Lehrstuhl für Softwaretechnik und Programmiersprachen
Professor Dr. Michael Leuschel

Heinrich Heine Universität Düsseldorf

Compilerbau - Wintersemester 2021/22

Übungsblatt 12 - Musterlösung

Aufgabe 12.1

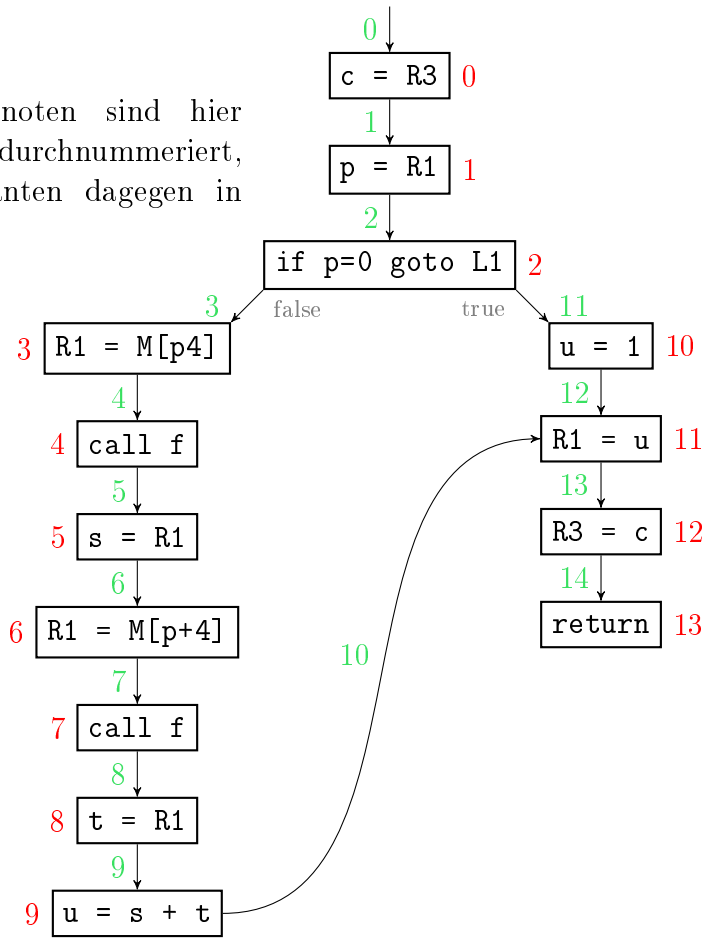
Das folgende Programm wurde für eine Maschine mit drei Registern erzeugt. Hierbei sind `R1` und `R2` caller-save Register für Argumente und R3 ist ein callee-save Register.

```
f:  c = R3
    p = R1
    if p = 0 goto L1
    R1 = M[p]
    call f            (Tipp: uses R1, defines R1, R2)
    s = R1
    R1 = M[p+4]
    call f            (Tipp: uses R1, defines R1, R2)
    t = R1
    u = s + t
    goto L2
L1: u = 1
L2: R1 = u
    R3 = c
    return            (Tipp: uses R1, R3)
```

a) Bauen Sie den Interferenzgraph auf und fügen Sie die Move-Kanten ein.

Zunächst sollen hier die Kontrollflussgraph aufgestellt und die USE- und DEF-Mengen für jeden Knote darin angegeben werden. Anschließend erfolgt eine Liveness-Analyse.

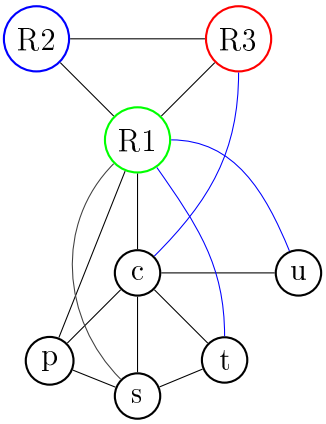
Die Knoten sind hier in Rot durchnummeriert, die Kanten dagegen in Grün.



Knoten	Use	Def
0	R3	c
1	R1	p
2	p	
3	p	R1
4	R1	R1, R2
5	R1	s
6	p	R1
7	R1	R1, R2
8	R1	t
9	s, t	u
10		u
11	u	R1
12	c	R3
13	R1, R3	

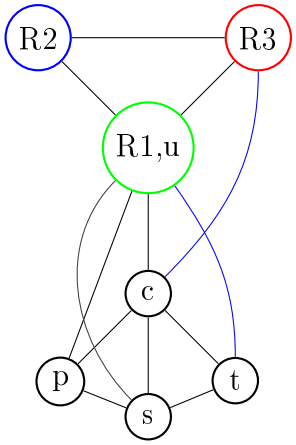
Kante	Lebende Variablen
0	R1,R3
1	c, R1
2	c, p
3	c, p
4	c, p, R1
5	c, p, R1
6	c, p, s
7	c, s, R1
8	c, s, R1
9	c, s, t
10	c, u
11	c
12	c, u
13	c, R1
14	R1, R3

Nun sollen der Interferenzgraph aufgestellt und die Move-Kanten eingefügt werden:

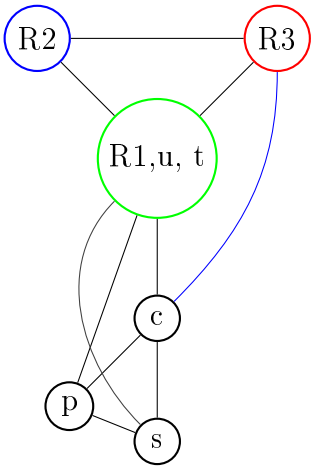


Die Move-Kanten sind hier in Blau eingezeichnet.

Verschmelzen von R1 und u:



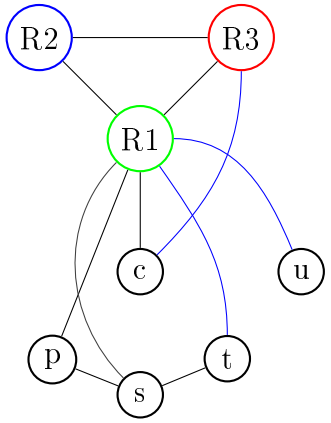
Verschmelzen von R1,u und t:



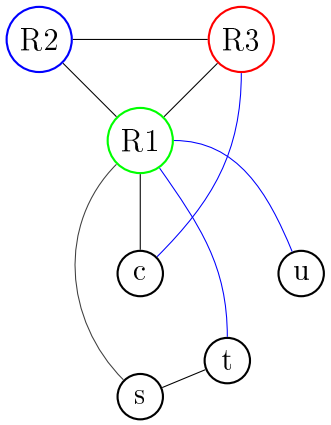
Kein Knoten kann auf dem Stack landen. Eine Färbung mit 3 Farben ist nicht möglich. Der Knoten s oder p oder c muss auf den Stack gespillt werden. Wir entscheiden uns für c. Hierdurch ergibt sich ein neues Programm:

```
f:  c = R3
    M[C_loc] = c
    p = R1
    if p = 0 goto L1
    R1 = M[p]
    call f
    s = R1
    R1 = M[p+4]
    call f
    t = R1
    u = s + t
    goto L2
L1: u = 1
L2: R1 = u
    c = M[C_loc]
    R3 = c
    return
```

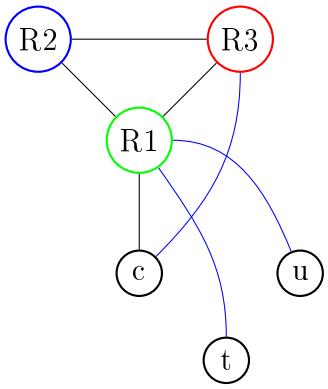
Und es ergibt sich ein neuer Graph:



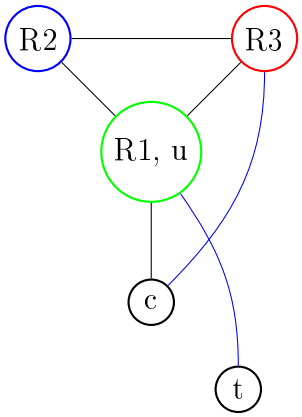
Push p. Stack p:



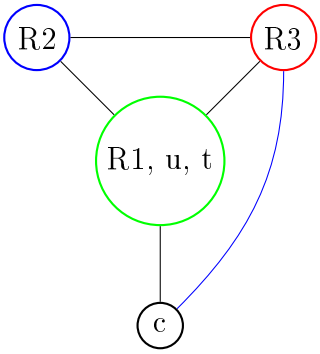
Push s. Stack p, s:



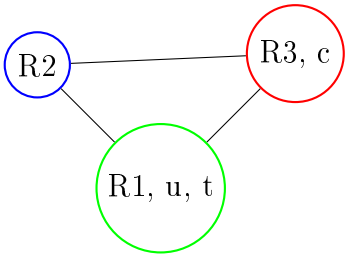
Verschmelze r1 und u. Stack p, s:



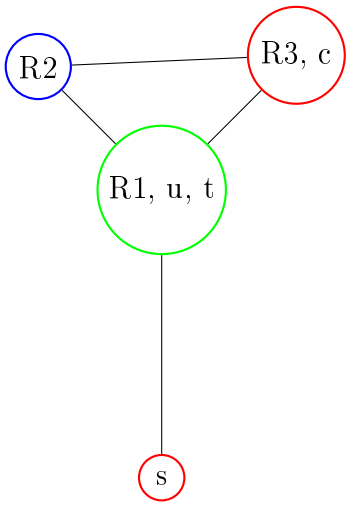
Verschmelze r1,u und t. Stack p, s:



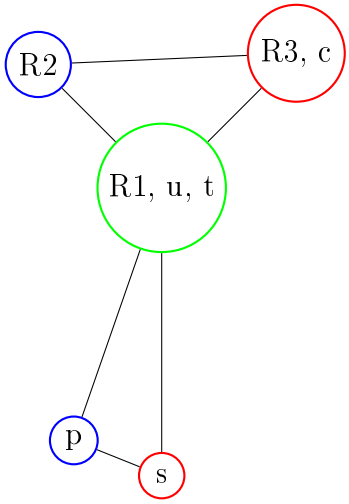
Verschmelze r3 und c. Stack p, s:



Push s und färbe rot oder blau. Stack p:



Push p und färbe blau:



Hieraus ergibt sich folgende Zuordnung von Registern:

```
f:  R3 = R3
    M[C_loc] = R3
    R2 = R1
    if R2 = 0 goto L1
    R1 = M[R2]
    call f
    R3 = R1
    R1 = M[R2+4]
    call f
    R1 = R1
    R1 = R3 + R1
```

```
    goto L2
L1: R1 = 1
L2: R1 = R1
    R3 = M[C_loc]
    R3 = R3
    return
```

Und nach Entfernung von dead code:

```
f:  M[C_loc] = R3
    R2 = R1
    if R2 = 0 goto L1
    R1 = M[R2]
    call f
    R3 = R1
    R1 = M[R2+4]
    call f
    R1 = R3 + R1
    goto L2
L1: R1 = 1
L2: R3 = M[C_loc]
    return
```