

Globale Registerallokation für irreguläre Registerarchitekturen durch Graphfärbung

Mario Schölzel

Brandenburgische Technische Universität Cottbus
Lehrstuhl Programmiersprachen und Compilerbau

Postfach 101344

D-03013 Cottbus

Telefon: +49 (355) 69 - 3886

mas@informatik.tu-cottbus.de

Zusammenfassung

Es wird ein Ansatz zur globalen Registerallokation für irreguläre Registerarchitekturen vorgestellt, der auf dem von Chaitin in [1] und [2] dargelegten Grundprinzip durch Graphfärbung beruht. Dieses wurde erweitert, um den Anforderungen irregulärer Registerarchitekturen gerecht zu werden. Dazu gehört die Behandlung von Spezialregistern, strukturierten und adjazenten Registern auf eine einheitliche Art und Weise. Die Allokation wird durch gezielte Berücksichtigung von Unterregistern noch effizienter gestaltet.

1. Einführung

Der hier dargelegte Ansatz zur globalen Registerallokation für irreguläre Registerarchitekturen orientiert sich an den Rahmenbedingungen, die durch die Arbeiten [5] und [4] gegeben sind und die eine Methode zur Erstellung effizienter Programme für irreguläre Prozessorarchitekturen vorstellen, zu denen typischerweise Digitale Signalprozessoren (DSPs) zählen. Irreguläre Registerarchitekturen zeichnen sich durch viele Spezialregister und keine oder nur wenige allgemeine Register aus. Spezialregister können nur in einigen Maschinenbefehlen als Operanden verwendet werden. Des weiteren gibt es Prozessorregister, die *Unterregister* enthalten, auf die separat zugegriffen werden kann, sowie Maschinenbefehle, die implizit Gebrauch von mehreren Prozessorregistern machen, um diese wie ein großes Register zu verwenden. Durch Verwendung eines Registers sind in diesem Fall alle übrigen verwendeten Register eindeutig festgelegt. Register auf die das zutrifft, werden als *adjazente Register* bezeichnet.

2. Grundlagen

Die Registerallokation wird für den Datenflussgraphen einer Routine durchgeführt. Das in [4] vorgestellte Konzept sieht vor, dass die Knoten dieses Graphen bereits konkrete Assembleroperationen der Zielarchitektur darstellen. Die Kanten repräsentieren virtuelle Prozessorregister, die vergleichbar sind mit realen Prozessorregistern, von denen aber beliebig viele im Programm verwendet werden dürfen. Virtuelle Register können, genauso wie Prozessorregister, Unterregister besitzen. Auf die Komponenten des Hauptregisters wird dann durch eine Qualifizierung zugegriffen. Ein Beispiel dafür ist in Abbildung 4 zu sehen, in der das virtuelle Hauptregister v die zwei virtuellen

Unterregister a und o besitzt, auf die mittels $v.a$ bzw. $v.o$ zugegriffen wird.

Die Knoten des Datenflussgraphen lassen sich in disjunkte Teilmengen zerlegen, die die Basisblöcke der Routine darstellen. Da in [4] noch eine zusätzliche Codeparallelisierung in den Basisblöcken vorgenommen wird, bezieht sich die hier vorgestellte Registerallokation nur auf globale virtuelle Register. Das sind solche virtuellen Register, die in mehreren Basisblöcken definiert oder verwendet werden. Alle anderen virtuellen Register werden als lokal bezeichnet. Abbildung 1 zeigt einen Datenflussgraphen, in dem die Knoten rechteckig dargestellt und die Basisblöcke durch Ellipsen eingerahmt sind. Globale virtuelle Register sind in diesem Beispiel $v1$, $v2$, $v3$ und $v4$.

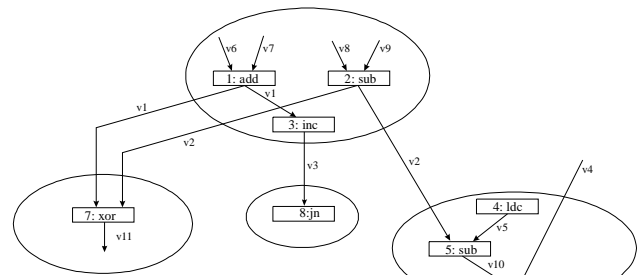


Abbildung 1: Teil eines Datenflussgraphen einer Routine

Da bei den folgenden Betrachtungen die lokalen virtuellen Register nicht von Interesse sind, kann der Datenflussgraph aus Abbildung 1 zu dem vereinfachten Datenflussgraphen in Abbildung 2 reduziert werden, in dem Basisblöcke als Knoten dargestellt sind.

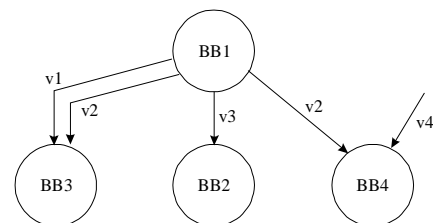


Abbildung 2: Vereinfachung des Datenflussgraphen aus Abbildung 1

Dieser vereinfachte Datenflussgraph wird verwendet, um die gleichzeitig lebendigen virtuellen Register zu berechnen.

3. Chaitin's Registerallokator

Chaitin konstruiert zu einem Datenflussgraphen¹ den Interferenzgraphen $I=(V, E)$. Für jedes virtuelle Register v im Datenflussgraphen gilt $v \in V$. Falls zwei virtuelle Register u und v gleichzeitig lebendig sind, so gilt $(u, v) \in E$. Eine Registerallokation lässt sich jetzt durch Färbung des Interferenzgraphen finden. Die Farben entsprechen den zur Verfügung stehenden Prozessorregistern. Zwei adjazente Knoten dürfen nicht die gleiche Farbe erhalten, weil sich sonst die Werte im zugeordneten Prozessorregister gegenseitig überschreiben würden.

Der Algorithmus für die Färbung lässt sich in eine *Reduktionsphase* und eine *Färbungsphase* gliedern. Angenommen es stehen k Farben zur Verfügung, dann kann ein Knoten mit weniger als k Nachbarn sicher gefärbt werden. Dieser Knoten wird zusammen mit seinen angrenzenden Kanten in der Reduktionsphase aus dem Graphen entfernt. Durch wiederholte Anwendung dieses Schrittes wird versucht, den Interferenzgraphen zum leeren Graphen zu reduzieren. Wenn das gelungen ist, so werden die Knoten in der umgekehrten Reihenfolge, in der sie entfernt wurden, wieder in den Graphen eingefügt und ihnen dabei unter Berücksichtigung der bereits eingefärbten Nachbarn eine Farbe zugeordnet.

Ist kein Knoten mit weniger als k Nachbarn im Graphen vorhanden, so wird ein Knoten als Spillkandidat ausgewählt und für das damit identifizierte virtuelle Register Programmcode zum Sichern und Restaurieren (*Spillcode*) im Programm eingefügt. Anschließend wird der Interferenzgraph erneut konstruiert und die Registerallokation neu gestartet. Die Menge des eingefügten Spillcodes lässt sich durch die optimistische Strategie von Briggs aus [3] noch verbessern.

4. Erweiterungen

Um Chaitin's Ansatz, der sich gut für Registerarchitekturen die nur Universalregister besitzen eignet, auf irreguläre Registerarchitekturen anzuwenden, müssen folgende Architektureigenschaften berücksichtigt werden:

- Jedes virtuelle Register besitzt eine individuelle Menge an Prozessorregistern, die ihm zugeordnet werden können,
- Strukturierte Register und
- Adjazente Register müssen behandelt werden können.

Aus diesem Grund wurden Erweiterungen an den Farben und am Interferenzgraphen vorgenommen.

4.1 Strukturierte Farben

Farben² werden jetzt als Mengen aufgefasst, die selbst wieder aus weiteren Farben bestehen dürfen. Dadurch lassen sich strukturierte Register sehr gut modellieren. Die Menge aller zur Verfügung stehenden Farben sei C . B soll die Menge aller elementaren Prozessorregister bezeichnen, die keine weiteren Prozessorregister enthalten. Dann muss C den folgenden drei Bedingungen genügen:

- $B \subseteq C$.

¹ Chaitin verwendet allerdings keinen vereinfachten Datenflussgraphen, sondern bezieht alle virtuellen Register in die Allokation ein.

² Die Begriffe Farbe und Prozessorregister werden hier synonym verwendet.

- Ist r ein Prozessorregister mit $r \notin B$ und r besteht aus den Unterprozessorregistern r_1, \dots, r_n , so gilt $\{r_1, \dots, r_n\} \in C$, falls $n = \min\{k \mid r \text{ besteht aus den Unterregistern } a_1, \dots, a_k\}$ und $r_1, \dots, r_n \in C$.
- Es gibt keine weiteren Elemente in C .

Mit der so definierten Farbmenge C und einer Funktion $c: V \rightarrow \wp(C)$ kann festgelegt werden, welche Farben zum Färben eines Knotens im Interferenzgraphen verwendet werden dürfen. Wenn $c(v) = F$, dann gilt: $f \in F$, gdw. v mit f gefärbt werden darf, falls also das durch f repräsentierte Prozessorregister dem virtuellen Register v zugeordnet werden kann.

Um im folgenden auf die Elemente einer Farbe aus C , die sich aus mehreren anderen Farben zusammensetzt, Bezug nehmen zu können, soll der Begriff der Teilfarbe eingeführt werden.

Definition 1 (Teilfarbe c_c)

Eine Farbe x ist Teilfarbe der Farbe y ($x \subset_c y$), falls $x = y$ oder $x \subset_c z$ für ein $z \in y$. Zur Vereinfachung einiger folgender Betrachtungen soll die Farbe x Teilfarbe einer Menge Y von Farben sein ($x \subset_c Y$), falls $x \subset_c y$ für ein $y \in Y$. \square

Der von Chaitin eingeführte Interferenzgraph wird erweitert, um die strukturierten Farben zu berücksichtigen.

Definition 2 (erweiterter Interferenzgraph I_E)

Ein erweiterter Interferenzgraph $I_E = (V, E_E, C, c, weight)$ ist ein gerichteter Graph mit der Kantengewichtungsfunktion $weight: E_E \rightarrow \mathbb{N}$, der Knotenbewertung $c: V \rightarrow \wp(C)$, den Knoten V , den Kanten $E_E = \{(u, v), (v, u) \mid (u, v) \in E\}$ und der zugehörigen Farbmenge C . \square

Die Gewichtungsfunktion $weight((u, v))$ gibt an, wie viele Farben aus $c(u)$ maximal durch eine beliebige Färbung von v nicht mehr für die Färbung von u genutzt werden können und ist definiert als: $weight((u, v)) = \max\{|C_t| \mid t \in c(v) \text{ und } C_t = \{s \mid s \in c(u) \text{ und } \exists f \subset_c t \text{ mit } f \subset_c s\}\}$.

Durch diesen erweiterten Interferenzgraphen ergibt sich auch eine veränderte Interpretation der Färbung. Eine korrekte Färbung des Interferenzgraphen ist nur dann noch eine korrekte Registerallokation, wenn die, zwei adjazenten Knoten zugewiesenen Farben, keine gemeinsame Teilfarbe besitzen. Aus diesen Änderungen ergibt sich auch ein neues Kriterium, wann ein Knoten in der Reduktionsphase aus dem erweiterten Interferenzgraphen entfernt werden darf.

4.2 Neues Kriterium

Das neue Kriterium muss insbesondere berücksichtigen, dass jeder Knoten über eine individuelle Farbmenge verfügen kann und dass verschiedene Farben gleiche Teilfarben enthalten können. Die sich daraus ergebenden Konsequenzen sind in Abbildung 3 beispielhaft dargestellt.

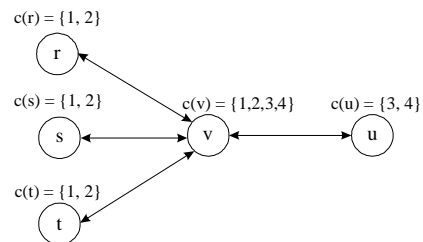


Abbildung 3: Interferenzgraph mit individuellen Farbmengeten

Es soll angenommen werden, dass r, s, t und u mit weiteren Knoten adjazent sind, so dass diese vier Knoten in der Reduktionsphase nicht vor v aus dem Graphen entfernt werden. Es stehen für v insgesamt vier Farben zur Verfügung. Er hat aber auch vier Nachbarn. Dennoch ist v sicher färbbar, da die Farben 1 und 2 nur von den Knoten r, s und t verwendet werden und u nur eine Farbe aus $\{3,4\}$ verwenden kann. Somit bleibt eine Farbe aus $\{3,4\}$ für v übrig.

Es ist also eine Zerlegung Z_f der Farben $c(v)$ in die Klassen f_1, \dots, f_n gesucht und eine Zerlegung Z_k der Nachbarknoten von v in die Klassen k_1, \dots, k_n , so dass die Farben der Klasse f_i mit all ihren Unterfarben ausschließlich von den Knoten der Klasse k_i verwendet werden. Weiterhin wird eine Klasse f_0 eingeführt, die die Farben enthält, die in keinem Nachbarknoten von v verwendet werden. Zu f_0 soll die Knotenklasse k_0 gehören, die per Definition leer ist. Für die Zerlegung Z_f von $c(v)$ müssen die folgenden Bedingungen gelten:

- $c(v) = \bigcup_{i=0}^n f_i$,
- $\forall i, j: i \neq j: f_i \cap f_j = \emptyset$ mit $0 \leq i \leq n, 0 \leq j \leq n$.

Für die Zerlegung Z_k aller Nachbarknoten W von v muss gelten:

- $W = \bigcup_{i=0}^n k_i$,
- $\forall i, j: i \neq j: k_i \cap k_j = \emptyset$ mit $0 \leq i \leq n, 0 \leq j \leq n$,
- $k_0 = \emptyset$,

wobei die Zerlegung $Z_k = \{k_0, \dots, k_n\}$ mit maximalem n gesucht ist, die noch folgende Forderungen erfüllt:

- $u \in k_i \wedge f \in c(u) \wedge \exists t \subset_c f: t \subset_c g \in c(v) \Rightarrow g \in f_i$ und
- $\forall f_i, f_j$ mit $i \neq j$ gilt: $\forall f \subset_c f_i \Rightarrow f \not\subset_c f_j$.

Diese Zerlegung existiert in jedem Fall, wenn $n=1$ gewählt wird. Aus diesen Forderungen folgt unmittelbar

Lemma 1

Für zwei Knoten $v_i \in k_i$ und $v_j \in k_j$ mit $i \neq j$ gilt: $\nexists f \in c(v)$, so dass $\exists f_1 \subset_c g \in c(v_i)$ und $\exists f_2 \subset_c h \in c(v_j)$ mit $f_1 \subset_c f$ und $f_2 \subset_c f$.

Beweis

f_i sei die von k_i und f_j die von k_j induzierte Farbmenge. Angenommen es gäbe ein $f \in c(v)$ und ein $f_1 \subset_c f$ und ein $f_2 \subset_c f$, dann gehört f nach Forderung (i) sowohl zu f_i als auch zu f_j , damit ist $f_i \cap f_j \neq \emptyset$, was ein Widerspruch zu Forderung (ii) ist. \square

Mit Hilfe von Lemma 1 ergibt sich die gesuchte Klasseneinteilung, indem mit einem beliebigen Nachbarknoten u von v gestartet und die Klasse k_u gebildet wird. Dann wird der Abschluss über k_u gebildet, indem alle weiteren Nachbarknoten von v in k_u aufgenommen werden, die eine Teilfarbe besitzen, die als Teilfarbe in einer Farbe $h \in c(v)$ vorkommt und h eine Teilfarbe enthält, die auch als Teilfarbe in $c(w)$ für ein $w \in k_u$ auftritt. Für alle Knoten x , die dadurch in k_u aufgenommen werden, werden auch alle Farben $h \in c(v)$ in f_u aufgenommen, die eine gemeinsame Teilfarbe mit einer Farbe aus $c(x)$ besitzen. Dieser Vorgang wird solange wiederholt, bis die Klassen k_u und f_u sich nicht mehr ändern. Die so gebildeten Klassen sind von minimaler Größe und damit ist die gefundene Zerlegung maximal. Diese Schritte werden für die Knoten, die noch nicht

in den bisher gebildeten Knotenklassen enthalten sind, wiederholt.

Die so bestimmten Knotenklassen k_i implizieren, wie oben definiert, eine Klasseneinteilung f_i über $c(v)$.

In der Reduktionsphase muss getestet werden, ob für einen Knoten v eine Knotenklasse i existiert, so dass folgende hinreichende Bedingung gilt:

$$\sum_{u \in k_i} \text{weight}((v, u)) < |f_i|, 0 \leq i \leq n.$$

Ist das der Fall, so kann v sicher gefärbt werden. Es ist zu beachten, dass sich die gebildeten Knotenklassen nach jedem Schritt in der Reduktionsphase verändern können. Sie müssen also immer neu berechnet werden.

Satz 1

Falls für einen Knoten v und eine maximale Zerlegung Z_k gilt:

$$\exists i: \sum_{u \in k_i} \text{weight}((v, u)) < |f_i|, 0 \leq i \leq n, \text{ dann ist } v \text{ sicher färbbar.}$$

Beweis

Sei k die Klasse k_i , die die Voraussetzung erfüllt, dann gehen v durch das Färben der Knoten in k höchstens m Farben verloren,

wobei $m = \sum_{u \in k_i} \text{weight}((v, u))$. Das folgt direkt aus der Definition

der Gewichtungsfunktion weight . Sei weiterhin W die Menge aller Nachbarknoten von v und $\bar{k} = W - k$, dann geht v keine Farbe aus f_i verloren, egal wie ein $u \in \bar{k}$ gefärbt wird. Angenommen das wäre doch der Fall, dann soll f diese Farbe sein. Für f gilt auch: $f \in c(v)$. Da f in f_i liegt, existiert ein Knoten $v_i \in k_i$, der eine Teilfarbe aus f verwendet. Nach Lemma 1 müssten v_i und u in der gleichen Knotenklasse liegen, was ein Widerspruch ist.

Damit können unter jeder beliebigen Färbung der Nachbarknoten, v höchstens m Farben entzogen werden. Da $m < |f_i|$ gilt, bleibt für v wenigstens eine Farbe aus f_i übrig. \square

4.3 Adjazente Register

Wie in (c) gefordert, müssen adjazente Register von der globalen Registerallokation gehandhabt werden können. Das bedeutet, dass die Zuweisung einer Farbe an einen Knoten gleichzeitig die Zuweisung einer eindeutig definierten Farbe an einen anderen Knoten zur Folge hat. Der Registerallokator kann adjazente Register wie Unterregister eines Hauptregisters behandeln. Die Farben, mit denen das Hauptregister gefärbt werden kann, werden aus allen zulässigen Farbkombinationen der adjazenten Register gebildet. Dadurch wird durch das Zuweisen einer Farbe an das Hauptregister den Unterregistern immer eine korrekte Farbkombination zugeordnet. Die neu gebildeten Farbmengen müssen noch in die Menge C aufgenommen werden, da sie in der Regel keine Register darstellen, die durch den Prozessor definiert werden. Somit können adjazente Register im Interferenzgraphen wie ein strukturiertes Register dargestellt und auch so behandelt werden.

4.4 Verbesserung

Strukturierte Register werden im Interferenzgraphen durch die Struktur der Farben beschrieben. Repräsentiert werden sie dabei von einem einzigen Knoten. Das kann zu einer schlechten Registerallokation führen, da beispielsweise bei adjazenten

Registern jedes Unterregister einzeln definiert werden kann. Damit wird das Hauptregister (einschließlich aller Unterregister) als lebendig betrachtet, obwohl es noch Unterregister geben kann, die noch nicht lebendig, da noch nicht definiert sind. Abbildung 4 soll diesen Fall verdeutlichen.

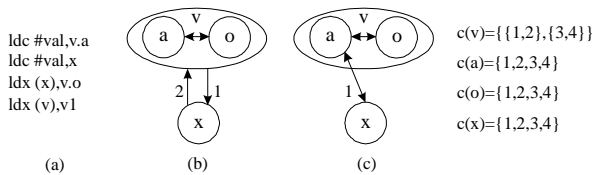


Abbildung 4: Beispiel für ineffiziente Registerallokation bei strukturierten Registern

Das virtuelle Hauptregister v wird durch einen strukturierten Knoten repräsentiert, der die Unterregister a und o als Unterknoten enthält. x ist ein virtuelles Register mit einem kurzen Lebendigkeitsbereich, der sich nicht mit dem Lebendigkeitsbereich des gesamten Registers v überschneidet (vgl. (a)), sondern nur mit dem des Unterregisters a , wie in (c) dargestellt. Somit könnte x die gleiche Farbe wie o zugewiesen bekommen. In der Reduktionsphase wird dieser Umstand aber bisher nicht berücksichtigt, da x mit v adjazent ist und x durch die Färbung von v zwei Farben entzogen werden könnten (vgl. (b)). Dieses Problem kann gelöst werden, indem die Bewertung der Kante (x,v) nicht auf Basis der unstrukturierten Knoten bestimmt wird, sondern unter Berücksichtigung der hierarchischen Struktur von v . Damit wird bei der Bestimmung der Kantenbewertung von (x,v) die genaue Anzahl der Farben berechnet, die x maximal durch die Färbung der Unterknoten von v entzogen werden. Ist x zu einem Unterknoten nicht adjazent, so kann ihm dieser auch keine Farben entziehen.

Die Unterknoten werden bei der Bestimmung der Kantenbewertung wie normale Knoten behandelt, wodurch zunächst für jede Kante die Kantenbewertung wie in 4.2 beschrieben, vorgenommen werden kann.

In der Reduktionsphase kann die hierarchische Struktur eines Knotens v vernachlässigt werden, da sie in der Farbmenge $c(v)$ und der Kantenbewertung berücksichtigt wird. Jeder strukturierte Knoten v muss somit durch einen einfachen Knoten ersetzt werden. Damit darf jeder andere Knoten x nur eine Kante zu v besitzen, auch wenn er tatsächlich mit mehreren Unterknoten von v adjazent ist. Die Bewertung dieser einen Kante lässt sich recht einfach aus den vorhandenen Kantenbewertungen zu den Unterknoten ermitteln. Folgende Notation soll dazu noch eingeführt werden.

Notation (Vorfahre)

Ein Knoten a ist Vorfahre vom Knoten b , falls b ein Unterknoten von a ist oder b ist Unterknoten von c und a ist Vorfahre von c . \square

Wird der strukturierte Knoten v durch den einfachen Knoten u ersetzt, so sind bei der resultierenden Kantenbewertung folgende Fälle zu unterscheiden:

- B.1 Ist v oder einer seiner Unterknoten mit einem zweiten strukturierten Knoten x oder einem seiner Unterknoten adjazent, dann gilt
 $weight(x,u) = weight(u,x) := weight(v,x)$.
- B.2 Ist ein einfacher Knoten x mit den Knoten a und b adjazent, wobei a Vorfahre von b ist und v Vorfahre von a , dann ist $weight(x,u) := weight(x,a)$.

- B.3 Ist ein einfacher Knoten x mit zwei Unterknoten a und b von v adjazent, wobei weder a Vorfahre von b noch b Vorfahre von a ist, dann ist
 $weight(x,u) := weight(x,a) + weight(x,b)$.

B.1 folgt aus der Tatsache, dass beide strukturierten Knoten sich wenigstens teilweise in ihren Lebendigkeitsbereichen überschneiden. Damit dürfen beide Knoten nicht das gleiche Prozessorregister zugewiesen bekommen und somit befinden sich auch alle Unterknoten in verschiedenen Registern. Daher ist es nicht nötig zu berücksichtigen, ob eventuell nur Unterknoten adjazent sind.

B.2 gilt, weil alle Unterknoten y eine Farbmenge $c(y)$ besitzen. Ist a Vorfahre von b , so kommen alle Farben von b als Teilfarben in $c(a)$ vor. Daher berücksichtigt a bereits alle Farben, die x durch jede beliebige Färbung (auch jede beliebige Färbung von b) entzogen werden können und die Kante zu b kann vernachlässigt werden.

In B.3 kommt zum Ausdruck, dass zwei Teilknoten, die mit verschiedenen Farben gefärbt werden, x alle diese Farben entziehen, da für diese Knoten angenommen werden kann, dass sie disjunkte Farbmengen besitzen.

4.5 Färben und Spillen

Das Färben der Knoten, die wieder in den Interferenzgraphen eingefügt werden, unterscheidet sich prinzipiell nicht vom Vorgehen Chaitin's. Zu beachten ist, dass die hierarchische Struktur der Knoten hier nicht vernachlässigt werden darf, da für jeden Knoten berechnet werden muss, welche Farben ihm durch die bereits eingefärbten adjazenten (Unter-) Knoten nicht mehr zur Verfügung stehen.

Müssen virtuelle Register ausgelagert werden, so wird der dafür benötigte Programmcode in die Basisblöcke eingefügt und später eine kombinierte Codeparallelisierung und Registerzuordnung zu den lokalen virtuellen Registern durchgeführt. Eine erneute Iteration der globalen Registerallokation, wie bei Chaitin oder Briggs ist damit nicht notwendig.

Heuristiken zur Auswahl einer Farbe für einen Knoten bzw. eines Spillkandidaten sind in [4] angegeben.

Literaturverzeichnis

- [1] Chaitin, G.J.; Auslander, M.A.; Chandra, A.K.; Cocke, J.; Hopkins, M.E.; Markstein, P.W.: Register allocation via graph coloring. Computer Languages, 6:47-57, 1981.
- [2] Chaitin, G.J.: Register allocation and spilling via graph coloring, ACM SIGPLAN Notices 17:6 201-207, 1982.
- [3] Briggs, P.: Register Allocation via Graph Coloring. Ph. D. Thesis Rice, Department of Computer Science, Rice University, 1992.
- [4] Müller, M.; Schölzel, M.: Methoden und Werkzeuge zur Entwicklung effizienter Programme für irreguläre Prozessorarchitekturen, Diplomarbeit, Lehrstuhl für Programmiersprachen und Compilerbau, BTU Cottbus, 2001.
- [5] Bachmann, P.: Projektskizze UPHA. BTU Cottbus, Institut für Informatik, 1999.