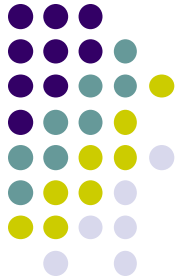
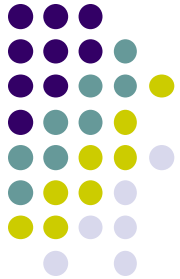
Kiến trúc hệ thống nhớ



4.1. Công nghệ hệ thống nhớ.

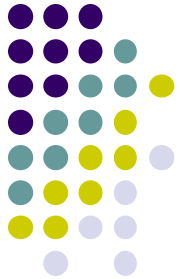
4.2. Hệ thống nhớ chính.

Các đặc trưng của hệ thống nhớ



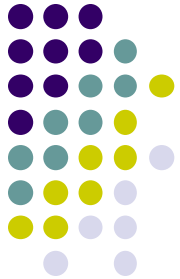
- Vị trí
  - Bên trong CPU:
    - Tập thanh ghi
  - Bộ nhớ trong:
    - Bộ nhớ chính
    - Bộ nhớ cache
  - Bộ nhớ ngoài: các thiết bị nhớ
- Dung lượng
  - Độ dài từ nhớ (tính bằng bit)
  - Số lượng từ nhớ

Các đặc trưng của hệ thống nhớ (tiếp)



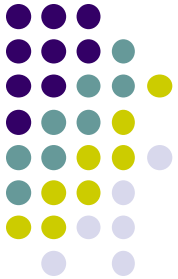
- Đơn vị truyền
  - Từ nhớ
  - Khối nhớ
- Phương pháp truy nhập
  - Truy nhập tuần tự (băng từ)
  - Truy nhập trực tiếp (các loại đĩa)
  - Truy nhập ngẫu nhiên (bộ nhớ bán dẫn)
  - Truy nhập liên kết (cache)

Các đặc trưng của hệ thống nhớ (tiếp)



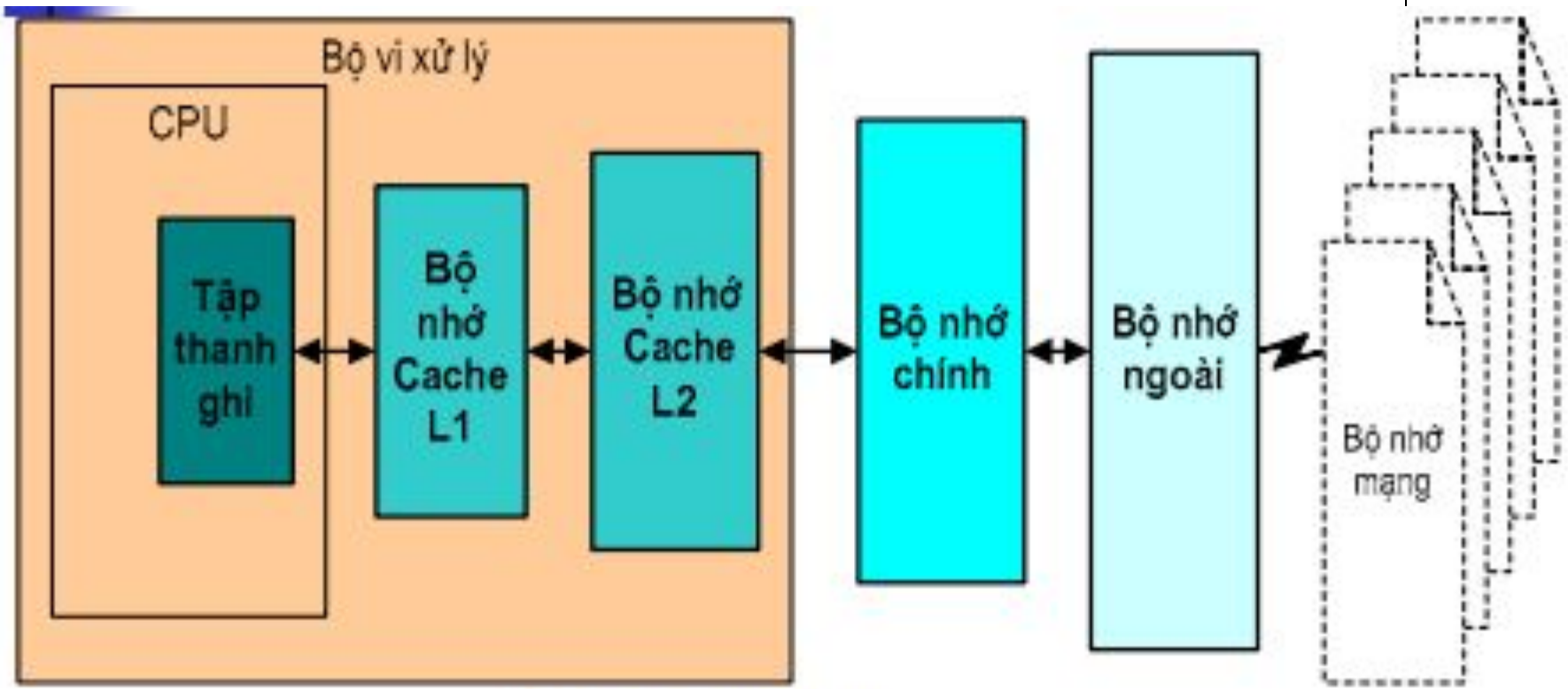
- Hiệu năng (performance)
  - Thời gian truy nhập
  - Chu kỳ nhớ
  - Tốc độ truyền
- Kiểu vật lý
  - Bộ nhớ bán dẫn
  - Bộ nhớ từ
  - Bộ nhớ quang

Các đặc trưng của hệ thống nhớ (tiếp)



- Các đặc tính vật lý
  - Khả biến / Không khả biến (volatile / nonvolatile)
  - Xoá được / không xoá được

Phân cấp hệ thống nhớ



Từ trái sang phải:

- dung lượng tăng dần
- tốc độ giảm dần
- giá thành/1bit giảm dần

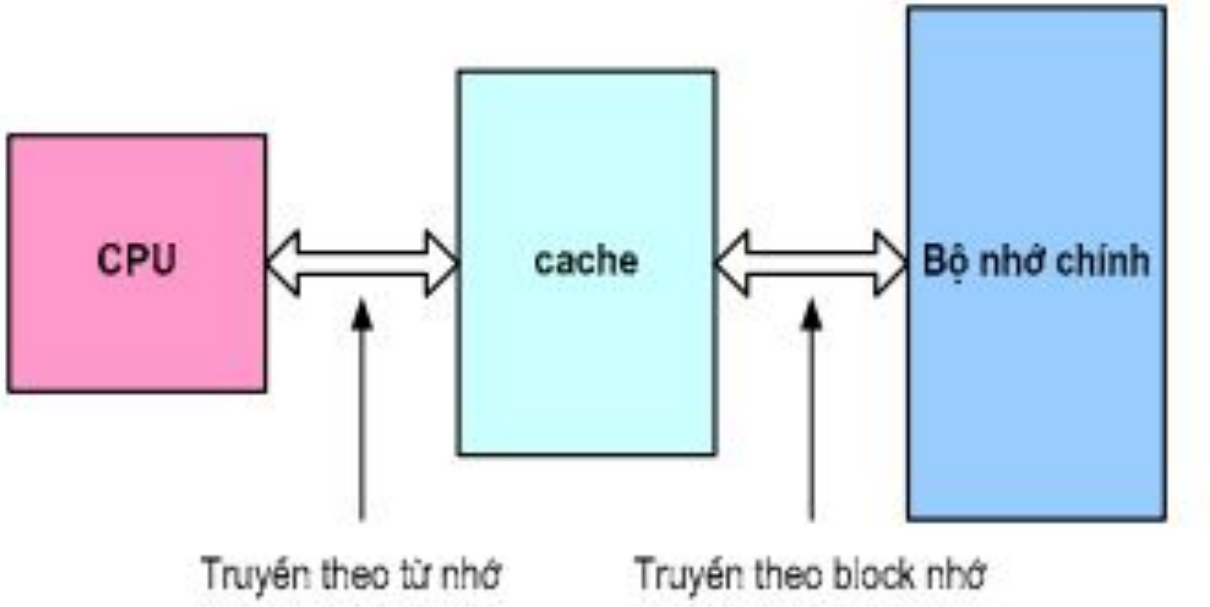
Bộ nhớ đệm (cache)

Nguyên tắc chung của cache

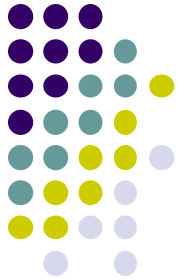
- Nguyên lý cục bộ hoá tham chiếu bộ nhớ:
  - Thời gian: Một lệnh hoặc dữ liệu vừa được truy nhập thì thường sẽ được truy nhập ngay sau đó.
  - Không gian: Một lệnh hoặc một dữ liệu vừa được truy nhập thì thường những lệnh hoặc dữ liệu lân cận sẽ được truy nhập ngay sau đó.
- Ví dụ:
  - Cấu trúc chương trình tuần tự
  - Vòng lặp có thân nhỏ
  - Cấu trúc dữ liệu mảng

Nguyên tắc chung của cache (tiếp)

- Cache có tốc độ nhanh hơn bộ nhớ chính.
- Cache được đặt giữa CPU và bộ nhớ chính nhằm tăng tốc độ CPU truy cập bộ nhớ.
- Cache có thể được đặt trên chip CPU.

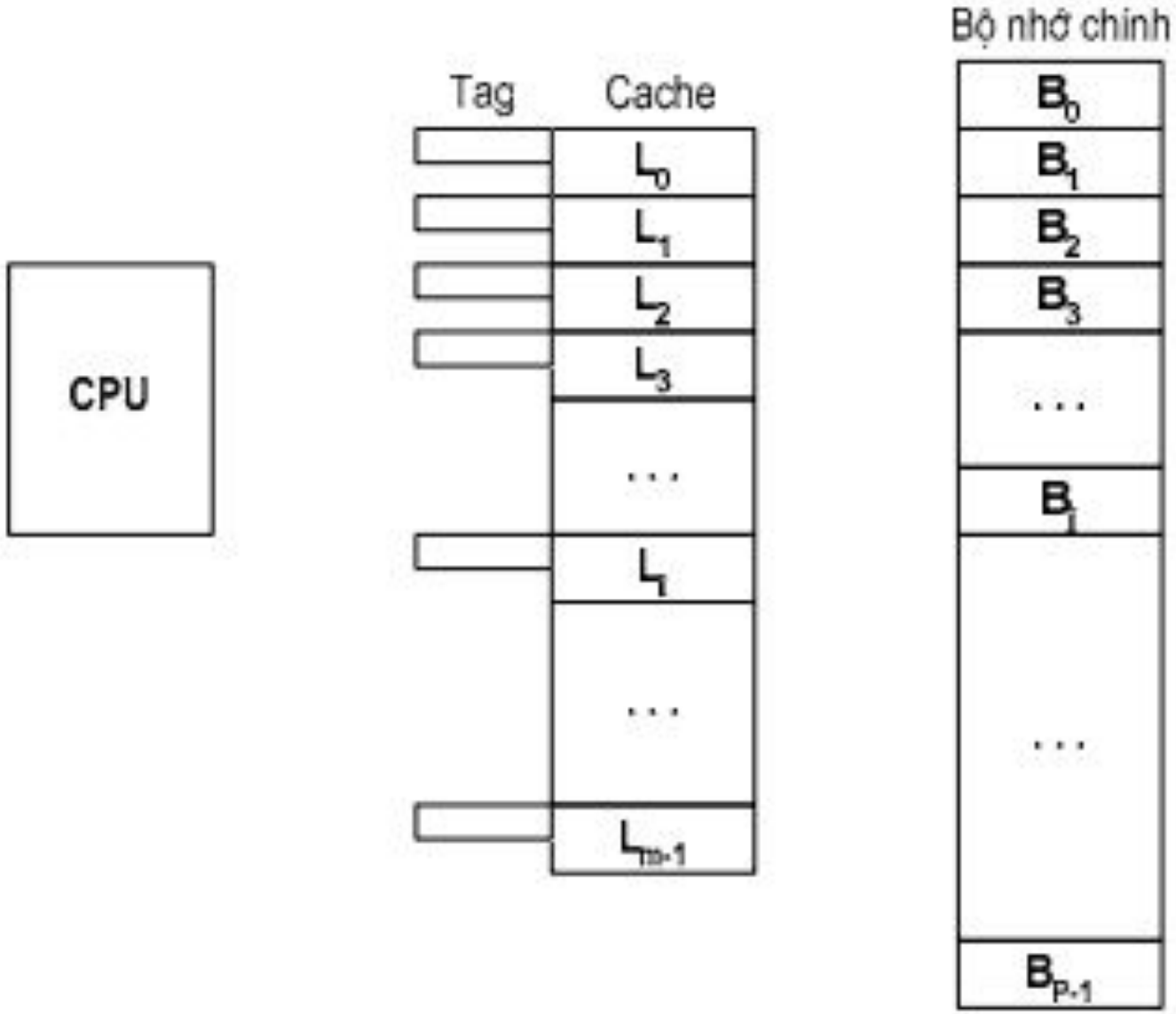


Ví dụ hoạt động của cache

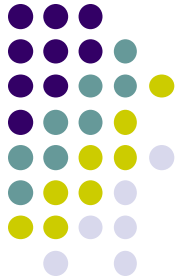


- CPU yêu cầu nội dung của ngăn nhớ
- CPU kiểm tra trên cache với dữ liệu này
- Nếu có, CPU nhận dữ liệu từ cache (nhanh) – Hit cache
- Nếu không có, đọc Block nhớ chứa dữ liệu từ bộ nhớ chính vào cache – Miss cache
- Tiếp đó chuyển dữ liệu từ cache vào CPU.

Tổ chức bộ nhớ cache và bộ nhớ chính

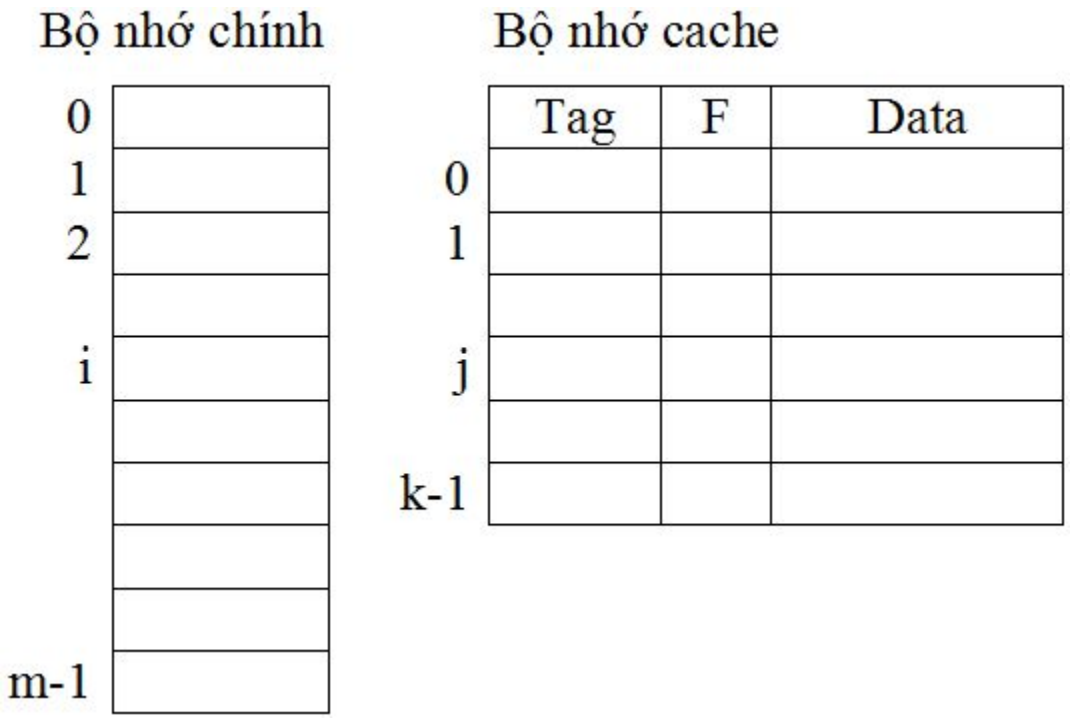


Tổ chức bộ nhớ cache và bộ nhớ chính (tiếp)



- Bộ nhớ chính có 2^N byte nhớ
- Bộ nhớ chính và cache được chia thành các khối có kích thước bằng nhau
  - Bộ nhớ chính: $B_0, B_1, B_2, \ldots, B_{p-1}$ (p Blocks)
  - Bộ nhớ cache: $L_0, L_1, L_2, \ldots, L_{m-1}$ (m Lines)
  - m<<p
- Kích thước của Block = 8,16,32,64,128 byte

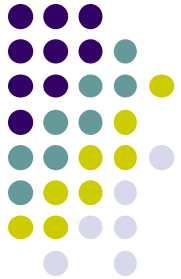
Tổ chức bộ nhớ cache và bộ nhớ chính (tiếp)



- Một số **Block** của bộ nhớ chính được nạp vào các **Line** của **Cache**.
- Nội dung **Tag** (thẻ nhớ) cho biết **Block** nào của bộ nhớ chính hiện đang được chứa ở **Line** đó.
- Bit F báo hiệu nội dung của **Line** có thay đổi hay không. F = 0 không đổi; F = 1 có thay đổi.

- Khi CPU truy nhập (đọc/ghi) một từ nhớ có 2 khả năng xảy ra:
  - Từ nhớ có trong cache (hit cache)
  - Từ nhớ không có trong cache (miss cache)

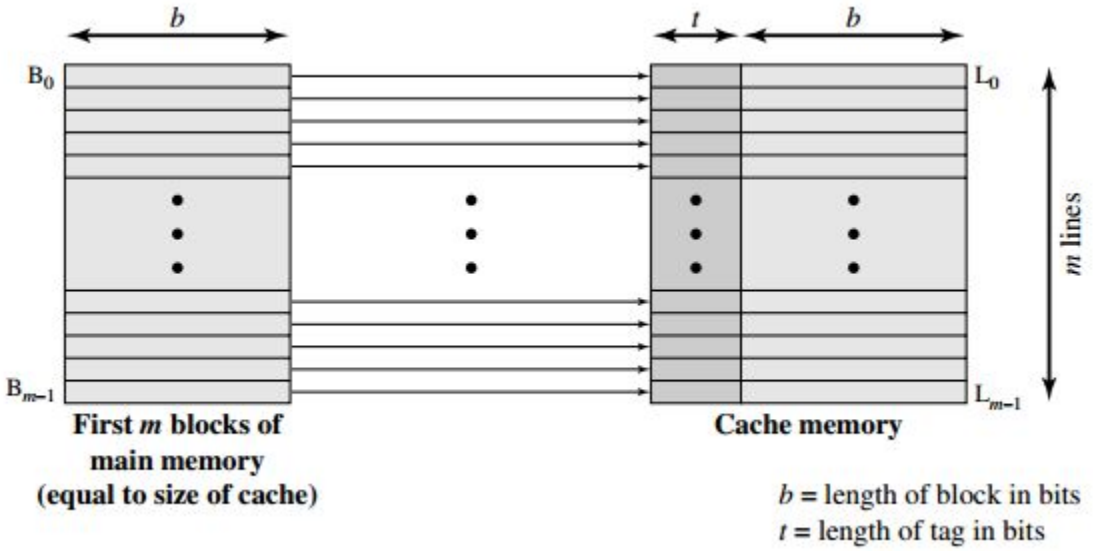
Các phương pháp ánh xạ



- Ánh xạ trực tiếp (direct mapping)
- Ánh xạ liên kết toàn phần (Fully associative mapping)
- **Ánh xạ liên kết tập (Set associative mapping)**

Ánh xạ trực tiếp

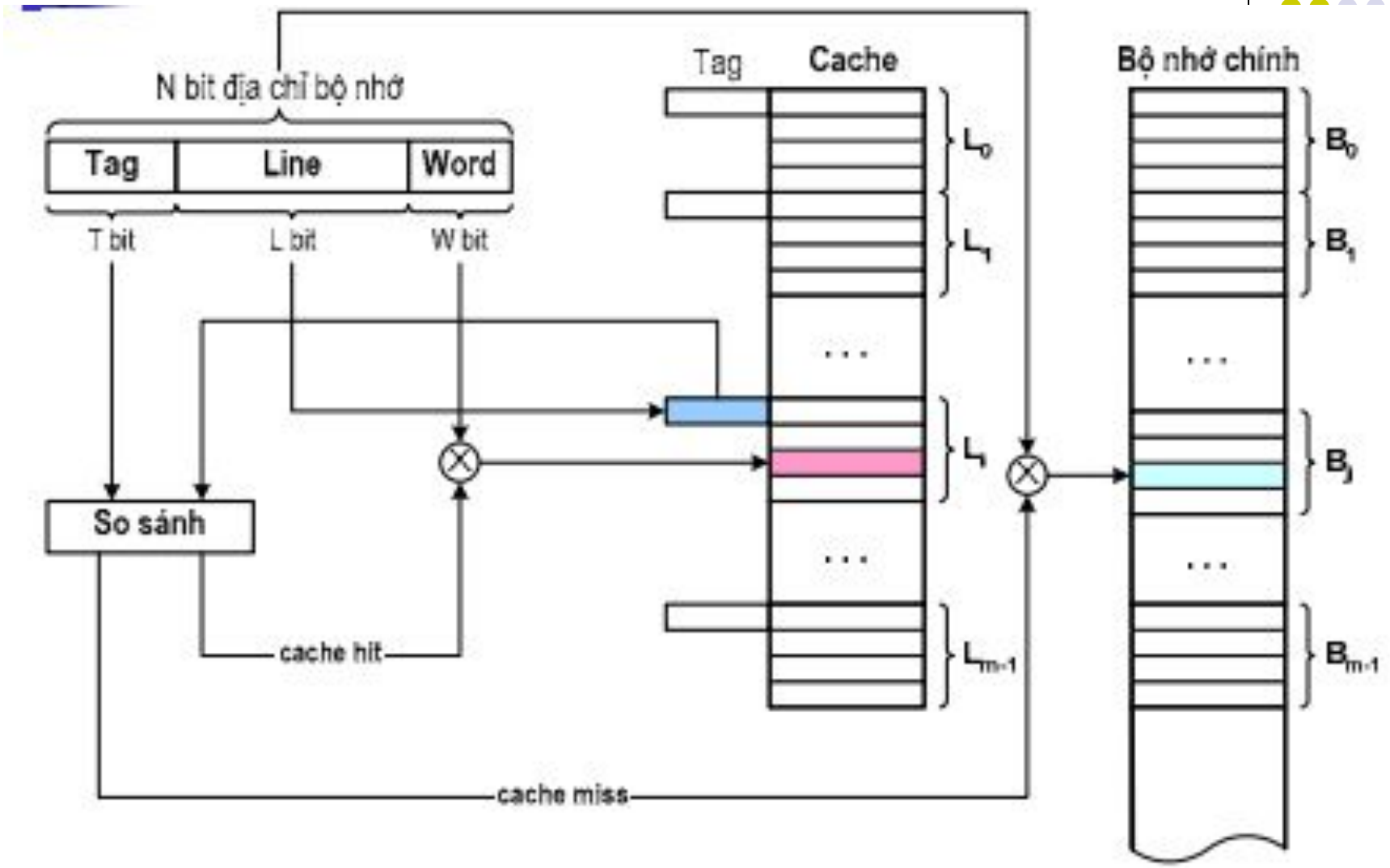
- Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một Line của cache:
  - $B_0 \rightarrow L_0$
  - $B_1 \rightarrow L_1$
  - ....
  - $B_{m-1} \rightarrow L_{m-1}$
  - $B_m \rightarrow L_0$
  - $B_{m+1} \rightarrow L_1$
  - ....



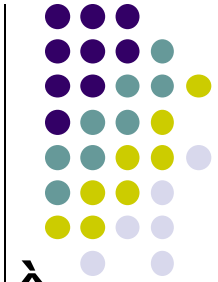
- Tổng quát
  - **B_j chỉ có thể nạp vào $L_{j \bmod m}$**
  - m là số *Line* của *cache*.
  - Mod là phép chia lấy phần dư.

Ví dụ: Cache có 16 lines. Hỏi block 100 nạp vào line nào của cache?
100 mod 16 = 4

Minh họa ánh xạ trực tiếp

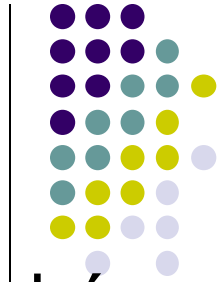


Đặc điểm của ánh xạ trực tiếp



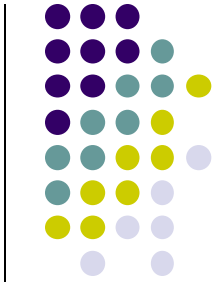
- Mỗi một địa chỉ N bit của bộ nhớ chính gồm ba trường:
  - Trường Word (n_1) gồm W bit xác định một từ nhớ trong *Block* hay *Line:*
    - 2^W = kích thước của *Block* hay *Line*
  - Trường *Line* (n_2) gồm L bit xác định một trong số các *Line* trong *cache:*
    - 2^L = số *Line* trong *cache* = m dung lượng cache = 2^{L+W}
  - Trường *Tag* (n_3) gồm T bit xác định block nhớ cần truy cập dựa vào số block: 2^T = số block.
    - N = T+L+W
- Bộ so sánh đơn giản
- Xác suất *cache hit* thấp

Ví dụ minh họa



- CPU có 24 bit địa chỉ (2^{24} byte = 16MB), bộ nhớ chính 256 KB chia làm 512 block nhớ, bộ nhớ cache có dung lượng 8 KB. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là **308842h**. Hãy trình bày chi tiết phương pháp đọc cache theo kỹ thuật ánh xạ **trực tiếp** cho trường hợp phát ra địa chỉ trên và cho biết địa chỉ ô nhớ cần truy cập trong bộ nhớ chính.
- Đỉa chỉ có hợp lệ không? Xác định chiều dài địa chỉ hợp lệ là bảo nhiêu bit? Block có nạp đúng line của cache không? T, L, W (n_3,n_2,n_1).

Giải ví dụ minh họa



- Bước 1: Xác định dung lượng của 1 block (line) nhớ.

$$C_{block} = \frac{C_{memory}}{n_{block}} = \frac{256KB}{512} = \frac{2^8 \times 2^{10}}{2^9} = 2^9 (byte)$$

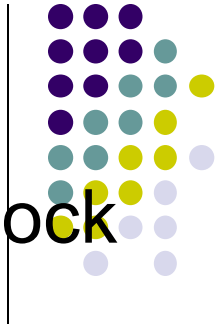
→ **W (n_1) = 9 bits**

- Bước 2: Xác định số line của cache

$$n_{line} = \frac{C_{cache}}{C_{block}} = \frac{8KB}{0.5KB} = 16 = 2^4 (lines)$$

→ **L (n_2) = 4 bits**. Dùng 4 bits để đánh số hiệu cho line (0 – 15)

Giải ví dụ minh họa (tiếp)



- Bước 3: Xác định số bits để đánh số hiệu block trong bộ nhớ.

  Số block = 512 = 2^9

  → **T (n_3)= 9 (bits)**

- Bước 4: Xác định số bits địa chỉ hợp lệ

  N = W + L + T = 9 + 4 + 9 = 22 (bits)

- Bước 5: Đọc bộ nhớ địa chỉ **308842**h
  - Xác định tính hợp lệ của địa chỉ

    308842h = **11000010001000010000010b**

    → **Địa chỉ hợp lệ vì số bits là 22 bits.**
  - Xác định số line trong cache cần truy cập: 0100b = 4
    → Truy cập line **4** trong cache.

Bước 5 (tiếp)

- Xác định số hiệu block của bộ nhớ cần truy cập.

  BI (Block index) = $110000100_2 = 2^8 + 2^7 + 2^2$
  $= 256 + 128 + 4 = 388.$

  → block nhớ cần truy cập là 388.

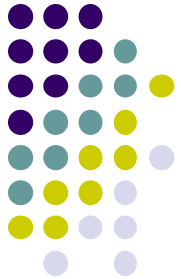
- Xác định block có nạp đúng line của cache không.

  Số hiệu line = số hiệu block mod số line của cache

  → số hiệu line = 388 mod 16 = **4** → block nạp đúng → **địa chỉ hợp lệ**

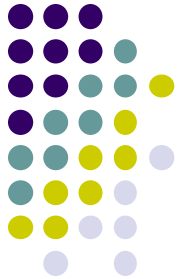
- CPU phát địa chỉ của ô nhớ cần truy cập ra bus địa chỉ.
- Bước 6. Bộ điều khiển cache (MMU – Memory Management Unit) sẽ truy cập vào line 4 của cache và đọc trường Tag của line này, sau đó đem so sánh với số hiệu block cần truy cập

Các khả năng xảy ra



- Khả năng 1: Miss cache → **Tag** ≠ 388.
  - MMU phải tiến hành nạp cache: trước khi nạp cache, nếu line 4 có F = 1, tức là nội dung có sự thay đổi, MMU phải ghi nội dung của nó ra block nhớ có số hiệu bằng giá trị trên **tag** của line, sửa F = 0 (còn trường hợp F = 0 thì thao tác này không cần thiết).
  - MMU ra bộ nhớ đọc block nhớ 388, nạp vào line 4, sửa tag của line 4 thành 388. Cuối cùng CPU tiến hành đọc byte trong line.
- Khả năng 2: Hit cache → **Tag** = 388 → CPU đọc byte trong line 4 có địa chỉ tương đối là 9 bits thấp của địa chỉ W = 001000010b = $2^6 + 2^1$ = 66.

Xác đinh địa chỉ vật lý của ô nhớ trong bộ nhớ chính



- Địa chỉ vật lý của ô nhớ trong bộ nhớ chính được tính bằng cách ghép địa chỉ của block và địa chỉ của từ nhớ lại.

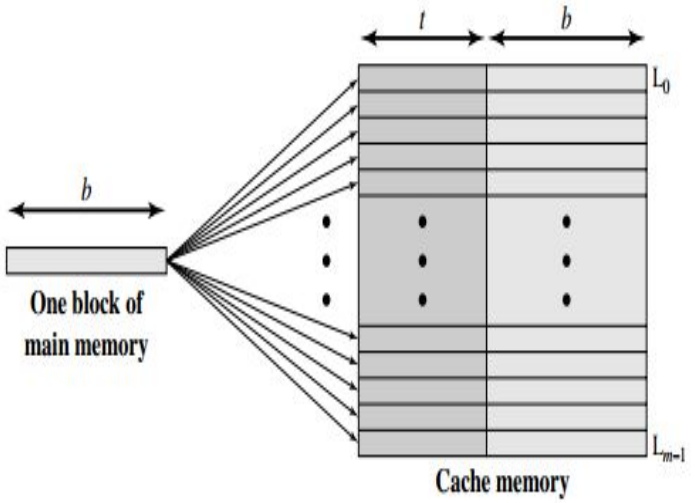
  → Địa chỉ vật lý = TW ($n_3 n_1$)

  → Địa chỉ vật lý = 11/0000/1000/0100/0010b

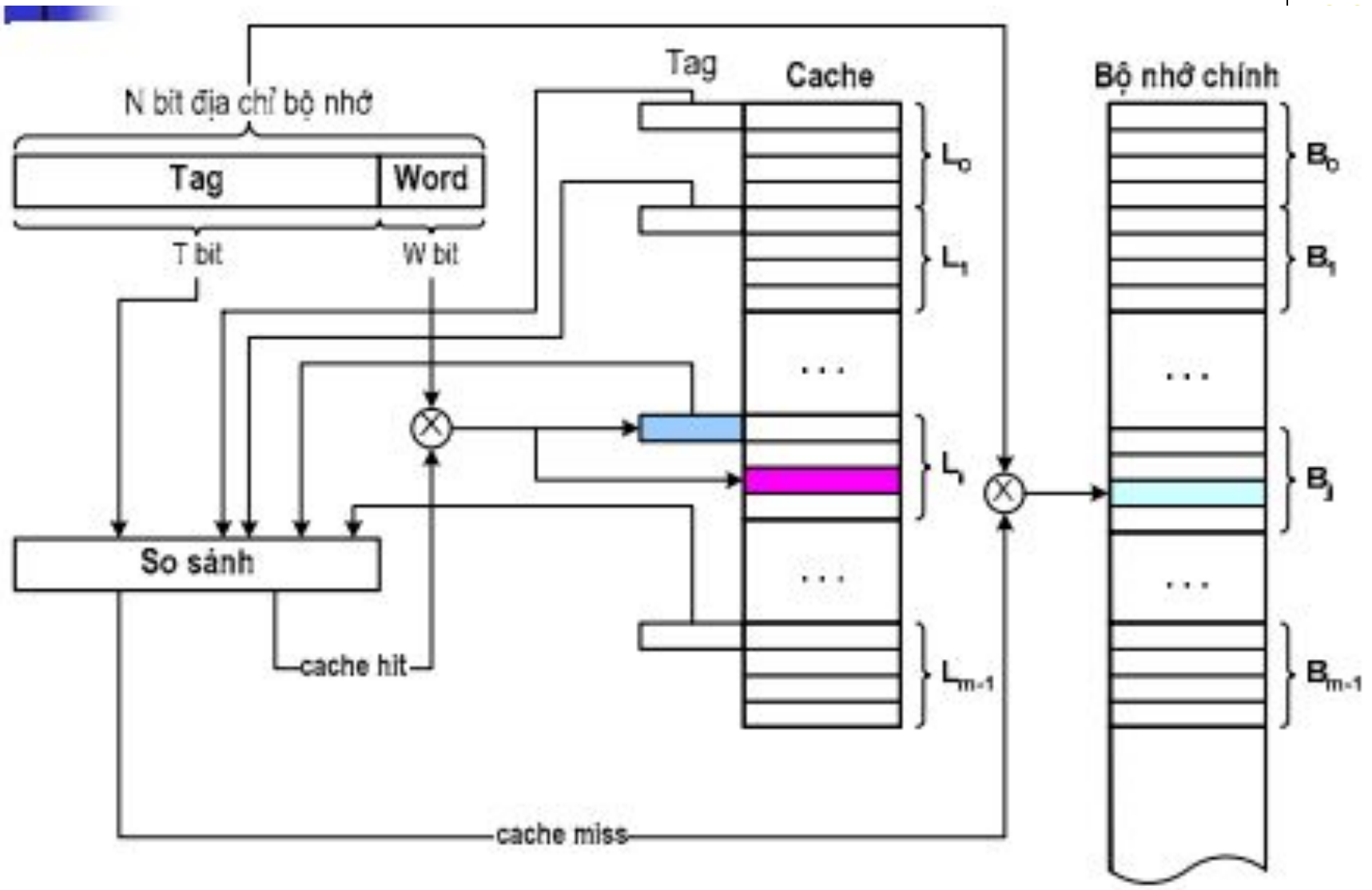
  = **30842h**

Ánh xạ liên kết toàn phần

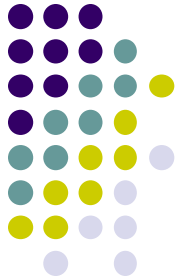
- Mỗi Block có thể được nạp vào line bất kỳ của cache.
- Địa chỉ của bộ nhớ chính bao gồm hai trường:
  - Trường **Word (n1)** giống như trường hợp ở trên.
  - Trường ***Tag (n3)*** dùng để xác định *Block* của bộ nhớ chính.
- Tag xác định Block nào đang nằm ở Line đó.



Minh họa ánh xạ toàn phần



Đặc điểm của ánh xạ liên kết toàn phần

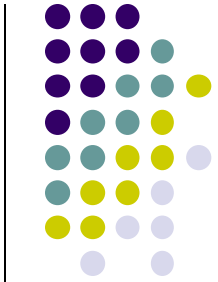


- Phải so sánh đồng thời với tất cả các Tag Tốn thời gian so sánh.
- Bộ so sánh phức tạp.
- Tỷ lệ cache hit cao hơn phương pháp ánh xạ trực tiếp.

Ví dụ minh họa

- CPU có 24 bit địa chỉ, Bộ nhớ chính **256** KB chia làm **512** block nhớ, bộ nhớ cache có dung lượng **8** KB. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là **30815**h. Hãy trình bày chi tiết phương pháp đọc cache theo kỹ thuật ánh xạ liên kết toàn phần cho trường hợp phát ra địa chỉ trên và chỉ ra địa chỉ vật lý của ô nhớ cần truy cập trong bộ nhớ chính.
- Đỉa chỉ có mấy phần: Tag+Word (n_3+n_1) chiều dài của địa chỉ hợp lệ Địa chỉ hợp lệ là địa chỉ có chiều dài <= (n_3+n_1) hoặc (T+W).
- n_1 (w) liên quan đến kích thước block. n_3 liên quan số block

Các bước giải ví dụ minh họa



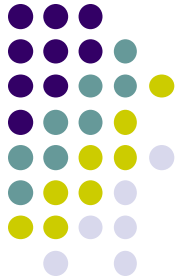
- Bước 1: Xác định dung lượng của 1 block (line) nhớ.

$$C_{block} = \frac{C_{memory}}{n_{block}} = \frac{256KB}{512} = \frac{2^8 \times 2^{10}}{2^9} = 2^9 (byte)$$

→ W (n_1) = 9 bits

- Bước 2: Xác định số bít cần thiết để đánh số hiệu block trong bộ nhớ.
  - Số block = 512 = 2^9 → Cần 9 bits để đánh số hiệu block □ T (n_3) = 9 (bits).
- Bước 3: Xác định số bít địa chỉ hợp lệ cho bộ nhớ 256KB là: 9 + 9 = 18 bits.
- Số line của cache = Dung lượng cache/Dung

Các bước giải ví dụ minh họa (tiếp)

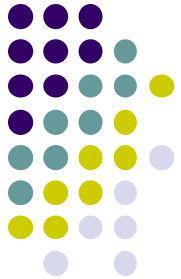


- Bước 4: Đọc cache với địa chỉ 30815h
- Địa chỉ có hợp lệ không?
  - Chuyển địa chỉ từ hệ hex sang hệ nhị phân:

30815h = **110000100**000010101b

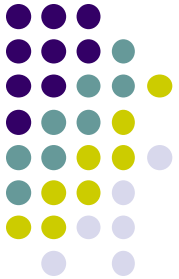
  - Kiểm tra xem địa chỉ cần truy cập có hợp lệ không: Hợp lệ vì số bít của địa chỉ là 18 bằng số bít địa chỉ hợp lệ CPU phát địa chỉ này lên bus địa chỉ.
  - Bộ điều khiển cache xác định giá trị của trường Tag là 9 bít cao của địa chỉ trên để xác định số hiệu block bộ nhớ cần truy cập: 110000100 = 256 + 128 + 4 = 388.
  - Bộ điều khiển cache sẽ so sánh trường Tag của tất cả các line với 388. Có 2 khả năng xảy ra.

Các bước giải ví dụ minh họa (tiếp)



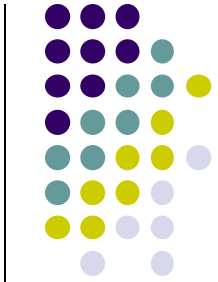
- Bước 4: (tiếp)
  - Trường hợp 1: Không có Tag của line nào có giá trị 388 Block cần truy cập chưa có trong cache (miss cache). CPU phải nạp block có số hiệu 388 từ bộ nhớ chính vào cache. CPU tìm line nào trong cache sử dụng kém hiệu quả nhất để thay bằng block mới. Nếu bít F của line bị thay thế = 1 thì nội dung của line đó sẽ được ghi trả lại bộ nhớ vào đúng block có giá trị bằng trường Tag của line đó. Sau đó xóa F = 0 rồi nạp block nhớ có số hiệu 388 vào line này và cập nhật trường Tag của nó bằng 388. Sau đó CPU sẽ đọc byte cần trong line này.

Các bước giải ví dụ minh họa (tiếp)



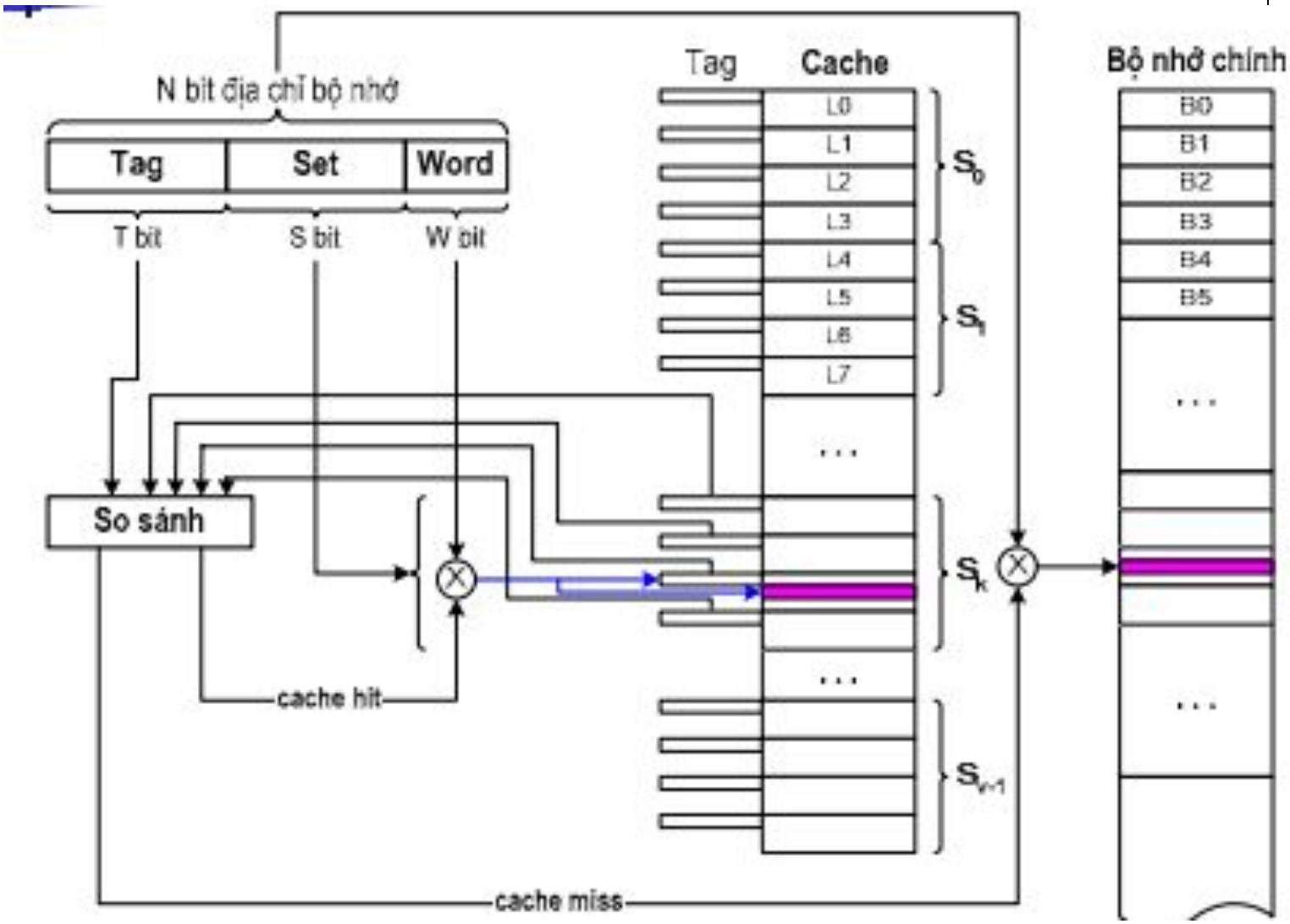
- Bước 4: (tiếp)
  - Trường hợp 2: Có một Tag của line nào đó = 388 (hit cache). CPU sẽ đọc byte có số hiệu là 9 bit thấp của địa chỉ: 000010101b = 16 + 4 + 1 = 21 □ đọc byte có số hiệu 21 của line đó.
  - Địa chỉ vật lý của ô nhớ cần truy cập chính là: **30815h (TW).**

Ánh xạ liên kết tập hợp

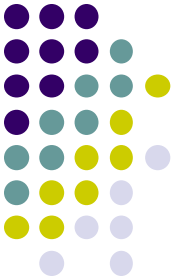


- Cache được chia thành các Tập (Set)
- Mỗi một Set chứa một số Line
- Ví dụ:
  - 4 Line/Set → 4-way associative mapping
- □ Ánh xạ theo nguyên tắc sau:
  - $B_0 \rightarrow S_0$
  - $B_1 \rightarrow S_1$
  - $B_2 \rightarrow S_2$
  - ...
  - $B_m \rightarrow S_{m \bmod q.}$
  - Trong đó: q là số tập (set) của cache.
  - □ Địa chỉ CPU phát phải có 3 phần: **T + S + W** ($n_3 n_2 n_1$)

Minh họa ánh xạ liên kết tập hợp



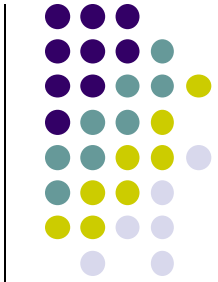
Đặc điểm của ánh xạ liên kết tập hợp



- N bits địa chỉ hợp lệ được chia làm 3 phần:
  - W bits thấp nhất dùng để xác định số hiệu của từ nhớ trong line (block) cần truy cập → kích thước của một block = 2^W
  - S bits kế tiếp dùng để xác định một trong 2^S set cần truy cập.
  - T bits trọng số cao nhất dùng để xác định block cần truy cập.

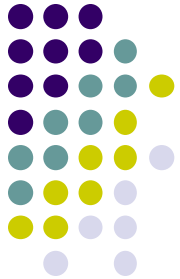
→ N = T + S + W

Thao tác đọc cache



- SI (Set Index)– là số hiệu Set cần truy nhập.
- BI (Block Index)- là số hiệu Block nhớ cần truy nhập.
  - **Bước 1**:MMU kiểm tra tính hợp lệ của địa chỉ truy nhập.
    - **Trường hợp 1**: Nếu số bit có nghĩa biểu diễn địa chỉ truy nhập lớn hơn số bit hợp lệ (lớn hơn n), thì địa chỉ này không hợp lệ, tiến trình phải dừng.
    - **Trường hợp 2**: Nếu SI ≠ (BI mod q), thì địa chỉ này không hợp lệ, tiến trình phải dừng (Số lượng Set trong cache là q).
    - Ngược lại, CPU thực hiện phát ra địa chỉ truy nhập lên Bus A, sang bước 2.
  - **Bước 2:** Đọc cache

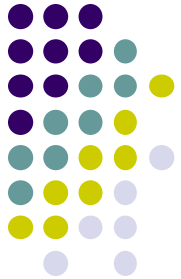
Ví dụ minh họa



- CPU có **24** bit địa chỉ, bộ nhớ chính **256 KB** chia làm **512** block nhớ, bộ nhớ cache có dung lượng **8 KB** chia làm **4 set**. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là **73426h**. Hãy trình bày chi tiết phương pháp đọc cache cho trường hợp phát ra địa chỉ trên và địa chỉ vật lý của ô nhớ cần truy nhập trong bộ nhớ.

- Ánh xạ liên kết tập hợp □**T S W** □ **Địa chỉ hợp lệ có độ dài tối đa là N = T + S + W.**
- **S liên quan đến số set của cache: Số set = 2^S** □ **S**

Phân tích

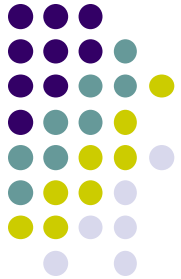


- Địa chỉ hợp lệ ? Hợp lệ nếu thỏa mãn 2 yêu cầu sau:
  - Hợp lệ về chiều dài: Số bit của địa chỉ <= số bit của địa chỉ hợp lệ (1).
  - Block phải nạp đúng set. SI = số hiệu block mod q (q là số set của cache) (2)

  Thỏa mãn đồng thời (1) và (2) địa chỉ mới hợp lệ.

  Số N = T + S + W

Bước 1: Xác định số bit W



- Xác định dung lượng của 1 block (line) nhớ.

$$C_{block} = \frac{C_{memory}}{n_{block}} = \frac{256KB}{512} = \frac{2^8 \times 2^{10}}{2^9} = 2^9 (byte)$$

$\rightarrow$ W = 9 (bits)

Bước 2

- Xác định số lượng line của cache.

$$n_{line} = \frac{C_{cache}}{C_{block}} = \frac{8KB}{0.5KB} = 16 = 2^4 (lines)$$

- Xác định số line trong một set

$$n_{line-set} = \frac{n_{line}}{n_{set}} = \frac{16}{4} = 4(lines)$$

- Xác định số bít để đánh số hiệu của set

$n_{set} = 4 = 2^2 \rightarrow$ ***S = 2 (bits)***

Bước 3 và 4

- Xác định số bit đánh số hiệu cho block

$$n_{block} = 512 = 2^9 (blocks)$$

  - **T = 9 bits**

- Xác định số bits tối đa của địa chỉ hợp lệ:

N = T + S + W = 9 + 2 + 9 = 20 (bits)

Bước 5

- Đọc cache với địa chỉ 73426h
  - Đổi địa chỉ sang hệ nhị phân:

    73426h = **11100110100**0001100110b → số bits có nghĩa của địa chỉ là 19 nhỏ hơn số bits địa chỉ hợp lệ địa chỉ này hợp lệ về chiều dài. (1)

  - S = 10b Số hiệu của tập cần truy cập SI = 2
  - T = 11100110b = 128 + 64 + 32 + 4 + 2 = 230

    BI = 230
  - Kiểm tra xem block có số hiệu 230 nạp vào set 2 của cache có hợp lệ không: 230 mod 4 = 2 → hợp lệ block nạp đúng set (2)(1) và (2) địa chỉ hợp lệ.

Bước 5 (tiếp)

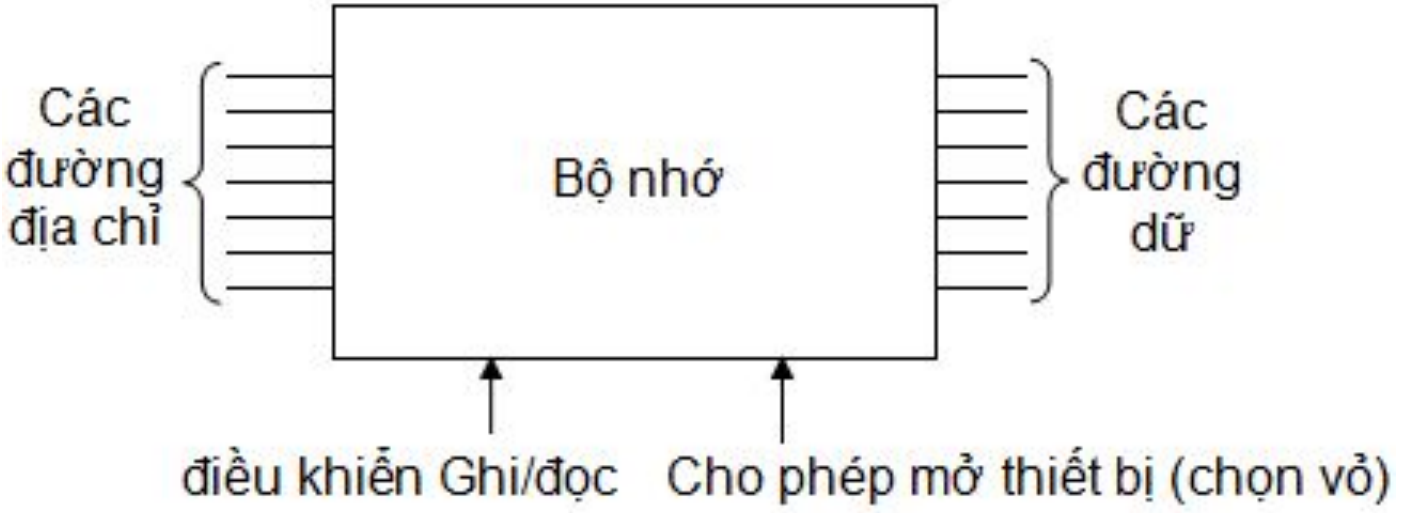
- Bộ điều khiển cache (MMU) tiến hành so sánh trường Tag của 4 line trong Set 2 với 230. Có 2 khả năng xảy ra.
  - Miss cache: Không có Tag nào trùng với 230. Khi đó MMU phải nạp block nhớ có số hiệu 230 vào một line của Set 2 → như trường hợp 2.
  - Hit cache: Một trong số 4 line của Set 2 có trường Tag = 230. Khi đó CPU sẽ đọc từ nhớ có số hiệu là 9 bits thấp của địa chỉ: 000100110b = 32 + 6 = 38 trong line này.

☞ Xác định địa chỉ ô nhớ cần truy nhập trong bộ nhớ: Ghép 9bits của Tag với 9 bits của Word ta được địa chỉ ô nhớ:

111001100001100110b = **1CC26**h

Bộ nhớ bán dẫn

- Sơ đồ khối của bộ nhớ bán dẫn.

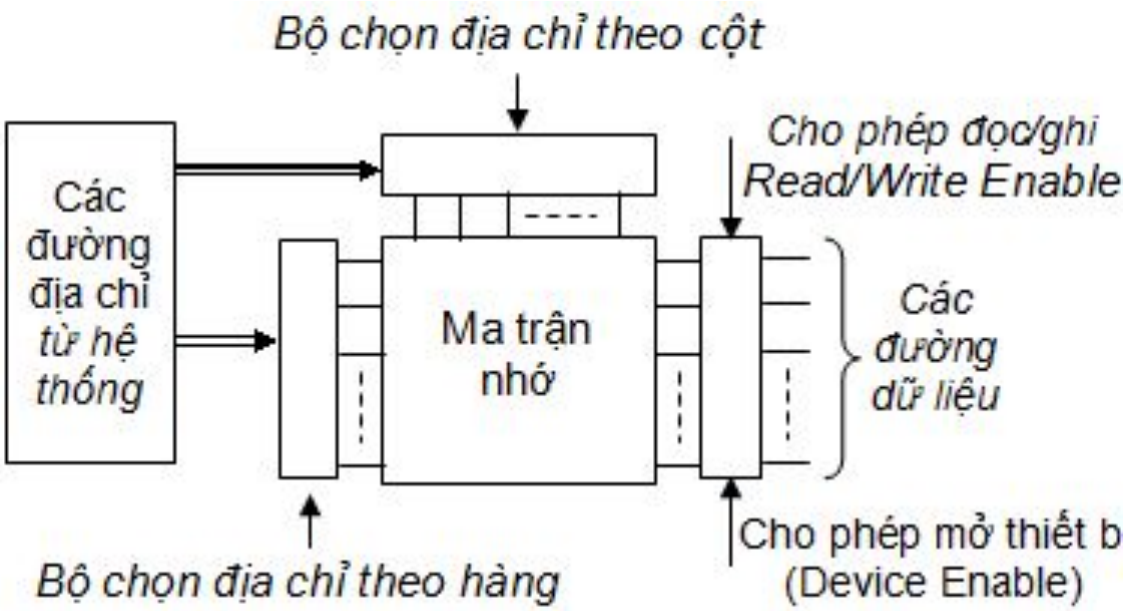


- Số đường dây địa chỉ là n → dung lượng bộ nhớ sẽ là 2^n

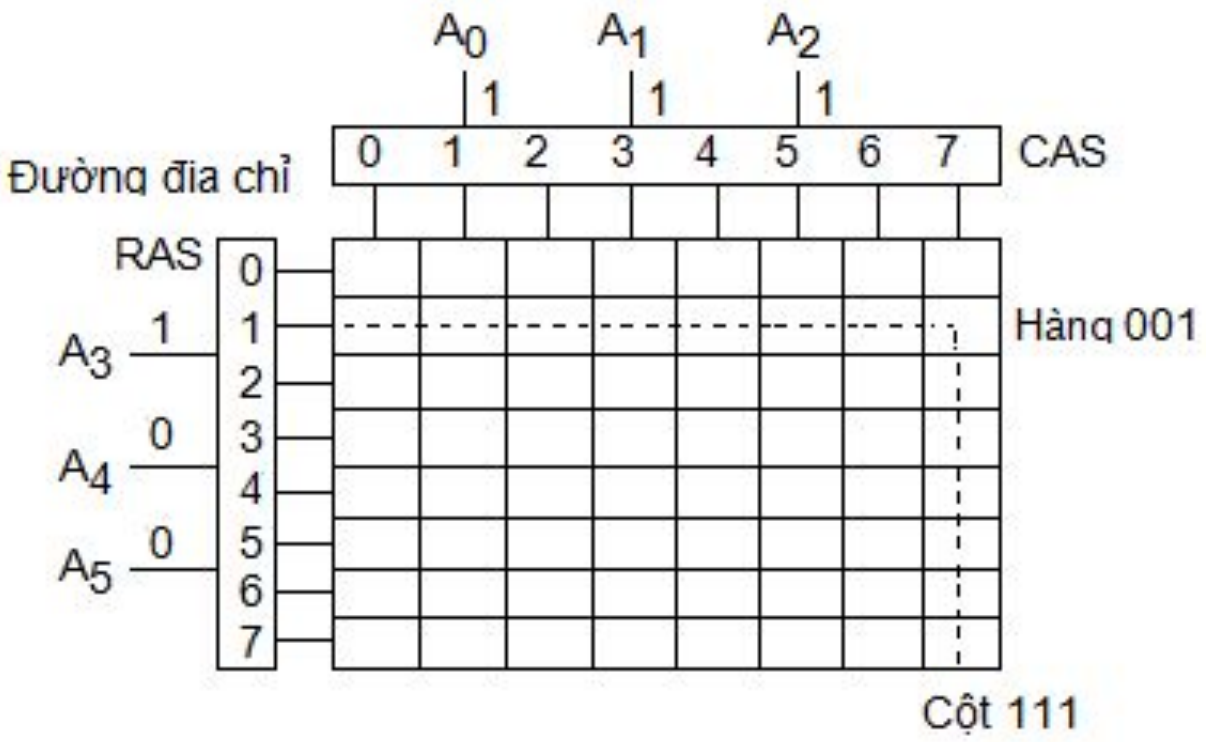
Tổ chức của bộ nhớ bán dẫn

- Tổ chức dưới dạng ma trận các dòng và cột

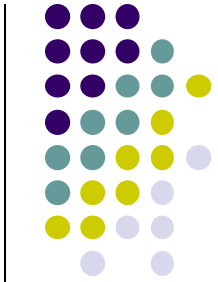
→ Để xác định một ô nhớ (cell) cần xác định dòng và cột → giải mã dòng và cột



Vi dụ tổ chức bộ nhớ

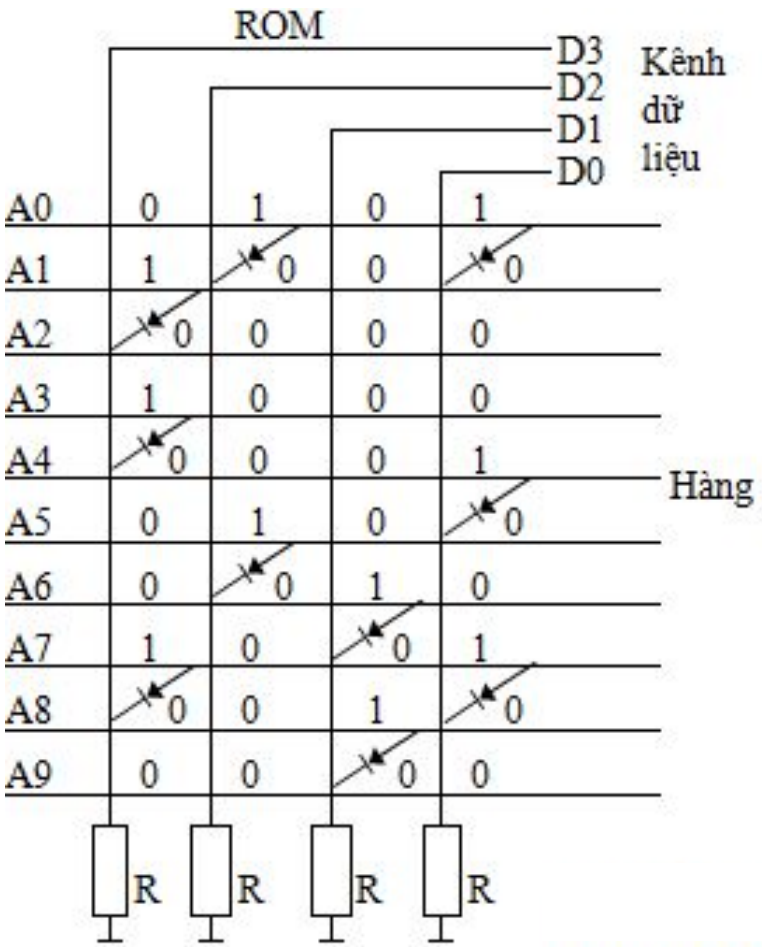


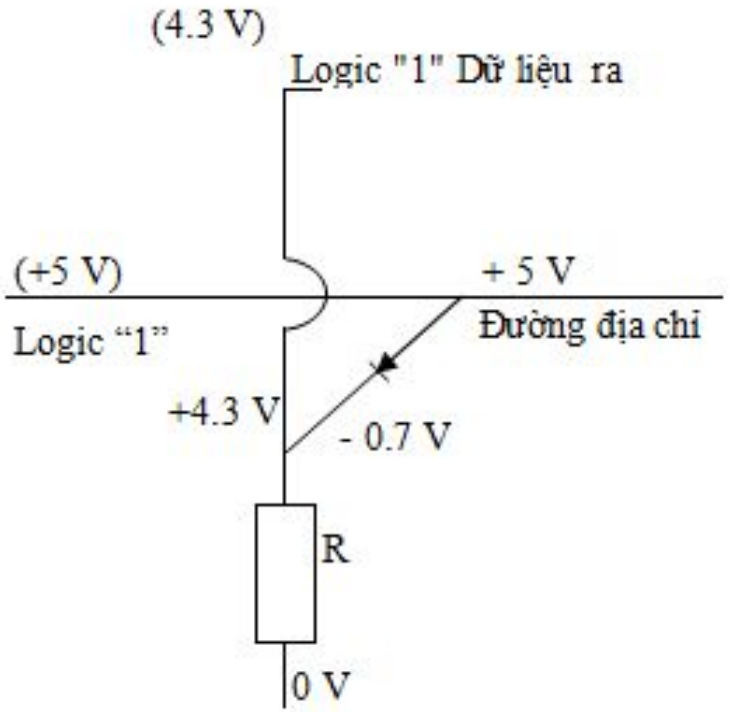
Bộ nhớ ROM



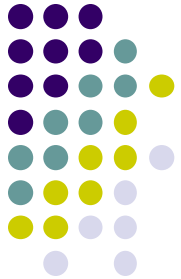
- Đặc điểm
  - Bộ nhớ chỉ đọc
  - Truy cập ngẫu nhiên
  - Mất điện không bị mất thông tin
- Phân loại
  - ROM (Read Only Memory)
  - PROM (Programmable ROM)
  - EPROM (Erasable PROM)
  - EEPROM (Electrical EPROM)

ROM dùng diode



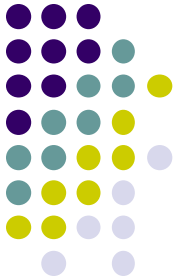


Bảng dữ liệu lưu trong ROM



INPUT ADDRESS										OUTPUT DATA			
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	1	0	1	0	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	1	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bộ nhớ RAM (Random Access Memory)

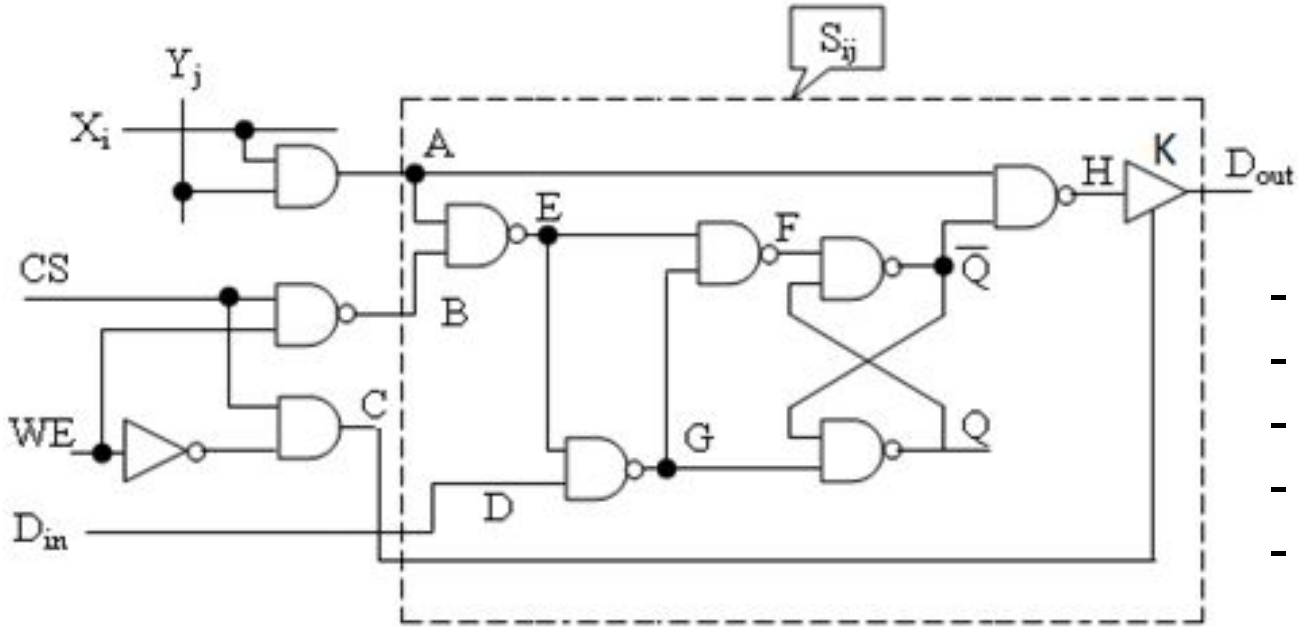


- Đặc điểm
  - Bộ nhớ vừa đọc vừa ghi được
  - Truy cập ngẫu nhiên
  - Mất điện bị mất thông tin
- Phân loại
  - SRAM (Static RAM)
  - DRAM (Dynamic RAM)

SRAM – RAM tĩnh

- Đặc điểm
  - Xây dựng dựa vào phần tử Flip - Flop
  - Tốc độ nhanh
  - Giá thành đắt
  - Dung lượng nhỏ → chế tạo cache L1 và L2

Cấu tạo phần tử nhớ 1 bit SRAM



- D_{in}: tín hiệu đầu vào.
- D_{out}: tín hiệu đầu ra.
- X_i, Y_j: các dây địa chỉ
- CS: Tín hiệu chọn chip
- WE: Đ/Kh đọc ghi

Ghi dữ liệu vào SRAM

- Chứng minh rằng điều kiện ghi: CS = 1, X_i = 1, Y_j = 1, WE = 1 thì Q = D_{in}, đồng thời giữa H và D_{out} ở trạng thái trở kháng cao (D_{out} = "Z").

+ WE = 1 ==> $\overline{WE} = 0$;

+ C = AND(CS, $\overline{WE}$) = AND(1, 0) = 0, nên thiết bị 3 trạng thái K ở trạng thái trở kháng cao, vậy D_{out} = "z"

+ A = AND(X_i, Y_j) = AND(1, 1) = 1

+ B = NAND(WE, CS) = NAND (1, 1) = 0

+ E = NAND(A, B) = NAND (1, 0) = 1

+ Nếu D_{in} = 0
  - G = NAND(E, D_{in}) = NAND (1, 0) = 1
  - F = NAND(E, G) = NAND (1, 1) = 0
  - $\overline{Q}$ = NAND(F, Q) = NAND (0, Q) = 1
  - Q = NAND(G, $\overline{Q}$) = NAND (1, 1) = 0

  hay Q = D_{in} = 0 (1)

+ Nếu D_{in} = 1
  - G = NAND(E, D_{in}) = NAND (1, 1) = 0
  - F = NAND(E, G) = NAND (1, 0) = 1
  - Q = NAND(G, $\overline{Q}$) = NAND (0, 1) = 1,

  hay Q = D_{in} = 1 (2)

Từ (1) & (2) ta thấy trong trường hợp này Q luôn = D_{in}

Đọc dữ liệu từ SRAM

- Chứng minh rằng điều kiện đọc: CS = 1, X_i = 1, Y_j = 1, WE = 0 thì Q không đổi, đồng thời D_{out} = H = Q.

+ WE = 0 ==> $\overline{WE}$ = 1;

+ C = AND(CS, $\overline{WE}$) = AND(1, 1) = 1,

nên thiết bị 3 trạng thái K ở trạng thái thông, vậy D_{out} = H

+ A = AND(X_i, Y_j) = AND(1, 1) = 1
+ B = NAND(WE, CS) = NAND (0, 1) = 1
+ E = NAND(A, B) = NAND (1, 1) = 0
+ G = NAND(E, D_{in}) = NAND (0, D_{in}) = 1
+ F = NAND(E, G) = NAND (0, 1) = 1

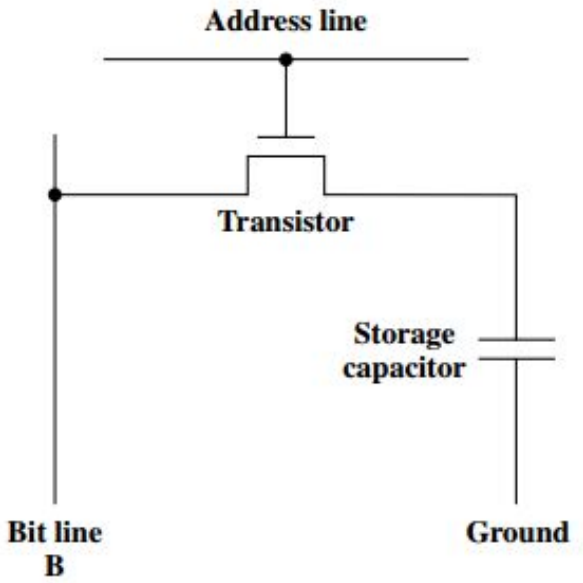
+ Giả sử ban đầu Q = 0
  - $\bar{Q}$ = NAND(F, Q) = NAND (1, 0) = 1
  - Q = NAND(G, $\bar{Q}$) = NAND (1, 1) = 0 (1')
+ Giả sử ban đầu Q = 1
  - $\bar{Q}$ = NAND(F, Q) = NAND (1, 1) = 0
  - Q = NAND(G, $\bar{Q}$) = NAND (1, 0) = 1 (2')

Từ (1') & (2') ta thấy trong trường hợp này Q không đổi

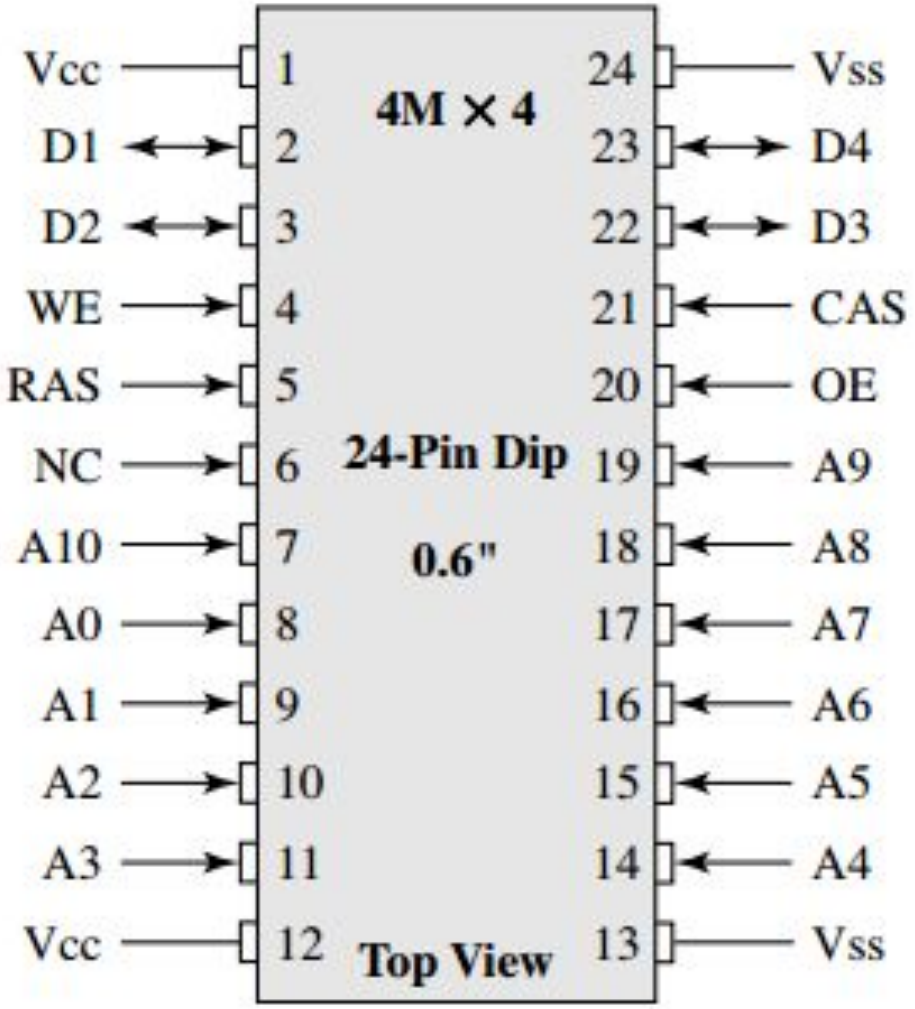
D_{out} = H = NAND(A, $\bar{Q}$) = NAND (1, $\bar{Q}$) = Q

DRAM – RAM động

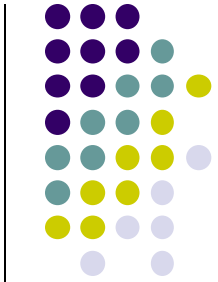
- Đặc điểm
  - Tốc độ chậm hơn SRAM do phải làm tươi
  - Giá thành rẻ hơn SRAM
  - Dung lượng lớn → chế tạo bộ nhớ chính
- Cấu tạo



Ví dụ đóng gói DRAM

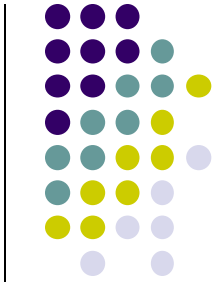


Thiết kế module nhớ



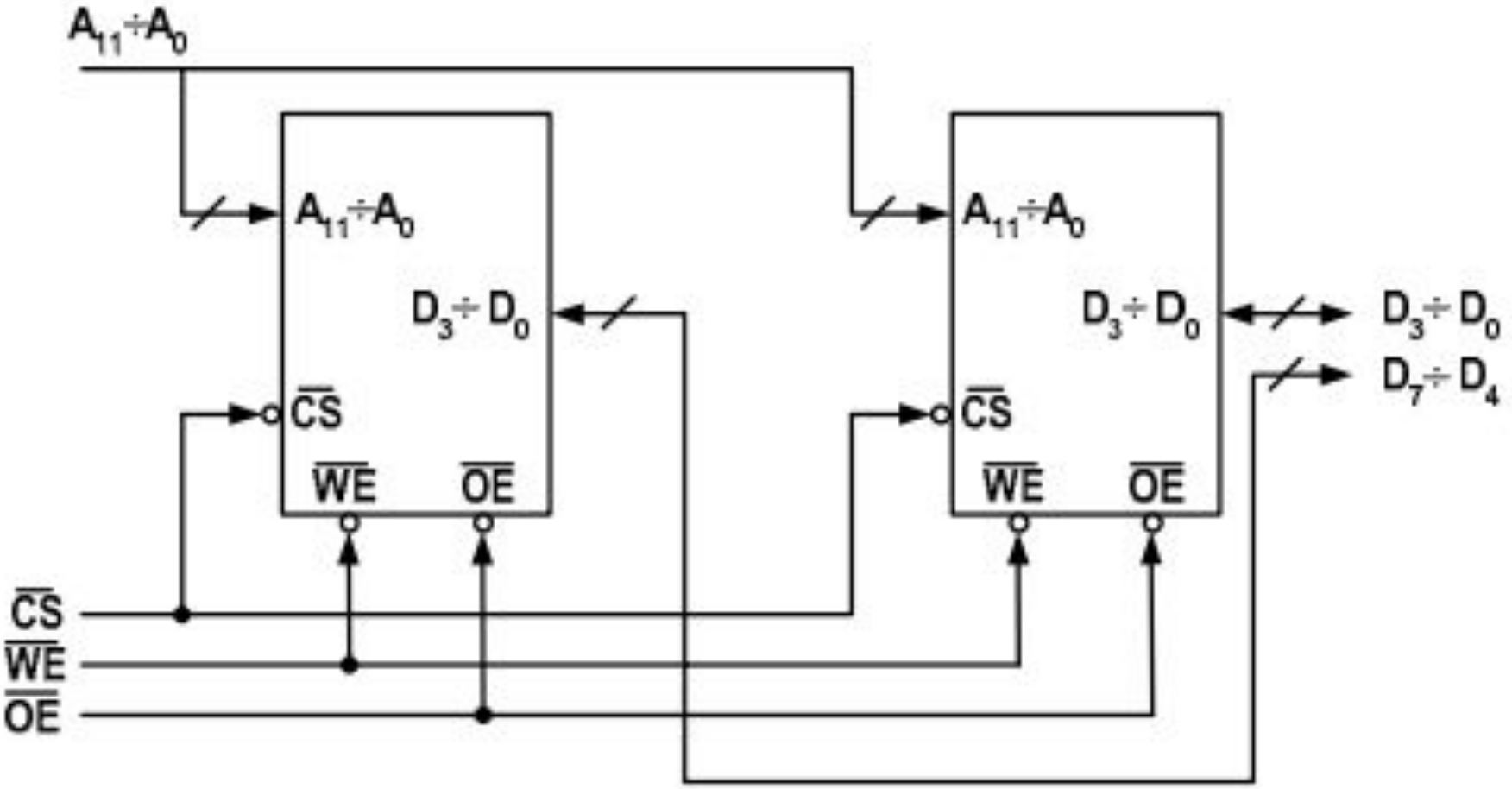
- Dung lượng chip nhớ 2n x m bit
- Cần thiết kế để tăng dung lượng:
  - Thiết kế tăng độ dài từ nhớ
  - Thiết kế tăng số lượng từ nhớ
  - Thiết kế kết hợp

Tăng độ dài từ nhớ

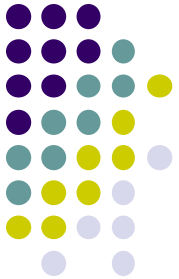


- Cho chip nhớ SRAM 4K x 4 bit
- Thiết kế mô-đun nhớ 4K x 8 bit
- Giải:
- Dung lượng chip nhớ = 2^{12} x 4 bit
- Chip nhớ có:
  - 12 chân địa chỉ
  - 4 chân dữ liệu
- Mô-đun nhớ cần có:
  - 12 chân địa chỉ
  - 8 chân dữ liệu

Tăng độ dài từ nhớ (tiếp)

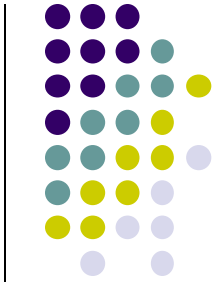


Tăng số lương từ nhớ



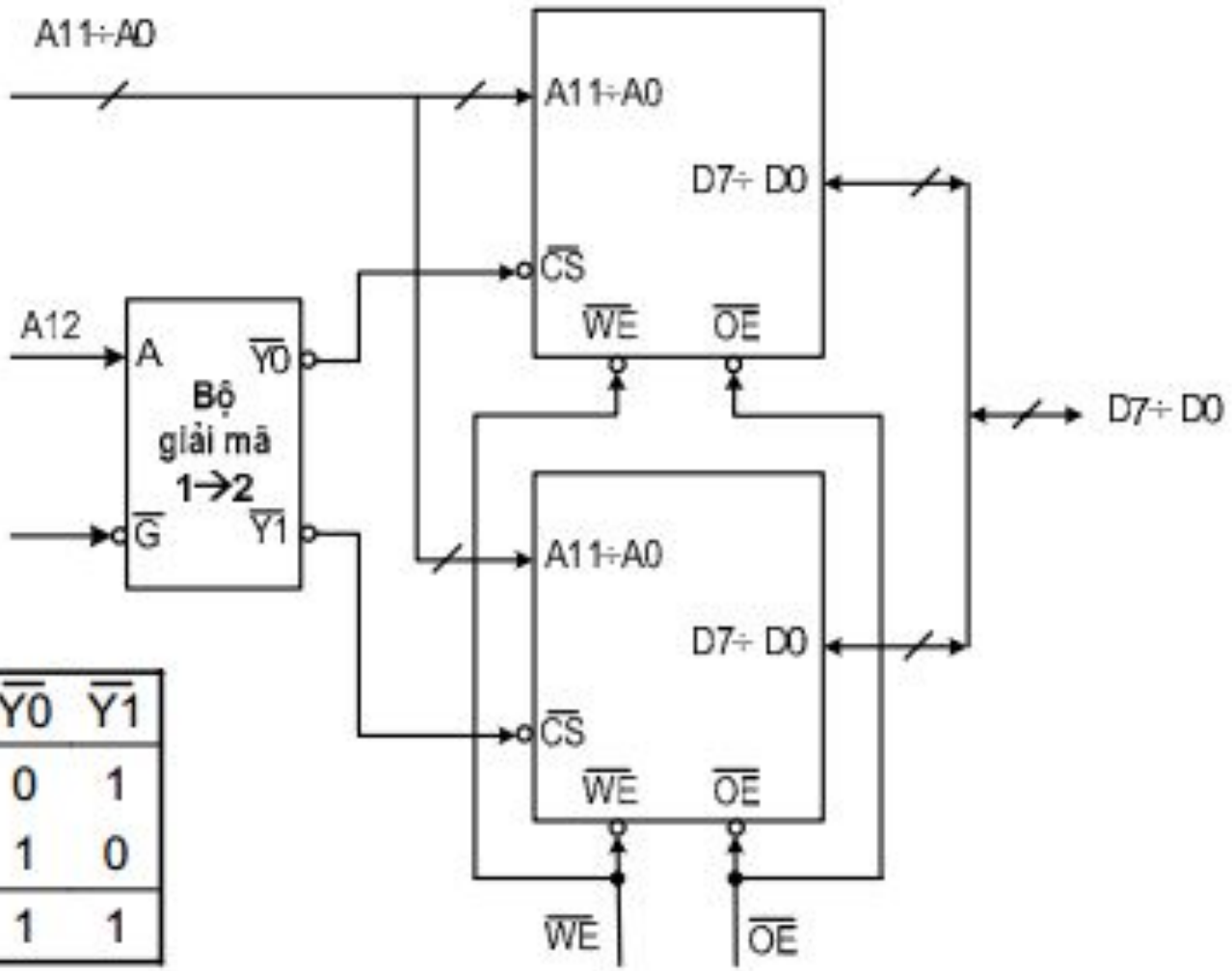
- Cho chip nhớ 2^n x mbit
- Thiết kế mô-đun nhớ 2^n x k.(m) bit
- □ Dùng k chip nhớ mắc nối tiếp

Ví dụ tăng số lương từ nhớ



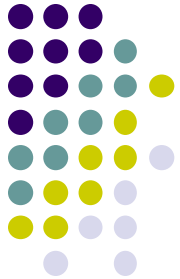
- Cho chip nhớ SRAM 4K x 8 bit
- Thiết kế mô-đun nhớ 8K x 8 bit
- Giải:
- Dung lượng chip nhớ = 2^{12} x 8 bit
- Chip nhớ có:
  - 12 chân địa chỉ
  - 8 chân dữ liệu
- Dung lượng mô-đun nhớ = 2^{13} x 8 bit
  - 13 chân địa chỉ
  - □ 8 chân dữ liệu

Ví dụ tăng số lương từ nhớ (tiếp)



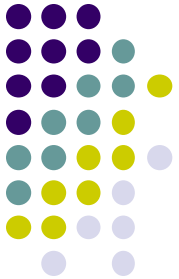
$\overline{G}$	A	$\overline{Y0}$	$\overline{Y1}$
0	0	0	1
0	1	1	0
1	x	1	1

Tăng kích thước và số lượng từ nhớ



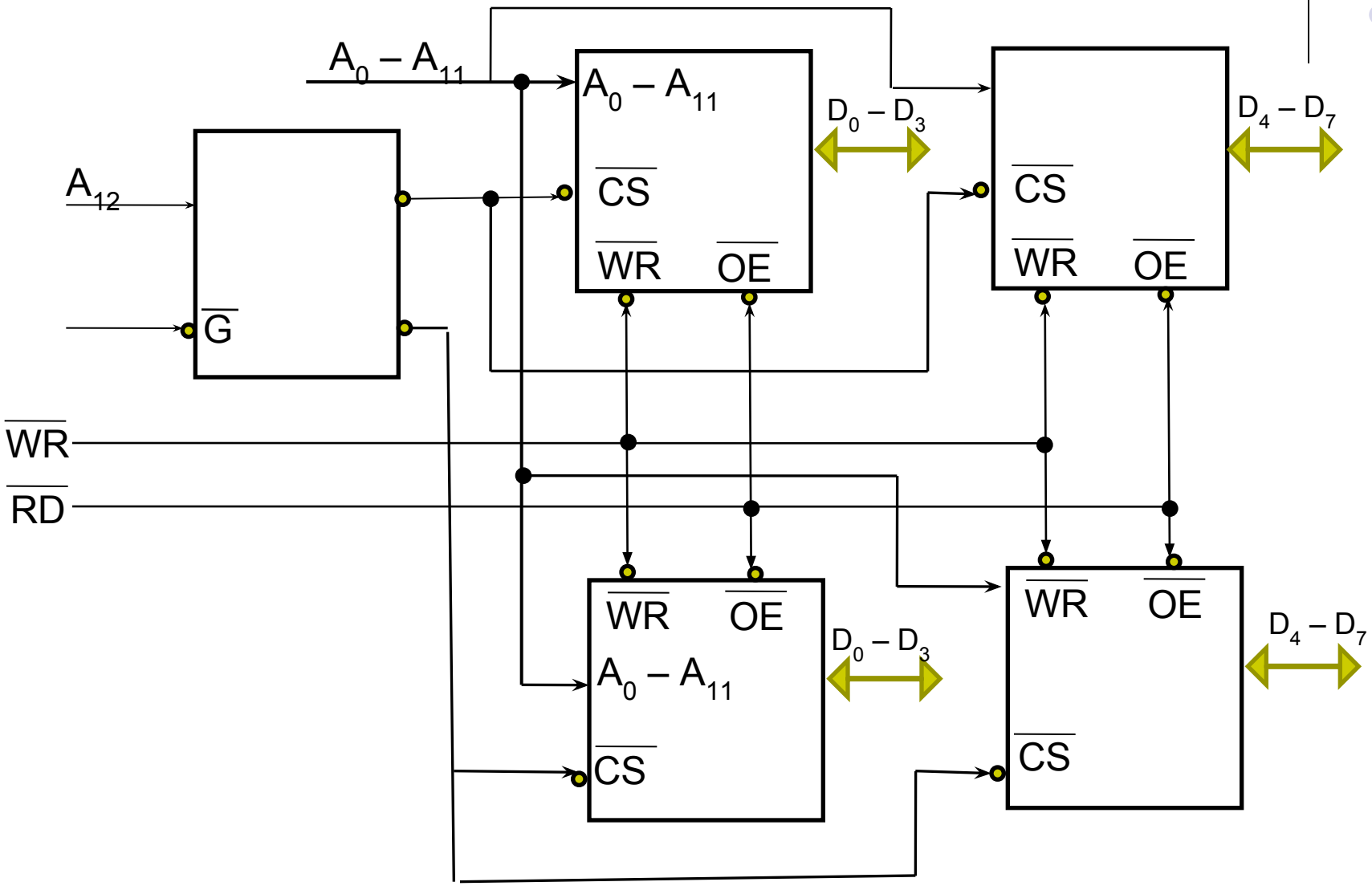
- Cho chip nhớ 2^n x mbit
- Thiết kế mô-đun nhớ 2^n x k.(m.p) bit
- □ Dùng k.p chip nhớ mắc hỗn hợp nối tiếp và song song trong đó:
  - p chip nhớ mắc song song
  - k dãy mắc nối tiếp

Ví dụ tăng số lương và kích thước từ nhớ

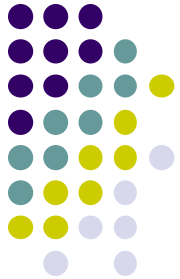


- Cho chip nhớ SRAM 4K x 4 bit
- Thiết kế mô-đun nhớ 8K x 8 bit
- Giải:
- Dung lượng chip nhớ = 2^{12} x 4 bit
- Chip nhớ có:
  - 12 chân địa chỉ
  - 4 chân dữ liệu
- Dung lượng mô-đun nhớ = 2^{13} x 8 bit
  - 13 chân địa chỉ
  - 8 chân dữ liệu

Ví dụ tăng kích thước và số lương từ nhớ (tiếp)

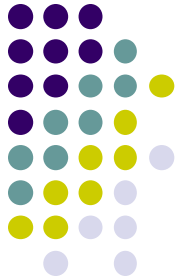


Bộ nhớ ảo



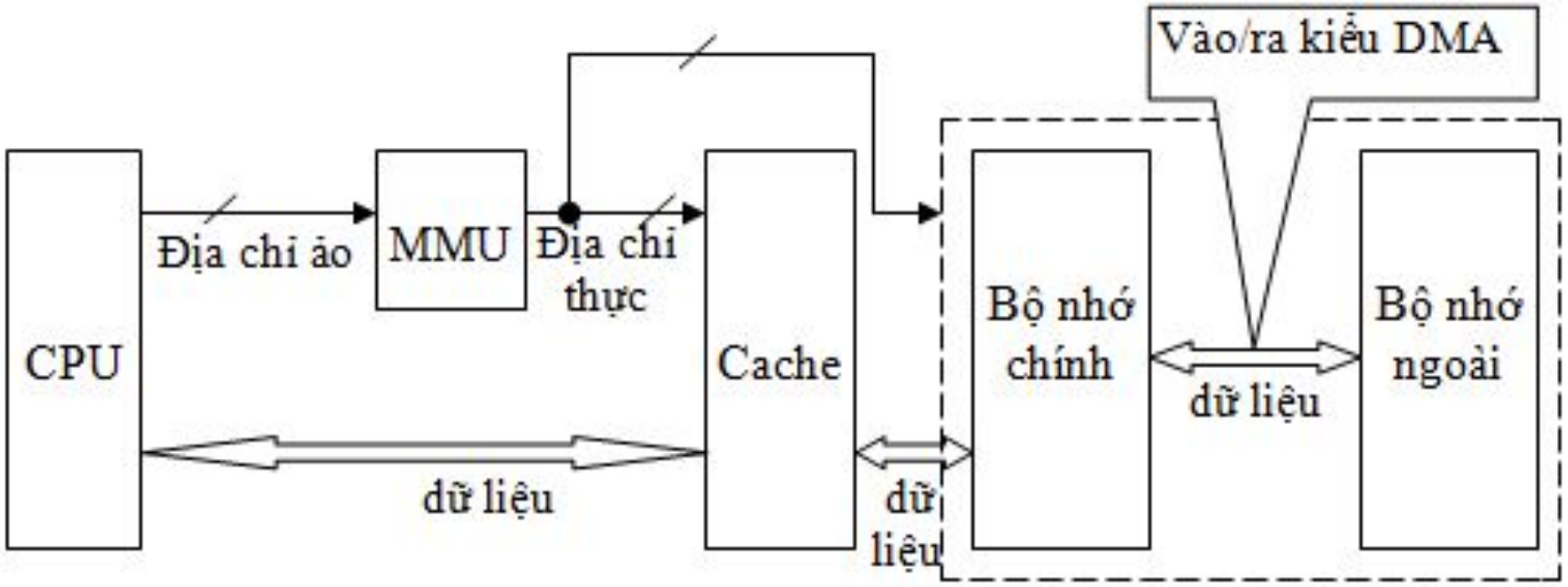
- **Bộ nhớ ảo (Virtual memory)** là: không gian trên đĩa cứng được giả lập như là phần thêm vào bộ nhớ chính. Hay nói cách khác thì **bộ nhớ ảo** một vùng nhớ trên đĩa cứng mà hệ điều hành coi nó như là một phần bộ nhớ RAM. Về bản chất, đây là một mẹo được thực hiện bởi hệ điều hành, nhờ đó nâng cao hiệu năng của hệ thống.

Kỹ thuật bộ nhớ ảo



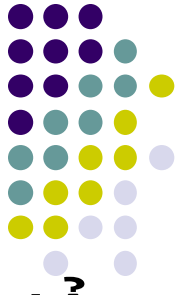
- Kỹ thuật bộ nhớ ảo sử dụng 2 loại địa chỉ:
  - Địa chỉ ảo (địa chỉ logic / địa chỉ chương trình): là các giá trị địa chỉ tiến trình tham chiếu, được tính từ 0 cho đến hết chương trình đã được biên dịch (khi chương trình được nạp vào thực thi thì trở thành tiến trình).
  - Bộ nhớ chương trình: là tập hợp tất cả các địa chỉ chương trình tạo thành. Bộ nhớ chương trình bằng dung lượng chương trình đã được biên dịch. Mỗi chương trình có dung lượng bộ nhớ chương trình khác nhau.
  - Địa chỉ vật lý hay còn gọi là địa chỉ thực trong bộ nhớ vật lý: được tính từ 0 cho đến hết vùng nhớ RAM có thực.
- Để truy nhập lệnh hoặc dữ liệu, cần có sự ánh xạ địa chỉ ảo về địa chỉ vật lý.

Minh họa nguyên lý bộ nhớ ảo



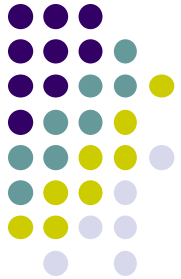
Sơ đồ nguyên lý kỹ thuật bộ nhớ ảo

Kỹ thuật thực hiện bộ nhớ ảo



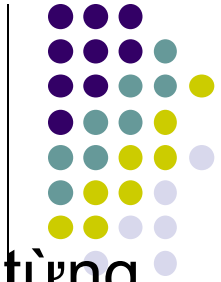
- 3 kỹ thuật thường được sử dụng để thực hiện bộ nhớ ảo:
  - Phân đoạn
  - Phân trang
  - Kết hợp 2 phương pháp trên

Đặc điểm của kỹ thuật phân trang và phân đoạn



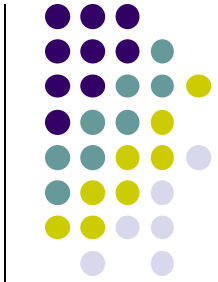
- Chương trình được chia làm nhiều phần.
- Nạp một hoặc nhiều phần chương trình vào bộ nhớ vật lý.
- Các phần còn lại chưa nhất thiết phải thi hành ngay thì được lưu trữ tạm ở một địa chỉ xác định trên ổ đĩa cứng (vùng swapping) do hệ điều hành quản lý.
- Khi phần chương trình được nạp đã thực hiện xong thì có thể được giải phóng khỏi bộ nhớ vật lý và đưa ra lưu trữ tạm trong ổ đĩa để tạo ra vùng nhớ vật lý tự do.
- Sau đó thực hiện nạp các phần chương trình còn lại để thực thi.

Kỹ thuật phân đoạn



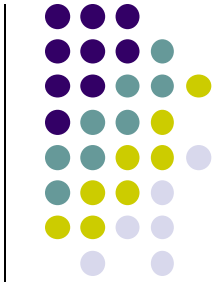
- Khi biên dịch, chương trình được biên dịch theo từng module.
- Mỗi module sẽ được nạp vào một vùng nhớ riêng biệt (một đoạn) trong bộ nhớ hoặc được cất trong vùng swap.
- Khi nạp và biên dịch chương trình, hệ thống sinh ra một bảng quản lý đoạn (Segment Control Block - SCB) để quản lý trạng thái các module trong chương trình.
- Địa chỉ đầu của bảng SCB được đưa vào thanh ghi Rs.
- Mỗi phần tử trong SCB phản ánh trạng thái của một module.

Kỹ thuật phân đoạn (tiếp)



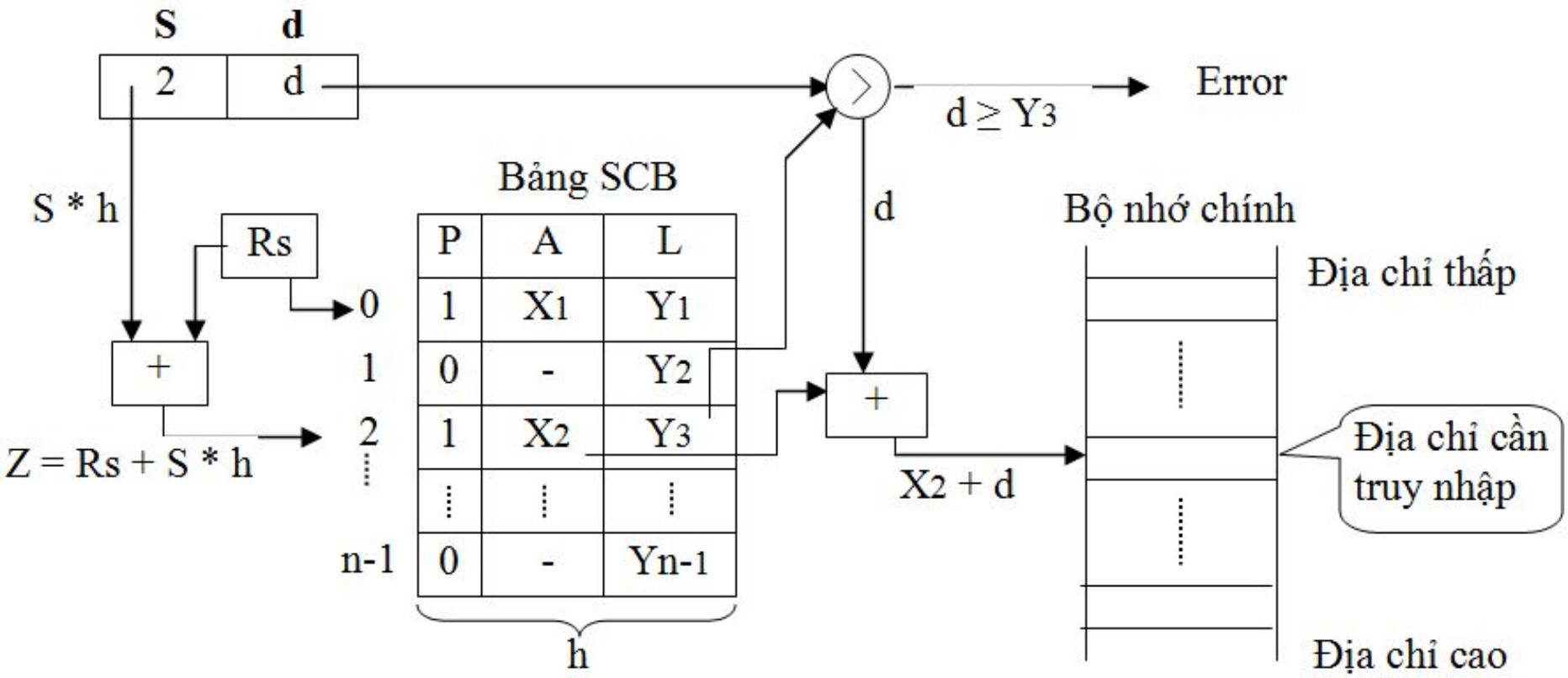
- Bảng Bảng SCB có 3 trường là P, A, L:
  - P - trường dấu hiệu:
    - P = 0 : đoạn chưa được nạp,nằm ở vùng swap do hệ thống quản lý.
    - P = 1: đoạn đã được nạp vào bộ nhớ chính.
  - A – trường địa chỉ: cho biết địa chỉ đầu vùng nhớ mà đoạn đã được nạp, trường này chỉ có ý nghĩa khi P = 1.
  - L – trường độ dài: cho biết độ dài của đoạn, hay chính là dung lượng của module khi biên dịch.

Kỹ thuật phân đoạn (tiếp)



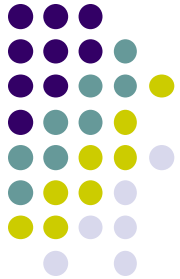
- Địa chỉ truy nhập lệnh hoặc dữ liệu được biểu diễn là bộ đôi giá trị **<S, d>**, địa chỉ này được gọi là địa chỉ logic.
  - S là số hiệu của đoạn cần truy nhập.
  - d là địa chỉ tương đối của lệnh hay dữ liệu trong đoạn.

Kỹ thuật phân đoạn (tiếp)



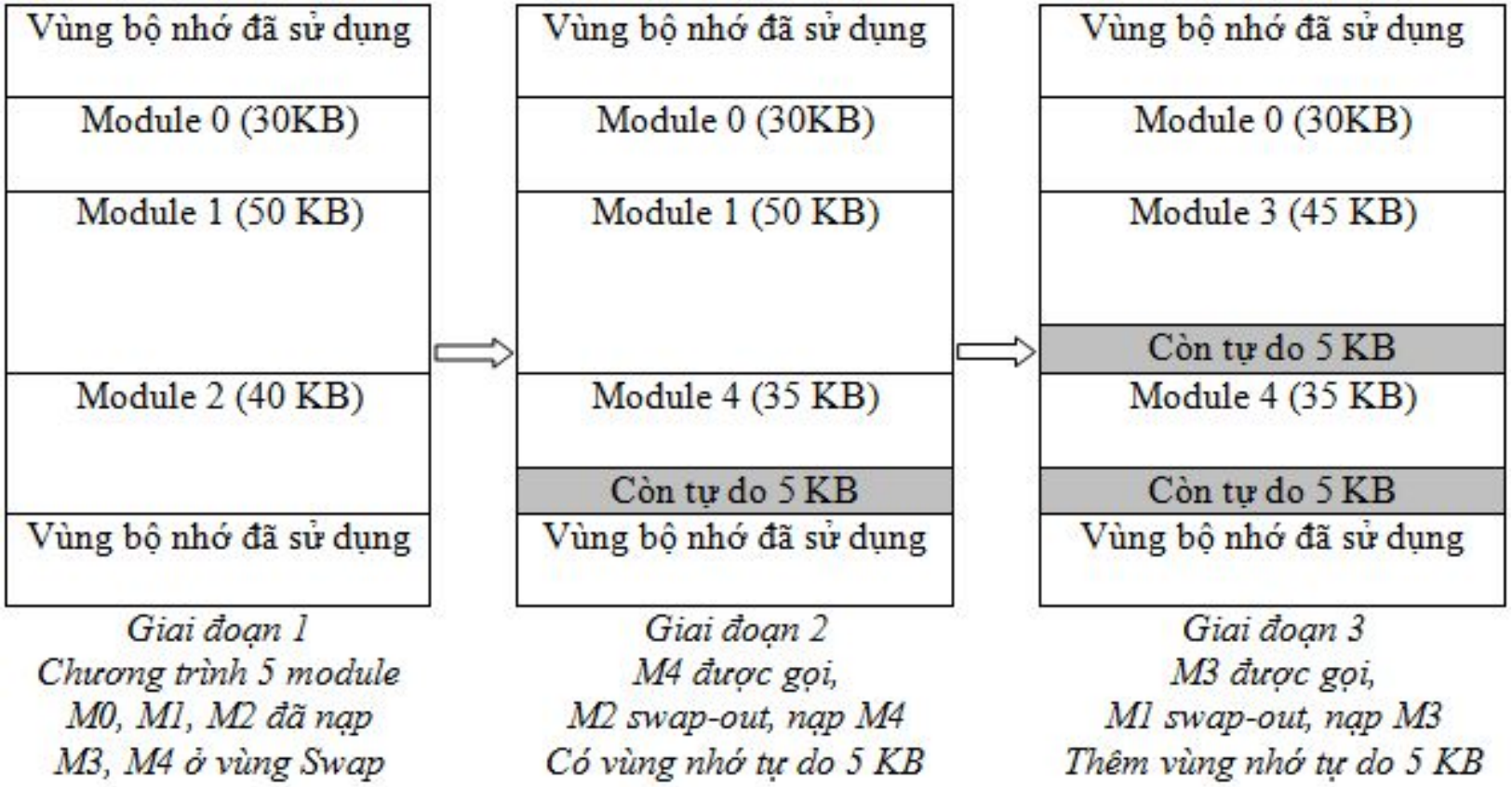
Sơ đồ cơ chế truy nhập bộ nhớ quản lý kiểu phân đoạn

Ưu và nhước điểm của kỹ thuật phân đoạn



- Ưu điểm:
  - Cho phép kích thước chương trình lớn hơn kích thước bộ nhớ còn tự do vẫn được thực thi.
- Nhược điểm:
  - Gây ra phân mảnh ngoại vi sau 1 thời gian sử dụng bộ nhớ, do vậy cần có kỹ thuật dồn bộ nhớ tự do.
  - Nếu module cần truy nhập chưa được nạp thì chương trình thực thi chậm do phải có thao tác nạp đoạn hoặc đổi đoạn
  - Sơ đồ này chỉ được áp dụng đồng bộ với chương trình có cấu trúc phân đoạn và hệ thống phần cứng hỗ trợ quản lý bộ nhớ kiểu phân đoạn (cần có các thanh ghi đoạn– segment register và các thanh ghi lệch – Offset register), ánh xạ địa chỉ logic về đ.ch vật lý.

Ví dụ minh họa phân mảnh bộ nhớ



Kỹ thuật phân trang

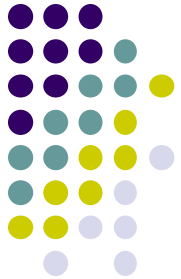
- Bộ nhớ vật lý được chia thành các trang bằng nhau có độ dài bằng L, gọi là các trang vật lý và được đánh số từ 0.
- Khi được gọi, chương trình được biên dịch thành một khối thống nhất, và cũng được chia thành các trang gọi là các trang logic (được đánh số từ 0) và kích thước 1 trang logic bằng kích thước của trang vật lý .
- Mỗi trang logic sẽ được nạp vào một trang vật lý trong bộ nhớ hoặc được cất trong vùng swap.
- Khi nạp và biên dịch chương trình, hệ thống sinh ra một bảng quản lý trang (Page Control Block - PCB) để quản lý trạng thái các trang logic trong chương trình.
- Địa chỉ đầu của bảng PCB được đưa vào thanh ghi Rp.
- Mỗi phần tử trong PCB phản ánh trạng thái của một trang logic.

Cấu trúc của bảng PCB

P	A
1	4
0	
1	8

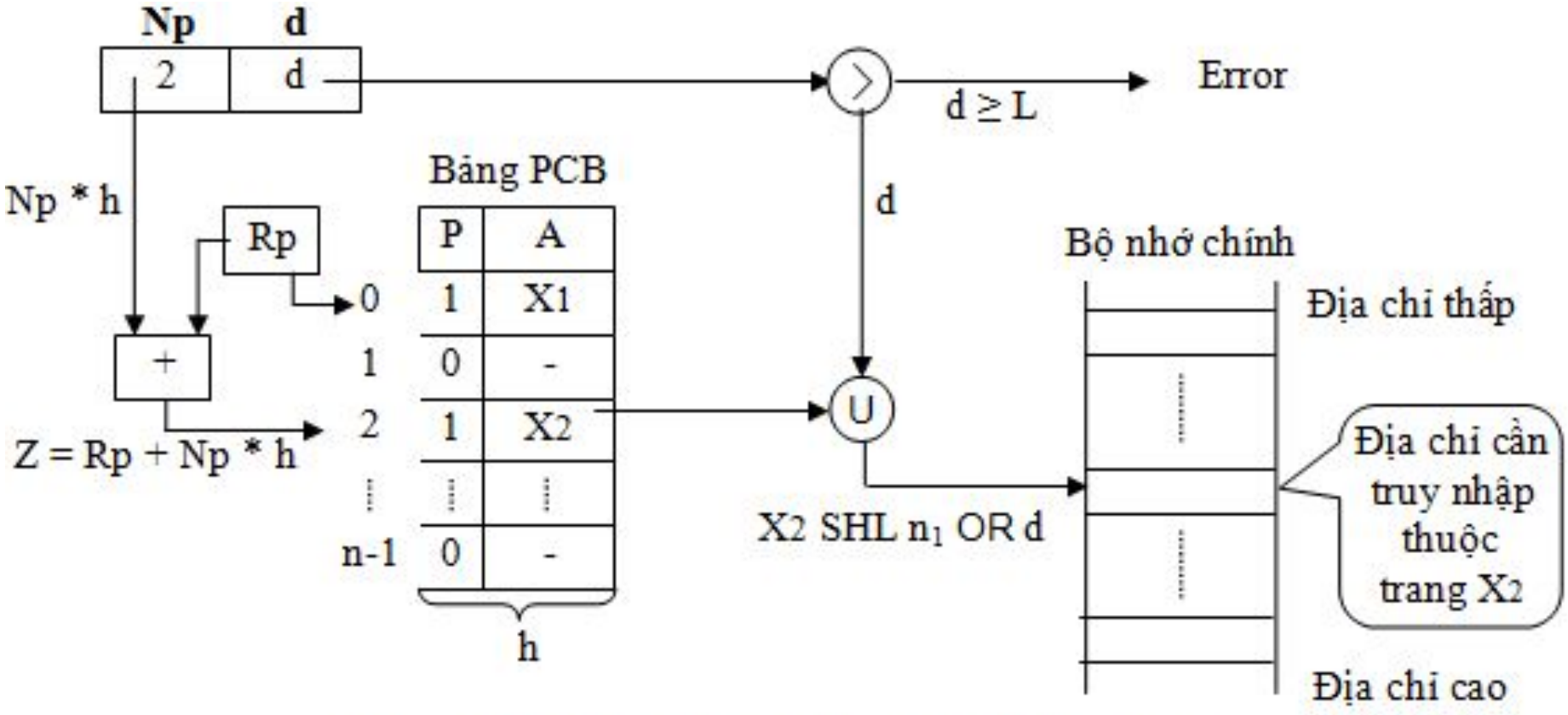
- Trường P
  - P = 0, trang chưa được nạp
  - P = 1, trang đã nạp
- Trường A: Cho biết trang vật lý nào đang nạp trang logic tương ứng. Nó chỉ có ý nghĩa khi P = 1.

Địa chỉ logic



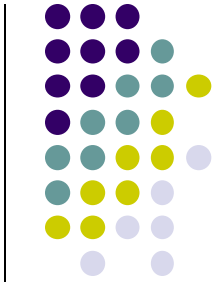
- Địa chỉ của lệnh hoặc dữ liệu được biểu diễn là bộ đôi giá trị <Np, d>, địa chỉ này được gọi là địa chỉ logic.
  - Np là số hiệu của trang logic cần truy nhập.
  - d là địa chỉ tương đối của lệnh hay dữ liệu trong trang.

Sơ đồ chuyển địa chỉ logic sang địa chỉ vật lý



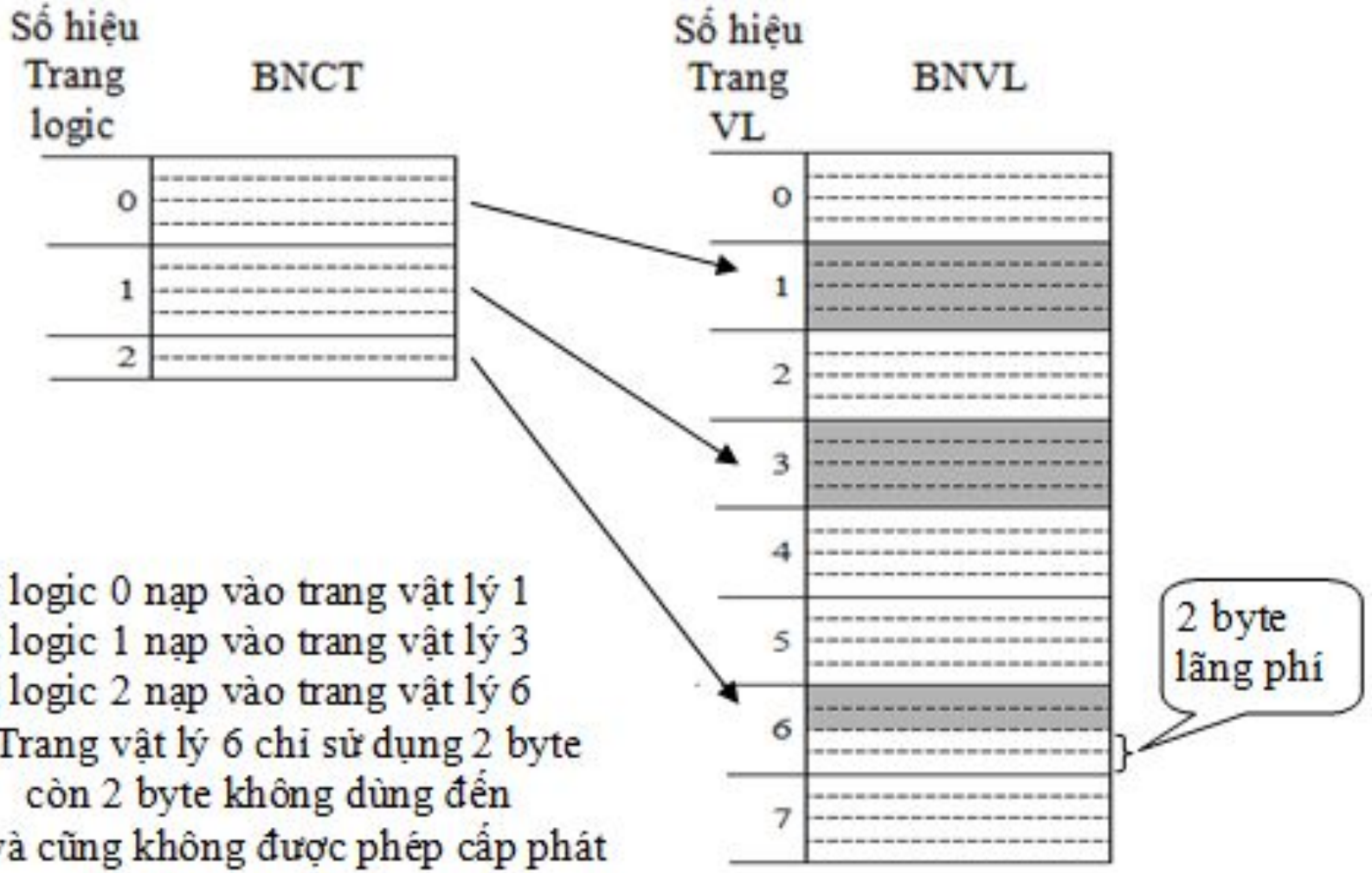
Sơ đồ cơ chế truy nhập bộ nhớ quản lý kiểu phân trang

Ưu nhược điểm của kỹ thuật phân trang



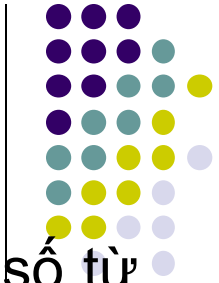
- Ưu điểm
  - Cho phép kích thước chương trình lớn hơn kích thước bộ nhớ còn tự do vẫn được thực thi (giống sơ đồ phân đoạn).
  - Không còn phân mảnh ngoại vi do cấp phát bộ nhớ theo trang.
  - Tốc độ truy cập bộ nhớ nhanh hơn so với sơ đồ phần đoạn vì phép cộng được thay thế bởi phép ghép.
- Nhược điểm
  - Vẫn tồn tại phân mảnh nội vì kích thước của trang logic cuối cùng trong chương trình thường nhỏ hơn kích thước của trang vật lý.
  - Cần có phần cứng hỗ trợ để định vị trang và chuyển địa chỉ logic sang vật lý

Minh họa phân mảnh nội khi phân trang



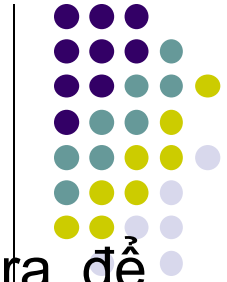
Trang logic 0 nạp vào trang vật lý 1
Trang logic 1 nạp vào trang vật lý 3
Trang logic 2 nạp vào trang vật lý 6
Vậy: Trang vật lý 6 chỉ sử dụng 2 byte
còn 2 byte không dùng đến
và cũng không được phép cấp phát

Kỹ thuật phân trang kết hợp với phân đoạn



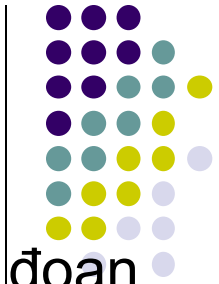
- Bộ nhớ vật lý được chia thành nhiều trang vật lý (đánh số từ 0, độ dài L) như kỹ thuật phân trang.
- Khi biên dịch, chương trình được biên dịch theo từng module và cũng sử dụng một bảng SCB để quản lý trạng thái của từng module như kỹ thuật phân đoạn.
- Mỗi module lại được chia thành nhiều trang logic, kích thước một trang logic bằng một trang vật lý.
- Khi một module được nạp hệ thống sinh ra một bảng PCB để quản lý trạng thái của từng trang logic trong module.
- Khi đoạn được giải phóng khỏi bộ nhớ chính (đưa ra vùng swap) thì bảng PCB tương ứng với nó cũng được giải phóng khỏi bộ nhớ.

Kỹ thuật phân trang kết hợp với phân đoạn (tiếp)



- Khi mỗi đoạn được nạp, một bảng PCB được sinh ra để quản lý trạng thái các trang trong đoạn.
- Địa chỉ đầu của bảng PCB được ghi trong trường Ap của bảng SCB dành cho đoạn.
- Bảng PCB có 2 trường là P, A:
  - P - trường dấu hiệu: cho biết trang đã nạp vào bộ nhớ chính hay chưa?
    - P = 0 : trang chưa được nạp, nằm ở vùng swap do hệ thống quản lý.
    - P = 1: trang đã được nạp vào bộ nhớ chính.
  - A – trường địa chỉ: cho biết số hiệu trang vật lý đang chứa trang logic, trường này chỉ có ý nghĩa khi P = 1.

Kỹ thuật phân trang kết hợp với phân đoạn (tiếp)

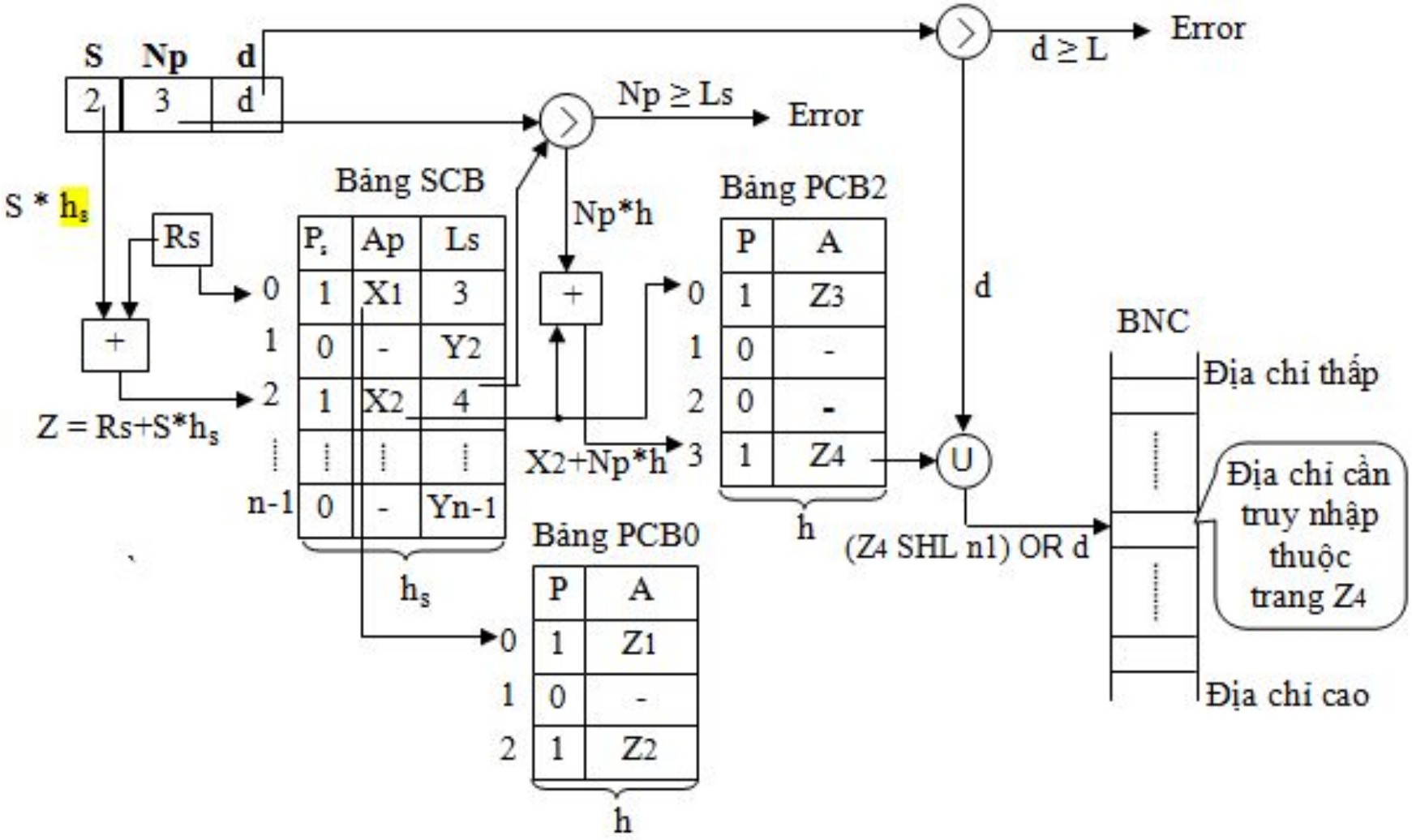


- Địa chỉ đầu của bảng SCB được quản lý nhờ thanh ghi đoạn Rs.
- Bảng SCB có 3 trường là Ps, Ap, Ls:
  - Ps - trường dấu hiệu: cho biết đoạn đã nạp vào bộ nhớ chính hay chưa?
    - Ps = 0 : đoạn chưa được nạp, nằm ở vùng swap do hệ thống quản lý.
    - Ps = 1: đoạn đã được nạp vào bộ nhớ chính.
  - Ap – trường địa chỉ: cho biết địa chỉ đầu của bảng PCB tương ứng với đoạn được nạp, trường này chỉ có ý nghĩa khi Ps = 1.
  - Ls – trường độ dài: cho biết số trang logic của đoạn.

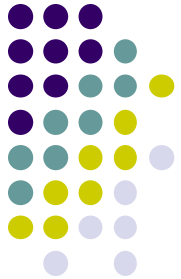
Địa chỉ logic

- S là số hiệu của đoạn cần truy nhập
- Np là số hiệu của trang logic cần truy nhập trong đoạn.
- d là địa chỉ tương đối của lệnh hay dữ liệu cần truy nhập trong trang.

Chuyển từ địa chỉ logic sang địa chỉ vật lý



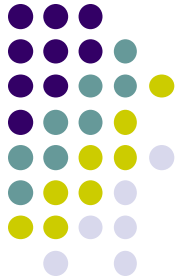
Sơ đồ cơ chế truy nhập bộ nhớ quản lý kiểu phân trang – phân đoạn kết hợp



- Cho bộ nhớ có dung lượng 512 byte. Dung lượng cache là 64 byte, chọn 4 line/1set. Chọn dung lượng của một line là 8 byte. Hãy xác định chi tiết đọc cache địa chỉ sau:

  2DEh

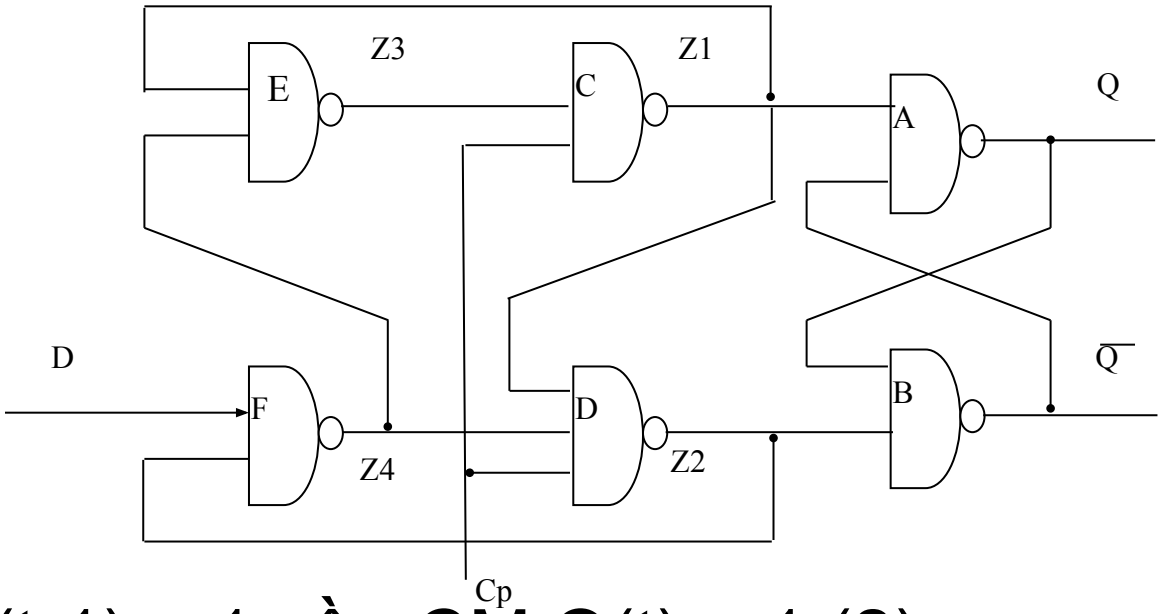
P	A	L
1	100h	200h
0	-	400h
1	300h	700h
0	-	500h
1	A00h	600h



Giả sử bộ nhớ vật lý có dung lượng 512 MB, chương trình gồm 5 module, xác định địa chỉ vật lý phát ra tương ứng với địa chỉ logic sau <2, 43h>, cho biết bảng quản lý phân đoạn như sau:

Xét: Q(t-1) = 0 cần cm Q(t) = 0 (1).
Z1(t) = NAND(Z3(t),Cp) = 1.
Q(t-1) = 0 thì /Q(t-1) = 1
Q(t) = NAND(Z1(t),/Q(t-1)) = NAND(1,1) = 0 (1)

•Cho sơ đồ mạch Flip-Flop sau, chứng minh rằng khi Cp = 0, trạng thái đầu ra Q không đổi

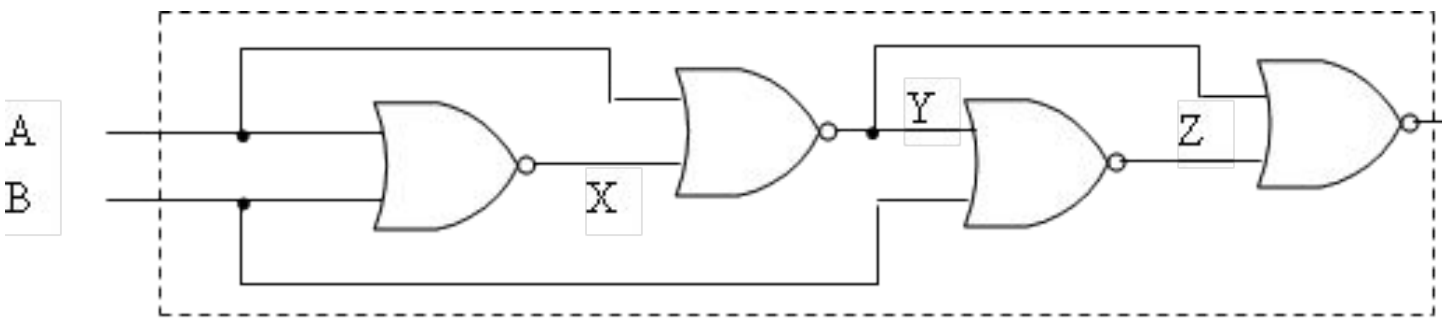


Xét Q(t-1) = 1 cần CM Q(t) = 1 (2)

Q(t-1) = 1 □/Q(t-1) = 0

Q(t) = NAND(Z1(t),/Q(t-1)) = 1 (2)

Từ (1) và (2) □Q(t) = Q(t-1) □Q không thay đổi



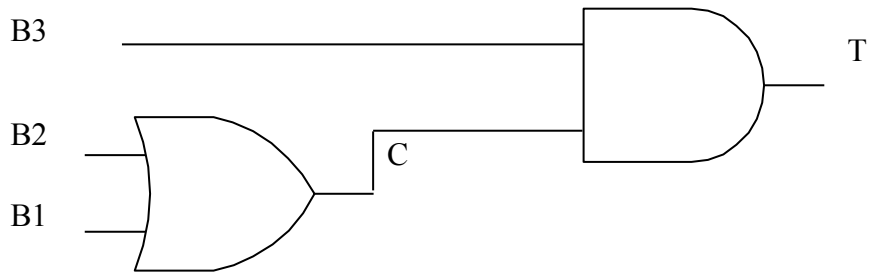
Dựa bảng chân lý của mạch AND

A	B	K
0	0	0
0	1	0
1	0	0
1	1	1

X = NOR(A,B)
Y = NOR(A,X)
Z = NOR(B,Y)
K = NOR(Y,Z)

A	B	X	Y	Z	K
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

KL: Đầu ra của mạch trùng với bảng chân lý của mạch AND Mạch trên là mạch AND 2 đầu vào A và B, đầu ra K



- Mạch logic được thiết kế để phát hiện lỗi trong mã BCD. Lối vào là 3
- bit cao của mã BCD, lối ra ở trạng thái 1 khi có lỗi.

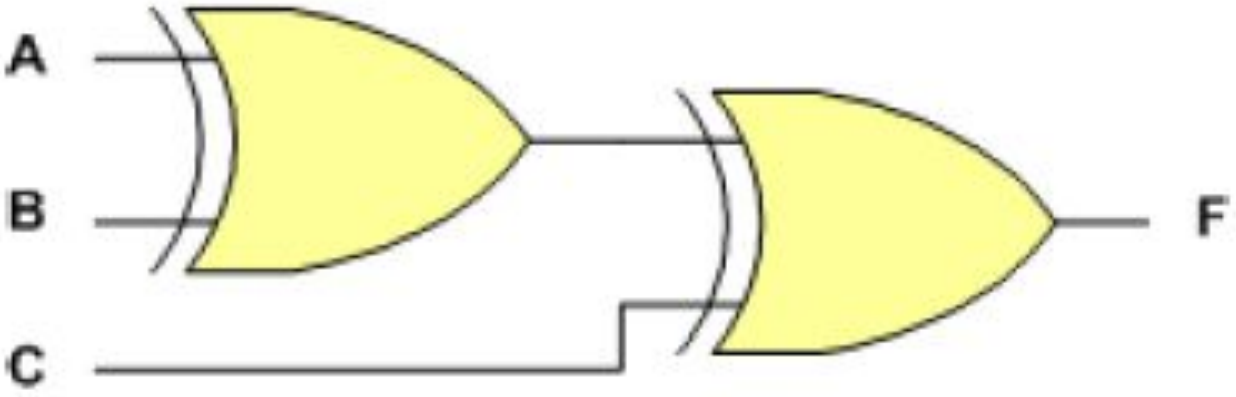
Chứng minh rằng B3 = 0, B2 = 0, B1 = 0, B0 = 0 hoặc B0 = 1 thì T = 0 (mã BCD không lỗi)

C = OR(B2,B1), T = AND(B3,C)

$B_3B_2B_1B_0$ Nếu B3 = 0 thì $0B_2B_1B_0$ tối đa là 7.

C = OR(0,0) = 0.

T = AND(0,0) = 0



A	B	C	F (lẻ)	Chẵn
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0