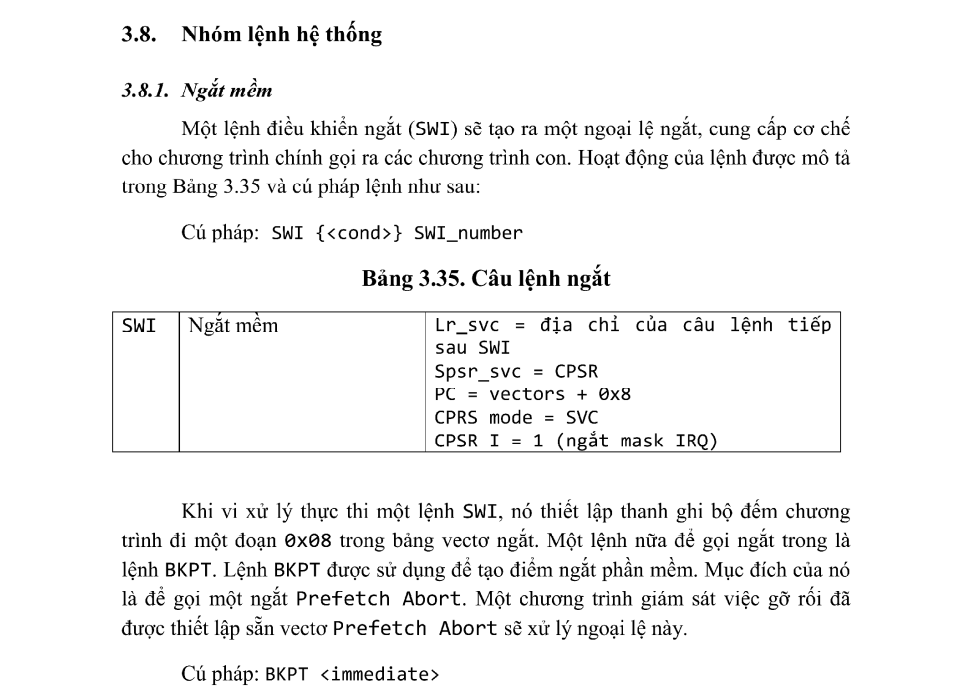
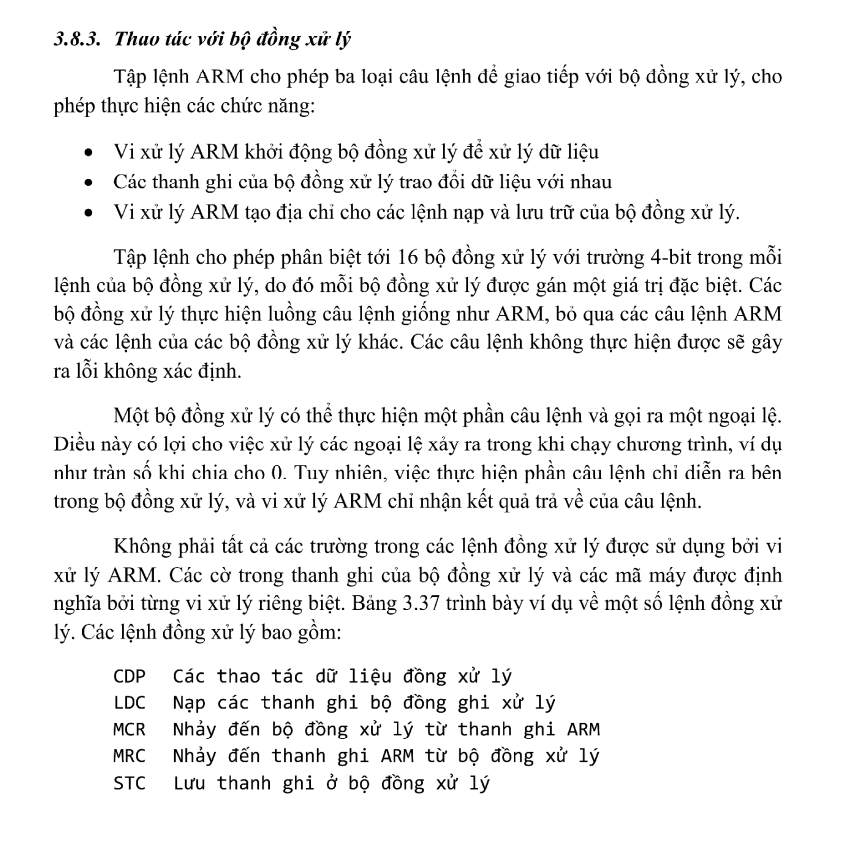
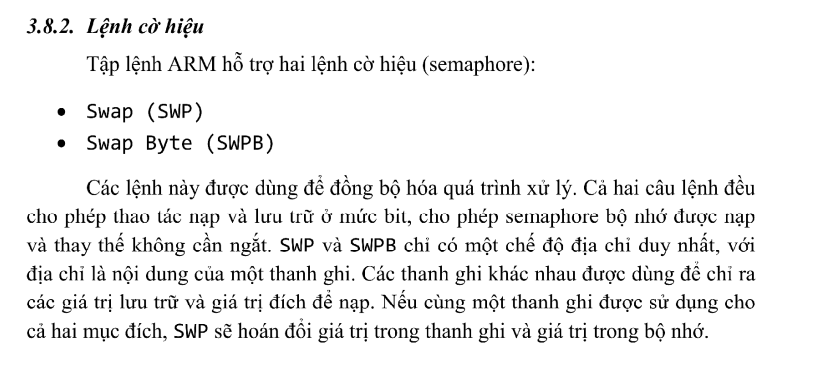
**1. Trình bày hoạt động và sử dụng các lệnh hệ thống**





**2. Phân biệt lệnh máy và mã phép toán, ánh xa giữa mã phép toán và các IC chức năng trong vi xử lí**

**Tức là phân biệt instruction và opcode**

* **Instruction** là một chỉ thị đầy đủ mà CPU có thể thực hiện, bao gồm cả opcode và operand.
* **Opcode** là một phần của instruction, xác định loại hành động mà CPU cần thực hiện
* **Ví dụ**: Trong lệnh ADD R1, R2, ADD là opcode

**- mã phép toán(opcode) là phần xác định hành động mà cpu cần thực hiện, mỗi mã phép toán ứng với 1 IC thực hiện chức năng ấy, ví dụ mã ADD ứng với IC thực hiện phép cộng. Mã phép toán ADD khi được dịch thành mã máy, mã máy dạng nhị phân sẽ được đưa vào một bộ giải mã decoder, bộ giải mã này có các chân liên kết với các IC thực hiện các phép toán. Khi decoder nhận vào mã nhị phân của phép cộng ADD, decoder kích hoạt chân IC thực hiện phép cộng**

**3. Trình bày bản chất địa chỉ vật lí, không gian địa chỉ vật lí, bộ giải mã địa chỉ**

* Địa chỉ vật lý = số thứ tự của byte = stt ô nhớ
* Bộ nhớ là một chuỗi các byte nhớ, số thứ tự của byte chính là địa chỉ vật lý

Không gian địa chỉ vật lý là phạm vi tất cả các địa chỉ mà bộ nhớ vật lý có thể hỗ trợ. Đây là dải địa chỉ từ 0 đến một giá trị tối đa, tương ứng với kích thước tối đa của bộ nhớ mà hệ thống có thể sử dụng.

Độ lớn của không gian địa chỉ vật lí phụ thuộc vào độ lớn của bus địa chỉ, ví dụ bus 64bit thì có thể đánh 2^64 địa chỉ để quản lí

Bộ giải mã địa chỉ là một thành phần phần cứng có chức năng xác định vị trí thực tế của dữ liệu hoặc lệnh trong bộ nhớ dựa trên địa chỉ mà CPU cung cấp.

**4. Trình bày về tập thanh ghi trong vi xử lí ARM và các cờ**

**- 16 thanh ghi R0 – R15**

**R0-R12: thanh ghi đa mục đích**

**R13: thanh ghi SP**

**R14: thanh ghi liên kết LR**

**R15: thanh ghi con trỏ lệnh IR**

**xPSR: thanh ghi trạng thái: chứa thông tin về trạng thái của CPU, chứa các cờ**

**N: cờ âm cho biết kết quả phép toán âm hay không,**

**Z: cờ zero chỉ ra kết quả phép toán = 0 hay không**

**C: cờ nhớ** chứa giá trị dư

**V: cờ check xem có bị tràn không**

**5. Trình bày các thanh ghi chuyên dụng như thanh ghi liên kết, con trỏ chương trình, con trỏ ngăn xếp**

- thanh ghi liên kết: LR – chứa địa chỉ trở về khi gọi 1 hàm hay một chương trình con, khi thực thi câu lệnh A nhảy sang chương trình con, địa chỉ của câu lệnh tiếp theo của A là A+1 được nạp vào thanh ghi LR, thực thi xong chương trình con thì nạp giá trị địa chỉ trong LR vào PC để nhảy đến câu lệnh A+1 để tiếp tục chương trình chính

- thanh ghi con trỏ chương trình: PC – chứa địa chỉ bộ nhớ thực hiện lệnh tiếp theo của chương trình

- thanh ghi con trỏ ngăn xếp: trỏ đến đỉnh ngăn xếp

**6. Mô tả quá trình thực hiện 1 lệnh bên trong vi xử lí**

Câu lệnh và dữ liệu đều là các chuỗi bit nhị phân được lưu trữ trong bộ nhớ nên vi xử lý chỉ có thể phân biệt được mã lệnh và dữ liệu thông qua địa chỉ của vùng nhớ chứa nó

Với họ vi xử lý 80x86, quá trình thực hiện lệnh gồm sáu giai đoạn là lấy lệnh, giải mã lệnh, tính địa chỉ toán hạng, lấy toán hạng, thực thi và lưu trữ

Với vi xử lý ARM, theo kiến trúc tập lệnh rút gọn RISC nên được phân chia đơn giản hơn, thành ba giai đoạn: lấy lệnh, thực thi và lưu kết

- lấy lệnh:

+ CPU phát địa chỉ câu lệnh ra bus địa chỉ

+ CPU phát tín hiệu đọc bộ nhớ

+ Lệnh đọc từ bộ nhớ được đặt lên bus dữ liệu và dc nạp vào thành ghi lệnh IR

+ CPU tăng nội dung thanh ghi PC để trỏ đến lệnh tiếp theo

- thực thi

+ giải mã lệnh để chọn dc mạch chức năng (IC) mà câu lệnh cần, rồi lấy toán hạng ra cho IC để thực thi blabla…

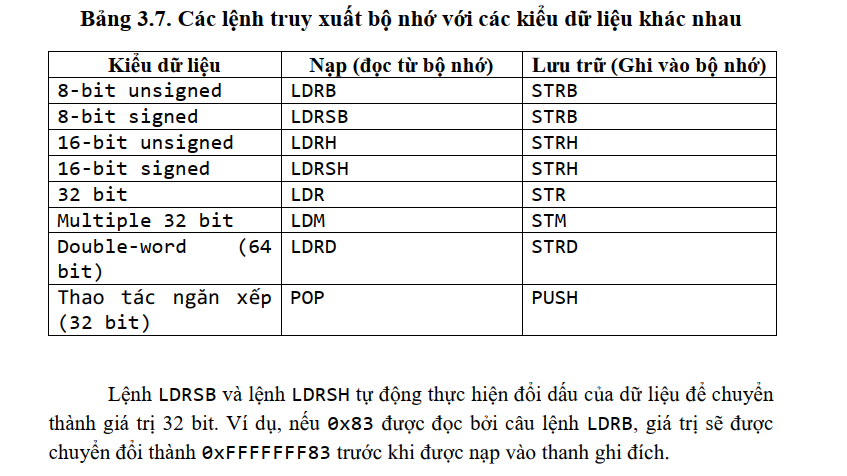
- Lưu kết quả

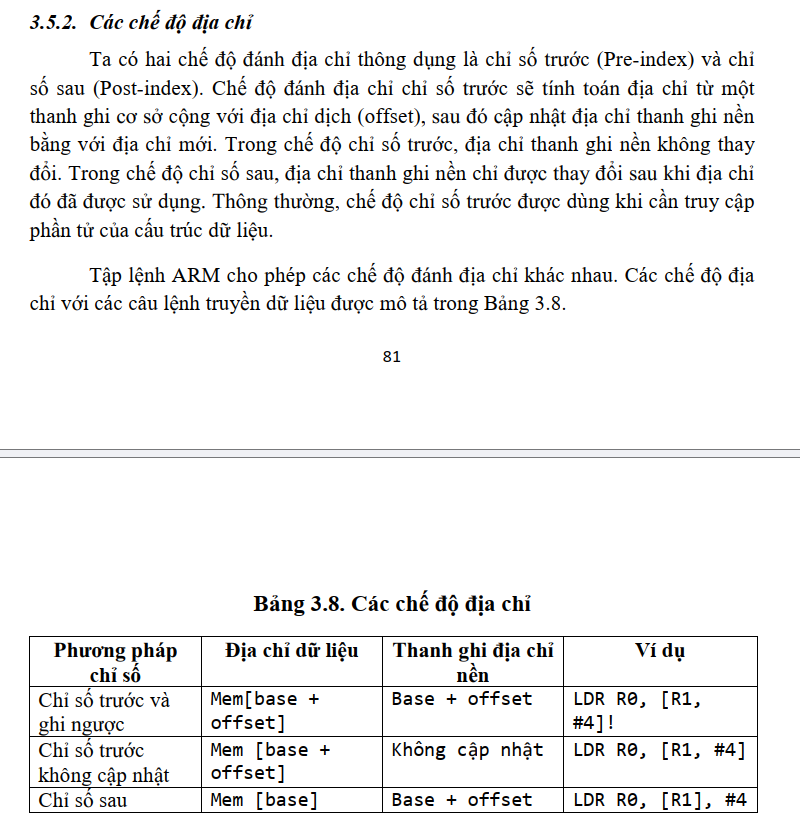
**7. Mô tả quá trình thực thi của một chương trình trong máy tính**

- Các chương trình trong ROM BIOS sẽ được thực hiện đầu tiên, và sẽ làm nhiệm vụ nạp nhân hệ điều hành, rồi chuyển điều khiển cho nhân hệ điều hành. Quá trình chuyển điều khiển chính là quá trình chuyển vi xử lý giữa các chương trình. Việc chuyển đổi này thực chất là thay đổi địa chỉ câu lệnh trong thanh ghi PC

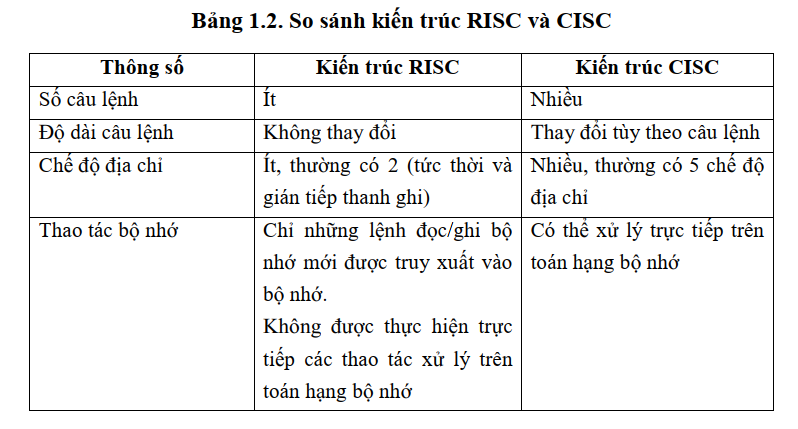
- đầu tiên, chương trình được lưu trữ lâu dài trong bộ nhớ ngoài; khi được thực hiện chương trình và dữ liệu sẽ được nạp vào trong bộ nhớ trong; trong quá trình thực hiện lần lượt từng câu lệnh của chương trình sẽ được nạp từ bộ nhớ vào vi xử lý để thực hiện. Câu lệnh đang thực hiện được chứa trong thanh ghi lệnh (IR) của vi xử lý

**8. Trình bày câu lệnh truy cập bộ nhớ, các chế độ địa chỉ**

****

****

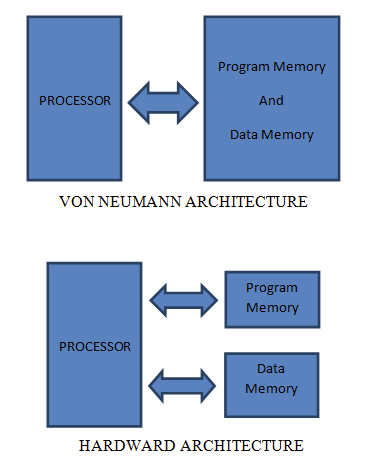
**9. So sánh RISC và CISC**

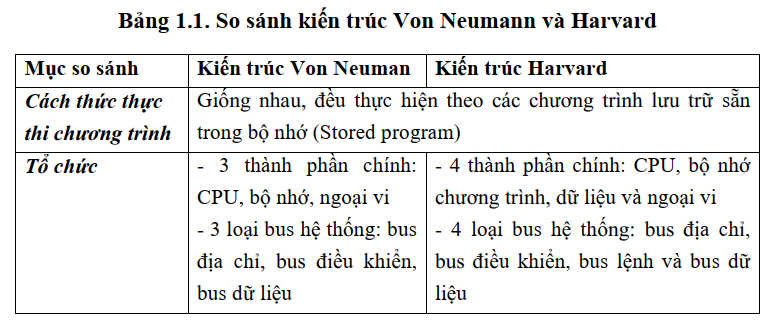
****

**Toán hạng bộ nhớ: các giá trị hoặc địa chỉ nằm trong bộ nhớ chính(RAM) mà các lệnh trong chương trình có thể truy cập và xử lí**

**Bao gồm: địa chỉ bộ nhớ (các vtri trong bộ nhớ nơi lưu trữ dữ liệu) + dữ liệu**

**10. So sánh kiến trúc Von Neuman và kiến trúc Havard**

****

****

Trong một máy tính sử dụng kiến trúc Von Neumann, CPU có thể đọc một lệnh, hoặc đọc/ghi dữ liệu từ bộ nhớ. Tuy nhiên, cả hai quá trình tương tác với lệnh hoặc dữ liệu không thể thực hiện cùng lúc vì nó sử dụng chung một đường truyền và bộ nhớ. Trong một máy tính kiến trúc Harvard, CPU có thể vừa đọc một lệnh, vừa truy cập dữ liệu từ bộ nhớ cùng lúc . trong kiến trúc Harvard cần phải thiết kế hai bus khác nhau cho dữ liệu và chương trình

Harvard có thể chạy nhanh hơn máy tính kiến trúc Von Neumann vì truy xuất 2 cái cùng lúc, không phải chia sẻ bus

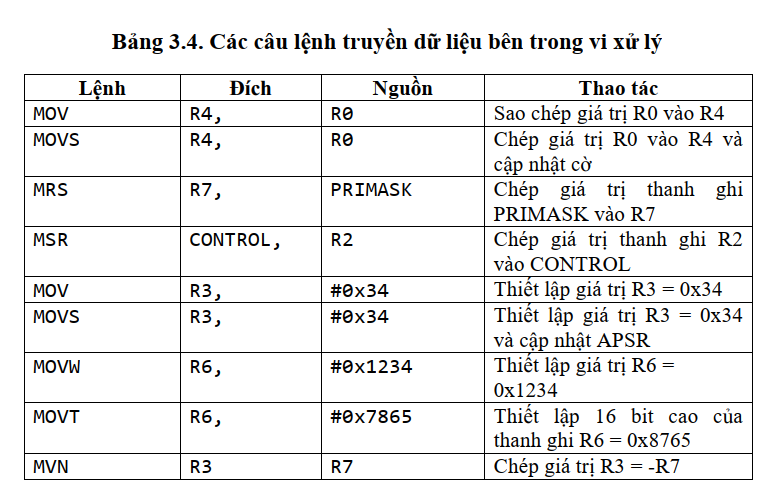
**11. Trình bày về tập lệnh máy, tại sao mỗi họ CPU lại có tập lệnh máy khác nhau**

Tập lệnh máy là tập hợp các chỉ thị mà một (CPU) có thể hiểu và thực thi trực tiếp. Mỗi chỉ thị thường là một chuỗi các bit và được thiết kế để thực hiện những thao tác cơ bản như tính toán số học cộng trừ nhân chia, di chuyển dữ liệu giữa các thanh ghi hoặc bộ nhớ, và điều khiển luồng chương trình (như các lệnh nhảy hoặc gọi hàm). Những lệnh này thường được viết dưới dạng mã nhị phân, nhưng trong quá trình lập trình, chúng ta sử dụng các ngôn ngữ assembly, là dạng ngôn ngữ cấp thấp hơn gần với tập lệnh máy.

**Vì kiến trúc mỗi họ CPU là khác nhau, kiến trúc CPU lại liên quan đến số lượng thanh ghi, cách quản lí bộ nhớ, cơ chế xử lí dữ liệu,…**

**Mỗi họ CPU sinh ra để phục vụ các tác vụ khác nhau, nên có tập lệnh khác nhau để phục vụ tốt tác vụ nó đảm nhiệm**

**12. Trình bày nhóm lệnh truyền dữ liệu bên trong vi xử lí**

****

**Các việc có thể thực hiện:**

* Truyền dữ liệu giữa hai thanh ghi đa năng với nhau
* Truyền dữ liệu giữa một thanh ghi đa năng và một thanh ghi đặc biệt (CONTROL, PRIMASK, BASEPRI, FAULTMASK)
* Truyền một giá trị tức thời vào thanh ghi.

**13. Trình bày vấn đề mã hóa tập lệnh và các tập lệnh Thumb 16bit, tập lệnh 32(bit)**

Mã hóa tập lệnh là quá trình biểu diễn các lệnh của CPU bằng các chuỗi bit (binary). Mỗi tập lệnh được mã hóa dưới dạng nhị phân nhằm tối ưu hóa việc xử lý và lưu trữ trong bộ nhớ

* **Tập lệnh Thumb 16-bit**: Đây là tập con của tập lệnh ARM và được thiết kế để có kích thước nhỏ hơn, nhằm tăng mật độ mã lệnh trong bộ nhớ. Các tập lệnh 16 bit này được sử dụng khi hệ thống yêu cầu tính hiệu quả cao về không gian bộ nhớ.

Ưu điểm của tập lệnh 16-bit:

* + Giảm kích thước mã chương trình, do đó tiết kiệm bộ nhớ.
  + Tăng cường hiệu suất trong các hệ thống nhúng, nơi tài nguyên bộ nhớ hạn chế.

Hạn chế:

* + Tập lệnh 16-bit ít phong phú hơn và không cung cấp nhiều tính năng như các lệnh 32-bit.
* **Tập lệnh Thumb 32-bit**: Để bổ sung cho tập lệnh 16-bit hạn chế, Thumb-2 (phiên bản cải tiến của Thumb) cho phép sử dụng các lệnh 32-bit kết hợp với các lệnh 16-bit, nhằm cung cấp tính linh hoạt và hiệu suất cao hơn. Điều này giúp tối ưu hóa cả về không gian lẫn hiệu năng.

Vi xử lý ARM có thể chuyển đổi linh hoạt giữa chế độ ARM và Thumb bằng cách sử dụng các lệnh đặc biệt như BX hoặc BLX

**Kết luận**

* **Tập lệnh Thumb 16-bit** thích hợp cho các ứng dụng đòi hỏi tiết kiệm bộ nhớ và tốc độ xử lý không quá cao.
* **Tập lệnh Thumb 32-bit** được bổ sung trong Thumb-2, cung cấp sự linh hoạt cao và khả năng kết hợp giữa các lệnh ngắn và dài.
* **Mã hóa tập lệnh** là yếu tố quan trọng ảnh hưởng đến kích thước mã chương trình và hiệu suất của hệ thống, đặc biệt là trong các ứng dụng nhúng hoặc yêu cầu tối ưu bộ nhớ.