom(X)\ sonst \end{cases}$$

  Theorem: $\forall_{Y \neq Root}$:

  $$sdom(Y) = min_{dfs(V)}(\{V | (V,W) \in E \wedge V < W\} \cup \{sdom(U) | U > W \wedge \exists_{(V,W) \in E} U \overset{*}{\rightarrow} V\})$$

- Beispiel: Kontrollfluß (siehe oben) mit:

X: Nummer für Block X
dfs: Nummer der Tiefensuche für Block X
parent_dfs: Vater innerhalb des deckenden Baumes (der Tiefensuche)
ev1: Nummer der Blöcke U des Ergebnisses von eval(Vorgänger(X))
Pfade: Teilpfade (für die Ergebnisse von eval)
$U < X$**:** zugehörige Werte $dfs(sdom(U)) < dfs(sdom(X))$
sdom: Semi-Dominator für Teilpfade zu Block X
bucket: Blocknr. X, wo implizite idom Berechnung durchgeführt wird
ev2: Nummer der Blöcke U des Ergebnisses des zweiten eval-Aufrufs
$U < Y$**:** zugehörige Werte $dfs(sdom(U)) < dfs(sdom(Y))$
ii: idom, d.h. impliziter, unmittelbarer Dominatorblock von X (Schritt 3)
ei: idom, d.h. expliziter, unmittelbarer Dominatorblock von X (Schritt 4)

X	dfs	ev1	Pfade	$U < X$	sdom	bucket	ev2	$U < Y$	ii	ei
7	12	2	2,7	2<12	2	7	7	2<2	2	2
4	11	3	3,4	3<11	3	4	4	3<3	3	3
12	10	11	11,12	9<10	11	12	12	9<9	11	11
11	9	9;10	9,11;10,11	7< 9	9	11	11	7<7	9	9
10	8	9	9,10	7< 8	9	11,10	10	7<7	9	9
9	7	8;9	8,9;9,10,11,9	6< 7	8	9	9	6<6	8	8
8	6	2;6	2,7,8;6,7	2< 6	2	8	8	2<2	2	2
6	5	3;5	3,4,6;5,6	3< 5	3	8,6	6	3<3	3	3
5	4	3	3,5	3< 4	3	8,6,5	5	3<3	3	3
3	3	2	2,3	2< 3	2	8,3	3	2<2	2	2
2	2	1;8	1,2;8,9,10,11,12,2;8,9,11,12,2	1< 2	1	2	2	1<1	1	1

Wie ändert sich der Dominatorbaum, wenn eine Kante von Block 1 nach Block 5 hinzugefügt würde, die Ordnung (Nummerierung) der Tiefensuche jedoch beibehalten würde?

X	dfs	ev1	Pfade	$U < X$	sdom	bucket	ev2	$U < Y$	ii	ei
...										
8	...						5	1<2	5	1
6	...						5	1<3	5	1
5	4	1;3	1,5;3,5	1< 4	1		5	1<1	1	1
3	...					8,6,5,3	...			
2		1;5	1,2;5,6,7,8,9,10,11,12,2;5,6,7,8,9,11,12,2			5,2	...			

Man beachte, daß die Blöcke 5, 6 und 8 nun Kinder des Blockes 1 im Dominatorbaum wären.

Dominatorgrenzen

- Z ist in *Dominatorgrenze* für Block X: $Z \in DF(X)$ genau dann wenn X kein strikter Dominator von Z ist und Z der erste solche Block ist (auf dem Pfad von X nach Z), so daß die Variableninstanzen V_0 und V_1 von verschiedenen Vorgändern von Z sichtbar sind. (Damit wird für V in Z eine ϕ-Funktion benötigt.)

- Datenflußgleichungen:

  – $DF(X) = DF_{local}(X) \cup \bigcup_{Z \in Kinder(X)} DF_{up}(Z)$ (Kinder bzgl. Dominatorbaum)

  – $DF_{local}(X) := \{Y \in Succs(X) | X$ ist kein strikter Dominator von $Y\}$
  (Nachfolger/Succs bzgl. Kontrollfluß)

  – $DF_{up}(Z) := \{Y \in DF(Z) | idom(Z)$ ist kein strikter Dominator von $Y\}$

- Beispiel wie vorher, Dominatorgrenzen für jeden Block X werden in einer Traversierung des Dominatorbaums von den Blättern aufwärts zur Wurzel berechnet.

X	local	Kinder	DF	up
12	2	–	2	2
11	9	12	9,2	9,2
10	11	–	11	–
9	–	10,11	9,2	2
8	–	9	2	2
7	8	–	8	–
6	8	–	8	8
5	6	–	6	–
4	6	–	6	–
3	–	4,5,6	8	–
2	–	3,7,8	2	–
1	–	2	–	–