PHENIKAA UNIVERSITY
Realizing your potential

CHƯƠNG TRÌNH DỊCH

Chương 1. Giới thiệu về chương trình dịch

TS. Phạm Văn Cảnh
Khoa Công nghệ thông tin

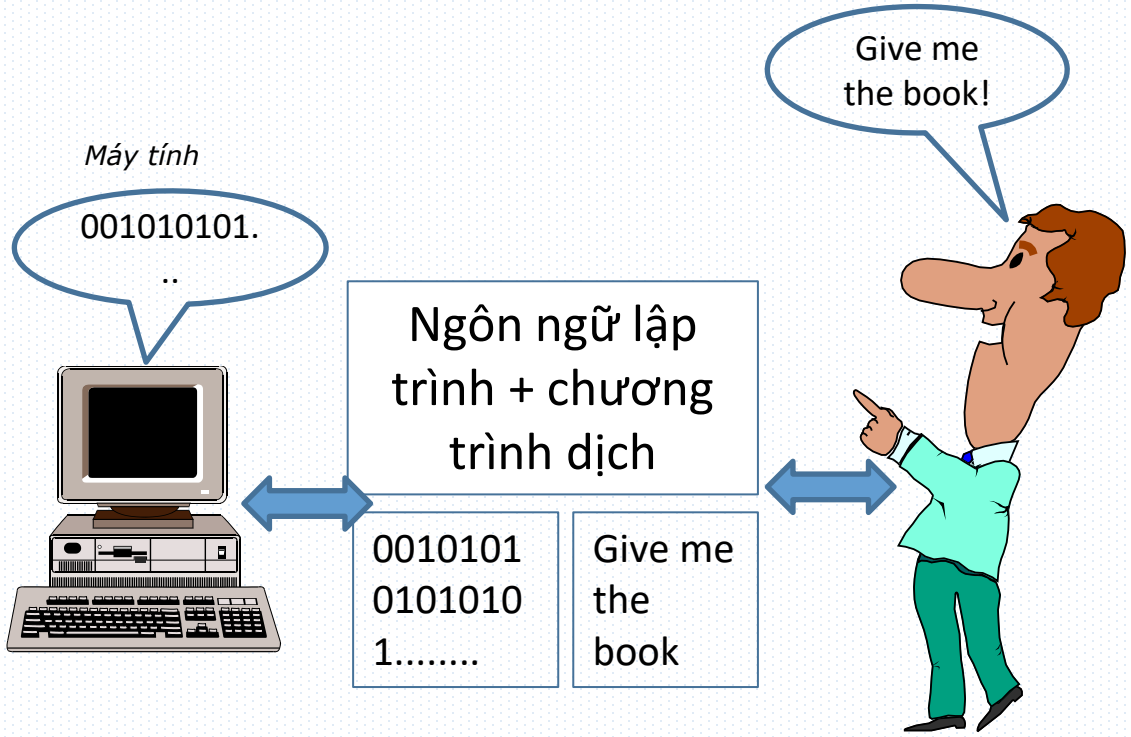
Email: canh.phamvan@phenikaa-uni.edu.vn

Nội dung

1. Giới thiệu về Chương trình dịch.
2. Ứng dụng của Chương trình dịch.
3. Phát triển dự án về Chương trình dịch.

1. Giới thiệu về Chương trình dịch

- Ngôn ngữ lập trình:
  - Để máy tính giao tiếp với con người.
  - Cú pháp đơn giản, gần gũi với ngôn ngữ tự nhiên.
  - Ví dụ: C/Pascal/Java v.v.. Gần giống Tiếng Anh.
- Chương trình dịch: chuyển tự động ngôn ngữ lập trình sang ngôn ngữ máy.

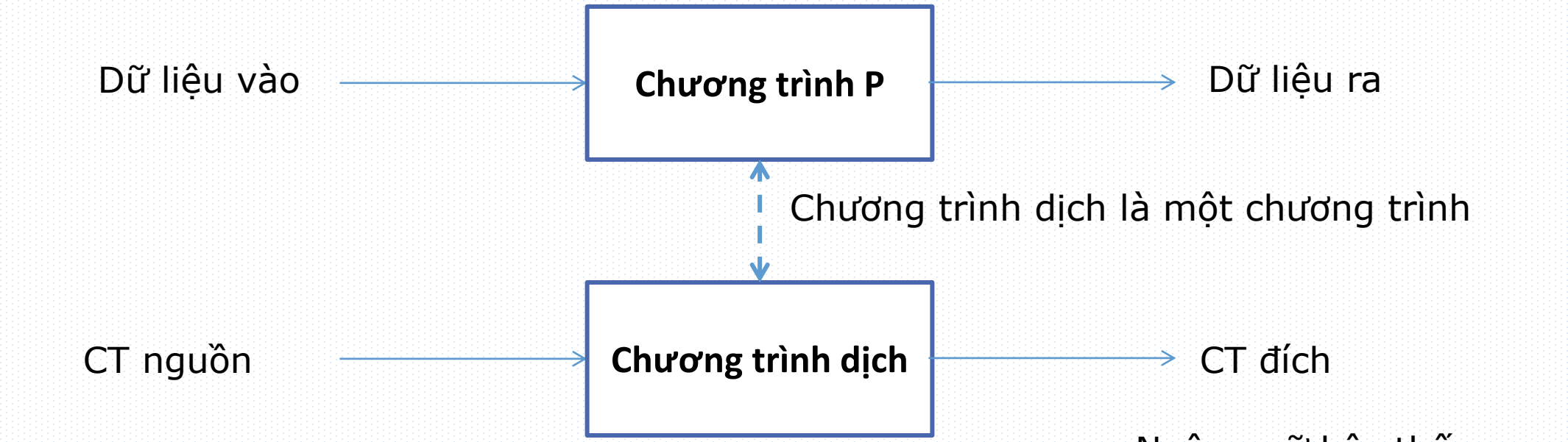


1. Giới thiệu về Chương trình dịch

PHENIKAA UNIVERSITY
Realizing your potential

- CCD nghiên cứu hai vấn đề
  - Lý thuyết thiết ké các ngôn ngữ lập trình.
  - Cách viết chương trình chuyển đổi từ một ngôn ngữ lập trình này sang một ngôn ngữ lập trình khác.
- Ứng dụng của chương trình dịch:
  - Các chương trình dịch dùng trong lập trình: Turbo Pascal, Turbo C/Borland C/DevC v.v..
  - Các bộ chuyển đổi văn bản: Phần mềm convert từ MS Word sang Pdf và ngược lại v.v..
  - Xử lý ngôn ngữ tự nhiên: dịch Anh – Việt, Việt –Anh tự động v.v..

1. Giới thiệu về Chương trình dịch



1. Giới thiệu về Chương trình dịch

PHENIKAA UNIVERSITY
Realizing your potential

- Khái niệm Chương trình dịch (CCD): chương trình dịch là hệ thống chuyển đổi ***đoạn văn*** viết trong ***ngôn ngữ A*** sang ***đoạn văn tương đương*** viết trong ***ngôn ngữ B***.
  - Khái niệm mang tính tổng quát.
  - Bài toán dịch ngôn ngữ chưa có lời giải đủ tốt.
- Thuật ngữ Compiler:
  - Xuất hiện vào năm 1950 do GraceMurray Hopper đưa ra.
  - Chương trình thật sự hoàn chỉnh đầu tiền: FORTRAN.

1. Giới thiệu về Chương trình dịch

PHENIKAA UNIVERSITY
Realizing your potential

- Các bài toán cụ thể của chương trình dịch:
  - Dịch một ngôn ngữ lập trình thành mã máy.
  - Dịch một ngôn ngữ lập trình bậc cao thành ngôn ngữ bậc thấp hơn.
  - Chuyển đổi đoạn mã giữa các ngôn ngữ lập trình.
  - Kiểm tra chính tả, ngữ pháp của các đoạn văn.
  - Mô tả hình ảnh (dịch từ hình ảnh thành văn bản).

1. Giới thiệu về Chương trình dịch

PHENIKAA UNIVERSITY
Realizing your potential

- Dịch từ ngôn ngữ lập trình thành mã máy là bài toán quan trọng, đóng góp rất lớn vào sự phát triển của ngành máy tính:
  - Chương trình lớn với mã máy vì quá phức tạp, dễ gây lỗi, nhàm chán.
  - Ban đầu chỉ là bộ dịch đơn giản từ ngôn ngữ cấp thấp (assembly) thành mã máy.
  - Tăng năng suất của lập trình viên (một dòng mã cấp cao tương đương với vài nghìn dòng mã máy).

Đặc trưng của CCD

- **Tính toàn vẹn**: kết quả ở ngôn ngữ đích phải hoàn toàn tương đương với đầu vào viết ở ngôn ngữ nguồn.
- **Tính trong suốt**:
  - Chương trình phải chia thành các bước độc lập.
  - Phải rõ ràng về kết quả sau từ bước thực hiện.
  - Có thể thực hiện việc hiệu chỉnh, sửa lỗi và tối ưu sau mỗi bước.
- **Tính hiệu quả**: chương trình dịch sử dụng không quá nhiều bộ nhớ và công suất tính toán, kết quả ở ngôn ngữ đích là đủ tốt.

Đặc trưng của CCD

PHENIKAA UNIVERSITY
Realizing your potential

❑ Tính chịu lỗi:

- Chương trình có thể chấp nhận một số lỗi của đầu vào và đưa ra các gợi ý xử lý phù hợp.
- Chương trình dừng ở ngay lỗi đầu tiên không thể coi là tốt.

Phân loại CCD

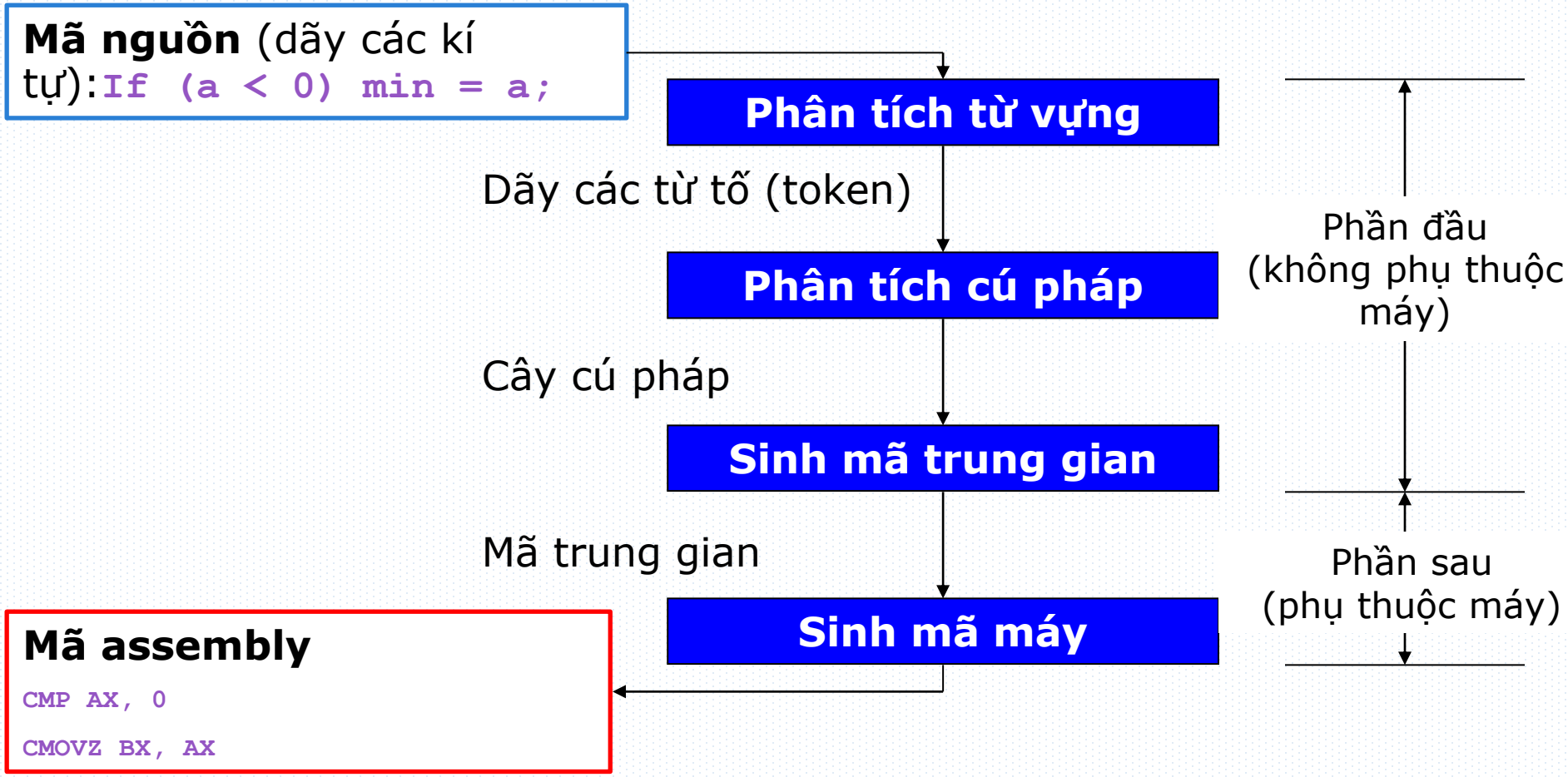
PHENIKAA UNIVERSITY
Realizing your potential

- Phân loại:
  - Theo số lần duyệt.
  - Theo mục đích.
  - Theo độ phức tạp: Assembly, Preproccessor, Compiler.
  - Theo phương pháp dịch chạy: Thông dịch, biên dịch.
  - Theo lớp văn phạm: LL(1), LR(1).

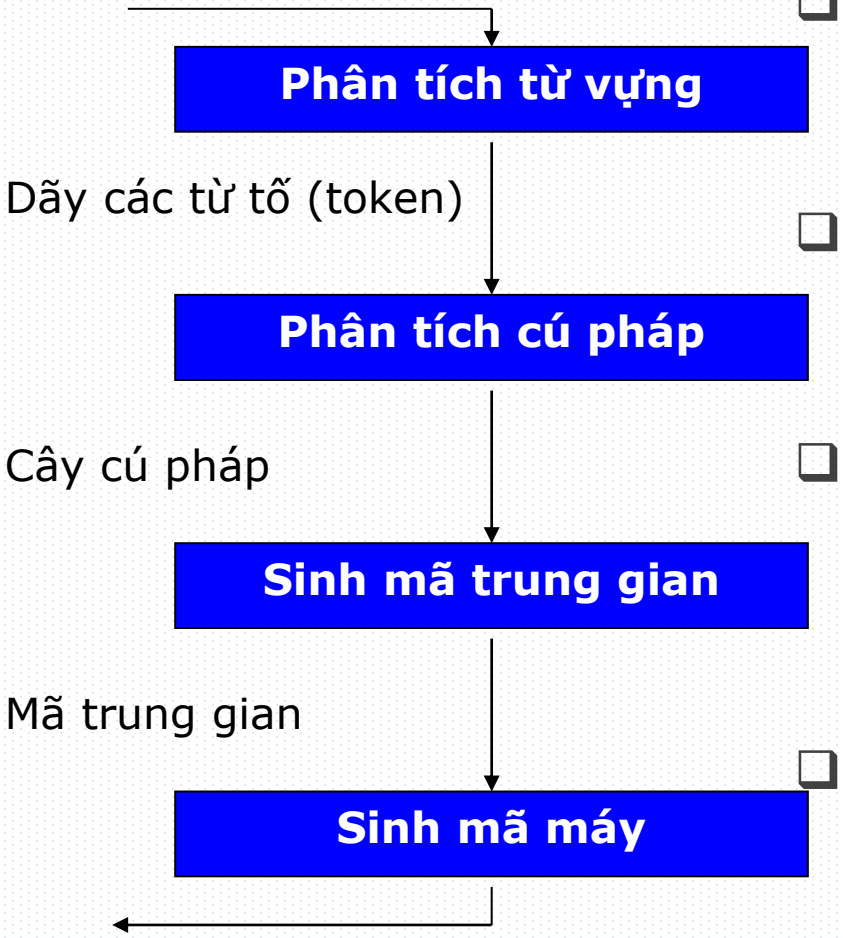
Phân loại CCD

- Trình biên dịch (compiler): nhận toàn bộ nguồn rồi dịch sang đích một lượt.
  - Compiler hoạt động giống như dịch giả.
- Trình thông dịch (interpreter): nhận mã nguồn từng phần, nhận được phần nào dịch (và thực thi) phần đó.
  - Interpreter hoạt động giống như người phiên dịch (các cuộc giao tiếp).

Cấu trúc của một CCD



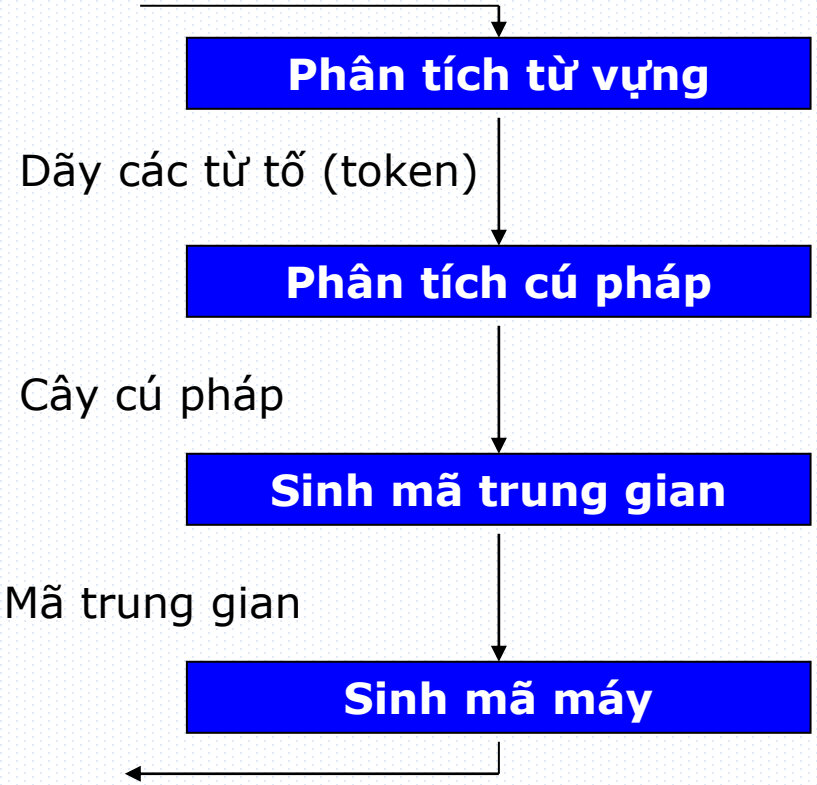
Cấu trúc của một CCD



Giai đoạn phân tích

- Phân tích từ vựng (lexical analyzer):
  - Đọc CT nguồn từ phải sang trái, nhóm các ký hiệu được gọi là từ tố: tên từ khóa, tên biến, số, phép toán v.v..
- Phân tích cú pháp (syntax analyzer):
  - Phân tích cấu trúc ngữ pháp của chương trình.
  - Các từ tố sẽ được nhóm lại theo các cấu trúc phân cấp.
- Phân tích ngữ nghĩa (sematic analyzer):
  - Phân tích tất cả các đặc tính khác của chương trình.
  - Tìm ra lỗi ngữ nghĩa, không tương thích. Ví dụ: gán bản ghi cho biến số nguyên.
- Phân tích cú pháp và phân tích ngữ nghĩa có thể tách rời hoặc kết hợp là một.

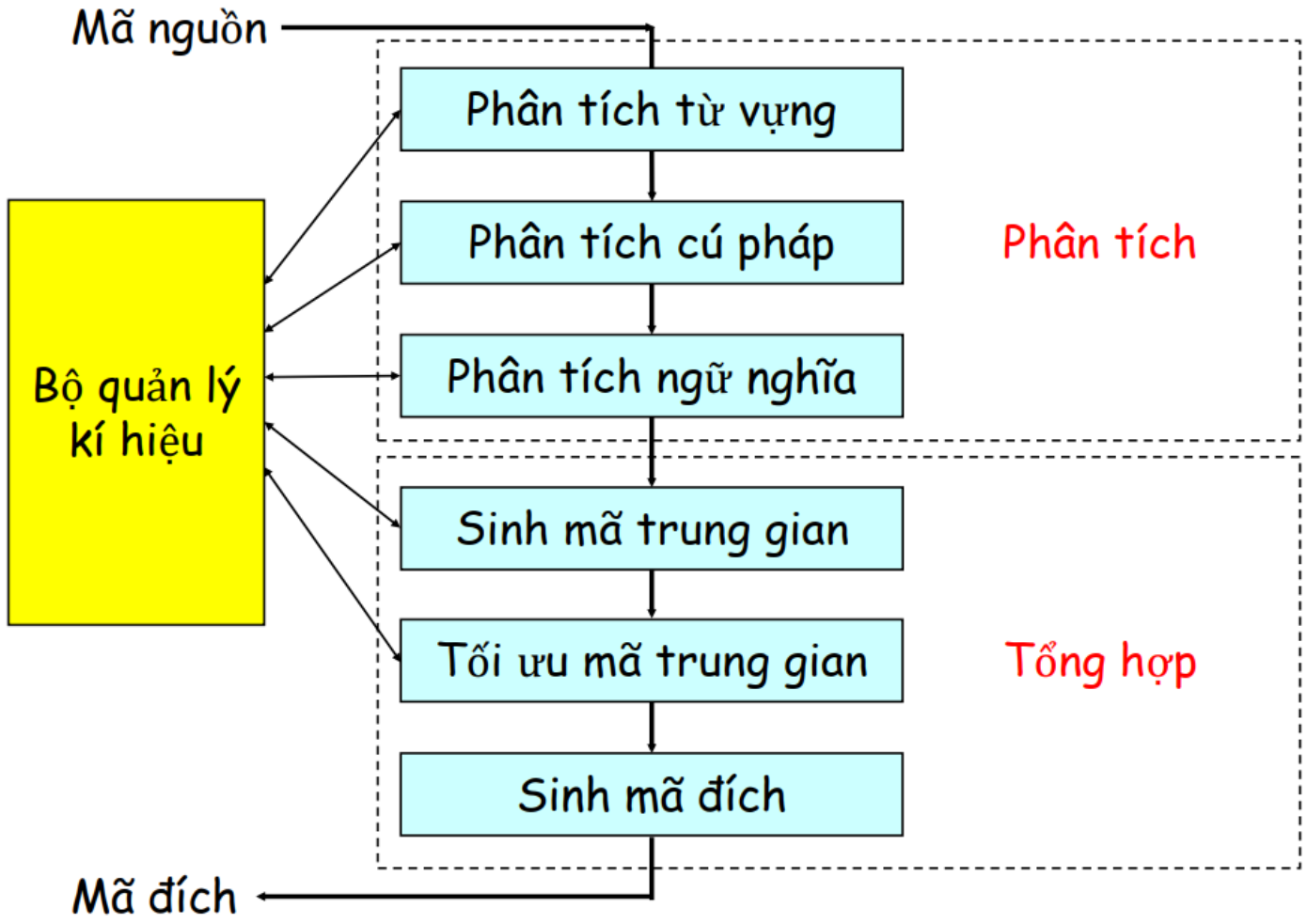
Cấu trúc của một CCD



Giai đoạn tổng hợp

- Sinh mã trung gian (intermediate code generator): sinh chương trình trong ngôn ngữ trung gian nhằm:
  - Dễ sinh và tối ưu hơn mã máy
  - Dễ chuyển đổi về mã máy
- Tối ưu mã (code optimizer): Sửa đổi chương trình trong ngôn ngữ trung gian nhằm cải tiến chương trình đích về hiệu năng.
- Sinh mã (code generator): Tạo ra chương trình đích từ ngôn ngữ trung gian đã tối ưu.

Cấu trúc chi tiết của một CCD



Cấu trúc của một CCD

PHENIKAA UNIVERSITY
Realizing your potential

❑Các thành phần khác

- **Quản lý bảng ký hiệu**
  - Ghi lại các ký hiệu, tên,.. Đã sử dụng trong chương trình nguồn cùng các thuộc tính kèm theo như kiểu, phạm vi, giá trị v.v..
- **Xử lý lỗi**
  - Ghi lại các vị trí có lỗi, loại lỗi, những lỗi khác có thể liên quan.
  - Giúp CCD có thể bỏ qua các lỗi không quan trọng.
  - Giúp CCD biết lỗi dây chuyền: dịch tiếp hay không?

Phân tích từ vựng

- Phân tích từ vựng (lexical analysis hay scanner) có nhiệm vụ chính sau đây:
  - Đọc dữ liệu đầu vào, loại bỏ các khối văn bản không cần thiết (dấu cách, dấu tab, các ghi chú,...).
  - Chia khối văn bản còn lại thành các từ vựng đồng thời xác định từ loại cho các từ vựng đó.
    - ✓ Các từ khóa của ngôn ngữ: for, if, switch,...
    - ✓ Tên riêng: "i", "j", "myList",...
    - ✓ Các hằng số: 17, 3.14, "%s", "\n",...
    - ✓ Các kí hiệu: "(", ")", ";", "+",...
- Các từ vựng thường được định nghĩa bởi từ điển (danh sách các từ khóa) và các biểu thức chính quy.

Phân tích từ vựng

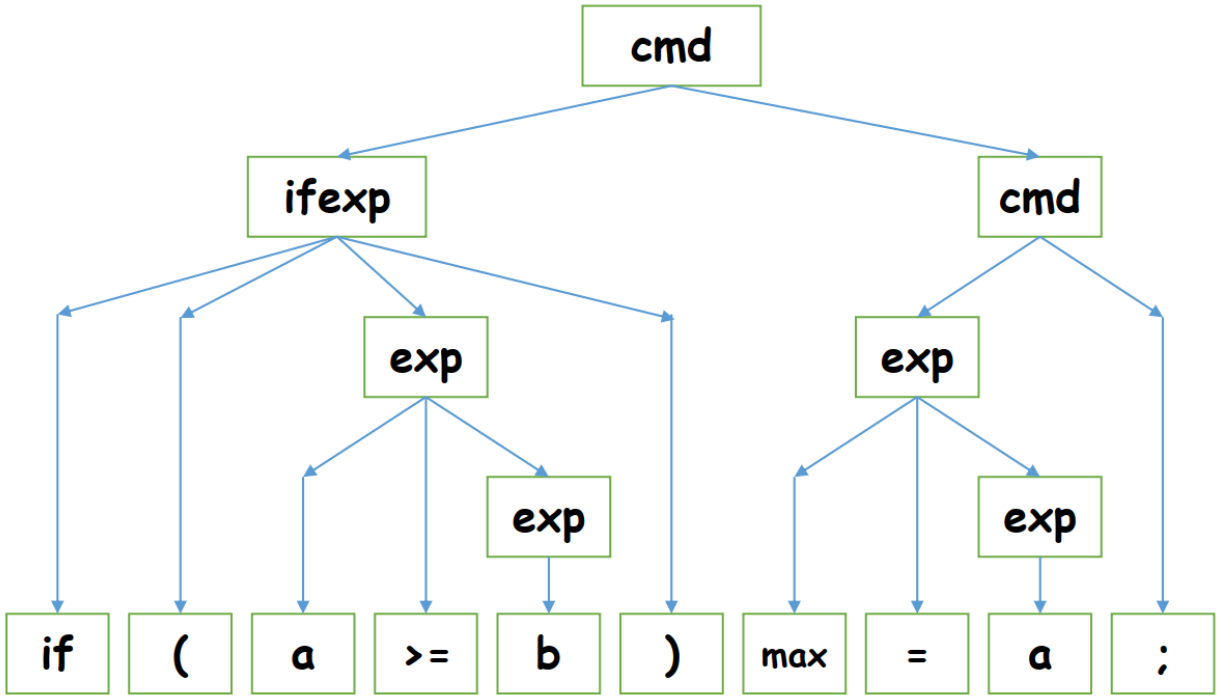
PHENIKAA UNIVERSITY
Realizing your potential

- Ví dụ về hoạt động của bộ phân tích từ vựng
  - Đầu vào: **if (a >= b) max = a;**
  - Kết quả:
    1. "**if**" – từ khóa
    2. "**(**" – kí hiệu (mở ngoặc)
    3. "**a**" – tên riêng
    4. "**>=**" – kí hiệu (lớn hơn hoặc bằng)*
    5. "**b**" – tên riêng
    6. "**)**" – kí hiệu (đóng ngoặc)
    7. "**max**" – tên riêng
    8. "**=**" – kí hiệu (bằng)
    9. "**a**" – tên riêng
    10. "**;**" – kí hiệu (chấm phẩy)

Phân tích cú pháp

- Phân tích cú pháp (syntax analysis hay parser) có nhiệm vụ chính là sinh cây phân tích (hay cây cú pháp – syntax tree) cho dãy từ vựng:
  - Các luật cú pháp thường được thể hiện ở dạng các luật phi ngữ cảnh (hoặc mở rộng)
  - Có rất nhiều phương pháp xây dựng một parser:
    - Sử dụng các kĩ thuật duyệt (top-down hoặc bottom-up).
    - Sử dụng kĩ thuật bảng phương án (automat đẩy xuống).
  - Thực tế: xây dựng một parser hiệu quả cho ngôn ngữ đó cũng là vấn đề khó.

- Ví dụ: Đầu vào **if (a >= b) max = a;**



Phân tích ngữ nghĩa

- Phân tích ngữ nghĩa (semantic analysis) sẽ dựa trên cây phân tích để thực hiện 2 việc chính:
  - Kiểm tra xem chương trình nguồn có các lỗi về ngữ nghĩa hay không
  - Tổng hợp các thông tin phục vụ cho giai đoạn sinh mã.
- Một vài tình huống về ngữ nghĩa:
  - Không tương thích kiểu (gán chuỗi vào số)
  - Không chuyển kiểu được.
- Tổng hợp thông tin để sinh mã:
  - Quyết định về kiểu của hằng số, biểu thức.
  - Tính toán trực tiếp các giá trị tĩnh.

Phân tích mã trung gian

- Các CCD sử dụng mã trung gian:
  - Mã 3 địa chỉ (TAC – three-address code)
  - Mã SSA (static single assignment)
- Các loại mã này giúp cho việc tối ưu hóa dễ dàng hơn (thực hiện ở pha sau).
- Ngoài ra bước này còn phân tích hoạt động của chương trình (control flow analysis) để cảnh báo một số rủi ro trong mã nguồn, chẳng hạn:
  - Biến sử dụng nhưng chưa khởi tạo
  - Đoạn mã vô dụng (không bao giờ chạy tới)

- Ví dụ:

```
// Mã nguồn
for (i = 0; i < 10; ++i) b[i] = i*i;

// Mã trung gian dạng TAC
     t1 := 0                ; initialize i
L1:  if t1 >= 10 goto L2    ; conditional jump
     t2 := t1 * t1          ; square of i
     t3 := t1 * 4           ; word-align address
     t4 := b + t3           ; address to store i*i
     *t4 := t2              ; store through pointer
     t1 := t1 + 1           ; increase i
     goto L1                ; repeat loop
L2:
```

Tối ưu mã trung gian

- Tối ưu (optimization) mã trung gian tạo ra mã có tốc độ chạy nhanh hơn.
  - Không phải mã ngắn hơn thì chạy nhanh hơn.
- Nhiều trình dịch cho phép lựa chọn chiến lược tối ưu mã thích hợp với mục đích sử dụng.
- Một số kĩ thuật tối ưu kinh điển:
  - Loại bỏ đoạn mã dư thừa.
  - Tận dụng lại kết quả tính toán đã có.
  - Sử dụng thanh ghi thay vì bộ nhớ.
  - Tách đoạn mã thành nhiều đoạn con chạy song song.

Tối ưu mã trung gian

- Nhiều kĩ thuật phụ thuộc vào kiểu của bộ xử lý
  - Máy tính có nhiều CPU.
  - Máy tính có siêu phân luồng.
  - Máy tính có bộ tiên đoán rẽ nhánh.
- Ví dụ về tối ưu mã trung gian:
  - Đoạn mã: **a=b;**
    c=a+10;
  - Được thay bằng: **a=b;**
    c=b+10;
  - Lý do: 2 lệnh trên có thể chạy song song trên máy nhiều CPU nên tốt hơn

Sinh mã đích

- Sinh mã đích (code generation) tạo ra mã đích (thường là dạng mã máy hoặc mã assembly) từ các mã TAC hoặc SSA đã được tối ưu.
- Đây là bước tương đối đơn giản (so với các bước khác), việc chuyển đổi từ mã TAC hoặc SSA sang các mã máy hoặc mã trung gian thường sử dụng các luật chuyển đổi đơn giản dạng nếu-thì.
- Ngoài ra bước này có thể bổ sung một số thông tin trong trường hợp mã đích là loại tái định vị.

2. Ứng dụng của CCD

- Từ ứng dụng ban đầu là chương trình dịch cho ngôn ngữ lập trình, nhiều kĩ thuật đã được áp dụng vào các nhiều ngành khác.
- Bộ kiểm tra chính tả.
  - Dùng cho các ngôn ngữ tự nhiên (trong các phần mềm soạn thảo văn bản).
  - Trợ giúp việc soạn thảo (intellisense, word suggestion).
- Bộ kiểm tra ngữ pháp.
  - Kiểm tra lỗi nhanh khi viết ngôn ngữ lập trình.
  - Soát lỗi khi viết tài liệu bằng ngôn ngữ tự nhiên.

2. Ứng dụng của CCD

- Sinh các bộ nhận dạng mã độc / virus:
  - Mỗi mã độc / virus được nhận dạng bởi một mẫu (pattern) hoặc một automat
  - Kết hợp nhiều automat làm một để tăng tốc độ của việc nhận dạng (thay vì phải chạy một automat cho mỗi virus, ta chạy một automat phát hiện đồng thời nhiều virus)
- Các công cụ về ngôn ngữ:
  - Bộ tìm kiếm văn bản hiệu quả
  - Bộ phát hiện ngôn ngữ
  - Bộ diễn dịch các giao thức

3. Phát triển dự án về CCD

1. Mục đích:

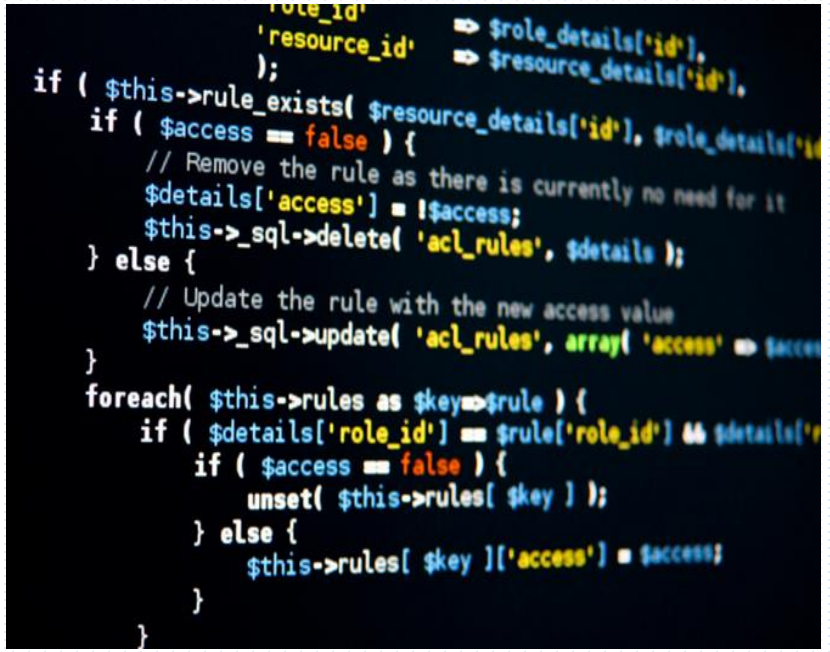
- Tạo một ngôn ngữ lập trình mới.
- Viết chương trình dịch hoàn chỉnh cho ngôn ngữ máy.

3. Phát triển dự án về CCD

2. Các bước tiến hành

a) Tìm hiểu/thiết kế ngôn ngữ nguồn/đích

- Có nhu cầu NNLT mới hay không?
  - Vi mạch mới sáng chế.
- Các NNLT hiện tại không đáp ứng được nhu cầu mới.
- Điều kiện cần đảm bảo:
  - Thiết kế tốt.
  - Tuân theo các tiêu chuẩn: begin, end, start, var v.v..
- Xây dựng CCD bằng các kỹ thuật hiện tại.



3. Phát triển dự án về CCD

PHENIKAA UNIVERSITY
Realizing your potential

b) Thiết kế và viết CCD

- Phân tích từ vựng: đọc văn bản tìm từ tố.
- Phân tích cú pháp: đọc từ tố phân tích cú pháp.
- Phân tích ngữ nghĩa: dựa vào luật xây dựng câu.
- Sinh mã trung gian: chuyển từ cây cú pháp sang MTG.
- Sinh mã: chương trình ở MTG → mã máy.
- Quản lý bảng ký hiệu.
- Xử lý lỗi.
- Xây dựng môi trường phát triển của CCD: gắn liền với một hệ thống cụ thể (ví dụ Turbo pascal gắn với công cụ soạn thảo chương trình)
- Kiểm tra và bảo trì.