**Câu 1: So sánh kiến trúc RISC và CISC**

· **RISC** thích hợp cho các ứng dụng yêu cầu hiệu suất cao, tiêu thụ năng lượng thấp, và thiết kế đơn giản hơn.

· **CISC** mạnh hơn trong các ứng dụng cần sử dụng nhiều lệnh phức tạp, chẳng hạn như máy tính để bàn, máy chủ và các ứng dụng công nghiệp.

· **RISC**: ARM, MIPS, PowerPC.

· **CISC**: Intel x86, AMD.

**Câu 2: Trình bày vấn đề mã hoá tập lệnh và các tập lệnh Thumb 16bit, tập lệnh 32**

**Mã hóa tập lệnh** (Instruction encoding) là quá trình biến đổi các lệnh trong ngôn ngữ lập trình cấp cao hoặc hợp ngữ (assembly) thành một chuỗi các mã nhị phân mà vi xử lý có thể hiểu và thực thi trực tiếp

Mã hóa tập lệnh là một yếu tố quan trọng trong thiết kế vi xử lý, vì nó ảnh hưởng đến cách mà các lệnh của bộ xử lý được biểu diễn và lưu trữ trong bộ nhớ. Việc mã hóa tập lệnh liên quan đến độ dài của lệnh, khả năng truy xuất bộ nhớ, và hiệu quả thực thi. Có hai cách tiếp cận chính trong mã hóa tập lệnh:

* **Mã hóa độ dài cố định** (Fixed-length encoding): Tất cả các lệnh đều có độ dài giống nhau. Đây là cách tiếp cận thường thấy trong kiến trúc RISC. Ưu điểm của mã hóa này là dễ dàng giải mã và đơn giản hóa thiết kế bộ xử lý, nhưng có thể gây lãng phí bộ nhớ do các lệnh đơn giản cũng chiếm một không gian cố định lớn.
* **Mã hóa độ dài biến đổi** (Variable-length encoding): Các lệnh có thể có độ dài khác nhau, với các lệnh đơn giản có thể ngắn và các lệnh phức tạp thì dài hơn. Kiến trúc CISC thường sử dụng phương pháp này. Nó giúp tiết kiệm bộ nhớ nhưng làm phức tạp quá trình giải mã.

**Thumb 16 bit:** Thumb là một dạng rút gọn của tập lệnh ARM được mã hóa bằng các lệnh có độ dài 16-bit, thay vì 32-bit như ARM. Tập lệnh Thumb giúp tăng hiệu quả sử dụng bộ nhớ mà vẫn duy trì phần lớn hiệu năng của ARM.

**Tập lệnh 32:** ARM 32-bit là tập lệnh chính của kiến trúc ARM, trong đó mỗi lệnh có độ dài 32-bit. Nó cung cấp nhiều tính năng mạnh mẽ hơn so với tập lệnh Thumb 16-bit.

· **Tập lệnh ARM 32-bit** phù hợp với các ứng dụng yêu cầu tính toán mạnh mẽ và độ phức tạp cao, nhưng chiếm nhiều bộ nhớ hơn.

· **Tập lệnh Thumb 16-bit** hiệu quả hơn về sử dụng bộ nhớ, nhưng có khả năng hạn chế hơn so với ARM 32-bit.

**Câu 3: Trình bày về tập lệnh máy? Tại sao mỗi Cpu lại có tập lệnh máy khác nhau**

Tập lệnh máy (Machine Instruction Set) là tập hợp các lệnh mà một bộ xử lý (CPU) có thể hiểu và thực thi trực tiếp. Đây là ngôn ngữ cấp thấp nhất của máy tính, bao gồm các chỉ thị nhị phân (binary) hoặc mã máy (machine code), được sử dụng để điều khiển hoạt động của CPU.

Mỗi lệnh trong tập lệnh máy tương ứng với một nhiệm vụ cụ thể mà CPU có thể thực hiện, như đọc/ghi dữ liệu từ bộ nhớ, thực hiện các phép toán số học và logic, hoặc điều khiển dòng chảy chương trình (như nhảy điều kiện, gọi hàm, v.v.).

**mỗi CPU có tập lệnh máy khác nhau** vì sự khác biệt trong kiến trúc, mục tiêu thiết kế, lịch sử phát triển, và yêu cầu ứng dụng. Điều này giúp tối ưu hóa hiệu suất, năng lượng tiêu thụ, và khả năng xử lý của CPU trong từng trường hợp cụ thể.

**Câu 4: Trình bày nhóm lệnh truyền bên trong vi xử lý**

Nhóm lệnh truyền trong vi xử lý là nền tảng cho mọi hoạt động xử lý dữ liệu, cho phép di chuyển dữ liệu giữa các thành phần khác nhau của hệ thống, từ thanh ghi, bộ nhớ, đến thiết bị ngoại vi. Việc sử dụng hiệu quả các lệnh truyền giúp tăng cường hiệu năng của vi xử lý trong các tác vụ tính toán và điều khiển.

Truyền dữ liệu giữa các thanh ghi MOV R1,R2

Truyền dữ liệu giữa thanh ghi và bộ nhớ LDR R1, [address]

Truyền dữ liệu giữa CPU và các thiết bị ngoại vi IN AL, port

Truyền dữ liệu gián tiếp

Truyền dữ liệu tức thời (Immediate Data Transfer) MOV R1, #5

Truyền dữ liệu qua ngăn xếp (Stack) PUSH , POP

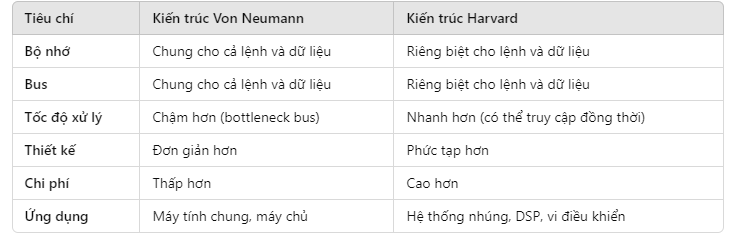
**Câu 5 : Trình bày bản chất địa chỉ vật lý, không gian địa chỉ vật lý, bộ giải mã địa chỉ**

· **Địa chỉ vật lý** là địa chỉ thực tế trong bộ nhớ mà CPU truy cập để lấy dữ liệu.

· **Không gian địa chỉ vật lý** là tập hợp tất cả các địa chỉ có thể truy cập trong bộ nhớ vật lý của hệ thống.

· **Bộ giải mã địa chỉ** là thành phần giải mã địa chỉ mà CPU đưa ra để xác định vùng bộ nhớ hoặc thiết bị ngoại vi cần truy cập.

**Câu 6 : so sánh kiến trúc von neumann và kiến trúc harvard**



Kết luận:

**Kiến trúc Von Neumann** phù hợp với các hệ thống không yêu cầu tốc độ cao nhưng cần thiết kế đơn giản và tiết kiệm chi phí.

**Kiến trúc Harvard** ưu việt hơn về hiệu suất và được sử dụng trong các hệ thống đòi hỏi khả năng xử lý nhanh và song song lệnh/dữ liệu, nhưng đi kèm với chi phí và độ phức tạp cao hơn.

**Câu 7: Trình bày về tập thanh ghi trong vi xử lý ARM và các cờ**

Tập thanh ghi trong vi xử lý ARM bao gồm các thanh ghi tổng quát (R0 - R12), thanh ghi ngăn xếp (SP), thanh ghi liên kết (LR), thanh ghi đếm chương trình (PC), và thanh ghi trạng thái chương trình (CPSR). Các cờ trạng thái như N, Z, C, V, I, F, và T được sử dụng để theo dõi kết quả của các phép toán số học và điều khiển hoạt động của CPU, giúp vi xử lý ARM có thể ra quyết định dựa trên trạng thái của các phép toán.