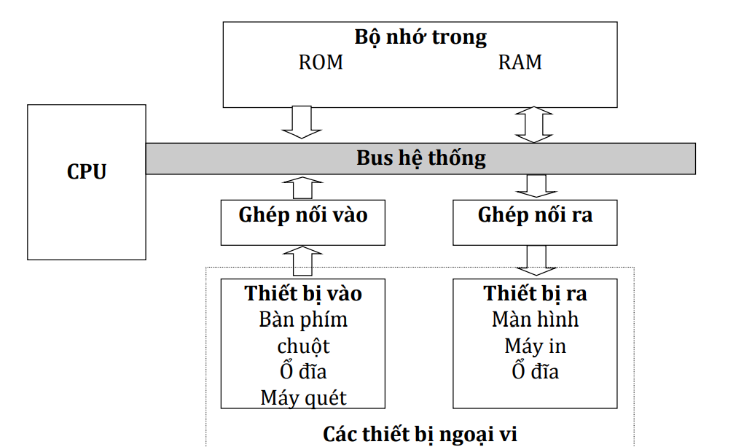
Computer Architecture

**Chương 1: Tổng quan**

* 1. **- KTMT và các thành phần. Vẽ sơ đồ khối chức năng của hệ thống máy tính:**
* Là khoa học về lựa chọn và kết nối các thành phần cứng của máy tính, nhằm đạt yêu cầu: *hiệu năng cao, chức năng nhiều, giá thành rẻ*
* 3 thành phần cơ bản:
  + Kiến trúc tập lệnh (ISA)
  + Vi kiến trúc
  + Thiết kế hệ thống

|  |  |  |
| --- | --- | --- |
|  | **Chức năng** | **Thành phần** |
| **CPU** | Bộ sử lý trung tâm:   * Đọc lệnh * Giải mã * Thực hiện lệnh | 4 phần:   * CU: Khối điều khiển * Thanh ghi * ALU * Bus trong CPU |
| **Bộ nhớ trong** | * Lưu trữ lệnh và dữ liệu của để CPU xử lý | 2 phần:   * ROM: thông tin vẫn tồn tại khi ngắt nguồn * RAM: thông tin mất khi ngắt nguồn |
| **Bus hệ thống** | Kết nối CPU với các thành phần khác | 3 phần:   * Bus Address * Bus Data * Bus Control |
| **Các thiết bị ngoại vi** | * Vào: nhập dữ liệu và điều khiển * Ra: kết xuất dữ liệu | 2 phần:   * Vào, vd: bàn phím, chuột,… * Ra, vd: Màn hình, máy in,… |

|  |  |  |  |
| --- | --- | --- | --- |
| **Giai đoạn** | **Sử dụng** | **Mật độ** | **Ví dụ** |
| 1949-1959 | Bóng đèn điện tử | 1.000 liên kết/foot3 | ENIAC |
| 1960-1964 | Transistors | 100.000 | UNIVAC 1107 |
| 1964-1975 | IC | 10 tr | UNIVAC 9000 |
| 1975-1989 | LSI | 1 tỷ | IBM System 3090 |
| 1990-nay | VLSI | <10nm-0.045µm | Pentium 2,3,4,… |

* 1. **- Lịch sử phát triển:**
  2. **- Von-Neuman & Harvard:**

1. **Von – Neuman (1945):**

* 3 KN cơ bản:

+ Data/lệnh – lưu trong bộ nhớ đọc/viết

+ Đánh giá địa chỉ dựa trên đoạn – không phụ thuộc vào dữ liệu

+ Các lệnh chạy lần lượt, nối tiếp

* 3 gđoạn chính:

+ CPU lấy lệnh từ memory

+ CPU giải mã – thực hiện lệnh; nếu cần thì lấy data từ mem

+ CPU viết kq vào mem (nếu có)

1. **Harvard:**

* 2 thành phần:

+ Program Memory

+ Data Memory

* CPU sử dụng 2 bus hệ thống:

+ có thể đọc – truy cập lệnh cùng lúc

+ 2 bus khác nhau về định dạng: 1 bus A,D cho program; 1 bus A,D cho data

* Đặc điểm:

+ Nhanh hơn (vì băng thông rộng, không cần tranh chấp)

+ Giảm xung đột (đọc/xuất data cùng lúc)

+ Được cải tiến cho máy tính hiện đại (intel x86)

+ Được sử dụng ở các hệ thống nhúng

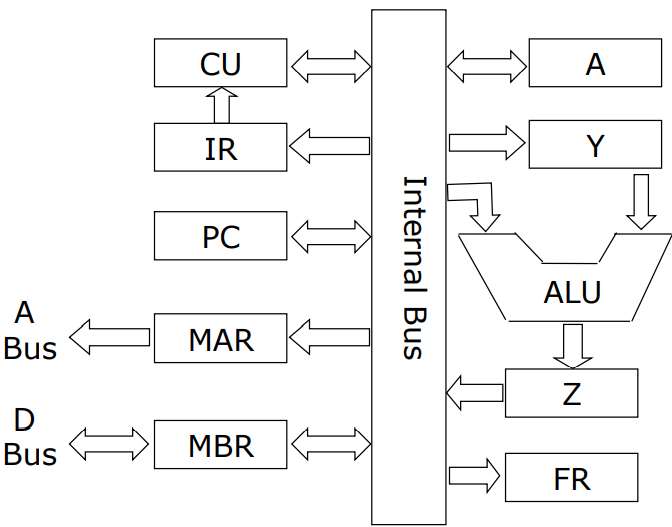
1. **So sánh:**

|  |  |
| --- | --- |
| **Von - Neuman** | **Harvard** |
| Sử dụng chung đường dẫn cho lệnh và data | Riêng biệt |
| Chung bộ bus A,D cho bộ nhớ đọc/ghi, thực hiện lệnh | Có 2 bộ bus A,D riêng biệt:  + 1 bộ để đọc/ghi  + 1 bộ để thực hiện lệnh |
| Đọc/ghi, thực hiện lệnh lần lượt theo thứ tự, chỉ thực hiện 1 lệnh 1 lúc | Có thể vừa đọc và thực hiện lệnh cùng lúc |
| Có chung địa chỉ data và địa chỉ lệnh | Các địa chỉ riêng biệt: Địa chỉ lệnh 0 ! địa chỉ data 0 |
| Tóm lai: là khởi đầu, nhiều bộ phận sử dụng chung | Tóm lại: hiện đại (các bộ phận riêng biệt, nhanh hơn, giảm xung đột, ứng dụng cao,…) |

* 1. **- Hệ đếm, tổ chức lưu trữ:**

1. Hệ đếm: 2, 8, 10, 16, …
2. Tổ chức lưu trữ dữ liệu:

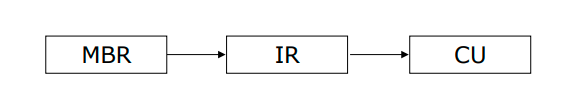
|  |  |  |
| --- | --- | --- |
| Bit |  | 1 or 0 |
| Nibble | 4 bits | 16 giá trị:  00002 – 11112 (hoặc từ 0-F) |
| Byte | 8 bits | 256 giá trị  0000 00002 – 1111 11112 (hoặc 00 – FF) |
| Word | 16 bits = 2 bytes | 216 giá trị  (0000)16 – (FFFF)16 |
| Double – Word | 32 bits = 4 bytes | 232 giá trị  (0000 0000)16 – (FFFF FFFF)16 |

**Chương 2: Khối xử lý trung tâm**

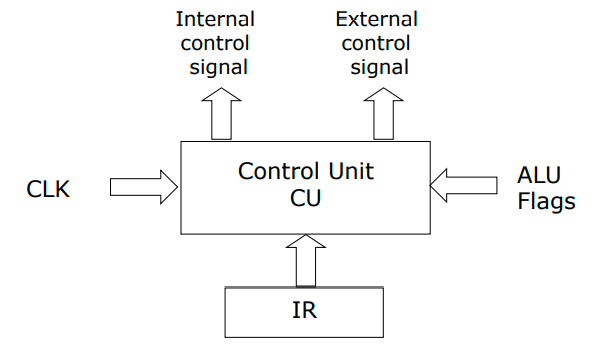
**2.1 - Các thành phần:**

|  |  |  |
| --- | --- | --- |
| **Thành phần** | | **Chức năng** |
| 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 đệm bộ nhớ |
| A | Accumulator Register | Thanh ghi tích lũy |
| Y, Z | Temporary Register | Thanh ghi tạm thời |
| FR | Flag Register | Thanh ghi cờ |
| ALU | Arthmetic and Logic Unit | Khối tính toán số học - logic |

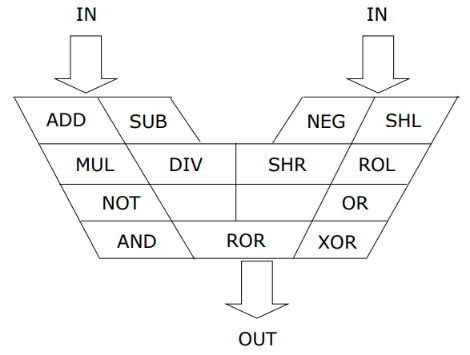
**2.2 - Register:**

* 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)

* 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:

* 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: Cache**

**3.1 Nguyên lý:** 2 nguyên lý lân cân/cục bộ về:

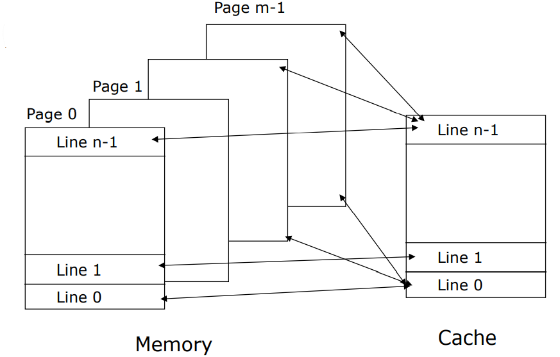
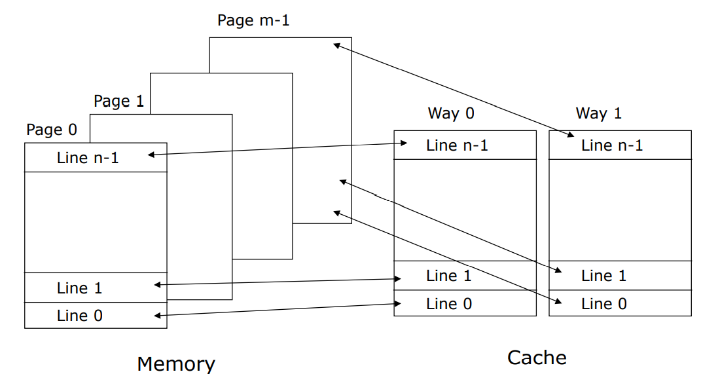
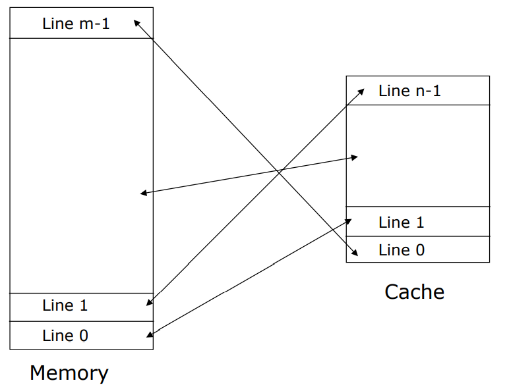
|  |  |
| --- | --- |
| Không gian (Spatial locality) | Thời gian (Temporal locality) |
| *Nếu một vị trí được truy cập*  *Thì khả năng cao* **các vị trí** gần đó cũng sẽ được truy cập trong lần tới | *Nếu một vị trí được truy cập*  *Thì khả năng cao* nó sẽ được truy cập **lại** trong thời gian tới |
| *Áp dụng với các mục dữ liệu và các lệnh* có **thứ tự tuần tự** trong chương trình | *Áp dụng với các mục dữ liệu và các lệnh* **trong vòng lặp** |
| Cache đọc 1 khối lệnh, bao gồm cả **các phần tử xung quanh** vị trí hiện tại được truy cập | Cache đọc một khối dữ liệu, bao gồm tất cả thành phần **trong vòng lặp** |

**3.2 Tổ chức bộ nhớ**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Ánh xạ trực tiếp**  *(Direct Mapping)* | **Ánh xạ kết hợp đầy đủ**  *(Fully Associative Mapping)* | **Ánh xạ tập kết hợp/theo bộ**  *(Set Associative Mapping)* |
| Ưu điểm | Đơn giản, nhanh | * Tránh xung đột * Tỉ lệ hit cao | * Nhanh, ít xung đột * Tỉ lệ hit cao |
| Nhược điểm | * Khả năng xung đột cao * Tỉ lệ hit thấp | Phức tạp, chậm | Phức tạp |
| Cache | Được chia thành n khối hoặc dòng (block/line)  Từ Line0 – Line n-1 | | * Được chia thành k đường (ways) có kích thước bằng nhau * Mỗi đường chia thành Line0 – Line n-1 |
| Bộ nhớ | * Được chia thành m page   từ page0 – pagem-1   * Mỗi page có bộ nhớ kích thước = cache * Mỗi trang có n lines Line0 – Line n-1 | * Được chia thành m khối từ Line0 – Linem-1 * Kích thước mỗi dòng cache = kích thước 1 dòng bộ nhớ * m có thể >>n | Được chia thành m page từ page0 – pagem-1   * Kích thước page = kích thước way(cache) * Mỗi trang có n lines Line0 – Line n-1 |
| Ánh xạ | * Cố định * Linei (page0 - pagen-1) ax tới Linei (cache) | * Linh hoạt * Linei (bộ nhớ) ax tới Linej (cache) | * Linh hoạt * Linei (pagek) ax tới Linei của wayj |
| Địa chỉ ax | = Tag + Line + Word | = Tag + Word | = Tag + Set + Word |

\*note:

- Tag(bit): địa chỉ của trang trong bộ nhớ - Set(bit): địa chỉ của line trong way của cache

- Line(bit): địa chỉ của line trong cache - Word(bit): địa chỉ của word trong line****

**Chương 3: Pineline**

* 1. **Cơ chế xử lý xen kẽ dòng mã lệnh (pipeline)**
* Khái niệm: 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
* 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