지정 레지스터 수의 증가를 최소화하는 레지스터 할당

박승진[0*] 한경숙** 표창우*
홍익대학교 컴퓨터공학과*, 한국산업기술대학교 컴퓨터공학과**
sjpark@cs.hongik.ac.kr[0*], khan@kpu.ac.kr**, pyo@cs.hongik.ac.kr*

Register Allocation Minimally Incrementing the Number of Assigned Registers

Seung-jin Park[0*] Kyung-sook Han** Changwoo Pyo*
Dept. of Computer Engineering, Hongik University* & Korea Polytechinc University**

요 약

지정 레지스터 수의 증가를 최소화하는 레지스터 할당 방법은 컬러링 과정에서 좀 더 적은 수의 레지스터를 사용하도록 하기 위하여 제안된 방법이다. 이 방법은 생존 범위가 서로 복잡하게 얽혀 있을 때 다른 레지스터 할당 알고리즘 보다 우수한 결과를 보였다. Appel의 간섭 그래프들을 사용하여 제시된 레지스터 할당 방법과 Chaitin의 알고리즘을 비교할 때 500개 이상의 에지를 포함하는 그래프중에 29.7%의 그래프에서 레지스터 요구 수를 적게 요구하였다. 전체 그래프를 대상으로 한 실험에서는 9.7%의 그래프에서 Chaitin의 알고리즘 보다 레지스터를 적게 요구하였고, 노드 병합 레지스터 할당 방법보다는 2.2%의 그래프에서 레지스터 요구수의 감소를 보였다. 제시된 알고리즘은 전역 변수의 사용이 많고, 함수 코드의 길이가 긴 프로그램의 실행 성능 개선에 도움이 될 것으로 예상된다.

1. 서 론

현재까지 개발된 레지스터 할당 방법은 Chaitin의 그래프 감축에 의한 그래프 컬러링 방식을 기반으로 개선되어 왔다. Chaitin 이후로 개선된 Briggs의 그래프 컬러링 알고리즘과 노드 병합을 이용한 레지스터 할당 방법을 기반으로 하여 지정 레지스터 수의 증가를 최소화하는 레지스터 할당 방법을 제시하였다. 이 방법은 컬러링 과정에서 좀 더 적은 수의 레지스터를 사용하도록 하기 위하여 제안된 방법이다. 컬러링 단계에 노드 병합의 개념을 도입하여 기존의 방법보다 메모리 접근을 최소화하고 레지스터 사용을 줄임으로써 프로그램 실행 성능을 향상시키기 위함이다[1].

본 논문의 구성은 다음과 같다. 2절에서는 관련 레지스터 할당 방법에 대해 알아보고, 3절에서는 본 논문에서 제시한 컬러링 방법에 대해 설명하였다. 4절에서는 실험을 통하여 본 논문에서 제시한 컬러링 방법의 우수성을 증명하였다. 5절에서는 결론 및 향후 연구 방향을 기술하였다.

2. 관련 연구

Chaitin의 레지스터 할당 방법은 전역 자료 흐름 분석을 통한 생존 범위 추출, 간섭 그래프 구축, 노드 병합, 그래프 감축, 컬러링 단계로 진행되며 그래프 감축에 의한 컬러링을 수행한다[2, 3].

Chaitin의 레지스터 할당 방법을 개선 시킨 Briggs의 알고리즘은 대피 코드의 삽입 시기를 컬러링 단계 이후로 미룸으로써 불필요한 대피 코드의 삽입을 감소 시켰다[4]. 실제 레지스터 수 이상의 간섭 관계를 가지는 경우라도 컬러링이 가능한 경우가 존재하므로 컬러링 단계에서 색을 할당할 수 없는 경우에만 대피 코드를 삽입하도록 하였다.

노드 병합을 이용한 레지스터 할당 방법은 그래프 감축 단계에서 더 이상 감축 가능한 노드가 존재하지 않는 경우 비용에 의해 두 개의 노드를 병합시킴으로써 그래프 감축 단계 수행 중에 병합하는 두 노드와 동시에 간섭하는 노드의 간섭 수를 줄여 그래프 감축이 지속될 가능성을 발생시키는 방법이다[5]. 사용 가능한 실제 레지스터 수보다 간섭 수가 작은 노드가 더 이상 존재하지 않는 경우 병합에 의해 간섭 수를 줄이고자 하는 것이다.

3. 지정 레지스터 수의 증가를 최소화하는 레지스터 할당

간섭 그래프에서 각 노드의 간섭 수는 중간 코드 상에 있는 한 변수의 생존 범위와 같은 시점에 생존하는 변수의 최대 개수를 의미한다. 간섭 수가 큰 노드는 많은 변수와 동시에 생존한다는 의미이므로 다른 노드와 같은 색을 할당받을 가능성이 적다. 하나의 노드가 색을 할당받으려면 기존에 그 색을 할당받은 모든 노드들과 간섭 관계에 있지 않아야 한다. 따라서 하나의 색을 할당받은 노드들의 간섭 관계를 모두 병합한 간섭 관계를 가지고 새로운 노드에 색을 할당할 수 있는지 여부를 결정한다.

기존의 레지스터 할당 방법에서는 간섭 그래프 감축 단계에 의해 노드의 컬러링 순서가 정해지면 컬러링 단계에서는 할당 대상이 되는 레지스터 수의 범위 안에서 할당 가능한 색을 결정한다. 이 때 어떤 색을 할당하느냐에 따라 다음에 색을 할

당받을 노드의 컬러링이 가능할 수도 있고 불가능할 수도 있다. 좀 더 많은 노드를 할당하여 대피 코드의 수를 감소시키기 위한 방법으로 기존의 색을 할당받은 노드들의 간섭 관계를 병합한 간섭 관계에서 간섭 수의 증가에 따른 지정 레지스터 수 증가를 최소화할 수 있는 색을 할당한다.

예를 들어 <그림 1>의 (a)와 같은 간섭 그래프를 컬러링 한다고 가정한다. 이 간섭 그래프를 3-컬러링 하는 과정을 통해 지정 레지스터 수 증가 최소화를 통한 레지스터 할당 방법을 볼 수 있다. <그림 1>에서 사용하는 용어와 그 의미는 다음과 같다.

C : 컬러, n : 노드

$adjacent(n)$: 노드 n과 간섭 관계에 있는 노드들의 집합

$degree(n) = |adjacent(n)|$

$color(n)$: 노드 n에 할당된 색

$$interf(C) = \bigcup_{color(n)=C} adjacent(n)$$

$$Uinterf(C, n) = interf(C) \cup adjacent(n)$$

$$interfNo(C) = |interf(C)|$$

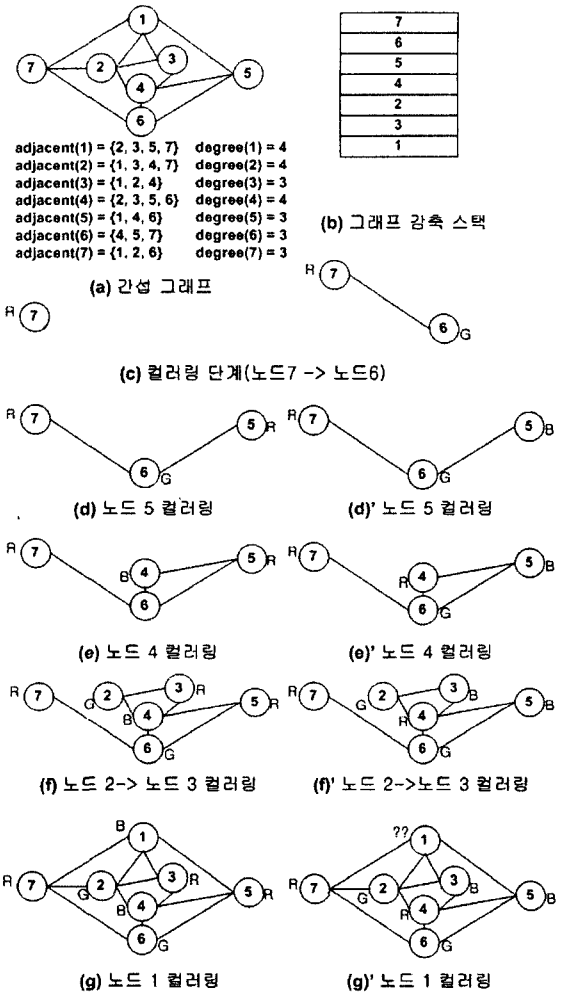
$$incInterfNo(C, n) = |Uinterf(C, n)| - |interf(C)|$$

adjacent(n)은 노드 n과 간섭 관계에 있는 노드들의 집합으로 노드 n과 같은 컬러를 할당받을 수 없는 노드의 집합을 말한다. degree(n)은 노드 n의 간섭 수로 노드 n과 간섭하는 노드들의 수를 나타내며 color(n)은 노드 n에 할당된 색이다. interf(C)는 컬러 C와 간섭 관계에 있는 노드들의 집합으로, 컬러 C를 할당받은 모든 노드들의 간섭 관계를 통합한 것이다. interf(C)에 포함되어 있는 노드들은 컬러 C를 할당받은 노드 중에 간섭 관계에 있는 노드가 존재하는 것이므로 컬러 C를 할당받을 수 없다. 컬러가 정해지지 않은 상태에서 계산하는 Uinterf(C, n)는 노드 n에 컬러 C를 할당했을 경우 컬러 C와 간섭 관계가 되는 노드들의 집합이다. 즉 컬러 C에 포함되어 있는 노드들과 간섭 관계에 있거나 노드 n과 간섭 관계에 있는 노드들의 합집합으로 볼 수 있다. interfNo(C)는 interf(C)에 포함되어 있는 노드의 수로 컬러 C의 간섭 수를 나타내며 incInterfNo(C, n)는 노드 n에 컬러 C를 할당했을 경우 컬러 C의 간섭 수 증가분을 나타낸다. 본 논문에서는 지정 레지스터 수 증가를 최소화하기 위한 간섭 수 증가분을 최소화하는 컬러를 선택하므로 노드를 컬러링 하는 단계에서 incInterfNo(C, n) 값을 최소로 하는 컬러 C를 선택하게 된다.

<그림 1>은 지정 레지스터 수 증가 최소화를 이용한 레지스터 할당의 예를 보여주고 있다. (a)에서는 간섭 그래프와 각 노드의 간섭 관계를 보여주고 있다. 그래프 감축 단계에서 노드들이 (b)와 같은 순서로 감축 스택에 삽입된 상태에서 (c) ~ (g)는 그래프의 컬러링 단계를 보여주고 있다. 노드 7을 컬러링 하는 경우 어떤 컬러를 할당받아도 각 컬러의 간섭 수 증가분은 노드 7의 간섭 수인 3이 된다. 노드 7에 컬러 R을 할당하면

interf(R) = adjacent(7) = {1, 2, 6}, interfNo(R) = 3

이 된다. 따라서 노드 6은 R을 제외한 모든 색을 할당받을 수 있다. 노드 6에 컬러 G를 할당했다고 가정하면 노드 5의 컬러링 시점에서 노드 6과 간섭하므로 노드 7이 할당받은 색인 R과 할당되지 않은 B중 어느 색을 할당할지 선택해야 한다. 노드 5의 할당 시점에서와 같이 할당 가능한 여러 개의 색 중 어느 색을 선택하느냐 하는 문제는 전체 그래프가 요구하는 색의 수 즉, 전체 프로그램이 요구하는 레지스터 수에 영향을 미친다. 레지스터 요구 수가 많아지면 대피 코드를 삽입해야 하는 경우가 많이 발생하며 결과적으로 프로그램의 실행 성능을 저하시킨다. 본 논문에서 제안하는 지정 레지스터 수 증가를 고려한 레지스터 할당 방법은 노드의 할당 시점에서 각 컬러에 이미 할당된 노드들의 간섭 수를 이용한 컬러의 간섭 수 interfNo(C)를 계산하게 된다. 노드 5의 할당 시점에서 간섭 관계에 있는 노드와 간섭 수, 간섭 수의 증가분은 다음과 같다. 이 때 노드 5는 R 또는 B를 할당받을 수 있으므로 노드 5의 R과 B를 할당 받을 경우 각각의 간섭 수를 계산한다.



<그림 1> 간섭 수 증가 최소화를 이용한 그래프 컬러링의 예

interf(R) = adjacent(7) = {1, 2, 6},
interfNo(R) = | interf(R) | = 3
interf(B) = { },
interfNo(B) = | interf(B) | = 0
< 노드 5에 컬러 R을 할당하는 경우 >
Uinterf(R, 5) = interf(R) U adjacent(5) = {1, 2, 4, 6}
incInterfNo(R, 5) = | Uinterf(R) | - | interf(R) | = 1

< 노드 5에 컬러 B를 할당하는 경우 >
Uinterf(B, 5) = interf(B) U adjacent(5) = {1, 4, 6}
incInterfNo(B, 5) = | Uinterf(B) | - | interf(B) | = 3

위의 incInterfNo(R, 5)과 incInterfNo(B, 5) 값에서 볼 수 있듯이 노드 5에 컬러 R을 할당하게 되면 컬러 R의 간섭 수 interfNo(R)은 1만큼 증가한다. 컬러 B를 할당하는 경우에는 B가 처음 할당되는 것이므로 컬러 B의 간섭 수 interfNo(B)가 노드 5의 간섭 수인 3만큼 증가하게 된다. 간섭 수의 증가는 간섭하는 노드의 수가 증가하는 것을 나타내므로 컬러링 되지 않은 노드가 해당 컬러를 할당받을 가능성이 줄어드는 것을 의미한다.

<그림 1>의 (e) ~ (g)는 노드 5의 할당 이후 과정을 나타낸 것이다. 노드 5에 R을 할당하는 경우와 B를 할당하는 경우를 비교하기 위하여 노드 5에 B를 할당하는 경우의 진행 과정을 (e)' ~ (g)'에 도시하였다. (f)'에서 노드 2와 노드 3의 색은 서로 치환 가능하나 (g)'에서 노드 1에 할당할 색이 존재하지 않는다는 사실에는 변함이 없게 된다.

4. 성능평가

지정 레지스터 수 증가를 고려한 레지스터 할당 방법을 구현하여 Appel[6]의 그래프 데이터베이스에서 요구하는 최소 레지스터 수를 산출하였다. Appel의 간섭 그래프 데이터베이스는 그래프 크기의 변화가 다양하고, 많은 간섭그래프들이 소량의 간섭 수를 포함하고 있으며, 53개의 그래프는 비어있다.

[표 1]에서 지정 레지스터 수 증가를 고려한 알고리즘은 DegInc로 표시하였으며 노드 병합 방법을 이용한 레지스터 할당 방법은 Merging으로 표현하였다. 실험 결과 Chaitin의 레지스터 할당 방법과 비교할 때 9.7%, Briggs의 레지스터 할당 방법과 비교할 때 1.02%의 그래프에서 레지스터 요구 수가 감소하는 것을 볼 수 있었다. 노드 병합 알고리즘에 비해서는 2.3%의 그래프에서 적은 레지스터를 사용함을 볼 수 있었다.

기존의 알고리즘보다 레지스터를 적게 요구하는 이유는 할당 가능한 컬러 중 간섭 수를 적게 증가시키는 컬러를 할당하기 때문이다. 간섭 수를 적게 증가시키는 컬러를 할당함으로써 현재 사용되고 있는 컬러에 좀 더 많은 노드를 포함시킬 가능성이 커지게 되어 적은 수의 컬러를 이용하여 컬러링이 가능하게 되는 것으로 추측할 수 있다.

[표 1]의 큰 그래프는 500개이상의 간섭 수를 포함하고 있으며, Appel의 간섭그래프 데이터베이스에서 큰 그래프는 1170개로 구성되어 있다. 기존의 레지스터 할당 방법들과 지정 레지스터 수 증가를 고려한 알고리즘의 큰 그래프 레지스터 요구수를 살펴 본다. 실험 결과 Chaitin의 레지스터 할당

[표 1] 알고리즘에 따른 레지스터 요구수

레지스터 요구수	그래프 수 및 비율	큰 그래프 수 및 비율
Chaitin > DegInc	2704 (9.7%)	347 (29.7%)
Chaitin = DegInc	25,018 (89.6%)	817 (69.8%)
Chaitin < DegInc	199 (0.7%)	6 (0.5%)
Briggs > DegInc	284 (1.02%)	70 (6.0%)
Briggs = DegInc	27,588 (98.8%)	1,078 (92.1%)
Briggs < DegInc	49 (0.18%)	22 (1.9%)
Merging > DegInc	635 (2.3%)	211 (18%)
Merging = DegInc	27,004 (96.7%)	952 (81.4%)
Merging < DegInc	282 (1.0%)	7 (0.6%)

방법과 비교해 보면 29.7%의 큰 그래프에서 레지스터 요구수가 줄어 든다. Briggs의 레지스터 할당 방법과 비교한 경우 6.0% 그리고 노드병합의 경우에는 18%의 큰 그래프에서 레지스터 요구수가 감소하는 것을 볼 수 있었다. 즉, 간섭 수가 많을수록 변화되는 간섭 수가 많아지므로 간섭 수가 적은 그래프보다 레지스터 요구 수에도 많은 변화를 가져오는 것을 확인할 수 있었다.

5. 결론 및 향후과제

본 논문에서는 그래프 감축을 이용한 레지스터 할당 방법에서 실제 레지스터를 할당하는 단계에서의 레지스터 선택에 대한 방법을 제안하였다.

지정 레지스터 수 증가 최소화를 이용한 레지스터 할당 방법은 Appel의 간섭 그래프 데이터베이스[6]를 이용한 실험 결과 레지스터 요구 수 측면에서 성능의 향상을 가져왔다.

앞으로 실제 컴파일러에 적용하여 실행 성능을 비교하는 작업이 이루어져야 할 것이며, 이 논문에서 제시된 방법에 그래프 감축 단계에서의 노드병합 방법을 적용한 레지스터 할당 방법도 연구가 진행되어야 한다.

6. 참고문헌

[1] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, *"Compilers, Principles, Techniqus, and Tools"*, 1986

[2] Gregory J. Chaitin, Mark A. Auslander, Ashok K. Chandra, John Coke, Martin E. Hopkins, Peter W. Markstein, *"Register Allocation via Coloring"*, Computer Languages, 1981

[3] Gregory J. Chaitin, *"Register Allocation & Spilling via Graph Coloring"*, Proceedings of the ACM SIGPLAN '82 Symposium on Compiler Construction, 1982

[4] Preston Briggs, Keith D. Cooper, Ken Kennedy, Linda Torczon, *"Coloring Heuristics for Register Allocation"*, Proceedings of the SIGPLAN '89 Conference on Programming Language Design and Implementation, 1989

[5] Steven R. Vegdahl, *"Using Node Merging to Enhance Graph Coloring"*, PLDI, 1999

[6] Andrew W. Appel, *A Sample Graph Coloring Problems*, URL:http://www.cs.princeton.edu/fac/appel/graphdata, 1996