CHƯƠNG TRÌNH DỊCH

Bài 1: Nhập môn

Nội dung

1. Giới thiệu
2. Khái niệm "chương trình dịch"
3. Một chương trình dịch điển hình
4. Hệ thống dịch
5. Ứng dụng chương trình dịch
6. Đối tượng nghiên cứu của môn học này
7. Mục tiêu của môn học
8. Thảo luận

Phần 1

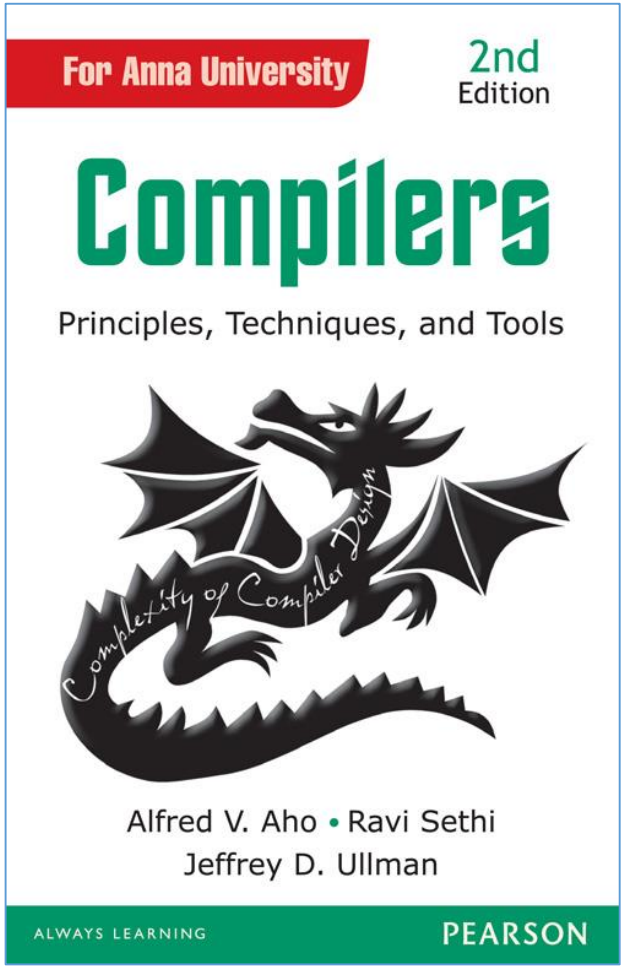
Giới thiệu

Môn học "chương trình dịch"

- Tên môn: chương trình dịch (compiler)
- Số tín chỉ: 4 (3 lý thuyết + 1 bài tập)
- Nội dung chính:
  - Tổng quan về môn học
  - Các khái niệm cơ sở
  - Phân tích từ vựng
  - Phân tích cú pháp
  - Các vấn đề khác
- Giảng viên: Trương Xuân Nam, khoa CNTT
- Email: truongxuannam@gmail.com

Tài liệu môn học

- Giáo trình chính: ***"Compilers: Principles, Techniques and Tools, 2nd edition"***
- Tài liệu tham khảo: ***"Nhập môn chương trình dịch"*** – Phạm Hồng Nguyên, ĐH Công nghệ
- Bài giảng, bài tập, mã nguồn, điểm số, thông báo,… sẽ được đưa lên site http://txnam.net, mục BÀI GIẢNG



Kiến thức yêu cầu

- Sử dụng được một ngôn ngữ lập trình phổ thông (C/C++, C#, Java,…) để viết chương trình
- Hiểu biết về tổ chức của máy tính:
  - Hoạt động của CPU (lệnh máy, cờ, thanh ghi, ô nhớ,…)
  - Cách làm việc của stack (trong máy tính)
  - Ngôn ngữ assembly
- Lý thuyết tính toán: automat, biểu thức chính quy, văn phạm phi ngữ cảnh, phân loại chomsky,…
- Cấu trúc dữ liệu: mảng, ngăn xếp, cây, danh sách,…
- Thuật toán: tìm kiếm, sắp xếp, từ điển, duyệt cây,…

Đánh giá kết quả

- Điểm môn học = ĐQT x **50%** + ĐTCK x **50%**
- Điểm quá trình:
  - Điểm danh
  - Bài làm trên lớp
  - Bài tập (nộp qua email)
- Điểm thi cuối kỳ:
  - Thi viết, 90 phút
  - Chỉ bài tập, không lý thuyết
  - Được sử dụng tài liệu tham khảo
  - Chi thi những gì học, không có giới hạn nội dung thi

Tại sao phải học môn này?

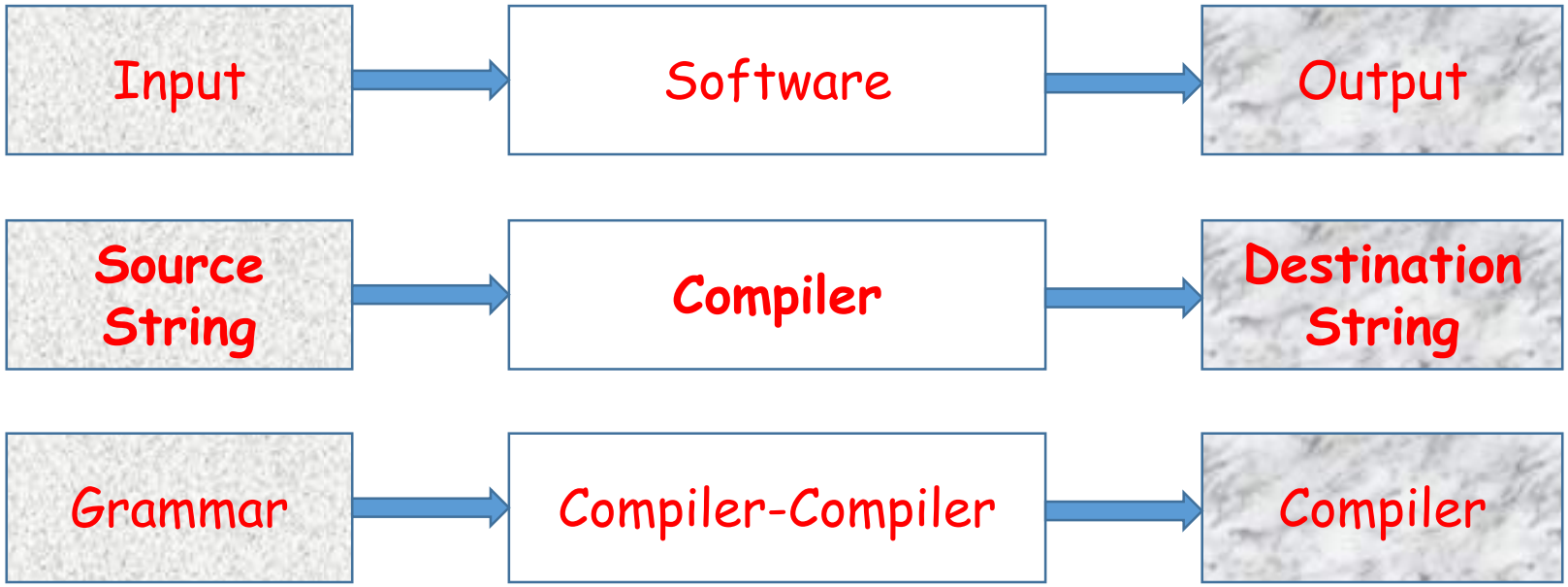
- Để có kiến thức về chương trình dịch
- Để có hiểu biết về cách thức hoạt động của các hệ thống dịch và khai thác tốt hơn các hệ thống đó
- Để có nâng cao kĩ năng viết chương trình
- Để có hiểu biết về điểm mạnh, điểm yếu của các ngôn ngữ lập trình, có lựa chọn ngôn ngữ lập trình phù hợp với công việc của bạn
- Có thêm lựa chọn cho đề tài làm tốt nghiệp
- Có điểm môn học và được ra trường

Phần 2

Khái niệm “chương trình dịch”

Khái niệm "chương trình dịch"

Tổng quát nhất: 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 "chương trình dịch"

- Định nghĩa như vậy quá tổng quát, bài toán dịch ngôn ngữ một cách tổng quát chưa có lời giải đủ tốt
- Người ta cố gắng giải quyết các bài toán cụ thể hơn và có ứng dụng thực tế hơn, chẳng hạn:
  - 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)

Biên dịch ngôn ngữ lập trình

- Trong các bài toán trên, "**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
  - Lập trình viên không thể viết 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)
- Đây cũng là bài toán nghiên cứu chính của môn học

Đặc trưng của chương trình dịch

- **Tính toàn vẹn**: kết quả ở ngôn ngữ đích phải hoàn toàn tương đương với đầu vào viết ở ngôn ngữ nguồn
  - Thành ngữ Italia: "traduttore, traditore" ("dịch thuật là sự phản bội")
- **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

Đặc trưng của chương trình dịch

- **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.
- **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.

Câu hỏi: Chương trình dịch nên có thêm những đặc trưng nào khác?

Phân loại chương trình dịch

- Phân loại cổ điển:
  - Trình biên dịch (compiler): nhận toàn bộ nguồn rồi dịch sang đích một lượt
  - Trình thông dịch (interpreter): nhận mã nguồn từng phần, nhận được phần nào dịch (và thực thi) phần đó
- Nhận xét:
  - Compiler hoạt động giống như dịch giả
  - Interpreter hoạt động giống như người phiên dịch (các cuộc giao tiếp)
- Hiện nay: ranh giới giữa compiler và interpreter ngày càng mờ dần

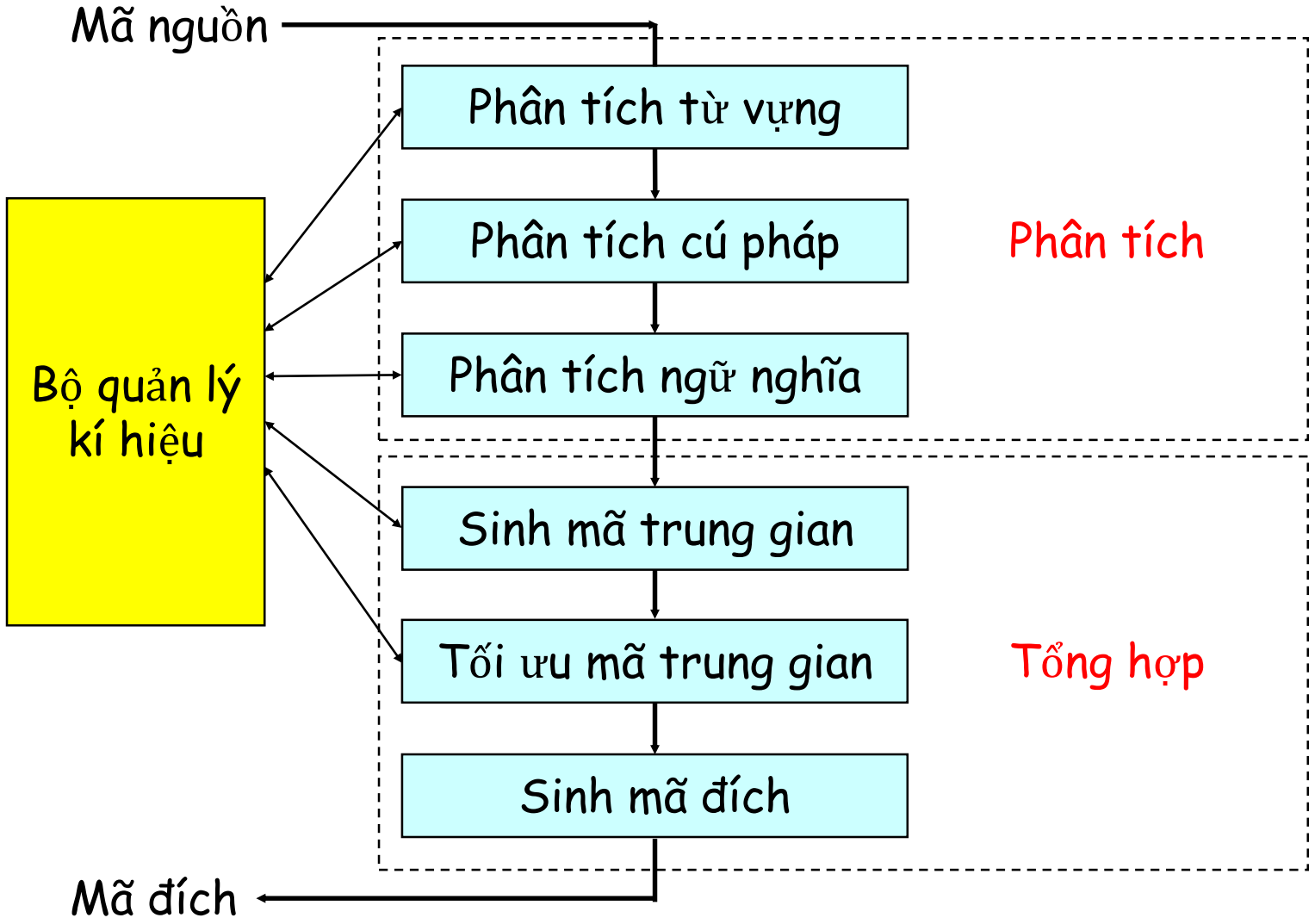
Phân loại chương trình dịch

- Ngay cả biên dịch cũng được chia thành 2 loại:
  - Tĩnh (statically): mã sinh ra chạy trực tiếp ngay
  - Động (dynamically): mã sinh ra cần thao tác tái định vị rồi mới có thể chạy được
- Một số ngôn ngữ lập trình kết hợp cả compiler và interpreter, chẳng hạn như java
  - Mã java được biên dịch thành mã bytecode
  - Máy ảo chạy mã bytecode ở dạng thông dịch
- Một số sử dụng compiler và just-in-time compiler
  - Mã C# được biên dịch thành mã IL
  - Mã IL được biên dịch thành mã máy trong lần chạy đầu

Phần 3

Một chương trình dịch điển hình

Một chương trình dịch điển hình



Pha 1: 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

Pha 1: phân tích từ vựng

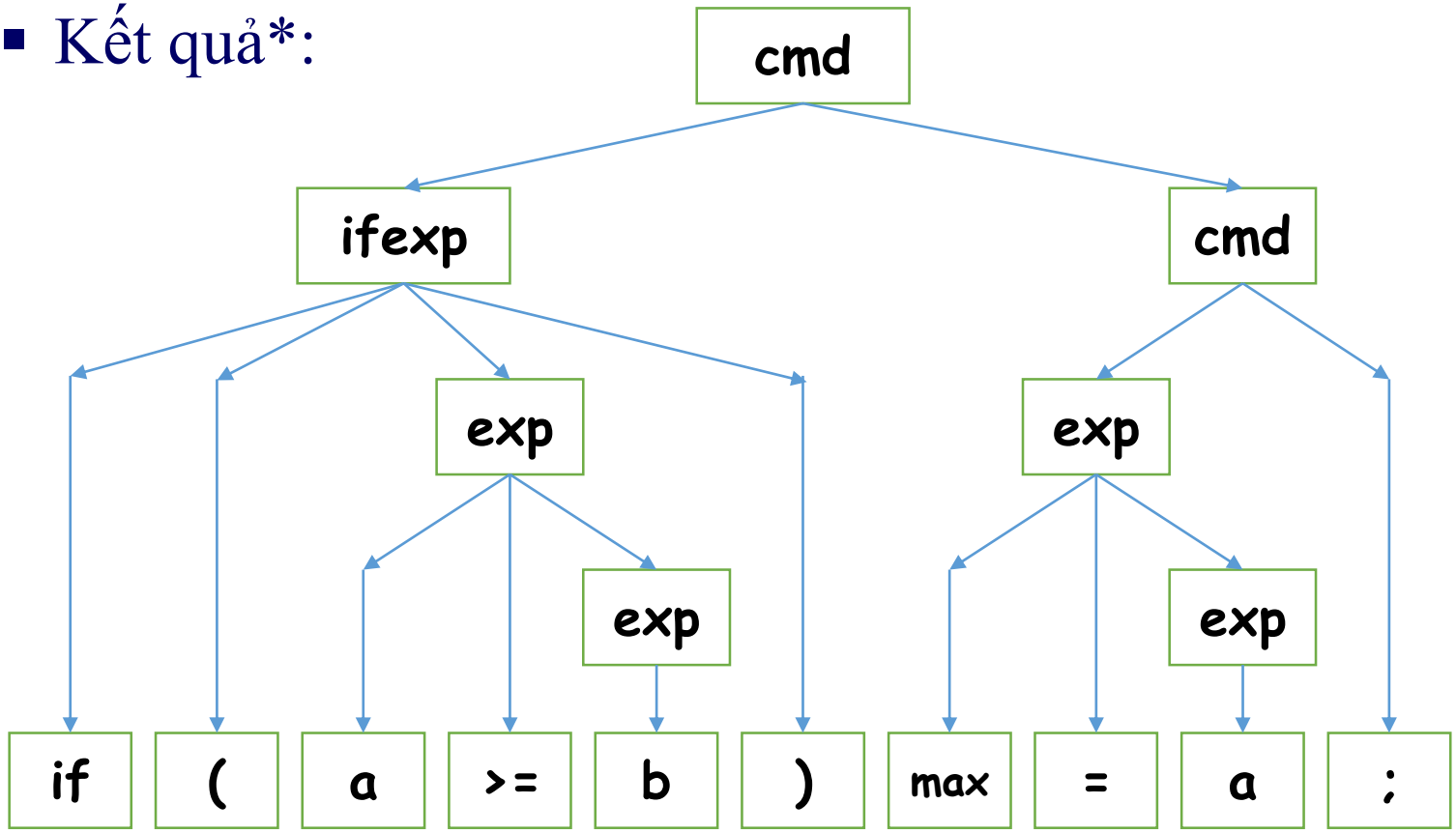
- Ví dụ về hoạt động của bộ phân tích từ vựng
  - Đầu vào: `if (a >= b) max = a;`
  - Kết quả:
    1. "**if**" – từ khóa
    2. "**(**" – kí hiệu (mở ngoặc)
    3. "**a**" – tên riêng
    4. "**>=**" – kí hiệu (lớn hơn hoặc bằng)*
    5. "**b**" – tên riêng
    6. "**)**" – kí hiệu (đóng ngoặc)
    7.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

Pha 2: 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: ngay cả với một ngôn ngữ có cú pháp đơn giản, xây dựng một parser hiệu quả cho ngôn ngữ đó cũng là vấn đề khó

Pha 2: phân tích cú pháp

- Ví dụ về hoạt động của bộ phân tích từ vựng
  - Đầu vào: `if (a >= b) max = a;`
  - Kết quả*:



Pha 3: 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

Pha 4: sinh mã trung gian

- Thông thường các trình dịch sử dụng loại mã 3 địa chỉ (TAC – three-address code) hoặc mã SSA (static single assignment) làm mã trung gian
- 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)

Pha 4: sinh mã trung gian

```
// 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:
```

Pha 5: 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

Pha 5: 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

Pha 6: 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

Phân tích và Tổng hợp

- 6 bước biên dịch có thể chia thành 2 giai đoạn:
  - Kỳ đầu (front-end), còn gọi là giai đoạn phân tích:
    - Gồm 3 bước đầu tiên
    - Giúp biến đổi từ ngôn ngữ nguồn sang một mô hình trung gian
    - Không phụ thuộc vào ngôn ngữ đích
  - Kỳ sau (back-end), còn gọi là giai đoạn tổng hợp:
    - Gồm 3 bước sau cùng
    - Giúp chuyển đổi từ mô hình trung gian sang ngôn ngữ đích
    - Không phụ thuộc vào ngôn ngữ nguồn
- Thực tế thì có nhiều ứng dụng chỉ sử dụng một vài phần của trình biên dịch; chẳng hạn như ứng dụng kiểm lỗi ngữ pháp trong các trình soạn thảo văn bản

Phần 4

Hệ thống dịch

Hệ thống dịch

- Chương trình dịch hiệu quả phải kèm với nó nhiều công cụ hỗ trợ, những công cụ này cùng với chương trình dịch tạo thành một hệ thống dịch hoàn chỉnh
  - Các IDE (môi trường phát triển tích hợp) của các ngôn ngữ lập trình, ngoài trình biên dịch thì còn nhiều công cụ khác như: bộ tiền xử lý mã nguồn, công cụ soạn thảo mã nguồn, công cụ hỗ trợ viết mã, công cụ trợ giúp, công cụ gỡ rối, công cụ phân tích mã,…
  - Các công cụ dịch tự động ngôn ngữ tự nhiên, ngoài module dịch tự động còn có các công cụ khác: bộ nhập liệu, từ điển, bộ nhận dạng, bộ tổng hợp tiếng nói,…

Hệ thống dịch

- Do việc nghiên cứu về chương trình dịch đã rất sâu sắc, nên nhiều thành phần của chương trình dịch đã được chuẩn hóa và có thể tách ra đứng độc lập
  - Bộ công cụ Lex/Flex: sinh tự động các scanner
  - Bộ công cụ Yacc/Bison: sinh tự động các parser
- Nhưng module độc lập này có thể có những ứng dụng riêng và có thể thay thế lẫn nhau
  - Hệ thống eclipse: có thể hỗ trợ nhiều ngôn ngữ lập trình bằng cách thêm các module mới
  - Bộ dịch Intel C++ có thể giúp dịch mã C++ tối ưu hơn nếu dùng chip Intel

Phần 5

Ứng dụng chương trình dịch

Ứng dụng chương trình dịch

- 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

Ứng dụng chương trình dịch

- 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

Phần 6

Đối tượng nghiên cứu của môn học này

Bài toán "Dịch"

- Như tìm hiểu ở các phần trên, chúng ta có thể thấy:
  - Chương trình dịch có rất nhiều biến thể: biên dịch, thông dịch, kiểm tra ngữ pháp, kiểm tra lỗi chính tả,…
  - Chương trình dịch rất phong phú ở đầu vào và đầu ra: dịch từ ngôn ngữ tự nhiên sang ngôn ngữ tự nhiên, dịch ngôn ngữ lập trình sang mã máy, dịch từ biểu thức chính quy thành automat,…
  - Chương trình dịch có nhiều ứng dụng: trong phần mềm dịch ngôn ngữ lập trình, trong xử lý các đoạn script nhúng trong file, trong các hệ thống phân tích, trong các bộ diễn dịch giao thức,…

Bài toán "Dịch"

- Để tránh vấn đề quá lan man, cần thu gọn đối tượng nghiên cứu của môn học bằng cách hạn chế chúng lại, ta phát biểu bài toán "Dịch" như sau: "***nghiên cứu các bước hoạt động của hệ thống biên dịch một ngôn ngữ lập trình đơn giản thành mã máy***"
  - Ngay cả với bài toán hạn chế như trên thì khối lượng kiến thức tìm hiểu rất lớn và các vấn đề được nêu ra cũng khá phức tạp
  - Phần kiến thức chúng ta được học sẽ tương đối sơ sài nếu muốn xây dựng một phần mềm compiler có giá trị sử dụng thật sự

Phần 7

Mục tiêu của môn học

Mục tiêu của môn học

- Mục tiêu về kiến thức:
  - Nắm được khái niệm chương trình dịch
  - Nắm được cách thức hoạt động của hệ thống dịch
  - Nắm được các phương pháp phân tích từ vựng đơn giản
  - Viết được module phân tích từ vựng đơn giản, hiệu quả
  - Nắm được một số phương pháp phân tích văn phạm
  - Viết được module phân tích văn phạm đơn giản, sử dụng các thuật toán được giảng dạy trên lớp
  - Hiểu được các vấn đề về ngữ nghĩa, sinh mã, tối ưu,…
  - Hiểu được hoạt động của một máy tính kiểu stack khi nó thực thi các câu lệnh

Mục tiêu của môn học

- Ứng dụng thực tế:
  - Áp dụng vào các lĩnh vực khác:
    - Giao thức trao đổi thông tin
    - Các công cụ xử lý văn bản
    - Xử lý ngôn ngữ tự nhiên
  - Nghiên cứu các vấn đề xa hơn trong chương trình dịch
    - Các thuật toán dịch hiệu quả dành cho các ngôn ngữ phức tạp hoặc ngôn ngữ tự nhiên
    - Các hệ thống trích rút thông tin từ văn bản
    - Các hệ thống dịch liên ngữ
  - Nghiên cứu các ứng dụng mới của hệ thống dịch

Phần 8

Thảo luận

Thảo luận

1. Hãy chỉ ra các bước hoạt động của trình dịch hợp ngữ (assembler) tương ứng với 6 pha của một compiler được mô tả ở trên
2. Một dịch giả dịch một bài thơ từ tiếng Anh sang tiếng Việt, theo bạn, dịch giả đó sẽ thực hiện những pha nào trong 6 pha của một complier? Hãy mô tả quá trình thực hiện và kết quả các bước đó
3. Một nhân chứng mô tả lại hình ảnh của đối tượng trong một vụ án cho cảnh sát, thì nhân chứng đó thực hiện những pha nào trong 6 pha trên?

Thảo luận

4. Các ưu điểm và nhược điểm của việc biên dịch so với việc thông dịch
5. Sự giống nhau và khác nhau giữa bài toán "dịch ngôn ngữ lập trình thành mã máy" và "dịch từ tiếng Anh sang tiếng Việt"
6. Sự giống nhau và khác nhau giữa một chương trình dịch và một người biên dịch
7. Kể tên một chương trình không phải chương trình dịch nhưng lại hoạt động như chương trình dịch
8. Việc diễn giải hành vi của động vật có phải là dịch?