Chương 3: Quản lý bộ nhớ

Hệ điều hành

ThS. Đinh Xuân Trường

truongdx@ptit.edu.vn

Posts and Telecommunications Institute of Technology

Faculty of Information Technology 1

CNTT1

Học viện Công nghệ Bưu chính Viễn thông

August 15, 2022

Mục lục

Nội dung chương 3

1. Địa chỉ và các vấn đề liên quan
2. Một số cách tổ chức chương trình
3. Các yêu cầu quản lý bộ nhớ
4. Phân chương bộ nhớ
5. Phân trang bộ nhớ
6. Phân đoạn bộ nhớ
7. Bộ nhớ ảo

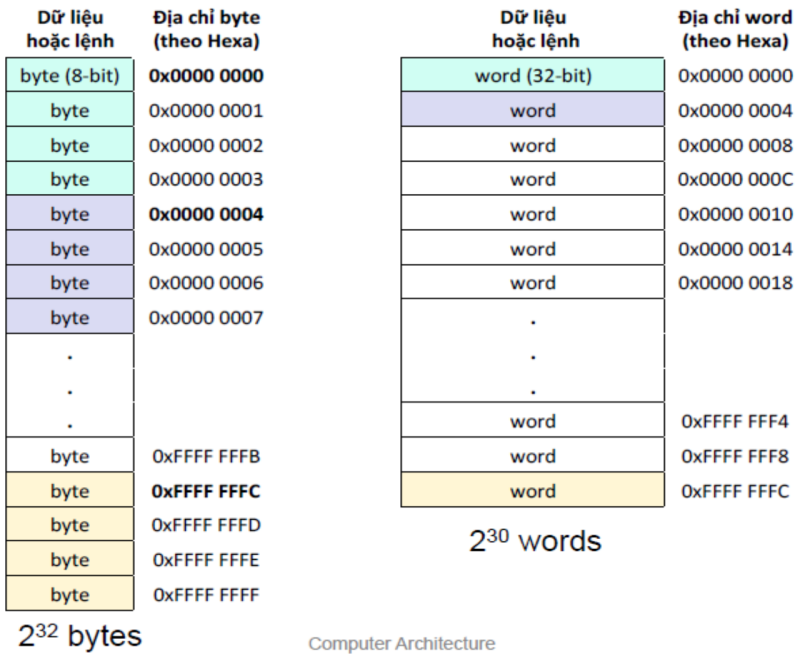
Quản lý bộ nhớ

Bộ nhớ *là tài nguyên quan trọng thứ hai sau CPU trong hệ thống máy tính, bao gồm các bytes và các từ nhớ được đánh địa chỉ.*

- Bộ nhớ là nơi chứa tiến trình và dữ liệu cho tiến trình
- Quản lý bộ nhớ ảnh hưởng tới tốc độ xử lý và khả năng tính toán của toàn bộ hệ thống
- Các công việc liên quan tới quản lý bộ nhớ bao gồm:
  - Quản lý bộ nhớ trống
  - Cấp phát và giải phóng bộ nhớ cho các tiến trình
  - Ngăn chặn truy cập trái phép tới các vùng nhớ
  - Ánh xạ địa chỉ giữa địa chỉ logic và địa chỉ vật lý

Địa chỉ và các vấn đề liên quan

- Bộ nhớ máy tính được đánh địa chỉ theo các bytes hoặc các từ nhớ



Dữ liệu hoặc lệnh	Địa chỉ byte (theo Hexa)
byte (8-bit)	**0x0000 0000**
byte	0x0000 0001
byte	0x0000 0002
byte	0x0000 0003
byte	**0x0000 0004**
byte	0x0000 0005
byte	0x0000 0006
byte	0x0000 0007
.	
.	
.	
byte	0xFFFF FFFB
byte	**0xFFFF FFFC**
byte	0xFFFF FFFD
byte	0xFFFF FFFE
byte	0xFFFF FFFF

2^{32} bytes

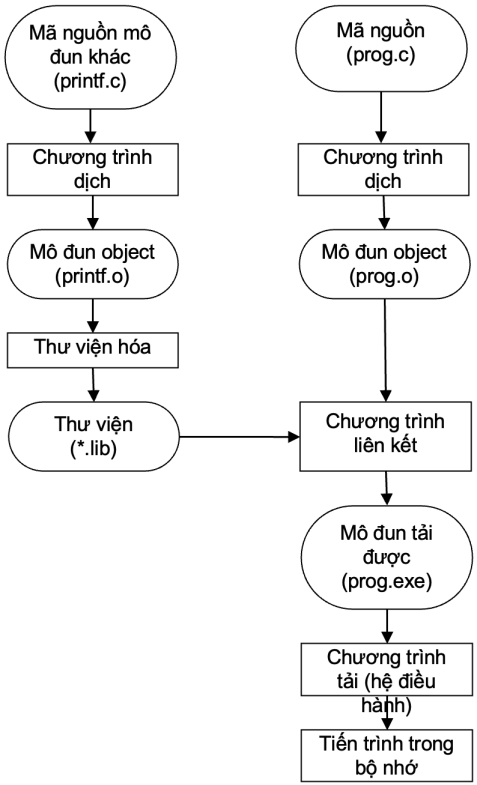
Dữ liệu hoặc lệnh	Địa chỉ word (theo Hexa)
word (32-bit)	0x0000 0000
word	0x0000 0004
word	0x0000 0008
word	0x0000 000C
word	0x0000 0010
word	0x0000 0014
word	0x0000 0018
.	
.	
.	
word	0xFFFF FFF4
word	0xFFFF FFF8
word	0xFFFF FFFC

2^{30} words

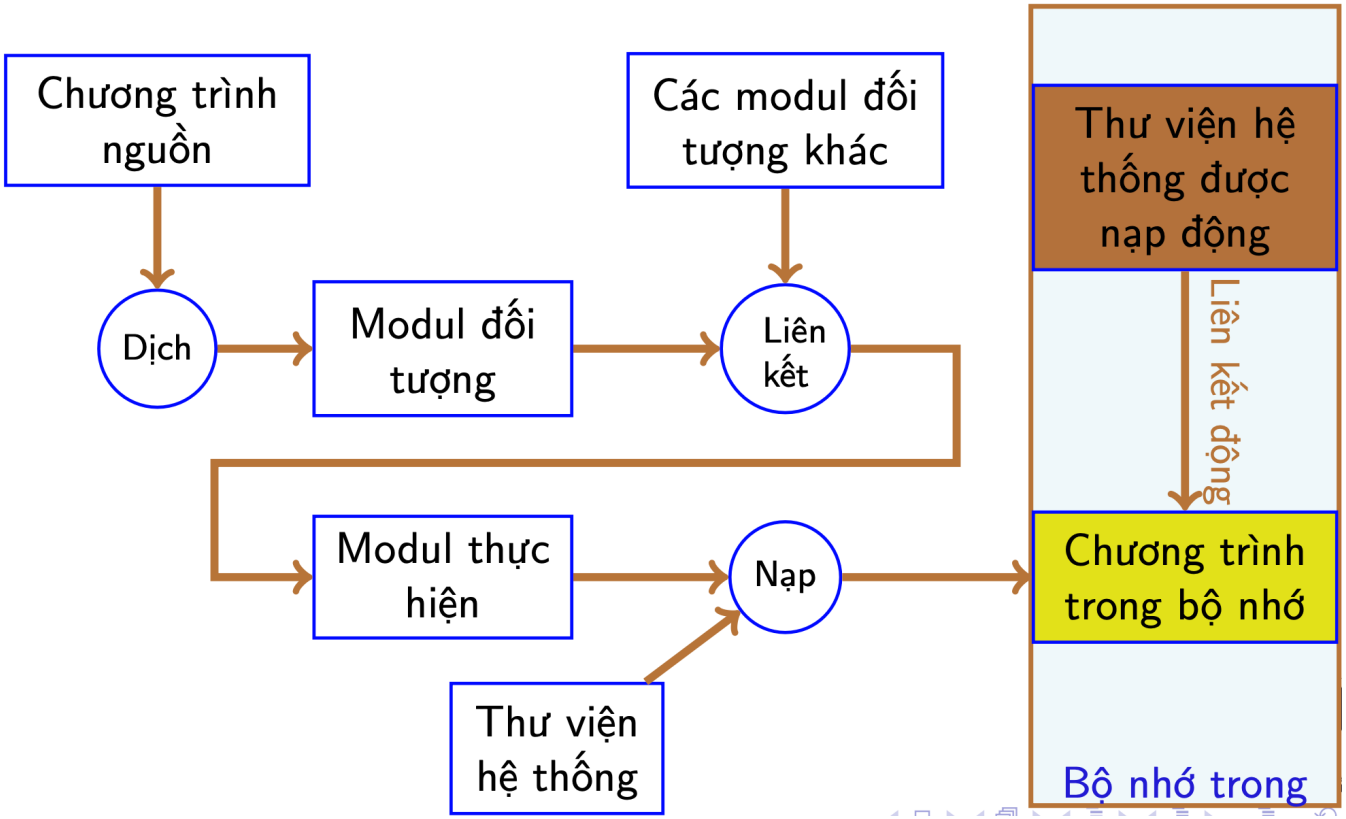
Computer Architecture

- Quá trình thực hiện tiến trình CPU đọc lần lượt các lệnh từ trong bộ nhớ và thực hiện các lệnh này
- Dữ liệu và lệnh đều được đánh địa
- CPU sẽ sử dụng địa chỉ để xác định lệnh và dữ liệu cụ thể

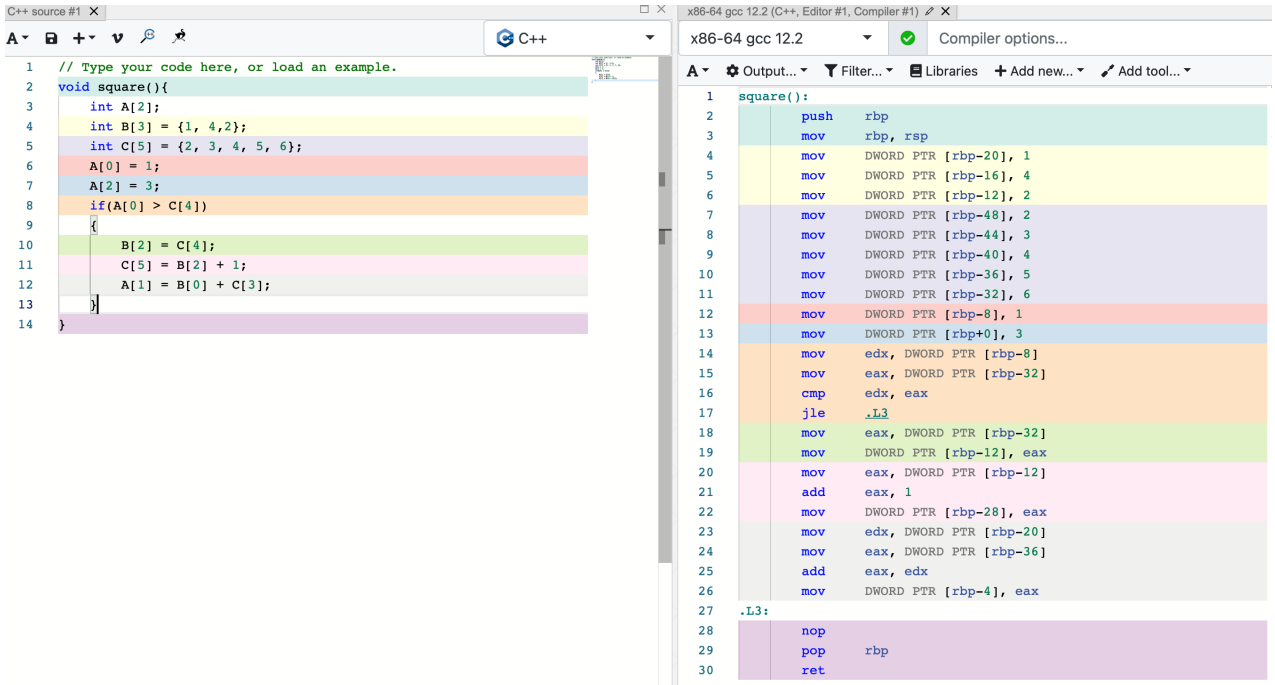
- Chương trình thường không được viết bằng ngôn ngữ máy
- Các chương trình phải trải qua một quá trình dịch và chương trình liên kết trước khi trở thành chương trình có thể tải vào và thực hiện



- Chương trình thường không được viết bằng ngôn ngữ máy
- Các chương trình phải trải qua một quá trình dịch và chương trình liên kết trước khi trở thành chương trình có thể tải vào và thực hiện

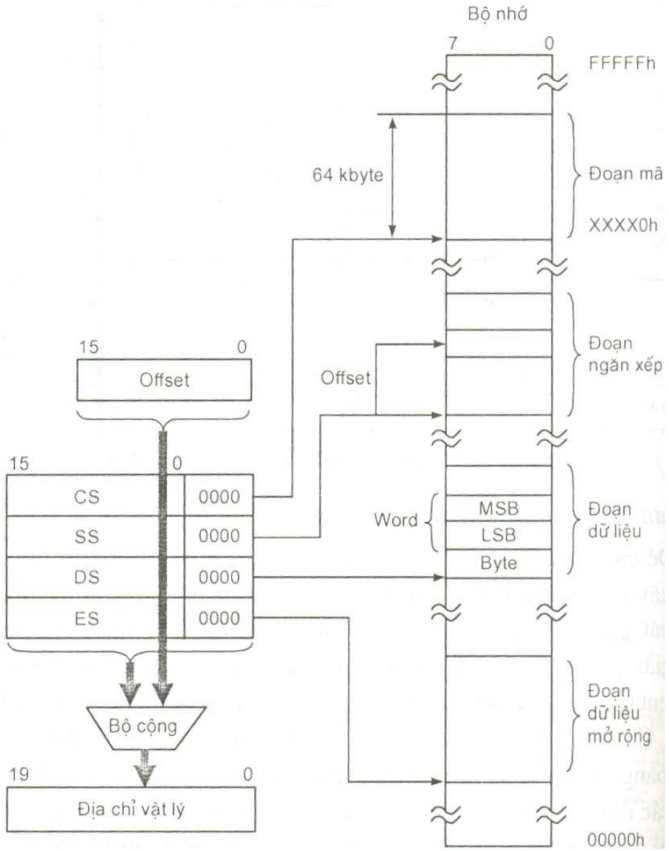


- Chương trình sử dụng địa chỉ dưới dạng tên (biến, hàm)
- Khi dịch, chương trình dịch ánh xạ các tên theo địa chỉ tương đối tính từ đầu file obj
- Chương trình liên kết ánh xạ địa chỉ thành địa chỉ tương đối tính từ đầu chương trình



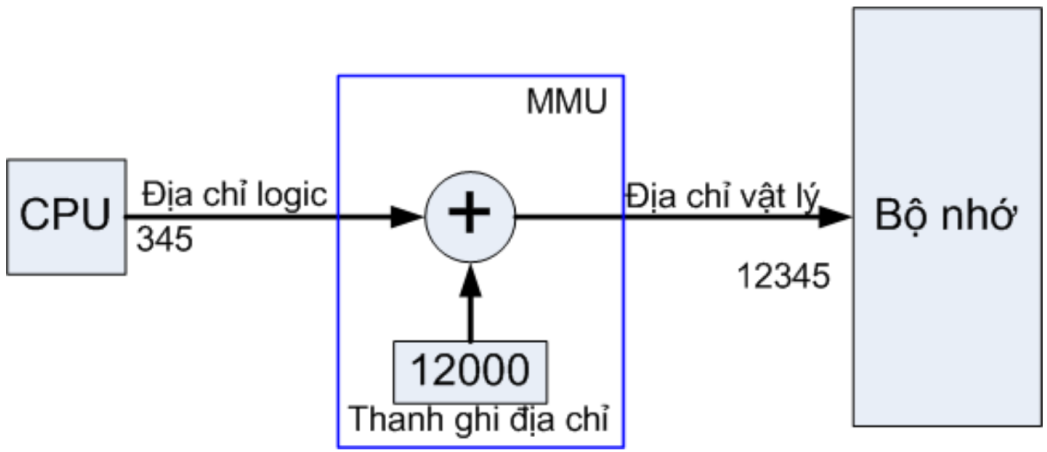
- Quản lý bộ nhớ:
  - **Cấp phát bộ nhớ** cho các tiến trình
  - **Tái định vị**, ánh xạ địa chỉ lệnh và dữ liệu từ các tiến trình vào trong bộ nhớ
  - **Bảo vệ tiến trình**: phần nào bộ nhớ cho tiến trình nào thì tiến trình khác không được sử dụng
  - **Chia sẻ:** các tiến trình có thể chia sẻ các không gian nhớ khi tương tác với nhau

VXL 8086 dùng 20 bit mã hoá địa chỉ 1MB từ 00000h - FFFFFh



Địa chỉ vật lý 20-bit của một ô nhớ = địa chỉ đoạn 16-bit được dịch trái 4 bít (nhân với 16) + địa chỉ lệnh 16-bit.

- Địa chỉ logic
  - Gán cho các lệnh và dữ liệu không phụ thuộc vào vị trí cụ thể tiến trình trong bộ nhớ. Được sinh ra trong tiến trình (CPU đưa ra)
  - CTƯD chỉ quan tâm tới địa chỉ logic còn địa chỉ vật lý là do HĐH
  - Khi thực hiện chương trình, CPU "nhìn thấy" và sử dụng địa chỉ lôgic này để trỏ đến các phần khác nhau của lệnh, dữ liệu
  - Để truy cập bộ nhớ, địa chỉ lôgic cần được ánh xạ thành địa chỉ vật lý và được khối quản lý bộ nhớ (MMU) chuyển sang địa chỉ vật lý khi truy nhập tới đối tượng trong chương trình



Ví dụ: JMP 010A: Nhảy tới ô nhớ có vị trí 010Ah tại cùng đoạn mã lệnh (CS - Code Segment)
Nếu CS = 1555h, sẽ đi tới vị trí: 1555h*10h + 010Ah = 1560Ah

- Địa chỉ vật lý:
  - Là địa chỉ chính xác trong bộ nhớ máy tính
  - Các mạch nhớ sử dụng để truy nhập tới chương trình và dữ liệu
- Địa chỉ logic được chuyển thành địa chỉ vật lý nhờ khối ánh xạ địa chỉ. (MMU=Memory Mapping Unit)

Một số cách tổ chức chương trình

Các vấn đề quan trọng trong tổ chức chương trình và quản lý bộ nhớ:

- Giảm không gian chương trình chiếm trên đĩa, trên bộ nhớ
- Sử dụng không gian nhớ hiệu quả
- Các kỹ thuật tổ chức và sử dụng bộ nhớ hiệu quả:
  - Tải trong quá trình chạy
  - Liên kết động và thư viện dùng chung

Toàn bộ chương trình thông thường được tải vào bộ nhớ để thực hiện.
Đối với chương trình lớn, trong một phiên làm việc, một số phần của chương trình có thể không dùng tới. Các hàm nãy sẽ chiếm vô ích trong bộ nhớ, đồng thời tăng thời gian tải chương trình lúc đầu.
Giải pháp:

- Hàm chưa bị gọi thì chưa tải vào bộ nhớ
- Chương trình chính được load vào bộ nhớ và chạy
- Khi có lời gọi hàm:
  - Chương trình sẽ kiểm tra hàm đó được tải vào chưa.
  - Nếu chưa, chương trình sẽ tiến hành tải sau đó ánh xạ địa chỉ hàm vào không gian chung của chương trình và thay đổi bảng địa chỉ để ghi lại các ánh xạ đó
- Lập trình viên đảm nhiệm kiểm tra và tải hàm, HĐH chỉ cung cấp các hàm thư viện cho module

Trong quá trình liên kết tĩnh các hàm và thư viện được liên kết luôn vào mã chương trình
Kích thước chương trình = Kích thước chương trình vừa được dịch + Kích thước các hàm thư viện

- Các hàm sẽ có mặt lặp đi lặp lại trong các chương trình
- Lãng phí không gian cả trên đĩa và bộ nhớ trong

Giải quyết sử dụng **kỹ thuật liên kết động** : *Trong giai đoạn liên kết, không kết nối các hàm thư viện vào chương trình mà chỉ chèn các thông tin về hàm thư viện đó (stub).*

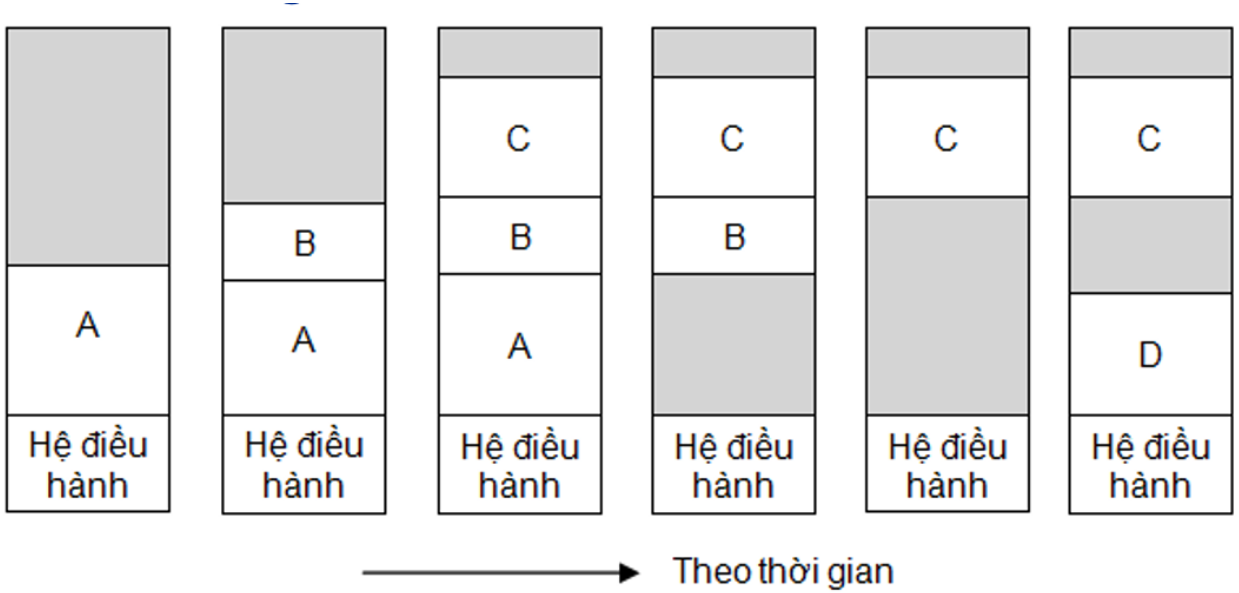
- Các module thư viện được liên kết trong quá trình thực hiện:
  - Không giữ bản sao các module thư viện mà tiến trình giữ đoạn mã nhỏ chứa thông tin về modul thư viện
  - Khi đoạn mã nhỏ được gọi, nó kiểm tra module tương ứng đã có trong bộ nhớ chưa. Nếu chưa, thì tải phần còn lại và dùng.
  - Lần tiếp theo cần sử dụng, modul thư viện sẽ được chạy trực tiếp
  - Mỗi module thư viện chỉ có 1 bản sao duy nhất chứa trong MEM
  - Cần hỗ trợ từ HĐH

Các yêu cầu quản lý bộ nhớ

Các yêu cầu liên quan đến việc quản lý bộ nhớ bao gồm:

- **Cấp phát lại** cho các tiến trình
- **Bảo vệ**: phần nào bộ nhớ cho tiến trình nào thì tiến trình khác không được sử dụng
- **Chia sẻ:** các tiến trình có thể chia sẻ các không gian nhớ khi tương tác với nhau
- **Cấu trúc:** cấu trúc logic và cấu trúc vật lý

- Cần có khả năng tráo đổi các tiến trình vào và ra ngoài MEM để tối đa sử dụng vi xử lý
- Không thể yêu cầu tiến trình được chuyển lại vào MEM thì phải vào đúng chỗ nó đã dùng trước khi bị chuyển ra



- Mỗi tiến trình phải được bảo vệ khỏi các tham chiếu không mong muốn từ các tiến trình khác vào vùng nhớ dành cho mình
- Khi 1 vùng nhớ đã dùng cho tiến trình này thì nó không cho phép tiến trình khác tham chiếu vào vùng nhớ đang dùng đó.
- Mọi tham chiếu bộ nhớ của 1 tiến trình phải được kiểm tra lúc chạy
- HĐH không đoán trước được mọi tham chiếu MEM nên phần cứng vi xử lý đảm nhiệm

- Nhiều tiến trình cần và được phép truy cập vào cùng 1 vùng nhớ
- Các tiến trình đang cộng tác cần chia sẻ truy nhập tới 1 cấu trúc dữ liệu
- Cho phép truy cập tới các vùng chia sẻ

Cấu trúc logic:

- MEM được cấu trúc 1 cách tuyến tính gồm các byte, còn chương trình được tổ chức thành các modul
- Phải đáp ứng để:
  - Các module có thể được viết và thông dịch 1 cách độc lập
  - Mức độ bảo vệ có thể khác nhau
  - Module có thể được chia sẻ giữa các tiến trình

Cấu trúc vật lý:

- Cấu trúc vật lý 2 mức:
  - Bộ nhớ chính: nhanh; chi phí cao, dung lượng ít
  - Bộ nhớ phụ: dung lượng lớn, cho phép lưu chương trình và dữ liệu trong thời gian dài
- Hệ thống có trách nhiệm chuyển đổi thông tin giữa 2 mức

Phân chương bộ nhớ

- Để thực hiện tiến trình, HĐH cần cấp phát cho tiến trình không gian nhớ cần thiết.
- Việc cấp phát và quản lý vùng nhớ là chức năng quan trọng của HĐH
- Một kỹ thuật cấp phát đơn giản nhất là mỗi tiến trình được cấp một vùng bộ nhớ liên tục
- HĐH tiến hành chia bộ nhớ thành các phần liên tục là chương (partition), mỗi tiến trình sẽ được cung cấp một chương để chứa lệnh và dữ liệu của mình.
- Quá trình phân chia bộ nhớ thành chương như vậy gọi là *phân chương bộ nhớ*.
- Tùy thuộc việc lựa chọn vị trí và kích thước của chương, có thể phân biệt phân chương cố định và phân chương động

Phân chương bộ nhớ (cont.)

Các chiến lược quản lý bộ nhớ:

- Phân chương cố định
- Phân chương động
- Phân trang
- Phân đoạn
- Kết hợp phân đoạn - phân trang

Nguyên tắc

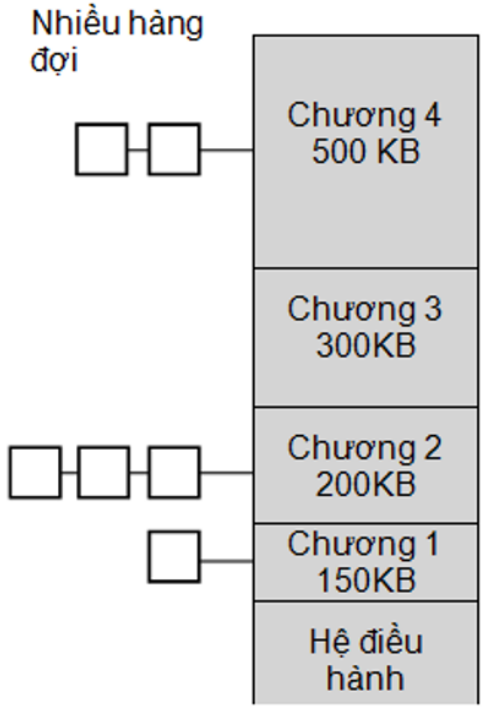
- Bộ nhớ được chia thành n phần
  - Mỗi phần được gọi là một chương (partition)
  - Mỗi chương ở một vị trí cố định
  - Chương được sử dụng như một vùng nhớ độc lập
    - Mỗi chương chứa được đúng một tiến trình
    - Khi được tải vào, tiến trình được cấp phát một chương. Sau khi tiến trình kết thúc, HĐH giải phóng chương và chương có thể được cấp phát cho tiến trình mới.
  - Chương có thể có kích thước bằng nhau hoặc khác nhau

Ví dụ: Xét hệ thống dưới đây bộ nhớ được tổ chức theo phân chương cố định, tính thời gian HĐH thực hiện xong các tiến trình sau:

Process	Size	time
P_1	120	20
P_2	80	15
P_3	70	5
P_4	50	5
P_5	140	12
Hàng đợi		

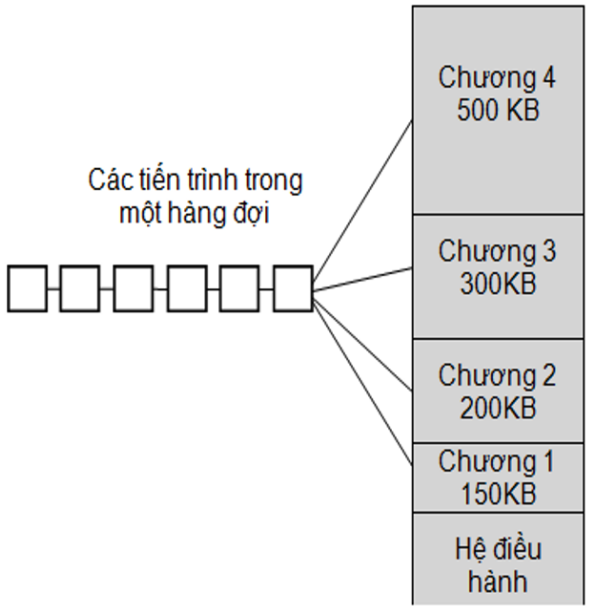
- Kích thước các chương bằng nhau:
  - Ưu điểm: Đơn giản
  - Nhược điểm: Kích thước chương trình > kích thước chương dẫn đến không thể cấp phát
  - Gây phân mảnh trong
- Kích thước các chương khác nhau:
  - Có hai cách lựa chọn chương nhớ để cấp cho tiến trình đang chờ đợi bằng cách chọn chương có kích thước nhỏ nhất
    - Mỗi chương có một hàng đợi riêng
    - Một hàng đợi chung cho tất cả các chương

Mỗi chương có một hàng đợi riêng: *tiến trình có kích thước phù hợp với chương nào sẽ nằm trong hàng đợi của chương đó*



- Ưu điểm: Tiết kiệm bộ nhớ, giảm phân mảnh trong
- Nhược điểm: Hệ thống không tối ưu, có thời điểm hàng đợi chương lớn thì rỗng, hàng đợi chương nhỏ hơn chứa nhiều tiến trình.

Một hàng đợi chung cho tất cả các chương: *Mỗi khi có một chương trống tiến trình nằm gần đầu hàng đợi nhất và có kích thước phù hợp với chương nhất sẽ được tải và thực hiện*



- Ưu điểm: Tiết kiệm bộ nhớ, giảm phân mảnh trong và tối ưu hệ thống

Nhận xét về phân chương cố định

- Ưu điểm
  - Đơn giản và ít xử lý
  - Giảm thời gian tìm kiếm
- Nhược điểm
  - Hệ số song song không thể vượt quá số lượng chương của bộ nhớ
  - Bị phân đoạn bộ nhớ
    - Kích thước chương trình lớn hơn kích thước chương lớn nhất
    - Tổng bộ nhớ tự do còn lớn, nhưng không dùng để nạp các chương trình khác

Tổng kết

Chương 3 Quản lý bộ nhớ

- Địa chỉ và các vấn đề liên quan
- Một số cách tổ chức chương trình
- Các yêu cầu quản lý bộ nhớ

Chương 3 Quản lý bộ nhớ

- Phân chương bộ nhớ
- Phân trang bộ nhớ
- Phân đoạn bộ nhớ
- Bộ nhớ ảo