

## GIẢI NGÂN HÀNG CÂU HỎI KIẾN TRÚC MÁY TÍNH

---

### Contents

Câu hỏi 1.1: Kiến trúc máy tính là gì ? Kiến trúc máy tính được cấu thành từ những thành phần nào ? .....	4
Câu hỏi 1.2: Nêu sơ đồ khối chức năng của hệ thống máy tính. ....	4
Câu hỏi 1.3: Thanh ghi của vi xử lý là gì? Nêu chức năng và đặc điểm của thanh ghi tích lũy A. ....	4
Câu hỏi 1.4: Nêu chức năng và đặc điểm của bộ đếm chương trình PC. ....	5
Câu hỏi 1.5: Thanh ghi cờ (hay thanh ghi trạng thái) của vi xử lý có chức năng gì? .....	5
Câu hỏi 1.6: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ tức thì. Cho ví dụ. ....	6
Câu hỏi 1.7: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ trực tiếp. Cho ví dụ. ....	6
Câu hỏi 1.8: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ gián tiếp qua thanh ghi. Cho ví dụ minh họa. ....	6
Câu hỏi 1.9: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ gián tiếp qua ô nhớ. Cho ví dụ minh họa. ....	7
Câu hỏi 1.10: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ chỉ số. Cho ví dụ minh họa. ....	7
Câu hỏi 1.11: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ tương đối. Cho ví dụ minh họa. ....	7
Câu hỏi 1.12: Nêu phương thức trao đổi dữ liệu giữa CPU, cache và bộ nhớ chính. ....	8
Câu hỏi 1.13: Nêu đặc điểm chính của đĩa CD và đĩa DVD. ....	8
Câu hỏi 1.14: Nêu nguyên lý hoạt động của chuột quang. ....	9
Câu hỏi 2.1: Nêu sơ đồ khối chức năng và chức năng chính của các thành phần trong một hệ thống máy tính ? .....	10
Câu hỏi 2.2: Nêu sơ đồ và các đặc điểm của kiến trúc máy tính von-Neumann. ....	11
Câu hỏi 2.3: Nêu sơ đồ và các đặc điểm của kiến trúc máy tính Harvard. ....	12
Câu hỏi 2.4: Nêu sơ đồ khối tổng quát và chu trình xử lý lệnh của CPU. ....	13

Câu hỏi 2.5: Nêu sơ đồ khối và chức năng của các khối điều khiển (CU) và khối tính toán số học và logic (ALU). ....	14
Câu hỏi 2.6: Lệnh máy tính là gì ? .....	15
Câu hỏi 2.7: Nêu các dạng địa chỉ của lệnh. Cho ví dụ minh họa với mỗi dạng địa chỉ. ....	16
Câu hỏi 2.8: Cơ chế xử lý xen kẽ dòng lệnh (ống lệnh – pipeline) là gì ? ....	17
Câu hỏi 2.9: Nêu cấu trúc phân cấp của hệ thống bộ nhớ máy tính. ....	18
Câu hỏi 2.10: Phân biệt bộ nhớ RAM tĩnh và RAM động. ....	20
Câu hỏi 2.11: Bộ nhớ cache là gì ? .....	21
Câu hỏi 2.12: So sánh các chuẩn ghép nối ổ đĩa cứng IDE, SATA và SCSI. ....	23
Câu hỏi 2.13: Trình bày nguyên lý đọc thông tin trên đĩa CD. ....	24
Câu hỏi 2.14: Nêu nguyên lý hoạt động của máy in laser. ....	25
Câu hỏi 2.15: Nêu nguyên lý tạo hình ảnh của màn hình LCD. ....	26
Câu hỏi 3.1: Nêu sơ đồ và đặc điểm của hai dạng kiến trúc cache : .....	27
Câu hỏi 3.2: So sánh 3 phương pháp ánh xạ cache: .....	27
Câu hỏi 3.3: Nêu các phương pháp đọc ghi và các chính sách thay thế dòng cache. ....	29
Câu hỏi 3.4: RAID là gì? .....	30
ADD R2, (R1) .....	36

Câu hỏi 3.4: RAID là gì? Tại sao RAID có thể nâng cao được tính tin cậy và tốc độ truy nhập 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?

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

Câu hỏi 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 R4, R4, #300      ; R4 <== R4+300
ADD R1, R2, R3        ; R1 <== R2+R3
CMP R1, #100          ; so sánh R1 với 100
SUB R5, #2000          ; R5 <== 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 (WB).

Câu hỏi 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    ; R4 <== R4+300
ADD R1, R1, R3    ; R1 <== R1+R3
SUB R1, R1, #100    ; R1 <== R1 - 100
SUB R5, #2000      ; R5 <== 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 (WB).

Câu hỏi 3.8: Cho đoạn chương trình sau (R1, R2 là các thanh ghi):

- (1) LOAD R2, #400
- (2) LOAD R1, #1200
- (3) STORE (R1), R2
- (4) SUBTRACT 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).

Câu hỏi 3.9: Cho đoạn chương trình sau (R1, R2 là các thanh ghi):

- (1) LOAD R2, #500
- (2) LOAD R1, #2000
- (3) STORE (R1), R2
- (4) ADD 2000, #30
- (5) SUBTRACT 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).

Câu hỏi 3.10: Cho một dãy số nguyên gồm 10 phần tử lưu trong bộ nhớ bắt đầu từ địa chỉ 1000. Viết chương trình sử dụng tập lệnh của CPU 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.

## • Câu hỏi loại 1 điểm

**Câu hỏi 1.1: Kiến trúc máy tính là gì ? Kiến trúc máy tính được cấu thành từ những thành phần nào ?**

**TL:** Kiến trúc máy tính (Computer architecture) là một khoa học về lựa chọn và kết nối các thành phần phần cứng của máy tính nhằm đạt được các yêu cầu:

- Hiệu năng/tốc độ (performance): nhanh.
- Chức năng (functionality): nhiều tính năng.
- Giá thành (cost): rẻ.

Kiến trúc máy tính là một trong hai khái niệm cơ bản của công nghệ máy tính.

KTMT bao gồm 3 thành phần cơ bản là:

- Kiến trúc tập lệnh (Instruction set architecture – ISA).
- Vi kiến trúc (micro-architecture).
- Thiết kế hệ thống (System Design).

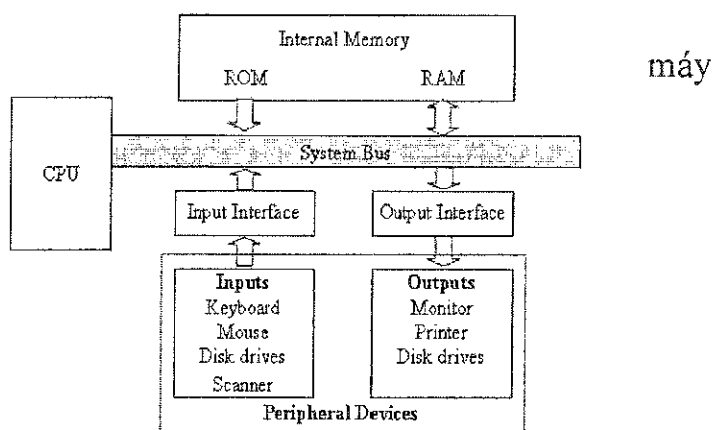
**Câu hỏi 1.2: Nêu sơ đồ khối chức năng của hệ thống máy tính.**

**TL:**

Sơ đồ khối chức năng hệ thống máy tính như hình bên:

Trong đó:

- Bộ xử lý trung tâm (CPU)
- Bộ nhớ trong (Internal Memory): bao gồm: ROM và RAM.
- Các thiết bị vào ra (Peripheral devices).
- Bus hệ thống (system bus).



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

**TL:**

Thanh ghi (registers) là các ô nhớ bên trong CPU có đặc điểm là kích thước nhỏ, tốc độ rất cao (bằng tốc độ CPU) và có chức năng lưu trữ tạm thời lệnh và dữ liệu cho CPU xử lý.

Số lượng thanh ghi tùy thuộc vào đời CPU, các CPU cũ (80x86) có 16-32 thanh ghi; các CPU hiện đại (Pentium 4 và Core Duo) có hàng trăm thanh ghi; Kích thước thanh ghi phụ thuộc vào thiết kế CPU. Các kích thước thông dụng của thanh ghi là 8, 16, 32, 64, 128 và 256 bit.

Thanh tích lũy A là một trong các thanh ghi quan trọng nhất của hầu hết các CPU. Thanh ghi tích lũy A có chức năng:

- dùng để chứa toán hạng đầu vào
- dùng để chứa kết quả đầu ra
- sử dụng để trao đổi dữ liệu với các thiết bị vào ra.

Kích thước của A bằng kích thước từ xử lý của CPU: 8, 16, 32 và 64 bit.

**Câu hỏi 1.4: Nêu chức năng và đặc điểm của bộ đếm chương trình PC**

**TL:**

- Bộ đếm chương trình PC (Program Counter) hoặc con trỏ lệnh (IP – Instruction Pointer) luôn chứa địa chỉ của ô nhớ chứa lệnh được thực hiện tiếp theo.
- PC chứa địa chỉ của ô nhớ chứa lệnh đầu tiên của chương trình khi nó được kích hoạt và được nạp vào bộ nhớ.
- Khi CPU thực hiện xong lệnh, địa chỉ của ô nhớ chứa lệnh tiếp theo được nạp vào PC.
- Kích thước PC phụ thuộc vào thiết kế CPU. Các kích thước thông dụng là 8, 16, 32 và 64 bit.

**Câu hỏi 1.5: Thanh ghi cờ (hay thanh ghi trạng thái) của vi xử lý có chức năng gì?**

Nêu ý nghĩa của các cờ nhớ (C), cờ không (Z), cờ dấu (S).

**TL:**

Thanh ghi trạng thái (SR - Status Register) hoặc thanh ghi cờ (FR – Flag Register) là thanh ghi đặc biệt của CPU. Mỗi bit của FR lưu trạng thái của kết quả của phép tính ALU thực hiện.

Hai loại bit cờ: Cờ trạng thái (CF, OF, AF, ZF, PF, SF) và cờ điều khiển (IF, TF, DF).

Các bit cờ thường được sử dụng như là các điều kiện trong các lệnh rẽ nhánh để tạo logic chương trình. Kích thước của thanh ghi FR phụ thuộc thiết kế CPU.

Ý nghĩa của một số cờ:

ZF: Cờ Zero, ZF=1 nếu kết quả=0 và ZF=0 nếu kết quả $\neq$ 0.

SF: Cờ dấu, SF=1 nếu kết quả âm và SF=0 nếu kết quả dương.

CF: Cờ nhớ, CF=1 nếu có nhớ/mượn, CF=0 trong trường hợp khác.

**Câu hỏi 1.6: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ tức thì. Cho ví dụ.**

**TL:**

Chế độ địa chỉ (Addressing modes) là phương thức CPU tổ chức các toán hạng của lệnh, nó cho phép CPU kiểm tra dạng và tìm các toán hạng của lệnh.

Chế độ địa chỉ tức thì: trong chế độ địa chỉ tức thì, giá trị hằng của toán hạng nguồn (source operand) nằm ngay sau mã lệnh, toán hạng đích có thể là 1 thanh ghi hoặc 1 địa chỉ ô nhớ.

Ví dụ: `LOAD R1, #1000;`       $R1 \leftarrow 1000;$       Nạp giá trị 1000 vào thanh ghi R1.

`LOAD B, #500;`       $M[B] \leftarrow 500;$       Nạp giá trị 500 vào ô nhớ B.

**Câu hỏi 1.7: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ trực tiếp. Cho ví dụ.**

**TL:**

Chế độ địa chỉ (Addressing modes) là phương thức CPU tổ chức các toán hạng của lệnh, nó cho phép CPU kiểm tra dạng và tìm các toán hạng của lệnh. Sử dụng một hằng để biểu diễn địa chỉ một ô nhớ làm một toán hạng, toán hạng còn lại có thể là 1 thanh ghi hoặc 1 địa chỉ ô nhớ;

Ví dụ: `LOAD R1, 1000;`  $R1 \leftarrow M[1000]$

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

**Câu hỏi 1.8: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ gián tiếp qua thanh ghi. Cho ví dụ minh họa.**

**TL:**

Chế độ địa chỉ (Addressing modes) là phương thức CPU tổ chức các toán hạng của lệnh, nó cho phép CPU kiểm tra dạng và tìm các toán hạng của lệnh.

Trong chế độ địa chỉ gián tiếp qua thanh ghi, một thanh ghi được sử dụng để lưu địa chỉ toán hạng, toán hạng còn lại có thể là một hằng, một thanh ghi hoặc một ô nhớ.

Ví dụ: `LOAD Rj, (Ri);`  $R_j \leftarrow M[R_i];$

Nạp nội dung ô nhớ có địa chỉ lưu trong thanh ghi Ri vào thanh ghi Rj.

**Câu hỏi 1.9: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ gián tiếp qua ô nhớ. Cho ví dụ minh họa.**

**TL:**

Chế độ địa chỉ (Addressing modes) là phương thức CPU tổ chức các toán hạng của lệnh, nó cho phép CPU kiểm tra dạng và tìm các toán hạng của lệnh.

Trong chế độ địa chỉ gián tiếp qua ô nhớ, một ô nhớ được sử dụng để lưu địa chỉ toán hạng, toán hạng còn lại có thể là một hằng, một thanh ghi hoặc một ô nhớ.

Ví dụ:  $\text{LOAD Ri, (1000); Ri} \leftarrow \text{M(M(1000))}$

Nạp nội dung ô nhớ có địa chỉ lưu trong ô nhớ 1000 vào thanh ghi Ri.

**Câu hỏi 1.10: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ chỉ số. Cho ví dụ minh họa.**

**TL:**

Chế độ địa chỉ (Addressing modes) là phương thức CPU tổ chức các toán hạng của lệnh, nó cho phép CPU kiểm tra dạng và tìm các toán hạng của lệnh.

Trong chế độ địa chỉ chỉ số, địa chỉ của 1 toán hạng được tạo thành bởi phép cộng giữa 1 hằng và thanh ghi chỉ số (index register), toán hạng còn lại có thể là một hằng, một thanh ghi hoặc một ô nhớ.

Ví dụ:  $\text{LOAD Ri, X(R}_{\text{ind}}\text{); Ri} \leftarrow \text{M[X+R}_{\text{ind}}\text{];}$  X là hằng và  $\text{R}_{\text{ind}}$  là thanh ghi chỉ số.

**Câu hỏi 1.11: Chế độ địa chỉ của vi xử lý là gì ? Mô tả chế độ địa chỉ tương đối. Cho ví dụ minh họa.**

**TL:**

Chế độ địa chỉ (Addressing modes) là phương thức CPU tổ chức các toán hạng của lệnh, nó cho phép CPU kiểm tra dạng và tìm các toán hạng của lệnh.

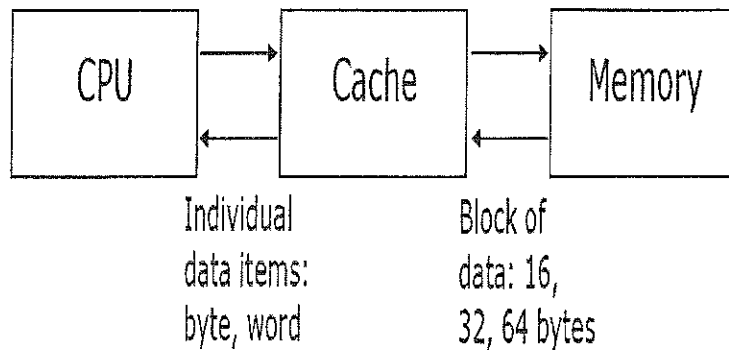
Chế độ địa chỉ tương đối: trong chế độ địa chỉ này, địa chỉ của 1 toán hạng được tạo thành bởi phép cộng giữa 1 hằng và bộ đếm chương trình PC (Program Counter), toán hạng còn lại có thể là một hằng, một thanh ghi hoặc một ô nhớ.

Ví dụ:  $\text{LOAD Ri, X(PC); Ri} \leftarrow \text{M[X+PC];}$  X là hằng và PC là bộ đếm chương trình.

**Câu hỏi 1.12: Nêu phương thức trao đổi dữ liệu giữa CPU, cache và bộ nhớ chính.**

**TL:** Sơ đồ trao đổi dữ liệu giữa CPU, cache và bộ nhớ chính như hình bên.

CPU trao đổi dữ liệu với cache



theo các đơn vị cơ sở như byte, từ, từ kép. Cache trao đổi dữ liệu với bộ nhớ chính theo các khối với kích thước 16, 32, 64 bytes. Sở dĩ CPU trao đổi dữ liệu với cache theo các đơn vị cơ sở mà không theo khối do dữ liệu được lưu trong các thanh ghi của CPU - vốn có dung lượng rất hạn chế. Vì vậy, CPU chỉ trao đổi các phần tử dữ liệu cần thiết theo yêu cầu của các lệnh. Ngược lại, cache trao đổi dữ liệu với bộ nhớ chính theo các khối, mỗi khối gồm nhiều byte kề nhau với mục đích bao phủ các mẫu dữ liệu lân cận theo không gian và thời gian. Ngoài ra, trao đổi dữ liệu theo khối (hay mẻ) với bộ nhớ chính giúp cache tận dụng tốt hơn băng thông đường truyền và nhờ vậy có thể tăng tốc độ truyền dữ liệu.

**Câu hỏi 1.13: Nêu đặc điểm chính của đĩa CD và đĩa DVD.**

**TL:** Đặc điểm chính của đĩa CD:

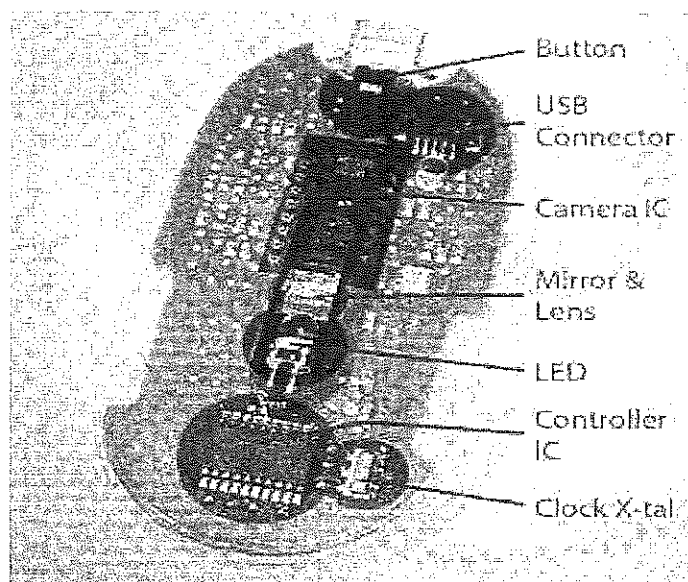
- Dung lượng tối đa của đĩa CD là 700MB hoặc 80 phút nếu lưu âm thanh
- Ổ đĩa sử dụng tia laser hồng ngoại với bước sóng 780 nm để đọc thông tin.
- Tốc độ truyền thông tin của đĩa CD được tính theo tốc độ cơ sở (150KB/s) nhân với hệ số nhân. Ví dụ, đĩa có tốc độ đọc 4x thì tốc độ tối đa có thể đọc là  $4 \times 150\text{KB/s} = 600 \text{KB/s}$ ; nếu đĩa có tốc độ đọc 50x thì tốc độ tối đa có thể đọc là  $50 \times 150\text{KB/s} = 7500 \text{KB/s}$ .

Đặc điểm chính của đĩa DVD:



- Dung lượng tối đa của đĩa DVD là 4,7GB với đĩa một mặt và 8,5GB với đĩa 2 mặt.
- Ổ đĩa DVD sử dụng tia laser hồng ngoại có bước sóng 650nm, ngắn hơn nhiều so với bước sóng tia laser dùng trong ổ đĩa CD nên có mật độ ghi cao hơn nhiều so với CD.
- Tốc độ truyền thông tin của đĩa DVD được tính theo tốc độ cơ sở (1350KB/s) nhân với hệ số nhân. Ví dụ, đĩa có tốc độ đọc 4x thì tốc độ tối đa có thể đọc là  $4 \times 1350\text{KB/s} = 5400 \text{ KB/s}$ ; nếu đĩa có tốc độ đọc 16x thì tốc độ tối đa có thể đọc là  $16 \times 1350\text{KB/s} = 21600 \text{ KB/s}$ .

**Câu hỏi 1.14: Nêu nguyên lý hoạt động của chuột quang.**



**TL :**Nguyên tắc hoạt động của chuột quang có thể tóm tắt như sau:

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.

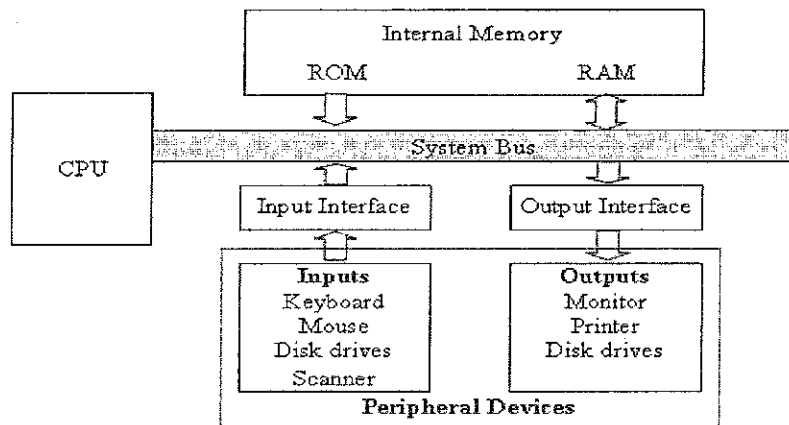
Một camera đặt phía dưới chuột liên tục chụp ảnh của bề mặt di chuột nhờ ánh sáng phản xạ. Tốc độ chụp là khoảng 1500 ảnh/giây.

IC điều khiển chuột sẽ phân tích và so sánh các ảnh kề nhau và qua đó phát hiện ra chuyển động chuột.

Tín hiệu biểu diễn chuyển động chuột do IC điều khiển chuột sinh ra được chuyển cho máy tính xử lý.

• Câu hỏi loại 2 điểm

**Câu hỏi 2.1:** *Nêu sơ đồ khối chức năng và chức năng chính của các thành phần trong một hệ thống máy tính ?*



**TL :**

Sơ đồ khối chức năng hệ thống máy tính như hình bên:

**Bộ xử lý trung tâm** (Central Processing Unit - CPU) có chức năng: đọc lệnh từ bộ nhớ, giải mã và thực hiện lệnh.

CPU bao gồm:

- Bộ điều khiển (Control Unit - CU)
- Bộ tính toán số học và logic (Arithmetic and Logic Unit - ALU)
- Các thanh ghi (Registers)

**Bộ nhớ trong** (Internal Memory) có chức năng: lưu trữ lệnh (instruction) và dữ liệu (data) cho CPU xử lý.

Bộ nhớ trong bao gồm:

- ROM (Read Only Memory): Lưu trữ lệnh và dữ liệu của hệ thống, thông tin trong ROM vẫn tồn tại khi mất nguồn nuôi.
- RAM (Random Access Memory): Lưu trữ lệnh và dữ liệu của hệ thống và người dùng; thông tin trong RAM sẽ mất khi mất nguồn nuôi.

**Các thiết bị vào ra** (Peripheral devices):

Thiết bị vào (Input devices): nhập dữ liệu và điều khiển. Bao gồm: Bàn phím (Keyboard), Chuột (Mice), Ổ đĩa (Disk drives), Máy quét (Scanner)...

Thiết bị ra (Output devices): kết xuất dữ liệu. Bao gồm: Màn hình (Monitor/screen), Máy in (Printer), Máy vẽ (Plotter), Ổ đĩa (Disk drives)...

**Bus hệ thống** (system bus):

Bus hệ thống là một tập các đường dây kết nối CPU với các thành phần khác của máy tính.

Bus hệ thống thường gồm:

- Bus địa chỉ (Address bus) – Bus A
- Bus dữ liệu (Data bus) – Bus D
- Bus điều khiển (Control bus) - Bus C

**Câu hỏi 2.2: Nêu sơ đồ và các đặc điểm của kiến trúc máy tính von-Neumann.**

**Kiến trúc máy tính von-Neumann hiện đại khác kiến trúc máy tính von-Neumann cổ điển ở những điểm chính nào ?**

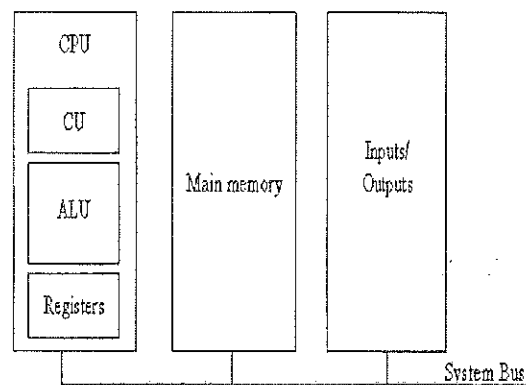
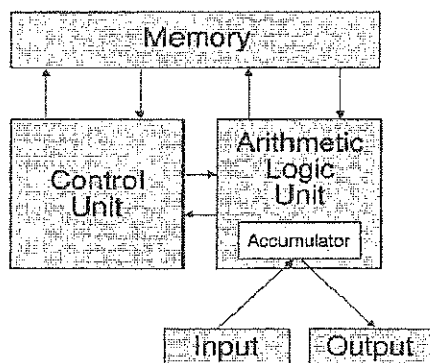
**TL :**

Kiến trúc von-Neumann dựa trên 3 khái niệm cơ sở:

- Lệnh và dữ liệu được lưu trữ trong bộ nhớ ghi chia sẻ;
- Bộ nhớ được đánh địa chỉ theo vùng, không phụ thuộc vào nội dung nó lưu trữ;
- Các lệnh của một chương trình được thực hiện tuần tự.

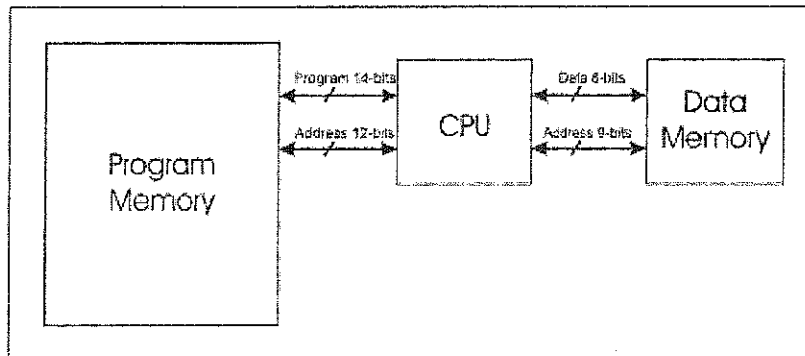
Các lệnh được thực hiện theo 3 giai đoạn (stages) chính:

- CPU đọc (fetch) lệnh từ bộ nhớ;
- CPU giải mã và thực hiện lệnh; nếu lệnh yêu cầu dữ liệu, CPU đọc dữ liệu từ bộ nhớ;
- CPU ghi kết quả thực hiện lệnh vào bộ nhớ (nếu có).



**Câu hỏi 2.3: Nêu sơ đồ và các đặc điểm của kiến trúc máy tính Harvard.**

Kiến trúc máy tính Harvard có những ưu điểm gì so với kiến trúc máy tính von-Neumann. Các máy tính hiện đại ngày nay sử dụng kiến trúc nào ?



**TL :** Bộ nhớ được chia thành 2 phần:

- Bộ nhớ lưu mã chương trình
- Bộ nhớ lưu dữ liệu

CPU sử dụng 2 hệ thống bus để giao tiếp với bộ nhớ:

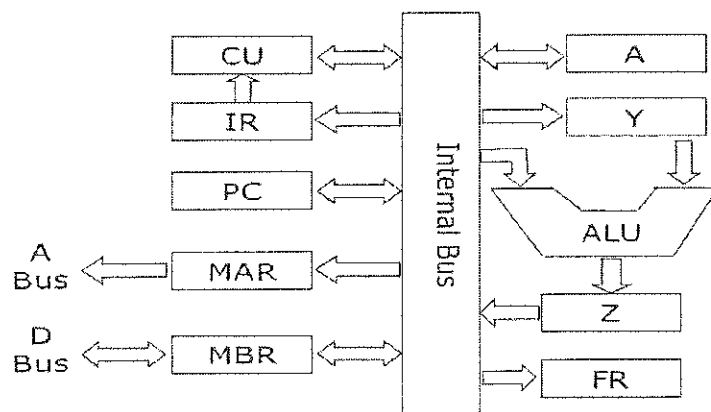
- Bus A, D và C giao tiếp với bộ nhớ chương trình
- Bus A, D và C giao tiếp với bộ nhớ dữ liệu.

So sánh Kiến trúc Harvard và Kiến trúc von-Neumann:

- Kiến trúc Harvard nhanh hơn kiến trúc von-Neumann do băng thông của bus lớn hơn.
- Hỗ trợ nhiều thao tác đọc/ghi bộ nhớ tại một thời điểm → giảm xung đột truy nhập bộ nhớ, đặc biệt khi CPU sử dụng kỹ thuật đường ống (pipeline).

Tuy nhiên hệ thống máy tính hiện nay phổ biến sử dụng kiến trúc von-Neumann hiện đại vì kiến trúc này ít phức tạp hơn, rẻ hơn so với hệ thống dùng kiến trúc Harvard, ngoài ra các máy tính đã phân cấp bộ nhớ nên tốc độ đã được cải thiện đáng kể. Hơn nữa, kiến trúc von-Neumann ra đời trước nên các nhà sản xuất máy tính đã phát triển công nghệ sản xuất trên nền tảng này, sự thay đổi là rất tốn kém.

**Câu hỏi 2.4: Nêu sơ đồ khối tổng quát và chu trình xử lý lệnh của CPU.**



**TL :**

CU : Bộ điều khiển

ALU : Bộ số học và logic

Internal Bus : Bus trong CPU

A : Thanh ghi tích lũy (Accumulator Register)

PC : Bộ đếm chương trình.

IR : Thanh ghi lệnh

MAR : Thanh ghi địa chỉ bộ nhớ.

MBR : Thanh ghi đệm dữ liệu.

Y, Z : Thanh ghi tạm thời.

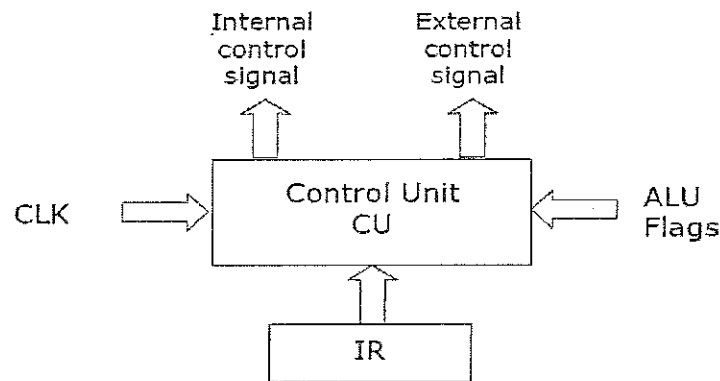
FR : Thanh ghi cờ.

### **Chu trình xử lý lệnh của CPU**

1. Khi một chương trình được thực hiện, hệ điều hành (OS - Operating System) nạp mã chương trình vào bộ nhớ trong.
2. Địa chỉ của ô nhớ chứa lệnh đầu tiên của chương trình được nạp vào bộ đếm chương trình PC.
3. Địa chỉ ô nhớ chứa lệnh từ PC được chuyển đến bus A thông qua thanh ghi MAR.
4. Bus A chuyển địa chỉ ô nhớ đến đơn vị quản lý bộ nhớ (MMU - Memory Management Unit).
5. MMU chọn ra ô nhớ và thực hiện lệnh đọc nội dung ô nhớ.
6. Lệnh (trong ô nhớ) được chuyển ra bus D và tiếp theo được chuyển tiếp đến thanh ghi MBR;

7. MBR chuyển lệnh đến thanh ghi lệnh IR; IR chuyển lệnh vào bộ điều khiển CU.
8. CU giải mã lệnh và sinh các tín hiệu điều khiển cần thiết, yêu cầu các bộ phận chức năng như ALU thực hiện lệnh.
9. Giá trị địa chỉ trong bộ đếm PC được tăng lên 1 đơn vị và nó trở về địa chỉ của ô nhớ chứa lệnh tiếp theo.
10. Các bước từ 3-9 được lặp lại với tất cả các lệnh của chương trình.

**Câu hỏi 2.5:** *Nêu sơ đồ khối và chức năng của các khối điều khiển (CU) và khối tính toán số học và logic (ALU).*



**TL :**

#### **Sơ đồ khối đơn vị điều khiển CU**

Đơn vị điều khiển CU (Control Unit) điều khiển toàn bộ các hoạt động của CPU theo xung nhịp đồng hồ

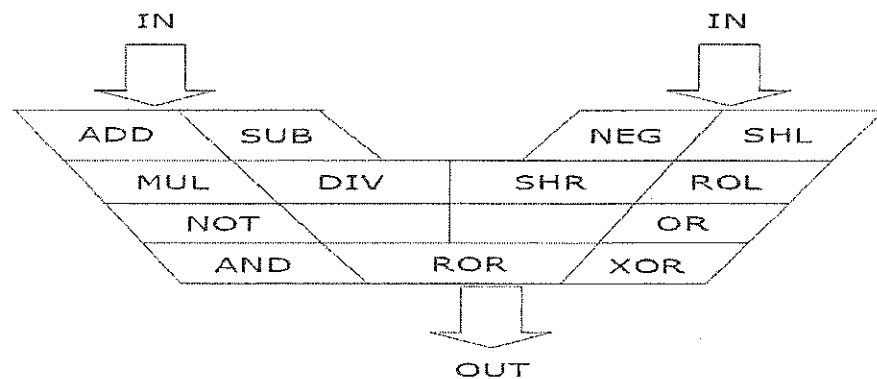
CU nhận 3 tín hiệu đầu vào:

- Lệnh từ IR
- Giá trị các cờ trạng thái
- Xung đồng hồ

CU sinh 2 nhóm tín hiệu đầu ra:

- Nhóm tín hiệu điều khiển các bộ phận bên trong CPU;
- Nhóm tín hiệu điều khiển các bộ phận bên ngoài CPU

CU sử dụng nhịp đồng hồ để đồng bộ các đơn vị chức năng trong CPU và giữa CPU với các bộ phận bên ngoài.



### Đơn vị số học & logic ALU

ALU (Arithmetic and Logic Unit) bao gồm một loạt 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 vào IN: nhận toán hạng từ các thanh ghi.
- 1 cổng ra OUT: kết nối với bus trong để chuyển kết quả đến thanh ghi.

### Câu hỏi 2.6: Lệnh máy tính là gì ? ...

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

**TL :**

Lệnh máy tính (Computer Instruction) là một từ nhị phân (binary word) được gán một nhiệm vụ cụ thể. Các lệnh của chương trình được lưu trong bộ nhớ và chúng lần lượt được CPU đọc, giải mã và thực hiện.

Chu kỳ thực hiện lệnh (Instruction execution cycle): là khoảng thời gian mà CPU thực hiện xong một lệnh:

- Một chu kỳ thực hiện lệnh có thể gồm một số giai đoạn thực hiện lệnh.
- Một giai đoạn thực hiện lệnh có thể gồm một số chu kỳ máy.
- Một chu kỳ máy có thể gồm một số chu kỳ đồng hồ.

Việc thực hiện lệnh có thể được chia thành các pha (phase) hay giai đoạn (stage). Mỗi lệnh có thể được thực hiện theo 4 giai đoạn:

- Đọc lệnh (Instruction fetch - IF): lệnh được đọc từ bộ nhớ về CPU;
- Giải mã (Instruction decode - ID): CPU giải mã lệnh;
- Thực hiện (Instruction execution – EX): CPU thực hiện lệnh;
- Lưu kết quả (Write back - WB): kết quả thực hiện lệnh (nếu có) được lưu vào bộ nhớ.

Dạng tổng quát của lệnh gồm 2 thành phần chính:

- Mã lệnh (Opcode - operation code): mỗi lệnh có mã lệnh riêng
- Địa chỉ của các toán hạng (Addresses of Operands): mỗi lệnh có thể gồm một hoặc nhiều toán hạng.

Opcode	Addresses of Operands
--------	-----------------------

Opcode	Source addr.	Destination addr.
--------	--------------	-------------------

Có thể có các dạng địa chỉ toán hạng sau: 3 địa chỉ; 2 địa chỉ; 1 địa chỉ; 1,5 địa chỉ; 0 địa chỉ.

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

**TL :**

**Toán hạng 3 địa chỉ:**

Dạng: opcode addr1, addr2, addr3

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

Ví dụ:

- ADD R1, R2, R3;  $R1 \leftarrow R2 + R3$   
R2 cộng với R3, kết quả gán vào R1. ( $R_i$  là thanh ghi của CPU)
- ADD A, B, C;  $M[A] \leftarrow M[B] + M[C]$   
A, B, C là địa chỉ các ô nhớ.

**Toán hạng 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;  $R1 \leftarrow R1 + R2$



R1 cộng với R2, kết quả gán vào R1. (R1 là thanh ghi của CPU)

- $\text{ADD A, B; } M[A] \leftarrow M[B] + M[B]$

A, B là địa chỉ các ô nhớ.

### Toán hạng 1 địa chỉ:

Dạng: opcode addr1

Địa chỉ addr1 tham chiếu đến một ô nhớ hoặc một thanh ghi.

Ở dạng 1 địa chỉ, thanh ghi tích lũy A (Accumulator) được sử dụng như địa chỉ addr2 trong dạng 2 địa chỉ.

Ví dụ:

- $\text{ADD R1; } R_{acc} \leftarrow R_{acc} + R1$   
R1 cộng với Racc, kết quả gán vào Racc. R1 là thanh ghi của CPU.
- $\text{ADD A; } R_{acc} \leftarrow R_{acc} + M[A]$   
A là địa chỉ một ô nhớ.

### Toán hạng 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 hỗn hợp giữa ô nhớ và thanh ghi.

Ví dụ:

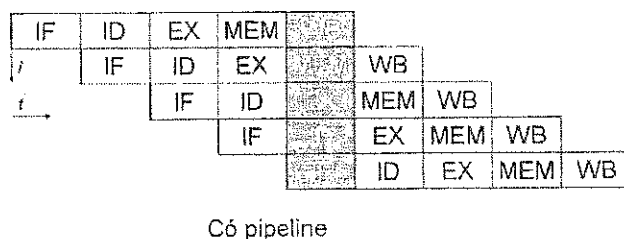
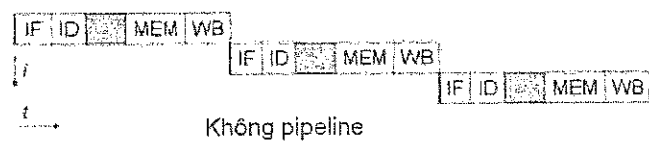
- $\text{ADD A, R1; } M[A] \leftarrow M[A] + R1$   
Nội dung ô nhớ A cộng với R1, kết quả lưu vào ô nhớ A  
R1 là thanh ghi của CPU và A là địa chỉ một ô nhớ.

### Toán hạng 0 địa chỉ:

Được sử dụng trong các lệnh thao tác với ngăn xếp: PUSH và POP.

## Câu hỏi 2.8: Cơ chế xử lý xen kẽ dòng lệnh (ống lệnh – pipeline) là gì ?

Nêu các đặc điểm của cơ chế ống lệnh.



**TL :**

**Cơ chế ống lệnh** (pipeline) hay còn gọi là cơ chế thực hiện xen kẽ các lệnh của chương trình là một phương pháp thực hiện lệnh tiên tiến, cho phép đồng thời thực hiện nhiều lệnh, giảm thời gian trung bình thực hiện mỗi lệnh và như vậy tăng được hiệu năng xử lý lệnh của CPU.

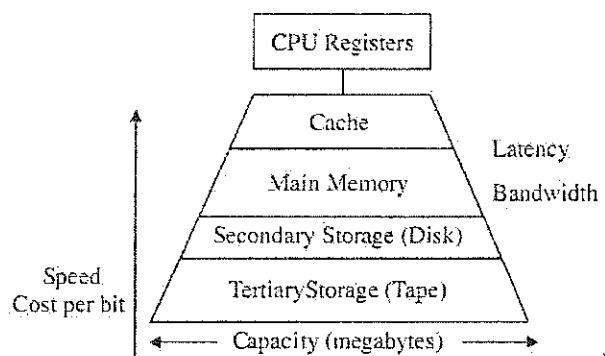
Việc thực hiện lệnh được chia thành một số giai đoạn và mỗi giai đoạn được thực thi bởi một đơn vị chức năng khác nhau của CPU. Nhờ vậy CPU có thể tận dụng tối đa năng lực xử lý của các đơn vị chức năng của mình, giảm thời gian chờ cho từng đơn vị chức năng. Có nhiều lệnh đồng thời được thực hiện gối nhau trong CPU và hầu hết các đơn vị chức năng của CPU liên tục tham gia vào quá trình xử lý lệnh. Số lượng lệnh được xử lý đồng thời đúng bằng số giai đoạn thực hiện lệnh.

**Đặc điểm của cơ chế ống lệnh:**

- Là dạng xử lý song song ở mức lệnh (instruction level parallelism (ILP)).
- Một pipeline là đầy đủ (*fully pipelined*) khi nó luôn tiếp nhận một lệnh mới tại mỗi chu kỳ đồng hồ.
- Ngược lại, một pipeline là không đầy đủ khi có một số chu kỳ trễ trong tiến trình thực hiện.
- Số lượng các giai đoạn (stages) trong pipeline phụ thuộc vào thiết kế vi xử lý:
  - 2,3, 5 giai đoạn (pipeline đơn giản)
  - 14 giai đoạn (PII, PIII)
  - 20-31 giai đoạn (P4)
  - 12-15 giai đoạn (Core)

**Câu hỏi 2.9: Nêu cấu trúc phân cấp của hệ thống bộ nhớ máy tính.**

Tại sao cấu trúc phân cấp của hệ thống bộ nhớ có thể giúp tăng hiệu năng và giảm giá thành sản xuất máy tính ?



**TL :**

**Cấu trúc phân cấp hệ thống bộ nhớ** được thể hiện như hình bên.

Trong cấu trúc phân cấp hệ thống nhớ, dung lượng các thành phần tăng theo chiều từ các thanh ghi của CPU đến bộ nhớ ngoài. Ngược lại, tốc độ truy nhập hay băng thông và giá thành một đơn vị nhớ tăng theo chiều từ bộ nhớ ngoài đến các thanh ghi của CPU. Như vậy, các thanh ghi của CPU có dung lượng nhỏ nhất nhưng có tốc độ truy cập nhanh nhất và cũng có giá thành cao nhất. Bộ nhớ ngoài có dung lượng lớn nhất, nhưng tốc độ truy cập thấp nhất. Bù lại, bộ nhớ ngoài có giá thành rẻ nên có thể được sử dụng với dung lượng lớn.

CPU registers (các thanh ghi của CPU):

- Dung lượng rất nhỏ, khoảng từ vài chục bytes đến vài KB
- Tốc độ truy nhập rất cao (các thanh ghi hoạt động với tốc độ của CPU); thời gian truy nhập khoảng 0,25ns
- Giá thành đắt
- Sử dụng để lưu toán hạng đầu vào và kết quả của các lệnh.
- Dung lượng tương đối nhỏ (khoảng 64KB đến 32MB)
- Tốc độ truy nhập cao; thời gian truy nhập khoảng 1-5ns
- Giá thành đắt
- Còn được gọi là “bộ nhớ thông minh” (smart memory)
- Sử dụng để lưu lệnh và dữ liệu cho CPU xử lý.

Main memory (bộ nhớ chính):

- Gồm ROM và RAM, có kích thước khá lớn; với hệ thống 32 bit, dung lượng khoảng 256MB-4GB
- Tốc độ truy nhập chậm; thời gian truy nhập khoảng 50-70ns
- Giá thành tương đối rẻ
- Sử dụng để lưu lệnh và dữ liệu của hệ thống và của người dùng.

Secondary memory (bộ nhớ thứ cấp – bộ nhớ ngoài):

- Có dung lượng rất lớn, khoảng từ 20GB-1000GB
- Tốc độ truy nhập rất chậm; thời gian truy nhập khoảng 5ms
- Giá thành rẻ
- Sử dụng để lưu dữ liệu lâu dài dưới dạng các tệp (files).

**Vai trò của việc phân cấp hệ thống bộ nhớ:**

Tăng hiệu năng hệ thống

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

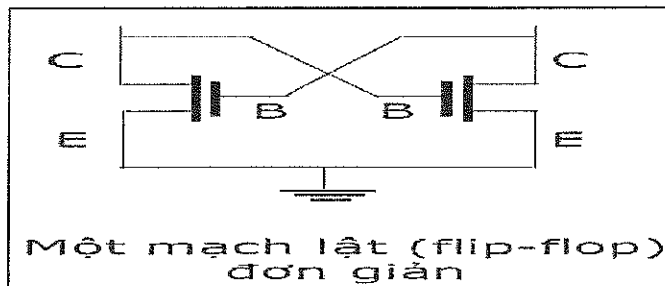
Giảm giá thành sản xuất

- Các thành phần đắt tiền (thanh ghi và cache) được sử dụng với dung lượng nhỏ;
- Các thành phần rẻ tiền hơn (bộ nhớ chính và bộ nhớ ngoài) được sử dụng với dung lượng lớn;

→ Tổng giá thành của hệ thống nhớ theo mô hình phân cấp sẽ rẻ hơn so với hệ thống nhớ không phân cấp có cùng tốc độ.

**Câu hỏi 2.10: Phân biệt bộ nhớ RAM tĩnh và RAM động.**

Tại sao bộ nhớ RAM động cần quá trình làm tươi và RAM động thường rẻ hơn RAM tĩnh ?

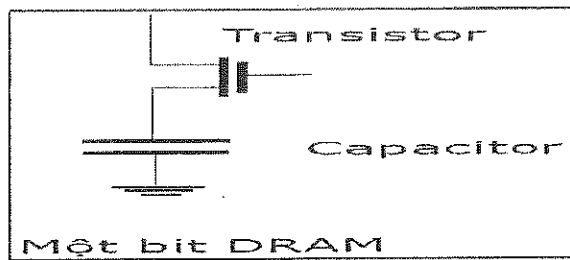


**TL :**

SRAM (Static Ram) là loại RAM sử dụng một mạch lật trigô lưỡng ổn để lưu một bit thông tin. Thông tin trong SRAM luôn ổn định và không phải “làm tươi” định kỳ. Tốc độ truy nhập SRAM cũng nhanh hơn nhiều so với DRAM.

Mỗi mạch lật lưu 1 bit thường sử dụng 6, 8 hoặc 10 transistors (gọi là mạch 6T, 8T hoặc 10T);

SRAM thường có tốc độ truy nhập nhanh do các bit của SRAM có cấu trúc đối xứng, Các mạch nhớ SRAM chấp nhận tất cả các chân địa chỉ tại một thời điểm (không dồn kênh).



DRAM (Dynamic Ram): Mỗi bit DRAM dựa trên một tụ điện và một transistor. Hai mức tích điện của tụ biểu diễn 2 mức logic 0 và 1:

- Không tích điện: mức 0
- Tích đầy điện: mức 1.

Thông tin lưu trong các bit DRAM không ổn định và phải được “làm tươi” định kỳ. DRAM chậm hơn nhưng rẻ hơn SRAM.

**RAM động cần quá trình làm tươi** vì tụ thường tự phóng điện, điện tích trong tụ có xu hướng bị tổn hao theo thời gian nên:

- Cần nạp lại thông tin trong tụ thường xuyên để tránh mất thông tin.
- Việc nạp lại thông tin cho tụ là quá trình làm tươi (refresh), phải theo định kỳ.

**DRAM thường rẻ hơn SRAM do:**

- Mỗi bit SRAM dùng nhiều transistor hơn so với 1 bit DRAM
- Do cấu trúc bên trong của SRAM bit phức tạp hơn nên mật độ cấy linh kiện trong SRAM thường thấp.

**Câu hỏi 2.11: Bộ nhớ cache là gì ?**

**Nêu vai trò của cache. Giải thích hai nguyên lý hoạt động của cache.**

**TL :**

Cache hay còn gọi là bộ nhớ đệm là một thành phần trong hệ thống nhớ phân cấp của máy tính, cache đóng vai trò 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 suất 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 suất 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;

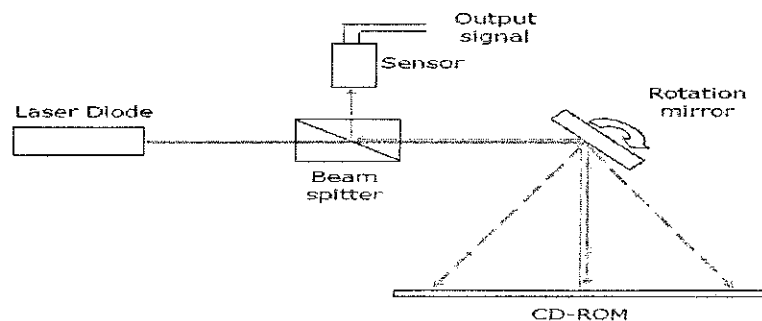
Cache đọc cả khối lệnh từ bộ nhớ chính → phủ được cả khối lệnh của vòng lặp.

**Câu hỏi 2.12: So sánh các chuẩn ghép nối ổ đĩa cứng IDE, SATA và SCSI.**

**TL :**

	IDE	SATA	SCSI
<b>Giống nhau</b>	Đều là các giao diện ghép nối ổ đĩa cứng với máy tính		
<b>Khác nhau</b>	ATA/IDE sử dụng cáp dẹt 40 hoặc 80 sợi để ghép nối ổ cứng với bảng mạch chính;	SATA sử dụng cùng tập lệnh mức thấp như ATA nhưng SATA sử dụng đường truyền tin nối tiếp tốc độ cao qua 2 đôi dây;	SCSI là một tập các chuẩn về kết nối vật lý và truyền dữ liệu giữa máy tính và thiết bị ngoại vi;
	Mỗi cáp thường hỗ trợ ghép nối với 2 ổ đĩa		Tất cả các thiết bị SCSI đều kết nối đến bus SCSI theo cùng một kiểu
<b>Ưu điểm</b>		Truyền dữ liệu nhanh và hiệu quả Cắm nóng Sử dụng ít dây hơn ATA để truyền dữ liệu.	SCSI cung cấp tốc độ truyền dữ liệu và tính ổn định rất cao; Cắm nóng
<b>Nhược điểm</b>	Tốc độ truyền dữ liệu còn chậm		Các ổ cứng SCSI thường rất đắt

**Câu hỏi 2.13: Trình bày nguyên lý đọc thông tin trên đĩa CD.**



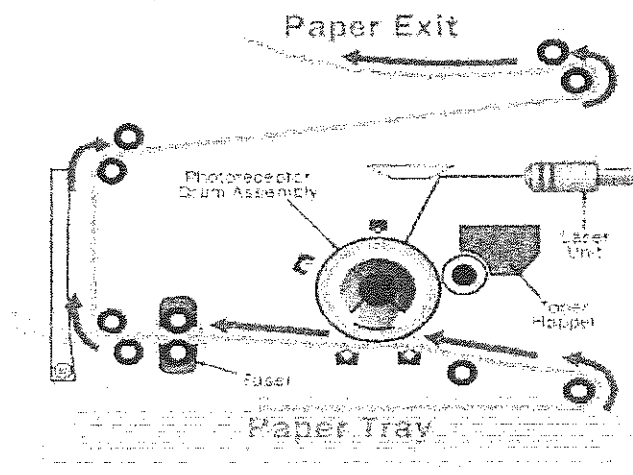
**TL :**

**Nguyên lý đọc CD-ROM**

- Tia laser từ diốt phát laser đi qua bộ tách tia 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 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 mức tín hiệu ra.



**Câu hỏi 2.14: Nêu nguyên lý hoạt động của máy in laser.**



**TL :**

Máy in laser hoạt động dựa trên nguyên tắc chụp ảnh điện tử bằng tia laser. Cụ thể:

- Trống cảm quang được nạp một lớp điện tích nhờ 1 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; 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 hỏi 2.15: Nêu nguyên lý tạo hình ảnh của màn hình LCD.**

**TL :**

Các tinh thể lỏng không thể tự phát sáng, chúng có khả năng điều khiển lượng ánh sáng đi qua theo nhiệt độ và dòng điện.

Có 2 loại LCD dựa trên phương pháp điều khiển:

- LCD ma trận thụ động (Passive matrix):

Sử dụng lưới hoặc ma trận để định nghĩa từng điểm ảnh (pixel) bởi hàng và cột của nó. Một điểm ảnh (giao giữa 1 hàng và 1 cột) được kích hoạt khi điện áp được đặt vào cột và dòng tương ứng được nối đất.

- LCD ma trận chủ động (Active matrix):

Sử dụng một TFT (Thin Film Transistor) để điều khiển một phần tử LCD. Các TFT hoạt động tương tự như các bộ chuyển mạch.

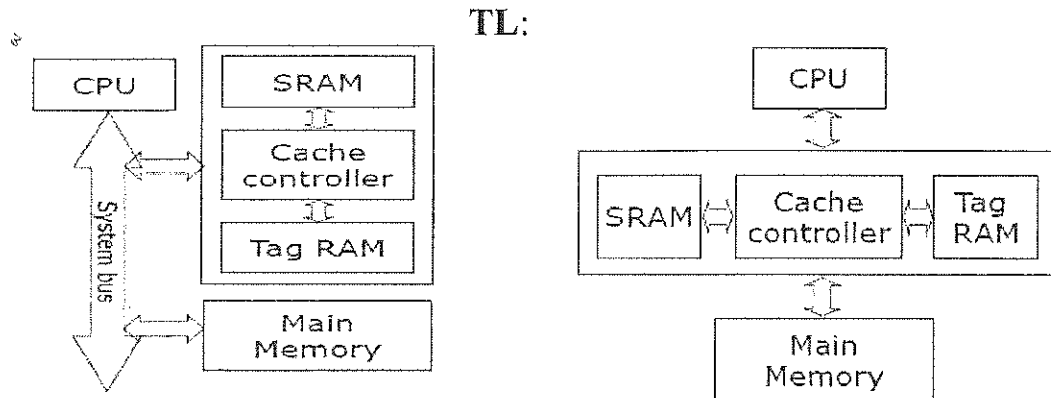
Nguyên lý TFT LCD:

- 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 (Indium Tin Oxide).
- Các phần tử tinh thể lỏng được sắp đặt theo các hướng khác nhau theo sự thay đổi điện 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 (còn gọi là mức xám) của ảnh hiển thị;
- Màu của hình ảnh được tạo bởi một lớp lọc màu;
- Mức xám của các điểm ảnh được thiết lập theo mức điện áp của tín hiệu video đưa vào.

• Câu hỏi loại 3 điểm

**Câu hỏi 3.1: Nêu sơ đồ và 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?



**Cache – Look aside**

SRAM: RAM lưu dữ liệu cache

Tag RAM: RAM lưu địa chỉ bộ nhớ

Cache và bộ nhớ chính cùng kết nối với bus hệ thống;

Cache và bộ nhớ chính “thấy” chu kỳ bus của CPU tại cùng một thời điểm;

Ưu điểm:

- Thiết kế đơn giản
- Miss nhanh

Nhược điểm:

- Hit chậm

**Cache – Look through**

SRAM: RAM lưu dữ liệu cache

Tag RAM: RAM lưu địa chỉ bộ nhớ

Cache nằm giữa CPU và bộ nhớ chính;

Cache “thấy” chu kỳ bus của CPU trước, sau đó nó chuyển chu kỳ bus cho bộ nhớ chính;

Ưu điểm:

- Hit nhanh

Nhược điểm:

- Thiết kế phức tạp
- Đắt tiền
- Miss chậm

**Câu hỏi 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?

**TL :**

Tiêu chí	Ánh xạ trực tiếp	Ánh xạ kết hợp đầy đủ	Ánh xạ tập kết hợp
Giống	Cùng là phương pháp ánh xạ cache-mem Cache được chia thành các dòng		
Khác nhau			
Bộ nhớ	m trang	1 trang	m trang
Ánh xạ	Ánh xạ dòng của trang đến dòng của đường (ánh xạ cố định)	Một dòng trong bộ nhớ có thể được ánh xạ vào một dòng bất kỳ trong cache;	Ánh xạ trang đến đường (ánh xạ mềm dẻo): 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).
Ưu điểm	Thiết kế đơn giản Nhanh do ánh xạ là cố định: khi biết địa chỉ ô nhớ có thể tìm được vị trí của nó trong cache rất nhanh chóng.	Giảm được xung đột do ánh xạ là không cố định Hệ số Hit cao hơn ánh xạ trực tiếp.	Nhanh do ánh xạ trực tiếp được sử dụng cho ánh xạ dòng (chiếm số lớn ánh xạ); Giảm xung đột do ánh xạ từ các trang bộ nhớ đến các đường cache là mềm dẻo. Hệ số Hit cao hơn.

Nhược điểm	Do ánh xạ cố định nên dễ gây xung đột Hệ số hit không cao.	Chậm do cần phải tìm địa chỉ ô nhớ trong cache Phức tạp do cần có n bộ so sánh địa chỉ bộ nhớ trong cache. Thường được sử dụng với cache có dung lượng nhỏ.	Phức tạp trong thiết kế và điều khiển vì cache được chia thành một số đường.
------------	---	---	--

**Câu hỏi 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 ?**

**TL :**

Đọc thông tin:

+Trường hợp hit (mẫu tin cần đọc có trong cache)

- Mẫu tin được đọc từ cache vào CPU;
- Bộ nhớ chính không tham gia.

+Trường hợp miss (mẫu tin cần đọc không có trong cache)

- Mẫu tin trước hết được đọc từ bộ nhớ chính vào cache;
- Sau đó nó được chuyển từ cache vào CPU.

→đây là trường hợp miss penalty: thời gian truy nhập mẫu tin bằng tổng thời gian truy nhập cache và bộ nhớ chính.

Ghi thông tin:

+Trường hợp hit (mẫu tin cần ghi có trong cache)

- Ghi thẳng (write through): mẫu tin được ghi đồng thời ra cache và bộ nhớ chính;
- Ghi trở (write back): mẫu tin trước hết được ghi ra cache và dòng chứa mẫu tin được ghi ra bộ nhớ chính khi dòng đó bị thay thế.

+Trường hợp miss (mẫu tin cần ghi không có trong cache)

- Ghi có đọc lại (write allocate / fetch on write): mẫu tin trước hết được ghi ra bộ nhớ chính và sau đó dòng chứa mẫu tin được đọc vào cache;
  - Ghi không đọc lại (write non-allocate): mẫu tin chỉ được ghi ra bộ nhớ chính (dòng chứa mẫu tin không được đọc vào cache).
- 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 (LRULeast Recently Used):

Các dòng cache ít được sử dụng gần đây nhất được lựa chọn để thay thế.

Ưu:

- Có hệ số miss thấp nhất so với thay thế ngẫu nhiên và thay thế FIFO
- Do thay thế LRU có xem xét đến các dòng đang được sử dụng

### ***Câu hỏi 3.4: RAID là gì?***

***Tại sao RAID có thể nâng cao được tính tin cậy và tốc độ truy nhập 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?***

***TL :***

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:

Một mảng của các đĩa cứng;

Các đĩa cứng theo chuẩn SATA và SCSI mới hỗ trợ tạo RAID.

***Hai kỹ thuật chính được sử dụng trong RAID:***

Tạo lát đĩa (Disk Stripping):

Ghi: Dữ liệu được chia thành các khối, mỗi khối được ghi đồng thời vào một đĩa độc lập;

Đọc: Các khối dữ liệu được đọc đồng thời ở các đĩa độc lập, và được ghép lại tạo dữ liệu hoàn chỉnh.

→ Tốc độ truy nhập được cải thiện.

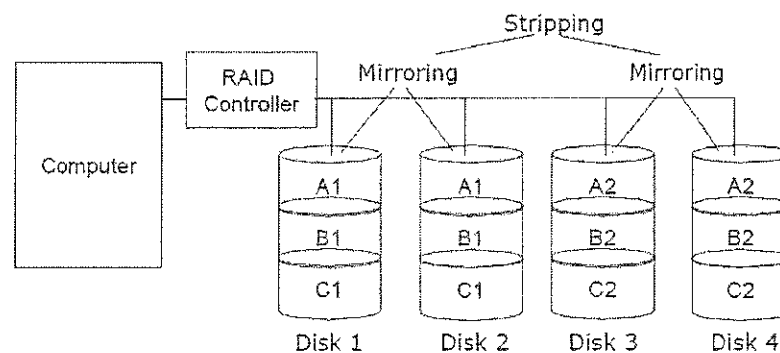
Soi gương đĩa (Disk Mirroring):

Ghi: Dữ liệu được chia thành các khối, mỗi khối được ghi đồng thời vào nhiều đĩa độc lập;

Tại mọi thời điểm ta luôn có nhiều hơn 1 bản sao vật lý của dữ liệu.

→ Tính tin cậy được cải thiện.

RAID 10



RAID 10 - Disk mirroring & stripping

Ưu điểm của RAID 10 là đạt được cả tốc độ cao và tính tin cậy cao, nên rất phù hợp với các hệ thống máy chủ đòi hỏi tính an toàn cao, hiệu năng lớn như máy chủ cơ sở dữ liệu. Dung lượng RAID 10 bằng một nửa tổng dung lượng các đĩa độc lập tham gia tạo RAID.

Nhược điểm duy nhất của RAID 10 là giá thành cao.

**Câu hỏi 3.5: Nêu các đặc điểm chính của kiến trúc bus PCI và PCI-Express.**

**Tại sao bus PCI-Express có khả năng hỗ trợ nhiều cặp thiết bị truyền dữ liệu đồng thời với tốc độ cao?**

**TL :**

**Bus PCI Express – Kiến trúc**

PCI Express được cấu trúc từ các liên kết nối tiếp điểm đến 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 cắm PCI Express vật lý có thể chứa từ 1 đến 32 làn.
- PCI Express sử dụng giao thức truyền nối tiếp và tránh được vấn đề timing skew (lệch thời gian) – một trong các yếu tố làm giảm tốc độ;
- Các loại bus song song (ISA, PCI, AGP) đòi hỏi tất cả các bit của một đơn vị dữ liệu phải đến đích tại một thời điểm;
- Do vấn đề timing skew, các bit của một đơn vị dữ liệu có thể không đến đích đồng thời gây khó khăn cho việc khôi phục đơn vị dữ liệu;
- Phương thức truyền nối tiếp không gặp phải vấn đề timing skew do giao thức này không đòi hỏi tất cả các bit của một đơn vị dữ liệu phải đến đích tại một thời điểm.

**Câu hỏi 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 R4, R4, #300 ; R4 ← R4+300**

**ADD R1, R2, R3 ; R1 ← R2+R3**

**CMP R1, #100 ; so sánh R1 với 100**

**SUB R5, #2000; R5 ← 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 (WB).**

**TL :**

a. Cơ chế ống lệnh (pipeline) của CPU thường gặp phải 3 vấn đề:

1. **Vấn đề xung đột tài nguyên**



Vấn đề xung đột tài nguyên xảy ra khi hệ thống không cung cấp đủ tài nguyên phần cứng phục vụ CPU thực hiện đồng thời nhiều lệnh trong cơ chế ống lệnh.

## 2. Vấn đề tranh chấp dữ liệu

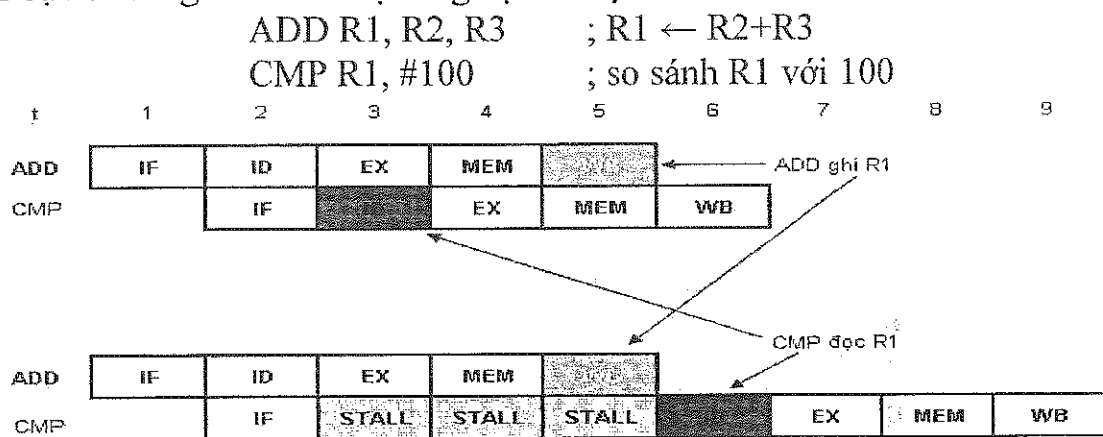
Tranh chấp dữ liệu cũng là một trong các vấn đề lớn của cơ chế ống lệnh và tranh chấp dữ liệu kiểu *đọc sau khi ghi* (RAW – Read After Write) là dạng xung đột dữ liệu hay gặp nhất.

## 3. Vấn đề nảy sinh do các lệnh rẽ nhánh

Theo thống kê, tỷ lệ các lệnh rẽ nhánh trong chương trình khoảng 10-30%. Do lệnh rẽ nhánh thay đổi nội dung của bộ đếm chương trình, chúng có thể phá vỡ tiến trình thực hiện tuần tự các lệnh trong ống lệnh vì lệnh được thực hiện sau lệnh rẽ nhánh có thể không phải là lệnh liên tiếp sau nó mà là một lệnh ở vị trí khác. Như vậy, do kiểu thực hiện gói đầu, các lệnh liên tiếp sau lệnh rẽ nhánh đã được nạp và thực hiện dở dang trong ống lệnh sẽ bị đẩy ra làm cho ống lệnh bị trống rỗng và hệ thống phải bắt đầu nạp mới các lệnh từ địa chỉ đích rẽ nhánh.

### b. Hướng giải quyết xung đột dữ liệu trong cơ chế Pipeline

Đoạn chương trình trên bị xung đột dữ liệu ở chỗ :



Đây là xung đột tranh chấp dữ liệu kiểu *đọc sau khi ghi* (RAW – Read After Write)

Để khắc phục xung đột dữ liệu trong đoạn chương trình trên ta có thể chèn thêm các lệnh NO-OP ở giữa 2 lệnh ADD và CMP.

t	1	2	3	4	5	6	7	8	9
ADD	IF	ID	EX	MEM					
NO-OP		NO-OP	NO-OP	NO-OP	NO-OP	NO-OP			
NO-OP			NO-OP	NO-OP	NO-OP	NO-OP	NO-OP		
NO-OP				NO-OP	NO-OP	NO-OP	NO-OP	NO-OP	
CMP					IF		EX	MEM	WB

**Câu hỏi 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 ;  $R4 \leftarrow R4 + 300$**

**ADD R1, R1, R3 ;  $R1 \leftarrow R1 + R3$**

**SUB R1, R1, #100 ;  $R1 \leftarrow R1 - 100$**

**SUB R5, #2000 ;  $R5 \leftarrow R5 - 2000$**

**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 (WB).**

**TL :**

a. Cơ chế ống lệnh (pipeline) của CPU thường gặp phải 3 vấn đề:

1. **Vấn đề xung đột tài nguyên**

Vấn đề xung đột tài nguyên xảy ra khi hệ thống không cung cấp đủ tài nguyên phần cứng phục vụ CPU thực hiện đồng thời nhiều lệnh trong cơ chế ống lệnh.

2. **Vấn đề tranh chấp dữ liệu**

Tranh chấp dữ liệu cũng là một trong các vấn đề lớn 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.

3. **Vấn đề nảy sinh do các lệnh rẽ nhánh**

Theo thống kê, tỷ lệ các lệnh rẽ nhánh trong chương trình khoảng 10-30%. Do lệnh rẽ nhánh thay đổi nội dung của bộ đếm chương trình, chúng có thể phá vỡ tiến trình thực hiện tuần tự các lệnh trong ống lệnh vì lệnh được thực hiện sau lệnh rẽ nhánh có thể không phải là lệnh liên sau nó mà là một lệnh ở vị trí khác. Như vậy, do kiểu thực hiện gộp đầu, các lệnh liên sau lệnh rẽ nhánh đã được nạp và thực hiện dở dang trong

ống lệnh sẽ bị đẩy ra làm cho ống lệnh bị trống rỗng và hệ thống phải bắt đầu nạp mới các lệnh từ địa chỉ đích rẽ nhánh.

#### B, Hướng giải quyết xung đột dữ liệu trong cơ chế Pipeline

Đoạn chương trình trên bị xung đột dữ liệu ở chỗ :

***ADD R1, R1, R3 ; R1 ← R1+R3***

***SUB R1, R1, #100 ; R1 ← R1 – 100***

Đây là xung đột tranh chấp dữ liệu kiểu *đọc sau khi ghi* (RAW – Read After Write)

Để khắc phục xung đột dữ liệu trong đoạn chương trình trên ta có thể chèn thêm các lệnh NO-OP ở giữa 2 lệnh ADD và SUB.

<Tương tự bài 3.6>

**Câu hỏi 3.8: Cho đoạn chương trình sau (R1, R2 là các thanh ghi):**

Lệnh	Ý nghĩa	Chế độ	GT R2	
LOAD R2, #400	Nạp giá trị 400 vào thanh ghi R2	Tức thì (immediate)	400	
LOAD R1, #1200	Nạp giá trị 1200 vào thanh ghi R1	Tức thì (immediate)	400	
STORE (R1), R2	Lưu nội dung của thanh ghi R2 vào ô nhớ có địa chỉ chứa trong thanh ghi R1	Gián tiếp (Indirect)	400	
SUBTRACT R2, #15	$R2 \leftarrow R2 - 15$ : Lấy nội dung thanh ghi R2 trừ đi 15, kết quả lưu vào thanh ghi R2	Tức thì (immediate)	385	
ADD 1200, #10	Lấy nội dung của ô nhớ có địa chỉ 1200 cộng với 10, kết quả lưu vào ô nhớ có địa chỉ 1200	Trực tiếp (Direct)	385	

ADD R2, (R1)	Lấy nội dung của thanh ghi R2 cộng với nội dung của ô nhớ có địa chỉ trong thanh ghi R1, kết quả lưu vào thanh ghi R2	Gián tiếp (Indirect)	795	
--------------	---	----------------------	-----	--

**Câu hỏi 3.9: Cho đoạn chương trình sau (R1, R2 là các thanh ghi):**

Lệnh	Ý nghĩa	Chế độ	GT R2	
LOAD R2, #500	$R2 \leftarrow 500$	Tức thì (immediate)	500	
LOAD R1, #2000	$R1 \leftarrow 2000$	Tức thì (immediate)	500	
STORE (R1), R2	Lưu nội dung của thanh ghi R2 vào ô nhớ có địa chỉ chứa trong thanh ghi R1	Gián tiếp (Indirect)	500	
ADD 2000, #30	Lấy nội dung của ô nhớ có địa chỉ 2000 cộng với 30, kết quả lưu vào ô nhớ có địa chỉ 2000	Trực tiếp (Direct)	500	
SUBTRACT R2, #15	$R2 \leftarrow R2 - 15$ : Lấy nội dung thanh ghi R2 trừ đi 15, kết quả lưu vào thanh ghi R2	Tức thì (immediate)	485	
ADD R2, (R1)	Lấy nội dung của thanh ghi R2 cộng với nội dung của ô nhớ có địa chỉ trong thanh ghi R1, kết quả lưu vào thanh ghi R2	Gián tiếp (Indirect)	1015	

**Câu hỏi 3.10:** Cho một dãy số nguyên gồm 10 phần tử lưu trong bộ nhớ bắt đầu từ địa chỉ 1000. Viết chương trình sử dụng tập lệnh của CPU tính:

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

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

**TL :**

CLEAR R0 ;

CLEAR R1 ;

LOAD R2, #10 ;

Câu a:/

Loop :

LOAD R3, 1000(R0) ;

BRANCH\_IF >0, Add ;

Continue :

INCREMENT R0 ;

DECREMENT R2 ;

BRANCH\_IF >0 Loop ;

JUMB Exit ;

Add :

ADD R1, R2 ;

JUMB Continue ;

Exit :

STORE 2000, R1 ;

Câu b:/

Loop :

LOAD R3, 1000(R0) ;

Add ;

Continue :

INCREMENT R0 ;

DECREMENT R2 ;

Loop ;

JUMB Exit ;

Add :

ADD R1, R2 ;

JUMB Continue ;

Exit :

STORE 2000, R1 ;

