**A white paper with black text

Description automatically generated**

**1. Computer Architecture (Kiến trúc máy tính):**

* **Định nghĩa:**
  + Tập trung vào thiết kế logic, khái niệm, và các thuộc tính chức năng của hệ thống máy tính.
  + Được định nghĩa ở mức trừu tượng, nhằm mô tả cách hệ thống hoạt động về mặt chức năng.
* **Mục tiêu:**
  + Đảm bảo hệ thống đáp ứng được các yêu cầu quan trọng như hiệu suất, chi phí, độ tin cậy, và khả năng mở rộng.
* **Nội dung chính:**
  + **Phối hợp thành phần:** Cách CPU, bộ nhớ, và I/O phối hợp để thực thi các lệnh.
  + **Instruction Set Architecture (ISA):** Tập hợp các lệnh mà CPU có thể thực thi, giao diện chính giữa phần cứng và phần mềm.
  + **Người dùng/lập trình viên:** Các khía cạnh liên quan đến cách lập trình viên hoặc người dùng tương tác với phần cứng.
* **Ví dụ:**
  + Thiết kế tập lệnh RISC (Reduced Instruction Set Computer) vs. CISC (Complex Instruction Set Computer).
  + Cấu trúc xử lý song song (Parallel Processing).
  + Bộ nhớ đệm nhiều cấp (Cache Levels: L1, L2, L3).

**2. Computer Organization (Tổ chức máy tính):**

* **Định nghĩa:**
  + Tập trung vào cách các thành phần vật lý trong hệ thống máy tính được kết nối và hoạt động cùng nhau để thực hiện kiến trúc đã định.
* **Mục tiêu:**
  + Hiểu và tối ưu hóa hiệu suất thực tế của hệ thống dựa trên thiết kế kiến trúc.
* **Nội dung chính:**
  + **Luồng dữ liệu:** Cách dữ liệu được di chuyển và lưu trữ trong hệ thống.
  + **Thiết kế phần cứng:** Tập trung vào thiết kế chi tiết của bộ vi xử lý, bus, thanh ghi, bộ đệm, và các thành phần vật lý khác.
  + **Điều khiển tín hiệu:** Tổ chức và điều khiển các mạch điện tử, tín hiệu vật lý để thực thi lệnh.
* **Ví dụ:**
  + Cách mạch logic của ALU thực hiện các phép toán.
  + Quản lý DMA (Direct Memory Access) để tăng tốc truyền dữ liệu.
  + Bộ điều khiển (Control Unit) xử lý các lệnh cụ thể.

**So sánh tóm tắt:**

| **Tiêu chí** | **Computer Architecture** | **Computer Organization** |
| --- | --- | --- |
| **Mức độ trừu tượng** | Cao, tập trung vào thiết kế logic | Thấp, tập trung vào phần cứng thực tế |
| **Đối tượng chính** | Người dùng, lập trình viên | Kỹ sư phần cứng |
| **Mục tiêu** | Xác định hệ thống sẽ làm gì | Hiện thực hóa cách hệ thống hoạt động |
| **Ví dụ câu hỏi chính** | "Hệ thống này có thể làm gì?" | "Hệ thống này sẽ làm thế nào?" |

**Phân Cấp Máy Tính**

**1. Digital Logic Level (Mức logic số)**

* **Chức năng:**
  + Là cấp thấp nhất, nơi dữ liệu được biểu diễn và xử lý dưới dạng tín hiệu nhị phân (0 và 1).
  + Dựa trên các mạch điện tử như transistor và các cổng logic (AND, OR, NOT).
* **Thành phần:**
  + Transistor, cổng logic, mạch điện tử.

**2. Microarchitecture Level (Mức vi kiến trúc)**

* **Chức năng:**
  + Xác định cách các thành phần vật lý trong bộ xử lý (CPU) được kết nối và phối hợp để thực hiện các lệnh cơ bản.
  + Xử lý thông qua ALU (Arithmetic Logic Unit), thanh ghi, bộ điều khiển, và bộ nhớ cache.
* **Thành phần:**
  + Bộ xử lý (CPU), thanh ghi, bộ đệm, bộ điều khiển lệnh.

**3. Instruction Set Architecture (ISA) Level (Mức tập lệnh)**

* **Chức năng:**
  + Giao diện giữa phần mềm và phần cứng.
  + Định nghĩa tập hợp các lệnh (Instruction Set) mà CPU có thể thực hiện, ví dụ: LOAD, STORE, ADD, SUBTRACT, JUMP.
* **Thành phần:**
  + Tập lệnh, bộ điều khiển lệnh.

**4. Operating System Level (Mức hệ điều hành)**

* **Chức năng:**
  + Quản lý tài nguyên phần cứng (CPU, bộ nhớ, thiết bị I/O) và cung cấp môi trường hoạt động cho các chương trình phần mềm.
  + Điều khiển luồng dữ liệu giữa các thành phần hệ thống.
* **Thành phần:**
  + Hệ điều hành (Windows, Linux, macOS), trình điều khiển thiết bị (Device Drivers).

**5. Assembly Language Level (Mức ngôn ngữ hợp ngữ)**

* **Chức năng:**
  + Cung cấp cú pháp dễ hiểu hơn so với mã máy, cho phép lập trình viên viết các chương trình gần với phần cứng.
  + Dịch hợp ngữ thành mã nhị phân để phần cứng thực thi.
* **Thành phần:**
  + Assembler (trình biên dịch hợp ngữ), mã hợp ngữ (MOV, ADD, JMP).

**6. High-Level Language Level (Mức ngôn ngữ cấp cao)**

* **Chức năng:**
  + Cho phép lập trình viên viết chương trình bằng ngôn ngữ gần với ngôn ngữ tự nhiên (Python, Java, C++).
  + Trình biên dịch (Compiler) hoặc trình thông dịch (Interpreter) sẽ chuyển đổi mã cấp cao thành mã máy.
* **Thành phần:**
  + Ngôn ngữ lập trình cấp cao (C, Java, Python), trình biên dịch (Compiler), trình thông dịch (Interpreter).

**Tóm tắt các cấp độ:**

| **Cấp** | **Tên gọi** | **Chức năng chính** |
| --- | --- | --- |
| **1** | Digital Logic | Xử lý tín hiệu nhị phân, nền tảng vật lý của máy tính. |
| **2** | Microarchitecture | Tổ chức và tối ưu hóa cách thực hiện lệnh trong CPU. |
| **3** | Instruction Set Architecture | Giao diện giữa phần mềm và phần cứng. |
| **4** | Operating System | Quản lý tài nguyên và cung cấp môi trường hoạt động. |
| **5** | Assembly Language | Ngôn ngữ gần phần cứng, dịch hợp ngữ thành mã nhị phân. |
| **6** | High-Level Language | Lập trình ngôn ngữ cấp cao, chuyển mã thành mã máy. |

**Các thành phần của hệ thống máy tính**

**1. Bộ xử lý trung tâm (CPU - Central Processing Unit):**

* **Chức năng:**
  + Là "bộ não" của máy tính, thực hiện xử lý dữ liệu và điều khiển hoạt động của các thành phần khác.
  + Thực hiện các phép tính toán số học, logic, và điều khiển chương trình.
* **Thành phần:**
  + **ALU (Arithmetic Logic Unit):** Xử lý các phép toán số học và logic.
  + **CU (Control Unit):** Điều phối các hoạt động trong máy tính, đọc lệnh từ bộ nhớ và thực thi.
  + **Thanh ghi (Registers):** Bộ nhớ tạm thời trong CPU, lưu trữ dữ liệu và lệnh trong quá trình xử lý.

**2. Bộ nhớ (Memory):**

* **Chức năng:**
  + Lưu trữ dữ liệu và chương trình cần thiết cho hoạt động của máy tính.
* **Phân loại:**
  + **Bộ nhớ chính (Primary Memory):**
    - **RAM (Random Access Memory):** Lưu trữ tạm thời dữ liệu và lệnh, bị mất khi tắt máy.
    - **ROM (Read-Only Memory):** Lưu trữ cố định, chứa chương trình khởi động (BIOS/firmware).
  + **Bộ nhớ phụ (Secondary Memory):**
    - Lưu trữ lâu dài dữ liệu, như ổ cứng (HDD), ổ SSD, USB.
  + **Bộ nhớ đệm (Cache):**
    - Tốc độ cao, nằm gần CPU, lưu trữ dữ liệu thường xuyên truy cập để tăng tốc độ xử lý.

**3. Thiết bị nhập/xuất (I/O - Input/Output Devices):**

* **Chức năng:**
  + **Input:** Nhận dữ liệu từ bên ngoài vào hệ thống.
  + **Output:** Xuất dữ liệu từ hệ thống ra bên ngoài.
* **Phân loại:**
  + **Thiết bị đầu vào (Input Devices):**
    - Bàn phím, chuột, máy quét, micro, camera.
  + **Thiết bị đầu ra (Output Devices):**
    - Màn hình, máy in, loa.
  + **Thiết bị I/O kết hợp:**
    - Màn hình cảm ứng, ổ USB, ổ đĩa.

**4. Bus (Hệ thống đường truyền):**

* **Chức năng:**
  + Truyền dữ liệu, địa chỉ, và tín hiệu điều khiển giữa các thành phần của hệ thống.
* **Phân loại:**
  + **Data Bus:** Truyền dữ liệu giữa các thành phần.
  + **Address Bus:** Truyền địa chỉ để xác định nơi lưu hoặc lấy dữ liệu.
  + **Control Bus:** Truyền tín hiệu điều khiển, đồng bộ hóa các hoạt động.

Thành phần Vai trò chính

CPU Xử lý dữ liệu và điều khiển hệ thống.

Bộ nhớ Lưu trữ dữ liệu và chương trình.

Thiết bị I/O Tương tác với người dùng và môi trường ngoài.

Bus Kết nối và truyền dữ liệu giữa các thành phần.

**Các thành phần trong CPU**

**1. CU (Control Unit - Bộ điều khiển):**

* **Chức năng:**
  + Điều khiển và phối hợp hoạt động giữa các thành phần của CPU.
  + Giải mã lệnh và gửi tín hiệu điều khiển đến các bộ phận khác để thực thi lệnh.
* **Vai trò:**
  + Đọc lệnh từ bộ nhớ thông qua **Program Counter (PC)**.
  + Giải mã lệnh trong **Instruction Register (IR)**.
  + Điều khiển dòng dữ liệu giữa CPU, bộ nhớ và các thiết bị ngoại vi.
* **Ví dụ hoạt động:**
  + Trong chu kỳ thực thi lệnh, CU đảm nhận việc quản lý trình tự **Fetch - Decode - Execute**.

**2. ALU (Arithmetic Logic Unit - Bộ số học và logic):**

* **Chức năng:**
  + Thực hiện các phép toán số học (cộng, trừ, nhân, chia).
  + Thực hiện các phép toán logic (AND, OR, NOT, XOR).
  + So sánh dữ liệu (lớn hơn, nhỏ hơn, bằng nhau).
* **Vai trò:**
  + Là trung tâm xử lý dữ liệu của CPU.
  + Kết quả từ ALU được lưu tạm thời trong các thanh ghi hoặc ghi trở lại bộ nhớ.
* **Ví dụ hoạt động:**
  + Khi thực hiện lệnh ADD R1, R2, R3 (cộng giá trị từ thanh ghi R2 và R3, lưu vào R1), ALU chịu trách nhiệm thực hiện phép cộng.

**3. Registers (Thanh ghi):**

* **Chức năng:**
  + Lưu trữ tạm thời dữ liệu, lệnh, hoặc địa chỉ trong quá trình xử lý.
  + Hoạt động nhanh hơn so với RAM, thường dùng để lưu trữ dữ liệu mà CPU cần truy cập liên tục.
* **Phân loại:**
  + **General Purpose Registers (Thanh ghi đa dụng):**
    - Lưu trữ dữ liệu hoặc địa chỉ phục vụ cho các phép toán.
  + **Special Purpose Registers (Thanh ghi chuyên dụng):**
    - **Program Counter (PC):** Chứa địa chỉ của lệnh tiếp theo cần thực thi.
    - **Instruction Register (IR):** Lưu trữ lệnh hiện tại đang được giải mã.
    - **Accumulator (AC):** Lưu kết quả tạm thời của các phép toán trong ALU.
    - **Status Register (SR):** Lưu trạng thái của các phép toán, như cờ (Flags) để xác định kết quả (ví dụ: Zero, Carry, Overflow).

**4. Internal Bus (Bus nội bộ):**

* **Chức năng:**
  + Truyền dữ liệu, địa chỉ, và tín hiệu điều khiển giữa các thành phần bên trong CPU.
  + Kết nối các bộ phận như ALU, CU, các thanh ghi, và bộ nhớ đệm.
* **Phân loại:**
  + **Data Bus (Bus dữ liệu):** Truyền dữ liệu giữa các thanh ghi, ALU và bộ nhớ.
  + **Address Bus (Bus địa chỉ):** Chỉ định địa chỉ trong bộ nhớ hoặc thiết bị I/O để CPU truy xuất.
  + **Control Bus (Bus điều khiển):** Truyền tín hiệu điều khiển, đồng bộ hóa hoạt động.
* **Vai trò:**
  + Đảm bảo các thành phần bên trong CPU có thể trao đổi dữ liệu một cách hiệu quả.

**Tóm tắt mối quan hệ giữa các thành phần:**

| **Thành phần** | **Vai trò chính** |
| --- | --- |
| **CU (Control Unit)** | Giải mã lệnh và điều phối hoạt động của các thành phần trong CPU. |
| **ALU (Arithmetic Logic Unit)** | Thực hiện các phép toán số học và logic. |
| **Registers (Thanh ghi)** | Lưu trữ tạm thời dữ liệu và lệnh trong quá trình xử lý. |
| **Internal Bus** | Kết nối và truyền dữ liệu giữa các thành phần bên trong CPU. |

**Quy trình phối hợp:**

1. **Fetch (CU):** CU lấy lệnh từ bộ nhớ thông qua PC và lưu vào IR.
2. **Decode (CU):** CU giải mã lệnh và chuẩn bị các tín hiệu điều khiển.
3. **Execute (ALU):** ALU thực hiện các phép toán theo lệnh, sử dụng dữ liệu từ các thanh ghi.
4. **Store (Registers):** Kết quả từ ALU được lưu vào thanh ghi hoặc ghi trở lại bộ nhớ qua internal bus.

**Vai trò của một số thanh ghi trong CPU**

**1. Program Counter (PC) / Instruction Pointer (IP):**

* **Chức năng:**
  + Chứa địa chỉ của lệnh tiếp theo cần được lấy và thực thi từ bộ nhớ.
  + Sau khi CPU thực thi một lệnh, PC sẽ tự động tăng lên để trỏ đến lệnh tiếp theo trong chương trình.
* **Vai trò:**
  + Giúp CPU theo dõi tiến trình thực thi của chương trình.
  + Đảm bảo rằng CPU thực thi các lệnh theo thứ tự từ bộ nhớ.
  + Trong các chương trình nhánh (branch), PC có thể thay đổi để trỏ đến các địa chỉ lệnh khác.

**2. Instruction Register (IR):**

* **Chức năng:**
  + Lưu trữ lệnh hiện tại mà CPU đang giải mã và thực thi.
  + Sau khi lệnh được tải từ bộ nhớ vào IR, nó sẽ được giải mã và thực hiện bởi CU (Control Unit).
* **Vai trò:**
  + Giữ lệnh trong quá trình giải mã và thực thi.
  + Là nơi mà CU đọc và kiểm tra lệnh để quyết định hành động tiếp theo.

**3. Memory Address Register (MAR):**

* **Chức năng:**
  + Lưu trữ địa chỉ của ô nhớ mà CPU muốn truy xuất (đọc hoặc ghi).
  + Khi CPU cần truy xuất dữ liệu từ bộ nhớ, MAR chứa địa chỉ của ô nhớ cần truy cập.
* **Vai trò:**
  + Chỉ định địa chỉ bộ nhớ mà CPU muốn đọc hoặc ghi vào.
  + Làm cầu nối giữa CPU và bộ nhớ trong quá trình truy xuất dữ liệu.

**4. Memory Buffer Register (MBR) / Memory Data Register (MDR):**

* **Chức năng:**
  + Lưu trữ dữ liệu được lấy từ bộ nhớ hoặc dữ liệu chuẩn bị để ghi vào bộ nhớ.
  + MBR nhận dữ liệu từ bộ nhớ (khi CPU thực hiện đọc dữ liệu) hoặc gửi dữ liệu tới bộ nhớ (khi CPU thực hiện ghi dữ liệu).
* **Vai trò:**
  + Giữ dữ liệu giữa bộ nhớ và các thành phần khác của CPU.
  + Trong chu kỳ truy xuất bộ nhớ, MBR giúp di chuyển dữ liệu giữa bộ nhớ và CPU.

**5. Flag Register (FlagR) / Status Register (SR):**

* **Chức năng:**
  + Lưu trữ các cờ trạng thái (flags) phản ánh kết quả của các phép toán mà ALU thực hiện.
  + Các cờ này có thể bao gồm: **Zero flag**, **Carry flag**, **Overflow flag**, **Sign flag**, **Parity flag**, v.v.
* **Vai trò:**
  + Cung cấp thông tin về trạng thái của các phép toán và so sánh trong ALU.
  + Dùng để ra quyết định trong các lệnh điều khiển, như nhảy (branch) dựa trên kết quả của phép toán.
  + Ví dụ: Nếu kết quả phép toán bằng 0, cờ **Zero flag** sẽ được thiết lập, giúp xác định việc nhảy hoặc tiếp tục chương trình.

**Tóm tắt vai trò các thanh ghi:**

| **Thanh ghi** | **Vai trò** |
| --- | --- |
| **Program Counter (PC)** | Lưu địa chỉ của lệnh tiếp theo cần thực thi. |
| **Instruction Register (IR)** | Lưu trữ lệnh hiện tại được giải mã và thực thi. |
| **Memory Address Register (MAR)** | Lưu địa chỉ bộ nhớ cần truy xuất. |
| **Memory Buffer Register (MBR)** | Lưu trữ dữ liệu được đọc hoặc chuẩn bị ghi vào bộ nhớ. |
| **Flag Register (FlagR)** | Lưu cờ trạng thái (flags) phản ánh kết quả phép toán và so sánh. |

**Mối quan hệ giữa các thanh ghi:**

1. **PC** trỏ đến địa chỉ lệnh tiếp theo trong bộ nhớ.
2. **MAR** chứa địa chỉ bộ nhớ mà CPU muốn truy xuất.
3. **IR** lưu trữ lệnh được lấy từ bộ nhớ.
4. **MBR** lưu trữ dữ liệu được truy xuất từ bộ nhớ hoặc chuẩn bị ghi vào bộ nhớ.
5. **Flag Register** giữ trạng thái của các phép toán trong ALU, giúp quyết định hành động tiếp theo.

Các thanh ghi này phối hợp với nhau trong chu kỳ thực thi lệnh để đảm bảo hoạt động chính xác và đồng bộ của CPU.

**Vai trò các cờ trong thanh ghi cờ**

| **Cờ** | **Chức năng** |
| --- | --- |

|  |  |
| --- | --- |
| **ZF** | Được đặt khi kết quả phép toán bằng 0 (Zero). |

|  |  |
| --- | --- |
| **SF** | Được đặt khi kết quả phép toán có dấu âm (Sign). |

|  |  |
| --- | --- |
| **CF** | Được đặt khi có sự chuyển bit (Carry) trong phép toán cộng hoặc mượn bit trong phép toán trừ. |

|  |  |
| --- | --- |
| **AF** | Được đặt khi có sự chuyển bit giữa các nibbles trong phép toán số học (Auxiliary Carry). |

|  |  |
| --- | --- |
| **OF** | Được đặt khi có sự tràn số học (Overflow) trong phép toán số nguyên có dấu. |

|  |  |
| --- | --- |
| **PF** | Được đặt khi số lượng bit 1 trong kết quả phép toán là chẵn (Parity). |

|  |  |
| --- | --- |
| **DF** | Điều khiển hướng của các lệnh chuỗi (Direction). Đặt hoặc xóa để di chuyển từ địa chỉ cao/thấp. |

**Trap Flag (TF)**: Cho phép chế độ gỡ lỗi (debugging mode), khi được bật, CPU sẽ tạm dừng sau mỗi lệnh thực thi để cho phép kiểm tra.

**Interrupt Flag (IF)**: Cho phép hoặc không cho phép các ngắt.

**Quá trình thi hành lệnh trong vi xử lý (không có ngắt)**

1. **Nạp lệnh (Fetch):**
   * Bộ xử lý nạp mã lệnh từ bộ nhớ chính (RAM) vào thanh ghi lệnh (Instruction Register - IR) thông qua bộ đếm chương trình (Program Counter - PC).
   * PC được tự động tăng lên để trỏ tới địa chỉ của lệnh tiếp theo.
2. **Giải mã lệnh (Decode):**
   * Bộ giải mã lệnh phân tích mã lệnh trong IR để xác định loại lệnh (ví dụ: tính toán, di chuyển dữ liệu, nhảy, v.v.) và các toán hạng liên quan.
3. **Đọc toán hạng (Operand Fetch):**
   * Các toán hạng (nếu có) được nạp từ bộ nhớ hoặc thanh ghi vào các thanh ghi nội bộ của CPU.
4. **Thi hành lệnh (Execute):**
   * Bộ xử lý thực hiện lệnh, chẳng hạn như thực hiện phép toán, di chuyển dữ liệu, hoặc thay đổi giá trị của thanh ghi hoặc bộ đếm chương trình.
5. **Lưu kết quả (Write Back):**
   * Kết quả được lưu trữ vào thanh ghi hoặc bộ nhớ.
6. **Cập nhật PC:**
   * Sau khi hoàn thành lệnh, PC được cập nhật để trỏ đến lệnh tiếp theo.

**Có ngắt (With Interrupt):**

1. **Nhận tín hiệu ngắt (Interrupt Request - IRQ):** CPU nhận tín hiệu từ thiết bị ngoại vi.
2. **Lưu ngữ cảnh (Context Saving):** Lưu trạng thái chương trình vào Stack.
3. **Xử lý ngắt (Interrupt Service Routine - ISR):** Thực hiện chương trình xử lý ngắt.
4. **Khôi phục ngữ cảnh (Context Restoring):** Phục hồi trạng thái và tiếp tục chương trình chính.

**Quá trình tiến hóa của máy tính, thế hệ 0,1,2,3,4**

**Thế hệ 0 (Trước năm 1940) - Máy tính cơ học**

* **Đặc điểm chính**: Sử dụng các thiết bị cơ học để thực hiện tính toán.
* **Công nghệ**:
  + Bánh răng, đòn bẩy và bánh xe.
  + Máy tính Pascaline (1642) và máy tính cơ học của Charles Babbage (1822).
* **Hạn chế**: Tốc độ chậm, cồng kềnh và không linh hoạt.

**Thế hệ 1 (1940-1956) - Đèn chân không**

* **Đặc điểm chính**: Sử dụng đèn chân không (vacuum tubes) để xử lý dữ liệu.
* **Công nghệ**:
  + Máy ENIAC (1945), UNIVAC (1951).
  + Sử dụng ngôn ngữ máy (machine language).
  + Bộ nhớ bằng trống từ (magnetic drum).
* **Hạn chế**:
  + Kích thước lớn, tiêu tốn nhiều năng lượng.
  + Sinh nhiệt cao, dễ hỏng hóc.

**Thế hệ 2 (1956-1963) - Transistor**

* **Đặc điểm chính**: Thay thế đèn chân không bằng transistor.
* **Công nghệ**:
  + Sử dụng ngôn ngữ lập trình cấp cao như COBOL, FORTRAN.
  + Bộ nhớ từ tính (magnetic core memory).
* **Ưu điểm**:
  + Nhỏ gọn hơn, tiêu thụ ít năng lượng hơn.
  + Độ tin cậy cao hơn.
* **Hạn chế**: Vẫn sinh nhiệt và tốc độ xử lý chưa cao.

**Thế hệ 3 (1964-1971) - Mạch tích hợp (IC)**

* **Đặc điểm chính**: Sử dụng mạch tích hợp (Integrated Circuits - IC).
* **Công nghệ**:
  + IC chứa hàng trăm transistor trên một chip.
  + Bắt đầu xuất hiện hệ điều hành và giao diện người dùng.
  + Hỗ trợ đa nhiệm.
* **Ưu điểm**:
  + Nhanh hơn, hiệu quả hơn và nhỏ gọn hơn so với thế hệ trước.
  + Chi phí giảm dần.
* **Hạn chế**: Công nghệ chế tạo IC vẫn đang phát triển.

**Thế hệ 4 (1971-nay) - Vi xử lý (Microprocessor)**

* **Đặc điểm chính**: Sử dụng vi xử lý, tích hợp hàng triệu transistor trên một chip.
* **Công nghệ**:
  + Máy tính cá nhân (PC) xuất hiện (IBM PC, Apple Macintosh).
  + Sử dụng bộ nhớ bán dẫn (RAM, ROM).
  + Giao diện đồ họa (GUI), internet và mạng toàn cầu phát triển.
* **Ưu điểm**:
  + Tốc độ xử lý nhanh, kích thước nhỏ gọn.
  + Dễ sử dụng, phổ biến rộng rãi.
* **Hạn chế**: Phụ thuộc vào năng lượng và có nguy cơ bảo mật.

**Định luật more**

**“Số lượng bóng bán dẫn trên mỗi vi mạch sẽ tăng gấp đôi sau mỗi 18-24 tháng, trong khi chi phí cho mỗi bóng bán dẫn sẽ giảm đi.”**

**Ý nghĩa chính của Định luật Moore**

1. **Tăng hiệu suất**:
   * Sự gia tăng số lượng bóng bán dẫn cho phép vi xử lý xử lý dữ liệu nhanh hơn, giúp máy tính trở nên mạnh mẽ hơn.
2. **Giảm chi phí**:
   * Nhờ tiến bộ trong công nghệ, sản xuất vi mạch trở nên hiệu quả hơn, làm giảm chi phí trên mỗi đơn vị.
3. **Thu nhỏ kích thước**:
   * Các vi mạch ngày càng nhỏ gọn, góp phần tạo ra các thiết bị như smartphone, laptop siêu mỏng, và các thiết bị IoT (Internet of Things).
4. **Đẩy mạnh sáng tạo**:
   * Định luật này là kim chỉ nam cho ngành công nghiệp bán dẫn, thúc đẩy các công ty không ngừng cải tiến để đạt được tốc độ phát triển như dự đoán.

**Hạn chế và thách thức**

* **Giới hạn vật lý**: Khi kích thước bóng bán dẫn ngày càng nhỏ, hiện tượng rò rỉ dòng điện và các vấn đề về nhiệt độ trở thành thách thức lớn.
* **Chi phí phát triển cao**: Các công nghệ mới như EUV (Extreme Ultraviolet Lithography) để thu nhỏ bóng bán dẫn đòi hỏi đầu tư lớn.
* **Chậm lại trong những năm gần đây**: Định luật Moore không còn giữ được tốc độ như trước, khi thời gian tăng gấp đôi đã kéo dài hơn 24 tháng.

**Công thức tính toán**

**1.1. Hiệu suất máy tính:**

Hiệu suất của máy tính được đánh giá dựa trên thời gian thực hiện chương trình, tốc độ xử lý lệnh và khả năng xử lý các tác vụ.

* **Clock Cycle (Chu kỳ xung):**
  + Một chu kỳ xung (T) là thời gian mà CPU thực hiện một bước nhỏ trong việc xử lý lệnh.
  + **T = 1 / f**, với f*f* là tần số xung nhịp (Hz).
  + Đơn vị thường dùng: giây (s), micro giây (µs), nano giây (ns).
  + A black screen with white text

    Description automatically generated

**CPI (Cycles Per Instruction):**

* Số chu kỳ xung cần để thực hiện một lệnh.
* A black background with white text

  Description automatically generated

A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

**Các kỹ thuật tăng tốc máy tính**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a black and white screen

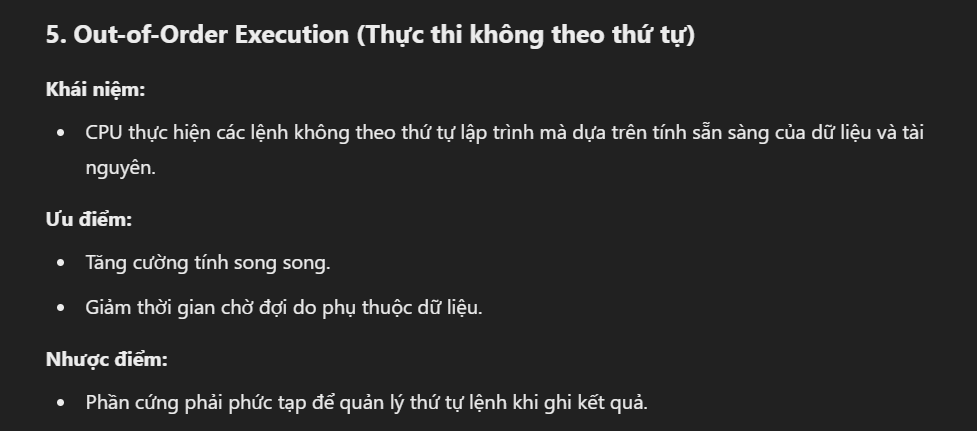
Description automatically generated**

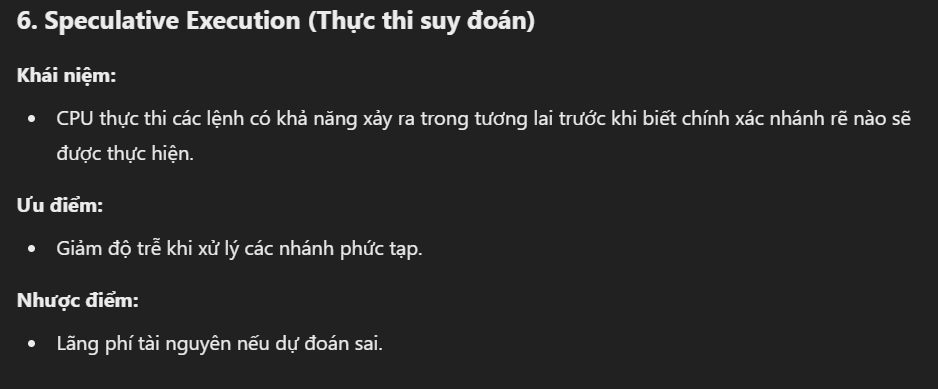
**A screenshot of a computer

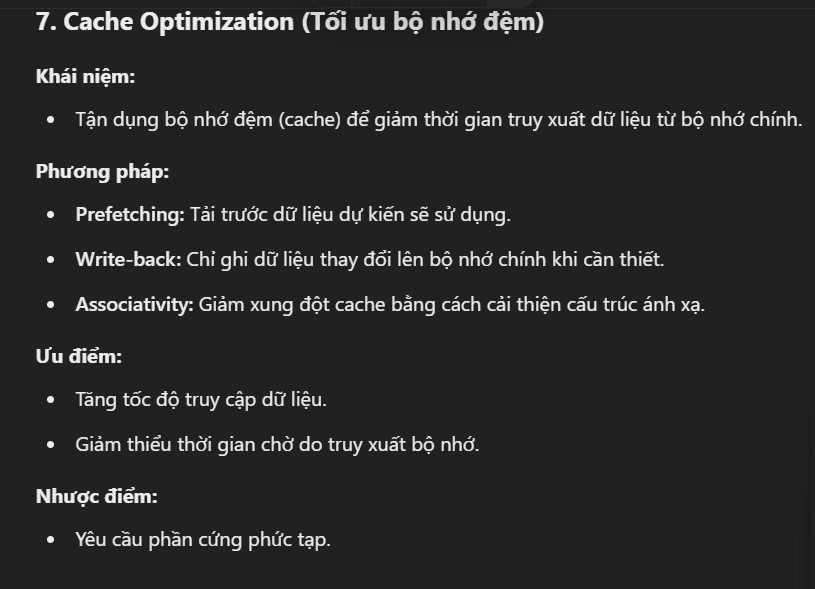
Description automatically generated**

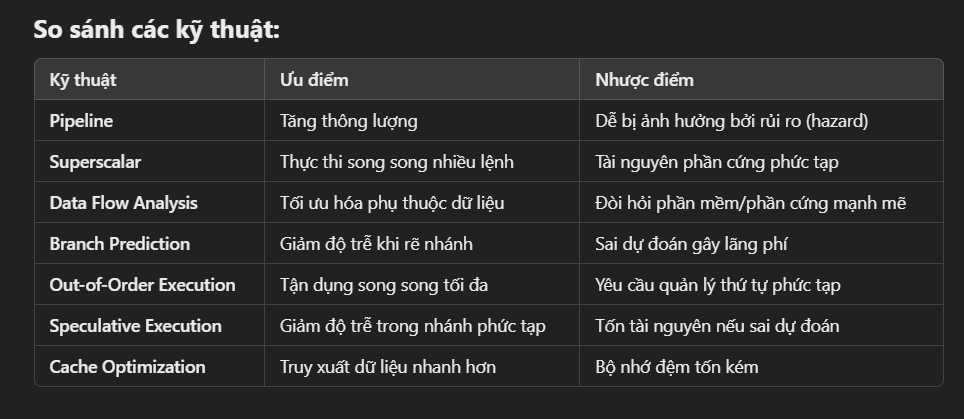
**A screenshot of a computer

Description automatically generated**

****

****





**Khái niệm về ngắt: là gì ? các nguyên nhân gây ra ngát, bảng vector ngắt, ưu điểm của ngắt, cơ chế ngắ,...**

A black screen with white text

Description automatically generated

A screenshot of a black screen

Description automatically generated

**. Thiết bị I0: các phương pháp điều khiển 10, các phương pháp đánh địa chỉ IO**

A screenshot of a black screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**Các thông tin liên quan tới bus: là gì, các loại bus, các thiết kế liên quan tới bus, phân cấp**

**bus, bus dong bo va khong dong bo**

A black background with white text

Description automatically generated

A black screen with white text

Description automatically generated

A screenshot of a black screen

Description automatically generated

MultipleBus hierarchies

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

BỘ nHỚA black table with white text

Description automatically generated

**Cac phuong phap truy cap: tuan ty, ngau nhien, associate**A black screen with white text

Description automatically generated

17. Phân cấp bộ nhớ theo mô hình kim tự tháp: thanh ghi, cache, bộ nhớ trong, đĩa,..., tương quan: giá thành, tốc độ, dung lượng khi đi trên xuống hoặc dưới lên

Mô hình kim tự tháp trong phân cấp bộ nhớ thể hiện sự phân chia bộ nhớ của hệ thống máy tính theo các cấp độ với sự tương quan về **giá thành**, **tốc độ** và **dung lượng** như sau:

1. **Thanh ghi (Registers)**:
   * **Tốc độ**: Nhanh nhất, vì đây là bộ nhớ trực tiếp trong CPU.
   * **Dung lượng**: Rất nhỏ (thường chỉ vài kilobyte hoặc ít hơn).
   * **Giá thành**: Rất đắt, vì phải tích hợp trực tiếp vào vi xử lý.
2. **Cache**:
   * **Tốc độ**: Rất nhanh, nhưng chậm hơn thanh ghi một chút.
   * **Dung lượng**: Nhỏ hơn bộ nhớ trong, thường từ vài kilobyte đến vài megabyte.
   * **Giá thành**: Cao hơn bộ nhớ trong, vì được thiết kế để đạt hiệu suất cao.
3. **Bộ nhớ trong (Main Memory / RAM)**:
   * **Tốc độ**: Chậm hơn cache, nhưng nhanh hơn bộ nhớ ngoài.
   * **Dung lượng**: Lớn hơn nhiều so với cache, từ vài gigabyte đến vài terabyte tùy theo hệ thống.
   * **Giá thành**: Rẻ hơn cache, nhưng đắt hơn bộ nhớ ngoài (đĩa cứng).
4. **Bộ nhớ ngoài (Disk, HDD, SSD)**:
   * **Tốc độ**: Chậm nhất trong các cấp bộ nhớ.
   * **Dung lượng**: Lớn nhất, từ vài terabyte đến hàng chục terabyte.
   * **Giá thành**: Rẻ nhất so với các cấp bộ nhớ khác.

**Tương quan khi đi từ trên xuống dưới:**

* **Giá thành**: Khi đi xuống từ thanh ghi đến đĩa, giá thành giảm dần.
* **Tốc độ**: Tốc độ giảm dần từ thanh ghi (nhanh nhất) đến đĩa (chậm nhất).
* **Dung lượng**: Dung lượng tăng dần từ thanh ghi (nhỏ nhất) đến đĩa (lớn nhất).

**Tương quan khi đi từ dưới lên trên:**

* **Giá thành**: Khi đi lên từ đĩa đến thanh ghi, giá thành tăng dần.
* **Tốc độ**: Tốc độ tăng dần từ đĩa (chậm nhất) đến thanh ghi (nhanh nhất).
* **Dung lượng**: Dung lượng giảm dần từ đĩa (lớn nhất) đến thanh ghi (nhỏ nhất).

A screenshot of a black screen

Description automatically generated

Bộ nhớ cache:

a. Khái niệm, vị trí của cache

b. Cache hit, cache miss

c. Nguyên lý cục bộ - không gian, thời gian

d. Các thông số liên quan tới thiết kế cache: dung lượng, chức năng ánh xạ, set, line,

18

block..

e. Các phương pháp ánh xạ: Direct mapping, fully associate mapping, set associate mapping

**a. Khái niệm và vị trí của cache**

* **Khái niệm**: Bộ nhớ **cache** là một bộ nhớ tốc độ cao được đặt gần hoặc bên trong vi xử lý (CPU) dùng để lưu trữ các dữ liệu và lệnh thường xuyên được truy cập, giúp giảm độ trễ khi truy xuất dữ liệu từ bộ nhớ chính (RAM).
* **Vị trí**: Cache thường nằm giữa CPU và bộ nhớ chính. Nó có thể được chia thành nhiều cấp (L1, L2, L3), với L1 là gần nhất với CPU và nhỏ nhất, L3 ở xa hơn nhưng có dung lượng lớn hơn.

**b. Cache hit, Cache miss**

* **Cache hit**: Khi dữ liệu yêu cầu đã có sẵn trong bộ nhớ cache, việc truy xuất dữ liệu từ cache được gọi là **cache hit**. Điều này giúp tăng tốc độ xử lý.
* **Cache miss**: Khi dữ liệu yêu cầu không có trong bộ nhớ cache, hệ thống phải truy xuất dữ liệu từ bộ nhớ chính, điều này làm giảm hiệu suất và gọi là **cache miss**.

**c. Nguyên lý cục bộ - không gian, thời gian**

* **Nguyên lý cục bộ về thời gian**: Nếu một phần dữ liệu được truy cập, khả năng cao nó sẽ được truy cập lại trong thời gian ngắn, do đó nó sẽ được giữ trong bộ nhớ cache.
* **Nguyên lý cục bộ về không gian**: Nếu một phần dữ liệu trong bộ nhớ được truy cập, khả năng cao các phần dữ liệu liền kề (gần nhau về mặt bộ nhớ) cũng sẽ được truy cập trong tương lai gần, và do đó sẽ được lưu vào bộ nhớ cache.

**d. Các thông số liên quan tới thiết kế cache**

* **Dung lượng (Capacity)**: Là tổng không gian lưu trữ của bộ nhớ cache, thường được đo bằng kilobyte (KB) hoặc megabyte (MB).
* **Chức năng ánh xạ (Mapping Function)**: Là cách thức quyết định vị trí lưu trữ của dữ liệu trong bộ nhớ cache, giúp liên kết giữa bộ nhớ chính và cache.
* **Set**: Bộ nhớ cache thường được chia thành nhiều nhóm gọi là **sets**. Mỗi set chứa một số lượng cache lines.
* **Cache line**: Là đơn vị nhỏ nhất của bộ nhớ cache, mỗi dòng có thể chứa một khối dữ liệu từ bộ nhớ chính.
* **Block**: Dữ liệu từ bộ nhớ chính được tải vào cache theo khối, mỗi khối chứa một phần dữ liệu và có thể chứa nhiều dòng (cache line).

**e. Các phương pháp ánh xạ**

1. **Direct Mapping**:
   * Mỗi vị trí trong bộ nhớ chính được ánh xạ vào đúng một vị trí trong bộ nhớ cache.
   * **Ưu điểm**: Đơn giản, dễ triển khai.
   * **Nhược điểm**: Dễ bị **cache miss** nếu nhiều dữ liệu trong bộ nhớ chính phải chia sẻ cùng một vị trí trong cache.
2. **Fully Associative Mapping**:
   * Mỗi vị trí trong bộ nhớ chính có thể được ánh xạ vào bất kỳ vị trí nào trong bộ nhớ cache.
   * **Ưu điểm**: Giảm thiểu khả năng **cache miss**.
   * **Nhược điểm**: Tốn chi phí phần cứng và phức tạp trong việc tìm kiếm và lưu trữ dữ liệu.
3. **Set-Associative Mapping**:
   * Là sự kết hợp của hai phương pháp trên. Cache được chia thành nhiều sets, và mỗi block trong bộ nhớ chính có thể được ánh xạ vào bất kỳ vị trí nào trong một set cụ thể.
   * **Ưu điểm**: Cân bằng giữa chi phí phần cứng và hiệu suất, giúp giảm **cache miss** so với Direct Mapping.
   * **Nhược điểm**: Phức tạp hơn trong việc quản lý.

**Các loại ROM**

A black and white screen with white text

Description automatically generated

20. Bộ nhớ RAM

a. Phân biệt SRAM, DRAM

A black and white text on a black background

Description automatically generated

b. Cấu trúc DRAM

A screenshot of a computer

Description automatically generated

c. Tính toán dung lượng, số đường địa chỉ, dữ liệu, lưu ý khi có tín hiệu CAS, RAS

A screenshot of a computer

Description automatically generated

A black and white text

Description automatically generated

21. Bộ nhớ ngoài (các loại đĩa)

a. Cấu trúc của đĩa cứng HDD: C, H, S

A black screen with white text

Description automatically generated

b. Các tham số: seek time, rotational time, transfer time

A black and white text

Description automatically generated

c. Thời gian truy cập

A black background with white text

Description automatically generated

d. Nguyên tắc làm việc​

A black screen with white text

Description automatically generated

22. Bộ nhớ ảo

a. Vai trò của bộ nhớ ảo

A screenshot of a black and white text

Description automatically generated

b. Kỹ thuật phân trang, phân đoạn

A black and white text on a black background

Description automatically generated

c. Cấu trúc của bảng trang, bảng đoạn

A screenshot of a computer

Description automatically generated

I

d. Cấu trúc địa chỉ ảo, chuyển từ địa chỉ ảo sang địa chỉ vật lý và ngược lại chuyển từ địa chỉ vật lý sang địa chỉ ảo.​

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

23. CPU: loại (bao nhiêu bit), kích thước các thanh ghi, tên, vai trò các thanh ghi, kích thước bus địa chỉ, bus dữ liệu, kích thước bộ nhớ có thể quản lý, cache L1, L2

A black screen with white text

Description automatically generated



A black and white text

Description automatically generated

24. Các chế độ định địa chỉ: chế độ tức thời, định vị thanh ghi, bộ nhớ, cách tính địa chỉ toán hạng trong chế độ định vị bộ nhớ

A black and white table with white text

Description automatically generated

**Giải thích chi tiết các chế độ định địa chỉ:**

1. **Chế độ tức thời (Immediate Addressing)**:
   * Trong chế độ này, toán hạng không được lưu trữ ở đâu đó trong bộ nhớ mà là một giá trị trực tiếp nằm trong chính lệnh. Ví dụ, lệnh MOV R1, #5 sẽ di chuyển giá trị 5 vào thanh ghi R1.
   * **Cách tính địa chỉ toán hạng**: Địa chỉ toán hạng chính là giá trị ngay trong lệnh (không có phép toán nào cần thực hiện).
2. **Định vị thanh ghi (Register Addressing)**:
   * Toán hạng được lưu trong một thanh ghi của CPU. Khi CPU thực hiện lệnh, toán hạng sẽ được lấy từ một trong các thanh ghi.
   * **Cách tính địa chỉ toán hạng**: Địa chỉ toán hạng là tên của thanh ghi chứa giá trị (ví dụ: MOV R1, R2 di chuyển giá trị trong R2 vào R1).
3. **Định vị bộ nhớ (Memory Addressing)**:
   * Toán hạng có thể được lưu trữ ở một địa chỉ bộ nhớ, và CPU cần truy xuất bộ nhớ để lấy toán hạng. Có nhiều cách tính địa chỉ toán hạng trong bộ nhớ:

a. **Chế độ chỉ mục (Indexed Addressing)**:

* + Địa chỉ toán hạng được tính bằng cách cộng địa chỉ của thanh ghi chỉ mục với một giá trị offset (hằng số).
  + **Ví dụ**: MOV AX, [BX + 5] có nghĩa là lấy giá trị từ bộ nhớ tại địa chỉ BX + 5 và di chuyển nó vào thanh ghi AX.

b. **Chế độ gián tiếp (Indirect Addressing)**:

* + Địa chỉ của toán hạng được lưu trữ trong một thanh ghi hoặc bộ nhớ khác. CPU sử dụng địa chỉ này để lấy toán hạng.
  + **Ví dụ**: MOV AX, [BX] có nghĩa là lấy giá trị từ địa chỉ bộ nhớ tại vị trí mà thanh ghi BX chỉ định.

c. **Chế độ dựa trên cơ sở (Base-Register Addressing)**:

* + Địa chỉ toán hạng được tính bằng cách cộng giá trị của thanh ghi cơ sở với một offset.
  + **Ví dụ**: MOV AX, [SI + 100] có nghĩa là lấy giá trị từ địa chỉ tại SI + 100 và di chuyển vào AX.

d. **Chế độ gián tiếp với chỉ mục (Indexed Indirect Addressing)**:

* + Cách tính này yêu cầu CPU lấy địa chỉ từ một thanh ghi chỉ mục, sau đó địa chỉ này lại được sử dụng để lấy toán hạng từ bộ nhớ.
  + **Ví dụ**: MOV AX, [[BX + 100]] có nghĩa là lấy giá trị từ địa chỉ bộ nhớ BX + 100, nơi địa chỉ đó chứa một địa chỉ khác, và lấy giá trị tại địa chỉ đó.

25. Biểu diễn số: các hệ thống số quan trọng (hệ nhị phân, bát phân, thập phân, thập lục phân), các phương pháp tính toán chuyển đổi cơ số​