Kỹ thuật dịch ngược

Ngày 23 tháng 11 năm 2016

Lời cam đoan

Lời cảm ơn

Tóm tắt luận văn

Mục lục

Danh sách hình vẽ

1 Giới thiệu

Chương này nhằm mục đích giới thiệu về bài toán sẽ được giải quyết trong luận văn và các khái niệm liên quan. Đầu chương sẽ nói về kỹ thuật dịch ngược, các ứng dụng của nó và những khó khăn trong quá trình dịch ngược. Phần tiếp theo trình bày bài toán đặt ra và các thách thức khi giải quyết bài toán. Phần cuối cùng tóm tắt cấu trúc của luận văn.

1.1 Kỹ thuật dịch ngược và ứng dụng

1.2 Bài toán đặt ra

Như đã đề cập ở phần trên, mục tiêu của luận văn là nghiên cứu về trình dịch ngược từ mã assembly lên mã cấp cao, các bài toán cần phải giải quyết và hiện thực giải pháp. Vì mã assembly cho kiến trúc máy khác nhau có những đặc điểm khác nhau, và đi cùng với đó là những vấn đề khác nhau cần giải quyết, nên giới hạn của luận văn sẽ là trình dịch ngược từ mã assembly 8051. Việc chọn kiến trúc máy 8051 là do 2 nguyên nhân sau:

- Chip 8051 đã xuất hiện trên thị trường từ lâu, hiện tại đã không còn được sản xuất. Tuy nhiên, vẫn còn nhiều hệ thống được chạy trên đây và cần phải chuyển đổi chúng sang một kiến trúc máy khác hiện đại hơn.
- Chip 8051 có một số đặc điểm khác biệt so với các con chip khác trên thị trường. Vì vậy việc dịch ngược từ mã 8051 sẽ gặp nhiều khó khăn hơn, vấn đề phải giải quyết phức tạp hơn.

Các đặc điểm khác biệt của 8051 gồm có:

- Trong khi hầu hết các kiến trúc máy khác sử dụng kiểu dữ liệu byte là kiểu dữ liệu nhỏ nhất, thì 8051 cho phép lập trình viên truy xuất tới mức bit trong một số thanh ghi và kèm theo đó là các câu lệnh xử lý bit. Tuy nhiên, các thanh ghi này của 8051 cũng có thể được truy xuất ở mức byte bình thường (xem ví dụ ở đoạn mã **??**).
- Một số assembler của 8051 cho phép sử dụng tên biến. Biến này dùng để lưu các giá trị hằng số, hằng số này thường là địa chỉ một vùng nhớ kích thước 1 byte (trong luận văn này sẽ gọi tắt là biến byte) hoặc đại diện cho bit của thanh ghi (gọi tắt là biến bit). Khi lập trình, người ta thường sử dụng biến byte và biến bit này theo bộ, nghĩa là chỉ khi thanh ghi được load vào giá trị vùng nhớ quy định bởi biến byte, thì các biến bit cùng bộ mới được sử dụng (xem ví dụ ở đoạn mã **??**).

Từ các đặc điểm trên, ta có thể thấy bài toán lớn nhất đặt ra trong luận văn này sẽ là tìm ra được mối liên hệ giữa biến byte và biến bit trong chương trình, lấy được các bộ biến byte và biến bit đúng. Có 2 cách để biết được điều này:

- Đưa ra quy định về việc khai báo biến byte và biến bit. Hiện nay, ở phần khai báo, các lập trình viên có thể khai báo các biến theo thứ tự tuỳ ý, và cũng không có quy định nào bắt buộc họ phải có phần comment chỉ rõ các biến byte và biến bit nào là cùng một bộ. Ta có thể đưa ra các mẫu khai báo cho biến byte và biến bit để trình dịch ngược có thể biết được các bộ biến bằng cách đọc theo mẫu mà không cần phân tích gì thêm. Tuy nhiên, sau khi đã xác định được các bộ biến này, cần có thêm một bước kiểm tra mã chương trình để đảm bảo rằng nguyên tắc sử dụng biến byte và biến bit được tuân thủ (Kiểm tra kiểu - Type checking). Giải pháp này có ưu điểm là đơn giản, dễ hiện thực nhưng gây bất tiện cho người dùng vì phải chuyển đổi bộ mã hiện tại về theo mẫu quy định.

- Dựa vào phân tích luồng dữ liệu của chương trình, tìm ra được địa chỉ vùng nhớ được load vào thanh ghi tại thời điểm sử dụng biến bit và từ đó suy ra biến byte cùng bộ với biến bit đó (Suy luận kiểu - Type inference). Với cách làm này, không cần phải thay đổi đoạn mã gốc. Tuy nhiên cách hiện thực sẽ phức tạp hơn nhiều vì có rất nhiều cách load dữ liệu vùng nhớ vào thanh ghi như: dùng trực tiếp hằng số, dùng trực tiếp biến byte, trung gian qua một thanh ghi khác... (xem ví dụ ở đoạn mã **??**)

Cả hai giải pháp này đều được hiện thực trong từng giai đoạn của luận văn và sẽ được trình bày trong các chương tiếp theo.

1.3 Cấu trúc luận văn

Luận văn sẽ gồm 6 chương như sau:

- Chương 1: Giới thiệu về kỹ thuật dịch ngược, bài toán đặt ra trong luận văn và cấu trúc của luận văn
- Chương 2: Nêu lên một số kiến thức cơ bản và các nghiên cứu liên quan đến luận văn
- Chương 3: Trình bày giải pháp Kiểm tra kiểu - Type checking
- Chương 4: Trình bày giải pháp Suy luận kiểu - Type inference
- Chương 5: Đánh giá kết quả của luận văn thông qua các mẫu thử (testcase)
- Chương 6: Kết luận

2 Kiểm tra kiểu - Type checking

Như đã giới thiệu ở phần trước, giải pháp Kiểm tra kiểu gồm có các bước sau:

- Đưa ra mẫu khai báo biến byte và biến bit. Chỉnh sửa chỉnh dịch ngược để đọc vào mẫu khai báo này và biết được những biến byte và biến bit nào cùng một bộ.
- Thực hiện phân tích dữ liệu để biết được giá trị của thanh ghi ACC tại một thời điểm nào đó có phải là giá trị của một vùng nhớ cố định hay không. Tại giai đoạn này của luận văn, kỹ thuật phân tích dữ liệu được sử dụng là reaching definition.
- Tại các câu lệnh sử dụng bit, nếu giá trị của thanh ghi ACC là của một vùng nhớ cố định, thì kiểm tra xem giá trị địa chỉ của vùng nhớ đó đang được lưu bởi biến nào. Nếu biến đó được khai báo cùng bộ với biến bit thì xem như tuân thủ đúng nguyên tắc sử dụng, nếu không thì báo lỗi.
- Nếu toàn bộ chương trình đều tuân thủ đúng nguyên tắc sử dụng biến byte - biến bit thì đưa các bộ biến byte - biến bit về dạng union ở ngôn ngữ cấp cao.

Chương này sẽ lần lượt trình bày các bước trên.

2.1 Mẫu khai báo biến byte và biến bit

Mẫu khai báo biến byte và biến bit được quy định như sau: Đây là mẫu khai báo khi muốn gom các biến vào cùng một bộ, ngoài ra, chương trình vẫn chấp nhận việc khai báo riêng lẻ từng biến. Như vậy, cần chỉnh lại phần parser của trình dịch ngược để chấp nhận cấu trúc mới này. Trong trình dịch ngược, cần có một cấu trúc mới để lưu trữ thông tin của các bộ biến byte và biến bit này. Như vậy, sau giai đoạn parser, ta đã thu được bộ biến theo khai báo của người dùng. Tuy nhiên, nguyên tắc sử dụng bộ biến này là không bắt buộc trong quá trình lập trình 8051, và các assembler cũng không hề kiểm tra việc người dùng có tuân thủ nguyên tắc này không. Vì vậy, trước khi chuyển hoá các bộ biến này sang cấu trúc union ở mã đầu ra, ta cần phải kiểm tra đoạn mã của người dùng sử dụng các bộ biến như thế nào.

2.2 Phân tích Reaching definitions

Mục đích của phân tích Reaching definitions là biết được ở một thời điểm của chương trình, các câu lệnh khai báo nào đang còn hiệu lực, hay nói cách khác là gía trị của các biến đang là gì. Như vậy, khi áp dụng vào trình dịch ngược, ta sẽ biết được tại thời điểm sử dụng biến bit, giá trị của thanh ghi ACC đang được định nghĩa như thế nào. Nếu đó là một câu lệnh load một vùng nhớ có địa chỉ quy định bởi biến byte, thì ta kiểm tra tiếp biến byte đó có cùng bộ với biến bit đang sử dụng hay không. Lưu ý là ta chỉ xét trường hợp thanh ghi ACC được load một giá trị vùng nhớ (biểu diễn ở ngôn ngữ cấp cao là pointer). Các bước

phân tích reaching definitions áp dụng vào cấu trúc dữ liệu hiện tại của trình dịch ngược như sau:

Tuy nhiên, phân tích Reaching definitions chỉ cho biết được câu lệnh khai báo có hiệu lực của một biến tại một thời điểm chương trình, chứ không cho biết được giá trị thực sự của biến đó. Nếu vế phải của câu lệnh khai báo này chỉ đơn giản là pointer của một biến byte, thì ta sẽ dễ dàng kiểm tra được. Nhưng ngoài ra, biểu thức nằm trong dấu pointer khi dịch ngược từ mã assembly 8051 lên có thể là các trường hợp sau đây:

- Một biểu thức có hai vế, các vế của biểu thức có thể là một biến, một thanh ghi hoặc một hằng số.
- Một thanh ghi, giá trị của thanh ghi có thể được khai báo ở các câu lệnh trước đó.

Với phương pháp Reaching definitions, ta sẽ không thể xử lý được với các trường hợp phức tạp hơn như trên. Một phân tích khác sẽ được áp dụng trong giai đoạn tiến hành giải pháp Suy luận kiểu. Còn với giai đoạn sơ khai này của luận văn, ta sẽ giả thuyết rằng ở đoạn mã đầu vào chỉ có các câu lệnh gán vùng nhớ đơn giản cho ACC với một biến byte duy nhất.

2.3 Kiểm tra cách sử dụng bộ biến trong toàn bộ chương trình và sinh mã

Sau khi đã có thông tin về reaching definitions ở tất các các điểm của chương trình, ta sẽ chuyển sang kiểm tra các ở các câu lệnh sử dụng bit.

Quá trình kiểm tra gồm các bước sau:

Nếu ở bất cứ câu lệnh sử dụng biến bit nào, nguyên tắc đưa ra bị vi phạm, thì trình dịch ngược sẽ hiện thông báo lỗi trên console cho người dùng. Sau đó, đoạn mã kiểm tra lỗi vẫn tiếp tục chạy để kiểm tra các câu lệnh khác nhằm tạo thuận lợi cho người dùng trong quá trình sửa lỗi chương trình đầu vào. Tuy nhiên, nếu có lỗi thì trình dịch ngược sẽ không sinh mã đầu ra vì sẽ không thể xử lý được sinh các union như thế nào.

Sau quá trình kiểm tra lỗi, trình dịch ngược sẽ tiến hành sinh ra các union ở mã đầu ra và thay thế các biến bit trong chương trình bằng các biểu thức truy xuất union, cũng như bỏ các câu lệnh gán cho ACC và thay thế biến ACC bằng biến byte. Nếu chương trình đầu vào tuân thủ đúng nguyên tắc sử dụng biến byte - biến bit, ta sẽ chuyển đổi các biến này thành một cấu trúc dữ liệu khác phù hợp hơn ở dạng ngôn ngữ cấp cao. Như đã phân tích từ trước ở chương 1, cấu trúc dữ liệu đó là union. Khi lập trình ở dạng mã assembly, lập trình viên không được phép xử lý các vùng nhớ trực tiếp mà phải thông qua thanh ghi, tuy nhiên, khi đã chuyển đổi về dạng ngôn ngữ cấp cao, ta có thể sử dụng trực tiếp tên biến trong các câu lệnh mà không cần trung gian qua thanh ghi nữa. Điều này giúp mã đầu ra dễ hiểu và trong sáng hơn.

Đặc điểm của giải pháp này là ta đã biết trước các bộ biến, nên ngay ở giai đoạn mã trung gian, ta sẽ thể hiện các biến bit dưới dạng truy xuất của union.

Như vậy, ta không cần phải xử lý lại các biến bit này ở giai đoạn sinh mã đầu ra nữa.

Như vậy, với giải pháp Kiểm tra kiểu này, ta đã có sẵn thông tin về bộ biến ngay từ đầu và chỉ cần kiểm tra xem người lập trình có tuân thủ đúng quy tắc không trước khi sinh ra mã ở ngôn ngữ cấp cao. Tuy nhiên, giải pháp còn nhiều hạn chế như phương pháp phân tích dữ liệu chưa đạt độ chính xác cao, cần người dùng phải chỉnh sửa lại khai báo theo mẫu quy định... Chính vì vậy, giai đoạn sau của luận văn đã phát triển một giải pháp mới có độ chính xác cao hơn và không cần chỉnh sửa mã đầu vào của người dùng, đó là giải pháp Suy luận kiểu. Giải pháp này sẽ được trình bày ở chương tiếp theo.

Tài liệu