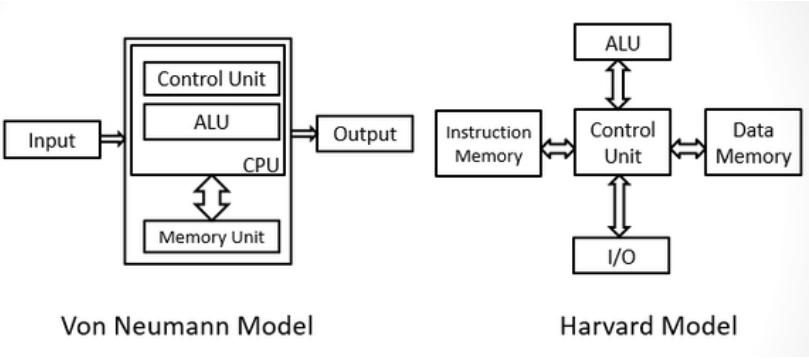
**Chương 1: Tổng quan về kiến trúc máy tính**

* 1. Giới thiệu về kiến trúc và tổ chức máy tính
* Kiến trúc máy tính(computer architecture):
  + 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: nhiều chức năng
    - Giá thành: càng rẻ càng tốt
  + Ba thành phần cơ bản của kiến trúc máy tính:
    - Kiến trúc tập lệnh(Instruction Set Architecture - ISA): là hình ảnh trừu tượng của máy tính ở mức ngôn ngữ máy (hoặc hợp ngữ), bao gồm:
      * Tập lệnh
      * Các chế độ địa chỉ bộ nhớ
      * Các thanh ghi
      * Khuôn dạng địa chỉ và dữ liệu
    - Vi kiến trúc(Micro Architecture): còn được gọi 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(System Design): bao gồm tất cả các thành phần phần cứng khác trong hệ thống máy tính.VD:
      * Các hệ thống kết nối như bus và chuyển mạch
      * Mạch điều khiển bộ nhớ và cấu trúc phân cấp bộ nhớ
      * Các kỹ thuật giảm tải cho CPU (như truy cập trực tiếp bộ nhớ)
      * Các vấn đề khác (như đa xử lý và xử lý song song)
* Tổ chức máy tính(computer organization): là khoa học nghiên cứu các thành phần của máy tính và phương thức làm việc của chúng dựa trên kiến trúc cho trước.
  1. Lịch sử phát triển của máy tính điện tử
* Thế hệ 1(1944-1959):
  + Sử dụng bóng đèn điện tử
  + Dùng băng từ làm các thiết bị đầu vào/ra
  + Mật độ tích hợp linh kiện: 1000 linh kiện/foot3(1 foot = 30.48cm)
  + Ví dụ: ENIAC – Electronic Numberical Integrator and Computer, 1946
* Thế hệ 2(1960-1964):
  + Sử dụng transistors
  + ~100,000 linh kiện/foot3
  + Ví dụ: UNIVAC 1107, UNIVAC III, IBM 7070, 7080, 7090, 1400 series, 1600 series
* Thế kệ 3(1964-1975):
  + Sử dụng mạch tích hợp(IC)
  + ~10 triệu linh kiện/foot3
  + Ví dụ: UNIVAC 9000 series, IBM System/360, System 3, System 7
* Thế hệ 4(1975-1989):
  + Sử dụng LSI – Large Scale Integrated Circuit
  + ~1 tỷ linh kiện/foot3
  + Ví dụ: IBM System 3090, IBM RISC 6000, IBM RT, Cray 2 XMP
* Thế hệ 5(1990-nay):
  + Sử dụng VLSI – Very Large Scale Integrated Circuit
  + <10nm – 0.045
  + Hỗ trợ xử lý song song
  + Hiệu năng rất cao
  + Kết hợp xử lý giọng nói và hình ảnh
  + Ví dụ: Pentium II, III, IV, M, D, Core Duo, Core 2 Duo, Core Quad,…
  1. Một số kiến trúc máy tính:
* Kiến trúc Von-Neumann:
  + Được giới thiệu bởi John von-Neumann vào năm 1945
  + Máy tính kiến trúc Von-Neumann dựa trên 3 khái niệm cơ bản:
    - Dữ liệu và lệnh được lưu trong một bộ nhớ đọc/viết chia sẻ
    - Bộ nhớ được đánh giá địa chỉ dựa trên đoạn và không phụ thuộc vào 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 được chia thành 3 giai đoạn chính:
    - CPU lấy lệnh 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)
* Kiến trúc Harvard:
  + Bộ nhớ được chia thành 2 phần:
    - Bộ nhớ chương trình(Program 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)
  + Đặc điểm của kiến trúc Harvard:
    - Nhanh hơn vì băng thông bus rộng vì 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 -> giảm xung đột truy cập bộ nhớ
    - Ngày nay kiến trúc Harvard cải tiến được ứng dụng cho các kiến trúc máy tính hiện đại: ARM, intel x86
    - Kiến trúc Harvard cũng được ứng dụng ở các hệ thống nhúng embedded system, chíp chuyên xử lý tín hiệu (DSP)
* So sánh kiến trúc Harvard và kiến trúc Von-Neumann:



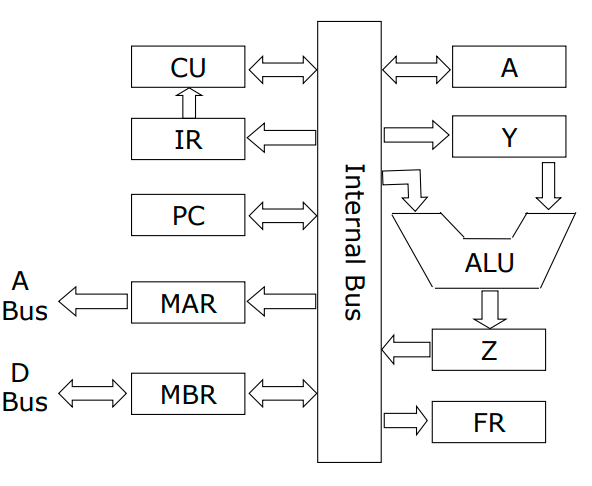
|  |  |
| --- | --- |
| Kiến trúc Harvard | Kiến trúc Von-Neumann |
| Kiến trúc Harvard có các đường dẫn vật lý riêng biệt cho lệnh và dữ liệu | Kiến trúc Von-Neumann sử dụng chung đường dẫn vật lý cho lệnh và dữ liệu |
| Có một bộ bus địa chỉ và dữ liệu chuyên dụng để đọc dữ liệu từu và viết dữ liệu đến bộ nhớ, và các bộ bus địa chỉ và dữ liệu khác để đọc lệnh | Có chung các bộ bus địa chỉ và dữ liệu cho bộ nhớ đọc/ghi và thực hiện lệnh |
| Dưới kiến trúc Harvard, CPU có thể vừa đọc một lệnh và thực hiện một truy cập bộ nhớ dữ liệu cùng lúc | Dưới kiến trúc Von Neumann nguyên thủy, CPU có thể đọc 1 lệnh hoặc đọc/viết dữ liệu từ/đến bộ nhớ. Cả hai không thể xảy ra cùng lúc vì các lệnh và dữ liệu sử dụng chung bus hệ thống |
| Máy tính kiến trúc Harvard có các địa chỉ dữ liệu và lệnh riêng biệt: địa chỉ lệnh 0 không giống địa chỉ dữ liệu 0 | Kiến trúc Von Neumann có chung địa chỉ dữ liệu và địa chỉ lệnh |

* 1. Hệ đếm và tổ chức lưu trữ dữ liệu trong hệ thống máy tính
* Hệ đếm :
  + Hệ nhị nhân: sử dụng 2 chữ số: 0 và 1
  + Hệ thập phân: sử dụng 10 chữ số: 0,1,2,3,4,5,6,7,8,9
  + Hệ thập lục phân: sử dụng 16 chữ số: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
* Tổ chức lưu trữ dữ liệu:
  + Dữ liệu trên máy tính được biểu diễn theo các đơn vị. Các đơn vị biểu diễn dữ liệu cơ sở gồm: bit, nibble, byte, word và double-word.
    - Bit:
      * Là đơn vị dữ liệu nhỏ nhất
      * Một bit chỉ có thể lưu trữ 2 giá trị: 0 hoặc 1, true hoặc false
    - Nibbles:
      * Nhóm 4 bits
      * Có thể lưu trữ tới 16 giá trị từ (0000)2 tới (1111)2, hoặc 1 số hệ hexa.
    - Bytes:
      * Nhóm 8 bít hoặc 2 nibbles
      * Có thể lưu trữ tới 256 giá trị, từ (0000 0000)2 tới(1111 1111)2 , hoặc từ (00)16­ tới (FF)2
    - Words:
      * Nhóm 16 bits hay 2 bytes
      * Có thể lưu tới 216 (65536) giá trị, từ (0000)­16 tới (FFFF)16
    - Double words:
      * Nhóm 32 bits, hoặc 4 bytes, hoặc 2 words
      * Có thể lưu tới 232 giá trị, từ (0000 0000)16 tới (FFFF FFFF)16

**Chương 2: Khối xử lý trung tâm**

2.1 Sơ đồ tổng quát và chức năng các thành phần

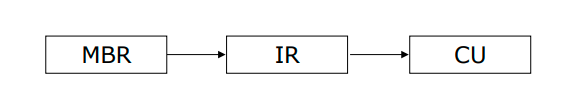
* Sơ đồ tổng quát:



* Chức năng các thành phần:
  + CU: (Control Unit)- Khối điều khiển
  + IR: (Instruction Register)- Thanh ghi lệnh
  + PC: (Program Counter)- Bộ đếm chương trình
  + MAR: (Memory Address Register)- Thanh ghi địa chỉ bộ nhớ
  + MBR: (Memory Buffer Register)- Thanh ghi nhớ đệm
  + A: (Accumulator Register)- Thanh ghi tích lũy
  + Y,Z: (Temporary Register)- Thanh ghi tạm thời
  + FR: (Flag Register)- Thanh ghi cờ
  + ALU: (Arithmetic and Logic Unit)- Khối tính toán số học- logic

2.2 Các thanh ghi:

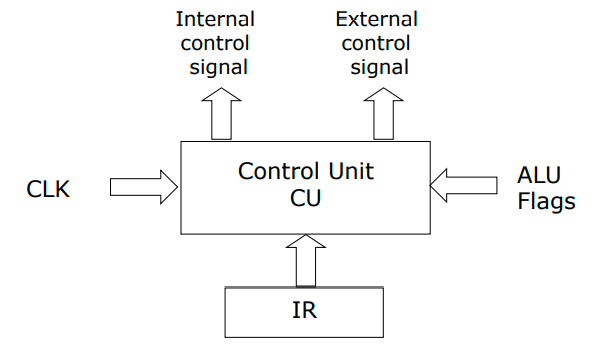
* Thanh ghi tích lũy A(Accumulator):
  + Thanh ghi tích lũy là một trong những thanh ghi quan trọng nhất của CPU
    - Lưu trữ các toán hạng đầu vào
    - Lưu kết quả đầu ra
  + Kích thước của thanh ghi A tương ứng với độ dài từ xử lý của CPU: 8, 16, 32, 64 bit
  + Cũng được sử dụng để trao đổi dữ liệu với các thiết bị vào ra
* Thanh ghi lệnh IR:
  + Lưu trữ lệnh đang được xử lý
  + IR lấy lệnh từ MBR và chuyển nó tới CU để giải mã lệnh



* Thanh ghi MBR và MAR:
  + MAR: (Memory Address Register)- Thanh ghi địa chỉ bộ nhớ:
    - Giao diện giữa CPU và bus địa chỉ
    - Nhận địa chỉ bộ nhớ của lệnh tiếp theo từ PC và chuyển nó tới bus địa chỉ
  + MBR: (Memory Buffer Register)- Thanh ghi đệm bộ nhớ:
    - Giao diện giữa CPU và bus dữ liệu
    - Nhận lệnh từ bus dữ liệu và chuyển nó tới IR
* Các thanh ghi tạm thời: CPU thường sử dụng một số thanh ghi tạm thời để:
  + Lưu trữ các toán hạng đầu vào
  + Lưu các kết quả đầu ra
  + Hỗ trợ xử lý song song (tại một điểm chạy nhiều hơn một lệnh)
  + Hỗ trợ thực hiện lệnh theo cơ chế thực hiện tiên tiến kiểu không trật tự
* Các thanh ghi đa năng:
  + Có thể sử dụng cho nhiều mục đích:
    - Lưu các toán hạng đầu vào
    - Lưu các kết quả đầu ra
  + VD: CPU 8086 có 4 thanh ghi đa năng:
    - AX: (Accumulator Register) – Thanh ghi tích lũy
    - BX: (Base Register) – Thanh ghi cơ sở
    - CX: (Counter Register) – Thanh ghi đếm
    - DX: (Data Register) – Thanh ghi dữ liệu
* Thanh ghi trạng thái/ Thanh ghi cờ(Flag Register):
  + 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
    - Cờ điều khiển: IF, TF, DF
  + Các bit cờ thường được dùng là các điều kiện rẽ nhánh lệnh tạo logic chương trình
  + Kích thước FR phụ thuộc thiết kế CPU

2.3 Khối điều khiển(CU:Control Unit)

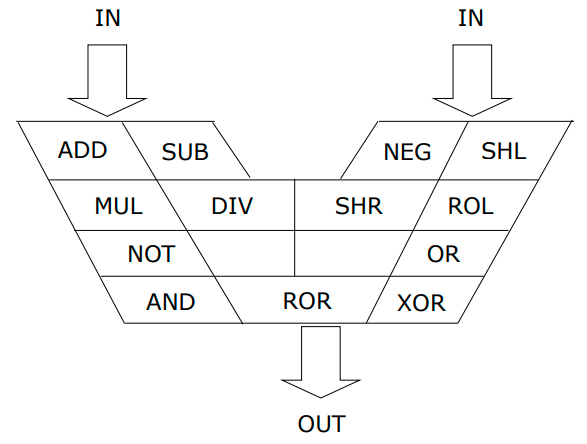
* Sơ đồ khối:



* Chức năng:
  + 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
  + CU sinh 2 nhóm tín hiệu đầu ra:
    - Internal control signal : Nhóm tín hiệu điều khiển các bộ phận bên trong CPU
    - External control signal: 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 vs các thành phần bên ngoài

2.4 Khối tính toán số học và lô-gic:

* Sơ đồ khối:



* Chức năng:
  + 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 phủ định(NOT), bộ và(AND), bộ hoặc(OR), bộ loại trừ(XOR)
  + ALU có:
    - 2 cổng IN để nhận đầu vào từ các thanh ghi
    - 1 cổng OUT được nối với bus trong để gửi kết quả tới các thanh ghi

2.5 Hệ thống bus trong:

* + 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 thay đổi thông tin với bus ngoài(bus hệ thống)
  + Bus trong luôn có băng thông lớn và tốc độ nhanh hơn so với bus ngoài

**Chương 3: Xử lý xen kẽ dòng mã lệnh và bộ nhớ Cache**

3.1 Cơ chế xử lý xen kẽ dòng mã lệnh (pipeline)

* Khái niệm:
  + Cơ chế xử lý xen kẽ dòng mã lệnh(pipeline) là một phương pháp thực hiện tiên tiến, cho phép đồng thời thực hiện nhiều lệnh, giảm thời gian trung bình thực hiện mỗi lệnh và như vậy tăng được hiệu năng xử lý của CPU
  + Việc thực hiện được chia thành một số giai đoạn và mỗi giai đoạn được thực thi bởi một đơn vị chức năng khác nhau của CPU. Nhờ vậy CPU có thể tận dụng tối đa năng lực xử lý của các đơn vị chức năng của mình, giảm thời gian chờ cho từng đơn vị chức năng. Có nhiều lệnh đồng thời được thực hiện gối nhau trong CPU và hầu hết các đơn vị chức năng của CPU liên tục tham gia vào quá trình xử lý lệnh. Số lượng lệnh được xử lý đồng thời đúng bằng số giai đoạn thực hiện lệnh
* Đặc điểm:
  + Pipeline là kỹ thuật song song ở mức lệnh(ILP: Instruction Level Parallelisn)
  + Một pipeline là đầy đủ nếu nó luôn nhận một lệnh mới tại mỗi chu kỳ đồng hồ
  + Một pipeline là không đầy đủ nếu có nhiều 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

3.2 Các vấn đề pipeline:

* Xung đột tài nguyên(Resource hazards):
  + 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
  + Hai xung đột tài nguyên thường gặp:
    - Xung đột truy cập bộ nhớ
    - Xung đột truy cập thanh ghi
  + Giải pháp:
    - Nâng cao khả năng tài nguyên
    - Memory/cache: hỗ trợ nhiều thao tác đọc/ ghi cùng lúc
    - Chia cache thành cache lệnh và cache dữ liệu để cải thiện truy nhập
* Xung đột dữ liệu(Data hazards):
  + Dạng xung đột dữ liệu hay gặp nhất là tranh chấp dữ liệu kiểu đọc sau khi ghi(RAW- Read After Write)
  + Giải pháp:
    - Nhận biết nó xảy ra
    - Ngưng pipeline (stall): làm trễ hoặc ngưng ống lệnh cho đến khi lệnh phía trước hoàn tất giai đoạn WB(Write Back- ghi kết quả vào các thanh ghi)
    - Sử dụng complier để nhận biết RAW và:
      * Chèn các lệnh NO-OP vào giữa các lệnh có RAW
      * 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 tranh chấp RAW (có trong các CPUs hiện đại) và dự đoán trước giá trị dữ liệu phụ thuộc
* Xung đột rẽ nhánh(Branch hazards):
  + Tỷ lệ các lệnh rẽ nhánh chiếm khoảng 10-30%.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
  + Giải pháp:
    - Đích rẽ nhánh(brach targets)
    - Làm chậm rẽ nhánh(delayed branching):
      * Chèn thêm một lệnh NO-OP vào ngay sau lệnh rẽ nhánh
      * Chèn một lệnh độc lập vào ngay sau lệnh rẽ nhánh
    - Dự báo rẽ nhánh(branch prediction)

3.3 Xử lý siêu xen kẽ dòng mã lệnh (hyper-pipeline):

* Siêu pipeline là kỹ thuật cho phép:
  + Tăng độ sâu ống lệnh
  + Tăng tốc độ đồng hồ
  + Giảm thời trễ cho từng giai đoạn thực hiện lệnh
* Ví dụ: nếu giai đoạn thực hiện lệnh bởi ALU kéo dài-> chia thành một số giai đoạn nhỏ-> giảm thời gian chờ cho các giai đoạn ngắn

3.4 Bộ nhớ Cache: nguyên lý và tổ chức:

* Nguyên lý hoạt động bộ nhớ cache:
  + Cache được coi là bộ nhớ thông minh do:
    - Cache 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 cập cache -> giảm thời gian truy cập bộ nhớ
  + Để có được sự thông minh, cache hoạt động dựa trên 2 nguyên lý cơ bản:
    - Nguyên lý cục bộ/lân cận về không gian (spatial locality):
      * Nếu một vị trí bộ nhớ được truy cập, thì khả năng/ xác suất các vị trí gần đó được truy cập trong thời gian gần tới là cao
      * Áp dụng với các mục dữ liệu và các lệnh có thứ tự tuần tự theo chương trình
      * Hầu hết các lệnh trong chương trình có thứ tự tuần tự, do đó cache đọc một khối lệnh trong bộ nhớ, mà bao gồm cả các phần tử xung quanh vị trí phần tử hiện tại được truy cập
    - Nguyên lý cục bộ/lân cận về thời gian (temporal locality):
      * Nếu một vị trí bộ nhớ được truy cập, thì khả năng nó sẽ được truy cập lại trong thời gian gần tới là cao
      * Áp dụng với các mục dữ liệu và các lệnh trong vòng lặp
      * Cache đọc khối dữ liệu trong bộ nhớ bao gồm tất cả các thành phần trong vòng lặp
* Tổ chức bộ nhớ cache:
  + Giải quyết vấn đề cache và bộ nhớ chính phối hợp làm việc với nhau như thế nào
  + Ba phương pháp tổ chức/ánh xạ cache:
    - Ánh xạ trực tiếp(direct mapping): đơn giản, nhanh, ánh xạ cố định :
      * Main memory chia thành n block ( block size= cache size ). Cache chia thành n line.
      * Mapping : line i of n block🡪 line i of cache.
      * Address = Tag + Block + Word
        + Tag là địa chỉ của block trong Main memory
        + Block là địa chỉ line ( offset-line )
        + Word là địa chỉ từng từ trong line ( offset-word )
    - Ánh xạ kết hợp đầy đủ(fully associative mapping): phức tạp, chậm, ánh xạ linh hoạt :
      * Main memory chia thành m line (m >> n). Cache chia thành n line.
      * Mapping : line i of line(m) -> line j of line(n).
      * Address = Tag + word
        + Tag là địa chỉ line trong RAM
        + Word là địa chỉ từng từ trong 1 line
    - Ánh xạ tập kết hợp/ theo bộ(set associative mapping): phức tạp, nhanh, ánh xạ linh hoạt :
      * Main memory chia thành m block ( block size = 1 way size). Cache chia thành L way, mỗi way có n line. Tập hợp các line giống nhau thuộc các way gọi là set. ( địa chỉ set và line như nhau )
      * Mapping : line i (0 – n-1 )of block x ( 0 -> m-1 ) 🡪 line j of block y ( y là way bất kì trong cache ).
      * Address = Tag + Set + Word
        + Tag là địa chỉ block trong RAM
        + Set là địa chỉ line
        + Word địa chỉ từ trong line.

3.5 Chính sách thay thế dòng và hiệu năng bộ nhớ Cache

* Chính sách thay thế(replacement policies):

Xác định cách thức lựa chọn các dòng trong cache để thay thế khi có dòng mới từ bộ nhớ cần chuyển vào

Ba chính sách thay thế:

* + Thay thế ngẫu nhiên(Random replacement): các dòng trong cache được chọn ngẫu nhiên để thay
    - Thiết kế đơn giản, dễ cài đặt
    - Tỷ lệ miss cao vì phương pháp này không xét tới dòng cache nào đang thực sự được sử dụng
  + Thay thế kiểu vào trước ra trước FIFO(First In First Out): các dòng cache được đọc vào cache trước sẽ được chọn để thay trước
    - Thiết kế, cài đặt phức tạp vì cần thêm mạch để giám sát thứ tự nạp các dòng bộ nhớ vào cache
    - Tỷ lệ miss thấp hơn thay thế ngẫu nhiên tuy nhiên vẫn cao vì phương pháp này vẫn chưa thực sự xem xét tới block nào đang thực sự được sử dụng
  + Thay thế các dòng ít được sử dụng gần đây nhất LRU(Least Recently Used): các dòng cache ít được sử dụng nhất trong thời gian gần đây sẽ được chọn để thay
    - Thiết kế, cài đặt phức tạp vì cần thêm mạch để giám sát tần suất sử dụng các dòng cache
    - Tỷ lệ miss thấp nhất so với thay thế ngẫu nhiên và thay thế FIFO vì thay thế LRU có xem xét đến các dòng đang thực sự được sử dụng-tuân theo yếu tố lân cận theo thời gian một cách chặt chẽ.
* Hiệu năng cache:
  + Đánh giá theo thời gian truy cập trung bình của hệ thống nhớ có cache:

taccess = (hit cost) + (miss cost)

= (hit cost) + (miss rate) \* (miss penalty)

= tcache + (1 - H) \* (tmemory)

Với taccess là thời gian truy nhập trung bình hệ thống nhớ

t­memory là thời gian truy nhập bộ nhớ chính

tcache là thời gian truy nhập cache

H là hệ số hit

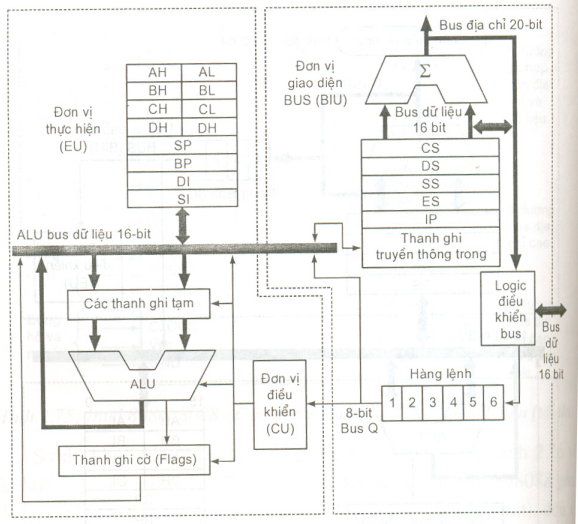
3.6 Công nghệ lưu trữ dữ liệu RAID:

* RAID(Redundant Array ò Independent Disks) là một công nghệ tạo các thiết bị lưu trữ tiên tiến trên cơ sở các ổ đĩa cứng, nhằm đạt các yêu cầu về tốc độ cao(high performance/speed), tính tin cậy cao(high reliability) và dung lượng lớn(large volume).
* Các kỹ thuật tạo RAID:
  + Kỹ thuật tạo lát đĩa(Disk Stripping): tốc độ đọc cao – các dữ liệu cần ghi được chia thành các khối cùng kích thước và được ghi đồng thời vào các ổ đĩa vật lý độc lập
  + Kỹ thuật soi gương đĩa(Disk Mirroring): tính an toàn cao – dữ liệu được chia thành các khối và mỗi khối được ghi đồng thời lên hai hay nhiều ổ đĩa độc lập
* RAID thông dụng:
  + RAID 0: dựa trên kỹ thuật tạo lát đĩa -> tốc độ cao
  + RAID 1: dựa trên kỹ thuật soi gương đĩa -> tính tin cậy cao
  + RAID 10: dựa trên cả 2 kỹ thuật -> tốc độ cao, tính tin cậy cao

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

4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088

* Kiến trúc của bộ vi xử lý 8086/8088:



* Các thành phần của bộ vi xử lý 8086/8088:
  + Đơn vị giao tiếp bus BIU(Bus Interface Unit):
    - Bộ cộng để tính địa chỉ
    - 4 thanh ghi đoạn 16-bit:CS, DS, SS, ES
    - Bộ đếm chương trình/con trỏ lệnh 16-bit (PC/IP)
    - Hàng đợi lệnhIQ(Instruction Queue): 4 bytes trong 8088 và 6 bytes trong 8086
    - Logic điều khiển bus
    - Bus dữ liệu 16 bit
  + Đơn vị thực hiện(EU:Execution Unit)
    - ALU
    - CU
    - 8 thanh ghi 16-bit: AX, BX, CX, DX, SP, BP, SI, DI
    - Thanh ghi cờ FR
  + Bus trong (Internal Bus): liên kết BIU và EU
    - 16-bit A-BUS trong 8088
    - 16-bit ALU-BUS trong 8086
  + Các thanh ghi đa năng:
    - Các thanh ghi con trỏ và chỉ số
    - Các thanh ghi đoạn
    - Con trỏ lệnh và thanh ghi cờ

4.2. Mã hóa lệnh và các chế độ địa chỉ

4.3 Tập lệnh và công cụ Emu8086.

4.4 Lập trình hợp ngữ

4.5 Một số ví dụ