Chương 8

Bộ nhớ

(Memory)

Nội dung

- Tổng quan về hệ thống nhớ
  - Bộ nhớ bán dẫn
  - Bộ nhớ chính
  - Bộ nhớ cache
  - Bộ nhớ ngoài
  - Bộ nhớ ảo
- Các giải thuật thay thế trang

Tổng quan về hệ thống nhớ

- 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 lưu trữ
  - Dung lượng
    - Độ dài từ nhớ (tính bằng bit)
    - Số lượng từ nhớ

Tổng quan về hệ thống nhớ

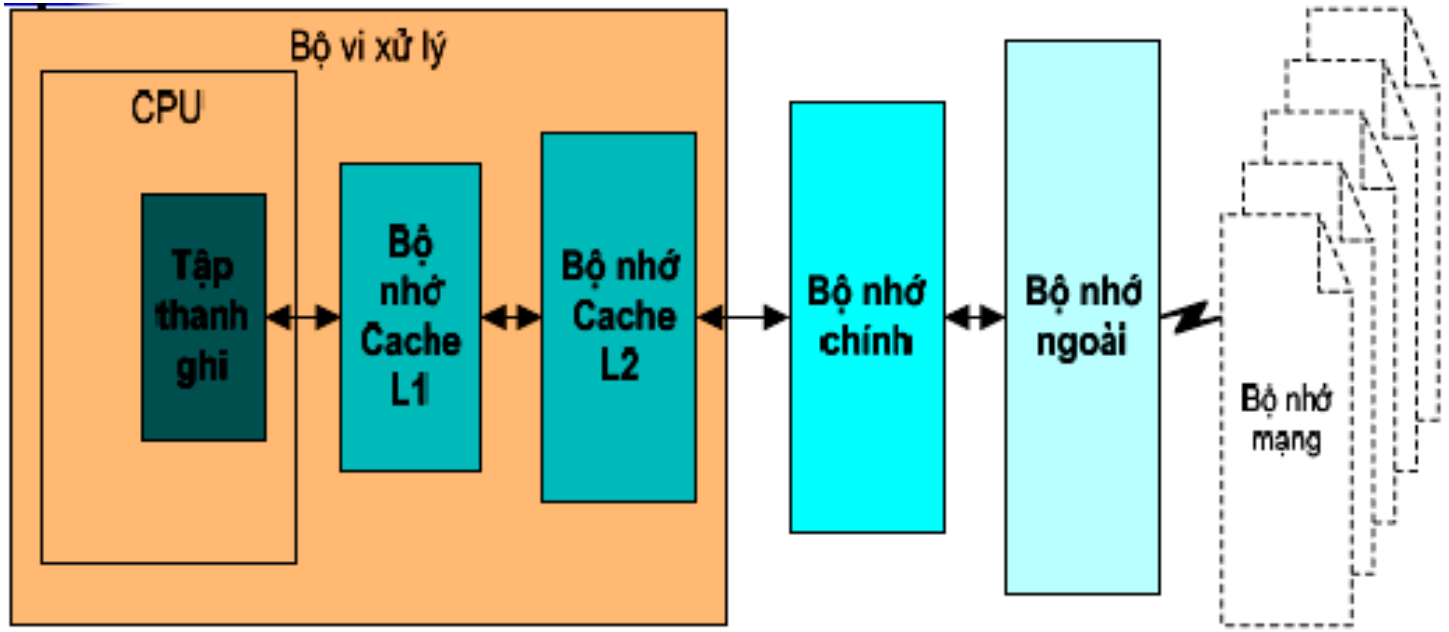
- Các đặc trưng của hệ thống nhớ (tiếp)
  - Đơn vị truyền
    - Từ nhớ (word)
    - Khối nhớ (block)
  - Phương pháp truy cập
    - Truy cập tuần tự (băng từ)
    - Truy cập trực tiếp (các loại đĩa)
    - Truy cập ngẫu nhiên (bộ nhớ bán dẫn)
    - Truy cập kết hợp (cache)

Tổng quan về hệ thống nhớ

- Các đặc trưng của hệ thống nhớ (tiếp)
  - Hiệu năng (performance)
    - Thời gian truy cập
    - Tốc độ truyền
  - Kiểu vật lý
    - Bộ nhớ bán dẫn
    - Bộ nhớ từ
    - Bộ nhớ quang
  - Các đặc tính vật lý
    - Tự mất/ Không tự mất (volatile/ nonvolatile)
    - Xoá được/ không xoá được

Tổng quan về hệ thống nhớ

- Phân cấp hệ thống nhớ



Từ trái sang phải:

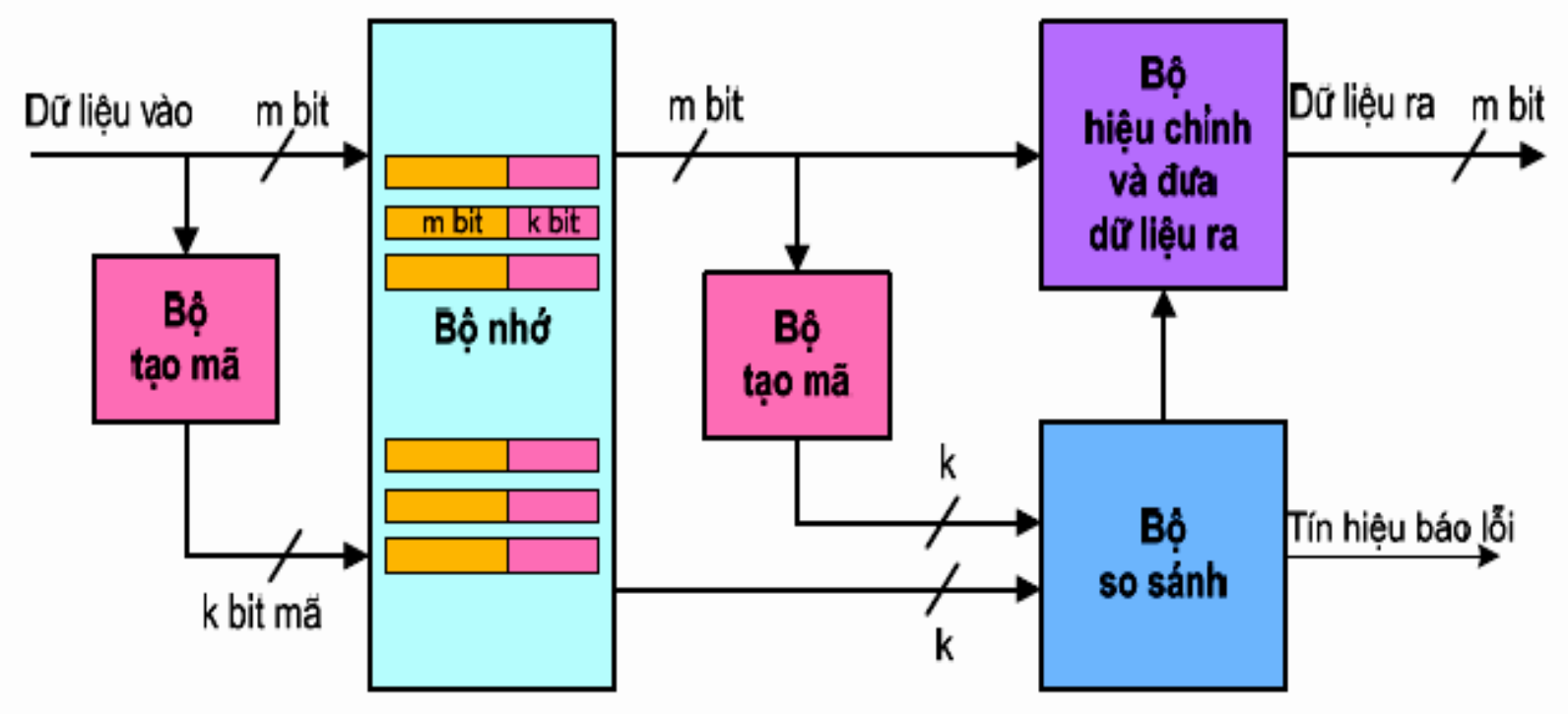
- dung lượng tăng dần
- tốc độ giảm dần
- giá thành/1bit giảm dần

Tổng quan về hệ thống nhớ

- Độ tin cậy bộ nhớ
  - Nguyên tắc chung: cần tạo ra và lưu trữ thêm thông tin dư thừa.
    - Từ dữ liệu cần ghi vào bộ nhớ: m bit
    - Cần tạo ra và lưu trữ từ mã: k bit
    - Lưu trữ (m+k) bit
  - Phát hiện lỗi
    - Kiểm tra chẵn/ lẻ (parity): Mỗi byte dữ liệu cần 1 bit kiểm tra
    - Checksum
    - CRC (Cyclic Redandancy Check)
  - Phát hiện và sửa lỗi
    - Dữ liệu được mã hoá bằng các bộ mã có khả năng sửa lỗi ECC (Error Correction Code), ví dụ : Mã Hamming
    - Mỗi byte hoặc block dữ liệu cần nhiều bit kiểm tra hơn

Tổng quan về hệ thống nhớ

- Độ tin cậy bộ nhớ (tiếp)



Bộ nhớ bán dẫn

- Phân loại
  - ROM (Read Only Memory)
    - Bộ nhớ chỉ đọc
    - Không tự mất dữ liệu khi cắt nguồn điện
  - RAM (Random Access Memory)
    - Bộ nhớ đọc/ ghi
    - Tự mất dữ liệu khi cắt nguồn điện
  - Cache
    - Bộ nhớ có tốc độ cao nhưng dung lượng thấp
    - Trung gian giữa bộ nhớ chính và thanh ghi trong CPU
    - Ngày nay thường được tích hợp sẵn trong CPU

Bộ nhớ bán dẫn

- ROM
  - Thông tin được ghi khi sản xuất
  - Không xoá/ sửa được nội dung khi sử dụng
  - Ứng dụng:
    - Thư viện các chương trình con
    - Các chương trình điều khiển hệ thống nhập xuất cơ bản BIOS (Basic Input Output System)
    - Phần mềm kiểm tra khi bật máy POST (Power On Self Test)
    - Phần mềm khởi động máy tính (OS loader)
    - Vi chương trình

Bộ nhớ bán dẫn

- Phân loại ROM
  - Mask ROM
    - Thông tin được ghi khi sản xuất
    - Không xoá/ sửa được nội dung
    - Giá thành rất đắt
  - PROM (Programmable ROM)
    - Khi sản xuất chưa có nội dung (ROM trắng)
    - Cần thiết bị chuyên dụng để ghi
    - Cho phép ghi được một lần, gọi là OTP (One Time Programmable) hoặc WORM (Write-Once-Read-Many)
  - EPROM (Erasable PROM)
    - Có thể xóa bằng tia cực tím UV (Ultra Violet)
    - Cần thiết bị chuyên dụng để ghi
    - Ghi/ xoá được nhiều lần

Bộ nhớ bán dẫn

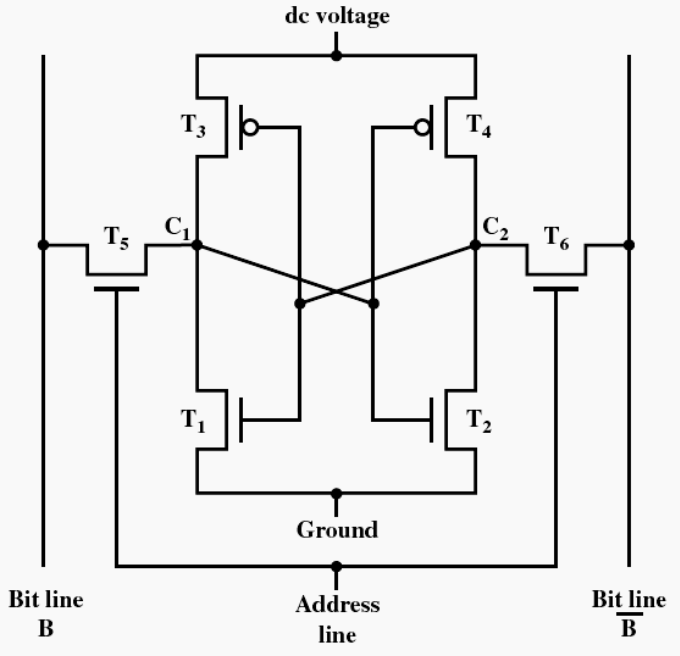
- Phân loại ROM (tiếp)
  - EEPROM (Electrically EPROM)
    - Xóa bằng mạch điện, không cần tia UV → Không cần tháo chip ROM ra khỏi máy tính
    - Có thể ghi theo từng byte
    - 2 chế độ điện áp:
      - Điện áp cao : Ghi + Xoá
      - Điện áp thấp : Chỉ đọc
  - Flash memory (Bộ nhớ cực nhanh)
    - EEPROM sản xuất bằng công nghệ NAND, tốc độ truy cập nhanh, mật độ cao
    - Xóa bằng mạch điện; Ghi theo từng block
    - Ngày nay được sử dụng rộng rãi dưới dạng thẻ nhớ (CF, SD,…) , thanh USB, ổ SSD (thay thế cho ổ đĩa cứng)

Bộ nhớ bán dẫn

- RAM
  - Bộ nhớ đọc-ghi (Read/Write Memory)
  - Có thể ghi/ xoá trong quá trình sử dụng → Làm bộ nhớ chính trong máy tính
  - Tự mất dữ liệu khi cắt nguồn điện. Chỉ lưu trữ thông tin tạm thời khi chạy chương trình, khi kết thúc chương trình cần lưu trữ dữ liệu ra bộ nhớ ngoài
  - Có hai loại:
    - SRAM (Static RAM): RAM tĩnh
    - DRAM (Dynamic RAM): RAM động

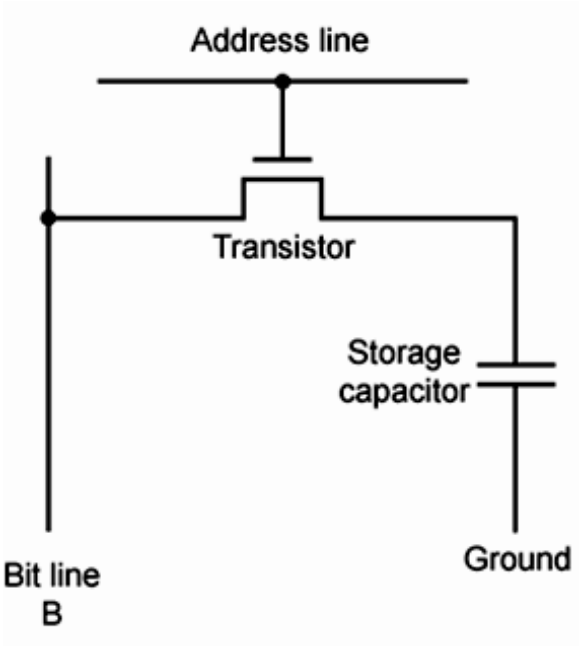
Bộ nhớ bán dẫn

- SRAM
  - Các bit được lưu trữ bằng các Flip-Flop
  - Thông tin ổn định, không tự mất dữ liệu theo thời gian
  - Cấu trúc phức tạp
  - Dung lượng chip nhỏ
  - Tốc độ truy cập nhanh
  - Đắt tiền
  - Dùng làm bộ nhớ cache



Bộ nhớ bán dẫn

- DRAM
  - Các bit được lưu trữ trên mạch tụ điện
  - Tự mất dữ liệu theo thời gian → cần phải có mạch làm tươi (refresh)
  - Cấu trúc đơn giản
  - Dung lượng lớn
  - Tốc độ chậm hơn
  - Rẻ tiền hơn
  - Dùng làm bộ nhớ chính

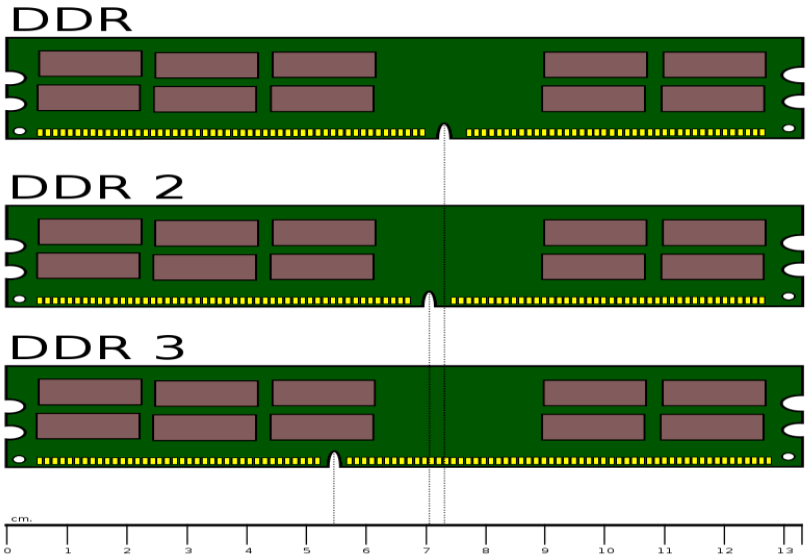


Bộ nhớ bán dẫn

- Phân loại DRAM theo cơ chế hoạt động
  - FPM (Fast Page Mode)
    - Truy cập theo từng trang bộ nhớ (cùng hàng khác cột)
  - EDO (Enhanced Data Out)
    - Khi xuất dữ liệu có thể đồng thời đọc địa chỉ của ô nhớ kế tiếp
    - Cho phép đọc nhanh gấp đôi so với RAM thường
  - SDRAM (Synchronous DRAM)
    - Đồng bộ với system clock → CPU không cần chu kỳ chờ
    - Truyền dữ liệu theo block
  - RDRAM (Rambus DRAM)
    - Bộ nhớ tốc độ cao, truyền dữ liệu theo block
    - Do công ty Rambus và Intel sản xuất để sử dụng cho CPU Pentium 4 khi mới xuất hiện năm 2000
    - Giá thành đắt nên ngày nay ít sử dụng

Bộ nhớ bán dẫn

- Phân loại DRAM theo cơ chế hoạt động (tiếp)
  - DDR-SDRAM (Double Data Rate-SDRAM)
    - Phiên bản cải tiến của SDRAM nhằm nâng cao tốc độ truy cập nhưng có giá thành rẻ hơn RDRAM
    - Gởi dữ liệu 2 lần trong 1 chu kỳ clock
  - DDR2/ DDR3: Gởi dữ liệu 4 hoặc 8 lần trong 1 chu kỳ clock



Bộ nhớ bán dẫn

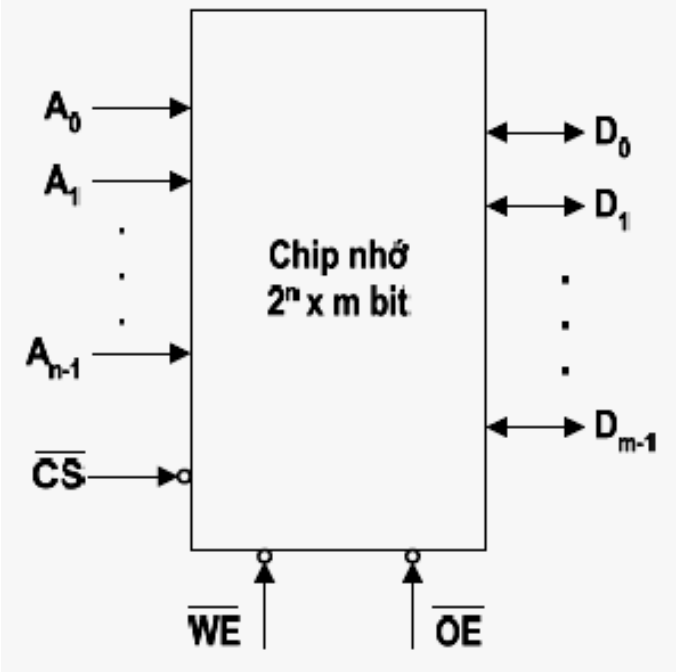
- Phân loại DRAM theo hình thức đóng gói
  - SIMM (Single Inline Memory Module)
  - DIMM (Dual Inline Memory Module)
  - RIMM (Rambus Inline Memory Module)
  - SO-DIMM (Small Outline DIMM)
  - SO-RIMM (Small Outline RIMM)





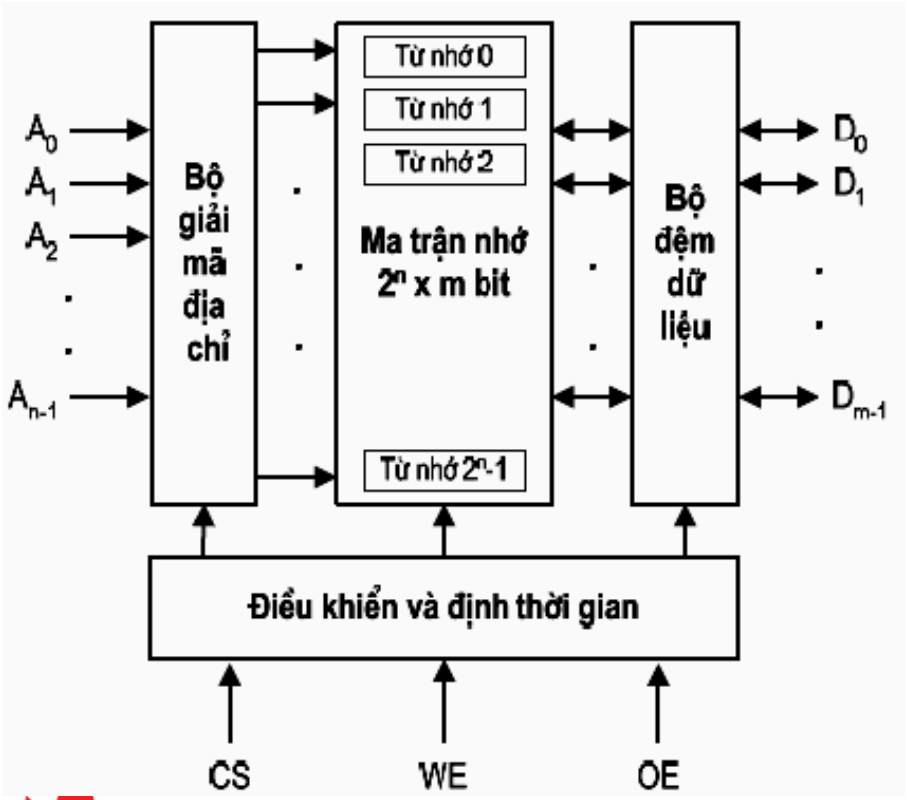
Bộ nhớ bán dẫn

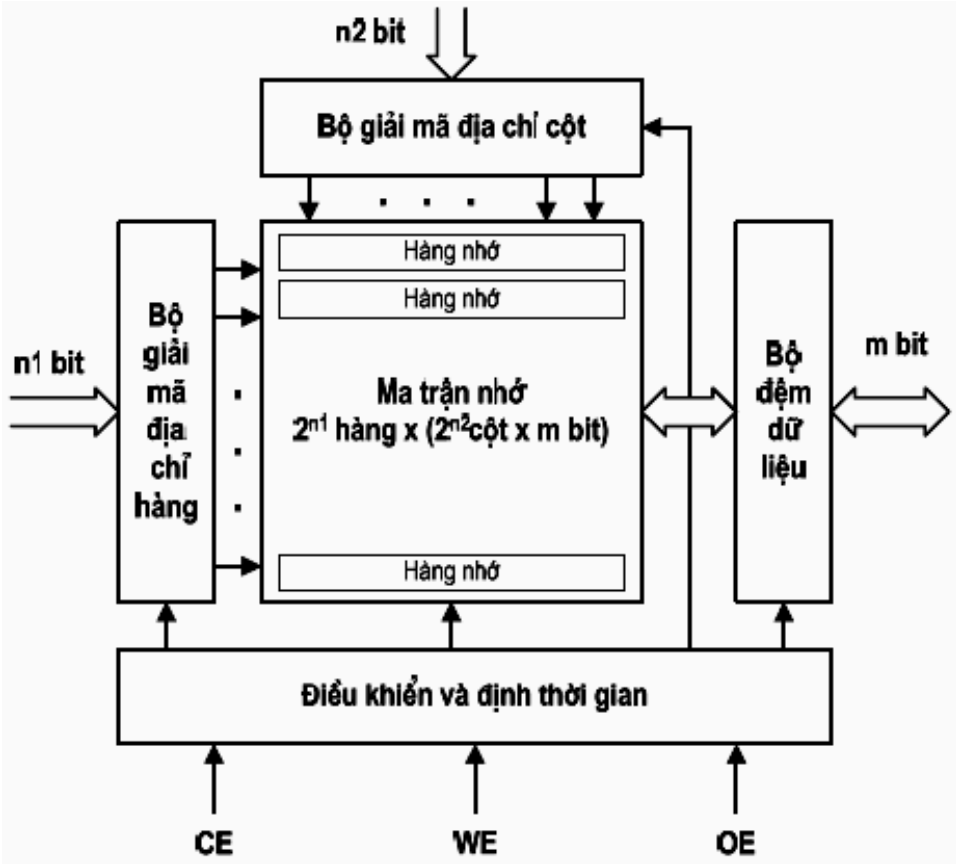
- Tổ chức của chip nhớ
  - Các đường địa chỉ: $A_{n-1} \div A_0$ → có 2^n từ nhớ
  - Các đường dữ liệu: $D_{m-1} \div D_0$ → độ dài từ nhớ = m bit
  - Dung lượng chip nhớ = 2^n * m bit
  - Các đường điều khiển:
    - Tín hiệu chọn chip CS (Chip Select)
    - Tín hiệu điều khiển đọc OE (Output Enable)
    - Tín hiệu điều khiển ghi WE (Write Enable)
    - Các tín hiệu điều khiển thường tích cực với mức 0



Bộ nhớ bán dẫn

- Tổ chức bộ nhớ 1 chiều và 2 chiều



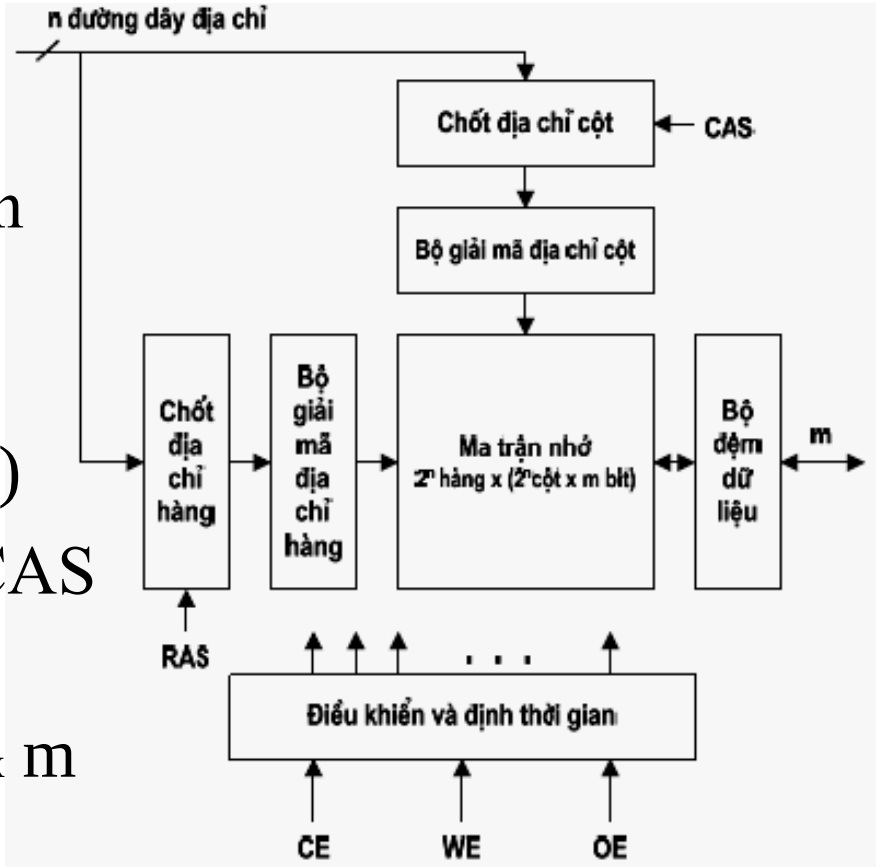


Bộ nhớ bán dẫn

- Tổ chức bộ nhớ hai chiều
  - Có n đường địa chỉ: n = n1 + n2
    - 2^{n1} hàng,
    - mỗi hàng có 2^{n2} từ nhớ,
  - Có m đường dữ liệu:
    - mỗi từ nhớ có độ dài m-bit.
  - Dung lượng của chip nhớ:
    - $[2^{n1} \text{ x } (2^{n2} \text{ x m})]$ bit = (2^{n1+n2} x m) bit = (2^{n} x m) bit.
  - Hoạt động giải mã địa chỉ:
    - Bước 1: bộ giải mã hàng chọn 1 trong 2^{n1} hàng.
    - Bước 2: bộ giải mã cột chọn 1 trong 2^{n2} từ nhớ (cột) của hàng đã được chọn.

Bộ nhớ bán dẫn

- Tổ chức của DRAM
  - Dùng n đường địa chỉ dồn kênh → cho phép truyền 2n bit địa chỉ
  - Tín hiệu chọn địa chỉ hàng RAS (Row Address Strobe)
  - Tín hiệu chọn địa chỉ cột CAS (Column Address Strobe)
  - Dung lượng DRAM = 2^{2n} x m bit



Bộ nhớ bán dẫn

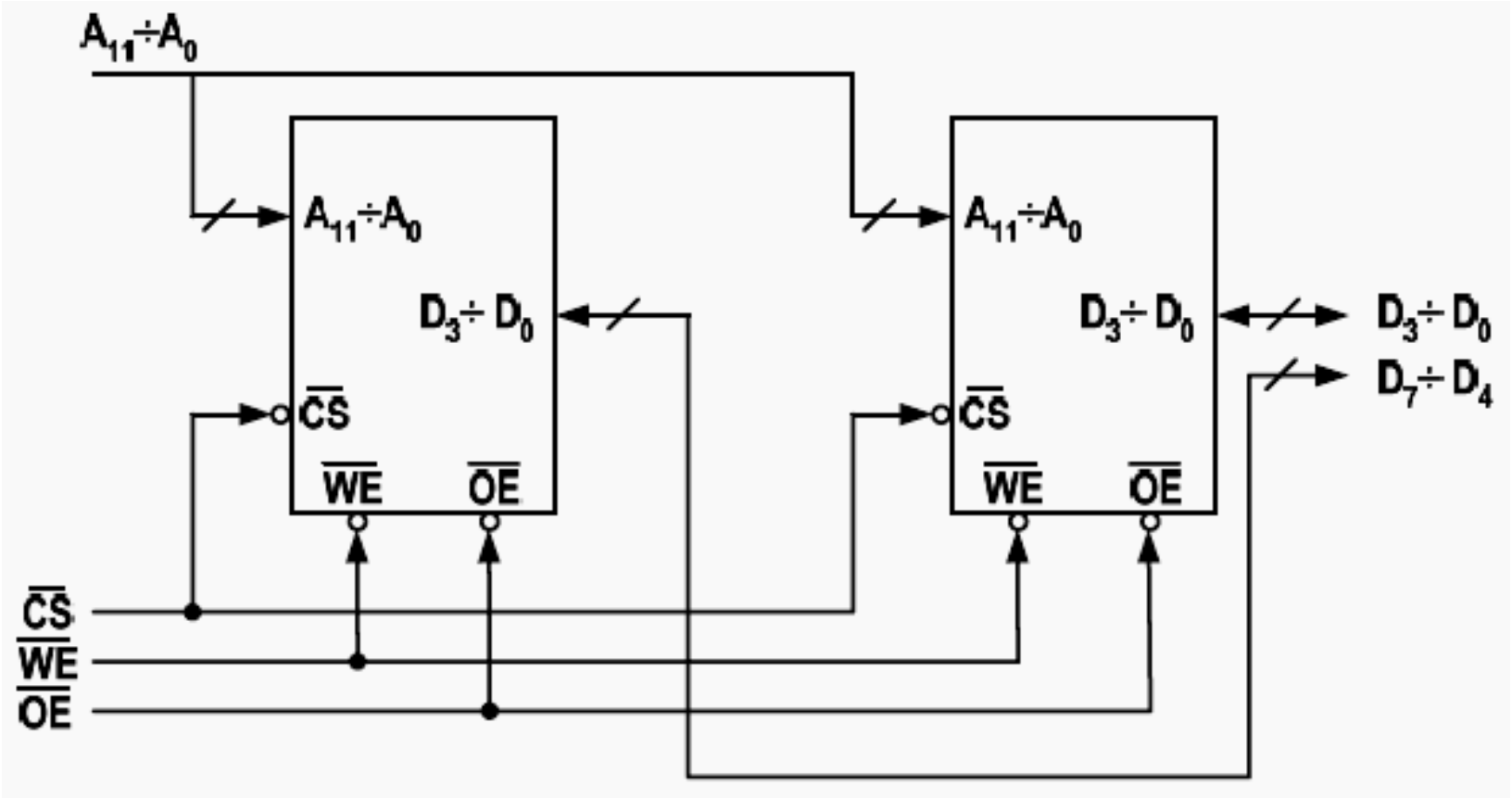
- Thiết kế mô-đun nhớ bán dẫn
  - Dung lượng chip nhớ 2^n x m bit
  - Cần thiết kế để tăng dung lượng:
    - Thiết kế tăng độ dài từ nhớ
    - Thiết kế tăng số lượng từ nhớ
    - Thiết kế kết hợp: tăng cả độ dài và số lượng từ nhớ
  - Qui tắc: ghép nối các chip nhớ song song (tăng độ dài) hoặc nối tiếp bằng mạch giải mã (tăng số lượng)

Bộ nhớ bán dẫn

- Tăng độ dài từ nhớ
  - VD:
    - 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
  - Tổng quát
    - Cho chip nhớ 2^n x m bit
    - Thiết kế mô-đun nhớ 2^n x (k.m) bit
    - Dùng k chip nhớ

Bộ nhớ bán dẫn

- Ví dụ: Tăng độ dài từ nhớ

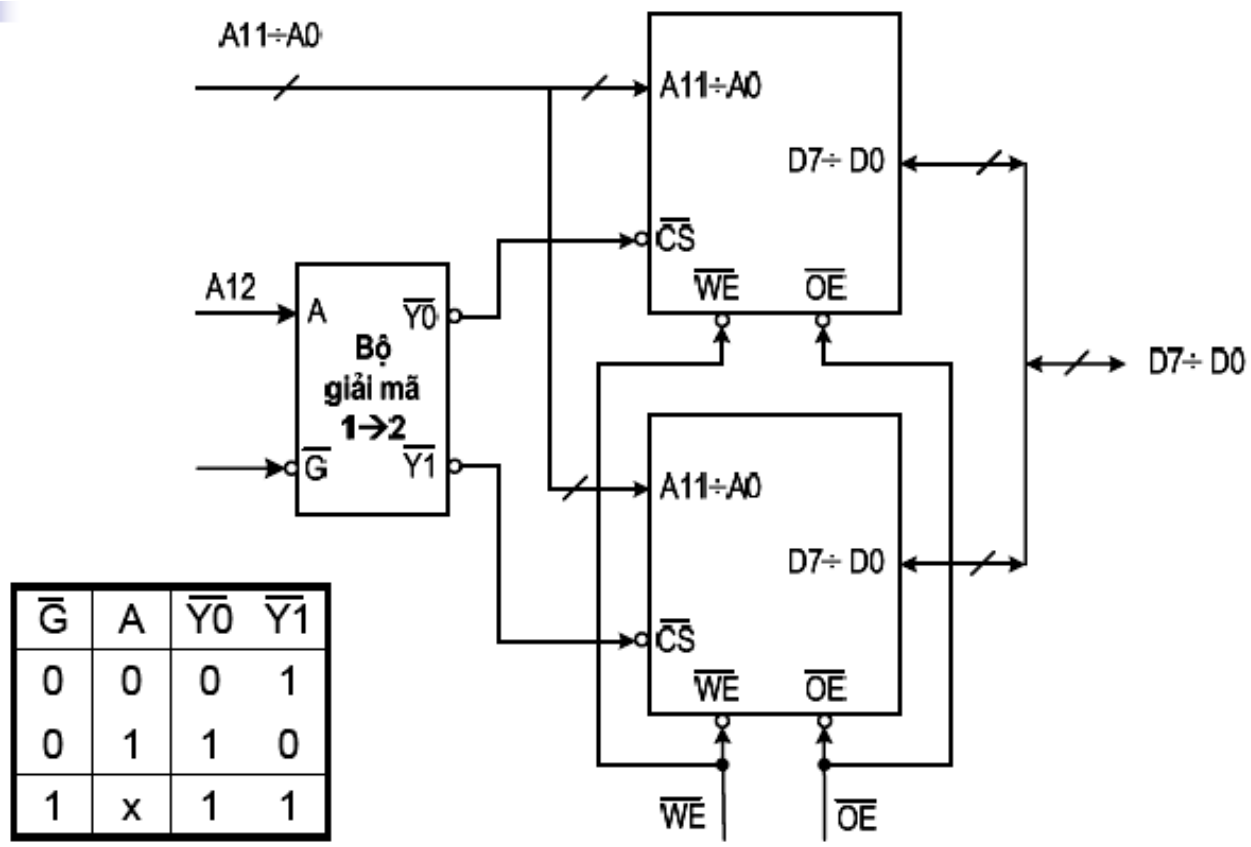


Bộ nhớ bán dẫn

- Tăng số lượng từ nhớ
  - VD:
    - 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

Bộ nhớ bán dẫn

- Ví dụ: Tăng số lượng từ nhớ



G	A	Y0	Y1
0	0	0	1
0	1	1	0
1	x	1	1

Bộ nhớ bán dẫn

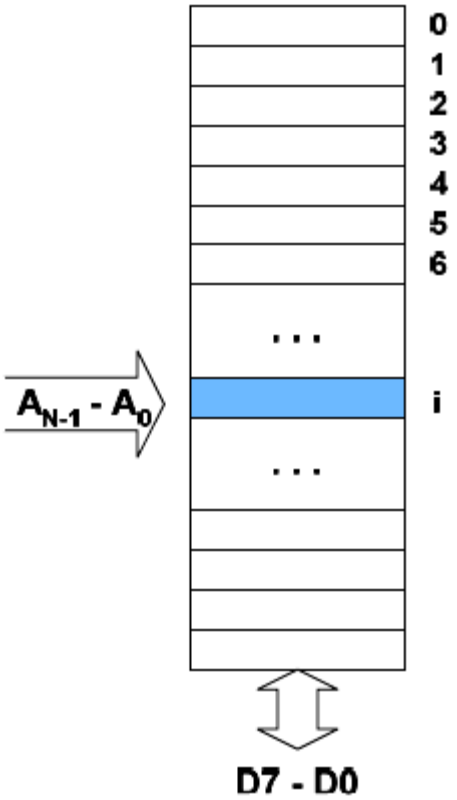
- Bài tập
  - Tăng số lượng từ gấp 4 lần:
    - Cho chip nhớ SRAM 4K x 8 bit
    - Thiết kế mô-đun nhớ 16K x 8 bit
    - Gợi ý: Dùng mạch giải mã 2 → 4
  - Tăng số lượng từ gấp 8 lần:
    - Cho chip nhớ SRAM 4K x 8 bit
    - Thiết kế mô-đun nhớ 32K x 8 bit
    - Gợi ý: Dùng mạch giải mã 3 → 8
  - Thiết kế kết hợp:
    - Cho chip nhớ SRAM 4K x 4 bit
    - Thiết kế mô-đun nhớ 8K x 8 bit

Bộ nhớ chính

- Các đặc trưng cơ bản
  - Chứa các chương trình đang thực hiện và các dữ liệu đang được sử dụng
  - Tồn tại trên mọi hệ thống máy tính
  - Bao gồm các ô nhớ được đánh địa chỉ trực tiếp bởi CPU
  - Dung lượng của bộ nhớ chính trên thực tế thường nhỏ hơn không gian địa chỉ bộ nhớ mà CPU có thể quản lý.
  - Việc quản lý logic bộ nhớ chính tuỳ thuộc vào hệ điều hành → người lập trình chỉ sử dụng bộ nhớ logic.

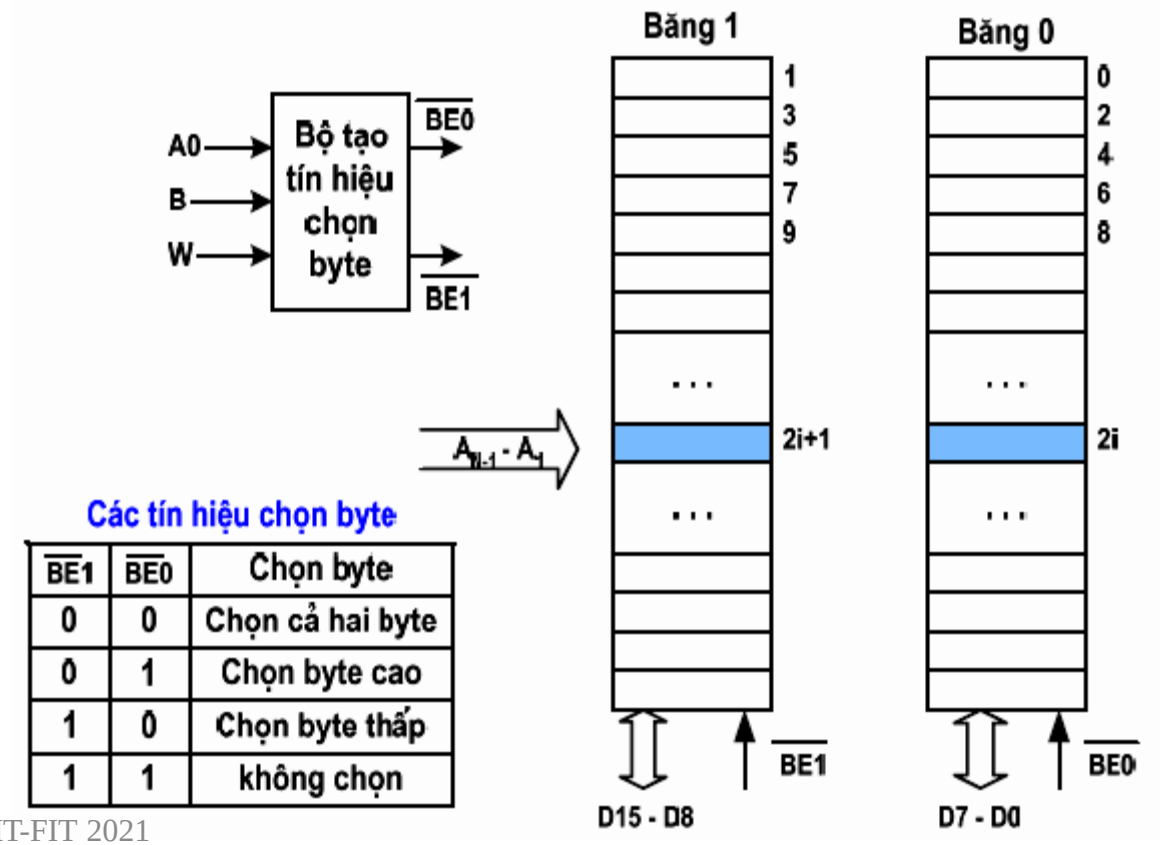
Bộ nhớ chính

- Tổ chức bộ nhớ đan xen (interleaved memory)
  - Độ rộng của bus dữ liệu để trao đổi với bộ nhớ: m = 8, 16, 32, 64,128 bit ...
  - Các ô nhớ được tổ chức theo byte → tổ chức bộ nhớ vật lý khác nhau
  - m=8 bit → một băng nhớ tuyến tính



Bộ nhớ chính

- Tổ chức bộ nhớ đan xen (tiếp)
  - m = 16 bit → hai băng nhớ đan xen

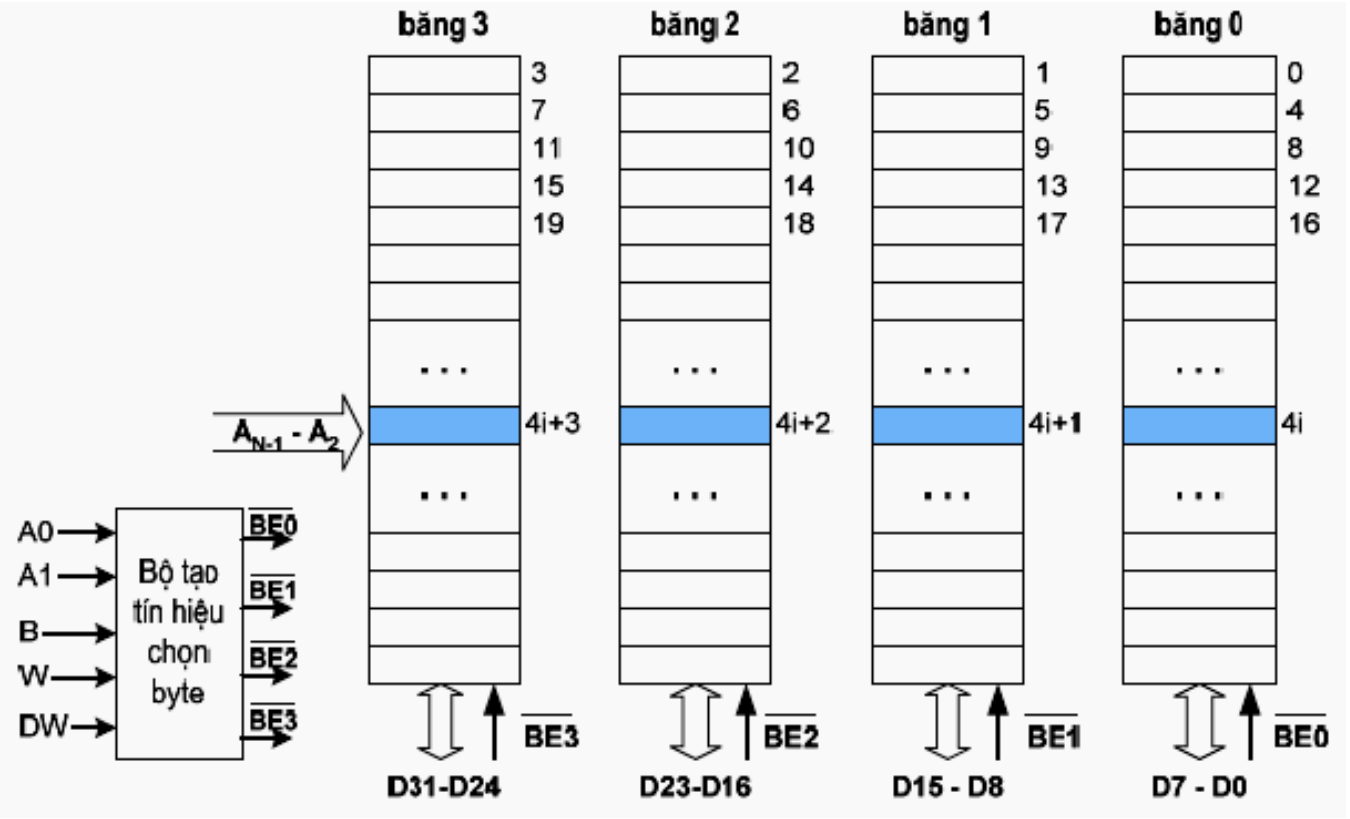


Các tín hiệu chọn byte

$\overline{BE1}$	$\overline{BE0}$	Chọn byte
0	0	Chọn cả hai byte
0	1	Chọn byte cao
1	0	Chọn byte thấp
1	1	không chọn

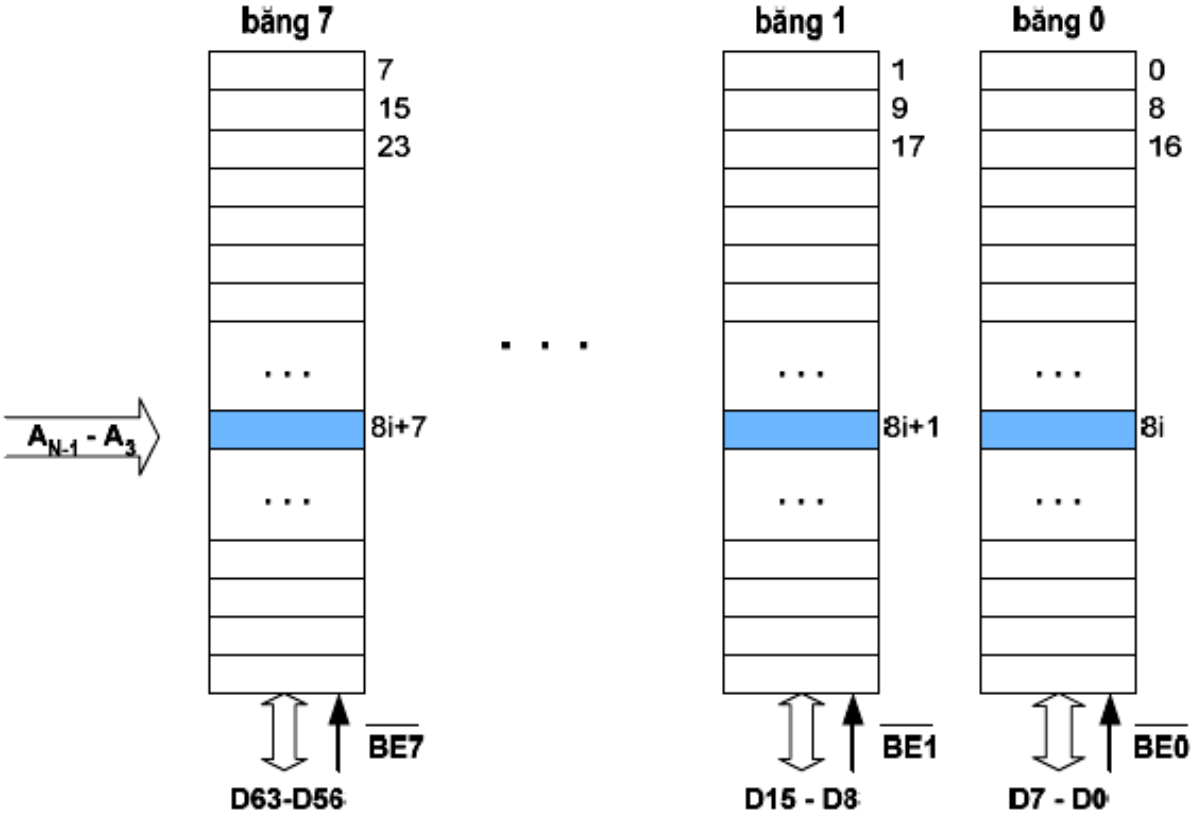
Bộ nhớ chính

- Tổ chức bộ nhớ đan xen (tiếp)
  - m = 32 bit → bốn băng nhớ đan xen



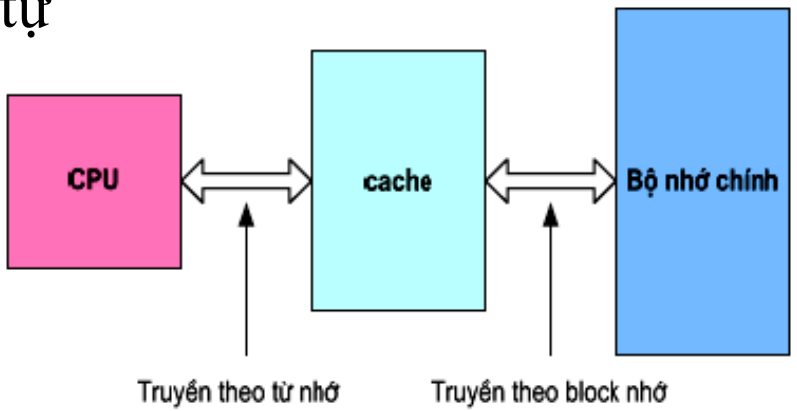
Bộ nhớ chính

- Tổ chức bộ nhớ đan xen (tiếp)
  - m = 64 bit → tám băng nhớ đan xen



Bộ nhớ cache

- Nguyên tắc chung của cache
  - Nguyên lý cục bộ : Một chương trình thường sử dụng 90% thời gian chỉ để thi hành 10% câu lệnh
  - Cache được đặt giữa CPU và bộ nhớ chính nhằm tăng tốc độ truy cập bộ nhớ của CPU
  - Ví dụ:
    - Cấu trúc chương trình tuần tự
    - Vòng lặp có thân nhỏ
    - Cấu trúc dữ liệu mảng

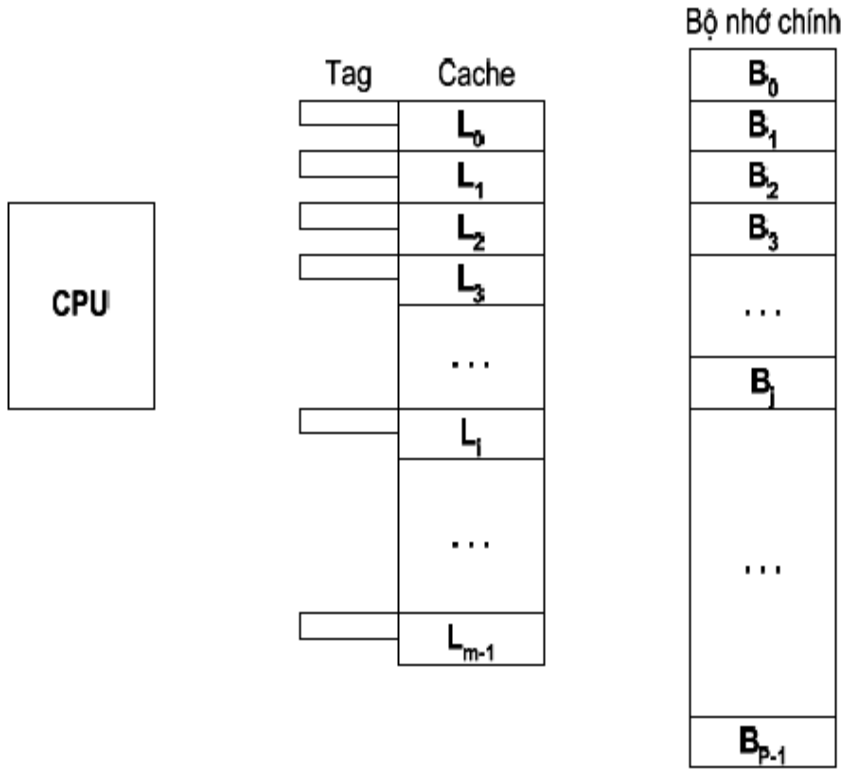


Bộ nhớ cache

- Thao tác trên bộ nhớ cache
  - CPU yêu cầu nội dung của ô nhớ
  - CPU kiểm tra trên cache với dữ liệu này
  - Nếu có, CPU nhận dữ liệu từ cache (nhanh)
  - Nếu không có thực hiện 2 bước sau:
    - Đọc Block chứa dữ liệu từ bộ nhớ chính vào cache (chậm hơn)
    - Chuyển dữ liệu từ cache vào CPU

Bộ nhớ cache

- Cấu trúc chung của cache
  - 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, ... , B_{p-1} (p Blocks)
    - Bộ nhớ cache: L_0, L_1, L_2, ... , L_{m-1} (m Lines)
    - Kích thước của Block = 8,16,32,64,128 byte



Bộ nhớ cache

- Cấu trúc chung của cache (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.
  - Khi CPU truy cập (đọc/ghi) một từ nhớ, có hai khả năng xảy ra:
    - Từ nhớ đó có trong cache (cache hit)
    - Từ nhớ đó không có trong cache (cache miss)

Bộ nhớ cache

- Tổ chức bộ nhớ cache
  - Kích thước cache
    - Cache càng lớn càng hiệu quả nhưng đắt tiền
    - Cần nhiều thời gian để giải mã và truy cập
  - Kỹ thuật ánh xạ : Phương pháp xác định vị trí dữ liệu trong cache
    - Ánh xạ trực tiếp (Direct mapping)
    - Ánh xạ kết hợp toàn phần (Fully associative mapping)
    - Ánh xạ kết hợp theo bộ (Set associative mapping)
  - Giải thuật thay thế
    - Phương pháp chọn lựa vùng nhớ nào lưu trong cache để tăng hiệu suất sử dụng cache

Bộ nhớ cache

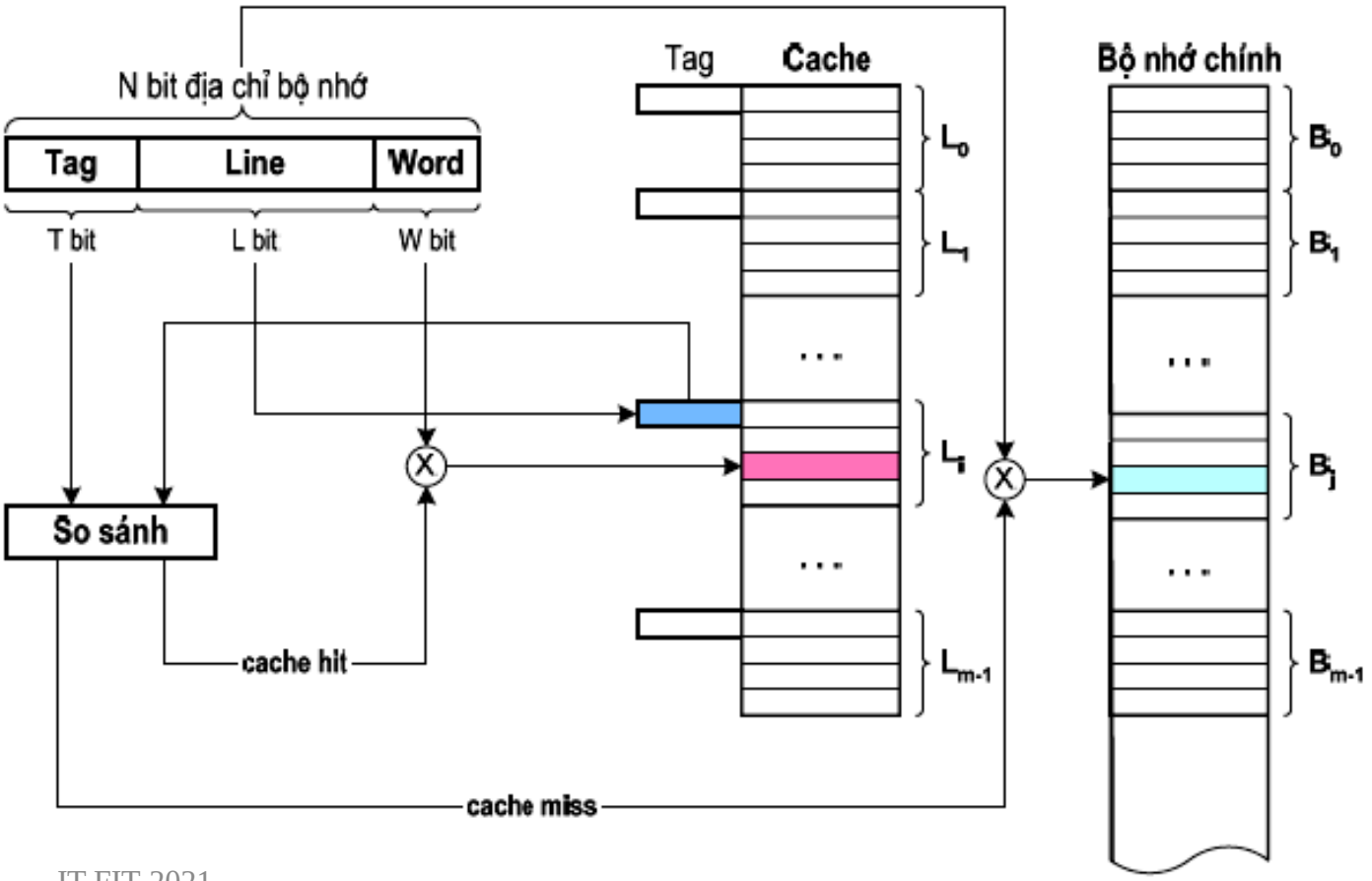
- Ánh xạ trực tiếp
  - Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một Line của cache:
    - $B_0 \rightarrow L_0$
    - $B_1 \rightarrow L_1$
    - ....
    - $B_{m-1} \rightarrow L_{m-1}$
    - $B_m \rightarrow L_0$
    - $B_{m+1} \rightarrow L_1$
    - ....

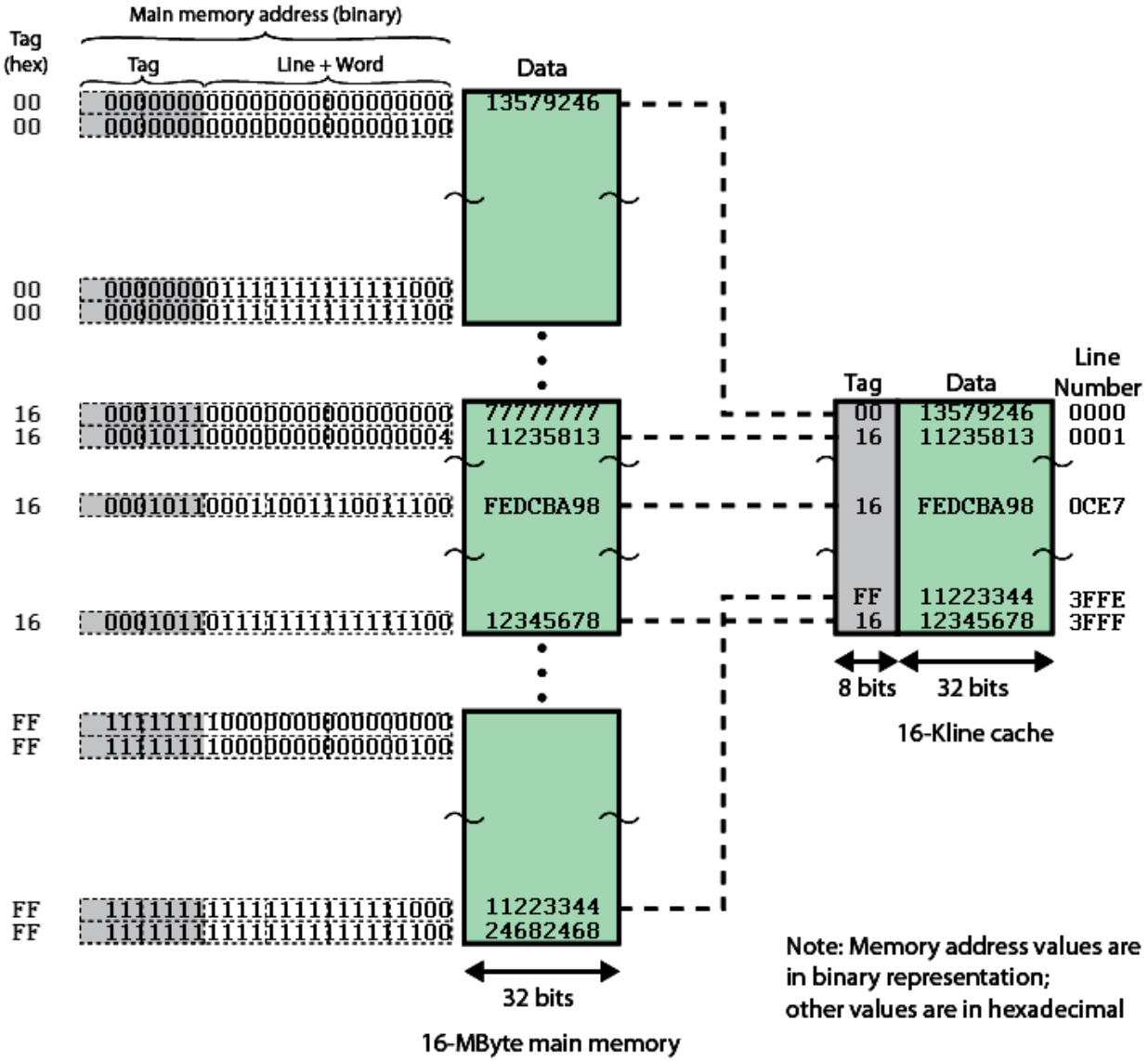
Cache line	Main Memory blocks s
0	0, m, 2m, 3m…2^s–m
1	1, m+1, 2m+1…2^s–m+1
m–1	m–1, 2m–1, 3m–1…2^s–1

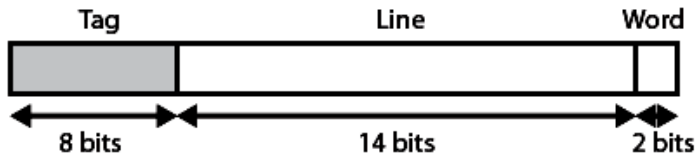
  - Tổng quát
    - $\mathbf{B_j}$ chỉ có thể nạp vào $\mathbf{L_{(j \bmod m)}}$
    - m là số Line của cache.

Bộ nhớ cache

- Ánh xạ trực tiếp (tiếp)




Main memory address (binary)
Tag (hex)
Tag
Line + Word
Data
00 00000000000000000000000000
00 00000000000000000000000100
13579246
00 00000000111111111111111000
00 00000000111111111111111100
16 00010110000000000000000000
16 00010110000000000000000004
77777777
11235813
16 00010110000110011100011100
FEDCBA98
16 00010110111111111111111100
12345678
FF 11111111000000000000000000
FF 11111111000000000000000100
FF 11111111111111111111111000
FF 11111111111111111111111100
11223344
24682468
32 bits
16-MByte main memory
Tag
Data
Line Number
00 13579246 0000
16 11235813 0001
16 FEDCBA98 0CE7
FF 11223344 3FFE
16 12345678 3FFF
8 bits
32 bits
16-Kline cache
Note: Memory address values are in binary representation; other values are in hexadecimal


Main memory address =
Tag
Line
Word
8 bits
14 bits
2 bits

Bộ nhớ cache

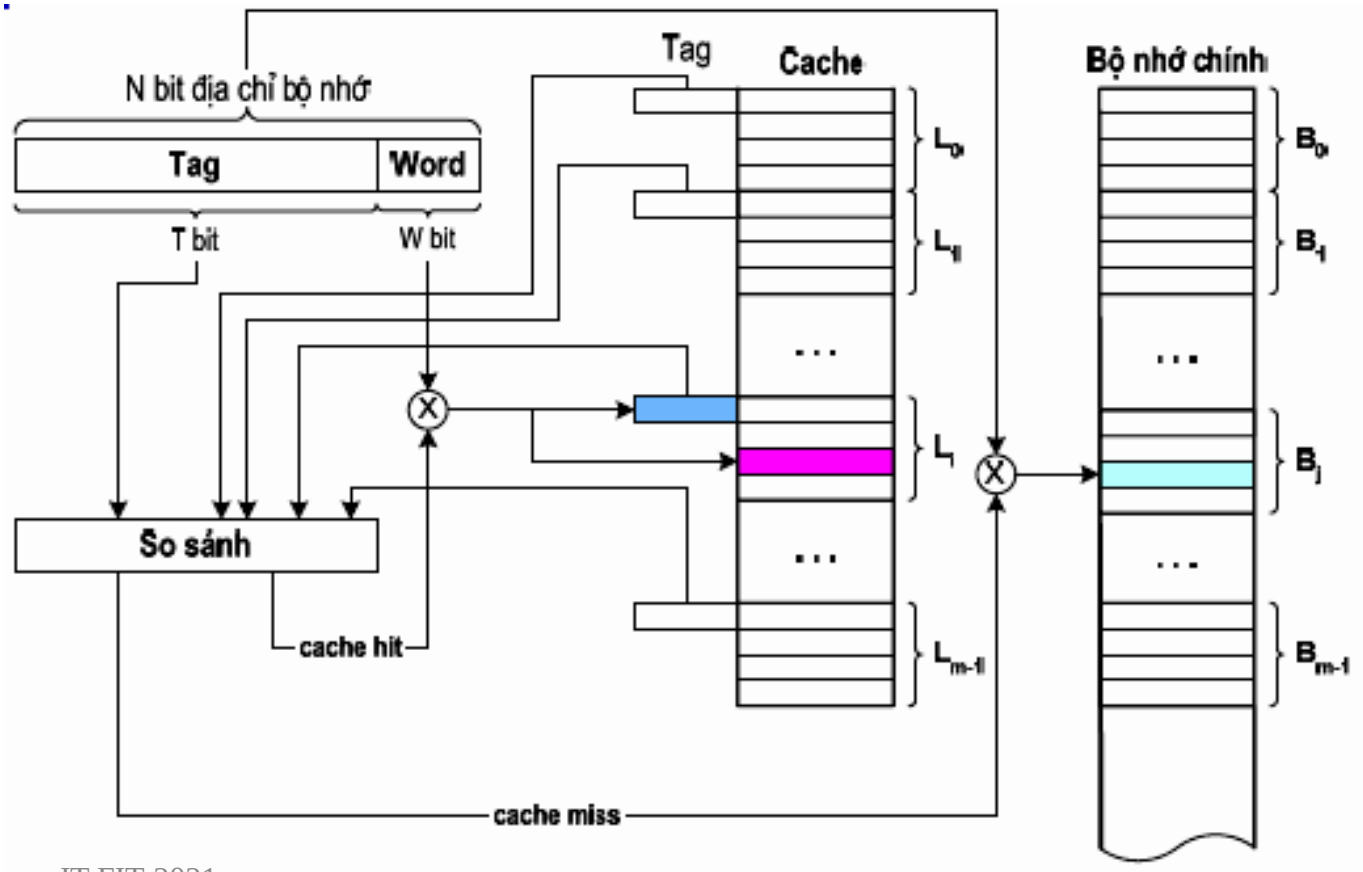
- Đặc điểm của ánh xạ trực tiếp
  - Mỗi một địa chỉ N bit của bộ nhớ chính gồm ba vùng:
    - Vùng *Word* gồm W bit xác định một từ nhớ trong *Block* hay *Line:*
      - 2^W = kích thước của *Block* hay *Line*
    - Vùng *Line* gồm L bit xác định một trong số các *Line* trong cache:
      - 2^L = số *Line* trong cache = m
    - Vùng *Tag* gồm T bit:
      - T = N - (W+L)
  - Bộ so sánh đơn giản
  - Giá thành rẻ
  - Tỷ lệ cache hit thấp (vd: 2 block cùng map vào 1 line)

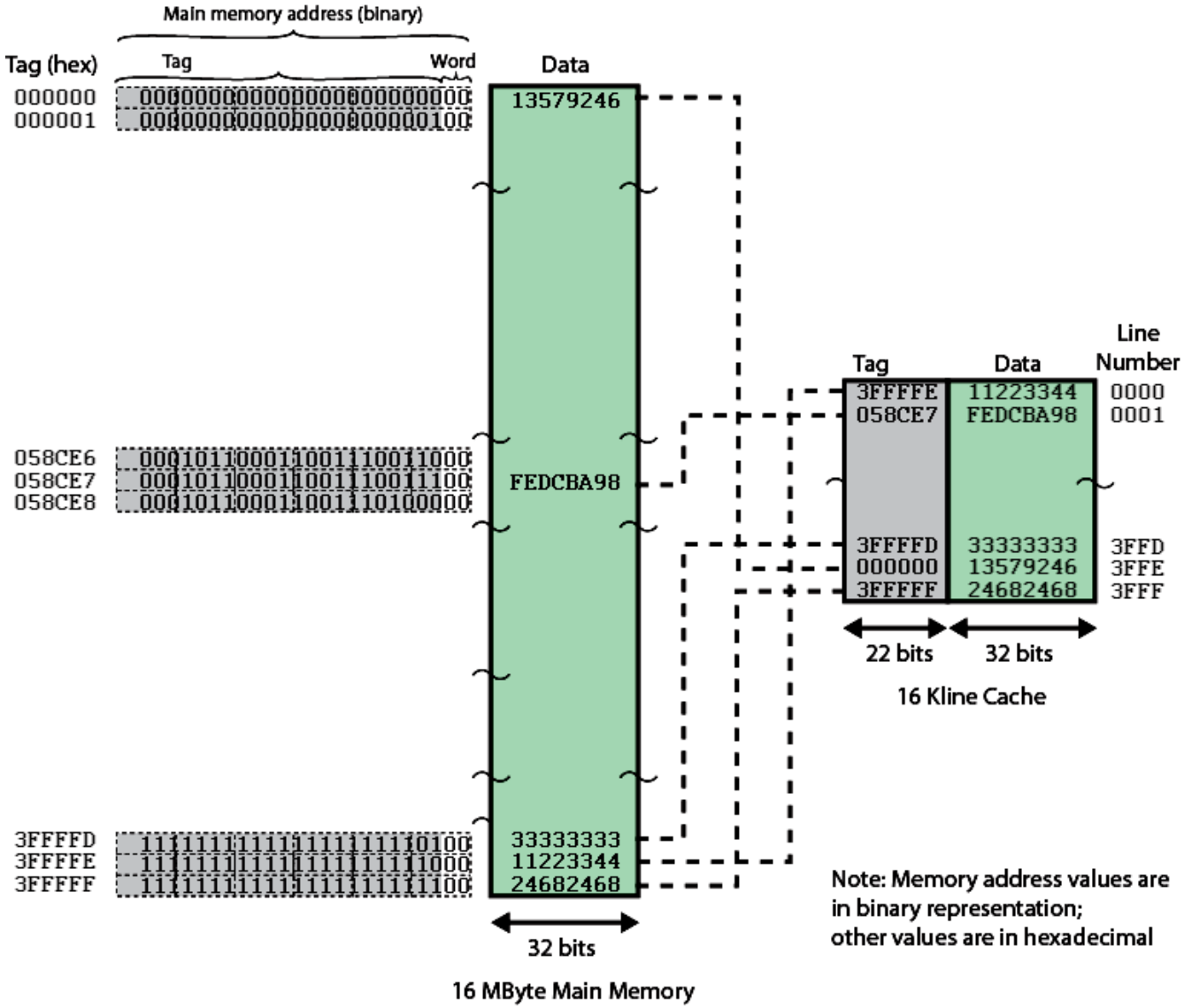
Bộ nhớ cache

- Ánh xạ kết hợp toàn phần
  - Mỗi Block có thể nạp vào bất kỳ Line nào của cache.
  - Địa chỉ của bộ nhớ chính bao gồm hai vùng:
    - Vùng Word giống như trường hợp ánh xạ trực tiếp.
    - Vùng Tag dùng để xác định Block của bộ nhớ chính.
  - Tag xác định Block đang nằm ở Line đó
- Đặc điểm
  - So sánh đồng thời Block nhớ với tất cả các Tag → mất nhiều thời gian
  - Tỷ lệ cache hit cao.
  - Bộ so sánh phức tạp.

Bộ nhớ cache

- Ánh xạ kết hợp toàn phần (tiếp)




Main memory address (binary)
Tag (hex)
Tag
Word
Data
000000
000001
13579246
058CE6
058CE7
058CE8
FEDCBA98
3FFFFD
3FFFFE
3FFFFF
33333333
11223344
24682468
32 bits
16 MByte Main Memory
Line Number
Tag
Data
3FFFFE
11223344
0000
058CE7
FEDCBA98
0001
3FFFFD
33333333
3FFD
000000
13579246
3FFE
3FFFFF
24682468
3FFF
22 bits
32 bits
16 Kline Cache
Note: Memory address values are in binary representation; other values are in hexadecimal

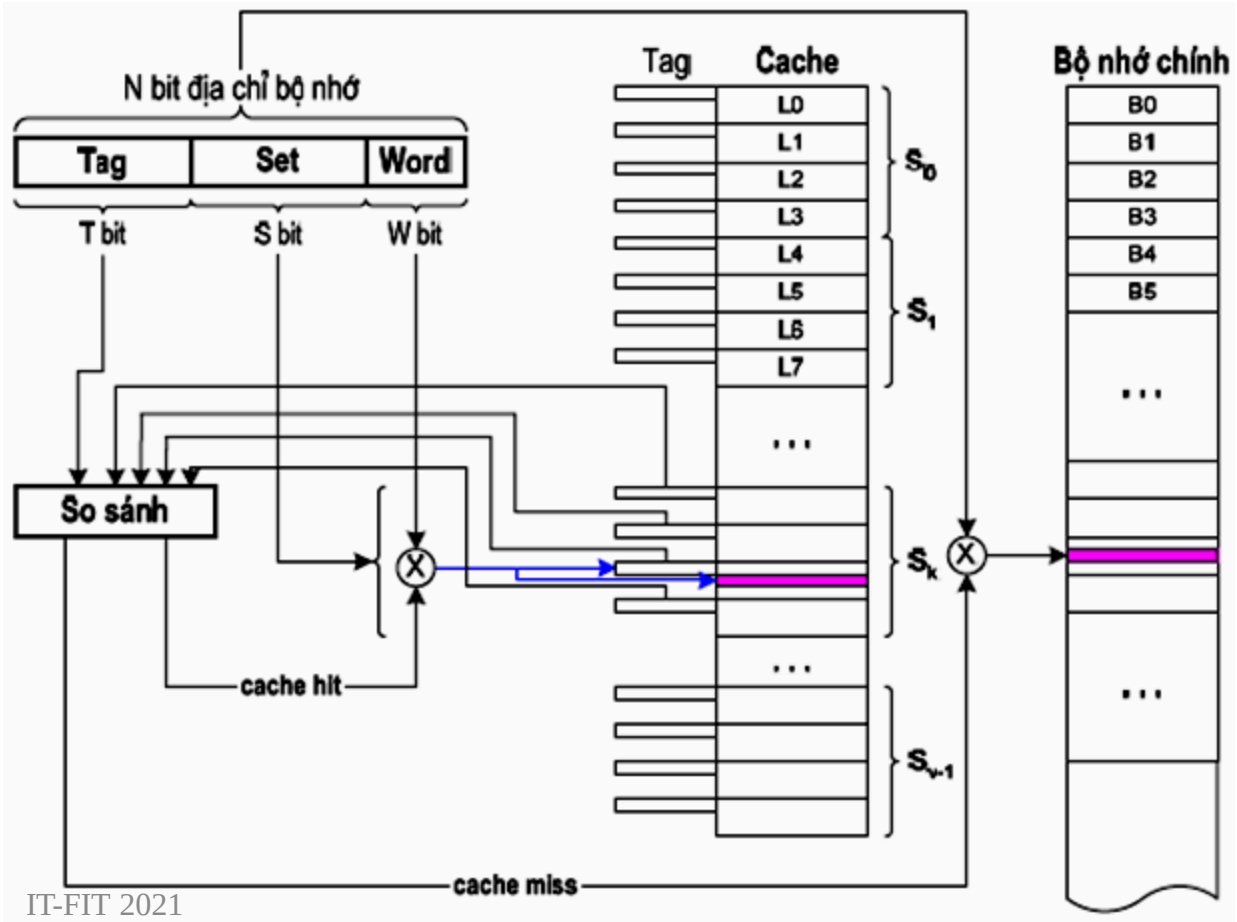

Tag
Word
Main Memory Address =
22 bits
2 bits

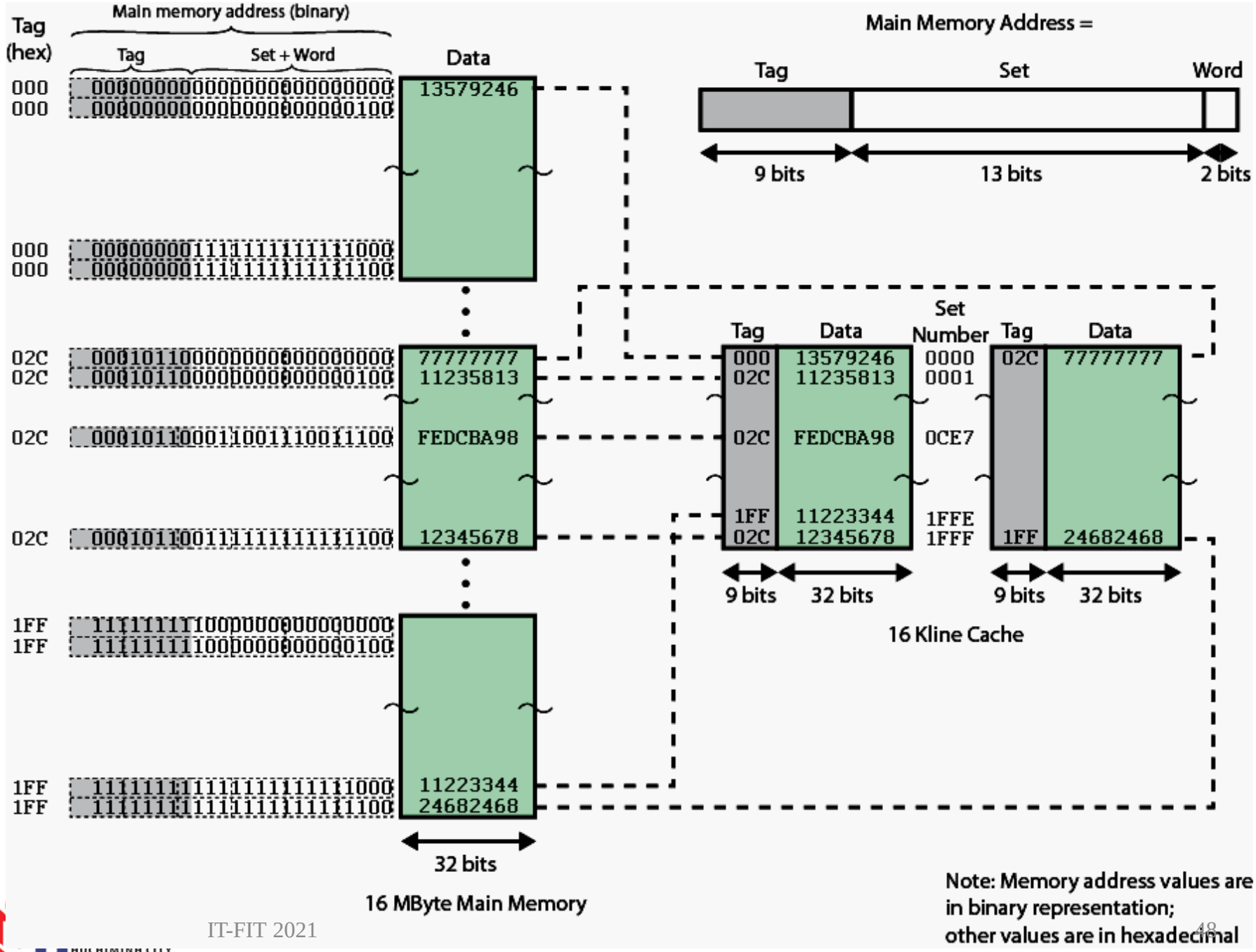
Bộ nhớ cache

- Ánh xạ kết hợp theo bộ
  - Cache đươc chia thành các bộ (Set)
  - Mỗi một Set chứa một số Line
  - Ví dụ:
    - 4 Line/Set → 4-way associative mapping
  - Ánh xạ theo nguyên tắc sau:
    - $B_0 \rightarrow S_0$
    - $B_1 \rightarrow S_1$
    - $B_2 \rightarrow S_2$
    - .....

Bộ nhớ cache

- Ánh xạ kết hợp theo bộ (tiếp)




Tag (hex)
Main memory address (binary)
Tag
Set + Word
Data
000 00000000000000000000000000
000 00000000000000000000000100
13579246
000 00000000111111111111111000
000 00000000111111111111111100
02C 00010110000000000000000000
02C 00010110000000000000000100
77777777
11235813
02C 00010110001001110011100
FEDCBA98
02C 00010110011111111111111100
12345678
1FF 11111111100000000000000000
1FF 11111111100000000000000100
1FF 11111111111111111111111000
1FF 11111111111111111111111100
11223344
24682468
32 bits
16 MByte Main Memory
Main Memory Address =
Tag
Set
Word
9 bits
13 bits
2 bits
Set Number
Tag
Data
000 13579246
02C 11235813
02C FEDCBA98
1FF 11223344
02C 12345678
0000
0001
0CE7
1FFE
1FFF
02C 77777777
1FF 24682468
9 bits
32 bits
16 Kline Cache
Note: Memory address values are in binary representation; other values are in hexadecimal

Bộ nhớ cache

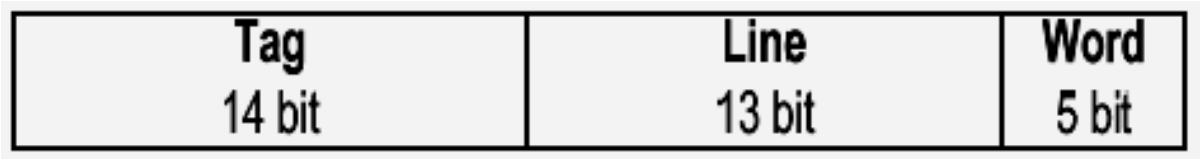
- Đặc điểm ánh xạ kết hợp theo bộ
  - Kích thước Block = 2^W Word
  - Vùng Set có S bit dùng để xác định một trong số V = 2^S Set (dùng để xác định Set nào trong cache)
  - Vùng Tag có T bit: T = N - (W+S) dùng để xác định Line nào có trong Set
  - Mỗi Block bộ nhớ có thể ánh xạ vào Line bất kỳ trong 1 Set tương ứng
  - Là phương pháp tổng hợp từ hai phương pháp trên
  - Thông thường sử dụng 2, 4, 8, 16 Lines/Set

Bộ nhớ cache

- Ví dụ về ánh xạ địa chỉ
  - Không gian địa chỉ bộ nhớ chính = 4GB
  - Dung lượng bộ nhớ cache là 256KB
  - Kích thước Line (Block) = 32byte.
  - Xác định số bit của các trường địa chỉ cho ba trường hợp tổ chức:
    - Direct mapping
    - Fully associative mapping
    - 4-way set associative mapping

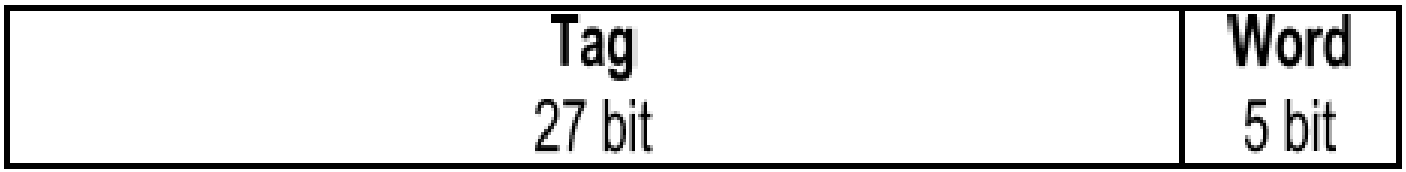
Bộ nhớ cache

- Ví dụ: Direct mapping
  - Bộ nhớ chính : 4GB = 2^{32} byte → N = 32 bit
  - Cache : 256 KB = 2^{18} byte.
  - Line : 32 byte = 2^5 byte → W = 5 bit
  - Line trong cache : $2^{18}/ 2^5 = 2^{13}$ Line → L = 13 bit
  - T = 32 - (13 + 5) = 14 bit



Bộ nhớ cache

- Ví dụ: Fully associative mapping
  - Bộ nhớ chính : 4GB = 2^{32} byte → N = 32 bit
  - Line : 32 byte = 2^5 byte → W = 5 bit
  - Số bit của vùng Tag : T = 32 - 5 = 27 bit



Bộ nhớ cache

- Ví dụ: 4-way set associative mapping
  - Bộ nhớ chính : 4GB = 2^{32} byte → N = 32 bit
  - Line : 32 byte = 2^5 byte → W = 5 bit
  - Line trong cache = 2^{18}/ 2^5 = 2^{13} Line
  - Một Set có 4 Line = 2^2 Line
  - Set trong cache = 2^{13}/ 2^2 = 2^{11} Set → S = 11 bit
  - Số bit của vùng Tag : T = 32 - (11 + 5) = 16 bit

Bộ nhớ cache

- Giải thuật thay thế cache
  - Mục đích: Giải thuật thay thế dùng để xác định Line nào trong cache sẽ được chọn để đưa dữ liệu vào cache khi không còn chỗ trống
  - Thường được cài đặt bằng phần cứng để tăng tốc độ xử lý
  - Đối với ánh xạ trực tiếp:
    - Không phải lựa chọn
    - Mỗi Block chỉ ánh xạ vào một Line xác định

Bộ nhớ cache

- Giải thuật thay thế với ánh xạ kết hợp
  - FIFO (First In First Out): Thay thế Block nào nằm lâu nhất ở trong Set đó
  - LFU (Least Frequently Used): Thay thế Block nào trong Set có số lần truy cập ít nhất trong cùng một khoảng thời gian
  - LRU (Least Recently Used): Thay thế Block ở trong Set tương ứng có thời gian lâu nhất không được tham chiếu tới.
  - Giải thuật tối ưu nhất: LRU

Bộ nhớ cache

- Đặc điểm ghi dữ liệu ra cache
  - Chỉ ghi vào 1 block trong cache khi nội dung trong bộ nhớ chính thay đổi
  - Nếu CPU ghi ra cache, ô nhớ tương ứng bị lạc hậu (invalid) → cần update ra BN chính. Ngược lại, nếu ghi vào BN chính, nội dung trong cache sẽ bị invalid → cần update lại cache
  - Trong hệ đa xử lý có nhiều CPU với cache riêng, khi ghi vào 1 cache, các cache khác sẽ bị invalid

Bộ nhớ cache

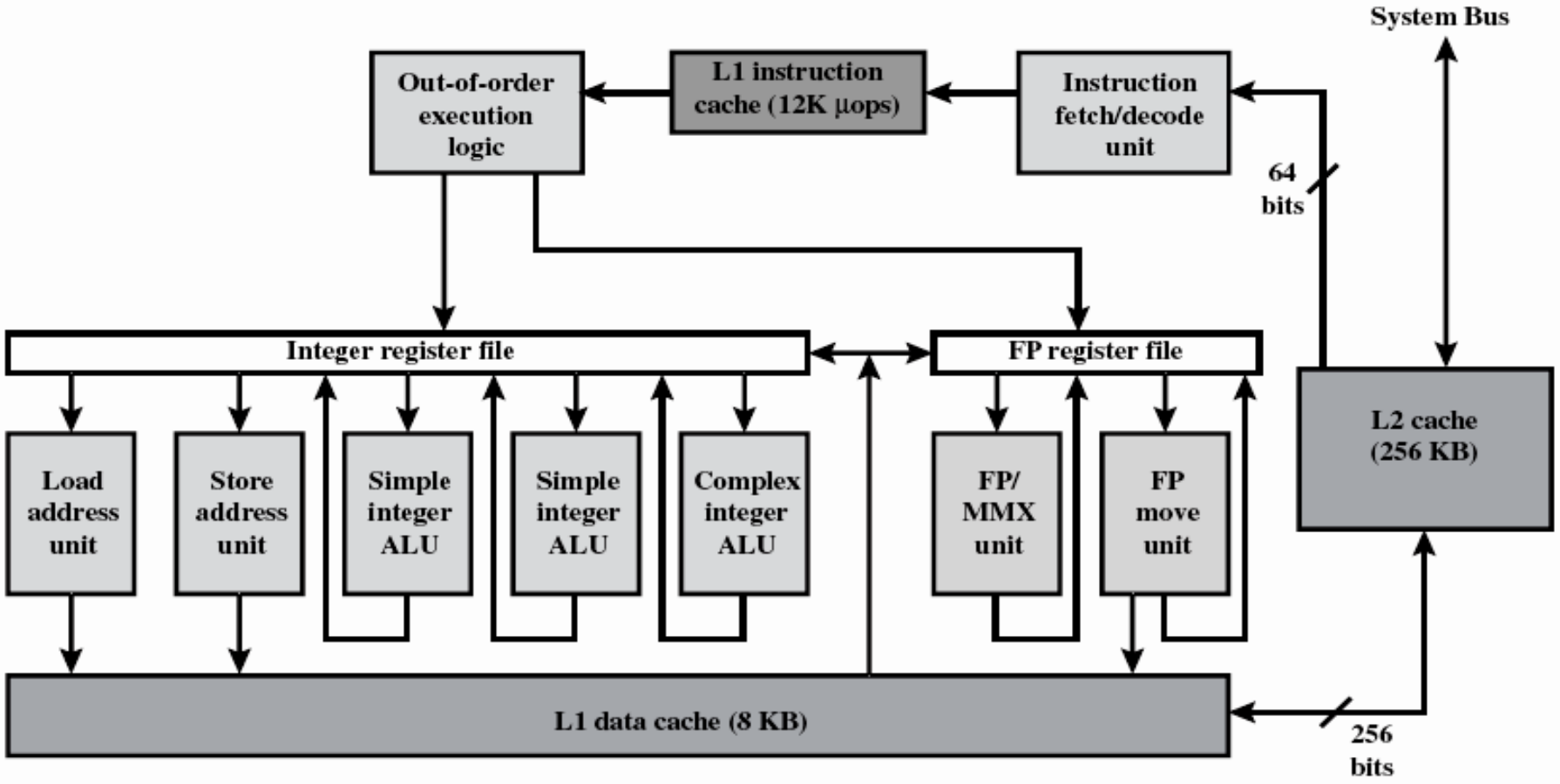
- Phương pháp ghi dữ liệu khi cache hit
  - Write-through:
    - Ghi cả cache và cả bộ nhớ chính
    - Tốc độ chậm
    - Cho phép CPU khác hoặc IO truy cập dữ liệu đã ghi từ BN
  - Write-back:
    - Chỉ ghi ra cache
    - Tốc độ nhanh
    - CPU khác hoặc IO không đọc được dữ liệu mới trong BN
    - Khi Block trong cache bị thay thế cần phải ghi Block này về bộ nhớ chính

Bộ nhớ cache

- Cache tách biệt và cache đồng nhất
  - Cache tách biệt (Split Cache): Tổ chức cache riêng cho dữ liệu và cache riêng cho lệnh chương trình
    - Tối ưu cho từng loại cache
    - Hỗ trợ kiến trúc pipeline
    - Phần cứng phức tạp
  - Cache đồng nhất (Unified Cache): Sử dụng 1 cache chung cho cả dữ liệu lẫn lệnh chương trình
    - Cân bằng về tỷ lệ cache hit
    - Phần cứng đơn giản

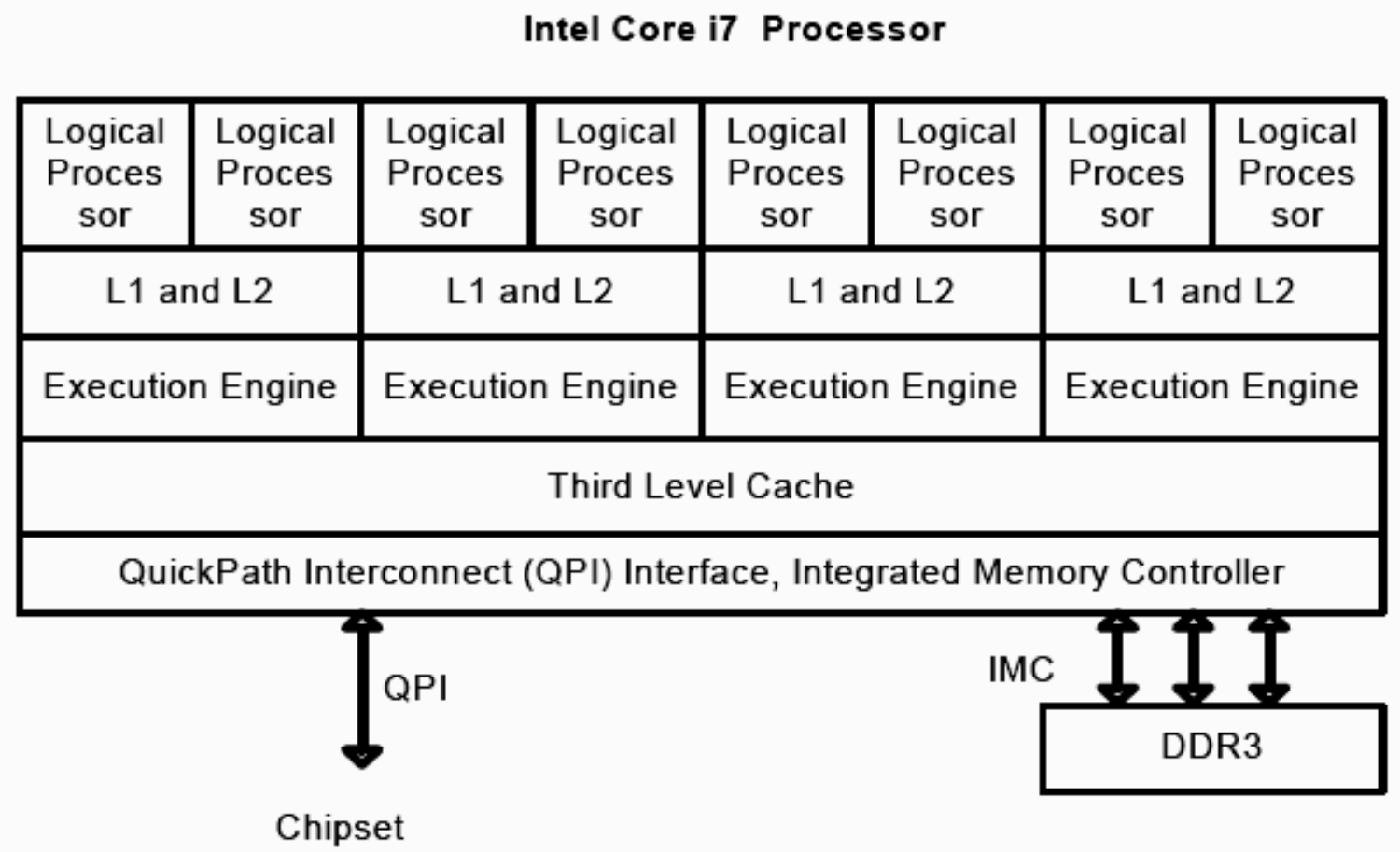
Bộ nhớ cache

- Ví dụ: Hệ thống cache trong CPU Intel Pentium 4



Bộ nhớ cache

- Ví dụ: Hệ thống cache trong CPU Intel Core i7



Bộ nhớ ngoài

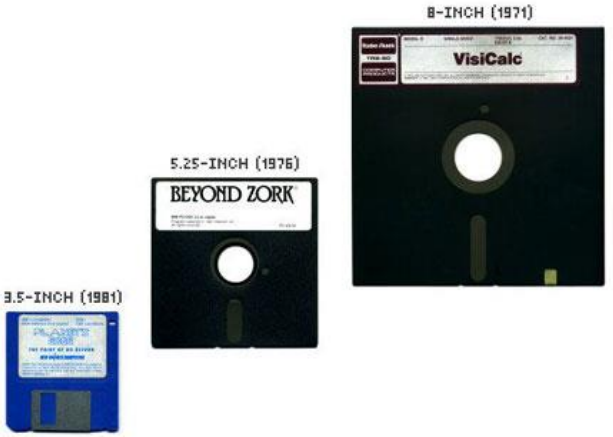
- Các kiểu bộ nhớ ngoài
  - Trống từ (Drum): Ngày nay không còn sử dụng
  - Băng từ (Tape): Chuyên dùng cho backup dữ liệu
  - Đĩa từ: Đang sử dụng rộng rãi nhất
  - Đĩa quang: Dùng để trao đổi dữ liệu giữa các máy tính và phân phối phần mềm
  - Flash Disk: Loại bộ nhớ bán dẫn gắn ngoài qua cổng USB, nhỏ gọn và thuận tiện để trao đổi dữ liệu
  - SSD (Solid State Disk): Cũng là bộ nhớ bán dẫn có dung lượng lớn giao tiếp với máy tính tương tự ổ đĩa cứng, tốc độ truy cập cao, ít tốn điện, không ồn, chống sốc tốt → rất phù hợp với máy xách tay. Nhược điểm giá thành đắt.

Bộ nhớ ngoài

- Đĩa từ
  - Bao gồm đĩa mềm (floppy disk) và đĩa cứng (hard disk)
  - Các đặc tính đĩa từ
    - Đầu từ cố định hay đầu từ di động
    - Đĩa cố định hay thay đổi được (removable)
    - Một mặt hay hai mặt
    - Một tấm đĩa (đĩa mềm) hay nhiều tấm đĩa (đĩa cứng)
    - Cơ chế đầu từ
      - Tiếp xúc (đĩa mềm)
      - Không tiếp xúc

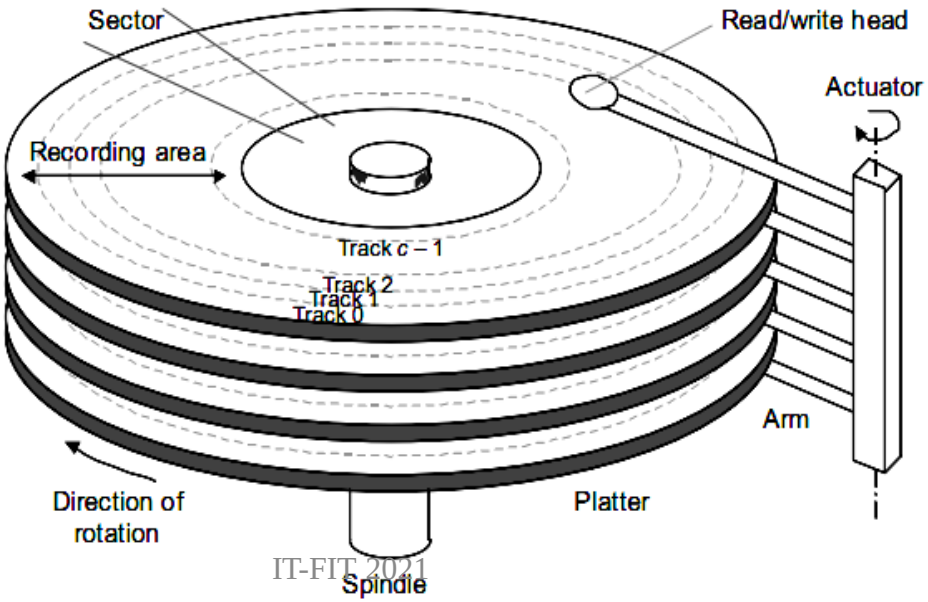
Bộ nhớ ngoài

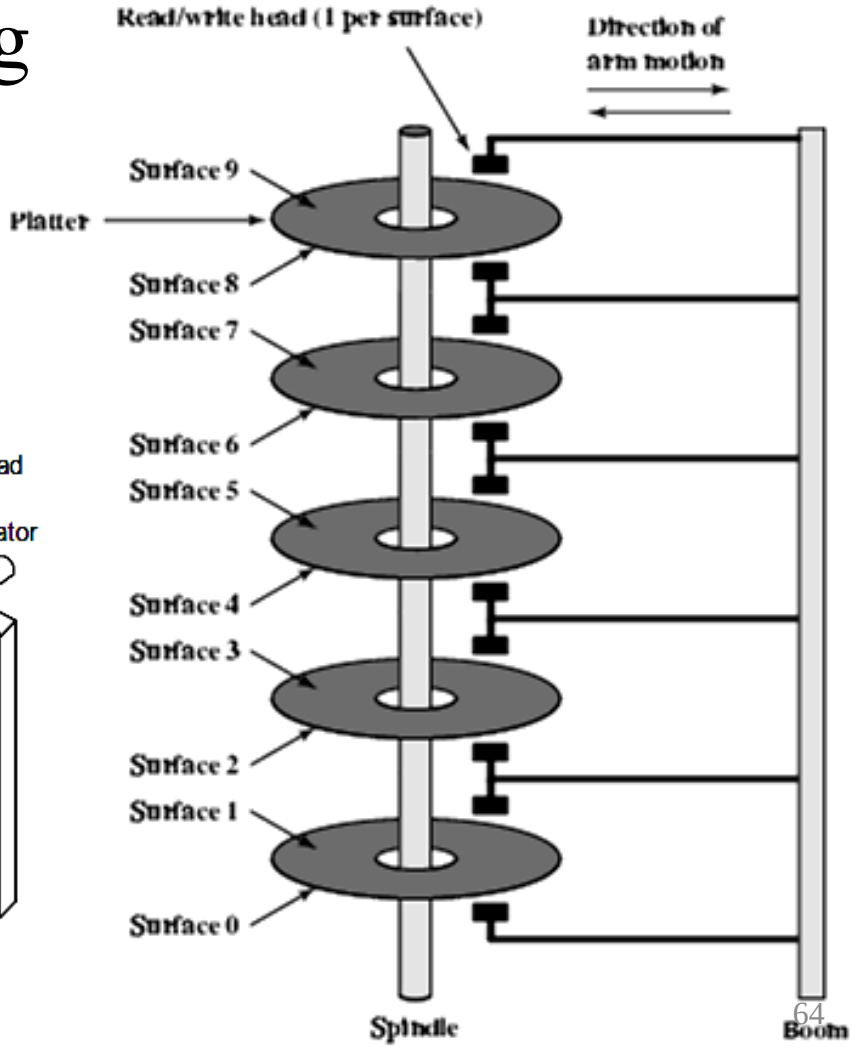
- Đĩa từ (tiếp)
  - Đĩa mềm
    - 8”, 5.25”, 3.5”
    - Dung lượng nhỏ: chỉ tới 1.44MB
    - Tốc độ chậm
    - Hiện nay không sản xuất nữa
  - Đĩa cứng
    - Thường có nhiều tấm đĩa
    - Đang sử dụng rộng rãi
    - Dung lượng lớn (hiện nay có ổ 3TB – 2011)
    - Tốc độ đọc/ghi nhanh
    - Rẻ tiền



Bộ nhớ ngoài

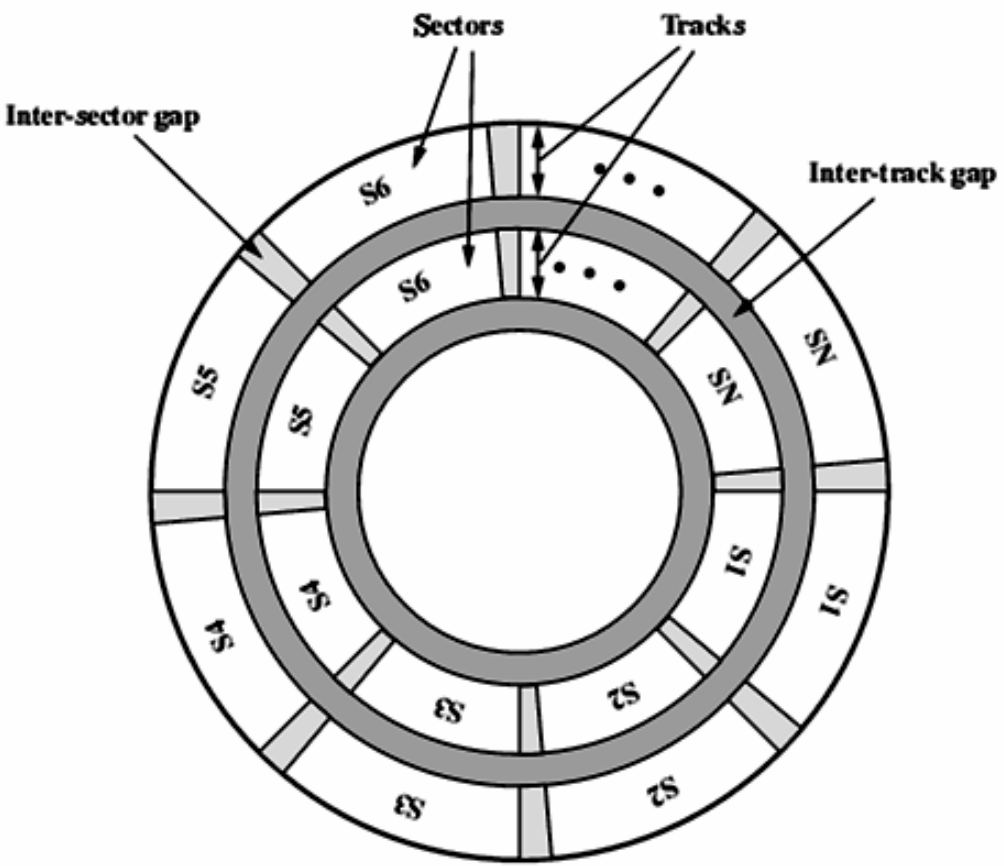
- Cấu trúc vật lý đĩa cứng
  - Mặt đĩa
  - Track (cylinder)
  - Sector





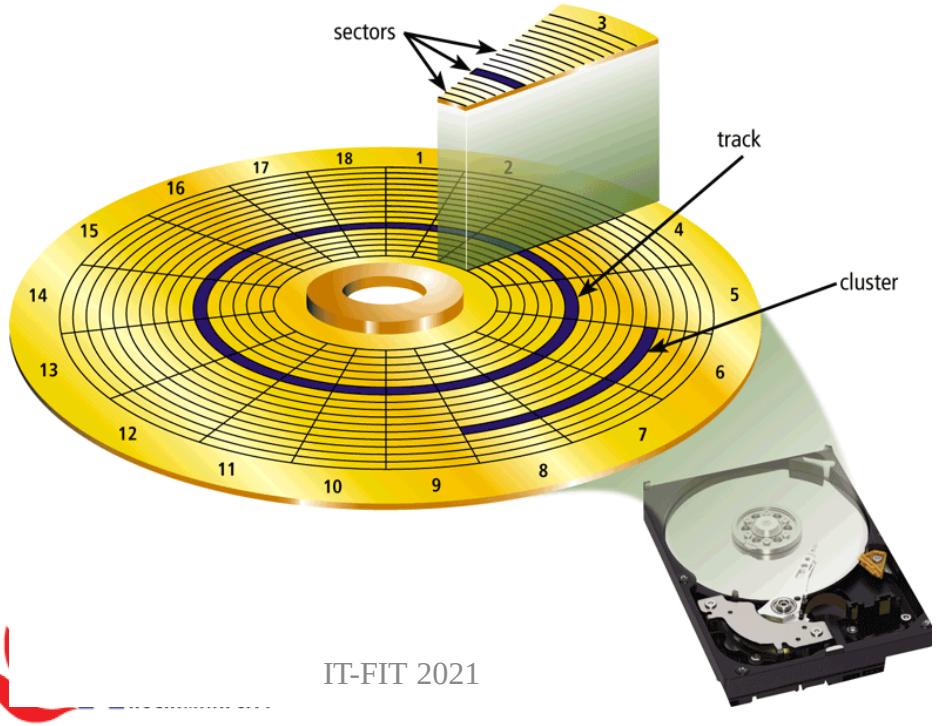
Bộ nhớ ngoài

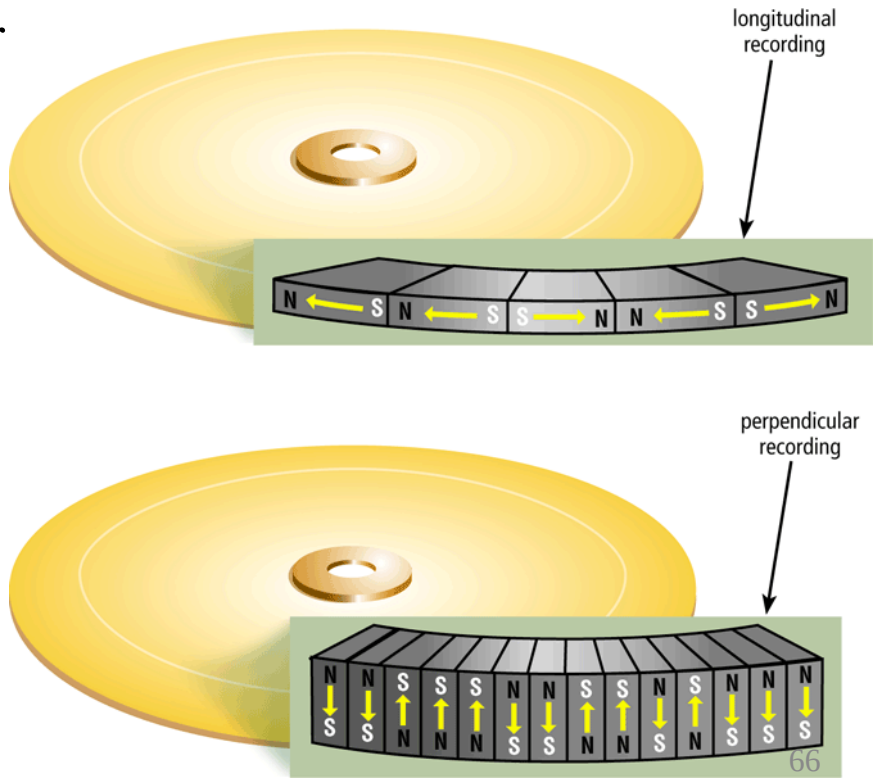
- Cấu trúc vật lý đĩa cứng (tiếp)
  - Track là các vòng tròn đồng tâm
  - Đơn vị đọc ghi: từng sector (~ 512Byte), có thể đọc ghi theo block nhiều sector (cluster)
  - Thời gian đọc ghi:
    - Seek time
    - Latency time
    - Transfer time
  - Đĩa quay với vận tốc góc không đổi CAV (constant angular velocity)



Bộ nhớ ngoài

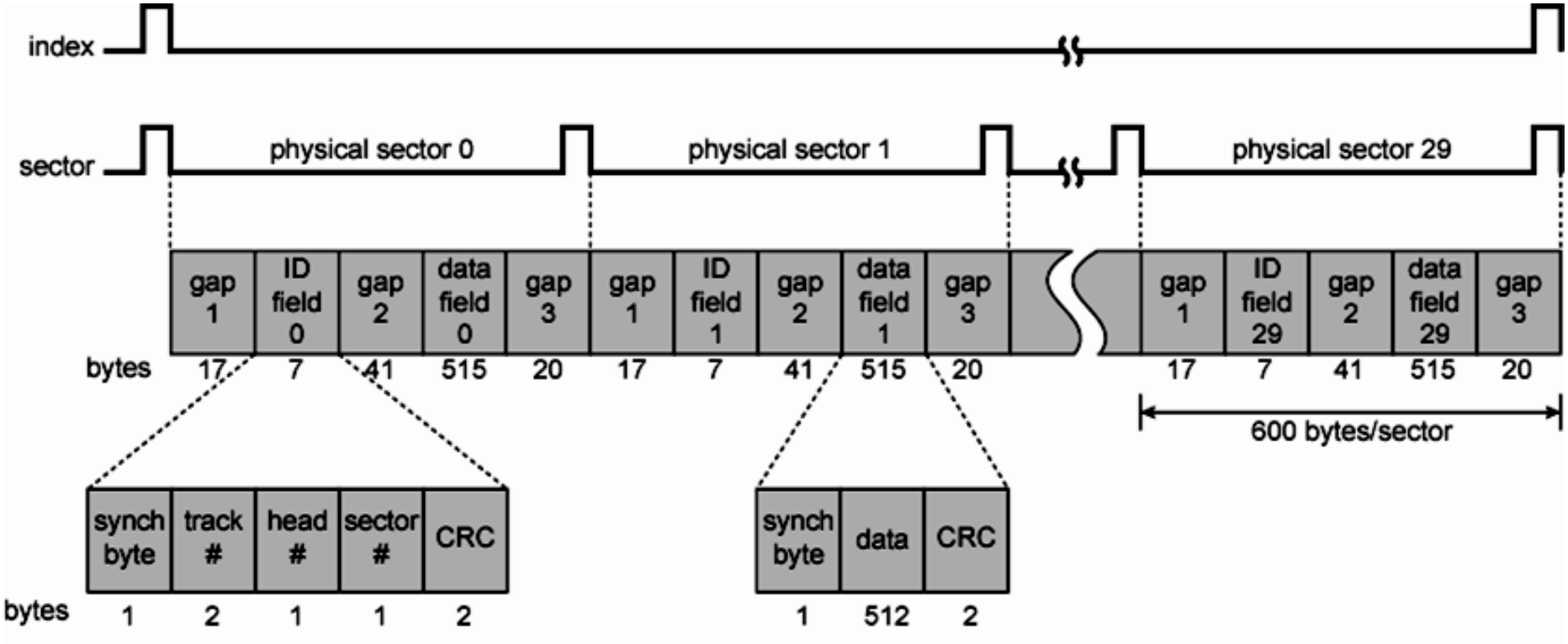
- Cấu trúc vật lý đĩa cứng (tiếp)
  - Longitudial recording: Ghi tuyến tính
  - Perpendicular recording: Ghi trực giao
  - Cluster: Một bộ gồm nhiều sector





Bộ nhớ ngoài

- Định dạng sector

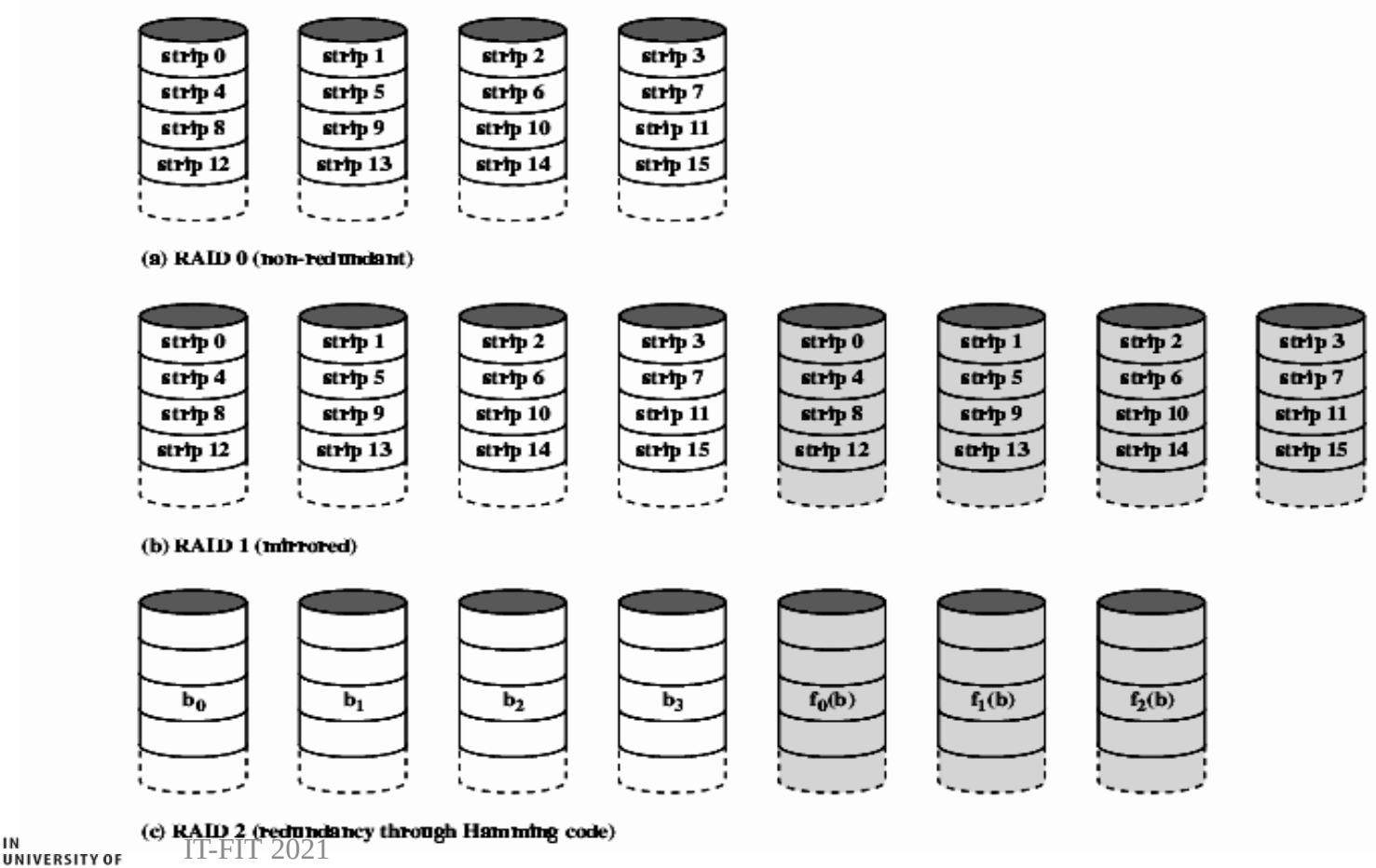


Bộ nhớ ngoài

- Kỹ thuật RAID
  - RAID: Redundant Array of Independent Disks
  - Ghép nhiều ổ đĩa vật lý để truy cập như 1 ổ luận lý
    - Tăng tốc độ truy cập (đọc ghi luân phiên và song song)
    - Tăng độ an toàn dữ liệu khi đĩa hư hỏng (ghi dư thừa hoặc ghi thêm thông tin ECC/parity)
    - Tăng dung lượng tối đa của đơn vị lưu trữ (nhiều đĩa)
  - Hiện có 7 loại thông dụng: RAID0 – RAID6

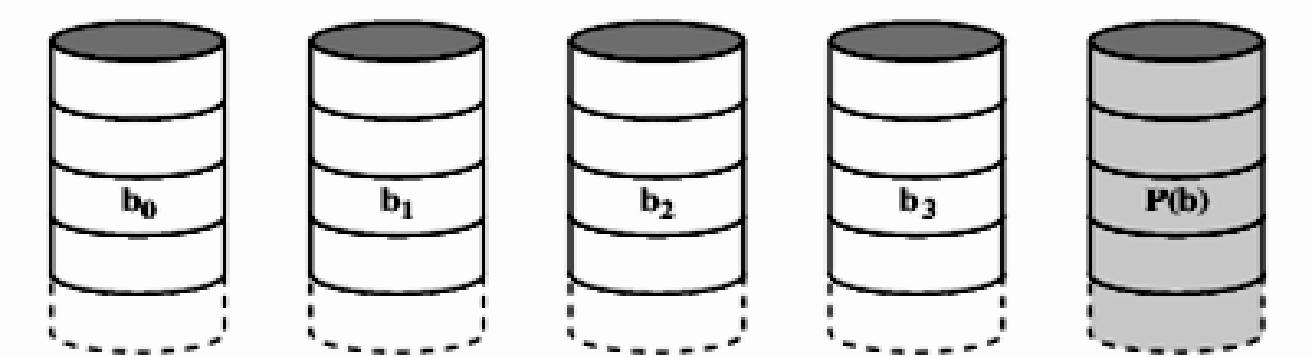
Bộ nhớ ngoài

- RAID 0, 1 và 2

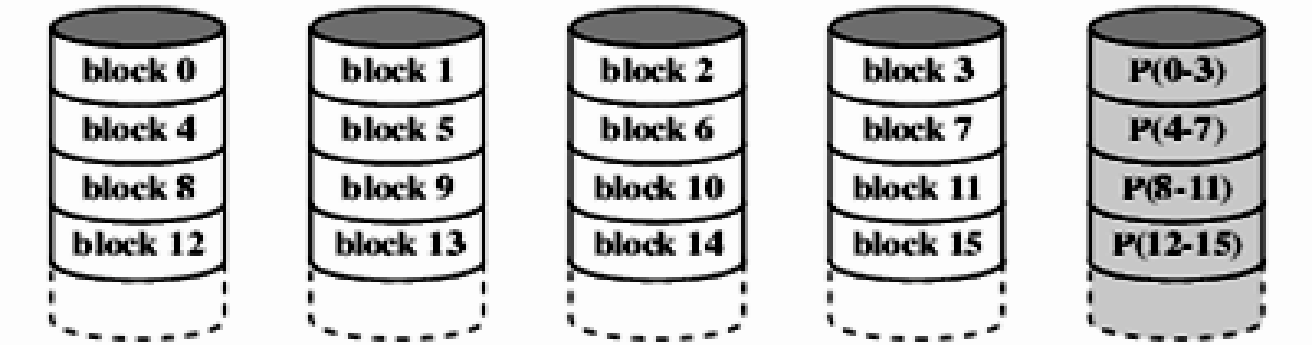


Bộ nhớ ngoài

- RAID 3 và 4



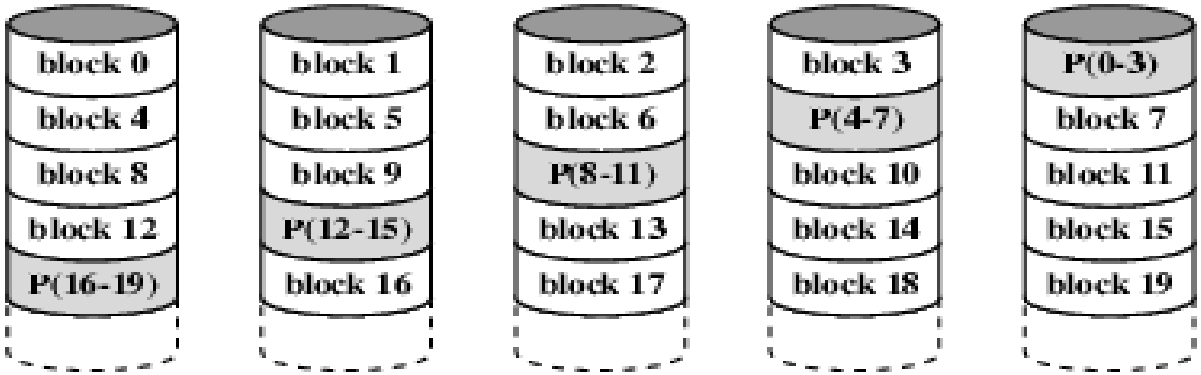
(d) RAID 3 (bit-interleaved parity)



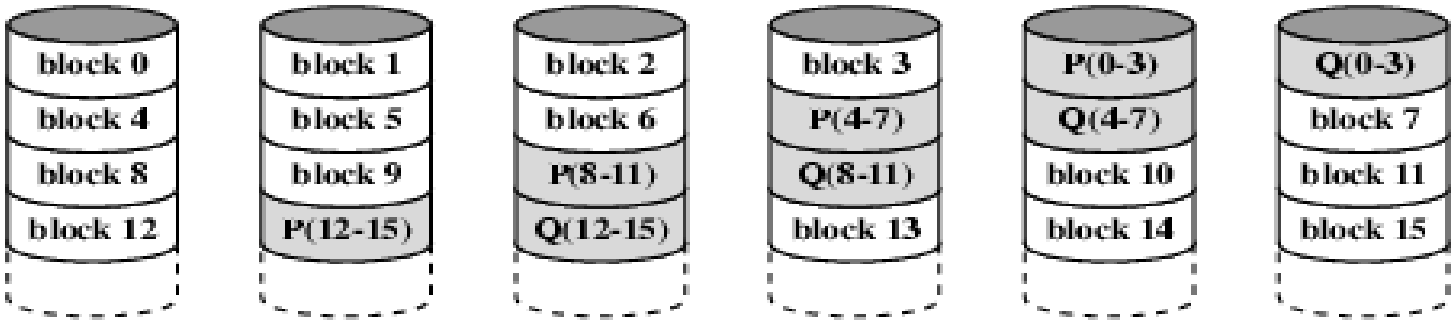
(e) RAID 4 (block-level parity)

Bộ nhớ ngoài

- RAID 5 và 6



(f) RAID 5 (block-level distributed parity)



(g) RAID 6 (dual redundancy)

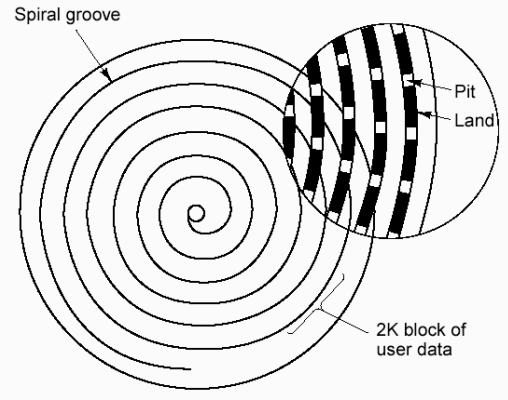
Bộ nhớ ngoài

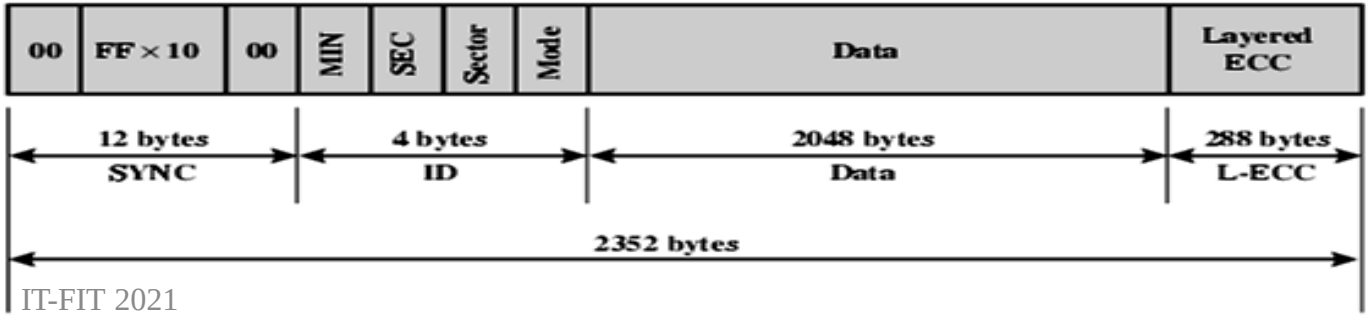
- Tóm tắt kỹ thuật RAID

Category	Level	Description	I/O Request Rate (Read/Write)	Data Transfer Rate (Read/Write)	Typical Application
Striping	0	Non-redundant	Large strips: Excellent	Small strips: Excellent	Applications requiring high performance for non-critical data
Mirroring	1	Mirrored	Good/Fair	Fair/Fair	System drives; critical files
Parallel access	2	Redundant via Hamming code	Poor	Excellent	
	3	Bit-interleaved parity	Poor	Excellent	Large I/O request size applications, such as imaging, CAD
Independent access	4	Block-interleaved parity	Excellent/Fair	Fair/Poor	
	5	Block-interleaved distributed parity	Excellent/Fair	Fair/Poor	High request rate, read-intensive, data lookup
	6	Block-interleaved dual distributed parity	Excellent/Poor	Fair/Poor	Applications requiring extremely high availablity

Bộ nhớ ngoài

- Đĩa quang (optical disk)
  - CD (compact disk)
    - Khả năng đọc/ghi: CD-ROM, CD-R, CD-RW
    - Đường kính: 12cm, 8cm
    - Dung lượng: 700MB, 200MB
    - Track: Ghi theo các vòng hướng tâm, tốc độ dài không đổi CLV (constant linear velocity)
    - Tốc độ đọc ghi: 1x – 52x (1x= ??)
    - Chuẩn định dạng: ISO 9660, UDF (Universal Disk Format)





Bộ nhớ ngoài

- Đĩa quang (tiếp)
  - DVD (Digital Versatile Disk): Loại đĩa dung lượng cao (so với CD), xuất phát từ đĩa phim video (Digital Video Disk)
  - Khả năng đọc ghi: DVD-ROM, DVD±R, DVD±RW, DVD-RAM
  - Số mặt/ số lớp: 1-2 mặt, 1-2 lớp/mặt
  - Đường kính: 12cm, 8cm
  - Tốc độ: 1x – 24x (1x=?)
  - Đĩa DVD dung lượng cao
    - HD-DVD (15-60GB)
    - Blue ray (25-50GB)

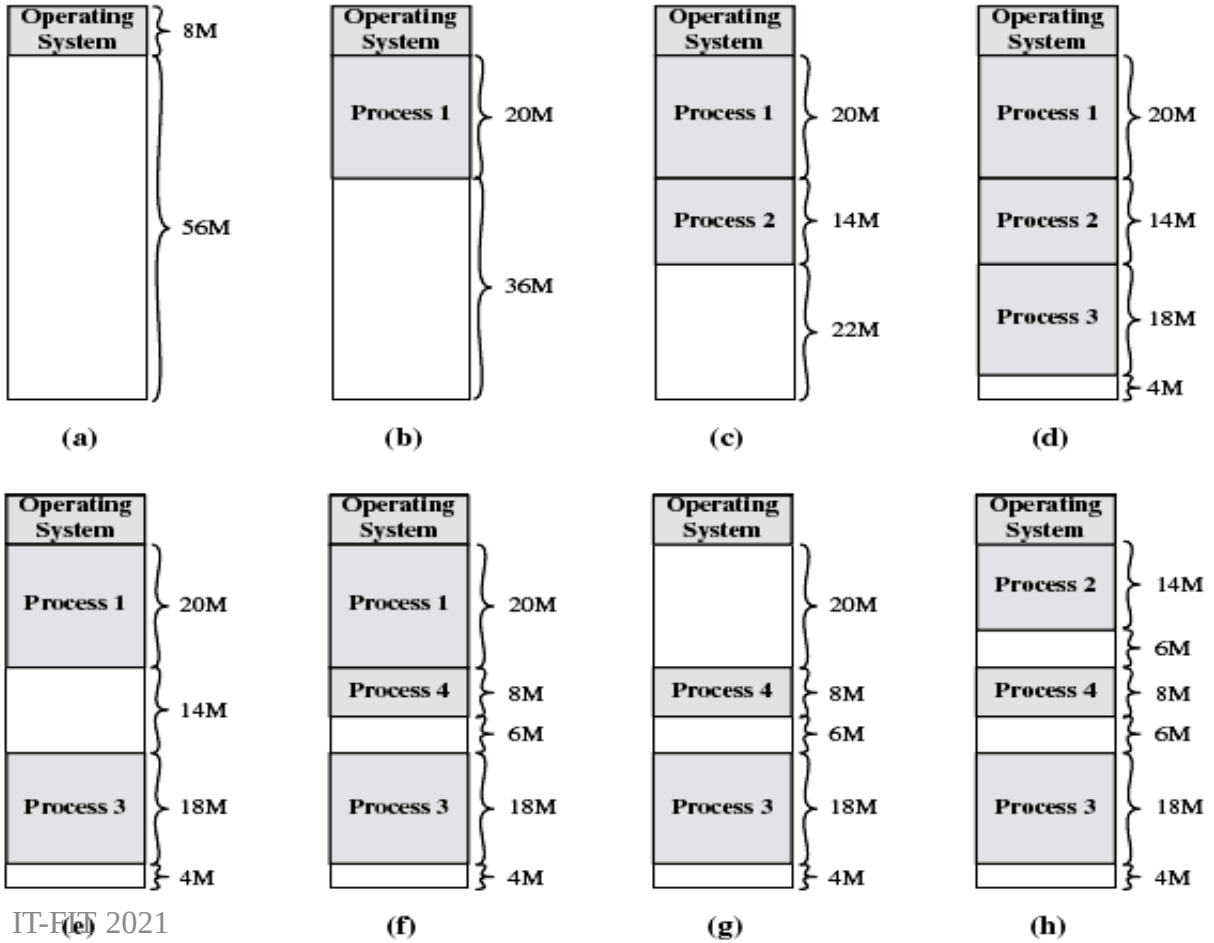
Sides	Layers	Diameter (cm)	Capacity (GB)
1	1	8	1.46
1	2	8	2.66
2	2	8	2.92
2	4	8	5.32
1	1	12	4.7
1	2	12	8.54
2	2	12	9.4
2	4	12	17.08

Bộ nhớ ảo

- Bộ nhớ thật
  - Không gian địa chỉ trong chương trình trùng với không gian địa chỉ trong bộ nhớ. Cho phép người lập trình truy cập trực tiếp vào 1 ô nhớ → Khó bảo vệ bộ nhớ.
  - Khi thi hành, hệ điều hành nạp toàn bộ chương trình vào bộ nhớ (nạp trước) → bộ nhớ máy tính phải đủ lớn để chạy các CT lớn
  - Chương trình được cấp phát 1 vùng nhớ có địa chỉ liên tục (cấp phát liên tục). HĐH sẽ thu hồi vùng nhớ sau khi chương trình kết thúc
  - Để thực hiện đa chương, HĐH cần chia BN ra nhiều vùng (partition), mỗi vùng cấp phát cho 1 CT
  - Khi bộ nhớ đầy
    - HĐH không cấp tiếp, các CT phải chờ đến khi có 1 vùng nhớ trống
    - HĐH cấp tiếp: Cần kỹ thuật tráo đổi (swapping) để ghi tạm vùng nhớ của 1 CT khác ra BN ngoài, lấy chỗ trống cấp cho CT mới

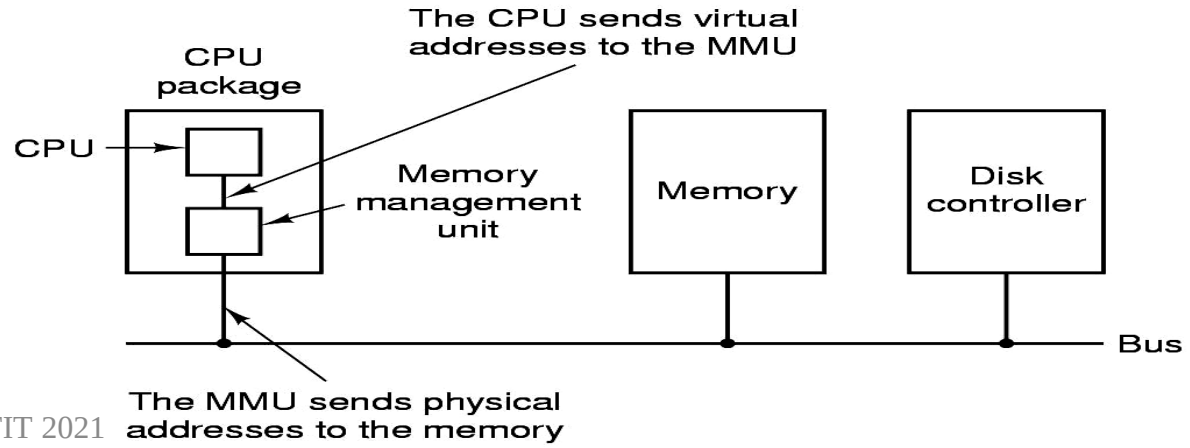
Bộ nhớ ảo

- Bộ nhớ thật (tiếp)



Bộ nhớ ảo

- Bộ nhớ ảo (Virtual Memory)
  - Không gian địa chỉ trong CT (địa chỉ ảo) được tách biệt với không gian địa chỉ trong BN (địa chỉ thực) → CPU và HĐH sẽ phối hợp để ánh xạ (mapping) địa chỉ ảo trong CT thành địa chỉ thật trong BN
  - Việc ánh xạ và quản lý BN ảo được thực hiện qua đơn vị MMU (Memory Management Unit)



Bộ nhớ ảo

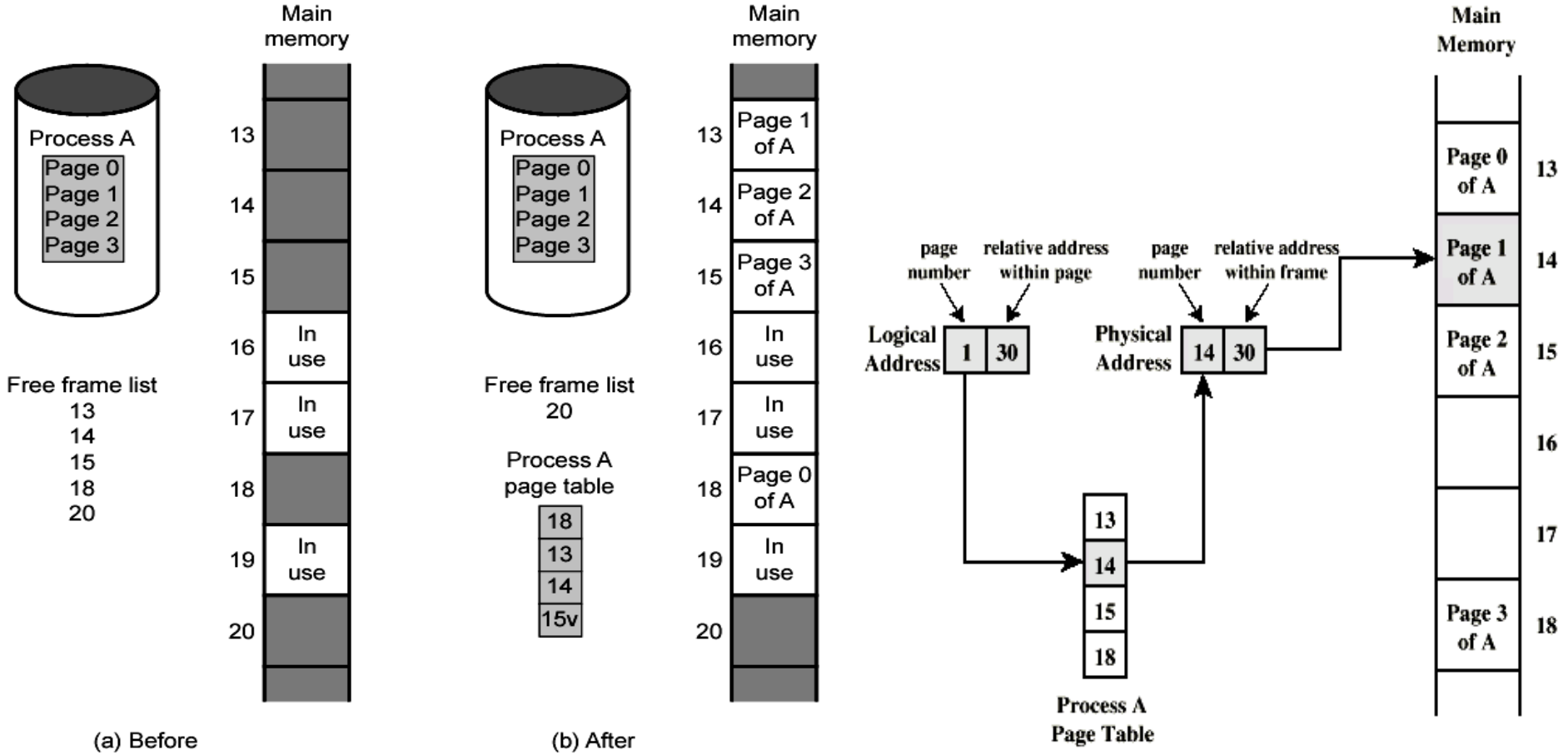
- Bộ nhớ ảo (tiếp)
  - Khi thi hành, hệ điều hành chỉ nạp các phần cần thiết của CT vào bộ nhớ (nạp theo yêu cầu), không cần nạp toàn bộ CT → tránh lãng phí BN
  - Các CT được cấp phát nhiều vùng nhớ có địa chỉ tách biệt nhau (cấp phát không liên tục).
  - Sử dụng kỹ thuật tráo đổi (swapping) để ghi tạm thời các vùng nhớ chưa cần đến ra BN ngoài (swap-out) để lấy chỗ trống nạp thông tin cần thiết vào BN (swap-in) khi cần đến
  - BN ngoài thông dụng là đĩa cứng
  - Có 2 kỹ thuật BN ảo:
    - Kỹ thuật phân trang : Kích thước các vùng nhớ cố định
    - Kỹ thuật phân đoạn : Kích thước các vùng nhớ thay đổi

Bộ nhớ ảo

- Kỹ thuật phân trang (paging)
  - Không gian địa chỉ ảo trong CT được chia đều ra các trang ảo (virtual page, gọi tắt là page) có kích thước bằng nhau, mỗi trang là 1 đơn vị cấp phát BN của HĐH
  - Không gian địa chỉ thật trong BN cũng được chia đều thành các khung trang (page frame, gọi tắt là frame) có kích thước bằng 1 trang (thường là 4KB)
  - Khi có yêu cầu cấp phát BN, HĐH có thể nạp 1 trang theo yêu cầu vào bất cứ frame nào trong BN thật
  - Khi CT truy cập vào 1 trang chưa được cấp phát sẽ gây ra lỗi trang (page fault) → HĐH phải xử lý bằng cách swapping với 1 trang khác chưa cần sử dụng đến (chậm)
  - HĐH cần 1 bảng quản lý để theo dõi trang nào đang được nạp vào frame nào trong BN cho mỗi CT, gọi là bảng trang (page table)

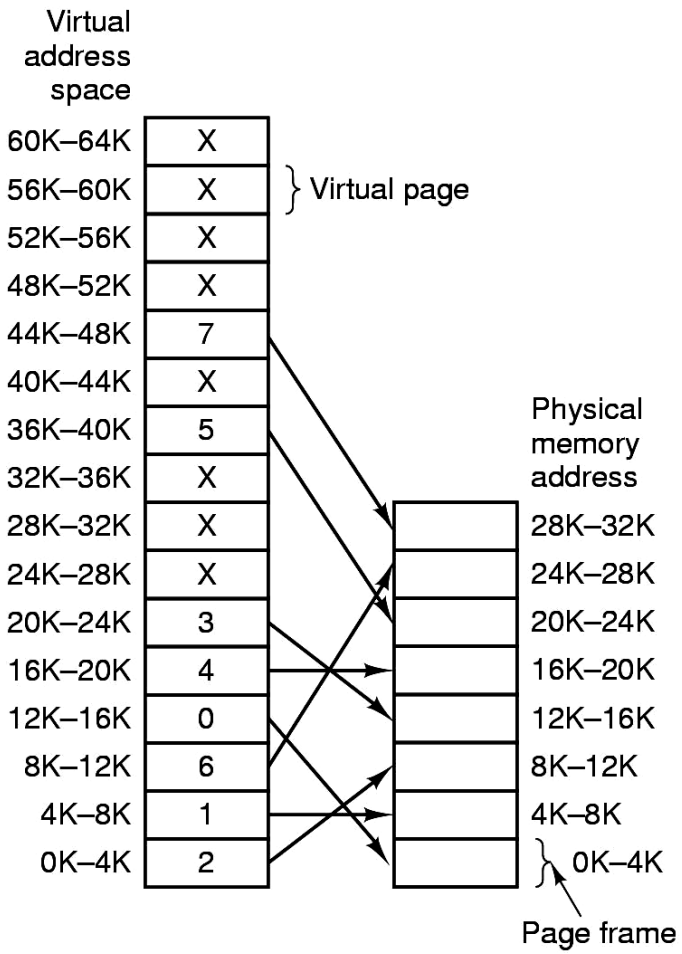
Bộ nhớ ảo

- Kỹ thuật phân trang (tiếp)



Bộ nhớ ảo

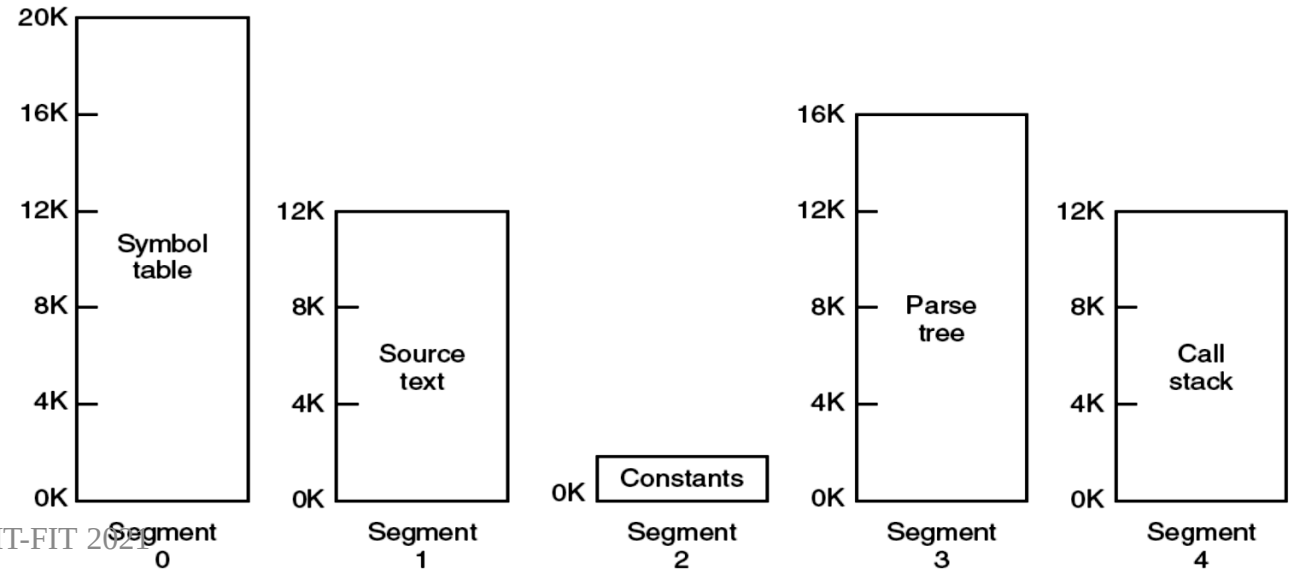
- Ví dụ về BN phân trang
  - BN ảo trong CT gồm 64KB được chia ra 16 trang, mỗi trang 4KB
  - BN thực gồm 32KB được chia ra 8 frame
  - BN đang được cấp phát như thể hiện trong bảng trang



Bài tập: Hãy tính địa chỉ thật từ các địa chỉ ảo
- 10.000
- 20.000
- 30.000

Bộ nhớ ảo

- Kỹ thuật phân đoạn (segmentation)
  - Quan điểm người lập trình về BN
    - Chương trình bao gồm nhiều module
    - Dữ liệu bao gồm nhiều array, chuỗi, …
    - Khi truy cập sẽ căn cứ vào địa chỉ tương đối của module (lệnh thứ mấy) hay array (phần tử thứ mấy)

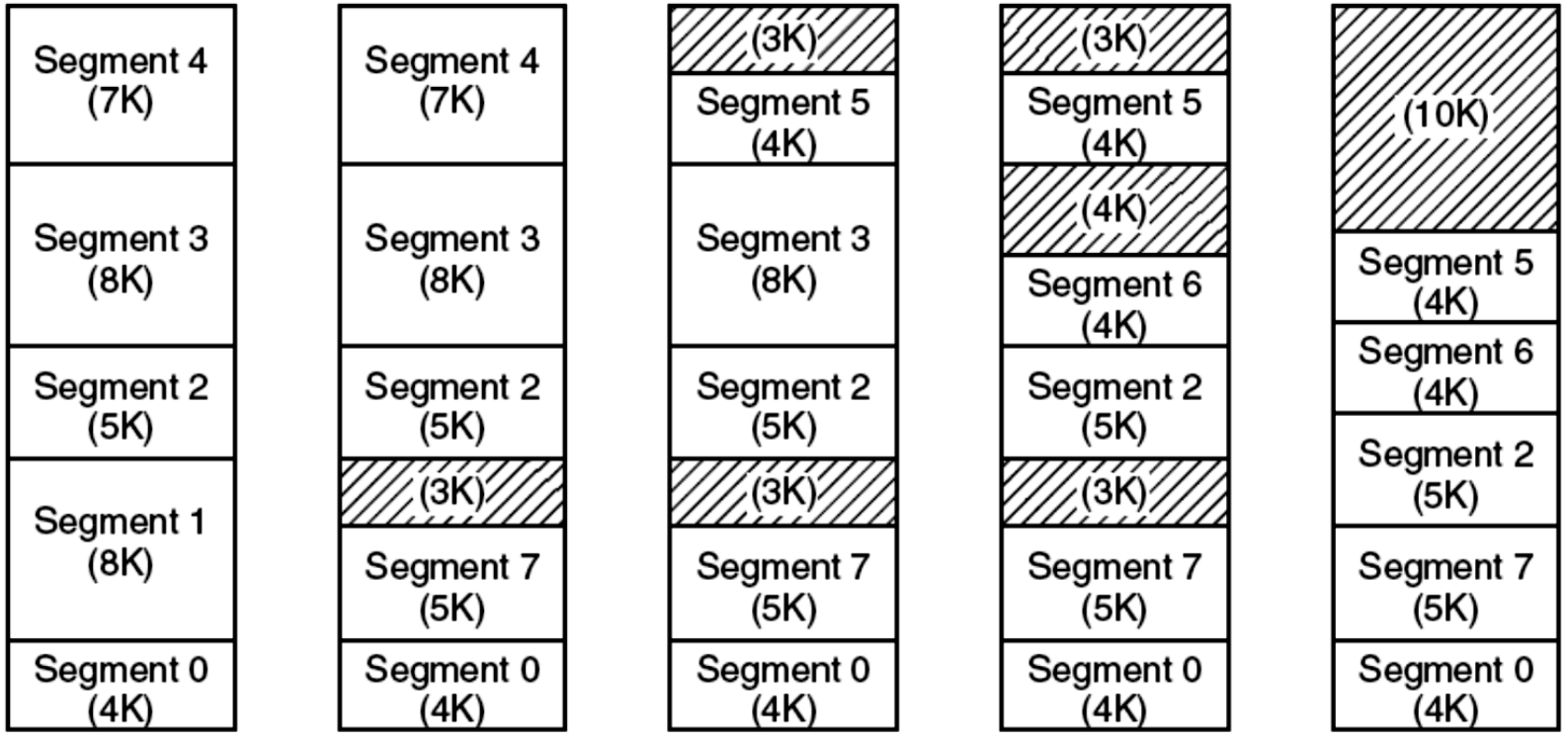


Bộ nhớ ảo

- Kỹ thuật phân đoạn (tiếp)
  - HĐH sẽ cấp phát BN theo từng đoạn (segment) có kích thước theo yêu cầu lập trình, người lập trình truy cập BN theo offset trong từng segment
  - Địa chỉ ảo có dạng (segment, offset)
  - Khi có yêu cầu cấp phát BN, HĐH có thể nạp 1 segment theo yêu cầu vào vùng trống trong BN thật. Nếu không có vùng trống đủ lớn HĐH cần dồn BN để tạo ra vùng trống đủ lớn.
  - Khi CT truy cập vào 1 segment chưa được cấp phát sẽ gây ra lỗi segment (segment fault) → HĐH phải xử lý bằng cách swapping với 1 hoặc vài segment khác chưa cần sử dụng đến (chậm)
  - HĐH cần 1 bảng quản lý để theo dõi segment nào đang được nạp vào vị trí nào trong BN cho mỗi CT, gọi là bảng segment (segment table)

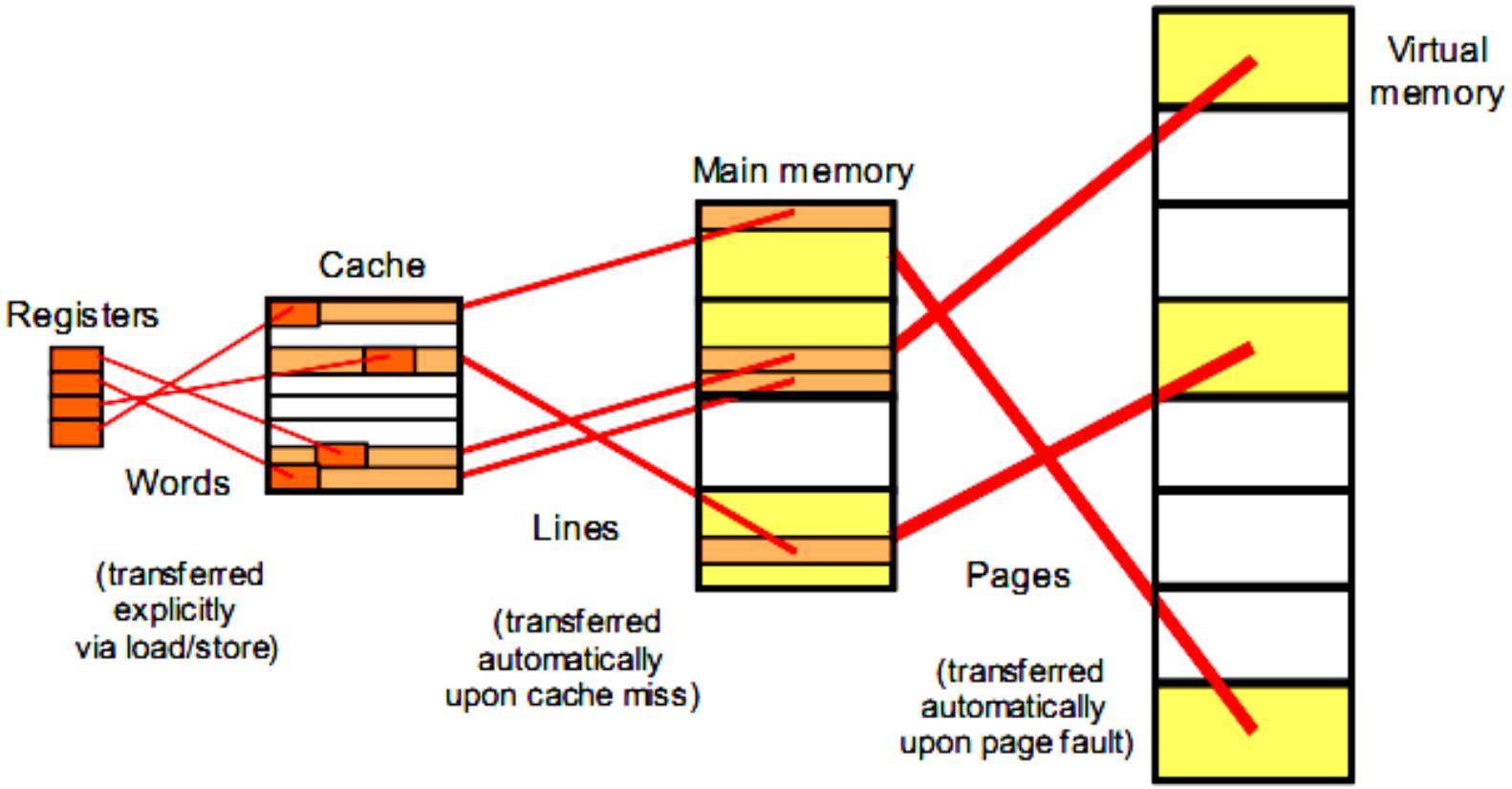
Bộ nhớ ảo

- Ví dụ về BN phân đoạn



Bộ nhớ ảo

- Tổng quát việc truy cập bộ nhớ trong máy tính

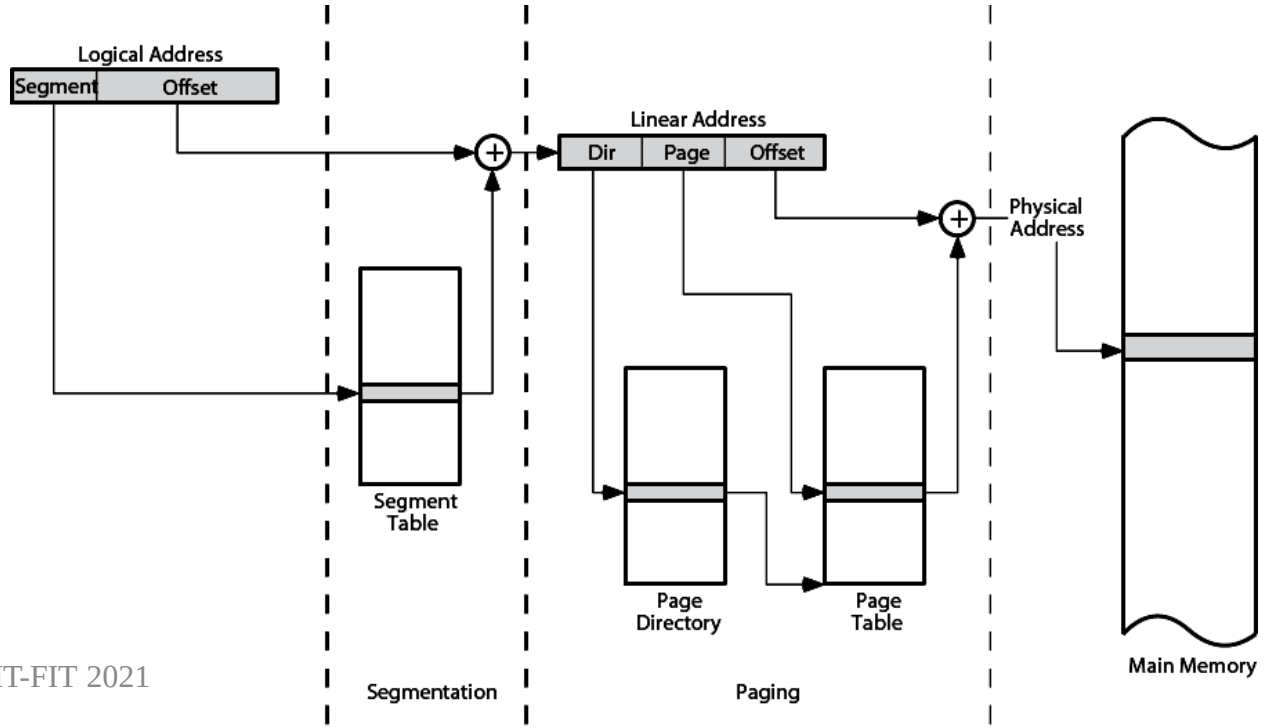


Bộ nhớ ảo

- Ưu điểm BN ảo
  - Cho phép CT lớn hơn BN vẫn chạy được
  - Chỉ nạp phần CT nào cần đến vào BN → tiết kiệm BN
- Nhược điểm BN ảo
  - Tăng phí tổn hệ thống (overhead): Tốn thời gian tính toán địa chỉ ảo sang địa chỉ thật, tốn không gian BN chứa bảng trang/ segment
  - Truy cập BN chậm hơn so với quản lý BN thực: Cần gấp đôi thời gian truy cập BN. Khi có page/ segment fault việc truy cập BN biến thành truy cập IO
- Cách khắc phục
  - Cần phần cứng đặc biệt hỗ trợ HĐH để quản lý BN
  - Cần giải thuật thay trang/ segment tối ưu

Bộ nhớ ảo

- Ví dụ: BN ảo trong CPU Intel Pentium 4
  - Phân segment kết hợp phân trang 2 cấp
    - Phân segment: Segment 16 bit, Offset: 32 bit.
    - Phân trang: Địa chỉ tuyến tính 32 bit chia ra: Directory 10 bit, page 10 bit và offset 12 bit (4KB/trang)



Các giải thuật thay thế trang
(Page replacement algorithms)

Thay thế trang

- Không dễ dàng để tìm được chính sách thay thế trang tốt
  - Khi thu hồi một trang, làm sao chúng ta biết là trang tốt nhất có thể giảm thiểu lỗi trang sau này?
- Có tồn tại thuật toán thay thế trang tối ưu?
- Nếu có, thuật toán thay thế trang tối ưu là gì?
- Xem ví dụ sau:
  - Giả sử chúng ta có 3 frames và chạy chương trình theo mẫu sau
  - 7, 0, 1, 2, 0, 3, 0, 4, 2, 3
- Giả sử chúng ta biết thứ tự yêu cầu trang

Giải thuật thay thế trang

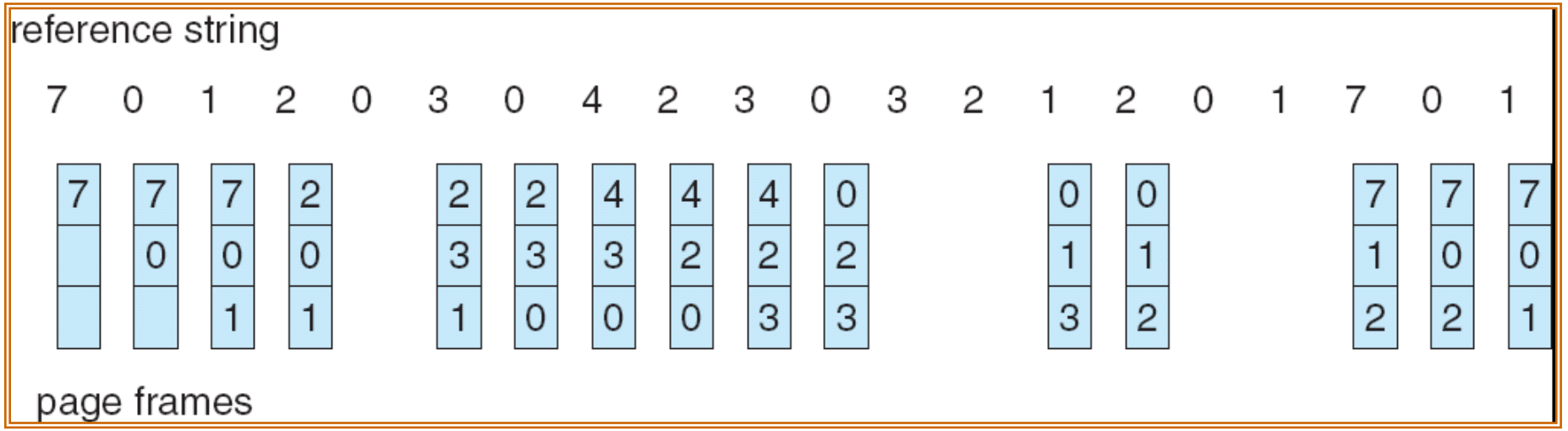
- FIFO
- LRU
- OPTIMAL

FIFO - First In First Out

- First-in, First-out
  - Công bằng, thời gian mỗi trang trên bộ nhớ gần như tương đương nhau
- Có vấn đề gì không?
  - Có phù hợp với yêu cầu của một chương trình?
- Có hiệu quả với ví dụ của chúng ta?

7, 0, 1, 2, 0, 3, 0, 4, 2, 3

FIFO - First In First Out



Ví dụ khác - FIFO

Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

3 frames (3 pages can be in memory at a time per process)

1	1	4	5	
2	2	1	3	9 page faults
3	3	2	4	

4 frames

1	1	5	4	
2	2	1	5	10 page faults
3	3	2		
4	4	3		

Belady's Anomaly: more frames ⇒ more page faults

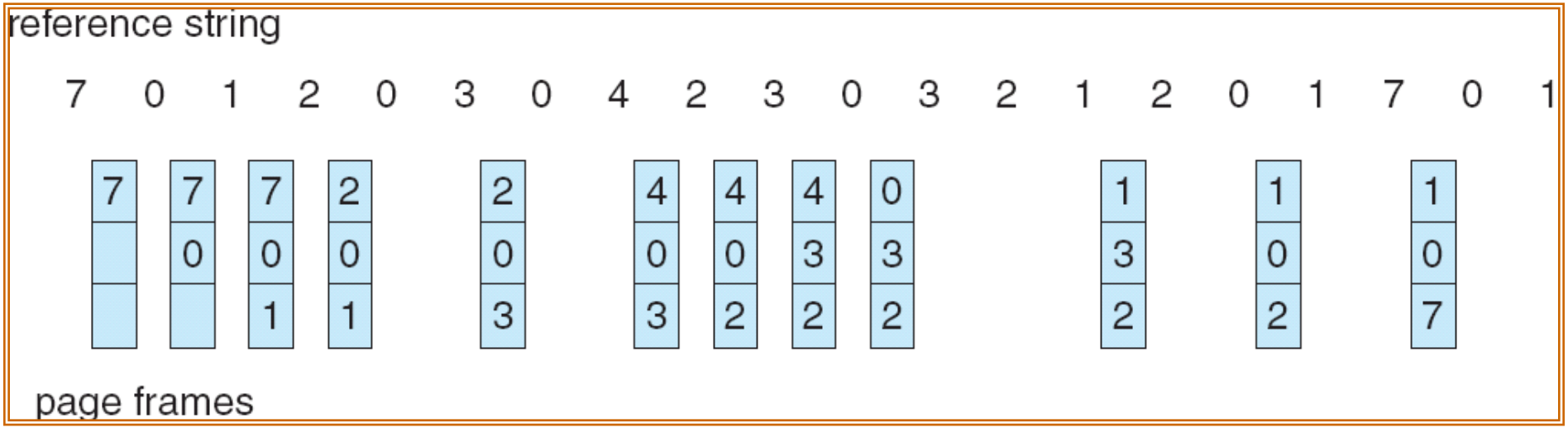
LRU – Least Recently Used

- Least Recently Used (ít sử dụng gần đây nhất)
  - Mỗi lần truy cập trang, dán nhãn thời gian lại
  - Khi cần thu hồi một trang, chọn trang với nhãn thời gian lâu nhất
- LRU có phải tối ưu nhất?
  - Trong thực tế, LRU là giải pháp tốt cho hầu hết chương trình
- Có dễ dàng cài đặt?

Thay thế trang ít sử dụng nhất

- Dùng 1 reference bit và bộ đếm cho mỗi trang (frame)
- Mỗi ngắt đồng hồ, HĐH cộng reference bit vào biến counter rồi xóa reference bit (bật reference bit nếu trang được sử dụng)
- Khi cần thay trang, chọn trang có số đếm ít nhất
- Có vấn đề gì không?
  - Lưu vết tất cả, khó thu hồi trang đã dùng rất nhiều trong quá khứ, nhưng hiện tại không còn dùng nữa
  - Chi phí cao để quản lý counter, bởi vì bộ nhớ ngày càng lớn
- Có thể cải tiến bằng lược đồ độ tuổi: counter được shift qua phải trước khi cộng reference bit và reference bit được cộng vào bit trái nhất

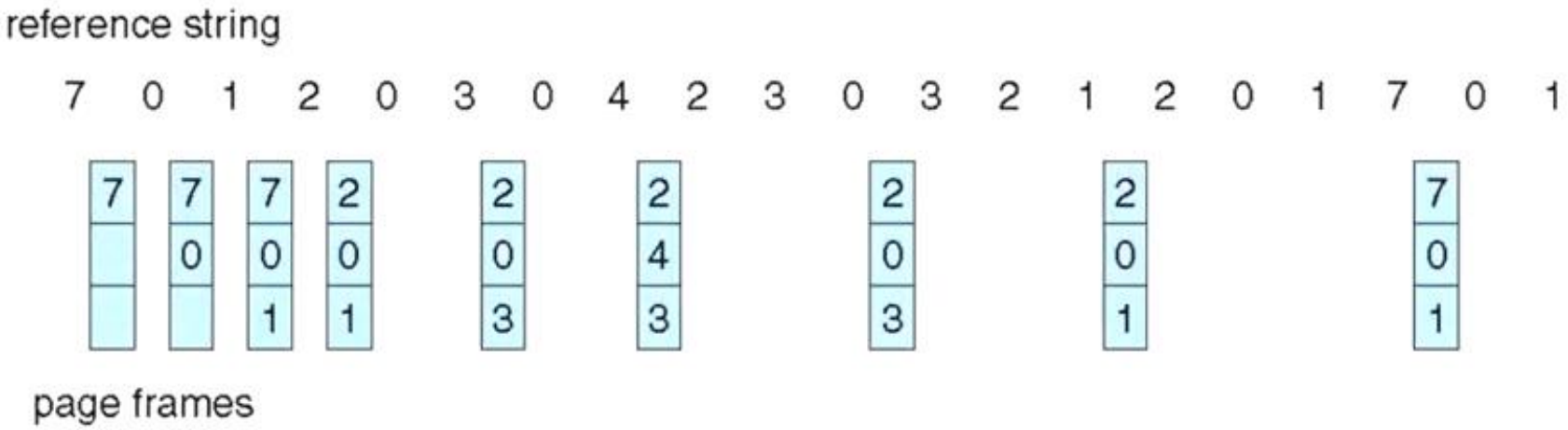
LRU – Least Recently Used



OPTIMAL

- Thuật toán tối ưu là **thay thế trang sẽ không dùng lại lâu nhất**
- Vấn đề của thuật toán này là gì?
- Giải pháp thực tế là dự đoán tương lai(sẽ yêu cầu trang nào) bằng quá khứ
  - Có thể đúng vì tính cục bộ

OPTIMAL



Clock (Cơ hội thứ hai)

- Sắp xếp các trang thành vòng tròn, và dùng 1 đồng hồ
- Dùng 1 use bit cho mỗi frame. Bật use bit lên khi mà frame đó được dùng.
  - Nếu use bit = 0, trang không sử dụng
- Khi lỗi trang:
  - Di chuyển kim đồng hồ
  - Kiểm tra use bit
    - If 1, mới sử dụng, xóa và tiếp tục
    - If 0, chọn trang này để thay thế
- Liệu chúng ta có thể luôn tìm được trang để thay thế?

Cơ hội thứ N^{th}

- Tương tự ý tưởng trên nhưng,
- Dùng 1 counter và 1 ***use bit***
- Khi lỗi trang:
  - Dịch kim đồng hồ
  - Kiểm tra ***use bit***
    - If 1, xóa use bit và đặt counter = 0
    - If 0, tăng counter, if counter < N, tiếp tục, ngược lại chọn trang này để thay thế
- Nhận xét
  - N lớn $\Rightarrow$ gần giống kết quả với LRU
- Nếu N quá lớn thì gặp vấn đề gì?

Tiếp cận khác của cơ hội thứ 2nd

- Luôn giữ một danh sách n > 0 trang trống
  - Khi lỗi trang, nếu danh sách trống có hơn n frames, chọn 1 frame từ danh sách trống
  - Nếu danh sách trống chỉ có n frames, chọn 1 frame từ danh sách, rồi chọn một frame đang sử dụng để đặt vào danh sách trống
- Khi lỗi trang, nếu trang lỗi là trong danh sách trống, không phải đọc lại trang đó lên bộ nhớ.
- Triển khai trên VAX … hiệu quả gần đạt LRU

Tập trang thường trú (working set)

- Một tiến trình nên nạp bao nhiêu trang lên bộ nhớ ?
- Số lượng trang thường trú này có thể cố định hoặc thay đổi
- Miền thay thế là cục bộ hay toàn cục
- Lược đồ hay được sử dụng:
  - Thay trang toàn cục: đơn giản – kích thước tập trang thường trú của tiến trình thay đổi mỗi lẫn thay trang
  - Thay trang cục bộ: phức tạp hơn – kích thước tập trang thường trú phải thay đổi xung quanh giá trị kích thước tập trang thường trú của tiến trình (working set size)

Working Set

- Là một tập các trang được sử dụng trong khoảng thời gian gần đây nhất
- Kích thước của working set có thể thay đổi trong suốt quá trình thực thi của tiến trình
- Nếu số lượng trang được cấp nhiều hơn working set thì số lỗi trang sẽ nhỏ
- Chỉ điều phối cho tiến trình khi mà bộ nhớ đủ để nạp working set của nó
- Làm sao để xác định/(xấp xỉ) kích thước của working set?

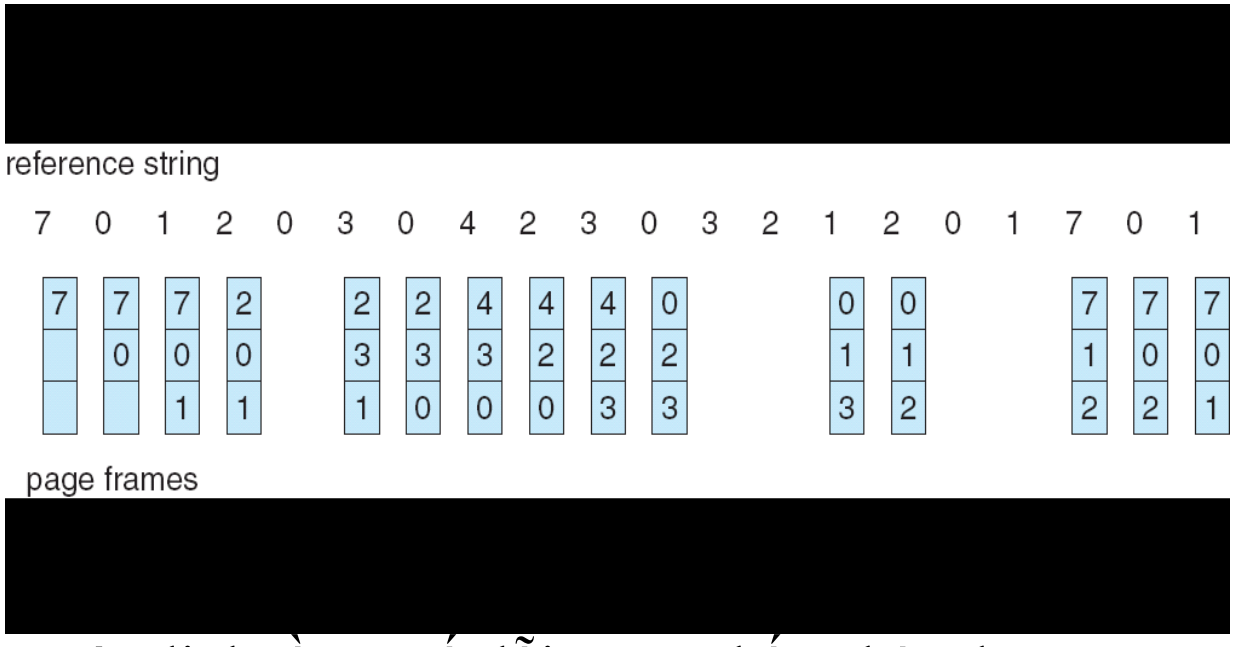
Working-Set

- $\Delta \equiv$ working-set window $\equiv$ số trang được gọi
  Ví dụ: Δ = 10,000 lệnh
- WSS_i (working set của tiến trình P_i) =
  tổng số trang được gọi trong khoảng tgian Δ vừa rồi(có thể thay đổi)
  - if Δ quá nhỏ thì không đủ chứa tập trang thường trú.
  - if Δ quá lớn thì có thể chứa nhiều tập trang thường trú.
  - if $\Delta = \infty \Rightarrow$ sẽ chứa tập trang toàn chương trình.
- $D = \Sigma\ WSS_i \equiv$ tổng trang được yêu cầu
- if $D > m \Rightarrow$ Trì trệ (Thrashing)
- Chính sách if $D >$ m, thì dừng một tiến trình.

Lưu vết Working Set

- Xấp xỉ khoảng thời gian + dùng một reference bit
- Ví dụ: Δ = 10,000
  - Đồng hồ ngắt sau mỗi 5000 đơn vị.
  - Dùng 2 reference bít cho mỗi trang.
  - Mỗi lần đồng hồ ngắt, thì lưu lại và gán lại giá trị 0 cho cả 2 reference bit.
  - Nếu 1 bit = 1 $\Rightarrow$ trang trong working set.
- Tại sao không thật sự chính xác?
- Cải tiến = dùng 10 bits và ngắt mỗi 1000 đơn vị.

Biểu đồ tần suất lỗi trang



Xác định tầng suất lỗi trang chấp nhận được.

Nếu tỉ lệ lỗi trang nhỏ, tiến trình bỏ bớt frame.

Nếu tỉ lệ lỗi trang cao, tiến trình cấp thêm frame.

Tầng suất lỗi trang

- Một bộ đếm cho mỗi trang để đếm "thời gian" giữa các lỗi trang("thời gian" = có thể là số lần trang được truy cập)
- Định nghĩa một ngưỡng trên cho biến "thời gian"
- Nếu thời gian giữa 2 lỗi trang nhỏ hơn ngưỡng trên, thì trang được thêm vào tập thường trú
- Và cũng cần một ngưỡng dưới để giảm bớt khung trang của tiến trình

Câu hỏi