TỐI ƯU HÓA VIỆC CẤP PHÁT THANH GHI BẰNG GIẢI THUẬT TÔ MÀU ĐỒ THỊ

Tạ Duy Công Chiến – Phan Thị Tươi
Trường Đại học Bách khoa – Đại học Quốc Gia Tp.HCM
(Bài nhận ngày 28 tháng 8 năm 2003)

TÓM TẮT: *Bài báo này sẽ trình bày giải thuật tô màu đồ thị dùng để giải quyết vấn đề cấp phát thanh ghi trong quá trình biên dịch khi số lượng thanh ghi là giới hạn và đồng thời trong bài báo cũng đề xuất một số ý tưởng cải thiện giải thuật để việc cấp phát thanh ghi hiệu quả hơn. Với giải thuật này, quá trình cấp phát thanh ghi toàn cục cho chương trình được hiện thực, sao cho số thanh ghi là tối thiểu. Số màu trong đồ thị sau khi tô, tượng trưng cho số thanh ghi vật lý cần phải cấp phát.*

Bài toán tô màu đồ thị là bài toán có độ phức tạp là lũy thừa, do đó heuristics được để xuất sẽ làm tăng tính hiệu quả của giải thuật cấp phát thanh ghi.

Các thanh ghi được các trình biên dịch sử dụng để lưu các giá trị trung gian của các tính toán nhằm tăng tốc độ thực thi ở giai đoạn sinh mã. Do đó chiến lược cấp phát và gán thanh ghi là một trong các vấn đề trọng tâm của giai đoạn tối ưu mã. Bài toán được đặt ra là: ***với số thanh ghi hạn chế trong CPU thì các trình biên dịch làm sao tận dụng được tối ưu chúng trong quá trình biên dịch***.

1. Cấp phát thanh ghi bằng giải thuật tô màu đồ thị

Việc cấp phát thanh ghi bằng phương pháp tô màu đồ thị đã được John Cocke đề xuất vào năm 1971[5]. Giải thuật xây dựng trên một số khái niệm sau:

- Gọi S1, S2, ...Si là tên của các thanh ghi ký hiệu được dùng cho các đối tượng trong mã trung gian ba địa chỉ.
- Ud-chain(x) là tập hợp các vị trí của x được định nghĩa (được gán trị) trong chương trình.
- Duchain(x) là tập hợp các vị trí của x được dùng trong chương trình tương ứng với định nghĩa của x và trước khi x được định nghĩa lại nếu có.
- Đồ thị giao thoa là đồ thị vô hướng có các nút là các thanh ghi ký hiệu, một cung nối giữa hai nút nếu như một nút còn sống tại thời điểm nút kia được định nghĩa.

Giải thuật bao gồm các bước sau

Bước1. Cấp phát thanh ghi ký hiệu

Giải thuật cấp phát thanh ghi ký hiệu

Nhập : Ud-chain và Duchain của các đối tượng.

Xuất : Đoạn mã với các thanh ghi ký hiệu được cấp phát cho các đối tượng trong chương trình.

Phương pháp:

- Xác định các Ud-chain và Duchain bằng các giải thuật ở [1].
- Cấp phát thanh ghi ký hiệu S1, S2,... Si cho các đối tượng trong mã trung gian dựa vào Ud-chain và Duchain .

Bước 2. Xây dựng Đồ thị giao thoa

Để xây dựng đồ thị giao thoa, trước hết ta xây dựng ma trận kề cho đồ thị vô hướng là ma trận vuông hai chiều, mỗi chiều có số phần tử bằng với số thanh ghi ký hiệu, phần tử ở hàng i cột j mang trị là một nếu như i(j) còn sống tại thời điểm j(i) được định nghĩa, ngược lại có giá trị không.

Giải thuật xây dựng đồ thị giao thoa

Nhập: Đoạn mã với các thanh ghi ký hiệu và ma trận kề.

Xuất: Đồ thị giao thoa.

Phương pháp:

- Xây dựng ma trận kề.
- Tiến hành xây dựng đồ thị giao thoa dựa trên ma trận kề.

Bước 3. Cấp phát và gán lại thanh ghi

- Sử dụng giải thuật tô màu trên đồ thị giao thoa để tiến hành cấp phát và gán lại thanh ghi.
- Số màu tô được cho một đồ thị sẽ tương ứng với số thanh ghi vật lý được cấp phát.

Giải thuật tô màu đồ thị.

Nhập: Đồ thị giao thoa chưa tô màu.

Xuất: Đồ thị giao thoa được tô màu

Phương pháp

- Duyệt đồ thị theo chiều sâu.
- Tiến hành tô màu đồ thị sao cho màu của các nút có liên kết với nhau là khác nhau và đảm bảo số màu là ít nhất.
- Cập nhật lại các thanh ghi trên tập mã trung gian đã cho.

Giải thuật cấp phát và gán lại thanh ghi.

Nhập:

- Đồ thị giao thoa đã tô màu
- Tập mã đã cấp phát với các thanh ghi ký hiệu.

Xuất:

Tập mã được cập nhật lại bằng số thanh ghi vật lý.

Phương pháp

- Xét các nút trên đồ thị giao thoa có cùng màu.
- Tìm các thanh ghi ký hiệu trên tập mã giống với các nút đó, thay thế chúng bằng thanh ghi Ri.
- Quay lại bước 1, cho đến khi nào duyệt hết tất cả các nút trên đồ thị.

Ví dụ1:

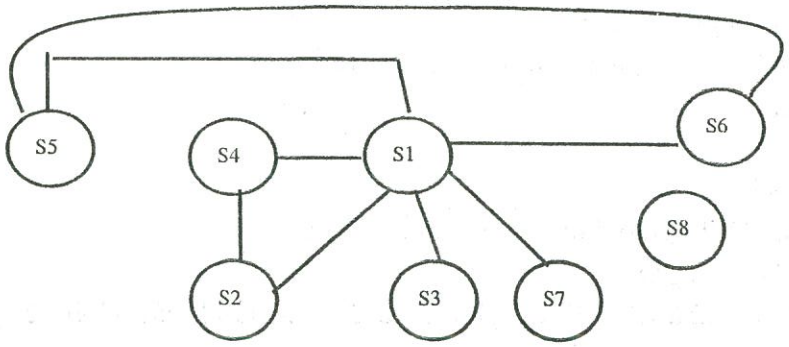
Mã ba địa chỉ	Thanh ghi ký hiệu	*Sau khi đã tối ưu*
b := 3 t1 := 1 + 4 t2 := t1 - b t3 := t2 + 2 a := t3 t2 := a + 6 t3 = a + t2 a = t3 -b	S1 := 3 S2 := 1 + 4 S3 := S2 – S1 S4 := S3 + 2 S5 := S4 S6 := S5 + 6 S7 := S5 + S6 S8 := S7 – S1	*R1 := 3* *R2 := 1 + 4* *R2 := R2 – R1* *R3 := R2 + 2* *R2 := R3 + 6* *R2 := R3 + R2* *R2 := R2 –R1*
(a)	(b)	*(c)*

H.1 Cấp phát thanh ghi cho mã ba địa chỉ bằng giải thuật tô màu đồ thị

H.1(b). Giải thuật sau khi đi qua bước 1. Với S1..S8 là các thanh ghi ký kiệu dùng trong chương trình. Ta nhận xét, với đoạn mã trên nếu muốn thực hiện toàn bộ trên thanh ghi thì phải dùng đến 8 thanh ghi.

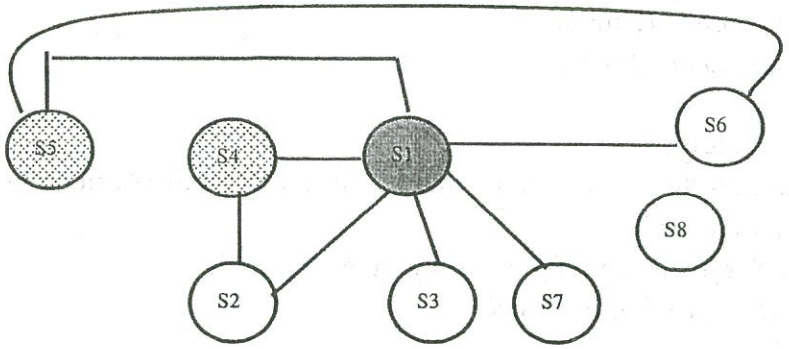
H.1(c). Giải thuật sau khi đã kết thúc, số thanh ghi vật lý cần thiết là: R1, R2, R3.

H.2 trình bày đồ thị giao thoa của tập mã trên.



H.2. Đồ thị giao thoa tương ứng với H.1(b)

Trong bước ba, tiến hành giải thuật tô màu đồ thị, H.3 trình bày kết quả thu được



H.3. Kết quả sau khi áp dụng giải thuật tô màu đồ thị

Dựa vào kết quả trên, tiến hành cấp phát lại các thanh ghi, mỗi màu tượng trưng cho một thanh ghi Ri, sau đó thay các thanh ghi ký hiệu Si bằng Ri, hình H.1(c) biểu diễn kết qủa thu được. So với bước 1 của giải thuật giờ đây chỉ cần 3 thanh ghi thay vì là 8 thanh ghi, vấn đề tối ưu đã được giải quyết.

2. Bài toán tô màu đồ thị với Heuristic

Số thanh ghi trong máy tính là giới hạn. Vấn đề đặt ra là làm sao giải quyết được bài toán với r thanh ghi cho trước?

Bằng phương pháp quy nạp, dễ dàng chứng minh được với một đồ thị G(V, E) có thể tô với số màu r cho trước, khi và chỉ khi đồ thị G(V-v,E) có thể tô với r màu, trong đó v là một nút trong G có bậc nhỏ hơn r[4].

Như vậy bài toán trở về là tô màu đồ thị với số màu (số thanh ghi) cho trước là r. Cách giải quyết vấn đề cũng tương tự ở phần một nhưng chỉ khác trong quá trình xử lý đồ thị giao thoa và stack.

Đồ thị có thể tô với số màu r, thì những nút trong đồ thị phải có bậc nhỏ hơn r. Có bốn phương pháp để giải quyết:

1. Chỉ đưa vào đồ thị những những thanh ghi ký hiệu mà có bậc nhỏ hơn r. Nghĩa là những đối tượng này trong chương trình có số phần tử trong tập Duchain nhỏ hơn r.

2. Với những đối tượng nào có tập Duchain lớn hơn r, dùng các lệnh nạp(load) và lưu trữ (store) để thay thế sau đó mới tiến hành xây dựng đồ thị giao thoa.
3. Sau khi xây dựng xong đồ thị giao thoa, trước hết đưa vào stack những nút có bậc nhỏ hơn r, còn những nút có bậc lớn hơn r thì đánh dấu sau khi đưa vào stack .
4. Sau khi xây dựng xong đồ thị giao thoa, trong quá trình tô màu, nếu số màu đạt tới số thanh ghi cho phép thì giải thuật chấm dứt. Phương pháp này tiết kiệm được thời gian chạy khi số thanh ghi cho trước lớn hơn bốn và số thanh ghi cần trong chương trình lớn hơn nhiều so với số thanh ghi cho phép.

Trong bốn phương pháp trên, chúng tôi chọn phương pháp ba và bốn để thực hiện. *Phương pháp ba bao gồm các bước sau:*

1. Xây dựng đồ thị giao thoa. Nếu số nút trên đồ thị giao thoa nhỏ hơn số màu r, thì mỗi màu tương ứng với một nút, thực hiện năm . Nếu trên đồ thị có những nút có bậc nhỏ hơn số màu r thì thực hiện hai, ngược lại thực hiện ba. Bậc của một nút v trong đồ thị là số cung tới v.
2. Lần lượt lấy các nút có bậc nhỏ hơn số màu r ra khỏi đồ thị và tất cả các cung đi từ nút đang xét đến các nút khác nếu có. Mỗi nút sau khi lấy ra khỏi đồ thị, thì bậc của các nút mà liên kết với nút đang xét sẽ giảm xuống một. Nút vừa lấy ra khỏi đồ thị được đưa vào stack. Nếu trong đồ thị không còn nút có bậc nhỏ hơn r mà đồ thị vẫn chưa rỗng thì thực hiện ba, ngược lại thực hiện bốn.
3. Đồ thị giờ đây chỉ còn các nút có bậc lớn hơn hay bằng r, tìm một nút có bậc cao nhất lấy ra khỏi đồ thị đồng thời loại bỏ các cung tới nó rồi đưa vào stack và đánh dấu nó, sau đó quay lại hai.
4. Lần lượt lấy các nút ra khỏi stack và xây dựng lại đồ thị để thực hiện tô màu, các nút trong bước hai thì luôn luôn tô được, còn các nút trong buớc ba(có đánh dấu) thì giữ nguyên, sau này sẽ dùng lệnh *store* lưu vào bộ nhớ.
5. Dựa vào đồ thị đã được tô màu, cập nhật lại và tối ưu số thanh ghi trong đoạn mã.

Phương pháp bốn bao gồm các bước sau

1. Xây dựng đồ thị giao thoa
2. Duyệt đồ thị theo chiều sâu
3. Tô màu đồ thị, trong quá trình tô, kiểm tra số màu, nếu số màu đạt đến số thanh ghi cho trước r thì giải thuật kết thúc.

Sau đây là một số ví dụ minh họa cho các giải thuật.

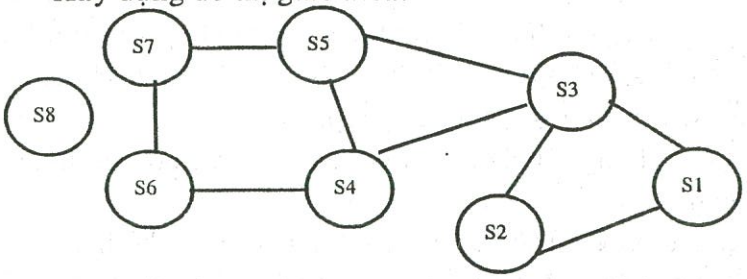
Ví dụ2: Dùng 3 thanh ghi để cấp phát cho đoạn mã sau (trường hợp không đánh dấu thanh ghi ký hiệu trên stack)

Mã	Thanh ghi ký hiệu	*Sau khi đã tối ưu*
b := 3	S1 := 3	*R1 := 3*
t1 := b + 4	S2 := S1 + 4	*R2 := R1 + 4*
t2 := t1 - 4	S3 := S2 – 4	*R3 := R2 – 4*
t3 := t1 + b	S4 := S2 + S1	*R2 := R2 + R1*
t4 := t2 + 3	S5 := S3 + 3	*R1 := R3 + 3*
t5 := t2 + 7	S6 := S3 + 7	*R1 := R3 + 7*
t6 = 7 + 9	S7 := 7 + 9	*R3 := 7 + 9*
t7 = t5 – t6	S8 := S6 – S7	*R2 := R1 – R3*
H.4(a)	H.4(b)	*H.4(c)*

H.4(a). Đoạn mã trung gian ba địa chỉ.

H.4.(b). Giải thuật sau khi đã cấp phát thanh ghi ký hiệu. Nhận xét, với đoạn mã trên nếu muốn thực hiện toàn bộ trên thanh ghi thì phải dùng đến 8 thanh ghi.
H.4(c). Giải thuật sau khi đã kết thúc, số thanh ghi vật lý cần thiết bây giờ chỉ còn ba: R1, R2, R3.

Bước1: Xây dựng đồ thị giao thoa.



H.2. Đồ thị giao thoa tương ứng với H.4(b)

Bậc của các nút trên đồ thị

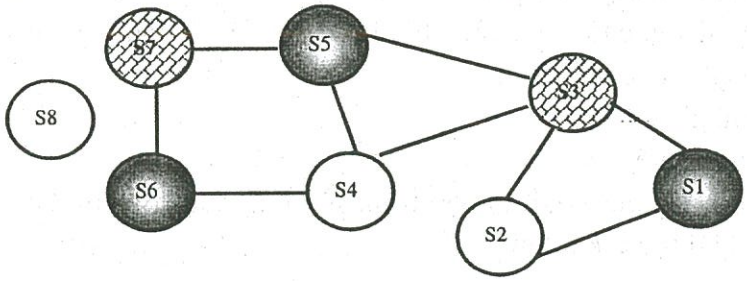
Nút	Bậc
S8	0
S1, S2, S6, S7	2
S5, S4	3
S3	4

Bước 2

Lần lượt lấy các nút có bậc nhỏ hơn 3 ra khỏi đồ thị và đưa vào stack bao gồm S8, S7, S6, S1. Sau đó tới S4, S3, S2 , đồ thị rỗng, thực hiện bước 4, H.6 biểu diễn kết quả thu được.

Bước 5

Cập nhật lại thanh ghi, H.4(c) biểu diễn kết quả thu được.



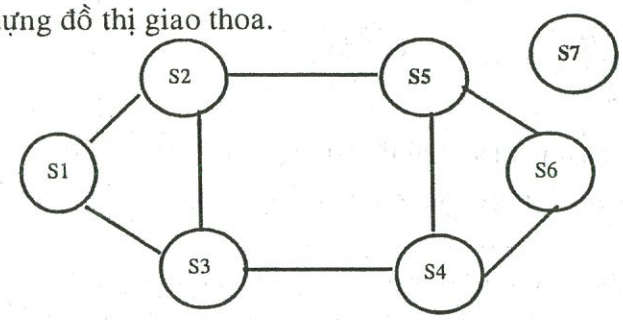
H.6. Kết quả sau khi tô màu đồ thị với heuristic

Ví dụ 3: Dùng 3 thanh ghi để cấp phát cho đoạn mã sau (trường hợp đánh dấu thanh ghi ký hiệu trên stack).

Mã ba địa chỉ	Thanh ghi ký hiệu	*Sau khi đã tối ưu*
a := 3 t1 := a + 6 t2 := a + t1 t3 := a - t2 t4 := t3 + 2 t5 := t1 – t2 t6 = t4 + t3 H.7(a)	S1 := 3 S2 := S1 + 6 S3 := S2 + S1 S4 := S1 - S3 S5 := S4 + 2 S6 := S2 - S3 S7 := S5 + S4 H.7(b)	*R1 := 3* *R2 := R1 + 6* *R3 := R1 + R2* *R1 := R1 - R3* *S5 := R1 + 2* *Store S5* *R3 := R2 – R3* *Load S5* *R2 := S5 + R1* *H.7(c)*

H.7(b). Giải thuật sau khi đã cấp phát thanh ghi ký hiệu. Nhận xét, với đoạn mã trên nếu muốn thực hiện toàn bộ trên thanh ghi thì phải dùng đến 7 thanh ghi.
H.7(c). Giải thuật sau khi đã kết thúc, số thanh ghi vật lý cần thiết bây giờ là ba: R1, R2, R3, còn S5 thì được lưu vào bộ nhớ qua lệnh Store và khi sử dụng thì được lấy từ bộ nhớ qua lệnh Load

Bước 1: Xây dựng đồ thị giao thoa.



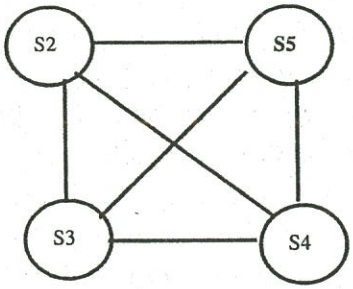
H.8 Đồ thị giao thoa tương ứng với H.7(b)

Bậc của các nút trên đồ thị giao thoa:

Nút	Bậc
S7	0
S1, S6	2
S2, S3, S4, S5	4

Bước 2

Lần lượt lấy S7, S6, S1 ra khỏi đồ thị và đưa vào stack, H.9 biểu diễn kết quả thu được.



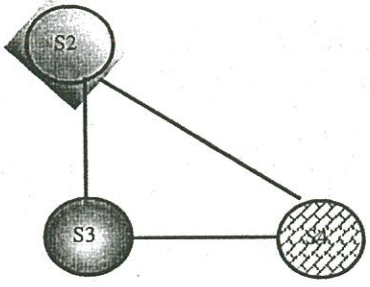
Stack

S1
S6
S7

H.9. Kết quả sau khi thực hiện bước 2

Bước 3

Lấy S5 ra khỏi đồ thị, đưa vào stack, do bậc của S5 bằng với số màu cho trước nên đánh dấu nó trên stack, sau đó quay trở lại bước 2.



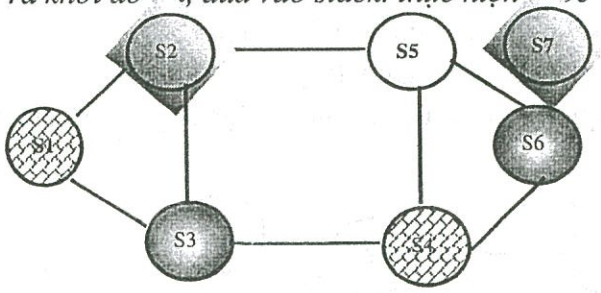
Đánh dấu

Stack

S5 *
S1
S6
S7

H.10. Kết quả sau khi thực hiện bước 3

Lấy S4, S3, S2 ra khỏi đồ thị, đưa vào stack. thực hiện bước 4



H.11. Kết quả sau khi thực hiện bước 4

Bước 5: *Cập nhật lại thanh ghi, H.7(c) biểu diễn kết quả thu được.*

Kết luận:

Quá trình cấp phát thanh ghi tự động bằng giải thuật tô màu đồ thị đã được áp dụng cho hàng loạt máy tính. Chúng tôi đã dùng giải thuật tô màu đồ thị để xây dựng các chương trình cấp phát thanh ghi của một số trình biên dịch và thu được một số kết quả kể cả trong trường hợp số thanh ghi là giới hạn.

OPTIMIZATION FOR REGISTER ALLOCATION USING GRAPH-COLORING ALGORITHM

Ta Duy Cong Chien - Phan Thi Tuoi

ABSTRACT: *One of the most interested problems of the compiler optimization is register allocation and assignment. The problem is addressed is how to minimize traffic between the CPU registers which are usually fast to access.*

The graph coloring algorithm usually results in very effective allocations with a low cost in compilation speed. It views the fact that two objects must be registers at the same time as excluding them from being in the same register. It represents the objects by nodes in a graph and the exclusions (called interferences) by arcs between the corresponding nodes. The nodes may represent real registers also, and the arcs may represent exclusions such as that the base address in a memory access may not be register. Given the graph corresponding to an entire procedure, this method then attempts to color the nodes, with the number of colors equal to the number of available real registers, so that every node is assigned a color that is distinct from those of all the nodes adjacent to it.

TÀI LIỆU THAM KHẢO

[1] Aho, Alfred V., and Jeffrey D.Ullman. Compilers: Principles, Techniques and Tools, Addison – Wesley Publishing Company, 1986.

[2] Steven S, Muchnick. Advanced Compiler Design and Implementation, Morgan Kaufmann Publishers, 1997.

[3] Thomas H. Cormen – Charles E. Leiserson –Ronald L. Rivest. Introdution to Algorithms, The MIT Press, 1990.

[4] Prof.Karen Tomko. ECES 686 Advanced Compiler Optimizations, Spring 2002, http://www.ececs.uc.edu/~ktomko/ACO/index.htm.

[5] Robert Morgan, Building an Optimizing Compiler, The Digital Press, 1998.