Giáo trình môn tin học

MÔN TIN HỌC

Đối tượng : SV đại học chính quy toàn trường

Nội dung chính gồm 12 chương :

1. Phương pháp giải quyết bài toán bằng máy tính số.
2. Thể hiện dữ liệu trong máy tính số.
3. Tổng quát về lập trình bằng VB.
4. Qui trình thiết kế trực quan giao diện.
5. Các kiểu dữ liệu của VB.
6. Các lệnh định nghĩa & khai báo.
7. Biểu thức VB.
8. Các lệnh thực thi VB.
9. Định nghĩa thủ tục & sử dụng.
10. Tương tác giữa người dùng & chương trình.
11. Quản lý hệ thống file.
12. Linh kiện phần mềm & truy xuất database.

Tài liệu tham khảo :

- Tập slide bài giảng & thực hành của môn học này.
- 3 CD MSDN trong Microsoft Visual Studio.

MÔN TIN HỌC

Chương 1

PHƯƠNG PHÁP GIẢI QUYẾT BÀI TOÁN BẰNG MÁY TÍNH SỐ

1.1 Các khái niệm cơ bản về máy tính số

- Con người thông minh hơn các động vật khác nhiều. Trong cuộc sống, họ đã chế tạo ngày càng nhiều công cụ, thiết bị để hỗ trợ mình trong hoạt động. Các công cụ, thiết bị do con người chế tạo ngày càng tinh vi, phức tạp và thực hiện nhiều công việc hơn trước đây. Mỗi công cụ, thiết bị thường chỉ thực hiện được 1 vài công việc cụ thể nào đó. Thí dụ, cây chổi để quét, radio để bắt và nghe đài audio...
- Máy tính số (digital computer) cũng là 1 thiết bị, nhưng thay vì chỉ thực hiện 1 số chức năng cụ thể, sát với nhu cầu đời thường của con người, nó có thể thực hiện 1 số hữu hạn các chức năng cơ bản (tập lệnh), mỗi lệnh rất sơ khai chưa giải quyết trực tiếp được nhu cầu đời thường nào của con người. Cơ chế thực hiện các lệnh là tự động, bắt đầu từ lệnh được chỉ định nào đó rồi tuần tự từng lệnh kế tiếp cho đến lệnh cuối cùng. Danh sách các lệnh được thực hiện này được gọi là chương trình.

Các khái niệm cơ bản về máy tính số

- Các lệnh mà máy hiểu và thực hiện được được gọi là lệnh máy. Ta dùng ngôn ngữ để miêu tả các lệnh. Ngôn ngữ lập trình cấu thành từ 2 yếu tố chính yếu : cú pháp và ngữ nghĩa. Cú pháp qui định trật tự kết hợp các phần tử để cấu thành 1 lệnh (câu), còn ngữ nghĩa cho biết ý nghĩa của lệnh đó.
- Bất kỳ công việc (bài toán) ngoài đời nào cũng có thể được chia thành trình tự nhiều công việc nhỏ hơn. Trình tự các công việc nhỏ này được gọi là giải thuật giải quyết công việc ngoài đời. Mỗi công việc nhỏ hơn cũng có thể được chia nhỏ hơn nữa nếu nó còn phức tạp,... $\Rightarrow$ công việc ngoài đời có thể được miêu tả bằng 1 trình tự các lệnh máy (chương trình ngôn ngữ máy).

Các khái niệm cơ bản về máy tính số

- Vấn đề mấu chốt của việc dùng máy tính giải quyết công việc ngoài đời là lập trình (được hiểu nôm na là qui trình xác định trình tự đúng các lệnh máy để thực hiện công việc). Cho đến nay, lập trình là công việc của con người (với sự trợ giúp ngày càng nhiều của máy tính).
- Với công nghệ phần cứng hiện nay, ta chỉ có thể chế tạo các máy tính mà tập lệnh máy rất sơ khai, mỗi lệnh máy chỉ có thể thực hiện 1 công việc rất nhỏ và đơn giản ⇒ công việc ngoài đời thường tương đương với trình tự rất lớn (hàng triệu) các lệnh máy ⇒ Lập trình bằng ngôn ngữ máy rất phức tạp, tốn nhiều thời gian, công sức, kết quả rất khó bảo trì, phát triển.
- Ta muốn có máy luận lý với tập lệnh (được đặc tả bởi ngôn ngữ lập trình) cao cấp và gần gủi hơn với con người. Ta thường hiện thực máy này bằng 1 máy vật lý + 1 chương trình dịch. Có 2 loại chương trình dịch : trình biên dịch (compiler) và trình thông dịch (interpreter).

Trình biên dịch (Compiler)

- Chương trình biên dịch nhận một chương trình nguồn (thường được viết bằng ngôn ngữ cấp cao) và tạo ra một chương trình đối tượng tương ứng về chức năng nhưng thường được viết bằng ngôn ngữ cấp thấp (thường là ngôn ngữ máy).
- Nếu có lỗi xảy ra trong lúc dịch, trình biên dịch sẽ báo lỗi, cố gắng tìm vị trí đúng kế tiếp rồi tiếp tục dịch… Nhờ vậy, mỗi lần dịch 1 chương trình, ta sẽ xác định được nhiều lỗi nhất có thể có.
- Sau mỗi lần dịch, nếu không có lỗi, trình biên dịch sẽ tạo ra file chứa chương trình đối tượng (thí dụ file chương trình khả thi *.exe trên Windows).
- Để chạy chương trình, người dùng chỉ cần kích hoạt file khả thi (người dùng không biết và không cần quan tâm đến file chương trình nguồn).

Trình thông dịch (Interpreter)

- Chương trình thông dịch không tạo ra và lưu giữ chương trình đối tượng.
- Mỗi lần thông dịch 1 chương trình nguồn là 1 lần cố gắng chạy chương trình này theo cách thức sau :
  - dịch và chuyển sang mã thực thi từng lệnh một rồi nhờ máy chạy đoạn lệnh tương ứng.
  - Nếu có lỗi thì báo lỗi, nếu không có lỗi thì thông dịch lệnh kế tiếp... cho đến khi hết chương trình.
  - Như vậy, mỗi lần thông dịch chương trình, trình thông dịch chỉ thông dịch các lệnh trong luồng thi hành cần thiết chứ không thông dịch hết mọi lệnh của chương trình nguồn. Do đó, sau khi thông dịch thành công 1 chương trình, ta không thể kết luận rằng chương trình này không có lỗi.

So sánh trình biên dịch & trình thông dịch

- Mọi hoạt động xử lý trên mọi mã nguồn của chương trình (kiểm tra lỗi, dịch ra các lệnh đối tượng tương đương,...) đều được chương trình biên dịch thực hiện để tạo được chương trình đối tượng. Do đó sau khi dịch các file mã nguồn của chương trình, nếu không có lỗi, ta có thể kết luận chương trình không thể có lỗi thời điểm dịch (từ vựng, cú pháp). Quá trình biên dịch và quá trình thực thi chương trình là tách rời nhau : biên dịch 1 lần và chạy nhiều lần cho đến khi cần cập nhật version mới của chương trình.
- Chương trình thông dịch sẽ thông dịch từng lệnh theo luồng thi hành của chương trình bắt đầu từ điểm nhập của chương trình, thông dịch 1 lệnh gồm 2 hoạt động : biên dịch lệnh đó và thực thi các lệnh kết quả. Nếu 1 đoạn lệnh cần được thực thi lặp lại thì trình thông dịch sẽ phải thông dịch lại tất cả đoạn lệnh đó. Điều này sẽ làm cho việc chạy chương trình trong chế độ thông dịch không hiệu quả.
- Việc chạy chương trình bằng cơ chế thông dịch đòi hỏi chương trình thông dịch và chương trình ứng dụng cần chạy phải tồn tại đồng thời trong bộ nhớ máy tính, do đó có nguy cơ chạy không được các chương trình lớn nếu tài nguyên của máy không đủ cho cả 2 chương trình thông dịch và chương trình ứng dụng.

Các khái niệm cơ bản về máy tính số

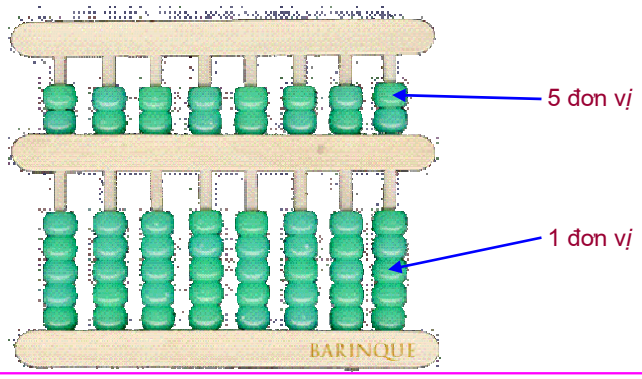
- Gọi ngôn ngữ máy vật lý là N_0. Trình biên dịch ngôn ngữ N_1 sang ngôn ngữ N_0 sẽ nhận đầu vào là chương trình được viết bằng ngôn ngữ N_1, phân tích từng lệnh N_1 rồi chuyển thành danh sách các lệnh ngôn ngữ N_0 có chức năng tương đương. Để viết chương trình dịch từ ngôn ngữ N_1 sang N_0 dễ dàng, độ phức tạp của từng lệnh ngôn ngữ N_1 không quá cao so với từng lệnh ngôn ngữ N_0.
- Sau khi có máy luận lý hiểu được ngôn ngữ luận lý N_1, ta có thể định nghĩa và hiện thực máy luận lý N_2 theo cách trên và tiếp tục đến khi ta có 1 máy luận lý hiểu được ngôn ngữ N_m rất gần g**ũi** với con người, dễ dàng miêu tả giải thuật của bài toán cần giải quyết...
- Nhưng qui trình trên chưa có điểm dừng, với yêu cầu ngày càng cao và kiến thức ngày càng nhiều, người ta tiếp tục định nghĩa những ngôn ngữ mới với tập lệnh ngày càng gần g**ũi** hơn với con người để miêu tả giải thuật càng dễ dàng, gọn nhẹ và trong sáng hơn.

Các cấp độ ngôn ngữ lập trình

- Ngôn ngữ máy vật lý là loại ngôn ngữ thấp nhất mà người lập trình bình thường có thể dùng được. Các lệnh và tham số của lệnh được miêu tả bởi các số binary (hay hexadecimal - sẽ được miêu tả chi tiết trong chương 2). Đây là loại ngôn ngữ mà máy vật lý có thể hiểu trực tiếp, nhưng con người thì gặp nhiều khó khăn trong việc viết và bảo trì chương trình ở cấp này.
- Ngôn ngữ assembly rất gần với ngôn ngữ máy, những lệnh cơ bản nhất của ngôn ngữ assembly tương ứng với lệnh máy nhưng được biểu diễn dưới dạng gợi nhớ. Ngoài ra, người ta tăng cường thêm khái niệm "lệnh macro" để nâng sức mạnh miêu tả giải thuật.
- Ngôn ngữ cấp cao theo trường phái lập trình cấu trúc như Pascal, C,... Tập lệnh của ngôn ngữ này khá mạnh và gần với tư duy của người bình thường.
- Ngôn ngữ hướng đối tượng như C++, Visual Basic, Java, C#,... cải tiến phương pháp cấu trúc chương trình sao cho trong sáng, ổn định, dễ phát triển và thay thế linh kiện.

1.2 Lịch sử phát triển máy tính số

- Máy tính xuất hiện từ rất lâu theo nhu cầu buôn bán và trao đổi tiền tệ.
- Bàn tính tay abacus là dạng sơ khai của máy tính.



Các thế hệ máy tính số

Blaise Pascal (Pháp-1642)

Charles Babbage (Anh-1830)

ENIAC (1946)
18.000 bóng đèn
1500 rờ le
30 tấn
140 KW

Intel 8080 (1974)
được xem như CPU đầu tiên được tích hợp trên 1 chip

IBM 360 (1965)

Von Neumann (1945)

PDP-1 (1961)

80x86 (1978)

Cơ

Đèn điện tử

Transistors

IC

?

(1642 - 1945)

(1945 - 1955)

(1955 - 1965)

(1965 - 1980)

(1980 - ????)

Herman Hollerith lập IBM (International Business Machine) ở Mỹ - 1890

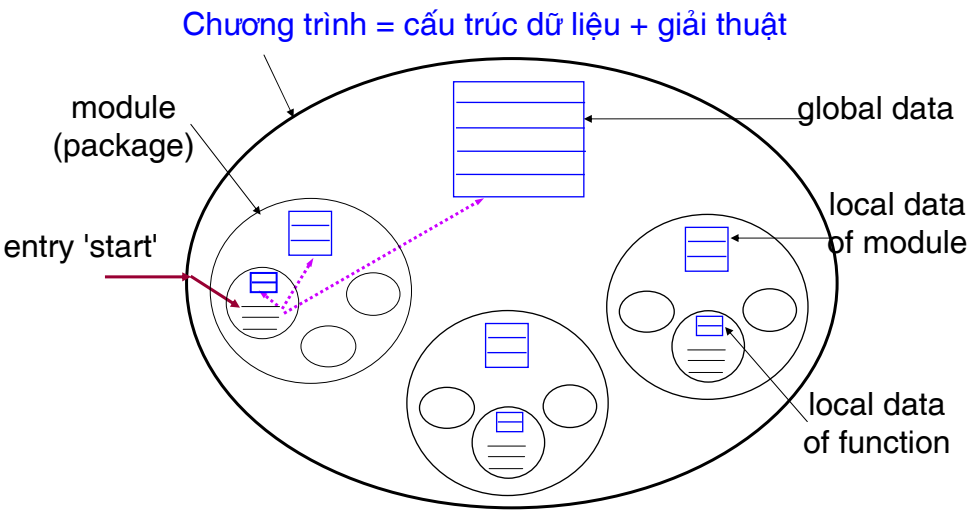
Bộ nhớ dây trễ, tĩnh điện. Giấy, phiếu đục lỗ. Băng từ

Bộ nhớ xuyến từ. Băng từ, trống từ, đĩa từ.

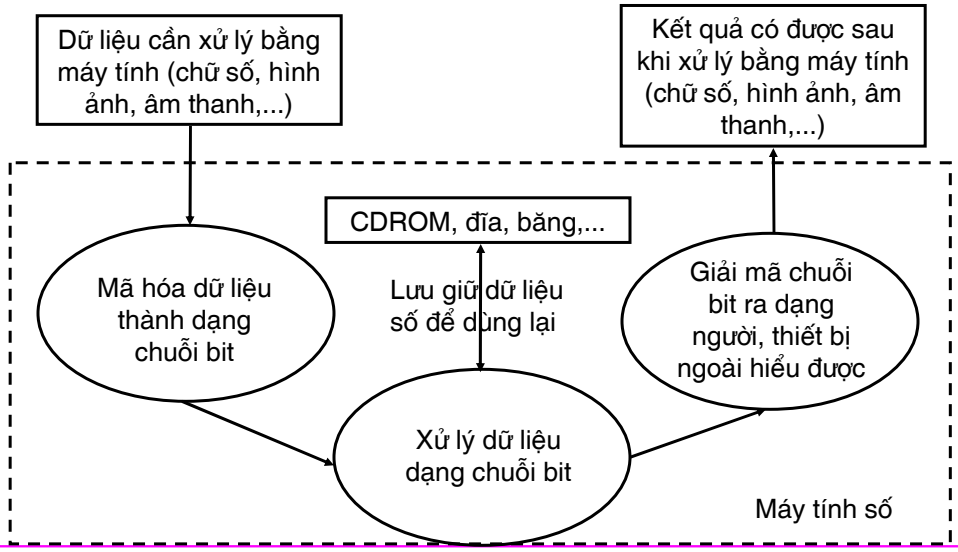
1.3 Dữ liệu & chương trình

- Các lệnh của chương trình (code) sẽ truy xuất (đọc và/hoặc ghi) thông tin (dữ liệu).
- Chương trình giải quyết bài toán nào đó có thể truy xuất nhiều dữ liệu khác nhau với tính chất rất đa dạng. Để truy xuất 1 dữ liệu cụ thể, ta cần 3 thông tin về dữ liệu đó :
  - **tên nhận dạng** (identifier) xác định vị trí của dữ liệu.
  - **kiểu dữ liệu** (type) miêu tả cấu trúc của dữ liệu.
  - **tầm vực truy xuất** (visibility) xác định các lệnh được phép truy xuất dữ liệu tương ứng.
- Chương trình cổ điển = dữ liệu + giải thuật.
- Chương trình con (function, subroutine,...) là 1 đoạn code thực hiện chức năng được dùng nhiều lần ở nhiều vị trí trong chương trình, được nhận dạng thông qua 1 tên gọi. Chương trình con cho phép cấu trúc chương trình, sử dụng lại code...

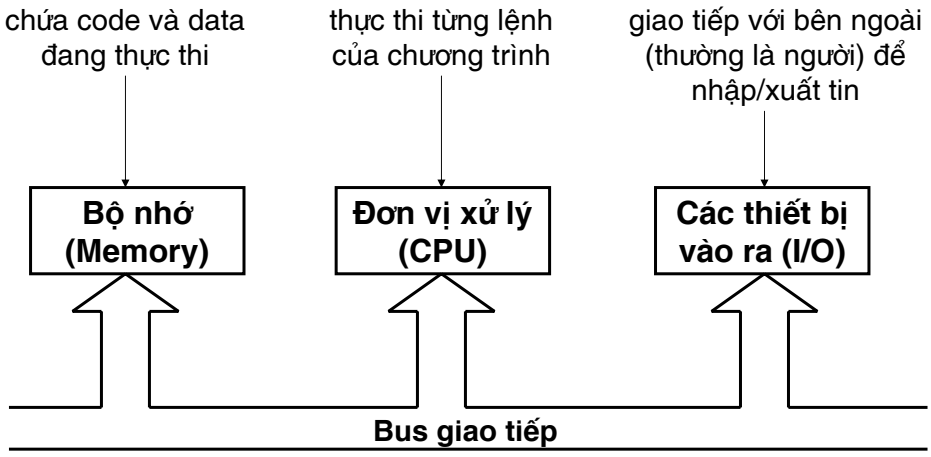
Cấu trúc 1 chương trình cổ điển



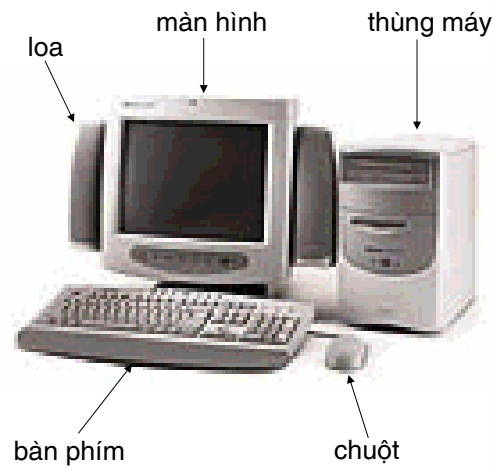
1.4 Qui trình tổng quát giải quyết bài toán bằng máy tính số

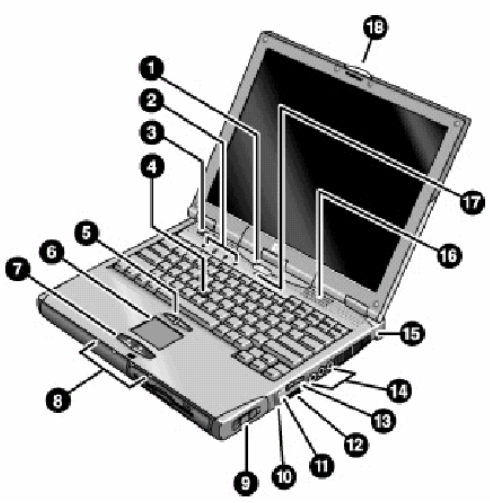


Mô hình máy tính số Von Neumann



Hình dạng vật lý của vài máy tính





1.5 Phương pháp phân tích từ-trên-xuống

Trong quá khứ, phương pháp thường sử dụng để phân tích bài toán là phương pháp từ-trên-xuống (top-down analysis).

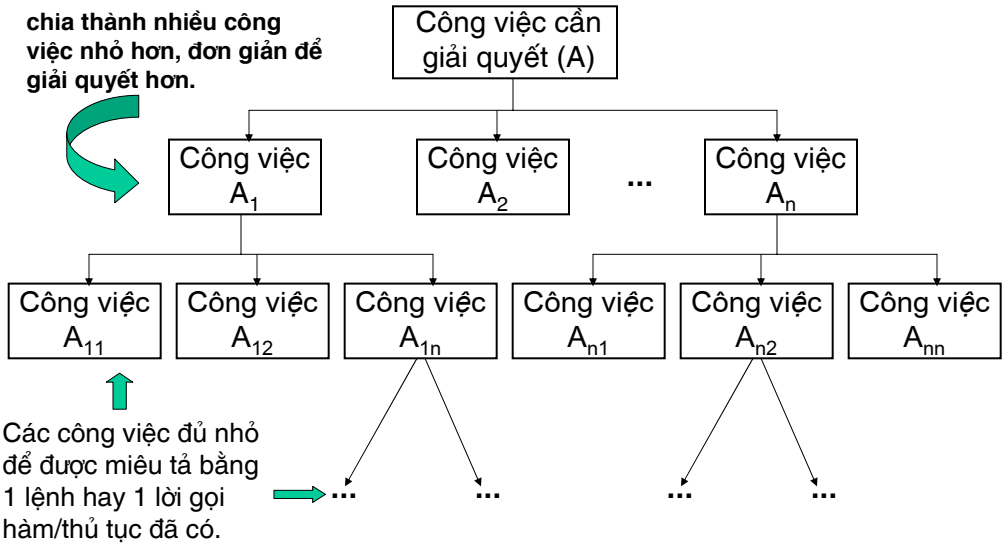
Nội dung của phương pháp này là xét xem, muốn giải quyết vấn đề nào đó thì cần phải làm những công việc nhỏ hơn nào. Mỗi công việc nhỏ hơn tìm được lại được phân thành những công việc nhỏ hơn nữa, cứ như vậy cho đến khi những công việc phải làm là những công việc thật đơn giản, có thể thực hiện dễ dàng.

Thí dụ việc học lấy bằng kỹ sư CNTT khoa CNTT ĐHBK TP.HCM có thể bao gồm 9 công việc nhỏ hơn là học từng học kỳ từ 1 tới 9, học học kỳ i là học n môn học của học kỳ đó, học 1 môn học là học m chương của môn đó,...

Hình vẽ của slide kế cho thấy trực quan của việc phân tích top-down.

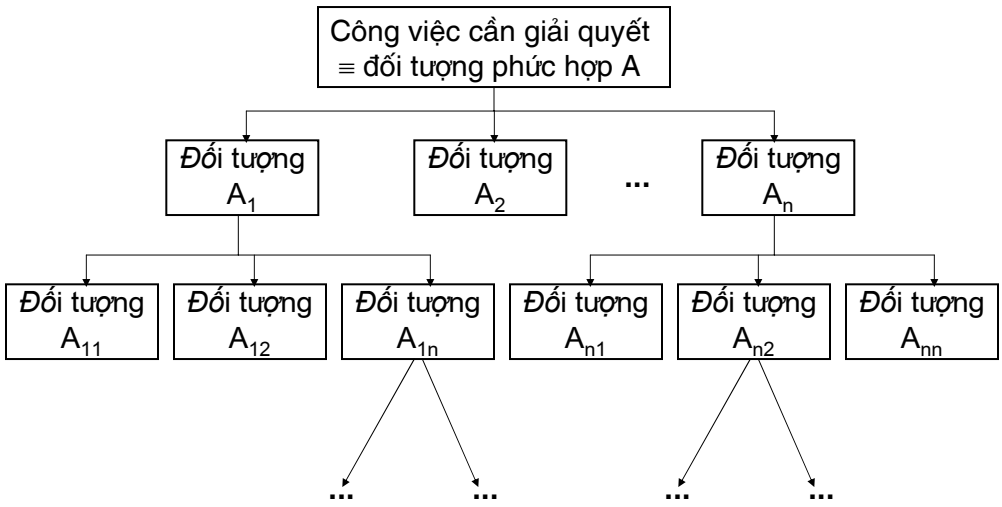
Phương pháp phân tích từ-trên-xuống

chia thành nhiều công việc nhỏ hơn, đơn giản để giải quyết hơn.



Các công việc đủ nhỏ để được miêu tả bằng 1 lệnh hay 1 lời gọi hàm/thủ tục đã có.

Phương pháp phân tích từ-trên-xuống



MÔN TIN HỌC

Chương 2

THỂ HIỆN DỮ LIỆU TRONG MÁY TÍNH SỐ

2.1 Cơ bản về việc lưu trữ và xử lý tin trong máy tính
2.2 Cơ bản về hệ thống số
2.3 Các phương pháp chuyển miêu tả số
2.4 Biểu diễn dữ liệu trong máy tính
2.5 Hệ thống file
2.6 Quản lý hệ thống file

2.1 Cơ bản về việc lưu trữ và xử lý tin trong máy tính

Phần tử nhớ nhỏ nhất của máy tính số chỉ có thể chứa 2 giá trị : 0 và 1 (ta gọi là bit).

Ta kết hợp nhiều phần tử nhớ để có thể miêu tả đại lượng lớn hơn. Thí dụ ta dùng 8 bit để miêu tả $2^8 = 256$ giá trị khác nhau. Dãy 8 bit nhớ được gọi là byte, đây là 1 ô nhớ trong bộ nhớ của máy tính.

Bộ nhớ trong của máy tính được dùng để chứa dữ liệu và code của chương trình đang thực thi. Nó là 1 dãy đồng nhất các ô nhớ 8 bit, mỗi ô nhớ được truy xuất độc lập thông qua địa chỉ của nó (tên nhận dạng). Thường ta dùng chỉ số từ 0 - n để miêu tả địa chỉ của từng ô nhớ.

Mặc dù ngoài đời ta đã quen dùng hệ thống số thập phân, nhưng về phần cứng bên trong máy tính, máy chỉ có thể chứa và xử lý trực tiếp dữ liệu ở dạng nhị phân. Do đó trong chương này, ta sẽ giới thiệu các khái niệm nền tảng về hệ thống số và cách miêu tả dữ liệu trong máy tính.

2.2 Cơ bản về hệ thống số

Hệ thống số (number system) là công cụ để biểu thị đại lượng. Một hệ thống số gồm 3 thành phần chính :

1. **cơ số** : số lượng ký số (ký hiệu để nhận dạng các số cơ bản).
2. **qui luật kết hợp các ký số** để miêu tả 1 đại lượng nào đó.
3. **các phép tính cơ bản** trên các số.

Trong 3 thành phần trên, chỉ có thành phần 1 là khác nhau giữa các hệ thống số, còn 2 thành phần 2 và 3 thì giống nhau giữa các hệ thống số.

Thí dụ :
- hệ thống số thập phân (**hệ thập phân**) dùng 10 ký số : 0,1,2,3,4,5,6,7,8,9.
- **hệ nhị phân** dùng 2 ký số : 0,1.
- hệ bát phân dùng 8 ký số : 0,1,2,3,4,5,6,7.
- **hệ thập lục phân** dùng 16 ký số : 0 đến 9,A,B,C,D,E,F.

Cơ bản về hệ thống số - Qui luật miêu tả lượng

Biểu diễn của lượng Q trong hệ thống số B (B>1) là :

$d_n d_{n-1} ... d_1 d_0 d_{-1} ... d_{-m} \Leftrightarrow$

$Q = d_n * B^n + d_{n-1} * B^{n-1} + ... + d_0 * B^0 + d_{-1} * B^{-1} + ... + d_{-m} * B^{-m}$

trong đó mỗi d_i là 1 ký số trong hệ thống B.

Trong thực tế lập trình bằng ngôn ngữ cấp cao, ta thường dùng hệ thống số thập phân để miêu tả dữ liệu số của chương trình (vì đã quen). Chỉ trong 1 số trường hợp đặc biệt, ta mới dùng hệ thống số thập lục phân (dạng ngắn của nhị phân) để miêu tả 1 vài giá trị nguyên, trong trường hợp này, qui luật biểu diễn của lượng nguyên Q trong hệ thống số B sẽ đơn giản là :

$d_n d_{n-1} ... d_1 d_0 \Leftrightarrow$

$Q = d_n * B^n + d_{n-1} * B^{n-1} + ... + d_1 * B^1 + d_0 * B^0$

trong đó mỗi d_i là 1 ký số trong hệ thống B.

Cơ bản về hệ thống số - Vài thí dụ

Thí dụ về biểu diễn các lượng trong các hệ thống số :

- lượng "mười bảy" được miêu tả là 17 trong hệ thập phân vì : $17 = 1*10^1+7*10^0$
- lượng "mười bảy" được miêu tả là 11 trong hệ thập lục phân vì : $11 = 1*16^1+1*16^0$
- lượng "mười bảy" được miêu tả là 10001 trong hệ nhị phân vì : $10001 = 1*2^4+0*2^3+0*2^2+0*2^1+1*2^0$

Trong môi trường sử dụng đồng thời nhiều hệ thống số, để tránh nhầm lẫn trong các biểu diễn của các lượng khác nhau, ta sẽ thêm ký tự nhận dạng hệ thống số được dùng trong biểu diễn liên quan. Thí dụ ta viết :

- 17_D để xác định sự biểu diễn trong hệ thống số thập phân.
- 11_H (hệ thống số thập lục phân.)
- 10001_B (hệ thống số nhị phân.)

2.3 Các phương pháp chuyển miêu tả số

Để chuyển 1 miêu tả số từ hệ thống số này sang hệ thống số khác, ta cần dùng 1 phương pháp chuyển thích hợp. Có 4 phương pháp sau tương ứng với từng yêu cầu chuyển tương ứng :

1. chuyển từ hệ thống số khác về thập phân.
2. chuyển từ nhị phân về thập lục phân (hay bát phân).
3. chuyển từ thập lục phân (hay bát phân) về nhị phân.
4. chuyển từ hệ thống số thập phân về hệ thống số khác.

Chuyển từ hệ thống khác về thập phân

Để chuyển 1 miêu tả số từ hệ thống số khác (nhị phân, thập lục phân hay bát phân) sang hệ thập phân, ta dùng công thức tính Q.

Thí dụ :

1. $1A2_H = 1*16^2+10*16^1+2*16^0 = 256+160+2 = 418_D$

2. $642_O = 6*8^2+4*8^1+2*8^0 = 384+32+2 = 418_D$

3. $110100010_B = 2^8 + 2^7+2^5+2^1 = 256+128+32+2 = 418_D$

Chuyển từ hệ thống nhị phân về thập lục phân

Lưu ý rằng có 1 mối quan hệ mật thiết giữa hệ nhị phân và thập lục phân (hay bát phân), đó là 4 ký số nhị phân tương đương với 1 ký số thập lục phân (hay 3 ký số nhị phân tương đương với 1 ký số bát phân) theo bảng tra sau :

Dec	Hex	Oct	Binary
0	0	00	0000
1	1	01	0001
2	2	02	0010
3	3	03	0011
4	4	04	0100
5	5	05	0101
6	6	06	0110
7	7	07	0111

Dec	Hex	Oct	Binary
8	8	10	1000
9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111

Chuyển từ hệ thống nhị phân về thập lục phân

Để đổi 1 số nhị phân về thập lục phân (hay bát phân), ta đi từ phải sang trái và chia thành từng nhóm 4 ký số nhị phân (hay 3 ký số nhị phân), sau đó đổi từng nhóm 4 ký số (hay 3 ký số) thành 1 ký số thập lục phân tương đương (hay 1 ký số bát phân tương đương).

Thí dụ :

1. $110100010_B = 0001.1010.0010 = 1A2_H$
2. $110100010_B = 110.100.010 = 642_O$

Chuyển từ hệ thống thập lục phân về nhị phân

Để đổi 1 số thập lục phân (hay bát phân) về nhị phân, ta đổi từng ký số thập lục phân (hay bát phân) thành từng nhóm 4 ký số nhị phân (hay 3 ký số nhị phân).

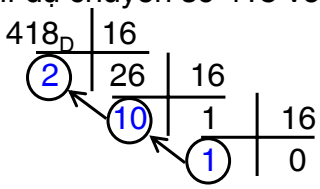
Thí dụ :

1. $1A2_H = 0001.1010.0010 = 110100010_B$
2. $642_O = 110.100.010 = 110100010_B$

Chuyển từ hệ thống thập phân về hệ thống khác

Để đổi 1 số thập phân về hệ thống số khác, ta hãy chia số cần đổi cho cơ số đích để có được thương và dư số, ta **lặp lại** hoạt động chia thương số cho cơ số đích để có được thương và dư số mới, cứ thế lặp mãi cho đến khi thương số = 0 thì dừng lại. Ghép các dư số theo chiều ngược chiều lặp để tạo ra kết quả (đó là sự miêu tả số tương đương nhưng ở hệ thống số khác).

Thí dụ chuyển số 418 về miêu tả tương ứng trong hệ thập lục :



Kết quả là $418_D = 1A2_H$

Chuyển từ hệ thống thập lục phân về bát phân

Để đổi 1 số thập lục phân về bát phân (hay ngược lại), ta nên chuyển tuần tự từ thập lục phân về nhị phân, rồi từ nhị phân về bát phân.

Cơ bản về hệ thống số - Các phép tính

Các phép tính cơ bản trong 1 hệ thống số là :

1. phép cộng (+).
2. phép trừ (-).
3. phép chia (/).
4. phép nhân (*).
5. phép dịch trái n ký số (<< n).
6. phép dịch phải n ký số (>> n).

Ngoài ra do đặc điểm của hệ nhị phân, hệ này còn cung cấp 1 số phép tính sau (các phép tính luận lý) :

1. phép OR bit (|).
2. phép AND bit (&).
3. phép XOR bit (^).
4. ....

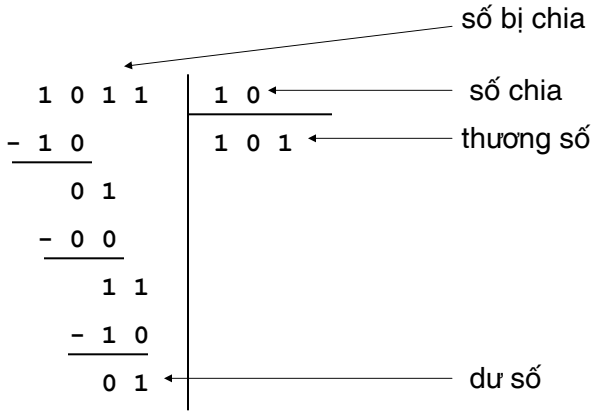
Thí dụ về phép cộng, trừ, nhân

Thí dụ về các phép tính cơ bản (các giá trị đều được biểu diễn bằng hệ nhị phân :

```
  0 1 1 0        1 0 0 1          1 0 0 1
+ 0 0 1 1      - 0 0 1 1        * 0 1 0 1
---------      ---------        ---------
  1 0 0 1        0 1 1 0          1 0 0 1
                                0 0 0 0
                              1 0 0 1
                              ---------
                              1 0 1 1 0 1
```

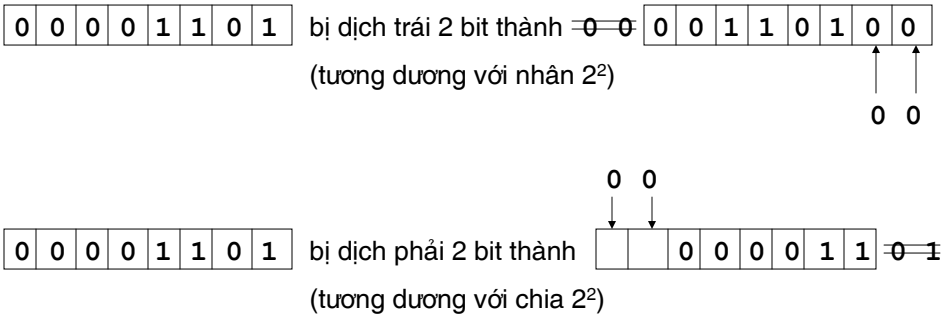
Thí dụ về phép chia

Thí dụ về các phép tính cơ bản (các giá trị đều được biểu diễn bằng hệ nhị phân) :



Thí dụ về phép dịch ký số

Thí dụ về các phép tính dịch ký số (các giá trị đều được biểu diễn bằng hệ nhị phân) :



Các phép tính của đại số Boole

Đại số Boole nghiên cứu các phép toán thực hiện trên các biến chỉ có 2 giá trị 0 và 1, tương ứng với hai thái cực luận lý "sai" và "đúng" (hay "không" và "có") của đời thường. Các phép toán này gồm :

x	y	**not** x	x **and** y	x **nand** y	x **or** y	x **nor** y	x **xor** y
0	0	1	0	1	0	1	0
0	1		0	1	1	0	1
1	0	0	0	1	1	0	1
1	1		1	0	1	0	0

Biểu thức Boole là 1 biểu thức toán hoc cấu thành từ các phép toán Boole trên các toán hạng là các biến chỉ chứa 2 trị 0 và 1.

Hàm Boole

Một hàm Boole theo n biến boole (hàm n ngôi) là 1 biểu thức boole cấu thành từ các phép toán Boole trên các biến boole.

Thay vì miêu tả hàm boole bằng biểu thức boole, ta có thể miêu tả hàm boole bằng bảng thực trị. Bảng thực trị của hàm boole n biến có 2^n hàng, mỗi hàng miêu tả 1 tổ hợp trị cụ thể của các biến và giá trị cụ thể của hàm tương ứng với tổ hợp trị này (xem slide ngay trước).

Như vậy 1 hàm boole n biến được miêu tả như 1 chuỗi 2^n bit $\Rightarrow$ có chính xác 2^{2^n} hàm boole n ngôi khác nhau. Cụ thể có :

$2^{2^1} = 4$ hàm boole 1 ngôi khac nhau

$2^{2^2} = 2^4 = 16$ hàm boole 2 ngôi khac nhau

$2^{2^3} = 2^8 = 256$ hàm boole 3 ngôi khac nhau