CHƯƠNG I
GIỚI THIỆU VỀ SỰ BIÊN DỊCH

Nội dung chính:
Để máy tính có thể hiểu và thực thi một chương trình được viết bằng ngôn ngữ cấp cao, ta cần phải có một *trình biên dịch* thực hiện việc chuyển đổi chương trình đó sang chương trình ở dạng ngôn ngữ đích. Chương này trình bày một cách tổng quan về *cấu trúc của một trình biên dịch* và mối liên hệ giữa nó với các thành phần khác - "họ hàng" của nó - như bộ tiền xử lý, bộ tải và soạn thảo liên kết,v.v. Cấu trúc của trình biên dịch được mô tả trong chương là một cấu trúc mức quan niệm bao gồm các giai đoạn: Phân tích từ vựng, Phân tích cú pháp, Phân tích ngữ nghĩa, Sinh mã trung gian, Tối ưu mã và Sinh mã đích.

Mục tiêu cần đạt:
Sau khi học xong chương này, sinh viên phải nắm được một cách tổng quan về nhiệm vụ của các thành phần của một trình biên dịch, mối liên hệ giữa các thành phần đó và môi trường nơi trình biên dịch thực hiện công việc của nó.

Tài liệu tham khảo:

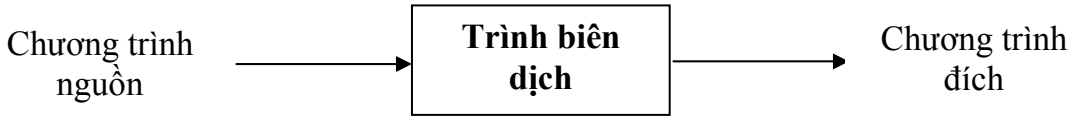
[1] **Trình Biên Dịch** - Phan Thị Tươi (Trường Đại học kỹ thuật Tp.HCM) - NXB Giáo dục, 1998.

[2] **Compilers : Principles, Technique and Tools** - Alfred V.Aho, Jeffrey D.Ullman - Addison - Wesley Publishing Company, 1986.

[3] **Compiler Design** – Reinhard Wilhelm, Dieter Maurer - Addison - Wesley Publishing Company, 1996.

I. TRÌNH BIÊN DỊCH

Nói một cách đơn giản, trình biên dịch là một chương trình làm nhiệm vụ đọc một chương trình được viết bằng một ngôn ngữ - *ngôn ngữ nguồn* (source language) - rồi dịch nó thành một chương trình tương đương ở một ngôn ngữ khác - *ngôn ngữ đích* (target languague). Một phần quan trọng trong quá trình dịch là ghi nhận lại các lỗi có trong chương trình nguồn để thông báo lại cho người viết chương trình.

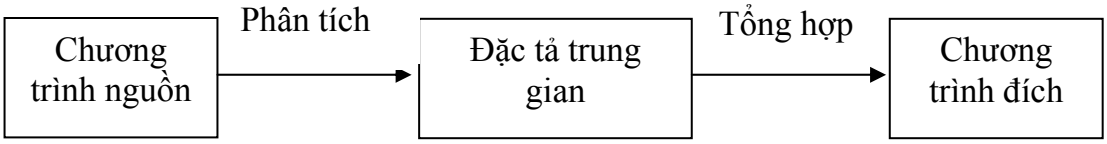


Hình 1.1 - *Một trình biên dịch*

1. Mô hình phân tích - tổng hợp của một trình biên dịch

Chương trình dịch thường bao gồm hai quá trình : phân tích và tổng hợp

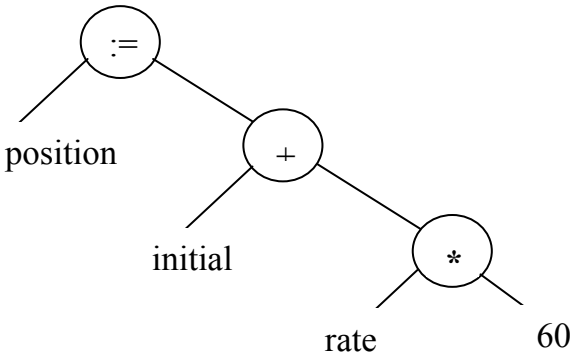
- Phân tích → đặc tả trung gian
- Tổng hợp → chương trình đích



Hình 1.2 - *Mô hình phân tích - tổng hợp*

Trong quá trình phân tích chương trình nguồn sẽ được phân rã thành một cấu trúc phân cấp, thường là dạng cây - *cây cú pháp* (syntax tree) mà trong đó có mỗi nút là một toán tử và các nhánh con là các toán hạng.

Ví dụ 1.1: Cây cú pháp cho câu lệnh gán position := initial + rate * 60



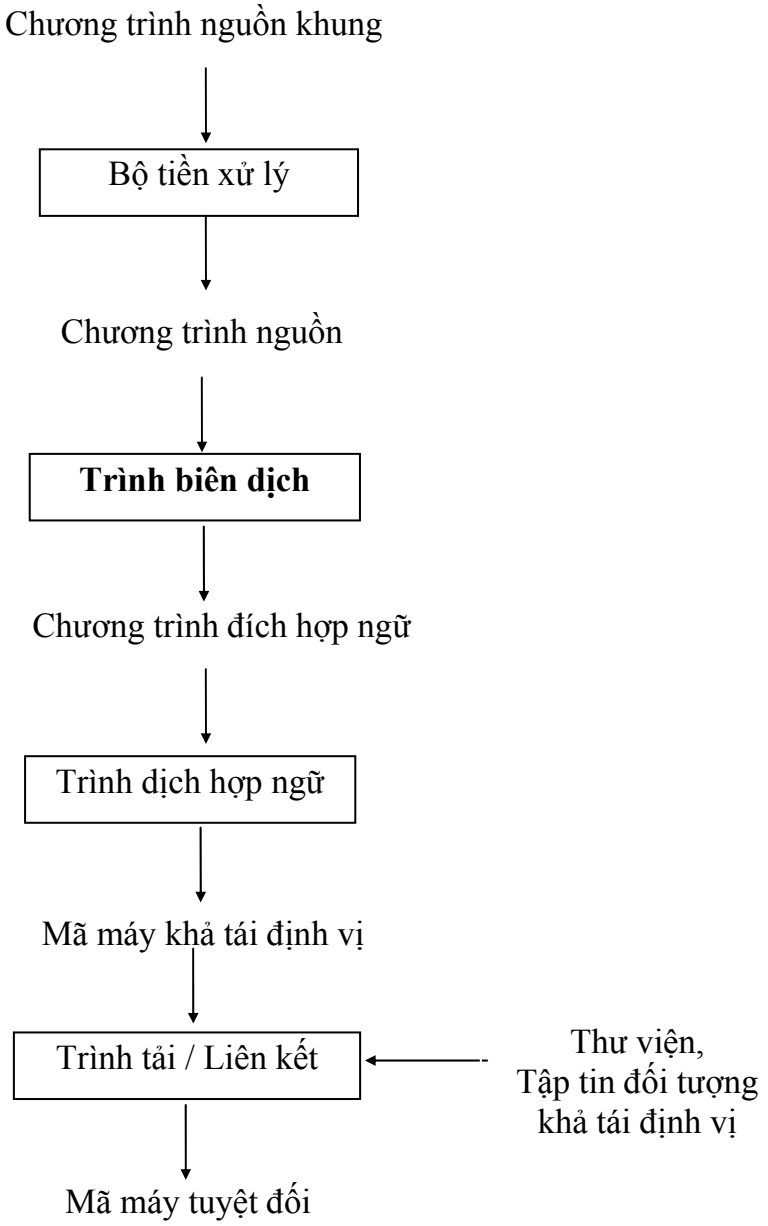
2. Môi trường của trình biên dịch

Ngoài trình biên dịch, chúng ta có thể cần dùng nhiều chương trình khác nữa để tạo ra một chương trình đích có thể thực thi được (executable). Các chương trình đó gồm: Bộ tiền xử lý, Trình dịch hợp ngữ, Bộ tải và soạn thảo liên kết.

Một chương trình nguồn có thể được phân thành các module và được lưu trong các tập tin riêng rẻ. Công việc tập hợp lại các tập tin này thường được giao cho một chương trình riêng biệt gọi là *bộ tiền xử lý* (preprocessor). Bộ tiền xử lý có thể "bung" các ký hiệu tắt được gọi là các macro thành các câu lệnh của ngôn ngữ nguồn.

Ngoài ra, chương trình đích được tạo ra bởi trình biên dịch có thể cần phải được xử lý thêm trước khi chúng có thể chạy được. Thông thường, trình biên dịch chỉ tạo ra mã lệnh hợp ngữ (assembly code) để *trình dịch hợp ngữ* (assembler) dịch thành dạng mã máy rồi được liên kết với một số thủ tục trong thư viện hệ thống thành các mã thực thi được trên máy.

Hình sau trình bày một quá trình biên dịch điển hình :



Hình 1.3 - *Một trình xử lý ngôn ngữ điển hình*

II. SỰ PHÂN TÍCH CHƯƠNG TRÌNH NGUỒN

Phần này giới thiệu về các quá trình phân tích và cách dùng nó thông qua một số ngôn ngữ định dạng văn bản.

1. Phân tích từ vựng (Lexical Analysis)

Trong một trình biên dịch, giai đoạn phân tích từ vựng sẽ đọc chương trình nguồn từ trái sang phải (quét nguyên liệu - scanning) để tách ra thành các thẻ từ (token).

Ví dụ 1.2: Quá trình phân tích từ vựng cho câu lệnh gán position := initial + rate * 60 sẽ tách thành các token như sau:

1. Danh biểu position
2. Ký hiệu phép gán :=
3. Danh biểu initial

4. Ký hiệu phép cộng (+)

5. Danh biểu rate

6. Ký hiệu phép nhân (*)

7. Số 60

Trong quá trình phân tích từ vựng các khoảng trắng (blank) sẽ bị bỏ qua.

2. Phân tích cú pháp (Syntax Analysis)

Giai đoạn phân tích cú pháp thực hiện công việc nhóm các thẻ từ của chương trình nguồn thành các *ngữ đoạn văn phạm* (grammatical phrase), mà sau đó sẽ được trình biên dịch tổng hợp ra thành phẩm. Thông thường, các ngữ đoạn văn phạm này được biểu diễn bằng dạng *cây phân tích cú pháp* (parse tree) với :

- Ngôn ngữ được đặc tả bởi các luật sinh.

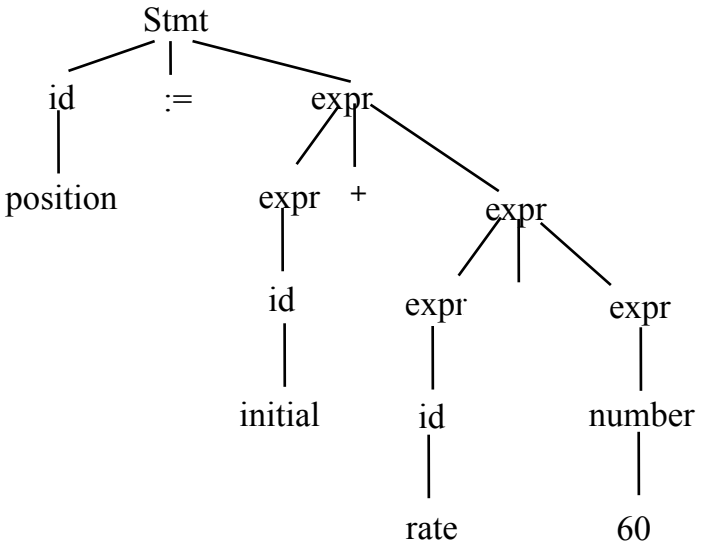
- Phân tích cú pháp dựa vào luật sinh để xây dựng cây phân tích cú pháp.

Ví dụ 1.3: Giả sử ngôn ngữ đặc tả bởi các luật sinh sau :

Stmt $\rightarrow$ id := expr

expr $\rightarrow$ expr + expr | expr * expr | id | number

Với câu nhập: position := initial + rate * 60, cây phân tích cú pháp được xây dựng như sau :



Hình 1.4 - *Một cây phân tích cú pháp*

Cấu trúc phân cấp của một chương trình thường được diễn tả bởi quy luật đệ qui.

Ví dụ 1.4:

1) Danh biểu (identifier) là một biểu thức (expr).

2) Số (number) là một biểu thức.

3) Nếu expr1 và expr2 là các biểu thức thì:

expr1 + expr2

expr1 * expr2

(expr)

cũng là những biểu thức.

Câu lệnh (statement) cũng có thể định nghĩa đệ qui :

1) Nếu id1 là một danh biểu và expr2 là một biểu thức thì id1 := expr2 là một lệnh (stmt).

2) Nếu expr1 là một biểu thức và stmt2 là một lệnh thì

while (expr1) do stmt2

if (expr1) then stmt2

đều là các lệnh.

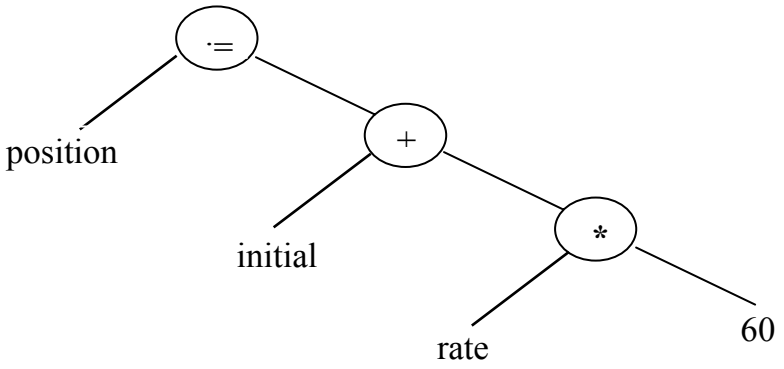
Người ta dùng các qui tắc đệ qui như trên để đặc tả luật sinh (production) cho ngôn ngữ. Sự phân chia giữa quá trình phân tích từ vựng và phân tích cú pháp cũng tuỳ theo công việc thực hiện.

3. Phân tích ngữ nghĩa (Semantic Analysis)

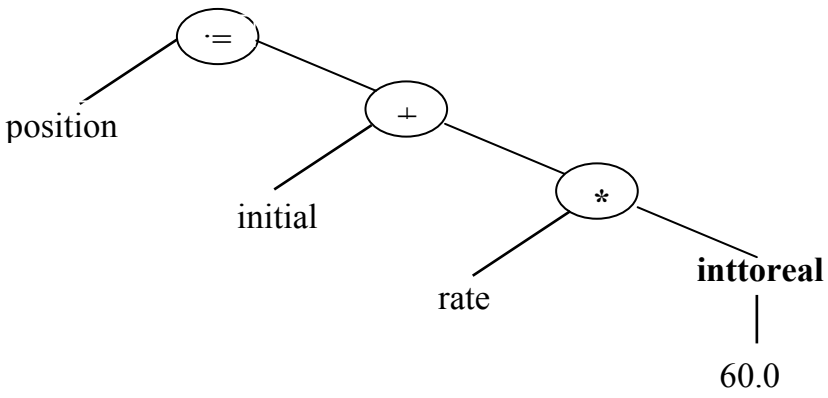
Giai đoạn phân tích ngữ nghĩa sẽ thực hiện việc kiểm tra xem chương trình nguồn có chứa lỗi về ngữ nghĩa hay không và tập hợp thông tin về kiểu cho giai đoạn sinh mã về sau. Một phần quan trọng trong giai đoạn phân tích ngữ nghĩa là *kiểm tra kiểu* (type checking) và ép chuyển đổi kiểu.

Ví dụ 1.5: Trong biểu thức position := initial + rate * 60

Các danh biểu (tên biến) được khai báo là real, 60 là số integer vì vậy trình biên dịch đổi số nguyên 60 thành số thực 60.0



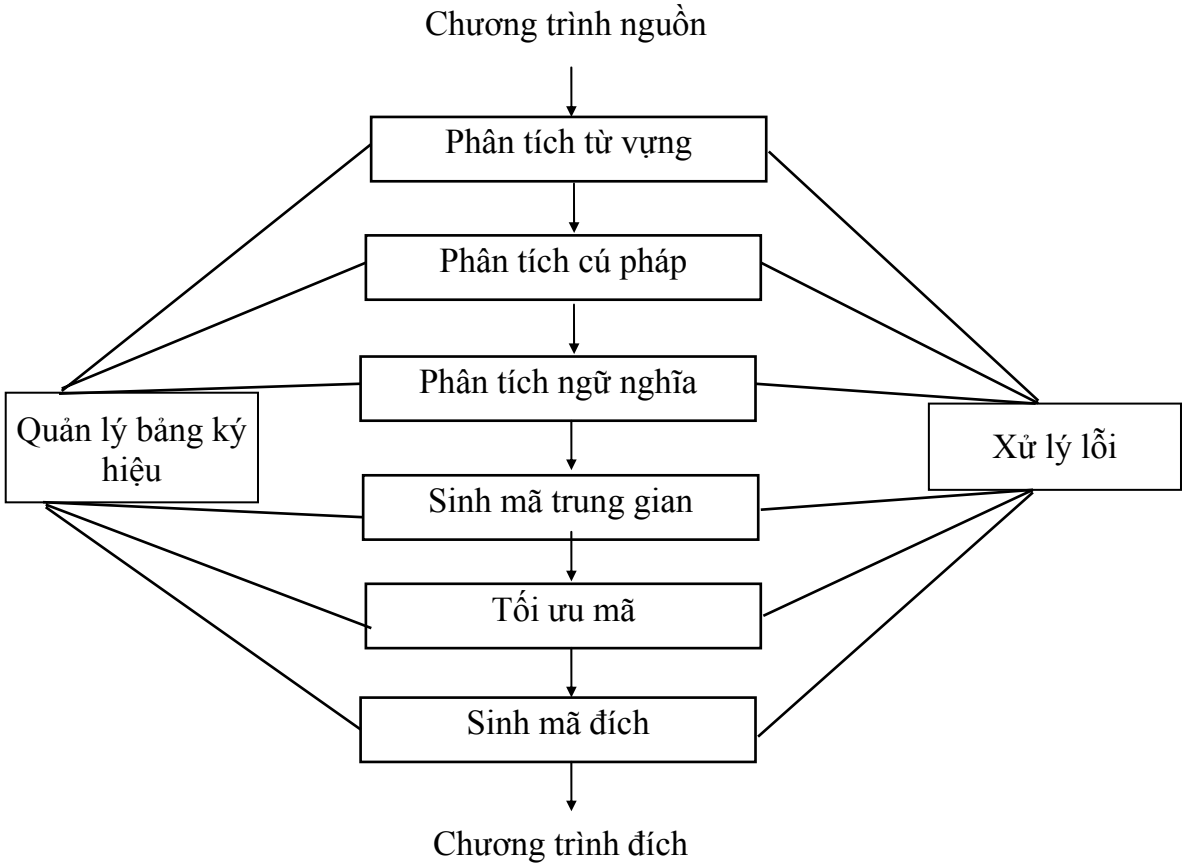
thành



Hình 1.5 - *Chuyển đổi kiểu trên cây phân tích cú pháp*

III. CÁC GIAI ĐOẠN BIÊN DỊCH

Để dễ hình dung, một trình biên dịch được chia thành các giai đoạn, mỗi giai đoạn chuyển chương trình nguồn từ một dạng biểu diễn này sang một dạng biểu diễn khác. Một cách phân rã điển hình trình biên dịch được trình bày trong hình sau.



Hình 1.6 - *Các giai đoạn của một trình biên dịch*

Việc quản lý bảng ký hiệu và xử lý lỗi được thực hiện xuyên suốt qua tất cả các giai đoạn.

1. Quản lý bảng ký hiệu

Một nhiệm vụ quan trọng của trình biên dịch là ghi lại các định danh được sử dụng trong chương trình nguồn và thu thập các thông tin về các thuộc tính khác nhau của mỗi định danh. Những thuộc tính này có thể cung cấp thông tin về vị trí lưu trữ được cấp phát cho một định danh, kiểu và tầm vực của định danh, và nếu định danh là tên của một thủ tục thì thuộc tính là các thông tin về số lượng và kiểu của các đối số, phương pháp truyền đối số và kiểu trả về của thủ tục nếu có.

Bảng ký hiệu (symbol table) là một cấu trúc dữ liệu mà mỗi phần tử là một mẩu tin dùng để lưu trữ một định danh, bao gồm các trường lưu giữ ký hiệu và các thuộc tính của nó. Cấu trúc này cho phép tìm kiếm, truy xuất danh biểu một cách nhanh chóng.

Trong quá trình phân tích từ vựng, danh biểu được tìm thấy và nó được đưa vào bảng ký hiệu nhưng nói chung các thuộc tính của nó có thể chưa xác định được trong giai đoạn này.

Ví dụ 1.6: Chẳng hạn, một khai báo trong Pascal có dạng

var position, initial, rate : real

thì thuộc tính kiểu real chưa thể xác định khi các danh biểu được xác định và đưa vào bảng ký hiệu. Các giai đoạn sau đó như phân tích ngữ nghĩa và sinh mã trung gian mới đưa thêm các thông tin này vào và sử dụng chúng. Nói chung giai đoạn sinh mã thường đưa các thông tin chi tiết về vị trí lưu trữ dành cho định danh và sẽ sử dụng chúng khi cần thiết.

Bảng ký hiệu

1	position	...
2	initial	...
3	rate	. . .
4		

2. Xử lý lỗi

Mỗi giai đoạn có thể gặp nhiều lỗi, tuy nhiên sau khi phát hiện ra lỗi, tùy thuộc vào trình biên dịch mà có các cách xử lý lỗi khác nhau, chẳng hạn :

- Dừng và thông báo lỗi khi gặp lỗi đầu tiên (Pascal).
- Ghi nhận lỗi và tiếp tục quá trình dịch (C).

Giai đoạn phân tích từ vựng thường gặp lỗi khi các ký tự không thể ghép thành một token.

Giai đoạn phân tích cú pháp gặp lỗi khi các token không thể kết hợp với nhau theo đúng cấu trúc ngôn ngữ.

Giai đoạn phân tích ngữ nghĩa báo lỗi khi các toán hạng có kiểu không đúng yêu cầu của phép toán hay các kết cấu không có nghĩa đối với thao tác thực hiện mặc dù chúng hoàn toàn đúng về mặt cú pháp.

3. Các giai đoạn phân tích

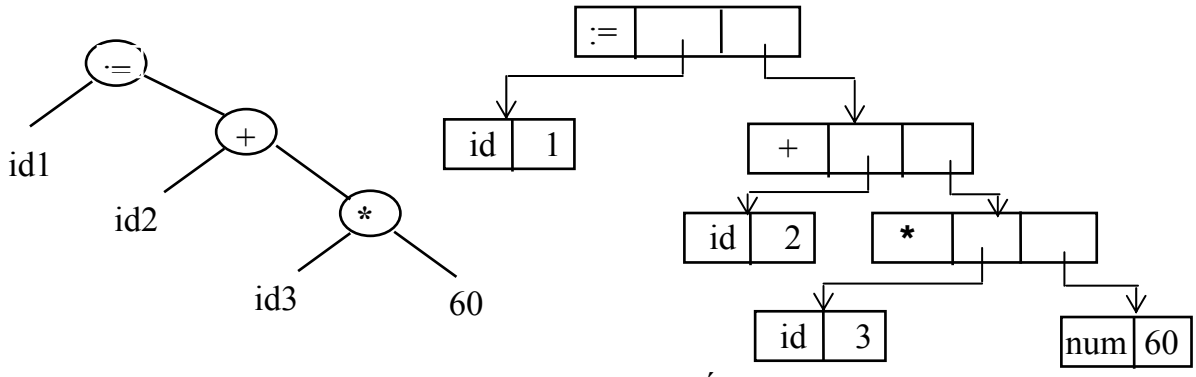
Giai đoạn phân tích từ vựng: Đọc từng ký tự gộp lại thành token, token có thể là một danh biểu, từ khóa, một ký hiệu,...Chuỗi ký tự tạo thành một token gọi là lexeme - trị từ vựng của token đó.

Ví dụ 1.7: Danh biểu rate có token id, trị từ vựng là rate và danh biểu này sẽ được đưa vào bảng ký hiệu nếu nó chưa có trong đó.

Giai đoạn phân tích cú pháp và phân tích ngữ nghĩa: Xây dựng cấu trúc phân cấp cho chuỗi các token, biểu diễn bởi cây cú pháp và kiểm tra ngôn ngữ theo cú pháp.

Ví dụ 1.8: Cây cú pháp và cấu trúc lưu trữ cho biểu thức

position := initial + rate * 60



Hình 1.7 - *Cây cú pháp và cấu trúc lưu trữ*

4. Sinh mã trung gian

Sau khi phân tích ngữ nghĩa, một số trình biên dịch sẽ tạo ra một dạng biểu diễn trung gian của chương trình nguồn. Chúng ta có thể xem dạng biểu diễn này như một chương trình dành cho một máy trừu tượng. Chúng có 2 đặc tính quan trọng : dễ sinh và dễ dịch thành chương trình đích.

Dạng biểu diễn trung gian có rất nhiều loại. Thông thường, người ta sử dụng dạng "mã máy 3 địa chỉ" (three-address code), tương tự như dạng hợp ngữ cho một máy mà trong đó mỗi vị trí bộ nhớ có thể đóng vai trò như một thanh ghi.

Mã máy 3 địa chỉ là một dãy các lệnh liên tiếp, mỗi lệnh có thể có tối đa 3 đối số.

Ví dụ 1.9: t1 := inttoreal (60)

t2 := id3 * t1

t3 := id2 + t2

id1 := t3

Dạng trung gian này có một số tính chất:

- Mỗi lệnh chỉ chứa nhiều nhất một toán tử. Do đó khi tạo ra lệnh này, trình biên dịch phải xác định thứ tự các phép toán, ví dụ * thực hiện trước +.

- Trình biên dịch phải tạo ra một biến tạm để lưu trữ giá trị tính toán cho mỗi lệnh.

- Một số lệnh có ít hơn 3 toán hạng.

5. Tối ưu mã

Giai đoạn tối ưu mã cố gắng cải thiện mã trung gian để có thể có mã máy thực hiện nhanh hơn. Một số phương pháp tối ưu hóa hoàn toàn bình thường.

Ví dụ 1.10:

Mã trung gian nêu trên có thể tối ưu thành:

t1 := id3 * 60.0

id1 := id2 + t1

Để tối ưu mã, ta thấy việc đổi số nguyên 60 thành số thực 60.0 có thể thực hiện một lần vào lúc biên dịch, vì vậy có thể loại bỏ phép toán inttoreal. Ngoài ra, t3 chỉ được dùng một lần để chuyển giá trị cho id1 nên có thể giảm bớt.

Có một khác biệt rất lớn giữa khối lượng tối ưu hoá mã được các trình biên dịch khác nhau thực hiện. Trong những trình biên dịch gọi là "trình biên dịch chuyên tối ưu", một phần thời gian đáng kể được dành cho giai đoạn này. Tuy nhiên, cũng có những phương pháp tối ưu giúp giảm đáng kể thời gian chạy của chương trình nguồn mà không làm chậm đi thời gian dịch quá nhiều.

6. Sinh mã

Giai đoạn cuối cùng của biên dịch là sinh mã đích, thường là mã máy hoặc mã hợp ngữ. Các vị trí vùng nhớ được chọn lựa cho mỗi biến được chương trình sử dụng. Sau đó, các chỉ thị trung gian được dịch lần lượt thành chuỗi các chỉ thị mã máy. Vấn đề quyết định là việc gán các biến cho các thanh ghi.

Ví dụ 1.11:

Sử dụng các thanh ghi (chẳng hạn R1, R2) cho việc sinh mã đích như sau:

```
MOVF    id3, R2
MULF    #60.0, R2
MOVF    id2, R1
ADDF    R2, R1
MOVF    R1, id1
```

Toán hạng thứ nhất và thứ hai của mỗi chỉ thị tương ứng mô tả đối tượng nguồn và đích. Chữ F trong mỗi chỉ thị cho biết chỉ thị đang xử lý các số chấm động (floating_point). Dấu # để xác định số 60.0 xem như một hằng số.

7. Ví dụ

Xem hình vẽ 1.8 (trang 10) mô tả các giai đoạn biên dịch cho biểu thức:

position := initial + rate * 60.

IV. NHÓM CÁC GIAI ĐOẠN

Các giai đoạn mà chúng ta đề cập ở trên là thực hiện theo trình tự logic của một trình biên dịch. Nhưng trong thực tế, cài đặt các hoạt động của nhiều hơn một giai đoạn có thể được nhóm lại với nhau. Thông thường chúng được nhóm thành hai nhóm cơ bản, gọi là: kỳ đầu (Front end) và kỳ sau (Back end).

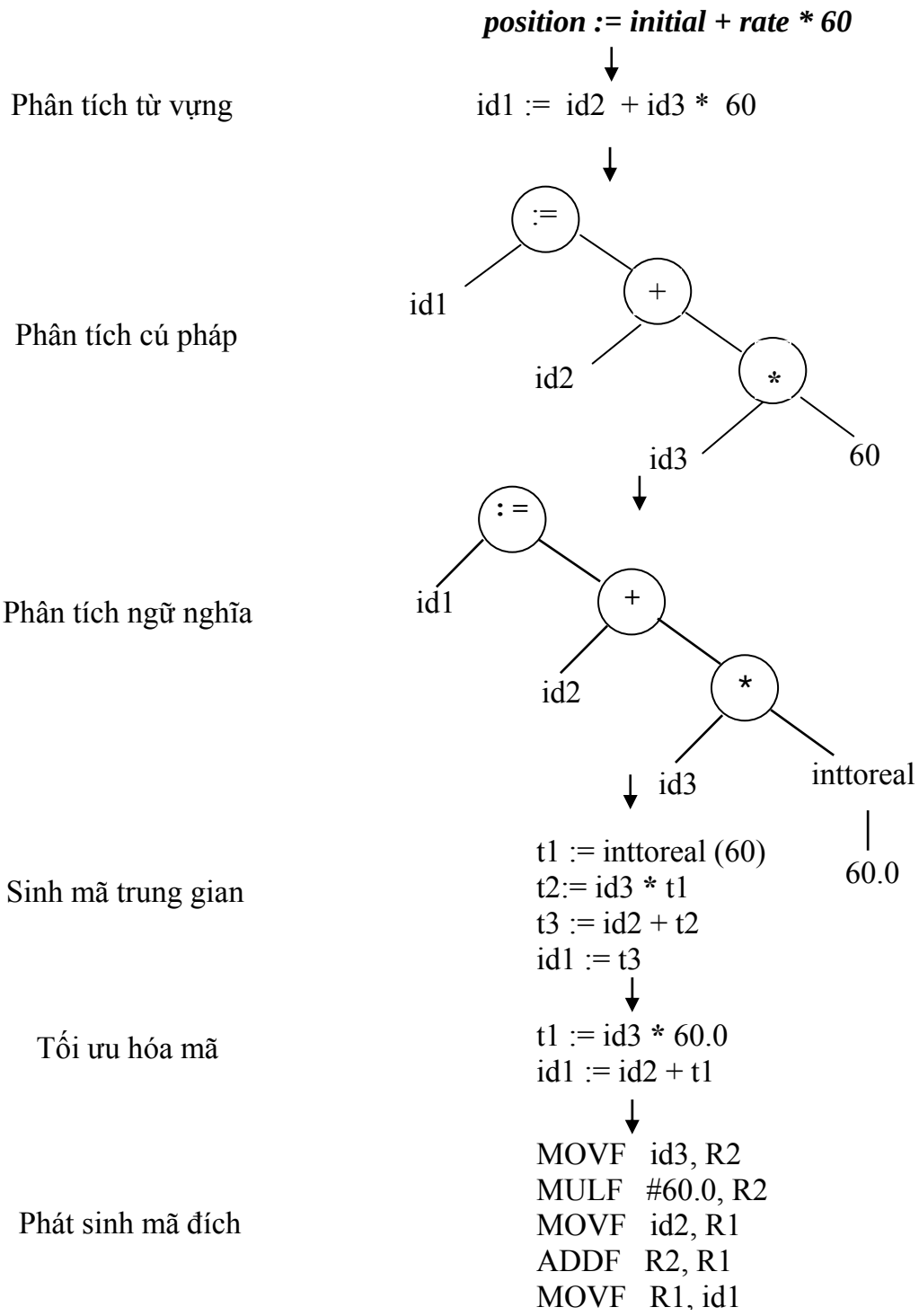
1. Kỳ đầu (Front End)

Kỳ đầu bao gồm các giai đoạn hoặc các phần giai đoạn phụ thuộc nhiều vào ngôn ngữ nguồn và hầu như độc lập với máy đích. Thông thường, nó chứa các giai đoạn sau: Phân tích từ vựng, Phân tích cú pháp, Phân tích ngữ nghĩa và Sinh mã trung gian. Một phần của công việc tối ưu hóa mã cũng được thực hiện ở kỳ đầu.

Front end cũng bao gồm cả việc xử lý lỗi xuất hiện trong từng giai đoạn.

2. Kỳ sau (Back End)

Kỳ sau bao gồm một số phần nào đó của trình biên dịch phụ thuộc vào máy đích và nói chung các phần này không phụ thuộc vào ngôn ngữ nguồn mà là ngôn ngữ trung gian. Trong kỳ sau, chúng ta gặp một số vấn đề tối ưu hoá mã, phát sinh mã đích cùng với việc xử lý lỗi và các thao tác trên bảng ký hiệu.



Hình 1.8 - *Minh họa các giai đoạn biên dịch một biểu thức*