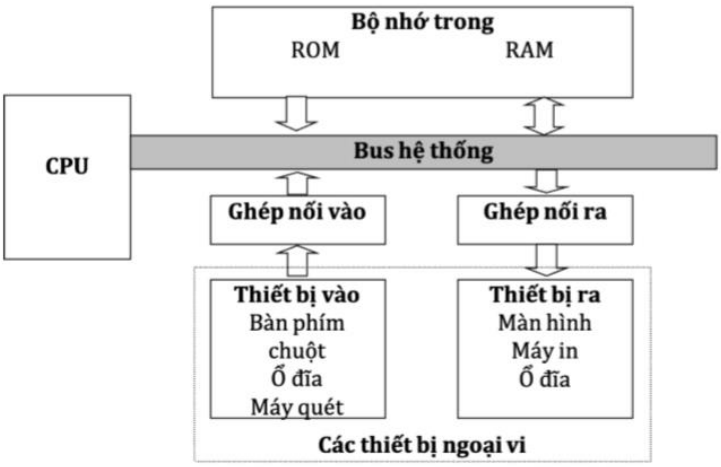
Chương 1: Tổng quan về kiến trúc máy tính

1. Khái niệm kiến trúc máy tính, các thành phần của kiến trúc máy tính.

- Kiến trúc máy tính là khoa học về lựa chọn và kết nối các thành phần phần cứng của máy tính nhằm đạt yêu cầu:
  - Hiệu năng: càng nhanh càng tốt
  - Chức năng: đáp ứng nhiều chức năng
  - Giá thành: càng rẻ càng tốt
- Thành phần cơ bản của kiến trúc máy tính gồm 3 thành phần:
  - Kiến trúc tập lệnh: Mô hình trừu tượng của máy tính ở mức ngôn ngữ máy (hợp ngữ) xác định những gì bộ xử lý thực hiện và cách thực hiện:
    + Tập lệnh
    + Các chế độ địa chỉ bộ nhớ
    + Các thanh ghi
    + Khuôn dạng địa chỉ và dữ liệu
  - Vi kiến trúc: là tổ chức máy tính, mô tả về hệ thống ở mức thấp, liên quan tới:
    + Các thành phần phần cứng kết nối với nhau như thế nào
    + Các thành phần phần cứng phối hợp, tương tác với nhau như thế nào để thực hiện tập lệnh
  - Thiết kế hệ thống: bao gồm tất cả các thành phần phần cứng khác trong hệ thống máy tính:
    + Các hệ thống kết nối như bus và chuyển mạch
    + Mạch điều khiển bộ nhớ, cấu trúc phân cấp bộ nhớ
    + Các kỹ thuật giảm tải cho CPU như truy cập trực tiếp bộ nhớ
    + Các vấn đề như đa xử lý

2. Sơ đồ khối, chức năng các thành phần trong 1 hệ thống máy tính.

- Sơ đồ khối chức năng của hệ thống máy tính:



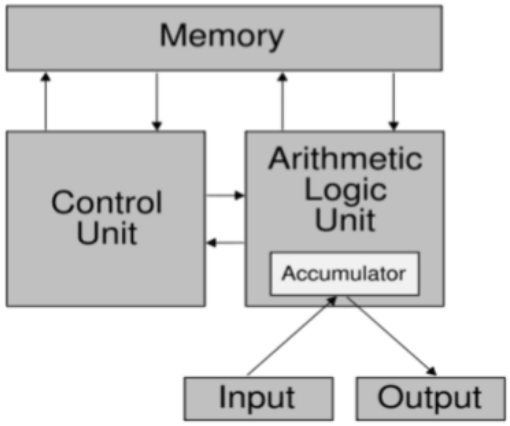
- Các thành phần và chức năng trong 1 hệ thống máy tính:
  - Bộ xử lý trung tâm (CPU):
    + Bao gồm: Khối điều khiển CU, khối tính toán số học và logic ALU, các thanh ghi và bus trong CPU.
    + Chức năng: Đọc lệnh từ bộ nhớ; Giải mã và thực hiện lệnh.
  - Bộ nhớ trong:

+ Bao gồm: ROM (lưu trữ lệnh và dữ liệu của hệ thống, thông tin vẫn tồn tại khi mất nguồn nuôi) và RAM (lưu trữ lệnh và dữ liệu của hệ thống và người dùng, thông tin sẽ mất khi mất nguồn nuôi).
+ Chức năng: Lưu trữ lệnh và dữ liệu để CPU xử lý.

- Các thiết bị vào ra:
  + Thiết bị vào: Nhập dữ liệu và điều khiển (bàn phím, chuột, ổ đĩa,…)
  + Thiết bị ra: Kết xuất dữ liệu (màn hình, máy in, loa,…)
- Bus hệ thống:
  + Bao gồm 3 loại: Bus địa chỉ (bus A), bus dữ liệu (bus D) và bus điều khiển (bus C)
  + Chức năng: Là tập các đường dây kết nối CPU với các thành phần khác của máy tính.

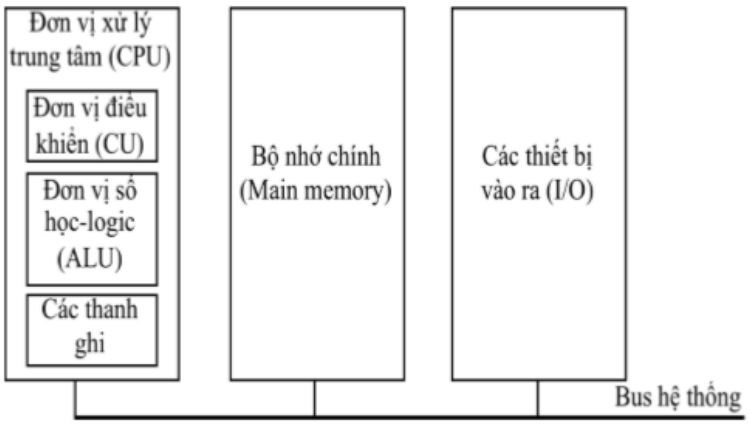
3. Sơ đồ và đặc điểm của kiến trúc máy tính Von-Neumann cổ điển. Vẽ sơ đồ và phân biệt kiến trúc máy tính von-Neumann hiện đại với cổ điển?

- Sơ đồ kiến trúc Von-Neumann cổ điển:



- Đặc điểm kiến trúc Von-Neumann cổ điển:
  - Dựa trên 3 khái niệm cơ bản:
    + Lệnh và dữ liệu được lưu trong 1 bộ nhớ đọc/ viết chia sẻ.
    + Bộ nhớ được đánh địa chỉ dựa trên đoạn và không phụ thuộc việc nó lưu trữ gì.
    + Các lệnh của chương trình được chạy lần lượt, lệnh nọ tiếp sau lệnh kia.
  - Quá trình thực hiện lệnh chia làm 3 giai đoạn chính:
    + CPU lấy lệnh (fetch) từ bộ nhớ
    + CPU giải mã lệnh và chạy lệnh; nếu lệnh cần dữ liệu thì đọc dữ liệu từ bộ nhớ
    + CPU viết kết quả vào bộ nhớ nếu có.

- Sơ đồ kiến trúc Von-Neumann hiện đại:
- Kiến trúc Von-Neumann hiện đại có sử dụng bus hệ thống so với kiến trúc cổ điển, cung cấp 1 con đường để truyền tải dữ liệu giữa các thành phần CPU, bộ nhớ và các thiết bị vào ra 1 cách nhanh chóng và hiệu quả hơn.



4. Sơ đồ và đặc điểm của kiến trúc máy tính Harvard. Những ưu điểm so với kiến trúc Von-neuman?

- Sơ đồ kiến trúc Harvard:

- Đặc điểm kiến trúc Harvard:
  - Bộ nhớ được chia làm 2 phần:
    + Bộ nhớ chương trình (Instruction Memory)
    + Bộ nhớ dữ liệu (Data Memory)
  - CPU sử dụng 2 bus hệ thống để liên hệ với bộ nhớ:
    + CPU có thể đọc lệnh và truy cập dữ liệu bộ nhớ cùng một lúc.
    + Một bus A, D cho bộ nhớ chương trình và 1 bus A, D cho bộ nhớ dữ liệu (khác nhau về định dạng)
- Ưu điểm của kiến trúc Harvard so với kiến trúc Von-Neumann:
  - Tốc độ xử lý nhanh hơn vì băng thông bus rộng, quá trình đọc lệnh không tranh chấp với quá trình truy xuất dữ liệu.
  - Hỗ trợ nhiều truy cập đọc/viết bộ nhớ cùng lúc nên giảm xung đột truy cập bộ nhớ.
  - (Được sử dụng trong vi điều khiển và xử lý tín hiệu; Von-Neumann được sử dụng trong máy tính cá nhân, xách tay và các máy trạm.)

5. Các máy tính để bàn, xách tay hiện nay thường sử dụng kiến trúc nào?

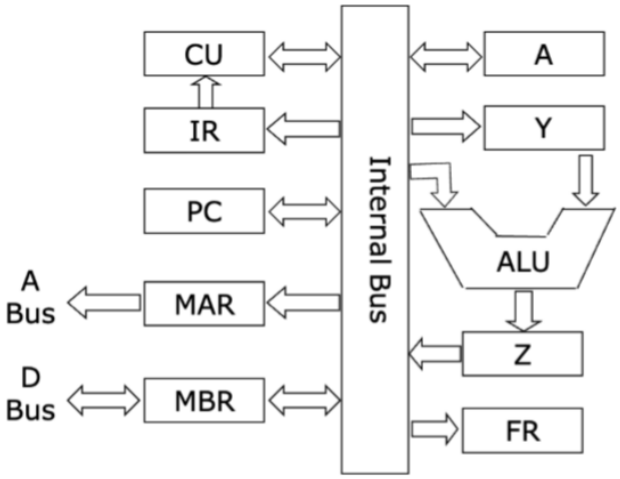
Các máy tính để bàn, xách tay hiện đại ngày nay thường sử dụng kiến trúc máy tính von Neumann hiện đại vì kiến trúc này ít phức tạp và rẻ hơn so với kiến trúc Harvard mặc dù Harvard có nhiều ưu việt hơn. Ngoài ra các máy tính đã phân cấp bộ nhớ nên tốc độ đã được cải thiện đáng kể. Hơn nữa, kiến trúc Von-

Neumann ra đời trước nên các nhà sản xuất máy tính đã phát triển công nghệ sản xuất trên nền tảng này, sự thay đổi là rất tốn kém.

Chương 2: Khối xử lí trung tâm CPU

1. Sơ đồ khối tổng quát, chu trình xử lí lệnh của CPU.

- Sơ đồ khối tổng quát của CPU:



- Chu trình xử lý lệnh của CPU:
  - Bước 1: Khi một chương trình được chạy, hệ điều hành tải mã chương trình vào bộ nhớ trong RAM
  - Bước 2: Địa chỉ lệnh đầu tiên của chương trình được đưa vào thanh ghi PC
  - Bước 3: Địa chỉ của ô nhớ chứa lệnh được chuyển tới bus A qua thanh ghi MAR
  - Bước 4: Bus A truyền địa chỉ tới khối quản lý bộ nhớ MMU.
  - Bước 5: MMU chọn ô nhớ và sinh ra tín hiệu READ
  - Bước 6: Lệnh chứa trong ô nhớ được chuyển tới thanh ghi MBR qua bus D
  - Bước 7: MBR chuyển lệnh tới thanh ghi IR. Sau đó IR lại chuyển lệnh tới CU
  - Bước 8: CU giải mã lệnh và sinh ra các tín hiệu xử lý cho các đơn vị khác, ví dụ như ALU,…
  - Bước 9: Địa chỉ trong PC được tăng lên để trỏ tới lệnh tiếp theo của chương trình sẽ được thực hiện
  - Bước 10: Thực hiện lại các bước 3->9 để chạy hết các lệnh của chương trình

2. Khái niệm thanh ghi trong CPU. Đặc điểm, chức năng các thanh ghi.

- Thanh ghi là thành phần nhớ ở bên trong CPU:
  - Lưu trữ tạm thời lệnh và dữ liệu cho CPU xử lý
  - Dung lượng nhỏ, số lượng ít
  - Tốc độ rất cao (bằng tốc độ CPU)
- Thanh ghi A (thanh ghi tích luỹ)
  - Đặc điểm: Là 1 trong những thanh ghi quan trọng nhất của CPU. Kích thước thanh ghi tương ứng với độ dài từ xử lý của CPU: 8, 16, 32, 64bit.
  - Chức năng:
    + Lưu trữ các toán hạng đầu vào và kết quả đầu ra.
    + Sử dụng để trao đổi dữ liệu với các thiết bị vào ra.
- Bộ đếm chương trình PC (thanh ghi lệnh IP)
  - Đặc điểm: Kích thước của PC phụ thuộc vào thiết kế CPU: 8, 16, 32, 64bit

- Chức năng: PC chứa địa chỉ ô nhớ chứa lệnh đầu tiên của chương trình khi nó được kích hoạt và được tải vào bộ nhớ. Khi CPU chạy xong 1 lệnh, địa chỉ lệnh tiếp theo được tải và lưu vào trong PC.

- Thanh ghi cờ (thanh ghi trạng thái) FR
  - Đặc điểm:
    + Mỗi bit của thanh ghi cờ lưu trữ trạng thái kết quả phép tính được ALU thực hiện.
    + Có 2 kiểu cờ: Cờ trạng thái (CF, OF, AF, ZF, PF, SF) và cờ điều khiển (IF, TF, DF)
    + Kích thước FR phụ thuộc vào thiết kế CPU.
  - Chức năng: Các bit cờ thường được dùng là các điều kiện rẽ nhánh lệnh tạo logic chương trình.
- Con trỏ ngăn xếp (SP)
  - Ngăn xếp là 1 đoạn bộ nhớ đặc biệt hoạt động theo nguyên tắc vào sau ra trước (LIFO)
  - PC chứa địa chỉ ô nhớ chứa lệnh đầu tiên của chương trình khi nó được kích hoạt và được tải vào bộ nhớ
  - Con trỏ ngăn xếp là thanh ghi luôn trỏ tới đỉnh của ngăn xếp
  - 2 thao tác với ngăn xếp:
    + Push: đẩy dữ liệu vào ngăn xếp:

      SP <– SP + 1

      SP <– Data

    + Pop: lấy dữ liệu ra khỏi ngăn xếp:

      Register <– SP

      SP <– SP – 1

- Thanh ghi lệnh IR

  Chức năng: IR lấy lệnh đang được xử lý từ MBR, lưu trữ và chuyển nó tới CU để giải mã lệnh.
- Thanh ghi MAR

  Chức năng: Đọc địa chỉ ô nhớ trong bộ nhớ và lưu trữ trong thanh ghi, sau đó chuyển tới bus địa chỉ A. CPU sẽ sử dụng địa chỉ trong thanh ghi để truy cập ô nhớ tương ứng sau đó lấy hoặc ghi dữ liệu vào đó.
- Thanh ghi MBR

  Chức năng: Lưu trữ dữ liệu được đọc từ bộ nhớ (qua bus dữ liệu D) và chuyển nó tới IR.

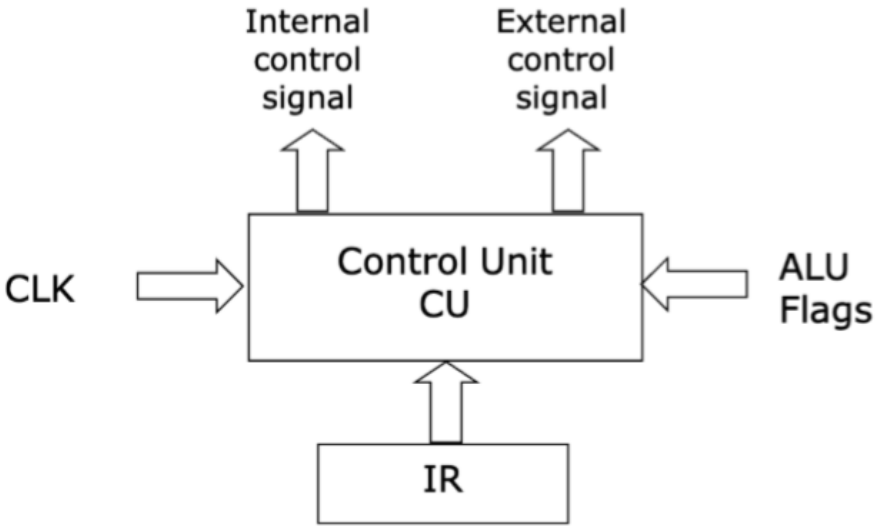
3. Ý nghĩa và nêu VD về các cờ trong thanh ghi cờ FR (Z, S, C, A, O, I, P)

- ZF (Zero Flag): ZF=1 nếu kết quả bằng 0 và =0 nếu kết quả khác 0.
- SF (Sign Flag/ Cờ dấu): SF=1 nếu kết quả âm và SF=0 nếu kết quả dương
- CF (Carry Flag/ Cờ nhớ): CF=1 nếu có nhớ/mượn ở bit trái nhất
- AF (Auxiliary Flag/ Cờ nhớ phụ): AF=1 nếu có nhớ ở bit trái nhất của nibble
- OF (Overflow Flag/ Cờ tràn): OF=1 nếu có tràn, OF=0 nếu không tràn.
- PF (Parity Flag/ Cờ chẵn lẻ): PF=1 nếu tổng số bit 1 trong kết quả là số lẻ, PF=0 nếu tổng số bit 1 trong kết quả là số chẵn.

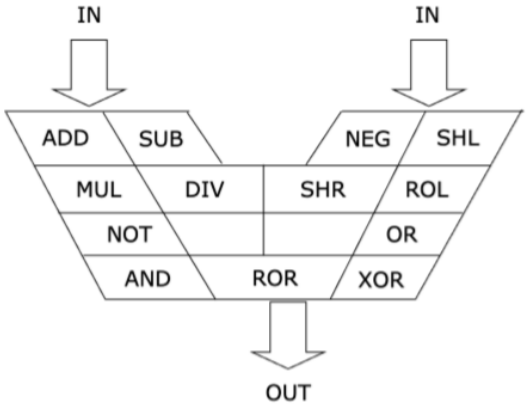
- IF (Interrupt Flag/ Cờ ngắt): IF=1: ngắt được phép, IF=0: cấm ngắt

4. Sơ đồ khối, chức năng của các khối trong CPU (khối điều khiển CU, khối tính toán số học và logic ALU, bus trong CPU)

- Sơ đồ khối điều khiển CU:



- Chức năng khối điều khiển CU:
  - Điều khiển tất cả các hoạt động của CPU theo xung nhịp đồng hồ.
  - Nhận 3 tín hiệu đầu vào: Lệnh từ IR; giá trị các cờ trạng thái; xung đồng hồ.
  - Sinh 2 nhóm tín hiệu đầu ra: Nhóm tín hiệu điều khiển các bộ phận bên trong CPU và nhóm tín hiệu điều khiển các bộ phận bên ngoài CPU.
  - Sử dụng nhịp đồng hồ để đồng bộ hóa các đơn vị bên trong CPU và giữa CPU với các thành phần bên ngoài.
- Sơ đồ khối tính toán số học và logic ALU:



- Chức năng khối tính toán số học và logic ALU: Đảm nhận nhiệm vụ tính toán trong CPU. Bao gồm các đơn vị chức năng con để thực hiện các phép toán số học và logic:
  - Bộ cộng (ADD), bộ trừ (SUB), bộ nhân (MUL), bộ chia (DIV), ...
  - Các bộ dịch (SHIFT) và quay (ROTATE).
  - Bộ logic như phủ định (NOT), bộ và (AND), bộ hoặc (OR), và bộ hoặc loại trừ (XOR).
- Bus trong CPU:
  - Bus trong là kênh liên lạc của tất cả các thành phần trong CPU
  - Hỗ trợ liên lạc 2 chiều
  - Bus trong có giao diện để trao đổi thông tin với bus ngoài
  - Bus trong có băng thông lớn và tốc độ nhanh hơn so với bus ngoài

Chương 3.1: Xử lý xen kẽ dòng mã lệnh

1. Khái niệm lệnh máy tính, chu kì thực hiện lệnh và các giai đoạn, các pha trong chu trình thực hiện lệnh.

- Lệnh máy tính là một từ nhị phân được gán một nhiệm vụ cụ thể, hướng dẫn cho máy tính biết phải làm gì.
  - Lệnh chương trình được lưu trong bộ nhớ
  - Lệnh được đọc từ bộ nhớ vào CPU để giải mã và thực hiện
  - Mỗi lệnh có một chức năng riêng
- Chu kì lệnh là khoảng thời gian để CPU thực hiện xong một lệnh kể từ khi CPU cấp phát tín hiệu địa chỉ ô nhớ chứa lệnh đến khi nó hoàn tất việc thực hiện lệnh đó. (1 chu kỳ lệnh có thể gồm 1 số giai đoạn thực hiện lệnh, 1 giai đoạn thực hiện lệnh có thể gồm 1 số chu kỳ máy, 1 chu kỳ máy có thể gồm 1 số chu kỳ đồng hồ)
- Việc thực hiện lệnh có thể được chia thành các pha hay giai đoạn. Mỗi lệnh có thể được thực hiện theo 4 giai đoạn:
  - Đọc lệnh (IF): lệnh được đọc từ bộ nhớ vào CPU
  - Giải mã lệnh (ID): CPU giải mã lệnh
  - Chạy lệnh (EX): CPU thực hiện lệnh
  - Ghi (WB): kết quả (nếu có) ghi vào thanh ghi/bộ nhớ

2. Dạng các toán hạng 0/ 1/ 1.5/ 2/ 3 địa chỉ. Cho VD? (?)

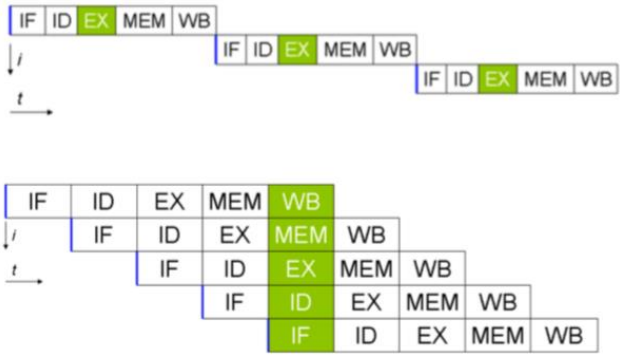
- Dạng toán hạng 3 địa chỉ: OPCODE Addr1, Addr2, Addr3 (mỗi địa chỉ Addri tham chiếu đến 1 ô nhớ hoặc 1 thanh ghi)
  VD: ADD R1, R2, R3; R1 <- R2 + R3 (Ri là các thanh ghi trong CPU)
- Dạng toán hạng 2 địa chỉ: OPCODE Addr1, Addr2 (mỗi địa chỉ Addri tham chiếu đến 1 ô nhớ hoặc 1 thanh ghi)
  VD: ADD R1, R2; R1 <- R1 + R2 (Ri là các thanh ghi trong CPU)
- Dạng toán hạng 1.5 địa chỉ: OPCODE Addr1, Addr2 (1 địa chỉ tham chiếu tới 1 ô nhớ và địa chỉ còn lại tham chiếu tới 1 thanh ghi)
  VD: ADD R1, B; R1 <- R1 + M[B]
- Dạng toán hạng 1 địa chỉ: OPCODE Addr (Địa chỉ Addr tham chiếu đến 1 ô nhớ hoặc 1 thanh ghi)
  VD: ADD R1; Racc <- Racc + R1 (R1 là thanh ghi trong CPU)
- Dạng toán hạng 0 địa chỉ: OPCODE (được thực hiện trong các lệnh mà thực hiện các thao tác ngăn xếp: PUSH và POP)
  VD: PUSH a
  PUSH b
  ADD POP c (nghĩa là c = a + b)

3. Các chế độ địa chỉ trong CPU (?)

Chế độ địa chỉ	Ý nghĩa	Ví dụ	Thực hiện
Tức thì	Giá trị của toán hạng được chứa trong lệnh	LOAD Ri, #1000	Ri ← 1000
Trực tiếp	Địa chỉ của toán hạng được chứa trong lệnh	LOAD Ri, 1000	Ri ← M[1000]
Gián tiếp thanh ghi	Giá trị của thanh ghi trong lệnh là địa chỉ bộ nhớ chứa toán hạng	LOAD Ri, (Rj)	Ri ← M[Rj]
Gián tiếp bộ nhớ	Địa chỉ bộ nhớ trong lệnh chứa địa chỉ bộ nhớ của toán hạng	LOAD Ri, (1000)	Ri ← M[M[1000]]
Chỉ số	Địa chỉ của toán hạng là tổng của hằng số (trong lệnh) và giá trị của một thanh ghi chỉ số	LOAD Ri, X(Rind)	Ri ← M[X+ Rind]
Tương đối	Địa chỉ của toán hạng là tổng của hằng số và giá trị của thanh ghi con đếm chương trình	LOAD Ri, X(PC)	Ri ← M[X+ PC]

4. Cơ chế xen kẽ dòng lệnh (cơ chế đường ống/ pipeline) là gì ? Ví dụ minh hoạ và đặc điểm phương pháp này.

- Cơ chế ống lệnh (pipeline) hay cơ chế thực hiện xen kẽ các lệnh của chương trình là 1 phương pháp thực hiện lệnh tiên tiến, giúp giảm thời gian trung bình thực hiện mỗi lệnh, tăng hiệu năng xử lý lệnh của CPU bằng cách cho phép đồng thời thực hiện nhiều lệnh, mỗi lệnh có thể được chia làm 5 giai đoạn thực hiện:
  - Đọc lệnh (IF): Lấy lệnh từ bộ nhớ (hoặc cache)
  - Giải mã lệnh (ID): Thực hiện giải mã lệnh và lấy các toán hạng
  - Thực thi lệnh (EX): Nếu là lệnh truy cập bộ nhớ thì tính toán địa chỉ bộ nhớ
  - Đọc - ghi bộ nhớ (MEM): Đọc và ghi bộ nhớ (nếu không truy cập bộ nhớ thì không có giai đoạn này)
  - Ghi (WB): Kết quả (nếu có) CPU xử lý được ghi vào thanh ghi/bộ nhớ lưu.



- Đặc điểm cơ chế ống lệnh:
  - Pipeline là kỹ thuật song song ở mức lệnh
  - Một pipeline đầy đủ luôn nhận 1 lệnh mới tại mỗi chu kỳ đồng hồ
  - Một pipeline không đầy đủ có các giai đoạn trễ trong quá trình xử lý
  - Số lượng giai đoạn của pipeline phụ thuộc vào thiết kế CPU:
    + 2, 3, 5 giai đoạn: pipeline đơn giản
    + 14 giai đoạn: Pen II, Pen III
    + 20 – 31 giai đoạn: Pen IV
    + 12 -15 giai đoạn: Core

5. Các vấn đề xảy ra và cách khắc phục khi sử dụng cơ chế ống lệnh

- Vấn đề xung đột tài nguyên:

- Xảy ra khi hệ thống không cung cấp đủ tài nguyên phần cứng phục vụ CPU thực hiện đồng thời nhiều lệnh trong cơ chế ống lệnh.
- 2 xung đột tài nguyên thường gặp nhất: Xung đột truy cập bộ nhớ và xung đột truy cập thanh ghi.
- Cách khắc phục:
  + Nâng cao khả năng các tài nguyên phần cứng.
  + Dùng hệ thống nhớ hỗ trợ nhiều thao tác đọc/ ghi cùng lúc hoặc sử dụng các bộ nhớ tiên tiến như bộ nhớ cache.
  + Chia cache thành cache lệnh và cache dữ liệu để cải thiện truy nhập.

- Vấn đề tranh chấp dữ liệu:
  - Tranh chấp dữ liệu kiểu đọc sau khi ghi (RAW) là khi dữ liệu chưa sẵn sàng cho các lệnh phụ thuộc tiếp theo gây xung đột dữ liệu hay sự phụ thuộc dữ liệu giữa các lệnh.
  - Cách khắc phục:
    + Nhận biết các lệnh phụ thuộc dữ liệu
    + Các lệnh phụ thuộc cần thực thi (EX) sau khi các lệnh mà nó phụ thuộc thực hiện xong ➔ Ngưng pipeline (stall): phải làm trễ hoặc ngưng pipeline bằng cách sử dụng một vài phương pháp tới khi có dữ liệu chính xác
    + Có thể sử dụng complier để nhận biết RAW và thực hiện:
      • Chèn các lệnh NO-OP vào giữa các lệnh có RAW (Lệnh NO-OP không thay đổi trạng thái CPU mà chỉ lấy đi 1 chu kỳ lệnh - thời gian 1 chu kỳ)
      • Thay đổi trình tự các lệnh trong chương trình và chèn các lệnh độc lập dữ liệu vào vị trí giữa 2 lệnh có RAW
    + Sử dụng phần cứng để xác định RAW (có trong các CPU hiện đại) và dự đoán trước giá trị dữ liệu phụ thuộc

- Vấn đề nảy sinh do các lệnh rẽ nhánh:
  - Các lệnh rẽ nhánh có thể gây ra:
    + Gián đoạn trong quá trình chạy bình thường của chương trình
    + Làm cho Pipeline rỗng nếu không có biện pháp ngăn chặn hiệu quả
  - Với các CPU mà pipeline dài (P4 với 31 giai đoạn) và nhiều pipeline chạy song song, vấn đề rẽ nhánh càng trở nên phức tạp hơn vì:
    + Phải đẩy mọi lệnh đang thực hiện ra ngoài pipeline khi gặp lệnh rẽ nhánh
    + Tải mới các lệnh từ địa chỉ rẽ nhánh vào pipeline. Tiêu tốn nhiều thời gian để điền đầy pipeline
  - Cách khắc phục:
    + Sử dụng đích rẽ nhánh (branch target):
      • Khi một lệnh rẽ nhánh được thực hiện, lệnh tiếp theo được lấy là lệnh ở địa chỉ đích rẽ nhánh (target) mà không phải lệnh tại vị trí tiếp theo lệnh nhảy
      • Các lệnh rẽ nhánh được xác định tại giai đoạn ID, vậy có thể biết trước chúng bằng cách giải mã trước
      • Sử dụng đệm đích rẽ nhánh (BTB: branch target buffer) để lưu vết của các lệnh rẽ nhánh đã được thực thi

• Cách này không hiệu quả với rẽ nhánh có điều kiện. Với rẽ nhánh có điều kiện, nên dùng 2 phương pháp sẽ được trình bày sau đây.

+ Làm chậm rẽ nhánh

• Lệnh rẽ nhánh không làm rẽ nhánh ngay lập tức mà nó sẽ bị làm chậm một vài chu kỳ đồng hồ phụ thuộc vào độ dài của pipeline.

• Phương pháp này khá hiệu quả với các ống lệnh ngắn (thường là 2 giai đoạn) và với ràng buộc là lệnh sau lệnh rẽ nhánh luôn được thực hiện, không phụ thuộc vào kết quả lệnh rẽ nhánh

• Cài đặt: Sử dụng complier để chèn NO-OP vào vị trí ngay sau lệnh rẽ nhánh hoặc chuyển một lệnh độc lập từ trước tới ngay sau lệnh rẽ nhánh.

+ Dự đoán rẽ nhánh: Có thể dự đoán lệnh đích của lệnh rẽ nhánh:

• Dự đoán đúng: nâng cao hiệu năng

• Dự đoán sai: đẩy các lệnh tiếp theo đã load và phải load lại các lệnh tại đích rẽ nhánh

• Trường hợp xấu của dự đoán là 50% đúng và 50% sai

• Các cơ sở để dự đoán:

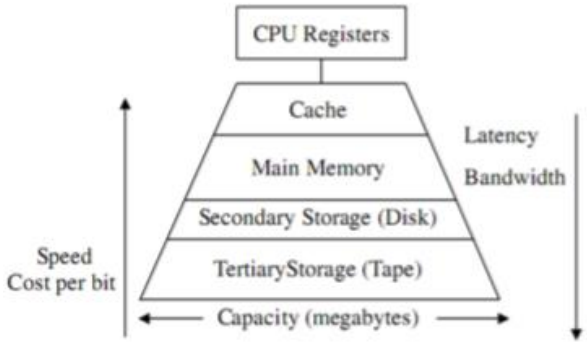
~ Đối với các lệnh nhảy ngược: Thường là một phần của vòng lặp hoặc các vòng lặp thường được thực hiện nhiều lần

~ Đối với các lệnh nhảy xuôi thì khó dự đoán hơn: Có thể là kết thúc lệnh loop hoặc là nhảy có điều kiện.

Chương 3.2: Bộ nhớ Cache

1. Sơ đồ phân cấp bộ nhớ trong hệ thống máy tính. Vai trò của việc phân cấp bộ nhớ của hệ thống máy tính.

- Sơ đồ phân cấp bộ nhớ trong hệ thống máy tính:



- Mô hình phân cấp bộ nhớ gồm các thành phần:
  - Thanh ghi của CPU:
    + Kích thước rất nhỏ (vài chục byte tới vài KB)
    + Tốc độ rất nhanh, thời gian truy cập khoảng 0.25 ns, giá thành đắt
    + Lưu trữ tạm thời dữ liệu đầu vào và ra cho các lệnh
  - Cache:
    + Kích thước nhỏ (64KB tới 32MB)
    + Tốc độ nhanh, thời gian truy cập khoảng 1 – 5ns, giá thành đắt
    + Lưu trữ lệnh và dữ liệu cho CPU
    + Còn được gọi là "bộ nhớ thông minh" (smart memory)
  - Bộ nhớ chính:

+ Gồm bộ nhớ ROM và bộ nhớ RAM, có kích thước lớn, dung lượng từ 256MB tới 4GB cho các hệ 32bits
+ Tốc độ chậm, thời gian truy cập từ 50 – 70ns
+ Lưu trữ lệnh và dữ liệu cho hệ thống và người dùng
+ Giá thành rẻ

- Bộ nhớ phụ:
+ Kích thước rất lớn, dung lượng từ 20GB tới 1000GB
+ Tốc độ rất chậm, thời gian truy cập khoảng 5ms
+ Lưu trữ lượng dữ liệu lớn dưới dạng các tệp (files) trong thời gian lâu dài
+ Giá thành rất rẻ

- Vai trò của việc phân cấp hệ thống bộ nhớ:
  - Nâng cao hiệu năng hệ thống:
    + Dung hòa được CPU có tốc độ cao với bộ nhớ chính và bộ nhớ phụ có tốc độ thấp
    + Thời gian truy cập dữ liệu trung bình của CPU từ hệ thống bộ nhớ gần bằng thời gian truy cập cache
  - Giảm giá thành sản xuất:
    + Các thành phần đắt tiền sẽ được sử dụng với dung lượng nhỏ hơn
    + Các thành phần rẻ hơn được sử dụng với dung lượng lớn hơn
    + Tổng giá thành của hệ thống nhớ theo mô hình phân cấp sẽ rẻ hơn so với hệ thống nhớ không phân cấp cùng tốc độ

2. Các phương pháp phân loại bộ nhớ máy tính (?)

- Dựa vào kiểu truy cập:
  - Bộ nhớ truy cập ngẫu nhiên (RAM)
  - Bộ nhớ truy cập tuần tự (SAM)
  - Bộ nhớ chỉ đọc (ROM)
- Dựa vào khả năng chịu đựng/ lưu trữ thông tin:
  - Bộ nhớ không ổn định: thông tin lưu trữ bị mất khi tắt nguồn
  - Bộ nhớ ổn định: thông tin lưu trữ được giữ lại khi tắt nguồn.
- Dựa vào công nghệ chế tạo:
  - Bộ nhớ bán dẫn: ROM, RAM, SSD, USB,...
  - Bộ nhớ từ: HDD, FDD, tape
  - Bộ nhớ quang: CD, DVD

3. Khái niệm, đặc điểm bộ nhớ ROM (?)

- Bộ nhớ ROM là bộ nhớ chỉ đọc. Muốn ghi thông tin vào ROM phải sử dụng các thiết bị hoặc phương pháp đặc biệt.
- Đặc điểm:
  - ROM là bộ nhớ ổn định: tất cả thông tin vẫn được duy trì khi mất nguồn nuôi
  - Là bộ nhớ bán dẫn: mỗi ô nhớ là một cổng bán dẫn
  - Thường dùng để lưu trữ thông tin hệ thống: thông tin phần cứng và BIOS

4. Khái niệm, đặc điểm bộ nhớ RAM (?)

- Bộ nhớ RAM là bộ nhớ truy cập ngẫu nhiên. Ô nhớ có thể được truy cập một cách ngẫu nhiên với tốc độ truy cập các ô nhớ là tương đương.
- Đặc điểm:
  - Là bộ nhớ không ổn định: mọi thông tin lưu trữ sẽ bị mất khi tắt nguồn
  - Là bộ nhớ bán dẫn: mỗi ô nhớ là một cổng bán dẫn
  - Thường dùng để lưu trữ thông tin hệ thống và của người dùng.

5. Khái niệm, vai trò của bộ nhớ Cache.

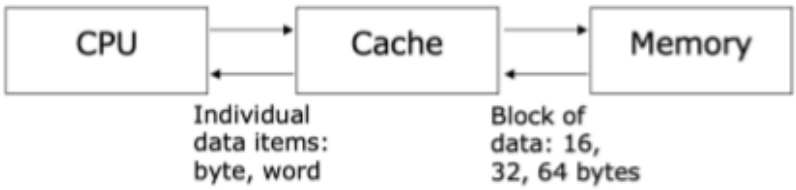
- Cache (hay còn gọi là bộ nhớ đệm) là thành phần nhớ trong sơ đồ phân cấp bộ nhớ máy tính. Nó hoạt động như thành phần trung gian, trung chuyển dữ liệu từ bộ nhớ chính về CPU và ngược lại.
- Vai trò của bộ nhớ Cache:
  - Nâng cao hiệu năng hệ thống:
    + Dung hòa giữa CPU có tốc độ cao và bộ nhớ chính tốc độ thấp (giảm số lượng truy cập trực tiếp của CPU vào bộ nhớ chính)
    + Thời gian trung bình CPU truy cập hệ thống bộ nhớ gần bằng thời gian truy cập cache.
  - Giảm giá thành sản xuất:
    + Nếu 2 hệ thống có cùng hiệu năng thì hệ thống có cache sẽ rẻ hơn
    + Nếu 2 hệ thống cùng giá thành, hệ thống có cache sẽ nhanh hơn

6. 2 nguyên lý hoạt động của bộ nhớ cache: nguyên lí lân cận về không gian và nguyên lí lân cận về thời gian.

- Cache được coi là bộ nhớ thông minh:
  - Có khả năng đoán trước yêu cầu về lệnh và dữ liệu của CPU.
  - Dữ liệu và lệnh cần thiết được chuyển trước từ bộ nhớ chính về cache –> CPU chỉ truy nhập cache –> giảm thời gian truy nhập bộ nhớ
- Cache hoạt động dựa trên 2 nguyên lý cơ bản:
  - Nguyên lý lân cận về không gian:
    + Nếu 1 vị trí bộ nhớ được truy cập, thì xác suất các vị trí gần đó được truy cập trong thời gian gần tới là cao
    + Áp dụng với dữ liệu và các lệnh có thứ tự tuần tự theo chương trình
    + Lệnh trong chương trình thường có thứ tự tuần tự, do đó cache đọc 1 khối lệnh trong bộ nhớ và bao gồm cả các phần tử xung quanh vị trí phần tử hiện tại được truy cập.
  - Nguyên lý lân cận về thời gian:
    + Nếu 1 vị trí bộ nhớ được truy cập, thì xác suất nó được truy cập lại trong thời gian gần là cao
    + Áp dụng với các mục dữ liệu và các lệnh trong vòng lặp
    + Cache đọc khối dữ liệu trong bộ nhớ gồm tất cả các thành phần trong vòng lặp

7. Phương thức trao đổi giữa CPU và bộ nhớ Cache, giữa bộ nhớ Cache và bộ nhớ chính.

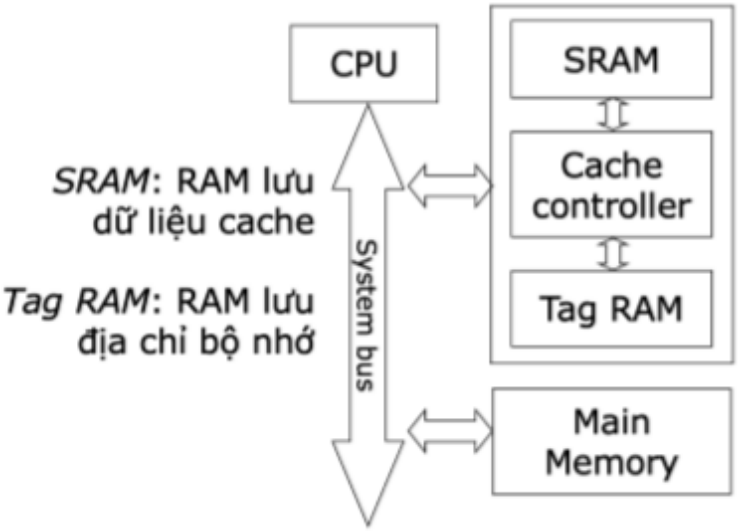
Sơ đồ trao đổi dữ liệu giữa CPU, Cache và bộ nhớ chính:



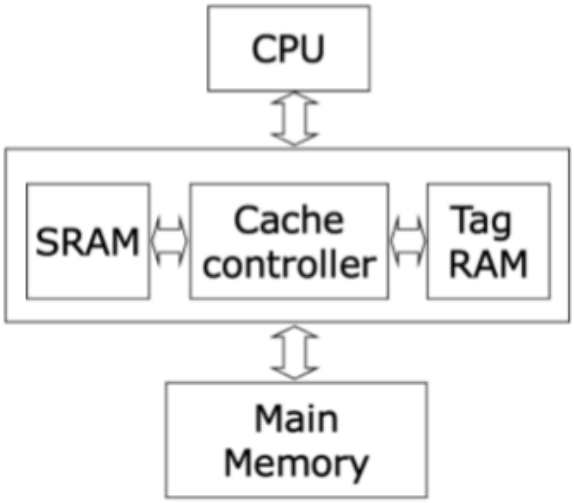
CPU trao đổi dữ liệu (đọc/ ghi) với cache theo các đơn vị cơ sở như byte, word và double word còn cache trao đổi dữ liệu với bộ nhớ chính theo các khối, với kích thước 16, 32 hoặc 64bytes.

8. Đặc điểm của 2 kiến trúc bộ nhớ Cache: look aside và look through.

- Kiến trúc look aside (kiến trúc kiểu ngang hàng – bình đẳng) (kiến trúc này được dùng phổ biến hiện nay hơn):
  - Cache và bộ nhớ cùng được kết nối tới bus hệ thống
  - Cache và bộ nhớ chính "thấy" chu kỳ bus CPU tại cùng 1 thời điểm
  - Ưu điểm:
    + Thiết kế đơn giản
    + Miss nhanh: khi CPU không tìm thấy mục dữ liệu trong cache nó đồng thời tìm trong bộ nhớ chính cùng 1 chu kỳ xung nhịp
  - Nhược điểm:
    + Hit chậm: do cache kết nối với CPU sử dụng bus hệ thống có tần số làm việc không cao và băng thông hẹp



- Kiến trúc look through (kiến trúc kiểu phân chia – phân cấp):
  - Cache nằm giữa CPU và bộ nhớ chính
  - Cache "thấy" chu kỳ bus CPU trước sau đó nó "truyền" lại cho bộ nhớ chính.
  - Ưu điểm:
    + Hit nhanh: Cache kết nối với CPU bằng bus riêng tốc độ cao và có băng thông lớn.
  - Nhược điểm:
    + Đắt và thiết kế phức tạp
    + Miss chậm: khi CPU không tìm thấy dữ liệu trong cache nó cần tìm trong bộ nhớ tại 1 xung nhịp tiếp theo



9. Tổ chức bộ nhớ Cache (?)

- Ánh xạ trực tiếp:
  - Cache được chia thành n khối hoặc dòng đánh số từ 0 đến n - 1
  - Bộ nhớ được chia thành m trang đánh số từ 0 đến m - 1, mỗi trang bộ nhớ có n dòng đánh số từ 0 đến n - 1 và có kích thước bằng cache.
  - Dòng thứ i của 1 trang bất kì trong bộ nhớ ánh xạ tới dòng thứ i của cache.
  - Ưu điểm:
    + Thiết kế đơn giản
    + Vì ánh xạ cố định: khi biết địa chỉ bộ nhớ có thể tìm thấy nó trong cache rất nhanh.
  - Nhược điểm:
    + Vì ánh xạ cố định nên khả năng xảy ra xung đột cao.
    + Tỷ lệ hit thấp
- Ánh xạ kết hợp đầy đủ:
  - Cache được chia thành n khối hoặc dòng đánh số từ 0 đến n - 1
  - Bộ nhớ được chia thành m dòng đánh số từ 0 đến m - 1, kích thước mỗi dòng cache bằng kích thước 1 dòng bộ nhớ; số lượng dòng trong bộ nhớ có thể lớn hơn nhiều số lượng dòng của cache (m >= n)
  - Dòng thứ i của bộ nhớ ánh xạ tới bất kì dòng thứ j nào của cache.
  - Ưu điểm:
    + Ít xung đột vì ánh xạ linh hoạt
    + Tỉ lệ hit cao hơn
  - Nhược điểm:
    + Chậm vì phải tìm kiếm địa chỉ bộ nhớ trong cache
    + Phức tạp vì có thêm n bộ so sánh địa chỉ trong cache
    + Thường chỉ sử dụng cho cache có kích thước nhỏ
- Ánh xạ tập kết hợp:
  - Cache được chia thành k đường có kích thước bằng nhau. Mỗi đường được chia thành n dòng đánh số từ 0 đến n – 1.
  - Bộ nhớ được chia thành m trang đánh số từ 0 đến m - 1, kích thước 1 trang bằng kích thước 1 way của cache. Mỗi trang được chia thành n dòng đánh số từ 0 đến n – 1.
  - 1 trang bộ nhớ có thể ánh xạ tới way bất kì của cache. Dòng thứ i của trang j được ánh xạ tới dòng thứ i của way k.
  - Ưu điểm:
    + Nhanh do ánh xạ trực tiếp được sử dụng cho ánh xạ dòng.
    + Ít xung đột do ánh xạ từ các trang bộ nhớ đến các đường cache là không cố định.
    + Tỷ lệ hit cao
  - Nhược điểm:
    + Thiết kế phức tạp.
    + Điều khiển phức tạp.

10. Các chính sách thay thế dòng Cache: thay thế ngẫu nhiên, FIFO và LRU

- Thay thế ngẫu nhiên:
  - Các dòng trong cache được chọn ngẫu nhiên để thay.
  - Thiết kế đơn giản, dễ cài đặt.
  - Hệ số miss cao vì phương pháp này không xét tới dòng cache nào đang thực sự được sử dụng. Nếu 1 dòng cache đang được sử dụng mà bị thay thế thì sự kiện miss xảy ra và cần phải đọc lại vào cache
- Vào trước ra trước (FIFO):
  - Dựa trên nguyên lý FIFO: dòng cache được đọc vào cache trước sẽ được chọn để thay trước
  - Hệ số miss thấp hơn so với phương pháp thay thế ngẫu nhiên, nhưng vẫn còn cao vì phương pháp này vẫn chưa thực sự xem xét tới dòng cache nào đang thực sự được sử dụng. Một dòng cache "cũ" có thể vẫn đang được sử dụng.
  - Cài đặt phức tạp vì cần thêm mạch để giám sát thứ tự nạp các dòng bộ nhớ vào cache.
- Dòng ít được sử dụng gần đây nhất (LRU)
  - Các dòng cache ít được sử dụng nhất trong thời gian gần đây sẽ được chọn để thay.
  - Hệ số miss thấp nhất so với phương pháp ngẫu nhiên và FIFO do có xét tới các dòng cache đang được sử dụng.
  - Cài đặt phức tạp vì cần thêm mạch để giám sát tần suất sử dụng các dòng cache.

11. Thao tác đọc/ ghi dữ liệu (khi hit/miss)

- Thao tác đọc:
  - Trường hợp tìm thấy (hit case): mục dữ liệu yêu cầu tìm thấy trong cache
    + Mục dữ liệu được đọc từ cache vào CPU
    + Bộ nhớ chính không tham gia
  - Trường hợp không tìm thấy (miss case):
    + Mục dữ liệu được đọc từ bộ nhớ vào cache
    + Sau đó dữ liệu được chuyển từ cache vào CPU
    ⇨ Miss penalty: thời gian truy cập mục dữ liệu bằng tổng thời gian truy cập cache và bộ nhớ chính
- Thao tác ghi:
  - Trường hợp tìm thấy (hit case): bản tin thuộc 1 khối trong cache
    + Write through (ghi thẳng): mục dữ liệu được ghi vào cache và ghi vào bộ nhớ đồng thời
    + Write back (ghi trễ): mục dữ liệu trước tiên được ghi vào cache và cả dòng (block) chứa nó ở trong cache sẽ được ghi lại vào bộ nhớ sau đó, khi mà dòng đó bị thay thế
  - Trường hợp không tìm thấy (miss case): bản tin không có trong cache
    + Write allocate (ghi có đọc lại): mục dữ liệu trước hết được ghi vào bộ nhớ sau đó cả dòng chứa nó sẽ được đọc vào cache

+ Write non-allocate (ghi không đọc lại): mục dữ liệu chỉ được ghi vào bộ nhớ

12. Các yếu tố ảnh hưởng đến hiệu năng cache: kích thước cache, cache nhiều mức và sự phân chia cache.

- Yếu tố về kích thước cache:
  - Kích thước cache lớn:
    + Có thể lưu trữ nhiều khối dữ liệu trong bộ nhớ hơn
    + Giảm tần suất trao đổi các khối dữ liệu của chương trình khác nhau giữa bộ nhớ và cache
    + Cache lớn chậm hơn cache nhỏ do phải tìm kiếm vị trí bộ nhớ trong không gian lớn hơn
  - Xu hướng hiện nay: cache càng ngày càng lớn
    + Hỗ trợ đa nhiệm (multi-tasking) tốt hơn
    + Hỗ trợ tốt hơn cho xử lý song song
    + Hỗ trợ tốt hơn cho các hệ thống CPU đa nhân
- Yếu tố cache nhiều mức:
  - Cache nhiều mức có thể dung hòa tốc độ của CPU và MEM tốt hơn cache 1 mức.
  - Thực tế, cache thường có 2 mức: L1 và L2. Một số cache có 3 mức: L1, L2 và L3.
  - Giảm giá thành hệ thống nhớ.
- Yếu tố phân chia cache:
  - Cache có thể được chia thành cache lệnh và cache dữ liệu để hiệu năng tốt hơn vì:
    + Dữ liệu và lệnh khác nhau về tính cục bộ: Dữ liệu có tính chất cục bộ về thời gian hơn còn lệnh có tính chất cục bộ về không gian hơn
    + Cache lệnh chỉ hỗ trợ thao tác đọc còn cache dữ liệu hỗ trợ cả 2 thao tác đọc ghi nên dễ tối ưu cache hơn
    + Hỗ trợ nhiều thao tác đọc/ ghi cùng lúc nên giảm xung đột tài nguyên
    + Kết hợp các chức năng khác như giải mã trước lệnh vào trong cache lệnh nên xử lý lệnh tốt hơn
  - Trong thực tế, hầu hết cache L1 được chia thành 2 phần:
    + I_cache (Instruction cache): cache lệnh
    + D_cache (Data cache): cache dữ liệu.
  - Các cache mức cao hơn không được chia:
    + Chia cache L1 có lợi ích cao nhất vì nó gần CPU nhất. CPU đọc/ ghi trực tiếp lên cache L1. Cache L1 cần hỗ trợ nhiều lệnh truy nhập đồng thời và các biện pháp tối ưu hóa
    + Chia các cache mức cao hơn không đem lại hiệu quả cao và làm phức tạp trong hệ thống điều khiển cache

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

1. Các thanh ghi đa năng

- 4 thanh ghi 16bits (hay 8 thanh ghi 8bits: AH, AL, BH, BL, CH, CL, DH, DL):
  - AX (accumulator): Thanh ghi tổng, thường dùng để lưu kết quả
  - BX (base): Thanh ghi cơ sở, thường dùng chứa địa chỉ ô nhớ
  - CX (count): Thanh ghi đếm, thường dùng làm biến đếm cho các lệnh lặp (Loop), chứa số lần dịch hoặc quay trong các lệnh dịch và quay thanh ghi
  - DX (data): Thanh ghi dữ liệu, cùng AX chứa dữ liệu trong các phép tính nhân/chia số 16 bit hoặc chứa địa chỉ cổng trong các lệnh vào ra dữ liệu trực tiếp (IN/OUT)
- Các thanh ghi con trỏ và chỉ số:
  - SP (Stack Pointer): con trỏ ngăn xếp, luôn chứa địa chỉ đỉnh ngăn xếp SS:SP
  - BP (Base Pointer): Con trỏ cơ sở, chứa địa chỉ của dữ liệu trong đoạn ngăn xếp SS hoặc các đoạn khác SS:BP
  - SI (Source Index): Thanh ghi chỉ số nguồn. SI thường dùng chứa địa chỉ ô nhớ nguồn ở đoạn dữ liệu DS như trong các lệnh chuỗi trong các thao tác chuyển dữ liệu DS:SI
  - DI (Destination Index): Thanh ghi chỉ số đích. DI thường dùng chứa địa chỉ ô nhớ đích ở đoạn dữ liệu DS trong các thao tác chuyển dữ liệu DS:DI
  - SI và DI có thể được sử dụng như thanh ghi đa năng
- Các thanh ghi đoạn:
  - CS (Code Segment): Thanh ghi đoạn mã. CS chứa địa chỉ bắt đầu đoạn mã
  - DS (Data Segment): Thanh ghi đoạn dữ liệu. DS chứa địa chỉ bắt đầu đoạn dữ liệu
  - SS (Stack Segment): Thanh ghi đoạn ngăn xếp. SS chứa địa chỉ bắt đầu đoạn ngăn xếp
  - ES (Extra Segment): Thanh ghi đoạn dữ liệu mở rộng. ES chứa địa chỉ bắt đầu đoạn dữ liệu mở rộng.
- Con trỏ lệnh và thanh ghi cờ
  - IP (Instruction Pointer): Con trỏ lệnh (còn gọi là bộ đếm chương trình PC). IP luôn chứa địa chỉ offset của lệnh tiếp theo sẽ được thực hiện. CS:IP chứa địa chỉ logic của lệnh tiếp theo đó.
  - FR (Flag Register) hoặc SR (Status Register): Thanh ghi cờ hoặc thanh ghi trạng thái.

    + Cờ trạng thái: Các bit của FR lưu các trạng thái của kết quả phép toán ALU thực hiện
      - ZF (Zero): cờ zero. ZF = 1 khi kết quả bằng 0; ngược lại ZF = 0
      - SF (Sign): cờ dấu. SF = 1 khi kết quả âm.
      - CF (Carry): cờ nhớ. CF = 1 là phép tính có nhớ, =0 nếu không nhớ.

    Nếu phép cộng có nhớ ra khỏi bit cao nhất hay phép toán trừ có mượn ra khỏi bit cao nhất thì CF được thiết lập (báo tràn với số nguyên không dấu).

• AF (Auxiliary): cờ nhớ phụ. AF = 1 là có nhớ phụ; AF = 0 là không nhớ phụ. Nếu phép cộng có nhớ từ bit 3 sang bit 4 hoặc phép trừ có mượn từ bit 3 sang bit 4 thì cờ AF được thiết lập.

• PF (Parity): cờ chẵn lẻ. PF = 1 khi tổng số bit 1 trong kết quả là lẻ và PF = 0 khi tổng số bit 1 trong kết quả là chẵn

• OF (Overflow): cờ tràn. OF = 1 khi kết quả bị tràn số. Nếu cộng 2 số cùng dấu mà kết quả có dấu ngược lại thì OF được thiết lập (báo tràn với số nguyên có dấu).

+ Cờ điều khiển: Các bit của FR lưu các trạng thái của tín hiệu điều khiển.

• DF (Direction): cờ hướng, chỉ hướng tăng giảm địa chỉ với các lệnh chuyển dữ liệu. DF = 0 là địa chỉ tăng, DF = 1 là địa chỉ giảm.

• TF (Trap/Trace): cờ bẫy/lần vết, được dùng khi gỡ rối chương trình. TF=1 là CPU ở chế độ chạy từng lệnh (chế độ gỡ rối chương trình).

• IF (Interrupt): cờ ngắt. Nếu IF = 1 thì bộ vi xử lý cho phép ngắt với yêu cầu ngắt đưa đến chân tín hiệu INTR (Interrupt Request) của bộ vi xử lý. Nếu IF = 0 thì cấm ngắt.

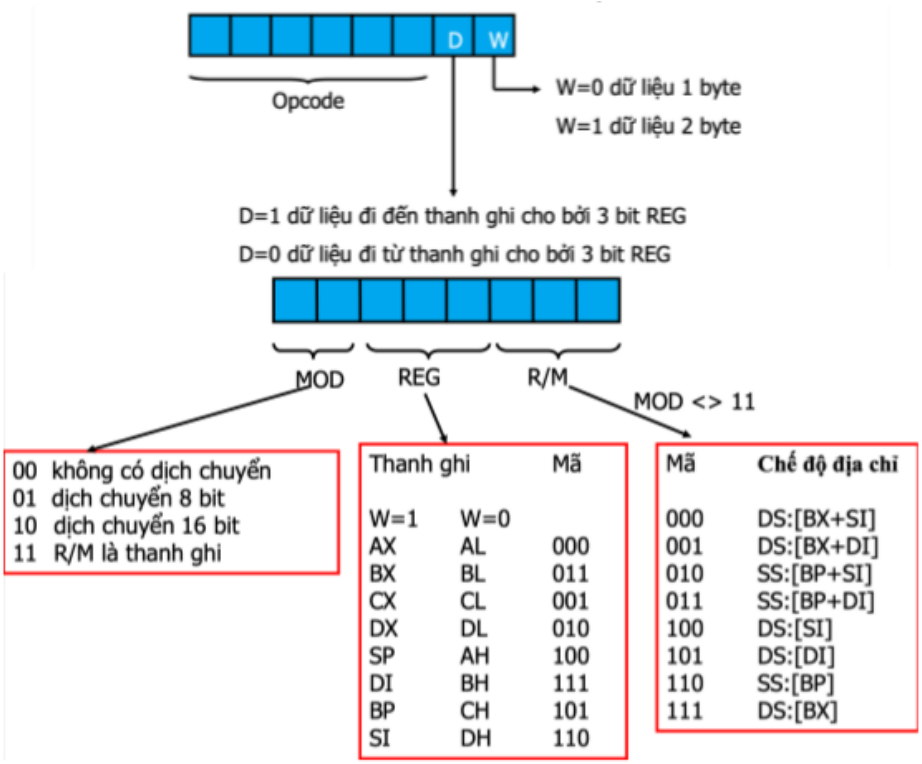
- Hàng đợi lệnh IQ (?)
  - Chứa lệnh đọc từ bộ nhớ cho EU thực hiện.
  - Trong 8088, IQ có 4 bytes, còn trong 8086, IQ có 6 bytes.
  - IQ là một thành phần quan trọng của cơ chế ống lệnh giúp tăng tốc độ xử lý lệnh.

2. Khuôn dạng lệnh tổng quát và các thành phần của nó (gồm 6 thành phần: OPCODE, D, W, MOD, REG, R/M)

- Dạng tổng quát của lệnh gồm 2 thành phần: mã lệnh và địa chỉ của các toán hạng

Opcode	Operands
Mã lệnh	Các toán hạng

- Độ dài của từ lệnh: 8, 16, 24, 32 và 64 bit.
- Lệnh của 8086/8088 có thể có độ dài 1-6 byte.



VD: MOV AX, BX => 1000 1011 1100 0011: 8BC3H

1	0	0	0	1	0	1	1	1	1	0	0	0	0	1	1
Opcode (MOV)						D (dữ liệu đi vào AX)	W (toán hạng 2bytes)	MOD (R/M là thanh ghi)		REG (AX)			R/M (BX)		

3. Các nhóm lệnh hợp ngữ của vi xử lý 8086/8088

- Các lệnh trao đổi dữ liệu:
  - Lệnh MOV: Chuyển (sao chép) dữ liệu từ gốc sang đích
    Dạng: MOV đích, gốc (VD: MOV AL, BL; AL <- BL)
  - Lệnh LEA: Nạp địa chỉ hiệu dụng vào thanh ghi
    Dạng: LEA đích, gốc (VD: LEA CX, [BX]; CX <- [DS:BX])
  - Lệnh IN: Đọc dữ liệu từ <địa chỉ cổng vào> lưu vào <thanh ghi>
    Dạng: IN <thanh ghi>, <địa chỉ cổng vào> (VD: IN AL, DX; AL <- (DX))
  - Lệnh OUT: Lưu dữ liệu từ gốc ra <địa chỉ cổng ra>
    Dạng: OUT <địa chỉ cổng ra>, <gốc> (VD: OUT 0F8H, AL; (0F8h) <- AL)
  - Lệnh PUSH/ POP: đẩy/ lấy dữ liệu vào/ ra ngăn xếp
- Các lệnh số học:
  - Các lệnh tính toán:
    + Lệnh ADD: Cộng 2 toán hạng (VD: ADD BX, AX; BX <- BX + AX)
    + Lệnh SUB: Trừ 2 toán hạng (VD: SUB AX, BX; AX <- AX – BX)
    + Lệnh MUL: Nhân 2 số không dấu (VD: MUL BL; AX <- AL * BL)
    + Lệnh DIV: Chia 2 số không dấu (VD: DIV BL; AL = AX / BL
      AH <- AX % BL)
    + Lệnh INC/ DEC: Tăng/ giảm 1 đơn vị của toán hạng
  - Các lệnh logic:

+ Lệnh NOT: Đảo các bit toán hạng (VD: MOV AL, 80H; 80H=10000000B
NOT AL; 7FH=01111111B)
+ Lệnh AND: Nhân các cặp bit của 2 toán hạng
+ Lệnh OR: Cộng các cặp bit của 2 toán hạng
+ Lệnh XOR: Cộng đảo các cặp bit của 2 toán hạng
+ Lệnh CMP: So sánh 2 toán hạng

- Các lệnh dịch và quay:
+ Lệnh SHL/ SHR: Dịch trái/ phải 1 bit (VD: SHL/ SHR AL, 1)
+ Lệnh ROL/ ROR: Quay trái/ phải 1 bit (VD: ROL/ ROR AL, 1)

- Các lệnh điều khiển, rẽ nhánh và lặp:
  - JMP: Nhảy đến một nhãn nào đó
  - LOOP: lặp lại đoạn chương trình do nhãn chỉ ra cho đến khi CX=0
  - (Các lệnh nhảy có điều kiện khác: JE, JZ, JNE, JL, JA, JB, …. tự viết)

4. Chế độ địa chỉ của vi xử lý là gì? 7 chế độ địa chỉ của vi xử lý 8086/8088.

- Chế độ địa chỉ là cách CPU tổ chức và lấy dữ liệu cho các toán hạng khi thực hiện lệnh.
- Chế độ địa chỉ thanh ghi:
  - Chế độ thanh ghi sử dụng các thanh ghi bên trong CPU như là các toán hạng để chứa dữ liệu cần thao tác.
  - Cả toán hạng gốc và đích đều là các thanh ghi.
  - VD: MOV BX, DX; BX <- DX
    ADD AL, DL; AL <- AL + DL
    MOV DS, AX; DS <- AX
    MOV ES, DS; không hợp lệ (segment to segment)
    MOV CS, AX; không hợp lệ vì CS không dùng làm thanh ghi đích
- Chế độ địa chỉ tức thì:
  - Toán hạng đích là 1 thanh ghi hay 1 ô nhớ, toán hạng gốc là một hằng số
  - Dùng để nạp hằng số vào thanh thi (trừ thanh ghi đoạn và thanh cờ) hoặc vào ô nhớ trong đoạn dữ liệu DS.
  - VD: MOV CL, 200; CL <- 200
    MOV AX, 0ff0h; AX <- 0ff0h
    MOV [BX], 200; Chuyển 200 vào ô nhớ có địa chỉ là DS:BX
- Chế độ địa chỉ trực tiếp:
  - 1 toán hạng là 1 hằng số biểu diễn địa chỉ lệch của ô nhớ
  - Toán hạng còn lại có thể là thanh ghi (không được là ô nhớ).
  - VD: MOV AL, [8088H]; AL <- DS:[8088H]
    MOV [1234H], DL; DS:[1234H] <- DL
- Chế độ địa chỉ gián tiếp thanh ghi:
  - 1 toán hạng là 1 thanh ghi chứa địa chỉ lệch của ô nhớ
  - Toán hạng còn lại có thể là thanh ghi
  - VD: MOV AL, [BX]; AL <- [DS:BX]
    MOV AL, [BP]; AL <- [SS:BP]
- Chế độ địa chỉ tương đối cơ sở:

- 1 toán hạng là địa chỉ của ô nhớ.
  + Địa chỉ của ô nhớ được tạo bởi thanh ghi cơ sở như BX (đoạn DS) hoặc BP (đoạn SS) và 1 hằng số
  + Hằng số là giá trị dịch chuyển để tính địa chỉ hiệu dụng của các toán hạng trong vùng nhớ DS và SS.
- Toán hạng còn lại có thể là thanh ghi (không được là ô nhớ)
- VD: MOV AL, [BX+100]; AL <- [DS:BX+100]
  MOV AL, [BP]+200; AL <- [SS:BP+200]

- Chế độ địa chỉ tương đối chỉ số:
  - 1 toán hạng là địa chỉ của ô nhớ:
    + Địa chỉ của ô nhớ tạo bởi thanh ghi cơ sở SI hoặc DI và 1 hằng số.
    + Hằng số biểu diễn các giá trị dịch chuyển được dùng để tính địa chỉ hiệu dụng của các toán hạng trong các vùng nhớ DS.
  - Toán hạng còn lại có thể là thanh ghi (không được là ô nhớ)
  - VD: MOV AL, [SI+100]; AL <- [DS:SI+100]
    MOV AX, [SI+10]; Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:SI+10 và DS:SI+11 vào AX
- Chế độ địa chỉ tương đối chỉ số cơ sở:
  - 1 toán hạng là địa chỉ của ô nhớ:
    + Địa chỉ của ô nhớ tạo bởi các thanh ghi BX+SI/DI (đoạn DS) hoặc BP+SI/DI (đoạn SS) và 1 hằng số.
    + Hằng số biểu diễn các giá trị dịch chuyển để tính địa chỉ hiệu dụng của các toán hạng trong các vùng nhớ DS và SS.
  - Toán hạng còn lại có thể là thanh ghi (không được là ô nhớ)
  - VD: MOV AL, [BX+SI+100]; AL <- [DS:BX+SI+100]
    MOV AX, [BX+SI+8]; Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:BX+SI+8 và DS:BX+SI+9 vào AX

Chương 5: Phối ghép và lập trình điều khiển thiết bị

1. Vai trò và phương pháp vào ra dữ liệu là gì?

- Vai trò của vào ra dữ liệu:
  - Là phương tiện giúp CPU giao tiếp với thế giới bên ngoài.
  - Cung cấp dữ liệu đầu vào cho CPU xử lý.
  - Cung cấp phương tiện để CPU kết xuất dữ liệu đầu ra.
- Các phương pháp vào ra chính:
  - Thăm dò.
  - Ngắt.
  - Truy nhập trực tiếp bộ nhớ.

2. Trình bày vào ra bằng thăm dò, vào ra bằng ngắt, vào ra bằng truy cập trực tiếp đến bộ nhớ. So sánh các phương pháp với nhau và cho các ví dụ minh hoạ.

- Vào ra bằng thăm dò:

- CPU kiểm tra lần lượt các thiết bị để phát hiện trạng thái sẵn sàng trao đổi dữ liệu của từng thiết bị và thực hiện các lệnh trao đổi dữ liệu. Các thiết bị được kiểm tra thăm dò theo trật tự ngẫu nhiên hoặc theo mức độ ưu tiên.
- CPU là bên chủ động trong quá trình trao đổi dữ liệu.
- Ưu điểm:
  + Đơn giản, dễ cài đặt
  + Có thể cài đặt được bằng phần mềm
- Nhược điểm:
  + Hiệu quả thấp do CPU tốn nhiều thời gian để thăm dò các thiết bị.
  + Không thực sự khả thi khi có nhiều thiết bị trong danh sách thăm dò.

- Vào ra bằng ngắt:
  - Khi thiết bị cần trao đổi dữ liệu, thiết bị sinh ra tín hiệu ngắt tới CPU. CPU sẽ hoàn thành câu lệnh hiện tại và lưu nội dung của bộ đếm chương trình và các thanh ghi trạng thái. Sau đó, máy tính tự động nạp địa chỉ của chương trình phục vụ ngắt vào thanh ghi đếm chương trình rồi chuyển sang xử lý ngắt. Sau khi hoàn thành xử lý ngắt, CPU tiếp tục thực hiện lệnh tiếp theo của chương trình chính.
  - Thiết bị vào ra là bên chủ động trong quá trình trao đổi dữ liệu.
  - Ưu điểm:
    + Hiệu quả hơn vào ra bằng thăm dò do CPU không cần thăm dò từng thiết bị.
  - Nhược điểm:
    + Phức tạp hơn vào ra bằng thăm dò.
    + Cần mạch phần cứng để điều khiển ngắt.
- Vào ra bằng cách truy cập trực tiếp đến bộ nhớ:
  - Khi thiết bị cần trao đổi dữ liệu, thiết bị sinh ra yêu cầu tới CPU thông qua DMAC. CPU nhận được yêu cầu sẽ gửi các tham số điều khiển trao đổi dữ liệu và tín hiệu xác nhận yêu cầu sử dụng bus cho DMAC và tự tách ra khỏi bus hệ thống. DMAC chiếm quyền điều khiển bus hệ thống và gửi tín hiệu xác nhận trao đổi dữ liệu cho thiết bị vào ra, điều khiển quá trình trao đổi dữ liệu trực tiếp giữa thiết bị vào ra và bộ nhớ. Sau khi kết thúc quá trình DMA, DMAC trả quyền điều khiển bus cho CPU.
  - Thiết bị vào ra là bên chủ động trong quá trình trao đổi dữ liệu.
  - Ưu điểm:
    + Hiệu suất rất cao do dữ liệu được trao đổi trực tiếp theo khối giữa thiết bị vào ra và bộ nhớ không cần thông qua CPU.
  - Nhược điểm:
    + Phức tạp hơn vào ra bằng thăm dò và ngắt.
    + Cần mạch phần cứng để điều khiển quá trình DMA.

Chương ?: WTF? (?)

1. RAID là gì? Trình bày các kỹ thuật cơ bản tạo RAID

- RAID là một công nghệ tạo các thiết bị lưu trữ tiên tiến trên cơ sở các ổ cứng nhằm đạt các yêu cầu về tốc độ cao, tính tin cậy cao và dung lượng lớn. Mặc dù

RAID là một mảng của các ổ cứng, nhưng không phải tất cả các loại ổ cứng đều có thể sử dụng để tạo RAID. Thực tế chỉ có các ổ cứng theo chuẩn SATA, SCSI và tương đường mới hỗ trợ tạo RAID.

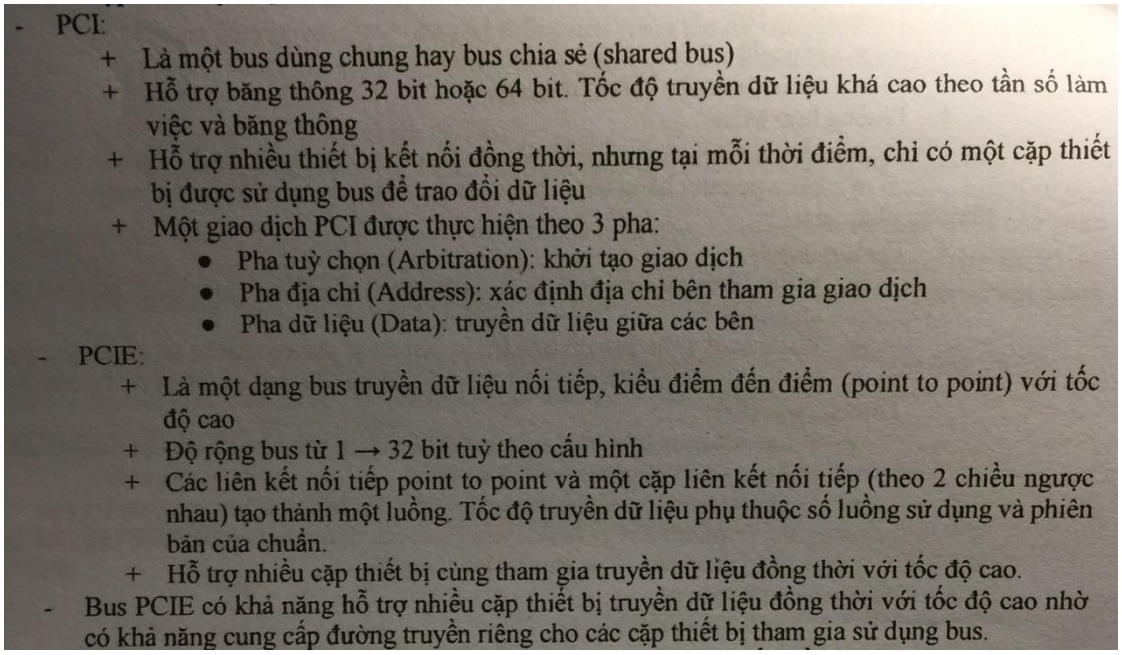
- 2 kỹ thuật cơ bản tạo RAID:
  - Kỹ thuật tạo lát đĩa: Các dữ liệu cần ghi được chia thành các khối cùng kích thước và được ghi đồng thời vào các ổ đĩa độc lập. Tương tự, trong quá trình sộc, các khối dữ liệu cần đọc được đọc đồng thời từ các đĩa cứng độc lập, giúp giảm thời gian đọc.
  - Kỹ thuật soi gương đĩa: Dữ liệu địa chia thành các khối và mỗi khối được ghi đồng thời lên hai hay nhiều ổ đĩa độc lập. Tại mọi thời điểm ta đều có nhiều bản sao dữ liệu trên các đĩa cứng độc lập, đảm bảo tính an toàn cao.

2. Trình bày các loại RAID cơ bản: RAID0, RAID1, RAID 10, RAID 5 và RAID 6

- RAID 0:
  - Là phương pháp tạo ra một mảng RAID bằng cách chia dữ liệu thành các đoạn nhỏ và lưu trữ chúng trên các ổ đĩa cứng khác nhau.
  - Đặc điểm: RAID 0 tăng hiệu suất đọc/ghi dữ liệu vì nó có thể đọc và ghi song song trên nhiều ổ đĩa. Tuy nhiên, nếu một ổ đĩa bị hỏng, toàn bộ dữ liệu trong mảng RAID có thể bị mất.
- RAID 1:
  - RAID 1 sử dụng hai ổ đĩa hoặc nhiều hơn để tạo ra một bản sao chính xác của dữ liệu trên mỗi ổ đĩa.
  - Đặc điểm: RAID 1 cung cấp độ an toàn cao vì dữ liệu được sao lưu. Nếu một ổ đĩa bị hỏng, dữ liệu vẫn được giữ nguyên trên ổ đĩa khác. Tuy nhiên, hiệu suất ghi dữ liệu có thể bị giảm.
- RAID 10:
  - RAID 10 là sự kết hợp của RAID 1 và RAID 0. Nó yêu cầu ít nhất bốn ổ đĩa.
  - Đặc điểm: Dữ liệu được chia thành các khối nhỏ và sao chép trên các ổ đĩa theo cách tương tự như RAID 1. Sau đó, các khối dữ liệu được striping trên các ổ đĩa đó như trong RAID 0. RAID 10 cung cấp độ an toàn cao và hiệu suất đọc/ghi dữ liệu tốt.
- RAID 5:
  - RAID 5 sử dụng ít nhất ba ổ đĩa và sử dụng phương pháp striping cùng với việc phân bố các khối dữ liệu và các mã sửa chữa (parity) trên các ổ đĩa.
  - Đặc điểm: RAID 5 cung cấp độ an toàn dữ liệu cao và hiệu suất đọc dữ liệu tốt. Nếu một ổ đĩa bị hỏng, dữ liệu có thể được khôi phục từ các thông tin parity trên các ổ đĩa khác. Tuy nhiên, hiệu suất ghi dữ liệu có thể bị ảnh hưởng trong trường hợp này.
- RAID 6:
  - RAID 6 tương tự như RAID 5 với việc sử dụng các thông tin parity. Tuy nhiên, RAID 6 sử dụng ít nhất bốn ổ đĩa và lưu trữ hai mã sửa chữa (parity) trên các ổ đĩa.

- Đặc điểm: RAID 6 cung cấp mức độ an toàn dữ liệu cao hơn so với RAID 5. Nó cho phép chịu đựng đồng thời hai ổ đĩa bị hỏng mà không mất dữ liệu. Hiệu suất đọc dữ liệu tương đối tốt, nhưng hiệu suất ghi dữ liệu có thể bị ảnh hưởng.

3. Nêu các đặc điểm chính của kiến trúc bus PCI và PCI-Express. Tại sao bus PCI-Express có khả năng hỗ trợ nhiều cặp thiết bị truyền dữ liệu đồng thời với tốc độ cao?



- PCI:
  - + Là một bus dùng chung hay bus chia sẻ (shared bus)
  - + Hỗ trợ băng thông 32 bit hoặc 64 bit. Tốc độ truyền dữ liệu khá cao theo tần số làm việc và băng thông
  - + Hỗ trợ nhiều thiết bị kết nối đồng thời, nhưng tại mỗi thời điểm, chỉ có một cặp thiết bị được sử dụng bus để trao đổi dữ liệu
  - + Một giao dịch PCI được thực hiện theo 3 pha:
    - Pha tuỳ chọn (Arbitration): khởi tạo giao dịch
    - Pha địa chỉ (Address): xác định địa chỉ bên tham gia giao dịch
    - Pha dữ liệu (Data): truyền dữ liệu giữa các bên
- PCIE:
  - + Là một dạng bus truyền dữ liệu nối tiếp, kiểu điểm đến điểm (point to point) với tốc độ cao
  - + Độ rộng bus từ 1 → 32 bit tuỳ theo cấu hình
  - + Các liên kết nối tiếp point to point và một cặp liên kết nối tiếp (theo 2 chiều ngược nhau) tạo thành một luồng. Tốc độ truyền dữ liệu phụ thuộc số luồng sử dụng và phiên bản của chuẩn.
  - + Hỗ trợ nhiều cặp thiết bị cùng tham gia truyền dữ liệu đồng thời với tốc độ cao.
- Bus PCIE có khả năng hỗ trợ nhiều cặp thiết bị truyền dữ liệu đồng thời với tốc độ cao nhờ có khả năng cung cấp đường truyền riêng cho các cặp thiết bị tham gia sử dụng bus.