|  |
| --- |
| **BỘ TÀI NGUYÊN VÀ MÔI TRƯỜNG**  **TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP.HCM**  **KHOA: HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM**    **ĐỒ ÁN MÔN HỌC**  **Trình Biên Dịch**  **Thiết kế một trình biên dịch cho phép dịch một đoạn chương trình viết bằng ngôn ngữ C sang mã máy**  Giảng viên hướng dẫn: **ThS. Đặng Đức Trung**  Sinh viên thực hiện: **Nhóm 1**  **Trần Thị Duyên Dương – 0450080018**  **Lê Văn Hiếu – 0450080041**  Lớp: ĐHCNTT01  Khóa: 04  **TP. Hồ Chí Minh, tháng 04 năm 2018** |

**LỜI MỞ ĐẦU**

Trình biên dịch là một môn học của ngành khoa học máy tính. Cùng với sự phát triển của các chuyên ngành: Lý thuyết ngôn ngữ hình thức & Ôtômat và các chuyên ngành khác, lý thuyết thiết kế trình biên dịch ngày càng hoàn thiện hơn.

Ngày nay lý thuyết trình biên dịch đã được ứng dụng không những trong việc thiết kế trình biên dịch cho ngôn ngữ lập trình mà còn trong lĩnh vực trí tuệ nhân tạo – hiểu ngôn ngữ tự nhiên, dịch máy,…

Đồ án môn học Trình Biên Dịch là dịp tốt để cho việc học tập của sinh viên chuyên ngành tin học nắm vững lý thuyết và cách thiết kế trình biên dịch được tốt.

Vì thế chúng em thực hiện đồ án với đề tài số 1:

Thiết kế một trình biên dịch cho phép dịch một đoạn chương trình viết bằng ngôn ngữ C sang mã máy”.

Đoạn chương trình gồm các phát biểu:

* Vòng lặp for
* Phép gán.
* Các toán tử +, -, \*, /, so sánh.

Các toán hạng gồm định danh, hằng số thực, hằng số nguyên, true, false, biểu thức.

**LỜI CẢM ƠN**

Chúng em xin cảm ơn sự hướng dẩn tận tình của thầy Đặng Đức Trung và sự đóng góp ý kiến của các bạn, nhằm giúp đỡ cho đồ án hoàn thành đúng yêu cầu. Mặc dù đã cố gắng tham khảo nhiều tài liệu nhưng quá trình thực hiện khó tránh khỏi những thiếu sót và hạn chế. Rất mong nhận được những ý kiến nhận xét, đóng góp của thầy cô và các bạn.

Nhóm sinh viên

Trần Thị Duyên Dương

Lê Văn Hiếu

**ĐÁNH GIÁ**

|  |
| --- |
| Điểm |
|  |

**Mục Lục**

[Chương 1: Phân tích 1](#_Toc513421565)

[1.1. Phân tích từ vựng: 1](#_Toc513421566)

[1.1.1. Xây dựng bảng phân tích từ vựng: 1](#_Toc513421567)

[1.1.2. Thiết kế các sơ đồ dịch cho token: 2](#_Toc513421568)

[1.1.3. Các thuật toán phân tích áp dụng trong đồ án: 5](#_Toc513421569)

[1.2. Phân tích cú pháp: 6](#_Toc513421570)

[1.2.1. Xây dựng văn phạm: 6](#_Toc513421571)

[1.2.2. Các thành phân trong văn phạm: 7](#_Toc513421572)

[1.2.3. Xây dựng bản phân tích: 8](#_Toc513421573)

[1.2.4. Thuật toán xây dựng cú pháp 9](#_Toc513421574)

[1.3. Phân tích ngữ nghĩa: 9](#_Toc513421575)

[1.3.1. Bảng quản lý ký hiệu: 9](#_Toc513421576)

[1.3.2. Các kỹ thuật kiểm tra kiểu: 9](#_Toc513421577)

[1.3.3. Quản lý phạm vi biến: 9](#_Toc513421578)

[1.3.4. Kỹ thuật quản lý phạm vi: 9](#_Toc513421579)

[Chương 2: Sinh mã 10](#_Toc513421580)

[2.1. Sinh mã trung gian: 10](#_Toc513421581)

[2.2. Sinh mã đích: 11](#_Toc513421582)

[Chương 3: Thực Nghiệm 12](#_Toc513421583)

[Chương 4: Kết luận 12](#_Toc513421584)

[Danh mục tài liệu tham khảo 13](#_Toc513421585)

**Danh mục hình**

|  |  |
| --- | --- |
| Hình 2.1: Cấu trúc sinh mã lệnh for | 10 |
| Hình 2.2: Ví dụ cây IR | 10 |

**Danh mục bảng**

|  |  |
| --- | --- |
| Bảng phân tích từ vựng | 1 |
| Bảng sơ đồ dịch | 2 |

# **Chương 1: Phân tích**

## **1.1. Phân tích từ vựng:**

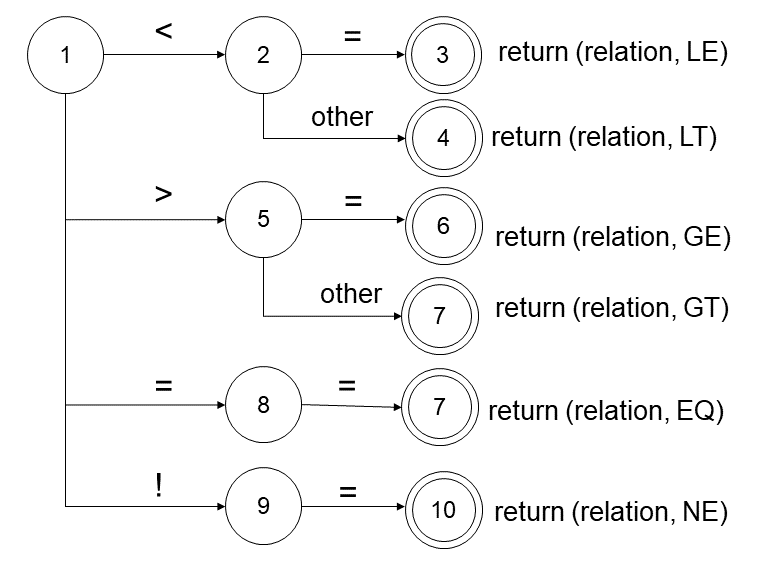
### **1.1.1. Xây dựng bảng phân tích từ vựng:**

|  |  |  |
| --- | --- | --- |
| **Token** | **Trị từ vựng minh họa** | **Mô tả của mẫu từ vựng** |
| for | for | for |
| ( | ( | ( |
| id | aA, a1, a2,… | (letter | ‘\_’)(letter|digit| ‘\_’)\* |
| = | = | ‘=’ |
| num | 1,2,3,123,… | Digit(digit)\* |
| Numreal | 1.2, 2E-3,0.5e+4, … | Digit\*.digit\* |
| ; | ; | ‘;’ |
| relation | <, >, <=, >=, ==, != | < hoặc > hoặc <= hoặc >= hoặc == hoặc != |
| operator | +, -, \*, /, % | + hoặc - hoặc \* hoặc / hoặc % |
| ) | ) | ) |
| type | Int, Float, Bool | (‘i’|‘I’) (‘n’|‘N’) (‘t’|‘T’) (‘f’|‘F’) (‘l’|‘L’) (‘o’|‘O’) (‘a’|‘A’) (‘t’|‘T’) (‘b’|‘B’) (‘o’|‘O’) (‘o’|‘O’) (‘l’|‘L’) |
| } | } | } |
| { | { | { |
| true | true | true |
| false | false | false |
| break | break | break |
| continue | continue | continue |

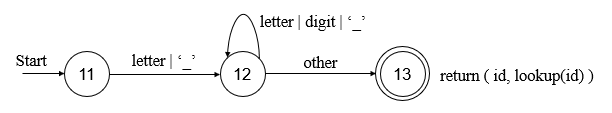
### **1.1.2. Thiết kế các sơ đồ dịch cho token:**

|  |  |  |
| --- | --- | --- |
| **Biểu thức chính quy** | **Token** | **Trị thuộc tính** |
| ws | - | - |
| for | FOR | - |
| true | TRUE |  |
| false | FALSE | - |
| int | INT | - |
| float | FLOAT | - |
| id | ID | Con trỏ trong bản ký hiệu |
| num | NUMBER | Giá trị số |
| < | RELATION | LT(Less Than) |
| > | RELATION | GT(Great Than) |
| >= | RELATION | GE(Greater or Equal) |
| <= | RELATION | LE(Less or Equal) |
| == | RELATION | EQ(Equal) |
| != | RELATION | NE(Not Equal) |
| + | OPERATOR | ADD(Addition) |
| - | OPERATOR | SUB(Subtraction) |
| \* | OPERATOR | MUL(Multiplication) |
| / | OPERATOR | DIV(Devision) |
| % | OPERATOR | MOD(Module) |
| ( | OPEN | - |
| ) | CLOSE | - |
| { | OPENFUNC | - |
| } | CLOSEFUNC | - |
| = | ASSIGNMENT | - |
| ; | END\_STAMENT | - |

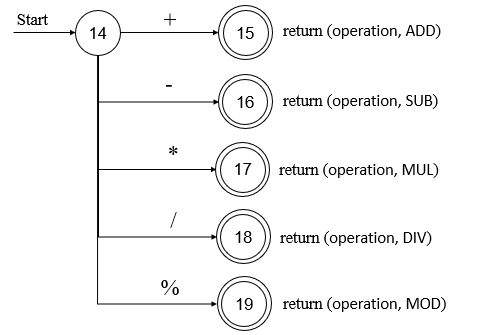
* Sơ đồ dịch token relation:

**

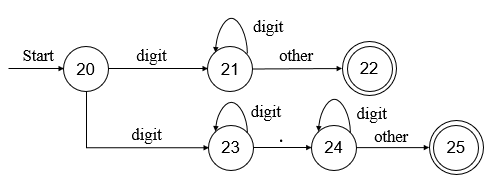
* Sơ đồ dịch id và từ khóa:



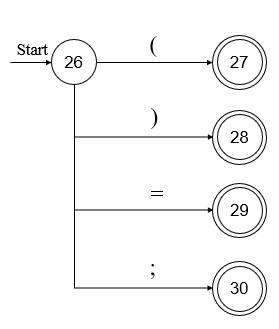
* Sơ đồ dịch operation:



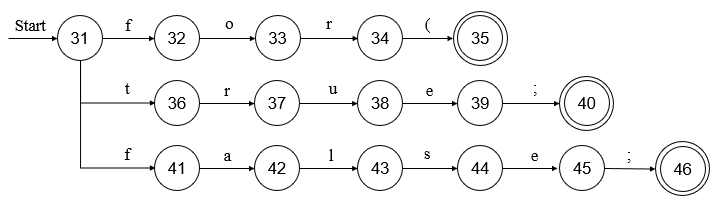
* Sơ đồ dịch num và Numreal:



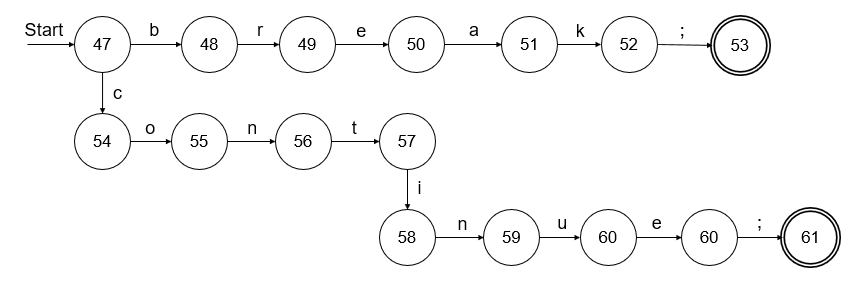
* Sơ đồ dịch (, ), ;, = :



* Sơ đồ dịch for, true, false :



* Sơ đồ dịch break và continue:



**1.1.3. Các thuật toán phân tích áp dụng trong đồ án:**

|  |
| --- |
| **Function**   lexan: integer;  var   lexbuf: array[0..100] of char;  c:  char  **begin**  **loop   begin**              đọc một ký tự vào c;  **if**  c là một ký tự trống blank  hoặc ký tự tab  **then**                  không thực hiện điều gì ;  **else  if**  c là ký tự  newline   **then**                   lineno = lineno + 1  **else   if**  c là một ký tự số   **then**  **begin**                         đặt tokenval là giá trị của ký số này và các ký số theo sau;  **return**NUM;  **end**  **else  if** c là một chữ cái   **then**  **begin**                         đặt c và các ký tự, ký số theo sau vào lexbuf;                         p := **lookup** (lexbuf);  **if**   p = 0   **then**  p := insert (lexbuf, id);                         tokenval := p;                         return trường token của ô có chỉ mục p;  **end**  **else**  **begin**          /\* token là một ký tự đơn \*/                          đặt tokenval là NONE;          /\* không có thuộc tính \*/                          return số nguyên mã hóa của ký tự  c;  **end;**  **end;**  **end;** |

**1.2. Phân tích cú pháp:**

**1.2.1. Xây dựng văn phạm:**

* Với các token đề bài ta thấy code C bao gồm các cú pháp hợp lệ gồm: khai báo biến và gán biến, vòng lặp for, biểu thức.

s → Program

Program → kbb\_gb | for\_stament

* Khai báo biến và gán biến: Ta có các trường hợp hợp lệ. VD: [int a = 1;] hoặc [int a = x;] hoặc [int a;] hoặc [int a = a + 1;] hoặc [a = a + 1;], …

kbb\_gb → kdl ID | kdl ID pg

kdl → INT | FLOAT

pg → ASSIGNMENT bt END\_STAMENT

bt → var op bt | ɛ

var → ID | NUMBER | OPEN bt CLOSE

op → OPERATOR | ɛ

* Vòng lặp for có cấu trúc [for(biểu thức gán ; biểu thức so sánh ; biểu thức gán) {stament}] nên ta có các luật sinh

for\_stament → FOR bt\_f stament\_f

bt\_f → OPEN kbb\_gb\_f END\_STAMENT btss END\_STAMENT btg CLOSE

kbb\_gb\_f → kbb\_gb | ɛ

btss → var rel dk | TRUE

rel → RELATION

dk → var | TRUE | FALSE

btg → ID pg

stament\_f → OPENFUNC stament CLOSEFUNC

stament → kbb\_gb stament | for\_stament stament | ɛ

### **1.2.2. Các thành phân trong văn phạm:**

* **Ký hiệu kết thúc:**

ID, INT, FLOAT, ASSIGNMENT, END\_STAMENT, NUMBER, OPEN, CLOSE, OPERATOR, FOR, TRUE, FALSE, RELATION, OPENFUNC, CLOSEFUNC

* **Ký hiệu chưa kết thúc:**

s, Program, kbb\_gb, for\_stament, kdl, pg, bt, var, op, bt\_f, stament\_f, btss, btg, rel, dk, stament, kbb\_gb\_f

* **Tập luật sinh:**

s → Program

Program → kbb\_gb s | for\_stament s | ɛ

kbb\_gb → kdl ID pg | ID pg

kdl → INT | FLOAT

pg → ASSIGNMENT bt END\_STAMENT | END\_STAMENT

bt → var op bt | ɛ

var → ID | NUMBER | OPEN bt CLOSE

op → OPERATOR | ɛ

for\_stament → FOR bt\_f stament\_f

bt\_f → OPEN kbb\_gb\_f END\_STAMENT btss END\_STAMENT btg CLOSE

kbb\_gb\_f → kbb\_gb | ɛ

btss → var rel dk | TRUE | ɛ

rel → RELATION

dk → var | TRUE | FALSE

btg → ID pg | ɛ

stament\_f → OPENFUNC stament CLOSEFUNC

stament → kbb\_gb stament | for\_stament stament | ɛ

* **Ký hiệu bắt đầu:** s

### **1.2.3. Xây dựng bản phân tích:**

* **Tính First:**

First(s) = { ɛ , ID, INT , FLOAT , FOR }

First(kbb\_gb) = {INT, FLOAT, ID}

First(kdl) = {INT, FLOAT}

First(pg) = {ASSIGNMENT, END\_STAMENT}

First(bt) = {ID, NUMBER, OPEN, ɛ}

First(var) = {ID, NUMBER, OPEN}

First(op) = {OPERATOR, ɛ}

First(for\_stament) = {FOR}

First(bt\_f) = { OPEN }

First(kbb\_gb\_f) = First(kbb\_gb)

First(btss) = {ID}

First(btg) = First(btss)

First(rel) = {RELATION}

First(dk) = {ID, NUMBER, OPEN, TRUE, FALSE}

First(stament) = {OPENFUNC}

* **Tính Follow:**

Follow(s) = {$, CLOSEFUNC}

Follow(kbb\_gb) = {INT, FLOAT, ID, FOR}

Follow(for\_stament) = {INT, FLOAT, ID, FOR}

Follow(kdl) = {ID}

Follow(pg) = {$, CLOSEFUNC}

Follow(bt) = {END\_STAMENT, CLOSE}

Follow(var) = {OPERATOR, END\_STAMENT, CLOSE, RELATION, OPENFUNC }

Follow(op) = {ID, NUMBER, OPEN}

Follow(btss) = {OPENFUNC}

Follow(stament) = {INT, FLOAT, ID, FOR}

Follow(rel) = {ID, NUMBER, OPEN, TRUE, FALSE}

Follow(dk) = {CLOSE}

### **1.2.4. Thuật toán xây dựng cú pháp**

Sử dụng phương pháp phân tích từ trên xuống. Ta sử dụng các hàm đệ quy gọi nhau theo cấu trúc cây và xây dựng các nút theo lời gọi đệ quy.

## **1.3. Phân tích ngữ nghĩa:**

### **1.3.1. Bảng quản lý ký hiệu:**

Sử dụng một danh sách lưu tên biến, số cùng với kiểu dữ liệu của nó. VD: (id1, int), (5, int), (2.2, float),… Ta sẽ được một danh sách các ký biến, số được sử dụng trong chương trình để tiến hành kiểm tra lỗi ngữ nghĩa.

**1.3.2. Các kỹ thuật kiểm tra kiểu:**

Ở đề tài này ta sử dụng thuật toán kiểm tra kiểu như sau:

Ta sẽ duyệt cây từ trên xuống, sử dụng đệ quy và trả về true nếu các lá của nút có cùng kiểu dữ liệu, và tiếp tục với các nút trên, nếu có một nút vi phạm điều kiện này có nghĩa là nó sai ngữ nghĩa. Như vậy mỗi khi đệ quy ta gọi một hàm trả về kiểu dữ liệu đã lưu trong list trên, hàm sẽ trả về kiểu dữ liệu nếu tìm thấy giá trị của nút trong danh sách.

**1.3.3. Quản lý phạm vi biến:**

Nếu các biến được khai báo trong vòng lặp for thì nó chỉ có phạm vi trong vòng for, bên ngoài vòng for không thể sử dụng biến này, những biến được khai báo bên ngoài vẫn có thể được dùng trong vòng for.

**1.3.4. Kỹ thuật quản lý phạm vi:**

Có rất nhiều kỹ thuật quản lý phạm vi, ở đây ta sẽ kiểm tra danh sách token để xác định dùng một danh sách lưu các biến được khởi tạo trong vòng for, là các token ID sau token INT hoặc FLOAT, sau đó duyệt danh sách token bỏ qua vòng for, nếu có một token ID nào có giá trị giống với giá trị trong mảng thì xuất ra lỗi ngữ nghĩa.

**Chương 2: Sinh mã**

**2.1. Sinh mã trung gian:**

Ta đã có cây cú pháp ở bước trên, từ cây cú pháp ta sẽ chuyển thành các cây IR. Từ đó ta cố thể sinh mã theo thứ tự chương trình thực hiện. Trong đề tài này ta sẽ có các nút:

CONST(i): Hằng số I.

TEMP(n): Thanh ghi n.

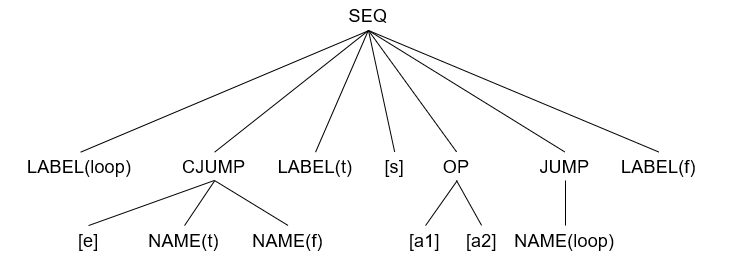
MOVE(a,b): Gán a = b.

JUMP(e): Nhảy đến địa chỉ e.

CJUMP(e, i1, i2): Nhày đến i1 hoặc i2 tùy vào e true hay false.

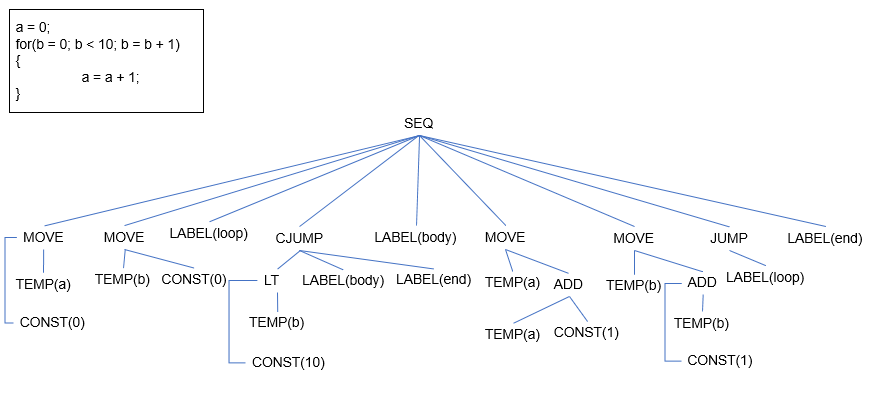
LABEL(n): Tạo nhãn n, để các lệnh nhảy nhảy đến.

OP(e1, e2): Tính giá trị của e1 và e2 theo các phép toán.



*Hình 2.1: Cấu trúc sinh mã lệnh for*

VD



*Hình 2.2: Ví dụ sinh cây IR*

Từ cây IR ta sinh được mã trung gian:

SEQ(

MOVE(TEMP(a), CONST(0)),

MOVE(TEMP(b), CONST(0)),

LABEL(loop),

CJUMP(

LT(TEMP(b), CONST(10)),

LABEL(body),

LABEL(end)

),

LABEL(body),

MOVE(TEMP(a), ADD(TEMP(a), CONST(1))),

MOVE(TEMP(b), ADD(TEMP(b), CONST(1))),

JUMP(LABEL(loop)),

LABEL(end)

)

**2.2. Sinh mã đích:**

**Chương 3: Thực Nghiệm**

**Chương 4: Kết luận**

Qua quá trình nghiên cứu và thực hiện đồ án, không chỉ giúp ta hiểu hơn về cơ chế hoạt động của một trình biên dịch, mà còn có thể xây dựng một trình biên dịch đơn giản dựa trên những kiến thức đó, tạo ra một sản phẩm có thể sử dụng và phát triển. Ngoài ra việc hiểu rõ cơ chế của trình biên dịch giúp ta optimize code tốt hơn, một kỹ năng cần thiết cho một lập trình viên.

**Danh mục tài liệu tham khảo**

[1]Slide bài giảng môn Trình Biên Dịch – Đặng Đức Trung.

[2] Compilers - Principles, Techniques, and Tools (2e) – Aho.

[3] Internet.