(12) **BẢN MÔ TẢ SÁNG CHẾ THUỘC BẰNG ĐỘC QUYỀN SÁNG CHẾ**

(19) **Cộng hòa xã hội chủ nghĩa Việt Nam (VN)**
CỤC SỞ HỮU TRÍ TUỆ

(11) **1-0020249**

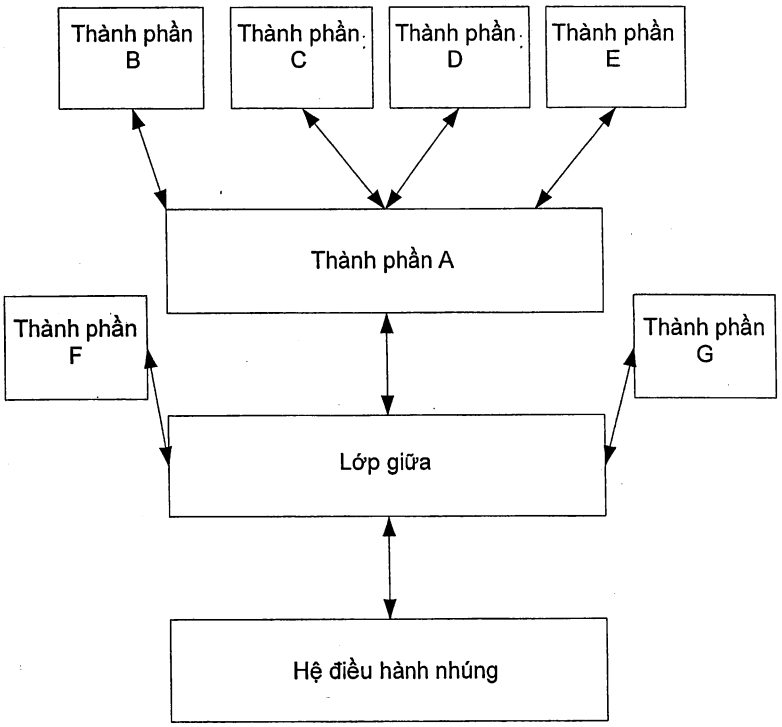
(51)[7] **G06F 9/44**, 9/445

(13) **B**

(21) 1-2011-01396
(22) 18.08.2009
(86) PCT/CN2009/073324 18.08.2009
(87) WO2010/057388A1 27.05.2010
(30) 200810180934.2 18.11.2008 CN
(45) 25.01.2019 370
(43) 25.09.2011 282
(73) TENCENT TECHNOLOGY (SHENZHEN) COMPANY LIMITED (CN)
Room 403, East Block 2, SEG Park, Zhenxing Road, Futian District, Shenzhen city 518044, Guangdong Province, P. R. China
(72) WU, Zhihua (CN), GU, Jia (CN), QIU, Xuan (CN)
(74) Công ty TNHH Sở hữu trí tuệ VIPATCO (VIPATCO CO., LTD.)

(54) PHƯƠNG PHÁP LIÊN KẾT ĐỘNG CHƯƠNG TRÌNH TRONG THIẾT BỊ NHÚNG VÀ THIẾT BỊ NHÚNG

(57) Sáng chế đề xuất phương pháp liên kết động chương trình trên thiết bị nhúng và thiết bị nhúng. Thiết bị nhúng bao gồm một lớp giữa mà bao gồm ít nhất một lớp chương trình ứng dụng để liên kết bao gồm ít nhất một lớp. Chương trình ứng dụng được biên dịch thành các mã độc lập vị trí (PICs), trong đó các vị trí lưu trữ của các phần mã khác trong PICs có được bằng việc biên dịch các hàm khác nhau của chương trình ứng dụng không chèn lên nhau; tất cả PICs được sao chép vào bộ nhớ của hệ điều hành nhúng; một con trỏ lớp được tạo ra, và con trỏ lớp được chuyển tới một lớp thuộc lớp giữa và lớp của chương trình ứng dụng để liên kết động chương trình ứng dụng.



Lĩnh vực đề cập của sáng chế

Sáng chế đề cập tới kỹ thuật nhúng, và cụ thể là phương pháp liên kết động chương trình trong thiết bị nhúng và thiết bị nhúng.

Tình trạng kỹ thuật của sáng chế

Như đã biết, liên kết chương trình bao gồm liên kết tĩnh và liên kết động. Khi liên kết tĩnh được chọn, địa chỉ của chương trình ứng dụng được xác định trong khi biên dịch. Nếu nhiều chương trình ứng dụng tham chiếu các hàm trong thư viện hàm hiện tại, thì số thư viện hàm cần được sao chép vào bộ nhớ bằng với số lượng chương trình ứng dụng, điều này sẽ làm lãng phí không gian bộ nhớ. Trong liên kết động, các hàm trong thư viện hàm được định vị thông qua thư viện nhập. Thư viện nhập chỉ lưu trữ mô tả của mối liên hệ tham chiếu giữa các hàm, mà không lưu trữ mã thực tế, do vậy có thể tiết kiệm được không gian bộ nhớ.

Hiện tại, trong các sản phẩm nhúng như điện thoại di động, liên kết động của chương trình được thực hiện bằng việc sử dụng chế độ định dạng liên kết và thực thi (ELF), thể hiện trên Hình 1.

Hình 1 là sơ đồ thể hiện liên kết động một chương trình bằng việc sử dụng chế độ ELF theo kỹ thuật hiện tại.

Như được thể hiện trên Hình 1, khi liên kết động của một chương trình được thực thi bằng việc sử dụng chế độ ELF, thì chương trình ứng dụng được biên dịch thành tệp tin ELF trước tiên, sau đó định vị lại hàm và biến được thực hiện cho tệp tin ELF bằng việc sử dụng chương trình ELF LOADER, tiếp theo tệp tin ELF mà việc định vị lại đã được thực hiện được tải vào hệ điều hành nhúng, và cuối cùng hệ điều hành nhúng sẽ liên kết các hàm và biến theo vị trí của hàm và biến đã định vị lại, để thực hiện chương trình liên kết động.

Khi chương trình liên kết động được thực hiện bằng việc sử dụng chế độ ELF, chương trình ELF LOADER sẽ thực hiện quy tắc quan trọng, mà được dùng để định vị lại địa chỉ của các hàm và biến trong chương trình ứng dụng với địa chỉ trong hệ điều hành nhúng. Cụ thể, chương trình ELF LOADER sẽ bung thông tin có giá trị từ tệp tin ELF, tính toán địa chỉ sử dụng sau khi định vị lại hàm và biến trong tệp tin ELF, và sau đó thực hiện việc định vị lại cho các hàm và biến theo địa chỉ đó.

Như đã thấy, khi chương trình liên kết động được thực hiện với chế độ ELF, do cần phải bung thông tin có giá trị từ tệp tin ELF và thực hiện xử lý thông tin đó, nên tốc độ của chương trình liên kết động bị chậm lại và tài nguyên tính toán bị lãng phí.

Bản chất kỹ thuật của sáng chế

Với quan điểm trên, sáng chế đề cập đến thiết bị nhúng và phương pháp liên kết động một chương trình trong thiết bị nhúng, để cải thiện tốc độ của chương trình liên kết động.

Giải pháp kỹ thuật của sáng chế được thực hiện như sau.

Phương pháp liên kết động chương trình trong thiết bị nhúng bao gồm: trong đó thiết bị nhúng bao gồm một lớp giữa, lớp giữa bao gồm ít nhất một lớp, mỗi lớp được đóng gói từ ít nhất một giao diện hàm của hệ điều hành nhúng và đáp ứng với bảng hàm ảo, và các phần tử của bảng hàm ảo tương ứng với ít nhất một giao diện hàm của hệ điều hành nhúng, chương trình ứng dụng được liên kết bao gồm ít nhất một lớp, mỗi lớp được đóng gói từ ít nhất một giao diện hàm của chương trình ứng dụng và tương ứng với bảng hàm ảo, và các phần tử của bảng hàm ảo tương ứng với ít nhất một giao diện hàm của chương trình ứng dụng; phương pháp bao gồm:

biên dịch chương trình ứng dụng thành các mã độc lập vị trí (PICs), trong đó lưu trữ vị trí của các phần mã khác nhau trong PICs có được bằng việc biên dịch các hàm khác nhau của chương trình ứng dụng không chèn lên nhau; sao lưu tất cả PICs vào bộ nhớ của hệ điều hành nhúng;

tạo con trỏ lớp, và chuyển con trỏ lớp giữa lớp của lớp giữa và lớp của chương trình ứng dụng tới chương trình ứng dụng liên kết động.

Thiết bị nhúng bao gồm: lớp giữa;

lớp giữa bao gồm ít nhất một lớp, mỗi lớp được đóng gói từ ít nhất một giao diện hàm của hệ điều hành nhúng và tương ứng với bảng hàm ảo, các phần tử trong bảng hàm ảo tương ứng với ít nhất một giao diện hàm; và

lớp giữa được cấu hình để thực hiện phương pháp ở trên.

Như đã thấy, lớp giữa được đóng gói từ hệ điều hành nhúng và bao gồm một hoặc nhiều lớp. Mỗi lớp được đóng gói từ nhiều giao diện hàm của hệ điều hành nhúng và tương ứng với bảng hàm ảo. Các phần tử của bảng hàm ảo tương ứng với các giao diện hàm. Chương trình ứng dụng được kết nối bao gồm một hoặc nhiều lớp, mỗi lớp được đóng gói từ nhiều giao diện hàm của chương trình ứng dụng và tương ứng với bảng hàm ảo. Các phần tử của bảng hàm ảo tương ứng với các giao diện hàm của chương trình ứng dụng. Chương

trình ứng dụng được biên dịch thành PICs, lưu trữ các vị trí của các phần mã khác nhau trong PICs có được bằng việc biên dịch các hàm khác nhau của chương trình ứng dụng để không chèn lên nhau, sao cho chương trình ứng dụng có thể hoạt động trong bất kỳ không gian lưu trữ nào, và các hàm của chương trình ứng dụng được viện dẫn một cách độc lập. Tất cả các PICs được sao lưu vào bộ nhớ của hệ điều hành nhúng; một con trỏ lớp được tạo ra, và con trỏ lớp được chuyển giữa lớp của lớp giữa và lớp của chương trình ứng dụng tới chương trình ứng dụng liên kết động.

Khi con trỏ lớp được tạo ra, các giao diện hàm mà tương ứng với các phần tử của bảng hàm ảo tương ứng với con trỏ lớp được tạo, thực chất các hàm được đưa ra cùng các địa chỉ ảo. Lúc này, các hàm tương ứng với các lớp của lớp giữa và các hàm tương ứng với chương trình ứng dụng được liên kết động thông qua bảng hàm ảo. Như vậy, với việc tạo ra con trỏ lớp và chuyển con trỏ lớp giữa lớp của lớp giữa và lớp của chương trình ứng dụng, liên kết động của chương trình ứng dụng được thực hiện.

Do liên kết động của chương trình được thực hiện bằng việc tạo con trỏ lớp và chuyển con trỏ lớp giữa lớp của lớp giữa và lớp của chương trình ứng dụng, khi tệp tin nhận được bằng việc biên dịch chương trình ứng dụng được tải vào bộ nhớ, không cần phải bung thông tin từ tệp tin để tính toán lại vị trí của mã biên dịch trong bộ nhớ, để giảm thời gian thực hiện liên kết động của chương trình và tăng tốc độ liên kết động của chương trình.

Mô tả vắn tắt các hình vẽ

Hình 1 là lưu đồ thể hiện liên kết động một chương trình với chế độ ELF theo kỹ thuật hiện tại.

Hình 2 là lưu đồ thể hiện cấu trúc thiết bị nhúng theo ví dụ của sáng chế.

Hình 3 là sơ đồ liên kết động chương trình ứng dụng trong thiết bị nhúng thể hiện trên Hình 2.

Hình 4 là sơ đồ thể hiện tương tác giữa lớp giữa và chương trình ứng dụng theo ví dụ của sáng chế.

Hình 5 thể hiện môi trường phát triển của chương trình ứng dụng động trong chương trình tin nhắn nhanh (IM) làm nền tảng.

Mô tả chi tiết sáng chế

Để thực hiện mục tiêu, các giải pháp kỹ thuật và bản chất của sáng chế rõ hơn, sáng chế sẽ được thể hiện chi tiết dưới đây cùng với các hình vẽ đi kèm và các ví dụ.

Hình 2 là sơ đồ thể hiện cấu trúc thiết bị nhúng theo ví dụ của sáng chế. Như được thể hiện trên Hình 2, thiết bị nhúng bao gồm một lớp giữa.

Trong ví dụ của sáng chế, lớp giữa được đóng gói từ hệ điều hành nhúng và bao gồm các lớp, mỗi lớp tương ứng với một bảng hàm ảo, và các phần tử của bảng hàm ảo tương ứng với các giao diện hàm của hệ điều hành nhúng.

Khi lớp giữa được đóng gói từ hệ điều hành nhúng, các giao diện hàm thường được dùng bởi chương trình ứng dụng thuộc tất cả các hàm của hệ điều hành nhúng được đóng gói thành một lớp giữa. Hàm chức năng mà liên quan đến hiệu quả được đóng gói thành một lớp. Ví dụ, các giao diện của các hàm dùng để vẽ, các giao diện của các hàm soạn thảo ký tự, các giao diện của các hàm kết nối mạng và các giao diện của các hàm truyền dẫn thông tin được đóng gói thành lớp giữa, các giao diện hàm vẽ và các giao diện hàm soạn thảo ký tự được đóng gói thành lớp xử lý tài liệu, và các giao diện các hàm kết nối mạng và các giao diện hàm truyền dẫn thông tin được đóng gói thành lớp mạng. Mỗi lớp tương đương với một thành phần, và như vậy lớp giữa thực tế là bao gồm nhiều thành phần.

Hình 3 là sơ đồ liên kết động chương trình ứng dụng trong thiết bị nhúng được thể hiện trên Hình 2.

Như được thể hiện trên Hình 3, thủ tục của liên kết động chương trình ứng dụng bao gồm: giai đoạn biên dịch và tạo tệp tin BIN, giai đoạn tải xuống và lưu trữ tệp tin BIN, giai đoạn tải tệp tin BIN và giai đoạn liên kết động chương trình ứng dụng.

Chương trình ứng dụng được liên kết động trong hệ điều hành được thể hiện trên Hình 2 bao gồm một hoặc nhiều lớp, mỗi lớp được đóng gói từ nhiều giao diện hàm của chương trình ứng dụng và tương ứng với một bảng hàm ảo, các phần tử trong bảng hàm ảo tương ứng với các giao diện hàm của chương trình ứng dụng. Tương tự với lớp của lớp giữa, các hàm tương ứng với liên kết trong lớp của chương trình ứng dụng có liên quan đến nhau. Trong chương trình ứng dụng, mỗi lớp tương đương với một thành phần. Do đó, thực thi chương trình ứng dụng thông qua lớp thực tế tương đương với việc chia chương trình ứng dụng thành nhiều thành phần.

Nhận thấy rằng, bốn vấn đề sau đây cần được giải quyết cho liên kết động chương trình trên thiết bị nhúng: vấn đề về nhập hàm, vấn đề về xuất hàm, vấn đề tham chiếu hàm thứ cấp và vấn đề vị trí biến toàn cục.

Vấn đề nhập hàm chính là cách tham chiếu hàm của hệ điều hành nhúng thông qua lớp giữa bởi chương trình ứng dụng được liên kết động; vấn đề xuất hàm chính là cách tham chiếu hàm của chương trình ứng dụng thông qua lớp

giữa bởi hệ điều hành nhúng; vấn đề tham chiếu hàm thứ cấp chính là cách tham chiếu hàm thứ cấp của chương trình ứng dụng bởi chương trình ứng dụng hoặc lớp giữa; và vấn đề định vị biến toàn cục chính là, nếu chương trình ứng dụng có biến toàn cục, cách định vị lại biến toàn cục của chương trình ứng dụng trong thiết bị nhúng sau khi chương trình ứng dụng được liên kết tới thiết bị nhúng.

Vấn đề nhập hàm và vấn đề xuất hàm liên quan chính đến giai đoạn biên dịch và tạo tệp tin BIN và giai đoạn liên kết động chương trình, vấn đề tham chiếu hàm thứ cấp và vấn đề định vị biến toàn cục liên quan chính đến giai đoạn biên dịch và tạo tệp tin BIN.

Lấy bốn giai đoạn được thể hiện trên Hình 3 làm ví dụ, phương pháp liên kết động chương trình trong thiết bị nhúng được thể hiện trên Hình 2 được mô tả theo đặc tính của thiết bị nhúng ARM.

1) Giai đoạn biên dịch và tạo tệp tin BIN

Trước tiên, người lập trình viết chương trình ứng dụng ở dạng lớp, và đảm bảo rằng chương trình ứng dụng bao gồm các lớp, mỗi lớp được đóng gói từ một hoặc nhiều các giao diện hàm của chương trình ứng dụng và tương ứng với một bảng hàm ảo, các phần tử của mỗi bảng hàm ảo tương ứng với các giao diện hàm khác nhau, nó tương đương với chương trình ứng dụng được chia thành nhiều thành phần, và mỗi thành phần được biên dịch thành một lớp.

Sau đó, chương trình ứng dụng được biên dịch thành các PICs, và PICs được tạo ở dạng tệp tin BIN. PICs chính là các mã có thể làm việc ở bất cứ không gian lưu trữ nào mà không phụ thuộc vào vị trí lưu trữ cố định.

Cụ thể, khi chương trình ứng dụng được biên dịch bằng việc sử dụng trình biên dịch ARM ADS1.2, các mã của chương trình được biên dịch thành PICs thông qua lựa chọn biên dịch acps/ropi, ở đây ảnh hưởng của lựa chọn ropi cho thấy rằng vị trí của các mã chỉ đọc được tạo ra là độc lập.

Do chương trình ứng dụng được biên dịch thành PICs, ở giai đoạn tải tệp tin BIN, thì tệp tin BIN được tải vào bất kỳ vị trí lưu trữ nào của bộ nhớ thiết bị nhúng.

Do chương trình ứng dụng được chia thành nhiều thành phần và lớp giữa được thực hiện thông qua các thành phần, chương trình ứng dụng và lớp giữa sẽ thực hiện nhập và xuất hàm chức năng thông qua tương tác giữa các thành phần và thông qua các lớp. Điều này được mô tả chi tiết trong giai đoạn liên kết động chương trình cách thực hiện nhập và xuất hàm thông qua các lớp bởi chương trình ứng dụng và lớp giữa.

Đối với vấn đề tham chiếu hàm thứ cấp, các hàm của chương trình ứng dụng được làm việc một cách độc lập với nhau thông qua biên dịch các hàm của chương trình ứng dụng thành PICs mà các hàm là độc lập và vị trí lưu trữ của các mã tương ứng với các hàm khác nhau mà không bị chèn lên nhau trong khi biên dịch và tạo tệp tin BIN, do đó thuận tiện cho tham chiếu từng hàm và không gây ảnh hưởng tới các hàm khác.

Cụ thể, khi chương trình ứng dụng được biên dịch bằng trình biên dịch ARM ADS1.2, các hàm của chương trình ứng dụng được biên dịch thành PICs thông qua lựa chọn biên dịch ZO, các PICs là độc lập và không chèn lên nhau.

Ví dụ, giả sử có một chương trình ứng dụng như sau:

```
int appMain(Ishell*ptr,void**ppobj)
{
  fun1();
  return 0;
}
void fun1(void)
{
}
```

Giả sử hàm appMain luôn được biên dịch tại địa chỉ khởi tạo 0x00000000 của tệp tin BIN và chương trình ứng dụng được biên dịch thông qua lựa chọn acps/ropi và lựa chọn ZO, thì kết quả biên dịch được cho trong Bảng 1.

Bảng 1

Hàm	Địa chỉ
appMain	PC+ 0x00
fun1	PC+0x0A

Trong Bảng 1, PC là địa chỉ khởi tạo của tệp tin BIN, và 0x0A là địa chỉ dịch chuyển của hàm fun1.

Như được thấy trong Bảng 1, sau khi chương trình ứng dụng được biên dịch thành PICs mà có các hàm độc lập và vị trí lưu trữ của các mã tương ứng với các hàm khác nhau không chèn lên nhau, việc nhảy giữa các hàm là nhảy tương đối, vị trí sau khi nhảy được xác định bởi địa chỉ dịch chuyển. Bằng cách này, mỗi hàm của chương trình ứng dụng được tham chiếu bởi các hàm bên trong chương trình ứng dụng và các hàm bên ngoài chương trình ứng dụng.

Đối với vấn đề vị trí biến toàn cục, hai chế độ xử lý sẽ được dùng. Ở chế độ xử lý thứ nhất, biến toàn cục được đặt trong hàm cấu trúc của chương trình ứng dụng, và như vậy biến toàn cục trong hàm cấu trúc cũng được biên dịch thành PICs khi cấu trúc được biên dịch thành PICs. Ở chế độ xử lý khác, biến toàn cục được biên dịch thành Dữ liệu độc lập vị trí (PID). Trong quá trình xử lý hình thành, nhiều chương trình ứng dụng được tải vào thiết bị nhúng cùng một lúc như được thể hiện trên Hình 2; do biến toàn cục được lưu trữ trong hàm cấu trúc của chương trình ứng dụng, nên cần phải lấy lại biến toàn cục từ hàm cấu trúc khi biến toàn cục được sử dụng, và như vậy thao tác rất phức tạp. Ở quá trình xử lý sau, do biến toàn cục được biên dịch thành PID, biến toàn cục được lưu trữ bên ngoài hàm cấu trúc, và nói chung được lưu trữ tiếp theo không gian lưu trữ của RAM mà các PICs đã định vị, để tránh truy cập chéo dữ liệu.

Khi chương trình ứng dụng được biên dịch bằng trình biên dịch ARM ADS1.2, các phần dữ liệu của chương trình ứng dụng được biên dịch thành PID thông qua lựa chọn biên dịch acps/rWpi. PDI chính là phần dữ liệu được sao chép vào không gian bộ nhớ của RAM và được hoạt động. Thông qua lựa chọn biên dịch acps/rWpi, các phần dữ liệu của chương trình ứng dụng được biên dịch thành PID, và PID được lưu trữ trong không gian bộ nhớ [R9] + OFFSET.[R9] tương ứng với giá trị của thanh ghi R9, OFFSET là giá trị địa chỉ dịch chuyển. Để tránh truy cập chéo dữ liệu, giá trị thanh ghi R9 được thiết lập là giá trị bằng với địa chỉ khởi tạo của BUFFER cộng với kích thước tệp tin BIN, nghĩa là biến toàn cục được lưu trữ theo tệp tin BIN có được bằng việc biên dịch chương trình ứng dụng.

Do thanh ghi R9 là cần thiết khi biến toàn cục được biên dịch thành PID, các chương trình ứng dụng chỉ được tải độc lập khi chế độ định vị biến toàn cục sau đó được sử dụng, thực chất là có thể tải nhiều chương trình ứng dụng vào cùng một thời điểm.

Ngoài ra, ở giai đoạn biên dịch và tạo tệp tin BIN, cần phải biên dịch hàm gốc của chương trình ứng dụng tại địa chỉ khởi đầu của tệp tin BIN, thực chất

là biên dịch điểm đầu vào của chương trình ứng dụng tại địa chỉ khởi đầu của tệp tin BIN, để đảm bảo định vị từng thành phần của chương trình ứng dụng khi liên kết động chương trình. Cụ thể, điểm đầu vào của chương trình được thiết lập là trước tiên khi chương trình ứng dụng được biên dịch.

Khi chương trình ứng dụng được biên dịch với trình biên dịch ARM ADS1.2, thì chương trình ứng dụng được biên dịch thành tệp tin BIN thông qua phương pháp ngón tay. Địa chỉ hàm của các mã được tạo ra thông qua phương pháp ngón tay nói chung là dịch đi 32 bít (tức 4 byte) so với địa chỉ hàm của mã được tạo thông qua phương pháp cánh tay. Khi hệ điều hành nhúng tham chiếu chương trình ứng dụng thông qua lớp giữa lần đầu tiên, thì con trỏ điểm đầu vào của chương trình ứng dụng sẽ trỏ tới vị trí địa chỉ đầu tiên của bộ đệm mà tệp tin BIN được định vị sau khi dịch đi 4 byte và chương trình ứng dụng được thực hiện, ví dụ, nếu địa chỉ ban đầu của tệp tin BIN là 0x8000, thì con trỏ điểm đầu vào sẽ trỏ tới 0x8004.

Khi chương trình ứng dụng được biên dịch thông qua phương pháp ngón tay, mỗi lệnh trong chương trình được biên dịch thành 2 byte (mỗi lệnh được biên dịch thành 4 byte nếu chương trình ứng dụng được biên dịch thông qua phương pháp cánh tay), điều này làm giảm kích thước của tệp tin BIN, tăng tốc độ xử lý của chương trình ứng dụng và tiết kiệm năng lượng tiêu thụ.

Tóm lại, trong giai đoạn biên dịch và tạo tệp tin BIN, chương trình ứng dụng được biên dịch thành tệp tin BIN, điểm đầu vào của chương trình ứng dụng được định vị tại địa chỉ ban đầu của tệp tin BIN; chương trình ứng dụng được biên dịch thành các PICs, không gian lưu trữ của PICs có được bằng việc biên dịch các hàm khác nhau của chương trình ứng dụng không chèn lên nhau; biến toàn cục trong chương trình ứng dụng được biên dịch thành PID, hoặc biến toàn cục được lưu trữ trong hàm cấu trúc của chương trình ứng dụng.

2) Giai đoạn tải xuống và lưu trữ tệp tin BIN

Ở giai đoạn tải xuống và lưu trữ tệp tin BIN, thì tệp tin BIN được sao chép vào tệp tin hệ thống một cách trực tiếp.

3) Giai đoạn tải tệp tin BIN

Khi tệp tin BIN được tải ra, thì tệp tin BIN được sao chép vào bộ nhớ từ tệp tin hệ thống trực tiếp mà không cần phân tích tệp tin BIN hoặc tính toán theo thông tin trong tệp tin BIN.

4) Giai đoạn liên kết động chương trình

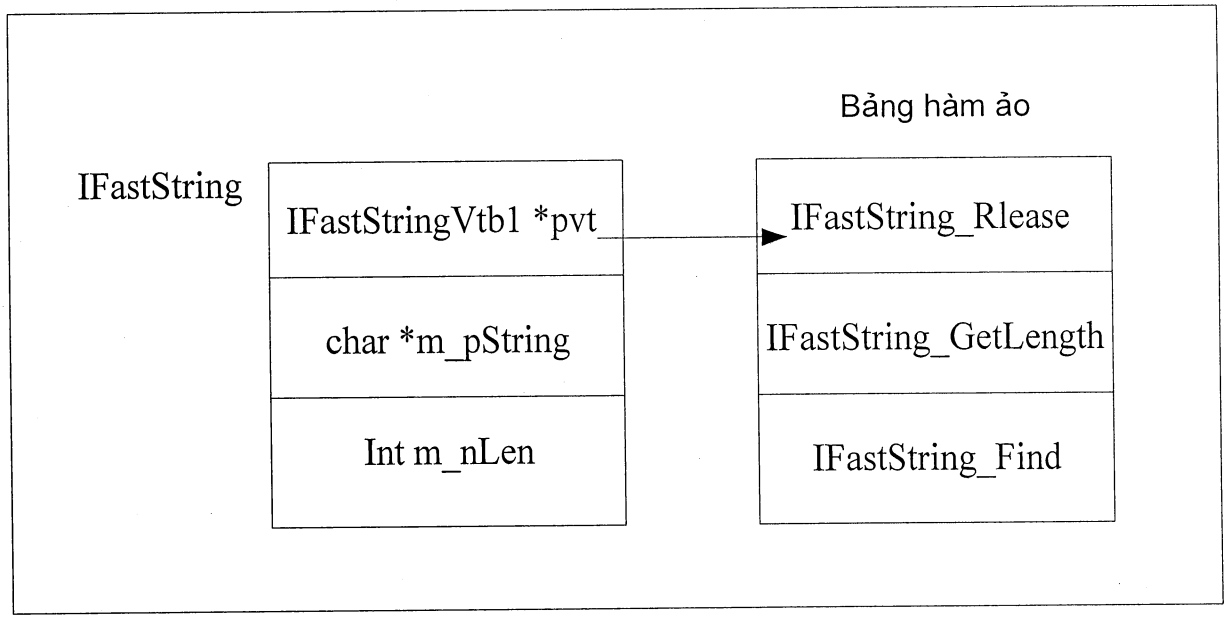
Ở giai đoạn liên kết động chương trình, một con trỏ lớp thực tế được tạo ra, và con trỏ lớp được chuyển giữa lớp của lớp giữa và lớp của chương trình ứng dụng, để thực hiện nhập và xuất các hàm.

Đối với vấn đề nhập hàm, bằng việc tạo con trỏ lớp của lớp giữa, chương trình ứng dụng thực hiện tạo con trỏ lớp trỏ tới lớp của lớp giữa trong đó các hàm của hệ điều hành nhúng được định vị, khởi tạo các hàm tương ứng thông qua các phần tử tương ứng trong bảng hàm ảo tương ứng với lớp của lớp giữa, để tham chiếu các hàm của hệ điều hành nhúng.

Ví dụ, lớp giữa bao gồm lớp IFastString, lớp IFastString có chứa con trỏ của bảng hàm ảo (pvt), và bảng hàm ảo có chứa ba hàm thứ cấp IFastString_Realase, IFastString_GetLength và IFastString_Find.

Bảng 2 là sơ đồ cấu trúc của lớp IfastString

Bảng 2



Khi chương trình ứng dụng tham chiếu các hàm của lớp IFastString, con trỏ lớp IFastString được tạo ra, và ba thành phần IFastString_Realase, IFastString_GetLength và IFastString_Find trong bảng hàm ảo mà con trỏ lớp được đưa ra với địa chỉ hàm thực tế. Bằng cách này, chương trình ứng dụng được định vị cho ba hàm thứ cấp IFastString_Realase, IFastString_GetLength và IFastString_Find trong hệ điều hành nhúng.

Như vậy, khi chương trình ứng dụng tham chiếu các hàm của hệ điều hành nhúng, các con trỏ lớp khác nhau trỏ đến các lớp thuộc lớp giữa được tạo ra thông qua lớp quản lý, ví dụ, lớp Shell, lớp được tạo theo ID của lớp thuộc lớp giữa, các giao diện hàm tương ứng với các phần tử của bảng hàm ảo trong lớp được khởi tạo, và chương trình ứng dụng sẽ định vị các giao diện hàm tương ứng với các phần tử trong bảng hàm ảo mà con trỏ lớp trỏ tới thông qua

con trỏ lớp được tạo ra. ID của lớp thuộc lớp giữa được xác định khi lớp giữa được biên dịch, và khi chương trình ứng dụng tham chiếu lớp của lớp giữa, chương trình ứng dụng sẽ chuyển ID của lớp được tham chiếu tới lớp quản lý, và lớp quản lý sẽ tạo con trỏ lớp tương ứng với ID của lớp và chuyển con trỏ lớp tới chương trình ứng dụng.

Như vậy, điểm đầu vào của chương trình ứng dụng được thể hiện theo dạng sau:

```
int  appMain(Shell*ptr,void**ppobj)

{

return 0;

}
```

Thông qua việc tạo con trỏ lớp và các lớp, chương trình ứng dụng làm cho con trỏ lớp trỏ tới bảng hàm ảo, thực chất là trỏ tới lớp của lớp giữa, và tham chiếu hàm tương ứng với giao diện hàm mà tương ứng với phần tử trong bảng hàm ảo.

Đối với vấn đề xuất hàm, do chương trình ứng dụng được biên dịch ở dạng lớp, thực chất là chương trình ứng dụng bao gồm các lớp, mỗi lớp tương ứng với một bảng hàm ảo, các phần tử trong bảng hàm ảo tương ứng với các giao diện hàm. Chương trình ứng dụng tạo các phần tử trong bảng hàm ảo tương ứng với lớp thông qua việc tạo con trỏ lớp của chương trình ứng dụng, và chuyển con trỏ lớp được tạo ra tới lớp giữa để lưu trữ. Lớp giữa trực tiếp tham chiếu giao diện hàm của lớp của chương trình ứng dụng khi cần tham chiếu các hàm của chương trình ứng dụng, để thực hiện thủ tục xuất hàm.

Đối với thủ tục nhập hàm, chương trình ứng dụng tạo con trỏ lớp trỏ tới lớp của lớp giữa thông qua lớp quản lý. Đối với thủ tục xuất hàm, mỗi lớp của chương trình ứng dụng được tạo khi chương trình ứng dụng được biên dịch, như vậy tải tệp tin BIN được tạo ra bằng việc biên dịch chương trình ứng dụng vào lớp giữa (thực chất là tải tệp tin BIN) tương đương với việc sao chép lớp được tạo của chương trình ứng dụng vào lớp giữa một cách trực tiếp; nếu hệ điều hành nhúng tham chiếu lớp của chương trình ứng dụng, thực tế là để thực hiện xuất hàm, lớp của chương trình ứng dụng mà được sao chép vào lớp giữa được tham chiếu trực tiếp.

Tóm lại, trong sáng chế, lớp giữa được đóng gói từ hệ điều hành nhúng, và chương trình ứng dụng được chia thành nhiều thành phần thông qua các

lớp, để thực hiện tương tác giữa lớp giữa và chương trình ứng dụng thông qua các thành phần, thể hiện trên Hình 4.

Hình 4 là sơ đồ thể hiện tương tác giữa lớp giữa và chương trình ứng dụng theo ví dụ của sáng chế.

Trên Hình 4, khi chương trình ứng dụng cần tham chiếu các hàm của hệ điều hành nhúng, thì con trỏ lớp sẽ trỏ tới lớp của lớp giữa được tạo ra để thực hiện tương tác thành phần; khi cần tham chiếu chương trình ứng dụng, thì lớp giữa sẽ tham chiếu chương trình ứng dụng thông qua con trỏ lớp được lưu trữ của chương trình ứng dụng, và con trỏ lớp của chương trình ứng dụng được tạo bởi chính chương trình ứng dụng và được lưu trữ ở lớp giữa.

Sau khi lớp giữa được đóng gói từ hệ điều hành nhúng, do tương tác giữa các thành phần của lớp giữa và các thành phần của chương trình ứng dụng được thực hiện thông qua các lớp, chương trình ứng dụng tương ứng được phát triển trên lớp giữa.

Hình 5 thể hiện môi trường phát triển chương trình ứng dụng động rađiô mà chương trình IM sử dụng làm nền tảng.

Như được thể hiện trên Hình 5, môi trường phát triển bao gồm một hệ điều hành nhúng, động lực chính của chương trình ứng dụng động (SDK) và chương trình ứng dụng động. SDK là lớp giữa được nói đến ở trên được đóng gói từ hệ điều hành nhúng, và chương trình ứng dụng động là chương trình ứng dụng nói đến ở trên được biên dịch thành dạng lớp.

Chương trình ứng dụng động thể hiện trên Hình 5 bao gồm thành phần A để thực hiện chức năng IM cơ bản, thành phần B để thực hiện trò chơi IM, thành phần C để thực hiện chức năng con vật nuôi IM, thành phần D để thực hiện chức năng viđêô của IM, thành phần E để thực hiện chức năng âm nhạc IM và thành phần F và G để thực hiện ứng dụng mở rộng. Thành phần A, thành phần F và thành phần G tương tác với lớp giữa thông qua các lớp, thành phần B tới E được tải và hoạt động sau khi thành phần A được tải, có thể nói rằng, các thành phần được hoạt động cùng lúc, thành phần B tới E sẽ truyền thông với các lớp của thành phần A thông qua các lớp của các thành phần B tới E, để thực hiện thủ tục đăng nhập vào (plug-in) của chương trình ứng dụng.

Trên Hình 5, SDK và hệ điều hành nhúng được biên dịch và tải xuống một vi mạch cùng một lúc. Chương trình ứng dụng liên kết động được viết ở dạng lớp để thực hiện các hàm ứng dụng cụ thể, và được biên dịch độc lập bằng phương pháp được thể hiện trên Hình 3, được tải xuống tệp tin hệ thống

của vi mạch, và được sao chép vào bộ nhớ của hệ điều hành nhúng khi cần phải tải, và SDK và chương trình ứng dụng tương tác với nhau thông qua lớp.

Như đã thấy từ các giải pháp kỹ thuật ở trên, sau khi chương trình ứng dụng được biên dịch thành tệp tin BIN và được tải xuống, khi chương trình ứng dụng cần để hoạt động, thì chỉ cần sao lưu trực tiếp chương trình ứng dụng vào bộ nhớ, và các vấn đề về nhập và xuất hàm tiếp theo là được thực hiện tự động thông qua việc tạo con trỏ lớp thực tế và chuyển con trỏ lớp giữa lớp giữa và chương trình ứng dụng, mà không cần phân tích hoặc tính toán tệp tin BIN, để cải thiện tốc độ kết nối động chương trình.

Hơn nữa, không gian lưu trữ bị chiếm dụng bởi tệp tin BIN có được bằng biên dịch chương trình ứng dụng là một trong ba không gian lưu trữ bị chiếm dụng bởi tệp tin ELF có được bằng việc biên dịch chương trình, và như vậy không qian lưu trữ được tiết kiệm nếu như chương trình ứng dụng được biên dịch thành tệp tin BIN.

Ngoài ra, tương tác giữa lớp giữa và chương trình ứng dụng được thực hiện thông qua lớp, và tương tác được dựa trên các thành phần, để thuận tiện cho việc cập nhật thành phần của chương trình ứng dụng, thành phần cập nhật của lớp giữa, hoặc cập nhật đồng bộ thành phần của chương trình và lớp giữa, thậm chí các thành phần của lớp giữa hoặc chương trình ứng dụng có thể được dùng để xây dựng ứng dụng mới, để thực thi plug-in của chương trình ứng dụng. Thông qua việc xây dựng thiết bị ứng dụng đúng đắn thì việc phát triển chương trình ứng dụng trở nên đơn giản, đó là lợi ích để phát triển chương trình ứng dụng của hệ điều hành điện thoại di động không thông minh.

Trên đây chỉ là thể hiện các phương án của sáng chế và không được dùng để hạn chế phạm vi bảo hộ của sáng chế này. Bất kỳ sự thay đổi hoặc thay thế tương đương hoặc cải tiến theo tinh thần và phạm vi của sáng chế cũng nằm trong phạm vi bảo hộ của sáng chế.

Yêu cầu bảo hộ

1. Phương pháp liên kết động chương trình ứng dụng trong thiết bị nhúng, trong đó thiết bị nhúng bao gồm lớp giữa, lớp giữa bao gồm ít nhất một lớp (class), mỗi lớp được đóng gói từ ít nhất một giao diện hàm của hệ điều hành nhúng và tương ứng với bảng hàm ảo, các phần tử của bảng hàm ảo tương ứng với ít nhất một giao diện hàm của hệ điều hành nhúng, mỗi chương trình ứng dụng được liên kết bao gồm ít nhất một lớp, mỗi lớp được đóng gói từ ít nhất một giao diện hàm của chương trình ứng dụng và tương ứng với bảng hàm ảo, các phần tử của bảng hàm ảo tương ứng với ít nhất một giao diện hàm của chương trình ứng dụng, phương pháp này bao gồm các bước:

biên dịch chương trình ứng dụng thành mã độc lập vị trí (PICs), trong đó các vị trí lưu trữ của các phần mã khác nhau trong PICs mà có được bằng việc biên dịch các hàm khác nhau của chương trình ứng dụng không bị chèn lên nhau; sao chép tất cả PICs vào bộ nhớ của hệ điều hành nhúng;

tạo con trỏ lớp, và chuyển con trỏ lớp của một lớp thuộc lớp giữa và lớp của chương trình ứng dụng tới liên kết động chương trình ứng dụng;

trong đó PICs là tệp tin nhị phân BIN; và

công đoạn biên dịch bao gồm các bước:

biên dịch chương trình ứng dụng bằng chế độ Thumb;

tạo con trỏ lớp và chuyển con trỏ lớp của một lớp thuộc lớp giữa và lớp của chương trình ứng dụng để liên kết động chương trình ứng dụng bao gồm:

tạo con trỏ điểm đầu vào của chương trình ứng dụng mà trỏ tới vị trí có được bằng việc dịch địa chỉ ban đầu của bộ đệm mà tệp tin BIN của chương trình ứng dụng được định vị đi 4 byte, và vận hành điểm đầu vào, khi hệ điều hành nhúng gọi ra chương trình ứng dụng thông qua lớp giữa lần đầu tiên.

2. Phương pháp theo điểm 1, trong đó chương trình ứng dụng bao gồm biến toàn cục, phương pháp này còn bao gồm bước:

biên dịch biến toàn cục của chương trình ứng dụng thành dữ liệu độc lập vị trí (PID), và lưu trữ PID theo PICs, để thực hiện định vị lại biến toàn cục trong quá trình liên kết động chương trình ứng dụng.

3. Phương pháp theo điểm 1, trong đó chương trình ứng dụng bao gồm biến toàn cục và hàm cấu trúc, phương pháp này còn bao gồm các bước:

lưu trữ biến toàn cục của chương trình ứng dụng vào hàm cấu trúc của chương trình ứng dụng;

biên dịch chương trình ứng dụng thành các mã PICs bao gồm: biên dịch hàm cấu trúc thành PICs.

4. Phương pháp theo điểm 1, trong đó việc biên dịch chương trình ứng dụng thành PICs, trong đó các vị trí lưu trữ của các phần mã khác nhau trong PICs có được bằng việc biên dịch các hàm khác nhau của chương trình ứng dụng không chèn lên nhau bao gồm bước:

biên dịch chương trình ứng dụng bằng cách lựa chọn biên dịch ACPS/ROPI và lựa chọn biên dịch ZO trong trình biên dịch ARM ADS1.2 của hệ điều hành nhúng.

5. Phương pháp theo điểm 2, trong đó việc biên dịch biến toàn cục của chương trình ứng dụng thành PID bao gồm bước:

biên dịch chương trình ứng dụng với lựa chọn biên dịch ACPS/RWPI trong trình biên dịch ARM ADS1.2 của hệ điều hành nhúng.

6. Phương pháp theo điểm 1, trong đó việc tạo con trỏ lớp, và chuyển con trỏ lớp của một lớp thuộc lớp giữa và lớp của chương trình ứng dụng tới liên kết động chương trình ứng dụng bao gồm các bước:

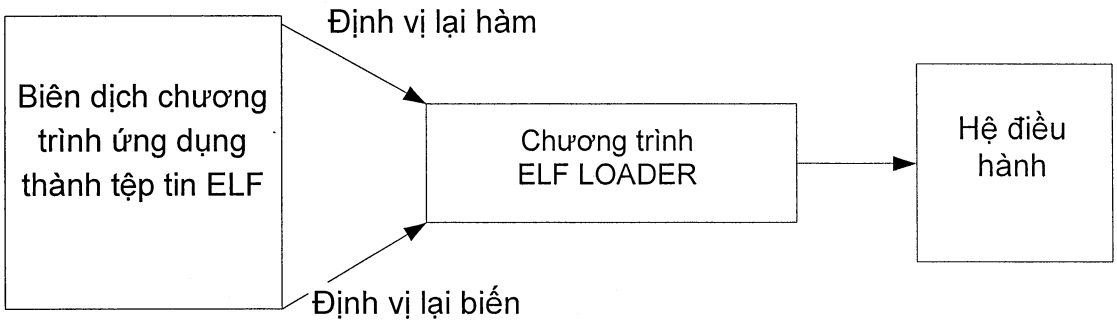
tạo, bởi chương trình ứng dụng, con trỏ lớp của lớp giữa thông qua lớp quản lý khi tham chiếu một hàm của hệ điều hành nhúng, trong đó con trỏ lớp trỏ tới lớp của lớp giữa; chuyển con trỏ lớp tới lớp của chương trình ứng dụng; hoặc

tạo, bởi chương trình ứng dụng, con trỏ lớp của chương trình ứng dụng, khởi tạo các phần tử của bảng hàm ảo tương ứng với lớp của chương trình ứng dụng, và chuyển con trỏ lớp tới lớp giữa để lưu trữ; khi cần thiết tham chiếu một hàm của chương trình ứng dụng, tham chiếu, bởi lớp giữa, con trỏ lớp của chương trình ứng dụng mà được lưu trữ bởi lớp giữa.

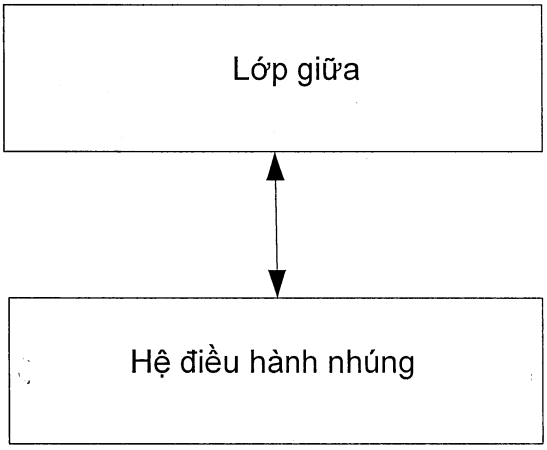
7. Thiết bị nhúng, bao gồm một lớp giữa, trong đó

lớp giữa bao gồm ít nhất một lớp, mỗi lớp được đóng gói từ ít nhất một giao diện hàm của hệ điều hành nhúng và tương ứng với bảng hàm ảo, và các phần tử của bảng hàm ảo tương ứng với ít nhất một giao diện hàm; và

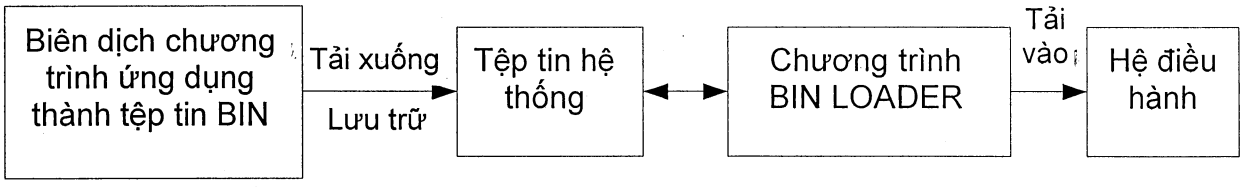
lớp giữa được cấu hình để thực hiện phương pháp theo bất kỳ điểm nào từ 1 đến 6.



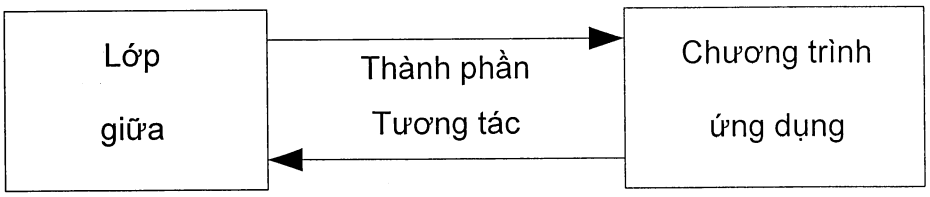
Hình 1



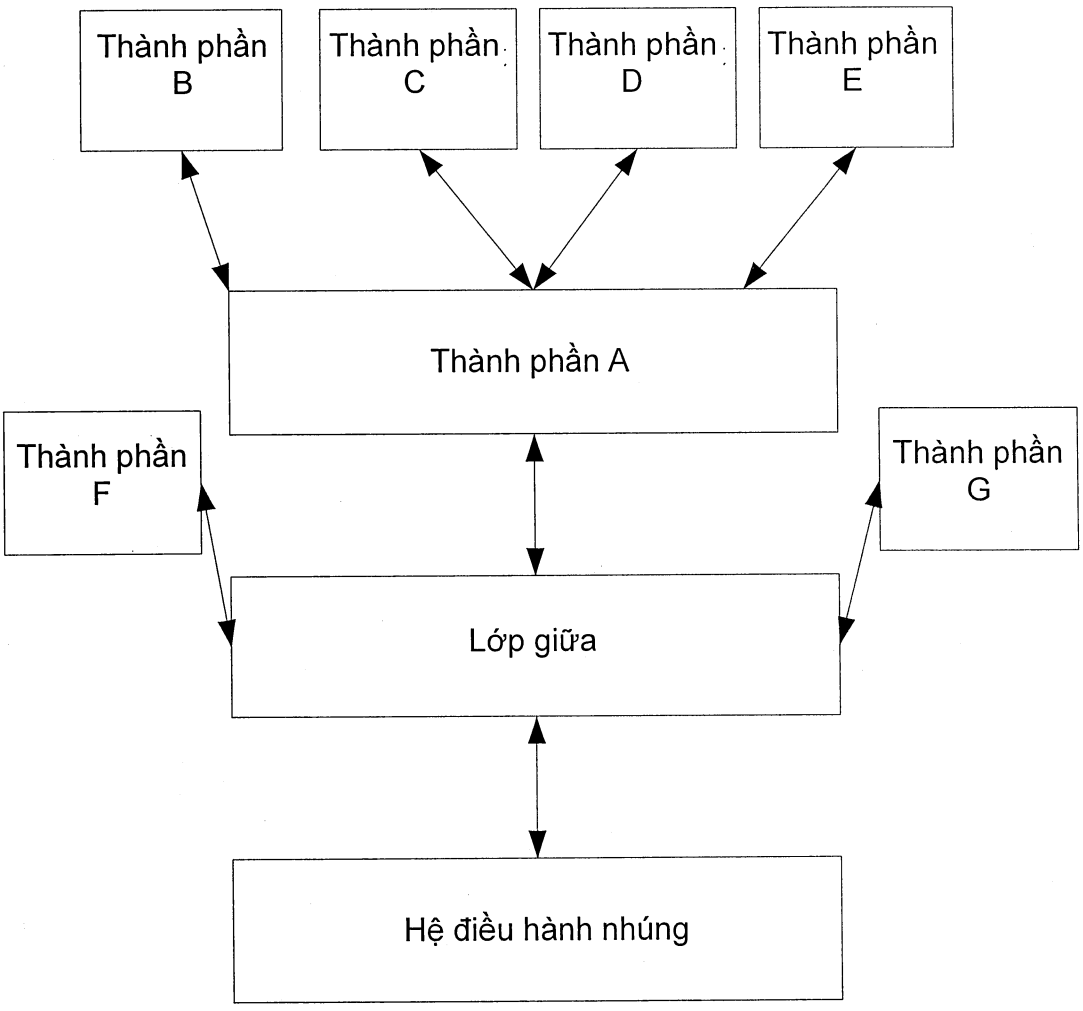
Hình 2



Hình 3


Lớp
giữa
Thành phần
Tương tác
Chương trình
ứng dụng

Hình 4


Thành phần
B
Thành phần
C
Thành phần
D
Thành phần
E
Thành phần A
Thành phần
F
Thành phần
G
Lớp giữa
Hệ điều hành nhúng

Hình 5