Tối ưu hóa mã C

Mục tiêu

- Tổng quan về tối ưu hóa mã
- Phân loại các loại tối ưu hóa
- Phương pháp và thuật ngữ tối ưu hóa
- Công cụ phân tích tài liệu
- Các thuật toán phổ biến

Tổng quan về tối ưu hóa mã

- Tối ưu hóa mã bao gồm việc áp dụng các quy tắc và thuật toán để lập trình mã với mục tiêu làm cho mã nhanh hơn, nhỏ hơn, hiệu quả hơn, v.v.
- Thông thường các kiểu tối ưu hóa này xung đột với nhau: ví dụ: mã nhanh hơn thường có kết quả lớn hơn chứ không phải nhỏ hơn
- Việc tối ưu hóa có thể được thực hiện ở nhiều cấp độ (ví dụ: mã nguồn, biểu diễn trung gian) và bởi nhiều bên khác nhau, chẳng hạn như nhà phát triển hoặc trình biên dịch/trình tối ưu hóa

Phân loại tối ưu hóa mã

- ***Tối ưu hóa cục bộ***-Được thực hiện trong một phần của một quy trình.
  - Loại bỏ biểu thức phụ phổ biến (ví dụ: những biểu thức xảy ra khi dịch các chỉ mục mảng sang địa chỉ bộ nhớ).
  - Sử dụng các thanh ghi cho kết quả tạm thời và nếu có thể cho các biến.
  - Thay thế phép nhân và chia bằng các phép tính dịch và cộng.
- ***Tối ưu hóa toàn cầu***-Được thực hiện với sự trợ giúp của phân tích luồng dữ liệu (xem bên dưới) và phân tích thời gian phân chia.
  - Chuyển động mã (nâng) bên ngoài vòng lặp
  - Tuyên truyền giá trị
  - Giảm sức mạnh
- ***Tối ưu hóa giữa các thủ tục***
  - Tối ưu hóa toàn cục cho phép trình biên dịch/trình tối ưu hóa xem xét chương trình tổng thể và xác định cách tốt nhất để áp dụng mức tối ưu hóa mong muốn. Lỗ nhìn trộm cung cấp các tối ưu hóa cục bộ, không tính đến toàn bộ mô hình hoặc điều kiện trong chương trình. Tối ưu hóa cục bộ có thể bao gồm thay thế hướng dẫn.

-Loại bỏ biểu thức phụ phổ biến

Nếu giá trị thu được từ phép tính biểu thức phụ được sử dụng nhiều lần, hãy thực hiện phép tính một lần và thay thế kết quả cho từng phép tính riêng lẻ

Bình thường	Tối ưu hóa

float x = a*min/max + sx;
 	
float y = a*min/max + sy;
	nhiệt độ nổi = a*min/max;
	
	float x = temp + sx;
	
	float y = temp + sy

-lan truyền liên tục

Thay thế các biến dựa vào giá trị không thay đổi bằng chính giá trị đó

Bình thường	Tối ưu hóa

x2 = 5;
 	
x3 = x1 + x2;
	x3 = x1 + 5;

-Sao chép lan truyền

Thay thế nhiều biến sử dụng cùng một giá trị được tính toán bằng một biến

Bình thường	Tối ưu hóa

x2 = x1;
 	
x3 = x1 + x2;
 	
x2 = 3;
	x3 = x1 + x1;
	
	x2 = 3;

Các phương pháp tối ưu hóa (tiếp theo)

-Loại bỏ mã chết

Mã không bao giờ được thực thi có thể bị xóa khỏi tệp đối tượng để giảm dung lượng lưu trữ

kích thước và dấu chân thời gian chạy

-Phân bổ đăng ký toàn cầu

Các biến không trùng lặp về phạm vi có thể được đặt trong sổ đăng ký, thay vì

còn lại trong RAM. Truy cập các giá trị được lưu trong sổ đăng ký nhanh hơn truy cập các giá trị trong RAM

-Cuộc gọi nội tuyến

Một hàm khá nhỏ có thể được thay thế các lệnh máy của nó tại thời điểm

của mỗi lệnh gọi hàm thay vì tạo ra một lệnh gọi thực tế. Điều này trao đổi không gian (kích thước của mã chức năng) để lấy tốc độ (không có chi phí gọi hàm).

Các phương pháp tối ưu hóa (tiếp theo)

-Lập kế hoạch hướng dẫn

Hướng dẫn cho một bộ xử lý cụ thể có thể được tạo ra, mang lại hiệu quả cao hơn mã cho bộ xử lý đó nhưng có thể có vấn đề về tính tương thích hoặc hiệu quả trên bộ xử lý khác. Sự tối ưu hóa này có thể được áp dụng tốt hơn cho các ứng dụng nhúng hệ thống, trong đó loại CPU được biết đến tại thời điểm xây dựng

-Phân tích trọn đời

Một thanh ghi có thể được sử dụng lại cho nhiều biến, miễn là các biến đó không chồng chéo về phạm vi

Các phương pháp tối ưu hóa (tiếp theo)

-Biểu thức bất biến vòng lặp (chuyển động mã)

Các giá trị không thay đổi trong quá trình thực hiện vòng lặp có thể được di chuyển ra khỏi vòng lặp, tăng tốc độ thực hiện vòng lặp.

Bình thường	Tối ưu hóa
cho (int i = 0; i < chiều dài; i++) x[i] += pi + cos(y);	nhiệt độ gấp đôi = pi + cos(y); cho (int i = 0; i < chiều dài; i ++) x[i] += tạm thời;

Các phương pháp tối ưu hóa (tiếp theo)

-Bỏ vòng lặp

Các câu lệnh trong vòng lặp dựa trên các chỉ mục hoặc truy cập tuần tự có thể được lặp đi lặp lại nhiều lần trong thân vòng lặp. Điều này dẫn đến việc kiểm tra vòng lặp có điều kiện ít thường xuyên hơn.

Bình thường	Tối ưu hóa

nhiệt độ gấp đôi = pi + cos(y);
 	
cho (int i = 0; i < chiều dài; i++)
 	
x[i] *= tạm thời;
	nhiệt độ gấp đôi = pi + cos(y);
	
	for (int i = 0; i < chiều dài; i += 2) {
	
	x[i] *= tạm thời;
	
	x[i+1] *= tạm thời;
	
	}

Các phương pháp tối ưu hóa (tiếp theo)

-Giảm sức mạnh

Một số hoạt động nhất định và hướng dẫn mã máy tương ứng của chúng yêu cầu nhiều hơn thời gian để thực hiện hơn các đối tác đơn giản hơn, có thể kém hiệu quả hơn.

Bình thường	Tối ưu hóa
x/4	x >> 2
x*2	x << 1

Các phương pháp tối ưu hóa (tiếp theo)

-Các bước tối ưu hóa kích thước mã

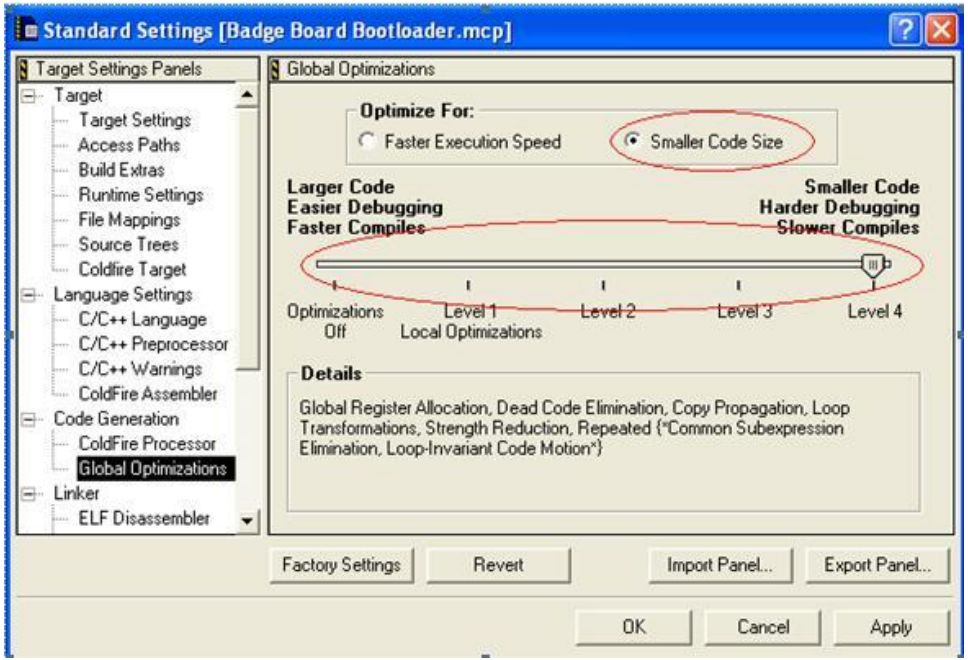
Bước 1. Tối ưu hóa thủ công

- Loại bỏ cửa hàng chết;
- Loại bỏ mã chết;
- Phân tích vòng đời: Một sổ đăng ký có thể được sử dụng lại cho nhiều biến, miễn là các biến đó không trùng nhau về phạm vi;
- Lan truyền liên tục: Thay thế các biến dựa vào một giá trị không thay đổi bằng chính giá trị đó;
- Sao chép lan truyền: Thay thế nhiều biến sử dụng cùng một giá trị được tính toán bằng một biến;
- Không sử dụng cuộc gọi nội tuyến.

Các phương pháp tối ưu hóa (tiếp theo)

-Các bước tối ưu hóa kích thước mã (tiếp)

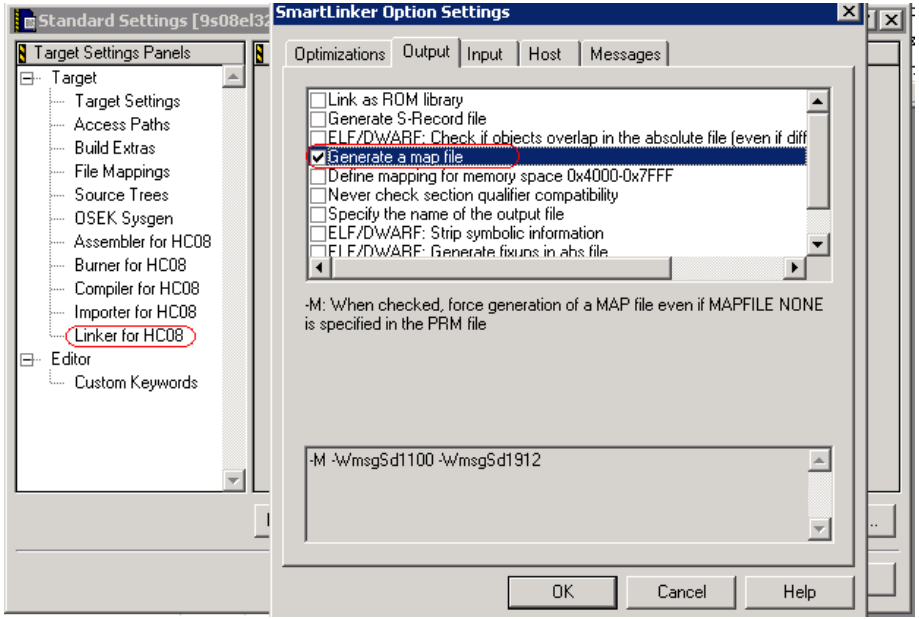
Bước 2. Sử dụng các tùy chọn trình biên dịch (công cụ)



Các phương pháp tối ưu hóa (tiếp theo)

-Các bước tối ưu hóa kích thước mã (Thực hành)

Bước 1.**Phân tích**phân phối mã trong bộ nhớ bằng Linker**tập tin BẢN ĐỒ**



Các phương pháp tối ưu hóa (tiếp theo)

-Các bước tối ưu hóa kích thước mã (Thực hành)

Bước 2. Phân tích tệp MAP liên kết và xác định các đối tượng có nhiều bộ nhớ nhất tiêu thụ

```
*******************************************************************************
MODULE STATISTIC
  Name                                          Data    Code   Const
-------------------------------------------------------------------------------
  Start08.c.o                                      0       7       0
  main.c.o                                         5     113       0
  mc9s08el32.c.o                                 128       0       0
  RTSHC08.C.o (ansiis.lib)                         0     206       0
  lin_cfg.c.o                                    164       0     273
  lin_common_api.c.o                               0     864       0
  lin_common_proto.c.o                             0    3405       0
  lin_j2602_proto.c.o                              0     438       0
  lin_lin21_proto.c.o                              0     639       0
  lin_commontl_api.c.o                             0     624       0
  lin.c.o                                          0     279       0
  lin_lld_slic.c.o                                 4     124       0
  slic_isr.c.o                                     2     333       2
  other                                           64      16       8
```

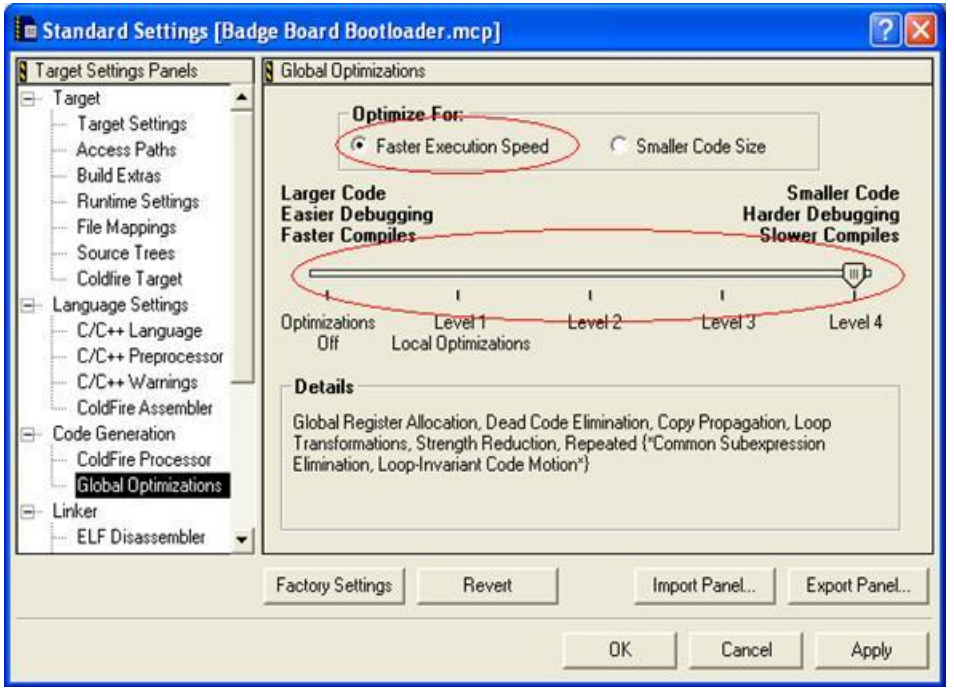
-Các bước tối ưu hóa tốc độ

Bước 1. Tối ưu hóa thủ công

- Loại bỏ mã chết;
- Loại bỏ biểu thức phụ phổ biến;
- Tuyên truyền liên tục;
- Sao chép lan truyền: Thay thế nhiều biến sử dụng cùng một giá trị được tính toán bằng một biến;
- Phân bổ đăng ký toàn cầu;
- Cuộc gọi nội tuyến;
- Lập kế hoạch giảng dạy;
- Vòng lặp các biểu thức bất biến (chuyển động mã);
- Biến đổi vòng lặp;
- Mở vòng lặp;
- Giảm sức mạnh;
- Sử dụng thuật toán nhanh;
- Viết bằng hợp ngữ

-Các bước tối ưu hóa tốc độ (tiếp)

Bước 2. Sử dụng các tùy chọn trình biên dịch (công cụ)



Các phương pháp tối ưu hóa (tiếp theo)

-Các bước tối ưu hóa tốc độ (Thực hành)

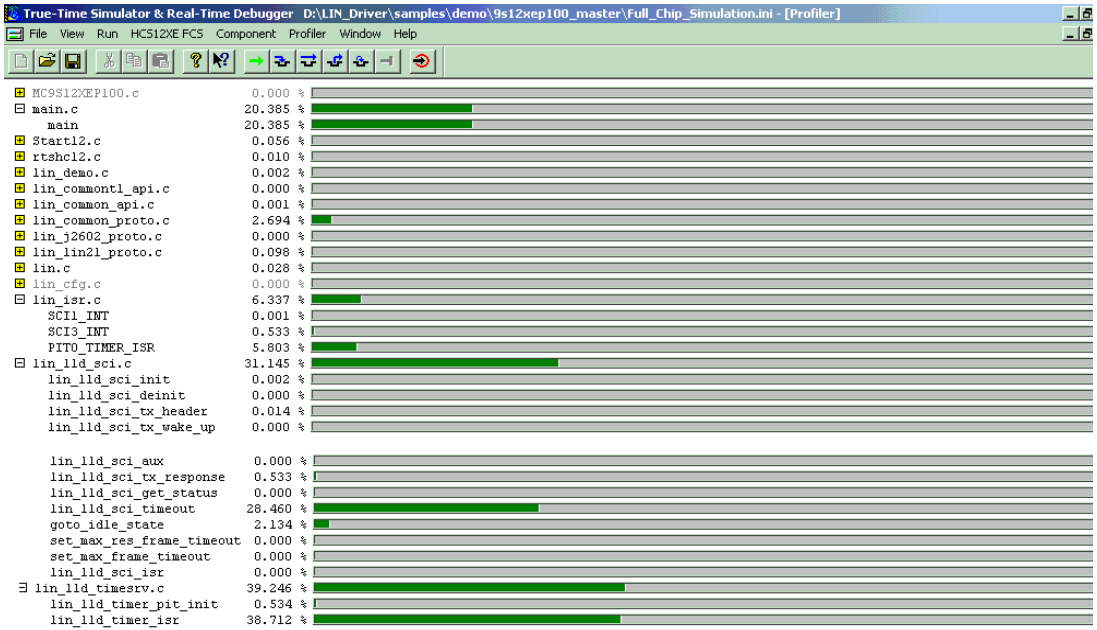
Bước 1. Lập hồ sơ mã

- Lập hồ sơ là quá trình phân tích phần mềm để xác định lượng thời gian, trung bình, một tệp thực thi sẽ sử dụng một lượng mã cụ thể.
- Mặc dù việc lập hồ sơ có thể tiết lộ nhiều thông tin hữu ích về mã của bạn. Hồ sơ xác định các nút thắt cổ chai -- các vùng mã cản trở toàn bộ hiệu suất của hệ thống. Tối ưu hóa cố gắng làm cho những nút cổ chai đó nhanh hơn.
- Hầu hết mã đều tuân theo mô hình thực thi 80/20 - 80% thực thi thời gian được dành để thực thi 20% mã

Các phương pháp tối ưu hóa (tiếp theo)

-Các bước tối ưu hóa tốc độ (Thực hành)

Bước 1. Lập hồ sơ mã (tiếp)



Các phương pháp tối ưu hóa (tiếp theo)

-Các bước tối ưu hóa tốc độ (Thực hành)

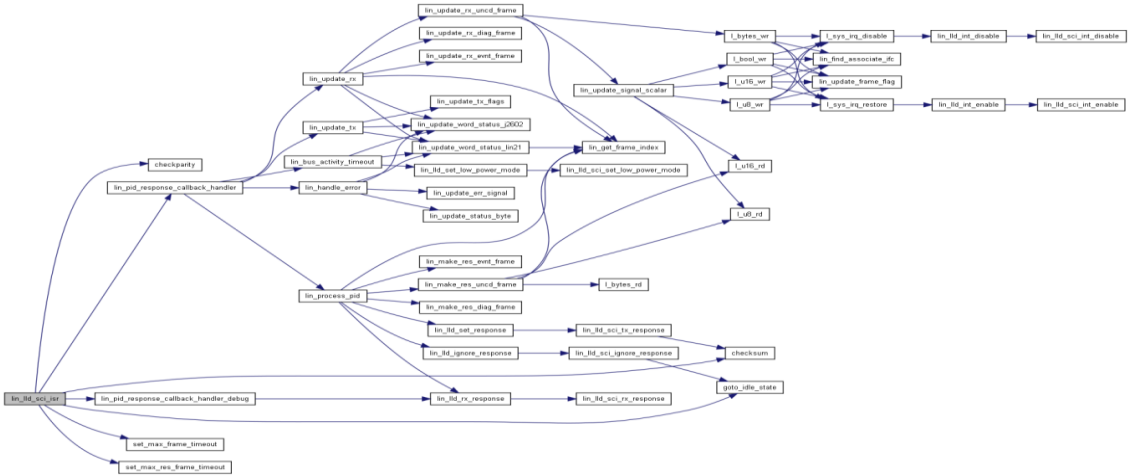
Bước 2. Tối ưu hóa tốc độ

-Chức năng nội tuyến/Hủy vòng lặp/Giảm cường độ http://www.eventhelix.com/realtimemantra/Basics/OptimizingCAndCPPCode.htm

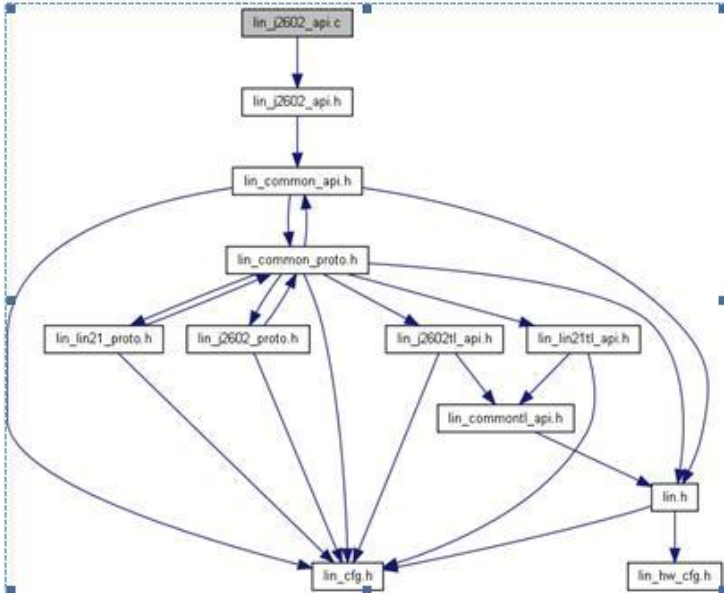
-Giảm thiểu chi phí định kỳ của dịch vụ gián đoạn bằng cách giữ cho ISR đơn giản

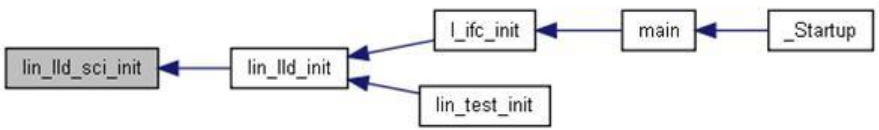
Quy trình dịch vụ gián đoạn (ISR) cải thiện hiệu suất và dễ dàng bảo trì. Vì ISR hoạt động không đồng bộ nên chúng vốn rất khó gỡ lỗi; vì vậy hãy giữ nhiệm vụ của họ ở mức tối thiểu.

Cố gắng chuyển mọi nhiệm vụ xử lý dữ liệu hoặc quản lý dữ liệu ra khỏi ISR và vào chương trình chính. Sau đó, ISR sẽ chỉ lấy dữ liệu từ phần cứng, đặt nó vào bộ đệm, nâng cờ sẵn sàng cho dữ liệu và kích hoạt lại ngắt.



Công cụ phân tích tài liệu (Doxygen)





Thuật toán chung

- Tìm kiếm và sắp xếp
  - Tìm kiếm nhị phân
  - Sắp xếp bong bóng
  - Sắp xếp lựa chọn
  - Sắp xếp chèn
- Dây
  - Thuật toán vét cạn

Cảm ơn

Hỏi đáp