Hochschule Osnabrück, Fakultät für Ingenieurwissenschaften und Informatik
Labor für Digital- und Mikroprozessortechnik, Prof. Dr.-Ing. Markus Weinhardt

Programming Exercises Compiler Construction

Code Generation

Auf der nachfolgenden Seite finden Sie den 3-Adress-Code einer Funktion f, die folgende Formel berechnet:

$$s = \sum_{i=0}^{n} |a[i] - b[i]|$$

Dabei sind die Parameter a, b und n bereits in Registern enthalten, und das Ergebnis s soll auch in einem Register zurückgegeben werden.

Aufgaben

a) Umrahmen Sie die Basisblöcke, aus denen der Code besteht, und ergänzen die Kanten des Kontrollfluß-Graphen durch Pfeile.

b) Zeichnen Sie für den größten Basisblock (der nach dem Befehl "if i>n goto done" beginnt) die DAG-Darstellung unter Berücksichtigung gemeinsamer Teilausdrücke.

c) Eliminieren Sie gemeinsame Teilausdrücke (falls vorhanden) im 3-Address-Code des Basisblocks, indem Sie eliminierte Instruktionen streichen und die TEMPs entsprechend anpassen.

d) Tragen Sie für alle (verbliebenen) Instruktionen der Funktion f die def- und use-Mengen in die Tabelle neben dem Code ein. Bestimmen Sie dann die live-in-Mengen aller Instruktionen durch Backward-Propagation von der return-Instruktion und tragen sie ebenfalls in die Tabelle ein.

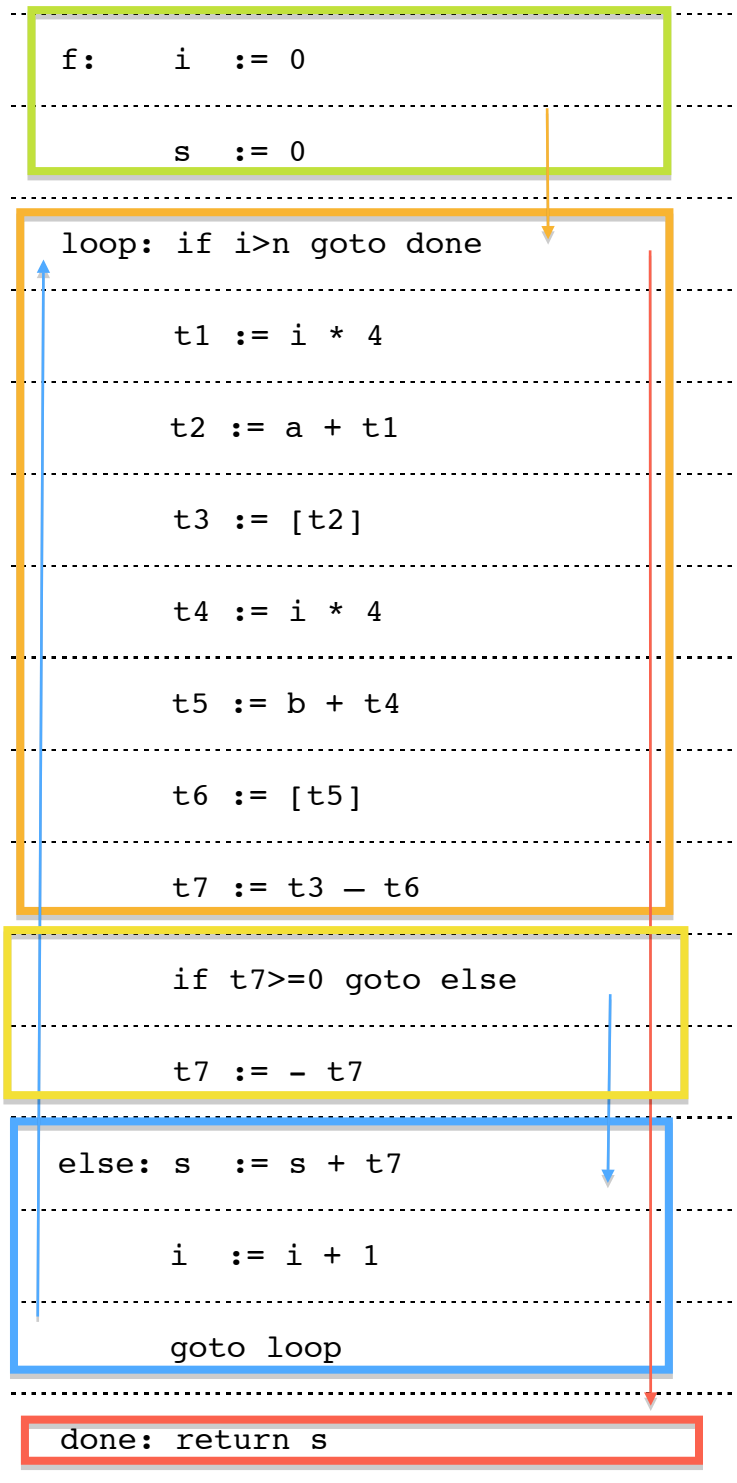
e) Bestimmen Sie aus den live-in-Mengen den Register-Konflikt-Graphen der Funktion f. Berücksichtigen Sie zur Vereinfachung dabei nur die TEMPs t1 - t7. (Gehen Sie davon aus, dass für die Variablen a, b, n, i und s bereits feste Register gleichen Namens vorgesehen sind.)

   Wieviele zusätzliche Register sind für die TEMPs mindestens nötig, damit die Funktion ohne Register-Spills implementiert werden kann? "Färben" Sie den Graphen, indem Sie jedem TEMP ein Register zuordnen.

f) Schreiben Sie mit der gefundenen Registerallokation einfachen Assemblercode für die Funktion.
   - Verwenden Sie a, b, n, i und s direkt als Register.
   - Verwenden Sie folgende 3-Address-Assembler-Befehle: MOV, CMP (compare), BRGT (branch if greater than), BRGE (branch if greater or equal), MUL, ADD, SUB, LD (load indirect), JUMP und RET (return).

   Beispiel: Der erste Befehl von f lautet: `MOV i, $0`

3-Adress-Code der Funktion f:



def	use	live-in
i		
s		
	i, n	
t1	i	
t2		

b)