ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

NGUYÊN LÝ NGÔN NGỮ LẬP TRÌNH (MỞ RỘNG)

Sinh mã MIPS cho toán tử + của ngôn ngữ BKOOL

GVHD: Nguyễn Hứa Phùng
SV: Ngô Lê Quốc Dũng - 1910101

TP. HỒ CHÍ MINH, THÁNG 3/2022

Mục lục

1 Sơ lược về mã MIPS

Hợp ngữ là ngôn ngữ có khả năng chuyển đổi 1-1 sang ngôn ngữ máy. Hợp ngữ và kiến trúc máy tính được chia làm hai loại: CISC và RISC. Đại diện tiêu biểu cho CISC là kiến trúc x86 được dùng cho các dòng vi xử lý của Intel. Đại diện cho RISC là kiến trúc ARM dùng cho các thiết bị di động, và MIPS dùng cho một số siêu máy tính và các thiết bị khác như router, máy chơi game cầm tay, game console.

Kiến trúc MIPS sử dụng các thanh ghi để lưu giá trị. Vi xử lý sẽ đọc/ghi trên bộ thanh ghi này để cho một tốc độ truy xuất và xử lý cao hơn nhiều so với việc truy xuất vào bộ nhớ chính. Kiến trúc MIPS có tổng cộng 32 thanh ghi, được đánh số từ `$0`, `$1`, ..., `$31` hay theo các tên dễ nhớ hơn như `$zero`, `$sp`,... Một số thanh ghi sẽ được để dành cho mục đích riêng, một số thanh ghi sẽ dùng để lưu trữ các giá trị tạm và một số thanh ghi sẽ dùng để lưu trữ các giá trị biến, cụ thể chi tiết sẽ như bảng dưới đây.

Tên	Thanh ghi	Ý nghĩa
`$zero`	0	Chứa giá trị 0
`$at`	1	Assembler Temporary
`$v0`, `$v1`	2, 3	Luu giá trị trả về của hàm
`$a0` - `$a3`	4 - 7	Lưu tham số truyền vào của hàm
`$t0` - `$t7`	8 - 15	Lưu biến tạm
`$s0` - `$s7`	16 - 23	Lưu biến
`$t8`, `$t9`	24, 25	Lưu biến tạm
`$k0`, `$k1`	26, 27	Để dành cho nhân hệ điều hành
`$gp`	28	Con trỏ tới global
`$sp`	29	Con trỏ tới đỉnh stack
`$fp`	30	Con trỏ tới frame
`$ra`	31	Lưu địa chỉ trả về

MIPS hỗ trợ các lệnh tính toán, thao tác với bộ nhớ, điều khiển luồng thực thi. Kết hợp các lệnh này, ta có thể sinh ra các hàm, và như ở mục tiêu của bài tập lớn này, ta sẽ sinh mã cho ngôn ngữ BKOOL - một ngôn ngữ lập trình hướng đối tượng.

2 Kết quả

2.1 Kiến trúc module của bộ sinh mã MIPS

Học hỏi từ cấu trúc module của bộ sinh mã JVM, bộ sinh mã MIPS cũng sẽ có các thành phần tương tự:

- MIPSMachineCode: Chứa các phương thức để sinh mã MIPS
- MIPSFrame (WIP): Quản lý các label và mô phỏng stack thực thi của hàm
- MIPSEmitter: Chứa các phương thức trung gian để sinh ra các đoạn mã thường dùng, là một bước trừu tượng hóa mã máy cho phía bộ sinh mã. Quản lý các thanh ghi.
- MIPSCodeGen: Chứa các hàm để visit tới các node trong cây cú pháp trừu tượng AST

Để sinh mã cho toán tử + của ngôn ngữ BKOOL, ta cần những phương thức tại MIPSEmitter để sinh mã cho phép cộng số nguyên, phép cộng số thực, và chuyển đổi từ số nguyên sang số thực để có thể chạy được ba testcase của nhiệm vụ trước.

Để sinh được những mã đó, phía MIPSMachineCode phải có những phương thức cần thiết để sinh mã MIPS (như phương thức `emitADD` để sinh mã `add rd, rs, rt`). Sau đó phía MIPSEmitter sẽ đóng gói các phương thức sinh mã trên thành những phương thức trừu tượng hơn, sinh mã cho phép cộng hai số nguyên, cộng hai số thực, chuyển từ số nguyên sang số thực, và in số nguyên, số thực ra console. Sau đó phía MIPSCodeGen sẽ sử dụng những phương thức của MIPSEmitter để sinh những mã phù hợp với cây AST đầu vào.

2.2 Hạn chế của bài làm này

Để sinh mã cho các phương thức, ta cần có MIPSFrame để mô phỏng stack thực thi, quản lý việc sinh các label. Đồng thời ta cũng cần những phương thức để quản lý việc phân bổ và gán dữ liệu vào thanh ghi (register allocation and assignment), nhưng vì thời gian hạn hẹp và quản lý thời gian không tốt nên tôi chỉ có thể tạo ra bộ sinh mã vừa đủ tốt để chạy pass các testcase của nhiệm vụ trước. Điều này sẽ khiến nhiệm vụ 3 trở nên rất khó khăn.

2.3 Xác thực kết quả

Để chạy thử, giải nén folder initial để vào folder src, chạy python3 run.py test CodeGenSuite để sinh mã MIPS bằng bộ mô phỏng Mars4.5 đi kèm (trong lib/-Mars4_5.jar).