Virtual Machine - Máy Ảo

TS. Nguyễn Hứa Phùng

Tháng 2/2021

1 Giới thiệu

Virtual machine (Máy ảo) là một chương trình phần mềm có thể chạy trên một máy vật lý để hoạt động như một hệ thống máy tính ảo với các bộ lệnh riêng cùng với các thành phần khác.

Máy ảo có rất nhiều ứng dụng khác nhau:

- Trình mô phỏng/giả lập hệ thống máy tính vật lý trong thực tế để có thể thực thi các phần mềm trên đó. Ví dụ như VirtualBox hay VMWare.
- Nền tảng mới để thực thi chương trình của nó theo cùng một cách trên các máy vật lý khác nhau. Cụ thể: máy ảo Java được sử dụng để thực thi chương trình Java theo cùng một cách trên các hệ thống máy tính khác nhau
- Các ứng dụng sử dụng dữ liệu phức tạp. Microsoft Word là một ứng dụng sử dụng nhiều loại dữ liệu khác nhau (văn bản, hình ảnh, bảng, định dạng, ...)

Các phần tiếp theo sẽ đặc tả cấu trúc và tập hợp các lệnh trong một máy ảo được sử dụng cho các bài tập lớn trong khóa học này.

2 Cấu trúc của Máy ảo

Máy ảo bao gồm một CPU, một dãy bộ nhớ dành cho các lệnh, một dãy bộ nhớ tĩnh, một bộ nhớ đệm (cache) và một ngăn xếp (stack) cho các địa chỉ trả về. CPU thực thi các lệnh trong dãy bộ nhớ lệnh (nơi lưu trữ các đoạn lệnh thực thi của chương trình); bộ nhớ tĩnh lưu giữ các biến được sử dụng trong chương trình; ngăn xếp giữ các địa chỉ trả về của các lệnh gọi; bộ nhớ đệm giữ các bộ địa chỉ và giá trị nhằm cải thiện tốc độ thời gian đọc bộ nhớ. CPU có 16 thanh ghi (register) bao gồm 1 thanh ghi đặc biệt (IP) và 15 thanh ghi chung (general register) từ R1 đến R15. Thanh ghi IP giữ một địa chỉ trong dãy bộ nhớ lệnh. Các thanh ghi chung có thể giữ một địa chỉ hoặc dữ liệu thuộc kiểu số nguyên (int), số thực (float) hoặc luận lý (boolean).

Để thực thi một chương trình trên máy ảo, chương trình này phải được viết trong 1 tập tin văn bản, bao gồm 1 chuỗi các câu lệnh được mô tả trong Phần 3. Chương trình này đầu tiên phải được nạp vào dãy bộ nhớ lệnh. Mỗi lệnh được đưa vào một ô (slot) của dãy bộ nhớ lệnh (có 65536 và được đánh địa chỉ từ 0). Sau đó, máy ảo để CPU thực hiện lặp đi lặp lại các bước sau:

1. Đọc lệnh tại địa chỉ được lưu trữ trong thanh ghi IP (giá trị của IP luôn được khởi tạo bằng 0).

2. Tăng địa chỉ trong thanh ghi IP lên 1.

3. Thực thi câu lệnh đã đọc ở bước 1. Ý nghĩa của mỗi lệnh và cách thực thi các câu lệnh tương ứng được mô tả chi tiết trong Phần 3,

Bộ nhớ tĩnh cũng có 65536 ô, mỗi ô có thể lưu dữ liệu có kiểu số nguyên, số thực hoặc luận lý. Các ô trong bộ nhớ tĩnh cũng được đánh địa chỉ từ 0 đến 65535. Giá trị trong mỗi ô có thể được thay đổi trong quá trình thực hiện chương trình.

Ngăn xếp địa chỉ trả về chỉ giữ địa chỉ của các lệnh trong dãy bộ nhớ lệnh. Kích thước của ngăn xếp là 1000.

Phần tiếp theo sẽ mô tả chi tiết các lệnh được sử dụng trong máy ảo.

3 Tập hợp các lệnh

Một lệnh được viết trên một dòng và luôn bắt đầu bằng một mã. Ngoài ra, một lệnh có thể không có hoặc có một hoặc hai toán hạng. Toán hạng đầu tiên trong lệnh, nếu có, sẽ cách mã bằng đúng một khoảng trắng (space). Toán hạng thứ hai của mã, nếu có, sẽ cách với toán hạng đầu tiên bằng một dấu phẩy và một khoảng trắng. Ngoài ra, không có ký tự phân cách và theo sau nào khác.

Trong phần mô tả các lệnh sau đây, toán hạng có tên là "dest", chỉ có thể là một thanh ghi chung, trong khi một toán hạng có tên là "src" có thể là một thanh ghi chung hoặc là một hằng (literal). Một hằng nguyên (integer literal) có thể được viết như một hằng số nguyên dạng thập phân trong C++; một hằng số thực (a float literal) có thể được viết bởi một chuỗi ký số theo sau là một dấu chấm và một chuỗi ký số khác. Các chuỗi kí số này không được để trống. Một hằng luận lý (boolean literal) viết bởi chữ true hoặc false. Một hằng địa chỉ (address literal) có thể được viết giống như một hằng số nguyên nhưng kết thúc bằng ký tự A.

Có 6 nhóm lệnh sẽ được mô tả trong các phần tiếp theo. Trong mỗi phần này, đoạn đầu tiên sẽ mô tả kiểu dữ liệu ứng với các toán hạng thuộc các lệnh trong nhóm này. Sau đó, tất cả các lệnh trong nhóm này được liệt kê, trong đó phần đầu tiên (trước dấu hai chấm) là cách viết lệnh và phần thứ hai (sau dấu hai chấm) là ý nghĩa của lệnh tương ứng.

3.1 Các lệnh số học

Tất cả các lệnh trong nhóm này có hai toán hạng; cả hai toán hạng đều là toán hạng nhập và kết quả được lưu vào toán hạng đầu tiên. Kiểu dữ liệu của các toán hạng này có thể là kiểu nguyên hoặc kiểu thực. Kiểu dữ liệu của kết quả chỉ là kiểu nguyên khi cả hai toán hạng nhập đều là kiểu nguyên, nếu không, kết quả sẽ ở kiểu thực.

Các lệnh trong nhóm này bao gồm:

- Add dest, src: dest = dest + src
- Minus dest, src: dest = dest - src
- Mul dest, src: dest = dest * src
- Div dest, src: dest = dest / src

3.2 Các lệnh so sánh

Kiểu dữ liệu của toán hạng của các lệnh này có thể là kiểu nguyên, kiểu thực hoặc kiểu luận lý. Nếu kiểu toán hạng là luận lý thì cả hai toán hạng phải cùng kiểu. Nếu không, kiểu toán hạng có thể là hỗn hợp giữa kiểu nguyên và kiểu thực. Kiểu kết quả luôn là kiểu boolean.

Các lệnh trong nhóm này bao gồm:

- CmpEQ dest, src: if (dest == src) dest = true else dest = false
- CmpNE dest, src: if (dest != src) dest = true else dest = false
- CmpLT dest, src: if (dest < src) dest = true else dest = false
- CmpLE dest, src: if (dest <= src) dest = true else dest = false
- CmpGT dest, src: if (dest > src) dest = true else dest = false
- CmpGE dest, src: if (dest >= src) dest = true else dest = false

3.3 Các lệnh luận lý

Kiểu dữ liệu của toán hạng của các lệnh này chỉ có thể là kiểu luận lý và kiểu kết quả cũng là luận lý.

Các lệnh trong nhóm này bao gồm:

- Not dest: dest = !dest
- And dest, src: dest = dest && src
- Or dest, src: dest = dest || src

3.4 Các lệnh đọc và ghi dữ liệu

Kiểu dữ liệu của **src** và **dest** được trình bày cụ thể trong mỗi lệnh.

Các lệnh trong nhóm này bao gồm:

- Move dest, src: dest = src ; kiểu dữ liệu của **src** và **dest** có thể là bất kỳ kiểu nào
- Load dest, src: dest = *src ; kiểu dữ liệu của **src** là kiểu địa chỉ trong khi kiểu dữ liệu của **dest** có thể là bất kỳ kiểu nào. Địa chỉ được sử dụng trong lệnh này là địa chỉ trong bộ nhớ tĩnh.
- Store dest, src: *dest = src ; kiểu dữ liệu của **dest** là kiểu địa chỉ trong khi kiểu dữ liệu của **src** có thể là bất kỳ kiểu nào. Địa chỉ được sử dụng trong lệnh này là địa chỉ trong bộ nhớ tĩnh.

3.5 Lệnh điều khiển trình tự

Kiểu dữ liệu của **src** là địa chỉ trong khi kiểu dữ liệu của **dest** là boolean. Địa chỉ được sử dụng trong nhóm này là địa chỉ trong dãy bộ nhớ lệnh.

Các lệnh trong nhóm này bao gồm:

- Jump src: IP = src
- JumpIf dest, src: if (dest) IP = src
- Call src: Đưa giá trị của thanh ghi IP vào ngăn xếp và sau đó gán IP = src
- Return: Lấy giá trị ra khỏi ngăn xếp và gán giá trị nó cho thanh ghi IP.
- Halt: Dừng chạy

3.6 Các lệnh nhập xuất dữ liệu

Kiểu dữ liệu của **dest** của các lệnh trong nhóm này có thể là số nguyên, số thực hoặc luận lý, trong khi kiểu dữ liệu của **src** của các lệnh trong nhóm này có thể là bất kỳ kiểu nào (số nguyên, số thực, luận lý hoặc địa chỉ).

Các lệnh trong nhóm này bao gồm:

- Input dest: cin >> dest
- Output src: cout << src

4 Lỗi

Một biến cố (exception) phải được ném ra cùng với địa chỉ của lệnh có lỗi, với các lỗi được mô tả cụ thể như sau:

4.1 Lỗi lúc nạp chương trình

Lỗi lúc nạp chương trình xảy ra khi máy ảo tải chương trình đầu vào từ tập tin văn bản.

- InvalidInstruction(address): xảy ra khi một lệnh được viết không đúng với đặc tả lệnh hoặc mã trong lệnh không có hoặc số lượng toán hạng không khớp với mô tả của lệnh tương ứng. Ví dụ, lệnh có thêm một khoảng trắng trước đoạn mã hoặc sau toán hạng cuối cùng.
- InvalidOperand(address): lỗi này xảy ra khi
  - toán hạng **dest** không phải là một thanh ghi chung, hoặc
  - toán hạng **src** không phải là một thanh ghi chung hoặc một hằng, hoặc
  - cách viết hằng trong câu lệnh không tuân theo mô tả của được nêu trong Phần 3.

4.2 Lỗi thực thi

Lỗi thực thi xảy ra khi CPU thực hiện một lệnh trong dãy bộ nhớ lệnh.

- TypeMismatch(address): các toán hạng nhập của lệnh không đúng với các kiểu dữ liệu được quy định cho các toán hạng của lệnh tương ứng.
- InvalidInput(address): giá trị nhập của lệnh **Input** không tuân thủ mô tả về hằng trong Phần 3.
- InvalidDestination(address): địa chỉ trong toán hạng **src** của lệnh **Jump**, **JumpIf** hoặc **Call** không có trong chương trình.
- DivideByZero(address): giá trị của toán hạng **src** của lệnh **Div** là 0 (kiểu số nguyên) hoặc 0.0 (kiểu số thực).
- StackFull(address): ngăn xếp của máy ảo đã đầy nên nó không thể nhận thêm một phần từ mới được đưa vào bởi lệnh tại địa chỉ đó.

Giả sử rằng không có các lỗi nào khác ngoài các lỗi trên.