Experiment in Compiler Construction

Sinh mã đích(3)

Nguyễn Hữu Đức

Bộ môn Hệ thống thông tin

Khoa Công nghệ Thông tin

Đại học Bách khoa Hà nội

Nội dung

- Sinh mã lấy địa chỉ/giá trị biến (có tính đến các biến phi cục bộ)
- Sinh mã lấy địa chỉ/giá trị tham số hình thức(có tính đến các biến phi cục bộ)
- Sinh mã lấy địa chỉ của giá trị trả về của hàm
- Sinh mã gọi hàm/thủ tục
  - Sinh mã tham số thực tế
- Sinh mã lấy địa chỉ/giá trị của phần tử mảng

Lấy địa chỉ/giá trị biến

- Khi lấy địa chỉ/giá trị một biến cần tính đến phạm vi của biến
  - Biến cục bộ được lấy từ frame hiện tại
  - Biến phi cục bộ được lấy theo các StaticLink với cấp độ lấy theo "độ sâu" của phạm vi hiện tại so với phạm vi của biến

```
computeNestedLevel(Scope* scope)
```

Lấy địa chỉ/giá trị tham số hình thức

- Ngoài số bước chuyển được tính toán giống như với biến, việc lấy địa chỉ/giá trị của tham số hình thức còn phụ thuộc vào tham số đó là tham trị hay tham biến
  - Với tham trị, lấy địa chỉ/giá trị giống như biến
  - Với tham biến, địa chỉ của biến chính là giá trị truyền vào cho hàm/thủ tục

Lấy địa chỉ của giá trị trả về

- Giá trị trả về luôn nằm ở offset 0 trên frame, do đó chỉ cần tính thêm số bước chuyển giống như với biến hay tham số hình thức

Sinh lời gọi hàm/thủ tục

- Lời gọi hàm/thủ tục được thực hiện tại lệnh Call hoặc trong việc sinh mã cho factor
- Trước khi sinh mã CALL cần phải nạp giá trị cho các tham số hình thức bằng cách
  - Tăng giá trị thanh ghi T lên 4 (bỏ qua RV,DL,RA,SL)
  - Sinh mã cho k tham số thực tế
  - Giảm giá trị thanh ghi T đi 4 + k
- "độ sâu" trong lệnh CALL được tính bằng độ sâu tính từ phạm vi của hàm/thủ tục gọi đến phạm vi chứa hàm/thủ tục gọi

Sinh lời gọi hàm/thủ tục

- Sau lời gọi hàm/thủ tục, điều khiển chuyển đến địa chỉ bắt đầu của chương trình con
- Lệnh đầu tiên thông thường là lệnh nhảy, để bỏ qua các hàm/ thủ tục cục bộ
- Lệnh tiếp theo là lệnh bỏ qua frame bằng cách tăng T đúng bằng kích thước frame
- Khi kết thúc thủ tục, toàn bộ frame được giải phóng (lệnh EP) bằng cách đặt con trỏ T lên đầu frame
- Khi kết thúc hàm, frame được giải phóng, chỉ chừa giá trị trả về tại vị trí 0 (lệnh EF)

Sinh địa chỉ của phần tử mảng

- Một biến mảng được định nghĩa

`A : array(.`n_1`.) of … of array(.`n_k`.) of integer/char`

sẽ chiếm $n_1 * \ldots * n_k$ từ trên frame

- A(.i1.)…(.ik.) được định vị tại địa chỉ

$A + i_1 * n_2 *\ldots* n_k + i_2 * n_3 *\ldots* n_k + \ldots + i_{k-1} * n_k + i_k$

- Địa chỉ này được tính tích lũy theo tiến trình duyệt chỉ số

Nhiệm vụ

- Bổ sung thêm vào codegen.c

```
int computeNestedLevel(Scope* scope);
void genVariableAddress(Object* var)
void genVariableValue(Object* var)
void genParameterAddress(Object* param)
void genParameterValue(Object* param)
void genReturnValueAddress(Object* func)
void genReturnValueValue(Object* func)
void genProcedureCall(Object* proc)
void genFunctionCall(Object* func)
```

Nhiệm vụ

- Cập nhật parser.c

```
Type* compileLValue(void);
void compileCallSt(void);
Type* compileFactor(void);
Type* compileIndexes(Type* arrayType);
```