

# MỤC LỤC

**Protected by PDF Anti-Copy Free**

Mục lục	I
Lời nói đầu	V
Các từ viết tắt	VI
<b>Chương 1: GIỚI THIỆU CHUNG VỀ KIẾN TRÚC MÁY TÍNH</b>	<b>1</b>
1.1. CÁC KHÁI NIỆM VÀ NGUYÊN CƠ BẢN	1
1.1.1. Mở đầu	1
1.1.2. Khái niệm máy tính	1
1.1.3. Kiến trúc máy tính và cấu trúc máy tính	2
1.2. CÁC THÀNH PHẦN CƠ BẢN CỦA MÁY TÍNH	3
1.2.1. Bộ nguồn	3
1.2.1.1. Nguồn cấp điện cho máy lớn	3
1.2.1.2. Nguồn pin cho máy tính xách tay	4
1.2.2. Bản mạch chính	2
1.2.2.1. Bộ xử lý trung tâm (CPU – Central Processing Unit)	4
1.2.2.2. Bộ nhớ cố định (ROM – Read Only Memory)	5
1.2.2.3. Bộ nhớ ghi/đọc (RAM – Random Access Memory)	5
1.2.2.4. Các bộ nhớ ngoài	5
1.2.3. Các thiết bị ngoại vi	6
1.2.3.1. Bàn phím (Keyboard)	6
1.2.3.2. Màn hình (Monitor)	6
1.2.3.3. Máy in (Printer)	7
1.2.3.4. Modem và các thiết bị ngoại vi khác	7
1.3. PHẦN MỀM MÁY TÍNH	7
1.4. LỊCH SỬ PHÁT TRIỂN MÁY TÍNH	8
1.5. KIẾN TRÚC MÁY TÍNH VON-NEUMANN	9
1.6. KIẾN TRÚC MÁY TÍNH HAVARD	10
1.7. ĐỊNH LUẬT MOORE	11
<b>Chương 2: BIỂU DIỄN THÔNG TIN TRONG MÁY TÍNH</b>	<b>13</b>
2.1. HỆ ĐẾM	13
2.1.1. Hệ thập phân	13
2.1.2. Hệ nhị phân	14
2.1.2.1. Khái niệm	14
2.1.2.2. Biến đổi từ nhị phân sang thập phân	15
2.1.2.3. Biến đổi thập phân sang nhị phân	14
2.1.3. Hệ thập lục phân	17
2.1.3.1. Khái niệm	17
2.1.3.2. Biến đổi thập lục phân sang thập phân	17
2.1.3.3. Biến đổi thập phân sang thập lục phân	18
2.1.3.4. Biến đổi thập lục phân sang nhị phân	18

2.1.3.5. Biến đổi nhị phân sang thập lục phân .....	19
2.2. BIỂU DIỄN DỮ LIỆU SỐ TRONG MÁY TÍNH .....	20
2.2.1. Nguyên tắc chung về mã hóa dữ liệu .....	20
2.2.2. Thứ tự lưu trữ các byte của dữ liệu .....	20
2.2.3. Biểu diễn số nguyên .....	21
2.2.3.1. Biểu diễn số nguyên dấu .....	21
2.2.3.2. Biểu diễn số nguyên không dấu .....	22
2.2.4. Các phép toán số học với số nguyên .....	23
2.2.5. Biểu diễn số thực .....	25
2.2.6. Biểu diễn ký tự .....	28
2.2.6.1. Bộ mã ASCII .....	28
2.2.6.2. Bộ mã Unicode .....	30
2.2.6.3. Mã BCD (Binary Coded Decimal) .....	30
<b>Chương 3: BỘ XỬ LÝ TRUNG TÂM CPU .....</b>	<b>33</b>
3.1. CẤU TRÚC, CHỨC NĂNG CỦA BỘ VI XỬ LÝ .....	33
3.1.1. Chức năng của bộ vi xử lý .....	33
3.1.2. Cấu trúc của bộ vi xử lý .....	33
3.2. CÁC THANH GHI .....	34
3.2.1. Các thanh ghi đa năng (general registers) .....	34
3.2.2. Các thanh ghi đoạn (segment registers) .....	35
3.2.3. Các thanh ghi con trỏ và chỉ số .....	36
3.2.4. Thanh ghi cờ FR (flag register) .....	37
3.3. ĐƠN VỊ SỐ HỌC VÀ LOGIC .....	37
3.3.1. Các phép toán cộng trừ .....	39
3.3.1.1. Phép toán cộng .....	39
3.3.1.2. Phép toán trừ .....	40
3.3.2. Phép nhân số nguyên không dấu .....	41
3.3.3. Phép nhân số nguyên có dấu .....	42
3.3.4. Phép chia số nguyên không dấu .....	49
3.3.5. Phép chia số nguyên có dấu .....	52
3.3.6. Phép toán với số dấu phẩy động .....	54
3.3.6.1. Phép cộng và trừ .....	54
3.3.6.2. Phép nhân và chia .....	59
3.3.6.3. Phép làm tròn .....	61
3.4. ĐƠN VỊ ĐIỀU KHIỂN .....	63
3.4.1. Tín hiệu điều khiển .....	63
3.4.1.1. Các tín hiệu vào .....	64
3.4.1.2. Các tín hiệu ra .....	64
3.4.2. Đơn vị điều khiển vi chương trình .....	65
<b>Chương 4: KIẾN TRÚC TẬP LỆNH .....</b>	<b>67</b>
4.1. CÁC ĐẶC TRƯNG CƠ BẢN CỦA LỆNH MÁY .....	67
4.1.1. Giới thiệu chung về tập lệnh .....	67



4.1.2. Các thành phần của lệnh máy.....	68
4.1.3. Mô tả lệnh.....	68
4.1.4. Các kiểu lệnh.....	68
4.1.5. Các thao tác khi thực hiện lệnh.....	69
4.1.6. Các vấn đề về thiết kế tập lệnh.....	69
4.2. CÁC KIỂU TOÁN HẠNG.....	69
4.2.1. Số lượng địa chỉ toán hạng.....	70
4.2.2. Đánh giá về số địa chỉ toán hạng.....	74
4.3. TẬP LỆNH.....	74
4.3.1. Các lệnh chuyển dữ liệu.....	74
4.3.2. Các lệnh số học.....	75
4.3.3. Các lệnh logic.....	75
4.3.4. Các lệnh vào ra chuyên dụng.....	76
4.3.5. Các lệnh chuyển điều kiện.....	76
4.3.6. Lệnh rẽ nhánh.....	76
4.3.7. Lệnh CALL và RETURN.....	78
4.3.8. Các lệnh điều khiển hệ thống.....	79
4.4. NGÔN NGỮ LẬP TRÌNH VÀ CHƯƠNG TRÌNH DỊCH.....	79
4.4.1. Khái niệm ngôn ngữ lập trình.....	79
4.4.2. Các loại ngôn ngữ lập trình thông dụng.....	79
4.4.3. Chương trình dịch.....	81
<b>Chương 5: HỆ THỐNG NHỚ.....</b>	<b>83</b>
5.1. TỔNG QUAN VỀ HỆ THỐNG NHỚ.....	83
5.1.1. Phân loại hệ thống nhớ.....	83
5.1.2. Phân cấp hệ thống nhớ.....	85
5.2. BỘ NHỚ BÁN DẪN.....	85
5.2.1. Phân loại bộ nhớ bán dẫn.....	85
5.2.1.1. ROM (Read Only Memory).....	86
5.2.1.2. RAM (Random Access Memory).....	88
5.2.2. Tổ chức của chip nhớ.....	90
5.2.3. Thiết kế mô-đun nhớ bán dẫn.....	91
5.3. BỘ NHỚ CACHE, BỘ NHỚ TRUY CẬP NHANH.....	96
5.3.1. Nguyên tắc chung của cache.....	96
5.3.2. Các phương pháp ánh xạ.....	98
5.3.2.1. Ánh xạ trực tiếp (Direct mapping).....	98
5.3.2.2. Ánh xạ liên kết toàn phần (Fully associative mapping).....	102
5.3.2.3. Ánh xạ liên kết tập hợp.....	105
5.3.3. Thuật giải thay thế.....	107
5.3.4. Phương pháp ghi dữ liệu cache hit.....	108
5.3.5. Cache trên các bộ xử lý Intel.....	108
5.4. BỘ NHỚ NGOÀI.....	109
5.4.1. Đĩa từ.....	109

5.4.2. Đĩa quang .....	111
5.4.3. Các loại thẻ nhớ .....	113
5.4.4. Băng từ .....	113
5.4.5. Biện pháp an toàn dữ liệu khi lưu trữ thông tin trong đĩa từ .....	114
<b>Chương 6: HỆ THỐNG VÀO RA .....</b>	<b>116</b>
6.1. CẤU TRÚC CHUNG CỦA HỆ THỐNG VÀO RA .....	116
6.1.1. Cấu trúc cơ bản của hệ thống vào-ra .....	116
6.1.2. Các thiết bị ngoại vi .....	117
6.1.3. Mô-đun vào-ra .....	118
6.1.4. Địa chỉ hóa cổng vào ra .....	119
6.1.4.1. Không gian địa chỉ của bộ xử lý .....	119
6.1.4.2. Các phương pháp địa chỉ hóa cổng vào-ra .....	120
6.2. CÁC PHƯƠNG PHÁP TRAO ĐỔI DỮ LIỆU .....	120
6.2.1. Vào-ra bằng chương trình .....	120
6.2.2. Vào-ra điều khiển bằng ngắt .....	121
6.2.3. Truy nhập bộ nhớ trực tiếp – DMA .....	125
6.2.4. Kênh vào-ra hay bộ xử lý vào-ra .....	126
6.3. GHÉP NỐI VỚI THIẾT BỊ NGOẠI VI .....	126
6.3.1. Các kiểu nối ghép vào ra .....	126
6.3.1.1. Nối ghép song song .....	126
6.3.1.2. Nối ghép nối tiếp .....	127
6.3.2. Các cấu hình ghép nối .....	127
<b>Chương 7: GIAO DIỆN TRUYỀN DỮ LIỆU .....</b>	<b>129</b>
7.1. GIAO DIỆN SONG SONG .....	129
7.1.1. Mạch thu/phát đệm dữ liệu SN74LS245 .....	130
7.1.2. Mạch tương thích với ngoại vi chương trình hoá 8255A .....	131
7.2. Giao diện tuần tự .....	136
7.3. Giao diện đa năng USB .....	141
7.4. Giao diện cao tốc IEEE 1394 .....	145
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>149</b>

## LỜI NÓI ĐẦU

**Protected by PDF Anti-Copy Free**

**(Upgrade to Pro Version to Remove the Watermark)**

Giáo trình Kiến trúc máy tính được biên soạn làm tài liệu giảng dạy của các giảng viên và là tài liệu tham khảo cho sinh viên chuyên ngành Điện tử viễn thông của Trường Đại học Công nghệ GTVT theo yêu cầu, mục tiêu đào tạo của Nhà trường.

Giáo trình Kiến trúc máy tính được biên soạn cấp cho sinh viên các kiến thức cơ sở của kiến trúc máy tính; hệ thống phân cấp bộ nhớ, bộ nhớ trong, bộ nhớ cache và các loại bộ nhớ ngoài và các thiết bị ngoại vi. Nội dung của giáo trình được biên soạn thành bảy chương:

**Chương 1:** Trình bày lịch sử phát triển của máy tính, cũng như các tính năng mới của máy tính trong từng giai đoạn; các thành phần cơ bản của máy tính.

**Chương 2:** Trình bày các phương pháp biến đổi cơ bản của hệ thống số; cách biểu diễn thông tin trong máy tính.

**Chương 3:** Giới thiệu cấu trúc của bộ xử lý trung tâm: tổ chức, chức năng và nguyên lý hoạt động của các bộ phận bên trong bộ xử lý như bộ tính toán logic số học, bộ điều khiển, tập các thanh ghi. Giới thiệu ngôn ngữ lập trình và chương trình dịch.

**Chương 4:** Giới thiệu bộ điều khiển tạo ra sự vận chuyển tín hiệu bên trong bộ xử lý nhằm thực hiện tập lệnh tương ứng với kiến trúc phần mềm đã đề ra. Mô tả diễn tiến thi hành một lệnh mã máy, đây là cơ sở để hiểu được các hoạt động xử lý lệnh trong máy tính.

**Chương 5:** Chương này giới thiệu chức năng và nguyên lý hoạt động của các cấp bộ nhớ máy tính: bộ nhớ cache: nguyên lý vận hành, phân loại các mức, đánh giá hiệu quả hoạt động; và nguyên lý vận hành của bộ nhớ ảo.

**Chương 6:** Hệ thống chuyển dữ liệu giữa máy tính và thiết bị ngoại vi, các phương pháp trao đổi dữ liệu và các cách ghép nối với thiết bị ngoại vi.

**Chương 7:** Giới thiệu chung về cấu trúc của hệ thống truyền dữ liệu, các phương pháp trao đổi dữ liệu song song, nối tiếp, giữa máy tính và thiết bị ngoại vi.

Giáo trình được biên soạn dựa trên kinh nghiệm giảng dạy môn học Kiến trúc máy tính tại Đại học Công nghệ GTVT của nhóm biên soạn. Tài liệu có thể được sử dụng làm tài liệu học tập, tham khảo cho sinh viên hệ đại học và cao đẳng ngành điện tử viễn thông và công nghệ thông tin.

Phân công biên soạn giáo trình như sau: TS. Đỗ Quang Hưng biên soạn lý thuyết và bài tập Chương 1 và Chương 2; KS. Phạm Trường Giang biên soạn lý thuyết và bài tập Chương 4, Chương 5 và Chương 6; ThS. Bùi Hải Đăng biên soạn lý thuyết và bài tập Chương 3 và Chương 7. Giáo trình được biên soạn lần đầu nên khó tránh khỏi những thiếu sót. Nhóm biên soạn rất mong nhận được các góp ý, cũng như ý kiến về việc cập nhật, hoàn thiện nội dung của giáo trình.

**Nhóm biên soạn**

**Hà Nội, tháng 2 năm 2017**

## CÁC TỪ VIẾT TẮT

**Protected by PDF Anti-Copy Free**

(Upgrade to Pro Version to Remove the Watermark)

ALU	Arithmetic and Logic Unit
BCD	Binary Coded Decimal
CD	Compact Disk
CD-ROM	Compact Disk Read Only Memory
CD-RW	CD ReWritable
CPU	Central Processing Unit
DMA	Direct memory Access
DRAM	Dynamic RAM
DVD	Digital Video Disk
FR	Flag register
IEEE	Institute of Electrical and Electronic Engineers
LSB	Least Significant Bit
MSB	Most Significant Bit
PIC	Programmable Interrupt Controller
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
ROM	Read Only Memory
USB	Universal Serial Bus

## Chương 1

# GIỚI THIỆU CHUNG VỀ KIẾN TRÚC MÁY TÍNH

Trang bị cho sinh viên những kiến thức cơ bản về kiến trúc máy tính: Các khái niệm, các thành phần cơ bản, phạm vi của máy tính và lịch sử phát triển của máy tính.



## 1.1. CÁC KHÁI NIỆM VÀ NGUYÊN LÝ CƠ BẢN

### 1.1.1. Mở đầu

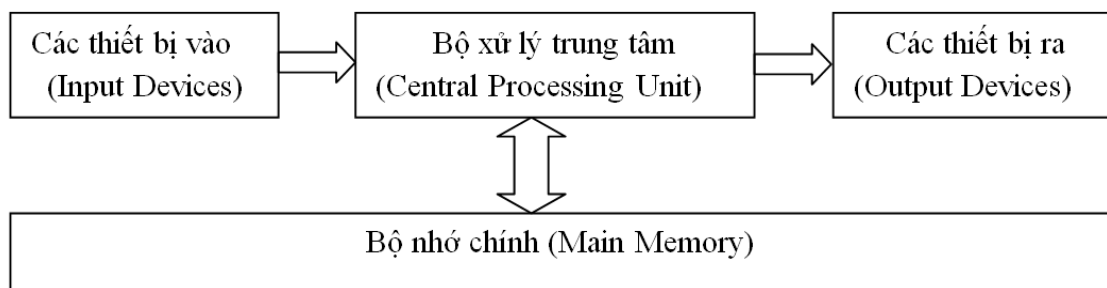
Kiến trúc máy tính là khoa học về việc lựa chọn và kết nối các thành phần phần cứng để tạo ra các máy tính đạt được các yêu cầu về chức năng (functionality), hiệu năng (performance) và giá thành (cost). Yêu cầu chức năng đòi hỏi máy tính phải có thêm nhiều tính năng phong phú và hữu ích; yêu cầu hiệu năng đòi hỏi máy tính phải đạt tốc độ xử lý cao hơn và yêu cầu giá thành đòi hỏi máy tính phải càng ngày càng rẻ hơn. Để đạt được cả ba yêu cầu về chức năng, hiệu năng và giá thành là rất khó khăn. Tuy nhiên, nhờ có sự phát triển rất mạnh mẽ của công nghệ vi xử lý, các máy tính ngày nay có tính năng phong phú, nhanh hơn và rẻ hơn so với máy tính các thế hệ trước.

### 1.1.2. Khái niệm máy tính

Máy tính (computer) là một thiết bị điện tử hoạt động dưới sự điều khiển của các chỉ thị được lưu trữ trong bộ nhớ. Nó thực hiện các công việc sau:

- Nhận thông tin đầu vào
- Xử lý thông tin theo chương trình được lưu trong bộ nhớ
- Đưa thông tin ra

Máy tính hoạt động theo chương trình.



Hình 1.1. Mô hình máy tính cơ bản

### 1.1.3. Kiến trúc máy tính và cấu trúc máy tính

- Kiến trúc máy tính (architecture) nghiên cứu những thuộc tính của một hệ thống mà người lập trình có thể nhìn thấy được, những thuộc tính quyết định trực tiếp đến việc thực thi một chương trình tính toán, xử lý dữ liệu

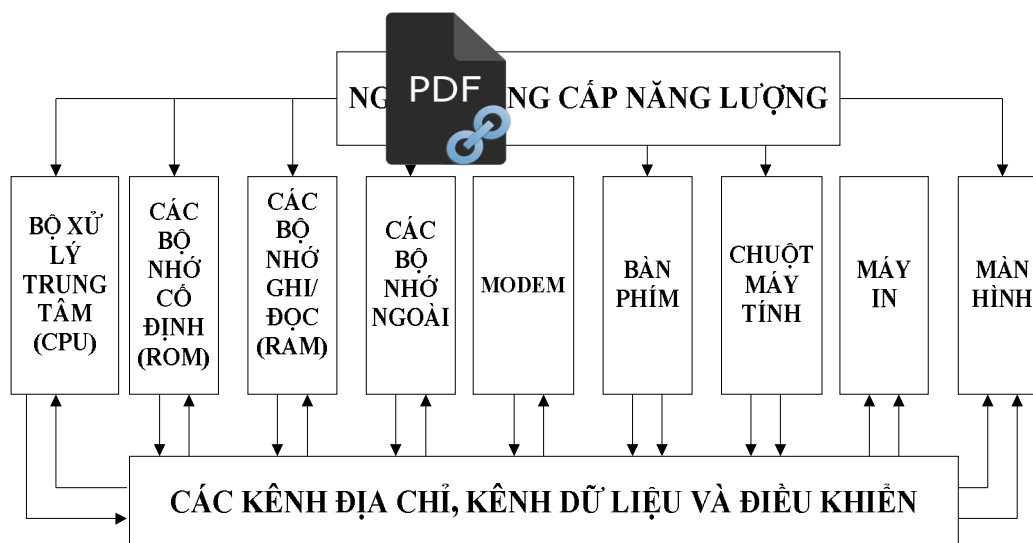
- Cấu trúc máy tính (structure) nghiên cứu về các thành phần chức năng và sự kết nối giữa chúng để tạo nên một hệ thống, nhằm thực hiện chức năng và tính toán kỹ thuật của kiến trúc.

Những thuộc tính liên quan đến kiến trúc bao gồm tập lệnh cơ bản mà CPU có thể thực hiện, số bit được sử dụng để biểu diễn các loại dữ liệu khác nhau, cơ chế nhập/xuất dữ liệu và các kỹ thuật đánh địa chỉ ô nhớ,...Cấu trúc máy tính lại bao gồm các thuộc tính kỹ thuật mà người lập trình không nhận biết được như các tín hiệu điều khiển, giao diện giữa máy tính và thiết bị ngoại vi, công nghệ xây dựng bộ nhớ...

Ví dụ việc quyết định máy tính có cần một lệnh cơ bản để thực hiện phép nhân hay không là vấn đề về kiến trúc. Còn thể hiện lệnh nhân bằng các đơn vị vật lý cụ thể nào (chẳng hạn, một đơn vị thuộc phần cứng đặc biệt, hay thực hiện lặp nhiều phép cộng) lại là vấn đề về cấu trúc. Để làm ví dụ minh họa sự khác biệt đó ta có thể xem các máy tính ở Trung tâm nghiên cứu nào đó. Các máy tính này có thể có kiến trúc rất giống nhau theo quan điểm của người lập trình. Chúng có cùng số thanh ghi (tức là thiết bị lưu trữ tạm thời), có cùng một tập lệnh cơ bản và dạng các toán hạng được nạp vào bộ nhớ giống nhau. Tuy nhiên các hệ thống này khác nhau về mặt cấu trúc: số bộ vi xử lý khác nhau, kích thước bộ nhớ của chúng cũng khác hẳn nhau, cách thức dữ liệu được truyền từ bộ nhớ đến bộ vi xử lý cũng không giống nhau. Kiến trúc máy tính thường được ứng dụng trong khoảng thời gian dài, hàng chục năm; trong khi cấu trúc thường thay đổi cùng với sự phát triển của công nghệ. Trên cùng một kiến trúc, các hãng chế tạo máy tính có thể đưa ra nhiều loại máy tính khác nhau về cấu trúc, do đó các đặc trưng về hiệu suất, giá thành cũng khác nhau. Các sản phẩm của IBM là một ví dụ điển hình. Kiến trúc máy tính của IBM vẫn còn được ứng dụng cho tới ngày nay và là ngọn cờ của thương hiệu IBM. Trong lĩnh vực máy PC, người ta thường không phân biệt rõ ràng giữa kiến trúc và cấu trúc vì sự khác biệt giữa hai khái niệm này đã rút ngắn đáng kể. Sự phát triển của công nghệ không chỉ tác động lên cấu trúc mà còn tạo điều kiện phát triển các kiến trúc mạnh hơn và nhiều tính năng hơn; và do đó tác động qua lại giữa kiến trúc và cấu trúc thường xuyên hơn [3,4].

## 1.2. CÁC THÀNH PHẦN CƠ BẢN CỦA MÁY TÍNH

Để đảm bảo tính tương thích, cấu trúc phần cứng bên trong các máy vi tính cá nhân về cơ bản là giống nhau. Vì thế chúng ta có cấu trúc chung của máy vi tính như sau:



Hình 1.2. Cấu trúc chung của máy vi tính

### 1.2.1. Bộ nguồn

#### 1.2.1.1. Nguồn cấp điện cho máy lớn

Bộ nguồn có chức năng chuyển điện xoay chiều AC 110 – 220V thành điện một chiều DC để cung cấp cho các mạch điện tử bên trong máy, cũng như các bộ phận ngoại vi.

Nguồn điện là điều kiện cơ bản cho các máy tính hoạt động nên một bộ nguồn hoạt động ổn định và cung cấp đủ công suất rất quan trọng đối với một máy vi tính cá nhân. Tùy theo chủng loại cấu hình, mỗi máy vi tính cá nhân cần một công suất khác nhau.

Ví dụ: Laptop công suất 60 – 80W; để bàn: 200W- 400W.

Để máy tính làm việc tốt thì bộ nguồn cần phải ổn định, làm nguội tốt, hiệu suất cao và phải có khả năng mở rộng. Bộ nguồn cung cấp cho bản mạch chính và các thiết bị ngoại vi những điện thế:  $\pm 5V$ ;  $\pm 12V$ ;  $\pm 3,3V$ ;  $0V$ . Ngày nay, có nhiều bộ nguồn với công nghệ mới nhất và chất lượng tốt nhất điển hình như: Corsair, Seasonic, Superflower....

Bộ nguồn được chia theo nguyên tắc hoạt động thành 2 loại:

*Bộ nguồn tuyến tính:* Gồm một biến thể để hạ điện áp, một mạch dẫn dòng (Dùng 4 Diode Bridge) và một bộ ổn áp để ổn định hiệu điện thế ( Có thể đổi 12V – 5V ). Do bộ nguồn tuyến tính giải phóng rất nhiều nhiệt lượng và hao tổn điện năng nên ngày nay gần như không tồn tại trong vi tính cá nhân.

*Bộ ổn áp ngắt:* Là bộ nguồn có hiệu suất cao. Năng lượng điện được điều tiết theo nguyên tắc đóng – mở.

#### **1.2.1.2. Nguồn pin cho máy tính xách tay**

Thế hệ đầu tiên bộ nguồn dành cho máy tính xách tay là pin NiCad (Nickel Cadmium). Thế hệ sau của nó là pin NiMH (Nickel Metal Hybride).

Năm 1998, trên thị trường xuất hiện thêm pin Li – Ion (Lithium Ion). Pin Li – Ion có thời gian làm việc lâu hơn pin NiMH (Khoảng 3 tiếng làm việc liên tục), nhẹ hơn và không cần xả hết trước khi nạp. Nhược điểm duy nhất của loại pin này là sẽ tự xả hết điện nếu như không được sử dụng trong thời gian dài.

Chính vì lý do trên nên tới năm 1999, đã xuất hiện loại pin mới Li – polymer. Pin này có mật độ điện tích cao hơn nhiều Li- Ion. Thay vì dùng điện môi lỏng, pin dùng điện môi dạng gel hay rắn ghép giữa các điện cực. Pin có cấu trúc lớp được sử dụng rộng rãi từ năm 2000 cho tới ngày nay [6].

#### **1.2.2. Bản mạch chính**

Bản mạch chính (Main Board) chứa đựng những linh kiện điện tử và những chi tiết quan trọng nhất của máy vi tính cá nhân như: Bộ vi xử lý CPU (Central Processing Unit), hệ thống bus và các vi mạch hỗ trợ. Vì vậy, bản mạch chính cần phải: nhỏ gọn, ổn định với nhiều bên ngoài và an toàn về điện.

##### **1.2.2.1. Bộ xử lý trung tâm (CPU- Central Processing Unit)**

Trung tâm đầu não của máy vi tính là bộ xử lý trung tâm, nó có nhiệm vụ quản lý điều hành và phân phối các tài nguyên của hệ thống tới các thiết bị làm việc khác trong hệ thống máy vi tính. Trong suốt quá trình làm việc của máy vi tính, nó thông qua các kênh điều khiển, kênh địa chỉ và kênh dữ liệu, tiến hành tất cả các phép gia công và xử lý thông tin, các tín hiệu điều khiển đều được thực hiện và điều phối trong bộ xử lý trung tâm.

Trong quá trình làm việc của bộ xử lý trung tâm, nó căn cứ vào lệnh nhận được, sau đó sẽ phát ra tín hiệu điều khiển đưa đến các thiết bị khác để yêu cầu chúng



hoạt động, chỉ cho chúng biết phải lấy tài nguyên ở bộ phận nào, địa chỉ nào và chỉ ra hướng truyền thông tin cho chúng. Khả năng, tốc độ của bộ xử lý trung tâm quyết định khả năng và tốc độ của máy tính.

#### **1.2.2.2. Bộ nhớ cố định (ROM- Read Only Memory)**

ROM chứa thông tin cố định, được phép đọc ra từ nó, đó là thông tin chương trình khởi động máy, chương trình quản lý máy, chương trình biên dịch ngôn ngữ. Nội dung của ROM sẽ không thay đổi khi bị mất điện. Thông tin ghi nhớ ở trong bộ nhớ ROM thường được nạp sẵn từ khi sản xuất. Ngoài ra để thuận tiện cho người sử dụng, thiết kế, xây dựng hệ thống chuyên dùng người ta còn sử dụng các loại EPROM, PROM (các loại bộ nhớ kiểu này có thể ghi lại bằng thiết bị đặc biệt).

#### **1.2.2.3. Bộ nhớ ghi/đọc (RAM- Random Access Memory)**

Bộ nhớ ghi/đọc (RAM) có thể truy xuất dữ liệu một cách ngẫu nhiên, chúng ta có thể ghi vào RAM, hoặc đọc dữ liệu ra từ RAM. Bất kỳ thời điểm nào bộ xử lý trung tâm có yêu cầu trao đổi thông tin thì bộ nhớ ghi/đọc (RAM) đều phải đáp ứng. Chúng thường được sử dụng để ghi nhớ tạm thời dữ liệu trong suốt quá trình hoạt động gia công, xử lý thông tin của máy tính.

Các bộ nhớ ghi/đọc thường có hai loại đó là RAM động (DRAM) và RAM tĩnh (SRAM).

RAM tĩnh được cấu tạo từ các vi mạch nhớ, xây dựng trên các TRIGƠ (flip-flop). Nếu chúng ta liên tục duy trì nguồn cung cấp thì nội dung trong RAM tĩnh được bảo toàn, thao tác trao đổi thông tin với RAM tĩnh đơn giản hơn, tốc độ truy cập nhanh hơn, nhưng dung lượng so với RAM động nhỏ hơn khi chúng cùng thể tích.

Nguyên tắc lưu trữ số liệu ở bộ nhớ RAM động giống như nguyên tắc lưu trữ năng lượng (điện áp) của tụ điện. Mỗi bit nhớ trong RAM động tương ứng với một tụ điện. Như vậy theo thời gian thì năng lượng lưu trữ ở trong RAM động sẽ bị suy giảm, dẫn đến hiện tượng mất dữ liệu trong quá trình làm việc, để đảm bảo duy trì dữ liệu trong bộ nhớ hệ thống máy tính phải liên tục thực hiện thao tác làm tươi bộ nhớ.

Quá trình làm tươi bộ nhớ RAM động thường thực hiện theo chu kỳ từ 2 đến 3 giây, tức là cứ 2 hoặc 3 giây tại các vị trí dữ liệu của bộ nhớ DRAM có mức logic 1 máy tính phải tiến hành nạp số liệu lại 1 lần.

#### **1.2.2.4. Các bộ nhớ ngoài**

- Ngoài các hệ thống lưu trữ dữ liệu bên trong như ROM, RAM, các hệ máy vi tính còn sử dụng các hệ thống ngoài để lưu trữ thông tin với mục đích tăng dung lượng nhớ cho hệ thống, lưu trữ thông tin để bảo quản lâu dài, đồng thời để sử dụng các thông tin đó chuyển sang các hệ máy vi tính khác, cũng như để cập nhật các chương trình điều hành, các chương trình ứng dụng,.... Các bộ nhớ ngoài thường được sử dụng hiện nay đó là các hệ thống đĩa cứng, đĩa mềm, đĩa quang, USB,...

- Về nguyên tắc thì hệ máy vi tính quản lý các hệ thống nhớ ngoài thông qua các cổng vào/ra giống như các thiết bị ngoại vi, chúng cũng được địa chỉ hoá và quản lý chặt chẽ. Trong quá trình làm việc với chúng hệ máy vi tính sẽ tiến hành kiểm tra để báo cho người sử dụng biết chúng có được phối ghép với hệ thống làm việc hay không.

### **1.2.3. Các thiết bị ngoại vi**

#### **1.2.3.1. Bàn phím (Keyboard)**

Là thiết bị đưa thông tin vào đơn giản và thông dụng nhất của máy vi tính, chúng là tổ hợp của các công tắc tức thời. Tín hiệu lối ra của các công tắc đưa tới bộ giải mã bàn phím, tại đó chúng được tạo ra các bộ mã tương ứng với chức năng của từng phím, chúng chính là dãy các con số mã nhị phân của các ký tự, ký hiệu, các chữ số thập phân, v...v... Các bộ mã này sẽ được đưa vào hệ máy vi tính để yêu cầu nó thực hiện các chức năng theo ý muốn của người sử dụng.

#### **1.2.3.2. Màn hình (Monitor)**

Màn hình chính là thiết bị đưa thông tin ra. Màn hình dùng để hiển thị các thông tin mang nội dung của các số liệu, cũng như các văn bản đưa vào, hay là các thông báo của máy vi tính đối với người sử dụng, trên cơ sở đó người sử dụng đánh giá nó có hoạt động đúng hay không và thực hiện các thao tác tiếp tục theo yêu cầu để máy tính tiếp tục hoạt động, hoặc thực hiện dừng hệ thống khi công việc thực hiện xong.

Ngày nay thường sử dụng hai loại màn hình cơ bản.

- + Hiển thị bằng ống tia điện tử CRT,
- + Tinh thể lỏng (hoặc ma trận điôt màu).

Sử dụng màn hình tinh thể lỏng thì hệ thống gọn, nhẹ, tiết kiệm năng lượng và an toàn hơn đối với người sử dụng.

#### 1.2.3.3. Máy in (Printer)

Được sử dụng để in các tài liệu để in các biểu đồ, các số liệu trên giấy để sử dụng. Máy in có thể in các tài liệu bằng các máy in đen trắng, máy in màu tùy theo nhu cầu và chất lượng hình ảnh cần in.

Có nhiều loại máy in khác nhau nhưng người ta thường phân thành hai loại:

- Máy in tiếp xúc: Là loại máy in sử dụng các ký tự, hình ảnh bằng cách tiếp xúc cơ học của các kim in lên các băng mực đặt trên mặt giấy in. Loại máy này tốc độ chậm, thường chỉ in được đen trắng, độ phân giải thấp, gây ra tiếng ồn trong quá trình làm việc. Ngày nay loại máy in này ít được sử dụng.

- Máy in không tiếp xúc: Loại này thực hiện in ra các ký tự không cần va đập, do đó chúng không gây tiếng ồn khi làm việc, chất lượng in tốt, độ phân giải cao, tốc độ làm việc cao và có thể in các hình ảnh màu. Thông dụng nhất của loại in không tiếp xúc là máy in laser, máy in phun.

#### 1.2.3.4. Modem và các thiết bị ngoại vi khác

Ngoài các thiết bị đã nêu trên người ta còn sử dụng thêm các thiết bị khác. Ví dụ như: chuột máy tính, modem phối ghép.

Chuột máy tính thường được kết nối thông qua cổng COM, USB giúp cho việc điều hành máy vi tính được thuận tiện, nhanh chóng hơn. Chuột máy tính có hai loại: Chuột không dây và chuột có dây, đối với chuột có dây lại có chuột cơ và chuột quang. Còn chuột không dây thì thường là chuột quang. Chuột máy tính là thiết bị chuột có thể nối trực tiếp hoặc nối không dây thông qua cổng thông tin hồng ngoại.

Modem phối ghép: Đó là các mạch điện tử dùng để phối ghép các thiết bị ngoại vi khác nhau như: các thiết bị xử lý âm thanh, xử lý hình ảnh, các thiết bị ghép nối mạng máy tính với mạng internet, mạng thông tin,...

### 1.3. PHẦN MỀM MÁY TÍNH

Phần mềm máy tính (Chương trình phần mềm): Là tổ hợp các lệnh trong kiến trúc tập lệnh. Đó là toàn bộ các chương trình điều hành máy tính và các chương trình ứng dụng thực hiện các thuật toán trong quá trình gia công và xử lý thông tin.

Các chương trình điều hành và các chương trình ứng dụng có thể được nhập từ bàn phím, đĩa cứng, đĩa mềm hoặc các thiết bị khác. Chương trình được lưu trữ ở dạng

tập hợp các bit nhị phân 0 và 1, chương trình được ghi nhớ dưới dạng mã máy. Phần mềm như là linh hồn của máy tính.

**Protected by PDF Anti-Copy Free**

Phần mềm (hay còn gọi là chương trình) là một tập hợp các lệnh máy tính được viết bằng ngôn ngữ lập trình để thực hiện một công việc nhất định.

Phần mềm hệ thống (thường được gọi là hệ điều hành): Bao gồm các hệ thống chương trình phần mềm, thực hiện các chức năng điều hành hệ thống, nó giúp cho máy tính quản lý, vận hành hệ thống, khởi động và sửa chữa các chương trình.

Chương trình ứng dụng: Được xây dựng đáp ứng cho người sử dụng khai thác và vận hành máy tính theo những mục đích khác nhau.

Ngoài việc tạo ra những phần mềm có ích con người còn tạo ra các phần mềm có hại cho máy tính. Những phần mềm như vậy chúng ta gọi là Virus.

#### 1.4. LỊCH SỬ PHÁT TRIỂN MÁY TÍNH

Nguồn gốc sơ khai của việc tính toán được sử dụng bằng công cụ là bàn tính gậy tay, nó được phát minh khoảng 3000 năm trước Công nguyên tại thành phố Babilon, sau này hàng loạt các công cụ tính toán khác nhau ra đời. Nhưng phải đến thế kỷ 17 thì máy tính cơ học đầu tiên có thể thực hiện được các phép tính cộng, trừ mới được chế tạo thành công do phát minh của nhà bác học Pascal người Pháp [1,4].

Năm 1930 tại Mỹ máy tính điện tử số được bác học Bush đề cập xây dựng, đến năm 1944 ở Mỹ hãng IBM do giáo sư Aiken chỉ đạo bắt đầu sản xuất máy tính hoàn thiện.

Kể từ đó máy tính phát triển mạnh mẽ và được chia thành các thế hệ như sau:

+ Máy tính thế hệ 1 (1950 – 1959): Là thế hệ dùng bóng điện tử chân không. Năm 1946 máy tính điện tử số ENIAC được xây dựng có chứa khoảng 18.000 bóng đèn điện tử, 1500 role, cùng với hàng triệu các linh kiện thụ động khác. Máy tính này nặng khoảng 30 tấn, chiếm diện tích 200m<sup>2</sup>, trên một toà nhà với không gian rộng lớn với các hệ thống làm mát tiêu thụ khá nhiều năng lượng, hiệu suất và khả năng làm việc rất thấp.

+ Máy tính thế hệ 2 (1959 – 1963): Thế hệ chế tạo các linh kiện bán dẫn rời ( Diode, Transistor ). Máy tính đầu tiên của thế hệ này là TX – 0 ( Transistorized Experimental Computer 0 ).

+ Máy tính thế hệ 3 (1964-1974): Máy tính được xây dựng trên các vi mạch cỡ nhỏ (SSI) và cỡ vừa (MSI). điển hình là thế hệ máy System 360 của IBM. Thế hệ máy tính này có những bước đột phá mới đó là: Tính tương thích cao, đặc tính đa chương trình, không gian địa chỉ lớn.

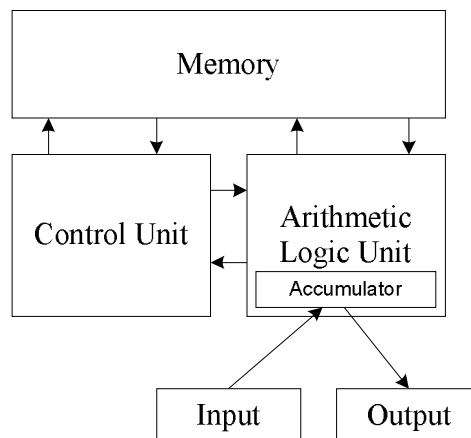
+ Máy tính thế hệ 4 (1974) Máy tính được xây dựng trên các vi mạch cỡ lớn ( LSI ) và cực lớn ( VLSI ).

+ Máy tính thế hệ 5 (2000 đến nay): Ứng dụng máy tính thông minh của trí tuệ nhân tạo với công nghệ Noron.

Ngày nay với sự phát triển mạnh mẽ của công nghệ vi điện tử và công nghệ bán dẫn thì cấu trúc của máy tính ngày càng được thu gọn, tốc độ làm việc và khả năng thao tác để giải quyết các bài toán lớn càng được tăng lên. Kể từ nửa sau của thế kỷ 20 máy tính điện tử đã phát triển qua nhiều thời kỳ và nhiều giai đoạn khác nhau với tốc độ cực kỳ nhanh chóng, cấu hình ngày càng hoàn thiện và được nâng cấp liên tục, kết cấu ngày càng được thu gọn và đặc biệt giá thành ngày càng giảm mạnh. Đây chính là thế mạnh của máy vi tính làm cho nó được ứng dụng rộng rãi trong nhiều lĩnh vực đời sống, kinh tế, xã hội,...

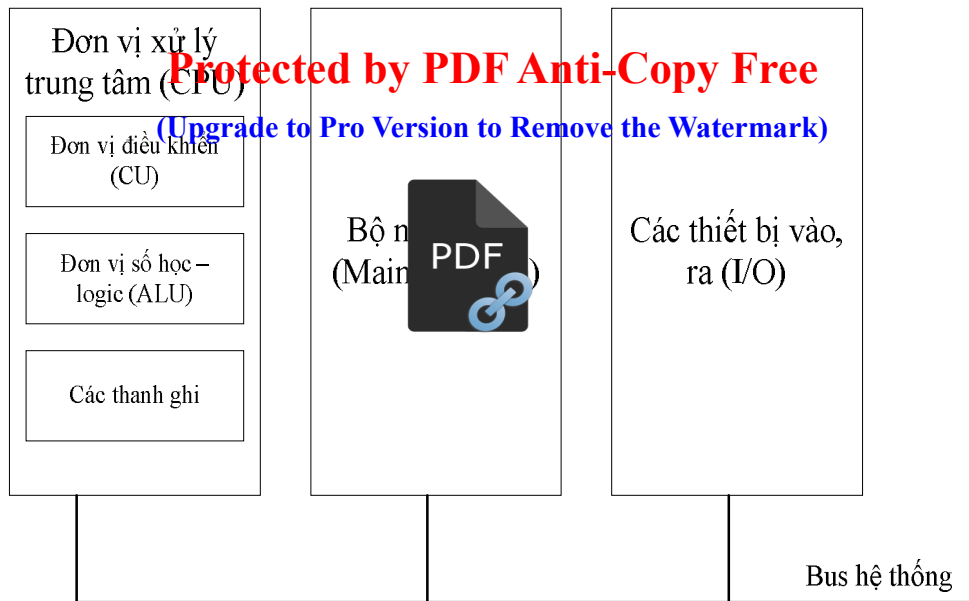
### 1.5. KIẾN TRÚC MÁY TÍNH VON-NEUMANN

Kiến trúc máy tính von-Neumann được John von-Neumann đưa ra vào năm 1945. Kiến trúc máy tính von-Neumann nguyên thủy như Hình 1.3.



**Hình 1.3. Kiến trúc máy tính von-Neumann nguyên thủy**

Các máy tính hiện đại ngày nay sử dụng kiến trúc máy tính von-Neumann cải tiến – còn gọi là kiến trúc máy tính von-Neumann hiện đại (Hình 1.4).



**Hình 1.4. Kiến trúc máy tính von-Neumann hiện đại**

Kiến trúc von-Neumann dựa trên các nguyên lý sau:

- + Lệnh và dữ liệu được lưu trữ trong bộ nhớ đọc ghi chia sẻ - một 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 và.

- + Các lệnh của một chương trình được thực hiện tuần tự.

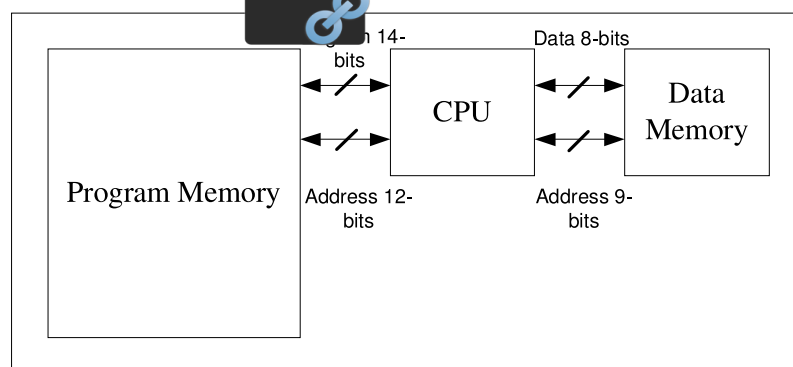
Quá trình thực hiện lệnh được chia thành 3 giai đoạn (stages) chính:

- + 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).

## 1.6. KIẾN TRÚC MÁY TÍNH HAVARD

Kiến trúc máy tính Harvard là một kiến trúc như Hình 1.5. Kiến trúc máy tính Harvard có bộ nhớ trong gồm hai thành phần: Bộ nhớ lưu chương trình (Program Memory) và Bộ nhớ lưu dữ liệu (Data Memory). Hai hệ thống bus riêng được sử dụng để kết nối CPU với bộ nhớ lưu chương trình và bộ nhớ lưu dữ liệu. Mỗi hệ thống bus đều có đầy đủ ba thành phần để truyền dẫn các tín hiệu địa chỉ, dữ liệu và điều khiển. Máy tính dựa trên kiến trúc Harvard có khả năng đạt được tốc độ xử lý cao hơn máy

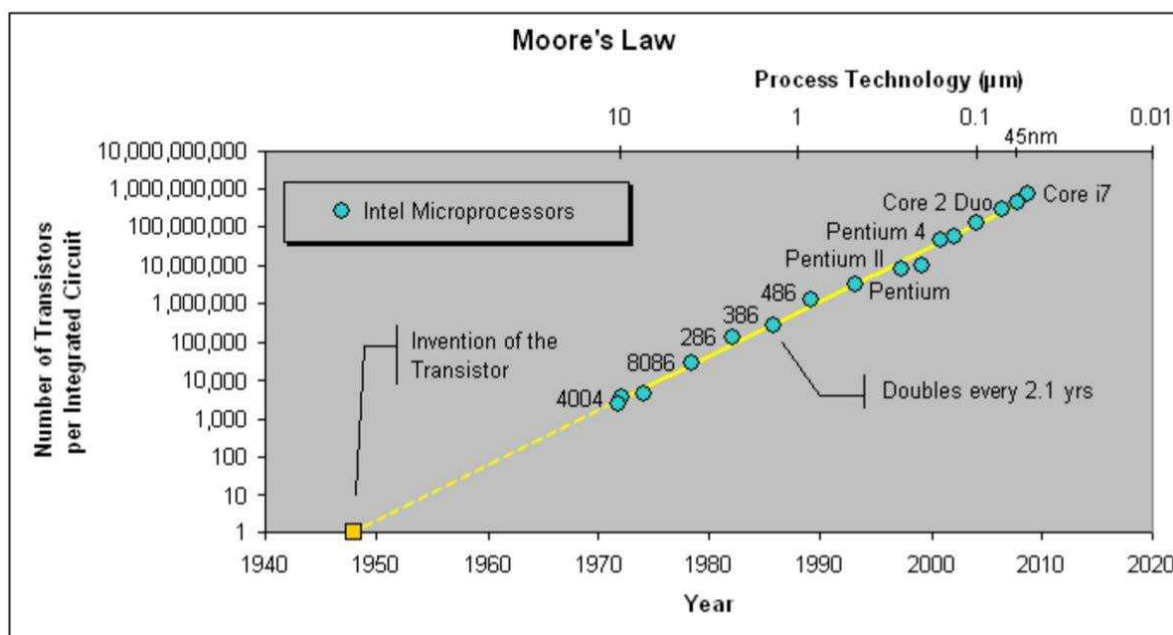
tính dựa trên kiến trúc von-Neumann do kiến trúc Harvard hỗ trợ hai hệ thống bus độc lập với băng thông lớn hơn. Ngoài ra nhờ có hai hệ thống bus độc lập, hệ thống nhớ trong kiến trúc Harvard hỗ trợ nhiều lệnh truy nhập bộ nhớ tại một thời điểm, giúp giảm xung đột truy nhập bộ nhớ đặc biệt khi CPU sử dụng kỹ thuật đường ống (pipeline).



Hình 1.5. Kiến trúc máy tính Havard

## 1.7. ĐỊNH LUẬT MOORE

Định luật Moore được xây dựng bởi Gordon Moore - một trong những sáng lập viên của tập đoàn sản xuất chip máy tính nổi tiếng Intel. Từ năm 1965, G. Moore đã đưa ra dự đoán: Khả năng của máy tính sẽ tăng lên gấp đôi sau 18 tháng với giá thành là như nhau.



Hình 1.6. Sự phát triển của bộ xử lý Intel theo qui luật Moore

Một số kết luận rút ra từ quy luật Moore:

- + Chi phí cho bộ vi xử lý sẽ giảm
- + Giảm kích thước các linh kiện, máy tính sẽ giảm kích thước
- + Hệ thống kết nối bên trong sẽ ngắn: tăng độ tin cậy, tăng tốc độ .
- + Tiết kiệm năng lượng cung cấp nhiệt thấp.
- + Các IC thay thế cho các linh kiện rời.

## CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 1

**Câu 1.** Nêu các thành phần cơ bản của máy tính ?

**Câu 2.** Phần mềm của máy tính là gì ?

**Câu 3.** Cấu trúc máy tính và kiến trúc máy tính giống và khác nhau như thế nào ?

**Câu 4.** Dựa vào tiêu chuẩn nào người ta phân chia máy tính thành các thế hệ ?

**Câu 5.** Đặc trưng cơ bản của các thế hệ máy tính ?

**Câu 6.** Khuynh hướng phát triển của các thế hệ máy tính ngày nay là gì ?



## Chương 2

### BIỂU DIỄN THÔNG TIN TRONG MÁY TÍNH

Trang bị cho sinh viên kiến thức về cách biến đổi cơ bản của hệ thống số, các bảng mã thông dụng được dùng để biểu diễn các ký tự. Cách chuyển đổi giữa các hệ thống sử dụng trong máy tính.

Sinh viên nắm vững kiến thức các khái niệm cơ bản liên quan đến các hệ thống số được dùng trong máy tính. Thành thạo các thao tác biến đổi số giữa các hệ thống số.

Các máy tính xử lý thông tin số và chữ. Các thông tin được biểu diễn dưới dạng mã nhất định. Bản chất vật lý của việc biểu diễn thông tin là điện áp ("0" ứng với không có điện áp, "1" ứng với điện áp ở mức quy chuẩn trong mạch điện tử) và việc mã hóa các thông tin số và chữ được tuân theo chuẩn quốc tế. Mã hiệu để mã hóa các thông tin cho máy tính xử lý là các giá trị của biến nhị phân 0,1. Một biến chỉ nhận được một trong hai giá trị duy nhất là 0 hoặc 1 được gọi là một bit. Hai trạng thái này của bit là các giá trị tương ứng với trạng thái "tắt" hoặc "đóng" của một công tắc điện, được sử dụng để mã hóa cho tất cả các ký tự (số, chữ, các ký tự đặc biệt).

Các bit được ghép lại thành các đơn vị mang thông tin đầy đủ. Với 8 bit ghép với nhau tạo thành đơn vị dữ liệu cơ sở của máy tính gọi là 1 Byte. Do được lưu giữ tương đương với một ký tự nên Byte là đơn vị cơ sở để đo khả năng lưu giữ, xử lý của các máy tính.

## 2.1. HỆ ĐẾM

### 2.1.1. Hệ thập phân

Hệ thập phân (hay hệ đếm cơ số 10) là một hệ đếm có 10 ký tự dùng chỉ số lượng.

Trong hệ thập phân, 10 ký tự (còn gọi là con số) khác nhau được dùng để biểu đạt 10 giá trị riêng biệt (0, 1, 2, 3, 4, 5, 6, 7, 8 và 9), tức là 10 con số. Những con số này còn được dùng cùng với dấu thập phân - ví dụ dấu "phẩy" - để định vị phần thập phân sau hàng đơn vị. Con số còn có thể được dẫn đầu bằng các ký hiệu "+" hay "-" để biểu đạt số dương và số âm.

Hệ thập phân là một hệ đếm dùng vị trí định lượng, bao gồm hàng đơn vị, hàng chục, hàng trăm v.v. Vị trí của một con số ám chỉ một phép nhân (mũ 10) với con số ở

vị trí đó, và mỗi con số về bên tay trái, có giá trị gấp mười lần con số kế bên, ở bên tay phải [7].

**Protected by PDF Anti-Copy Free**

Công thức chuyển đổi từ hệ thống số đếm khác sang hệ thập phân :

$$N = \underbrace{a_{n-1} \dots a_0}_n = a_0.s^0 + a_1.s^1 + \dots + a_{n-1}.s^{n-1} = \sum_{i=0}^{n-1} a_i s^i \quad (2.1)$$

Trong đó N là một số nguyên hữu số. Chữ số  $a_i$  tại vị trí  $i$  ( $i=0 \dots n-1$ ) được gọi là trị số (hay còn gọi là trọng số). Giá trị  $s$  là cơ số của hệ đếm. Hệ đếm được đặt tên theo giá trị cơ số  $s$ . Chẳng hạn, với  $s=2$  ta có hệ đếm 2, với  $s=10$  ta có hệ đếm 10, với  $s=16$  ta có hệ đếm 16. Giá trị  $s$  cũng xác định số ký tự cần dùng để biểu diễn trị số. Chẳng hạn với  $s=2$  hệ đếm này sẽ cần hai ký tự để biểu diễn, vì thế có khái niệm hệ nhị phân. Tương tự, có hệ đếm 10 và hệ đếm 16 còn được gọi là hệ thập phân và thập lục phân.

**Ví dụ 2.1.1:** Số nguyên  $(1537)_{10}$ . Giá trị số nguyên này được tính theo công thức (2.1) như sau:

$$N=1537D= 7 \times 10^0 + 3 \times 10^1 + 5 \times 10^2 + 1 \times 10^3$$

## 2.1.2. Hệ nhị phân (Binary)

### 2.1.2.1. Khái niệm

Hệ nhị phân hay hệ đếm cơ số 2 chỉ có hai con số 0 và 1. Đó là hệ đếm dựa theo vị trí. Giá trị của một số bất kỳ nào đó tùy thuộc vào vị trí của nó. Các vị trí có trọng số bằng bậc lũy thừa của cơ số 2. Chấm cơ số được gọi là chấm nhị phân trong hệ đếm cơ số 2. Mỗi một con số nhị phân được gọi là một bit (Binary digit). Bit ngoài cùng bên trái là bit có trọng số lớn nhất (MSB, Most Significant Bit) và bit ngoài cùng bên phải là bit có trọng số nhỏ nhất (LSB, Least Significant Bit)

$$\begin{array}{ccccccc} 2^3 & 2^2 & 2^1 & 2^0 & 2^{-1} & 2^{-2} & \\ \text{MSB} & 1 & 0 & 1 & 0 & . & 1 & 1 & \text{LSB} \end{array}$$

Chấm nhị phân

Dùng  $n$  bit có thể biểu diễn  $2^n$  giá trị

Cho một số nhị phân:

$$A = a_n a_{n-1} \dots a_1 a_0, a_{-1} \dots a_{-m}$$

Giá trị của A được tính như sau:

$$A = a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_1 2^1 + a_0 2^0 + a_{-1} 2^{-1} + \dots + a_{-m} 2^{-m}$$

### 2.1.2.2. **Protected by PDF Anti Copy Free**

(Upgrade to Pro Version to Remove the Watermark)

Các số nhị phân có thể được biến đổi thành thập phân bằng cách tính tổng của các con số, nhân với giá trị vị trí của chúng.

$$N = \underbrace{a_{n-1} \dots a_0}_n = a_0 \cdot s^0 + a_1 \cdot s^1 + \dots + a_{n-1} \cdot s^{n-1} = \sum_{i=0}^{n-1} a_i s^i$$

**Ví dụ 2.1.2:** Biến đổi số nhị phân  $(11001)_2$  thành số thập phân:

Trọng số vị trí:  $2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$

Giá trị vị trí:  $16 \quad 8 \quad 4 \quad 2 \quad 1$

Số nhị phân:  $1 \quad 1 \quad 0 \quad 0 \quad 1$

Số thập phân:  $1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (25)_{10}$

### 2.1.2.3. **Biến đổi thập phân sang nhị phân**

Có hai phương pháp:

*Phương pháp 1:* Chia dần cho 2 rồi lấy phần dư.

*Phương pháp 2:* Phân tích thành tổng của các số  $2^i$  (Nhanh hơn với các số thập phân có giá trị nhỏ).

*Phương pháp 1:*

Với phần nguyên, ta thực hiện phương pháp chia lặp, tức là chia liên tiếp số thập phân cho cơ số của hệ đếm (cơ số 2) cho đến khi thương số bằng 0 và thực hiện lấy số dư theo thứ tự số dư cuối cùng là chữ số có trọng số lớn nhất và số dư đầu tiên là chữ số có trọng số nhỏ nhất.

Đối với phần lẻ của các số thập phân, số lẻ được nhân với cơ số (cơ số 2) và số nhớ được ghi lại làm một số nhị phân. Trong quá trình biến đổi, số nhớ đầu chính là **bit** MSB và số nhớ cuối là **bit** LSB.

**Ví dụ 2.1.3:** Biến đổi số thập phân  $(29.625)_{10}$  thành nhị phân:

Biến đổi số  $(29)_{10}$  sang nhị phân:

$$29/2 = 14 \quad \text{dư} \quad 1(\text{LSB})$$

$$14/2 = 7 \quad \text{dư} \quad 0$$

$$7/2 = 3 \quad \text{dư} \quad 1$$

$$3/2 = 1 \text{ dư } 1$$

**Protected by PDF Anti-Copy Free**

**Upgrade to Pro Version to Remove the Watermark**

$$\Rightarrow (29)_{10} = (11101)_2.$$

Biến đổi số thập phân (0.625)<sub>10</sub> sang hệ nhị phân:

$$0.625 \times 2 = 1.250. \text{ Số nhớ là 1, là bit MSB.}$$

$$0.250 \times 2 = 0.500. \text{ Số nhớ là 0}$$

$$0.500 \times 2 = 1.000. \text{ Số nhớ là 1, là bit LSB.}$$

$$\Rightarrow (0.625)_{10} = (0.101)_2.$$

$$\text{Vậy : } (29.625)_{10} = (11101.101)_2$$

**Ví dụ 2.1.4 :** Biến đổi số thập phân (24.25)<sub>10</sub> sang hệ nhị phân:

Biến đổi (24)<sub>10</sub> sang hệ nhị phân:

$$24/2 = 12 \quad \text{dư } 0$$

$$12/2 = 6 \quad \text{dư } 0$$

$$6/2 = 3 \quad \text{dư } 0$$

$$3/2 = 1 \quad \text{dư } 1$$

$$1/2 = 0 \quad \text{dư } 1$$

$$\text{Vậy : } (24)_{10} = (11000)_2$$

Biến đổi (0.25)<sub>10</sub> sang hệ nhị phân:

$$0.25 \times 2 = 0.5 \quad \text{số nhớ là 0}$$

$$0.5 \times 2 = 1.0 \quad \text{số nhớ là 1}$$

$$\text{Vậy : } (0.25)_{10} = (0.01)_2$$

$$\text{Vậy : } (24.25)_{10} = (11000.01)_2$$

*Phương pháp 2:*

Ta chỉ việc phân tích số thập phân thành tổng của các số  $2^i$ .

**Ví dụ 2.1.5 :** Chuyển đổi  $105_{(10)}$  sang hệ nhị phân :

$$105 = 64 + 32 + 8 + 1 = 2^6 + 2^5 + 2^3 + 2^0$$

$$\text{Kết quả : } 105_{(10)} = 01101001_{(2)}$$

**Ví dụ 2.1.6 :** Chuyển đổi  $17000_{10}$  sang hệ nhị phân :

$$17000_{10} = 16384 + 616 = 16384 + 512 + 104 = 16384 + 512 + 64 + 40 = 16384 + 512 + 64 + 32 + 8 = 2^{14} + 2^{10} + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$$

Kết quả :  $17000_{10} = 010000100001000_2$

### 2.1.3. Hệ thập lục phân (Hexadecimal)

#### 2.1.3.1. Khái niệm

Các hệ máy tính hiện đại thường dùng một hệ đếm khác là hệ thập lục phân. Hệ thập lục phân là hệ đếm dựa vào vị trí với cơ số là 16. Hệ này dùng các con số từ 0 đến 9 và các ký tự từ A đến F như trong bảng sau:

**Bảng 2.1. Hệ thập lục phân**

Thập lục phân	Thập phân	Nhị phân
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

#### 2.1.3.2. Biến đổi thập lục phân sang thập phân

Các số thập lục phân có thể được biến đổi thành thập phân bằng cách tính tổng của các con số, nhân với giá trị vị trí của nó.

$$N = \underbrace{a_{n-1} \dots a_0}_n = a_0.s^0 + a_1.s^1 + \dots + a_{n-1}.s^{n-1} = \sum_{i=0}^{n-1} a_i s^i$$

**Ví dụ 2.1.7:** Biến đổi các số  $(2AF)_{16}$  thành thập phân.

Số thập phân:  $2AF$   
Trọng số vị trí:  $16^2$   $16^1$   $16^0$

Giá trị vị trí: 256

Số thập phân:  $2 \times 256 + F \times 16 + A \times 1 = (687)_{10}$ .

### 2.1.3.3. Biến đổi thập phân sang thập lục phân

Để biến đổi các số thập phân thành thập lục phân, ta sử dụng phương pháp chia lặp, với cơ số 16.

**Ví dụ 2.1.8 :** Biến đổi  $(1776)_{10}$  thành thập lục phân.

$$1776/16 = 111 \text{ dư } 0 \text{ (LSB).}$$

$$111/16 = 6 \text{ dư } 15 \text{ hoặc F.}$$

$$6/16 = 0 \text{ dư } 6 \text{ (MSB).}$$

Số thập lục phân:  $(6F0)_{16}$ .

**Ví dụ 2.1.9:**

Biến đổi  $(24A)_{16}$  thành thập phân.

Biến đổi  $(586)_{10}$  thành thập lục phân.

Biến đổi  $(24A)_{16}$  sang hệ thập phân:

- Số thập lục phân: 2 4 A

Trọng số vị trí:  $16^2$   $16^1$   $16^0$

Giá trị vị trí: 256 16 1

Số thập phân:  $2 \times 256 + 4 \times 16 + 10 \times 1 = 586_{10}$

Vậy:  $(24A)_{16} = (586)_{10}$

- Biến đổi  $(586)_{10}$  sang hệ thập lục phân:

$$586/16 = 36 \text{ dư } 10 \text{ hay A}$$

$$36/16 = 2 \text{ dư } 4$$

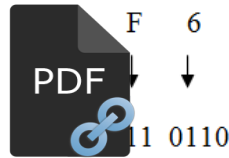
$$2/16 = 0 \text{ dư } 2$$

Vậy:  $(586)_{10} = (24A)_{16}$

### 2.1.3.4. Biến đổi thập lục phân sang nhị phân

Để đổi các số thập lục phân thành nhị phân, chỉ cần thay thế một cách đơn giản từng con số thập lục phân bằng bốn bit nhị phân tương đương của nó.

**Ví dụ 2.1.10:** Đổi số thập lục phân  $(DF6)_{16}$  thành nhị phân.

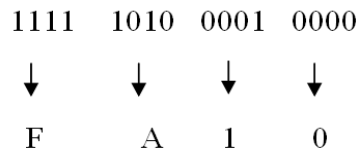


$$(DF6)_{16} = (110111110110)_2.$$

### 2.1.3.5. Biến đổi nhị phân sang thập lục phân

Để biến đổi một số nhị phân thành số thập lục phân tương đương thì chỉ cần gộp lại thành từng nhóm gồm 4 bit nhị phân, bắt đầu từ dấu chấm nhị phân.

**Ví dụ 2.1.11:** Biến đổi số nhị phân  $(1111101000010000)_2$  thành thập lục phân.

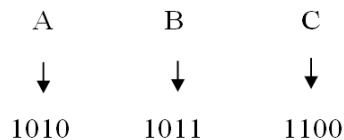


Số thập lục phân:  $(FA10)_{16}$ .

**Ví dụ 2.1.12:** Chuyển số  $(ABC)_{16}$  sang hệ nhị phân.

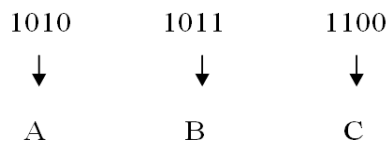
Chuyển  $(1010\ 1011\ 1100)_2$  sang hệ thập lục phân.

Chuyển số  $(ABC)_{16}$  sang hệ nhị phân:



$$(ABC)_{16} = (1010\ 1011\ 1100)_2$$

Chuyển số  $(1010\ 1011\ 1100)_2$  sang hệ thập lục phân:

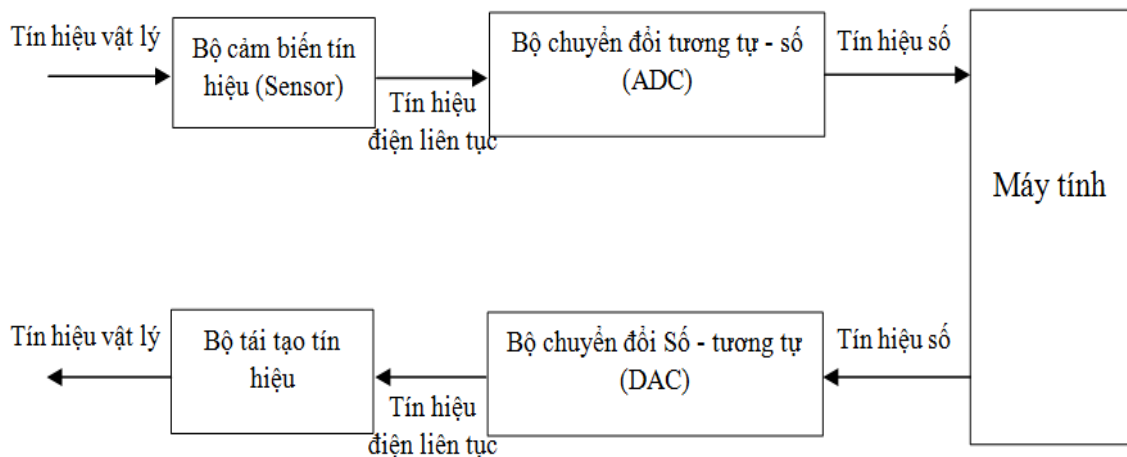


$$(1010\ 1011\ 1100)_2 = (ABC)_{16}$$

## 2.2. BIỂU DIỄN DỮ LIỆU SỐ TRONG MÁY TÍNH

### 2.2.1. Nguyên tắc mã hóa dữ liệu

- Mọi dữ liệu đưa vào máy tính đều phải mã hóa thành số nhị phân.
- Có hai loại dữ liệu: Dữ liệu nhân tạo và dữ liệu tự nhiên
  - Mã hóa dữ liệu nhân tạo
    - + Dữ liệu số nguyên: Mã hóa theo tiêu chuẩn quy ước
    - + Dữ liệu số thực: Mã hóa bằng số dấu phẩy động.
    - + Dữ liệu ký tự: Mã hóa theo bộ mã ký tự.
  - Mã hóa và tái tạo dữ liệu vật lý (Tín hiệu tự nhiên):



**Hình 2.1. Sơ đồ khối mã hóa và tái tạo dữ liệu vật lý**

Các dữ liệu vật lý thông dụng: Âm thanh và hình ảnh.

Độ dài dữ liệu: Độ dài từ dữ liệu là số bit được sử dụng để mã hóa loại dữ liệu tương ứng. Độ dài từ dữ liệu thường là bội của 8 – bit: 8, 16, 32, 64 bit

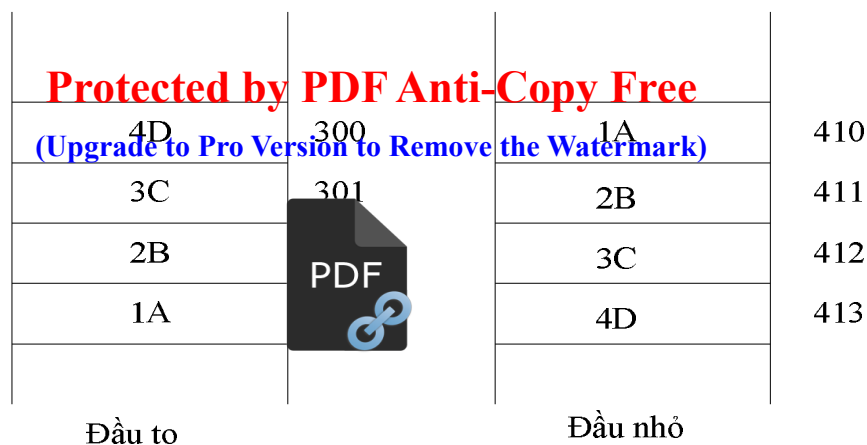
### 2.2.2. Thứ tự lưu trữ các byte của dữ liệu

- Bộ nhớ chính thường được tổ chức theo byte.
- Độ dài từ dữ liệu có thể chiếm từ 1 đến nhiều byte. Đối với dữ liệu nhiều byte ta cần phải biết thứ tự lưu trữ các byte trong bộ nhớ chính.
- Có hai cách lưu trữ:

Đầu nhỏ (Little - endian): Byte có ý nghĩa thấp hơn được lưu trữ trong bộ nhớ ở vị trí có địa chỉ nhỏ hơn.

Đầu to (Big - endian): Byte có ý nghĩa thấp hơn được lưu trữ trong bộ nhớ ở vị trí có địa chỉ lớn hơn.





**Hình 2.2. Lưu trữ các byte của dữ liệu**

- Các cách lưu trữ của các bộ xử lý điển hình:

Intel 80x86 và các loại Pentium lưu trữ kiểu đầu nhỏ

Motorola 680x0 và các bộ xử lý RISC lưu trữ kiểu đầu to

Power PC và Itanium lưu trữ cả 2 kiểu

**Ví dụ 2.2.1:** Lưu trữ dữ liệu 32-bit

0001	1010	0010	1011	0011	1100	0100	1101
1	A	2	B	3	C	4	D

### 2.2.3. Biểu diễn số nguyên

#### 2.2.3.1. Biểu diễn số nguyên không dấu

Tất cả các số cũng như các mã trong máy vi tính đều được biểu diễn bằng các chữ số nhị phân. Để biểu diễn các số nguyên không dấu, người ta dùng  $n$  bit.

$$A = a_{n-1} a_{n-2} \dots a_1 a_0 \quad (a = 1; 0)$$

Giá trị của  $A$  được tính như sau:

$$A = a_{n-1} 2^{n-1} + a_{n-2} 2^{n-2} + \dots + a_1 2^1 + a_0 2^0$$

Với  $n = 8$  bit biểu diễn được các giá trị từ 0...255

Giá trị 255 được biểu diễn là 1111 1111.

1111 1111

+0000 0001

---

10000 0000

Vậy:  $255+1 = 0$

Trục số hợp nhất của Bit để biểu diễn A

Theo trục số hợp nhất với  $n = 8$  thì biểu diễn được 256 giá trị từ 0...255. Tới giá trị 255 thì trục số lại quay lại giá trị 0.

Tương ứng với độ dài của số nhị phân sử dụng, ta có các khoảng giá trị xác định như sau:

Số bit	Khoảng giá trị
n bit:	$0.. 2^n - 1$
8 bit	0.. 255 Byte
16 bit	0.. 65535 Word

**Ví dụ 2.2.2:** Biểu diễn các số nguyên không dấu sau bằng thanh ghi 8 bit

$$A = (41)_{10}; B = (150)_{10}$$

*Hướng dẫn:*

$$A = (41)_{10} = 32 + 8 + 1 = 2^5 + 2^3 + 2^0$$

$$41_{10} = 0010\ 1001$$

$$B = (150)_{10} = 128 + 16 + 4 + 2 = 2^7 + 2^4 + 2^2 + 2^1$$

$$150_{10} = 1001\ 0110$$

**Ví dụ 2.2.3:** Cho các số nguyên không dấu M, N được biểu diễn bằng thanh ghi 8 bit như sau:

$$M = 0001\ 0010$$

$$N = 1011\ 1001$$

Xác định giá trị của chúng.

*Hướng dẫn:*

$$M = 0001\ 0010 = 2^4 + 2^1 = 16 + 2 = 18_{10}$$

$$\begin{aligned} N &= 1011\ 1001 = 2^7 + 2^5 + 2^4 + 2^3 + 2^0 \\ &= 128 + 32 + 16 + 8 + 1 = 185 \end{aligned}$$

### 2.2.3.2. Biểu diễn số nguyên có dấu

Số bù một và số bù hai:

Gọi A là một số nhị phân.

Số bù 1 của A nhận được bằng cách đảo giá trị các bit của A

Số bù 2 của A = Số bù 1 của A + 1

Ví dụ 2.2.4: Giả sử A = 0010 0101

Số bù 1 của A = 1101 1010

$$\begin{array}{r} \text{Số bù 1 của A} \\ 1101\ 1010 \\ + 1 \\ \hline \text{Số bù 2 của A} = 1101\ 1011 \end{array}$$

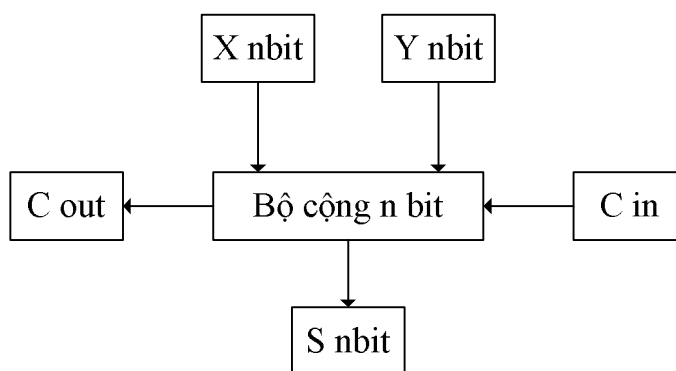
Ta thấy: A + (Số bù 2 của A) = 0

Nên số bù 2 biểu diễn được số âm. Chính vì thế mà số bù 2 dùng để biểu diễn số nguyên có dấu.

Người ta sử dụng bit cao nhất biểu diễn dấu; bit dấu có giá trị 0 tương ứng với số nguyên dương, bit dấu có giá trị 1 biểu diễn số âm. Như vậy khoảng giá trị số được biểu diễn sẽ được tính như sau:

Số bit	Khoảng giá trị:	
n bit	$-2^{n-1} \dots 2^{n-1} - 1$	
8 bit	$-128 \div 127$	Short integer
16 bit	$-32768 \div 32767$	Integer
32 bit	$-2147483648 \div 2147483647$	Long integer

#### 2.2.4. Các phép toán số học với số nguyên



**Hình 2.3. Sơ đồ khối phép toán số học với số nguyên**

*Nguyên tắc thực hiện phép toán với số nguyên*

Bit có trọng số lớn nhất là bit dấu:

Nếu bit dấu là:

00: Số dương

01: Trần dương

10: Trần a

11: Số an

*Phép cộng số nguyên không dấu:*

Nếu không có nhớ ra khỏi bit

$$C_{out} = 0$$

Nếu có nhớ ra khỏi bit MSB thì kết quả nhận được là sai:

$$C_{out} = 1 \text{ (Hiện tượng tràn số)}$$

Hiện tượng tràn số xảy ra khi tổng lớn hơn  $2^n - 1$

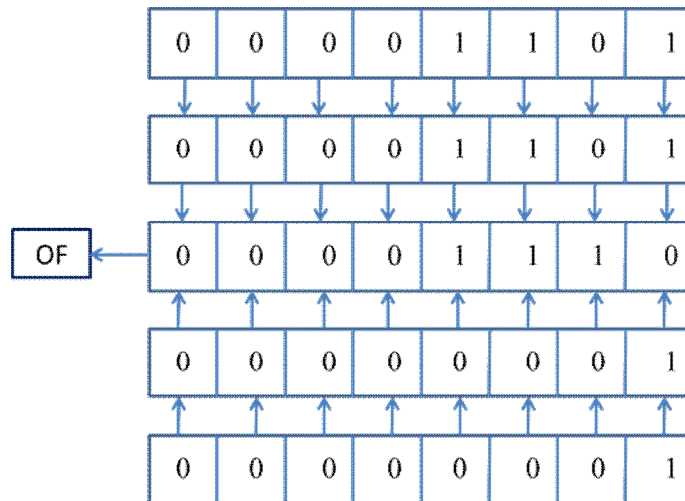
Nếu xuất hiện, hiện tượng tràn số thì ta tăng số lượng các bit lên (tăng n). Ví dụ từ 8 lên 16 bit

**Ví dụ 2.2.5:** Cho C = 1101; D = 0001

$$S = C + D$$

*Hướng dẫn:*

Tính S:



Kết quả trong thanh ghi là: 00001110. Ta thấy bit dấu là 00 nên kết quả thu được sẽ là số dương 001110 có giá trị là 3

$$\text{Vậy } S = +3$$

*Phép cộng số nguyên có dấu:*

Biểu diễn số nguyên có dấu dưới dạng mã bù 2 của số không dấu trong thanh ghi n bit

**Protected by PDF Anti-Copy Free**

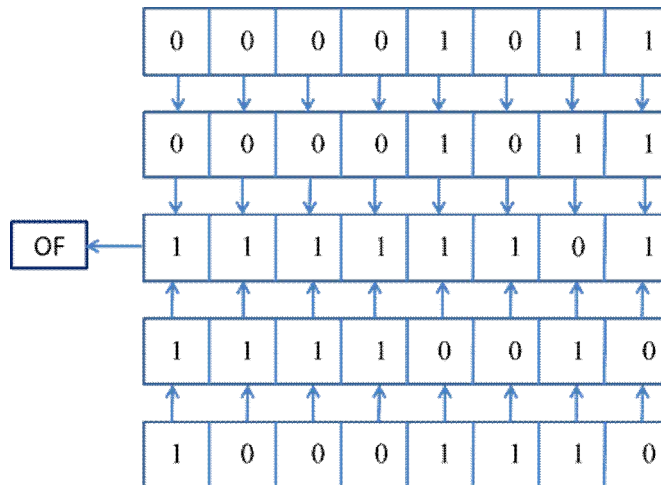
Thực hiện phép cộng hai số có dấu. (Remove the Watermark)

Chúng ta phải chú ý hiện tượng tràn số. Nếu xuất hiện tràn số thì ta tăng số lượng các bit lên (tăng n). Ví dụ từ

**Ví dụ 2.2.6:** Cho A = 1011; B = -1 và S = A + B

*Hướng dẫn:*

Tính S:



Kết quả trong thanh ghi là:

1111101. Ta thấy bit dấu là 11 nên kết quả nhận được là một số âm.

111101 có số bù 2 là: 000011.

Vậy kết quả của S = -3

## 2.2.5. Biểu diễn số thực

Có hai cách biểu diễn số thực trong một hệ nhị phân: Số có dấu chấm cố định (fixed point number) và số có dấu chấm động (floating point number). Cách thứ nhất được dùng trong những bộ vi xử lý (micro processor) và những bộ vi điều khiển (micro controller) cũ. Cách thứ hai được sử dụng hiện nay có độ chính xác cao hơn. Đối với cách biểu diễn số thực dấu chấm động có khả năng hiệu chỉnh theo giá trị của số thực. Cách biểu diễn chung cho mọi hệ đếm như sau:

$$R = m.B^e$$

Trong đó: m là phần định trị, trong hệ thập phân giá trị tuyệt đối của nó phải luôn nhỏ hơn 1. Số là phần mũ và E là cơ số của hệ đếm.

Có ba chuẩn định dạng dấu chấm động hiện nay là (University of Waterloo) Chuẩn MSBIN của Microsoft, chuẩn IEEE và chuẩn IBM 360. Cả hai chuẩn MSBIN và IEEE đều dùng hệ đếm nhị phân. Còn chuẩn IBM 360 dùng hệ đếm thập lục phân.

Thường dùng là theo tiêu chuẩn biểu diễn số thực của IEEE 754, đây là chuẩn được mọi hãng chấp nhận và được dùng trong bộ xử lý toán học của Intel. Bit dấu nằm tại vị trí cao nhất; kích thước phần mũ và khuôn dạng phần định trị thay đổi theo từng loại số thực.

Giá trị số thực IEEE được tính theo biểu thức:

$$R = (-1)^S \times (1 + M_1 \cdot 2^{-1} + \dots + M_n \cdot 2^{-n}) \times 2^{E - E_0 - 127} \quad (2.2)$$

Chú ý: Giá trị đầu tiên  $M_0$  luôn mặc định là 1.

- Dùng 32 bit để biểu diễn số thực, được số thực ngắn:  $-3,4 \cdot 10^{38} < R < 3,4 \cdot 10^{38}$

31	30.....23	22.....0
S	E7 - E0	Định trị (M1 - M23)

- Dùng 64 bit để biểu diễn số thực, được số thực dài:  $-1,7 \cdot 10^{308} < R < 1,7 \cdot 10^{308}$

63	62.....52	51.....0
S	E7 - E0	Định trị (M1 - M23)

**Ví dụ 2.2.7:** Tính số thực

0100 0011 0 111 0110 0000 0000 0000 0000

---

Phần định trị:  $1 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-6}$   
 $= 1 + 0.5 + 0.25 + 0.125 + 0.03125 + 0.015625 = 1.921875$

Phần mũ:  $2^7 + 2^2 + 2^1 = 134$

Giá trị thực (bit cao nhất là bit dấu):  $134 - 127 = 7$ .

---

Dấu: 0 = số dương

Giá trị số thực là:  $R = 1,921875 \cdot 2^7$

\*) *Phuong phap di so thuc lang so du phay dong* 2016

- Đổi số thập phân thành số nhị phân.

- Biểu diễn số nhị phân dạng  $\pm 1, xxx.2^y$

- Bit cao nhất 31: Lấy giá trị số dương, 1 với số âm.

- Xác định bit từ 30-23: được xác định bằng cách:

$$y + (7F)_{16} \text{ hay } y + (127)_{10}$$

- Phần xxx là phần định trị, được đưa vào từ bit 22...0

**Ví dụ 2.2.8:** Biểu diễn số thực  $A = (-9.75)_{10}$  dưới dạng dấu phẩy động 32 bit theo chuẩn IEEE 754.

*Hướng dẫn:*

Đổi sang dạng nhị phân:  $(-9.75)_{10} = (-1001,11)_2 = -1,00111.2^3$

Bit dấu: do là một số âm nên  $s=1$

Xác định bit từ 30-23:  $y + (7F)_{16} = 3 + 127 = (130)_{10} = (10000010)_2$

Xác định các bit từ 22...0: 00111...0

Cuối cùng số thực  $(-9.75)_{10}$  được biểu diễn dưới dạng dấu phẩy động 32 bit như sau:

1 100 0001 0 001 1100 0000 0000 0000 0000  
 bit |31|30 23|22 0|

bit |31|30

23|22

01

**Ví dụ 2.2.9:** Biểu diễn số thực  $A = (123)_{10}$  dưới dạng dấu phẩy động 32 bit theo chuẩn IEEE 754

*Hướng dẫn:*

Ta đổi sang dạng nhị phân:  $(123)_{10} = (0111\ 1011)_2 = 1,111011\ 2^6$

Bit dấu: do là một số dương nên bit 31 = 0.

Xác định bit từ 30-23:  $6 + 127 = (1000\ 0101)_2$ .

Xác định các bit từ 22...0: 111011...0

Cuối cùng số thực  $(123)_{10}$  được biểu diễn dưới dạng dấu phẩy động 32 bit như sau: 0100 0010 1111 0110 0000 0000 0000 0000

### 2.2.6. Biểu diễn ký tự

Để biểu diễn thông tin mà con người có thể hiểu được, máy vi tính cần một giao diện thích hợp. Nên con người đã xây dựng bộ mã để biểu diễn cho các ký tự cũng như các con số và các ký hiệu đặc biệt khác. Các mã đó gọi là *bộ mã ký tự và số*. Ta có hai bộ mã thông dụng là bộ mã (American Standard Code For Information Interchange) và bộ mã Unicode. Nó còn có chuẩn Latin1 dùng 8 bit để biểu diễn các ký tự, chuẩn ISO 10646 dùng 24 bit để biểu diễn các ký tự.

#### 2.2.6.1. Bộ mã ASCII

Bộ mã ASCII do ANSI (American National Standard Institute) thiết kế.

Bảng mã ASCII là mã 7 bit được dùng phổ biến trong các hệ máy tính hiện nay. Với mã 7 bit nên có  $2^7 = 128$  tổ hợp mã. Mỗi ký tự (chữ hoa và chữ thường) cũng như các con số thập phân từ 0...9 và các ký hiệu đặc biệt khác đều được biểu diễn bằng một mã số.

**Bảng 2.2. Phân bố mã trong ASCII cơ bản**

Ký tự	Mã thập phân	Mã thập lục phân
Null	0	00
31 ký tự điều khiển	1-31	01-1F
Các dấu	32-47	20-2F
Số 0-9	48-57	30-39
Các dấu khác	58-64	3A-40
A-Z	65-90	41-5A
Các dấu khác	91-96	5B-60
a-z	97-122	61-7A
Các dấu khác	123-127	7B-7F



**Bảng 2.3. Bảng mã ASCII**

**Protected by PDF Anti-Copy Free**

(Upgrade to Pro Version to Remove the Watermark)

Bits(row)					000	001	010	011	100	101	110	111
R	B <sub>4</sub>	B <sub>3</sub>	B <sub>2</sub>	B <sub>1</sub>		1	2	3	4	5	6	7
O						↓	↓	↓	↓	↓	↓	↓
W												
0	0	0	0	0	→	NUL	DLE	SP	0	@	P	\
1	0	0	0	1	→	SOH	DC1	!	1	A	Q	a
2	0	0	1	0	→	STX	DC2	"	2	B	R	b
3	0	0	1	1	→	ETX	DC3	#	3	C	S	c
4	0	1	0	0	→	EOT	DC4	\$	4	D	T	d
5	0	1	0	1	→	ENQ	NAK	%	5	E	U	e
6	0	1	1	0	→	ACK	SYN	&	6	F	V	f
7	0	1	1	1	→	BEL	ETB	'	7	G	W	g
8	1	0	0	0	→	BS	CAN	(	8	H	X	h
9	1	0	0	1	→	HT	EM	)	9	I	Y	i
A	1	0	1	0	→	LF	SUB	*	:	J	Z	j
B	1	0	1	1	→	VT	ESC	+	;	K	[	k
C	1	1	0	0	→	FF	FS	-	<	L	\	l
D	1	1	0	1	→	CR	GS	,	=	M	]	m
E	1	1	1	0	→	SO	RS	.	>	N	^	n
F	1	1	1	1	→	SI	US	/	?	O	_	o
												DEL

**Ví dụ 2.2.10:** Đổi ký tự BILL thành mã ASCII

Ký tự	B	I	L	L
ASCII	1000010	1001001	1001100	1001100
HEXA	42	49	4C	4C

Một đặc điểm lý thú là mã ký tự chữ cái trừ đi 64 sẽ ra mã điều khiển.

**Ví dụ 2.2.11:** Mã ký tự chữ cái và mã điều khiển

Ký tự	D	H
ASCII	1000100	1001000

Thực hiện phép tính như hướng dẫn sau:

Sau khi lấy giá trị tương ứng tại bảng mã ASC II của D đem trừ đi  $64_{10}$  ta sẽ được giá trị mới là 0000100.

Giá trị này tương ứng là BS (Protected by PDF Anti-Copy Free)

Tương tự : Ký tự H mã điều khiển tương ứng của nó là BS

Đối với bảng chữ cái không tiếng Anh chẳng hạn như tiếng Việt, ta cần nhiều mã hơn bộ mã ASCII cơ bản để biểu diễn những ký tự này:

Dùng phần ASCII mở rộng gồm cả 8 bit của một byte. Ta sẽ có cả thảy 256 ký tự, chúng đủ cho chữ cái của các thứ tiếng châu Âu, tạm đủ cho tiếng Việt, nhưng không đủ cho các chữ cái tượng hình (Hán, Hàn, Nhật).

Dùng nhiều byte để biểu diễn mã một ký tự. Phương án này sẽ khiến văn bản lớn lên rất nhiều lần so với văn bản dùng mã ASCII. Trong phương án này có chuẩn Unicode do hãng Xerox đề nghị dùng 2 byte để mã hóa một ký tự.

#### 2.2.6.2. Bộ mã Unicode

Chuẩn Unicode do hãng Xerox đề nghị dung 2 byte để mã hóa một ký tự.

Bộ mã Unicode do các hãng sản xuất máy tính hàng đầu thiết kế. Nó là một bộ mã 6 bit và là bộ mã đa ngôn ngữ.

Bộ mã có hỗ trợ ký tự tiếng Việt (bao gồm cả chữ Nôm)

#### 2.2.6.3. Mã BCD (Binary Coded Decimal )

Giữa hệ thập phân và hệ nhị phân còn tồn tại một hệ lai: Hệ BCD cho các số *hệ thập phân mã hoá bằng hệ nhị phân*. Ở đây dùng bốn số hệ nhị phân (bốn bit) để mã hoá một số hệ thập phân có giá trị nằm trong khoảng từ 0÷9.

0 -> 0000

1 -> 0001

2 -> 0010

3 -> 0011

4 -> 0100

5 -> 0101

6 -> 0110

7 -> 0111

8 -> 1000

9 -> 1001

Như vậy BCD dùng để biểu diễn các tổ hợp 4 bit. Vì tầm quan trọng của các số BCD nên có bộ vi xử lý được tích hợp các mạch này.

6 tổ hợp không sử dụng là: 1011, 1100, 1101, 1110, 1111.

*Phép cộng số BCD*

*Ví dụ 2.2.12:* Thực hiện phép cộng

$$\begin{array}{r} 35 + 61 \\ 35 \rightarrow (0011\ 0101)_{BCD} \\ +\ 61 \rightarrow (0110\ 0001)_{BCD} \\ \hline 96 \leftarrow (1001\ 0110)_{BCD} \end{array}$$

*Ví dụ 2.2.13:* Thực hiện phép cộng số BCD

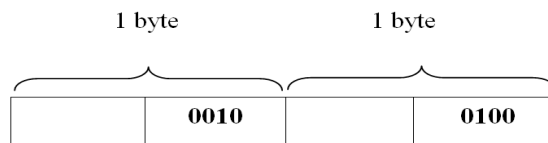
$$\begin{array}{r} 87 \rightarrow (1000\ 0111)_{BCD} \\ +\ 96 \rightarrow (1001\ 0110)_{BCD} \\ \hline 183\ \ (10001\ 1101) \rightarrow \text{Kết quả sai} \\ +\ 0110\ 0110 \leftarrow \text{Hiệu chỉnh} \\ \hline 1\ 1000\ 0011 \rightarrow 183 \end{array}$$

Thực hiện bước hiệu chỉnh bằng cách cộng 0110 (6) vào các vị trí có nhớ

### Các kiểu lưu trữ số BCD

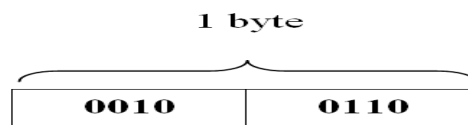
BCD không gói (Unpacked BCD): Mỗi số BCD 4 bit được lưu trữ trong 4 bit thấp của mỗi Byte.

*Ví dụ 2.2.14:* Số 26 được lưu trữ như sau:



BCD gói (Packed BCD): Hai số BCD được lưu trữ trong một byte

*Ví dụ 2.2.15:* Số 26 được lưu trữ như sau:



## CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 2

**Protected by PDF Anti-Copy Free**

**Câu 1:** Thực hiện phép toán sau bằng thanh ghi 8 bit

a,  $S1 = 61 - 79$

b,  $S2 = 65 - 9$

c,  $S3 = 12 - 25$

d,  $S4 = 24 - 12$



**Câu 2:** Thực hiện phép toán sau bằng thanh ghi 16 bit

a,  $S1 = 329 - 163$

b,  $S2 = 365 - 139$

c,  $S3 = 134 - 134$

d,  $S4 = 120 - 86$

**Câu 3:** Thực hiện phép toán sau

a,  $S1 = (123)_{BCD} + (546)_{BCD}$

b,  $S2 = (789)_{BCD} + (456)_{BCD}$

c,  $S3 = (123)_{BCD} + (115)_{BCD}$

d,  $S4 = (127)_{BCD} + (240)_{BCD}$

**Câu 4:** Biểu diễn các số thực sau về dạng dấu phẩy động 32 bit.

$(58,25)_{10} ; (-25,75)_{10}$

*(Upgrade to Pro Version to Remove the Watermark)*

Trang bị cho sinh viên kiến thức về cấu trúc, chức năng của bộ xử lý trung tâm CPU. Giúp sinh viên nắm vững kiến thức về các thanh ghi, đơn vị số học và logic, đơn vị điều khiển nằm bên trong bộ xử lý.



### **3.1. CẤU TRÚC, CHỨC NĂNG CỦA BỘ XỬ LÝ**

#### **3.1.1. Chức năng của bộ xử lý**

Bộ xử lý trung tâm CPU là cốt lõi của một máy vi tính. CPU thực hiện mọi tính toán và xử lý của hệ thống (ngoại trừ xử lý tăng cường tính toán đặc biệt trong những hệ thống có một chip đơn vị đồng xử lý toán, mà chip này cũng đã được tích hợp ngay trong các CPU hiện nay).

Một trong những bộ xử lý điển hình thuộc họ 80x86 của Intel là bộ xử lý 8088. Đây là bộ xử lý khá đơn giản. Vì vậy việc tìm hiểu nó là tương đối dễ đối với những người bắt đầu thâm nhập vào lĩnh vực vi xử lý, mặt khác việc nắm vững các vấn đề kỹ thuật của bộ vi xử lý 8088 sẽ là cơ sở để nắm bắt được các kỹ thuật của các bộ xử lý khác trong họ 80x86 của Intel, của các họ khác và của các bộ xử lý hiện đại ngày nay.

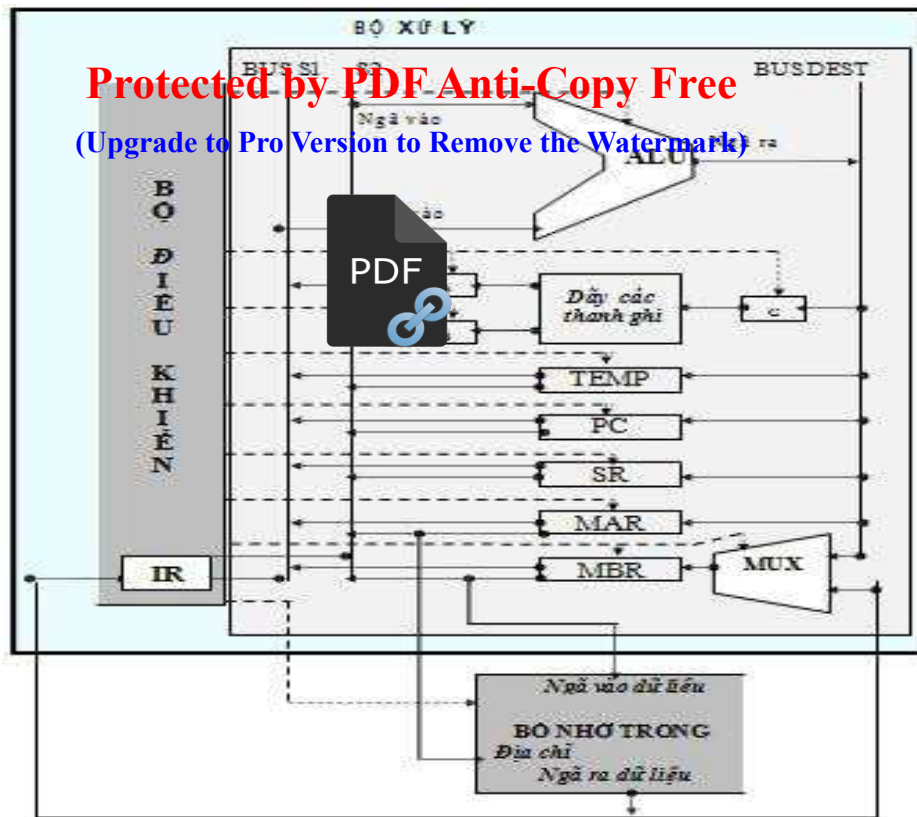
#### **3.1.2. Cấu trúc của bộ vi xử lý**

CPU có 3 bộ phận chính:

- Khối điều khiển (CU): Nhận lệnh của chương trình từ bộ nhớ trong đưa vào CPU. Nó có nhiệm vụ giải mã các lệnh, tạo ra các tín hiệu điều khiển công việc của các bộ phận khác của máy tính theo yêu cầu người sử dụng.

- Khối tính toán số học và logic (ALU): Bao gồm các thiết bị thực hiện các phép toán số học, phép tính logic và các tính quan hệ. Độ dài của các toán hạng được đưa vào tính toán trực tiếp ở khối ALU. Độ dài phổ biến với các máy tính hiện nay là 32 hay 64 bit.

- Tập các thanh ghi: Các thanh ghi mang chức năng chuyên dụng tăng tốc độ trao đổi thông tin trong máy tính.



Hình 3.1. Tổ chức của một xử lý điển hình

## 3.2. CÁC THANH GHI

### 3.2.1. Các thanh ghi đa năng (general registers)

Có nhiệm vụ ghi tham số cho mã lệnh, đây cũng là nơi lệnh trả kết quả về sau khi được thực hiện. Những thanh ghi đa năng của vi xử lý 16 bit là [6,7]:

- *AX (accumulator)* rộng 16 bit, được chia làm hai phần: 1 byte cao AH và 1 byte thấp AL. Đây là thanh ghi quan trọng nhất và chuyên được dùng để chứa kết quả các thao tác lệnh. Cả ba cách viết AX, AH, AL đều có thể sử dụng như những thanh ghi riêng biệt.

- *BX (base)* thanh ghi cơ sở, rộng 16 bit, cũng được chia ra làm BH và BL. Đây là thanh ghi thường dùng chứa địa chỉ cơ sở của một bảng dùng trong lệnh XLAT. Cả ba cách viết BX, BH, BL đều có thể sử dụng như những thanh ghi riêng biệt.

- *CX (count)* bộ đếm, rộng 16 bit. Được chia ra làm CH và CL. Thanh ghi CX được dùng để chứa số lần lặp trong trường hợp các lệnh LOOP. Thanh ghi thấp CL được dùng để chứa (nhớ) số lần quay hoặc dịch của các lệnh quay (rotate) và dịch (shift).

- *DX (data)* thanh ghi dữ liệu, rộng 16 bit. Thanh ghi này cùng thanh ghi AX tham gia vào các thao tác của phép nhân hoặc chia. Các số 16 bit DX còn dùng để chứa địa chỉ 16 bit của các cổng cứng (đài hơn 8 bit) trong các lệnh truy nhập các cổng ngoại vi (I/O port).

### 3.2.2. Các thanh ghi đoạn (segment registers)

Các thanh ghi đoạn dùng để chỉ một đoạn bộ nhớ. Vì mạch 8088/8086 có 20 đường dây trên bus địa chỉ. Do các thanh ghi con trỏ, thanh ghi chỉ số chỉ rộng 16 bit nên không thể định địa chỉ cho toàn bộ nhớ vật lý của máy tính là  $(2^{20} = 1.048.576 = 1\text{Mbyte})$ . Vì vậy trong chế độ thực (real mode) bộ nhớ được chia làm nhiều đoạn để một thanh ghi con trỏ 16 bit có thể quản lý được. Các thanh ghi đoạn 16 bit sẽ chỉ ra địa chỉ đầu của 4 đoạn trong bộ nhớ, dung lượng lớn nhất của mỗi đoạn nhớ sẽ dài  $2^{16} = 64\text{Kbyte}$  và tại một thời điểm nhất định bộ vi xử lý chỉ làm việc được với 4 đoạn nhớ 64Kbyte này. Việc thay đổi giá trị của các thanh ghi đoạn làm cho các đoạn có thể dịch chuyển linh hoạt trong không gian 1 Mbyte, vì vậy các đoạn có thể nằm cách nhau khi thông tin cần lưu trong chúng đòi hỏi dung lượng đủ 64 Kbyte hoặc cũng có thể nằm trùm nhau do có những đoạn không dùng hết độ dài 64 Kbyte và vì thế các đoạn khác có thể bắt đầu nối tiếp ngay sau đó. Địa chỉ của ô nhớ nằm ở đầu đoạn được ghi trong một thanh ghi đoạn 16 bit, địa chỉ này gọi là *địa chỉ cơ sở*. Mười sáu bit này tương ứng với các đường dây địa chỉ từ A4 đến A20. Như vậy giá trị vật lý của địa chỉ đoạn là giá trị trong thanh ghi đoạn dịch sang trái 4 vị trí. Điều này tương đương với phép nhân với  $2^4 = 16$ . Địa chỉ của các ô nhớ khác nằm trong đoạn tính được bằng cách cộng thêm vào địa chỉ cơ sở một giá trị gọi là địa chỉ lệch hay độ lệch (offset), gọi như thế vì nó ứng với khoảng lệch của toạ độ một ô nhớ cụ thể nào đó so với ô đầu đoạn. Độ lệch này được xác định bởi các thanh ghi 16 bit khác đóng vai trò thanh ghi lệch (offset register). Nguyên tắc này dẫn đến công thức tính địa chỉ vật lý (physical address) từ địa chỉ đoạn (segment) trong thanh ghi đoạn và địa chỉ lệch (offset) trong thanh ghi con trỏ như sau:

$$\text{Địa chỉ vật lý} = \text{Thanh ghi đoạn} \times 16 + \text{Thanh ghi lệch}$$

Việc dùng hai thanh ghi để nhớ thông tin về địa chỉ thực chất tạo ra một loại địa chỉ gọi là địa chỉ logic và được ký hiệu như sau:

*Thanh ghi đoạn: Thanh ghi lệch hay segment:offset.*

Địa chỉ kiểu *segment: offset* là logic vì nó tồn tại dưới dạng giá trị của các thanh ghi cụ thể bên trong CPU và khi cần thiết truy cập bộ nhớ ra ngoài thì nó phải đổi ra địa chỉ vật lý để rồi đưa lên bus địa chỉ. Việc chuyển đổi này do một bộ tạo địa chỉ thực hiện.

Vi xử lý 16 bit có 4 thanh ghi như sau:

- *CS (code segment)* là thanh ghi đoạn mã 16 bit. Thanh ghi này phối hợp với con trỏ lệnh IP để ghi địa chỉ mã lệnh trong bộ nhớ. Địa chỉ đầy đủ là CS:IP.

- *DS (data segment)* là thanh ghi đoạn 16 bit cho một đoạn dữ liệu. Thanh ghi này phối hợp với hai thanh ghi chỉ số SI và DI để đánh địa chỉ cho dữ liệu. Địa chỉ đầy đủ cho dữ liệu cần đọc vào là DS:SI, cho dữ liệu cần ghi ra là DS:DI.

- *SS (stack segment)* là thanh ghi đoạn 16 bit cho một ngăn xếp. Địa chỉ đỉnh của ngăn xếp được biểu diễn cùng với con trỏ ngăn xếp SP là SS:SP.

- *ES (extra segment)* là thanh ghi dữ liệu phụ có chiều dài 16 bit. Thường được dùng để đánh địa chỉ một chuỗi. ES:DI là địa chỉ chuỗi cần viết đến (chuỗi đích) và DS:SI là địa chỉ chuỗi đọc vào (chuỗi nguồn).

### 3.2.3. Các thanh ghi con trỏ và chỉ số

Các thanh ghi con trỏ và chỉ số có thể được dùng như một thanh ghi đa năng 16 bit. Vì mạch 8088 có tất cả ba thanh ghi con trỏ là (IP, BP, SP) và hai thanh ghi chỉ số (SI, DI). Nhiệm vụ của từng thanh ghi như sau:

- *IP (instruction pointer)* là con trỏ chỉ tới lệnh máy tiếp theo. Lệnh này nằm trong bộ nhớ mà địa chỉ đoạn được ghi trong CS. Như vậy địa chỉ của mã k=lệnh này là CS:IP.

- *BP (base pointer)* là con trỏ cơ sở trỏ về dữ liệu bộ nhớ mà địa chỉ đoạn được ghi trong SS. Địa chỉ đầy đủ sẽ là SS:BP.

- *SP (stack pointer)* là con trỏ ngăn xếp luôn trỏ vào đỉnh ngăn xếp mà địa chỉ đoạn được ghi trong SS. Địa chỉ đầy đủ của dữ liệu là DS:SP.

- *SI (source index)* là chỉ số nguồn, trỏ vào dữ liệu mà địa chỉ đoạn được ghi trong DS. Địa chỉ đầy đủ của dữ liệu là DS:SI.

- *DI (destination index)* là chỉ số đích, cũng trỏ vào đoạn dữ liệu mà địa chỉ đoạn ghi trong DS. Địa chỉ đầy đủ của đoạn dữ liệu là DS:SI.



### 3.2.4. Thanh ghi cờ FR (flag register)

Đây là thanh ghi khá đặc biệt trong CPU, dùng để ghi trạng thái kết quả các phép xử lý trong đơn vị số học và logic ALU hoặc một trạng thái hoạt động của EU. Dựa vào các cờ này người lập trình có thể có các lệnh thích hợp tiếp theo cho bộ vi xử lý (các lệnh nhảy có điều kiện). Thanh ghi này là một thanh ghi 16 bit trong 8088/8086. Nhưng chỉ có 9 bit trong thanh ghi được định nghĩa và sử dụng [2], đó là:

x	x	x	x	O	D	I	T	S	Z	x	A	x	P	x	C
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

x: bit không được định nghĩa.

**Hình 3.2. Sơ đồ thanh ghi cờ của bộ vi xử lý 8086/8088**

- Bit 0: *CF* (carry flag) cờ nhớ,  $CF=1$  khi có nhớ hoặc mượn từ MSB.
- Bit 2: *PF* (parity flag) cờ parity, PF phản ánh tính chẵn (parity) của tổng số bit 1 có trong kết quả. Cờ  $PF=1$  khi tổng số bit 1 trong kết quả là chẵn (even parity, parity chẵn).
- Bit 4: *AF* (auxiliary carry flag) cờ nhớ phụ dùng cho các phép tính với mã BCD.  $AF=1$  khi có nhớ hoặc mượn từ một số BCD thấp (4 bit thấp) sang một số BCD cao (4 bit cao).
- Bit 6: *ZF* (zero flag) cờ rỗng,  $ZF=1$  khi kết quả bằng 0.
- Bit 7: *SF* (sign flag) cờ dấu,  $SF=1$  khi kết quả âm.
- Bit 8: *TF* (trap flag) cờ bẫy,  $TF=1$  khi vi xử lý ở trong chế độ chạy từng lệnh (chế độ này dùng khi cần tìm lỗi trong một chương trình).
- Bit 9: *IF* (interrupt enable flag) cờ cho phép ngắt,  $IF=1$  cho phép các yêu cầu ngắt che được (maskable interrupt) được tác động.
- Bit A: *DF* (direction flag) cờ hướng.  $DF=1$  khi CPU làm việc với chuỗi ký tự theo thứ tự từ phải sang trái (lùi).
- Bit B: *OF* (overflow) cờ tràn,  $OF=1$  khi kết quả vượt ra ngoài giới hạn, xảy ra đối với phép tính có dấu.

### 3.3. ĐƠN VỊ SỐ HỌC VÀ LOGIC

Đơn vị số học và logic (Arithmetic and Logic Unit - ALU) thực hiện các phép toán số học và logic trên các dữ liệu cụ thể. ALU bao gồm các thiết bị thực hiện các phép tính số học và các phép tính logic. Thông thường các phép tính cơ bản được thực

hiện ở đây như: cộng, trừ (các số nhị phân) hai toán hạng; các phép toán logic như AND, OR, NOT hai toán hạng; các phép toán đảo ngược, dịch các bit ... Mặt khác chức năng của ALU còn bao gồm cả việc quyết định các trình tự thao tác đối với hệ thống, nó hình thành và quản lý toàn bộ các tín hiệu điều khiển để sắp xếp hợp lý các phép toán và dòng dữ liệu bên trong như bên ngoài ALU, nó điều khiển dòng dữ liệu của BUS địa chỉ, BUS dữ liệu và biên dịch các tín hiệu điều khiển trên BUS điều khiển của hệ thống.

Để đi vào các phép toán số học trong ALU chúng ta nghiên cứu và xây dựng các thuật toán để thực hiện các phép tính này.

### Số bù:

Trong hệ thống số thông thường, để biểu diễn số âm ta chỉ cần thêm dấu “-” vào các chữ số. Tuy nhiên, trong hệ thống máy tính, ta không thể biểu diễn được như trên. Phương pháp thông dụng là bit có ý nghĩa lớn nhất MSB làm bit dấu. Nếu MSB = 1 sẽ là số âm còn MSB = 0 là số dương. Khi đó các bit còn lại sẽ biểu diễn độ lớn của nó. Như vậy, nếu ta dùng 8 bit để biểu diễn thì sẽ thu được 256 tổ hợp ứng dụng với các giá trị 0 ÷ 255 (số không dấu) hay -127... -0 +0...+127 (số có dấu).

Để thuận tiện hơn trong việc tính toán số có dấu, ta dùng một dạng biểu diễn đặc biệt là số bù hai. Số bù hai của một số nhị phân xác định bằng cách lấy đảo của các bit rồi cộng thêm 1.

#### Ví dụ 3.3.1:

Số 7 biểu diễn là:  $0000\ 0111_2$  có MSB = 0 (biểu diễn số dương).

Số bù 2 là:  $1111\ 1000_2 + 1_2 = 1111\ 1001_2$

Số đại diện cho số -7 là:  $1111\ 1001_2$  có MSB = 1 (biểu diễn số âm).

Như ta đã biết đảo của đảo của một số sẽ là chính nó:

$$-7 = 1111\ 1001_2$$

Đảo bit:  $0000\ 0110_2 + 1_2 = 0000\ 0111_2 = 7_{10}$

Ta thấy, để thực hiện việc xác định số bù 2 của một số A cần phải:

Biểu diễn số A theo mã bù 2 của nó.

Đảo các bit (tìm số bù 1 của A).

Cộng thêm 1 vào để nhận được số bù 2.

Khi biểu diễn theo số bù 2, nếu sử dụng 8 bit ta sẽ có các giá trị số thay đổi từ -128...127.

**Bảng 3.1. Biểu diễn các số theo hệ 2, hệ 2 có dấu và mã bù 2**

Số 8 bit hệ hai	Số thập lục phân tương đương	Số hệ mười theo mã hệ nhị cơ dấu	Số hệ mười theo mã bù hai
0000 0000	0	+0	+0
0000 0001	1	+1	+1
0000 0010	2	+2	+2
....	....	....	....
0111 1101	125	+125	+125
0111 1110	126	+126	+126
0111 1111	127	+127	+127
1000 0000	128	-0	-128
1000 0001	129	-1	-127
1000 0010	130	-2	-126
....	....	....	....
1111 1101	223	-125	-3
1111 1110	224	-126	-2
1111 1111	225	-127	-1

### 3.3.1. Các phép toán cộng trừ

#### 3.3.1.1. Phép toán cộng

Phép cộng các số hệ 2 thực hiện giống như đối với các số hệ mười. Quy tắc cộng như sau:  $Y = A + B$

A	B	Y	C (Carry – Nhớ)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

**Ví dụ 3.3.2:**

Nhớ **Protected by PDF Anti-Copy Free**  
Số hạng 1 1111 1110  
(Upgrade to Pro Version to Remove the Watermark)

Số hạng 2 0101 1111

Tổng 1100 0010



Nếu kết quả của phép toán thì chúng ta có được một số dương trong ký hiệu thập phân bình thường. Còn nếu kết quả là một số âm thì chúng ta có được một số âm ở dạng mã bù hai. Chú ý rằng, trong một vài ví dụ có một vài bit nhớ vượt quá giới hạn của từ thì nó sẽ bị lờ đi. Ở phép cộng bất kỳ thì kết quả có thể lớn hơn độ lớn của từ được sử dụng. Trạng thái này được gọi là tràn trên (overflow). Nếu hai số được cộng có hiện tượng tràn trên thì kết quả là ngược dấu. Khi có hiện tượng tràn trên thì ALU phải báo sự việc này ra ngoài để không sử dụng kết quả này. Chú ý rằng tràn trên có thể xuất hiện có nhớ hoặc là không [1,3,7].

**3.3.1.2. Phép toán trừ**

Phép trừ các số hệ 2 được thực hiện theo quy tắc sau:  $Y = A - B$

A	B	Y	B (Borrow – Mượn)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

**Ví dụ 3.3.3:**

Mượn 0110 0000

Số trừ 0110 1101

Số bị trừ 0011 0001

Hiệu 0011 1100

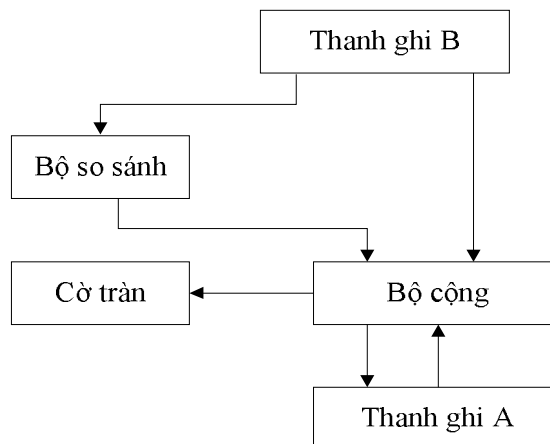
Để trừ một số (số bị trừ) cho một số khác (số trừ) có thể thực hiện bằng cách tạo số bù hai của số trừ và cộng với số bị trừ để được kết quả. Vì vậy, phép trừ được thực hiện thực chất là nhờ phép cộng.

Ví dụ 3.3.4:

**Protected by PDF Anti-Copy Free**  
(Upgrade to Pro Version to Remove the Watermark)

$  \begin{aligned}  M &= 2_{10} = 0010_2 \\  S &= 7_{10} = 0111_2 \\  S' &= 1001_2 \\  M + S' &= 0010_2 \\  &+ 1001_2 \\  \hline  &1011_2 = -5_{10} \\  \\  M &= 7 = 0111_2 \\  S &= -7 = 1001_2 \\  S' &= 0111_2 \\  M + S' &= 0111_2 \\  &+ 0111_2 \\  \hline  &1110_2 = \text{Tràn trên} \\  \\  S' &= \text{Số bù 2.}  \end{aligned}  $	$  \begin{aligned}  M &= 3_{10} = 0011_2 \\  S &= -3_{10} = 1101_2 \\  S' &= 1110_2 \\  M + S' &= 1011_2 \\  &+ 1110_2 \\  \hline  &1001_2 = -7_2 \\  \\  M &= -6_{10} = 1010_2 \\  S &= 4_{10} = 0100_2 \\  S' &= 1100_2 \\  M + S' &= 1010_2 \\  &+ 1100_2 \\  \hline  &0110_2 = \text{Tràn trên}  \end{aligned}  $
---	---

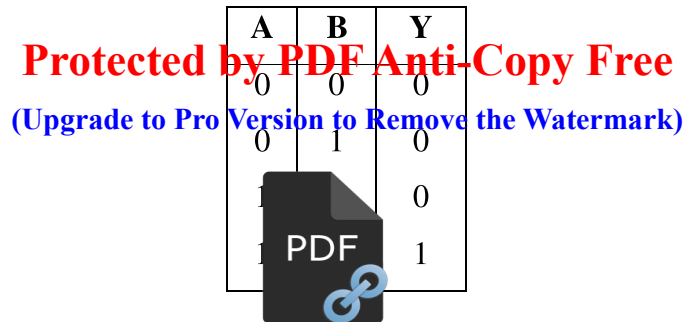
Hình dưới đưa ra đường đi của dữ liệu và yếu tố phần cứng để thực hiện phép cộng và phép trừ. Phần tử chính là một bộ cộng nhị phân, nó nhận hai số vào và đưa ra là tổng và tín hiệu báo nếu tràn. Bộ cộng nhị phân coi hai số như là các số nguyên không dấu. Đối với phép cộng thì hai số được đưa ra để cộng là từ hai thanh ghi, như trên hình vẽ là thanh ghi A và B. Kết quả được lưu vào một trong các thanh ghi này chứ không phải là thanh ghi thứ ba. Tín hiệu báo tràn được lưu vào trong 1 bit cờ tràn (nếu không tràn thì bit này bằng 0, nếu tràn thì bằng 1). Đối với phép trừ, số bị trừ (thanh ghi B) được gửi đến một bộ tạo mã bù hai và gửi tiếp đến bộ cộng [1,3,7].



Hình 3.3. Sơ đồ khối phần cứng của bộ cộng và trừ

### 3.3.2. Phép nhân số nguyên không dấu

Phép nhân trong hệ 2 được thực hiện theo quy tắc sau:  $Y = A \times B$



**Ví dụ 3.3.5:**

$$\begin{array}{rcl}
 \text{Số bị nhân} & 1\ 0\ 0\ 1 & =\ 9d \\
 \text{Số nhân} & 0\ 1\ 1\ 0 & =\ 6d \\
 \hline
 & 0\ 0\ 0\ 0 & \\
 & 1\ 0\ 0\ 1 & \\
 & 1\ 0\ 0\ 1 & \\
 & 0\ 0\ 0\ 0 & \\
 \hline
 & 0\ 1\ 1\ 0\ 1\ 1\ 0 & =\ 54d
 \end{array}$$

Theo quy tắc trên ta thấy, phép nhân hệ hai của các số nguyên nhị phân không đầu có thể thực hiện theo thuật toán cộng và dịch như sau:

- Thành phần đầu tiên của tổng tích lũy thu được là tích của số LSB trong số bị nhân với số nhân. Nếu LSB = 0 thì thành phần này cũng bằng 0, còn nếu LSB = 1 thì thành phần này chính bằng số nhân.

- Mỗi thành phần thứ  $i$  tiếp theo của tổng tích lũy sẽ được tính bằng cách tương tự nhưng phải dịch trái  $i$  bit (có thể bỏ qua các thành phần bằng 0)

- Tổng của các thành phần là tích cần tìm.

- Kết quả phép nhân hai số nguyên nhị phân  $n$  bit có độ dài  $2n$  bit.

Tuy nhiên, chúng ta có thể thực hiện phép nhân một cách có hiệu quả hơn. Đầu tiên, ta thực hiện cộng liên tiếp các tích từng phần cho đến khi kết thúc. Điều này sẽ làm triệt tiêu sự lưu lại của tất cả các tích thành phần, dẫn đến việc sử dụng các thanh ghi ít hơn để lưu giữ các kết quả. Mặt khác, chúng ta có thể loại ra các phân tử phát sinh của tích thành phần. Đối với bit 1 trong số bị nhân thì thực hiện phép cộng và một phép dịch, nhưng đối với bit 0, thì chỉ có phép dịch là được thực hiện.

### 3.3.3. Phép nhân số nguyên có dấu

Chúng ta biết rằng phép cộng và phép trừ được số nguyên không dấu được tính như ví dụ sau:

$$1001_2$$

**Protected by PDF Anti-Copy Free**

(Upgrade to Pro Version to Remove the Watermark)

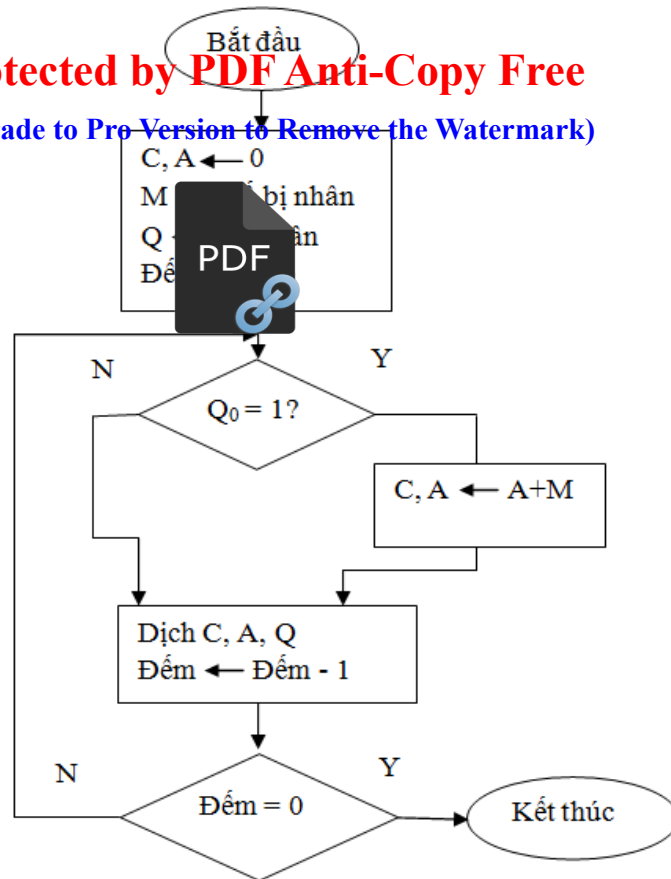
$$1100_2 = 12_{10}$$

Nếu các số trên được xem là các số nguyên bù 2, chúng ta thấy phép cộng  $1001 (-7_{10})$  với  $0011 (3_{10})$  sẽ được  $1100 (-4_{10})$ . Từ đó ta nhận thấy rằng các phép tính đơn giản này sẽ không áp dụng được cho phép nhân. Để thấy rõ điều này ta lấy ví dụ nhân  $1011 (11_{10})$  với  $1101 (13_{10})$  và được  $10001111 (143_{10})$ . Nếu xem các số này như các số bù hai thì ta có  $1011 (-5_{10})$  nhân với  $1101 (-3_{10})$  sẽ được kết quả là  $1000 1111 (-113_{10})$ .

Quá trình thực hiện phép nhân này được diễn giải như sau:

Số nhân và số bị nhân được tải vào hai thanh ghi (Q và M), còn thanh ghi thứ ba là thanh ghi A, ban đầu nó được thiết lập ở mức 0. Thanh ghi 1 bit C được khởi đầu bằng 0 được dùng để lưu bit nhớ của phép cộng. Các bit của số nhân được đọc lần lượt một cách logic. Nếu  $Q_0$  là 1 thì số bị nhân được cộng với thanh ghi A và kết quả được lưu trong thanh ghi A. Sau đó tất cả các bit của các thanh ghi C, A và Q được dịch sang phải một bit, các bit C dịch tới  $A_{n-1}$ ,  $A_0$  tới  $Q_{n-1}$  và  $Q_0$  thì thôi.

Nếu  $Q_0$  là 0 thì phép cộng không được thực hiện, chỉ thực hiện phép dịch phải. Quá trình này được thực hiện lặp lại với mỗi bit của số nhân. Kết quả của tích  $2n$  bit được chứa trong các thanh ghi A và Q. Chú ý rằng trong chu kỳ thứ hai, khi một bit của số nhân là 0 thì phép cộng không được thực hiện [1,3,7].



**Hình 3.4.** Sơ đồ khối phép nhân hai số nhị phân không dấu

**Ví dụ 3.3.6:** Thực hiện phép tính  $(11)_{10} \times (13)_{10}$

Hướng dẫn:

Chuyển đổi  $M = (11)_{10} = 1011$ ;  $Q = (13)_{10} = 1101$

Các bước thực hiện	C	A	Q
Giá trị khởi tạo	0	0000	1101
<b><u>Chu kỳ 1:</u></b>			
Do $Q_0=1$ Thực hiện phép cộng $A \leftarrow A+M$	0	1011	1101
Dịch phải C, A, Q	0	0101	1110
<b><u>Chu kỳ 2:</u></b>			
Do $Q_0=0$	0	0010	1111
Thực hiện dịch phải C, A, Q			



Các bước thực hiện	C	A	Q
<b>Chu kỳ 3:</b> Do $Q_0=1$ Thực hiện phép cộng $A \leftarrow A+M$ Dịch phải C, A, Q	0	1101	1111
	0	0110	1111
<b>Chu kỳ 4:</b> Do $Q_0=1$ Thực hiện phép cộng $A \leftarrow A+M$ Dịch phải C, A, Q	1	0001	1111
	0	1000	1111
<b>Kết quả:</b>		<b>1000</b>	<b>1111</b>

Ví dụ này chứng minh rõ phép nhân sẽ không thực hiện được nếu cả hai số nhân và số bị nhân là âm. Trong thực tế nó sẽ không thực hiện được nếu một trong hai số nhân hoặc số bị nhân là âm. Ta biết rằng, phép nhân một số nhị phân  $2n$  bit được thực hiện nhờ sự dịch chuyển các số đó tới  $n$  bit bên trái. Vì lý do này ta có thể tạo ra tích từng phần bằng cách thực hiện phép nhân theo cách khác. Tích từng phần sẽ được coi như số  $2n$  bit được tạo ra từ số bị nhân  $n$  bit. Vì vậy, cũng như số nguyên không dấu, số bị nhân 4 bit 1011 sẽ được lưu ở trong 1 từ 8 bit là 0000 1011. Mỗi tích từng phần (khác  $2^0$ ) có được từ số đó đã được dịch sang trái, các bit ở vị trí bên phải sẽ được điền bằng các số 0.

$$\begin{array}{r}
 1011 \\
 \times 1101 \\
 \hline
 00001011 \quad 1011 \times 1 \times 2^0 \\
 00000000 \quad 1011 \times 0 \times 2^1 \\
 00101100 \quad 1011 \times 1 \times 2^2 \\
 01011000 \quad 1011 \times 1 \times 2^3 \\
 \hline
 10001111
 \end{array}$$

Giờ đây chúng ta có thể làm sáng tỏ được rằng phép nhân sẽ không làm việc nếu số bị nhân là âm. Vấn đề là mỗi lần số bị nhân là âm thì tích từng phần phải là số âm trên vùng  $2n$  bit. Bit dấu của tích từng phần phải sắp xếp lại. Ta chứng minh điều này dựa vào ví dụ bảng 3.2, biểu diễn phép nhân số 1001 ( $9_{10}$ ) với 0011 ( $3_{10}$ ). Nếu được coi như là số nguyên không dấu thì phép nhân  $9 \times 3 = 27$  được thực hiện đơn giản.

Tuy nhiên nếu 1001 thì được coi như là số bù hai thì giá trị của nó là -7. Sau đó mỗi tích từng phần phải được coi như là số bù hai của các bit  $2n$  (8 bit), được biểu diễn như bảng 3.2. Chú ý rằng phép tính phải được thực hiện bằng cách kéo dài mỗi tích từng phần sang bên trái với số bit nhị phân bằng 1.

Bảng 3.2.

PDF

in số nguyên có dấu

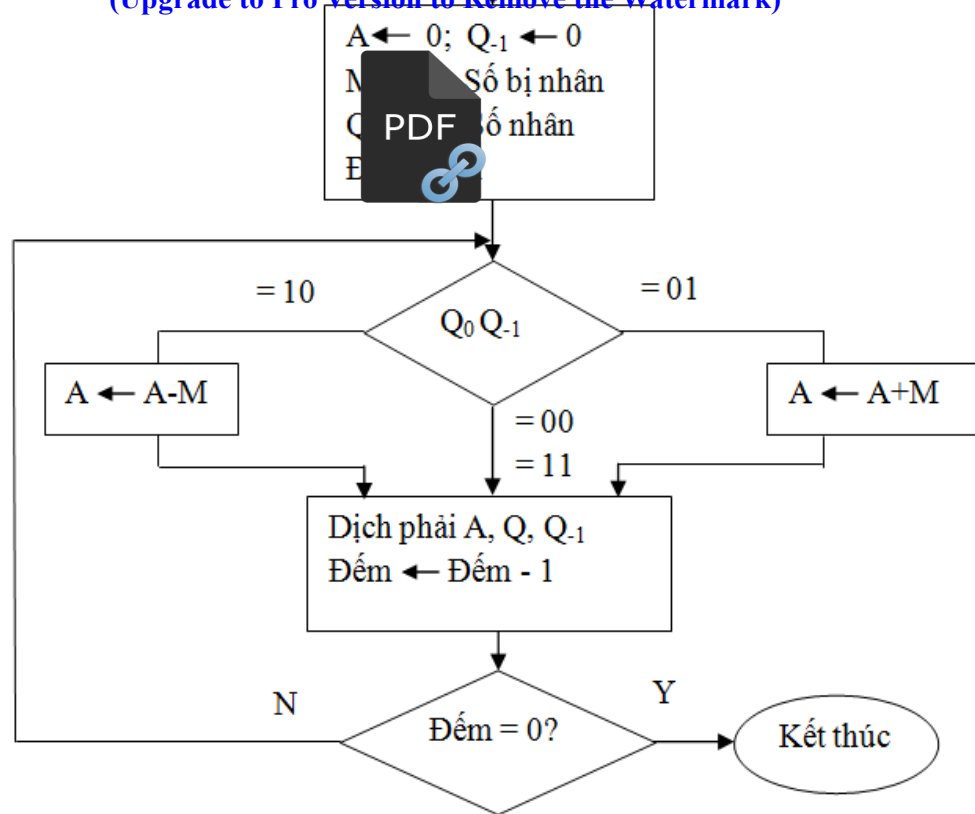
1011 (9)	1001 (-7)
x 0011 (3)	x 0011 (3)
00001001 (1001)x $2^0$	11111001 (-7)x $2^0 = (-7)$
00010010 (1001)x $2^1$	11110010 (-7)x $2^1 = (-14)$
00011011 (27)	11101011 (-21)
Các số nguyên không dấu	Các số nguyên bù 2

Một trong số những thuật toán được sử dụng để thực hiện phép nhân các số nguyên bù hai là thuật toán Booth. Thuật toán này rất có lợi để tăng tốc độ thực hiện các phép nhân, nó có liên quan đến việc tính toán gần đúng.

\*) Thuật toán Booth được miêu tả trong hình 3.5 và được mô tả như sau:

Giống như phần trên, số nhân và số bị nhân được đặt riêng rẽ trong các thanh ghi Q và M. Ở đây thanh ghi 1 bit được đặt một cách logic vào bên phải của bit có nghĩa nhỏ nhất của thanh ghi Q và được định nghĩa là Q-1. Kết quả của phép nhân sẽ xuất hiện trong các thanh ghi A và Q. A và Q được khởi tạo bằng 0. Giống như trước thì các bit của phép nhân sẽ kiểm tra một cách có hệ thống theo thời gian. Bây giờ, mỗi bit đã được kiểm tra và bit của bên phải của nó cũng được kiểm tra.

Nếu hai bit giống nhau (11 hay 00) thì tất cả các bit của thanh ghi A và Q và Q1 được dịch sang bên phải 1 bit. Nếu hai bit này khác nhau thì số bị nhân được cộng hay trừ từ thanh ghi A, tùy theo hai bit là 01 hay 10. Khi thực hiện phép cộng hay phép trừ thì đều được dịch sang phải. Kết quả của tích  $2n$  bit được chứa trong các thanh ghi A và Q. Trong mỗi trường hợp, sự dịch phải là ta thực hiện dịch bit ở phía bên trái nhất của A đó là  $A_{n-1}$ , không những được dịch vào  $A_{n-2}$  mà còn vào  $A_{n-1}$ . Đây là yêu cầu để giữ nguyên dấu của các số trong A và Q. Đây được hiểu như là phép dịch số học, từ đó sẽ giữ nguyên bit dấu.



**Hình 3.5. Thuật toán Booth cho phép nhân số bù hai**

Để rõ hơn ta mô tả phép toán trong các ví dụ sau. Ở các ví dụ này ta có thể thấy nó thực hiện phép nhân với bất kỳ các số dương và số âm.

**Ví dụ 3.3.7:** Thực hiện phép nhân của số  $(-7)_{10}$  với  $(5)_{10}$  theo thuật toán Booth.

*Hướng dẫn:*

Chuyển đổi  $M = (-7)_{10} = 1001$ ;  $Q = (5)_{10} = 0101$

Các bước thực hiện	A	Q	Q <sub>-1</sub>
Giá trị khởi tạo	0000	0101	0
<b><u>Chu kỳ 1:</u></b>			
Do $Q_0 Q_{-1} = 10$ Thực hiện phép gán $A \leftarrow A - M$	0111	0101	0
Dịch phải A, Q, Q <sub>-1</sub>	0011	1010	1

Các bước thực hiện	A	Q	Q <sub>-1</sub>
<b><u>Chu kỳ 2:</u></b> Do $Q_0Q_{-1} = 01$ Thực hiện phép gán $A \leftarrow A+M$ Dịch phải A, Q, Q <sub>-1</sub>	1100 1110	1010 0101	1 0
<b><u>Chu kỳ 3:</u></b> Do $Q_0Q_{-1} = 10$ Thực hiện phép gán $A \leftarrow A-M$ Dịch phải A, Q, Q <sub>-1</sub>	0101 0010	0101 1010	0 1
<b><u>Chu kỳ 4:</u></b> Do $Q_0Q_{-1} = 01$ Thực hiện phép gán $A \leftarrow A+M$ Dịch phải A, Q, Q <sub>-1</sub>	1011 1101	1010 1101	1 0
<b>Kết quả:</b>	<b>1101</b>	<b>1101</b>	

**Ví dụ 3.3.8:** Thực hiện phép nhân của số  $(-5)_{10}$  với  $(-6)_{10}$

*Hướng dẫn:*

Chuyển đổi  $M = (-5)_{10} = 1011$ ;  $Q = (-6)_{10} = 1010$

Các bước thực hiện	A	Q	Q <sub>-1</sub>
Giá trị khởi tạo	0000	1010	0
<b><u>Chu kỳ 1:</u></b> Do $Q_0Q_{-1} = 00$ Dịch phải A, Q, Q <sub>-1</sub>	0000	0101	0
<b><u>Chu kỳ 2:</u></b> Do $Q_0Q_{-1} = 10$ Thực hiện phép gán $A \leftarrow A-M$ Dịch phải A, Q, Q <sub>-1</sub>	0101 0010	0101 1010	0 1
<b><u>Chu kỳ 3:</u></b> Do $Q_0Q_{-1} = 01$ Thực hiện phép gán $A \leftarrow A+M$ Dịch phải A, Q, Q <sub>-1</sub>	1101 1110	1010 1101	1 0
<b><u>Chu kỳ 4:</u></b> Do $Q_0Q_{-1} = 10$ Thực hiện phép gán $A \leftarrow A-M$	0011	1101	0

Dịch phải A, Q, Q <sub>-1</sub>	0001	1110	1
Kết quả: <b>Protected by PDF Anti-CopyFree</b>	0001	1110	

(Upgrade to Pro Version to Remove the Watermark)

### 3.3.4. Phép chia số nguyên không dấu

Phép chia là một phép tính đơn giản hơn so với các phép tính khác nhưng về cơ bản thì nó vẫn được dựa trên những nguyên tắc chung.

Đầu tiên là việc xây dựng thuật toán và các phép dịch các bit, phép cộng hoặc trừ. Phép chia có thể được thực hiện bằng các phép trừ và phép dịch liên tiếp cho đến khi không còn gì để trừ hoặc số bị trừ nhỏ hơn số chia.

**Ví dụ 3.3.9:**  $215 / 22 = 9$  dư 17. Thực hiện phép chia này trong hệ 2.

Số bị chia 215	=	1 1 0 1 0 1 1 1	
Số chia 22	=	1 0 1 1 0	
Số bù 1 của số chia	=	0 1 0 0 1	
Số bù 2 của số chia	=	0 1 0 1 0	
Số bị chia		1 1 0 1 0 1 1 1	1 0 1 1 0
Số bù 2		0 1 0 1 0	1 0 0 1
Tổng nhỏ hơn số chia		0 0 1 0 0 1	
Hạ hàng		0 0 1 0 0 1 1	
Hạ hàng		0 0 1 0 0 1 1 1	
Số bù 2		0 0 1 0 1 0 1 0	
Số dư		0 0 0 1 0 0 0 1	

Ví dụ trên được diễn giải chi tiết như sau: Đầu tiên kiểm tra từng bit của số bị chia từ trái qua phải cho đến khi nào nhóm các bit đó biểu diễn một số lớn hơn hoặc bằng số chia, có nghĩa là số bị chia có thể chia được cho số chia. Nếu còn chưa chia được cho số chia thì ta thêm các số 0 vào thương số từ trái sang phải. Khi chia được cho số chia thì ta thêm 1 vào thương số và lấy số đó trừ đi số chia. Kết quả của phép trừ được gọi là số dư thành phần (partial remainder). Từ đây, phép chia lặp lại các bước như trên. Mỗi lần lặp như vậy ta bổ sung một bit từ số bị chia vào số dư thành phần cho đến khi nào nó lớn hơn số bị chia. Giống như trên, ta lại trừ đi số chia để được một số dư thành phần mới. Quá trình cứ lặp đi lặp lại như trên cho đến khi tất cả các bit của số bị chia đã hết. Tuy nhiên, phép chia thực hiện theo cách trên rất khó khăn cho các

VXL vốn chỉ gồm các phần tử để thực hiện phép cộng và dịch. Để khắc phục các khó khăn này người ta dùng thuật toán số bù 2 [3, 7].

- Đổi số chia ra số bù 2 của nó.
- Lấy số bị chia cộng với số bù 2 của số chia (trừ đi số chia).

Nếu kết quả này có bit dấu bằng 0 (nghĩa là phần này của số bị chia chia được cho số chia) thì bit tương ứng của thương bằng 1.

Nếu kết quả này có bit dấu bằng 1 (nghĩa là phần này của số bị chia không chia được cho số chia) thì bit tương ứng của thương bằng 0 và buộc phải khôi phục lại giá trị ban đầu của số bị chia bằng cách cộng kết quả với số chia ở mã hệ 2.

- Dịch trái kết quả thu được ở trên và làm lại bước 2 cho đến khi nhận được kết quả là 0 (chia hết) hoặc nhỏ hơn số chia (chia còn dư).

Bây giờ ta thực hiện lại ví dụ trên theo thuật toán này.

Số bị chia 215 = 0 1 1 0 1 0 1 1 1

Số chia 22 = 0 1 0 1 1 0

Số bù 2 của số chia = 1 0 1 0 1 0

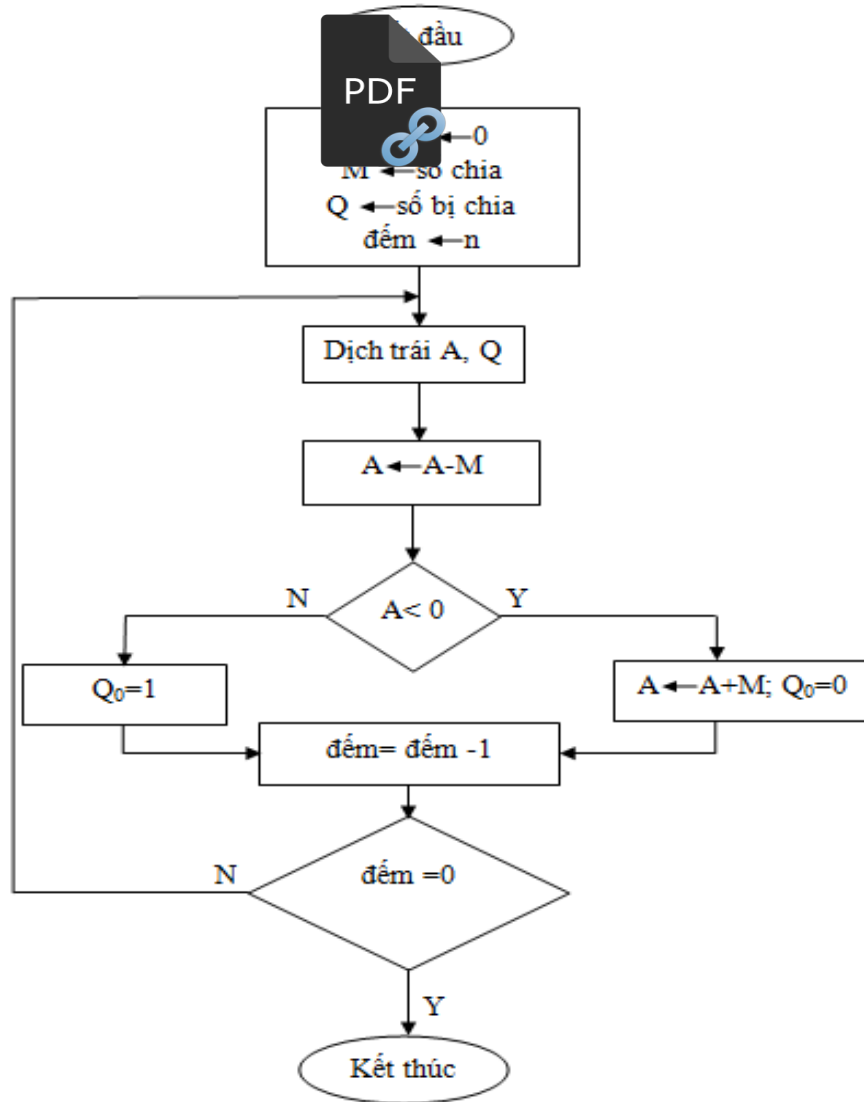
Các bước thực hiện như sau:

Số bị chia	0 1101 0111	
Số chia ở mã bù 2 để cộng	1 0101 0	
Tổng 1	0 0010 0111	Bit dấu = 0 → thương = 1
Dịch trái tổng 1	0 0100 111	
Số chia ở mã bù 2 để cộng	1 0101 0	
Tổng 2	1 1001 111	Bit dấu = 1 → thương = 0
Số chia ở mã hệ 2 để cộng	0 1011 0	
Số bị chia	0 0100 111	
Dịch trái số bị chia	0 1001 11	
Số chia ở mã bù 2 để cộng	1 0101 0	
Tổng 3	1 1110 11	Bit dấu = 1 → thương = 0
Số chia ở mã hệ 2 để cộng	0 1011 0	
Số bị chia	0 1001 11	
Dịch trái số bị chia	1 0011 1	
Số chia ở mã bù 2 để cộng	1 0101 0	
Tổng 4	0 1000 1	Bit dấu = 0 → thương = 1

Bit dấu

Ta cũng có kết quả tương tự. Hình 3.6 là thuật toán tương ứng để thực hiện phép chia dài. Số chia được đặt trong thanh ghi M, số bị chia thì đặt trong thanh ghi Q. Ở mỗi bước, cùng một lúc thanh ghi A và Q đều được dịch trái một bit. M được trừ A để xác định là có chia phần số dư A hay không. Nếu có thì  $Q_0$  nhận được một bit 1,

nếu không thì  $Q_0$  nhận được một bit 0 và M phải được cộng ngược trở lại A để trở lại giá trị trước. Bộ đếm sẽ giảm xuống 1 và tiếp tục thực hiện cho đến bước thứ n. Để kết thúc thì thương số phải nằm trong thanh ghi Q và phần dư phải nằm trong thanh ghi A.



**Hình 3.6. Lưu đồ thuật toán phép chia số nhị phân không dấu**

**Ví dụ 3.3.10:** Thực hiện phép tính  $(9)_{10} : (3)_{10}$

Hướng dẫn:

Đổi ra mã nhị phân:  $Q = (9)_{10} = 1001$ ;  $M = (3)_{10} = 0011$

Các bước thực hiện	A	Q	M
Giá trị khởi tạo	0000	1001	0011

Các bước thực hiện	A	Q	M
<b>Chu kỳ 1:</b> Dịch trái A, Q Gán $A \leftarrow A-M$ $A < 0$ , thực hiện phép phục hồi $A \leftarrow A+M$ và gán $Q_0=0$	0001 1110 0001	0010 0010 0010	
<b>Chu kỳ 2:</b> Dịch trái A, Q Gán $A \leftarrow A-M$ $A < 0$ , thực hiện phép phục hồi $A \leftarrow A+M$ và gán $Q_0=0$	0010 1111 0010	0100 0100 0100	
<b>Chu kỳ 3:</b> Dịch trái A, Q Gán $A \leftarrow A-M$ $A > 0$ , thực hiện phép gán $Q_0=1$	0100 0001 0001	1000 1000 1001	
<b>Chu kỳ 4:</b> Dịch trái A, Q Gán $A \leftarrow A-M$ $A = 0$ , thực hiện phép gán $Q_0=1$	0011 0000 0000	0010 0010 0011	
<b>Kết quả:</b>	<b>0000</b> (Số dư)	<b>0011</b> (Thương số)	

### 3.3.5. Phép chia số nguyên có dấu

Phép chia như trên có thể thực hiện được, nhưng gặp một số khó khăn khi mở rộng tới các số âm. Vì vậy, khi số bị chia hoặc số chia là một số âm thì thuật toán có thể được tổng quát thành một số bước như sau [1,3,7]:

- Bước 1. Chuyển đổi số bị chia và số chia về thành số dương tương ứng.
- Bước 2. Sử dụng thuật giải chia số nguyên không dấu để chia hai số dương, kết quả nhận được là thương Q và phần dư A đều là dương.
- Bước 3. Hiệu chỉnh dấu của kết quả như sau:

(Lưu ý: phép đảo dấu thực chất là thực hiện phép lấy bù hai)



Số bị chia	Số chia	Thương	Số dư
Dương	Dương	Giữ nguyên	Giữ nguyên
Dương	Âm	Đảo dấu	Giữ nguyên
Âm	Dương	Đảo dấu	Đảo dấu
Âm	Âm	Giữ nguyên	Đảo dấu

**Ví dụ 3.3.11:** Thực hiện phép chia  $(-10)_{10} : (4)_{10}$ .

Hướng dẫn:

Bước 1: Chuyển đổi số bị chia và số chia thành số dương tương ứng:

$Q = (10)_{10} = 1010$ ;  $M = (4)_{10} = 0100$

Bước 2: Thực hiện phép chia số nguyên không dấu

Các bước thực hiện	A	Q	M
Giá trị khởi tạo	0000	1010	0100
<b><u>Chu kỳ 1:</u></b>			
Dịch trái A, Q	0001	0100	
Gán $A \leftarrow A - M$	1101	0100	
$A < 0$ , thực hiện phép phục hồi $A \leftarrow A + M$ và gán $Q_0 = 0$	0001	0100	
<b><u>Chu kỳ 2:</u></b>			
Dịch trái A, Q	0010	1000	
Gán $A \leftarrow A - M$	1110	1000	
$A < 0$ , thực hiện phép phục hồi $A \leftarrow A + M$ và gán $Q_0 = 0$	0010	1000	
<b><u>Chu kỳ 3:</u></b>			
Dịch trái A, Q	0101	0000	
Gán $A \leftarrow A - M$	0001	0000	
$A > 0$ , thực hiện phép gán $Q_0 = 1$	0001	0001	
<b><u>Chu kỳ 4:</u></b>			

Các bước thực hiện	A	Q	M
Dịch trái A ← 1 bit	0010	0010	
Gán A ← A + M	1110	0010	
A < 0, thực hiện phép phục hồi A ← A + M và gán Q <sub>0</sub> = 0	0010	0010	
<b>Kết quả:</b>	<b>0010</b> <b>Số dư</b>	<b>0010</b> <b>Thương số</b>	

Bước 3: Hiệu chỉnh dấu kết quả ta được:

Thương số = Q = 1110; Số dư = A = 1110

**Ví dụ 3.3.12:** Thực hiện phép chia:

a)  $(10)_{10} : (-4)_{10}$

b)  $(-10)_{10} : (-4)_{10}$

Dựa vào phép chia ở ví dụ trên ta chỉ cần thực hiện hiệu chỉnh dấu kết quả và thu được kết quả như sau:

a) Thương số = Q = 1110; Số dư = A = 0010

b) Thương số = Q = 0010; Số dư = A = 1110

### 3.3.6. Phép toán với số dấu phẩy động

Một dấu phẩy động được biểu diễn như sau:

$$A = \pm M_A \cdot 2^{\pm E_A}$$

$$B = \pm M_B \cdot 2^{\pm E_B}$$

Trong đó:

$M_A, M_B$  là phần định trị

$E_A, E_B$  là các số mũ

Với các phép cộng, phép trừ, chúng ta cần phải đảm bảo là chúng có cùng số mũ (thường đưa về số mũ lớn hơn). Điều này có thể dẫn đến yêu cầu phải thay đổi cơ số của một trong hai toán hạng. Phép chia và phép nhân thì đơn giản hơn nhiều.

### 3.3.6.1. Phép cộng và trừ

Trong các phép tính số học với số đã học, phép cộng và trừ phức tạp hơn nhiều so với phép nhân và phép chia. Để đơn giản hóa việc tính toán cho phép cộng và phép trừ có 4 bước cơ bản sau:

Bước 1: Kiểm tra số mũ:

Nếu  $E_A > E_B$  lấy  $E_A$  làm chuẩn và ngược lại



Bước 2: Quy đồng phân định trị

Nếu  $E_A > E_B$  đặt  $M'_B = M_B \cdot 2^{-(E_A - E_B)}$

Nếu  $E_A < E_B$  đặt  $M'_A = M_A \cdot 2^{-(E_B - E_A)}$

Bước 3: Cộng hoặc trừ phân định trị

Kết quả  $= (M_A \pm M'_B) \cdot 2^{E_A}$  nếu  $E_A > E_B$

Kết quả  $= (M'_A \pm M_B) \cdot 2^{E_B}$  nếu  $E_A < E_B$

Bước 4: Chuẩn hóa kết quả và báo tràn nếu có.

56

Chúng ta cần phải đưa các số theo định dạng dấu phẩy động. Để thực hiện được các phép cộng và phép trừ cần phải đưa các số vào các thanh ghi mà ta sẽ sử dụng cho ALU. **Protected by PDF Anti-Copy Free**  
(Upgrade to Pro Version to Remove the Watermark)

Nếu định dạng dấu phẩy động là bit phần định trị được ẩn đi thì bit đó phải ẩn trong quá trình tính toán. Các phần phân định trị được lưu trong các thanh ghi riêng rẽ và sẽ được tổng hợp lại khi cần thiết.

Vấn đề của phép cộng và phép trừ là đồng nhất dấu, nếu chúng ta thực hiện phép trừ thì quá trình được bắt đầu bằng cách thay đổi dấu của số bị trừ. Tiếp theo, nếu một trong hai toán hạng bằng 0 thì toán hạng kia sẽ được lưu như là kết quả.

Bước tiếp theo là thao tác với các số, trong đó hai phần mũ được cân bằng nhau. Để hiểu rõ điều này, ta coi như cộng số thập phân như sau:

$$123 \times 100 + 456 \times 10^{-2}$$

Chúng ta không chỉ cộng phần định trị. Đầu tiên các số phải thiết lập về vị trí tương đương nhau, như là, số 4 ở số thứ hai phải thẳng hàng với số 3 ở số thứ nhất. Với các qui định dưới đây, các số mũ sẽ bằng nhau, các số được cộng sẽ được biểu diễn như sau:

$$123 \times 100 + 456 \times 10^{-2} = 123 \times 100 + 4,56 \times 100$$

Để làm tròn ta di chuyển các số nhỏ hơn về bên phải (tăng số mũ) hoặc di chuyển số lớn hơn về phía trái. Việc này có thể dẫn tới kết quả bị mất các số và kết quả thu được nhỏ hơn số bị làm tròn. Bất kỳ số nào bị mất đi đều có phần định trị tương đối nhỏ. Quá trình làm tròn được lặp đi lặp lại cho đến khi nào bên phải phần định trị của số bị chia có giá trị là 1 và tăng số mũ tương ứng lên đúng bằng số lần dịch chuyển (chú ý rằng với cơ số 16 thì một lần di chuyển một số là phải di chuyển 4 bit). Nếu quá trình này cho kết quả phần định trị là 0 thì số khác được lưu lại như là kết quả. Vì vậy nếu hai số có cùng phần mũ, nhưng khác nhau phần định trị, thì số nhỏ hơn sẽ bị mất.

Tiếp theo, hai phần định trị được cộng với nhau, kèm theo cả việc tính toán phần dấu của chúng. Dấu của chúng có thể khác nhau do đó kết quả có thể bằng 0. Cũng có các số có phần định trị có khả năng bị tràn trên một số hạng. Cũng như phần định trị của kết quả đã bị di chuyển về phía bên phải và phần mũ đã được tăng. Một số

mũ bị tràn có thể xuất hiện như là một kết quả và việc này có thể được lưu lại và kết thúc mọi hoạt động.

**Protected by PDF Anti-Copy Free**

Bước tiếp theo là điều chỉnh kết quả (theo chuẩn IEEE 754) phù hợp là việc di chuyển phần định trị sang trái cho tới khi số cao nhất của phần định trị (1 bit hoặc 4 bit với cơ số 16) là khác 0. Mỗi lần di chuyển tương ứng với một lần giảm số mũ đi 1 và điều này có thể dẫn tới số mũ bị tràn. Tóm lại là kết quả phải được làm tròn và lưu lại.

**Ví dụ 3.3.13:** Cho  $A = 0,101 \cdot 2^{110}$ ;  $B = 0,111 \cdot 2^{101}$

Thực hiện phép tính:

a)  $A+B$

b)  $A-B$

*Hướng dẫn:*

a) Thực hiện phép tính  $A + B$

Bước 1:

Xét:  $E_A = 0110 \Rightarrow E_{A \text{ thuận}} = 0000 \ 0110$

$E_B = 0101 \Rightarrow -E_B = -0101$

$\Rightarrow -E_{B \text{ thuận}} = 1111 \ 0101 \Rightarrow -E_{B \text{ ngược}} = 1111 \ 1010 \Rightarrow -E_{B \text{ bù}} = 1111 \ 1011$

$n = E_A - E_B = E_A + (-E_{B \text{ bù}}) = 0000 \ 0110 + 1111 \ 1011 = 0000 \ 0001$

$\Rightarrow n=1 > 0$  chọn  $E_A$  làm chuẩn

Bước 2: đặt  $M'_B = M_B \cdot 2^{-(E_A - E_B)} = 0,111 \cdot 2^{-1} = 0,0111$

Bước 3: thực hiện phép toán đối với phần định trị

$M_A + M'_B = 0101 \ 0000 + 0011 \ 1000 = 1000 \ 1000$

Kết quả =  $(M_A + M'_B) \cdot 2^{E_A} = 1,0001 \cdot 2^{110}$

Bước 4: Chuẩn hóa kết quả

Kết quả =  $0,10001 \cdot 2^{111}$

b) Thực hiện phép tính  $A - B$

có  $M_A = 0101 \ 000$ ;  $M'_B = 0011 \ 1000$

$M_A - M'_B = 0101 \ 0000 - 0011 \ 1000 = 0001 \ 1000$  (có tràn)

$$\text{Kết quả} = (M_A - M'_B) \cdot 2^{E_A} = 0,0011000 \cdot 2^{110}$$

**Protected by PDF Anti-Copy Free**

Ví dụ 3.3.14: Cho  $A = -0,101 \cdot 2^{111}$ ;  $B = -0,1 \cdot 2^{100}$

(Upgrade to Pro Version to Remove the Watermark)

Thực hiện phép tính:

a)  $A+B$

b)  $A-B$



Hướng dẫn:

a) Thực hiện phép tính  $A + B$

Bước 1:

$$E_A = -101 \Rightarrow E_{A \text{ thuận}} = 1111 \ 0101 \Rightarrow E_{A \text{ ngược}} = 1111 \ 1010 \Rightarrow E_{A \text{ bù}} = 1111 \ 1011$$

$$E_B = -1000 \Rightarrow -E_B = 1000 \Rightarrow -E_{B \text{ thuận}} = -E_{B \text{ bù}} = 0000 \ 1000$$

$$n = E_{A \text{ bù}} + (-E_{B \text{ bù}}) = 1111 \ 1011 + 0000 \ 1000 = 0000 \ 0011$$

$$\Rightarrow n = 3 > 0 \text{ chọn } E_A \text{ làm chuẩn}$$

$$\text{Bước 2: đặt } M'_B = M_B \cdot 2^{-(E_A - E_B)} = -0,1 \cdot 2^{-3} = -0,0001$$

Bước 3: thực hiện phép toán đối với phần định trị

$$\text{có } M'_B = 0000 \ 1000 \Rightarrow M'_{B \text{ ngược}} = 1111 \ 0111 \Rightarrow M'_{B \text{ bù}} = 1111 \ 1000$$

$$M_A = 0101 \ 0000 \Rightarrow M'_{A \text{ ngược}} = 1010 \ 1111 \Rightarrow M_{A \text{ bù}} = 1011 \ 0000$$

$$M'_B + M_A = M'_{B \text{ bù}} + M_{A \text{ bù}} = 1111 \ 1000 + 1011 \ 0000 = 1010 \ 1000$$

$$\text{Kết quả} = (M'_{B \text{ bù}} + M_{A \text{ bù}}) \cdot 2^{E_A} = 1,0101000 \cdot 2^{-101} = 0,10101 \cdot 2^{-100}$$

b) Thực hiện phép tính  $A - B$

$$\text{có } M_A - M'_B = M_{A \text{ bù}} - M'_{B \text{ bù}} = 1011 \ 0000 - 1111 \ 1000 = 1101 \ 1000$$

$$\text{Kết quả} = (M_A - M'_B) \cdot 2^{E_A} = 1,1011000 \cdot 2^{-101} = 0,11011 \cdot 2^{-100}$$

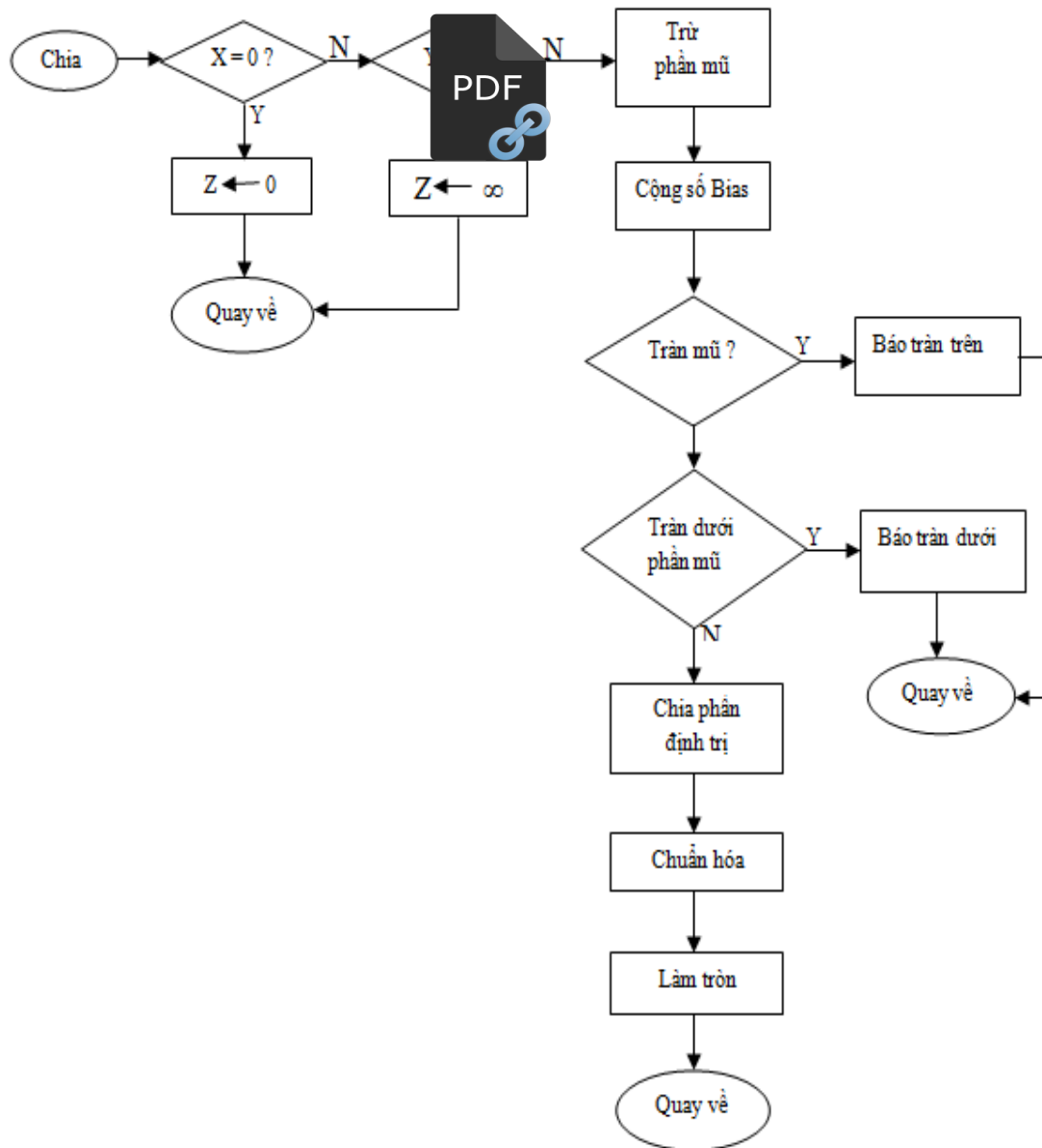
### 3.3.6.2. Phép nhân và chia

Phép nhân và chia dấu phẩy động là một việc đơn giản hơn so với phép cộng và phép trừ.

Các bước tính toán với phép nhân, được minh họa trong hình 3.8.

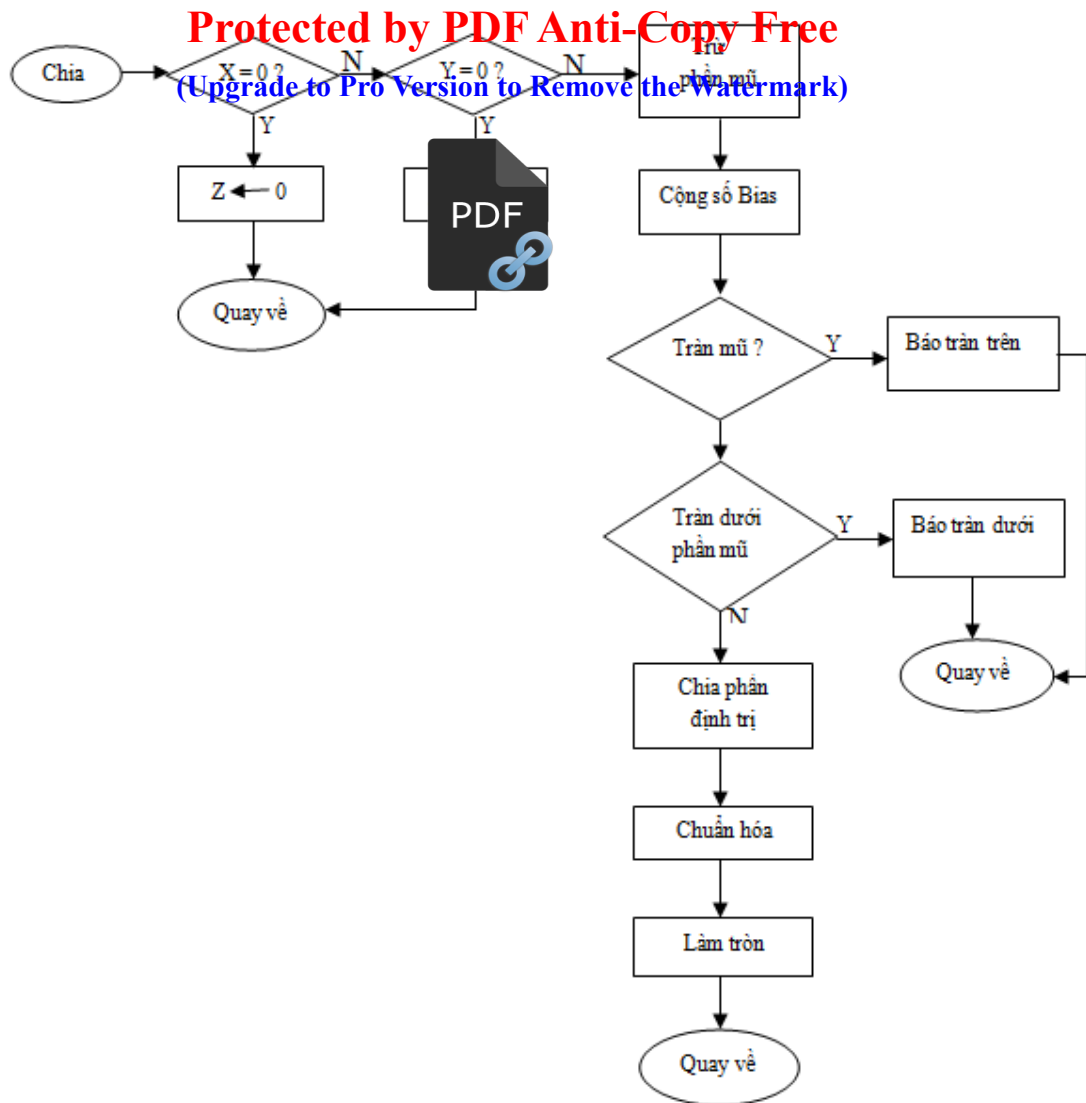
Trước tiên là thực hiện kiểm tra 0, nếu một toán hạng bằng không thì kết quả bằng 0. Bước tiếp theo là cộng số mũ. Nếu số mũ được lưu dưới dạng độ lệch (Bias) thì tổng của số mũ có thể gấp đôi số Bias. Do đó giá trị của độ lệch phải được trừ từ

tổng. Kết quả là có thể từng số mũ bị tràn trên hoặc tràn dưới, kết quả này có thể được lưu lại và kết thúc thuật toán [3,7]. Các bước tính toán với phép chia dấu phẩy động được minh họa trong hình 3.9.



**Hình 3.8. Phép nhân dấu phẩy động**





**Hình 3.9. Phép chia dấu phẩy động**

### 3.3.6.3. Phép làm tròn

Để hiểu rõ hơn về cách làm tròn. Kết quả của bất kỳ phép tính nào thì phần định trị cũng được lưu ở thanh ghi dài hơn. Khi kết quả được đưa ra dưới dạng dấu phẩy động thì các bit thêm vào phải được xấp xếp.

Mỗi một con số đều có phương pháp cho phép làm tròn. Thực ra trong tiêu chuẩn IEEE có 4 phương pháp làm tròn.

- Làm tròn đến số gần nhất: kết quả được làm tròn tới số gần nhất được biểu diễn.
- Làm tròn tới  $+\infty$ : kết quả được làm tròn lên đến dương vô cùng.

- Làm tròn tới  $-\infty$ : kết quả được làm tròn về âm vô cùng.

- Làm tròn tới kết quả được làm tròn tới giá trị gần nhất.

(Upgrade to Pro Version to Remove the Watermark)

Mỗi phương pháp có những đặc điểm riêng. Đối với làm tròn tới số gần nhất là cách làm tròn ngầm định trong chuẩn IEEE và được định nghĩa như sau: là giá trị gần nhất được biểu diễn sẽ là: giá trị được biểu diễn gần nhất là gần bằng nhau, thì bit 0 bit nhỏ nhất của phần thập phân sẽ được gán bằng.

Với ví dụ, nếu các bit thêm vào mà vượt khỏi 23 bit thì có thể được lưu lại, với số là 10010 thì các bit thêm vào lên đến hơn một nửa của các bit đã được biểu diễn với số dương trước đó. Trong trường hợp này câu trả lời đúng là cộng thêm số binary 1 vào bit cuối cùng, tiếp tục làm tròn lên tới số tiếp theo để biểu diễn. Bây giờ thì các bit thêm vào là 01111. Trong trường hợp này, các bit được thêm vào lên đến hơn một nửa của các bit đã được biểu diễn số dương trước đó. Câu trả lời đúng đơn giản là rơi vào bit thêm vào, mà có kết quả của phép làm tròn xuống tới số tiếp theo được biểu diễn.

Trong tiêu chuẩn cũng chỉ ra trường hợp đặc biệt của các bit thêm vào của định dạng 1000. . . ở đây kết quả là chưa hoàn toàn chính xác giữa hai giá trị có thể được biểu diễn. một có thể làm tròn theo phương pháp luôn luôn cắt bỏ. Tuy nhiên có một phương pháp khác đơn giản là đưa ra một số nhỏ nhưng độ lệch tích lũy trong bước tiếp theo của các phép tính. Yêu cầu ở đây là phương pháp không làm tròn độ lệch. Một có thể làm tròn lên hoặc xuống với độ lệch của các số ngẫu nhiên, hoặc trung bình, thì kết quả có thể là không làm tròn độ lệch. Ngược lại việc làm tròn với đối số là không thể dự đoán theo một qui định trước nào. Làm tròn số chẵn theo tiêu chuẩn IEEE là bắt buộc: Nếu kết quả có độ chính xác trung bình giữa hai số có thể được biểu diễn, giá trị được làm tròn lên nếu cuối cùng của số được biểu diễn là 1 và kết quả được làm tròn xuống nếu bit thấp nhất của số được biểu diễn có trạng thái hiện hành là 0.

Tiếp theo là sự lựa chọn thứ hai, việc làm tròn về dương và âm vô cùng, ta biết một phương pháp hoàn toàn có thể thực hiện được như với các phép tính số học. Một số khái niệm về phép tính số học trước đây: Tại bước cuối cùng của hàng loạt phép tính toán dấu phẩy động, chúng ta không biết được kết quả chính xác bởi vì điều đó còn phụ thuộc vào phần cứng, mà tạo ra việc làm tròn kết quả. Nếu chúng ta cho phép mỗi một phép tính được thực hiện hai lần liên tục: lần thứ nhất là làm tròn lên và lần

thứ hai là làm tròn xuống, và kết quả đúng là trung bình cộng của hai kết quả đó. Nếu giới hạn của số làm tròn lên và làm tròn xuống được thu hẹp lại một cách đầy đủ, thì kết quả thu được thật sự chính xác. Nếu không, thì ít nhất chúng ta cũng biết điều đó và có thể cho phép phân tích được phép cộng.

Phương pháp có tính quyết định tiêu chuẩn là làm tròn về 0. Đây chỉ đơn giản là sự cắt bỏ: các bit thêm vào. Đây quả thật là phương pháp đơn giản nhất. Tuy nhiên kết quả mà bị cắt bỏ các bit đi thì thường nhỏ hơn hoặc bằng với giá trị gốc, quá trình này tạo ra một độ lệch xuống (downward). Đây là một độ lệch liên tiếp mà ta đã bàn tới trước kia, từ độ lệch này ảnh hưởng đến mọi tính toán cho mỗi một bit thêm vào là 0.

### 3.4. ĐƠN VỊ ĐIỀU KHIỂN

*Chức năng:*

- Điều khiển nhận lệnh từ bộ nhớ đưa vào thanh ghi lệnh.
- Tăng nội dung của PC để trở sang lệnh kế tiếp.
- Giải mã lệnh đã nhận được và thao tác với lệnh yêu cầu.
- Phát ra các tín hiệu điều khiển thực hiện lệnh.
- Nhận các tín hiệu từ BUS hệ thống và đáp ứng các yêu cầu đó.
- Đơn vị điều khiển tạo ra các thao tác xảy ra trong CPU.

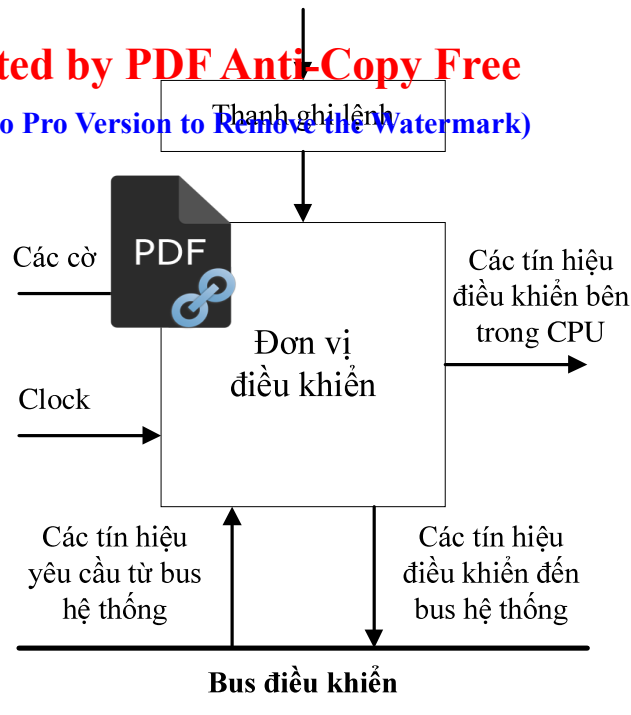
Đơn vị điều khiển thực hiện hai thao tác chính sau:

Sự sắp xếp chuỗi (Sequencing): Đơn vị điều khiển CPU sắp xếp chuỗi vi thao tác vào một chuỗi liên tục thích hợp, dựa trên chương trình đang được thực hiện.

Sự thi hành (Execution): Đơn vị điều khiển mỗi vi thao tác được thực hiện. Đơn vị điều khiển thao tác dựa vào việc sử dụng các tín hiệu điều khiển.

#### 3.4.1. Tín hiệu điều khiển

Đối với đơn vị điều khiển để thực hiện được chức năng của nó, nó phải có dữ liệu và cho phép nó xác định trạng thái của hệ thống và mục ra cho phép nó điều khiển tác động của hệ thống. Nội tại, đơn vị điều khiển phải có logic yêu cầu thực hiện chuỗi vi thao tác và thi hành các chức năng.



**Hình 3.10. Mô hình kết nối đơn vị điều khiển**

Tín hiệu điều khiển được thể hiện trong hình trên bao gồm tín hiệu vào và tín hiệu ra.

#### **3.4.1.1. Các tín hiệu vào**

- *Clock*: Đây là cách đơn vị điều khiển giữ thời gian. Đơn vị điều khiển tạo ra một vi thao tác (hoặc một tập các thao tác đồng thời) được thực hiện với mỗi xung đồng hồ.

- *Thanh ghi chỉ lệnh*: Được dùng để xác định vi thao tác nào được thực hiện trong chu kỳ thi hành.

- *Cờ*: Xác định trạng thái của CPU và kết quả của thao tác ALU.

- *Các tín hiệu điều khiển từ BUS điều khiển*: BUS điều khiển của BUS hệ thống cung cấp tín hiệu cho đơn vị điều khiển, như là tín hiệu ngắt và sự công nhận.

#### **3.4.1.2. Các tín hiệu ra**

- *Tín hiệu điều khiển trong CPU*: Khiến dữ liệu di chuyển từ một thanh ghi tới các thanh ghi khác và làm hoạt động các chức năng ALU cụ thể.

- *Các tín hiệu điều khiển BUS*: Có tín hiệu điều khiển bộ nhớ và tín hiệu điều khiển module vào ra.

### 3.4.2. Đơn vị điều khiển vi chương trình

Để thực hiện một lệnh, đơn vị logic tuần tự đưa ra một lệnh đọc tới bộ nhớ điều khiển. **(Upgrade to Pro Version to Remove the Watermark)**

- Từ mã địa chỉ được xác định trong thanh ghi địa chỉ điều khiển được đọc vào thanh ghi bộ đệm điều khiển.

- Nội dung của thanh ghi điều khiển phát ra tín hiệu điều khiển và thông tin địa chỉ tiếp theo cho đơn vị logic tuần tự.

- Đơn vị logic tuần tự tải địa chỉ mới vào trong thanh ghi địa chỉ điều khiển dựa vào thông tin địa chỉ tiếp theo từ thanh ghi bộ đệm điều khiển và các cờ ALU.

Tất cả xảy ra trong một xung đồng hồ.

## CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 3

**Câu 1:** Trình bày chức năng và cấu trúc của một bộ vi xử lý.

**Câu 2:** Trình bày chức năng và nhiệm vụ các thanh ghi đoạn.

**Câu 3:** Trình bày chức năng và nhiệm vụ các thanh ghi đa năng .

**Câu 4:** Trình bày chức năng và nhiệm vụ các thanh ghi con trỏ và chỉ số.

**Câu 5:** Trình bày chức năng và nhiệm vụ của đơn vị số học và logic (ALU).

**Câu 6:** Trình bày chức năng và nhiệm vụ các thanh ghi cờ FR

**Câu 7:** Nêu các thao tác chính trong đơn vị điều khiển

**Câu 8 :** Thực hiện phép cộng

$$(2)_{10} + (3)_{10}$$

**Câu 9 :** Thực hiện phép trừ theo 2 cách

$$(4)_{10} - (3)_{10}$$

**Câu 10 :** Thực hiện phép nhân

a)  $(12)_{10} \times (7)_{10}$

b)  $(14)_{10} \times (4)_{10}$

**Câu 11 :** Thực hiện phép chia

a)  $(13)_{10} : (6)_{10}$

b)  $(-12)_{10} : (-5)_{10}$

c)  $(10)_{10} : (-3)_{10}$

d)  $(-10)_{10} : (-3)_{10}$

**Câu 12 :** Thực hiện phép tính:  $A \pm B$

Khi biết:  $A = -0,125 \times 2^{-1001}$ ;  $B = -0,101 \times 2^{101}$



## Chương 4

# Protected by PDF Anti-Copy Free

Kiến trúc Tập lệnh

Trang bị (Upgraded to Protected to Remove the Watermark) tạo ra sự vận chuyển tín hiệu bên trong bộ xử lý nhằm thể hiện tập lệnh tương ứng với kiến trúc phần mềm đã đề ra. Mô tả diễn tiến thi hành lệnh mã máy, đây là cơ sở để hiểu được các hoạt động xử lý lệnh trong máy tính.



Sinh viên cần nắm vững cấu trúc của bộ xử lý trung tâm và diễn tiến thi hành một lệnh mã máy, vì đây là cơ sở để hiểu được các hoạt động xử lý lệnh trong các kỹ thuật xử lý thông tin trong máy tính.

### 4.1. CÁC ĐẶC TRƯNG CƠ BẢN CỦA LỆNH MÁY

#### 4.1.1. Giới thiệu chung về tập lệnh

Nếu coi phần mạch điện tử của CPU là “phần xác” thì tập lệnh (Instruction Set) chính là “phần hồn” của bộ não máy tính. Nhờ có tập lệnh, CPU có khả năng lập trình được để thực hiện các công việc hữu ích cho người dùng.

Có thể định nghĩa lệnh máy tính một cách đơn giản: 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. Tập lệnh máy tính thường gồm nhiều lệnh có thể được chia thành một số nhóm theo chức năng: nhóm các lệnh vận chuyển dữ liệu (data movement), nhóm các lệnh tính toán (computational), nhóm các lệnh điều kiện rẽ nhánh conditional and branching) và một số lệnh khác.

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: (1) Đọc lệnh (Instruction fetch - IF): lệnh được đọc từ bộ nhớ về CPU; (2) Giải mã (Instruction decode - ID): CPU giải mã lệnh; (3) Thực hiện lệnh (Instruction execution – EX): CPU thực hiện lệnh; và (4) Lưu kết quả (Write back - WB): kết quả thực hiện lệnh (nếu có) được lưu vào bộ nhớ [2].

Mỗi bộ xử lý đều có một tập lệnh xác định.

Tập lệnh thường có hàng chục tới hàng trăm lệnh.

Mỗi lệnh là một chuỗi số nhị phân mà bộ xử lý hiểu được để thực hiện một thao tác xác định.

Các lệnh được miêu tả bằng các ký hiệu gọi nhớ ( Chính là các lệnh của hợp ngữ).

**Protected by PDF Anti-Copy Free**

#### 4.1.2. Các thành phần của lệnh máy (Upgraded Version to Remove the Watermark)

Dạng tổng quát của lệnh máy tính gồm có 2 phần chính: (1) mã lệnh hay mã thao tác (opcode – operation code) và (2) địa chỉ của các toán hạng (Addresses of Operands). Mỗi lệnh có một mã lệnh và được biểu diễn bằng một số bit. Chẳng hạn, mã lệnh của CPU Intel 8086 được biểu diễn bởi 6 bit. Mỗi lệnh có thể có một hoặc nhiều toán hạng và mỗi toán hạng là một địa chỉ.

Mã thao tác	Địa chỉ các toán hạng
-------------	-----------------------

Mã thao tác (Operation code – opcode ): Mã hóa cho thao tác mà bộ xử lý phải thực hiện.

Địa chỉ toán hạng (Operand): Chỉ ra nơi chứa các toán hạng mà thao tác sẽ tác động.

Toán hạng nguồn: Dữ liệu vào của thao tác.

Toán hạng đích: Dữ liệu ra của thao tác.

Không phải lệnh nào cũng có Địa chỉ các toán hạng.

Một lệnh chỉ cần chỉ ra:

Chức năng của lệnh

Nơi chứa dữ liệu

Nơi lưu kết quả

#### 4.1.3. Mô tả lệnh

Lệnh máy là nhị phân.

Để dễ hiểu và dễ nhớ đối với con người, người ta mô tả lệnh bằng các ký hiệu gọi nhớ. VD: ADD, SUB, LOAD...

Toán hạng có thể được miêu tả như sau: ADD A, B

#### 4.1.4. Các kiểu lệnh

Xử lý dữ liệu

Lưu trữ dữ liệu (Store, Load)

Vận chuyển dữ liệu (Vào/ra – IN, OUT, IOR, IOW)

Lệnh làm việc với thanh ghi, bộ nhớ



Lệnh nhảy (Điều khiển thay đổi trật tự của chương trình): JMP, BR..

Lệnh biến đổi dữ liệu

Protected by PDF Anti-Copy Free

(Upgrade to Pro Version to Remove the Watermark)

#### 4.1.5. Các thao tác khi thực hiện lệnh

IF: Instruction Fetch: Nhận lệnh

ID: Instruction Decode: Giải mã

DF: Data: Nhận dữ liệu

EX: Execution: Thực hiện

DS: Data Store: Lưu trữ kết quả



Thông thường địa chỉ của lệnh tiếp theo  $\rightarrow PC (+1)$

#### 4.1.6. Các vấn đề về thiết kế tập lệnh

- Về thao tác. Chúng ta phải quan tâm tới các vấn đề như: Bao nhiêu thao tác?

Các thao tác như thế nào? Mức độ phức tạp của thao tác.

- Các kiểu dữ liệu.

- Các khuôn dạng lệnh. Độ dài của trường mã thao tác, số lượng địa chỉ toán hạng.

- Các thanh ghi: Quan tâm tới số thanh ghi của CPU được sử dụng, các thao tác nào được sử dụng trên các thanh ghi.

- Các phương pháp địa chỉ (Addressing Modes)

- RISC hay CISC (Reduced Instruction Set Computing, Complex Instruction Set Computing)

### 4.2. CÁC KIỂU TOÁN HẠNG

Kiểu của toán hạng thường được đưa vào trong mã tác vụ của lệnh. Có bốn kiểu toán hạng được dùng trong các hệ thống:

- Kiểu địa chỉ.
- Kiểu dạng số: số nguyên, dấu chấm động,...
- Kiểu dạng chuỗi ký tự: ASCII, EBCDIC,...
- Kiểu dữ liệu logic: các bit, cờ,...

Tuy nhiên một số ít máy tính dùng các nhãn để xác định kiểu toán hạng. Thông thường loại của toán hạng xác định luôn chiều dài của nó. Toán hạng thường có chiều

dài là byte (8 bit), nửa từ máy tính (16 bit), từ máy tính (32 bit), từ đôi máy tính (64 bit). Đặc biệt, kiểu trúc x86 của hãng Intel (New-Intel Packard) có khả năng tính toán với các số thập phân BCD. Một vài bộ xử lý có thể xử lý các chuỗi ký tự.

#### 4.2.1. Số lượng địa chỉ toán hạng trong lệnh

##### - 0 địa chỉ toán hạng:

Các toán hạng đều được ngầm định.

Sử dụng toán hạng Stack.



**Ví dụ 4.2.1:** push a

push b

add

poc c

Có nghĩa là:  $c = a + b$

Nhưng kiểu này không thông dụng.

##### - 1 địa chỉ toán hạng

Một toán hạng được chỉ ra trong lệnh.

Một toán hạng ngầm định, thường là thanh ghi (thanh chứa - accumulator)

Được sử dụng trên các máy của thế hệ trước.

##### - 2 địa chỉ toán hạng

Một toán hạng vừa là toán hạng nguồn vừa là toán hạng đích, toán hạng còn lại là toán hạng nguồn.

$a = a + b$

Giá trị cũ của một toán hạng nguồn bị mất do phải chứa kết quả.

Ưu điểm: Rút gọn độ dài từ lệnh và phổ biến.

##### -3 địa chỉ toán hạng

Hai toán hạng nguồn và một toán hạng đích.

$c = a + b$

Từ lệnh dài vì phải mã hóa địa chỉ cho cả ba toán hạng.

Số toán hạng	Dạng lệnh	Chức năng	Ghi chú
		70	

3	PT A,B,C	$A \leftarrow B \text{ PT } C$	
2	PT A,B	$A \leftarrow A \text{ PT } B$	
1	PT A	$A \leftarrow AC \text{ PT } A$	
0	PT	$(S-2) \text{ PT } (S-1)$	Làm việc với ngăn xếp

Được sử dụng trên các bộ vi xử lý tiên tiến.

### Một số lệnh ví dụ

Không địa chỉ	Một địa chỉ	Hai địa chỉ	Ba địa chỉ
PUSH M	LOAD M	MOV A,B	ADD A,B,C
POP M	STOR M	ADD A,B	SUB A,B,C
ADD	ADD M	SUB A,B	MPY A,B,C
SUB	SUB M	MPY A,B	DIV A,B,C
MPY	MPY M	DIV A,B	
DIV	DIV M		

Chú ý: Trong một lệnh tối thiểu phải có một toán hạng là thanh ghi

**Ví dụ 4.2.2:** Thực hiện phép toán sau.

$$f = (A + B/D) (C - D/E) + E/F$$

Dùng các lệnh 3, 2, 1, 0 địa chỉ

**Với lệnh 3 địa chỉ**

DIV R1,B,D	;	$R1 \leftarrow B/D$
ADD R1,R1,A	;	$R1 \leftarrow A + B/D$

DIV R2,D,E	;	$R2 \leftarrow D/E$
SUB R2,C,R2	;	$R2 \leftarrow C-D/E$
MPY R1,R1,R2	;	$R1 \leftarrow (A+B/D)(C-D/E)$
DIV R2,E,F	;	$R2 \leftarrow E/F$
ADD R1,R2,R1	;	$R1 \leftarrow (A+B/D)(C-D/E)+E/F$

*Với lệnh 2 địa chỉ*

MOV R1,B	;	$R1 \leftarrow B$
DIV R1,D	;	$R1 \leftarrow B/D$
ADD R1,A	;	$R1 \leftarrow A+B/D$
MOV R2,D	;	$R2 \leftarrow D$
DIV R2,E	;	$R2 \leftarrow D/E$
SUB C,R2	;	$C \leftarrow C-D/E$ (ô nhớ chứa C)
MPY R1,C	;	$R1 \leftarrow (A+B/D)(C-D/E)$
MOV R2,E	;	$R2 \leftarrow E$
DIV R2,F	;	$R2 \leftarrow E/F$
ADD R1,R1	;	$R1 \leftarrow (A+B/D)(C-D/E)+E/F$

*Với lệnh 1 địa chỉ*

LOAD B	;	$AC \leftarrow B$
--------	---	-------------------

DIV D	;	$AC \leftarrow B/D$
ADD A	;	$AC \leftarrow A+B/D$
STOR M1	;	$M1 \leftarrow A+B/D$
LOAD D	;	$D \leftarrow D$
DIV E	;	$AC \leftarrow D/E$
STOR M2	;	$M2 \leftarrow D/E$
LOAD C	;	$AC \leftarrow C$
SUB M2	;	$AC \leftarrow C-D/E$
MPY M1	;	$AC \leftarrow (A+B/D)(C-D/E)$
STOR M1	;	$M1 \leftarrow (A+B/D)(C-D/E)$
LOAD E	;	$AC \leftarrow E$
DIV F	;	$AC \leftarrow E/F$
ADD M1	;	$AC \leftarrow (A+B/D)(C-D/E)+E/F$

Với lệnh 0 địa chỉ

PUSH D	;	$S-1 \leftarrow D$
PUSH B	;	$S-2 \leftarrow B$
DIV	;	$S-1 \leftarrow B/D$
PUSH A	;	$S-2 \leftarrow A$
ADD	;	$S-1 \leftarrow A+B/D$
PUSH E	;	$S-2 \leftarrow E$

PUSH D	;	$S - 3 \leftarrow D$
DIV	;	$S - 2 \leftarrow D/E$
PUSH C	;	$S - 3 \leftarrow C$
SUB	;	$(C-D/E)$
MPY	;	$(A+B/D)(C-D/E)$
PUSH F	;	$S - 2 \leftarrow F$
PUSH E	;	$S - 3 \leftarrow F$
DIV	;	$S - 2 \leftarrow E/F$
ADD	;	$S - 1 \leftarrow (A+B/D)(C-D/E)+E/F$

#### 4.2.2. Đánh giá về số địa chỉ toán hạng

Nhiều địa chỉ toán hạng: Các lệnh phức tạp hơn

Cần nhiều thanh ghi hơn.

Chương trình có ít lệnh hơn.

Nhận lệnh và thực hiện lệnh chậm hơn.

Ít địa chỉ toán hạng: Các lệnh đơn giản hơn

Cần ít thanh ghi hơn

Chương trình có nhiều lệnh hơn

Nhận lệnh và thực hiện lệnh nhanh hơn.

### 4.3. TẬP LỆNH

#### 4.3.1. Các lệnh chuyển dữ liệu

MOVE: Copy dữ liệu từ nguồn tới đích.

LOAD: Nạp dữ liệu từ bộ nhớ tới bộ xử lý.

STORE: Cất dữ liệu từ bộ xử lý đến bộ nhớ.

EXCHANGE: Trao đổi nội dung của nguồn và đích.

CLEAR: Chuyển các bit 0 vào toán hạng đích.

SET: Chuyển các bit 1 vào toán hạng đích.

PUSH: Cất nội dung toán hạng nguồn vào ngăn xếp.

POP: Lấy nội dung đỉnh ngăn xếp đưa đến toán hạng đích

#### 4.3.2. Các lệnh số học

ADD: Cộng hai toán hạng.

SUBTRACT: Trừ hai toán hạng.

MULTIPLY: Nhân hai toán hạng.

DIVIDE: Chia hai toán hạng.

ABSOLUTE: Lấy trị tuyệt đối của toán hạng.

NEGATE: Đảo dấu toán hạng (Lấy bù 2 )

INCREMENT: Tăng toán hạng thêm 1

DECREMENT: Giảm toán hạng đi 1

COMPARE: Trừ hai toán hạng để lập cờ

#### 4.3.3. Các lệnh logic

AND: Thực hiện phép AND hai toán hạng

OR: Thực hiện phép OR hai toán hạng

XOR: Thực hiện phép XOR hai toán hạng

NOT: Đảo bit hai toán hạng (lấy bù 1 )

TEST: Thực hiện phép AND hai toán hạng để lập cờ

SHIFT: Dịch trái (phải) toán hạng

ROTATE: Quay trái (phải) toán hạng

Minh họa các lệnh AND, OR, XOR

Giả sử có hai thanh ghi chứa dữ liệu như sau:

$$(R1) = 1010 \ 1010$$

$$(R2) = 0000 \ 1111$$

$$R1 \leftarrow (R1) \text{ AND } (R2) \quad = 0000 \ 1010$$

Phép toán AND dùng để xóa một số bit và giữ nguyên một số bit còn lại của toán hạng.

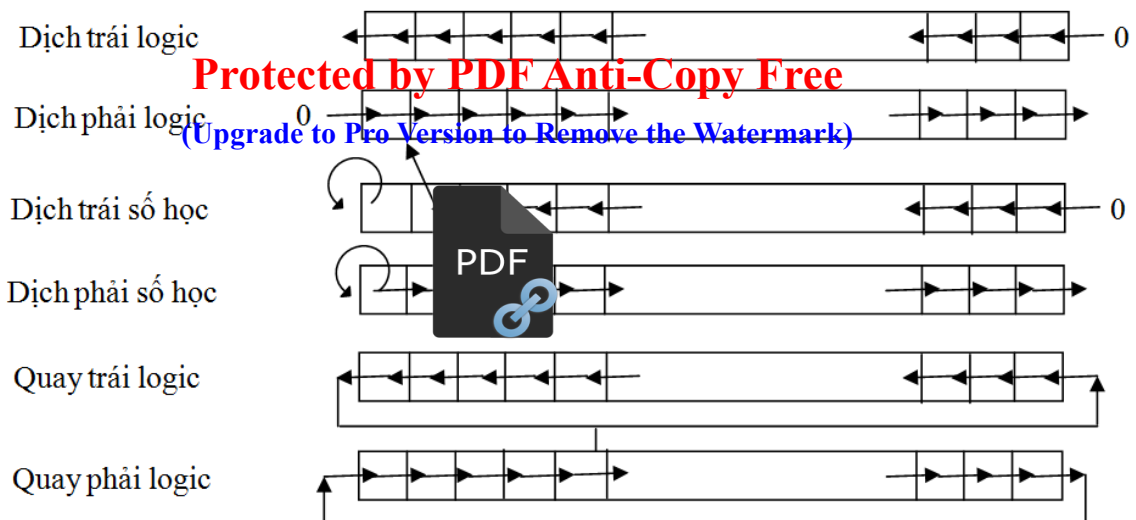
$$R1 \leftarrow (R1) \text{ OR } (R2) \quad = 1010 \ 1111$$

Phép toán OR dùng để thiết lập một số bit và giữ nguyên một số bit của toán hạng.

$$R1 \leftarrow (R1) \text{ XOR } (R2) \quad = 1010 \ 0101$$

Phép toán XOR dùng để đảo một số bit và giữ nguyên một số bit còn lại của toán hạng.

Các thao tác SHIFT và ROTATE:



#### 4.3.4. Các lệnh vào ra chuyên dụng

INPUT: Copy dữ liệu từ một cổng xác định đưa tới đích.

OUTPUT: Copy dữ liệu từ nguồn tới một cổng xác định.

#### 4.3.5. Các lệnh chuyển điều kiện

JUMP (BRANCH): Lệnh nhảy không điều kiện. Nạp vào PC một địa chỉ xác định.

JUMP CONDITIONAL: Lệnh nhảy có điều kiện. Điều kiện đúng -> nạp vào PC một địa chỉ xác định. Điều kiện sai -> Không làm gì cả.

CALL: Lệnh gọi chương trình con. Cất nội dung của PC (địa chỉ trở về) ra một vị trí xác định (thường ở Stack). Nạp vào PC địa chỉ của lệnh đầu tiên của chương trình con.

RETURN: Lệnh trở về từ chương trình con. Khôi phục địa chỉ trở về trả lại cho PC để trở về chương trình chính

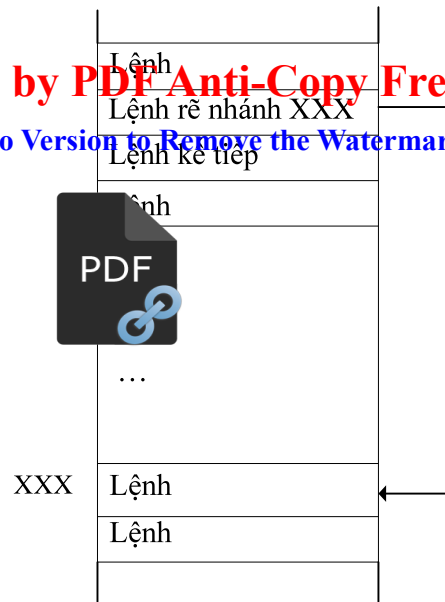
#### 4.3.6. Lệnh rẽ nhánh

- Lệnh rẽ nhánh không điều kiện.

Chuyển tới thực hiện lệnh ở vị trí có địa chỉ XXX. PC không trở sang lệnh kế tiếp mà nhảy xuống lệnh có địa chỉ XXX

PC ← XXX





**Hình 4.1. Lệnh rẽ nhánh không điều kiện**

- Lệnh rẽ nhánh có điều kiện.

Trong lệnh có kèm theo điều kiện

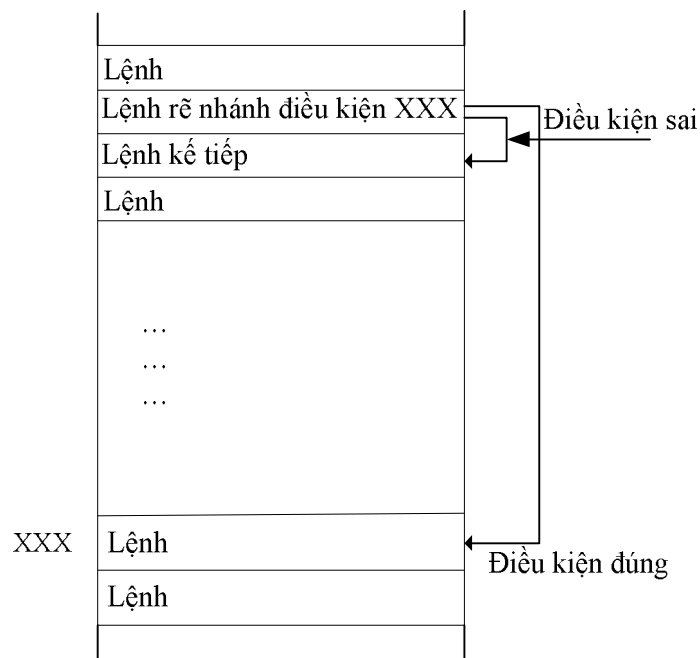
Kiểm tra điều kiện trong lệnh:

Nếu lệnh đúng -> thực hiện lệnh ở vị trí có địa chỉ XXX

Nếu điều kiện sai -> thực hiện lệnh\_kế\_tiếp.

Điều kiện thường được kiểm tra thông qua các cờ.

Có nhiều lệnh rẽ nhánh có điều kiện.



**Hình 4.2. Lệnh rẽ nhánh có điều kiện**

#### 4.3.7. Lệnh CALL và RETURN

- Lệnh gọi chương trình con CALL.

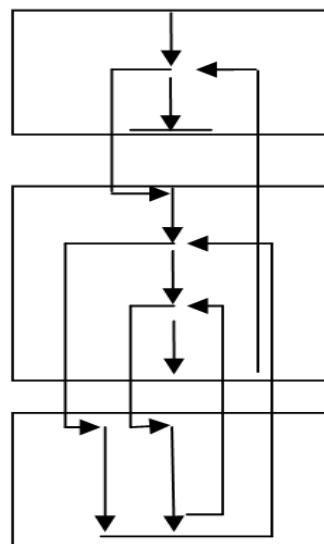
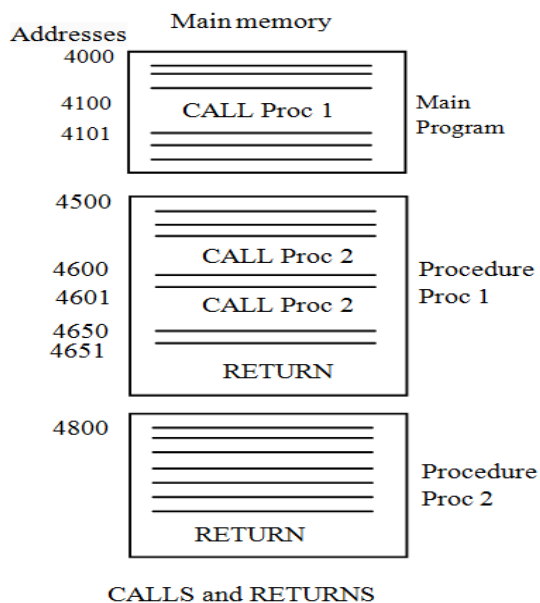
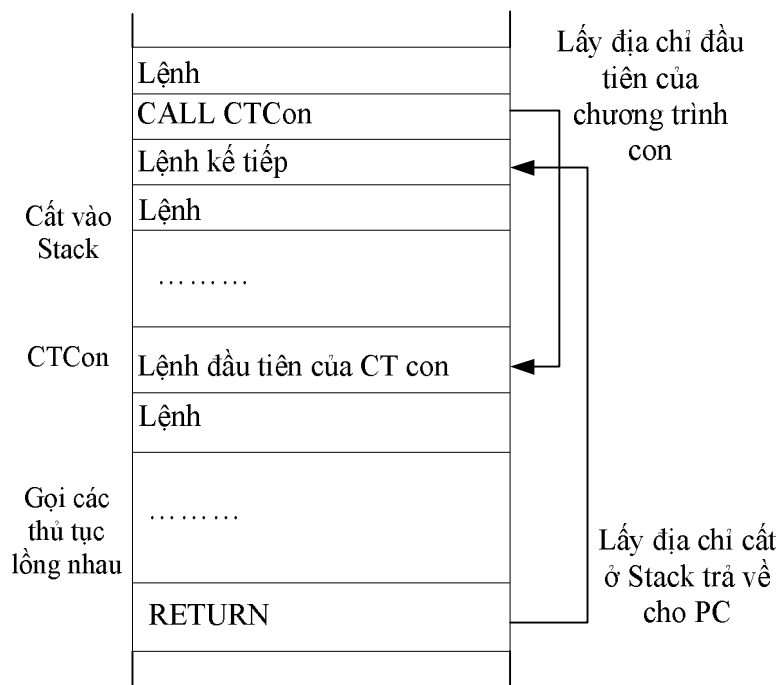
(Upgrade to Pro Version to Remove the Watermark)

Cất nội dung PC (chứa địa chỉ của lệnh kế tiếp) ra Stack.

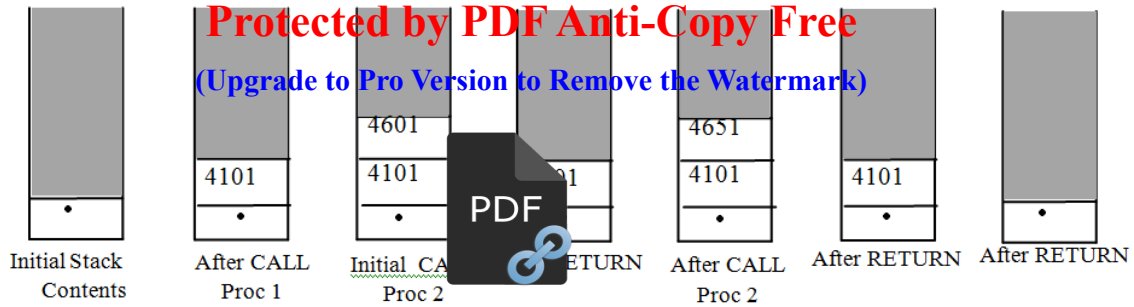
Nạp vào PC địa chỉ của lệnh đầu tiên của chương trình con được gọi -> Bộ xử lý được chuyển sang thực hiện chương trình con tương ứng.

Lệnh trở về chương trình con RETURN.

Lấy địa chỉ của lệnh kế tiếp cất ở Stack nạp trả lại cho PC -> Bộ xử lý được điều khiển quay trở về thực hiện lệnh nằm sau lệnh CALL.



### Sử dụng Stack



#### 4.3.8. Các lệnh điều khiển hệ thống

HALT: Dừng thực hiện chương trình

WAIT: Tạm dừng thực hiện chương trình, lặp kiểm tra điều kiện cho đến khi thỏa mãn thì tiếp tục thực hiện.

NO OPERATION: Không thực hiện gì cả.

LOCK: Cấm không cho xin chuyển nhượng BUS

UNLOCK: Cho phép xin chuyển nhượng BUS.

### 4.4. NGÔN NGỮ LẬP TRÌNH VÀ CHƯƠNG TRÌNH DỊCH

#### 4.4.1. Khái niệm ngôn ngữ lập trình

Ngôn ngữ lập trình (Programming language) là một tập con của ngôn ngữ máy tính. Đây là một dạng ngôn ngữ được chuẩn hóa (đối lập với ngôn ngữ tự nhiên). Nó được dùng để miêu tả những quá trình, những ngữ cảnh một cách rất chi tiết. Nói cách khác, ngôn ngữ lập trình là một hệ thống được ký hiệu hóa để miêu tả những tính toán (qua máy tính) trong một dạng mà cả con người và máy đều có thể đọc và hiểu được.

#### 4.4.2. Các loại ngôn ngữ lập trình thông dụng

Có hàng trăm loại ngôn ngữ lập trình khác nhau, mỗi loại ngôn ngữ đều có cú pháp riêng của nó. Một số ngôn ngữ thì được phát triển để dùng trên các loại máy tính chuyên biệt, một số ngôn ngữ khác thì - do sự thành công của nó - đã trở thành chuẩn và được áp dụng trên đa số các máy tính. Ngôn ngữ lập trình có thể được phân chia thành 3 loại chính: Ngôn ngữ máy, hợp ngữ và ngôn ngữ cấp cao.

##### ➤ Ngôn ngữ máy

Ngôn ngữ máy (mã máy) là ngôn ngữ nền tảng của bộ vi xử lý. Các chương trình được viết trong tất cả các loại ngôn ngữ khác cuối cùng đều được chuyển thành ngôn ngữ máy trước khi chương trình đó được thi hành. Vì tập lệnh của ngôn ngữ máy

phụ thuộc vào loại vi xử lý nên ngôn ngữ máy sẽ khác nhau trên những máy tính có sử dụng bộ vi xử lý khác nhau. Một đặc điểm của việc chương trình hành ngôn ngữ máy là lập trình viên có thể điều khiển máy tính trực tiếp và đạt được chính xác điều mình muốn làm. Do đó, các chương trình ngôn ngữ máy được viết tốt là những chương trình rất hiệu quả (tốc độ thi hành nhanh, kích thước nhỏ). Bất lợi của chương trình ngôn ngữ máy là thông thường sẽ mất rất nhiều thời gian để viết, rất khó đọc, theo dõi để tìm lỗi. Thêm vào đó, bởi vì chương trình được viết bằng tập lệnh phụ thuộc vào bộ vi xử lý nên chương trình chỉ chạy được trên những máy tính có cùng bộ vi xử lý mà thôi. Ngôn ngữ máy cũng được gọi là ngôn ngữ cấp thấp (*low-level language*)

#### ➤ Hợp ngữ

Hợp ngữ được phát triển nhằm giúp các lập trình viên dễ nhớ các chỉ thị của chương trình hơn. Hợp ngữ tương tự như ngôn ngữ máy nhưng lại sử dụng các ký hiệu gợi nhớ (mnemonics hay mã lệnh hình thức - symbolic operation code) để biểu diễn cho các mã lệnh của máy. Một đặc điểm khác nữa là hợp ngữ thông thường cho phép định địa chỉ hình thức (symbolic addressing), nghĩa là một vị trí bộ nhớ trong máy tính có thể được tham chiếu tới thông qua một cái tên hoặc ký hiệu, chẳng hạn như TOTAL thay vì phải sử dụng địa chỉ thực sự của nó (bằng con số nhị phân) trong ngôn ngữ máy. Các chương trình hợp ngữ còn bao gồm các chỉ thị vĩ mô (*macro instruction*) có thể tạo ra nhiều lệnh mã máy. Các chương trình hợp ngữ được chuyển sang mã máy thông qua một chương trình đặc biệt gọi là trình hợp dịch (assembler). Mặc dù hợp ngữ tương đối dễ dùng hơn mã máy nhưng hợp ngữ vẫn được xem là ngôn ngữ cấp thấp bởi vì nó vẫn còn rất gần với từng thiết kế của máy tính.

Nói chung một trình hợp ngữ được chia làm 4 đoạn:

Đoạn mã

Đoạn dữ liệu

Đoạn dữ liệu mở rộng

Đoạn ngăn xếp

Trong chương trình đòi hỏi phải có ít nhất một đoạn ngăn xếp. Đoạn ngăn xếp giúp lưu trữ các kết quả trung gian khi thực hiện chương trình.

#### ➤ Ngôn ngữ cấp cao

Cuộc cách mạng của ngôn ngữ máy tính bắt đầu với sự phát triển của ngôn ngữ cấp cao vào cuối thập kỷ 1950 và 1960. Ngôn ngữ cấp cao gần gũi hơn với ý niệm ngôn ngữ mà hầu hết mọi người đều biết như bao gồm các danh từ, động từ, ký hiệu toán học, liên hệ và các thao tác luận lý. Các yếu tố này có thể được phối hợp, liên kết với nhau tạo thành một hình thức "câu". Các "câu" này được gọi là các mệnh đề của chương trình (program statement). Nhờ những đặc điểm này, các lập trình viên dễ dàng đọc và dễ học ngôn ngữ cấp cao hơn so với ngôn ngữ máy hoặc hợp ngữ. Một lợi điểm quan trọng là ngôn ngữ cấp cao thông thường không phụ thuộc vào máy tính, nghĩa là các chương trình viết bằng ngôn ngữ cấp cao có thể chạy trên các loại máy tính khác nhau (sử dụng các bộ vi xử lý khác nhau).

Ví dụ: Ngôn ngữ cấp cao như: C, C++, PASCAL, BASIC, COBOL, FORTRAN,...

#### 4.4.3. Chương trình dịch

Mọi chương trình được viết bằng các ngôn ngữ không phải là ngôn ngữ máy cuối cùng đều phải được chuyển đổi sang ngôn ngữ máy trước khi được thi hành. Chương trình ngôn ngữ cấp cao được dịch sang ngôn ngữ máy bằng một trong hai cách: Trình biên dịch (compiler) hoặc trình thông dịch (interpreter).

##### ➤ Trình biên dịch

Sẽ chuyển đổi toàn bộ chương trình sang mã máy, rồi chứa kết quả vào đĩa để có thể thi hành về sau. Chương trình ngôn ngữ cấp cao được chuyển đổi được gọi là chương trình nguồn (source program) và chương trình ngôn ngữ máy được tạo ra được gọi là chương trình đối tượng (object program) hoặc mã đối tượng (object code). Khi người dùng muốn chạy chương trình, chương trình đối tượng sẽ được nạp lên bộ nhớ chính của CPU và các chỉ thị của chương trình sẽ được thi hành. Khi được hướng dẫn bởi các chỉ thị của chương trình, CPU sẽ truy xuất dữ liệu và tạo ra các kết quả. Trình biên dịch sẽ kiểm tra cú pháp chương trình, thực hiện các phép kiểm tra logic và đảm bảo các dữ liệu sắp được sử dụng trong các phép so sánh, tính toán đã được định nghĩa một cách hợp lý ở một nơi nào đó trong chương trình. Một chức năng quan trọng của trình biên dịch là nó sẽ tạo ra một danh sách lỗi của tất cả mệnh đề trong chương trình vi phạm cú pháp của ngôn ngữ. Danh sách này giúp lập trình viên dễ dàng sửa đổi chương trình.

Do ngôn ngữ máy phụ thuộc vào bộ vi xử lý nên các máy tính khác nhau sẽ cần có các trình biên dịch khác nhau để chạy cùng một bộ ngôn ngữ cấp cao. Ví dụ, một máy mainframe, máy mini và máy tính cá nhân cần có các trình biên dịch khác nhau để biên dịch cùng một chương trình nguồn sang mã máy của từng loại máy này.

➤ Trình thông dịch

Thay vì chuyển đổi toàn bộ trình nguồn như trình biên dịch, trình thông dịch chỉ chuyển đổi một mệnh đề của chương trình và thực hiện đoạn mã kết quả ngay, sau đó nó tiếp tục chuyển đổi mệnh đề thứ 2 rồi thi hành đoạn mã kết quả thứ 2 và cứ thế. Khi sử dụng trình thông dịch, mỗi lần chạy chương trình là mỗi lần chương trình nguồn được thông dịch sang ngôn ngữ máy. Không có chương trình đối tượng nào được tạo ra.

Các trình thông dịch thường được dùng trên các máy tính cá nhân không có đủ bộ nhớ hoặc sức mạnh tính toán cần thiết để dùng trình biên dịch. Lợi điểm của trình thông dịch là lập trình viên vẫn có thể chạy một chương trình vẫn còn lỗi cú pháp. Chỉ đến lúc thông dịch đến câu lệnh có lỗi cú pháp, quá trình thi hành chương trình mới bị ngừng lại và trình thông dịch sẽ thông báo lỗi. Điểm bất lợi là các chương trình thông dịch chạy không nhanh bằng các chương trình được biên dịch vì quá trình chuyển đổi sang ngôn ngữ máy được thực hiện cùng với quá trình thi hành chương trình. Vì lý do này, ngày nay, đa số các ngôn ngữ cấp cao đều dùng trình biên dịch.

## CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 4

**Câu 1:** Nêu các loại ngôn ngữ lập trình thông dụng

**Câu 2:** Nêu các chương trình dịch và ưu nhược điểm của mỗi loại.

**Câu 3:** Thực hiện phép toán sau.

$$f = E - (A + B/C) / (D + B)$$

Dùng các lệnh 3, 2, 1, 0 địa chỉ

## Chương 5

# Protected by PDF Anti Copy Free

HỆ THỐNG NHỚ

Trang bị (Upgrade) và (Pro Version) và (Remove the Watermark) lý hoạt động của các cấp bộ nhớ máy tính: bộ nhớ cache, nguyên lý vận hành, phân loại các mức, đánh giá hiệu quả hoạt động; và nguyên nhân hành của bộ nhớ ảo.

Sinh viên cần hiểu được các cấp bộ nhớ và cách thức vận hành của các loại bộ nhớ được giới thiệu để có thể đánh giá được hiệu năng hoạt động của các loại bộ nhớ.

## 5.1. TỔNG QUAN VỀ HỆ THỐNG NHỚ

### 5.1.1. Phân loại hệ thống nhớ

❖ Vị trí: Hệ thống nhớ nằm:

Bên trong CPU

Tập các thanh ghi.

Bộ nhớ vi chương trình trong đơn vị điều khiển.

Bộ nhớ trong

Bộ nhớ chính.

Bộ nhớ cache.

Bộ nhớ ngoài

Các thiết bị nhớ ngoài như ổ cứng, đĩa từ, đĩa quang, USB...

❖ Dung lượng

Độ dài từ nhớ: Tính bằng bit, thường là 8, 16, 32, 64 bit.

Số lượng từ nhớ.

❖ Đơn vị trao đổi:

Trao đổi theo từ nhớ: Đơn vị tự nhiên ở tổ chức bộ nhớ. Kích thước từ nhớ thường là số bit dùng để biểu diễn số hoặc độ dài lệnh.

Trao đổi theo khối nhớ: Là đơn vị truyền dữ liệu lớn hơn từ nhớ, thường được dùng truyền dữ liệu với bộ nhớ ngoài.

❖ Phương pháp truy nhập:

Truy nhập tuần tự: Bằng từ.

Truy nhập trực tiếp: Giống như truy nhập tuần tự, truy nhập trực tiếp bao hàm việc chia sẻ đọc và ghi. Những từ nhớ của ta chỉ có địa chỉ cơ sở duy nhất trên vị trí vật lý. Việc truy nhập được hoàn thành bởi truy nhập trực tiếp là đi đến vùng lân cận chung cộng với tìm kiếm tuần tự, đếm hoặc đợi để đi đến vị trí cuối cùng. Thời gian truy nhập có thể thay đổi được tại đĩa sử dụng phương pháp truy nhập trực tiếp.



Truy nhập ngẫu nhiên: Mỗi vị trí địa chỉ trong bộ nhớ là độc nhất. Thời gian truy nhập các vị trí đã cho là độc lập với dãy truy nhập ưu tiên và là hằng số. Như vậy, vị trí nào cũng có thể được chọn ngẫu nhiên, và địa chỉ trực tiếp. Bộ nhớ chính là truy nhập ngẫu nhiên.

Truy nhập liên kết: Đây là kiểu truy nhập ngẫu nhiên có thể làm sự so sánh vị trí bit trong từ cho một phép toán cụ thể và làm việc này cho tất cả các từ đồng thời. Vì vậy một từ được tìm lại được dựa vào chính nội dung của nó thay vì địa chỉ của nó. Với truy nhập ngẫu nhiên thông thường, mỗi vị trí có địa chỉ cơ khí của mình, và thời gian tìm là hằng số độc lập với vị trí hay mẫu hình truy nhập ưu tiên. Bộ nhớ cache dùng cách truy nhập này.

❖ *Hiệu năng:*

Thời gian truy nhập: Đối với truy nhập ngẫu nhiên đó là thời gian để thực hiện hoạt động đọc ghi. Đó là thời gian từ khi địa chỉ đã sẵn sàng trong bộ nhớ đến khi dữ liệu được cất trữ hoặc được làm có thể sử dụng được. Đối với truy nhập không phải là ngẫu nhiên thời gian truy nhập là thời gian đưa vị trí đọc viết cơ khí đến vị trí mong muốn.

Chu kỳ nhớ: Khoảng cách giữa hai lần truy nhập.

Tốc độ truyền: Bao nhiêu byte trong một đơn vị thời gian.

❖ *Kiểu vật lý:*

Bộ nhớ bán dẫn

Bộ nhớ từ

Bộ nhớ quang

❖ *Các đặc tính vật lý:*

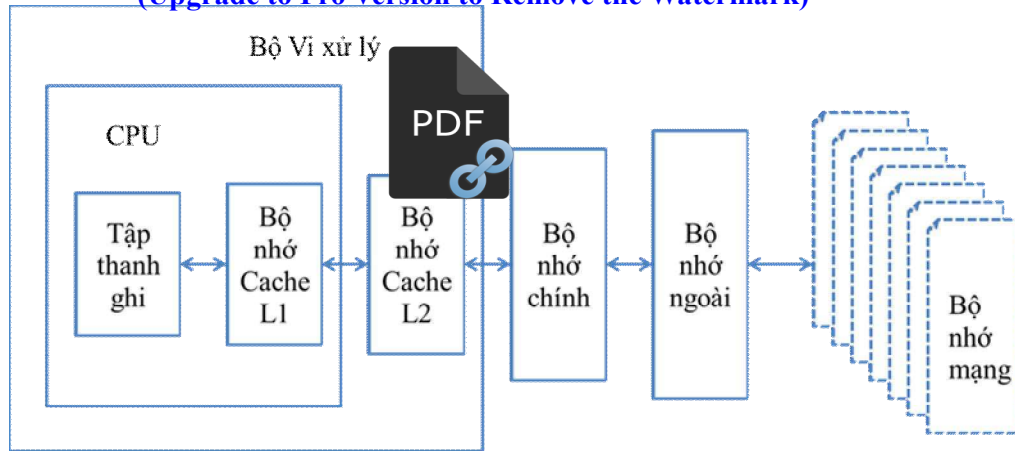
Khả biến / không khả biến



Xóa được / không xóa được

### 5.1.2. Phân cấp hệ thống nhớ

(Upgrade to Pro Version to Remove the Watermark)

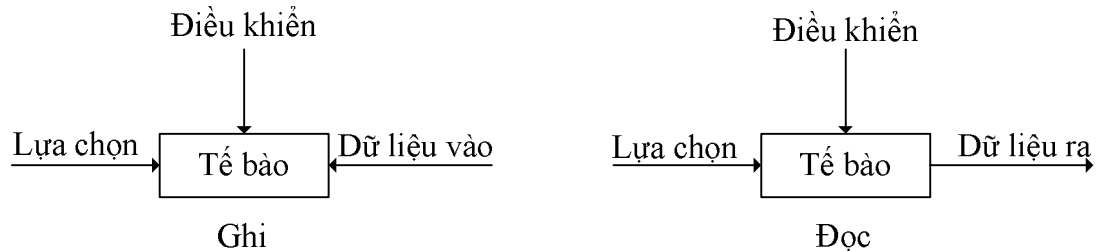


Hình 5.1. Phân cấp hệ thống nhớ

Dung lượng tăng dần, tốc độ giảm dần, giá thành/ 1 bit giảm dần

## 5.2. BỘ NHỚ BÁN DẪN

Hoạt động của một ô nhớ



Hình 5.2. Hoạt động của ô nhớ

### 5.2.1. Phân loại bộ nhớ bán dẫn

Có 3 loại bộ nhớ bán dẫn:

- Bộ nhớ bán dẫn chỉ đọc: (Read Only Memory, ROM)
- Bộ nhớ truy xuất ngẫu nhiên: (Random Access Memory, RAM)

Thực tế ROM và RAM đều là loại bộ nhớ truy xuất ngẫu nhiên, nhưng RAM được giữ tên gọi này. Để phân biệt chính xác ROM và RAM ta có thể gọi ROM là bộ nhớ chết (nonvolatile, vĩnh cửu) và RAM là bộ nhớ sống (volatile, không vĩnh cửu) hoặc nếu coi ROM là bộ nhớ chỉ đọc thì RAM là bộ nhớ đọc được - viết được (Read-Write Memory)

- Thiết bị logic khả trình : (Programmable Logic Devices, PLD) có thể nói điểm khác biệt giữa PLD với ROM và RAM là qui mô tích hợp từ. PLD thường không lớn như ROM và RAM và các tác vụ của PLD thì có phần hạn chế.

Kiểu bộ nhớ	Tiêu chí	Khả năng xóa	Cơ chế ghi	Tính khả biến
Read Only Memory(ROM)	Bộ nhớ đọc	Không xóa được	Mặt nạ	Không khả biến
Programmable ROM(PROM)			Bằng điện	
ERASABLE prom(eprom)	Bộ nhớ hầu như chỉ đọc	Bằng tia cực tím, cả chip.		
Electrically Erasable PROM(EEPROM)		Bằng điện mức từng byte		
Flash memory	Bộ nhớ ghi-đọc	Bằng điện, từng khối		
Random Access Memory(RAM)		Bằng điện mức từng byte		Bằng điện

#### 5.2.1.1. ROM (Read Only Memory)

Mặc dù có tên gọi như thế nhưng chúng ta phải hiểu là khi sử dụng ROM, tác vụ đọc được thực hiện rất nhiều lần so với tác vụ ghi. Thậm chí có loại ROM chỉ ghi một lần khi xuất xưởng.

ROM lưu trữ các thông tin:

Thư viện các chương trình con

Các chương trình điều khiển hệ thống (BIOS)

Các bảng chức năng

Vì chương trình

ROM là Bộ nhớ không khả biến

**Protected by PDF Anti-Copy Free**

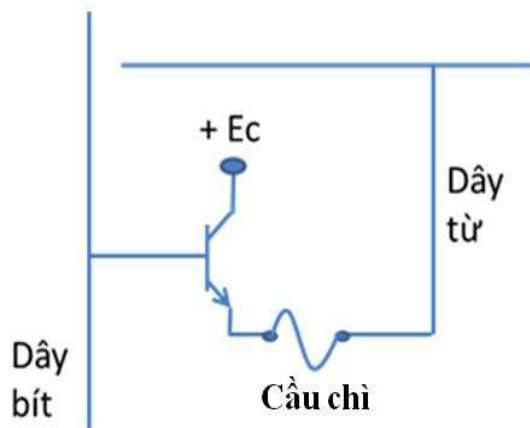
**(Upgrade to Pro Version to Remove the Watermark)**

Đây là loại ROM được chế tạo để thực hiện một công việc cụ thể như các bảng tính, bảng lượng giác, bảng logarithm... sau khi xuất xưởng và có giá thành rất đắt.

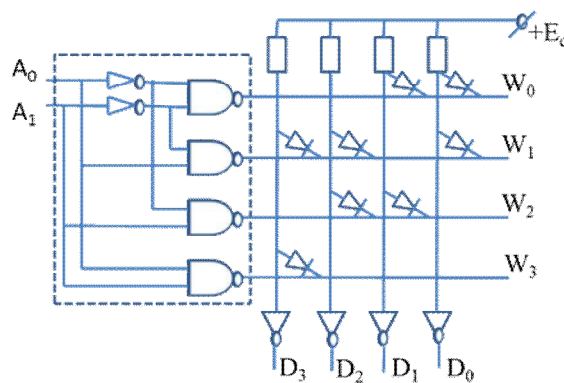
ROM khả trình (Programmable ROM, PROM)

Có cấu tạo giống MROM nhưng mỗi vị trí nhớ đều có linh kiện nối với cầu chì.

Một khi cầu chì đã bị phá vỡ thì không thể nối lại được do đó loại ROM này cho phép lập trình một lần duy nhất để sử dụng, nếu bị lỗi không thể sửa chữa được.



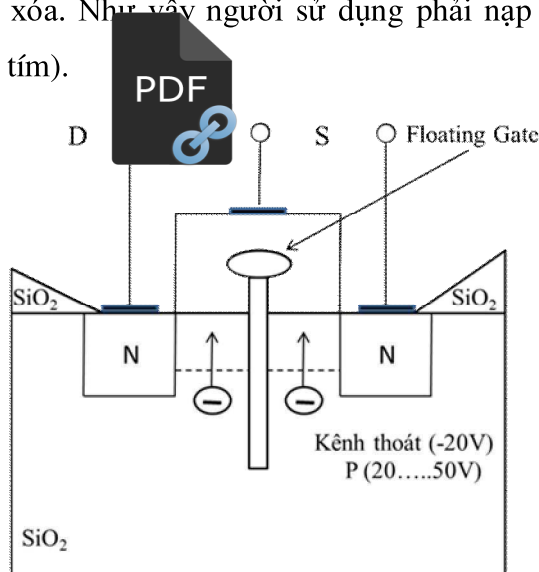
**Hình 5.3. Sơ đồ PROM**



**Hình 5.4. Sơ đồ ROM Diode**

ROM khả trình, xóa được bằng tia U.V. (Ultra Violet Erasable Programmable ROM, U.V. EPROM)

Đây là loại ROM rất tiện cho người sử dụng vì có thể dùng được nhiều lần bằng cách xóa và nạp lại. Tuy nhiên, để ghi lại P.Đ. thì xóa bằng tia cực tím. Điểm bất tiện của U.V EPROM là cần thiết bị xóa đặc biệt phát tia U.V và mỗi lần xóa tất cả tế bào nhớ trong một IC nhớ đều bị xóa. Như vậy người sử dụng phải nạp lại toàn bộ chương trình (UV chính là tia cực tím).



**Hình 5.5. Sơ đồ EPROM**

*ROM khả trình và xóa được bằng xung điện (Electrically Erasable PROM, EEPROM hay Electrically Alterable PROM, EAPROM)*

Đây là loại ROM khả trình và xóa được nhờ xung điện và đặc biệt là có thể xóa để sửa trên từng byte.

#### **FLASH ROM**

EPROM là loại nonvolatile, có tốc độ truy xuất nhanh (khoảng 120ns), mật độ tích hợp cao, giá thành rẻ tuy nhiên để xóa và nạp lại phải dùng thiết bị đặc biệt và lấy ra khỏi mạch.

#### **5.2.1.2. RAM (Random Access Memory)**

Bộ nhớ đọc ghi, khả biến, lưu trữ thông tin tạm thời

Có hai loại RAM: RAM tĩnh và RAM động.

##### **RAM tĩnh.**

Các bit được lưu trữ bằng FF nên thông tin ổn định. Có cấu trúc phức tạp. Dung lượng chíp nhỏ. Tốc độ nhanh. Đắt tiền. Dùng làm bộ nhớ cache

##### **RAM động (Dynamic RAM, DRAM)**

Các bit được lưu trữ trên tụ điện nên cần phải có mạch làm tươi, nó cấu trúc đơn giản, dùng điện áp lớn hơn để đọc thì nó sẽ tự làm tươi, dùng làm bộ nhớ chính.

(Upgrade to Pro Version to Remove the Watermark)

### **Các DRAM tiên tiến**

EDRAM (Enhanced DRAM)

CDRAM (Cache DRAM)



SDRAM (Synchronous DRAM): Được làm việc đồng bộ bởi xung clock

DDR – SDRAM (Double Data Rate SDRAM)

RDRAM (Rambus DRAM)

### **Làm tươi bộ nhớ DRAM**

Bộ nhớ DRAM có các hàng cần phải được làm tươi trong mỗi chu kỳ 2mS. Mạch làm tươi trong chip nhớ phải kiểm tra điện áp các ô nhớ, nếu nó lớn hơn  $V_{cc}/2$  thì nạp nó tới  $V_{cc}$ , nếu bé hơn  $V_{cc}/2$  thì xả hết về 0V.

Để đọc một từ từ BANK nhớ DRAM, trước hết DRAM Controller hoặc một mạch khác cấp tín hiệu  $WE\# = 1$ . Sau đó gửi nửa thấp của địa chỉ, ứng với địa chỉ hàng, rồi tín hiệu  $RAS\# = 0$ . Sau 1 thời gian, controller cấp nửa địa chỉ cao, ứng với địa chỉ cột, rồi tín hiệu  $CAS\# = 0$ . Sau thời gian nhất định, từ cần có sẽ xuất hiện trên Output Data của nhớ.

Để viết vào DRAM, các tín hiệu cũng tương tự, ngoại trừ sau tín hiệu  $CAS\# = 0$ , controller cấp  $WE\# = 0$  để quy định viết vào RAM.

Controller làm tươi DRAM bằng cách gửi ra mỗi địa chỉ trong 512 địa chỉ hàng và cấp  $RAS\# = 0$  theo chu kỳ, khoảng 4mS. Việc làm tươi được tiến hành hoặc theo *burst mode* hoặc theo *distributed mode*. Trong burst mode toàn bộ 512 hàng được định địa chỉ và đánh nhịp lần lượt cách nhau 4mS. Còn ở distributed mode hàng được định địa chỉ và đánh nhịp sau 4/512 mS. Là mạch làm tươi DRAM với controller làm tươi 8208.

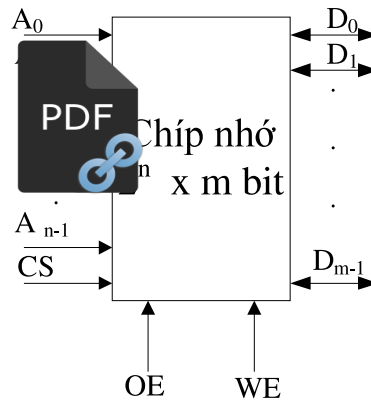
*Những nhiệm vụ chính của việc điều khiển nhớ DRAM của máy tính là:*

- Làm tươi mỗi ô nhớ sau một khoảng thời gian vài mS.
- Cấp hai nửa địa chỉ cùng các tín hiệu  $RAS\#$ ,  $CAS\#$  thích hợp.
- Bảo đảm thao tác đọc/viết và làm tươi không xảy ra đồng thời.

- Cấp tín hiệu đọc/viết để điều khiển chiều số liệu.

### 5.2.2. Tổ chức của chip nhớ

**Protected by PDF Anti-Copy Free**  
(Upgrade to Pro Version to Remove the Watermark)



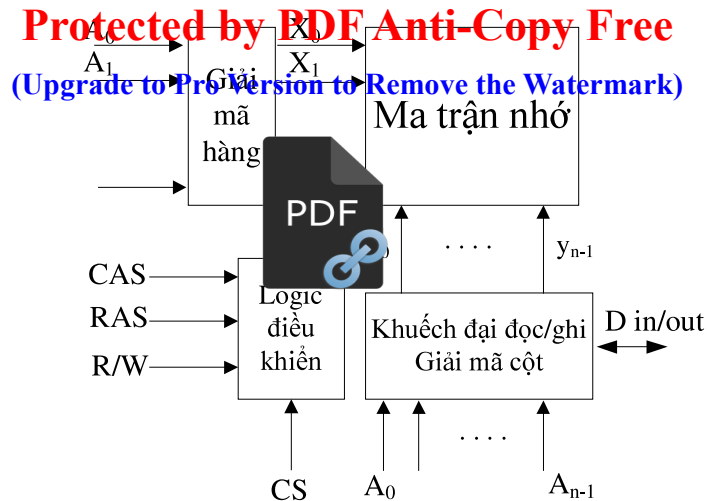
**Hình 5.6. Sơ đồ cơ bản của chip nhớ**

#### *Các tín hiệu của chip nhớ*

- Các đường địa chỉ:  
 $A_{n-1} \div A_0$  có  $n$  từ nhớ
- Các đường dữ liệu:  
 $D_{m-1} \div D_0$  có độ dài từ nhớ bằng  $m$  bit  
Dung lượng chip nhớ =  $2^n \times m$  bit
- Các đường điều khiển:  
Tín hiệu chọn chip CS (Chip select)  
Tín hiệu điều khiển đọc OE (Output Enable)  
Tín hiệu điều khiển ghi WE (Write Enable)

#### *Tổ chức của DRAM*

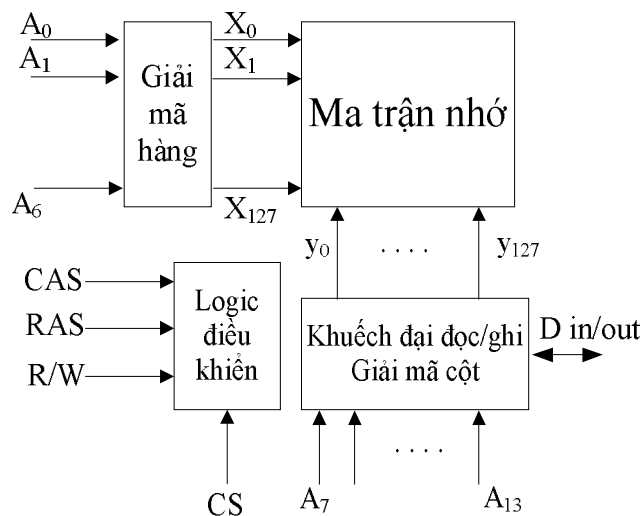
- Dùng  $n$  đường địa chỉ dồn kênh nên cho phép truyền  $2n$  bit địa chỉ
- Tín hiệu chọn địa chỉ hàng RAS (Row Address Select)
- Tín hiệu chọn địa chỉ cột CAS (Column Address Select)
- Dung lượng của DRAM =  $2^{2n} \times m$  bit



**Hình 5.7. Cấu trúc RAM**

**Ví dụ 5.2.1:** Vẽ cấu trúc RAM (16K x 1 bit)

Ta có:  $16k \times 1 \text{ bit} = 2^4 \times 2^{10} = 2^{14}$



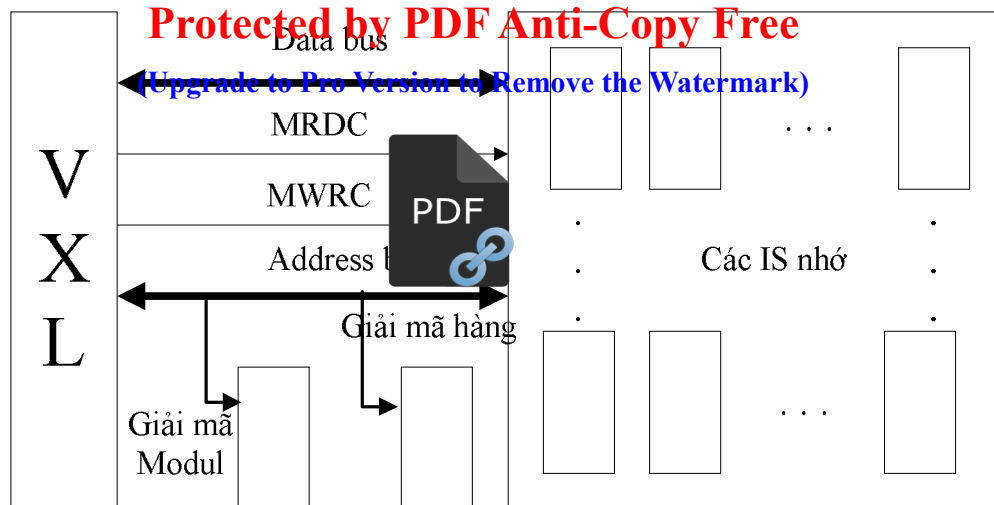
### 5.2.3. Thiết kế mô-đun nhớ bán dẫn

Dung lượng chip nhớ  $2^n \times m$  bit

Chính vì thế cần thiết kế để tăng dung lượng:

- Thiết kế tăng độ dài từ nhớ
- Thiết kế tăng số lượng từ nhớ
- Kết hợp cả 2 cách trên

Cấu trúc của một bộ nhớ



**Hình 5.8. Cấu trúc của bộ nhớ**

### ***Xác định ma trận IC***

- Số IC trên một hàng (Số cột) = Số bit dữ liệu bộ nhớ/ Số bit dữ liệu IC
- Số IC trên một cột (Số hàng) = Số từ nhớ của bộ nhớ/ Số từ nhớ của IC

Các IC trên cùng một hàng nối chung CE

Các IC trên cùng một cột được nối chung với các bit dữ liệu tương ứng

Tất cả các IC của ma trận được nối chung bit địa chỉ

Giải mã chọn hàng:

Đầu ra nối với các tín hiệu CE của từng hàng

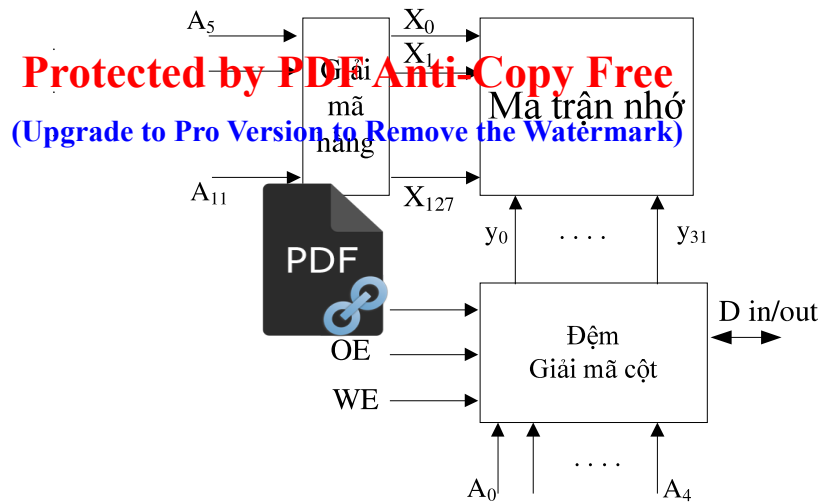
Đầu vào là các bit được còn lại

Giải mã chọn module:

Đầu ra đưa vào CE của IC giải mã

**Ví dụ 5.2.2:** Xây dựng bộ nhớ 64KB cho một máy tính sử dụng CPU 8085 (16 bit địa chỉ, 8 bit dữ liệu) từ các RAM được cấu tạo từ các ma trận 128x128 bit với 4 bit dữ liệu: Địa chỉ bộ nhớ bắt đầu: C000H





$$a_H = \log_2 N = \log_2 128 = 7$$

$$a_C = \log_2 (N/m) = \log_2 (128/4) = 5$$

$$a = a_C + a_H = 7 + 5 = 12$$

Vậy dung lượng 1 IC là  $2^a \times m = 4K \times 4$

$$(128 \times 128) = 2^{14} = 4K \times 4$$

Ma trận IC:

$$\text{Số IC trên một hàng: } 8/4 = 2$$

$$\text{Số IC trên một cột: } 16K/4K = 4$$

$$\text{Số bit dùng cho giải mã: } 16 - 12 = 4 \text{ bit}$$

$$\text{Số bit dùng cho giải mã chọn hàng: } 2$$

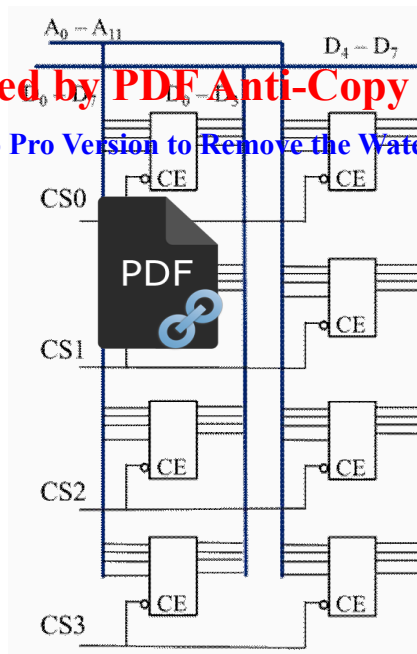
$$\text{Số bit dùng cho giải mã chọn module: } 2$$

$$\text{CS0: C0000H – CFFFFH : } 1 \ 1 \ 0 \ 0$$

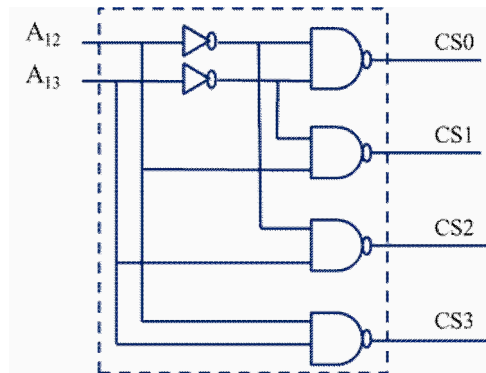
$$\text{CS1: D0000H – DFFFFH : } 1 \ 1 \ 0 \ 1$$

$$\text{CS2: E0000H – EFFFFH : } 1 \ 1 \ 1 \ 0$$

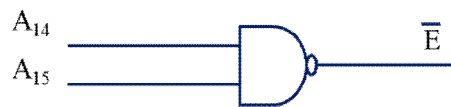
$$\text{CS3: F0000H – FFFFFH : } 1 \ 1 \ 1 \ 1$$



Giải mã chọn hàng



Giải mã chọn module



**Ví dụ 5.2.3:** (Tăng độ dài từ nhớ)

Cho chip nhớ SRAM 4K x 4 bit

Thiết kế module nhớ 4K x 8 bit

*Hướng dẫn:*

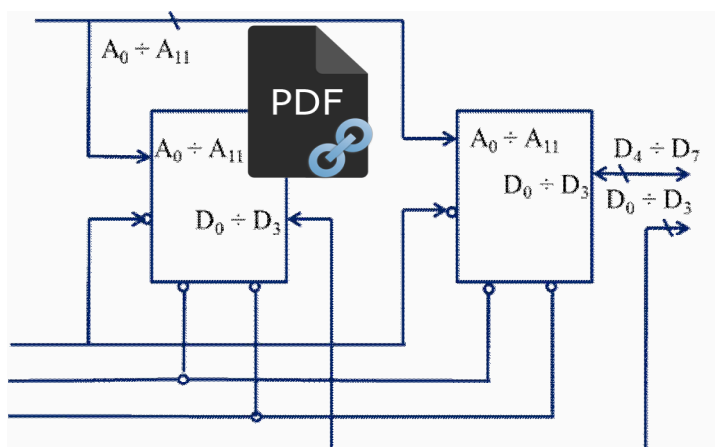
Dung lượng chip nhớ 4K x 4 bit =  $2^{12}$  x 4 bit. Chip nhớ có: 12 chân địa chỉ và 4 chân dữ liệu.

Module nhớ 4K x 8 bit =  $2^{12}$  x 8 bit. Module nhớ có: 12 chân địa chỉ và 8 chân dữ liệu.

Số IC trên một hàng:  $8/4 = 2$

Số IC trên một cột:  $4K/4K = 1$

(Upgrade to Pro Version to Remove the Watermark)



**Ví dụ 5.2.4:** (Tăng số lượng từ nhớ )

Cho chip nhớ SRAM 4K x 8 bit

Thiết kế module nhớ 8K x 8bit

*Hướng dẫn:*

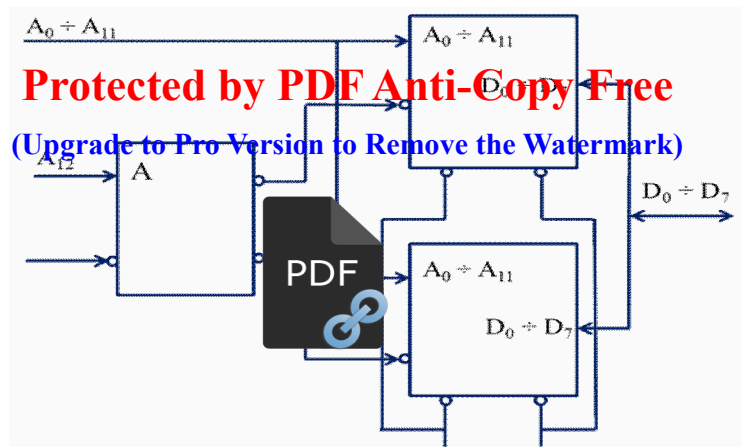
Dung lượng chip nhớ 4K x 8 bit =  $2^{12}$  x 8 bit. Chip nhớ có: 12 chân địa chỉ, 8 chân dữ liệu

Dung lượng module nhớ 8K x 8 bit =  $2^{13}$  x 8 bit. Module nhớ có: 13 chân địa chỉ, 8 chân dữ liệu

Số IC trên một hàng:  $8/8 = 1$

Số IC trên một cột :  $8K/4K = 2$

$\overline{G}$	A	$Y_0$	$Y_1$
0	0	0	1
0	1	1	0
1	X	1	1



### 5.3. BỘ NHỚ CACHE, BỘ NHỚ TRUY CẬP NHANH

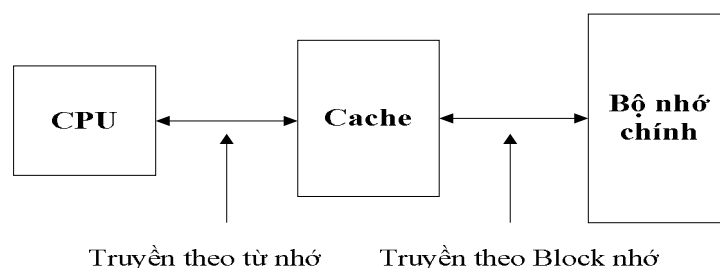
Khi tốc độ của bộ vi xử lý ngày càng vượt xa tốc độ truy nhập bộ nhớ chính (DRAM làm việc nhanh nhất chỉ 60ns), có nghĩa là bộ vi xử lý phải mất thêm vài chu kỳ đợi bộ nhớ hoàn thành quá trình đọc/ghi. Điều này làm giảm hiệu suất làm việc của bộ vi xử lý. Một giải pháp hữu hiệu là sử dụng bộ nhớ đệm cache với tốc độ truy nhập chỉ vài ns. Cache tiết kiệm thời gian truy xuất bộ nhớ của CPU bằng cách dự đoán trước lệnh mà CPU sẽ cần và nạp nó vào trong cache trước khi CPU thực sự cần đến nó. Nếu lệnh cần thiết đã có sẵn trong cache thì CPU sẽ truy xuất dữ liệu từ cache, nếu không, CPU mới truy xuất lên bộ nhớ chính.

#### 5.3.1. Nguyên tắc chung của cache

Bộ nhớ cache dựa theo nguyên lý cục bộ hóa tham chiếu bộ nhớ. Tức là, trong một khoảng thời gian đủ nhỏ CPU thường chỉ tham chiếu các thông tin trong một khối nhớ cục bộ.

Cache có tốc độ nhanh hơn bộ nhớ chính. Cache được đặt giữa CPU và bộ nhớ chính nhằm tăng tốc độ CPU truy cập bộ nhớ. Cache có thể được đặt trên chip CPU.

Hình 5.9 thể hiện vị trí của bộ nhớ cache so với CPU và bộ nhớ chính.



**Hình 5.9. Vị trí đặt bộ nhớ cache**

## Các đặc điểm của bộ nhớ Cache

Cache có tốc độ nhanh hơn bộ nhớ chính.

Bộ nhớ bán dẫn tốc độ cao.

Dung lượng rất nhỏ so với ổ cứng (256 – 512 KB)

Cache HIT: Xác định từ nhớ nằm trong Cache

Cache Miss: Xác định từ nhớ nằm ngoài cache

Tỷ lệ Hit = (Số lần tìm thấy từ nhớ/Tổng số lần tìm)

$H = (85 - 90\%)$

Tỷ lệ Miss =  $1 - H$

## Thao tác của bộ nhớ Cache:

CPU yêu cầu nội dung của ngăn nhớ.

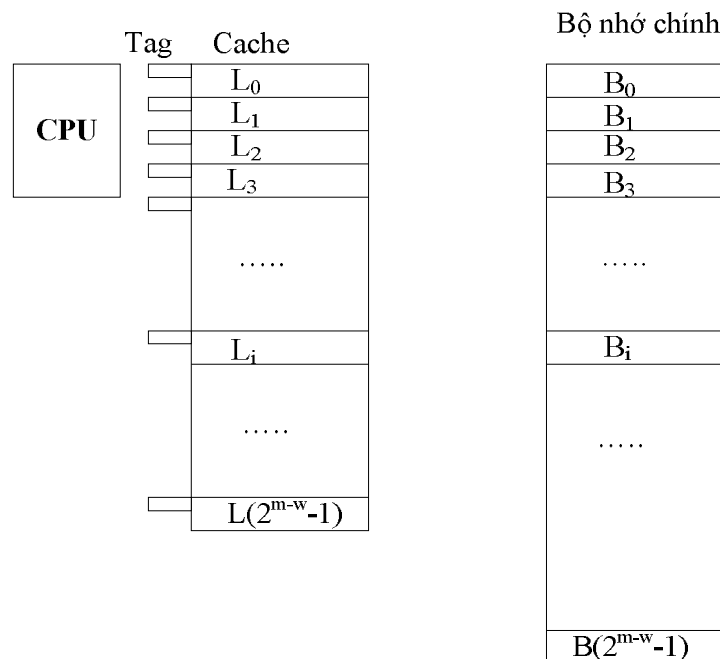
CPU kiểm tra trên cache với dữ liệu này.

Nếu có, CPU nhận dữ liệu từ cache.

Nếu không có, đọc Block nhớ chứa dữ liệu từ bộ nhớ chính vào cache.

Tiếp đó chuyển dữ liệu từ cache vào CPU.

## Cấu trúc chung của cache/ bộ nhớ chính



**Hình 5.10. Cấu trúc chung của cache**

Sử dụng  $n$  bit địa chỉ với độ dài từ nhớ xác định. Số lượng từ nhớ xác định là  $2^n$ .

Bộ nhớ chính được chia thành các Block, mỗi Block có chứa  $2^w$  ô nhớ. Số Block =  $2^{n-w} = 2^r$

Trong n bit địa chỉ dùng w bit LSB để xác định vị trí của từ nhớ trong Block.

Kích thước của Block = 8, 16, 32, 64, 128 byte.

Một số Block của bộ nhớ chính được nạp vào các Line của cache.

Số Line của Cache =  $2^r$

Nội dung TAG (thẻ nhớ) cho biết Block nào của bộ nhớ chính hiện đang được chứa ở Line đó.

Khi CPU truy nhập (đọc/ghi) một từ nhớ, có hai khả năng xảy ra:

Từ nhớ đó có trong cache (cache hit)

Từ nhớ đó không có trong cache (cache miss)

Bộ nhớ cache:

Có m bit địa chỉ ( $m < n$ )

Cache cũng được chia thành các Block, được gọi là Line. Kích thước của 1 Line bằng với các kích thước của một Block

Số Line của Cache =  $2^{m-w} = 2^r$

Trong thực tế thì số Line  $\ll$  số Block

Trong trường hợp Cache miss cần có một thuật toán thay thế thích hợp

Để CPU phân biệt được các Block khi đưa vào trong Cache ta sử dụng địa chỉ TAG. Địa chỉ TAG cho biết Block nào đang nằm trong Cache. Mỗi Block đều có một địa chỉ TAG duy nhất.

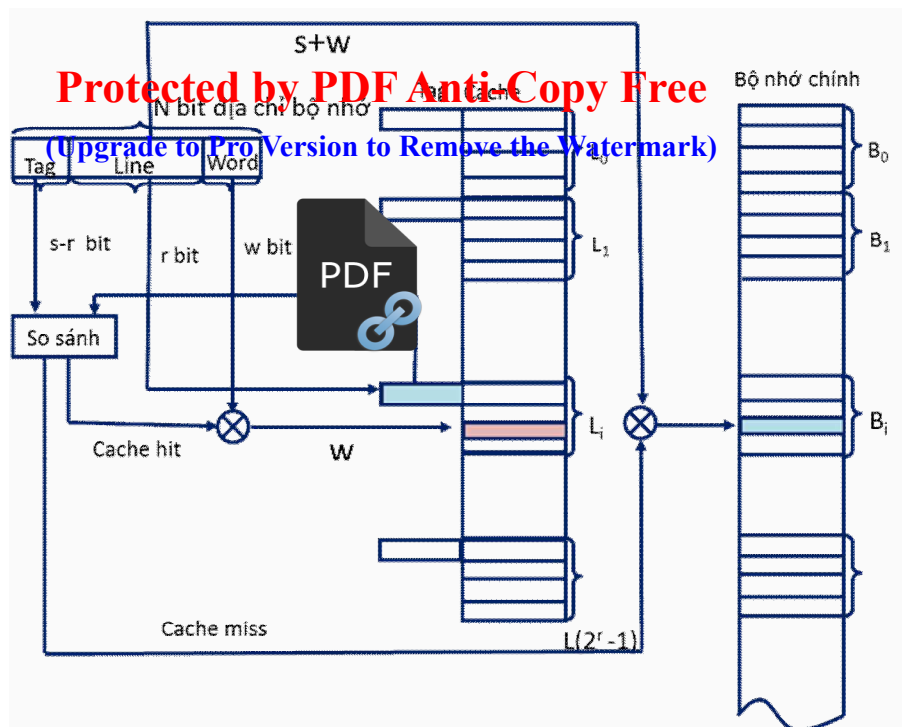
Khi xác định 1 từ nhớ nào đó có trong nằm trong cache hay không ta chỉ việc so sánh phần địa chỉ TAG của từ nhớ cần tìm với phần địa chỉ TAG của tất cả các Line có trong Cache.

### 5.3.2. Các phương pháp ánh xạ

Các phương pháp ánh xạ, chính là các phương pháp tổ chức bộ nhớ cache. Có ba phương pháp ánh xạ: Ánh xạ trực tiếp, ánh xạ liên kết toàn phần và loại thứ ba là phương pháp tổng quát của 2 phương pháp trên đó là ánh xạ liên kết tập hợp.

#### 5.3.2.1. Ánh xạ trực tiếp (Direct mapping)

Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một Line của cache:



Hình 5.11. Minh họa ánh xạ trực tiếp

$B_0 \rightarrow L_0$

$B_1 \rightarrow L_1$

.....

$B_{m-1} \rightarrow L_{m-1}$

$B_m \rightarrow L_0$

$B_{m+1} \rightarrow L_1$

.....

Tổng quát

$B_j$  chỉ có thể được nạp vào  $L_{j \bmod M}$

$M$  là số Line của cache.

Số lượng Block:  $2^s$ ; ( $s = n - w$ )

Trường Word : Kích thước Block hay Line :  $2^w$  ( $w$  là độ dài từ nhớ)

Trường Line : Số Line của cache  $2^r$  (có  $r$  bit)

Trường TAG: Số bit =  $s - r$

Bộ nhớ chính được tổ chức như sau:

TAG	LINE	WORD
s- r	r	w
địa chỉ		

**Ví dụ 5.3.1:** Cho bộ nhớ chính có 16 từ nhớ chia thành các Block, mỗi Block 2 từ nhớ. Cache có dung lượng 8 từ nhớ. Xác định từ địa chỉ để quản lý bộ nhớ cache theo phương pháp ánh xạ trực tiếp.

*Hướng dẫn:*

Bộ nhớ chính có 16 từ nhớ  $= 2^4$ , vậy  $n = 4$

Mỗi Block có 2 từ nhớ  $= 2^1$ , vậy  $w = 1$

$$s = n - w = 4 - 1 = 3$$

Cache có dung lượng 8 từ nhớ  $= 2^3$ , vậy  $m = 3$

$$r = m - w = 3 - 1 = 2$$

$$s - r = 1$$

Tổ chức của một Block:

TAG	LINE	WORD
1	2	1
N bit địa chỉ		

*Tổ chức của cache:*

TAG	W	Cache	Line
00	0	DL1	L0
	1	DL2	
01	0	DL3	L1
	1	DL4	
10	0	DL5	L2



	1	DL6	
Protected by PDF Anti-Copy Free	0	DL7	
(Upgrade to Pro Version to Remove the Watermark)	11	DL8	L3

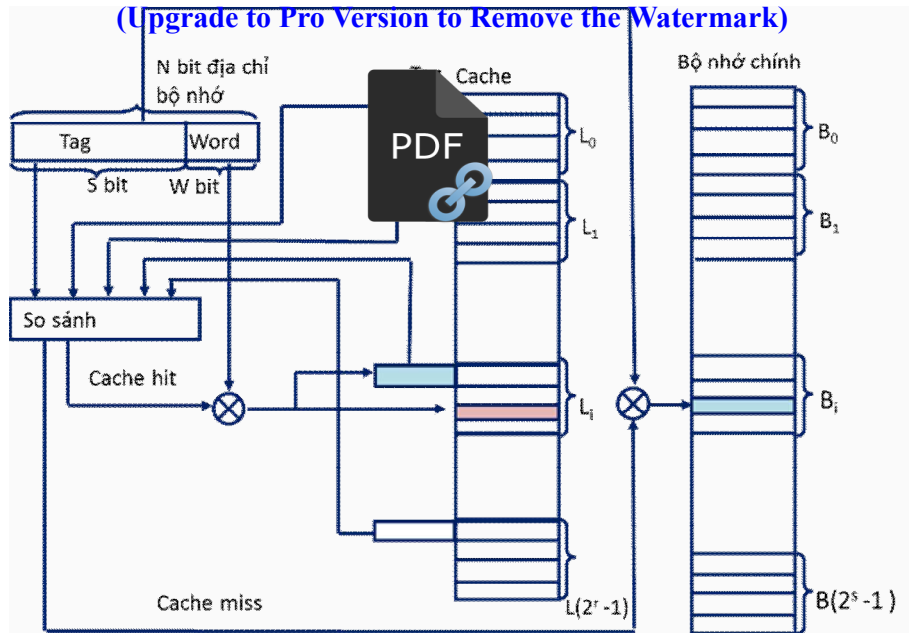


*Tổ chức bộ nhớ chính*

TAG	LINE	Word	BNC	Block
0	00	0	DL1	B0
		1	DL2	
	01	0	DL3	B1
		1	DL4	
	10	0	DL5	B2
		1	DL6	
	11	0	DL7	B3
		1	DL8	
1	00	0	DL9	B4
		1	DL10	
	01	0	DL11	B5
		1	DL12	
	10	0	DL13	B6
		1	DL14	
	11	0	DL15	B7
		1	DL16	

### 5.3.2.2. Ánh xạ liên kết toàn phần (Fully associative mapping)

Mỗi Block có thể nạp vào bất kỳ line nào của cache, miễn là Line đó còn rỗng.



**Hình 5.12. Minh họa ánh xạ liên kết toàn phần**

Địa chỉ của bộ nhớ chính bao gồm hai trường:

TAG	WORD
s	w
n bit địa chỉ	

Trường Word giống như trường hợp ánh xạ trực tiếp.

Trường Tag dùng để xác định Block của bộ nhớ chính.

Tag xác định Block đang nằm ở Line đó.

Đặc điểm của ánh xạ toàn phần:

- Kích thước bộ so sánh lớn nên tốc độ giảm
- Xác suất cache hit cao
- Bộ so sánh phức tạp
- Cần một thuật toán thay thế khi cache đầy
- Các Block có thể nằm ở mọi vị trí trong cache

**Ví dụ 5.3.2:**

Cho  $n = 4$ ,  $w = 1$ ,  $s = 3$ ,  $r = 2$

Tổ chức của Block:

**Protected by PDF Anti-Copy Free**  
(Upgrade to Pro Version to Remove the Watermark)

TAG	WORD
3	1
bit địa chỉ	

Tổ chức của cache:

r=2	T	WORD	Line
00	111	DL7	L0
		DL8	
01	011	DL3	L1
		DL4	
10	001	DL1	L2
		DL2	
11	101	DL5	L3
		DL6	

Tổ chức bộ nhớ chính

TAG	W	BNC	Block
000	0	DL1	B0
	1	DL2	
001	0	DL3	B1
	1	DL4	
010	0	DL5	B2
	1	DL6	
011	0	DL7	B3
	1	DL8	

**Protected by PDF Anti-Copy Free**  
(Upgrade to Pro Version to Remove the Watermark)

TAG	W	BNC	Block
100	0	DL9	B4
	1	DL10	
101	0	DL11	B5
	1	DL12	
110	0	DL13	B6
	1	DL14	
111	0	DL15	B7
	1	DL16	

**Ví dụ 5.3.3:** Ánh xạ địa chỉ

Không gian địa chỉ bộ nhớ chính = 64 MB.

Dung lượng bộ nhớ cache là 64 KB.

Kích thước Line (Block) = 64 byte.

Xác định số bit của các trường địa chỉ cho hai trường hợp :

- Ánh xạ trực tiếp
- Ánh xạ liên kết toàn phần

+) Với ánh xạ trực tiếp

Bộ nhớ chính = 64 MB =  $2^{26}$  byte  $\rightarrow n = 26$  bit

Dung lượng bộ nhớ Cache = 64 KB =  $2^{16}$  byte  $\rightarrow m = 16$  bit

Kích thước Block = 64 byte =  $2^6$  byte  $\rightarrow w = 6$  bit

$r = m - w = 16 - 6 = 10$  bit

$s = n - w = 26 - 6 = 20$  bit

$\rightarrow s - r = 20 - 10 = 10$  bit

Tag	Line	Word
s-r	r	w
10	10	6

+) Ánh xạ liên kết toàn phần

Bộ nhớ chính = 64Mb =  $2^{26}$  byte  $\rightarrow n = 26$  bit

Line = 64 byte =  $2^6$  byte  $\rightarrow w = 6$  bit

Số bit của trường địa chỉ  $s = \frac{P}{w} = \frac{20}{6} \in 20$  bit

(Upgrade to Tag Version to Remove the Wordmark)

Tag	Word
s	w
20	6

### 5.3.2.3. Ánh xạ liên kết tập

- Đây là phương pháp tổ hợp từ 2 phương pháp trên

$B_0 \rightarrow S_0$

$B_1 \rightarrow S_1$

.....

- Cache chia thành các tập (set), mỗi set chứa 1 số line

Số set được tính như sau:

$$\frac{2^r}{k} = 2^d \quad (k \text{ là số line trong set})$$

Nếu  $k=1 \rightarrow r=d$ : phương pháp ánh xạ trực tiếp

Nếu  $k=2^r \rightarrow d=0$ : phương pháp ánh xạ liên kết toàn phần ( $k=2,4,6,8\dots$ )

*Tổ chức của Block:*

TAG	SET	WORD
s-d	d	w

### Ví dụ 5.3.4:

Cho không gian địa chỉ bộ nhớ 4GB, dung lượng bộ nhớ Cache là 256KB, kích thước line là 32 byte

Xác định số bit của trường địa chỉ theo phương pháp:

- Ánh xạ trực tiếp
- Ánh xạ liên kết toàn phần
- Ánh xạ liên kết tập hợp 4 line.

*Hướng dẫn:*

Bộ nhớ chính = 4GB =  $2^{32}$  byte  $\Rightarrow n=32$  bit

Cache = 256 KB =  $2^{18}$  byte  $\Rightarrow m=18$  bit

Line = 32 byte =  $2^5$  byte  $\Rightarrow w=5$  bit

Có:  $r = m - w = 18 - 5 = 13$  bit

$$s = n - w = 32 - 5 = 27 \text{ bit}$$

a) Ánh xạ trực tiếp

**Protected by PDF Anti-Copy Free**  
(Upgrade to Pro Version to Remove the Watermark)

TAG	LINE	WORD
s-r	r	w
14	PDF	5

b) Ánh xạ liên kết toàn phần

TAG	WORD
s	w
27	5

c) Ánh xạ liên kết tập hợp

$$\frac{2^r}{k} = \frac{2^{13}}{4} = 2^{11} \rightarrow d = 11$$

$$\text{Có } s - d = 27 - 11 = 16 \text{ bit}$$

TAG	SET	WORD
s-d	d	w
16	11	5

### Ví dụ 5.3.5:

Giả thiết rằng máy tính có 128KB Cache tổ chức theo kiểu ánh xạ liên kết tập hợp 4 line, Cache có tất cả 1024 set tính từ s0-s1023. Địa chỉ bộ nhớ chính là 32 bit và được đánh địa chỉ cho từng byte

a) Tính số bit cho từng trường địa chỉ cho từng bộ nhớ Cache

b) Hãy xác định byte nhớ địa chỉ 003D024AF (H) được xác định vào set nào trong bộ nhớ Cache?

*Hướng dẫn:*

a) Bộ nhớ chính = 32 bit  $\Rightarrow n = 32$  bit

$$\text{Cache} = 128 \text{ KB} = 2^{17} \text{ byte} \Rightarrow m = 17 \text{ bit}$$

Theo giả thiết bài cho có  $k = 4$

$$\rightarrow \frac{2^r}{4} = 1024 \rightarrow r = 12$$

**Protected by PDF Anti-Copy Free**

$$r = m - w \Rightarrow w = 17 - 12 = 5 \text{ bit}$$

$$s = n - w = 32 - 5 = 27 \text{ bit}$$

\*) Ảnh xạ trực tiếp:

TAG	INDEX	WORD
s-r	r	w
15	12	5

\*) Ảnh xạ liên kết toàn phần

TAG	WORD
s	w
27	5

\*) Ảnh xạ liên kết tập hợp:

$$\text{Số set} = 2^d = 2^{10} \Rightarrow d = 10$$

$$s - d = 27 - 10 = 17 \text{ bit}$$

TAG	SET	WORD
s-d	d	w
17	10	5

b) Địa chỉ 003D02AF = 0000 0000 0011 1101 0000 0010 1010 1111

Các bit lần lượt được sắp xếp:

TAG	SET	WORD
17 bit	10 bit	5 bit
0000 0000 0011 1101 0	000 0010 101	0 1111

$$\text{Số set} = 000 0010 101 = 1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^0 = 21.$$

Vậy địa chỉ 003D02AF được xác định vào S<sub>21</sub>.

### 5.3.3. Thuật giải thay thế

Không phải lựa chọn

Mỗi Block chỉ ánh xạ vào một Line xác định

Thay thế Block ở time đó.

Được thực hiện bằng phần cứng.

Random: Thay thế ngẫu nhiên

FIFO (First In First Out): Thay thế Block nào nằm lâu nhất trong Set đó.

LFU (Least Frequently Used): Thay thế Block nào trong Set có số lần truy nhập ít nhất trong cùng một khoảng thời gian.

LRU (Least Recently Used): Thay thế Block ở trong Set tương ứng có thời gian lâu nhất không được tham chiếu tới.

Tối ưu nhất trong thuật giải thay thế là LRU

#### 5.3.4. Phương pháp ghi dữ liệu cache hit

Ghi xuyên qua (Write - through): Ghi cả cache và cả bộ nhớ chính

Tốc độ chậm

Ghi trả sau (Write - back): Chỉ ghi ra cache

Tốc độ nhanh

Khi Block trong cache bị thay thế cần phải ghi trả cả Block về bộ nhớ chính

#### 5.3.5. Cache trên các bộ xử lý Intel

- 80486: 8KB cache L1 trên chip

- Pentium: Có 2 cache L1 trên chip.

Cache lệnh = 8Kb và Cache dữ liệu = 8Kb

- Pentium 4 (2000): Có hai mức cache L1 và L2 trên chip

Cache L1: Mỗi cache 8 Kb

Kích thước Line = 64 byte

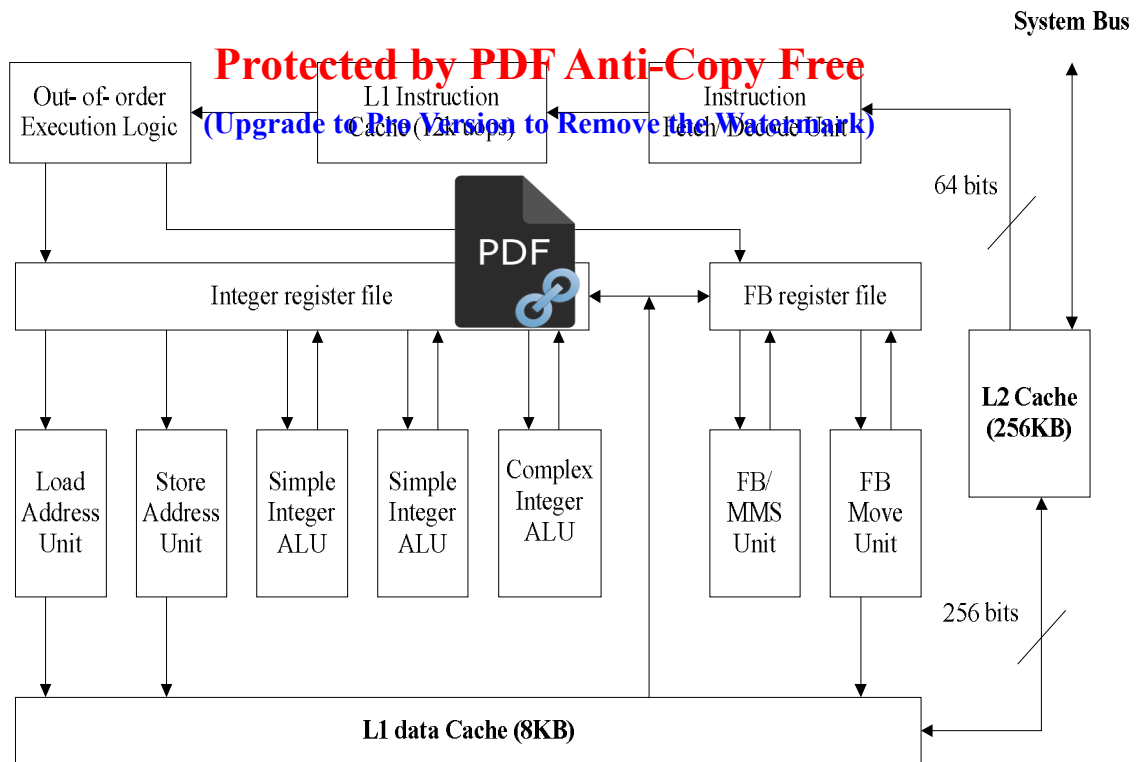
Ánh xạ liên kết tập hợp 4 đường

Cache L2: Mỗi cache 256 Kb

Kích thước Line = 128 byte

Ánh xạ liên kết tập hợp 8 đường.





**Hình 5.13. Sơ đồ Pentium 4**

## 5.4. BỘ NHỚ NGOÀI

### 5.4.1. Đĩa từ

Dù rằng công nghệ mới không ngừng phát minh nhiều loại bộ phận lưu trữ một lượng thông tin lớn thì đĩa từ vẫn giữ vị trí quan trọng từ năm 1965. Đĩa từ có hai nhiệm vụ trong máy tính.

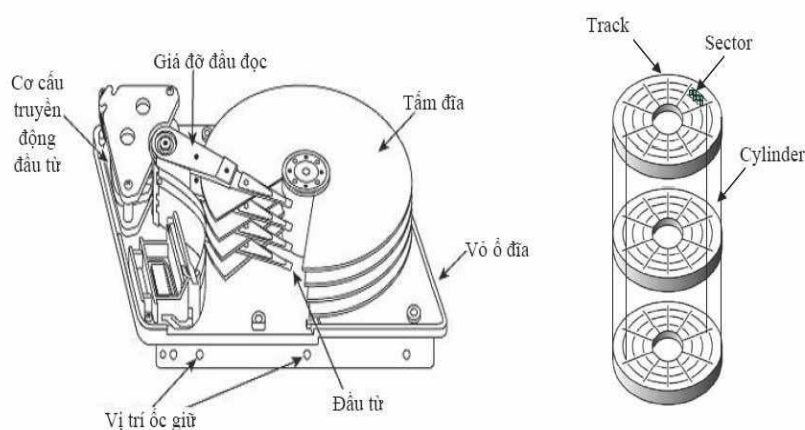
- Lưu trữ dài hạn các tập tin.
- Thiết lập một cấp bộ nhớ bên dưới bộ nhớ trong để làm bộ nhớ ảo lúc chạy chương trình.

Do đĩa mềm dần được các thiết bị lưu trữ khác có các tính năng ưu việt hơn nên chúng ta không xét đến thiết bị này trong chương trình mà chỉ nói đến đĩa cứng. Trong tài liệu này mô tả một cách khái quát cấu tạo, cách vận hành cũng như đề cập đến các tính chất quan trọng của đĩa cứng.

Một đĩa cứng chứa nhiều lớp đĩa (từ 1 đến 4) quay quanh một trục khoảng 3.600 - 15.000 vòng mỗi phút. Các lớp đĩa này được làm bằng kim loại với hai mặt được phủ một chất từ tính. Đường kính của đĩa thay đổi từ 1,3 inch đến 8 inch. Mỗi mặt của một lớp đĩa được chia thành nhiều đường tròn đồng trục gọi là *rãnh*. Thông

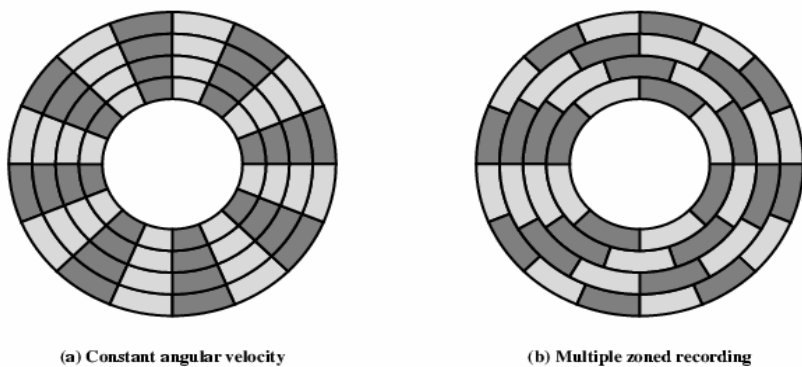
thường mỗi mặt của một lớp đĩa có từ 10.000 đến gần 30.000 rãnh. Mỗi rãnh được chia thành nhiều (sector) dùng để lưu trữ thông tin. Một rãnh có thể chứa từ 64 đến 800 cung. Cung là đơn vị nhỏ nhất mà máy tính có thể đọc hoặc viết (thông thường khoảng 512 bytes). Chuỗi thông tin ghi trên mỗi cung gồm có: số thứ tự của cung, một khoảng trống, số liệu của cung, một khoảng trống, một khoảng trống, số thứ tự của cung tiếp theo [4].

Với kỹ thuật ghi mật độ không đều, tất cả các rãnh đều có cùng một số cung, điều này làm cho các cung dài hơn ở các rãnh xa trục quay có mật độ ghi thông tin thấp hơn mật độ ghi trên các cung nằm gần trục quay.



**Hình 5.14. Cấu tạo của một đĩa cứng**

Với công nghệ ghi với mật độ đều, người ta cho ghi nhiều thông tin hơn ở các rãnh xa trục quay. Công nghệ ghi này ngày càng được dùng nhiều với sự ra đời của các chuẩn giao diện thông minh như chuẩn SCSI.



**Mật độ ghi không đều**

**Mật độ ghi đều**

**Hình 5.15. Mật độ ghi đĩa**

Để đọc hoặc ghi thông tin vào một cung, ta dùng một đầu đọc ghi di động áp vào mỗi mặt của mỗi lớp đĩa. Các đầu đọc ghi này được gắn chặt vào một thanh làm cho chúng cùng di chuyển trên một đường bán kính của mỗi lớp đĩa và như thế tất cả các đầu này đều ở trên những rãnh có cùng bán kính của các lớp đĩa. Từ “trụ” (cylinder) được dùng để gọi tất cả các đầu đọc ghi của các lớp đĩa có cùng bán kính và nằm trên một hình trụ. Người ta luôn luôn quay nhanh đĩa từ nên thông thường ổ đĩa đọc nhiều hơn số dữ liệu cần đọc; người ta nói đây là cách đọc trước. Để quản lý các phức tạp khi kết nối (hoặc ngưng kết nối) lúc đọc (hoặc ghi) thông tin, và việc đọc trước, ổ đĩa cần có bộ điều khiển đĩa. Công nghiệp chế tạo đĩa từ tập trung vào việc nâng cao dung lượng của đĩa mà đơn vị đo lường là mật độ trên một đơn vị bề mặt.

**Bảng 5.1. Bảng thông số kỹ thuật đĩa cứng**

Dung lượng tối đa	có thể đạt 500GB
Số lượng đầu đọc	1-8
Số tấm ghi ( đĩa)	1-4
Cache (bộ đệm)	2-16 MB
Số cung (Sector- 512 bytes/ sector)	xxx,xxx,xxx
Tốc độ quay đĩa (RPM)	3600-15000
Mật độ	có thể đạt 95 Gb/in <sup>2</sup>
Mật độ rãnh (TP1- Max Tracks/ Inch)	có thể đạt 120,000
Mật độ ghi BP1 (Max Bits/Inch)	có thể đạt 702,000
Tốc độ dữ liệu tối đa (Internal)	có thể đạt 900 Mb/s
Tốc độ truyền dữ liệu với ngoại vi	có thể đạt 320 Mb/s
Thời gian chuyển track R/W	có thể đạt 15ms
Thời gian quay nửa vòng	có thể đạt 6 ms

#### 5.4.2. Đĩa quang

Các thiết bị lưu trữ quang rất thích hợp cho việc phát hành các sản phẩm văn hoá, sao lưu dữ liệu trên các hệ thống máy tính hiện nay. Ra đời vào năm 1978, đây là sản phẩm của sự hợp tác nghiên cứu giữa hai công ty Sony và Philips trong công nghiệp giải trí. Từ năm 1980 đến nay, công nghiệp đĩa quang phát triển mạnh trong cả hai lĩnh vực giải trí và lưu trữ dữ liệu máy tính. Quá trình đọc thông tin

dựa trên sự phản chiếu của các tia laser năng lượng thấp từ lớp lưu trữ dữ liệu. Bộ phận tiếp nhận phản xạ sẽ nhận biết được những điểm mà tại đó tia laser bị phản xạ mạnh hay biến đổi (một do các vết khắc (pit) trên bề mặt đĩa. Các tia phản xạ mạnh chỉ ra rằng tại điểm đó không có lỗ khắc và điểm này được gọi là điểm nền (land). Bộ phận ánh sáng trong ổ đĩa thu nhận các tia phản xạ và khuếch tán được khúc xạ từ bề mặt đĩa. Khi các nguồn sáng được thu nhận và xử lý sẽ dịch các mẫu sáng thành các bit dữ liệu hay âm thanh. Các lỗ trên CD sâu 0,12micron và rộng 0,6 micron (1 micron bằng một phần ngàn mm). Các lỗ này được khắc theo một track hình xoắn ốc với khoảng cách 1,6 micron giữa các vòng, khoảng 16.000 track/inch. Các lỗ (pit) và nền (land) kéo dài khoảng 0,9 đến 3,3 micron. Track bắt đầu từ phía trong và kết thúc ở phía ngoài theo một đường khép kín các rìa đĩa 5mm. Dữ liệu lưu trên CD thành từng khối, mỗi khối chứa 2.352 byte. Trong đó, 304 byte chứa các thông tin về bit đồng bộ, bit nhận dạng (ID), mã sửa lỗi (ECC), mã phát hiện lỗi (EDC) [4].

Còn lại 2.048 byte chứa dữ liệu. Tốc độ đọc chuẩn của CD-ROM là 75 khối/s hay 153.600 byte/s hay 150KB/s (1X).

Dưới đây là một số loại đĩa quang thông dụng:

- + CD (Compact Disk): Đĩa quang không thể xóa được, dùng trong công nghiệp giải trí (các đĩa âm thanh được số hoá). Chuẩn đĩa có đường kính 12 cm, âm thanh phát từ đĩa khoảng 60 phút (không dùng).

- + CD-ROM (Compact Disk Read Only Memory): Đĩa không xóa dùng để chứa các dữ liệu máy tính. Chuẩn đĩa có đường kính 12 cm, lưu trữ dữ liệu hơn 650 MB.

- + CD-R (CD-Recordable): Giống như đĩa CD, đĩa mới chưa có thông tin, người dùng có thể ghi dữ liệu lên đĩa một lần và đọc được nhiều lần, không xóa được.

- + CD-RW (CD-Rewritable): có thể ghi và đọc dữ liệu nhiều lần.

- + DVD (Digital Video Disk – Digital Versatile Disk): Đĩa chứa các hình ảnh video được số hóa. Kích thước đĩa có hai loại: 8cm và 12cm. Đĩa DVD có thể chứa dữ liệu trên cả hai mặt đĩa, dung lượng tối đa lên đến 17GB.

Tốc độ đọc chuẩn của DVD – ROM (1X) là 1.3 MB/S (1X của DVD tương đương khoảng 9X của CD ROM).

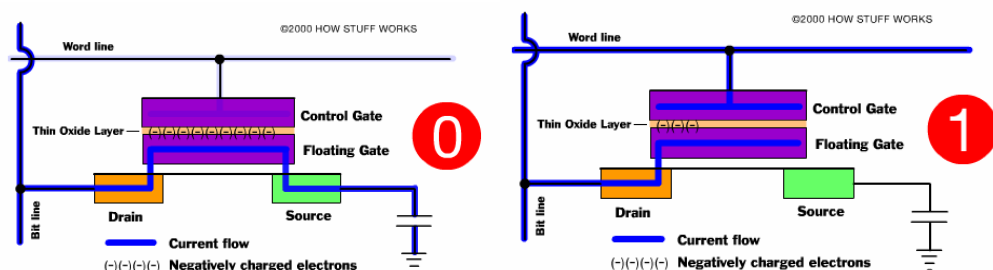
**Bảng 5.2. So sánh một số thông số của hai loại đĩa CDROM và DVDROM**

Dạng đĩa	CDROM	DVDROM
Kích thước Pit	0.834 micron	0.4 micron
Khoảng cách rãnh	1.6 micron	0.74 micron
Số lớp dữ liệu trên đĩa	1 lớp	2 lớp
Số mặt đĩa	1 mặt	1 - 2 mặt
Dung lượng	~700MB	1.36 - 17 GB
Độ phân giải phim	VCD= 320x200	720x640

Với các đặc tính của đĩa quang, giá thành ngày càng thấp, được xem như một phương tiện thích hợp để phân phối các phần mềm cho máy vi tính. Ngoài ra, đĩa quang còn được dùng để lưu trữ lâu dài các dữ liệu thay thế cho băng từ.

#### 5.4.3. Các loại thẻ nhớ

Hiện nay, thẻ nhớ là một trong những công nghệ mới nhất được dùng làm thiết bị lưu trữ. Thẻ nhớ flash là một dạng bộ nhớ bán dẫn EEPROM(công nghệ dùng để chế tạo các chip BIOS trên các vi mạch chính), được cấu tạo bởi các hàng và các cột. Mỗi vị trí giao nhau là một ô nhớ gồm có hai transistor, hai transistor này cách nhau bởi một lớp ô-xít mỏng. Một transistor được gọi là *floating gate* và transistor còn lại được gọi là *control gate*. Floating gate chỉ có thể nối kết với hàng (word line) thông qua control gate. Khi đường kết nối được thiết lập, bit có giá trị 1. Để chuyển sang giá trị 0 theo một qui trình có tên *Fowler-Nordheim tunneling*. Tốc độ, yêu cầu về dòng điện cung cấp thấp và đặc biệt với kích thước nhỏ gọn của các loại thẻ nhớ làm cho kiểu bộ nhớ này được dùng rộng rãi trong công nghệ lưu trữ và giải trí hiện nay [4,7].



**Hình 5.16. Minh hoạ hai trạng thái của một bit nhớ trong thẻ nhớ**

#### 5.4.4. Băng từ

Băng từ có cùng công nghệ với các đĩa từ nhưng khác đĩa từ hai điểm:

- Việc thâm nhập vào đĩa từ là ngẫu nhiên còn việc thâm nhập vào băng từ là tuần tự. Như vậy việc tìm thông tin trên băng từ mất nhiều thời gian hơn việc tìm thông tin trên đĩa từ.

Protected by PDF Anti-Copy Free  
(Upgrade to Pro Version to Remove the Watermark)

- Đĩa từ có dung lượng hạn chế còn băng từ gồm có nhiều cuộn băng có thể lấy ra khỏi máy đọc băng nên dung lượng của băng từ là rất lớn (hàng trăm GB). Với chi phí thấp, băng từ vẫn còn được ứng dụng rộng rãi trong việc lưu trữ dữ liệu dự phòng.

Các băng từ có chiều rộng thay đổi từ 0,38cm đến 1,27 cm được đóng thành cuộn và được chứa trong một hộp bảo vệ. Dữ liệu ghi trên băng từ có cấu trúc gồm một số các rãnh song song theo chiều dọc của băng.

Có hai cách ghi dữ liệu lên băng từ:

Ghi nối tiếp: với kỹ thuật ghi xoắn ốc, dữ liệu ghi nối tiếp trên một rãnh của băng từ, khi kết thúc một rãnh, băng từ sẽ quay ngược lại, đầu từ sẽ ghi dữ liệu trên rãnh mới tiếp theo nhưng với hướng ngược lại. Quá trình ghi cứ tiếp diễn cho đến khi đầy băng từ.

Ghi song song: để tăng tốc độ đọc-ghi dữ liệu trên băng từ, đầu đọc - ghi có thể đọc-ghi một số rãnh kề nhau đồng thời. Dữ liệu vẫn được ghi theo chiều dọc băng từ nhưng các khối dữ liệu được xem như ghi trên các rãnh kề nhau. Số rãnh ghi đồng thời trên băng từ thông thường là 9 rãnh (8 rãnh dữ liệu - 1 byte và một rãnh kiểm tra lỗi).

#### 5.4.5. Biện pháp an toàn dữ liệu khi lưu trữ thông tin trong đĩa từ

Người ta thường chú trọng đến sự an toàn trong lưu giữ thông tin ở đĩa từ hơn là sự an toàn của thông tin trong bộ xử lý. Bộ xử lý có thể hư mà không làm tổn hại đến thông tin. Ổ đĩa của máy tính bị hư có thể gây ra các thiệt hại rất to lớn.

Một phương pháp giúp tăng cường độ an toàn của thông tin trên đĩa từ là dùng một mảng đĩa từ. Mảng đĩa từ này được gọi là Hệ thống đĩa dự phòng (RAID - Redundant Array of Independent Disks). Cách lưu trữ dư thông tin làm tăng giá tiền và sự an toàn (ngoại trừ RAID 0). Cơ chế RAID có các đặc tính sau [6]:

- RAID là một tập hợp các ổ đĩa cứng (vật lý) được thiết lập theo một kỹ thuật mà hệ điều hành chỉ “nhìn thấy” chỉ là một ổ đĩa (logic) duy nhất.

- Với cơ chế đọc/ghi thông tin diễn ra trên nhiều đĩa (ghi đan chéo hay soi gương).

-Trong mảng đĩa có lưu các thông tin kiểm tra lỗi dữ liệu; do đó, dữ liệu có thể được phục hồi nếu có một đĩa trong mảng đĩa bị hư hỏng.

**Protected by PDF Anti-Copy Free**  
(Upgrade to Pro Version to Remove the Watermark)

## CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 5



**Câu 1 :** Nêu các loại bộ nhớ và đặc điểm của từng loại?

**Câu 2 :** Nêu những nhiệm vụ chính của việc điều khiển nhớ DRAM của máy tính?

**Câu 3 :** Nêu nguyên tắc chung của cache

**Câu 4 :** Nêu các phương pháp tổ chức bộ nhớ cache và đặc điểm của các phương pháp này.

**Câu 5 :** Nêu đặc điểm của đĩa từ?

**Câu 6 :** Nêu đặc điểm của đĩa quang ?

**Câu 7:** Vẽ cấu trúc Ram (16K x 1 bit)

**Câu 8:** Xây dựng bộ nhớ 64KB cho một máy tính sử dụng CPU 8085 (16 bit địa chỉ, 8 bit dữ liệu) từ các RAM được cấu tạo từ các ma trận 128x128 bit với 4 bit dữ liệu: Địa chỉ bộ nhớ bắt đầu: C000H

**Câu 9:** Cho chip nhớ SRAM 4K x 4 bit. Thiết kế module nhớ 4K x 8 bit

**Câu 10:** Cho chip nhớ SRAM 4k x 8 bit. Thiết kế module nhớ 8K x 8bit

**Câu 11:** Cho bộ nhớ chính có 16 từ nhớ chia thành các Block, mỗi Block 2 từ nhớ. Cache có dung lượng 8 từ nhớ. Xác định từ địa chỉ để quản lý bộ nhớ cache theo phương pháp ánh xạ trực tiếp.

**Câu 12:** Một máy tính có dung lượng bộ nhớ chính là 64MByte được tổ chức theo các block nhớ. Bộ nhớ cache có dung lượng là 64Kbyte, kích thước đường là 64 byte. Tính dạng địa chỉ truy cập cache trong trường hợp ánh xạ liên kết hoàn toàn.

**Câu 13:** Giả thiết rằng máy tính có 128KB cache tổ chức theo kiểu ánh xạ liên kết tập hợp 4 line. Cache có tất cả là 1024 Set từ S0->S1023. Địa chỉ bộ nhớ chính là 32 bit và đánh địa chỉ cho từng byte.

Xác định byte nhớ có địa chỉ 115B14CF (H) được ánh xạ vào Set nào của Cache?

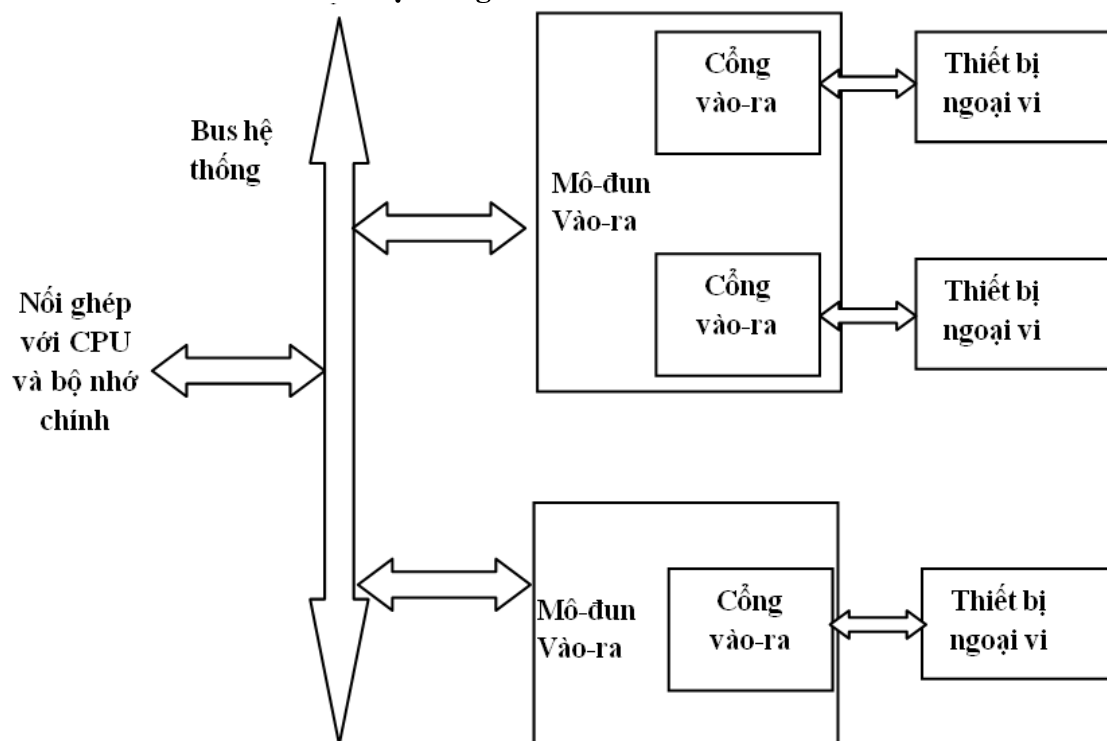
Trang bị cho sinh viên kiến thức về hệ thống chuyển dữ liệu giữa máy tính và thiết bị ngoại vi, các phương pháp truyền dữ liệu và các cách ghép nối với thiết bị ngoại vi.



Sinh viên cần nắm vững các kiến thức về hệ thống kết nối cơ bản các bộ phận bên trong máy tính, cách giao tiếp giữa các ngoại vi và bộ xử lý. Thông qua hệ thống BUS truyền dữ liệu.

## 6.1. CẤU TRÚC CHUNG CỦA HỆ THỐNG VÀO RA

### 6.1.1. Cấu trúc cơ bản của hệ thống vào ra



**Hình 6.1. Cấu trúc cơ bản của hệ thống vào ra**

- Chức năng của hệ thống vào ra: Trao đổi thông tin giữa máy tính với thế giới bên ngoài.

- Các thao tác cơ bản: Vào dữ liệu (Input)

Ra dữ liệu (Output)



- Các thành phần chính: Các thiết bị ngoại vi

**Protected by PDF Anti-Copy Free**

- Đặc điểm của vào-ra: (Upgrade to Pro Version to Remove the Watermark)

- + Tồn tại đa dạng các thiết bị ngoại vi khác nhau về:

Nguyên tắc hoạt động

Tốc độ

Khuôn dạng dữ liệu



- + Tất cả các thiết bị ngoại vi đều chậm hơn CPU và RAM, nên cần có các mô-đun vào-ra để nối ghép các thiết bị ngoại vi với CPU và bộ nhớ chính

### 6.1.2. Các thiết bị ngoại vi

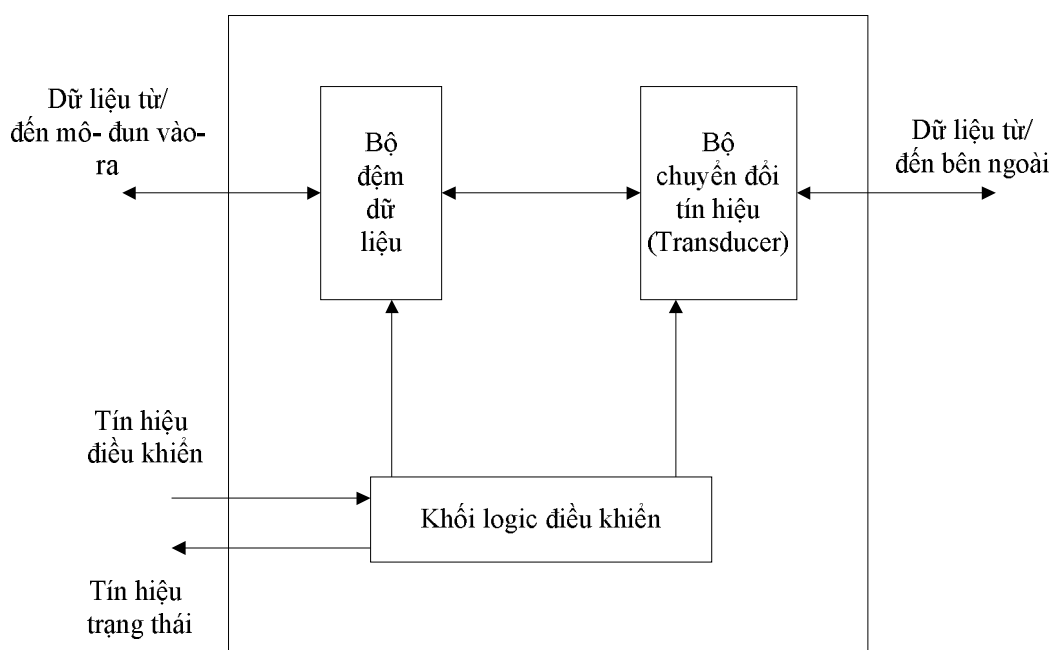
- Chức năng: Chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính

- Phân loại:

Thiết bị ngoại vi giao tiếp người với máy. Ví dụ: Bàn phím, màn hình, máy in....

Thiết bị ngoại vi giao tiếp máy với máy. Gồm các thiết bị theo dõi và kiểm tra

Thiết bị ngoại vi truyền thông. Modem, Network Interface Card (NIC).



**Hình 6.2. Cấu trúc chung của thiết bị ngoại vi**

- Các thành phần của thiết bị ngoại vi:

Bộ chuyển đổi tín hiệu: Chuyển đổi dữ liệu giữa bên ngoài với bên trong máy tính

**Protected by PDF Anti-Copy Free**

Bộ đệm dữ liệu: Đệm dữ liệu khi trao đổi giữa thiết bị ngoại vi.

Khối logic điều khiển: Điều khiển hoạt động của thiết bị ngoại vi đáp ứng theo yêu cầu từ mô-đun vào-ra.



### 6.1.3. Mô-đun vào-ra

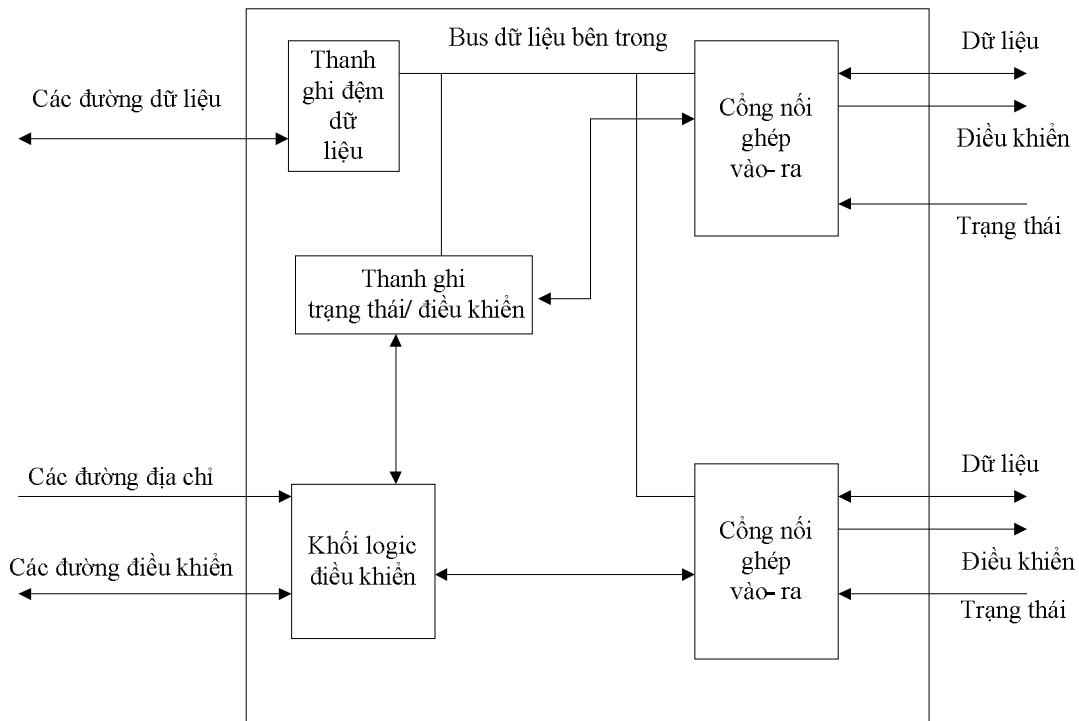
Điều khiển và định thời

Trao đổi thông tin với CPU hoặc bộ nhớ chính

Trao đổi thông tin với thiết bị ngoại vi

Đệm giữa bên trong máy tính với thiết bị ngoại vi

Phát hiện lỗi của thiết bị ngoại vi



**Hình 6.3. Cấu trúc chung của mô-đun vào-ra**

- Các thành phần của mô-đun vào-ra:

Thanh ghi đệm dữ liệu: Đệm dữ liệu trong quá trình trao đổi

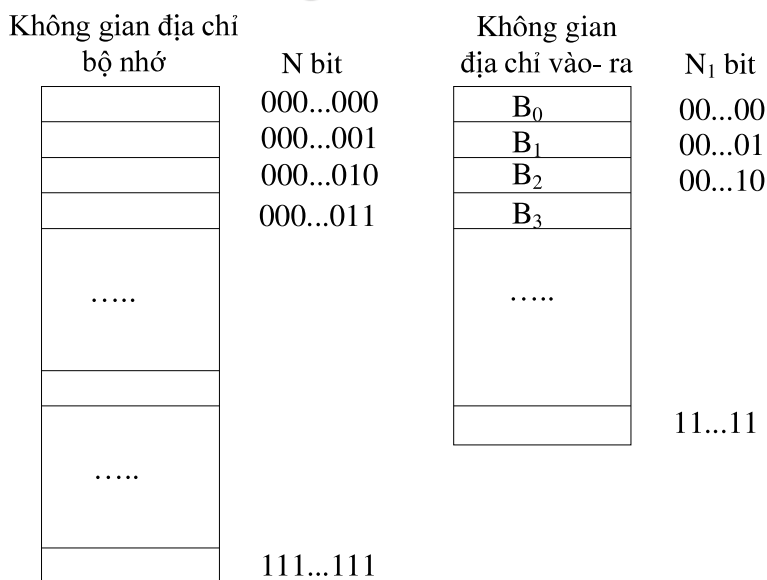
Các cổng vào ra (I/O port): Kết nối với thiết bị ngoại vi, mỗi cổng có một địa chỉ xác định

Thanh ghi trạng thái/ điều khiển: Lưu giữ thông tin trạng thái/điều khiển cho các cổng vào ra

Khởi lập (Upd) điều khiển (Pro) điều khiển bộ nhớ vào- ra (M) điều khiển bộ nhớ vào- ra (M)

#### 6.1.4. Địa chỉ hóa cổng vào ra

##### 6.1.4.1. Không gian địa chỉ của bộ xử lý



**Hình 6.4. Không gian địa chỉ của bộ xử lý**

- Một số bộ xử lý chỉ quản lý duy nhất một không gian địa chỉ. Không gian địa chỉ của bộ nhớ là  $2^N$  địa chỉ. Ví dụ: Các bộ xử lý 680x0 (Motorola)

- Một số bộ xử lý quản lý hai không gian địa chỉ tách biệt:

Không gian địa chỉ bộ nhớ:  $2^N$  địa chỉ

Không gian địa chỉ vào ra:  $2^{N_1}$  địa chỉ

Có tín hiệu điều khiển phân biệt truy nhập không gian địa chỉ

Tập lệnh có các lệnh vào-ra chuyên dụng

-Ví dụ: pentium (Intel)

Không gian địa chỉ bộ nhớ=  $2^{32}$  byte = 4 GB

Không gian địa chỉ vào-ra=  $2^{16}$  byte = 64 KB

Tín hiệu điều khiển  $M/\overline{IO}$

Lệnh vào-ra chuyên dụng: In , Out

#### 6.1.4.2. Các phương pháp địa chỉ hóa cổng vào-ra

- Vào ra bằng lệnh

Cổng vào-ra được đánh địa chỉ theo không gian địa chỉ vào-ra

CPU trao đổi dữ liệu với cổng vào-ra thông qua các lệnh vào-ra chuyên dụng (In, OUT)

Chỉ có thể thực hiện trên cổng có quản lý không gian địa chỉ vào-ra riêng biệt

- Vào ra theo bản đồ bộ nhớ:

Cổng vào-ra được đánh địa chỉ theo không gian địa chỉ bộ nhớ

Vào-ra giống như đọc/ghi bộ nhớ

CPU trao đổi dữ liệu với cổng vào-ra thông qua các lệnh truy nhập dữ liệu bộ nhớ

Có thể thực hiện trên mọi hệ thống

### 6.2. CÁC PHƯƠNG PHÁP TRAO ĐỔI DỮ LIỆU

#### 6.2.1. Vào-ra bằng chương trình

- Nguyên tắc chung: CPU điều khiển trực tiếp vào-ra bằng chương trình nên cần phải lập trình vào-ra

- Các tín hiệu điều khiển vào-ra:

Tín hiệu điều khiển (control): Kích hoạt thiết bị ngoại vi

Tín hiệu kiểm tra (test): Kiểm tra trạng thái của mô-đun vào-ra và thiết bị ngoại vi

Tín hiệu điều khiển (read): Yêu cầu mô-đun vào-ra nhận dữ liệu bằng thiết bị ngoại vi và đưa vào thanh ghi đệm dữ liệu, rồi CPU nhận dữ liệu đó

Tín hiệu điều khiển ghi (Write): Yêu cầu mô-đun vào-ra lấy dữ liệu trên bus dữ liệu đưa đến thanh ghi đệm dữ liệu và chuyển ra thiết bị ngoại vi

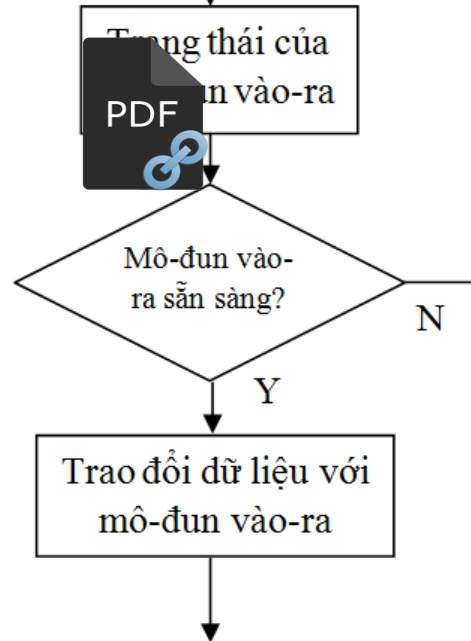
- Các lệnh vào ra:

Vào- ra riêng biệt: Sử dụng lệnh vào-ra chuyên dụng (In, Out)

Vào-ra theo bản đồ bộ nhớ: Sử dụng các lệnh trao đổi dữ liệu với bộ nhớ để trao đổi dữ liệu với cổng vào ra

- Lưu đồ đoạn chương trình vào-ra:

**Protected by PDF Anti-Copy Free**  
(Upgrade to Pro Version to Remove the Watermark)



**Hình 6.5. Lưu đồ đoạn chương trình vào-ra**

- Hoạt động của vào-ra bằng chương trình [5]:

CPU yêu cầu thao tác vào-ra

Mô-đun vào-ra thực hiện thao tác

Mô-đun vào-ra thiết lập các bit trạng thái

CPU kiểm tra các bit trạng thái: Nếu chưa sẵn sàng thì quay lại kiểm tra. Nếu sẵn sàng thì chuyển sang trao đổi dữ liệu với mô-đun vào-ra

- Đặc điểm của phương pháp vào-ra bằng chương trình:

Vào-ra do ý muốn của người lập trình

CPU trực tiếp điều khiển vào-ra

CPU đợi mô-đun vào-ra, nên tiêu tốn thời gian của CPU

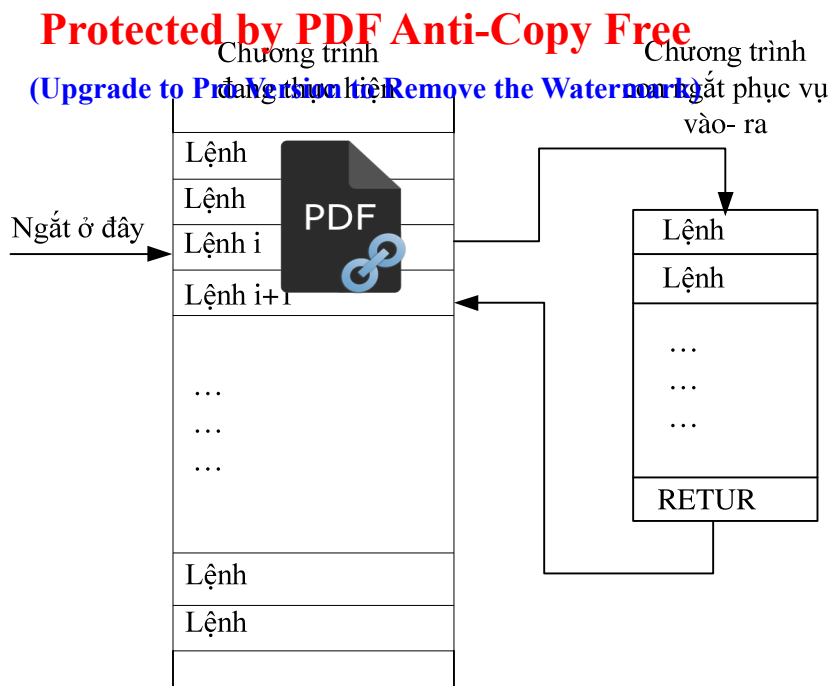
### 6.2.2. Vào-ra điều khiển bằng ngắt

- Nguyên tắc chung: CPU không phải đợi trạng thái sẵn sàng của mô-đun vào-ra, CPU thực hiện một chương trình nào đó

Khi mô-đun vào-ra sẵn sàng thì nó phát tín hiệu ngắt CPU

CPU thực hiện chương trình con vào-ra tương ứng để trao đổi dữ liệu

CPU trở lại tiếp tục thực hiện chương trình đang bị ngắt



**Hình 6.6. Vào-ra điều khiển bằng ngắt**

- Hoạt động vào dữ liệu:

+ Nhìn từ mô-đun vào-ra:

Mô-đun vào-ra nhận tín hiệu điều khiển đọc từ CPU

Mô-đun vào-ra nhận tín hiệu từ thiết bị ngoại vi, trong khi đó CPU làm việc khác

Khi đã có dữ liệu -> mô-đun vào-ra phát tín hiệu ngắt CPU

CPU yêu cầu dữ liệu

Mô-đun vào-ra chuyển dữ liệu đến CPU

+ Nhìn từ CPU:

Phát tín hiệu điều khiển đọc

Làm việc khác

Cuối mỗi chu kỳ lệnh, kiểm tra tín hiệu ngắt. Nếu bị ngắt thì cất giữ cảnh (nội dung các thanh ghi), thực hiện chương trình con ngắt để vào dữ liệu, khôi phục ngữ cảnh của chương trình đang thực hiện.

- Các phương pháp nối ghép ngắt:

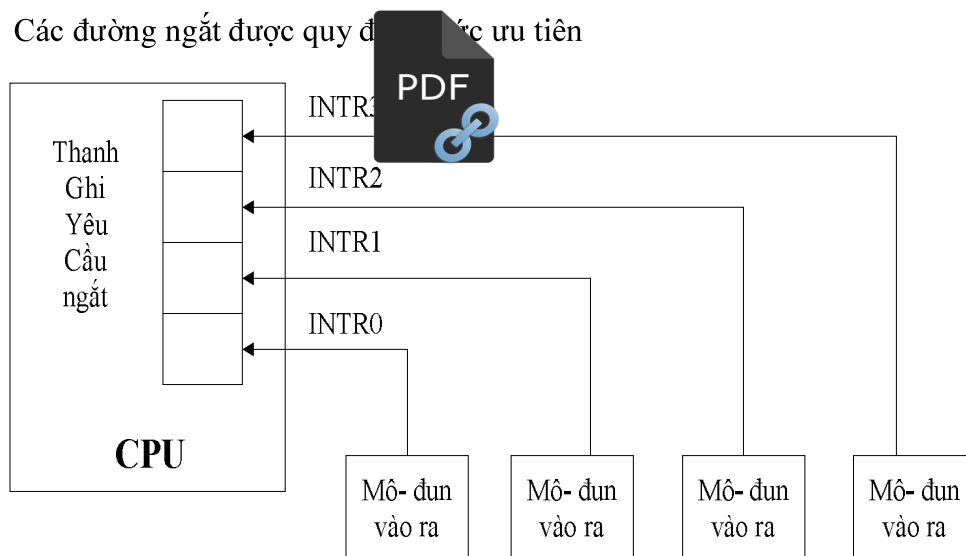
+ Sử dụng nhiều đường yêu cầu ngắt

Mỗi mô-đun vào-ra được nối với đường yêu cầu ngắt

CPU phải chờ tín hiệu đồng ý tín hiệu yêu cầu ngắt

Hạn chế số lượng mô-đun vào-ra

Các đường ngắt được quy định theo ưu tiên



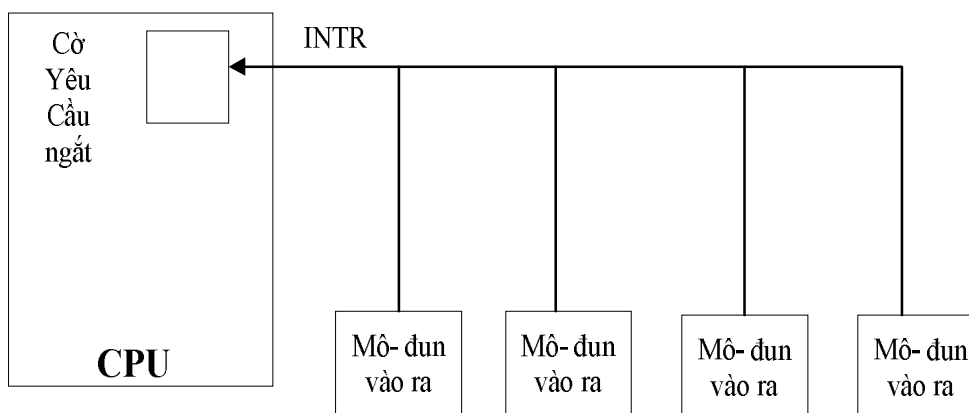
**Hình 6.7. Phương pháp nối ghép ngắt sử dụng nhiều đường yêu cầu ngắt**

+ Hỏi vòng bằng phần mềm:

CPU sử dụng phần mềm thực hiện hỏi lần lượt từng mô-đun vào-ra

Hỏi vòng bằng phần mềm chậm

Thứ tự các mô-đun được hỏi vòng chính là thứ tự ưu tiên



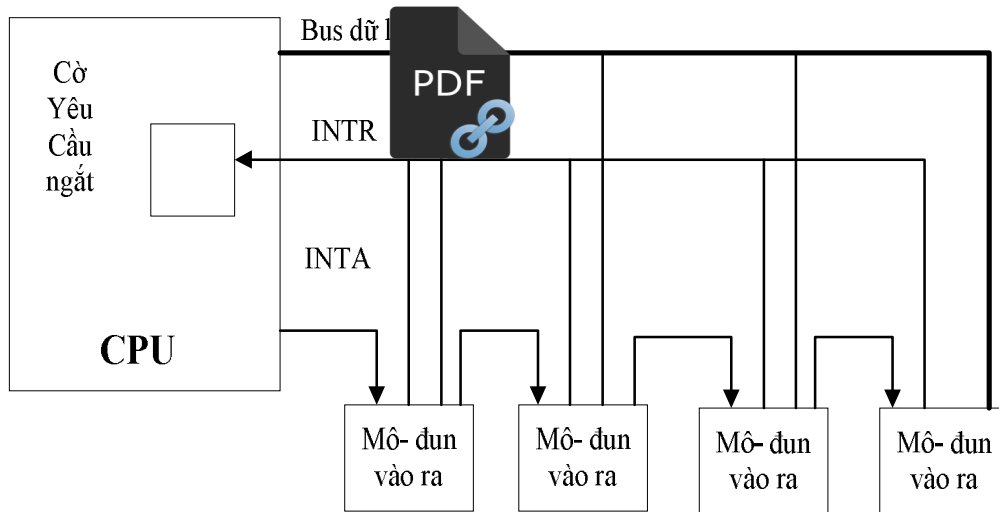
**Hình 6.8. Phương pháp nối ghép ngắt hỏi vòng bằng phần mềm**

+ Hỏi vòng bằng phần cứng:

CPU phát tín hiệu chấp nhận ngắt (INTA) đến mô-đun vào-ra đầu tiên

Nếu mô-đun vào ra đó không gây ra ngắt thì nó gửi tín hiệu đến mô-đun kế tiếp cho đến khi xác định được mô-đun gây ngắt

Thứ tự các mô-đun vào ra trong chuỗi xác định mức ưu tiên



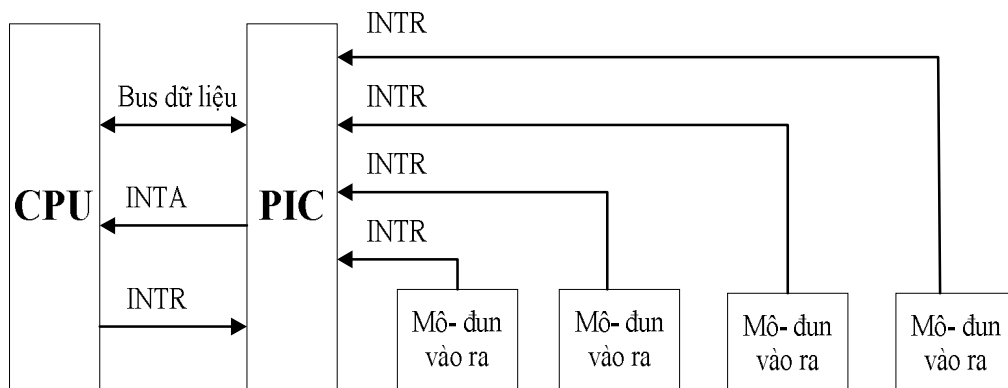
**Hình 6.9. Phương pháp nối ghép ngắt hồi vòng bằng phần cứng**

+ Sử dụng bộ điều khiển ngắt (PIC)

PIC: Programmable Interrupt Controller

PIC có nhiều đường vào yêu cầu ngắt có quy định mức ưu tiên

PIC chọn một yêu cầu ngắt không bị cấm có mức ưu tiên cao nhất gửi đến CPU



**Hình 6.10. Phương pháp nối ghép ngắt sử dụng bộ điều khiển ngắt PIC**

- Đặc điểm của vào ra điều khiển bằng ngắt:

Có sự kết hợp giữa phần cứng và phần mềm. Phần cứng gây ngắt CPU, phần mềm trao đổi dữ liệu

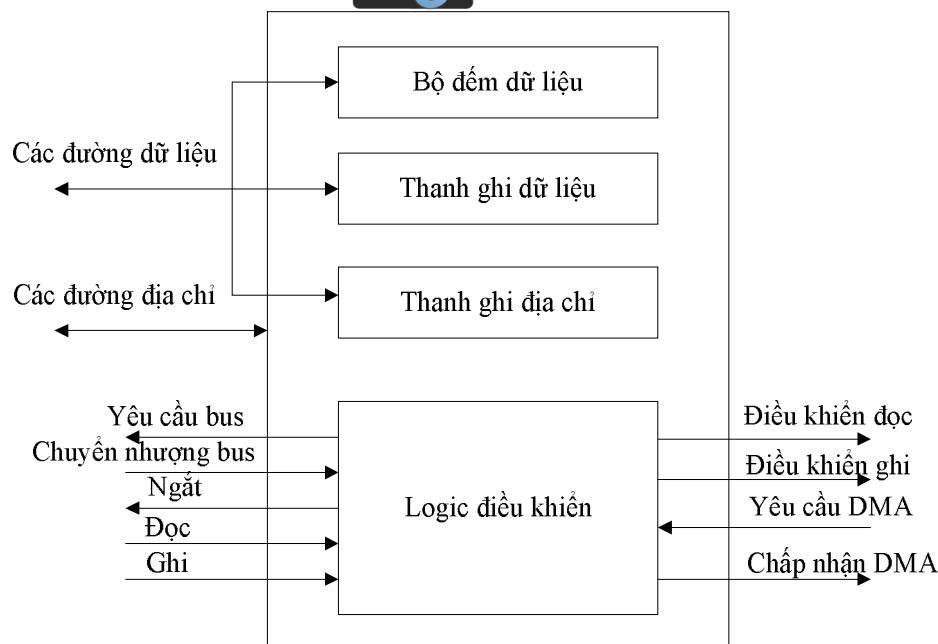
CPU trực tiếp điều khiển vào-ra



CPU không phải đợi mô-đun vào-ra nên hiệu quả sử dụng CPU tốt hơn

### 6.2.3. Truy nhập bộ nhớ trực tiếp – DMA (Direct Memory Access)

Vào ra bằng chương trình và ngắt CPU trực tiếp điều khiển nên chiếm thời gian của CPU, tốc độ truyền bị hạn chế vì phải chuyển qua CPU. Để khắc phục dùng DMA, thêm mô-đun phần cứng trên bus (bộ điều khiển DMA). DMAC điều khiển trao đổi dữ liệu giữa mô-đun vào-ra với bộ nhớ chính [6].



**Hình 6.11. Sơ đồ cấu trúc của DMAC**

- Các thành phần của DMAC:

Thanh ghi dữ liệu: Chứa dữ liệu trao đổi

Thanh ghi địa chỉ: Chứa địa chỉ ngăn nhớ dữ liệu

Bộ đếm dữ liệu: Chứa số từ dữ liệu cần trao đổi

Logic điều khiển: Điều khiển hoạt động của DMAC

- Hoạt động của DMA:

+ CPU “nói” cho DMAC:

Vào hay ra dữ liệu

Địa chỉ thiết bị vào-ra (Cổng Vào-Ra tương ứng)

Địa chỉ đầu của mảng nhớ chứa dữ liệu, sau đó nạp vào thanh ghi địa chỉ

Số từ dữ liệu cần truyền, sau đó nạp vào bộ đếm dữ liệu

+ CPU làm việc khác

+ DMAC điều khiển trao đổi dữ liệu

+ Sau khi truyền được một từ dữ liệu thì.

Nội dung thanh ghi địa chỉ

Nội dung bộ đếm dữ liệu ghi

+ Khi bộ đếm dữ liệu = 0, DMAC gửi tín hiệu ngắt CPU để báo kết thúc DMA

- Các kiểu thực hiện DMA

DMA truyền theo khối: DMAC sử dụng bus để truyền xong cả khối dữ liệu

DMA lấy chu kỳ: DMAC cưỡng bức CPU treo tạm thời từng chu kỳ bus, DMAC chiếm bus thực hiện truyền một từ dữ liệu

DMA trong suốt: DMAC nhận biết những chu kỳ nào CPU không sử dụng bus thì chiếm bus để trao đổi một từ dữ liệu

- Đặc điểm của DMA:

CPU không tham gia trong quá trình trao đổi dữ liệu.

DMAC điều khiển trao đổi dữ liệu giữa bộ nhớ chính và mô-đun vào-ra (hoàn toàn bằng phần cứng) nên tốc độ nhanh

Phù hợp với những yêu cầu trao đổi mảng dữ liệu lớn

#### **6.2.4. Kênh vào-ra hay bộ xử lý vào-ra**

Việc điều khiển vào-ra được thực hiện bởi một bộ xử lý vào-ra chuyên dụng

Bộ xử lý vào-ra hoạt động theo chương trình riêng của nó

Chương trình của bộ xử lý vào-ra có thể nằm trong bộ nhớ chính hoặc nằm trong bộ nhớ riêng

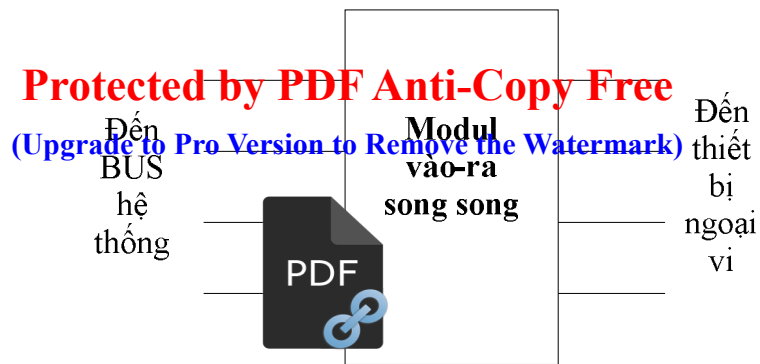
Hoạt động theo kiến trúc đa xử lý

### **6.3. GHÉP NỐI VỚI THIẾT BỊ NGOẠI VI**

#### **6.3.1. Các kiểu nối ghép vào ra**

Có hai kiểu nối ghép vào ra đó là: Nối ghép song song và nối ghép nối tiếp [7].

##### **6.3.1.1. Nối ghép song song**



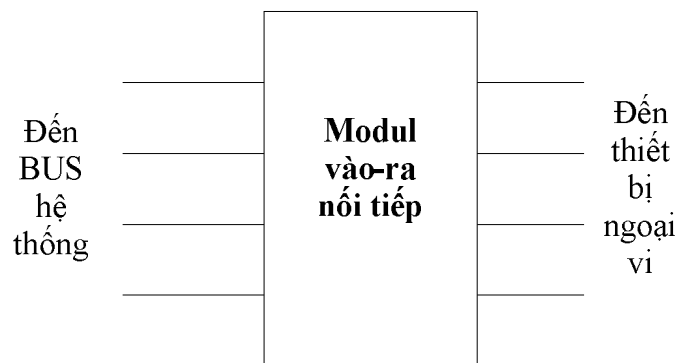
**Hình 6.12. Nối ghép vào- ra song song**

Truyền nhiều bit song song.

Tốc độ nhanh.

Cần nhiều đường truyền dữ liệu.

#### **6.3.1.2. Nối ghép nối tiếp**



**Hình 6.13. Nối ghép vào- ra nối tiếp**

Cần truyền lần lượt từng bit.

Cần có bộ truyền đổi dữ liệu từ song song sang nối tiếp và ngược lại.

Tốc độ chậm hơn.

Cần ít đường truyền dữ liệu.

#### **6.3.2. Các cấu hình ghép nối**

Điểm tới điểm (Point to point). Thông qua một cổng vào – ra nối ghép với một thiết bị ngoại vi.

Điểm tới đa điểm (Point to Multipoint). Thông qua một cổng vào – ra cho phép nối ghép được với nhiều thiết bị ngoại vi.

Ví dụ:

SCSI (Small Computer System Interface): 7 hoặc 15 thiết bị.

USB (Universal Serial Bus): 127 thiết bị.

IEEE 1394 (FireWire): 63 thiết bị.

**Protected by PDF Anti-Copy Free**  
(Upgrade to Pro Version to Remove the Watermark)

## CÂU HỎI TRẮC NGHIỆM VÀ BÀI TẬP CHƯƠNG 6



**Câu 1:** Nêu cấu trúc cơ bản của hệ thống vào-ra ?

**Câu 2:** Nêu các phương pháp địa chỉ hóa cổng vào-ra ?

**Câu 3:** Nêu các phương pháp trao đổi dữ liệu và đặc điểm của mỗi phương pháp ?

**Câu 4:** Các kiểu ghép nối vào-ra ?

**Câu 5:** Các cấu hình ghép nối ?

Trang bị cho sinh viên kiến thức về cấu trúc của hệ thống truyền dữ liệu, các phương pháp trao đổi dữ liệu song song, nối tiếp, giữa máy tính và thiết bị ngoại vi.

Sinh viên cần nắm vững các kiến thức về hệ thống kết nối cơ bản các bộ phận bên trong máy tính, các phương pháp trao đổi dữ liệu song song, nối tiếp, giao diện đa năng, giao diện cao tốc.

### 7.1. GIAO DIỆN SONG SONG

Hệ thống ghép nối vào/ra đáp ứng việc truyền dữ liệu giữa hệ máy tính cơ sở với thiết bị ngoại vi được nối ghép với chúng.

Cấu trúc ghép nối của hệ thống được chỉ ra trên hình 7.1.

Hệ thống vào /ra trao đổi số liệu với các thiết bị ngoại vi thông qua mạch tương thích gọi là cổng (PORT). Thiết bị ngoại vi nối trực tiếp với cổng, cổng được nối trực tiếp với mạch kiểm soát kênh.

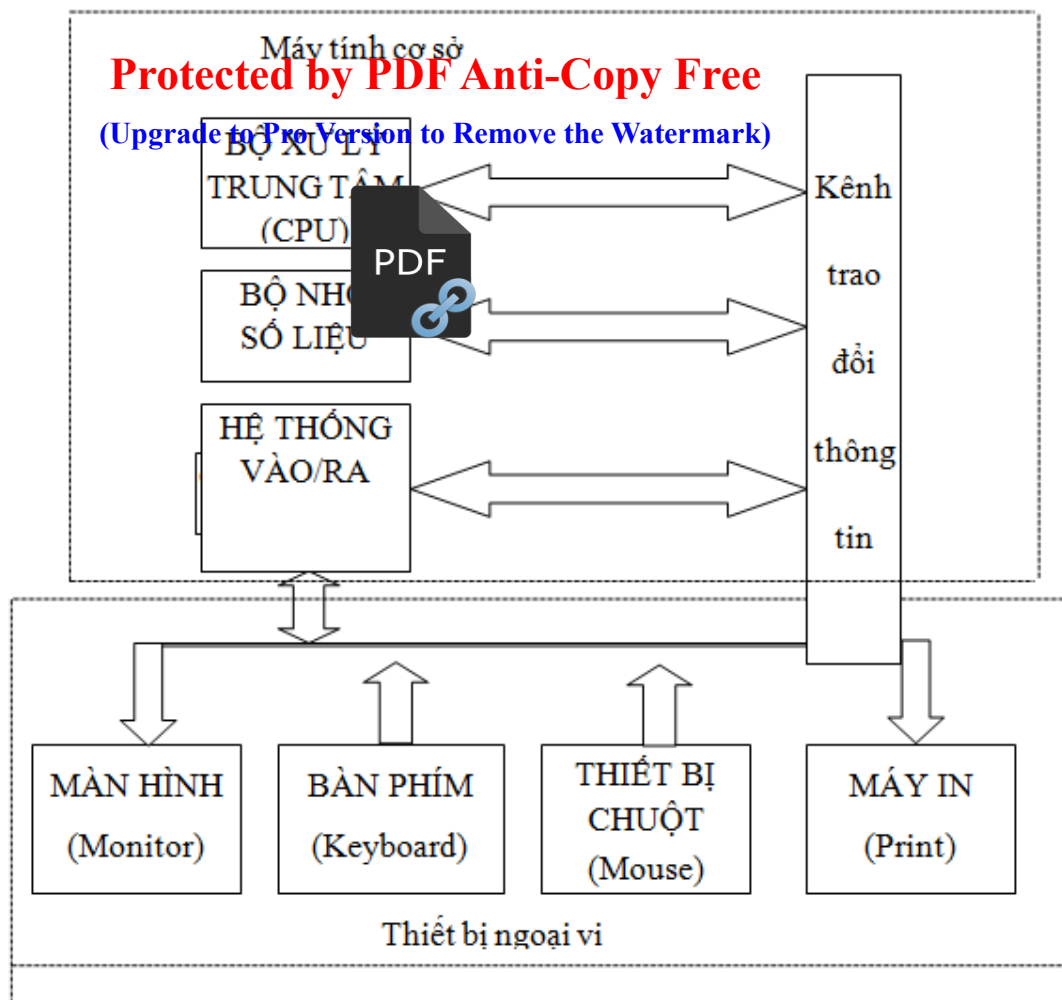
Khi thiết bị ngoại vi nối với cổng thì nó yêu cầu phải có một phương pháp thâm nhập vào bộ xử lý trung tâm, phương pháp này gọi là thao tác ngắt (INTERRUPT). Thao tác ngắt sẽ tạm thời dừng chương trình mà bộ xử lý trung tâm đang thực hiện, CPU sẽ đáp ứng cho thao tác ngắt bằng một chương trình con phục vụ ngắt (Interrupt Service Routine). Có hai loại thao tác ngắt cơ bản là: Ngắt có che mặt nạ (Maskable) và ngắt không che mặt nạ (Nonmaskable).

Thao tác ngắt không che mặt nạ đòi hỏi CPU phải đáp ứng ngay và được dùng trong những trường hợp khẩn cấp. Ví dụ như mất điện nguồn cung cấp.

Ngắt che mặt nạ là thao tác ngắt mà CPU có thể bỏ qua, tùy thuộc vào trạng thái của thanh ghi cờ trạng thái.

Ngắt có thể là có các mức ưu tiên khác nhau, tùy thuộc vào từng yêu cầu và độ cấp bách. Ví dụ như ngắt do yêu cầu phục vụ nhận dữ liệu vào của thiết bị thu thông tin ưu tiên cao hơn ngắt khi có yêu cầu đưa thông tin ra từ bộ vi xử lý.

Các lệnh ngắt có thể phát ra từ cả hai phần cứng và phần mềm.



**Hình 7.1. Cấu trúc ghép nối máy tính cơ sở với thiết bị ngoại vi**

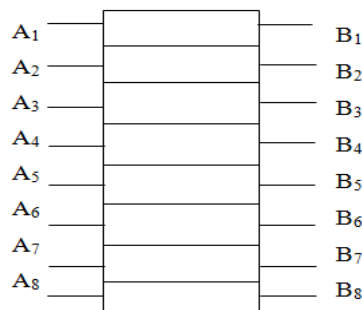
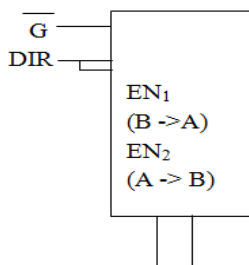
### 7.1.1. Mạch thu/phát đệm dữ liệu SN74LS245

Mạch thu/phát đệm kênh dữ liệu SN74LS245 là vi mạch thu/phát kênh hai chiều 8 bit. Vi mạch ba trạng thái này được dùng để đệm và điều khiển chiều chuyển động của số liệu trên kênh dữ liệu. Sơ đồ cấu trúc và bảng chân lý của vi mạch SN74LS245 được chỉ ra trên hình 7.2 [3].

Khi chân Enable  $\overline{G}$  có mức logic cao thì vi mạch ở trạng thái trở kháng cao. Khi  $\overline{G}$  xuống mức logic thấp thì số liệu sẽ di chuyển theo chiều do chân DIR (Direction) quyết định.

Nếu chân DIR ở mức logic thấp, thì số liệu đi từ lối vào B đến lối ra A. Ngược lại, nếu chân DIR ở mức logic cao, thì số liệu đi từ lối vào A đến lối ra B.

Trong hệ vi xử lý, vi mạch SN74LS245 thường dùng để kiểm soát chiều di chuyển của số liệu trên kênh dữ liệu. Khi chân DIR được nối với đường dây tín hiệu DT/ $\overline{R}$  thì nó dùng để quy định dữ liệu sẽ được CPU đưa ra hay thu về. Lối vào  $\overline{G}$  nối với đường tín hiệu DEN để kiểm soát việc nối ghép thông tin lối ra với kênh dữ liệu.



a. Sơ đồ cấu trúc vi mạch SN74LS245

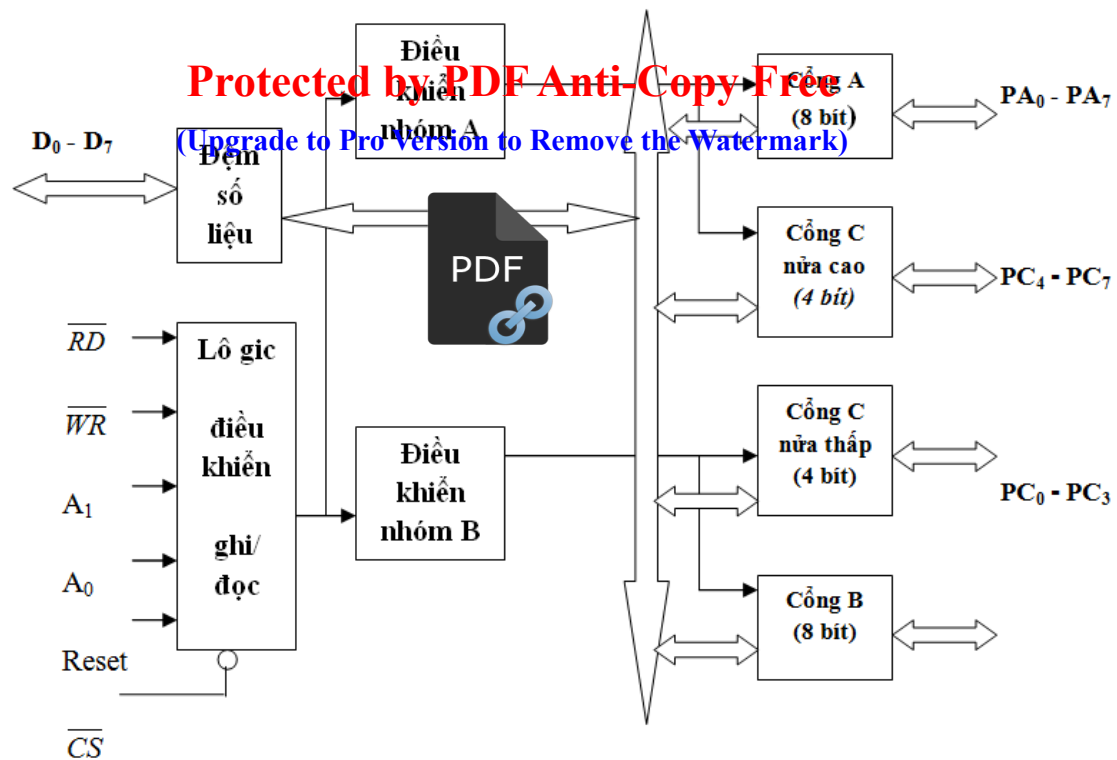
$\overline{G}$	DIR	OUT
L	L	B → A
L	H	A → B
H	X	Trở kháng cao

b. Bảng chân lý của vi mạch

**Hình 7.2. Sơ đồ cấu trúc và bảng chân lý của vi mạch SN74LS245**

### 7.1.2. Mạch tương thích với ngoại vi khả trình 8255A

Mạch tương thích với ngoại vi khả trình 8255A là vi mạch tương thích với các ngoại vi của hệ vi xử lý (PPI- Programmable Peripheral Interface) dùng để nối các hệ ngoại vi với máy vi tính. Vi mạch 8255A thực hiện chức năng tương hợp song song rất linh hoạt và được điều khiển bằng phần mềm.



**Hình 7.3. Sơ đồ khối của mạch tương thích với ngoại vi khả trình 8255A**

Về hướng CPU, vi mạch 8255A có các mạch đệm kênh dữ liệu hai chiều ( $D_0 \div D_7$ ), các tín hiệu kiểm soát ghi / đọc ( $\overline{RD}$ ,  $\overline{WR}$ ,  $A_0$ ,  $A_1$ , RESET,  $\overline{CS}$ ).

Về phía đầu ra của vi mạch 8255A có các cổng:

- Cổng A có các tín hiệu I/O:  $PA_0 \div PA_7$ .
- Cổng B có các tín hiệu I/O:  $PB_0 \div PB_7$ .
- Cổng C có các tín hiệu I/O:  $PC_0 \div PC_7$ .

Các cổng A, B là các cổng 8 bit 2 chiều.

Cổng C được chia thành 2 cổng, mỗi cổng 4 bit. Các bit cao từ  $PC_7 \div PC_4$  các bit thấp từ  $PC_3 \div PC_0$ . Các đường tín hiệu này dùng để di chuyển số liệu, các lệnh và thông tin trạng thái giữa CPU, vi mạch 8255A với các thiết bị ngoại vi.

Sự ấn định thời gian cho vi mạch 8255A trong việc di chuyển dữ liệu được điều khiển bằng tín hiệu kiểm soát đọc ( $\overline{RD}$ ) và ghi ( $\overline{WR}$ ). Các tín hiệu này cho phép CPU đọc thông tin ra từ vi mạch 8255A hay ghi thông tin vào vi mạch 8255A. Khi tín hiệu đọc có mức hiệu lực thấp thì CPU đọc dữ liệu hoặc thông tin trạng thái từ vi mạch 8255A ra qua kênh dữ liệu.



Khi tín hiệu ghi có mức hiệu lực thấp thì CPU ghi dữ liệu hoặc lời điều khiển vào vi mạch 8255A cũng thông qua Rẽ đầu dữ liệu.

Việc lựa chọn cổng được chỉ ra trong bảng các cổng của vi mạch 8255A như chỉ ra trong bảng 7.1.

**Bảng 7.1. Bảng xác định cổng chọn các cổng của vi mạch 8255A**

A <sub>1</sub>	A <sub>0</sub>	Cổng
0	0	A
0	1	B
1	0	C
1	1	Kiểm soát

Vi mạch 8255A có 3 chế độ hoạt động cơ bản là 0, 1 và 2.

- Chế độ 0: Chế độ vào/ra cơ sở.
- Chế độ 1: Chế độ vào/ra chốt (Strobe).
- Chế độ 2: Chế độ hoạt động 2 chiều.

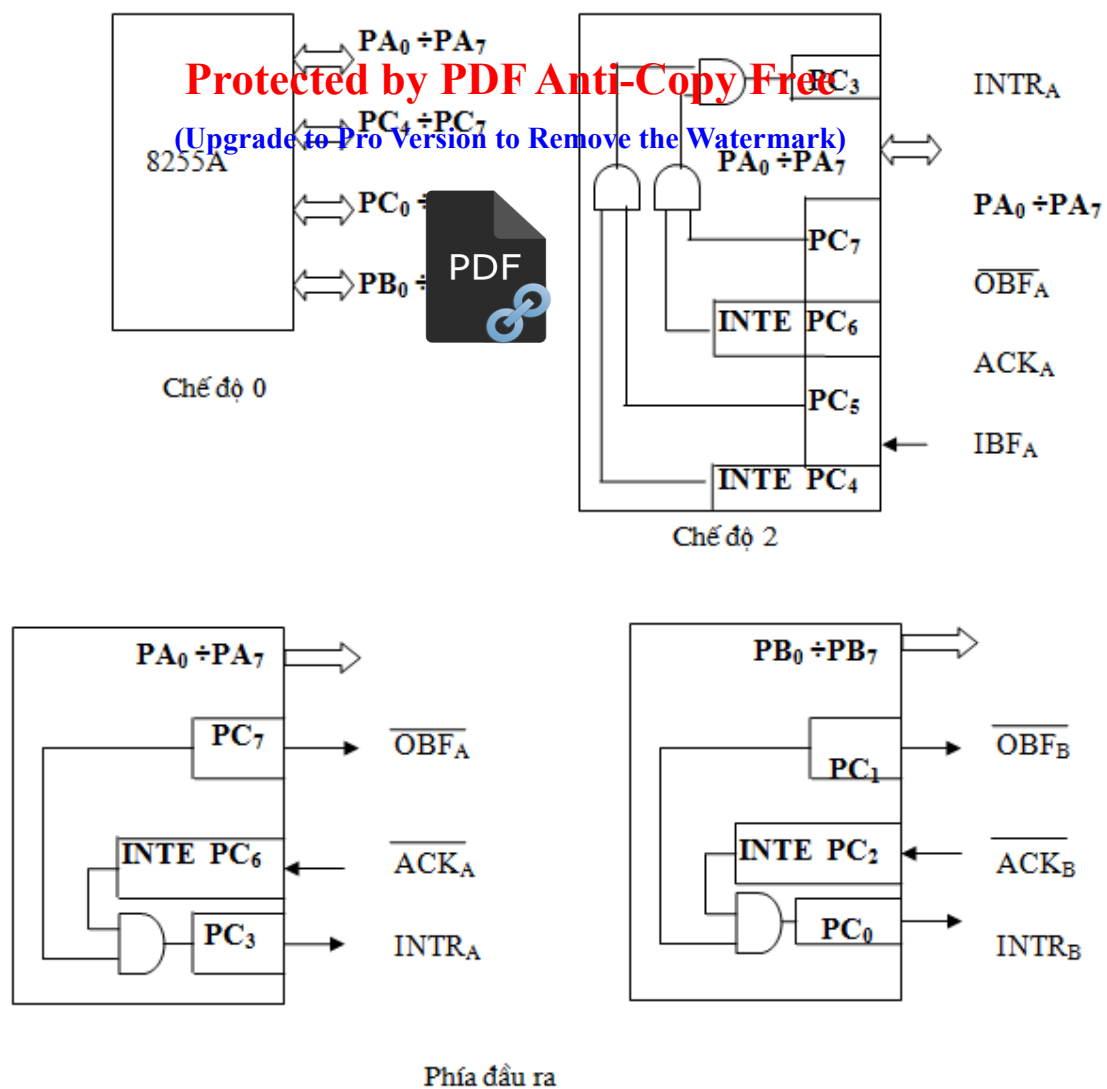
Trong chế độ 0: Vi mạch 8255A làm chức năng chốt, tức là dữ liệu được CPU đưa ra sẽ được giữ lại trong vi mạch 8255A cho đến khi nó chuyển trạng thái.

Chế độ 1: Dữ liệu chỉ được giữ lại trong vi mạch 8255A một khoảng thời gian ngắn, sau đó phải được chốt vào một mạch chốt ngoài nếu cần phải lưu lại số liệu.

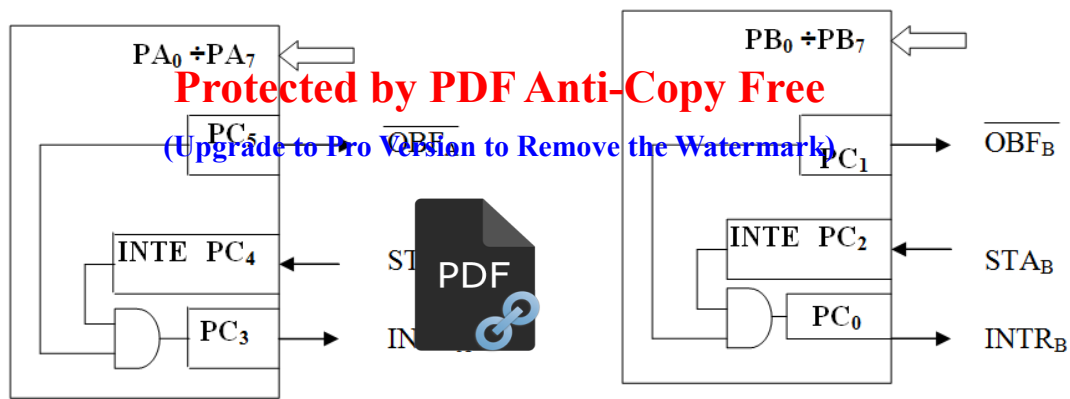
Chế độ 2: Là chế độ dùng cho hoạt động 2 chiều để trao đổi thông tin của hệ vi xử lý.

Các tín hiệu đối thoại và trạng thái của PPI 8255A ở các chế độ 1, 2, 3 được chỉ ra trên hình 7.4a và 7.4b.

Chú ý rằng: Cổng B và cổng C các bit thấp từ PC<sub>3</sub> ÷ PC<sub>0</sub> chỉ có thể hoạt động trong chế độ 0 hoặc chế độ 1, bởi vì chỉ có 1 bit dữ liệu để xác định chế độ hoạt động. Bit điều khiển cuối cùng D<sub>7</sub> là cờ thiết lập chế độ. Nó phải được đặt lên mức logic 1 khi chế độ hoạt động thay đổi.



**Hình 7.4a. Các mạch logic bên trong và các tín hiệu ở các chế độ 0 và 2 của PPI 8255A**



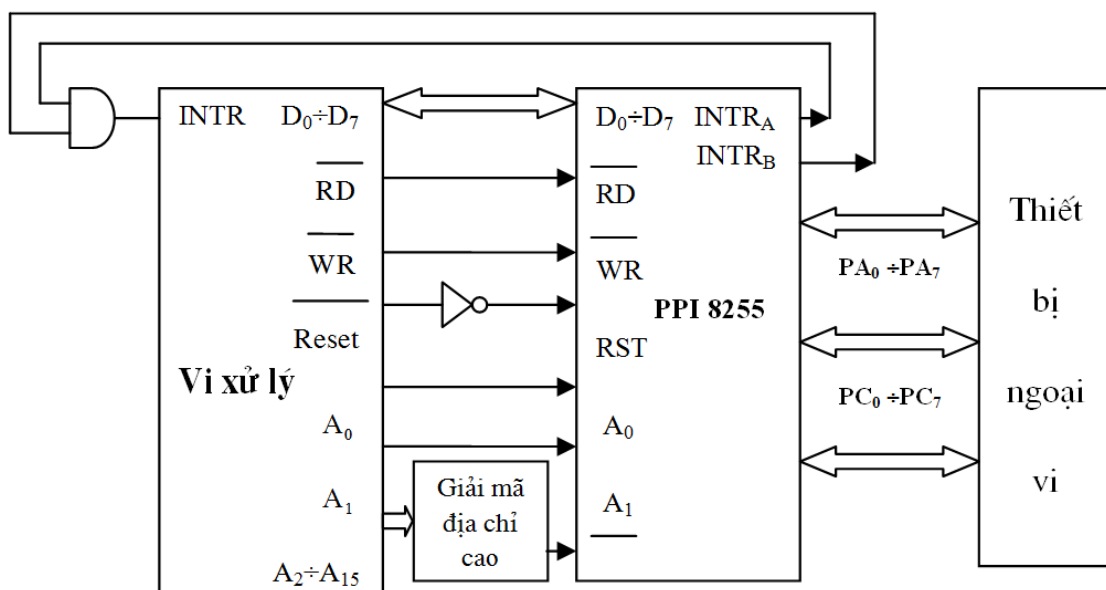
Phía đầu vào

**Hình 7.4b. Các mạch logic bên trong và các tín hiệu ở các chế độ 1 của PPI 8255A**

Sơ đồ ảnh hưởng nội cộng vào/ra theo chương trình với bộ vi xử lý và thiết bị ngoại vi khác nhau ra trên hình 7.5.

PPI 8255A khác biệt bộ vi xử lý và thiết bị ngoại vi, nó đảm vai trò trung chuyển thông tin giữa bộ vi xử lý với thiết bị ngoại vi qua các kênh thông tin của máy vi tính với thiết bị ngoại vi.

Thông thường khi ghép với ngoại vi thì các cổng A và B dùng để ghép nội trao đổi dữ liệu, cổng C dùng cho các thông tin điều khiển và trạng thái và chèn dữ liệu cho thiết bị.



Hình 7.5. Ghép nối giữa PPI 8255A với máy vi tính và thiết bị ngoại vi

**Protected by PDF Anti-Copy Free**  
**7.2. GIAO DIỆN TUẦN TỰ**  
(Upgrade to Pro Version to Remove the Watermark)

**Mạch kiểm soát ngắt khả trình 8259A (PIC - Programmable Interrupt Controller )**

Trong hầu hết các hệ thống nh đều có dùng một phương pháp cho phép mạch vào/ra tăng cường sự chú ý của CPU. Khi CPU nhận được một yêu cầu ngắt (INTR hoặc NMI) thì nó tiến hành tuần tự một loạt các bước để đáp ứng yêu cầu đó. Đầu tiên nó hoàn thành lệnh đang thực hiện, rồi quyết định có báo nhận cho yêu cầu ngắt hay không. Trường hợp yêu cầu ngắt không có mặt nạ che (NMI) thì nó bắt buộc phải báo nhận và phục vụ cho yêu cầu đó. Trong trường hợp yêu cầu ngắt bình thường (INTR) thì trước hết CPU tiến hành kiểm tra thanh ghi cờ để biết lệnh ngắt có bị che mặt nạ hay không. Sau đó nó sẽ phục vụ ngắt nếu cần thiết, nếu chưa cần thiết thì nó có thể bỏ qua thực hiện tiếp chương trình nó đang làm, sau khi hoàn thành chương trình mới chuyển sang phục vụ ngắt [5].

Khi CPU phục vụ ngắt, đầu tiên nó gửi cất nội dung của con trỏ lệnh và các thanh ghi đang được sử dụng có hiệu lực vào ngăn xếp (STACK). Tiếp theo là nó phục vụ ngắt bằng cách tìm địa chỉ cư trú của chương trình con phục vụ ngắt tương ứng được lưu trữ trong ROM. Chương trình con phục vụ ngắt là chương trình quy định các bước phải theo để phục vụ cho từng loại ngắt riêng biệt.

*Cấu trúc của PIC 8259A được chỉ ra trên hình 7.6a.*

Cấu trúc bên trong của mạch kiểm soát ngắt khả trình 8259A gồm 8 khối chức năng:

- Mạch đệm kênh dữ liệu để tương thích kênh dữ liệu với kênh bên trong của PIC 8259A. Mạch đệm 3 trạng thái 8 bit này là mạch trao đổi thông tin 2 chiều, nó được mở bằng khối logic ghi/đọc.

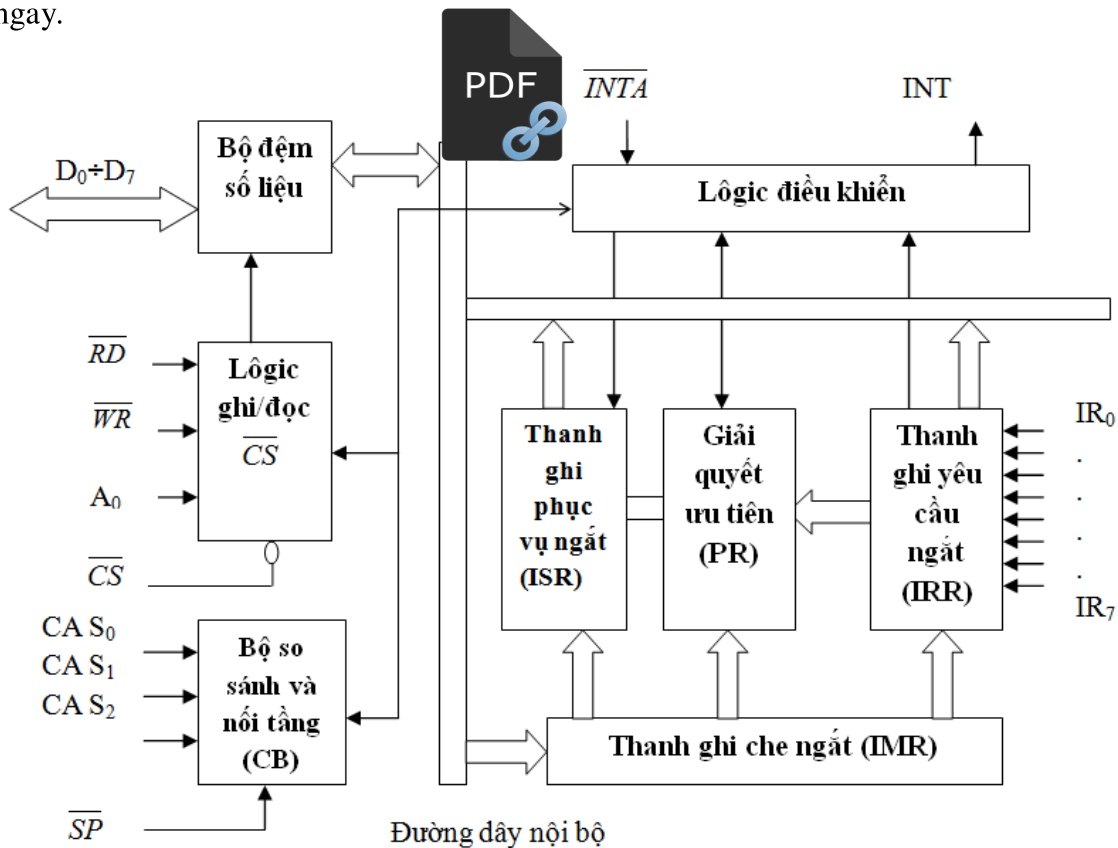
- Khối logic ghi/đọc sẽ cung cấp chiều, thời gian, nguồn hoặc nơi đến cho dữ liệu di chuyển qua khối mạch đệm của kênh dữ liệu.

Các tín hiệu điều khiển lối vào  $\overline{RD}$ ,  $\overline{WR}$ ,  $A_0$  và  $\overline{CS}$  sẽ kiểm soát khối logic ghi/đọc.

- Thanh ghi yêu cầu ngắt (IRR) lưu trữ trạng thái của những tín hiệu vào yêu cầu ngắt.

**Protected by PDF Anti-Copy Free**

- Thanh ghi phục vụ ngắt (ISR) dùng để lưu trữ mã ngắt sẽ được phục vụ ngay.



**Hình 7.6a. Sơ đồ khối của PIC 8259A**

- Mạch giải quyết ưu tiên (PR) dùng để quyết định tín hiệu ngắt nào có mức ưu tiên cao nhất.

- Khối so sánh và nối tầng (Cascade Buffer) dùng để tương thích giữa PIC 8259A chính và phụ trong các thao tác nối tiếp.

- Thanh ghi mặt nạ ngắt (IMR) dùng để che hoặc không che mặt nạ đối với các tín hiệu yêu cầu ngắt khác nhau.

Khối lôgic kiểm soát (điều khiển) dùng để sử dụng các thông tin do IRR, ISR và PR đưa vào để kiểm soát tín hiệu ngắt  $INT$  ở lối ra. Tín hiệu ngắt  $INT$  này sẽ yêu cầu ngắt đối với CPU. Khối lôgic kiểm soát này đồng thời cũng quản lý tín hiệu  $\overline{INTA}$  từ CPU đưa sang.

Các tín hiệu của PIC 8259A thực hiện các chức năng sau:

1. Vcc (Chân 28) - Nguồn cung cấp điện áp +5V.

2. GND (Chân 40) - Điểm nối đất của nguồn nuôi.

3.  $\overline{CS}$  (Chân 1) - Tín hiệu chọn chip, là tín hiệu vào hiệu lực thấp để mở PIC 8259A.

4.  $\overline{WR}$  (Chân 2) - Tín hiệu vào hiệu lực thấp. Tín hiệu này liên kết với  $\overline{CS}$  sẽ mở PIC 8259A để nhận các lời lệnh từ CPU đưa sang.

5.  $\overline{RD}$  (Chân 3) - Tín hiệu đọc, là tín hiệu vào hiệu lực thấp tín hiệu này cùng với tín hiệu  $\overline{CS}$  sẽ mở mạch PIC 8259A để đưa thông tin trạng thái lên kênh dữ liệu của CPU.

6.  $CAS_0 \div CAS_2$  (Các chân 12, 13, 15), là các đường dây mắc nối tiếp (Cascade) này sẽ hình thành sự kiểm soát kênh, dùng với hệ sử dụng nhiều mạch PIC 8259A ghép với nhau để tăng số lượng các ngoại vi yêu cầu ngắt.

7.  $\overline{SP} / \overline{EN}$  (Chân 16). Đây là chân 2 chức năng: Chương trình phụ (Slave Program) và mở mạch đệm (Enable Buffer). Chân này được dùng làm đầu ra để kiểm soát các mạch thu/phát đệm trong chế độ mạch đệm (EN) nó được làm đầu vào để mắc nối tiếp các mạch PIC 8259A trong chế độ SP.

8. ITR (Chân 17) - Tín hiệu ngắt, là tín hiệu ra hiệu lực cao, dùng để ngắt CPU.

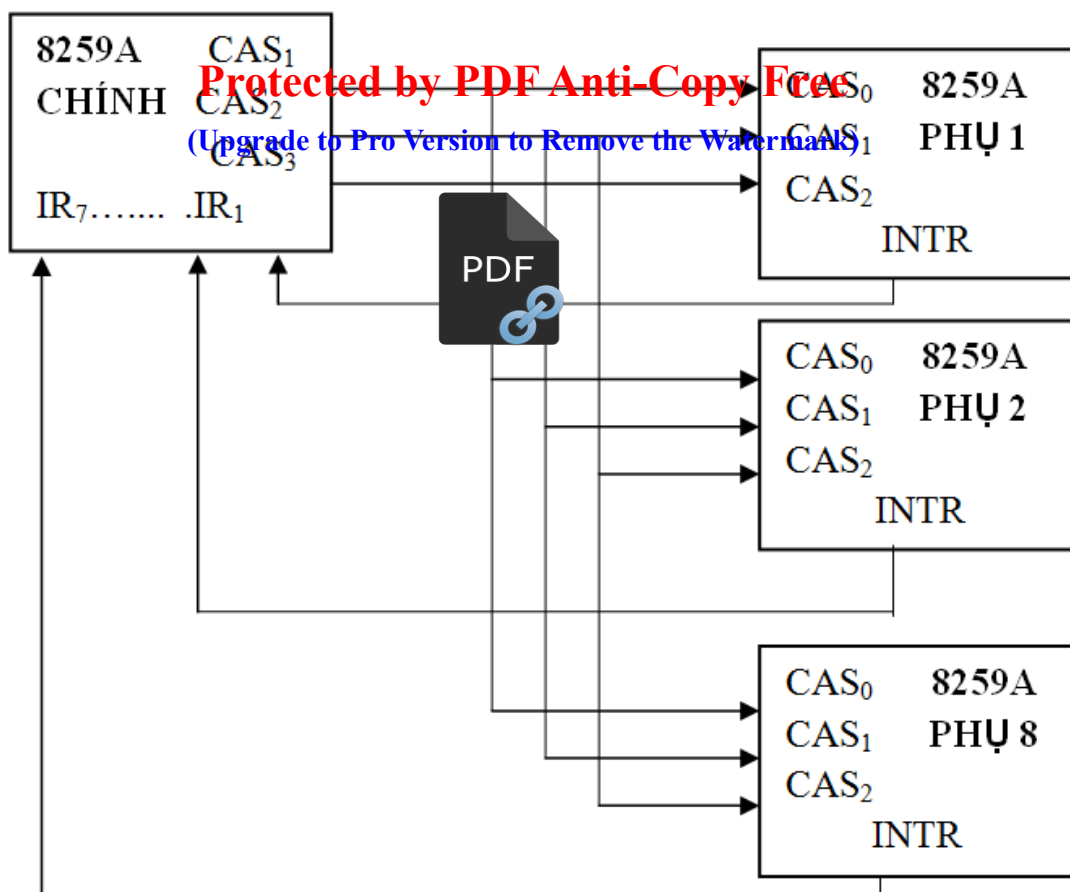
9.  $IR_0 \div IR_7$  (Chân 18 ÷ 25) - Các tín hiệu yêu cầu ngắt (Interrupt Request), là các tín hiệu vào hiệu lực cao, không đồng bộ, dùng để thông báo yêu cầu ngắt từ một ngoại vi.

10.  $\overline{INTA}$  (chân 26) - Tín hiệu báo nhận ngắt (Interrupt Acknowledge), là tín hiệu vào hiệu lực thấp từ CPU, dùng để báo rằng CPU sắp tiến hành phục vụ ngắt.

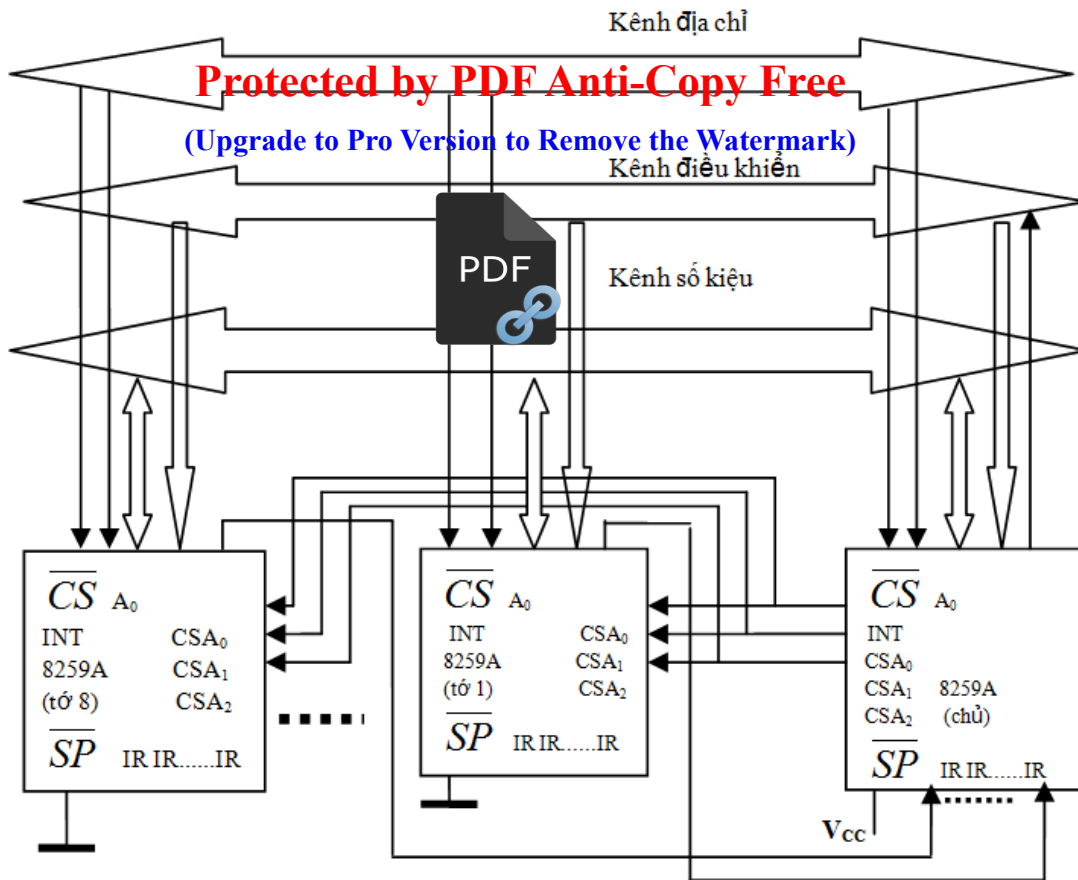
11.  $A_0$  (Chân 27) - Đường địa chỉ  $A_0$  cùng với các tín hiệu chọn chip, ghi và đọc, dùng để chọn các lời lệnh khác nhau cho vi mạch 8259A.

12.  $D_7 \div D_0$  (Các chân 4 ÷ 11) - Các tín hiệu kênh dữ liệu hai chiều này dùng cho việc di chuyển, kiểm soát, trạng thái và thông tin vector ngắt.

Cấu trúc ghép nối các vi mạch 8259A với nhau theo cơ chế chính-phụ (chủ - tớ) khi dùng nhiều vi mạch nối ghép với nhau để dùng cho nhiều ngoại vi được chỉ ra trên hình 7.6 b và c.



Hình 7.6b. Sơ đồ ghép nối các vi mạch 8259A với nhau trong máy vi tính để tăng số lượng thiết bị ngoại vi (Phương pháp nối các chân điều khiển)



**Hình 7.6c. Sơ đồ ghép nối các vi mạch 8259A với nhau trong máy vi tính để tăng số lượng thiết bị ngoại vi (Phương pháp nối tín hiệu và các kênh thông tin)**

Vi mạch 8259A chủ chịu trách nhiệm kiểm soát các vi mạch 8259A tổ khác thông qua tín hiệu INT của vi mạch tổ đưa về các chân IR<sub>i</sub> tương ứng. Chân  $\overline{SP}$  của vi mạch chủ được nối với V<sub>cc</sub>, còn chân  $\overline{SP}$  của các vi mạch tổ được nối với đất (GND). Các chân CAS<sub>0</sub> ÷ CAS<sub>2</sub> cũng được nối song song với nhau.

Các yêu cầu ngắt của các thiết bị ngoại vi khác nhau, có các mức ưu tiên khác nhau được chỉ ra trên bảng 7.2.



**Bảng 7.2. Các giá trị đọc của các mức ưu tiên**

**Protected by PDF Anti-Copy Free**  
(Upgrade to Pro Version to Remove the Watermark)

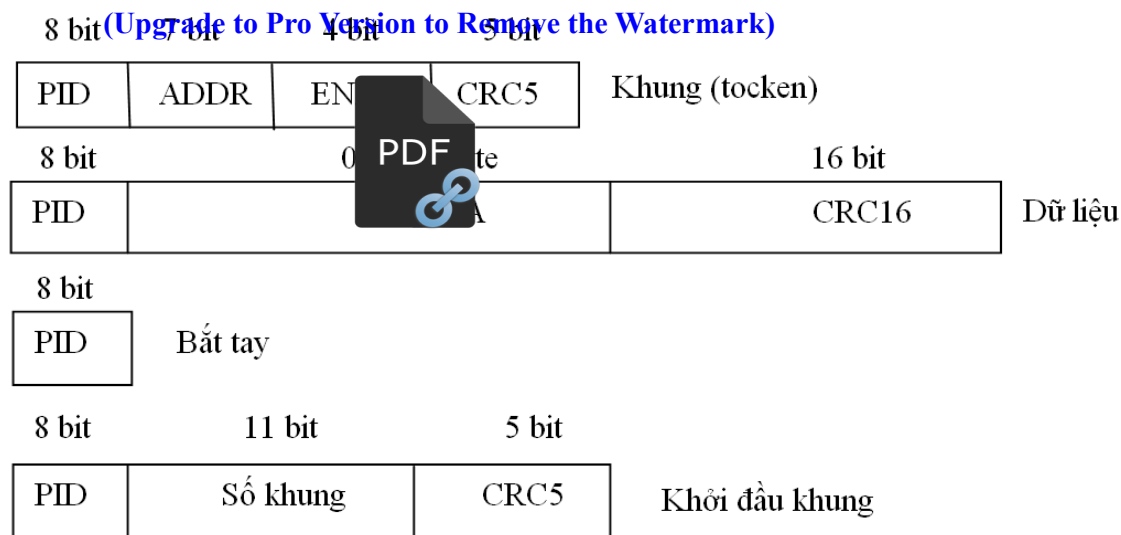
Yêu cầu ưu tiên		RST	D3	D2	D1	D0	A3	A2	A1	A0
Thấp nhất ↓      Cao nhất	0	7	1	1	1	1	1	1	1	1
	1	6	1	1	1	1	0	1	1	1
	2	5	1	1	1	0	1	1	0	1
	3	4	1	1	1	0	0	1	1	1
	4	3	1	1	0	1	1	1	1	1
	5	2	1	1	0	1	0	1	1	1
	6	1	1	1	0	0	1	1	1	1
	7	0	1	1	0	0	0	1	1	1

### 7.3. GIAO DIỆN ĐA NĂNG USB

Công nghệ USB (Universal Serial Bus) trong máy vi tính cá nhân đáp ứng nhu cầu một giao diện đơn giản, linh hoạt, dễ sử dụng và rẻ tiền. USB là một giao thức truyền dữ liệu tuần tự giữa máy vi tính (hay chủ USB) với các thiết bị ngoại vi. USB là một tiêu chuẩn và được xác định dựa vào "tín hiệu tốc độ" để giao tiếp. Theo lý thuyết, tốc độ chuẩn của USB 2.0 tối đa là 480 Mbps, tức 60MB/s. Trong khi chuẩn USB 3.0 (mới nhất hiện nay) được xác định với tốc độ tối đa là 4.8 - 5 Gbps, tức 600 - 625MB/s. Như vậy, về mặt lý thuyết USB 3.0 nhanh hơn USB 2.0 hơn 10 lần. Một ưu điểm của USB là tính năng cắm là chạy (người sử dụng có thể cắm thêm hoặc tháo ra một thiết bị ngoại vi mà không cần tắt máy chủ hay cài đặt lại hệ thống).

#### Giao thức USB:

Một lần truyền tin USB cần đến 3 gói: Gói khung (token packet), gói dữ liệu (data packet), gói bắt tay (handshake packet).



**Hình 7.7. Cấu trúc giao thức USB**

*PID: loại gói (packet identification);*

*ADDR: địa chỉ (address);*

*ENP: điểm cuối (endpoint);*

*CRC: mã kiểm tra quay vòng dư (cyclic redundancy code);*

*DATA: dữ liệu*

### **Mạch định thời gian khả trình 8253 (Programmable Interval Time / Counter)**

Mạch định thời gian khả trình 8253 có công dụng chính là thực hiện chức năng định thời gian, đếm, cũng như là phát các trị số biến, phát tín hiệu đồng bộ, kiểm soát one-shot, đếm sự kiện, phát tần số và điều khiển mô tơ.

Mạch định thời gian khả trình 8253 gồm có 3 bộ đếm 16 bit, có khả năng định trước chương trình hoạt động.

Các bộ đếm 16 bit đều có khả năng đặt trước chế độ đếm ngược (giảm). Chúng có thể hoạt động trong chế độ đếm nhị phân hoặc BCD.

Sơ đồ khối của mạch định thời gian khả trình 8253 được chỉ ra trên hình 7.8.

Mạch định thời gian lập trình hoá 8253 có thể chia ra làm 3 phần:

- Phần lõi vào chứa các mạch đếm ra số liệu, mạch kiểm soát logic đọc/ghi và thanh ghi lời điều khiển.

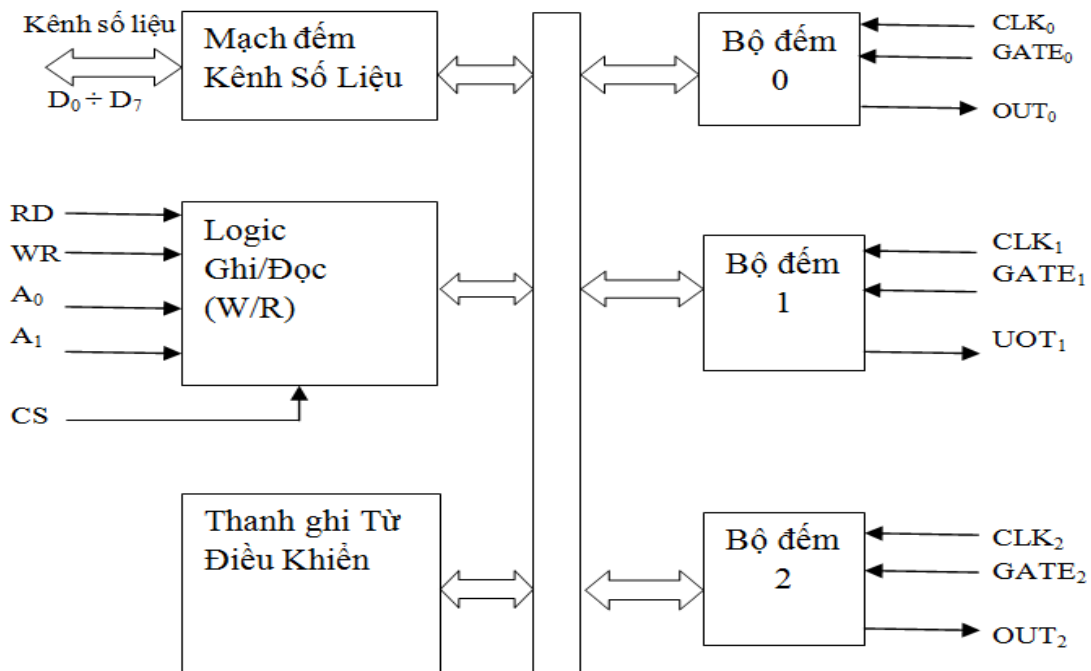
- Phần kernel bộ đếm và thanh ghi điều khiển giữa lõi vào và lõi ra.

- Phần lõi ra có 3 bộ đếm riêng biệt nhau.

- Các chân số liệu:  $D_0 \div D_7$  tín hiệu hai chiều, ba trạng thái nối với các mạch đếm kênh dữ liệu. Các mạch đếm có thể phát, hoặc thu dữ liệu bằng lệnh OUT, hoặc IN của CPU. Các đường tín hiệu này dùng để đặt chế độ hoạt động cho mạch định thời gian khả trình 8253, đặt giá trị đầu tiên cho các bộ đếm và đọc dữ liệu ở bộ đếm đầu ra (các chân  $1 \div 8$ ).

- Chân  $\overline{RD}$  (Chân 22)- Chân đọc, là tín hiệu vào hiệu lực thấp dùng để báo cho mạch định thời gian khả trình 8253 biết rằng CPU đang đưa dữ liệu ra.

- Các chân  $A_0/A_1$  (Các chân 19 và 20) - Các tín hiệu của các chân địa chỉ này dùng để chọn một trong ba bộ đếm hoặc thanh ghi lời điều khiển.



**Hình 7.8. Sơ đồ cấu trúc của mạch định thời gian khả trình 8253**

Việc chọn các bộ đếm và thanh ghi lời điều khiển được chỉ ra trên bảng 7.3.

Thanh ghi từ điều khiển dùng để định chương trình và khởi đầu cho mạch định thời gian khả trình 8253. Khi cả hai chân  $A_0$  và  $A_1$  đều có mức logic 1 thì thanh ghi

này được chọn. Sau đó có khả năng nhận thông tin từ các mạch đếm kênh dữ liệu. Thông tin này được lưu trữ trong thanh ghi điều khiển và được dùng để chọn lựa chế độ hoạt động của từng bộ đếm. Mỗi bộ đếm của vi mạch định thời gian khả trình 8253 được định chương trình riêng bằng cách ghi một từ điều khiển vào thanh ghi này.

**Bảng 7.3. Chọn các thanh ghi hoặc thanh ghi lời điều khiển**

A <sub>1</sub>	A <sub>0</sub>	Chức năng
0	0	Chọn bộ đếm 0
0	1	Chọn bộ đếm 1
1	0	Chọn bộ đếm 2
1	1	Chọn thanh ghi lời điều khiển

Dạng của lời điều khiển (kiểm soát) như sau:

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
SC <sub>1</sub>	SC <sub>0</sub>	RL <sub>1</sub>	RL <sub>0</sub>	M <sub>2</sub>	M <sub>1</sub>	M <sub>0</sub>	BCD

Các bit lời điều khiển này có các chức năng được miêu tả trên bảng 7.4, 7.5, 7.6. Các bit SC (Select Counter) là các bit chọn bộ đếm. Các bit RL (Read/Load) là các bit đọc/ nạp số liệu. Các bit M (Mode) là các bit xác định chế độ.

**Bảng 7.4. Chức năng của các bit chọn bộ đếm**

D <sub>7</sub> (SC <sub>1</sub> )	D <sub>6</sub> (SC <sub>0</sub> )	Chức năng
0	0	Chọn bộ đếm 0
0	1	Chọn bộ đếm 1
1	0	Chọn bộ đếm 2
1	1	Trái luật

**Bảng 7.5. Chức năng của các bit đọc/nạp số liệu**

D <sub>5</sub> (RL <sub>1</sub> )	D <sub>4</sub> (RL <sub>0</sub> )	Chức năng
0	0	Thao tác chốt bộ đếm
0	1	Đọc/nạp byte thấp
1	0	Đọc/nạp byte cao
1	1	Đọc/nạp byte thấp trước, rồi đến byte cao

$D_3(M_2)$	$D_2(M_1)$	$D_1(M_0)$	Chức năng
0		0	Chế độ 0
0		1	Chế độ 1
x		0	Chế độ 2
x	1	1	Chế độ 3
1	0	0	Chế độ 4
1	0	1	Chế độ 5

Ghi chú

Giá trị x là trạng thái không xác định (bít đó không cần quan tâm).

Chế độ 0: Ngừng ở số đếm cuối; Lỗi ra có mức cao khi kết thúc đếm.

Chế độ 1: Đếm từng bít theo các chương trình.

Chế độ 2: Mạch phát các trị số.

Chế độ 3: Mạch phát trị số xung vuông.

Chế độ 4: Trgger chốt (Strobe trigger) phần mềm.

Chế độ 5: Trgger chốt (Strobe trigger) phần cứng.

- Chân  $\overline{CS}$  (Chân 21) - Tín hiệu chọn chip, là tín hiệu vào hiệu lực thấp, dùng để mở mạch logic điều khiển. Do vậy tín hiệu này sẽ hiệu lực hoá cho các tín hiệu  $\overline{RD}$ ,  $\overline{WR}$ ,  $A_0$  và  $A_1$ .

-  $CLK_0$ ,  $CLK_1$ ,  $CLK_2$  (Các chân 9, 14, 18). Đây là các tín hiệu đồng bộ lỗi vào, ứng với đầu vào thời gian của các mạch đếm.

-  $OUT_0$ ,  $OUT_1$ ,  $OUT_2$  (Các chân 10, 13, 17). Đây là các tín hiệu ra riêng biệt ứng với các mạch đếm.

-  $GATE_0$ ,  $GATE_1$ ,  $GATE_2$  (Các chân 11, 14, 16). Đây là các tín hiệu cổng lỗi vào, hiệu lực cao, dùng để mở các mạch đếm.

#### 7.4. GIAO DIỆN CAO TỐC IEEE 1394

IEEE 1394 là một chuẩn giao tiếp với băng thông cao do IEEE (Institute of Electrical and Electronic Engineers) công bố vào cuối năm 1995 (theo thứ tự công bố chuẩn thứ 1394 như một sự tình cờ hoặc là lý do để chuẩn này được đặt tên như vậy).

IEEE 1394 cũng còn được biết đến với tên khác như: FireWire (hãng Apple) i.LINK (hãng Sony). IEEE 1394 không phải là một loại cổng, chúng chỉ là một chuẩn giao tiếp để các hãng phần cứng khác phát triển ra các cổng giao tiếp dựa trên chuẩn này nếu được chấp nhận rộng rãi [6].

Tiêu chuẩn chung nhất của IEEE 1394 là IEEE 1394a hoặc IEEE 1394a-2000 với con số 2000 là năm mà chuẩn này được giới thiệu. Chuẩn IEEE 1394b được giới thiệu vào đầu năm 2003, chuẩn này hỗ trợ băng thông lên đến 800 Mbps và còn có khả năng mở rộng lên 3.200 Mbps trong tương lai. IEEE 1394b có tốc độ cao hơn các chuẩn IEEE 1394/IEEE 1394a bởi vì chúng hỗ trợ các công nghệ mạng bằng cáp quang và các cáp theo Category 5 UTP. IEEE 1394b hoàn toàn tương thích ngược với các thiết bị theo chuẩn IEEE 1394a.

IEEE 1394a hiện tại hỗ trợ các mức băng thông 100 Mbps, 200 Mbps, và 400 Mbps (tương ứng 12,5 MBps, 25 MBps, và 50 MBps). IEEE 1394 cho phép kết nối đồng thời đến 63 thiết bị bằng các hình thức phân nhánh. IEEE 1394a dùng cáp 6 sợi, trong đó 4 sợi cho truyền tín hiệu, 2 sợi cho cung cấp nguồn điện. Tuy nhiên một loại đầu cắm nhỏ hơn dùng cho các thiết bị tự cung cấp năng lượng chỉ có 4 sợi, trong đó không bố trí 2 sợi cung cấp điện năng. Các DV camcorder thường sử dụng loại giao tiếp IEEE 1394 có 4 sợi bởi chúng tự cung cấp năng lượng qua pin hoặc có nguồn điện riêng.

IEEE 1394b là thế hệ thứ 2 của chuẩn IEEE 1394 với những ứng dụng đầu tiên vào năm 2003. IEEE 1394b có 9 chân, hỗ trợ tốc độ truyền 800/3200 Mbps nên cao hơn, nó có các cải tiến sau so với thế hệ trước nó (IEEE 1394a):

- Tự sửa chữa lỗi (Self-healing loops)
- Hỗ trợ các cáp dài hơn.
- Hỗ trợ cáp CAT5 cũng như cáp quang.

IEEE 1394b có thể giao tiếp với nhiều loại thiết bị có sử dụng các chuẩn giao tiếp theo chuẩn này thông qua các loại cáp chuyển đổi số chân cắm: 9 chân -> 6 chân hoặc 4 chân để phù hợp với các thiết bị sử dụng các cổng giao tiếp theo chuẩn IEEE 1394a.

Các loại cáp sử dụng với IEEE 1394b bao gồm:

- Beta: Chỉ dùng riêng với IEEE 1394b (không tương thích với IEEE 1394a)

**Protected by PDF Anti-Copy Free**

- ~~Blue~~ Upgrade: Cấp loại này dùng đồng thời ~~White~~ (bị IEEE 1394a/b, giúp các cổng theo IEEE 1394b sử dụng tương thích ngược với các thiết bị theo chuẩn thể hệ trước nó. Có hai loại: 9 chân vào 9 chân và 9 chân ra 4 chân giúp việc tương thích giữa hai chuẩn a/b.



Ở các máy tính cá nhân phổ thông có giá thành thấp, chuẩn này chưa được đưa vào sử dụng bởi chúng làm tăng giá thành sản phẩm. Ở các hệ máy tính tầm trung và cao cấp chúng đã được tích hợp sẵn vào bo mạch chủ.


Nhiều bo mạch chủ bán rời cho các người dùng tự lắp ráp máy tính cũng được tích hợp các cổng IEEE 1394 ở dòng sản phẩm trung và cao cấp (đa số chúng có giá lớn hơn 100 USD ở thời điểm cuối năm 2007).

Một số bo mạch âm thanh thuộc dòng cao cấp cũng được tích hợp sẵn các cổng IEEE 1394, ví dụ các bo mạch âm thanh của Creative cao cấp thường tích hợp sẵn IEEE 1394.

Cũng giống như các giao tiếp khác có nhu cầu sử dụng trên các máy tính cá nhân nhưng không được tích hợp sẵn, một số nhà sản xuất phần cứng đã sản xuất các bo mạch cung cấp các cổng I/O như IEEE 1394. Chúng thường được gắn vào các khe PCI trong máy tính. Không những phục vụ cho các máy tính cá nhân để bàn, nếu các máy tính xách tay chưa được tích hợp sẵn các cổng giao tiếp theo chuẩn IEEE 1394 thì có thể sử dụng các PCMCIA card để mở rộng ra các cổng theo chuẩn IEEE 1394.

Do sự các thiết bị cần băng thông cao chưa phổ biến trong người tiêu dùng nên các cổng IEEE 1394 chưa được thôi thúc tích hợp sẵn trên tất cả các máy tính cá nhân. Đa số các thiết bị ngoại vi hiện nay chỉ khai thác đến bus USB như: ổ usb flash, bàn phím, chuột..., giao tiếp với các máy ảnh số do tốc độ truy cập dữ liệu còn chậm nên cũng chỉ sử dụng USB. Mặt khác USB 2.0 với băng thông 480 Mbps cũng đáp ứng khá tốt cho các thiết bị kể trên.

IEEE 1394 mới chỉ được khai thác ở các ứng dụng cần băng thông lớn, đặc biệt là (một số trong chúng vẫn chưa phổ biến với chuẩn này):

DV cameras: Hầu hết các DV camera (digital camera) ngày nay được tích hợp sẵn một cổng IEEE 1394 (còn IEEE 1394 thì quen được gọi là cổng DV hay DV port). Sự lựa chọn giao tiếp với chuẩn IEEE 1394 bởi chúng đáp ứng với băng thông cho video. Nhiều phần mềm, tiện ích kèm theo các DV camera này được phát triển trên nền chuẩn IEEE 1394 như: chuyển video từ DV cam sang PC, convert các định dạng video trực tiếp theo thời gian thực...  


- Các ổ đĩa cứng gắn ngoài và các thiết bị lưu trữ khác gắn ngoài khác yêu cầu băng thông cao, như: ổ đĩa quang (ổ CD, ổ DVD).
- Các máy quét cao cấp với độ phân giải cao.
- Kết nối các máy tính với nhau với băng thông cao.
- Trong tương lai, khi mà nhiều thiết bị sử dụng các chuẩn IEEE 1394 thì chúng có thể thay thế các giao tiếp USB đang phổ biến hiện nay cũng như USB đã thay thế các cổng tuần tự và song song thế hệ trước nó.

## CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 7

**Câu 1:** Nêu các đặc điểm của mạch thu/phát đệm dữ liệu SN74LS245 ?

**Câu 2:** Nêu các đặc điểm của mạch tương thích với ngoại vi chương trình hóa 8255a ?

**Câu 3:** Vẽ sơ đồ, nêu chức năng và nhiệm vụ của mạch kiểm soát ngắt sử dụng chương trình hoá 8259A ?

**Câu 4:** Vẽ sơ đồ, nêu chức năng và nhiệm vụ của mạch định thời gian lập trình hoá 8253 ?



## TÀI LIỆU THAM KHẢO

- Protected by PDF Anti-Copy Free  
(Upgrade to Pro Version to Remove the Watermark)
1. J.L. Hennessy, D.A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 4th Edition, 2006.
  2. Hồ Khánh Lâm, *Kỹ thuật vi xử lý (tập 1 và tập 2)*, NXB Thông tin và truyền thông, 2008.
  3. Li-Shiuan Peh, Abhinav Arora, Elliott Fleming, Sang Woo Jun, Asif Khan, Myron King (MIT); Derek Chiou and Jihong Kim, *Computer architecture*, Seoul National University, 2012.
  4. Mostafa Abd-El-Barr and Hesham El-Rewini, *Fundamentals of Computer Organization and Architecture*, John Wiley & Sons, Inc, 2005.
  5. Nguyễn Đình Việt, *Kiến trúc máy tính*, Nhà xuất bản Đại học Quốc gia Hà Nội, 2006.
  6. Trần Quang Vinh, *Cấu trúc máy vi tính*, Nhà xuất bản Đại học Quốc gia Hà Nội, 2007.
  7. William Stallings, *Computer Organization and Architecture, Designing for Performance*, 8<sup>th</sup> Edition, Prentice Hall, 2009.