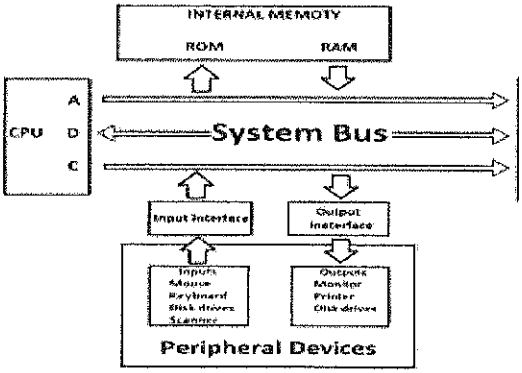
PHOTO HUYỀN TRANG 20 NGÕ 2 AO SEN

KIẾN TRÚC MÁY TÍNH – VXL 2023

1.1 Kiến trúc máy tính là gì ? Kiến trúc máy tính được cấu thành từ những thành phần nào ?

- Kiến trúc máy tính là khoa học về việc lựa chọn và kết nối các thành phần phần cứng để tạo ra các máy tính đạt được yêu cầu:
  - + Chức năng
  - + Hiệu năng
  - + Giá thành
- Kiến trúc máy tính được cấu thành từ 3 thành phần con:
  - + Kiến trúc tập lệnh: là hình ảnh của một hệ thống máy tính ở mức ngôn ngữ máy. Gồm: tập lệnh, các chế độ địa chỉ, các thanh ghi, khuôn dạng địa chỉ và dữ liệu.
  - + Vi kiến trúc: là mô tả mức thấp về các thành phần của hệ thống máy tính, phối ghép và việc trao đổi thông tin giữa chúng.
  - + Thiết kế hệ thống: bao gồm các thành phần p hần cứng của hệ thống máy tính.

1.2 Vẽ sơ đồ khối chức năng của hệ thống máy tính



1.3 Thanh ghi của vi xử lý là gì ? Nêu chức năng và đặc điểm của thanh ghi tích lũy A

- Thanh ghi là các ô nhớ bên trong CPU, có nhiệm vụ lưu trữ tạm thời lệnh và dữ liệu cho CPU xử lý. Thanh ghi thường có kích thước nhỏ nhưng tốc độ làm việc rất cao (ngang CPU). Số lượng thanh ghi phụ thuộc vào thế hệ của CPU, kích thước thanh ghi phụ thuộc vào thiết kế CPU.
- Thanh ghi tích lũy A được sử dụng để lưu toán hạng vào và kết quả ra, thường được dùng trong các lệnh trao đổi dữ liệu với thiết bị vào ra. Kích thước của thanh ghi bằng kích thước từ xử lý của CPU.

1.4 Nêu chức năng và đặc điểm của bộ đếm chương trình PC

Bộ đếm chương trình PC còn được gọi là con trỏ lệnh IP (thanh ghi EIP)

- Chức năng:
  - + Chứa địa chỉ của ô nhớ chứa *lệnh kế tiếp được thực hiện.*

+ Lưu giá trị EBP trở về cho chương trình.
+ Khi CPU thực hiện xong một lệnh, địa chỉ của ô nhớ chứa lệnh kế tiếp cần thực hiện được nạp vào PC.

- Đặc điểm: kích thước phụ thuộc vào thiết kế CPU

1.5 Nêu chức năng và phương thức hoạt động của con trỏ ngăn xếp SP

- Stack Pointer là một thanh ghi luôn chứa địa chỉ đỉnh của stack.
- SP hoạt động theo nguyên lý LIFO với hai thao tác chính:
  + Push - đẩy dữ liệu vào ngăn xếp:
    SP ← SP + 1 ; tăng địa chỉ đỉnh stack
    {SP} ← dữ liệu ; nạp dữ liệu vào stack
  + Pop - lấy dữ liệu ra khỏi ngăn xếp:
    Thanh ghi ← {SP} ; đẩy dữ liệu từ đỉnh stack vào thanh ghi
    SP ← SP - 1 ; giảm địa chỉ đỉnh stack

1.6 Thanh ghi cờ (FR) / Thanh ghi trạng thái (SR) của vi xử lý có chức năng gì ? Nêu ý nghĩa của các cờ nhớ CF, cờ không ZF và cờ dấu SF.

- Mỗi bit của thanh ghi cờ lưu trạng thái của kết quả phép tính do ALU thực hiện.
- Các bit cờ thường được sử dụng như điều kiện trong các câu lệnh rẽ nhánh để tạo logic cho chương trình. Các bit cờ gồm 2 loại:
  + Cờ trạng thái
  + Cờ điều khiển
- Cờ nhớ CF: CF = 1 nếu có nhớ / mượn. CF = 0 trong các trường hợp khác.
- Cờ không ZF: ZF = 1 nếu kết quả bằng 0. ZF = 0 nếu kết quả < 0 hoặc > 0.
- Cờ dấu SF: SF = 1 nếu kết quả âm. SF = 0 nếu kết quả dương.

1.7 Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ tức thì, cho VD.

- Chế độ địa chỉ là phương thức CPU tổ chức các toán hạng của lệnh. Chế độ địa chỉ cho phép CPU kiểm tra dạng lệnh và tìm các toán hạng của lệnh.
- Trong chế độ địa chỉ tức thì, **giá trị hằng** của toán hạng **nguồn** được đặt sau mã lệnh, còn toán hạng đích có thể là 1 thanh ghi hoặc 1 địa chỉ ô nhớ.
  VD: LOAD R1, **#1000** ; R1 <- **1000**

1.8 Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ trực tiếp, cho VD.

- Chế độ địa chỉ là phương thức CPU tổ chức các toán hạng của lệnh. Chế độ địa chỉ cho phép CPU kiểm tra dạng lệnh và tìm các toán hạng của lệnh.
- Chế độ địa chỉ trực tiếp sử dụng **một hằng** để biểu diễn **địa chỉ một ô nhớ** làm toán hạng. Toán hạng còn lại có thể là 1 thanh ghi hoặc 1 địa chỉ ô nhớ
  VD: LOAD R1, **1000**; R1 <- **M[1000]**

1.9 Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ gián tiếp qua thanh ghi, cho VD.

- Chế độ địa chỉ là phương thức CPU tổ chức các toán hạng của lệnh. Chế độ địa chỉ cho phép CPU kiểm tra dạng lệnh và tìm các toán hạng của lệnh. Số lượng các chế độ địa chỉ phụ thuộc vào thiết kế CPU.
- Chế độ địa chỉ gián tiếp qua thanh ghi sử dụng 1 **thanh ghi** để lưu **địa chỉ ô nhớ** dùng làm toán hạng. Toán hạng còn lại có thể là 1 hằng, 1 thanh ghi hoặc 1 ô nhớ
  VD: LOAD R1, **(R2)** ; R1 <- **M[R2]**

1.10 Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ gián tiếp qua ô nhớ, cho VD.

- Chế độ địa chỉ là phương thức CPU tổ chức các toán hạng của lệnh. Chế độ địa chỉ cho phép CPU kiểm tra dạng lệnh và tìm các toán hạng của lệnh. Số lượng các chế độ địa chỉ phụ thuộc vào thiết kế CPU.
- Chế độ địa chỉ gián tiếp qua ô nhớ sử dụng 1 **ô nhớ** để lưu **địa chỉ ô nhớ** dùng làm toán hạng. Toán hạng còn lại có thể là 1 hằng, 1 thanh ghi hoặc 1 ô nhớ
  VD: LOAD R1, **(100)** ; R1 <- **M[M[100]]**

1.11 Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ chỉ số, cho VD.

- Chế độ địa chỉ là phương thức CPU tổ chức các toán hạng của lệnh. Chế độ địa chỉ cho phép CPU kiểm tra dạng lệnh và tìm các toán hạng của lệnh. Số lượng các chế độ địa chỉ phụ thuộc vào thiết kế CPU.
- Trong chế độ địa chỉ chỉ số, địa chỉ của một toán hạng được tạo thành bởi **phép cộng giữa 1 hằng và 1 thanh ghi chỉ số**. Toán hạng còn lại có thể là 1 hằng, 1 thanh ghi hoặc 1 địa chỉ ô nhớ.

  VD: LOAD 100, **R1(100)** ; M[100] <- **M[R1 + 100]**

1.12 Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ tương đối, cho VD.

- Chế độ địa chỉ là phương thức CPU tổ chức các toán hạng của lệnh. Chế độ địa chỉ cho phép CPU kiểm tra dạng lệnh và tìm các toán hạng của lệnh. Số lượng các chế độ địa chỉ phụ thuộc vào thiết kế CPU.
- Trong chế độ địa chỉ tương đối, địa chỉ của một toán hạng được tạo thành bởi **phép cộng giữa 1 hằng và bộ đếm chương trình PC**. Toán hạng còn lại có thể là 1 hằng, thanh ghi hoặc địa chỉ ô nhớ.

  VD: LOAD R1, **100(PC)**; ; R1 <- **M[100 + PC]**

1.13 Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ thanh ghi? Cho VD minh họa?

- Chế độ địa chỉ là phương thức CPU tổ chức các toán hạng của lệnh. Chế độ địa chỉ cho phép CPU kiểm tra dạng lệnh và tìm các toán hạng của lệnh. Số lượng các chế độ địa chỉ phụ thuộc vào thiết kế CPU.

1.14 Nêu phương pháp phân loại bộ nhớ máy tính

Có thể phân loại bộ nhớ máy tính dựa trên 3 tiêu chí:

- Kiểu truy cập:
  + Bộ nhớ truy cập tuần tự (SAM)
  + Bộ nhớ truy cập ngẫu nhiên (RAM)
  + Bộ nhớ chỉ đọc (ROM)
- Khả năng duy trì dữ liệu:
  + Bộ nhớ ổn định
  + Bộ nhớ không ổn định
- Công nghệ chế tạo:
  + Bộ nhớ bán dẫn
  + Bộ nhớ từ tính
  + Bộ nhớ quang học

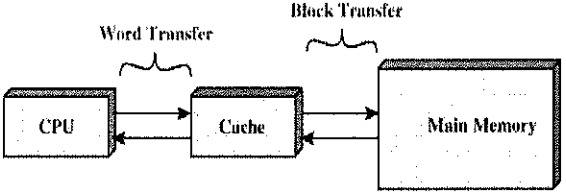
1.15 Bộ nhớ ROM là gì ? Nêu các đặc điểm của bộ nhớ ROM

- ROM là bộ nhớ chỉ đọc, thông tin lưu trữ trong ROM chỉ có thể đọc ra mà không thể ghi vào.
- Đặc điểm:
  + Thông tin trong ROM thường được nhà sản xuất ghi sẵn, gồm các thông tin về cấu hình máy và BIOS.
  + ROM thuộc loại bộ nhớ bán dẫn.
  + ROM là bộ nhớ ổn định, thông tin trong ROM vẫn được duy trì kể cả khi không có nguồn điện.

1.16 Bộ nhớ RAM là gì ? Nêu các đặc điểm của bộ nhớ RAM

- RAM là bộ nhớ truy cập ngẫu nhiên, các ô nhớ của RAM có thể được truy cập ngẫu nhiên không theo trật tự với tốc độ tương đương nhau.
- Đặc điểm:
  + RAM thường chứa các thông tin của hệ thống và người dùng.
  + RAM thuộc loại bộ nhớ bán dẫn.
  + RAM là bộ nhớ không ổn định, thông tin trong RAM sẽ mất khi không có nguồn điện nuôi.

1.17 Nêu phương thức trao đổi dữ liệu giữa CPU, cache và bộ nhớ chính

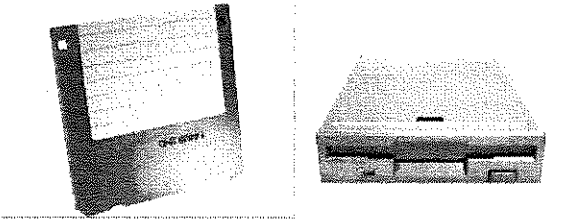


- CPU trao đổi dữ liệu với cache theo các đơn vị cơ sở như byte, word. Vì dữ liệu được lưu trong thanh ghi của CPU thường có dung lượng rất hạn chế, nên CPU chỉ trao đổi các phần tử dữ liệu cần thiết theo yêu cầu của các lệnh.
- Cache trao đổi dữ liệu với bộ nhớ chính theo các khối có kích thước 16, 32 hoặc 64 bytes. Cách trao đổi nhiều bytes kề nhau theo khối này có tác dụng bao phủ các mẫu dữ liệu theo không gian và thời gian, đồng thời giúp cache tận dụng tốt hơn băng thông đường truyền, nhờ vậy có thể tăng tốc độ truyền dữ liệu.

1.18 Nêu đặc điểm của đĩa từ và các loại đĩa từ?

Đĩa từ (Magnetic Disks) là một trong các loại thiết bị lưu trữ được sử dụng rộng rãi nhất trong các thiết bị tính toán nói chung và các máy tính cá nhân nói riêng. Đĩa từ thuộc loại bộ nhớ ổn định – thông tin lưu trên đĩa từ luôn được duy trì, không phụ thuộc vào nguồn điện nuôi bên ngoài. Đĩa từ cũng là bộ nhớ kiểu khối có dung lượng lớn, đặc biệt là các đĩa cứng, dùng để lưu trữ thông tin lâu dài dưới dạng các tệp (files). Để lưu được thông tin, đĩa từ sử dụng các đĩa nhựa hoặc đĩa kim loại có phủ lớp bột từ trên bề mặt. Bột từ được sử dụng thường là oxit sắt hoặc các hợp kim của sắt.

Có hai dạng đĩa từ chủ yếu là đĩa từ mềm (gọi tắt là đĩa mềm – Floppy Disks) và đĩa từ cứng (gọi tắt là đĩa cứng – Hard Disks). Đĩa mềm làm bằng plastic, có dung lượng nhỏ, tốc độ chậm và dễ bị hư hỏng. Người ta sử dụng ổ đĩa mềm (FDD – Floppy Disk Drive) để đọc ghi đĩa mềm. Hình 51 minh hoạ đĩa mềm và ổ đĩa mềm dung lượng 1,44MB với kích thước đĩa 3,5 inches. Ngày nay, do sự phát triển mạnh mẽ của các loại đĩa quang và đặc biệt là các thẻ nhớ flash kết nối qua cổng USB, đĩa mềm ngày càng ít được sử dụng. Nhiều hệ thống máy tính lắp mới không đi kèm ổ đĩa mềm.

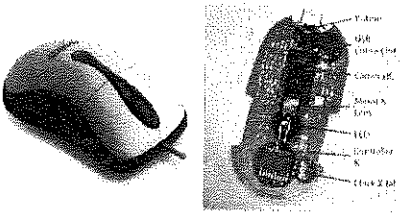


Hình 51 Đĩa mềm và ổ đĩa mềm kích thước 3,5 inches

1.19 Nêu đặc điểm chính của đĩa CD và đĩa DVD

- Đặc điểm chính của đĩa CD:
  - + Dung lượng nhỏ: tối đa 700MB / 80 phút
  - + Tốc độ truy cập chậm, tốc độ cơ sở: 150KB/s nhân với hệ số nhân
  - + Sử dụng tia hồng ngoại bước sóng 780nm
- Đặc điểm chính của đĩa DVD:
  - + Dung lượng lớn: 4.7GB với đĩa 1 mặt, 8.5GB với đĩa 2 mặt.
  - + Tốc độ truy cập cao, tốc độ cơ sở: 1350KB/s nhân với hệ số nhân
  - + Sử dụng tia hồng ngoại bước sóng 650nm

1.20 Nêu nguyên lý hoạt động của chuột quang



Hình 80 Chuột quang với cấu tạo

Chuột quang sử dụng nguyên tắc liên tục chụp và phân tích ảnh bề mặt chuột để phát hiện di chuyển:

1. Một đi-ốt phát ánh sáng đỏ qua ống kính chiếu xuống mặt phẳng di chuột. Ánh sáng phản xạ từ mặt phẳng di chuột quay ngược trở lại phía dưới chuột.
2. Một camera đặt phía dưới chuột liên tục chụp ảnh bề mặt di chuột nhờ ánh sáng phản xạ về với tốc độ khoảng 1500 ảnh/s
3. IC điều khiển chuột sẽ phân tích và so sánh các ảnh kề nhau, qua đó phát hiện chuyển động của chuột.
4. Tín hiệu biểu diễn chuyển động chuột do IC điều khiển chuột sinh ra được chuyển cho máy tính xử lý.

1.21 Nêu dạng các toán hạng 0 địa chỉ, 1 địa chỉ , 1.5 địa chỉ, 2 địa chỉ,? Cho ví dụ?

Toán hạng dạng 2 địa chỉ

Dạng: opcode addr1, addr2

Mỗi địa chỉ addr1, addr2 tham chiếu đến một ô nhớ hoặc một thanh ghi.

Ví dụ:

ADD R1, R2; R1 ←□R1 + R2; R1 là thanh ghi của CPU.	R1 cộng với R2, kết quả lọu vào R1.

ADD A, B; M[A] ←□M[A] + M[B];

Lấy nội dung của ô nhớ A cộng với nội dung của ô nhớ B, kết qua lọu vào ô nhớ A

A, B là địa chỉ các ô nhớ

Toán hạng dạng 1 địa chỉ

Dạng:

opcode addr2

Địa chỉ addr2 tham chiếu đến một ô nhớ hoặc một thanh ghi. Ngoài ra, thanh ghi tích luỹ Racc được sử dụng và có vai trò như addr1 trong toán hạng dạng 2 địa chỉ.

Ví dụ:

ADD R2; Racc ←□Racc + R2; Racc cộng với R2, kết quả lọu vào Racc.

R2 là thanh ghi của CPU

ADD B; Racc ← Racc + M[B];

Lấy nội dung của thanh ghi Racc cộng với nội dung của ô nhớ B, kết qua lọu vào Racc.

A là địa chỉ một ô nhớ.

Toán hạng dạng 1,5 địa chỉ

Dạng:

opcode addr1, addr2

Một địa chỉ tham chiếu đến một ô nhớ và địa chỉ còn lại tham chiếu đến một thanh ghi.

Dạng 1,5 địa chỉ là dạng toán hạng hỗn hợp giữa ô nhớ và thanh ghi.

Ví dụ:

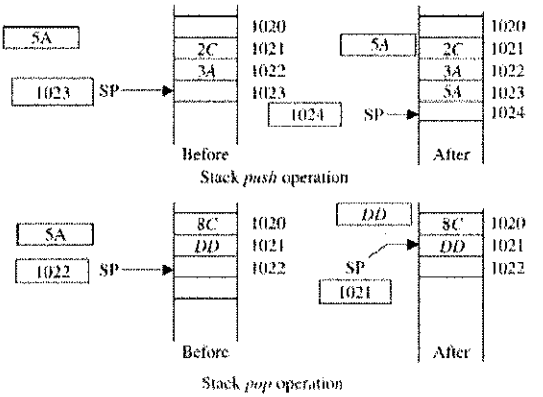
ADD R1, A; R1 ←□R1 + M[A];

Lấy nội dung của R1 cộng nội dung của ô nhớ A, kết quả lọu vào R1.

R1 là thanh ghi của CPU và A là địa chỉ ô nhớ.

Toán hạng dạng 0 địa chỉ

Toán hạng 0 địa chỉ thoịờng đoịợc sử dụng trong các lệnh thao tác với ngăn xếp: PUSH và POP như minh hoạ trên Hình 20



Hình 20 Thao tác PUSH và POP với ngăn xếp

1.22 Trình bày 5 lệnh vận chuyển dữa liệu thông dụng? cho ví dụ?

Các lệnh vận chuyển dữ liệu vận chuyển dữ liệu giữa các bộ phận của máy tính. Cụ thể, vận chuyển dữ liệu giữa các thanh ghi của CPU, nạp dữ liệu từ các ô nhớ về các thanh ghi của CPU và ngoịợc lại ghi dữ liệu từ các thanh ghi ra các ô nhớ. Ngoài ra, dữ liệu cũng có thể đoịợc vận chuyển giữa các ô nhớ trong bộ nhớ trong.

Ví dụ:

Vận chuyển dữ liệu giữa các thanh ghi của CPU:

MOVE Ri, Rj;	Ri ←□Rj
Chuyển (sao chép) nội dung của thanh ghi Rj sang thanh ghi Ri. Vận chuyển dữ liệu giữa 1 thanh ghi của CPU và một ô nhớ:	
MOVE 1000, Rj;	M[1000] ←□Rj

Lưu nội dung của thanh ghi Rj vào ô nhớ có địa chỉ 1000.

Vận chuyển dữ liệu giữa các ô nhớ:

MOVE 1000, (Rj); M[1000] ←□M[Rj]

Chuyển (sao chép) nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Rj sang số nhớ có địa chỉ 1000.

Một số lệnh vận chuyển dữ liệu thông dụng

Tên lệnh	**Ý nghĩa**
MOVE	Chuyển dữ liệu giữa thanh ghi – thanh ghi, ô nhớ - thanh ghi và ô nhớ - ô nhớ.
LOAD	Nạp nội dung 1 ô nhớ vào 1 thanh ghi.
STORE	Lọu nội dung 1 thanh ghi ra 1 ô nhớ.
PUSH	Đẩy dữ liệu vào ngăn xếp.
POP	Lấy dữ liệu ra khỏi ngăn xếp.

1.23 và 1.24 Trình bày các lệnh toán học(cộng, trừ, nhân, chia, tăng 1 đơn vị, giảm 1 đơn vị)? cho vd? Trình bày các lệnh logic thông dụng (and, or, xor, compare)? Cho vd?

Các lệnh tính toán số học và logic được sử dụng để thực hiện các thao tác tính toán trên nội dung các thanh ghi và / hoặc nội dung các ô nhớ. Các lệnh tính toán hỗ trợ hầu hết các phép toán số học thông dụng nhọi cộng, trừ, nhân, chia các số nguyên và các phép toán logic, như phủ định, và, hoặc, hoặc loại trừ.

Ví dụ:

Lệnh cộng:

ADD R1, R2, R3;	R1 ←□R2 + R3
Cộng nội dung 2 thanh ghi R2 và R3, kết quả lưu vào thanh ghi R1.	
ADD A, B, C;	M[A] ←□M[B] + M[C]
Cộng nội dung 2 ô nhớ B và C, kết quả lọọu vào ô nhớ A.	
Lệnh trừ: SUBSTRACT R1, R2, R3;	R1 ←□R2 - R3

Lấy nội dung thanh ghi R2 trừ đi nội dung thanh ghi R3, kết quả lưu vào thanh ghi R1.

Lệnh logic:

NOT R1; R1 ←□!(R1)

Lấy giá trị đảo (phủ định) của nội dung thanh ghi R1.

AND R1, R2; R1 ←□R1 ⊗□R2

Nhân bit nội dung 2 thanh ghi R1 và R2, kết quả lưu vào R1

Một số lệnh tính toán và logic thông dụng

Tên lệnh	**Ý nghĩa**
ADD	Cộng các toán hạng
SUBSTRACT	Trừ các toán hạng
MULTIPLY	Nhân các toán hạng
DIVIDE	Chia các toán hạng
INCREMENT	Tăng một đơn vị
DECREMENT	Giảm một đơn vị
NOT	Phủ định bit
AND	Phép và (nhân) bit
OR	Phép hoặc (cộng) bit
XOR	Phép hoặc loại trừ bit
COMPARE	So sánh 2 toán hạng
SHIFT	Phép dịch bit (dịch trái, dịch phải)
ROTATE	Phép quay bit (quay trái, quay phải)

1.25 Trình bày lệnh dịch thông dụng (shift left, shift right, rotate)? Cho vd?

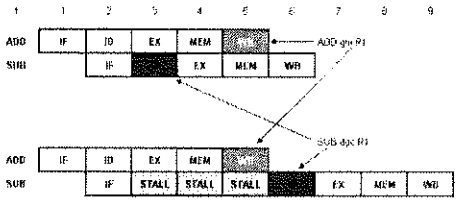
1.26 Trình bày về vấn đề tranh chấp dữ liệu kiểu đọc sau khi ghi trong cơ thể ống lệnh và cách khắc phục? Cho vd?

Tranh chấp dữ liệu cũng là một trong các vấn đề 11o+ớn của cơ chế ống lệnh và tranh chấp dữ liệu kiểu *đọc sau khi ghi* (RAW – Read After Write) là dạng xung đột dữ liệu hay gặp nhất.

Để hiểu rõ tranh chấp dữ liệu kiểu RAW, ta xem xét hai lệnh sau:

ADD R1, R2, R3; R1 ←□R2 + R3 (1)

SUB R4, R1, R2; R4 ←□R1 + R2 (2)



Hình 27 Tranh chấp dữ liệu kiểu RAW

Hình 27 minh hoạ tranh chấp dữ liệu kiểu RAW giữa hai lệnh ADD và SUB được thực hiện kế nhau trong cơ chế ống lệnh. Có thể thấy lệnh SUB sử dụng kết quả của lệnh ADD (thanh ghi R1 là kết quả của ADD và là đầu vào cho SUB) và nhơ vậy hai lệnh có sự phụ thuộc dữ liệu. Tuy nhiên, lệnh SUB đọc thanh ghi R1 tại giai đoạn giải mã (ID), trước khi lệnh ADD ghi kết quả vào thanh ghi R1 ở giai đoạn lưu kết quả (WB). Nhơ vậy, giá trị SUB đọc được từ thanh ghi R1 là giá trị cũ, không phải là kết quả tạo ra bởi ADD. Để SUB đọc được giá trị mới nhất của R1, giai đoạn ID của SUB phải lùi 3 nhịp, đến vị trí giai đoạn WB của ADD kết thúc. Có một số giải pháp cho vấn đề tranh chấp dữ liệu kiểu RAW. Cụ thể:

1. Nhận dạng tranh chấp RAW khi nó diễn ra;

2. Khi tranh chấp RAW xảy ra, tạm dừng (stall) ống lệnh cho đến khi lệnh phía trước hoàn tất giai đoạn WB;

3. Có thể sử dụng trình biên dịch (compiler) để nhận dạng tranh chấp RAW và thực hiện:
- Chèn thêm các lệnh NO-OP vào giữa các lệnh có thể gây ra tranh chấp RAW;

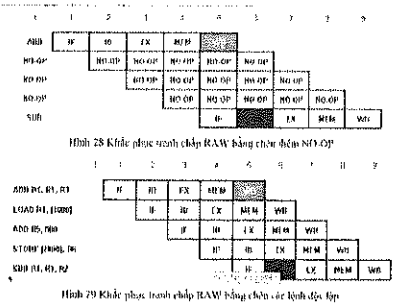
NO-OP là lệnh rỗng, không thực hiện tác vụ hữu ích mà chỉ tiêu tốn thời gian CPU.

- Thay đổi trật tự các lệnh trong chương trình và chèn các lệnh độc lập vào giữa các lệnh có thể gây ra tranh chấp RAW;

Mục đích của cả hai phương pháp kể trên là lùi việc thực hiện lệnh gây tranh chấp dữ liệu cho đến khi lệnh trước nó hoàn tất việc lưu kết quả.

4. Sử dụng phần cứng để nhận dạng tranh chấp RAW và dự đoán trước giá trị dữ liệu phụ thuộc.

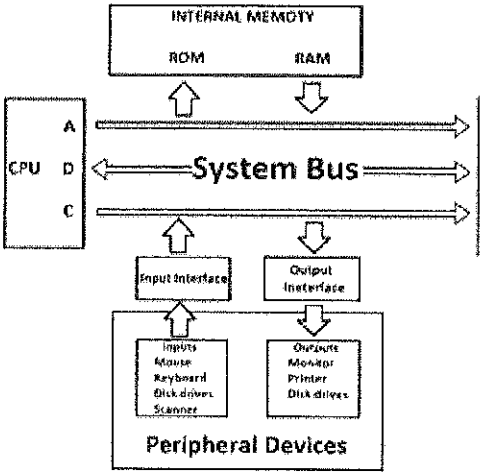
Hình 28 minh hoạ giải pháp khắc phục tranh chấp RAW bằng cách chèn thêm các lệnh NO-OP. Hình 29 minh hoạ giải pháp khắc phục tranh chấp RAW bằng cách chèn thêm các lệnh độc lập với hai lệnh có tranh chấp. Các lệnh độc lập có thể có được bằng cách thay đổi trật tự thực hiện các lệnh của chương trình mà không thay đổi kết quả thực hiện nó. Cũng có thể sử dụng giải pháp kết hợp chèn NO-OP và lệnh độc lập.



Hình 28 Khắc phục tranh chấp RAW bằng chèn thêm NO-OP

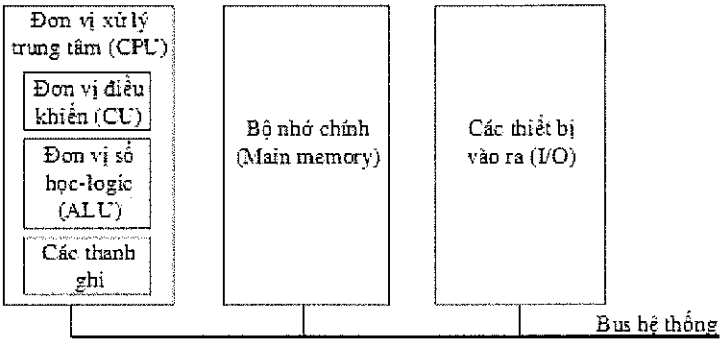
Hình 29 Khắc phục tranh chấp RAW bằng chèn các lệnh độc lập

2.1 Nêu sơ đồ khối chức năng và chức năng chính của các thành phần trong một hệ thống máy tính



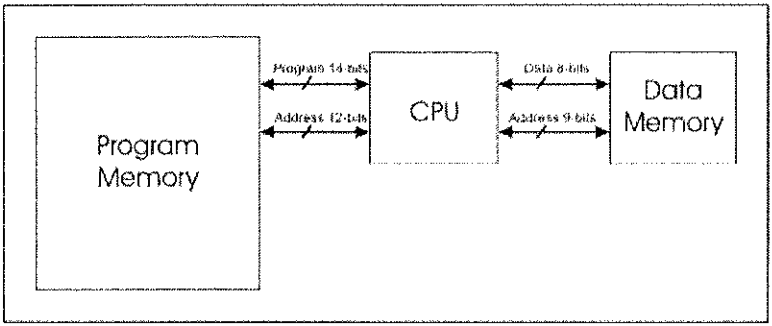
- **CPU**: là thành phần quan trọng nhất, được cho là đầu não của máy tính. CPU có chức năng đọc các lệnh của chương trình từ bộ nhớ, giải mã và thực hiện lệnh. CPU gồm:
  + Bộ điều khiển CU
  + Bộ tính toán số học và logic ALU
  + Các thanh ghi Registers
- **Bộ nhớ trong**: có chức năng chứa lệnh và dữ liệu của hệ thống và người dùng để phục vụ cho việc xử lý của CPU. Thông tin trong ROM vẫn tồn tại khi mất nguồn nuôi, còn thông tin trong RAM sẽ mất khi mất nguồn nuôi.
- **Các thiết bị ngoại vi**: có chức năng nhập dữ liệu vào, điều khiển hệ thống và xuất dữ liệu ra.
- **Bus hệ thống**: có chức năng kết nối CPU với các thành phần khác của máy tính.
  + Bus địa chỉ (Bus A) truyền tín hiệu địa chỉ từ CPU đến bộ nhớ và các thiết bị ngoại vi.
  + Bus dữ liệu (Bus D) vận chuyển các tín hiệu dữ liệu theo 2 chiều đi và đến CPU.
  + Bus điều khiển (Bus C) truyền tín hiệu điều khiển từ CPU tới các thành phần khác, đồng thời truyền tín hiệu trạng thái của các thành phần khác đến CPU.

2.2 Vẽ sơ đồ và các đặc điểm của kiến trúc máy tính Von Neumann



- Kiến trúc Von Neumann dựa trên 3 khái niệm cơ sở:
  + Lệnh và dữ liệu được lưu trữ trong bộ nhớ đọc ghi chia sẻ - 1 bộ nhớ duy nhất được sử dụng để lưu trữ cả lệnh và dữ liệu.
  + Bộ nhớ được đánh địa chỉ theo vùng, không phụ thuộc vào nội dung nó lưu trữ.
  + Các lệnh của 1 chương trình được thực hiện tuần tự.
- Quá trình thực hiện 1 lệnh được chia thành 3 giai đoạn chính:
  1. CPU đọc lệnh từ bộ nhớ.
  2. CPU giải mã và thực hiện lệnh. Nếu lệnh yêu cầu dữ liệu, CPU đọc dữ liệu từ bộ nhớ.
  3. CPU ghi kết quả thực hiện lệnh vào bộ nhớ (nếu có).

2.3 vẽ sơ đồ và các đặc điểm của kiến trúc máy tính Harvard



- Kiến trúc máy tính Harvard chia bộ nhớ trong thành 2 phần riêng rẽ:
  + Bộ nhớ lưu chương trình (Program Memory)
  + Bộ nhớ lưu dữ liệu (Data Memory)
- Hai hệ thống bus riêng được được sử dụng để kết nối CPU với bộ nhớ lưu chương trình và bộ nhớ lưu dữ liệu. Mỗi hệ thống đều có đủ 3 thành phần để truyền dẫn các tín hiệu địa chỉ, dữ liệu và điều khiển.
- Máy tính dựa trên kiến trúc Harvard có tốc độ xử lý cao hơn máy tính sử dụng kiến trúc Von Neumann nhờ:
  + Có 2 hệ thống bus độc lập với băng thông lớn
  + Hệ thống nhớ hỗ trợ nhiều lệnh truy cập bộ nhớ tại 1 thời điểm, giúp giảm xung đột truy nhập bộ nhớ, đặc biệt khi CPU sử dụng pipeline.

2.4 Nêu sơ đồ khối tổng quát và chu trình xử lý lệnh của CPU

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

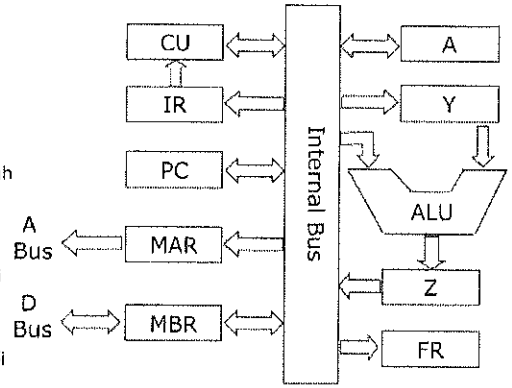
MBR: (Memory Buffer Register) Thanh ghi nhớ đệm

A: (Accumulator Register) Thanh ghi tích lũy

Y, Z: (Temporary Register) Thanh ghi tạm thời

FR: (Flag Register) Thanh ghi cờ

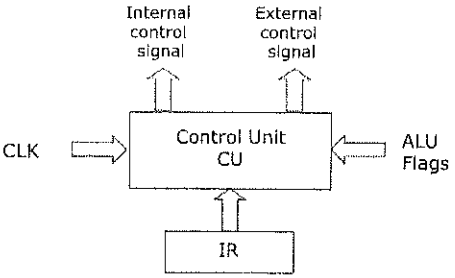
ALU: (Arithmetic and Logic Unit) Khối tính toán số học -logic



- Chu trình xử lý lệnh của CPU:
  1. Khi một chương trình được chạy, hệ điều hành tải mã chương trình vào bộ nhớ trong.
  2. Địa chỉ lệnh đầu tiên của chương trình được đưa vào thanh ghi PC.
  3. Địa chỉ của ô nhớ chứa lệnh được chuyển tới bus A qua thanh ghi MAR.
  4. Tiếp theo, bus A truyền địa chỉ tới khối quản lý bộ nhớ MMU.
  5. MMU chọn ô nhớ và sinh ra tín hiệu READ.
  6. Lệnh chứa trong ô nhớ được chuyển tới thanh ghi MBR qua bus D.
  7. MBR chuyển lệnh tới thanh ghi IR. Sau đó IR lại chuyển lệnh tới CU
  8. CU giải mã lệnh và sinh ra các tín hiệu xử lý cho các đơn vị khác, ví dụ như ALU để thực hiện lệnh.
  9. Địa chỉ trong PC được tăng lên để trỏ tới lệnh tiếp theo của chương trình sẽ được thực hiện.
  10. Thực hiện lại các bước 3 -> 9 để chạy hết các lệnh của chương trình.

2.5 Nêu sơ đồ khối và chức năng của các khối điều khiển CU và khối tính toán số học và logic ALU

Khối điều khiển CU

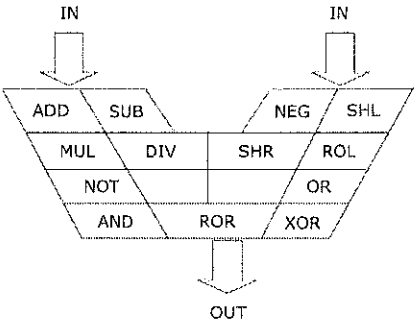


- Điều khiển tất cả các hoạt động của CPU theo xung nhịp đồng hồ.
- CU nhận 3 tín hiệu đầu vào:

+ Lệnh từ thanh ghi lệnh IR
+ Giá trị các cờ trạng thái của ALU
+ Xung nhịp đồng hồ CLK

- CU sinh 2 nhóm tín hiệu đầu ra:
  + Nhóm tín hiệu điều khiển các bộ phận bên trong CPU
  + 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 với các thành phần bên ngoài.

Khối số học và logic ALU



- Đảm nhiệm chức năng tính toán trong CPU. 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) và bộ hoặc loại trừ (XOR)
- ALU có:
  + 2 cổng IN để nhận đầu vào từ các thanh ghi.
  + 1 cổng OUT được kết nối với bus trong để gửi kết quả tới các thanh ghi.

2.6 Lệnh máy tính là gì ? Chu kỳ lệnh là gì ? Nêu các pha điển hình trong chu kỳ thực hiện lệnh. Nêu các dạng lệnh tổng quát và các thành phần của nó.

- Lệnh máy tính là 1 từ nhị phân được gán 1 nhiệm vụ cụ thể. Các lệnh của chương trình được lưu trong bộ nhớ và lần lượt được CPU đọc, giải mã và thực hiện.
- Chu kỳ lệnh là khoảng thời gian CPU thực hiện xong 1 lệnh.
- Việc thực hiện lệnh có thể được chia thành 4 giai đoạn:
  1. Đọc lệnh: lệnh được đọc từ bộ nhớ về CPU
  2. Giải mã: CPU giải mã lệnh
  3. Thực hiện lệnh
  4. Lưu kết quả thực hiện lệnh vào bộ nhớ (nếu có).
- Dạng lệnh tổng quát gồm 2 phần chính:
  + Opcode
  + Địa chỉ của các toán hạng.

2.7 Nêu các dạng địa chỉ của lệnh. Cho ví dụ minh họa với mỗi dạng địa chỉ.

- Toán hạng 3 địa chỉ:
  opcode addr1, addr2, addr3
  Mỗi địa chỉ addr1, addr2, addr3 tham chiếu đến 1 ô nhớ hoặc thanh ghi.
  VD: **ADD R1, R2, R3**

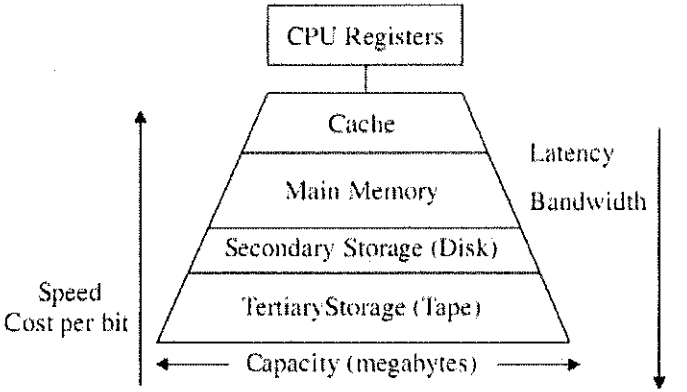
- Toán hạng 2 địa chỉ:
  opcode addr1, addr2
  Mỗi địa chỉ addr1, addr2 tham chiếu đến 1 ô nhớ hoặc thanh ghi.
  VD: **ADD R1, R2**
- Toán hạng 1 địa chỉ:
  opcode addr2
  Địa chỉ addr2 tham chiếu đến 1 ô nhớ hoặc thanh ghi. Ngoài ra, thanh ghi tích lũy R_{acc} được sử dụng với vai trò như addr1 trong toán hạng 2 địa chỉ.
  VD: **ADD R2** ; $R_{acc} \leftarrow R_{acc}$ + R2
- Toán hạng 1.5 địa chỉ:
  opcode addr1, addr2
  Một địa chỉ tham chiếu đến 1 ô nhớ, địa chỉ còn lại tham chiếu đến 1 thanh ghi. Dạng 1.5 địa chỉ là dạng **hỗn hợp giữa ô nhớ và thanh ghi**.
  VD: **ADD R1, A** ; R1 ← R1 + M[A]
- Toán hạng 0 địa chỉ:
  Thường được sử dụng trong các lệnh thao tác với stack như **PUSH** và **POP**.

2.8 Cơ chế xử lý xen kẽ dòng lệnh (pipeline) là gì ? Nêu các đặc điểm của cơ chế ống lệnh.

- Cơ chế ống lệnh là 1 phương pháp cho phép đồng thời thực hiện nhiều lệnh bằng cách chia quá trình thực hiện lệnh thành 5 giai đoạn:
  + IF: đọc lệnh từ bộ nhớ hoặc cache
  + ID: giải mã lệnh và đọc các toán hạng
  + EX: thực hiện lệnh. Nếu là lệnh truy cập bộ nhớ thì tính toán địa chỉ bộ nhớ.
  + MEM: đọc / ghi bộ nhớ, no-op nếu không truy cập bộ nhớ.
  + WB: ghi kết quả vào các thanh ghi.

  Mỗi giai đoạn được thực thi bởi 1 đơn vị chức năng khác nhau của CPU, giúp tận dụng tối đa năng lực xử lý của CPU, giảm thời gian chờ cho từng đơn vị chức năng. Nhờ vậy làm giảm thời gian thực hiện mỗi lệnh, tăng hiệu năng xử lý của CPU.
- Đặc điểm của cơ chế ống lệnh:
  + Là dạng xử lý song song ở mức lệnh.
  + Trong cùng một thời điểm thì hầu hết các đơn vị chức năng của CPU liên tục tham gia vào quá trình xử lý lệnh.
  + Số lệnh được xử lý đúng bằng số giai đoạn thực hiện lệnh.
  + Số giai đoạn thực hiện lệnh phụ thuộc vào thiết kế của CPU.

2.9 Nêu cấu trúc phân cấp của hệ thống bộ nhớ máy tính. Mô tả đặc điểm các thành phần. Tại sao cấu trúc này có thể giúp tăng hiệu năng và giảm giá thành sản xuất máy tính ?

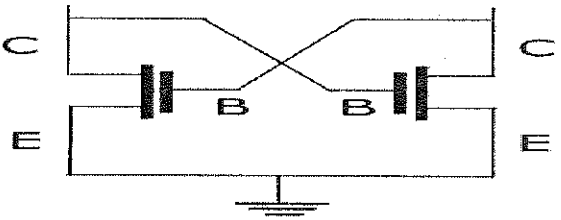


- Trong phân cấp bộ nhớ máy tính:
  + Thanh ghi CPU có cấp cao nhất, sau đó đến cache, bộ nhớ chính, và cấp thấp nhất là các bộ nhớ ngoài.
  + Các bộ nhớ cấp cao như thanh ghi CPU và cache có tốc độ cao hơn, độ trễ thấp hơn so với các bộ nhớ cấp thấp như bộ nhớ chính và bộ nhớ ngoài.
  + Giá thành với mỗi MB dung lượng bộ nhớ của các bộ nhớ cấp cao đắt hơn nhiều các bộ nhớ cấp thấp.
  + Các bộ nhớ cấp thấp như bộ nhớ ngoài có dung lượng lưu trữ lớn hơn các bộ nhớ cấp cao.
- Đặc điểm các thành phần:
  + Thanh ghi được tích hợp trong CPU và thường hoạt động theo tần số của CPU nên có tốc độ truy cập cao. Tuy nhiên, thanh ghi có dung lượng rất hạn chế nên thường để lưu toán hạng đầu vào và kết quả đầu ra của các lệnh phục vụ CPU làm việc.
  + Cache có dung lượng tương đối nhỏ và tốc độ truy cập cao. Nhiệm vụ chủ yếu của cache là dự đoán và tải trước các lệnh và dữ liệu CPU cần từ bộ nhớ chính để tăng tốc độ xử lý của CPU.
  + Bộ nhớ chính gồm ROM và RAM, có dung lượng khá lớn nhưng tốc độ truy cập chậm hơn cache. Được sử dụng để lưu lệnh và dữ liệu của hệ thống và người dùng.
  + Bộ nhớ ngoài gồm các loại thiết bị đĩa từ, đĩa quang, băng từ. Có dung lượng rất lớn nhưng tốc độ truy cập rất chậm. Vì có ưu điểm giá thành rẻ nên được sử dụng để lưu dữ liệu lâu dài.
- Cấu trúc phân cấp của hệ thống bộ nhớ có thể giúp tăng hiệu năng hệ thống vì CPU sẽ chủ yếu trực tiếp truy cập cache có tốc độ cao, còn cache sẽ chuyển trước các dữ liệu cần thiết về từ bộ nhớ chính.
- Cấu trúc phân cấp của hệ thống bộ nhớ có thể giúp giảm giá thành sản xuất vì các thành phần có tốc độ cao và đắt tiền được sử dụng với dung lượng rất nhỏ, còn các thành phần có tốc độ thấp và rẻ tiền được sử dụng với dung lượng lớn.

Câu hỏi 2.10: Vẽ sơ đồ cấu tạo cơ sở của một bít bộ nhớ RAM tĩnh (SRAM). Nêu các đặc điểm của bộ nhớ RAM tĩnh.

Trả lời:

Sơ đồ cấu tạo:



Một mạch lật (flip-flop) đơn giản

Đặc điểm của bộ nhớ RAM tĩnh (SRAM):

- SRAM sử dụng một mạch lật trigo lưỡng ổn để lưu một bit thông tin.
- Tốc độ truy cập cao do các bit SRAM có cấu trúc đối xứng.
- Thông tin trong bit SRAM ổn định nên không cần *làm tươi*.
- Cấu trúc phức tạp cần nhiều transistor nên mật độ cấy linh kiện thấp và giá cao.

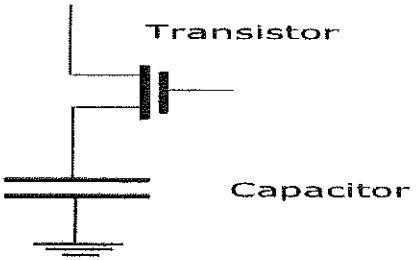
Câu hỏi 2.11: Vẽ sơ đồ cấu tạo cơ sở của một bit bộ nhớ RAM động (DRAM). Nêu các đặc điểm của bộ nhớ RAM động.

Trả lời:

Đặc điểm của bộ nhớ RAM động (DRAM):

- Cấu tạo đơn giản gồm 1 tụ điện và 1 transistor cấp nguồn.Mức điện tích trong tụ dùng để biểu diễn các giá trị 0 và 1.Có đầy điện tích tương ứng với 1 và không tích điện là 0.
- Cần *làm tươi* thường xuyên để tránh bị mất thông tin do tụ phóng điện.
- Tốc độ truy cập thấp hơn SRAM
- DRAM có cấu tạo đơn giản sử dụng ít transistor nên mật độ cấy linh kiện cao và giá rẻ.

Sơ đồ cấu tạo cơ sở DRAM:



Một bit DRAM

Câu hỏi 2.12: Phân biệt bộ nhớ RAM tĩnh và RAM động. Đánh giá ưu nhược điểm của RAM động so với RAM tĩnh?

Trả lời:

SRAM	DRAM
- Cấu tạo từ mạch lật trigo lưỡng ổn để lưu một bit thông tin - Tốc độ truy cập cao - Thông tin ổn định không cần *làm tươi*.	- Cấu tạo đơn giản gồm 1 tụ điện và 1 transistor cấp nguồn.Tụ có đầy điện tích tương ứng với 1 và không tích điện là 0. - Tốc độ truy cập thấp

- Cấu trúc phức tạp giá cao	- Cần *làm tươi* để tránh mất tt do tụ phóng điện. - Cấu trúc đơn giản giá rẻ.

Ưu nhược điểm của DRAM so với SRAM:

- Ưu điểm giá rẻ,cấu trúc đơn giản nên mật độ cấy linh kiện cao.
- Nhược điểm thông tin không ổn định cần *làm tươi* để tránh bị mất thông tin,tốc độ truy cập thấp hơn.

Câu hỏi 2.13: Trình bày khái niệm về bộ nhớ cache. Nêu vai trò của cache. Giải thích hai nguyên lý hoạt động của cache.

Trả lời:

Cache hay còn gọi là bộ nhớ đệm là một thành phần trong hệ thống nhớ phân cấp của máy tính, cache đóng vai trong trung gian, trung chuyển dữ liệu từ bộ nhớ chính về CPU và ngược lại.

Vai trò của cache

Tăng hiệu năng hệ thống

- Dung hoà được CPU có tốc độ cao và bộ nhớ chính có tốc độ thấp;
- Thời gian trung bình CPU truy nhập dữ liệu từ hệ thống nhớ tiệm cận thời gian truy nhập cache.

Giảm giá thành sản xuất

- Nếu hai hệ thống nhớ có cùng giá thành, hệ thống nhớ có cache có tốc độ truy nhập nhanh hơn;
- Nếu hai hệ thống nhớ có cùng tốc độ, hệ thống nhớ có cache có giá thành rẻ hơn.

Các nguyên lý hoạt động của cache

Cache được coi là bộ nhớ thông minh:

- Cache có khả năng đoán trước yêu cầu về dữ liệu và lệnh của CPU;
- Dữ liệu và lệnh cần thiết được chuyển trước từ bộ nhớ chính về cache → CPU chỉ truy nhập cache → giảm thời gian truy nhập hệ thống nhớ.

Cache hoạt động dựa trên 2 nguyên lý cơ bản:

- Nguyên lý lân cận về không gian (Spatial locality)
- Nguyên lý lân cận về thời gian (Temporal locality)

***Nguyên lý lân cận về không gian*:**

Nếu một ô nhớ đang được truy nhập thì xác xuất các ô nhớ liền kề với nó được truy nhập trong tương lai gần là rất cao;

Áp dụng:

- Lân cận về không gian được áp dụng cho nhóm lệnh/dữ liệu có tính tuần tự cao trong không gian chương trình;

Giải thích:

- Do các lệnh trong một chương trình thường tuần tự → cache đọc cả khối lệnh từ bộ nhớ chính → phủ được các ô nhớ lân cận của ô nhớ đang được truy nhập.

***Nguyên lý lân cận về thời gian*:**

Nếu một ô nhớ đang được truy nhập thì xác xuất nó được truy nhập lại trong tương lai gần là rất cao;

Áp dụng:

- Lân cận về thời gian được áp dụng cho dữ liệu và nhóm lệnh trong vòng lặp;

Giải thích:

- Các phần tử dữ liệu thường được cập nhật, sửa đổi thường xuyên;

Cache đọc cả khối lệnh từ bộ nhớ chính → phủ được cả khối lệnh của vòng lặp.

2.14 Nêu các đặc tính kỹ thuật cơ bản của các chuẩn ghép nối ổ đĩa cứng IDE, SATA, SCSI

	IDE	SATA	SCSI
Giống	Đều là những chuẩn ghép nối ổ đĩa cứng thông dụng		
Khác	- Tốc độ truyền tải dữ liệu tối đa: 133 MB/s - Sử dụng cáp dẹt 40 hoặc 80 sợi để ghép nối ổ đĩa. Mỗi cáp hỗ trợ ghép với 2 ổ cứng: 1 master và 1 slave	- Tốc độ truyền tải dữ liệu tối đa: 6 GB/s - Sử dụng đường truyền tín nối tiếp tốc độ cao qua 2 đôi dây với bộ điều khiển SATA sử dụng chuẩn AHCI	- Tốc độ truyền tải dữ liệu tối đa: 640 MB/s - Là một tập các chuẩn về kết nối vật lý và truyền dữ liệu giữa máy tính và thiết bị ngoại vi. Tất cả các thiết bị SCSI đều kết nối đến bus SCSI theo cùng một kiểu
Ưu		- Truyền dữ liệu nhanh và hiệu quả - Hỗ trợ cắm nóng	- Tốc độ truyền dữ liệu và tính ổn định rất cao - Hỗ trợ cắm nóng
Nhược	- Sử dụng tập lệnh mức thấp - Tốc độ truyền dữ liệu chậm - Không hỗ trợ cắm nóng	- Sử dụng tập lệnh mức thấp	- Rất đắt tiền

2.15 Trình bày nguyên lý đọc thông tin trên đĩa CD

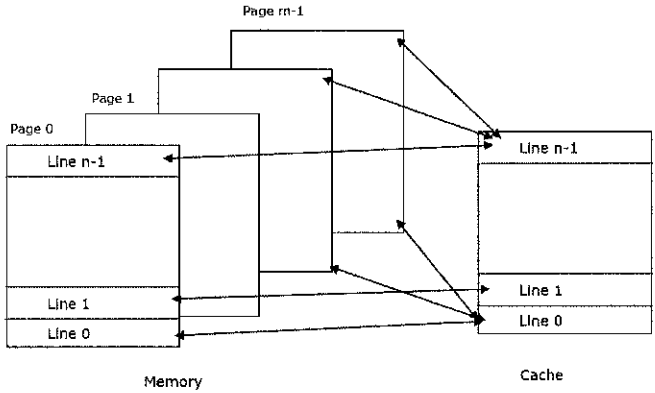
- Đĩa CD hoạt động dựa trên nguyên lý quang học:
  - + Tia laser từ điốt phát laser đi qua bộ tách đĩa đến gương quay
  - + Gương quay được điều khiển bởi tín hiệu đọc, lái tia laser đến vị trí cần đọc trên mặt đĩa
  - + Tia phản xạ từ mặt đĩa phản ánh mức độ lồi - lõm trên mặt đĩa quay trở lại gương quay
  - + Gương quay chuyển tia phản xạ về bộ tách tia và sau đó đến bộ cảm biến quang điện
  - + Bộ cảm biến quang điện chuyển đổi tia laser phản xạ thành tín hiệu điện đầu ra. Cường độ của tia laser được biểu diễn thành tín hiệu ra.2.16 Nêu nguyên lý hoạt động của máy in laser
- Máy in laser sử dụng phương pháp chụp ảnh điện tích bằng tia laser:
  - + Trống cảm quang được nạp 1 lớp điện tích nhờ điện cực.
  - + Tia laser từ nguồn sáng laser đi qua một gương quay và bộ điều chế tia được điều khiển bởi tín hiệu cần in đến mặt trống.
  - + Ánh sáng laser làm thay đổi mật độ điện tích trên mặt trống. Như vậy, mật độ điện tích trên mặt trống thay đổi theo tín hiệu cần in.
  - + Khi trống cảm quang quay đến hộp mực thì điện tích trên trống hút các hạt mực được tích điện trái dấu. Các hạt mực dính trên trống biểu diễn âm bản của văn bản / thông tin cần in.
  - + Giấy từ khay được kéo lên cũng được điện cực nạp điện tích trái dấu với điện tích của mực nên hút các hạt mực khỏi trống cảm quang.
  - + Giấy tiếp tục đi qua ống sấy nóng làm các hạt mực chảy ra và bị ép chặt vào giấy.

2.17 Nêu nguyên lý tạo hình ảnh của màn hình LCD

- Màn hình LCD tạo hình ảnh dựa trên sự linh động của các tinh thể lỏng:
  - + TFT LCD là thiết bị được điều khiển bằng các tín hiệu điện.

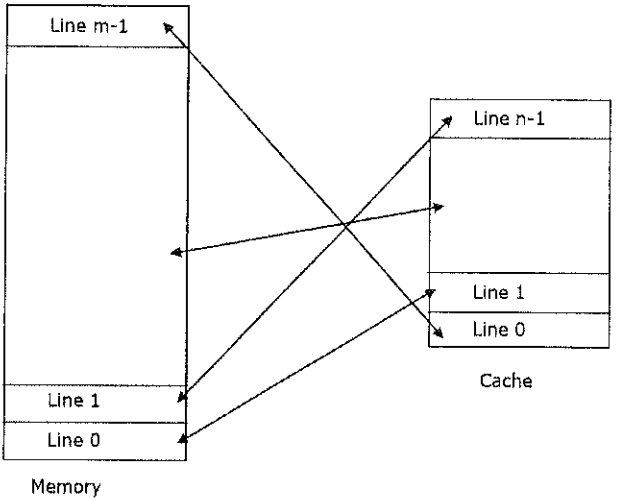
+ Lớp tinh thể lỏng nằm giữa 2 lớp trong suốt chứa các điện cực ITO.
+ Các phần tử tinh thể lỏng được sắp đặt theo các hướng khác nhau theo sự thay đổi điệp áp đặt vào các điện cực ITO.
+ Hướng của các phần tử tinh thể lỏng trực tiếp ảnh hưởng đến cường độ ánh sáng đi qua và nó gián tiếp điều khiển mức sáng / tối của ảnh hiển thị.
+ Màu của hình ảnh được tạo bởi 1 lớp lọc màu.
+ Mức xám của các điểm ảnh được thiết lập theo mức điện áp của tín hiệu video đưa vào điện cực điều khiển.

2.18 Mô tả phương pháp ánh xạ cache trực tiếp. Ưu và nhược ?



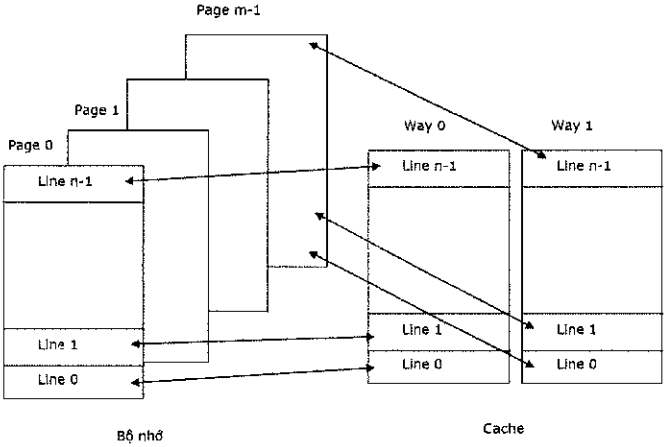
- Cache được chia thành n dòng, đánh số từ 0 → n-1.
  Bộ nhớ chính được chia thành m trang, đánh số từ 0 → m-1. Mỗi trang nhớ lại được chia thành n dòng, đánh số từ 0 → n-1.
- Ánh xạ từ bộ nhớ chính vào cache được thực hiện theo quy tắc: $Line_i$ của các trang $Page_j$ ($j \in [0; m-1]$)ánh xạ đến $Line_i$ của cache.
  Tại mỗi thời điểm, luôn có m dòng của bộ nhớ cùng cạnh tranh 1 dòng cache.
- Để có thể quản lý các ô nhớ được nạp, cache sử dụng địa chỉ ánh xạ trực tiếp gồm 3 thành phần: Tag, Line và Word
  + Tag (bit) là địa chỉ trang trong bộ nhớ chứa dòng được nạp vào cache
  + Line (bit) là địa chỉ dòng trong cache
  + Word (bit) là địa chỉ của từ trong dòng
- Ưu điểm:
  + Có thiết kế đơn giản
  + Khi biết địa chỉ ô nhớ có thể tìm được vị trí của nó trong cache rất nhanh chóng
- Nhược điểm:
  + Dễ gây xung đột dòng cache
  + Hệ số hit thấp

2.19 Mô tả phương pháp ánh xạ cache kết hợp đầy đủ. Ưu và nhược ?



- Cache được chia thành n dòng, đánh số từ 0 → n-1.
  Bộ nhớ chính được chia thành m dòng, đánh số từ 0 → m-1.
- Ánh xạ từ bộ nhớ chính vào cache được thực hiện theo quy tắc: Một dòng trong bộ nhớ chính có thể ánh xạ đến một dòng bất kỳ trong cache hay $Line_i$ ($i \in [0; m-1]$) của bộ nhớ chính ánh xạ đến $Line_j$ ($j \in [0; n-1]$) của cache.
- Để có thể quản lý được các ô nhớ được nạp, cache sử dụng địa chỉ ánh xạ kết hợp đầy đủ chỉ gồm 2 thành phần: Tag và Word
  + Tag (bit) là địa chỉ trang trong bộ nhớ chứa dòng được nạp vào cache
  + Word (bit) là địa chỉ của từ trong dòng
- Ưu điểm:
  + Mềm dẻo, giảm được xung đột sử dụng dòng cache
  + Hệ số hit cao
- Nhược điểm:
  + Thiết kế phức tạp
  + Việc truy tìm địa dòng nhớ trong cache tốn nhiều thời gian

2.20 Mô tả phương pháp ánh xạ cache tập kết hợp. Ưu và nhược ?



- Cache được chia thành k đường (way) đánh số từ 0 đến k-1. Mỗi đường cache lại được chia thành n dòng (line) đánh số từ 0 đến n-1.
  Bộ nhớ chính được chia thành m trang (page), đánh số từ 0 đến m-1. Mỗi trang lại được chia thành n dòng (line) đánh số từ 0 đến n-1.
- Ánh xạ từ bộ nhớ chính vào cache được thực hiện theo quy tắc:
  + Ánh xạ trang bộ nhớ đến đường cache (ánh xạ không cố định): Một trang của bộ nhớ có thể ánh xạ đến một đường bất kỳ của cache.
  + Ánh xạ dòng của trang đến dòng của đường (ánh xạ cố định): $Line_k$ ($k \in [0; n-1]$) của $Page_i$của bộ nhớ ánh xạ đến $Line_k$ ($k \in [0; n-1]$) của Way_j cache.
- Để có thể quản lý đọ̣c các ô nhớ đọ̣c nạp, cache sử dụng địa chỉ ánh xạ trực tiếp gồm 3 thành phần: Tag, Set và Word
  + Tag (bit) là địa chỉ trang trong bộ nhớ chứa dòng được nạp vào cache
  + Set (bit) là địa chỉ dòng trong đường cache
  + Word (bit) là địa chỉ của từ trong dòng
- Ưu điểm:
  + Nhanh do ánh xạ trực tiếp được sử dụng cho ánh xạ dòng
  + Ít xung đột do ánh xạ từ các trang bộ nhớ đến các đường cache là không cố định
  + Hệ số hit cao
- Nhược điểm:
  + Độ phức tạp thiết kế cao
  + Độ phức tạp điều khiển cao

2.21 Tại sao bộ nhớ cache thường được chia thành nhiều mức ?

Cache được chia thành nhiều mức với kích thước tăng dần và tốc độ truy nhập giảm dần giúp cải thiện hiệu năng của hệ thống. Cache nhiều mức có khả năng dung hòa tốt hơn tốc độ của CPU với tốc độ của bộ nhớ chính và có thời gian truy nhập trung bình hệ thống nhớ thấp hơn.

2.22 Nêu các phương pháp thay thế dòng cache. Phương pháp nào cho hiệu suất cao nhất, tại sao ?

- Chính sách thay thế xác định các dòng cache nào được chọn để thay thế bởi các dòng khác từ bộ nhớ. Các chính sách thay thế:
  + Ngẫu nhiên (Random)

+ Vào trước ra trước (FIFO)
+ Thay thế các dòng ít được sử dụng gần đây nhất (LRU)

- Phương pháp thay thế LRU (Least Recently Used) cho hiệu suất cao nhất vì LRU xem xét đến các dòng đang thực sự được sử dụng tuân theo yếu tố lân cận về thời gian và lựa chọn các dòng cache ít được sử dụng gần đây nhất để thay thế.

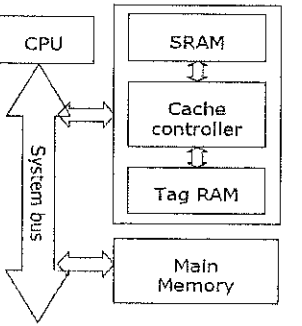
2.22 Tại sao bộ nhớ cache mức 1 thường được chia thành 2 phần: I-Cache và D-Cache ?

- Bộ nhớ cache mức 1 thường được chia thành 2 phần I-Cache và D-Cache để cải thiện hiệu năng do:
  + Dữ liệu và lệnh có tính lân cận khác nhau: Dữ liệu có tính lân cận về thời gian cao hơn lân cận về không gian còn lệnh thì ngược lại.
  + Tối ưu dễ dàng hơn vì I-Cache chỉ cần hỗ trợ thao tác đọc còn D-Cache cần được hỗ trợ cả thao tác đọc và ghi.
  + Tách Cache hỗ trợ nhiều lệnh truy cập đồng thời vào hệ thống nhớ giúp giảm xung đột tài nguyên cho ống lệnh.

3.1: Vẽ sơ đồ nguyên lý và nêu đặc điểm của hai dạng kiến trúc cache: Look Aside và Look Through. Trong hai dạng kiến trúc trên, dạng nào được sử dụng nhiều hơn trong thực tế hiện nay? Tại sao?

- Kiến trúc look aside:

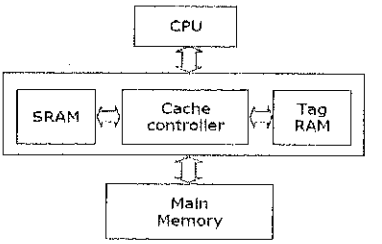
SRAM: RAM lưu dữ liệu cache
Tag RAM: RAM lưu địa chỉ dòng nhớ trong bộ nhớ
Cache controller: bộ điều khiển cache
Main memory: bộ nhớ chính
System bus: bus hệ thống



Đặc điểm:

+ Cache và bộ nhớ chính cùng được kết nối vào bus hệ thống.
+ Thiết kế đơn giản, dễ thực hiện
+ Hit chậm do sử dụng bus hệ thống có tần số không cao và băng thông hẹp
+ Miss tìm thấy nhanh do CPU đồng thời tìm tin trong cả Cache và Main memory tại cùng 1 chu kỳ xung nhịp
+ Giá thành thấp

- Kiến trúc look through:



Đặc điểm:

+ Cache được đặt nằm giữa CPU và Main memory
+ Cache kết nối với CPU bằng hệ thống bus riêng tốc độ cao (BSB – Back Side Bus) và kết nối với Main memory bằng bus hệ thống (FSB – Front Side Bus)
+ Thiết kế phức tạp
+ Hit nhanh do BSB có tốc độ cao
+ Miss chậm do tốn thêm 1 chu kỳ xung nhịp để Cache tìm tin trong Main memory
+ Giá thành cao

- Thực tế hiện nay, kiến trúc Look Through phổ biến hơn, vì cùng với sự phát triển của phần cứng, dung lượng của cache ngày càng tăng, kéo theo đó là tỉ lệ Hit cũng rất cao (lên tới 95%). Việc sử dụng kiến trúc này giúp cho máy tính hoạt động nhanh hơn rất nhiều so với Look Aside.

Câu 3.2: So sánh 3 phương pháp ánh xạ cache: ánh xạ trực tiếp, ánh xạ kết hợp đầy đủ và ánh xạ tập kết hợp ? Phương pháp ánh xạ nào trong các phương pháp trên được sử dụng nhiều nhất trong thực tế? Tại sao?

ánh xạ trực tiếp	ánh xạ kết hợp đầy đủ	ánh xạ tập kết hợp
Chia thành m trang mỗi trang n dòng.	Chia thành m trang mỗi trang n dòng.	Cache chia thành k đường và n dòng. Bộ nhớ chính được chia thành m trang và n dòng.
Tại mỗi thời điểm cache nạp 1 trang do kích thước cache chính bằng kích thước 1 trang.	Kích thước cache chính bằng kích thước 1 trang tuy nhiên một dòng trong bộ nhớ có thể ánh xạ tới dòng bất kỳ trong cache	Kích thước mỗi trang bằng kích thước mỗi đường trong cache, Một trang trong bộ nhớ có thể ánh xạ đến một đường bất kỳ trong cache. Kết hợp ánh xạ cố định và không cố định.
Được gọi là ánh xạ cứng hay ánh xạ cố định.	Được gọi là ánh xạ mềm hay ánh xạ không cố định.	Gồm 3 thành phần: Tag – Set – Word
Gồm 3 thành phần: Tag – Line – Word	Gồm 2 thành phần: Tag – Word	Thiết kế rất phức tạp
Thiết kế đơn giản	Thiết kế phức tạp	Nhanh do số lượng đường ít và không tốn nhiều thời gian tìm địa chỉ ô nhớ trong mỗi đường
Nhanh do không tốn nhiều thời gian truy tìm địa chỉ ô nhớ trong cache.	Chậm hơn do tốn nhiều thời gian tìm địa chỉ ô nhớ trong cache	Ít xung đột Hit cao
Dễ gây xung đột	Ít xung đột	
Hit thấp	Hit cao	

- Trong thực tế, phương pháp ánh xạ tập kết hợp được sử dụng nhiều nhất do kết hợp được nhiều ưu điểm của cả 2 phương pháp tiền nhiệm. Cùng với đó là sự phát triển của kĩ thuật phần cứng khiến trở ngại về thiết kế không còn quá lớn.

3.3 Nêu các phương pháp đọc ghi và các chính sách thay thế dòng cache. Tại sao thay thế dòng cache sử dụng phương pháp LRU có khả năng cho hệ số đoán trúng (hit) cao nhất?

- Phương pháp đọc:
  + Trường hợp Hit:
    - Mục dữ liệu được đọc từ cache vào CPU
    - Main memory không tham gia
  + Trường hợp Miss:
    - Mục dữ liệu được đọc từ bộ nhớ vào cache trước khi cache chuyển nó tới CPU
    - Miss penalty bằng tổng thời gian truy cập cache và Main memory
- Phương pháp ghi:
  + Trường hợp Hit:
    - Write through (ghi thẳng): dữ liệu được ghi đồng thời vào Cache và Main memory
    - Write back (ghi trễ): dữ liệu trước tiên được ghi vào cache và cả dòng chứa nó ở trong cache sẽ được ghi lại vào bộ nhớ sau đó, khi mà dòng đó bị thay thế
  + Trường hợp Miss:
    - Write Allocate (ghi có đọc lại): Dữ liệu được ghi bộ nhớ sau đó dòng chứa nó sẽ được đọc vào cache
    - Write Non-allocate (ghi không đọc lại): Dữ liệu chỉ được ghi vào bộ nhớ
- Chính sách thay thế (replacement policies) xác định các dòng cache nào được chọn để thay thế bởi các dòng khác từ bộ nhớ. Các chính sách thay thế:
  + Ngẫu nhiên (Random)
  + Vào trước ra trước (FIFO)
  + Thay thế các dòng ít được sử dụng gần đây nhất (LRU)
- Thay thế các dòng ít được sử dụng gần đây nhất LRU (Least Recently Used) có khả năng cho hệ số Hit cao nhất vì LRU xem xét đến các dòng đang thực sự được sử dụng tuân theo yếu tố lân cận về thời gian và lựa chọn các dòng cache ít được sử dụng gần đây nhất để thay thế.

3.4 RAID là gì ? Tại sao RAID có thể nâng cao được tính tin cậy, tốc độ truy nhập và dung lượng hệ thống lưu trữ ? Cấu hình RAID nào phù hợp hơn với máy chủ cơ sở dữ liệu trong ba loại RAID 0, RAID 1 và RAID 10 ? Giải thích.

- RAID (Redundant Array of Independent Disks) là một công nghệ tạo các thiết bị lưu trữ tiên tiến trên cơ sở đĩa cứng, nhằm đạt được các mục đích:
  + Tốc độ cao (High performance / speed)
  + Tính tin cậy cao (High reliability)
  + Dung lượng lớn (Large volume)
- RAID có thể nâng cao được tính tin cậy, tốc độ truy nhập và dung lượng hệ thống lưu trữ vì: (dựa trên 2 loại RAID cơ bản là RAID 0 và RAID 1)
- Raid 0 Disk Stripping (Tối thiểu 2 đĩa):
  + Dữ liệu được chia thành các khối và mỗi khối được đồng thời ghi vào 1 đĩa độc lập trong quá trình ghi
  + Khi đọc dữ liệu được đọc song song từ 2 đĩa và ghép lại dẫn tới tốc độ đọc nhanh hơn

  → Tốc độ đọc ghi được cải thiện, hệ số sử dụng bộ nhớ 1:1 tuy nhiên nguy cơ mất mát dữ liệu khi một trong 2 đĩa không hoạt động.
- Raid 1 Disk Mirroring (Tối thiểu 2 đĩa):
  + Dữ liệu được chia thành các khối và mỗi khối được ghi đồng thời vào nhiều đĩa trong quá trình ghi
  + Dữ liệu trong mỗi đĩa có 1 bản sao dữ liệu ở đĩa khác

  → Tốc độ đọc ghi giống như một địa, hệ số sử dụng bộ nhớ cao nhất 1:2 và chỉ mất mát dữ liệu khi cả 2 đĩa không hoạt động
- RAID 10 phù hợp với máy chủ cơ sở dữ liệu nhất bởi tính an toàn, độ tin cậy cao và tốc độ truy cập nhanh.

3.5 Nêu các đặc điểm chính của kiến trúc bus PCI và PCIE. Tại sao bus PCIE có khả năng hỗ trợ nhiều cặp thiết bị truyền dữ liệu đồng thời với tốc độ cao ?

- PCI:
  + Là một bus dùng chung hay bus chia sẻ (shared bus)
  + Hỗ trợ băng thông 32 bit hoặc 64 bit. Tốc độ truyền dữ liệu khá cao theo tần số làm việc và băng thông
  + Hỗ trợ nhiều thiết bị kết nối đồng thời, nhưng tại mỗi thời điểm, chỉ có một cặp thiết bị được sử dụng bus để trao đổi dữ liệu
  + Một giao dịch PCI được thực hiện theo 3 pha:
    - Pha tuỳ chọn (Arbitration): khởi tạo giao dịch
    - Pha địa chỉ (Address): xác định địa chỉ bên tham gia giao dịch
    - Pha dữ liệu (Data): truyền dữ liệu giữa các bên
- PCIE:
  + Là một dạng bus truyền dữ liệu nối tiếp, kiểu điểm đến điểm (point to point) với tốc độ cao
  + Độ rộng bus từ 1 → 32 bit tuỳ theo cấu hình
  + Các liên kết nối tiếp point to point và một cặp liên kết nối tiếp (theo 2 chiều ngược nhau) tạo thành một luồng. Tốc độ truyền dữ liệu phụ thuộc số luồng sử dụng và phiên bản của chuẩn.
  + Hỗ trợ nhiều cặp thiết bị cùng tham gia truyền dữ liệu đồng thời với tốc độ cao.
- Bus PCIE có khả năng hỗ trợ nhiều cặp thiết bị truyền dữ liệu đồng thời với tốc độ cao nhờ có khả năng cung cấp đường truyền riêng cho các cặp thiết bị tham gia sử dụng bus.

3.6 Cơ chế ống lệnh (pipeline) của CPU thường gặp phải những vấn đề gì? Nêu một hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình sau:

ADD R1, R2, R3
ADD R4, R4, #300
CMP R1, #100
SUB R5, #2000

Biết rằng mỗi lệnh được chia thành 5 giai đoạn trong pipeline: Đọc lệnh (IF), giải mã & đọc toán hạng (ID), truy nhập bộ nhớ (MEM), thực hiện (EX) và lưu kết quả (W).

- Cơ chế ống lệnh pipeline của CPU thường gặp phải 3 loại vấn đề:
  + Xung đột tài nguyên: Tại một thời điểm có 2 hay nhiều lệnh cùng đọc/ghi một thanh ghi hoặc bộ nhớ.
  + Tranh chấp dữ liệu (RAW – Read After Write): Câu lệnh phía sau sử dụng kết quả của câu lệnh phía trước chưa được write back.
  + Các lệnh rẽ nhánh: Do các câu lệnh rẽ nhánh phá vỡ tiến trình thực hiện tuần tự của ống lệnh, khiến ống lệnh bị đẩy ra trống rỗng và nạp lại mới.

	1	2	3	4	5	6	7	8
ADD R1, R2, R3	IF	ID	EX	MEM	**WB**			
ADD R4, R4, #300		IF	ID	EX	MEM	WB		
CMP R1, #100			IF	**ID**	EX	MEM	WB	
SUB R5, #2000				IF	ID	EX	MEM	WB

Có thể thấy rằng giá trị R1 trong hàm CMP đã được đọc trước khi kết quả của ADD R1,R2,R3 được ghi vào bộ nhớ.

→ Hướng giải quyết (Sắp xếp lại thứ tự câu lệnh trong ống và chèn thêm no-op):

1 2 3 4 5 6 7 8 9

ADD R1, R2, R3	IF	ID	EX	MEM	WB				
NOP		NOP	NOP	NOP	NOP	NOP			
ADD R4, R4, #300			IF	ID	EX	MEM	WB		
SUB R5, #2000				IF	ID	EX	MEM	WB	
CMP R1, #100					IF	ID	EX	MEM	WB

3.7 Cơ chế ống lệnh (pipeline) của CPU thường gặp phải những vấn đề gì? Nêu một hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình sau:

ADD R4, R4, #300
ADD R1, R1, R3
SUB R5, #2000
SUB R1, R1, #100

biết rằng mỗi lệnh được chia thành 5 giai đoạn trong pipeline: Đọc lệnh (IF), giải mã & đọc toán hạng (ID), truy nhập bộ nhớ (MEM), thực hiện (EX) và lưu kết quả (W).

- < Tương tự 3.6 >

	1	2	3	4	5	6	7	8
ADD R4, R4, #300	IF	ID	EX	MEM	WB			
ADD R1, R1, R3		IF	ID	EX	MEM	**WB**		
SUB R5, #2000			IF	ID	EX	MEM	WB	
SUB R1, R1, #100				IF	**ID**	EX	MEM	WB

Có thể thấy rằng giá trị R1 trong hàm SUB đã được đọc trước khi kết quả của ADD R1,R1,R3 được ghi vào bộ nhớ.

→ Hướng giải quyết (Sắp xếp lại thứ tự câu lệnh trong ống và chèn thêm no-op):

	1	2	3	4	5	6	7	8	9
ADD R1, R1, R3	IF	ID	EX	MEM	WB				
NOP		NOP	NOP	NOP	NOP	NOP			
ADD R4, R4, #300			IF	ID	EX	MEM	WB		
SUB R5, #2000				IF	ID	EX	MEM	WB	
SUB R1, R1, #100					IF	ID	EX	MEM	WB

3.8 Cho đoạn chương trình sau (R1, R2 là các thanh ghi và các lệnh quy ước theo dạng LỆNH <ĐÍCH> <GỐC >):

(1) LOAD R2, #400
(2) LOAD R1, #1200
(3) STORE (R1), R2
(4) SUBSTRACT R2, #20
(5) ADD 1200, #10
(6) ADD R2, (R1)

a. Xác định chế độ địa chỉ và ý nghĩa của từng lệnh
b. Xác định giá trị của thanh ghi R2 sau khi thực hiện xong lệnh số (6)

Lệnh	Chế độ địa chỉ	Ý nghĩa	Kết quả
LOAD R2, #400	Tức thì	R2 ← 400	R2 = 400
LOAD R1, #1200	Tức thì	R1 ← 1200	R1 = 1200
STORE (R1), R2	Gián tiếp qua thanh ghi	M[R1] ← R2	M[1200] = 400
SUBSTRACT R2, #20	Tức thì	R2 ← R2 - 20	R2 = 380
ADD 1200, #10	Tức thì	M[1200] ← M[1200] + 10	M[1200] = 410
ADD R2, (R1)	Gián tiếp qua thanh ghi	R2 ← R2 + M[R1]	R2 = 790

3.9 Cho đoạn chương trình sau (R1, R2 là các thanh ghi và các lệnh quy ước theo dạng LỆNH <ĐÍCH> <GÓC >):

(1) LOAD R2, #500
(2) LOAD R1, #2000
(3) STORE (R1), R2
(4) ADD 2000, #30
(5) SUBSTRACT R2, #15
(6) ADD R2, (R1)

a. Xác định chế độ địa chỉ và ý nghĩa của từng lệnh;
b. Xác định giá trị của thanh ghi R2 sau khi thực hiện xong lệnh số (6).

Lệnh	Chế độ địa chỉ	Ý nghĩa	Kết quả
LOAD R2, #500	Tức thì	R2 ← 500	R2 = 500
LOAD R1, #2000	Tức thì	R1 ← 2000	R1 = 2000
STORE (R1), R2	Gián tiếp qua thanh ghi	M[R1] ← R2	M[2000] = 500
ADD 2000, #30	Tức thì	M[2000] ← M[2000] + 30	M[2000] = 530
SUBSTRACT R2, #15	Tức thì	R2 ← R2 - 15	R2 = 485
ADD R2, (R1)	Gián tiếp qua thanh ghi	R2 ← R2 + M[R1]	R2 = 1015

3.10 Cho một dãy số nguyên gồm 50 phần tử lưu trong bộ nhớ bắt đầu từ địa chỉ 1000. Viết chương trình sử dụng tập lệnh của CPU (các lệnh quy ước theo dạng LỆNH <ĐÍCH> <GỐC >) tính:

a. Tổng của các số dương – lưu kết quả vào ô nhớ có địa chỉ 2000.

b. Tổng của tất cả các số trong dãy – lưu kết quả vào ô nhớ có địa chỉ 2010.

Tổng của các số dương	Tổng của tất cả các số

MOVE R0, #50;
CLEAR R1;
CLEAR R2;
CLEAR R3;
LAP:
LOAD R1, 1000(R2);
DECREMENT R0;
INCREMENT R2;
COMPARE R1, #0;
IF (SF=0) ADD R3, R1;
COMPARE R0, #0;
IF (ZF=0) JMP LAP;
STORE 2000, R3;
	MOVE R0, #50;
	CLEAR R1;
	CLEAR R2;
	LAP:
	ADD R1, 1000(R2);
	DECREMENT R0;
	INCREMENT R2;
	COMPARE R0, #0;
	IF (ZF=0) JMP LAP;
	STORE 2010, R1;

3.11 Cho một dãy số nguyên gồm 30 phần tử lưu trong bộ nhớ kết thúc tại địa chỉ 1500. Viết chương trình sử dụng tập lệnh của CPU (các lệnh quy ước theo dạng LỆNH <ĐÍCH> <GỐC>) tính:

a. Tổng của các số âm – lưu kết quả vào ô nhớ có địa chỉ 1800.

b. Tổng của tất cả các số trong dãy – lưu kết quả vào ô nhớ có địa chỉ 1900.

Tổng của các số âm	Tổng của tất cả các số

MOVE R0 #30;
CLEAR R1;
CLEAR R2;
CLEAR R3;
LAP:
LOAD R1, 1500(R2);
DECREMENT R0;
INCREMENT R2;
COMPARE R1, #0;
IF (SF=1) ADD R3, R1;
COMPARE R0, #0;
IF (ZF=0) JMP LAP;
STORE 1800, R3;
	MOVE R0, #30;
	CLEAR R1;
	CLEAR R2;
	LAP:
	ADD R1, 1000(R2);
	DECREMENT R0;
	INCREMENT R2;
	COMPARE R0, #0;
	IF (ZF=0) JMP LAP;
	STORE 1900, R1;

Một số dạng lệnh thường dùng

- Lệnh vận chuyển dữ liệu:
  - + MOVE: giữa thanh ghi - thanh ghi, ô nhớ - thanh ghi, ô nhớ - ô nhớ
  - + LOAD: nạp nội dung trong 1 ô nhớ vào thanh ghi
  - + STORE: lưu nội dung trong 1 thanh ghi ra ô nhớ
- Lệnh toán học và logic:
  - + ADD: cộng
  - + SUBSTRACT: trừ
  - + INCREMENT: cộng thêm 1
  - + DECREMENT: trừ đi 1
  - + CLEAR: reset về giá trị 0

+ COMPARE: so sánh
- Lệnh rẽ nhánh:
  + IF
  + JMP: nhảy đến đoạn lệnh, hàm

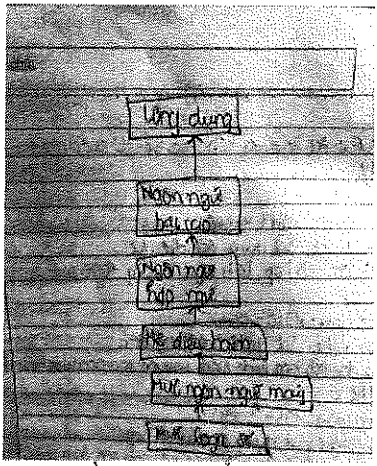
PHẦN 2 ĐỀ CƯƠNG THẦY SỸ

Câu 1: Trình bày các thế hệ máy tính theo sự phát triển của công nghệ

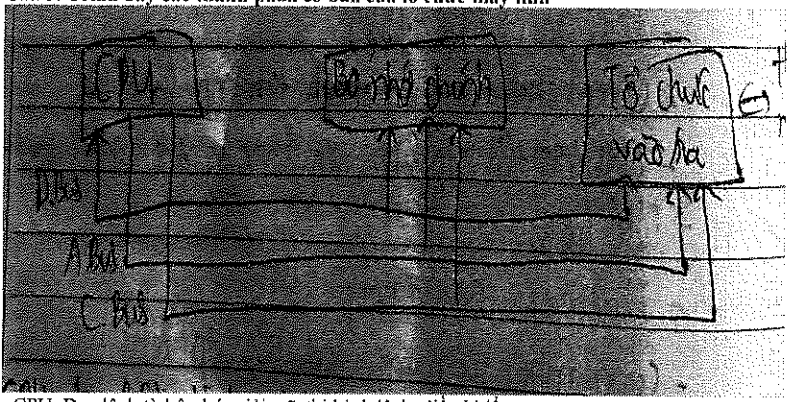
-Thế hệ 1:
+) Thời gian: 1944-1949
+) Công nghệ: Sử dụng đèn điện tử
+) Mật độ tích hợp linh kiện: khoảng 1000 linh kiện/foot^3
-Thế hệ 2:
+) Thời gian: 1960-1964
+) Công nghệ: Sử dụng bóng bán dẫn
+) Mật độ tích hợp linh kiện: khoảng 100000 linh kiện/foot^3
-Thế hệ 3:
+) Thời gian: 1964-1975
+) Công nghệ: sử dụng mạch tích hợp
+) Mật độ tích hợp linh kiện: 10000000 linh kiện/foot^3
-Thế hệ 4:
+) Thời gian:1975-1989
+) Công nghệ: Sử dụng mạch tích hợp loại lớn
+) Mật độ tích hợp linh kiện : khoảng 1 tỉ linh kiện/foot^3
-Thế hệ 5:
+)Thời gian: 1990-> nay
+) Công nghệ: Sử dụng mạch tích hợp loại siêu lớn
+) Mật độ tích hợp linh kiện: rất cao (0,180um-0,045um)

Câu 2: Trình bày khái niệm Kiến trúc máy tính (và mô hình 6 mức của máy tính hiện đại)

- Kiến trúc máy tính là khoa học về lựa chọn và kết nối với các thành phần cứng để tạo ra các máy tính đạt yêu cầu về chức năng, hiệu năng, giá thành.
- Các thành phần cơ bản của kiến trúc máy tính:
  - Kiến trúc tập lệnh: hình ảnh của hệ thống máy tính ở mức ngôn ngữ máy.
  - Vi kiến trúc (Tổ chức máy tính): mô tả mức thấp việc phối ghép và trao đổi thông tin giữa các thành phần của hệ thống.
  - Thiết kế hệ thống: kết nối các thành phần phần cứng.
- Mô hình 6 mức của máy tính hiện đại:

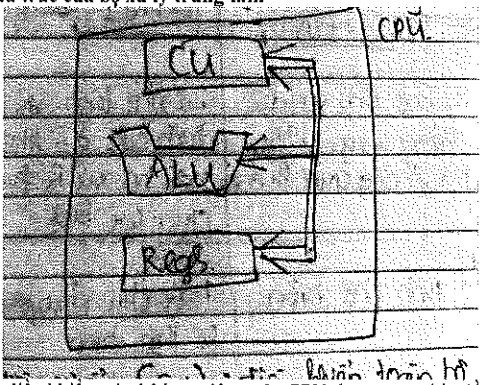


Câu 3: Trình bày các thành phần cơ bản của tổ chức máy tính



- CPU: Đọc lệnh từ bộ nhớ, giải mã, thi hành lệnh, điều khiển.
- Bộ nhớ chính: chứa các lệnh và dữ liệu
- Tổ chức vào/ra: Cung cấp dữ liệu để kết nối với thiết bị ngoại vi
- Bus dữ liệu (D.Bus): Vận chuyển dữ liệu qua lại giữa CPU, bộ nhớ chính
- Bus địa chỉ (A.Bus): Truyền tín hiệu địa chỉ từ CPU đến bộ nhớ và các thiết bị ngoại vi.
- Bus điều khiển(C.Bus): Truyền tín hiệu điều khiển từ CPU đến bộ nhớ chính và thiết bị ngoại vi và các tín hiệu trạng thái từ các thành phần khác về CPU.

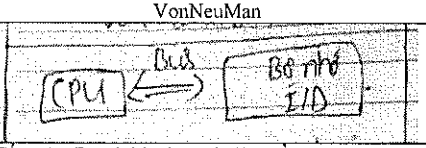
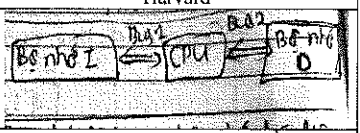
Câu 4: Trình bày cấu trúc của bộ xử lý trung tâm



- Bộ điều khiển(CU): điều khiển toàn bộ hoạt động của CPU theo xung nhịp đồng hồ.
- Bộ tính toán số học và logic(ALU): chức năng tính toán
- Thanh ghi(Regs): Lưu trữ các lệnh và dữ liệu cho hoạt động của CPU

+) Thanh ghi tích lũy A: lưu toán hạng, chứa kết quả ra
+) Bộ đếm chương trình PC: chứa địa chỉ của ô nhớ chứa lệnh kế tiếp được thực hiện
+) Thanh ghi IR: lưu lệnh được thực hiện
+) Thanh ghi địa chỉ bộ nhớ(MAR): nhận địa chỉ ô nhớ chứa lệnh tiếp theo từ PC và chuyển tiếp ra bus địa chỉ
+) Thanh ghi đệm(MBR): nhận lệnh từ Bus đ/c và chuyển tiếp lệnh đến IR thông qua bus trong CPU
+) Thanh ghi FR(cờ): mỗi bit của thanh ghi lưu kết quả phép tính ALU thực hiện
+) Thanh ghi tạm: chứa toán hạng vào, kết quả đầu ra.

Câu 5: Phân biệt kiến trúc VonNeuman và Harvard

VonNeuMan	Harvard
	
Cấu trúc: Bộ nhớ lệnh và dữ liệu nằm trong cùng 1 khối , nối với CPU thông qua Bus	Bộ nhớ lệnh và bộ nhớ dữ liệu nằm tách biệt ở 2 khối khác nhau nối với bộ xử lý trung tâm qua các Bus.
Ứng dụng: Máy tính phổ thông	Máy chủ chuyên dụng, bộ xử lý tín hiệu DSP

Câu 6: Trình bày các loại thanh ghi điển hình trong bộ vi xử lý

- Thanh ghi tích lũy A: lưu toán hạng và chứa kết quả ra
- Bộ đếm chương trình PC(IP): chứa địa chỉ của ô nhớ chứa lệnh kế tiếp đc thực hiện
- Thanh ghi IR(lệnh): lưu lệnh đc thực hiện
- Thanh ghi địa chỉ bộ nhớ(MAR): nhận địa chỉ ô nhớ chứa lệnh tiếp theo từ PC và chuyển tiếp ra bus địa chỉ
- Thanh ghi đệm(MBR): nhận lệnh từ Bus đ/c và chuyển tiếp lệnh đến IR thông qua bus trong CPU
- Thanh ghi tạm thời: chứa toán hạng đầu vào và kết quả đầu ra.
- Thanh ghi tổng quát: sd cho nhiều mục đích, chứa toán hạng đầu vào và kết quả đầu ra.

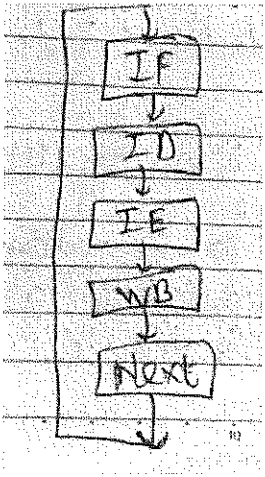
- Thanh ghi FR(cờ): mỗi bit của thanh ghi lưu kết quả phép tính ALU thực hiện

Câu 7: Phân biệt hai loại máy tính CISC và RISC

Tiêu chí	RISC	CISC
Cấu tạo	Máy trạng thái hữu hạn	Vi chương trình
Tập lệnh	80 -> 100 lệnh	200 -> 300 lệnh
Tốc độ xử lý lệnh	Nhanh hơn	Chậm hơn
Lập trình	Phức tạp hơn	Đơn giản hơn
Ứng dụng	- Các hệ VXL đơn giản - Trong bộ vi điều khiển	- Trong các hệ thống máy tính phổ thông

Câu 8: Trình bày khái niệm lệnh và quá trình thực hiện lệnh

- Lệnh: Là từ mã nhị phân để mã hóa cho 1 tác vụ (hoạt động) của CPU.
- Quá trình thực hiện lệnh gồm 5 bước:



- IF: CPU lấy lệnh, đọc lệnh từ bộ nhớ
- ID: giải mã lệnh và đọc các toán hạng
- IE: Thực hiện lệnh, nếu là lệnh truy nhập bộ nhớ tính toán đ/c bộ nhớ
- WB: Ghi kết quả vào các thanh ghi
- Next: Chuyển lệnh tiếp theo

Câu 9: Trình bày các loại toán hạng và cho VD minh họa

- ❖ Toán hạng dạng 0 địa chỉ: được sử dụng trong các lệnh thao tác với ngăn xếp.
  VD: PUSH và POP vào ngăn xếp.
- ❖ Toán hạng dạng 1 địa chỉ:
  Dạng: opcode addr2
  - Địa chỉ addr2 tham chiếu đến 1 ô nhớ hoặc 1 thanh ghi.
  - Thanh ghi tích lũy R_{acc} được sử dụng và có vai trò như addr1 trong toán hạng dạng 2 địa chỉ.

  VD:
  ADD R1; R_{acc} <- R_{acc} + R1 ;
  ADD B; R_{acc} <- R_{acc} + M[B];
- ❖ Toán hạng dạng 2 địa chỉ :

Dạng: opcode addr1, addr2

Mỗi địa chỉ addr1, addr2 tham chiếu đến 1 ô nhớ hoặc 1 thanh ghi.

VD:

ADD R1, R2; R2 <- R1+R2;

ADD A, B; M[A] <- M[A] + M[B];

❖ Toán hạng dạng 3 địa chỉ:

Dạng: opcode addr1, addr2, addr3

Mỗi địa chỉ addr1, addr2, addr3 tham chiếu đến 1 ô nhớ hoặc 1 thanh ghi

VD:

ADD R1,R2,R3; R1 <- R2 + R3;

ADD A,B,C; M[A] <- M[B] + M[C];

Câu 10: Trình bày khái niệm chế độ định địa chỉ, phân loại các chế độ định địa chỉ của máy tính và cho vd?

❖ Khái niệm Chế độ định địa chỉ: là phương thức CPU tổ chức các toán hạng của lệnh.

❖ Phân loại:

- Chế độ địa chỉ tức thì: Giá trị hằng của toán hạng nguồn được đặt ngay sau mã lệnh, toán hạng đích là 1 thanh ghi/ô nhớ.

  VD:

  LOAD R1, #1000; R1 <- 1000;

  LOAD B, #500; M[B] <- 500;

- Chế độ địa chỉ trực tiếp: Sử dụng 1 hằng để biểu diễn địa chỉ 1 ô nhớ làm 1 toán hạng. Toán hạng còn lại có thể là 1 thanh ghi hoặc 1 địa chỉ ô nhớ.

  VD:

  LOAD R1, 1000; R1 <- M[1000];

  (Nạp nội dung ô nhớ có địa chỉ 1000 vào thanh ghi R1)

- Chế độ địa chỉ gián tiếp

  Gián tiếp qua thanh ghi là chế độ địa chỉ mà một thanh ghi được sử dụng để ghi địa chỉ một ô nhớ làm toán hạng nguồn.

  VD: LOAD Rj, (Ri); Rj <- M[Ri];

  Gián tiếp qua ô nhớ là chế độ địa chỉ mà ô nhớ sử dụng để lưu địa chỉ ô nhớ làm toán hạng nguồn.

  VD: LOAD (1000), Ri; Ri <- M[1000];

- Chế độ địa chỉ chỉ số: Địa chỉ của 1 toán hạng đc tạo bởi 1 hằng và 1 thanh ghi chỉ số.

  VD:

  LOAD Ri, X(Rind); Ri <- M[X + Rind];

  (X là hằng số và Rind là thanh ghi chỉ số)

- Chế độ địa chỉ tương đối: Địa chỉ của 1 toán hạng được tạo thành bởi 1 hằng và bộ đếm chương trình PC.

  VD:

  LOAD Ri, X(PC); Ri <- M[X + PC];

  -Chế độ địa chỉ tương đối cơ sở là chế độ địa chỉ mà địa chỉ toán hạng nguồn tạo thành bởi phép cộng giữa thanh ghi cơ sở và hằng số

  VD: LOAD Ri, X(Rbase); Ri<- M[X+Rbase];

Câu 11: Trình bày các nhóm lệnh thông dụng (Phân loại lệnh) và cho VD bằng lệnh hợp ngữ.

- Các nhóm lệnh thông dụng:

  - Nhóm lệnh vận chuyển dữ liệu: là sao chép dữ liệu từ vị trí này đến vị trí khác.

VD:

LOAD Ri #1000; Ri <-1000

Nạp giá trị 1000 vào thanh ghi Ri

STORE 1000,Ri; M[1000] <- Ri

Lưu nội dung thanh ghi Ri ra ô nhớ 1000

MOVE Ri, Rj; Ri <- Rj;
Sao chép nội dung dữ liệu thanh ghi Rj sang Ri
PUSH: đẩy dữ liệu vào stack;
POP: lấy dữ liệu ra khỏi stack;

- Nhóm lệnh toán số học và logic: được sử dụng để thực hiện các phép tính số học và logic trên nội dung thanh ghi/ô nhớ.
  VD:
  ADD R1, R2, R3; R1 <- R2+R3;
  SUBSTRACT R1, R2, R3; R1 <- R2 – R3;
  INCREMENT R1; R1 <- R1+1;
  NOT R1; R1 <- !(R1); (lệnh logic)
- Nhóm lệnh chuyển điều khiển chương trình

Thay đổi thứ tự thực hiện các lệnh trong chương trình (gây rẽ nhánh hoặc nhảy) bằng cách thay đổi nội dung của bộ đếm chương trình PC.
Các Cờ của ALU giúp xác định điều kiện rẽ nhánh hoặc nhảy.

- Nhóm lệnh điều khiển bit:

Thay đổi các bit trạng thái và điều khiển các thanh ghi.

Câu 12: Trình bày kỹ thuật đường ống 5 đoạn. Tranh chấp dữ liệu và khắc phục.

- Là phương pháp thực hiện lệnh cho phép đồng thời thực hiện nhiều lệnh, giảm thời gian trung bình thực hiện lệnh và tăng hiệu năng xử lý lệnh CPU.

Mô tả: 5 giai đoạn thực hiện lệnh
+ Vẽ hình
+ IF: đọc lệnh, lệnh được đọc từ bộ nhớ về CPU.
+ ID: CPU giải mã lệnh.
+ IE: Thực hiện lệnh: CPU thi hành nhiều lệnh, lấy DL bộ nhớ và các t.bị ngoại vi.
+ WB: Lưu dữ liệu vào thanh ghi.
+NEXT: Chuyển lệnh tiếp theo.

- Nguyên tắc : thực hiện đồng thời các lệnh.
- Tranh chấp dữ liệu: là một trong các vấn đề lớn của cơ chế ống lệnh và tranh chấp dữ liệu kiểu đọc sau ghi (RAW) là dạng xung đột dữ liệu hay gặp.
- Cách khắc phục :

+ Nhận dạng RAW khi nó xảy ra.
+ Tạm dừng ống lệnh cho đến khi lệnh phía trước hoàn thành.
+ Chèn các lệnh NO-OP hoặc các lệnh độc lập vào giữa.
+ Sử dụng phần cứng nhận dạng và dự đoán RAW.
-VD: ADD R1,R2,R3;
NO-OP;
NO-OP;
NO-OP;
SUB R1,R2,R3;

Câu 13: Trình bày cấu trúc lệnh và các dạng toán hạng

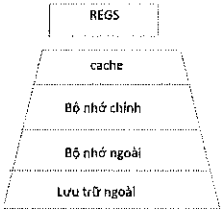
-Cấu trúc lệnh:

Mã lệnh Toán hạng

- Mã lệnh: Từ mã nhị phân dùng để phân biệt các lệnh với nhau.
- Toán hạng: Dữ liệu hoặc địa chỉ của lệnh.
- Chức năng: các lệnh của chương trình đc lưu trong bộ nhớ và lần lượt đc CPU đọc, giải mã và thực hiện. Từ đó CPU có khả năng lập trình đc để thực hiện công việc có ích cho người dùng.
- Phân loại lệnh: 0 toán hạng
  1 toán hạng
  2 toán hạng
  3 toán hạng.

(phân tích loại lệnh ở **câu 9**)

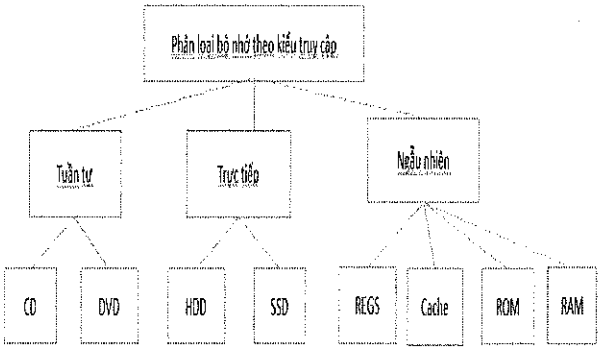
Câu 14: Trình bày hệ thống bộ nhớ phân cấp trong các hệ thống máy tính



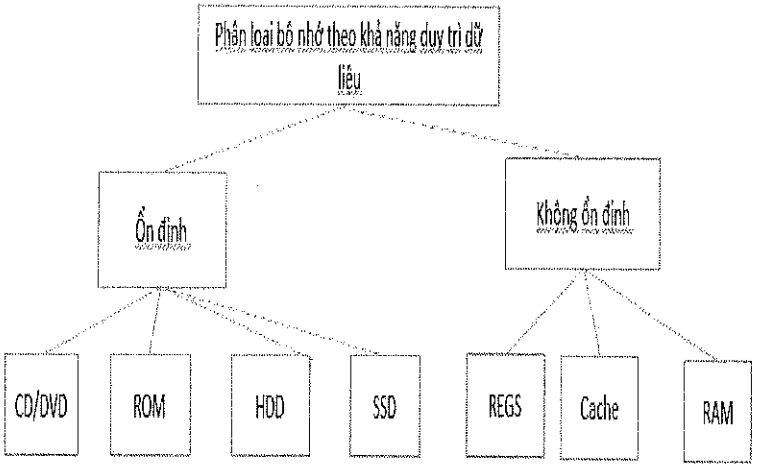
- Hệ thống bộ nhớ phân cấp:
- Độ trễ dung lượng của các thành phần tăng theo chiều từ REGs đến lưu trữ ngoài .
- Tốc độ và giá thành thăng theo chiều ngược lại
- Các thanh ghi có dung lượng nhỏ nhất (vài chục byte-> vài KB) nhưng có tốc độ cao nhất nhưng giá thành đắt nhất. Thường dùng để lưu các toàn hàng vào và kết quả ra của các lệnh.
- Bộ nhớ cache dung lượng tương đối nhỏ (vài chục KB->MB). Tốc độ khá cao nhưng giá đắt. Cache đoán trước được nhu cầu lệnh và dữ liệu của CPU-> tăng tốc xử lý .
- Bộ nhớ chính (ROM,RAM): dung lượng khá lớn (256MB-> 4GB) nhưng tốc độ truy cập chậm và có gia rẻ. Thường dùng để lưu lệnh và dữ liệu của hệt hống và người dùng.
- Bộ nhớ ngoài và bộ nhớ lưu trữ ngoài có giá rẻ, dung lượng lớn nhưng tốc độ truy cập chậm nhất. Thường dùng để lưu dữ liệu lâu dài.
- Vài trò:
  + Tăng hiệu năng nhờ dung hoà được CPU có tốc độ cao với các bộ nhớ có tốc độ thấp (CPU-> Cache->MEM)
  +Giảm giá thành: các thành phần có tốc độ và giá cao được dùng với dung lượng nhỏ và ngược lại. Nhờ đó giảm được giá thành mà vẫn đảm bảo được tốc độ, cao.

Câu 15: Các phương pháp phân loại bộ nhớ;

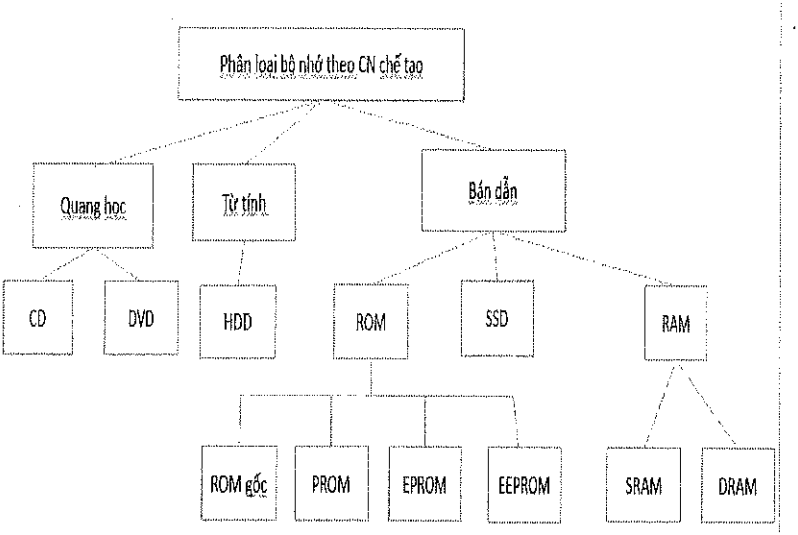
+ Dựa trên kiểu truy nhập:



+Dựa trên khả năng duy trì dữ liệu:



+ Dựa trên công nghệ chế tạo:

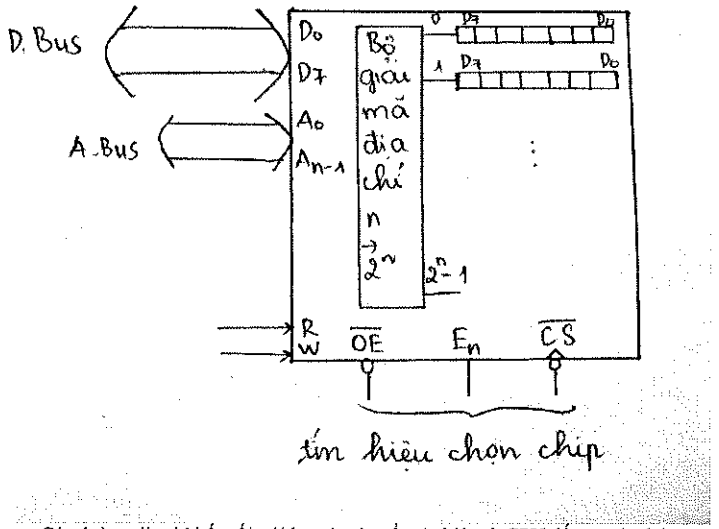


- ROM: là bộ nhớ chỉ đọc, việc ghi thông tin vào ROM chỉ có thể thực hiện đc bằng các thiết bị chuyên dùng và các thông tin đc ghi sẵn thường là cấu hình máy, BIOS. Ngoài ra, ROM là bộ nhớ ổn định, dữ liệu không bị mất khi bị mất nguồn.
  +ROM gốc: ROM sử dụng tia cực tím để ghi thông tin
  +PROM: ROM có thể lập trình được bằng bộ lập trình PROM.
  +EPROM: thông tin trong ROM có thể được xóa bằng tia cực tím có cường độ cao.

+EEPROM: có thể xóa đc bằng điện và ghi bằng phần mềm chuyên dụng.

- RAM: Bộ nhớ không ổn didjnhvaf cho phép truy cập ngẫu nhiên, thường dùng để lưu thông tin của hệ thống và người dùng.
  + SRAM: cấu tạo dựa trên mạch lật. Thông tin ổn định và không cần "làm tươi: định kỳ. Tốc độ truy cập nhanh hơn DRAM.
  + DRAM: cấu tạo dựa trên tụ điện. Thông tin sẽ dần bị mất do tụ phóng điện nên cần "làm tươi". DRAM tuy có tốc độ thấp hơn nhưng có cấu trúc gọn nhẹ và giá thành rẻ hơn.

Câu 16: Tổ chức và hoạt động của IC nhớ:



- Các đường địa chỉ kết nối với bus A, chuyển tín hiệu từ CPU đến mạch nhớ.
- Bộ giải mã địa chỉ sử dụng tín hiệu địa chỉ để chọn ra và kích hoạt ô nhớ hoặc dòng nhớ cần truy nhập.
- Các đường DL kết nối với D bus: truyền dữ liệu từ bộ nhớ về CPU và ngược lại.

Câu 17: Bộ nhớ ROM:

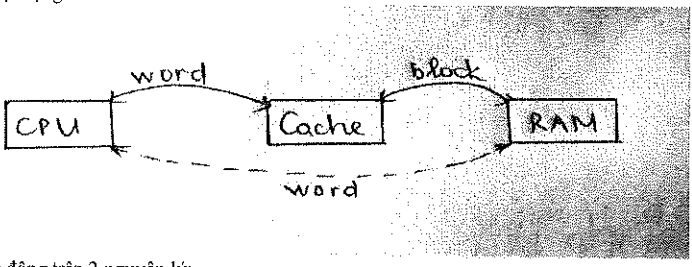
- Đặc điểm:
- Là bộ nhớ chỉ đọc
- Là bộ nhớ ổn định
- Là bộ nhớ bán dẫn: mỗi ô nhớ của ROM là một cổng bán dẫn
- Ứng dụng:
- Thường được sử dụng để lưu chương trình khởi động của máy tính
- Phân loại ROM:
- ROM gốc: sử dụng tia cực tím để ghi thông tin.
- PROM: thông tin được ghi vào PROM nhờ bộ lập trình PROM.
- EPROM: thông tin trong EEPROM có thể xóa được sử dụng tia cực tím ở cường độ cao.
- EEPROM: là loại ROM tiên tiến nhất, có thể xóa đc bằng điện và ghi thông tin sử dụng phần mềm chuyên dụng.

Câu 18: Bộ nhớ RAM:

- Đặc điểm:
- Là bộ nhớ truy cập ngẫu nhiên:
+ Mỗi ô nhớ của RAM có thể được truy nhập không theo trật tự nào.
+ Tốc độ truy nhập ô nhớ là tương đương
- Là bộ nhớ không ổn định: thông tin sẽ bị mất khi mất nguồn
- Là bộ nhớ bán dẫn: mỗi ô nhớ của RAM là một cổng bán dẫn
- Sử dụng để lưu thông tin hệ thống và người dùng
+ Thông tin hệ thống: phần cứng và hệ điều hành
+ Thông tin người dùng: các chương trình ứng dụng và dữ liệu
- Phân loại (2 loại)
- RAM tĩnh: (static RAM)
+ Mỗi bit SRAM là một mạch lật flip-flop
+ Thông tin trong bit luôn ổn định
+ Nhanh nhưng đắt hơn DRAM
- RAM động (dynamic RAM)
- + Mỗi bit DRAM dựa trên một tụ điện
  + Thông tin lưu trong các bit không ổn định và phải được "làm tươi" định kỳ.
  + Chậm nhưng rẻ hơn SRAM.

Câu 19: Bộ nhớ cache

- Khái niệm: là một thành phần trong hệ thống nhớ phân cấp MT, đóng vai trò trung gian, chuyển giữ liệu từ bộ nhớ chính đến CPU và ngược lại
- Hoạt động:

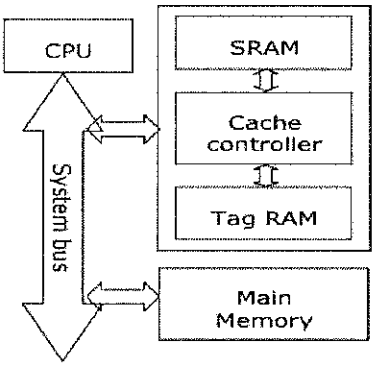


cache hoạt động trên 2 nguyên lý:

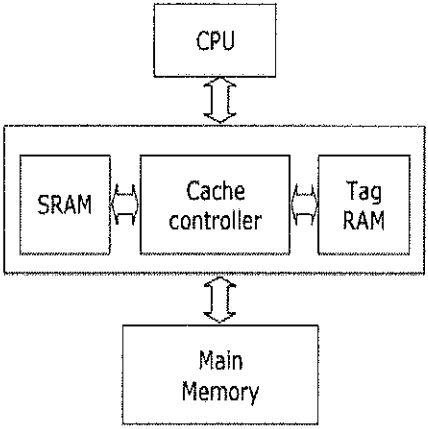
- Nguyên lý lân cận về không gian
  + nếu một ô nhớ được truy cập thì xác suất các ô nhớ liền kề với nó trong tương lai gần là rất cao
  + lân cận về thời gian được AD cho nhóm lệnh hoặc DL có tính tuần tự cao.
- Nguyên lý lân cận về thời gian
  + Nếu một ô nhớ đang đc truy nhập thì xác suất nó đc truy nhập lại trong tương lai gần là rất cao.
  + lân cận về thời gian được AD cho DL và nhóm lệnh trong vòng lặp.

Câu 20:

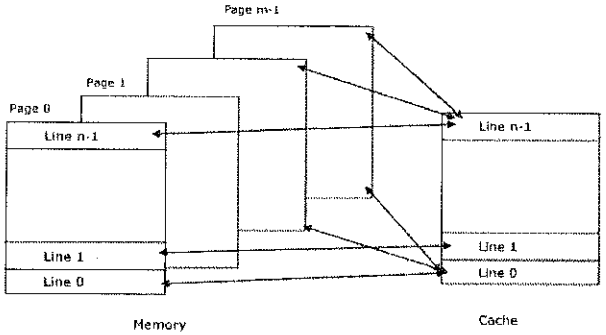
- Kiến trúc cache Look aside
- Cache và bộ nhớ chính cùng kết nối với bus hệ thống
- Cache và bộ nhớ chính thấy chu kì bus của CPU tại cùng 1 thời điểm



- Kiến trúc cache Look through
- Cache nằm giữa CPU và bộ nhớ chính
- Cache "thấy" chu kỳ bus của CPU trước sau đó chuyển chu kỳ bus cho bộ nhớ chính
- Ưu điểm
  + Hit nhanh
- Nhược điểm
  + Thiết kế phức tạp
  + Đắt tiền
  + Miss chậm

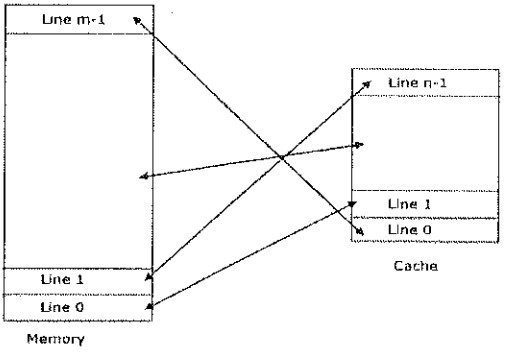


Câu 21: Phương pháp ánh xạ trực tiếp của bộ nhớ cache:



- Cache chia thành n khối hoặc dòng, từ $Line_0$ đến $Line_{n-1}$
- Bộ nhớ chính:
  - + Chia thành m trang, từ $page_m$ đến $page_{m-1}$
  - + Mỗi một bộ nhớ có kích thước bằng cache
  - + Mỗi trang có n dòng từ $Line_0$ đến $Line_{n-1}$
- Quy tắc ánh xạ
  - + $Line_0$ của ($page_0$ đến $page_{m-1}$) ánh xạ đến $Line_0$
  - + $Line_1$ của ($page_0$ đến $page_{m-1}$) ánh xạ đến $Line_1$
- Đặc điểm:
  - Ưu điểm:
    - Đơn giản, dễ cài đặt
    - Nhanh do không mất nhiều thời gian tìm kiếm.
  - Nhược điểm:
    - Xung đột dòng cache nếu có nhiều dòng cùng sử dụng.
    - Lãng phí dung lượng cache.
    - Hệ số Hit thấp.

Câu 22: Phương pháp ánh xạ kết hợp đầy đủ (ánh xạ liên kết)



- Cache được chia thành n khối hoặc dòng, từ $Line_0$ đến $Line_{n-1}$
- Bộ nhớ:

+ Được chia thành m khối hoặc dòng, từ $Line_0$ đến $Line_{m-1}$
+ Kích thước mỗi dòng cache bằng kích thước một dòng bộ nhớ
+ Số dòng trong bộ nhớ rất lớn so với số dòng của cache (m>n)

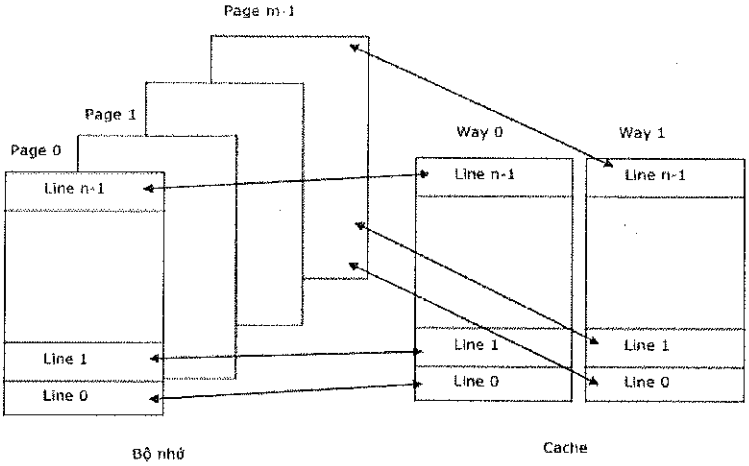
- Quy tắc ánh xạ:
  + Một dòng trong bộ nhớ có thể ánh xạ vào một dòng bất kỳ trong cache
  + $Line_i$ trong bộ nhớ có thể ánh xạ vào $Line_j$ trong cache
- Đặc điểm:
  - Ưu điểm:
    - Giảm xung đột dòng.
    - Sử dụng không gian cache hiệu quả.
    - Hệ số Hit cao.
  - Nhược điểm:
    - Thiết kế và cài đặt khó khăn.
    - Tìm kiếm dòng cache mất nhiều thời gian.
    - Chỉ phù hợp với cache dung lượng nhỏ.

Câu 23: Phương pháp ánh xạ kết hợp tập hợp (tập kết hợp / liên kết tổ hợp) của tổ chức bộ nhớ cache.

- Phương pháp ánh xạ tập ánh xạ liên kết nhóm.
- Cache:
  + Được chia thành k đường (way) và có kích thước bằng nhau.
  + Mỗi đường được chia thành n dòng, từ $Line_0$ đến $Line_{n-1}$
- Bộ nhớ:
  + Được chia thành m trang từ $page_0$ đến $page_{m-1}$
  + Mỗi trang có n dòng từ $Line_0$ đến $Line_{n-1}$
  + Kích thước 1 trang bộ nhớ bằng kích thước 1 đường cache
- Ánh xạ:
  + Ánh xạ trang đến đường (ánh xạ mềm dẻo). Một trang bộ nhớ có thể ánh xạ đến 1 đường bất kỳ của cache
  + Ánh xạ dòng của trang đến dòng của đường (ánh xạ cố định)

$Line_0$ của $page_i$ của bộ nhớ ánh xạ đến $Line_0$ của way_j của cache
$Line_1$ của $page_i$ của bộ nhớ ánh xạ đến $Line_1$ của way_j của cache
$Line_{n-1}$ của $page_i$ của bộ nhớ ánh xạ đến $Line_{n-1}$ của way_j của cache

- Đặc điểm
  - Ưu điểm:
    - Nhanh do ánh xạ trực tiếp là chủ yếu.
    - Giảm xung đột dòng.
    - Sử dụng không gian cache hiệu quả.
    - Hệ số Hit cao.
  - Nhược điểm:
    - Phức tạp.
    - Khó quản lý do cache bị chia thành nhiều đường.



Câu 24. Trình bày các chính sách thay thế dòng của bộ nhớ cache.

- Định nghĩa: Là phương thức xác định dòng cache bị thay thế bằng dòng cache mới nhằm đạt hệ số Hit cao nhất.
- Nguyên nhân: Các lệnh, dữ liệu CPU cần không có trong cache nên cache cần đọc và lấy chúng từ bộ nhớ. Mà dung lượng (số dòng) cache có giới hạn nên để nạp dòng mới cần thay thế dòng cũ.
- Các phương pháp thay thế

	Thay thế ngẫu nhiên	Thay thế kiểu vào trước ra trước	Thay thế dòng ít sử dụng gần đây nhất
Định nghĩa	Dòng cache được chọn để thay thế là ngẫu nhiên, không theo quy luật nào cả.	Dòng cache được nạp trước sẽ bị thay thế trước.	Dòng cache được thay thế là dòng ít được sử dụng gần đây nhất.
Ưu điểm	Đơn giản, dễ cài đặt.	Hệ số Miss thấp hơn ngẫu nhiên do có xét đến yếu tố lân cận thời gian.	Hệ số Miss thấp nhất do có xét đến các dòng cache đang sử dụng.
Nhược điểm	Hệ số Miss cao do không xem xét những dòng cach đang được sử dụng.	- Phức tạp, khó cài đặt do cần thiết bị để theo dõi thứ tự dòng được nạp. - Chưa thực sự xét các dòng cache đang sử dụng.	- Thiết kế và cài đặt phức tạp. - Cần thiết bị để theo dõi tần suất sử dụng dòng cache.

Câu 25. Các phương thứ ghi dữ liệu trong bộ nhớ cache

a. Trường hợp Hit - mẩu tin cần ghi có trong cache

- Ghi thẳng (Write Through):
  - Mẩu tin cần ghi được lưu đồng thời ra cache và bộ nhớ chính.

- Đảm bảo tính nhất quán dữ liệu giữa cache và bộ nhớ chính.
- Nhược điểm:
  - Chậm.
  - Tốn băng thông nếu ghi nhiều mẩu tin có kích thước nhỏ.

- Ghi trễ (Write Back):
  - Mẩu tin được ghi ra cache trước. Khi dòng cache chứa mẩu tin bị thay thế thì nó được ghi ra bộ nhớ chính.
  - Tăng tốc độ do ghi ra cache nhiều lần còn ghi ra bộ nhớ chính chỉ 1 lần.
  - Giảm băng thông sử dụng.

b. Trường hợp Miss - mẩu tin cần ghi không có trong cache.

- Ghi có đọc lại (Write Allocate):
  - Mẩu tin được ghi ra bộ nhớ chính trước, sau đó dòng nhớ chứa mẩu tin được đọc lại vào cache.
  - Giảm Miss đọc kế tiếp do sử dụng nguyên tắc lân cận về thời gian.
- Ghi không đọc lại (Write Non-Allocate): mẩu tin chỉ được ghi ra bộ nhớ chính mà không được đọc lại vào cache.

Câu 26. Trình bày các phương pháp đọc cache.

a. Trường hợp Hit
- Mẩu tin được đọc từ cache vào CPU và bộ nhớ chính không tham gia.
- Thời gian truy nhập mẩu tin = Thời gian CPU truy nhập cache.

b. Trường hợp Miss
- Mẩu tin được chuyển từ bộ nhớ chính vào cache trước, sau đó được đọc từ cache vào CPU.
- Thời gian truy nhập mẩu tin = Thời gian truy nhập cache + Thời gian truy nhập bộ nhớ chính.

Câu 27: Trình bày các phương pháp tăng Hệ số Hit trong cache.

a. Tăng dung lượng cache
- Tăng dung lượng làm tăng số dòng bộ nhớ lưu trong cache.
- Giảm Miss bắt buộc do không gian cache lớn hơn, có độ bao phủ tốt hơn, giảm tần suất thay thế dòng cache.
- Hỗ trợ đa nhiệm, xử lý song song các hệ thống CPU nhiều nhân.
- Bị chậm do không gian tìm kiếm rộng hơn.
- Lãng phí dung lượng cache do có thể có dòng cache không được sử dụng.

b. Chia tách cache
- Tách cache thành 2 phần: Cache lệnh và Cache dữ liệu.
- Dữ liệu và lệnh có tính lân cận khác nhau: dữ liệu - không gian, lệnh - thời gian nên tách cache giúp tăng hiệu năng.
- Cache lệnh chỉ cần thao tác đọc trong khi cache dữ liệu cần cả đọc và ghi nên tách cache giúp tối ưu tốt hơn.
- Giảm xung đột tài nguyên cho CPU pipeline do tách cache hỗ trợ nhiều lệnh truy nhập đồng thời.

c. Tạo cache nhiều mức
- Chia cache thành nhiều mức với kích thước tăng dần và tốc độ giảm dần sẽ cải thiện hiệu năng do cache dung hòa tốt hơn tốc độ CPU và bộ nhớ chính.

Câu 28: Trình bày các tham số hiệu năng của cache

- Hiệu năng của cache được đánh giá theo thời gian truy nhập trung bình của CPU đến hệ thống nhớ:

$$T_{truy\ nhập\ trung\ bình} = \text{Cache Hit} + \text{Cache Miss} * T_{truy\ nhập\ bộ\ nhớ}$$
$$= T_{cache} + (1 - \text{Cache Hit}) * T_{truy\ nhập\ bộ\ nhớ}$$

trong đó:

+ $T_{truy\ nhập\ trung\ bình}$: Thời gian truy nhập trung bình của CPU đến hệ thống nhớ.
+ Cache Hit: Hệ số Hit của cache.
+ Cache Miss: Hệ số Miss của cache.
+ $T_{truy\ nhập\ bộ\ nhớ}$: Thời gian truy nhập hệ thống nhớ.

+ T_{cache} : Thời gian truy nhập cache.

Câu 29: Trình bày đặc điểm của đĩa từ và các loại đĩa từ

a. Đặc điểm của đĩa từ

- Là 1 loại thiết bị lưu trữ thuộc loại bộ nhớ ổn định dữ liệu.
- Ngoài ra đĩa từ cũng là 1 bộ nhớ kiểu khối có dung lượng lớn, dùng để lưu trữ thông tin lâu dài dưới dạng các file.
- Để lưu được thông tin, đĩa từ sử dụng các đĩa nhựa hoặc kim loại có lớp phủ bột từ trên bề mặt.

b. Các loại đĩa từ

	Đĩa từ mềm	Đĩa từ cứng
Chất liệu	Nhựa, dễ hư hỏng	Kim loại hoặc thủy tinh
Dung lượng	Nhỏ (tính bằng KB)	Lớn (tính bằng GB - TB)
Tốc độ	Chậm (300-360 rpm)	Nhanh (3600-10000 rpm)
Kích thước	8", 5.25", 3.5"	2.5", 3.5", 5.25"

Câu 30: . Trình bày đặc điểm cơ bản của các giao diện ghép nối đĩa từ

a. Chuẩn ATA/ IDE/ PATA

- Sử dụng cáp dẹt 40/80 sợi để ghép nối ổ cứng với bảng mạch chính của máy tính.
- Mỗi cáp hỗ trợ ghép nối với 2 ổ đĩa gọi là chủ - tớ (master - slave).
- Băng thông đường truyền: 16 bit.
- Các mức thông lượng: 16, 33, 66, 100, 133 MB/s.

b. Chuẩn SATA

- Sử dụng đường truyền nối tiếp tốc độ cao qua 2 đôi dây với bộ điều khiển SATA sử dụng chuẩn AHCI.
- Truyền dữ liệu nhanh và hiệu quả hơn ATA, hỗ trợ cắm nóng (cắm thêm ổ cứng mà không cần tắt nguồn).
- Thông lượng cao hơn nhiều so với ATA: 150/300/600 MB/s với lần lượt các chuẩn SATA 1/2/3.

c. Chuẩn SCSI

- Là 1 tập các chuẩn về kết nối vật lý và truyền dữ liệu giữa máy tính và thiết bị ngoại vi, thường dùng cho máy chủ.
- Tất cả các thiết bị SCSI đều kết nối đến BUS SCSI, mỗi BUS có thể kết nối 8-16 thiết bị.
- Tốc độ truyền dữ liệu và tính ổn định rất cao, hỗ trợ cắm nóng.
- Tốc độ truyền dữ liệu:
  + Chuẩn cũ: 5, 10, 20, 40 MB/s.
  + Chuẩn mới: 160, 320, 640 MB/s.

Câu 31 : Trình bày đặc điểm của đĩa quang và các loại đĩa quang.

a. Đặc điểm

- Sử dụng ánh sáng để đọc ghi thông tin trên đĩa.
- Được chế tạo bằng plastic với 1 mặt được tráng lớp nhôm mỏng để phản xạ laser, mặt đĩa quang được khắc rãnh với mức lõm của rãnh để biểu diễn các bit thông tin.

b. Các loại đĩa quang

- Sơ đồ:
- CD:
  - Đặc điểm chung:
    - Dung lượng nhỏ: 700MB hoặc 80' lưu âm thanh.
    - Tốc độ chậm: 150 KB/s * hệ số nhân.
    - Lưu dữ liệu, âm thanh và phim ảnh chất lượng thấp.
  - CD-ROM: được ghi sẵn nội dung khi sản xuất, chỉ có thể đọc.
  - CD-R: có thể ghi lại 1 lần bởi người sử dụng, sau đó chỉ có thể đọc.
  - CD-RW: có thể xóa và ghi lại thông tin nhiều lần.
- DVD:
  - Đặc điểm chung:

- Dung lượng lớn: 4.7GB với đĩa 1 mặt và 8.5 GB với đĩa 2 mặt.
- Tốc độ cao: 1350 KB/s * hệ số nhân.
- Lưu dữ liệu, âm thanh và phim ảnh chất lượng cao.

- DVD-ROM: đĩa DVD chỉ đọc.
- DVD-R: đĩa có thể ghi 1 lần.
- DVD-RW: đĩa có thể ghi lại nhiều lần.
- HD-DVD: đĩa DVD có mật độ cao, dung lượng từ 15-30GB, tốc độ truy nhập cao.
- Bluray-DVD: đĩa DVD có mật độ siêu cao, dung lượng từ 30-50GB

Câu 32 : Nguyên lý hoạt động của chuột quang

- Một điốt phát ánh sáng đỏ qua ống kính chiếu xuống mặt phẳng di chuột. Ánh sáng phản xạ từ mặt phẳng di chuột quay ngược trở lại chuột
- Camera đặt phía dưới chuột liên tục chụp ảnh bề mặt di chuột nhờ ánh sáng phản xạ. Tốc độ khoảng 1500 ảnh/ 1s
- Bộ điều khiển chuột xử lý và so sánh các bức ảnh kề nhau để tìm ra sự di chuyển của chuột
- Tín hiệu biểu diễn di chuyển chuột được gửi tới máy tính để xử lý tiếp theo

Câu 33 : RAID

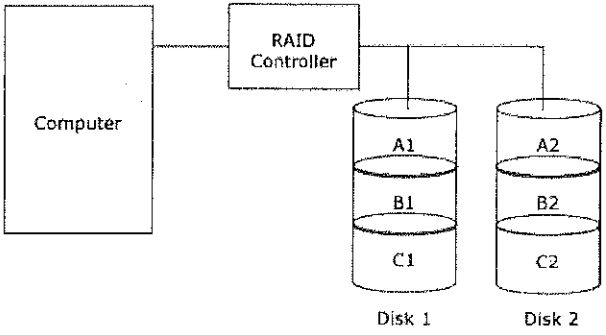
Raid là :

- RAID (Redundant Array of Independent Disks) là công nghệ tạo các thiết bị lưu trữ tiên tiến trên cơ sở đĩa cứng những mục đích sau:
  - Hiệu năng, tốc độ cao (high performance/ speed)
  - Độ tin cậy cao (high reliability)
  - Dung lượng lớn (large volume)
- RAID là một tập/ mảng các HDD nhưng được HDH coi như 1 ổ đĩa logic
- Các đĩa cứng theo chuẩn SATA và SCSI mới hỗ trợ tạo RAID
- Dữ liệu được phân tán trên các đĩa vật lý
- Đĩa dư thừa được sử dụng để lưu trữ thông tin parity => đảm bảo khôi phục dữ liệu

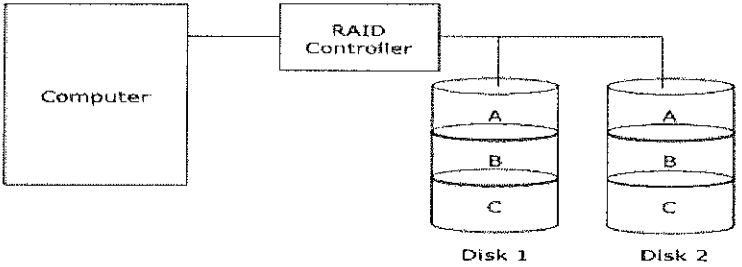
Kỹ thuật cơ bản tạo RAID :

- 2 kỹ thuật cơ bản được sử dụng trong RAID:

- Disk stripping (tạo lát đĩa):
  - Dữ liệu được chia thành các khối và mỗi khối được ghi đồng thời vào một đĩa độc lập
  - Sau đó, các khối dữ liệu có thể được đọc từ HDD một cách đồng thời
    ⇒ Cải thiện tốc độ truy cập



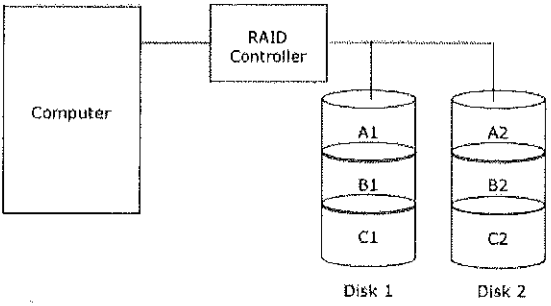
- Disk mirroring (soi gương):
  - Dữ liệu được chia thành các khối và mỗi khối được ghi vào một số đĩa
  - Tại thời điểm bất kì, luôn có nhiều hơn 1 bản sao dữ liệu => độ tin cậy tăng



Câu 34 : Trình bày 3 loại RAID

RAID 0 :

- Các đặc điểm:
  - Dựa trên kĩ thuật disk stripping (đọc/ ghi song song), cần tối thiểu 2 đĩa
  - Dữ liệu được phân bố trên các đĩa trong mảng
  - Tối thiểu cần 2 HDD
- Ưu:
  - Tốc độ nhanh
  - Đáp ứng tốt các hệ thống nhu cầu I/O cao
  - Dung lượng là tổng của tất cả các đĩa
- Nhược:
  - Độ tin cậy như một đĩa

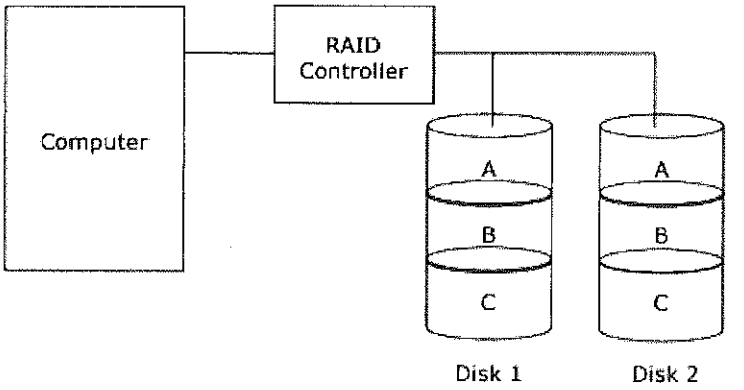


RAID 1 :

- Các đặc điểm:
  - Dựa trên kĩ thuật disk mirroring (nhiều bản sao)
  - Tính dư thừa có được đơn giản bằng cách sao tất cả dữ liệu
  - Tối thiểu cần 2 HDD
  - Dữ liệu cũng phân mảnh (data stripping) như RAID 0 nhưng mỗi mảnh logic được ánh xạ tới 2 đĩa vật lý khác nhau

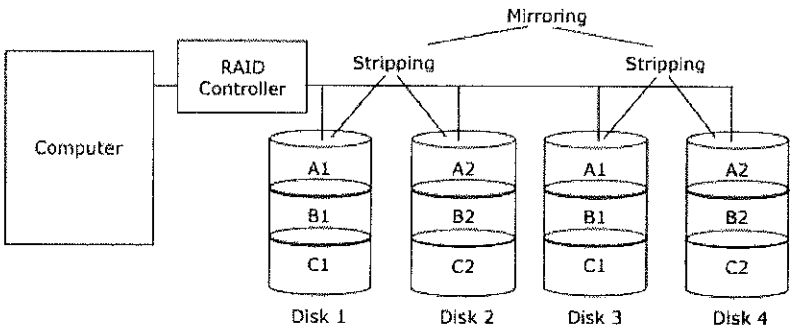
=> Mỗi đĩa trong mảng có một bản sao cùng dữ liệu (mirror)

- Ưu:
  - Độ tin cậy cao
- Nhược:
  - Dung lượng thực sự bằng 1/2 tổng số đĩa
  - Chi phí cao



RAID 10 :

- Các đặc điểm:
  - Tối thiểu cần 4 HDD
  - Dựa trên kĩ thuật disk mirroring và stripping
- Ưu:
  - Nhanh hơn so với một đĩa
  - Tin cậy hơn so với một đĩa
- Nhược:
  - Dung lượng bằng một nửa dung lượng tổng số đĩa



Câu 35 : NAS và đặc điểm

- NAS là server chuyên dụng cho lưu trữ
- NAS được kết nối vào mạng và cung cấp các dịch vụ lưu trữ qua mạng
- NAS dựa trên nền tảng là RAID tốc độ cao, dung lượng lớn, độ tin cậy cao
- NAS có thể cung cấp dịch vụ lưu trữ cho hầu hết tất cả các loại server có cấu hình phần cứng khác nhau và chạy trên các hệ điều hành khác nhau

Câu 36 : SAN và đặc điểm

- SAN là một mạng các server chuyên dụng cung cấp dịch vụ lưu trữ
- SAN thường cung cấp dịch vụ lưu trữ với đặc điểm:
  - Tốc độ truy cập rất cao

- Dung lượng cực kỳ lớn
- Độ tin cậy rất cao:
  - An toàn dữ liệu cục bộ
  - An toàn dữ liệu với các bản copy được đồng bộ ở khoảng cách xa về địa lý

➢ SAN thường được tổ chức dưới dạng hệ thống file phân tán (Distributed File System)

Câu 37 : Nguyên lý hoạt động của máy in laser

- Máy in laser hoạt động dựa trên nguyên tắc chụp ảnh điện tử bằng tia laser:
- Trống cảm quang được nạp 1 lớp điện tích nhờ 1 điện cực
- Tia laser từ nguồn sáng laser đi qua 1 gương quay và bộ điều chế tia được điều khiển bởi tín hiệu cần in đến mặt trống
- Ánh sáng laser làm thay đổi mật độ điện tích trên mặt trống; mật độ điện tích trên mặt trống thay đổi theo tín hiệu cần in
- Khi trống cảm quang quay đến hộp mực thì điện tích trên trống hút các hạt mực được tích điện trái dấu. Các hạt mực dính trên trống biểu diễn âm bản của văn bản/thông tin cần in
- Giấy từ khay được kéo lên cũng được điện cực nạp điện tích trái dấu với điện tích của mực nên hút các hạt mực khỏi trống cảm quang
- Giấy tiếp tục đi qua trống sấy nóng làm các hạt mực chảy ra và bị ép chặt vào giấy

Câu 38: Nguyên lý hoạt động của màn hình LCD

➢ Bản thân tinh thể lỏng không thể phát sáng
- Tuy nhiên, chúng có thể điều khiển lượng ánh sáng đi qua theo nhiệt độ và điện

➢ 2 loại LCD dựa vào phương pháp điều khiển:
- LCD ma trận thụ động (Passive matrix):
  - Sử dụng ma trận/ lưới để định nghĩa từng điểm ảnh bởi hàng và cột của nó.
  - Một điểm ảnh (giao của hàng và cột) được kích hoạt khi điện áp được đặt vào cột và dòng tương ứng được tiếp đất
- LCD ma trận chủ động (Active matrix):
  - Sử dụng một TFT (Thin Film Transistor) để điều khiển một phần tử LCD
  - TFT hoạt động như bộ chuyển mạch

➢ TFT LCD là thiết bị được điều khiển bởi các tín hiệu điện
➢ Lớp tinh thể lỏng ở giữa 2 lớp trong suốt chứa các điện cự ITO (Indium Tin Oxide)
➢ Các phân tử tinh thể lỏng được sắp xếp theo các hướng khác nhau theo sự thay đổi điện áp đặt vào các điện cực ITO
➢ Hướng của các phân tử tinh thể lỏng ảnh hưởng trực tiếp tới cường độ ánh sáng đi qua và nó gián tiếp điều khiển mức sáng/ tối (còn gọi là mức xám : grayscale) của ảnh hiển thị
➢ Màu của hình ảnh được tạo bởi một lớp lọc màu
➢ Mức xám của điểm ảnh được xác định bởi mức điện áp của tín hiệu video đưa vào

Câu 39 : PCI Express:

Đặc điểm PCI:

➢ PCI (Peripheral Component Interconnect) bus được Intel phát triển năm 1993
➢ Băng thông: 32 hoặc 64 bit
➢ Tốc độ truyền dữ liệu:
- 133 MB/s (32bit at 33MHz) n 266 MB/s (32bit at 66MHz or 64bit at 33MHz)
- 533 MB/s (64bit at 66MHz)

Hoạt động PCI:

➢ Một giao dịch PCI (một phiên truyền dữ liệu trên bus PCI – transaction) thường gồm 3 giai đoạn:
- Arbitration (pha tùy chọn): khởi tạo giao dịch
- Address (pha địa chỉ): xác định địa chỉ bên tham gia giao dịch
- Data (pha dữ liệu): truyền dữ liệu

Đặc điểm PCIe:

- PCIe được cấu trúc từ các liên kết nối tiếp điểm tới điểm
- Một cặp liên kết nối tiếp (theo 2 chiều ngược nhau) tạo thành một luồng(lane)
- Các luồng được định tuyến qua một bộ chuyển mạch (crossbar switch) trên bảng mạch chính
- Các khe PCIe vật lý có thể chứa từ 1 – 32 làn

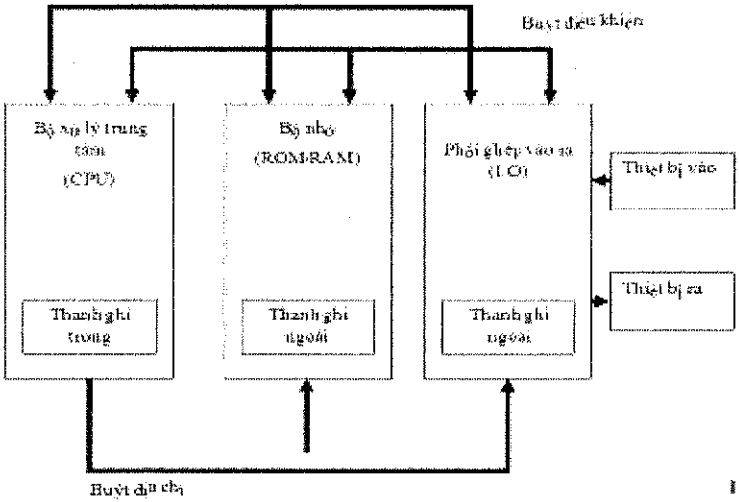
Hoạt động PCIe:

➢ PCIe sử dụng giao thức truyền nối tiếp và tránh được vấn đề lệch thời gian (time skew) – một trong các yếu tố làm giảm tốc độ:
- Các bus song song (ISA, PCI, AGP) yêu cầu tất cả các bit dữ liệu cần truyền tới điểm đích cùng thời điểm
- Vì vấn đề lệch thời gian, các bit của khối dữ liệu cần truyền có thể không đến đích cùng thời gian, sẽ gây khó khăn trong việc phục hồi từ dữ liệu cuối cùng
- Đối với bus nối tiếp, không có vấn đề về thời gian vì chúng không yêu cầu mọi bit của khối dữ liệu cần truyền tới đích cùng thời điểm

VI XỬ LÝ (KTMT)

• Câu hỏi loại 2 điểm

2.1.Trình bày sơ đồ khối của hệ vi xử lý tiêu biểu và giải thích ngắn gọn vai trò của các khối chức năng chính?



Trong sơ đồ này ta thấy rõ các khối chức năng chính của hệ vi xử lý gồm:

_Khối xử lý trung tâm (CPU)

_Bộ nhớ bán dẫn (ROM-RAM)

_Khối phối ghép với các thiết bị ngoại vi (I/O)

_Các bus truyền thông tin.

Ba khối chức năng đầu liên hệ với nhau thông qua qập các đường dây để truyền tín hiệu gọi chung là *Bus hệ thống.* Bus hệ thống bao gồm 3 bus thành phần ứng với các tín hiệu địa chỉ, dữ liệu và điều khiển ta có *bus địa chỉ, bus dữ liệu và bus điều khiển.*

CPU đóng vai trò chủ đạo trong hệ vi xử lý.

Bộ nhớ bán dẫn hay còn gọi là *bộ nhớ trong* là một bộ phận khác rất quan trọng của hệ vi xử lý.

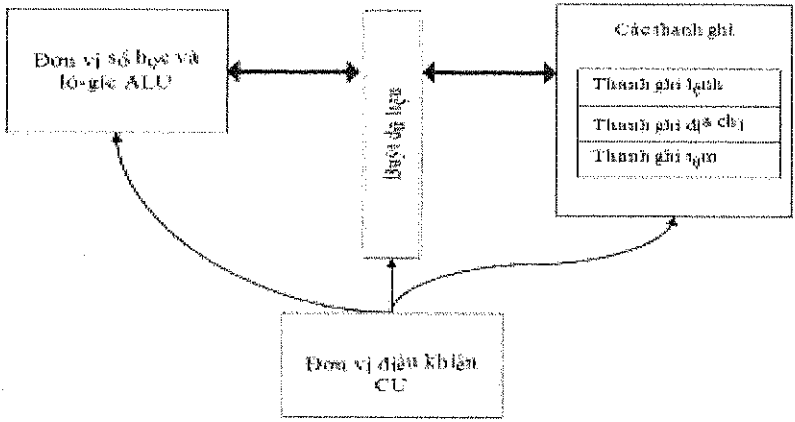
Khối phối ghép vào/ra (I/O) tạo ra khả năng giao tiếp giữa hệ vi xử lý với thế giới bên ngoài.

Buýt địa chỉ (address bus) thường có từ 16, 20, 24, 32 hay 64 đường dây song song chuyển tải thông tin của các bít địa chỉ.

Buýt dữ liệu (data bus) thường có từ 8, 16, 20, 24, 32, 64 (hoặc hơn) đường dây tùy theo các bộ vi xử lý cụ thể. Số lượng đường dây này quyết định số bít dữ liệu mà CPU có khả năng xử lý cùng một lúc.

Buýt điều khiển (control bus) thường gồm hàng chục đường dây tín hiệu khác nhau. Mỗi tín hiệu điều khiển có *một chiều nhất định*

2.2.Trình bày sơ đồ chức năng vi xử lý và giải thích vai trò của các khối chức năng trong quá trình thực hiện chương trình?



Thanh ghi lệnh: lưu các lệnh. Sau khi nạp mã lệnh từ bộ nhớ, vi xử lý lưu mã lệnh trong thanh ghi lệnh. Giá trị trong thanh ghi này luôn được vi xử lý giải mã để xác định lệnh.

Bộ đếm chương trình: chứa địa chỉ của lệnh hay mã thực thi (op-code). Thông thường, thanh ghi này chứa địa chỉ của câu lệnh kế.

Thanh ghi địa chỉ bộ nhớ: chứa địa chỉ của dữ liệu. Vi xử lý sử dụng các địa chỉ này như là các con trỏ trực tiếp tới bộ nhớ.

Thanh ghi dùng chung: dùng thể lưu hầu hết các kết quả tính toán của đơn vị xử lý số học và lô-gíc ALU. Thanh ghi này còn dùng để trao đổi dữ liệu với các thiết bị vào/ra.

ALU thực hiện tất cả các thao tác xử lý dữ liệu bên trong vi xử lý như là các phép toán lô-gíc, số học.

Chức năng chính của đơn vị điều khiển CU là đọc và giải mã các lệnh từ bộ nhớ chương trình.

2.5.Phân biệt chế độ địa chỉ trực tiếp và chế độ địa chỉ gián tiếp qua thanh ghi? Cho ví dụ?

Trực tiếp một toán hạng chứa địa chỉ lệnh của ô nhớ dùng chứa dữ liệu còn toán hạng kia chỉ có thể là thanh ghi mà không được là ô nhớ.	Gián tiếp một toán hạng là một thanh ghi được sử dụng để chứa địa chỉ lệch của ô nhớ chứa dữ liệu, còn toán hạng kia chỉ có thể là thanh ghi mà không được là ô nhớ
MOV AL, (1234H) MOV (4320H), CX	MOV AL, (BX) MOV (SI), CL

2.6.Phân biệt chế độ địa chỉ thanh ghi và chế độ địa chỉ tức thì? Cho ví dụ?

Thanh ghi người ta dùng các thanh ghi bên trong CPU như là các toán hạng để chứa dữ liệu cần thao tác.	địa chỉ tức thì toán hạng đích là một thanh ghi hay một ô nhớ, còn toán hạng nguồn là một hằng số và vị trí của toán hạng này ở ngay sau mã lệnh.
MOV BX, DX	MOV CL, 100

MOV DS, AX	MOV AX, 0FF0H

2.3.Phân biệt kiến trúc RISC và CISC ?

Thiết kế vi xử lý RISC sử dụng điều khiển cứng (*hardwired control*) không hoặc rất ít sử dụng vi mã. Tất cả các lệnh RISC có định dạng cố định vì vậy việc sử dụng vi mã không cần thiết. Vi xử lý RISC xử lý hầu hết các lệnh trong một chu kỳ. Tập lệnh của vi xử lý RISC chủ yếu sử dụng các lệnh với thanh ghi, nạp và lưu. Tất cả các lệnh số học và lô-gíc sử dụng thanh ghi, còn các lệnh nạp và lưu dùng để truy nhập bộ nhớ. Các lệnh có một định dạng cố định và ít chế độ địa chỉ. Vi xử lý RISC có một số thanh ghi dùng chung. Vi xử lý RISC xử lý một vài lệnh đồng thời và thường áp dụng kỹ thuật đường ống (*pipeline*).	vi xử lý CISC bao gồm số lượng lớn các lệnh và nhiều chế độ địa chỉ mà nhiều kiểu rất ít được sử dụng. Với CISC hầu hết các lệnh đều có thể truy nhập bộ nhớ trong khi đó RISC chỉ có các lệnh nạp và lưu. Do tập lệnh phức tạp, CISC cần đơn vị điều khiển phức tạp và ví chương trình. Kiến trúc CISC khó triển khai kỹ thuật đường ống.
Các chương trình phức tạp có thể chỉ cần vài lệnh với vài chu trình nạp	RISC cần một số lượng lớn các lệnh để thực hiện cùng nhiệm vụ. Tuy nhiên, RISC có thể cải thiện hiệu năng đáng kể nhờ xung nhịp nhanh hơn, kỹ thuật đường ống và tối ưu hóa quá trình biên dịch

2.4.Trình bày cách thức tổ chức và xác định địa chỉ ô nhớ của vi xử lý 8086? Cho ví dụ ?

CPU 8086 có 2 khối chính: *khối phối ghép* BIU *(Bus Interface Unit) và khối thực hiện lệnh* EU (*Execution Unit*). Việc chia CPU ra thành 2 phần làm việc đồng thời có liên hệ với nhau qua đệm lệnh làm tăng đáng kể tốc độ xử lý của CPU. Các buýt bên trong CPU có nhiệm vụ chuyển tải tín hiệu giữa các khối. Trong số các buýt đó có buýt dữ liệu 16 bit của ALU, buýt các tín hiệu điều khiển ở EU và buýt trong của hệ thống ở BIU. Trước khi đi ra buýt ngoài hoặc đi vào buýt trong của bộ vi xử lý, các tín hiệu truyền trên buýt thường được cho đi qua các bộ đệm để nâng cao tính tương thích cho nối ghép hoặc nâng cao phối ghép.

BIU đưa ra địa chỉ, đọc mã lệnh từ bộ nhớ, đọc/ghi dữ liệu từ vào cổng hoặc bộ nhớ.

EU bao gồm một *đơn vị điều khiển*, khối này có *mạch giải mã lệnh*. Mã lệnh đọc vào từ bộ nhớ được đưa đến đầu vào của bộ giải mã, các thông tin thu được từ đầu ra của nó sẽ được đưa đến mạch tạo xung điều khiển. Ngoài ra, EU còn có *khối số học và lôgic* (*Arithmetic and Logic Unit - ALU*) dùng để thực hiện các thao tác khác nhau với các toán hạng của lệnh. Tóm lại, khi CPU hoạt động EU sẽ cung cấp thông tin về địa chỉ cho BIU để khối này đọc lệnh và dữ liệu, còn bản thân nó thì đọc lệnh và giải mã lệnh.

Trong BIU còn có một *bộ nhớ đệm lệnh* với dung lượng 6 byte dùng để chứa các mã lệnh để chờ EU xử lý (bộ đệm lệnh này còn được gọi là *hàng đợi lệnh).*

2.7.Phân biệt chế độ địa chỉ gián tiếp qua thanh ghi và các chế độ địa chỉ tương đối cơ sở? Cho ví dụ?

Gián tiếp một toán hạng là một thanh ghi được sử dụng để chứa địa chỉ lệch của ô nhớ chứa dữ liệu, còn toán hạng kia chỉ có thể là thanh ghi mà không được là ô nhớ	Địa chỉ tdcs các thanh ghi cơ sở như BX và BP và các hằng số biểu diễn các giá trị dịch chuyển (*displacement values)* được dùng để tính địa chỉ hiệu dụng của toán hạng trong các vùng nhớ DS và SS. Sự có mặt của các giá trị dịch chuyển xác định tính tương đối của địa chỉ so với địa chỉ cơ sở.
MOV AL, (BX) MOV (SI), CL	MOV CX, (BX) +10 MOV CX, (BX+10)

2.8.Tại sao phải ngắt đơn vị xử lý trung tâm? Các loại ngắt trong hệ 8086?

Ngắt là việc tạm dừng việc chương trình đang chạy để CPU có thể chạy một chương trình khác nhằm xử lý một yêu cầu do bên ngoài đưa tới CPU như yêu cầu vào/ra hoặc do chính yêu cầu của bên trong CPU như lỗi trong khi tính toán. Trong cách tổ chức trao đổi dữ liệu thông qua việc thăm dò trạng thái sẵn sàng của thiết bị ngoại vi, trước khi tiến hành bất kỳ một cuộc trao đổi dữ liệu nào CPU phải dành toàn bộ thời gian vào việc xác định trạng thái sẵn sàng làm việc của thiết bị ngoại vi. Để tận dụng khả năng của CPU để làm thêm được nhiều công việc khác nữa, chỉ khi nào có yêu cầu trao đổi dữ liệu thì mới yêu cầu CPU tạm dừng công việc hiện tại để phục vụ việc trao đổi dữ liệu. Sau khi hoàn thành việc trao đổi dữ liệu thì CPU lại phải quay về để làm tiếp công việc hiện đang bị gián đoạn.

Các loại ngắt

_Nhóm các ngắt cứng: đó là các yêu cầu ngắt CPU do các tín hiệu đến từ các chân INTR và NMI. Ngắt cứng INTR là yêu cầu ngắt che được. Các lệnh CLI và STI có ảnh hưởng trực tiếp tới trạng thái của cờ IF trong bộ vi xử lý, tức là ảnh hưởng tới việc CPU có nhận biết yêu cầu ngắt tại chân này hay không. Yêu cầu ngắt tại chân INTR có thể có kiểu ngắt N nằm trong khoảng 0-FFH. Kiểu ngắt này phải được đưa vào buýt dữ liệu để CPU có thể đọc được khi có xung trong chu kỳ trả lời chấp nhận ngắt.

_Nhóm các ngắt mềm: khi CPU thực hiện các lệnh ngắt dạng INT N, trong đó N là số hiệu (kiểu) ngắt nằm trong khoảng 00-FFH (0-255).

_Nhóm các hiện tượng ngoại lệ: đó là các ngắt do các lỗi nảy sinh trong quá trình hoạt động của CPU như phép chia cho 0, xảy ra tràn khi tính toán.

2.9.Trình bày đáp ứng của CPU khi có yêu cầu ngắt ? Khi có nhiều yêu cầu ngắt thì CPU phải xử lý thế nào ?

Khi có yêu cầu ngắt kiểu N đến CPU và nếu yêu cầu đó được phép, CPU thực hiện các công việc sau:

1.SP <- SP-2, [SP] <- FR, trong đó [SP] là ô nhớ do SP chỉ ra.

(chỉ ra đỉnh mới của ngăn xếp, cất thanh ghi cờ vào đỉnh ngăn xếp)

2.IF <- 0, TF <- 0.

(cấm các ngắt khác tác động vào CPU, cho CPU chạy ở chế độ bình thường)

3.SP <- SP-2, [SP] <- CS.

(chỉ ra đỉnh mới của ngăn xếp, cất phần địa chỉ đoạn của địa chỉ trở về vào đỉnh ngăn xếp)

4.SP <- SP-2, [SP] <- IP

(chỉ ra đỉnh mới của ngăn xếp, cất phần địa chỉ lệch của địa chỉ trở về vào đỉnh ngăn xếp)

5.[N*4] <- IP, [N*4+2] <- CS (lấy lệnh tại địa chỉ mới của chương trình con phục vụ ngắt kiểu N tương ứng trong bảng vectơ ngắt)

6.Tại cuối chương trình phục vụ ngắt, khi gặp lệnh IRET

[SP] <- IP, SP <- SP+2

[SP] <- CS, SP <- SP+2

[SP] <- FR, SP <- SP+2

(bộ vi xử lý quay lại chương trình chính tại địa chỉ trở về và với giá trị cũ của thanh ghi cờ được lấy ra từ ngăn xếp).

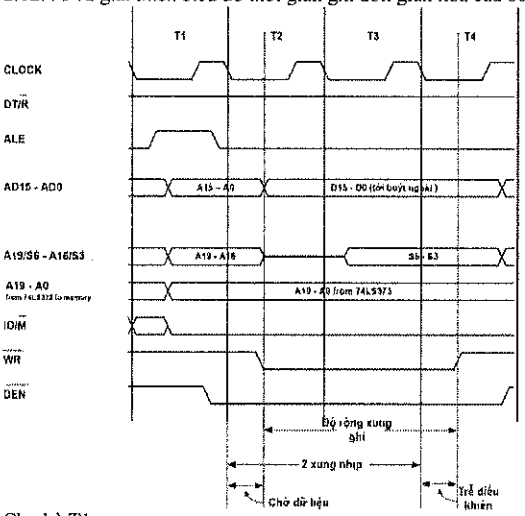
2.20.Trình bày phương pháp vào/ra trực tiếp bộ nhớ ? So sánh với phương pháp vào/ra sử dụng ngắt?

Trong thực tế có những khi ta cần trao đổi dữ liệu thật nhanh với thiết bị ngoại vi: như khi cần đưa dữ liệu hiện thị ra màn hình hoặc trao đổi dữ liệu với bộ điều khiển đĩa. Trong các trường hợp đó ta cần có khả năng ghi /đọc dữ liệu trực tiếp với bộ nhớ thì mới đáp ứng được yêu cầu về tốc độ trao đổi dữ liệu. Để làm được điều này các hệ vi xử lý nói chung đều phải dùng thêm mạch chuyên dụng để điều khiển việc truy nhập trực tiếp vào bộ nhớ DMAC (*Direct Memory Access Controller*)

Để hỗ trợ cho việc trao đổi dữ liệu với thiết bị ngoại vi bằng cách truy nhập trực tiếp vào bộ nhớ. CPU thường có tín hiệu yêu cầu treo HOLD để mỗi khi thiết bị cần dùng buýt cho việc trao đổi dữ liệu với bộ nhớ thì thông qua chân này mà báo cho CPU biết. Đến lượt CPU, khi nhận được yêu cầu treo thì nó tự treo lên (tự tách ra khỏi hệ thống bằng cách đưa các bít vào trạng thái trở kháng cao) và đưa xung HLDA ra ngoài để thông báo CPU cho phép sử dụng buýt.

Phương pháp này tốc độ nhanh hơn phương pháp vào rã bằng ngắt

2.12.Vẽ và giải thích biểu đồ thời gian ghi đơn giản hóa của 8086?



Chu kỳ T1:

Trong chu kỳ này địa chỉ của bộ nhớ hay thiết bị ngoại vi được đưa ra trên các đường địa chỉ, hoặc địa chỉ/dữ liệu và địa chỉ/ trạng thái. Các tín hiệu điều khiển ALE, DT/R̄ ,IO/M cũng được đưa ra để giúp việc hoàn tất việc giữ thông tin địa chỉ này.

Chu kỳ T2:

Trong chu này CPU đưa ra các tín hiệu điều khiển RD hoặc WR, DEN và tín hiệu dữ liệu trên D0 - D7 nếu là lệnh ghi. DEN thường dùng để mở các bộ đệm của buýt dữ liệu nếu như chúng được dùng trong hệ. Tại cuối kỳ T2 (và giữa mỗi chu kỳ T của T_w, nếu có) CPU lấy mẫu tín hiệu READY để xử lý trong chu kỳ tiếp theo khi nó phải làm việc với bộ nhớ hoặc thiết bị ngoại vi chậm.

Chu kỳ T3:

Trong chu kỳ này CPU dành thời giờ cho bộ nhớ hay thiết bị ngoại vi khi nhập dữ liệu. Nếu là chu kỳ đọc dữ liệu thì tại cuối T3 CPU sẽ lấy mẫu tín hiệu của buýt dữ liệu.

Nếu tại cuối chu kỳ đồng hồ T2 (hoặc giữa mỗi chu kỳ T của T_w) mà CPU phát hiện ra tín hiệu READY=0 (do bộ nhớ hay thiết bị ngoại vi đưa đến) thì CPU tự xen vào sau T3 một vài chu kỳ T để tạo chu kỳ đợi $T_w = n*T$ nhằm kéo dài thời gian thực hiện lệnh, tạo điều kiện cho bộ nhớ hoặc thiết bị ngoại vi có đủ thời gian hoàn tất việc ghi/đọc dữ liệu.

Chu kỳ T4:

Trong chu kỳ này các tín hiệu trên buýt được đưa về trạng thái bị động để chuẩn bị cho chu kỳ buýt mới. Tín hiệu WR trong khi chuyển trạng thái từ 0 lên 1 sẽ kích hoạt động quá trình đưa vào bộ nhớ hay thiết bị ngoại vi.

Trên các biểu đồ đọc ghi cũng biểu diễn các thông số quan trọng về mặt thời gian liên quan đến tốc độ hoạt động tối thiểu cần thiết của các bộ nhớ hoặc thiết bị ngoại vi nếu chúng muốn làm việc với CPU 5MHz.

2.19.Trình bày phương pháp vào/ra thăm dò ? So sánh phương pháp này với phương pháp vào/ra sử dụng ngắt?

Vấn đề điều khiển vào/ra dữ liệu sẽ trở nên đơn giản nếu thiết bị ngoại nếu lúc nào cũng sẵn sàng để làm việc với CPU. Tuy nhiên trong thực tế không phải lúc nào CPU cũng làm việc với các đối tượng "liên tục sẵn sàng" như trên. Thông thường khi CPU muốn làm việc với một đối tượng nào đó, trước

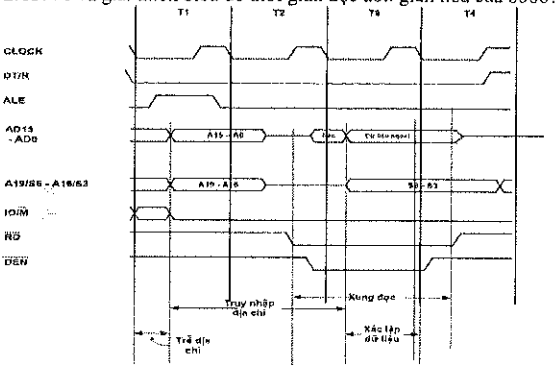
tiên nó phải kiểm tra xem thiết bị đó có đang ở trạng thái sẵn sàng làm việc hay không; nếu có thì nó mới thực hiện vào việc trao đổi dữ liệu. Như vậy, nếu làm việc theo phương thức thăm dò thì thông thường CPU chia sẻ thời gian hoạt động cho việc trao đổi dữ liệu và việc kiểm tra trạng thái sẵn sàng của thiết bị ngoại vi thông qua các tín hiệu móc nối (*handshake signal*).

Với bàn phím, CPU liên tục kiểm tra trạng thái các phím và nếu có phím được bấm CPU sẽ đọc thông tin trên cổng vào để xác định phím nào được bấm. Với bộ hiển thị LED, CPU liên tục đưa dữ liệu ra các cổng ra để thiết bị có thể hiển thị các thông tin.

Khi số lượng các thiết bị vào/ra tăng lên thì thời gian dành cho việc xác định trạng thái của thiết bị vào/ra cũng tăng lên nhanh chóng. CPU kiểm tra lần lượt các thiết bị để phát hiện trạng thái sẵn sàng trao đổi dữ liệu của từng thiết bị và thực hiện các lệnh trao đổi dữ liệu. Các thiết bị được kiểm tra thăm dò theo trật tự ngẫu nhiên hoặc theo mức độ ưu tiên của các thiết bị. Cách thức này đủ đơn giản song có nhược điểm là thời gian quét trạng thái của các thiết bị chiếm tỷ trọng rất đáng kể trong suốt quá trình vào/ra nhất là khi các thiết bị chưa có dữ liệu để trao đổi.

Nhược điểm của vào/ra thăm dò là máy tính cần kiểm tra bit trạng thái bằng cách chờ. Với các thiết bị chậm, việc chờ làm giảm khả năng xử lý dữ liệu khác của máy tính. Kỹ thuật ngắt cho phép giải quyết vấn đề này.

2.13. Vẽ và giải thích biểu đồ thời gian đọc đơn giản hóa của 8086?



Chu kỳ T1:

Trong chu kỳ này địa chỉ của bộ nhớ hay thiết bị ngoại vi được đưa ra trên các đường địa chỉ, hoặc địa chỉ/dữ liệu và địa chỉ/ trạng thái. Các tín hiệu điều khiển ALE, DT/$\overline{R}$,IO/M cũng được đưa ra để giúp việc hoàn tất việc giữ thông tin địa chỉ này.

Chu kỳ T2: Trong chu này CPU đưa ra các tín hiệu điều khiển $\overline{RD}$ hoặc $\overline{WR}$, $\overline{DEN}$ và tín hiệu dữ liệu trên D0 - D7 nếu là lệnh ghi. $\overline{DEN}$ thường dùng để mở các bộ đệm của buýt dữ liệu nếu như chúng được dùng trong hệ. Tại cuối kỳ T2 (và giữa mỗi chu kỳ T của T_w, nếu có) CPU lấy mẫu tín hiệu READY để xử lý trong chu kỳ tiếp theo khi nó phải làm việc với bộ nhớ hoặc thiết bị ngoại vi chậm.

Chu kỳ T3: Trong chu kỳ này CPU dành thời giờ cho bộ nhớ hay thiết bị ngoại vi khi nhập dữ liệu. Nếu là chu kỳ đọc dữ liệu thì tại cuối T3 CPU sẽ lấy mẫu tín hiệu của buýt dữ liệu. Nếu tại cuối chu kỳ đồng hồ T2 (hoặc giữa mỗi chu kỳ T của T_w) mà CPU phát hiện ra tín hiệu READY=0 (do bộ nhớ hay thiết bị ngoại vi đưa đến) thì CPU tự xen vào sau T3 một vài chu kỳ T để tạo chu kỳ đợi T_w = n*T nhằm kéo dài thời gian thực hiện lệnh, tạo điều kiện cho bộ nhớ hoặc thiết bị ngoại vi có đủ thời gian hoàn tất việc ghi/đọc dữ liệu.

Chu kỳ T4: Trong chu kỳ này các tín hiệu trên buýt được đưa về trạng thái bị động để chuẩn bị cho chu kỳ buýt mới. Tín hiệu WR trong khi chuyển trạng thái từ 0 lên 1 sẽ kích hoạt động quá trình đưa vào bộ nhớ hay thiết bị ngoại vi.

Trên các biểu đồ đọc ghi cũng biểu diễn các thông số quan trọng về mặt thời gian liên quan đến tốc độ hoạt động tối thiểu cần thiết của các bộ nhớ hoặc thiết bị ngoại vi nếu chúng muốn làm việc với CPU 5MHz.

2.14.Phân loại bộ nhớ? Trình bày các tín hiệu căn bản của vi mạch nhớ khái quát ?

1 vi mạnh nhớ có các nhóm tín hiệu sau:

Nhóm tín hiệu địa chỉ: Các tín hiệu địa chỉ có tác dụng chọn ra một ô nhớ của vi mạch nhớ. Các ô nhớ có độ dài khác nhau (còn gọi là từ nhớ) tuỳ theo nhà sản xuất: 1, 4, 8, bít. Số đường tín hiệu địa chỉ có liên quan đến dung lượng của mạch nhớ. Với một mạch nhớ có m bít địa chỉ thì dung lượng của mạnh nhớ đó là 2^m từ nhớ. Ví dụ, với m = 10 dung lượng mạch nhớ là 1K ô nhớ (1 kilô = 2^{10} = 1024) và với m=20 dung lượng mạch nhớ là 1M ô nhớ (1 Mêga = 2^{20} = 1048576).

Nhóm tín hiệu dữ liệu: Các tín hiệu dữ liệu thường là đầu ra đối với mạch ROM hoặc đầu vào/ra dữ liệu chung (hai chiều) đối với mạch RAM. Ngoài ra có loại mạch nhớ RAM với đầu ra và đầu vào dữ liệu riêng biệt. Các mạch nhớ thường có đầu ra dữ liệu kiểu 3 trạng thái. Số đường dây dữ liệu quyết định độ dài từ nhớ của mạch nhớ. Thông thường người ta hay nói rõ dung lượng và độ dài từ nhớ cùng một lúc. Ví dụ mạch nhớ dung lượng 1 Kx8 (tức là 1KB) hoặc 16Kx4. .

Tín hiệu chọn vi mạch (chọn vỏ): Các tín hiệu chọn vi mạch là *CS* (*chip select*) hoặc *CE* (*chip enable*) thường được dùng để tạo ra vi mạch nhớ cụ thể để ghi/đọc. Tín hiệu chọn vi mạch ở các mạch RAM thường là *CS* , còn ở mạch ROM thường là *CE* . Các tín hiệu chọn vi mạch thường được nối với đầu ra của bộ giải mã địa chỉ. Khi một mạnh nhớ không được chọn thì buýt dữ liệu của nó bị treo (ở trạng thái trở kháng cao).

Nhóm tín hiệu điều khiển: Tín hiệu điều khiển cần có trong tất cả các mạch nhớ. Các mạch nhớ ROM thường có một đầu vào điều khiển *OE* (*output enable*) để cho phép dữ liệu được đưa ra buýt. Khi mạch nhớ không được mở bởi *OE* thì buýt dữ liệu được treo. Một mạch nhớ RAM nếu chỉ có một tín hiệu điều khiển thì thường đó là *R* /*W* để điều khiển quá trình ghi/đọc. Nếu mạch nhớ RAM có hai tín hiệu điều khiển đó thường là *WE* (*write enable*) để điều khiển ghi và *OE* để điều khiển đọc. Hai tín hiệu này phải loại trừ lẫn nhau (ngược pha) để điều khiển việc ghi/đọc mạch nhớ.

2.15.Tại sao cần giải mã địa chỉ ô nhớ? Phân biệt giải mã địa chỉ đủ và giải mã thiếu?

Mỗi mạch nhớ nối ghép với CPU cần phải được CPU tham chiếu chính xác khi thực hiện các thao tác ghi/đọc. Điều đó có nghĩa là mỗi mạch nhớ phải được gán cho một vùng riêng biệt có địa chỉ xác định nằm trong không gian địa chỉ tổng thể của bộ nhớ. Việc gán địa chỉ cụ thể cho mạch nhớ được thực hiện nhờ một xung chọn vi mạch lấy từ mạch giải mã địa chỉ. Việc phân định không gian địa chỉ tổng thể thành các cùng nhớ khác nhau để thực hiện những chức năng nhất định gọi là phân vùng bộ nhớ. Giải mã đầy đủ cho một mạch nhớ đòi hỏi ta phải đưa đến đầu vào của mạch giải mã các tín hiệu địa chỉ sao cho tín hiệu ở đầu ra của nó chỉ chọn riêng mạch nhớ đã định.

Trong trường hợp này ta phải dùng tổ hợp đầy đủ của các đầu vào địa chỉ tương ứng để chọn được mạch nhớ. Nói cách khác, từ một tổ hợp tín hiệu địa chỉ, bộ giải mã sẽ chỉ sinh ra một tín hiệu chọn vỏ duy nhất ứng với không gian địa chỉ cấp cho vi mạch nhớ.

Giải mã địa chỉ thiếu hay giải mã rút gọn thì ta chỉ dùng một nhóm trong số các tín hiệu địa chỉ để sinh ra tín hiệu chọn vỏ cho mạch nhớ. Như vậy, từ một tổ hợp các tín hiệu địa chỉ có thể sinh ra nhiều tín hiệu chọn vỏ khác nhau. Vì sử dụng ít tín hiệu hơn nên mạch giải mã thiếu cần ít linh kiện hơn nhưng lại làm mất tính đơn trị của xung chọn thu được ở đầu ra.

2.16.Phân biệt các loại thiết bị vào/ra theo địa chỉ của chúng? Việc phân biệt các thiết bị vào/ra theo cách này ảnh hưởng như thế nào đến việc giải mã địa chỉ thiết bị vào/ra?

Thiết bị vào/ra có không gian địa chỉ tách biệt: Trong cách phối ghép này, bộ nhớ được dùng toàn bộ không gian 1MB mà CPU dành cho nó. Các thiết bị ngoại vi (các cổng) sẽ được dành riêng một không gian 64KB cho mỗi loại cổng vào hoặc ra. Để phân biệt các thao tác truy nhập, ta phải dùng tín hiệu IO/M =1, và các lệnh trao đổi dữ liệu một cách thích hợp cho mỗi không gian đó. Với các thiết bị này cần sử dụng các câu lệnh IN, OUT để trao đổi dữ liệu.

Thiết bị vào/ra và bộ nhớ có chung không gian địa chỉ: Trong cách phối ghép này, bộ nhớ và thiết bị ngoại vi cùng chia nhau không gian địa chỉ 1MB mà CPU 8086 có khả năng địa chỉ hóa. Các thiết bị ngoại vi sẽ chiếm một vùng nào đó trong không gian 1MB, phần còn lại là của bộ nhớ. Tất nhiên trong trường hợp này ta dùng chung tín hiệu IO/M =0 và lệnh trao đổi dữ liệu kiểu lệnh MOV cho cả bộ nhớ và thiết bị ngoại vi

2.17.Phân biệt chế độ vào/ra cơ sở và vào/ra thăm dò của vi mạch 8255A?

Chế độ 0: Vào/ra cơ sở:

Chế độ này cung cấp thao tác vào/ra đơn giản cho từng cổng, trên các cổng không có tín hiệu kết nối. Các cổng A, B và C có thể được chia thành 2 cổng 8 bít (A,B) và 2 cổng 4 bít (C thấp PC_0-PC_3, C cao PC_4-PC_7). Bất kỳ cổng nào có thể dùng làm cổng vào/ra.

Chế độ 1: Vào/ra thăm dò

Chế độ này chỉ được cung cấp trên hai cổng A,B, mỗi cổng có kênh dữ liệu là 8 bít và 4 tín hiệu điều khiển lấy từ cổng C. Dữ liệu trên kênh có thể là vào hay ra.

Các nhóm tín hiệu điều khiển vào/ra như sau:

Đầu vào	**Đầu ra**
STB: Kiểm tra đầu vào (mức thấp)	OBF: Dữ liệu ra sẵn sàng (mức thấp)
IBF: Dữ liệu sẵn sàng (mức cao)	ACK: Nhận xong dữ liệu (mức thấp)
INTR: Báo ngắt CPU (mức cao)	INTR: Báo ngắt CPU (mức cao)

2.18.So sánh truyền thông nối tiếp đồng bộ và dị bộ?

Nối tiếp đồng bộ	Dị bộ
dữ liệu được truyền theo từng khối với một tốc độ xác định. Khối dữ liệu trước khi được truyền đi sẽ được bổ sung thêm các phần tử đặc biệt ở đầu và ở cuối tạo thành khung. Các phần tử này dùng để đánh dấu điểm bắt đầu của khối dữ liệu hay các thông tin giúp phát hiện lỗi trong quá trình truyền. Đây thực chất là cách điều khiển hướng ký tự vì các ký tự đặc biệt được dùng để đánh dấu các phần khác nhau trong khung.	dữ liệu được truyền đi theo từng ký tự riêng biệt. Độ dài ký tự có thể thay đổi từ 5 đến 8 bít. Ký tự cần truyền đi được gắn thêm 1 bít đánh dấu ở đầu để báo bắt đầu kí tự (*Start bit*) và một hoặc hai bít báo kết thúc kí tự (*Stop bit*), và một bít kiểm tra tính toàn vẹn dữ liệu (*Parity bit*). Vì mỗi kí tự được nhận dạng riêng biệt nên nó có thể được truyền đi vào bất kì lúc nào. Giữa các kí tự truyền đi có thể có các khoảng cách về thời gian.

2.22.Phân biệt hệ vi điều khiển và hệ vi xử lý ? Cho ví dụ ứng dụng hệ vi điều khiển?

Hệ vi điều khiển là một máy tính trong đó các vi mạch cần thiết được bố trí trên một vi mạch duy nhất. Hệ vi điều khiển và hệ vi xử lí đều có một số điểm chung như sau:

_Đơn vị xử lý trung tâm (CPU) thực hiện các chương trình

_Bộ nhớ truy nhập ngẫu nhiên RAM chứa dữ liệu thay đổi

_Bộ nhớ chỉ đọc ROM chức các chương trình

_Các thiết bị vào/ra để liên lạc với thế giới bên ngoài như bàn phím, màn hình …

Hệ vi điều khiển có thể được mô tả bằng các đặc trưng khác. Nếu một máy tính có các đặc điểm chung như thế thì chúng có thể coi như là hệ vi điều khiển. Hệ vi điều khiển có thể:

_được nhúng bên trong các thiết bị khác (thường là các sản phẩm tiêu dùng) để kiểm soát các chức năng hay hoạt động của sản phẩm. Hệ vi điều khiển cũng được coi như bộ điều khiển nhúng;

_chỉ dùng cho một nhiệm vụ và chạy một chương trình xác định. Chương trình này thường được lưu trong ROM và không thay đổi;

_là thiết bị tiêu thụ điện thấp. Bộ vi điều khiển sử dụng pin có thể tiêu thụ chỉ 50 mA

Bộ vi điều khiển có thể nhận đầu vào từ thiết bị và điều khiển thiết bị này bằng cách gửi các tín hiệu tới các bộ phận khác nhau trong thiết bị được điều khiển. Bộ vi điều khiển thường nhỏ và chi phí thấp. Bộ xử lý được dùng trong một bộ vi điều khiển có thể thay đổi rất nhiều. Trong nhiều sản phẩm như lò vi sóng, yêu cầu về CPU khá thấp và sức ép về giá thành lại lớn nên các nhà sản xuất lựa chọn các vi mạch vi điều khiển chuyên dụng. Đó là các thiết bị CPU nhúng, giá rẻ, tiêu thụ điện thấp. Các

vi mạch Motorola 6811 và Intel 8051 là các ví dụ tiêu biểu. Các vi mạch vi điều khiển cấp thấp thường có sẵn 1KB ROM và 20 B RAM trên vi mạch cùng với 8 tín hiệu vào/ra.

2.23.Trình bày các chế độ hoạt động và mô hình tổ chức bộ nhớ của kiến trúc IA-32 ?

Các chế độ hoạt động quyết định các lệnh và các chức năng mà chương trình có thể truy nhập:

_Chế độ bảo vệ: là chế độ căn bản của bộ xử lý. Chế độ này cho phép chạy các phần mềm 8086 trong môi trường đa nhiệm và bảo vệ. Chế độ này còn được gọi là chế độ 8086 ảo.

_Chế độ địa chỉ thực: Chế độ này cung cấp môi trường lập trình 8086 với một số tính năng mở rộng như chuyển sang chế độ bảo vệ. Để bộ xử lý hoạt động ở chế độ này thông thường phải khởi động lại bộ xử lý.

_Chế độ quản lý hệ thống - SMM: Chế độ này cung cấp cho hệ điều hành các cơ chế trong suốt phục vụ nhiệm vụ cụ thể như quản lý năng lượng hay bảo mật hệ thống. Chế độ này được kích hoạt thông qua tín hiệu SMM hoặc tín hiệu này nhận được từ bộ điều khiển ngắt tiên tiến.

Trong chế độ này bộ xử lý chuyển qua lại các không gian địa chỉ riêng biệt trong khi lưu lại ngữ cảnh căn bản của các chương trình đang chạy. Các đoạn mã SMM có thể được thực hiện hoàn toàn trong suốt. Ngay khi quay trở lại từ chế độ SMM, bộ xử lý được khôi phục lại trạng thái giống như trước khi ngắt SMM xảy ra.

Về mô hình bộ nhớ, các chương trình không truy nhập trực tiếp vào bộ nhớ vật lý. Thay vào đó, các chương trình có thể sử dụng các mô hình truy nhập:

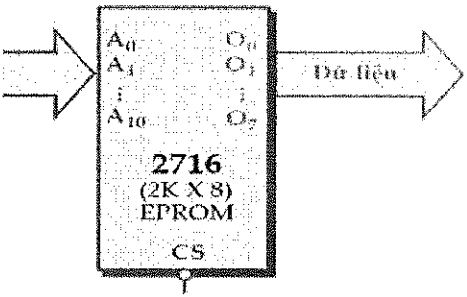
_Tuyến tính: Chương trình coi bộ nhớ như một chuỗi liên tiếp các byte. Đoạn mã, dữ liệu và ngăn xếp đều nằm trong không gian địa chỉ này.

_Phân đoạn: Bộ nhớ được chia thành các không gian khác nhau được gọi là đoạn. Thông thường dữ liệu, đoạn mã, ngăn xếp sử dụng các đoạn khác nhau. Bộ xử lý hỗ trợ IA-32 có thể cung cấp 16383 đoạn với các kích cỡ khác nhau, kích cỡ lớn nhất của 1 đoạn là 4GB.

_Địa chỉ thực: đây là mô hình bộ nhớ của 8086.

_Phân trang và bộ nhớ ảo: khi này bộ nhớ chương trình được chia thành các trang ánh xạ vào bộ nhớ ảo. Sau đó, bộ nhớ ảo được ánh xạ vào bộ nhớ thực. Nếu hệ điều hành sử dụng phân trang, cơ chế ánh xạ hoàn toàn trong suốt đối với chương trình ứng dụng

3.1 Xây dựng mạch giải mã địa chỉ dùng các mạch lô-gíc cơ bản cho bộ nhớ ROM dung lượng 4KB có địa chỉ cơ sở 5800H dùng vi mạch nhớ 2Kx8 (như hình vẽ)



Theo bài dung lượng bộ nhớ 4KB=4.1023=2^{12}

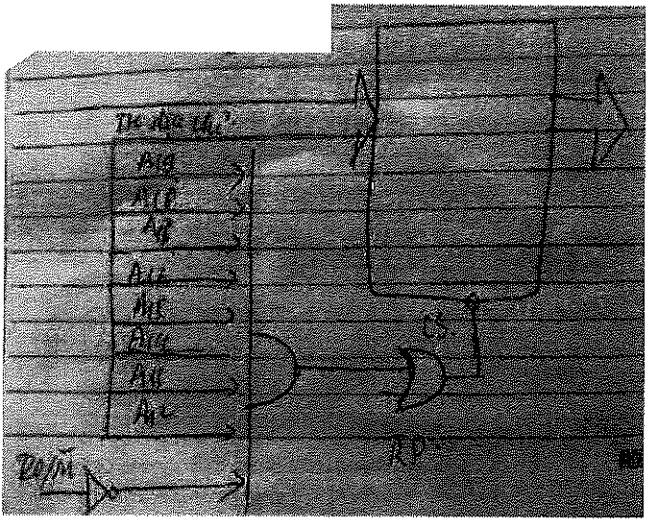
A19...A0, trong đó tín hiệu địa chỉ để giải mã là A0...A11

Địa chỉ cơ sở: 5800H=0000 0101 1000 0000

Tín hiệu địa chỉ để chọn chip A19 đến A12

CS=RD OR NOT((0000 0101) AND (IO/$\overline{M}$))

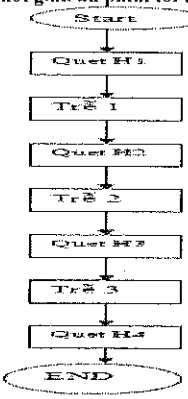
Mạch giải mã:



3.11 Cho bàn phím được nối với hệ vi xử lý 8086 tại các cổng vào 16H và cổng ra 15H:

1.Viết đoạn chương trình (bằng ngôn ngữ assembly) và vẽ lưu đồ quét các hàng của bàn phím với độ trễ giữa các hàng là 100 lệnh NOP

2.Viết đoạn chương trình (bằng ngôn ngữ assembly) và vẽ lưu đồ đọc cổng vào để xác định trường hợp có 3 phím được bấm. Thời gian ấn phím tối thiểu 100 lệnh NOP.



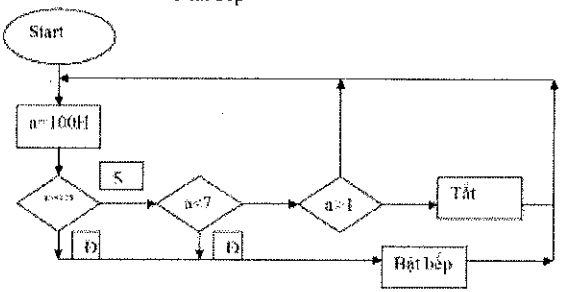

```
Model Small
Stack 100H
Data Nhap EQU 16H
          EQU 15H
Code:
START MOV AX,@DATA
          MOV DS,AX
          MOV AL,00010000b
```

```
        OUT XUAT,AL
        MOV CX,100
TRE1:   NOP
        LOOP TRE1
        ROL AX,1
        OUT XUAT,AL
        MOV CX,100
TRE2:   NOP
        LOOP TRE2
        ROL AX,1
        OUT XUAT,AL
        MOV CX,100
TRE3:   NOP
        LOOP TRE3
        ROL AX,1
        OUT XUAT,AL
        MOV CX,100
        MOV AH,4CH
        INT 21H
END START
```

3.12 Vẽ lưu đồ và viết chương trình điều khiển bếp làm sao cho nhiệt độ bếp luôn ổn định trong dải 70°C đến 100°C. Biết rằng hệ thống trên được nối với hệ vi xử lý 8086 trong đó Cổng đọc nhiệt độ là cổng 100H, giá trị nhiệt độ là số 8 bit có dấu tương ứng với giá trị nhiệt độ thực tế. Cổng điều khiển bếp là 105H, khi đưa giá trị 0 ra cổng thì bếp tắt còn đưa giá trị 1 thì bếp sẽ được đốt.
Viết chương trình điều khiển bếp

D7|D6D5D4D3D2D1D0

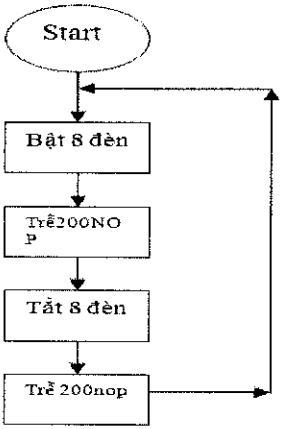
Am 1
Duong 0
>=128 Am
<128 Duong
Địa chỉ cổng nhiệt độ 100H
Cổng Đk 105H → 1 bật bếp
0 tắt bếp




```
Chương trình
 Model Small
Stack 100H
Data
 Nhiet do EQH 100H
        Đk   105H
```

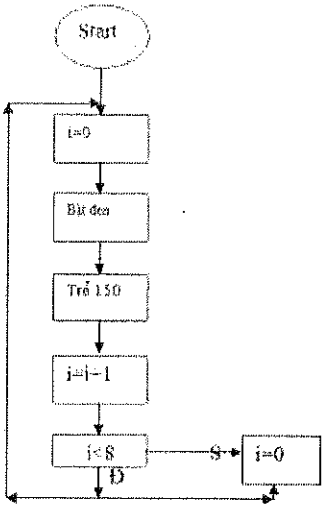
```
Code   Start MOV AX, @DATA
              MOV DS,AX
Bắt đầu IN AL,nhietdo
              CMP AL,128
              JG Batbep
               CMP AL,70
               JL BATBEP
              CMP AL,100
              JG TATBEP
              JMP BATDAU
BATBEP MOV BL,1
            OUT DIEUKHIEN,BL
JMP BATDAU
TATBEP MOV BL,0
            OUT DIEUKHIEN,BL
            JMP BATDAU
            MOV AH,4CH
INT 21H
END START
```

3.8 Xây dựng mạch giải mã địa chỉ cho mạch điều khiển truyền thông nối tiếp 8251? Biết địa chỉ cơ sở là 9DE4H và không gian địa chỉ tách biệt với bộ nhớ.
9DE4H 0000 1001 1101 1110 0100
vxl 8086 có ko gian nhớ 1MB,,dành 64KB làm ko gian cho mỗi loại cổng vào/ra :64x1024=2^{16}
Tín hiệu chọn chip:A16-A19
tín hiệu giải mã:A0-A15
hình vẽ:tương tự 3.1 :))
3.13 Cho mạch điều khiển 8 đèn (như hình vẽ) được nối với hệ vi xử lý 8086 tại cổng ra 120H. Biết rằng đèn được bật sáng nếu bít điều khiển tương ứng nhận giá trị 1. Ngược lại khi bít điều khiển bằng 0 thì đèn sẽ tắt. Vẽ lưu đồ và viết chương trình (bằng ngôn ngữ assembly) tạo các hiệu ứng sau:
1.Tất cả các đèn đều cùng bật tắt liên tục với khoảng trễ giữa hai lần bật tắt là 200 chu kỳ lệnh NOP.
2.Tạo hiệu ứng hai đèn kế nhau chạy liên tục từ trái sang phải, mỗi bước dịch chuyển tương ứng với 1 đèn, với khoảng nghỉ của 1 bước dịch chuyển là 150 chu kỳ lệnh NOP.
a




```
MODEL SMALL
STACK 100H
DATA
CODE START: MOV AX,@DATA
                MOV DS,AX
TIEPTUC: MOV AL,FFH
          OUT (110H),AL;
           MOV CX,250;
TRE1 NOP
      LOOP TRE1
      MOV AL,0
      OUT (110H),AL
       MOV CX,250
TRE2 NOP
       LOOP TRE2
      JMP TIEPTUC
END START
```

b.




```
MODEL SMALL
STACK 100H
DATA
CODE
START MOV AX,@DATA
        MOV DS,AX
        MOV AL,00000001b
OUT DIEUKHIEN,AL ; BẬT ĐÈN
     TIEPTUC MOV CX,150
TRE NOP
     LOOP TRE
     ROL AL,1
     OUT DIEUKHIEN,AL
     JMP TIEPTUC
     MOV AH,4CH
     INT 20H
END START
```

.17 Viết các đoạn chương trình (bằng ngôn ngữ assembly) cho 8251 được nối với cổng 12E4H thực hiện các công việc sau:

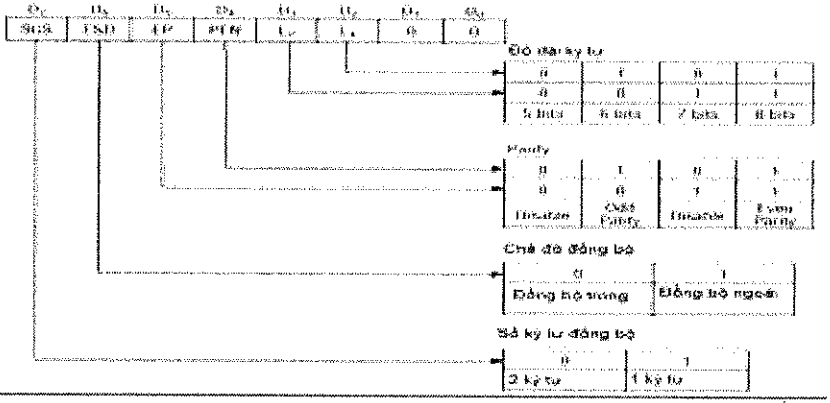
C/D (A0)	RD	WR	Thanh ghi
0	0	1	Thanh ghi đệm thu (Đọc)
0	1	0	Thanh ghi đệm phát (Ghi)
1	0	1	Thanh ghi trạng thái (Đọc)

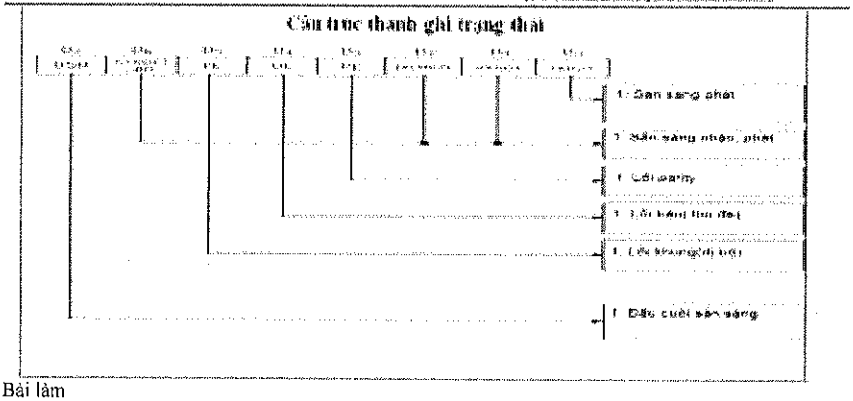
1	1	0	Thanh ghi điều khiển (Ghi)

1.Xác lập 8251 hoạt động ở chế độ đồng bộ trong, sử dụng 2 ký tự đồng bộ, mỗi ký tự được mã hoá bằng 7 bít, không dùng kiểm tra chẵn lẻ.
2.Viết đoạn chương trình (vẽ lưu đồ) luôn kiểm tra trạng thái đường phát, khi đường phát sẵn sàng gửi 1 byte dữ liệu (từ biến out_data) ra cổng phát của 8251.

C/D (A0)	RD	WR	Thanh ghi
0	0	1	Thanh ghi đệm thu (Đọc)
0	1	0	Thanh ghi đệm phát (Ghi)
1	0	1	Thanh ghi trạng thái (Đọc)
1	1	0	Thanh ghi điều khiển (Ghi)

Thanh ghi điều khiển chế độ (đồng bộ)

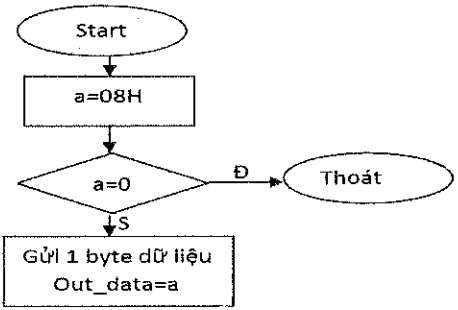




Bài làm

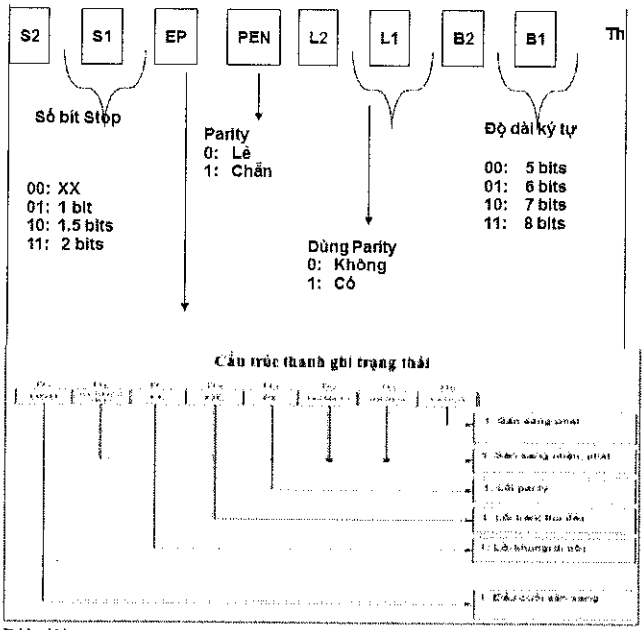
SCS	ESD	EP	PEN	L2	L1	0	0

-Theo bài có: 8251 hđ ở chế độ đồng bộ trong.
Sd 2 ký tự đồng bộ trong SCS 0
ESD=0
Ko sử dụng parity EP=0, PEN=0
Độ dài ký tự 7 bits=>L2L1=10
=>CW: 00001000=8 H
2.Kiểm tra trạng thái đường phát khi đường phát sẵn sang gửi 1 byte DL< từ biến out_data> ra cổng của 8251




```
.Mode Small
.Stack 100H
.Data congphat 12E4H
.Code
Start
Mov AX, @Data
Mov DS,AX
Mov Al,08H
Mov (out_data), AL
CMP AL,0
JE END
JMP tiep
Tiep: out congphat,out_data
JMP END
END: MOV AH, 4CH
INT 21 H
endStart.
```

3.18 Viết các đoạn chương trình (bằng ngôn ngữ assembly) cho 8251 được nối với cổng EF0H thực hiện các công việc sau:
1.Xác lập 8251 hoạt động ở chế độ đồng bộ dị bộ, tốc độ x16, mỗi ký tự được mã hoá bằng 7 bít, với 2 bít stop và sử dụng kiểm tra chẵn lẻ.
2.Viết đoạn chương trình (vẽ lưu đồ) luôn kiểm tra trạng thái đường thu, khi đường thu có dữ liệu thì đọc vào biến rd_data.



Bài giải :

1.Xác lập 8251 hđ ở chế độ đồng bộ dị bội, tốc độ x16, mỗi ký tự được mã hóa =7bit, với 2 bit stop và sử dụng kiểm tra chẵn lẻ.

-XĐ từ điều khiển TG chế độ

S2 S1 EP PEN L2 L1 B2 B1

1 1

-Số bit stop 2 bit =>S2S1=11

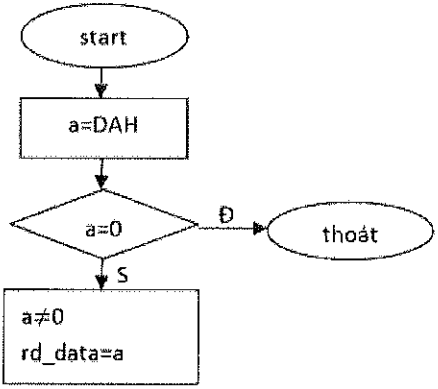
-Sd kiểm tra chẵn lẻ PEN=1, 7 bits parity lẻ

-Đc dài ký tự 7 bits=>L2L1=10

-tốc độ x16=> B2B1=10

-Cw=11011010 DA H

2.Kiểm tra trạng thái đường thu khi đường thu có dữ liệu thì đọc vào biến rd_data




```
.Model
.Stack 100H
.Data
DK EQU EF0H
.Code
Mov AX,@Data
Mov DS,AX
Mov AL,DAH
CMP AL, 0
JE END
Jmp Doc
Doc: IN (rd_data),DK
Jmp END
END: Mov AH,4CH
INT 21H
endStart.
```

Bài 1: Viết chương trình hiện ra câu "Hello Assembly 2021"

```
.MODEL small
.STACK 100h
.DATA
tb1 db 'Hello Assembly 2021!$'
.CODE
main proc
Mov AX,DATA
Mov DS,AX
Lea DX,tb1Mov AH,9
Int 21h
Mov AH,4ChInt 21h
endp mainend main
```

Bài 2: Viết chương trình hiện ra hai câu "Chao mung ban den voi Assembly 2021" "Assembly 2021 that de!". Mỗi câu trên một dòng.

```
.MODEL small
.STACK 100h
.DATA
tb1 db "Chao mung ban den voi Assembly 2021$"
tb2 db 0DH,0AH,"Assembly 2021 that de!$"
.CODE
main proc
Mov AX,DATA
Mov DS,AX
;xuat thong bao 1Lea DX,tb1
Mov AH,9
Int 21h
;xuat thong bao 2Lea DX,tb2
Mov AH,9
Int 21h
Mov AH,4ChInt 21h
endp mainend main
```

Bài 3: Viết chương trình yêu cầu nhập một ký tự từ bàn phím và xuấtra màn hình ký tự vừa nhập.

```
.model small
.stack
.data
tb1 db 13,10,"Hay nhap mot ky tu: $"
tb2 db 13,10,"Ky tu da nhap: $"KyTu db ?
.code main proc
Mov ax,data

Mov ds,ax

; nhac nhapLea dx, tb1Mov ah, 9 int 21h
; nhap 1 ky tuMov ah, 1
Int 21h

Mov KyTu, al

; thong bao ket qualea dx, tb2
mov ah, 9int 21h
; hien thi ky tu da nhapMov ah, 2
Mov dl, KyTuInt 21h
; ve dos
Mov ah, 4ChInt 21h
endp mainend main
```

Bài 4: Viết chương nhập vào một ký tự từ bàn phim. Chuyển ký tựđó sang ký tự hoa.

```
.MODEL SMALL

.STACK 100h

.DATA

tb1 DB 'Nhap vao ki tu thuong : $'

tb2 DB 13,10,'Chuyen sang ki tu hoa la : 'Char
DB ?,'$'
.CODE

Main PROC

MOV AX,DATA

MOV DS,AX

; In ra thong bao 1LEA DX,tb1 MOV AH,9
INT 21h

; Nhap vao 1 ki tu thuong va doi thanh ki tu hoa
MOV AH,1
INT 21h ; Doc 1 ki tu thuong va luu vao AL
SUB AL,20h ; Doi thanh ki tu hoaMOV
Char,AL
; Hien len chu hoaLEA DX,tb2 MOV AH,9
INT 21h

; Ket thuc chuong trinhMOV AH,4Ch
INT 21h

endp mainend main
```

Bài 5: Viết chương trình chuyển đổi ký tự hoa thành ký tự thườngtrong môn kỹ thuật vi xử lý.

```
.MODEL SMALL

.STACK 100h

.DATA
```

```
tb1 DB 'Nhap vao ki tu hoa : $'

tb22 DB 0Dh,0Ah,'Chuyen sang ki tu thuong la :
'Char DB ?,'$'
.CODE

Main PROC

MOV AX,DATA

MOV DS,AX
; In ra thong bao 1LEA DX,tb1 MOV AH,9
INT 21h

; Nhap vao 1 ki tu hoa va doi thanh ki tu thuong
MOV AH,1
INT 21h ; Doc 1 ki tu hoa va luu vao ALADD
AL,20h ; Doi thanh ki tu thuong MOV Char,AL
; Hien len chu thuongLEA DX,tb2
MOV AH,9INT 21h
; Ket thuc chuong trinhMOV AH,4Ch
INT 21h

endp main end main nhap chuoi data segment
tb1 db "Hay nhap vao mot chuoi ky tu: $"tb2 db
13, 10,"chuoi vua nhap la: $"
str db 200,?,200 dup('$')
ends

stack segmentdw 128 dub(?)ends
code segmentmov ax,data mov  ds,ax lea dx,tb1
mov ah,09h int 21h
mov ah,0Ahlea  dx,str int 21h
lea DX,tb2 mov ah,09hint 21h
lea bx,str

mov al,[bx+01h]mov ah,00h
add bx,ax

;mov <bx+2>,"$"mov ah,09h
lea dx,str+2
int 21hends
```

Bài 6: Viết chương trình nhập vào một chuỗi. In ra màn hình chuỗithường, chuỗi in. Dùng chương con.

```
.MODEL small

.STACK

.DATA

tb1 DB 'Nhap vao 1 chuoi: $'

tb2 DB 10,13,'Doi thanh chu thuong: $'tb3 DB
10,13,'Doi thanh chu hoa: $'
s DB 100,?,101 dup('$')

.CODE BEGIN:
MOV AX,DATA

MOV DS,AX

;xuat chuoi tb1MOV AH,09hLEA DX,tb1 INT
21h
;nhap chuoi s MOV AH,0AHLEA DX,s
INT 21h
;xuat chuoi tb2MOV AH,09hLEA DX,tb2 INT
21h
; Goi chuong trinh con in chuoi thuongCALL
InChuoiThuong
; xuat chuoi tb3MOV AH,09h LEA DX,tb3 INT
21h
; Goi chuong trinh con in chuoi thuongCALL
InChuoiHoa
MOV AH,4chINT 21h
;*************************************

; Doi thanh chuoi ky tu thuongInChuoiThuong
PROC
LEA SI,s+1 XOR CX,CXMOV CL, INC SI
LapThuong: MOV AH,02h
MOV DL, CMP DL,'A'JB  LT1 CMP DL,'Z'JA
LT1 ADD DL,32 LT1: INC SIINT 21h
LOOP LapThuongRET
InChuoiThuong ENDP

; Doi thanh chuoi ky tu hoaInChuoiHoa PROC
LEA SI,s+1 XOR CX,CXMOV CL, INC SI
LapHoa: MOV AH,02hMOV DL, CMP DL,'a'
JB LH1
CMP DL,'z'
JA  LH1 SUB DL,32 LH1: INC SIINT 21h
LOOP LapHoaRET
InChuoiHoa ENDPEND BEGIN
```

Bài 8: Viết chương trình nhập vào một chuỗi. Đếm chiều dài củachuỗi nhập vừa nhập trong môn kỹ thuật vi xử lý.

```
.MODEL small

.STACK
```

```
.DATA

tb1 DB 'Nhap vao 1 chuoi: $'

tb2 DB 10,13,'Tong chieu dai cua chuoi: $'s DB
100,?,101 dup('$')
.CODE BEGIN:
MOV AX,DATA

MOV DS,AX

;xuat chuoi tb1MOV AH,09hLEA DX,tb1
INT 21h

;nhap chuoi s MOV AH,0AHLEA DX,s
INT 21h

;xuat chuoi tb2MOV AH,09hLEA DX,tb2 INT
21h
;Tinh chieu dai chuoiXOR AX,AX
MOV AL,s+1 ;Chuyen chieu dai chuoi vao ax
MOV CX,0 ;Khoi tao bien dem
MOV BX,10

LapDem1: MOV DX,0DIV BX PUSH DX INC
CX CMP AX,0
JNZ LapDem1

;xuat chieu dai chuoiMOV AH,2
LapDem2:

POP DX

OR DL,'0' ;chuyen chu so -> soINT 21H
LOOP LapDem2MOV AH,4ch INT 21h
END BEGIN
```

Bài 9: Viết chương trình nhập vào 2 số kiểu byte, in ra màn hình tích2 số vừa nhập trong môn kỹ thuật vi xử lý.

```
.model tiny

.stack 100h

.data

tb1 db 'nhap a=$' tb2 13,10,'nhap b=$'
tb3 13,10,'tich 2 so la:$'so1 db 0
so2 db 0

.code main proc
Mov ax,Data

Mov ds,axMov ah,9
;in thong bao nhap so thu 1Lea dx,tb1
int 21hnhap1:
mov ah,1int 21h
cmp al,13 ;so sanh al voi 13

je nhap2 ;neu bang thi nhay den nhap 2sub
al,30h ; chuyen ky tu thanh so
mov dl,al ;cat al vao dl

mov al,so1 ; dua so vua nhap ve kieu bytemov
bl,10 ;gan bl =10
mul bl ;nhan al voi 10

add al,dl ;lay ket qua vua nhan cong voi so vua
nhapmov s01,al ;cat ket qua sau khi doi vao bien
so1
jmp nhap1 ;nhay den nhan nhap 1nhap2:
lea dx,tb2; in thong bao nhap so thu 2mov ah,9
int 21h

nhap: mov ah,1int 21h
cmp al,13 ; so sanh so vua nhap voi enter
je tinhtich ;neu bang thi tinh tich

sub al,30h ; chuyen xau vua nhap thanh somov
dl,al ;cat so vua nhap vao dl
mov al,so2 ;dua so vua nhap ve kieu bytemov
bl,10 ;gan bl=10
mul bl ; lay so ban dau nhan voi 10

add al,dl ;lay ket qua vua nhan cong voi so vua
nhapmov s02,al ;cat ket qua sau khi doi vao bien
so2
jmp nhaptinhtich:
mov al,so1 ;dua so vua nhap ra thanh ghi almul
so2 ;nhan voi so 2
mov bx,ax ;lay ket qua vua tinh chuyen vao
thanh ghi bxjmp tinhtich
;in tich

mov ah,9 ;hien thong bao in tichlea dx,tb3
int 21h

mov ax,bx ;chuyen ket qua ra thanh ghi axmov
bx,10 ;gan bx=10
xor cx,cx ;khoi tao bien demchia: xor dx,dx ;xoa
bit cao
div bx ;lay ket qua chia cho 10 du dat dx,thuong
dat ax
```

```
push dx ;day du trong dx vao ngan xepinc cx
;tang bien dem
cmp ax,0 ;so sanh thuong voi 0ja chia ;neu lon
hon thi chia
mov ah,2 ;lay chuc nang in ky tu ra man hinhlay:
pop dx ;lay du trong stasck khoi dx
add dl,30h ;chuyen so vua nhap sang dang ky tu

int 21h ;thuc hien in ky tu nam trong dl ra man
hinhloop lay
ra: mov ah,4chint 21h
Main endpEnd main
```

Bài 10 : Viết chương trình nhập vào 2 số kiểu word, in ra màn hìnhtổng 2 số vừa nhập trong môn kỹ thuật vi xử lý.

```
.model tiny

.stack 100h

.data

tb1 db 'nhap so thu 1:$'

tb2 db 13,10 ,'nhap so thu 2:$'tb3 db 13,10,'tong
2 so$'
so1 dw 0

so2 dw 0

tong dw 0
.code

main proc

mov ax,data

mov ds,ax

;in thong bao nhap so thu nhatlea dx,tb1
mov ah,9int 21h nhap1:
mov ah,1int 21h
cmp al,13 ;so sanh ky tu vua nhap voi 13je
nhap2 ;neu bang nhap so thu 2
sub al,30h ;doi ky tu sang somov ah,0 ;xoa bit
cao
mov cx,ax ;cat so vua nhap vào cx

mov ax,so1 ;đua bien số 1 vê kiêu byte de chuan
bi nhann với 10mov bx,10 ;gan bx =10
mul bx ; nhân ax voi 10
add ax,cx ;công ket qua vua nhan voi so vua
nhap ket qua cât vào axmov so1,ax ; cat kêt qua
vao biến số1
jmp nhap1nhap2:
lea dx,tb2 ;hiên thông báo nhập số thu 2mov ah,9
int 21h

nhap: mov ah,1 ;nhap số thu 2int 21h
cmp al,13 ;so sánh ký tu vua nhập voi 13je
tinhtong ;nêu băng thì tính tông
sub al,30h ;chuyên ký tu sang dang sômov ah,0
;xoá bít cao
mov cx,ax ;cât kêt qua vua nhap vào cxmov
ax,so2 ;đua biến số 2 vê kiêu bytemov bx,10 ;gan
bx=10
mul bx ;nhân kêt qua vua nhap voi 10

add ax,cx ;công kêt qua vua nhân với số vua
nhậpmov so2,ax ;cât kêt qua vào biên số 2
jmp nhaptinhtong:
mov dx,tong

mov ax,so1 ;đua biên số 1 ra thanh ghi ax mov
bx,so2 ;đua biên só 2 ra thanh ghi bx add ax,bx
;công ax voi bx kêt qua cat vao ax mov tong,ax
;đua kêt qua tu ax vào biên tônginso: mov ah,9
;hiên thông báo in tông
lea dx,tb3int 21h
mov ax,tong ;đua kêt qua trongv bien tông ra
thanh ghi axmov dx,0 ;xoa bit cao dx
mov bx,10 ;gán bx=10

mov cx,0 ;khoi tao biên đêm

chia: div bx ;lây kêt qua chia cho 10push dx ;du
o dx đây vao ngăn xêp inc cx ;tăng biên đêm
cmp ax,0 ;so sanh thuong voi 0

jc hienkq ;neu băng thì hiên kêt quaxor dx,dx
;xoa bit cao trong dx
jmp chia

hienkq: pop dx ;lây du trong ngăn xêp ra khoi dx
add dl,30h ;chuyên sô thành dang ký tu
mov ah,2 ;in tôngint 21h
loop  hienkq ra: mov ah,4chint 21h
Main endpEnd main
```

Bài 11 : Cho một mảng M gồm 20 phần tử kiểu Word giá trị tuỳ ý (không phải nhập giá trị các phần tử).Tính tổng giá trị các phần tửcó giá trị chia hết cho 7 trong môn kỹ thuật vi xử lý.

```
.model tiny

.stack 100h

.data

st1 db 13,10,'tong cac phan tu chia het cho 7:$'
st2 db 13,10,'$'
m db
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,2
0

a db 0

.code main proc
mov ax,data

mov ds,ax

;in thông báo nhập số thứ 1lea dx,st1
mov ah,9int 21h
mov cx,20 ;gán cx=20

lea si,m ;si trỏ đến ngăn nhớ đầu tiên của mảng
Mmov a,0 ; khởi tạo a=0
duyet:

mov al, ; đưa các giá trị trong mảng do si trỏ đến
vào almov bl,7 ;gán bl=7
mov ah,0 ;xoá bit caodiv bl ;chia al cho 7
cmp ah,0 ;so sánh thương với 0je tong ;nếu bằng
thì tính tổng jmp tiep
tong:

mov al, ; đưa các giá trị trong mảng do si trỏ đến
vào almov bl,a ; đ ưa số a vào bl
add al,bl ;c ộng aí v ơi bl kết quả cất vào almov
a,al ;chuyển kết quae\r vào biến a tiep:
inc si ;tăng chỉ số mảnginc dl ;tăng dl
loop duyet

mov al,a ;chuyển số trở lại thanh ghi almov bl,10
;gán bl =10
mov cx,0 ;khởi tạo biến đếmchia:
mov ah,0 ;xoá bít cao

div bl ;lấy kết quả chia cho 10mov dl,ah ;chuyển
dư vào dl
add dl,30h ;chuyển số sang dạng ký tựpush dx
;đẩy dư vào ngăn xếp
inc cx ;tăng biến đếm
cmp al,0 ;so sánh thương với 0je inso ;nếu bằng
thì in số
jmp chiainso:
pop dx mov ah,2int 21h loop insora:
mov ah,4chint 21h main endp end main
```

Bài 12 : Viết chương trình nhập vào 1 số kiểu word in ra màn hìnhmã nhị phân tương ứng của số đó trong môn kỹ thuật vi xử lý.

```
.model tiny

.stack 100h

.data

st1 db 'Nhap so kieu WORD :$'

st2 db 13,10,'Ma nhi phan tuong ung:$'so dw 0
.code main proc
mov ax,data

mov ds,axlea dx,st1 mov ah,9
int 21h ;In xau st1nhap:
mov ah,1 int 21h cmp al,13 je nhiphansub al,30h
mov ah,0 mov cx,axmov bx,10mov ax,somul bx
add ax,cx mov so,axjmp nhap nhiphan: lea
dx,st2 mov ah,9 int 21h
mov cx,0 mov bx,2 mov ax,sochia:
mov dx,0div bx
add dx,30hpush  dx inc cx
cmp ax,0je inso jmp chiainso:
pop dx mov ah,2int 21h loop insora:
mov ah,4chint 21h main endp end main
```

Bài 13 : Viết chương trình nhập vào 1 số kiểu word in ra màn hìnhmã Hexa tương ứng của số đó trong môn kỹ thuật vi xử lý.

```
.model tiny

.stack 100h

.data

st1 db 'nhap so kieu word:$'

st2 db 13,10,'so do duoi dang hecxa:$'a dw 0
.code main proc
;

mov ax,data
```

```
mov ds,ax

;

lea dx,st1 mov ah,9 int 21h nhap: mov ah,1 int
21h cmp al,13je inso mov ah,0 sub al,30hmov
cx,ax
mov ax,a mov bx,10mul bx add ax,cx mov a,ax
jmp nhap inso:
;

lea dx,st2 mov ah,9 int 21h mov bx,16mov ax,a
mov cx,0 chia:
mov dx,0 div bx cmp dx,10jae doi
add dx,30hjmp cat
doi: add dx,37hcat:
push dx inc cx cmp ax,0je hien jmp chiahien:
pop dx mov ah,2int 21h loop hienra:
mov ah,4chint 21h main endp end main
```

Bài 14 : Viết chương trình nhập vào 1 mảng 15 phần tử kiểu word inra màn hình mã Hexa tương ứng của số đótrong môn kỹ thuật vi xử lý.

```
.model tiny

.stack 100h

.data

tb1 db 'Nhap mang 15 phan tu kieu word: $'tb2
db 10,13,'Nhap phan tu : $'
tb3 db 10,13,'Phan tu co gia tri lon nhat la: $'
a dw 20 dup(0)

.code

main proc

mov ax,data

mov ds,ax

;_____ lea dx,tb1 mov  ah,9 int 21h
mov cx,15 ;Nhap 15 phan tu kieu wordxor si,si
nhapmang:mov ah,9 lea dx,tb2 int 21h push cx
nhapso: mov ah,1 int 21h cmp al,13 je catkq sub
al,30h mov cl,al
xor ch,ch mov bx,10mov ax,a mul bx add ax,cx
mov a,ax
jmp nhapsocatkq:
add si,2pop cx
loop nhapmanglea dx,tb3
mov ah,9int 21h xor si,si mov ax,a
mov cx,15 ; mov cx,14 add si,2duyet:
cmp a,axjbe qua mov ax,aqua:
add si,2
loop duyet

;mov ax,a<0>xor  cx,cx mov bx,10 chia:
xor dx,dxdiv bx push dx inc cx cmp ax,0jne chia
mov ah,2hienso:
pop dx add dl,30hint 21h
loop hienso

;_____

mov ah,4chint 21h main endp end main
```

Bài 15 : Tính tổng 2 số:

```
Code Segment Assume cs: CodeOrg 100h
Start: jmp over

tb1 db 'Nhap a = $'

tb2 db 10, 13, 'Nhap b = $'tb3 db 10, 13, 'Tong
2 so da nhap la $'
over:
Tính hiệu hai số:
Code Segment Assume cs: CodeOrg 100h
Start: jmp over

tb1 db 'Nhap a = $'

tb2 db 10, 13, 'Nhap b = $'tb3 db 10, 13, 'Hieu 2
so danhap la $'
over:
```

Bài 16 : Viết chương trình nhập 1 kí tự cho ra số có mã Hexa tươngứngtrong môn kỹ thuật vi xử lý.

```
Code Segment
 Assume cs:
 CodeOrg 100h
Start: jmp over

tb1 db 10, 13, 'Nhap ki tu kt = $'
tb2 db 10, 13, 'So thapphan tuong ung la 1$'
Trang 3
Mov ah,9 lea dx,tb1int 21h mov ah,1 int 21h
mov bl,al mov ah,9 lea dx,tb2int 21h mov ah,1
int 21h add bl,al mov ah,9 lea dx,tb3int 21h sub
bl,30h
cmp bl,39h
jbe thoat sub bl,10 mov ah,2 mov dl,'1'int 21h
```

```
thoat: mov dl,bl mov ah,2 int  21h int 20h Mov
ah,9 lea dx,tb1int 21h mov ah,1 int 21h mov
bl,al mov ah,9 lea dx,tb2int 21h mov ah,1 int
21h mov cl,al mov ah,9
lea dx,tb3int 21h cmp bl,cl jae thoat xchg bl,cl
mov ah,2 mov dl,'-'int 21h thoat:
sub bl,cl add bl,30hmov ah,2 mov dl,bl int 21h
tb3 db 10, 13, 'Hay nhaplai voi ( A
kt over:
nhap: Mov ah,9lea dx,tb1
int 21h mov ah,1int 21h
mov bl,al cmp bl,'A'jae  ss1 jmp loi
ss1: cmp bl,'F'jbe thoat1
cmp bl,'a'jae ss2 jmp loi
ss2: cmp bl,'F'jbe thoat2
loi: Mov ah,9lea dx,tb3
int 21h jmp nhapthoat1:
sub bl,11hjmp thoat3thoat2:
```

Trang 4 Code EndsEnd Start

Bài 17 : Viết chương trình kiểm tra tính chẵn lẻ của 1 chữ sốtrongmôn kỹ thuật vi xử lý.

Xem thêm: Hàng Lỗi Tiếng Anh Là Gì - Hàng Hóa Lỗi Thời TiếngAnh Là Gì

```
Code Segment Assume cs: CodeOrg 100h
Start: jmp over

tb1 db 10, 13, 'Nhap ki tuso kt = $'
tb2 db 10, 13, 'Do la sochan $'
tb3 db 10, 13, 'Do la so le$'tb4 db 10, 13, 'Hay
nhap
lai voi ( 0over:
nhap: Mov ah,9lea dx,tb1
int 21h mov ah,1 int 21h mov bl,al cmp bl,30h
jae sosanh jmp loi sosanh:
cmp bl,39hjbe inra
int 20h Code EndsEnd Start
```

Câu 18: Viết chương trình in theo thứ tự bảng chữ cáitrong môn kỹthuật vi xử lý.

```
Code Segment Assume cs: CodeOrg 100h
Start: jmp over

tb1 db 'Nhap ki tu thu 1 : $'tb2 db 10, 13, 'Nhap
ki tu thu 2 : $'
tb3 db 10, 13, 'Thu tu bangma la : $'
over: Mov ah,9lea dx,tb1int 21h mov ah,1int 21h
mov bl,almov ah,9
lea dx,tb2int 21h mov ah,1 int 21h mov cl,al sub
bl,31hthoat3:
Mov ah,9lea dx,tb2int 21h mov ah,2mov dl,blint
21h int 20h
Code EndsEnd Start
```

Bài 17 : Viết chương trình tính thương – dư và xuất ra màn hìnhtrongmôn kỹ thuật vi xử lý.

```
Code Segment Assume cs: CodeOrg 100h
Start: jmp over

tb1 db 'Nhap mot so bat kya = $'
tb2 db 10, 13, 'Thuong cuaBL : $'
tb3 db 10, 13, 'Du cua BL :

$' over:
Mov bl,19Mov ah,9 lea dx,tb1int 21h mov cl,0
mov ah,1
loi: mov ah,9lea dx,tb4
int 21h jmp nhap
inra: Test bl,1jne sole
mov ah,9lea dx,tb2jmp thoat
sole: mov ah,9lea dx,tb3 thoat: int 21h
int 20h Code EndsEnd Start
```

Bài 18 : Viết chương trình kiểm tra số nguyên tốtrong môn kỹ thuậtvi xử lý.

```
include mylib.macCode Segment Assume cs:
Code Org 100h
Start:

write 'So da nhap la sochinh phuong'
thoat:

int 20h

include proc.asmCode Ends
End Start
```

Bài 19 : Viết chương trình tính tổng các phần tử lêtrong môn kỹ thuật vixử lý.

```
Include Mylib.macCode Segment Trang 12
chia: xor dx,dxdiv bx
push dx inc cx cmp ax,0ja chia mov ah,2
inra: pop dxor dx,30h int  21h loop  inra int 20h
Code Ends End Start
```

Bài 20: Kiểm tra số hoàn thiệnInclude Mylib.mac

```
Code Segment Assume cs : codeOrg 100h
Start : jmp overn dw ?
over:

Write 'Nhap n = 'Call Nhap_so mov n,ax
mov bx,2xor cx,cxlap:
Include Mylib.macCode Segment Assume Cs :
```

```
CodeOrg 100h
Start : jmp overn dw ?
over:

Write 'Nhap x = 'Call  nhap_so cmp ax,2
jbe ngt mov n,ax shr ax,1 mov cx,axmov bx,2
chia:
xor dx,dxmov ax,ndiv  bx inc bx
cmp dx,0 ; hay or dx,dxje hopso
loop chiangt:

write 'Nhap 1 phan tu: '

Call nhap_somov  a,ax add bx,2 loop nhap mov
cx,n xor  bx,bx xor ax,ax lap:
xor dx,dxmov dx,aTest dx,1je tiep add ax,a tiep:
add bx,2 loop lap xor dx,dxdiv bx
cmp dx,0jne tiep add cx,axtiep:
inc bx mov ax,n cmp bx,axjbe lap cmp cx,n jne
khong

write 'So da cho ko hoanthien'
thoat:
int 20h
Include Proc.asmCode Ends
End Start
```

Bài 21: Viết chương trình tính tích 2 sốtrong môn kỹ thuật vi xử lý.

```
include mylib.maccode segment assume cs:code
org 100h
start:
thoat:
int 20h
Include Proc.asmCode ends
End Start
```

Bài 22: Viết chương trình tính số Fibonacy thứ ntrong môn kỹ thuậtvi xử lý.

```
Include mylib.macCode Segment Assume cs:
Code Org 100h Start:jmp over
n dw ?over:
xuongdongcmp ax,2 jbe thoat mov n,ax mov
ax,1 mov bx,1
mov cx,2 ;tinh tu n>2tinh: add bx,ax
sub ax,bx
neg ax inc cx cmp n,cxje thoat1
write 'tong la : '
Call in_soint 20h
Include Proc.asmCode Ends
End Start
```

Câu 23: Tìm Min-Max của mảng:Include Mylib.mac

```
Max Macro w1,w2local thoat
mov ax,w1cmp ax,w2ja thoat mov ax,w2thoat:
EndM
Min Macro w1,w2local thoat
mov ax,w1
cmp ax,w2jbe thoat mov ax,w2thoat:
EndM
Code Segment Assume Cs : CodeOrg 100h
Start : jmp over
write 'tich cua a*b la:'mov ax,cx
call in_soint 20h
include proc.asmcode ends
end start
```

Bài 24 : Viết chương trình sắp xếp các phần tử tăng dầntrong môn kỹthuật vi xử lý.

```
include mylib.maccode segment assume cs: code
jmp tinh
thoat:
mov bx,1thoat1:
write 'So Fibonaci thu nia '
Mov ax,bxCall in_so int 20h
include proc.asmCode Ends
End Start
```

Bài 25 : Viết chương trình nhập xâu kí tự từ bàn phím chuyển kí tựđó thành chữ hoa sang xâu khác và in ra màn hìnhtrong môn kỹ thuật vi xử lý.

```
Include Mylib.macCode Segment Assume CS :
CodeOrg 100h
Start : Jmp overx1 db 80 dup(?)x2 db 80 dup(?)
over:
write 'Nhap xau : 'lea di,x1
xor bx,bxcld
nhap:
mov ah,1 int 21h cmp al,0dhje chuyen stosb
inc bx
cmp ax,a<bx+2>jlc qua_
xchg ax,a<bx+2>mov a,ax
qua_:
add bx,2 cmp bx,dxjb lap_ loop for_
write ' ' loop forin_int 20h
include proc.asmcode ends
end start
```

Câu 26 : Viết chương trình nhập xâu kí tự từ bàn phím chuyển kí tựvừa nhập thành chữ thường sang xâu khác và in ran màn hìnhtrongmôn kỹ thuật vi xử lý.

```
Include Mylib.macCode Segment Assume CS :
CodeOrg 100h
Start : Jmp overx1 db 80 dup(?)x2 db 80 dup(?)
over:
write 'Nhap xau : 'lea di,x1
```

```
xor bx,bxcld
nhap: mov ah,1int 21h
cmp al,0dhje chuyen stosb
inc bx jmp nhap
chuyen:

mov byte ptr,'$'inc bx
lea si,x1

xuongdonglea di,x2 mov cx,bx lap:
mov al, cmp al,'A'jb nhay cmp al,'Z'ja nhay
stosb
nhay: inc siloop lap
mov byte ptr,'$'lea si,x2

xuongdongmov ah,9 lea dx,x2 int 21h
int 20h Code Ends
```

Bài 27 : Viết chương trình nhập họ tên và tách tên đó hiển thị ra mànhìnhtrong môn kỹ thuật vi xử lý.

```
write 'Nhap ho & ten:'lea di,hten
cld nhap:
mov ah,1 int 21h cmp al,0dhje chuyen stosb
jmp nhapchuyen:
mov byte ptr,'$'
write 'Ho ten la : 'mov ah,9
lea dx,htenint  21h dec di
std
mov al,' '
repe scasbinc cx
inc  di mov bx,cx
repne scasbinc cx
add di,2 sub bx,cx mov cx,bxlap:
mov al, cmp al,'a'jb nhay cmp al,'z'ja nhay
stosb
nhay: inc siloop lap
mov byte ptr,'$'lea si,x2
xuongdongmov ah,9 lea dx,x2
int 21hint 20h
Code EndsEnd Start
```

Bài 28 : Viết chương trình kiểm tra tính đối xứng của xâutrong mônkỹ thuật vi xử lý.

```
include mylib.maccode segment assume cs:code
org 100h
start:jmp over

xau db 80 dup ('$')End Start
```

Bài 29 : Đếm từ trong xâuinclude mylib.mac

```
code segment assume cs:codeorg 100h start:jmp
over
xau db 80 dup ('$')over:

write 'nhap xau: 'xor cx,cx
xor bx,bxlea di,xaucld
mov ah,1nhap:
int 21h cmp al,13je  tiep inc cx stosb
jmp nhaptiep:
jcxz inradec di std
mov al,' 'lap:
repe scasbor  cx,cx je inra
inc cx Trang 18 mov cx,bx
mov si,dilea di,tencld
rep movsb

mov byte ptr,'$'

write 'Ten la:'lea  dx,ten mov ah,9
int 21hint 20h
Code EndsEnd Start
```

Bài 30 : Viết chương trình kiểm tra chữ hoatrong môn kỹ thuật vi xửlý.

```
code segment assume cs:codeorg 100h start:jmp
over
tb1 db 'nhap mot ki tu:$' tb2 db 10,13,'la chu
hoa$'tb3 db 10,13,'khong$' over:
mov ah,9
lea dx,tb1 int 21h mov ah,1 int 21h cmp al,'A'jb
ko
cmp al,'Z'ja ko
mov ah,9lea dx,tb2int 21h jmp het over:

xuongdonglea di,xau xor cx,cx cld
mov ah,1nhap:
int 21h cmp al,13je  tiep inc cx
stosb jmp nhaptiep:
dec  di shr cx,1 lea si,xaulap:
cmpsb

jne kdxungsub di,2 loop lap
write 'Xau khong doixung'
thoat:
int 20h
include proc.asminc di
inc bx repne scasbinc cx
inc di jmp lapinra:
write 'So tu trong xau la:'mov ax,bx
call in_soint 20h
include proc.asmcode ends
end start
```