TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HỒ CHÍ MINH

KHOA ĐIỆN

# VI XỬ LÝ

## Chương 1: Giới thiệu chung về vi điều khiển

1.1 Vài nét về họ vi điều khiển 8051

1.2 Hệ thống máy tính:

1.3 Đơn vị xử lý trung tâm (CPU):

1.4 Bộ nhớ bán dẫn RAM và ROM:
1.4.1. Nguyên tắc vận hành của bộ nhớ:
1.4.2. Cấu trúc ROM:
1.4.3. Phân loại một số loại ROM:
1.4.4. Cấu trúc RAM:
1.4.5. Phân loại một số loại RAM:
1.4.6. Cách xác định dung lượng bộ nhớ bán dẫn 8 bit

1.5. Hệ thống bus:
1.5.1. Bus địa chỉ (address bus)
1.5.2. Bus dữ liệu (data bus)
1.5.3. Bus điều khiển (Control bus)

## Chương 2: Giới thiệu phần cứng 89C51

2.1 Họ MCS51:

**2.1.1** Tóm tắt phần cứng họ MSC51(8951):

**2.1.2** Khảo sát cấu trúc bên trong của vi điều khiển 89C51:

2.1.2.1 Sơ đồ cấu trúc bên trong của vi điều khiển:

**2.1.3** Các phiên bản của 8051 từ Atmel (Flash ROM)

2.2 Khảo sát sơ đồ chân 89C51

2.2.1 Port 0

2.2.2 Port 1

2.2.3 Port 2

2.2.4 Port 3

2.2.5 Các tín hiệu điều khiển

2.2.6 Tổ chức bộ nhớ

2.3 Mở rộng port cho 89C51

## Chương 3: Tập lệnh 89C51

3.1 Các kiểu định địa chỉ

**3.2.1** Kiểu định địa chỉ dùng thanh ghi (Register Addressing) :

**3.2.2** Kiểu định địa chỉ trực tiếp (Direct Addressing) :

**3.2.3** Định địa chỉ gián tiếp thanh ghi (indirect Addressing):

# Chương 1: Giới thiệu chung về vi điều khiển

## 1.1 Vài nét về họ vi điều khiển 8051

Vào năm 1971 tập đoàn Intel đã giới thiệu 8080, bộ vi xử lý (micro-processor) thành công đầu tiên. Sau đó không lâu Motorola, RCA, kế đến là MOS Techlogy và Zilog đã giới thiệu các bộ vi xử lý tương tự : 6800, 1801, 6502 và Z80. Vào năm 1976 Intel đã giới thiệu bộ vi điều khiển (microcontroller) 8748, một chip tương tự như các bộ vi điều khiển và là chip đầu tiên trong họ vi điều khiển MCS-48. 8748 là vi mạch chứa trên 17000 transistor bao gồm một CPU, 1 K byte EPROM, 64 byte RAM, 27 chân xuất nhập và một bộ định thời 8-bit. IC này và các IC khác tiếp theo của họ MCS-48 đã nhanh chóng trở thành chuẩn công nghiệp trong các ứng dụng hướng điều khiển (control–oriented application). Việc thay thế các thành phần cơ điện trong các sản phẩm như các máy giặt và các bộ điều khiển được tìm thấy bao gồm xe ô tô, thiết bị công nghiệp, các sản phẩm tiêu dùng và các thiết bị ngoại vi của máy tính (bàn phím của IBM-PC là một thí dụ sử dụng bộ vi điều khiển trong các thiết kế tối thiểu thành phần...).

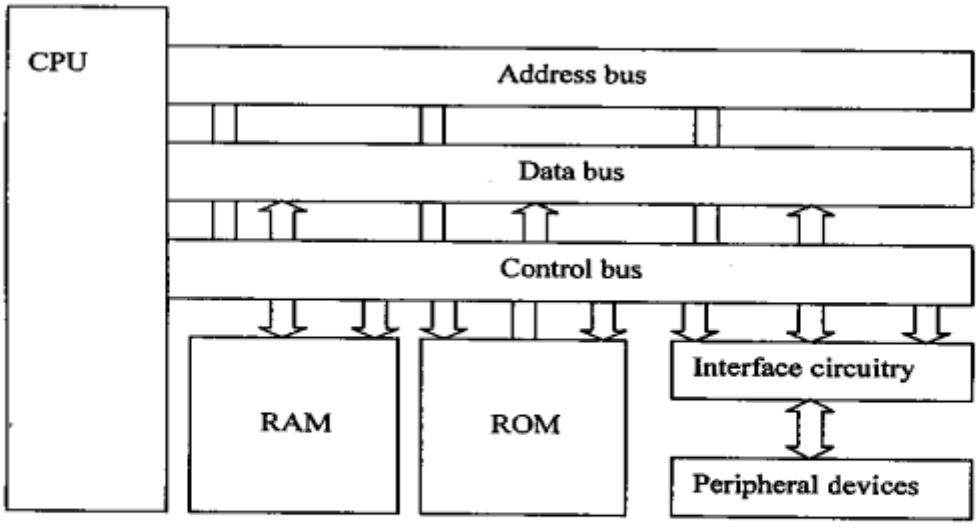
Độ phức tạp, kích thước và khả năng của các bộ vi điều khiển được tăng thêm một bậc quan trọng vào năm 1990 khi Intel công bố chip 8051, bộ vi điều khiển đầu tiên của họ vi điều khiển. So với 8048, chip 8051 chứa trên 60000 transistor bao gồm 4 K byte ROM, 128 byte RAM, 32 đường xuất nhập, 1 port nối tiếp và 2 bộ định thời 16- bit một số lượng mạch đáng chú ý trong một IC đơn. Các thành viên mới được thêm vào cho họ MCS-51 và các biến thể ngày nay có gấp đôi các đặc trưng này. Tập đoàn Siemens, nguồn sản xuất thứ hai các bộ vi điều khiển thuộc họ MCS-51 cung cấp chip SAB80515, một cải tiến của 8051 chứa trong một vỏ 68 chân, có 6 port xuất nhập 8 bit, 13 nguồn tạo ra ngắt và một bộ biến đổi A/D 8 bit với 8 kênh ngõ vào. Họ 8051 là một trong những bộ vi điều khiển 8-bit mạnh và linh hoạt nhất, đã trở thành bộ vi điều khiển hàng đầu trong những năm gần đây.

## 1.2 Hệ thống máy tính:

Một máy tính được định nghĩa bởi 2 ý chính:

- Khả năng được lập trình để thao tác trên dữ liệu mà không cần đến sự can thiệp của con người.
- Khả năng lưu trữ và khôi phục dữ liệu
- Bên cạnh đó, một hệ máy tính cũng bao gồm các thiết bị ngoại vi để truyền thông với con người cũng như các chương trình để xử lý dữ liệu.

Một hệ máy tính bao gồm một đơn vị xử lý trung tâm CPU (Central Processing Unit), đơn vị này kết nối với bộ nhớ truy xuất ngẫu nhiên RAM (Random Access Memory) và bộ nhớ chỉ đọc ROM (Read Only Memory) thông qua bus địa chỉ (address bus), bus dữ liệu (data bus) và bus điều khiển ( control bus). Các mạch giao tiếp kết nối các bus của hệ thống với các thiết bị ngoại vi.




### 1.3 Đơn vị xử lý trung tâm (CPU):

CPU quản lý tất cả các hoạt động của hệ và thực hiện tất cả các thao tác trên dữ liệu.

- Hầu hết các CPU chỉ bao gồm 1 tập các mạch logic thực hiện liên tục 2 thao tác: tìm nạp lệnh và thực thi lệnh.
- CPU có khả năng hiểu và thực thi các lệnh dựa trên 1 tập các mã nhị phân, mỗi một mã nhị phân biểu thị 1 thao tác đơn giản. Các lệnh số học, các lệnh logic, các lệnh di chuyển dữ liệu hoặc các lệnh rẽ nhánh được biểu thị bởi 1 tập các mã nhị phân và được gọi là tập lệnh.
- Bên trong CPU gồm :
  - Một tập các thanh ghi ( register) có nhiệm vụ lưu giữ tạm thời các thông tin,
  - Một đơn vị số học ALU ( Arithmetic logic unit) có nhiệm vụ thực hiện các thao tác trên các thông tin này.
  - Một đơn vị giải mã lệnh và điều khiển ( instruction decode and control unit) có nhiệm vụ xác định thao tác cần thực hiện và thiết lập các hoạt động cần thiết để thực hiện thao tác.
  - Thanh ghi lệnh IR ( instruction register) lưu giữ mã nhị phân của lệnh để được thực thi.
  - Bộ đếm chương trình PC (Program counter) lưu giữ địa chỉ của lệnh kế tiếp trong bộ nhớ cần thực thi.

### 1.4 Bộ nhớ bán dẫn RAM và ROM:

Các chương trình và dữ liệu được lưu giữ trong bộ nhớ. Các bộ nhớ được truy xuất trực tiếp bởi CPU bao gồm các IC bán dẫn gọi là RAM và ROM

- ROM(Read Only Memory): Bộ nhớ bán dẫn chỉ đọc :
- RAM (Random Access Memory): Bộ nhớ truy xuất ngẫu nhiên.
- RAM không tiếp tục lưu giữ nội dung khi bị mất nguồn cấp điện trong khi ROM thì ngược lại.

#### 1.4.1 Nguyên tắc vận hành của bộ nhớ:

Mỗi hệ thống nhớ luôn có một số yêu cầu ở các ngã vào và ra để hoàn thành một số tác vụ:

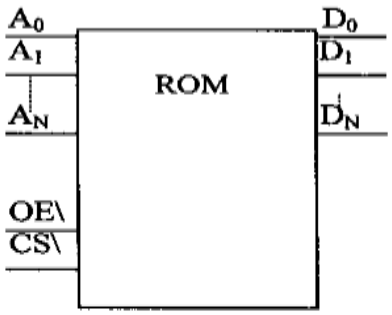
- Chọn địa chỉ trong bộ nhớ để truy xuất (đọc hoặc viết)
- Chọn tác vụ đọc hoặc viết để thực hiện
- Cung cấp dữ liệu để lưu vào bộ nhớ trong tác vụ viết
- Gửi dữ liệu ra từ bộ nhớ trong tác vụ đọc
- Cho phép (Enable) hay Không (Disable) bộ nhớ đáp ứng (hay không) đối với lệnh đọc/ghi ở địa chỉ đã gọi đến.

Mỗi IC nhớ có một số ngã vào ra như sau:

- **Ngã vào địa chỉ** : mỗi vị trí nhớ xác định bởi một địa chỉ duy nhất, khi cần đọc dữ liệu ra hoặc ghi dữ liệu vào ta phải tác động vào chân địa chỉ của vị trí nhớ đó. Một IC có n chân địa chỉ sẽ có $2^n$ vị trí nhớ. Ký hiệu các chân địa chỉ là $A_0$ đến $A_{n-1}$ Một IC có 10 chân địa chỉ sẽ có 1024 (1K) vị trí nhơ.
- **Ngã vào/ra dữ liệu: Các chân dữ liệu là các ngã vào/ra, nghĩa là dữ liệu luôn** được xử lý theo hai chiều. Thường thì dữ liệu vào/ra chung trên một chân nên các ngã này thuộc loại ngã ra 3 trạng thái. Số chân địa chỉ và dữ liệu của một IC xác định dung lượng nhớ của IC đó. Thí dụ một IC nhớ có 10 chân địa chỉ và 8 chân dữ liệu thì dung lượng nhớ của IC đó là 1Kx8 (8K bit hoặc 1K Byte).
- **Các ngã vào điều khiển:** Mỗi khi IC nhớ được chọn hoặc có yêu cầu xuất nhập dữ liệu các chân tương ứng sẽ được tác động. Ta có thể kể ra một số ngã vào điều khiển:
  - CS\: Chip select - Chọn chip - Khi chân này xuống thấp IC được chọn
  - CE\: Chip Enable - Cho phép chip - Chức năng như chân CS
  - OE: Output Enable - Cho phép xuất - Dùng khi đọc dữ liệu
  - WR\: Read/Write - Đọc/Viết - Cho phép **Đọc** dữ liệu ra khi ở mức cao và **Ghi** dữ liệu vào khi ở mức thấp
  - CAS: Column Address Strobe - Chốt địa chỉ cột
  - RAS: Row Address Strobe - Chốt địa chỉ hàng.

Trong trường hợp chip nhớ có dung lượng lớn, để giảm kích thước của mạch giải mã địa chỉ bên trong IC, người ta chia số chân ra làm 2: địa chỉ hàng và địa chỉ cột. Như vậy phải dùng 2 mạch giải mã địa chỉ nhưng mỗi mạch nhỏ hơn rất nhiều. Thí dụ với 10 chân địa chỉ, thay vì dùng 1 mạch giải mã 10 đường sang 1024 đường, người ta dùng 2 mạch giải mã 5 đường sang 32 đường, hai mạch này rất đơn giản so với một mạch kia. Một vị trí nhớ bây giờ có 2 địa chỉ : hàng và cột, dĩ nhiên muốn truy xuất một vị trí nhớ phải có đủ 2 địa chỉ nhờ 2 tín hiệu RAS và CAS.

### 1.4.2 Cấu trúc ROM:

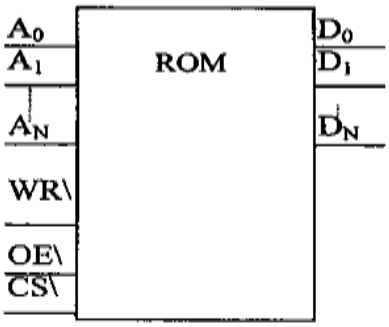



$A_0,\ldots, A_N$ : Các chân địa chỉ
$D_0,\ldots, D_N$ : Các chân dữ liệu
OE\(Output Enable): ngõ vào cho phép xuất dữ liệu.
CS\ (Chip Select): ngõ vào cho phép ROM hoạt động.

**1.4.3 Phân loại một số loại ROM:**

- MPROM (MASK ROM): ROM mặt nạ
- PROM (Programmable ROM): ROM lập trình được.
- EPROM (Erasable PROM): ROM lập trình và xóa được.
- UV-EPROM (Ultra Violet EPROM): Rom xóa bằng tia cực tím.
- EEPROM (Electric EPROM): ROM lập trình và xóa bằng điện.
- Flash ROM: ROM lập trình và xóa bằng tín hiệu điện.

**1.4.4 Cấu trúc RAM:**




$A_0,\ldots, A_N$ : Các chân địa chỉ
$D_0,\ldots, D_N$ : Các chân dữ liệu
OE\(Output Enable): ngõ vào cho phép xuất dữ liệu.
CS\ (Chip Select): ngõ vào cho phép ROM hoạt động.
WR\ (Write): ngõ vào cho phép ghi dữ liệu.

**1.4.5 Phân loại một số loại RAM:**

- DRAM (Dynamic RAM): RAM động.
- SRAM (Static RAM): RAM tĩnh.

**1.4.6 Cách xác định dung lượng bộ nhớ bán dẫn 8 bit**

- Dựa vào số chân địa chỉ:

Dung lượng bộ nhớ = $2^N$ với N là số đường địa chỉ của bộ nhớ.

- Dựa vào mã số của bộ nhớ:

Mã số XXYYYY XX: Loại bộ nhớ YYYY: Dung lượng (Kbit)

- 27: UV-EPROM
- 28: EEPROM
- 61,62: SRAM
- 40,41: DRAM

**Ví dụ 1:** Bộ nhớ có 10 đường địa chỉ, dung lượng bộ nhớ là bao nhiêu?
N = 10 nên dung lượng bộ nhớ = $2^{10}$ = 1024 = 1KB.

**Ví dụ 2:** Bộ nhớ có mã số 27256, dung lượng bộ nhớ là bao nhiêu?
Mã số 27 cho biết ta sử dụng bộ nhớ UV-EPROM
Mã số 256 cho biết dung lượng bộ nhớ = 256 Kbit = 32KB.

### 1.5 Hệ thống bus:

Bus là một tập các dây mang thông tin có cùng một chức năng. Việc truy xuất tới một mạch xung quanh CPU sử dụng 3 bus: Bus địa chỉ, bus dữ liệu và bus điều khiển.

Thao tác đọc học ghi:

- CPU xác định rõ vị trí của dữ liệu bằng cách đặt 1 địa chỉ lên bus địa chỉ.
- Tích cực 1 tín hiệu trên bus điều khiển để chỉ ra thao tác là đọc hay ghi.
- Thao tác đọc lấy 1 byte dữ liệu từ bộ nhớ ở vị trí đã xác định và đặt byte này lên bus dữ liệu. CPU đọc dữ liệu và đặt dữ liệu vào 1 trong các thanh ghi nội của CPU.
- Với thao tác ghi, CPU xuất dữ liệu lên bus dữ liệu. Nhờ tín hiệu điều khiển CPU nhận biết đây là thao tác ghi và lưu dữ liệu vào vị trí đã được xác định.

#### 1.5.1 Bus địa chỉ (address bus)

- Dùng để truyền tải thông tin giữa các bit địa chỉ.
- Là loại bus 1 chiều.
- Dùng để xác định bộ nhớ hoặc thiết bị ngoại vi mà CPU cần trao đổi thông tin.
- Để xác định dung lượng bộ nhớ hoặc ngoại vi mà CPU có khả năng truy xuất.

#### 1.5.2 Bus dữ liệu (data bus)

- Để truyền tải thông tin giữa CPU và bộ nhớ, giữa CPU và các thiết bị xuất nhập.
- Là loại bus 2 chiều.
- Để xác định số bit dữ liệu mà CPU có khả năng xử lý cùng 1 lúc.

#### 1.5.3 Bus điều khiển (Control bus)

- Là 1 hỗn hợp các tín hiệu, mỗi một tín hiệu có 1 vai trò riêng trong việc điều khiển có trực tự hoạt động của hệ thống.
- Các tín hiệu điều khiển là các tín hiệu định thời được cung cấp bởi CPU để đồng bộ việc di chuyển thông tin trên các bus địa chỉ và dữ liệu.
- Là loại bus 1 chiều.

# Chương 2: Giới thiệu phần cứng 89C51

## 2.1 Họ MCS51:

Có rất nhiều hãng chế tạo được vi điều khiển, hãng sản xuất nổi tiếng là ATMEL. Hãng Intel là nhà thiết kế. Nhiều họ vi điều khiển mang các mã số khác nhau, một trong họ nổi tiếng là họ ***MCS-51.***

Trong họ MCS-51 thì vi điều khiển đầu tiên là 80C31 không có bộ nhớ bên trong là do không tích hợp được.

Vi điều khiển 80C51 tích hợp được 4 kbyte bộ nhớ Prom. Chỉ lập trình 1 lần không thể xóa để lập trình lại được.

Vi điều khiển 87C51 tích hợp được 4 kbyte bộ nhớ eprom. Cho phép lập trình nhiều lần và xóa bằng tia cực tím.

Vi điều khiển 89C51 tích hợp được 4 kbyte bộ nhớ flash rom nạp và xóa bằng điện một cách tiện lợi và nhanh chóng. Có thể cho phép nạp xóa hàng ngàn lần.

Song song với họ MCS-51 là họ MCS-52 có 3 timer nhiều hơn họ MCS-51 một timer và dung lượng bộ nhớ nội lớn gấp đôi tức là 8kbyte.

Hiện nay có rất nhiều vi điều khiển thế hệ sau có nhiều đặc tính hay hơn, nhiều thanh ghi hơn, dung lượng bộ nhớ lớn hơn.

Ứng dụng của vi điều khiển rất nhiều trong các hệ thống điều khiển công nghiệp, các dây chuyền sản xuất, các bộ điều khiển lập trình, máy giặt, máy điều hòa nhiệt độ, máy bơm xăng tự động... có thể nói vi xử lý và vi điều khiển được ứng dụng trong hầu hết mọi lĩnh vực.

### 2.1.1 Tóm tắt phần cứng họ MSC51(8951):

AT89C51 là vi điều khiển do Atmel sản xuất, chế tạo theo công nghệ CMOS có các đặc tính như sau:

- 4 KB PEROM (Flash Programmable and Erasable Read Only Memory), có khả năng tới 1000 chu kỳ ghi xoá
- Tần số hoạt động từ: 0Hz đến 24 MHz
- 3 mức khóa bộ nhớ lập trình
- 128 Byte RAM nội.
- 4 Port xuất /nhập I/O 8 bit.
- 2 bộ Timer/counter 16 Bit.
- 6 nguồn ngắt.
- Giao tiếp nối tiếp điều khiển bằng phần cứng.
- 64 KB vùng nhớ mã ngoài
- 64 KB vùng nhớ dữ liệu ngoài.
- Cho phép xử lý bit.
- 210 vị trí nhớ có thể định vị bit.
- 4 chu kỳ máy (4 µs đối với thạch anh 12MHz) cho hoạt động nhân hoặc chia.

Tất cả các vi điều khiển cùng họ MCS-51 hoặc MCS-52 đều có các đặc tính cơ bản giống nhau như phần mềm, còn phần cứng thì khác nhau, các vi điều khiển sau này sẽ

có nhiều tính năng hay hơn các vi điều khiển thế hệ trước. Ví dụ vi điều khiển 89C51 sẽ tiện cho việc sử dụng hơn vi điều khiển 80C51 hay 87C51. Vi điều khiển 89S51 sẽ hay hơn 89C51 vì có nhiều thanh ghi hơn, có thêm chế độ nạp nối tiếp rất tiện lợi. Những thế hệ đi sau sẽ kế thừa tất cả những gì của thế hệ đi trước. Trong phần này chỉ đề cập đến vi điều khiển 89C51/89C52.

### 2.1.2 Khảo sát cấu trúc bên trong của vi điều khiển 89C51:

#### 2.1.2.1 Sơ đồ cấu trúc bên trong của vi điều khiển:

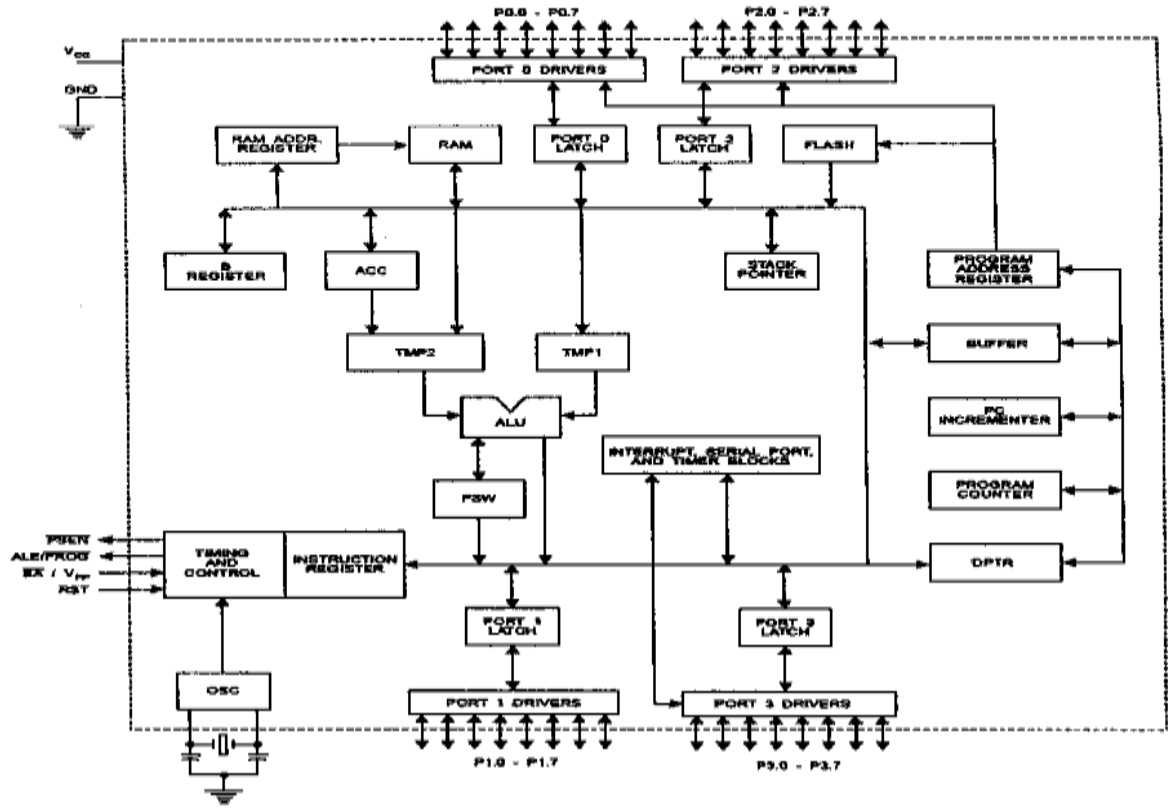
Các thanh ghi có trong vi điều khiển bao gồm:

- Khối ALU đi kèm với các thanh ghi temp1, temp2 và thanh ghi trạng thái PSW.
- Bộ điều khiển logic (timing and control).
- Vùng nhớ RAM nội và vùng nhớ Flash Rom lưu trữ chương trình.
- Mạch tạo dao động nội kết hợp với tụ thạch anh bên ngoài để tạo dao động.
- Khối xử lý ngắt, truyền dữ liệu, khối timer/counmter.
- Thanh ghi A, B, dptr và 4 port0, port1, port2, port3 có chốt và đệm.
- Thanh ghi bộ đếm chương trình PC (program counter).
- Con trỏ dữ liệu dptr (data pointer).
- Thanh ghi con trỏ ngăn xếp SP (stack pointer).
- Thanh ghi lệnh IR (instruction register).
- Ngoài ra còn có 1 số các thanh ghi hỗ trợ để quản lý địa chỉ bộ nhớ ram nội bên trong cũng như các thanh ghi quản lý địa chỉ truy xuất bộ nhớ bên ngoài.

Khi khảo sát các khối này không cần thiết phải hiểu hết chức năng của từng khối hoạt động ra sao vào thời điểm này, các thông tin ở phần sau sẽ giúp hiểu rõ thêm về tổ chức phần cứng này.

Tập lệnh cho người lập trình là kết quả của sự liên kết các khối bên trong của vi điều khiển – những gì tập lệnh cung cấp là đều do phần cứng xây dựng nên.

**Block Diagram**




### 2.1.3 Các phiên bản của 8051 từ Atmel (Flash ROM)

| Số linh kiện | ROM | RAM | Chân I/O | Timer | Ngắt | Vcc | Đóng vỏ |
|---|---|---|---|---|---|---|---|
| AT89C51 | 4K | 128 | 32 | 2 | 6 | 5V | 40 |
| AT89LV51 | 4K | 128 | 32 | 2 | 6 | 3V | 40 |
| AT89C1051 | 1K | 64 | 15 | 1 | 3 | 3V | 20 |
| AT89C2051 | 2K | 128 | 15 | 2 | 6 | 3V | 20 |
| AT89C52 | 8K | 128 | 32 | 3 | 8 | 5V | 40 |
| AT89LV52 | 8K | 128 | 32 | 3 | 8 | 3V | 40 |

## 2.2 Khảo sát sơ đồ chân 89C51

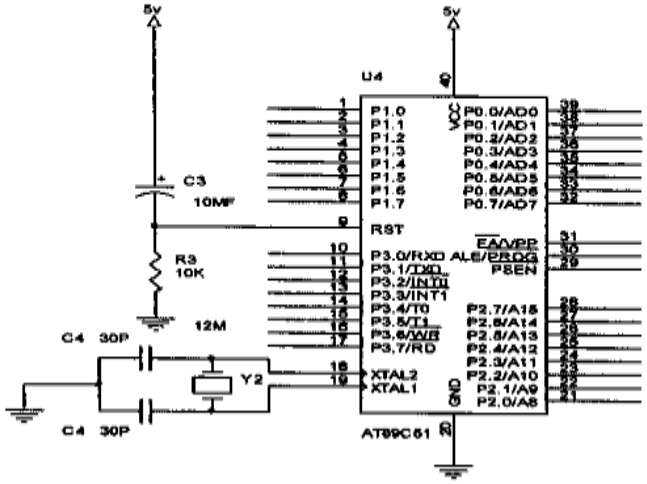
Vi điều khiển 89C51 có tất cả 40 chân. Trong đó có 24 chân có tác dụng kép (có nghĩa là 1 chân có 2 chức năng), mỗi đường có thể hoạt động như đường xuất nhập điều khiển IO [input output] hoặc là thành phần của các bus dữ liệu và bus địa chỉ để tải địa chỉ và dữ liệu khi giao tiếp với bộ nhớ ngoài.

Chip 89c51 có 40 chân:

- 2 chân nguồn cấp điện( VCC, GND)
- 6 chân chức năng(EA, ALE, PSEN, XTAL1, XTAL2, RST)

– 32 chân xuất nhập:

Port xuất/nhập 8 bit (P0.0 – P0.7)
Port xuất/nhập 8 bit (P1.0 – P1.7)
Port xuất/nhập 8 bit (P2.0 – P2.7)
Port xuất/nhập 8 bit (P3.0 – P3.7)



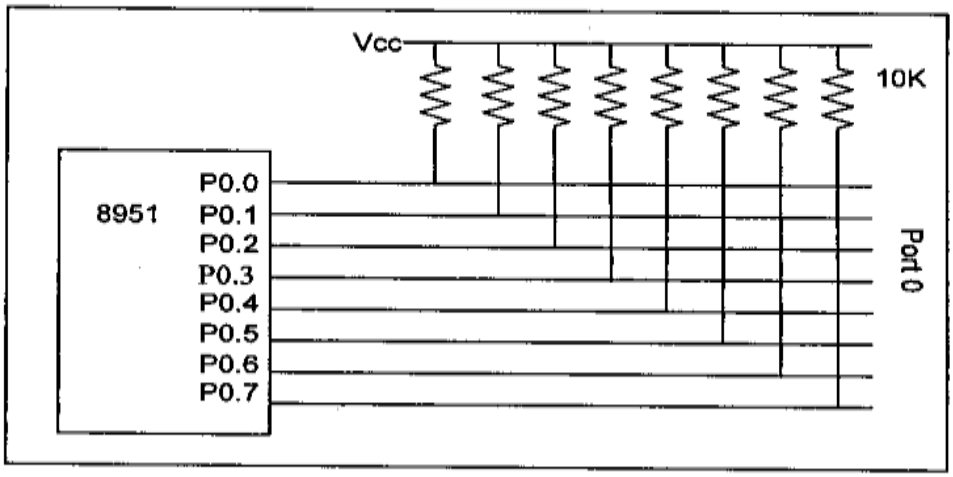

### 2.2.1 Port 0

Port 0 là port có 2 chức năng với số thứ tự chân 32 – 39.

– Trong các hệ thống điều khiển đơn giản sử dụng bộ nhớ bên trong không dùng bộ nhớ mở rộng bên ngoài thì port 0 được dùng làm các đường điều khiển IO (Input-Output).

– Trong các hệ thống điều khiển lớn sử dụng bộ nhớ mở rộng bên ngoài thì port 0 có chức năng là bus địa chỉ và bus dữ liệu AD7 - AD0. (Address: địa chỉ, data: dữ liệu)

Lưu ý: Khi Port 0 đóng vai trò là port xuất nhập dữ liệu thì mỗi chân phải được nối tới một điện trở kéo bên ngoài 10kΩ.



### 2.2.2 Port 1

Port 1: có số chân từ 1 – 8
Port 1 có 1 chức năng là port xuất nhập dữ liệu 8 bit. So với cổng P0 thì cổng này không cần đến điện trở kéo vì nó đã có các điện trở kéo bên trong.

### 2.2.3 Port 2

Port 2 là port có 2 chức năng với số thứ tự chân 21 – 28.

– Trong các hệ thống điều khiển đơn giản sử dụng bộ nhớ bên trong không dùng bộ nhớ mở rộng bên ngoài thì port 2 được dùng làm các đường điều khiển IO (Input-Output).

– Trong các hệ thống điều khiển lớn sử dụng bộ nhớ mở rộng bên ngoài thì port 2 có chức năng là bus địa chỉ cao A8 - A15.
Cổng P2 cũng không cần điện trở kéo vì nó đã có các điện trở kéo bên trong

### 2.2.4 Port 3

Port 3 là port có 2 chức năng với số thứ tự chân 10 -17. Các chân của port này có nhiều chức năng, các công dụng chuyển đổi có liên hệ với các đặc tính đặc biệt của 89C51 như ở bảng sau:

| Bit | Tên | Chức năng chuyển đổi |
|---|---|---|
| P3.0 | RXT | Dữ liệu nhận cho port nối tiếp. |
| P3.1 | TXD | Dữ liệu phát cho port nối tiếp. |
| P3.2 | INT0\ | Ngắt 0 bên ngoài |
| P3.3 | INT1\ | Ngắt 1 bên ngoài |
| P3.4 | T0 | Ngõ vào của Timer/Counter 0. |
| P3.5 | T1 | Ngõ vào của Timer/Counter 1. |
| P3.6 | WR\ | Xung ghi bộ nhớ dữ liệu ngoài |
| P3.7 | RD\ | Xung đọc bộ nhớ dữ liệu ngoài |

### 2.2.5 Các tín hiệu điều khiển

**Ngõ tín hiệu $\overline{PSEN}$ (Program store enable):**

– $\overline{PSEN}$ là tín hiệu ngõ ra ở chân 29 có tác dụng cho phép đọc bộ nhớ chương trình mở rộng thường nối đến chân $\overline{OE}$ ( output enable hoặc $\overline{RD}$ ) của Eprom cho phép đọc các byte mã lệnh.

– Khi có giao tiếp với bộ nhớ chương trình bên ngoài thì mới dùng đến $\overline{PSEN}$, nếu không có giao tiếp thì chân $\overline{PSEN}$ bỏ trống.

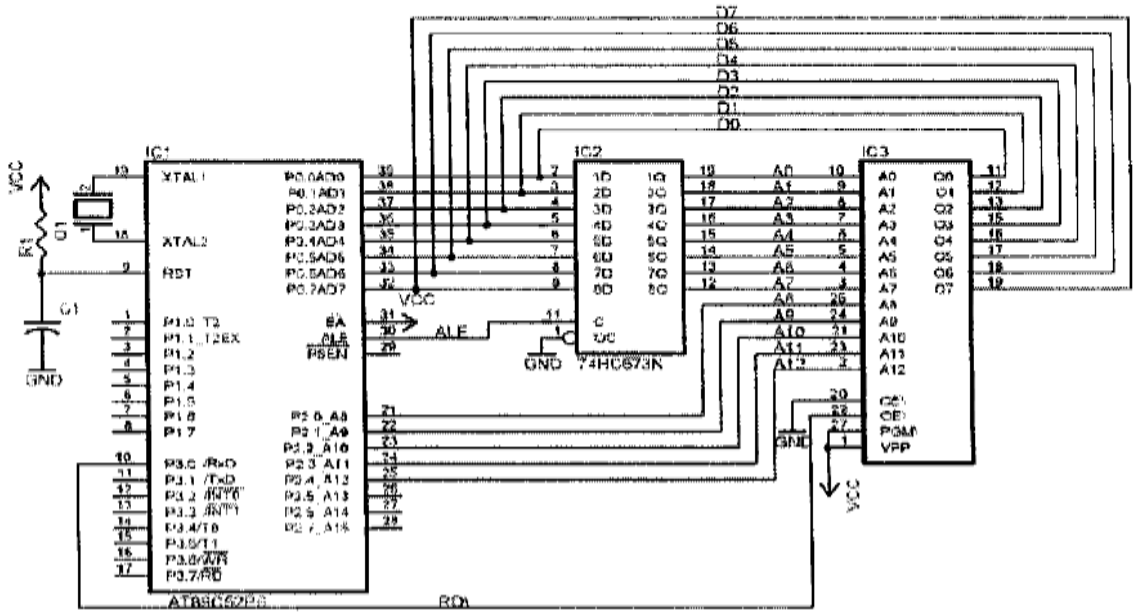
($\overline{PSEN}$ ở mức thấp trong thời gian vi điều khiển 89C51 lấy lệnh. Các mã lệnh của chương trình đọc từ Eprom qua bus dữ liệu và được chốt vào thanh ghi lệnh bên trong 89C51 để giải mã lệnh. Khi 89C51 thi hành chương trình trong EPROM nội thì $\overline{PSEN}$ ở mức logic 1).

**Ngõ tín hiệu điều khiển ALE (Address Latch Enable ) :**

– Khi vi điều khiển 89C51 truy xuất bộ nhớ bên ngoài, port 0 có chức năng là bus

tải địa chỉ và bus dữ liệu [AD7 – AD0] do đó phải tách các đường dữ liệu và địa chỉ. Tín hiệu ra ALE ở chân thứ 30 dùng làm tín hiệu điều khiển để giải đa hợp các đường địa chỉ và dữ liệu khi kết nối chúng với IC chốt

– Tín hiệu ra ở chân ALE là một xung trong khoảng thời gian port 0 đóng vai trò là địa chỉ thấp nên việc chốt địa chỉ được thực hiện 1 cách hoàn toàn tự động.




– Các xung tín hiệu ALE có tốc độ bằng 1/6 lần tần số dao động của tụ thạch anh gắn vào vi điều khiển và có thể dùng tín hiệu xung ngõ ra ALE làm xung clock cung cấp cho các phần khác của hệ thống.

– Trong chế độ lập trình cho bộ nhớ nội của vi điều khiển thì chân ALE được dùng làm ngõ vào nhận xung lập trình từ bên ngoài để lập trình cho bộ nhớ flash rom trong 89C51.

**Ngõ tín hiệu $\overline{EA}$ (External Access):**

Tín hiệu vào $\overline{EA}$ ở chân 31 thường nối lên mức 1 hoặc mức 0.

– Nếu nối $\overline{EA}$ lên mức logic 1 (+5v) thì vi điều khiển sẽ thi hành chương trình từ bộ nhớ nội.

– Nếu nối $\overline{EA}$ với mức logic 0 (0V) thì vi điều khiển sẽ thi hành chương trình từ bộ nhớ ngoại.

**Ngõ tín hiệu RST (Reset):**

– Ngõ vào RST ở chân 9 là ngõ vào Reset của 89C51. Khi cấp điện cho hệ thống hoặc khi nhấn nút reset thì mạch sẽ reset vi điều khiển. Khi reset thì tín hiệu reset phải ở mức cao ít nhất là 2 chu kỳ máy, khi đó các thanh ghi bên trong được nạp những giá trị thích hợp để khởi động hệ thống.

– Trạng thái của tất cả các thanh ghi trong 89C51 sau khi reset hệ thống được

tóm tắt như sau:

| Thanh ghi | Nội dung |
|---|---|
| Đếm chương trình PC | 0000H |
| Thanh ghi tích lũy A | 00H |
| Thanh ghi B | 00H |
| Thanh ghi thái PSW | 00H |
| SP | 07H |
| DPRT | 0000H |
| Port 0 đến port 3 | FFH |
| IP | XXX0 0000 B |
| IE | 0X0X 0000 B |
| Các thanh ghi định thời | 00H<br>00H |
| SCON SBUF | 00H |
| PCON (MHOS) | 0XXX XXXXH |
| PCON (CMOS) | 0XXX 0000 B |

– Thanh ghi quan trọng nhất là thanh ghi bộ đếm chương trình PC = 0000H. Sau khi reset vi điều khiển luôn bắt đầu thực hiện chương trình tại địa chỉ 0000H của bộ nhớ chương trình nên các chương trình viết cho vi điều khiển luôn bắt đầu viết tại địa chỉ 0000H.

– Nội dung của RAM trên chip không bị thay đổi bởi tác động của ngõ vào reset [có nghĩa là vi điều khiển đang sử dụng các thanh ghi để lưu trữ dữ liệu nhưng nếu vi điều khiển bị reset thì dữ liệu trong các thanh ghi vẫn không đổi].

**Các ngõ vào bộ dao động Xtal1, Xtal2:**

Bộ dao động được được tích hợp bên trong 89C51, khi sử dụng 89C51 người thiết kế chỉ cần kết nối thêm tụ thạch anh và các tụ. Tần số tụ thạch anh thường sử dụng cho 89C51 là 12Mhz ÷ 24Mhz.

**Chân 40 (Vcc) được nối lên nguồn 5V, chân 20 GND nối mass.**

### 2.2.6 Tổ chức bộ nhớ

Vi điều khiển 89C51 có ***bộ nhớ nội bên trong*** và có thêm khả năng giao tiếp với ***bộ nhớ bên ngoài*** nếu bộ nhớ bên trong không đủ khả năng lưu trữ chương trình.

– Bộ nhớ nội bên trong gồm có 2 loại bộ nhớ: bộ nhớ dữ liệu và bộ chương trình. Bộ nhớ dữ liệu có 256 byte, bộ nhớ chương trình có dung lượng 4kbyte. [89C52 có 8 kbyte, 89W55 có 16kbyte].

– Bộ nhớ mở rộng bên ngoài cũng được chia ra làm 2 loại bộ nhớ: bộ nhớ dữ liệu và bộ nhớ chương trình. Khả năng giao tiếp là 64kbyte cho mỗi loại.

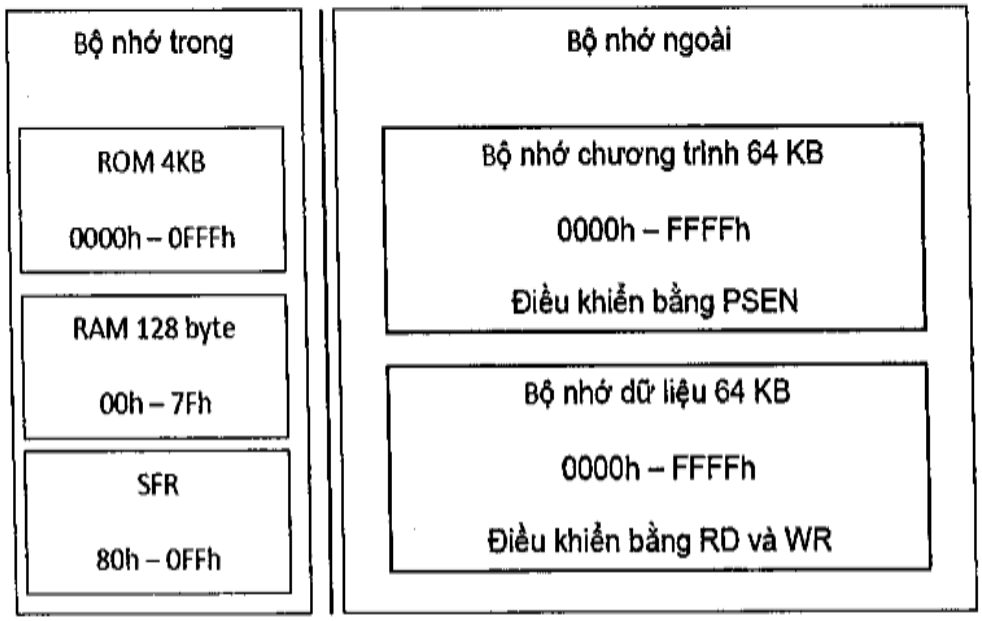
– Bộ nhớ mở rộng bên ngoài và bộ nhớ chương trình bên trong không có gì

đặc biệt – chỉ có chức năng lưu trữ dữ liệu và mã chương trình nên không cần phải khảo sát.

– Bộ nhớ chương trình bên trong của vi điều khiển thuộc loại bộ nhớ Flash rom cho phép xóa bằng xung điện và lập trình lại.

– Bộ nhớ ram nội bên trong là một bộ nhớ đặc biệt người sử dụng vi điều khiển cần phải nắm rõ các tổ chức và các chức năng đặc biệt của bộ nhớ này.

Sơ đồ cấu trúc bên trong của bộ nhớ này được trình bày như hình vẽ




**RAM bên trong 89C51 được phân chia như sau:**

- ☐ Các bank thanh ghi có địa chỉ từ 00H đến 1FH.
- ☐ RAM địa chỉ hóa từng bit có địa chỉ từ 20H đến 2FH.
- ☐ RAM đa dụng từ 30H đến 7FH.
- ☐ Các thanh ghi chức năng đặc biệt từ 80H đến FFH.

Bản đồ bộ nhớ Data trên Chip như sau :

| Địa chỉ byte | Địa chỉ bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7F | RAM đa dụng | | | | | | | |
| 30 | | | | | | | | |
| 2F | 7F | 7E | 7D | 7C | 7B | 7A | 79 | 78 |
| 2E | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 |
| 2D | 6F | 6E | 6D | 6C | 6B | 6A | 69 | 68 |
| 2C | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 |
| 2B | 5F | 5E | 5D | 5C | 5B | 5A | 59 | 58 |
| 2A | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 |
| 29 | 4F | 4E | 4D | 4C | 4B | 4A | 49 | 48 |
| 28 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| 27 | 3F | 3E | 3D | 3C | 3B | 3A | 39 | 38 |
| 26 | 37 | 36 | 35 | 34 | 33 | 32 | 31 | 30 |
| 25 | 2F | 2E | 2D | 2C | 2B | 2A | 29 | 28 |
| 24 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 23 | 1F | 1E | 1D | 1C | 1B | 1A | 19 | 18 |
| 22 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| 21 | 0F | 0E | 0D | 0C | 0B | 0A | 09 | 08 |
| 20 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 1F | Bank 3 | | | | | | | |
| 18 | | | | | | | | |
| 17 | Bank 2 | | | | | | | |
| 10 | | | | | | | | |
| 0F | Bank 1 | | | | | | | |
| 08 | | | | | | | | |
| 07 | Bank thanh ghi 0 | | | | | | | |
| 00 | (mặc định cho R0 -R7) | | | | | | | |

| Địa chỉ byte | Địa chỉ bit | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| FF | | | | | | | | | |
| F0 | F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 | B |
| E0 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 | ACC |
| D0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | PSW |
| B8 | - | - | - | BC | BB | BA | B9 | B8 | IP |
| B0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | P.3 |
| A8 | AF | | | AC | AB | AA | A9 | A8 | IE |
| A0 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | P2 |
| 99 | không được địa chỉ hoá bit | | | | | | | | SBUF |
| 98 | 9F | 9E | 9D | 9C | 9B | 9A | 99 | 98 | SCON |
| 90 | 97 | 96 | 95 | 94 | 93 | 92 | 91 | 90 | P1 |
| 8D | không được địa chỉ hoá bit | | | | | | | | TH1 |
| 8C | không được địa chỉ hoá bit | | | | | | | | TH0 |
| 8B | không được địa chỉ hoá bit | | | | | | | | TL1 |
| 8A | không được địa chỉ hoá bit | | | | | | | | TL0 |
| 89 | không được địa chỉ hoá bit | | | | | | | | TMOD |
| 88 | 8F | 8E | 8D | 8C | 8B | 8A | 89 | 88 | TCON |
| 87 | không được địa chỉ hoá bit | | | | | | | | PCON |
| 83 | không được địa chỉ hoá bit | | | | | | | | DPH |
| 82 | không được địa chỉ hoá bit | | | | | | | | DPL |
| 81 | không được địa chỉ hoá bit | | | | | | | | SP |
| 80 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | P0 |

**Các bank thanh ghi :** 32 byte thấp của bộ nhớ nội được dành cho 4 bank thanh ghi.

– Bộ lệnh 89C51 hỗ trợ thêm 8 thanh ghi có tên là R0 đến R7 và theo mặc định sau khi reset hệ thống thì các thanh ghi R0 đến R7 được gán cho 8 ô nhớ có địa chỉ từ 00H đến 07H, khi đó bank 0 có 2 cách truy xuất bằng địa chỉ trực tiếp và bằng thanh ghi R.

– Các lệnh dùng các thanh ghi R0 đến R7 sẽ có số lượng byte mã lệnh ít hơn và thời gian thực hiện lệnh nhanh hơn so với các lệnh có chức năng tương ứng nếu dùng kiểu địa chỉ trực tiếp.

– Các dữ liệu được dùng thường xuyên nên lưu trữ ở một trong các thanh ghi này.

– Do có 4 bank thanh ghi nên tại một thời điểm chỉ có một bank thanh ghi được truy xuất bởi các thanh ghi R0 đến R7, để chuyển đổi việc truy xuất các bank thanh ghi ta phải thay đổi các bit chọn bank trong thanh ghi trạng thái.

– Chức năng chính của 4 bank thanh ghi này là nếu trong hệ thống có sử dụng nhiều chương trình thì chương trình thứ nhất bạn có thể sử dụng hết các thanh ghi R0 đến R7 của bank0, khi bạn chuyển sang chương trình thứ 2 để xử lý một công việc gì đó và vẫn sử dụng các thanh ghi R0 đến R7 để lưu trữ cho việc sử lý dữ liệu mà không làm ảnh hưởng đến các dữ liệu R0 đến R7 trước đây và không cần phải thực hiện công việc cất dữ liệu thì cách nhanh nhất là bạn gán nhóm thanh ghi R0 đến R7 cho bank1 là xong. Tương tự bạn có thể mở thêm hai chương trình nữa và gán cho các bank 3 và 4.

**RAM có thể truy xuất từng bit:**

Vi điều khiển 89C51 có 210 ô nhớ bit có thể truy xuất từng bit, trong đó có 128 bit nằm ở các các ô nhớ byte có địa chỉ từ 20H đến 2FH và các bit còn lại chứa trong nhóm thanh ghi có chức năng đặc biệt.

– Các ô nhớ cho phép truy xuất từng bit và các lệnh xử lý bit là một thế mạnh của vi điều khiển. Các bit có thể được đặt, xóa, AND, OR bằng 1 lệnh duy nhất trong khi đó để xử lý các bit thì ***vi xử lý*** vẫn có thể xử lý được nhưng phải sử dụng rất nhiều lệnh để đạt được cùng một kết quả vì ***vi xử lý*** thường xử lý byte.

– Các port cũng có thể truy xuất được từng bit.

– 128 ô nhớ bit cho phép truy xuất từng bit và cũng có thể truy xuất byte phụ thuộc vào lệnh được dùng là lệnh xử bit hay lệnh xử lý byte. Chú ý địa chỉ của ô nhớ byte và bit trùng nhau.

– Người lập trình dùng vùng nhớ này để lưu trữ dữ liệu phục vụ cho việc xử lý dữ liệu byte hoặc bit. Các dữ liệu xử lý bit nên lưu vào vùng nhớ này.

Chú ý: các ô nhớ nào mà chia ra làm 8 và có các con số bên trong là các ô nhớ vừa cho truy xuất byte và cả truy xuất bit. Những ô nhớ còn lại thì không thể truy xuất bit.

**RAM đa dụng :**

Vùng nhớ ram đa dụng gồm có 80 byte có địa chỉ từ 30H đến 7FH – vùng nhớ này không có gì đặc biệt so với 2 vùng nhớ trên. Vùng nhớ bank thanh ghi 32 byte từ 00H đến 1FH cũng có thể dùng làm vùng nhớ ram đa dụng mặc dù các các ô nhớ này đã có chức năng như đã trình bày.

– Mọi địa chỉ trong vùng RAM đa dụng đều có thể truy xuất tự do dùng kiểu địa chỉ trực tiếp hoặc gián tiếp.

– Bộ nhớ ngăn xếp của vi điều khiển dùng bộ nhớ Ram nội nên dung lượng của bộ nhớ ngăn xếp nhỏ trong khi đó các bộ vi xử lý dùng bộ nhớ bên ngoài làm bộ nhớ ngăn xếp nên dung lượng tùy ý mở rộng.

**Các thanh ghi có chức năng đặc biệt :**

Các thanh ghi nội của 89C51 được truy xuất ngầm định bởi bộ lệnh. Các thanh ghi trong 89C51 được định dạng như một phần của RAM trên chip vì vậy mỗi thanh ghi sẽ

có một địa chỉ (ngoại trừ thanh ghi bộ đếm chương trình và thanh ghi lưu trữ mã lệnh vì các thanh ghi này đã có chức năng cố định). Cũng như các thanh ghi R0 đến R7, vi điều khiển 89C51 có 21 thanh ghi có chức năng đặc biệt nằm ở vùng trên của RAM nội có địa chỉ từ 80H đến FFH.

***Chú ý:*** 128 ô nhớ có địa chỉ từ 80H đến FFH thì chỉ có 21 thanh ghi có chức năng đặc biệt được xác định các địa chỉ – còn các ô nhớ còn lại thì chưa thiết lập và trong tương lai sẽ được các nhà thiết kế vi điều khiển thiết lập thêm khi đó sẽ có các vi điều khiển thế hệ mới hơn.

***Các ô nhớ có địa chỉ 80H, 90H, A0h, B0h:*** Là các Port của 89C51 bao gồm Port0 có địa chỉ 80H, Port1 có địa chỉ 90H, Port2 có địa chỉ A0H và Port3 có địa chỉ B0H. Tất cả các Port này đều có thể truy xuất từng bit nên rất thuận tiện trong điều khiển IO. Địa chỉ của các bit được đặt tên với ô bắt đầu chính là địa chỉ của port tương ứng ví dụ như bit đầu tiên của port 0 là 80h cũng chính là địa chỉ bắt đầu của port 0. Người lập trình không cần nhớ địa chỉ các bit trong các port vì phần mềm lập trình cho phép truy xuất bằng tên từng bit dễ nhớ như sau: P0.0 chính là bit có địa chỉ 80h của port0.

Ngoại trừ thanh ghi A có thể được truy xuất ngầm, đa số các thanh ghi có chức năng đặc biệt SFR có thể địa chỉ hóa từng bit hoặc byte.

***Ô nhớ có địa chỉ 81h:***

Là thanh ghi con trỏ ngăn xếp SP (stack pointer) - có chức năng quản lý địa chỉ của bộ nhớ ngăn xếp. Bộ nhớ ngăn xếp dùng để lưu trữ tạm thời các dữ liệu trong quá trình thực hiện chương trình của vi điều khiển.

– Các lệnh liên quan đến ngăn xếp bao gồm các lệnh cất dữ liệu vào ngăn xếp (lệnh push) và lấy dữ liệu ra khỏi ngăn xếp (lệnh pop).

– Lệnh cất dữ liệu vào ngăn xếp sẽ làm tăng SP trước khi ghi dữ liệu vào.

– Sau lệnh lấy ra khỏi ngăn xếp sẽ làm giảm SP.

– Bộ nhớ ngăn xếp của 89C51 nằm trong RAM nội và bị giới hạn về cách truy xuất địa chỉ - chỉ cho phép truy xuất địa chỉ gián tiếp. Dung lượng bộ nhớ ngăn xếp lớn nhất là 128 byte ram nội của 89C51.

– Khi Reset 89C51 thì thanh ghi SP sẽ mang giá trị mặc định là 07H và dữ liệu đầu tiên sẽ được cất vào ô nhớ ngăn xếp có địa chỉ 08H.

– Nếu phần mềm ứng dụng không khởi tạo SP một giá trị mới thì bank 1 có thể cả 2 và 3 sẽ không dùng được vì vùng nhớ này đã được dùng làm ngăn xếp.

– Ngăn xếp được truy xuất trực tiếp bằng các lệnh PUSH và POP để lưu trữ tạm thời và lấy lại dữ liệu, hoặc truy xuất ngầm bằng lệnh gọi chương trình con ( ACALL, LCALL) và các lệnh trở về (RET, RETI) để lưu trữ địa chỉ của bộ đếm chương trình khi bắt đầu thực hiện chương trình con và lấy lại địa chỉ khi kết thúc chương trình con.

***Ô nhớ có địa chỉ 82h và 83h :***

Là 2 thanh ghi dpl (byte thấp) có địa chỉ là 82H và dph (byte cao) có địa chỉ 83H. Hai thanh ghi này có thể sử dụng độc lập để lưu trữ dữ liệu và có thể kết hợp lại tạo thành 1 thanh ghi 16 bit có tên là dptr và gọi là con trỏ dữ liệu - được dùng để lưu địa chỉ 16 bit khi truy xuất dữ liệu của bộ nhớ dữ liệu bên ngoài. Các ví điều khiển sau này có thêm một thanh ghi dptr.

***Ô nhớ có địa chỉ 87h:***

Là thanh ghi pcon (power control) có chức năng điều khiển công suất khi vi điều khiển làm việc hay ở chế độ chờ. Khi vi điều khiển không còn xử lý gì nữa thì người lập trình có thể lập trình cho vi điều khiển chuyển sang chế độ chờ để giảm bớt công suất tiêu thụ nhất là khi nguồn cung cấp cho vi điều khiển là pin.

***Các ô nhớ có địa chỉ từ 88h đến 8dh :***

Là các thanh ghi phục vụ cho 2 timer/ counter T1, T0.

– Thanh ghi tcon (timer control): thanh ghi điều khiển timer / counter.

– Thanh ghi tmod (timer mode): thanh ghi lựa chọn mode hoạt động cho timer counter.

– Thanh ghi TH0 và TL0 kết hợp lại tạo thành 1 thanh ghi 16 bit có chức năng lưu trữ xung đếm cho timer/counter T0. Tương tự cho 2 thanh ghi TH1 và TL1 kết hợp lại để lưu trữ xung đếm cho timer/counter T1. Khả năng lưu trữ số lượng xung đếm được là 65536 xung.

***Các ô nhớ có địa chỉ từ 98h đến 99h :***

Sbuf (series buffer ): thanh ghi đệm dữ liệu truyền nối tiếp. Dữ liệu muốn truyền đi thì phải lưu vào thanh ghi SBUF và dữ liệu nhận về nối tiếp cũng lưu ở thanh ghi này. Khi có sử dụng truyền dữ liệu thì phải sử dụng 2 thanh ghi này.

Chức năng của các thanh ghi này sẽ được trình bày rõ ở chương truyền dữ liệu.

***Các ô nhớ có địa chỉ từ a8h đến b9h :*** Là 2 thanh ghi IE và IP

– Thanh ghi IE (interrupt enable) thanh ghi điều khiển cho phép / không cho phép ngắt.

– IP (interrupt priority): thanh ghi điều khiển ưu tiên ngắt. Khi có sử dụng đến ngắt thì phải dùng đến 2 thanh ghi này. Mặc nhiên các thanh ghi này được khởi tạo ở chế độ cấm ngắt.

***Thanh ghi trạng thái chương trình (PSW: Program Status Word):***

Thanh ghi trạng thái chương trình ở địa chỉ D0H được tóm tắt như sau:

| BIT | SYMBOL | ADDRESS | DESCRIPTION |
|---|---|---|---|
| PSW.7 | CY | D7H | Cary Flag |
| PSW.6 | AC | D6H | Auxiliary Cary Flag |
| PSW.5 | F0 | D5H | Flag 0 |
| PSW4 | RS1 | D4H | Register Bank Select 1 |
| PSW.3 | RS0 | D3H | Register Bank Select 0 |
| | | | 00=Bank 0; address 00H÷07H |
| | | | 01=Bank 1; address 08H÷0FH |
| | | | 10=Bank 2; address 10H÷17H |
| | | | 11=Bank 3; address 18H÷1FH |

| | | | |
|---|---|---|---|
| PSW.2 | OV | D2H | Overlow Flag |
| PSW.1 | - | D1H | Reserved |
| PSW.0 | P | DOH | Even Parity Flag |

***Chức năng từng bit trạng thái:***

*Cờ Carry CY (Carry Flag):* Cờ nhớ có tác dụng kép. Cờ C được sử dụng cho các lệnh toán học:

– C = 1 nếu phép toán cộng có tràn hoặc phép trừ có mượn.
– C = 0 nếu phép toán cộng không tràn và phép trừ không có mượn.

*Cờ Carry phụ AC (Auxiliary Carry Flag):*

Khi cộng những giá trị BCD (Binary Code Decimal), cờ nhớ phụ AC được set [AC=1] nếu kết quả 4 bit lớn hơn 09H, ngược lại AC= 0. Cờ AC được dùng để chỉnh số BCD khi thực hiện lệnh cộng 2 số BCD.

*Cờ 0 (Flag 0):*

Cờ 0 (F0) còn gọi là cờ zero, cờ zero =1 khi kết qủa xử lý bằng 0 và cờ zero = 0 khi kết quả xử lý khác 0.

*Các bit chọn bank thanh ghi truy xuất:*

Hai bit RS1 và RS0 dùng để thay đổi cách gán 8 thanh ghi R7 – R0 cho 1 trong 4 bank thanh ghi. Hai bit này sẽ bị xóa sau khi reset vi điều khiển và được thay đổi bởi chương trình của người lập trình.

Hai bit RS1, RS0 = 00, 01, 10, 11 sẽ được chọn Bank thanh ghi tích cực tương ứng là Bank0, Bank1, Bank2, Bank3.

| RS1 | RS0 | Bank thanh ghi được chọn |
|---|---|---|
| 0 | 0 | Bank 0 |
| 0 | 1 | Bank 1 |
| 1 | 0 | Bank 2 |
| 1 | 1 | Bank 3 |

*Cờ tràn OV (Over Flag) :*

Khi các số có dấu được cộng hoặc trừ với nhau, phần mềm có thể kiểm tra bit này để xác định xem kết quả có nằm trong vùng giá trị xác định hay không. Với số nhị phân 8 bit có dấu thì số dương từ 0 đến +127, số âm từ -128 đến – 1. Nếu kết quả cộng 2 số dương lớn hơn +127 hoặc cộng 2 số âm kết quả nhỏ hơn –128 thì kết quả đã vượt ra ngoài vùng giá trị cho phép thì khối ALU trong vi điều khiển sẽ làm bit OV = 1.

Khi cộng các số nhị phân không dấu thì không cần quan tâm đến bit OV.

*Bit Parity (P) :*

Bit P tự động được Set hay Clear ở mỗi chu kỳ máy để lập Parity chẵn với thanh ghi A. Đếm các bit 1 trong thanh ghi A cộng với bit Parity luôn luôn là số chẵn. Ví dụ thanh ghi A chứa nhị phân 10101101B thì bit P set lên một để cho biết tổng số bit 1 trong thanh ghi A và cả bit P tạo thành số chẵn.

Bit Parity thường được dùng kết hợp với những thủ tục truyền dữ liệu nối tiếp để tạo ra bit Parity cho dữ liệu trước khi truyền đi hoặc kiểm tra bit Parity sau khi nhận dữ liệu.

**Thanh ghi A:** Công dụng chứa dữ liệu của các phép toán mà vi điều khiển xử lý.
- Accumulator: Thanh ghi tích lũy
- Địa chỉ byte: E0H
- Địa chỉ bit: E0H – E7H

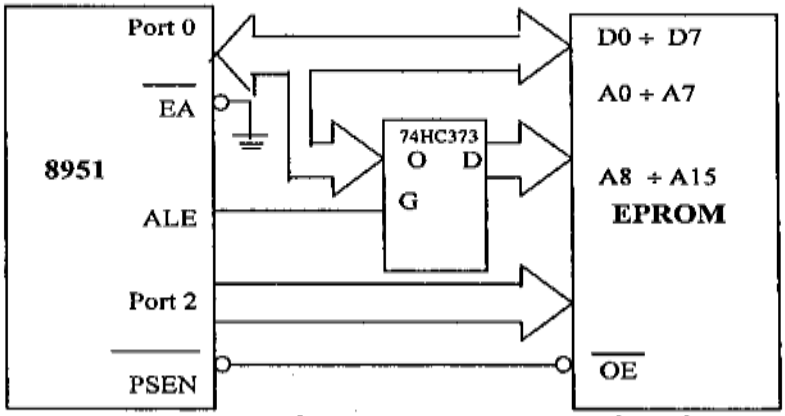
**Thanh ghi B:**
- Được sử dụng cùng với thanh ghi A cho các phép tóan nhân và chia hai số 8 bit
- Phép nhân 2 số 8 bit không dấu, kết quả là 16 bit.
  - Byte cao chứa vào TG B
  - Byte thấp chứa vào TG A.
- Phép chia 2 số 8 bit, thương số và số dư là số 8 bit.
  - Thương số chứa vào tg A.
  - Số dư chứa vào thanh ghi B

## 2.3 Mở rộng port cho 89C51

8951 có khả năng mở rộng bộ nhớ lên đến 64Kbyte bộ nhớ chương trình và 64Kbyte bộ nhớ dữ liệu ngoài. Do đó có thể dùng thêm RAM và ROM nếu cần.

Khi dùng bộ nhớ ngoài, Port 0 không còn chức năng I/O nữa. Nó được kết hợp giữa bus địa chỉ (A0-A7) và bus dữ liệu (D0-D7) với tín hiệu ALE để chốt byte của bus địa chỉ khi bắt đầu mỗi chu kỳ bộ nhớ. Port được cho là byte cao của bus địa chỉ.

***Truy xuất bộ nhớ mã ngoài (Accessing External Code Memory) :***
- Bộ nhớ chương trình bên ngoài là bộ nhớ ROM được cho phép của tín hiệu PSEN\.
- Sự kết nối phần cứng của bộ nhớ EPROM như sau:



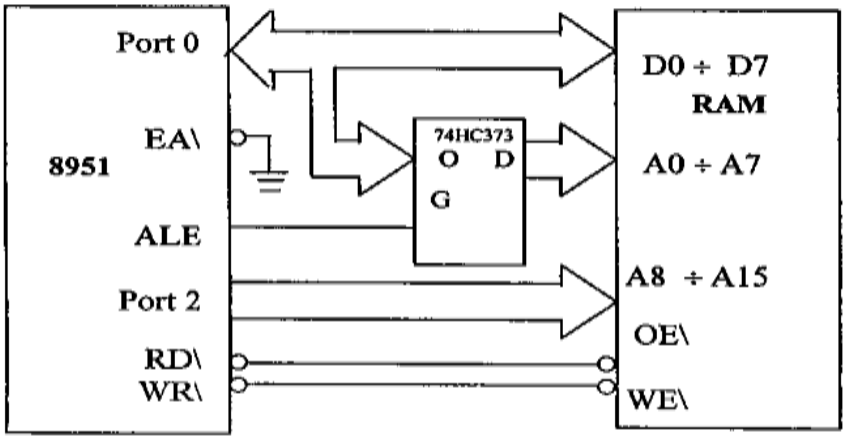

- Trong một chu kỳ máy tiêu biểu, tín hiệu ALE tích 2 lần. Lần thứ nhất cho phép

74HC373 mở cổng chốt địa chỉ byte thấp, khi ALE xuống 0 thì byte thấp và byte cao của bộ đếm chương trình đều có nhưng EPROM chưa xuất vì PSEN\ chưa tích cực, khi tín hiệu lên một trở lại thì Port 0 đã có dữ liệu là Opcode. ALE tích cực lần thứ hai được giải thích tương tự và byte 2 được đọc từ bộ nhớ chương trình. Nếu lệnh đang hiện hành là lệnh 1 byte thì CPU chỉ đọc Opcode, còn byte thứ hai bỏ đi.

**Truy xuất bộ nhớ dữ liệu ngoài (Accessing External Data Memory):**

– Bộ nhớ dữ liệu ngoài là một bộ nhớ RAM được đọc hoặc ghi khi được cho phép của tín hiệu RD\ và WR\. Hai tín hiệu này nằm ở chân P3.7 (RD\) và P3.6 (WR\). Lệnh MOVX được dùng để truy xuất bộ nhớ dữ liệu ngoài và dùng một bộ đệm dữ liệu 16 bit (DPTR), R0 hoặc R1 như là một thanh ghi địa chỉ.

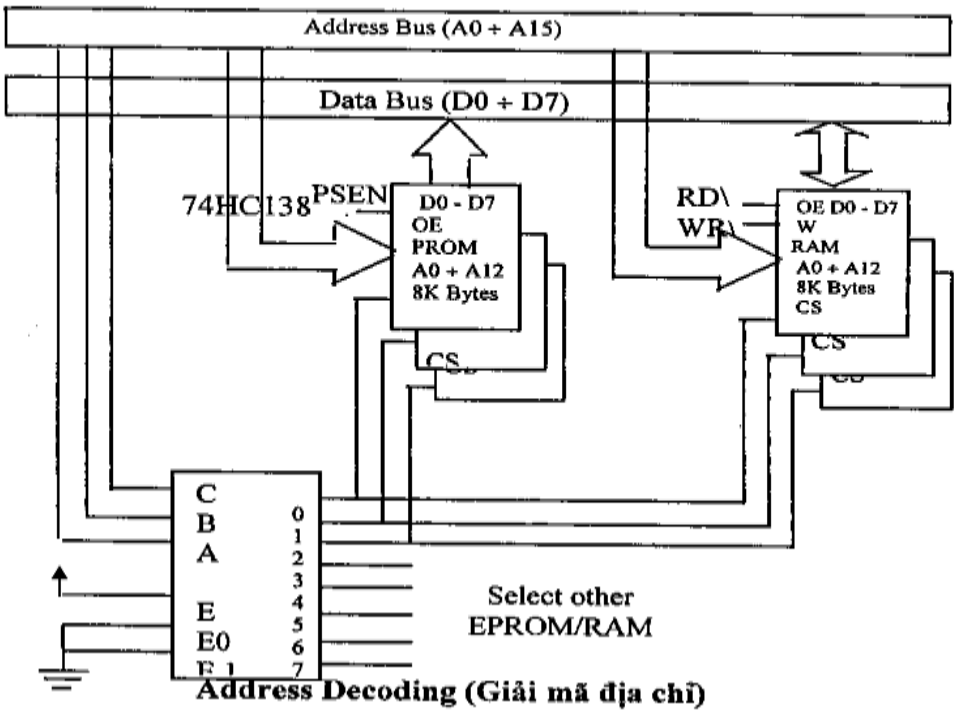
– Các RAM có thể giao tiếp với 8951 tương tự cách thức như EPROM ngoại trừ chân RD\ của 8951 nối với chân OE\ (Output Enable) của RAM và chân WR\ của 8951 nối với chân WE\ của RAM. Sự nối các bus địa chỉ và dữ liệu tương tự như cách nối của EPROM.




**Sự giải mã địa chỉ (Address Decoding):**

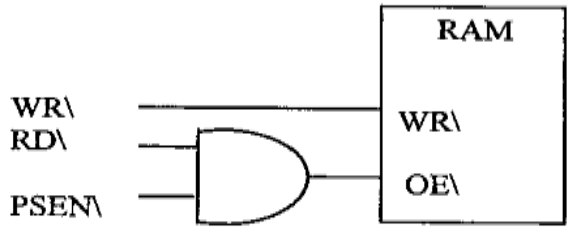
– Sự giải mã địa chỉ là một yêu cầu tất yếu để chọn EPROM, RAM, 8279, ... Sự giải mã địa chỉ đối với 8951 để chọn các vùng nhớ ngoài như các vi điều khiển. Nếu các con EPROM hoặc RAM 8K được dùng thì các bus địa chỉ phải được giải mã để chọn các IC nhớ nằm trong phạm vi giới hạn 8K: 0000H÷1FFFH, 2000H÷3FFFH, . .

– Một cách cụ thể, IC giải mã 74HC138 được dùng với những ngõ ra của nó được nối với những ngõ vào chọn Chip CS (Chip Select) trên những IC nhớ EPROM, RAM, ... Hình sau đây cho phép kết nối nhiều EPROM và RAM.




**Address Decoding (Giải mã địa chỉ)**

**Sự đè lên nhau của các vùng nhớ dữ liệu ngoài:**
Vì bộ nhớ chương trình là ROM, nên nảy sinh một vấn đề bất tiện khi phát triển phần mềm cho vi điều khiển. Một nhược điểm chung của 8951 là các vùng nhớ dữ liệu ngoài nằm đè lên nhau, vì tín hiệu PSEN\ được dùng để đọc bộ nhớ mã ngoài và tín hiệu RD\ được dùng để đọc bộ nhớ dữ liệu, nên một bộ nhớ RAM có thể chứa cả chương trình và dữ liệu bằng cách nối đường OE\ của RAM đến ngõ ra một cổng AND có hai ngõ vào PSEN\ và RD\. Sơ đồ mạch như hình sau cho phép cho phép bộ nhớ RAM có hai chức năng vừa là bộ nhớ chương trình vừa là bộ nhớ dữ liệu:




– Vậy một chương trình có thể được tải vào RAM bằng cách xem nó như bộ nhớ dữ liệu và thi hành chương trình bằng cách xem nó như bộ nhớ chương trình.

## Chương 3: Tập lệnh 89C51

Vi điều khiển hay vi xử lý là các IC lập trình, khi bạn đã thiết kế hệ thống điều khiển có sử dụng vi xử lý hay vi điều khiển thì đó mới chỉ là phần cứng, muốn hệ thống vận hành thì bạn phải viết một chương trình điều khiển nạp vào bộ nhớ nội bên trong vi điều khiển hoặc bộ nhớ bên ngoài và gắn vào trong hệ thống để hệ thống vận hành và dĩ nhiên bạn phải viết đúng thì hệ thống mới vận hành đúng. Chương trình gọi là phần mềm.

– Phần mềm và phần cứng có quan hệ với nhau, người lập trình phải hiểu rõ hoạt động của phần cứng để viết chương trình. Ở chương này sẽ trình bày chi tiết về tập lệnh của vi điều khiển giúp bạn hiểu rõ từng lệnh để bạn có thể lập trình được.

– ***Chương trình*** là một tập hợp các lệnh được tổ chức theo một trình tự hợp lí để giải quyết đúng các yêu cầu của người lập trình.

– Người lập trình là người biết giải thuật để viết chương trình và sắp xếp đúng các lệnh theo giải thuật. Người lập trình phải biết chức năng của tất cả các lệnh của vi điều khiển để viết chương trình.

– Tất cả các lệnh có thể có của một ngôn ngữ lập trình còn gọi là tập lệnh.

– Họ vi điều khiển MCS-51 đều có chung 1 tập lệnh, các vi điều khiển thế hệ sau chỉ phát triển nhiều về phần cứng còn lệnh thì ít mở rộng.

– Tập lệnh họ MCS-51 có mã lệnh 8 bit nên có khả năng cung cấp $2^8$= 256 lệnh.

– Trong toàn bộ tập lệnh của vi điều khiển có139 lệnh 1 byte, 92 lệnh 2 byte và 24 lệnh 3 byte.

– ***Lệnh*** của vi điều khiển là một số nhị phân 8 bit [còn gọi là mã máy]. 256 byte từ 0000 0000b đến 1111 1111b tương ứng với 256 lệnh khác nhau. Do mã lệnh dạng số nhị phân quá dài và khó nhớ nên các nhà lập trình đã xây dựng một ngôn ngữ lập trình Assembly cho dễ nhớ, điều này giúp cho việc lập trình được thực hiện một cách dễ dàng và nhanh chóng cũng như đọc hiểu và gỡ rối chương trình.

– Khi viết chương trình bằng ngôn ngữ lập trình Assembly thì vi điều khiển sẽ không thực hiện được mà phải dùng chương trình biên dịch Assembler để chuyển đổi các lệnh viết bằng Assembly ra mã lệnh nhị phân tương ứng rồi nạp vào bộ nhớ – khi đó vi điều khiển mới thực hiện được chương trình.

Ngôn ngữ lập trình Assembly do con người tạo ra, khi sử dụng ngôn ngữ Assembly để viết thì người lập trình vi điều khiển phải học hết tất cả các lệnh và viết đúng theo qui ước về cú pháp, trình tự sắp xếp dữ liệu để chương trình biên dịch có thể biên dịch đúng.

### 3.1 Các kiểu định địa chỉ

Các kiểu định địa chỉ cho phép định rõ nơi lấy dữ liệu hoặc nơi nhận dữ liệu tùy thuộc vào cách thức sử dụng lệnh của người lập trình.

Vi điều khiển họ MCS-51 có 8 kiểu định địa chỉ như sau:

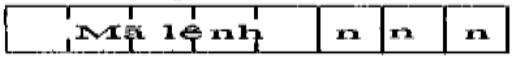
√ Kiểu định địa chỉ dùng thanh ghi.
√ Kiểu định địa chỉ trực tiếp.
√ Kiểu định địa chỉ gián tiếp.
√ Kiểu định địa chỉ tức thời.
√ Kiểu định địa chỉ tương đối.
√ Kiểu định địa chỉ tuyệt đối.
√ Kiểu định địa chỉ dài.
√ Kiểu định địa chỉ định vị.

**3.1.1 Kiểu định địa chỉ dùng thanh ghi (Register Addressing) :**

– Kiểu này thường được dùng cho các lệnh xử lý dữ liệu mà dữ liệu luôn lưu trong **các thanh ghi**.

– Đối với vi điều khiển thì mã lệnh thuộc kiểu này chỉ có 1 byte.

– Trong định địa chỉ thanh ghi, mã lệnh luôn luôn có 3 bit để chỉ thị một thanh ghi




| $n_2n_1n_0$ | Thanh ghi |
|---|---|
| 000 | R0 |
| 001 | R1 |
| 010 | R2 |
| 011 | R3 |
| 100 | R4 |
| 101 | R5 |
| 110 | R6 |
| 111 | R7 |

Ví dụ: Mov A,R1 ; copy nội dung thanh ghi R1 vào thanh ghi A

**3.1.2 Kiểu định địa chỉ trực tiếp (Direct Addressing) :**

– Kiểu này thường được dùng để truy xuất dữ liệu của bất kỳ ô nhớ nào trong 256 byte bộ nhớ RAM nội của vi điều khiển 89C51.

– Các lệnh thuộc kiểu này thường có mã lệnh 2 byte: byte thứ nhất là mã lệnh, byte thứ 2 là **địa chỉ của ô nhớ**


Ví dụ: Mov A,05H ; copy nội dung ô nhớ có địa chỉ 05H vào thanh ghi A

**3.1.3 Định địa chỉ gián tiếp thanh ghi (indirect Addressing):**

– Kiểu định địa chỉ gián tiếp được tượng trưng bởi ký hiệu @ và được đặt trước các thanh ghi R0, R1 hay DPTR. R0 và R1 có thể hoạt động như một thanh ghi con trỏ, nội dung của nó cho biết địa chỉ của một ô nhớ trong RAM nội mà dữ liệu sẽ ghi hoặc sẽ đọc. Còn dptr dùng để truy xuất ô nhớ ngoại.

– Các lệnh thuộc dạng này chỉ có 1 byte.


Ví dụ: Mov A,@R1 ; copy nội dung ô nhớ có địa chỉ trong thanh ghi R1 vào thanh ghi A

**3.1.4 Định địa chỉ tức thời**

– Kiểu định địa chỉ tức thời được tượng trưng bởi ký hiệu # và được đặt trước một hằng số.

– Lệnh này thường dùng để nạp 1 giá trị là 1 hằng số ở byte thứ 2 (hoặc byte thứ 3) vào thanh ghi hoặc ô nhớ.

Ví dụ: Mov a,#30H ; nạp dữ liệu là con số 30H vào thanh ghi A

**3.1.5 Kiểu định địa chỉ tương đối:**

– Kiểu định địa chỉ tương đối chỉ sử dụng với những lệnh nhảy. Nơi nhảy đến có địa chỉ bằng địa chỉ đang lưu trong thanh ghi PC cộng với 1 giá trị 8 bit [còn gọi là giá trị lệch tương đối: relative offset] có giá trị từ – 128 đến +127 nên vi điều khiển có thể nhảy lùi [nếu số cộng với số âm] và nhảy tới [nếu số cộng với số dương].

– Lệnh này có mã lệnh 2 byte, byte thứ 2 chính là giá trị lệch tương đối:


– Nơi nhảy đến thường được xác định bởi nhãn (label) và trình biên dịch sẽ tính toán giá trị lệch.

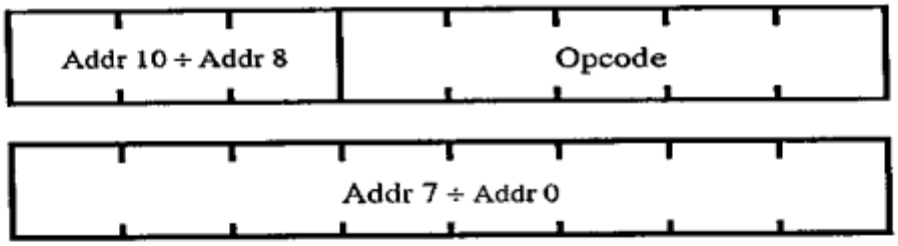
– Định vị tương đối có ưu điểm là mã lệnh cố định, nhưng khuyết điểm là chỉ nhảy ngắn trong phạm vi -128÷127 byte [256byte], nếu nơi nhảy đến xa hơn thì lệnh này không đáp ứng được – sẽ có lỗi.

Ví dụ: Sjmp X1 ;nhảy đến nhản có tên là X1 nằm trong tầm vực

**3.1.6 Định địa chỉ tuyệt đối(Absolute Addressing):**

– Kiểu định địa chỉ tuyệt đối được dùng với các lệnh ACALL và AJMP.

– Các lệnh này có mã lệnh 2 byte cho phép phân chia bộ nhớ theo trang - mỗi trang có kích thước đúng bằng 2Kbyte so với giá trị chứa trong thanh ghi PC hiện hành. 11 bit địa chỉ A10÷A0 được thay thế cho 11 địa chỉ thấp trong thanh ghi PC nằm trong cấu trúc mã lệnh như sau:



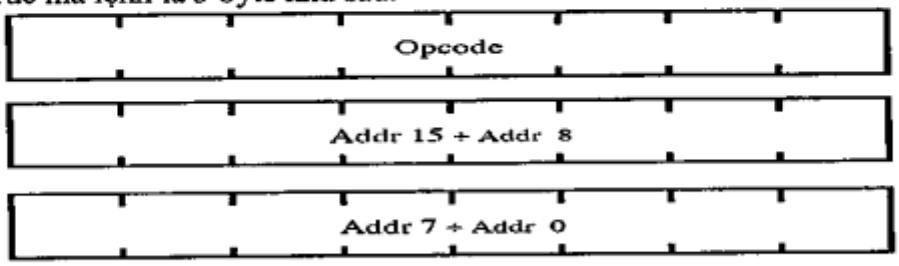

– Định địa chỉ tuyệt đối có ưu điểm là mã lệnh ngắn (2 byte), nhưng khuyết điểm là mã lệnh thay đổi và giới hạn phạm vi nơi nhảy đến, gọi đến không quá 2 kbyte.

Ví dụ: Ajmp X1 ;nhảy đến nhản có tên là X1 nằm trong tầm vực 2 kbyte

**3.1.7 Định địa chỉ dài (Long Addressing):**

– Kiểu định địa chỉ dài được dùng với lệnh LCALL và LJMP.

– Các lệnh này có mã lệnh 3 byte – trong đó có 2 byte (16bit) là địa chỉ của nơi đến. Cấu trúc mã lệnh là 3 byte như sau:




– Ưu điểm của định địa chỉ dài là có thể gọi 1 chương trình con hoặc có thể nhảy

đến bất kỳ vùng nhớ nào vùng nhớ 64K, nhược điểm là các lệnh kiểu này dài 3 byte và phụ thuộc vào vị trí đến – điều này sẽ bất tiện bởi không thể dời toàn bộ mã lệnh của chương trình từ vùng nhớ này sang các vùng nhớ khác – có nghĩa là khi chương trình đã viết nơi đến tại địa chỉ 1000h thì sau khi dịch ra mã lệnh dạng số nhị phân thì sau đó nạp vào bộ nhớ thì địa chỉ bắt đầu phải đúng với địa chỉ đã viết là 1000h; nếu nạp ở vùng địa chỉ khác địa chỉ 1000h thì chương trình sẽ thực hiện sai.
Ví dụ: Ljmp X1 ;nhảy đến nhãn có tên là X1 nằm trong tầm vực 64kbyte

**3.1.8 Định địa chỉ chỉ số (Index Addressing):**

– Kiểu định địa chỉ chỉ số "dùng một thanh ghi cơ bản: là bộ đếm chương trình PC hoặc bộ đếm dữ liệu DPTR" kết hợp với "một giá trị lệch (offset) còn gọi là giá trị tương đối [thường lưu trong thanh ghi]" để tạo ra 1 địa chỉ của ô nhớ cần truy xuất hoặc là địa chỉ của nơi nhảy đến. Việc kết hợp được minh họa như sau:

| Base Register | | Offset | | Effective Address |
|---|---|---|---|---|
| PC (or PDTR) | + | A | = | |

Ví dụ: MOVX A, @A + DPTR ;lấy dữ liệu trong ô nhớ có địa chỉ bằng DPTR + A

## 3.2 Tập lệnh 89C51

Thống nhất một số qui định về các từ ngữ kí hiệu trong tập lệnh thường được sử dụng:

– Direct tượng trưng cho ô nhớ nội có địa chỉ trực tiếp
– Rn tượng trưng cho các thanh ghi từ thanh ghi R0 đến thanh ghi R7.
– @Ri tượng trưng cho ô nhớ có địa chỉ lưu trong thanh ghi Ri và Ri chỉ có 2 thanh ghi là R0 và R1.

Các lệnh thường xảy ra giữa các đối tượng sau:

– Thanh ghi A.
– Thanh ghi Rn.
– Ô nhớ có địa chỉ direct.
– Ô nhớ có địa chỉ lưu trong thanh ghi @Ri.
– Dữ liệu 8 bit #data.
– Addr11 là địa chỉ 11 bit từ A11 – A0: địa chỉ này phục vụ cho lệnh nhảy hoặc lệnh gọi chương trình con trong phạm vi 2 kbyte.
– Addr16 là địa chỉ 16 bit từ A15 – A0: địa chỉ này phục vụ cho lệnh nhảy và lệnh gọi chương trình con ở xa trong phạm vi 64 kbyte – đó chính là địa chỉ nhảy đến, hoặc địa chỉ của chương trình con.
– Khi viết chương trình người lập trình có thể thay thế địa chỉ bằng nhãn (label) để khỏi phải tính toán các địa chỉ cụ thể. Nhãn (label) sẽ được đặt tại vị trí addr thay cho addr và phải có một nhãn đặt tại nơi muốn nhảy đến - gọi là 1 cặp nhãn cùng tên. Có thể nhiều nơi nhảy đến cùng một nhãn. Không được đặt các nhãn cùng tên. Các lệnh có ảnh hưởng đến thanh ghi trạng thái thì có trình bày trong lệnh, còn các lệnh không đề cập đến thanh ghi trạng thái thì có nghĩa là nó không ảnh hưởng.

### 3.2.1 Nhóm lệnh chuyển dữ liệu(8 bit) :

**1. Lệnh chuyển dữ liệu từ một thanh ghi vào thanh ghi A:**

– Cú pháp : **Mov A,Rn**
– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | n2 | n1 | n0 |

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
– Chức năng: Chuyển nội dung của thanh ghi Rn vào thanh ghi A, nội dung thanh ghi Rn vẫn giữ nguyên.

Ví dụ: Giả sử thanh ghi R0 có nội dung là 32h , lệnh:
Mov A, R0 ; kết quả như sau: (A) = 32h, (R0) = 32h.

**2. Lệnh chuyển dữ liệu từ ô nhớ trực tiếp vào thanh ghi A :**
– Cú pháp : **Mov A, direct**
– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
– Chức năng: Chuyển nội dung của ô nhớ trong Ram nội có địa chỉ trực tiếp ở byte thứ hai vào thanh ghi A. Trực tiếp có nghĩa là địa chỉ của ô nhớ được ghi ở trong lệnh.

Ví dụ : Giả sử ô nhớ có địa chỉ 30h lưu nội dung 32h. Lệnh:
Mov A, 30h ;chuyển nội dung của ô nhớ có địa chỉ là 30h sang thanh ghi A.
;Kết quả như sau: (A)= 32h.

**3. Lệnh chuyển dữ liệu từ ô nhớ gián tiếp vào thanh ghi A :**
– Cú pháp : **MOV A,@Ri**
– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | i |

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
– Chức năng: Chuyển nội dung ô nhớ trong Ram nội, có địa chỉ chứa trong thanh ghi Ri, vào thanh ghi A.

Ví dụ : Giả sử R0 có nội dung là 70h, ô nhớ có địa chỉ 70h chứa nội dung là 0B8h. Lệnh: Mov A,@R0 ; kết quả như sau: (A) = 0B8h.

**4. Lệnh nạp dữ liệu 8 bit vào thanh ghi A :**
– Cú pháp : **MOV A, #data**
– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
– Chức năng: Nạp dữ liệu 8 bit data (d0 đến d7) vào thanh ghi A.

Ví dụ : Giả sử A có nội dung 47h, dữ liệu trực tiếp là 32h, lệnh:
MOV A, #32h ;kết quả như sau: (A) = 32h.
;d7..d0 = 00110010b

**5. Lệnh chuyển dữ liệu từ thanh ghi A vào thanh ghi :**
– Cú pháp : **Mov Rn, A**
– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | n2 | n1 | n0 |

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
– Chức năng: Chuyển nội dung của thanh ghi A vào thanh ghi Rn.

Ví dụ : Giả sử A có nội dung 47h , lệnh:

MOV R0, A ; kết quả như sau: (R0) = 47h, (A) = 47h.

**6. Lệnh chuyển dữ liệu từ ô nhớ trực tiếp vào thanh ghi Rn :**

– Cú pháp : **MOV Rn, direct**

– Mã lệnh :

| 1 | 0 | 1 | 0 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Chuyển nội dung của ô nhớ trong Ram nội có địa chỉ direct vào thanh ghi Rn.

Ví dụ : Giả sử R1 có nội dung 47h, ô nhớ có địa chỉ 30h chứa nội dung 0afh.

Lệnh: MOV R1, 30h Lệnh chuyển nội dung ô nhớ có địa chỉ 30h sang thanh ghi R1.

Kết quả như sau: (R1) = 0afh, dữ liệu trong ô nhớ có địa chỉ 30h không đổi.

**7. Lệnh chuyển tức thời dữ liệu 8 bit vào thanh ghi Rn :**

– Cú pháp : **MOV Rn, #data**

– Mã lệnh :

| 0 | 1 | 1 | 1 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nạp dữ liệu 8 bit data (d0 đến d7) vào thanh ghi Rn.

Ví dụ: Giả sử muốn chuyển dữ liệu 47h vào thanh ghi R1:

MOV R1,#47h ; kết quả như sau: (R1)= 47h.

**8. Lệnh chuyển dữ liệu từ thanh ghi A vào ô nhớ trực tiếp :**

– Cú pháp : **MOV direct, A**

– Mã lệnh :

| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Chuyển nội dung của thanh ghi A vào ô nhớ trong Ram nội có địa chỉ direct.

Ví dụ : Cho nội dung thanh ghi (A ) = 35H, nội dung ô nhớ có địa chỉ 10H bằng 50H.

MOV 10h, A ;Sau khi thực hiện xong thì nội dung ô nhớ có địa chỉ 10h bằng 35H.

**9. Lệnh chuyển dữ liệu từ thanh ghi Rn vào ô nhớ trực tiếp :**

– Cú pháp : **MOV direct, Rn**

– Mã lệnh :

| 1 | 0 | 0 | 0 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Chuyển nội dung của thanh ghi Rn vào ô nhớ trong Ram nội có địa chỉ direct.

Ví dụ : Cho nội dung thanh ghi (R0 ) = 35H, nội dung ô nhớ 10H bằng 50H.

MOV 10h,R0

Sau khi thực hiện xong thì nội dung ô nhớ có địa chỉ 10h bằng 35H.

**10. Lệnh chuyển dữ liệu từ ô nhớ trực tiếp vào ô nhớ trực tiếp :**

– Cú pháp : **MOV direct, direct**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Chuyển nội dung của ô nhớ trong Ram nội có địa chỉ direct vào ô nhớ có địa chỉ trực direct.

Ví dụ : Cho nội dung ô nhớ có địa chỉ 20H bằng 35H và nội dung ô nhớ có địa chỉ 10H bằng 50H.

MOV 10h,20h

Sau khi thực hiện xong thì nội dung ô nhớ có địa chỉ 10h bằng 35H.

**11. Lệnh chuyển dữ liệu từ ô nhớ gián tiếp vào ô nhớ trực tiếp :**

– Cú pháp : **MOV direct, @Ri**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | i |
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Chuyển nội dung ô nhớ có địa chỉ chứa trong thanh ghi Ri vào ô nhớ có địa chỉ direct.

Ví dụ : Cho nội dung thanh ghi (R0 ) = 05H, nội dung ô nhớ có địa chỉ 05h bằng FFH và nội dung ô nhớ có địa chỉ 10H bằng 50H.

MOV 10h,@r0

Sau khi thực hiện xong thì nội dung ô nhớ có địa chỉ 10h bằng FFH.

**12. Lệnh chuyển dữ liệu vào ô nhớ trực tiếp :**

– Cú pháp : **MOV direct, #data**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Nạp dữ liệu data 8 bit (d0 đến d7 ) vào ô nhớ có địa chỉ direct.

Ví dụ : Cho nội dung ô nhớ có địa chỉ 05h bằng FFH.

MOV 05h,#25H

Sau khi thực hiện xong thì nội dung ô nhớ có địa chỉ 05h bằng 25H.

**13. Lệnh chuyển dữ liệu từ thanh ghi A vào ô nhớ gián tiếp :**

– Cú pháp : **MOV @Ri, A**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | i |

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : Chuyển nội dung của thanh ghi A vào ô nhớ trong Ram nội có địa

chỉ chứa trong thanh ghi Ri.

**14. Lệnh chuyển dữ liệu từ ô nhớ trực tiếp vào ô nhớ gián tiếp :**

– Cú pháp : **MOV @Ri, direct**

– Mã lệnh :

| 1 | 0 | 1 | 0 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Chuyển nội dung ô nhớ có địa chỉ direct vào ô nhớ có địa chỉ chứa trong thanh ghi Ri.

**15. Lệnh chuyển dữ liệu tức thời vào ô nhớ gián tiếp :**

– Cú pháp : **MOV @Ri, #data**

– Mã lệnh :

| 0 | 1 | 1 | 1 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nạp dữ liệu data 8 bit (d0 đến d7) vào ô nhớ có địa chỉ chứa trong thanh ghi Ri.

**16. Lệnh chuyển dữ liệu tức thời 16 bit vào thanh ghi con trỏ dữ liệu :**

– Cú pháp : **MOV dptr, #data16**

– Mã lệnh :

1 0 0 1 0 0 0 0

d7 d6 d5 d4 d3 d2 d1 d0

| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| d15 | d14 | d13 | d12 | d11 | d10 | d9 | d8 |
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Nạp dữ liệu data 16 bit vào thanh ghi con trỏ dữ liệu dptr.

**17. Lệnh chuyển dữ liệu từ ô nhớ có địa chỉ là Dptr + A vào thanh ghi A :**

– Cú pháp : **MOVC A,@A+DPTR**

– Mã lệnh :

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng : chuyển nội dung của ô nhớ ngoài, có địa chỉ chứa bằng dptr cộng với giá trị chứa trong A, chuyển vào thanh ghi A.

**18. Lệnh chuyển dữ liệu từ ô nhớ có địa chỉ là PC + A vào thanh ghi A :**

– Cú pháp : **MOVC A,@A+PC**

– Mã lệnh :

| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng : chuyển nội dung của ô nhớ ngoài có địa chỉ chứa bằng PC cộng với giá trị chứa trong A được chuyển vào thanh ghi A.

**19. Lệnh chuyển dữ liệu từ ô nhớ ngoài gián tiếp (8 bit địa chỉ) vào thanh ghi A :**

– Cú pháp : **MOVX A, @Ri**

– Mã lệnh :

| 1 | 1 | 1 | 0 | 0 | 0 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng : chuyển nội dung ô nhớ ngoài có địa chỉ chứa trong thanh ghi Ri vào thanh ghi A.

**20. Lệnh chuyển dữ liệu từ ô nhớ ngoài gián tiếp (16 bit địa chỉ) vào thanh ghi A :**

– Cú pháp : **MOVX A,@DPTR**

– Mã lệnh :

| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng : chuyển nội dung của ô nhớ ngoài có địa chỉ chứa trong thanh ghi dptr vào thanh ghi A.

**21. Lệnh chuyển dữ liệu từ thanh ghi A vào ô nhớ ngoài gián tiếp (8 bit địa chỉ) :**

– Cú pháp : **MOVX @ Ri, A**

– Mã lệnh :

| 1 | 1 | 1 | 1 | 0 | 0 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng : chuyển nội dung của thanh ghi A ra ô nhớ ngoài có địa chỉ chứa trong thanh ghi Ri.

**22. Lệnh chuyển dữ liệu từ thanh ghi A vào ô nhớ ngoài gián tiếp (16 bit địa chỉ) :**

– Cú pháp : **MOVX @DPTR, A**

– Mã lệnh :

| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng : Chuyển nội dung của thanh ghi A ra ô nhớ ngoài có địa chỉ chứa trong thanh ghi dptr.

**23. Lệnh cất nội dung ô nhớ trực tiếp vào ngăn xếp :**

– Cú pháp : **PUSH direct**

– Mã lệnh :

| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: cất nội dung của ô nhớ có địa chỉ direct vào ô nhớ ngăn xếp. Con trỏ ngăn xếp SP tăng lên 1 trước khi lưu nội dung.

**24. Lệnh lấy dữ liệu từ ngăn xếp trả về ô nhớ trực tiếp :**

– Cú pháp : **POP direct**

– Mã lệnh :

| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: lấy nội dung của ô nhớ ngăn xếp trả cho ô nhớ có địa chỉ direct.

con trỏ ngăn xếp SP giảm 1 sau khi lấy dữ liệu ra.

**25. Lệnh trao đổi dữ liệu giữa thanh ghi với thanh ghi A :**

– Cú pháp : **XCH A,Rn**

– Mã lệnh :

| 1 | 1 | 0 | 0 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : Trao đổi nội dung của thanh ghi Rn với thanh ghi A.

Ví dụ : cho nội dung của thanh ghi (A) = 35H và (R0) = 70H

Lệnh XCH A, R0

Kết quả sau khi thực hiện (A) = 70H và (R0) = 35H

**26. Lệnh trao đổi dữ liệu giữa ô nhớ trực tiếp với thanh ghi A :**

– Cú pháp : **XCH A,Direct**

– Mã lệnh :

| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : Trao đổi nội dung của thanh ghi A với nộ dung ô nhớ có địa chỉ direct.

**27. Lệnh trao đổi dữ liệu giữa ô nhớ gián tiếp với thanh ghi A :**

– Cú pháp : **XCH A,@Ri**

– Mã lệnh :

| 1 | 1 | 0 | 0 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : Trao đổi nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri với thanh ghi A.

**28. Lệnh trao đổi 4 bit dữ liệu giữa ô nhớ gián tiếp với thanh ghi A :**

– Cú pháp : **XCHD A,@Ri**

– Mã lệnh :

| 1 | 1 | 0 | 1 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : Trao đổi dữ liệu 4 bit thấp của ô nhớ có địa chỉ chứa trong thanh ghi Ri với dữ liệu 4 bit thấp trong thang ghi A.

### 3.2.2 Nhóm lệnh số học

**1. Lệnh cộng thanh ghi A với thanh ghi :**

– Cú pháp : **ADD A,Rn**

– Mã lệnh :

| 0 | 0 | 1 | 0 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : cộng nội dung thanh ghi A với nội dung thanh ghi Rn, kết quả lưu trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ 1: Giả sử A có nội dung 47h và R0 có nội dung là 32h, lệnh:

ADD A, R0 ;kết quả như sau: (A) = 79h, (C) = 0.

Ví dụ 2: Giả sử A có nội dung 0D9h và R0 có nội dung là 0B8h, lệnh:

ADD A, R0 ;kết quả như sau: (A) = 91h, (C) =1.

**2. Lệnh cộng nội dung ô nhớ trực tiếp vào thanh ghi A :**

– Cú pháp : **ADD A, direct**

– Mã lệnh :

| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Cộng nội dung của ô nhớ có địa chỉ direct với nội dung thanh ghi A, kết quả chứa ở thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ 1: Giả sử A có nội dung 0D9h và ô nhớ có địa chỉ 30h lưu nội dung 0B8h, lệnh:

ADD A,30h ;kết quả như sau: (A) = 81h, (C) =1.

Ví dụ 2: Giả sử A có nội dung 47h và ô nhớ có địa chỉ 30h lưu nội dung 32h, lệnh:

ADD A,30h ;kết quả như sau: (A) = 79h, (C) =0.

**3. Lệnh cộng nội dung ô nhớ gián tiếp vào thanh ghi A :**

– Cú pháp: **ADD A,@Ri**

– Mã lệnh:

| 0 | 0 | 1 | 0 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: cộng nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri với thanh ghi A, kết qủa lưu trữ trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ: Giả sử A có nội dung 0D9h, ô nhớ có địa chỉ 30h có nội dung là 0B8h, R0 có nội dung là 30h, lệnh:

ADD A,@R0 ;kết qủa như sau: (A) = 91h, (C) =1.

**4. Lệnh cộng dữ liệu tức thời 8 bit vào thanh ghi A :**

– Cú pháp : **ADD A, #data**

– Mã lệnh :

| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Cộng dữ liệu data 8 bit (d0 đến d7) với nội dung thanh ghi A, kết quả lưu trữ trong A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ1: Giả sử A có nội dung 47h, dữ liệu trực tiếp là 32h, lệnh:

ADD A,#32h ;kết quả như sau: (A) = 79h, (C) = 0.

Ví dụ2: Giả sử A có nội dung D9h, dữ liệu trực tiếp là B8h, lệnh:

ADD A,#0B8h ; kết quả như sau: (A) = 91h, (C) = 1.

**5. Lệnh cộng thanh ghi A với thanh ghi có bit carry :**

– Cú pháp : **ADDC A,Rn**

– Mã lệnh :

| 0 | 0 | 1 | 1 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : cộng nội dung thanh ghi A với nội dung thanh ghi Rn với bit C, kết quả lưu trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ: Giả sử A có nội dung 47h, R1 có nội dung 32h và cờ (C) = 1, lệnh:

ADDC A, R1 ;kết quả như sau: (A) = 7ah, (C) = 0.

Ví dụ: Giả sử A có nội dung 0D9h, R0 có nội dung là 0B8h, (C) =1, lệnh:

ADDC A,R0 ;kết quả như sau: (A) = 92h, (C)=1.

**6. Lệnh cộng nội dung ô nhớ trực tiếp vào thanh ghi A có bit carry :**

– Cú pháp : **ADDC A, direct**

– Mã lệnh :

| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Cộng nội dung của ô nhớ có địa direct nội dung thanh ghi A và bit C, kết quả chứa ở thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ 1: Giả sử A có nội dung 47h, ô nhớ 30h có nội dung 32h và cờ (C) = 0, lệnh:

ADDC A,30h ;kết quả như sau: (A) = 79h, (C) = 0.

Ví dụ 2: Giả sử A có nội dung 0D9h, ô nhớ 30h có nội dung là 0B8h, C:=1, lệnh:

ADDC A,30h ;kết quả như sau: (A) = 92h, (C) = 1.

**7. Lệnh cộng nội dung ô nhớ gián tiếp vào thanh ghi A có bit carry :**

– Cú pháp : **ADDC A,@Ri**

– Mã lệnh :

| 0 | 0 | 1 | 1 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : cộng nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri với thanh ghi A với bit C, kết quả lưu trữ trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ: Giả sử A có nội dung 47h, ô nhớ 30h có nội dung 32h, R0 có nội dung là 30h và cờ (C) = 0, lệnh:

ADDC A,@R0 ;kết quả như sau: (A) = 79h, (C) = 0.

Ví dụ: Giả sử A có nội dung 0D9h, ô nhớ 30h có nội dung là 0B8h, R0 có nội dung 30h và (C) =1, lệnh:

ADDC A,@R0 ;kết quả như sau: (A) = 92h, (C)=1.

**8. Lệnh cộng dữ liệu 8 bit vào thanh ghi A có bit carry :**

– Cú pháp : **ADDC A, #data**

– Mã lệnh :

| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy.

– Chức năng: Cộng dữ liệu data 8 bit (d0 đến d7) với nội dung thanh ghi A và bit C, kết quả lưu trữ trong A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

**9. Lệnh trừ thanh ghi A với thanh ghi :**

– Cú pháp : **SUBB A,Rn**

– Mã lệnh :

| 1 | 0 | 0 | 1 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : Trừ nội dung thanh ghi A với nội dung thanh ghi Rn và trừ cho cờ Carry, kết quả lưu trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ: Giả sử A có nội dung 47h, nội dung thanh ghi R0 là 32h và cờ (C)=0, lệnh:

SUBB A,R0 ;kết quả như sau: (A) = 15h (C)=0.

Ví dụ: Giả sử A có nội dung 0B9h, thanh ghi R0 có nội dung là 5Ah và (C)=1, lệnh:

SUBB A,R0 ;kết quả như sau: (A) = 5Eh, (C) =0.

**10. Lệnh trừ nội dung thanh ghi A cho nội dung ô nhớ trực tiếp :**

– Cú pháp : **SUBB A, direct**

– Mã lệnh :

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Trừ nội dung thanh ghi A cho nội dung của ô nhớ có địa chỉ direct và trừ cho cờ Carry, kết quả chứa ở thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

**11. Lệnh trừ nội dung thanh ghi A cho nội dung ô nhớ gián tiếp :**

– Cú pháp : **SUBB A,@Ri**

– Mã lệnh :

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Trừ nội dung của thanh ghi A cho dữ liệu của ô nhớ có địa chỉ chứa trong thanh ghi Ri và trừ cho cờ carry, kết qủa lưu trữ trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

**12. Lệnh trừ nội dung thanh ghi A cho dữ liệu tức thời 8 bit :**

– Cú pháp : **SUBB A, #data (subbtract: trừ)**

– Mã lệnh :

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Trừ nội dung thanh ghi A cho dữ liệu 8 bit d0 đến d7 và trừ cho cờ carry, kết qủa lưu trữ trong A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

**13. Lệnh tăng nội dung thanh ghi A :**

– Cú pháp : **INC A (increment: tăng lên 1 đơn vị)**

– Mã lệnh :

| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Tăng nội dung thanh ghi A lên 1.

Ví dụ: Giả sử A có nội dung 35h, lệnh:

INC A ;kết qủa như sau: (A) = 36h.

Ví dụ: Giả sử A có nội dung FFh, lệnh:

INC A ;kết qủa như sau: (A) = 00h.

**14. Lệnh tăng nội dung của thanh ghi :**

– Cú pháp : **INC Rn**

– Mã lệnh :

| 0 | 0 | 0 | 0 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Tăng nội dung thanh ghi Rn lên 1.

**15. Lệnh tăng nội dung ô nhớ trực tiếp :**

– Cú pháp : **INC direct**

– Mã lệnh :

| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Cú pháp : **ADDC A, direct**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Cộng nội dung của ô nhớ có địa direct nội dung thanh ghi A và bit C, kết quả chứa ở thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ 1: Giả sử A có nội dung 47h, ô nhớ 30h có nội dung 32h và cờ (C) = 0, lệnh:

ADDC A,30h ;kết quả như sau: (A) = 79h, (C) = 0.

Ví dụ 2: Giả sử A có nội dung 0D9h, ô nhớ 30h có nội dung là 0B8h, C:=1, lệnh:

ADDC A,30h ;kết quả như sau: (A) = 92h, (C) = 1.

**7. Lệnh cộng nội dung ô nhớ gián tiếp vào thanh ghi A có bit carry :**

– Cú pháp : **ADDC A,@Ri**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | i |

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : cộng nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri với thanh ghi A với bit C, kết quả lưu trữ trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ: Giả sử A có nội dung 47h, ô nhớ 30h có nội dung 32h, R0 có nội dung là 30h và cờ (C) = 0, lệnh:

ADDC A,@R0 ;kết quả như sau: (A) = 79h, (C) = 0.

Ví dụ: Giả sử A có nội dung 0D9h, ô nhớ 30h có nội dung là 0B8h, R0 có nội dung 30h và (C) =1, lệnh:

ADDC A,@R0 ;kết quả như sau: (A) = 92h, (C)=1.

**8. Lệnh cộng dữ liệu 8 bit vào thanh ghi A có bit carry :**

– Cú pháp : **ADDC A, #data**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy.

– Chức năng: Cộng dữ liệu data 8 bit (d0 đến d7) với nội dung thanh ghi A và bit C, kết quả lưu trữ trong A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

**9. Lệnh trừ thanh ghi A với thanh ghi :**

– Cú pháp : **SUBB A,Rn**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | n2 | n1 | n0 |

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng : Trừ nội dung thanh ghi A với nội dung thanh ghi Rn và trừ cho cờ Carry, kết quả lưu trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ: Giả sử A có nội dung 47h, nội dung thanh ghi R0 là 32h và cờ (C)=0, lệnh:

SUBB A,R0 ;kết quả như sau: (A) = 15h (C)=0.

Ví dụ: Giả sử A có nội dung 0B9h, thanh ghi R0 có nội dung là 5Ah và (C)=1, lệnh:

SUBB A,R0 ;kết quả như sau: (A) = 5Eh, (C) =0.

**10. Lệnh trừ nội dung thanh ghi A cho nội dung ô nhớ trực tiếp :**

– Cú pháp : **SUBB A, direct**

– Mã lệnh :

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Trừ nội dung thanh ghi A cho nội dung của ô nhớ có địa chỉ direct và trừ cho cờ Carry, kết quả chứa ở thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

**11. Lệnh trừ nội dung thanh ghi A cho nội dung ô nhớ gián tiếp :**

– Cú pháp : **SUBB A,@Ri**

– Mã lệnh :

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Trừ nội dung của thanh ghi A cho dữ liệu của ô nhớ có địa chỉ chứa trong thanh ghi Ri và trừ cho cờ carry, kết qủa lưu trữ trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

**12. Lệnh trừ nội dung thanh ghi A cho dữ liệu tức thời 8 bit :**

– Cú pháp : **SUBB A, #data (subbtract: trừ)**

– Mã lệnh :

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Trừ nội dung thanh ghi A cho dữ liệu 8 bit d0 đến d7 và trừ cho cờ carry, kết qủa lưu trữ trong A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

**13. Lệnh tăng nội dung thanh ghi A :**

– Cú pháp : **INC A (increment: tăng lên 1 đơn vị)**

– Mã lệnh :

| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Tăng nội dung thanh ghi A lên 1.

Ví dụ: Giả sử A có nội dung 35h, lệnh:

INC A ;kết qủa như sau: (A) = 36h.

Ví dụ: Giả sử A có nội dung FFh, lệnh:

INC A ;kết qủa như sau: (A) = 00h.

**14. Lệnh tăng nội dung của thanh ghi :**

– Cú pháp : **INC Rn**

– Mã lệnh :

| 0 | 0 | 0 | 0 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Tăng nội dung thanh ghi Rn lên 1.

**15. Lệnh tăng nội dung ô nhớ trực tiếp :**

– Cú pháp : **INC direct**

– Mã lệnh :

| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Tăng nội dung của ô nhớ có địa chỉ trực tiếp ở byte thứ 2 lên 1.

**16. Lệnh tăng nội dung ô nhớ gián tiếp :**

– Cú pháp : **INC @Ri**

– Mã lệnh :

| 0 | 0 | 0 | 0 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Tăng nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri lên 1.

Ví dụ : Giả sử nội dung ô nhớ 30h là 35h, thanh ghi R0 có nội dung là 30h, lệnh:
INC R0 ;kết quả như sau: ô nhớ (30h) = 36h

**17. Lệnh tăng nội dung con trỏ dữ liệu Dptr :**

– Cú pháp : **INC dptr**

– Mã lệnh :

| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Tăng nội dung của thanh ghi con trỏ dữ liệu dptr lên 1.

**18. Lệnh giảm nội dung thanh ghi A :**

– Cú pháp : **DEC A**

– Mã lệnh :

| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Giảm nội dung thanh ghi A xuống 1.

Ví dụ: Giả sử A có nội dung 35h, lệnh:
DEC A ;kết qủa như sau: (A) = 34h.

Ví dụ: Giả sử A có nội dung 00h, lệnh:
DEC A ;kết qủa như sau: (A) = FFh.

**19. Lệnh giảm nội dung của thanh ghi :**

– Cú pháp : **DEC Rn**

– Mã lệnh :

| 0 | 0 | 0 | 1 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy.

– Chức năng: Giảm nội dung thanh ghi Rn xuống 1.

Ví dụ : Giả sử R0 là 35h , lệnh:
DEC R0 ;kết quả như sau: (R0) =34h.

**20. Lệnh giảm nội dung ô nhớ trực tiếp :**

– Cú pháp : **DEC direct**

– Mã lệnh :

| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Giảm nội dung của ô nhớ có địa chỉ direct ở byte thứ 2 xuống 1.

Ví dụ: Giả sử ô nhớ 30h có nội dung là 35h , lệnh:
DEC 30h ;kết quả như sau: ô nhớ có địa chỉ là 30h lưu 34h.

**21. Lệnh giảm nội dung ô nhớ gián tiếp :**

– Cú pháp : **DEC @Ri**

– Mã lệnh :

| 0 | 0 | 0 | 1 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
– Chức năng: Giảm nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri xuống 1.

**22. Lệnh nhân thanh ghi A với thanh ghi B :**

– Cú pháp : **MUL AB**
– Mã lệnh :

| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 4 chu kỳ máy
– Chức năng: Nội dung của thanh ghi A nhân với nội dung của thanh ghi B, kết qủa là một dữ liệu 16 bit, 8 bit thấp lưu trữ trong thanh ghi A, 8 bit cao lưu trữ trong thanh ghi B.

Ví dụ : Giả sử thanh ghi A có nội dung là 50h,thanh ghi B có nội dung 0A0h , lệnh:
MUL AB ;Kết qủa như sau: 50h*A0h:= 3200h thì (A) = 00 và (B) = 32h.

**23. Lệnh chia thanh ghi A cho thanh ghi B :**

– Cú pháp : **DIV AB**
– Mã lệnh :

| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 4 chu kỳ máy
– Chức năng: Nội dung của thanh ghi A chia cho nội dung của thanh ghi B, kết qủa của phép chia lưu trữ trong thanh ghi A, số dư lưu trữ trong thanh ghi B. Lệnh ảnh hưởng đến thanh ghi trạng thái: Bit C và bit OV bị xóa về 0, nếu phép chia này mà dữ liệu trong thanh ghi B = 00h thì nội dung thanh ghi A không thay đổi, nội dung chứa trong thanh ghi B không xác định và bit OV = 1, bit Cy = 0.

**24. Lệnh điều chỉnh thập phân nội dung thanh ghi A :**

– Cú pháp : **DA A**
– Mã lệnh :

| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 4 chu kỳ máy
– Chức năng: Nếu 4 bit thấp A3A2A1A0>9 hoặc bit AC = 1thì A3A2A1A0 + 6, kết qủa lưu trở lại trong A. Nếu 4 bit cao A7A6A5A4>9 hoặc bit Cy = 1 thì A7A6A5A4 + 6, kết quả lưu trở lại thanh ghi A. Kết quả sau cùng trong thanh ghi A là số BCD.

**3.2.3 Nhóm lệnh logic**

**1. Lệnh and thanh ghi A với thanh ghi :**

– Cú pháp : **ANL A,Rn (and logic)**
– Mã lệnh :

| 0 | 1 | 0 | 1 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
– Chức năng: Nội dung thanh ghi A and với nội dung thanh ghi Rn, kết quả lưu trữ trong thanh ghi A.

Ví dụ :

```
MOV A, #10110011b
MOV R0, #11001011b
ANL a, r0 ;kết quả (A) = 10000011b
```

**2. 2. Lệnh and thanh ghi A với nội dung ô nhớ trực tiếp :**

– Cú pháp : **ANL A, direct**
– Mã lệnh :

| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A and với nội dung của ô nhớ có địa chỉ direct, kết quả chứa ở thanh ghi A.

**3. Lệnh and thanh ghi A với nội dung ô nhớ gián tiếp :**

– Cú pháp : **ANL A, @Ri**

– Mã lệnh :

| 0 | 1 | 0 | 1 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A and với ô nhớ có địa chỉ chứa trong thanh ghi Ri, kết quả lưu trữ trong thanh ghi A.

**4. Lệnh and thanh ghi A với dữ liệu tức thời 8 bit :**

– Cú pháp : **ANL A, #data**

– Mã lệnh :

| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung của thanh ghi A and với dữ liệu d0 đến d7 , kết quả lưu trữ trong thanh ghi A.

Ví dụ :

```
MOV A , #10110011b
ANL a, #00001111b ;kết quả (A) = 00000011b
```

**5. Lệnh and nội dung ô nhớ trực tiếp với nội dung thanh ghi A :**

– Cú pháp : **ANL direct, A**

– Mã lệnh :

| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung ô nhớ có địa chỉ direct and với nội dung của thanh ghi A, kết qủa lưu trữ vào ô nhớ.

Ví dụ :

```
MOV A ,#10110011b
MOV 10h,#11110000b
ANL 10h,A ;kết quả ô nhớ có địa chỉ 10h lưu 10110000b.
```

**6. Lệnh and nội dung ô nhớ trực tiếp với dữ liệu tức thời 8 bit :**

– Cú pháp : **ANL direct, #data**

– Mã lệnh :

| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Nội dung của ô nhớ có địa chỉ direct and với 8 bit dữ liệu 8 bit, kết quả lưu trữ vào ô nhớ.

**7. Lệnh or thanh ghi A với thanh ghi :**

– Cú pháp : **ORL A, Rn**

– Mã lệnh :

| 0 | 1 | 0 | 0 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A or với nội dung thanh ghi Rn, kết quả lưu trữ trong thanh ghi A.

Ví dụ :

```
MOV A ,#10110011b
MOV R0,#11001011b
ORL A, R0 ;kết quả (A) = 11111011b.
```

**8. Lệnh or thanh ghi A với nội dung ô nhớ trực tiếp :**

– Cú pháp : **ORL A, direct**

– Mã lệnh :

| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A or với nội dung của ô nhớ có địa chỉ direct, kết qủa chứa ở thanh ghi A.

**9. Lệnh or thanh ghi A với nội dung ô nhớ gián tiếp :**

– Cú pháp : **ORL A, @Ri**

– Mã lệnh :

| 0 | 1 | 0 | 0 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A or với ô nhớ có địa chỉ chứa trong thanh ghi Ri, kết quả lưu trữ trong thanh ghi A.

**10. Lệnh or thanh ghi A với dữ liệu tức thời 8 bit :**

– Cú pháp : **ORL A, #data**

– Mã lệnh :

| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung của thanh ghi A or với dữ liệu 8 bit data (từ d0 đến d7), kết quả lưu trữ trong thanh ghi A.

**11. Lệnh or nội dung ô nhớ trực tiếp với nội dung thanh ghi A :**

– Cú pháp : **ORL direct, A**

– Mã lệnh :

| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung ô nhớ có địa chỉ direct or với nội dung của thanh ghi A, kết qủa lưu trữ trong ô nhớ có địa chỉ direct.

**12. Lệnh or nội dung ô nhớ trực tiếp với dữ liệu tức thời 8 bit :**

– Cú pháp : **ORL direct, #data**

– Mã lệnh :

| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Nội dung của ô nhớ có địa chỉ direct or với dữ liệu 8 bit (từ d0 đến d7 ) ở byte thứ 3, kết quả lưu trữ trong ô nhớ.

**13. Lệnh ex-or thanh ghi A với thanh ghi :**

– Cú pháp : **XRL A, Rn**

– Mã lệnh :

| 0 | 1 | 1 | 0 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A ex-or với nội dung thanh ghi Rn, kết quả lưu trữ trong thanh ghi A.

**14. Lệnh ex-or thanh ghi A với nội dung ô nhớ trực tiếp :**

– Cú pháp : **XRL A, direct**

– Mã lệnh :

| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A ex-or với nội dung của ô nhớ có địa chỉ direct, kết qủa chứa ở thanh ghi A.

**15. Lệnh ex-or thanh ghi A với nội dung ô nhớ gián tiếp :**

– Cú pháp : **XRL A, @Ri**

– Mã lệnh :

| 0 | 1 | 1 | 0 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A ex-or với ô nhớ có địa chỉ chứa trong thanh ghi Ri, kết quả lưu trữ trong thanh ghi A.

**16. Lệnh ex-or thanh ghi A với dữ liệu tức thời 8 bit :**

– Cú pháp : **XRL A, #data**

– Mã lệnh :

| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung của thanh ghi A ex-or với dữ liệu datat, kết qủa lưu trữ trong thanh ghi A.

**17. Lệnh ex-or nội dung ô nhớ trực tiếp với nội dung thanh ghi A :**

– Cú pháp : **XRL direct, A**

– Mã lệnh :

| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung ô nhớ có địa chỉ direct ex-or với nội dung của thanh ghi A, kết qủa lưu trữ vào ô nhớ.

**18. Lệnh ex-or nội dung ô nhớ trực tiếp với dữ liệu tức thời 8 bit :**

– Cú pháp : **XRL direct, #data**

– Mã lệnh :

| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Nội dung của ô nhớ có địa chỉ direct ex-or với 8 bit dữ liệu data 8 bit, kết quả lưu trữ vào ô nhớ.

**19. Lệnh xóa nội dung thanh ghi A :**

– Cú pháp : **CLR A (clear a)**

– Mã lệnh :

1 1 1 0 0 1 0 0

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A bằng zero.

**20. Lệnh bù nội dung thanh ghi A :**

– Cú pháp : **CPL A (complement A)**

– Mã lệnh :

| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A được lấy bù 1, kết quả chứa trong A.

Ví dụ :

```
MOV A ,#10110011b
CPL A ;kết quả (A) = 01001100b.
```

**21. Lệnh xoay trái nội dung thanh ghi A :**

– Cú pháp : **RL A (rotate left)**

– Mã lệnh :

| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A được xoay trái 1 bit.

Ví dụ :

```
MOV A,#1011 0011b ;
RL A ;lệnh thứ nhất
```

Gía trị ban đầu của C ta không cần quan tâm đến kết quả sau khi xoay thì (A) = 0110 0111b và cờ (C) = 1 là do bit A7 bằng 1 chuyển sang.

```
RL A ;lệnh thứ 2
```

Kết quả sau khi xoay thì (A) = 11001110b và cờ (C) = 0 là do bit A7 bằng 0 chuyển sang.

**22. Lệnh xoay trái nội dung thanh ghi A và bit carry :**

– Cú pháp : **RLC A**

– Mã lệnh :

| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung thanh ghi A và bit C được xoay trái 1 bit.

Ví dụ: Giả sử cho cờ C = 0 trước khi thực hiện lệnh

```
MOV A ,#10110011b
RLC A ;kết quả (A) = 01100110b và cờ (C) = 1
```

Ví dụ:

```
SETB C ;làm cờ C bằng 1
MOV A,#00000000b
RLC A ;kết quả (A) = 0000 0001b và cờ (C) = 0
```

```
SETB C ;làm cờ C bằng 1
RLC A ;kết quả (A) = 0000 0011b và cờ (C) = 0
....
SETB C ;làm cờ C bằng 1
RLC A ;kết quả (A) = 0111 1111b và cờ (C) = 0
SETB C ;làm cờ C bằng 1 (lần thứ 8)
RLC A ;kết quả (A) = 1111 1111b và cờ (C) = 0
SETB C ;làm cờ C bằng 1 (lần thứ 9)
RLC A ;kết quả (A) = 1111 1111b và cờ (C) = 1
;xxxx
Clr C ;làm cờ C bằng 0
RLC A ;kết quả (A) = 1111 1110b và cờ (C) = 1
```

**23. Lệnh xoay phải nội dung thanh ghi A :**

- Cú pháp : **RR A (rotate right)**
- Mã lệnh :

| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A được xoay phải 1 bit ngược với lệnh RL A.

**24. Lệnh xoay phải nội dung thanh ghi A và bit carry :**

- Cú pháp : **RRC A**
- Mã lệnh :

| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A và bit C được xoay phải 1 bit ngược với lệnh RLC A.

**25. Lệnh xoay thanh ghi A 4 bit :**

- Cú pháp : **SWAP A**
- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: hoán chuyển 4 bit thấp và 4 bit cao trong thanh ghi A.

Ví dụ :

```
MOV A,#3EH
SWAP A ;kết quả (A) = E3H
```

**3.2.4 . Nhóm lệnh xử lý bit :**

**1. Lệnh xóa bit carry :**

- Cú pháp : **CLR C**
- Mã lệnh :

| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Xóa bit C về 0.

**2. Lệnh xóa bit :**

- Cú pháp : **CLR bit**
- Mã lệnh :

| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Xóa bit có địa chỉ được xác định bởi byte thứ 2 về 0.

**3. Lệnh đặt bit carry :**

– Cú pháp : **SETB C**

– Mã lệnh :

| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Đặt bit C = 1.

**4. Lệnh đặt bit :**

– Cú pháp : **SETB bit**

– Mã lệnh :

| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Đặt bit có địa chỉ được xác định bởi byte thứ 2 lên 1.

**5. Lệnh bù bit carry :**

– Cú pháp : **CPL C**

– Mã lệnh :

| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Bù bit carry, nếu trước đó C = 1 thì C = 0, ngược lại C = 0 thì C = 1.

**6. Lệnh bù bit :**

– Cú pháp : **CPL bit**

– Mã lệnh :

| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Bù bit có địa chỉ xác định bởi byte thứ 2, nếu trước đó bit này = 0 thì kết quả bit này bằng 1 và ngược lại nếu trước đó bằng 1 thì nó sẽ bằng 0.

**7. Lệnh and bit carry với bit :**

– Cú pháp : **ANL C, bit**

– Mã lệnh :

| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Bit C and với bit có địa chỉ được xác định bởi byte thứ 2, kết quả chứa ở bit C.

**8. Lệnh and bit carry với bù bit :**

– Cú pháp : **ANL C, /bit**

– Mã lệnh :

| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Bit C and với bù bit có địa chỉ được xác định bởi byte thứ 2, kết quả chứa ở bit C.

**9. Lệnh or bit carry với bit :**

– Cú pháp : **ORL C, bit**

– Mã lệnh :

| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Bit C or với bit có địa chỉ được xác định bởi byte thứ 2, kết quả chứa ở bit C.

**10. Lệnh or bit carry với bù bit :**

– Cú pháp : **ORL C, /bit**

– Mã lệnh :

| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Bit C or với bù bit có địa chỉ được xác định bởi byte thứ 2, kết quả chứa ở bit C.

**11. Lệnh di chuyển bit vào bit carry :**

– Cú pháp : **MOV C, bit**

– Mã lệnh :

| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Bit có địa chỉ được xác định bởi byte thứ 2 được chuyển vào bit C.

**12. Lệnh di chuyển bit carry vào bit :**

– Cú pháp : **MOV bit, C**

– Mã lệnh :

| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Bit C được chuyển vào bit có địa chỉ được xác định bởi byte thứ 2.

### 3.2.5. Nhóm lệnh điều khiển

**1. Lệnh gọi chương trình con dùng địa chỉ tuyệt đối :**

– Cú pháp : **ACALL addr11**

– Mã lệnh :

| A10 | A9 | A8 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy.

– Chức năng: Khi lệnh này được thực hiện thì vi điều khiển sẽ thực hiện chương trình con tại địa chỉ addr11. Chương trình con không được cách lệnh gọi quá 2 kbyte. Addr11 của chương trình con có thể thay bằng nhản (tên của chương trình con).

– Chú ý: Trước khi nạp địa chỉ mới vào thanh ghi PC thì địa chỉ của lệnh kế trong chương trình chính được cất vào bộ nhớ ngăn xếp.

2. Lệnh gọi chương trình con dùng địa chỉ dài 16 bit :

– Cú pháp : **LCALL addr16**

– Mã lệnh :

| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Khi lệnh này được thực hiện thì vi điều khiển sẽ thực hiện chương trình con tại địa chỉ addr16. Lệnh này có thể gọi chương trình con ở đâu cũng được trong vùng 64kbyte. Addr16 của chương trình con có thể thay bằng nhản (tên của chương trình con).

– 16 bit địa chỉ A15 – A0 được nạp vào PC, vi điều khiển sẽ thực hiện chương trình con tại địa chỉ vừa nạp vào PC. Chú ý: Trước khi nạp địa chỉ vào thanh ghi PC thì địa chỉ của lệnh kế trong chương trình chính được cất vào bộ nhớ ngăn xếp.

**3. Lệnh trở về từ chương trình con :**

– Cú pháp : **RET**

– Mã lệnh :

| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Lệnh này sẽ kết thúc chương trình con, vi điều khiển sẽ trở lại chương trình chính để tiếp tục thực hiện chương trình.

– Chú ý: lệnh này sẽ lấy địa chỉ của lệnh kế đã lưu trong bộ nhớ ngăn xếp (khi thực hiện lệnh gọi) trả lại cho thanh ghi PC để tiếp tục thực hiện chương trình chính.

***Khi viết chương trình con thì phải luôn luôn kết thúc bằng lệnh ret.***

4. Lệnh trở về từ chương trình con phục vụ ngắt :

– Cú pháp : **RETI**

– Mã lệnh :

| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Lệnh này sẽ kết thúc chương trình phục vụ ngắt, vi điều khiển sẽ trở lại chương trình chính để tiếp tục thực hiện chương trình.

5. Lệnh nhảy dùng địa chỉ tuyệt đối :

– Cú pháp : **AJMP addr11**

– Mã lệnh :

| A10 | A9 | A8 | 0 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Ý nghĩa của lệnh: vi điều khiển sẽ nhảy đến đến địa chỉ addr11 để thực hiện chương trình tại đó. Addr11 có thể thay thế bằng nhản. Nhản hay địa chỉ nhảy đến không quá 2 kbyte.

– 11 bit địa chỉ A10 – A0 được nạp vào PC, các bit cao của PC không thay đổi, vi điều khiển sẽ nhảy đến thực hiện lệnh tại địa chỉ PC mới vừa nạp.

– Lệnh này khác với lệnh gọi chương trình con là không cất địa chỉ trở về. Nơi nhảy đến không quá 2 kbyte so với lệnh nhảy.

6. Lệnh nhảy dùng địa chỉ 16 bit :

– Cú pháp : **LJMP addr16**

– Mã lệnh :

| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 |

| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: vi điều khiển sẽ nhảy đến địa chỉ addr16 để thực hiện chương trình tại đó. Nơi nhảy đến tùy ý nằm trong vùng 64 kbyte.

7. Lệnh nhảy tương đối :

– Cú pháp : **SJMP rel**

– Mã lệnh :

| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: vi điều khiển sẽ nhảy đến lệnh có địa chỉ tương đối (rel) để thực hiện tiếp. Có thể thay thế rel bằng nhản.

– Lệnh này chỉ nhảy trong tầm vực 256 byte: có thể nhảy tới 128 byte và có thể nhảy lùi 128 byte. Khi tầm vực nhảy xa hơn ta nên dùng lệnh AJMP hay LJMP.

– Chú ý: rel [relative: tương đối]: các lệnh có xuất hiện "rel" đều liên quan đến lệnh nhảy: nơi nhảy đến được tính bằng cách lấy nội dung của PC cộng với số lượng byte của các lệnh nằm giữa lệnh nhảy và nơi nhảy đến. Chúng ta không cần quan tâm đến điều này vì chương trình biên dịch của máy tính sẽ tính giúp chúng ta.

8. Lệnh nhảy gián tiếp :

– Cú pháp : **JMP @A + DPTR**

– Mã lệnh :

| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: lệnh sẽ nhảy đến nơi có địa chỉ bằng nội dung của A cộng với dptr để

tiếp tục thực hiện chương trình tại đó.

9. Lệnh nhảy nếu cờ Z =1 (nội dung thanh ghi A bằng 0) :

– Cú pháp : **JZ rel (jump zero)**

– Mã lệnh :

–

| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: nếu bit Z = 1 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel (thỏa điều kiện), nếu Z = 0 thì vi điều khiển sẽ tiếp tục thực hiện lệnh kế (không thỏa điều kiện).

10. Lệnh nhảy nếu cờ Z = 0 (nội dung thanh ghi A khác 0):

– Cú pháp : **JNZ rel**

– Mã lệnh :

| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: nếu Z = 0 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel.

11. Lệnh nhảy nếu bit carry = 1 :

– Cú pháp : **JC rel**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: nếu bit carry C = 1 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel.

12. Lệnh nhảy nếu bit carry = 0 :

– Cú pháp : **JNC rel**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: nếu bit carry C = 0 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel.

13. Lệnh nhảy nếu bit = 1 :

– Cú pháp : **JB bit, rel**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: nếu nội dung của bit có địa chỉ bit [được xác định bởi byte thứ 2] bằng 1 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel.

14. Lệnh nhảy nếu bit = 0 :

– Cú pháp : **JNB bit, rel**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: nếu nội dung của bit có địa chỉ bit [được xác định bởi byte thứ 2] bằng 0 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ bằng rel.

15. Lệnh nhảy nếu bit = 1 và xóa bit :

– Cú pháp : **JBC bit, rel**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: nếu bit được xác định bởi byte thứ 2 bằng 1 thì bit này được xóa về 0 và vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel.

16. Lệnh so sánh ô nhớ trực tiếp với nội dung thanh ghi A :

– Cú pháp : **CJNE A, direct, rel (compare jump if not equal)**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
|---|---|---|---|---|---|---|---|
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
– Chức năng: lệnh này ảnh hưởng đến cờ C và thực hiện việc nhảy như sau:
– Nếu nội dung của A ≥ nội dung của ô nhớ có địa chỉ direct thì bit C = 0.
– Nếu nội dung của A < nội dung của ô nhớ có địa chỉ direct thì bit C = 1.
– Nếu nội dung của A khác nội dung của ô nhớ có địa chỉ direct thì lệnh sẽ nhảy đến và thực hiện lệnh tại địa chỉ rel.
– Nếu nội dung của A bằng nội dung của ô nhớ có địa chỉ direct thì không nhảy và làm lệnh kế.

17. Lệnh so sánh dữ liệu tức thời với nội dung thanh ghi A :
– Cú pháp : **CJNE A, #data, rel**
– Mã lệnh :

| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
– Chức năng: lệnh này ảnh hưởng đến cờ C và thực hiện việc nhảy như sau:
– Nếu nội dung của A ≥ data 8 bit thì bit C = 0.
– Nếu nội dung của A < data 8 bit thì bit C = 1.
– Nếu nội dung của A khác data 8 bit thì lệnh sẽ nhảy đến thực hiện lệnh tại địa chỉ rel.
– Nếu nội dung của A bằng data 8 bit thì không nhảy và làm lệnh kế.

18. Lệnh so sánh dữ liệu tức thời với nội dung thanh ghi :
– Cú pháp : **CJNE Rn, #data, rel**
– Mã lệnh :

| 1 | 0 | 1 | 1 | 1 | n2 | n1 | n0 |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
– Chức năng: lệnh này ảnh hưởng đến cờ C và thực hiện việc nhảy như sau:
– Nếu nội dung của thanh ghi Rn ≥ data 8 bit thì bit C = 0.
– Nếu nội dung của thanh ghi Rn < data 8 bit thì bit C = 1.
– Nếu nội dung của thanh ghi Rn khác data 8 bit thì lệnh sẽ nhảy đến thực hiện lệnh tại địa chỉ rel.
– Nếu nội dung của thanh ghi Rn bằng data 8 bit thì không nhảy và làm lệnh kế.

19. Lệnh so sánh dữ liệu tức thời với dữ liệu gián tiếp :
– Cú pháp : **CJNE @Ri, #data, rel**
– Mã lệnh :

| 1 | 0 | 1 | 1 | 0 | 1 | 1 | i |
|---|---|---|---|---|---|---|---|
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
– Chức năng: lệnh này ảnh hưởng đến cờ C và thực hiện việc nhảy như sau:
– Nếu nội dung của ô nhớ có địa chỉ lưu trong thanh ghi Ri ≥ data 8 bit thì bit C

= 0.

– Nếu nội dung của ô nhớ có địa chỉ lưu trong thanh ghi Ri < data 8 bit thì bit C =1.

– Nếu nội dung của ô nhớ có địa chỉ lưu trong thanh ghi Ri khác data 8 bit thì lệnh sẽ nhảy đến thực hiện lệnh tại địa chỉ rel.

– Nếu nội dung của ô nhớ có địa chỉ lưu trong thanh ghi Ri bằng data 8 bit thì không nhảy và làm lệnh kế.

20. Lệnh giảm thanh ghi và nhảy :

– Cú pháp : **DJNZ Rn, rel (decrement and jump if not zero)**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | n2 | n1 | n0 |
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Nội dung của thanh ghi Rn giảm đi 1 và nếu kết quả trong thanh ghi Rn sau khi giảm khác 0 thì vi điều khiển sẽ thực hiện chương trình tại địa chỉ rel, nếu kết quả bằng 0 thì vi điều khiển sẽ tiếp tục thực hiện lệnh kế.

21. Lệnh giảm ô nhớ trực tiếp và nhảy :

– Cú pháp : **DJNZ direct, rel**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
| r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 |

– Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

– Chức năng: Nếu nội dung của ô nhớ có địa chỉ direct giảm đi 1 và nếu kết quả sau khi giảm khác 0 thì vi điều khiển sẽ thực hiện chương trình tại địa chỉ rel, ngược lại nếu kết quả bằng 0 thì vi điều khiển sẽ tiếp tục thực hiện lệnh kế.

22. Lệnh Nop :

– Cú pháp : **NOP**

– Mã lệnh :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

– Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

– Chức năng: Nội dung của PC tăng lên 1 và tiếp tục thực hiện lệnh tiếp theo.

**3.3** Khuôn dạng của chương trình hợp ngữ:

Các chương trình hợp ngữ chứa:

– Các lệnh(instruction) của bộ vi xử lý, bộ vi điều khiển

– Các chỉ dẫn (directive) của trình dịch hợp ngữ

– Các điều khiển (control) của trình dịch hợp ngữ.

– Các chú thích (comment).

Khuôn dạng tổng quát của mỗi dòng lệnh như sau:

**[Nhãn:] mã gọi nhớ [ Toán hạng][,Toán hạng][...][;chú thích]**

**Nhãn** : biểu thị địa chỉ của lệnh hoặc dữ liệu theo sau

– Nhãn là một loại ký hiệu và được nhận dạng bằng dấu : (kết thúc nhãn)

– Một ký hiệu hoặc nhãn phải bắt đầu bằng 1 ký tự chữ hoặc dấu (?) hoặc dấu nối dưới ( _ ) và tiếp theo phải là các ký tự chữ, các số, các dấu (?) hoặc ( _ ).

– Nhãn có thể dài 31 ký tự ở dạng chữ thường hoặc chữ in.

– Nhãn không thể trùng với các từ khóa ( các mã gợi nhớ, các chỉ dẫn, các toán tử hoặc các ký hiệu tiền định nghĩa).

**Mã gợi nhớ:** của lệnh hoặc chỉ dẫn của trình dịch hợp theo sau nhãn.

– Các thí dụ về mã gợi nhớ của lệnh là ADD, MOV, DIV, INC…

**Toán hạng:** Theo sau mã gợi nhớ. Trường toán hạng chứa địa chỉ hoặc dữ liệu mà lệnh sẽ sử dụng.

**Chú thích:** Các ghi chú để làm rõ chương trình được đặt trong trường chú thích ở cuối dòng lệnh.

– Các chú thích cần phải bắt đầu bằng dấu chấm phẩy(;). Các chú thích có thể chiếm nhiều dòng riêng và cũng phải bắt đầu bằng dấu chấm phẩy (;).

**3.3.1** Các ký hiệu đặc biệt:

Được dùng cho các kiểu định địa chỉ thanh ghi. Các ký hiệu này bao gồm A, R0 đến R7, DPTR, PC, C và AB. Dấu $ cũng là 1 ký hiệu đặc biệt được dùng để tham chiếu đến gía trị hiện hành của bộ đếm vị trí.

# Chương 4: Bộ định thời Timer

## 4.1 GIỚI THIỆU:

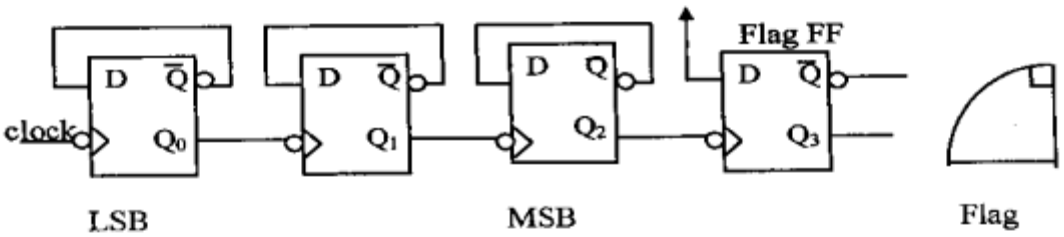
Một bộ định thời là một chuỗi các flipflop với mỗi flipflop là một mạch chia 2, chuỗi này nhận tín hiệu ngõ vào làm nguồn xung clock. Xung clock đặt vào flippop thứ nhất, flippop này chia đôi tần số xung clock. Ngõ ra của flippop thứ nhất trở thành nguồn xung clock cho flippop thứ hai, nguồn xung clock này cũng được chia cho hai... Vì mỗi một tầng kế tiếp nhau đều chia cho hai nên một bộ định thời có n tầng sẽ chia tần số xung clock ở ngõ vào của bộ này cho $2^n$.

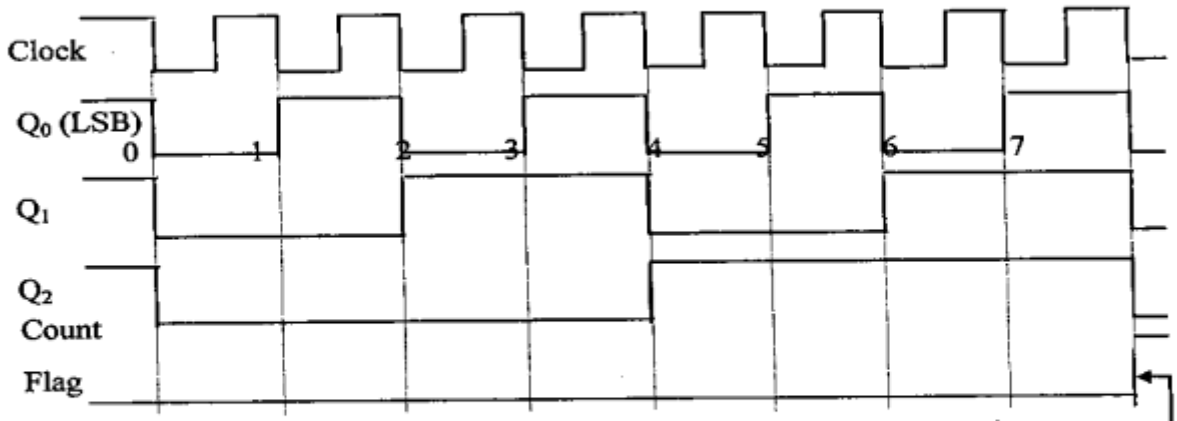
Bộ định thời dù đếm thời gian hay đếm sự kiện đều đếm tăng. Giá trị bắt đầu đếm được xác định bởi phần mềm. Khi bộ định thời đếm hết thì chương trình sẽ bật tắc cờ tràn, dấu hiệu cho phép thực hiện chương trình tiếp theo.

Ví dụ Timer 16 bit có thể đếm đến từ FFFFH sang 0000H.

Hoạt động của Timer đơn giản 3 bit được minh họa như sau:

**Timer Flip Flops(3)**




Trong hình trên mỗi tầng là một FF loại D phủ định tác động cạnh xuống được hoạt động ở chế độ chia cho 2 (ngõ ra Q\ được nối vào D). flad FF là một bộ chốt đơn giản loại D được set bởi tầng cuối cùng trong Timer. Trong biểu đồ thời gian, tầng đầu đổi trạng thái ở ½ tần số clock, tầng thứ hai đổi trạng thái ở tần số ¼ tần số clock . . . Số đếm

được biết ở dạng thập phân và được kiểm tra lại dễ dàng bởi việc kiểm tra các tầng của 3 FF. Ví dụ số đếm "4" xuất hiện khi Q2=1, Q1=0, Q0=0 ($4_{10}=100_2$).

89C51 có 2 timer 16bit, mỗi timer có 4 chế độ làm việc

- Timer được sử dụng để:
  - Định khoảng thời gian
  - Đếm sự kiện
  - Phát tốc độ baud cho port nối tiếp
- Timer được truy cập khi sử dụng các thanh ghi chức năng đặc biệt: TCON, TMOD, TH0, TL0, TH1, TL1

### 4.2 Các thanh ghi dùng cho Timer

#### 4.2.1 Thanh ghi chế độ định thời TMOD (Timer Mode Register)

Thanh ghi TMOD là thanh ghi 8 bít gồm có 4 bít thấp được thiết lập dành cho bộ Timer 0 và 4 bít cao dành cho Timer 1. Trong đó hai bít thấp của chúng dùng để thiết lập chế độ của bộ định thời, còn 2 bít cao dùng để xác định phép toán.

| Bit | Name | Timer | Description |
|---|---|---|---|
| 7 | GATE | 1 | Khi GATE = 1, Timer chỉ làm việc khi INT1=1 |
| 6 | C/T | 1 | Bit cho đếm sự kiện hay ghi giờ |
| | | | C/T = 1 : Đếm sự kiện |
| | | | C/T = 0 : Ghi giờ đều đặn |
| 5 | M1 | 1 | Bit chọn mode của Timer 1 |
| 4 | M0 | 1 | Bit chọn mode của Timer 1 |
| 3 | GATE | 0 | Bit cổng của Timer 0 |
| 2 | C/T | 0 | Bit chọn Counter/Timer của Timer 0 |
| 1 | M1 | 0 | Bit chọn mode của Timer 0 |
| 0 | M0 | 0 | Bit chọn mode của Timer 0 |

Cấu trúc thanh ghi TMOD:

| (MSB) | | | | | | | (MSB) |
|---|---|---|---|---|---|---|---|
| GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 |
| Timer1 | | | | Timer0 | | | |

Gate: bit mở cổng cho timer

- Khi Gate=1: Timer/Counterx chỉ hoạt động khi chân INTx ở mức 1(phần cứng) và bit TRx =1 (trong thanh ghi TCON)
- Khi Gate=0: Timerx hoạt động khi bit TRx=1

Ghi chú: X = 0,1

- TRx : TR0 hoặc TR1

– Timerx: Timer 0 hoặc Timer 1
C/T : bit chọn Timer hay Counter
– C/T=1 : chọn Counter (đếm sự kiện)
– C/T=0 : chọn Timer (định khoảng thời gian)
M1,M0 : các bit chọn chế độ

| M1 | M0 | MODE | DESCRIPTION |
|---|---|---|---|
| 0 | 0 | 0 | Chế độ định thời 13 bit |
| 0 | 1 | 1 | Chế độ định thời 16 bit |
| 1 | 0 | 2 | Chế độ định thời 8 bit tự động nạp lại |
| 1 | 1 | 3 | Chế độ định thời chia xẻ |

**Ví dụ:** Cho biết chế độ hoạt động của các Timer 0 và Timer 1.
a) MOV TMOD, #01H b) MOV TMOD, #20H c) MOV TMDO, #12H
**Lời giải:** Chúng ta chuyển đổi giá trị từ số Hex sang nhị phân và đối chiếu hình 93 ta có:
a) TMOD = 0000 0001, chế độ 1 của bộ định thời Timer 0 được chọn.
b) TMOD = 0010 0000, chế độ 1 của bộ định thời Timer 1 được chọn.
c) TMOD = 0001 0010, chế độ 1 của bộ định thời Timer 0 và chế độ 1 của Timer 1 được chọn.
Ví dụ: Cho biết giá trị cần nạp cho thanh ghi TMOD để
Timer 0 là bộ định thời gian 16 bit, được điều khiển bằng phần mềm (bit TR0).
Timer 1 là bộ đếm xung 13 bit, được điều khiển bằng phần cứng (chân INT1\)
Lời giải:
TMOD = 1100 0001B = C1H

### 4.2.2 Thanh ghi điều khiển định thời TCON (Timer Control Register)

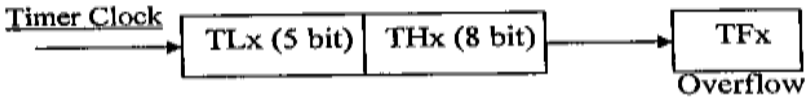
Thanh ghi TCON chứa các bit trạng thái và điều khiển cho Timer 0 và Timer 1.
Được địa chỉ hóa từng bit

| Bit | Symbol | Bit Address | Description |
|---|---|---|---|
| TCON.7 | TF1 | 8FH | Cờ tràn của Timer 1 (TF1=1: timer 1 bị tràn) |
| TCON.6 | TR1 | 8EH | Bit điều khiển hoạt động của Timer 1 (TR1=0:Timer 1 dừng, TR1=1:Timer 1 chạy |
| TCON.5 | TF0 | 8DH | Cờ tràn của Timer 0(TF0=1: timer 1 bị tràn) |
| TCON.4 | TR0 | 8CH | Bit điều khiển hoạt động Timer 0 (TR0=0:Timer 0 dừng, TR0=1:Timer 0 chạy) |
| TCON.3 | IE1 | 8BH | Cờ ngắt ngoài 1(IE1=1: khi tín hiệu ngắt xuất |

| | | | |
|---|---|---|---|
| | | | hiện tại chân INT1\) |
| TCON.2 | IT1 | 8AH | Bit điều khiển chọn kiểu tác động của tín hiệu ngắt ngoài 1<br>IT1 = 0: Ngắt ngoài 0 kích khởi mức thấp<br>IT1 = 1: Ngắt ngoài 0 kích khởi cạnh âm |
| TCON.1 | IE0 | 89H | Cờ ngắt ngoài 0(IE0=1: khi tín hiệu ngắt xuất hiện tại chân INT0\) |
| TCON | IT0 | 88H | Bit điều khiển chọn kiểu tác động của tín hiệu ngắt ngoài 0.<br>IT0 = 0: Ngắt ngoài 0 kích khởi mức thấp<br>IT0 = 1: Ngắt ngoài 0 kích khởi cạnh âm |

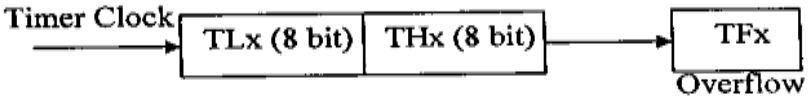
**4.2.3 Các chế độ hoạt động của TIMER**

**4.2.3.1 Chế độ 0:**



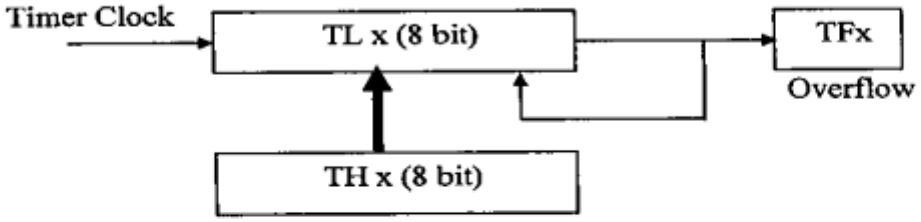

- Chế độ định thời 13bit.
- Sử dụng 8 bit của thanh ghi THx và 5bit thấp của thanh ghi TLx
- Giá trị đếm được là từ 0000→1FFFH nghĩa là từ 0 đến $2^{13}$ -1 = 8191. Thời gian định thời: từ 1.$T_{Timer}$ đến $2^{13}$ nghiã là từ 1 đến 8192$T_{timer}$.
- Thanh ghi THx và TLx chứa giá trị của bộ định thời.
- Khi có xung clock, bộ định thời bắt đầu đếm lên từ giá trị chứa trong THx/TLx.
- Xảy ra tràn (TFx = 1) khi số đếm chuyển từ 1FFFH sang 0000H và việc đếm sẽ tiếp tục đếm lên từ giá trị 0000H.

**4.2.3.2 Chế độ 1:**



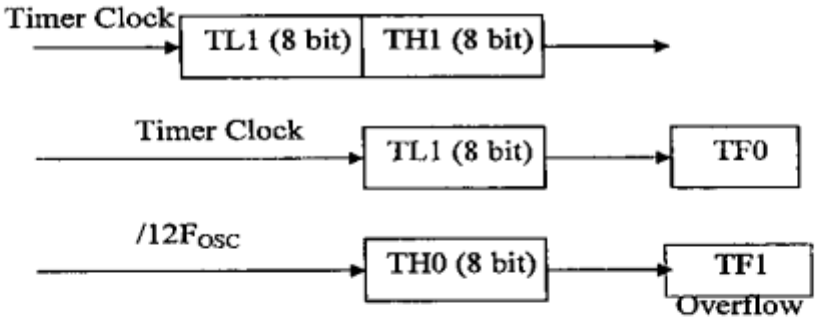

- Chế độ định thời 16 bit
- Trong chế độ này sử dụng cả 2 thanh ghi THx và TLx để tạo ra bộ định thời.
- Giá trị đếm được là từ 0000→FFFFH nghĩa là từ 0 đến $2^{16}$ -1 = 65535. Thời gian định thời: từ 1.$T_{Timer}$ đến $2^{16}$ nghiã là từ 1 đến 65536$T_{timer}$.
- Xảy ra tràn (TFx = 1) khi số đếm chuyển từ FFFFH sang 0000H và việc đếm sẽ tiếp tục đếm lên từ giá trị 0000H.

**4.2.3.3 Chế độ 2:**




- Chế độ định thời 8 bit tự động nạp lại.
- Sử dụng thanh ghi TLx để tạo ra bộ định thời.
- Giá trị đếm được là từ 00→FFH nghĩa là từ 0 đến $2^8$ -1 = 255. Thời gian định thời: từ $1.T_{Timer}$ đến $2^8$ nghiã là từ 1 đến $256T_{timer}$.
- Thanh ghi TLx chứa giá trị của bộ định thời và thanh ghi THx chứa giá trị sẽ được dùng để nạp lại cho bộ định thời.
- Khi có xung clock, bộ định thời bắt đầu đếm lên từ giá trị chứa trong TLx (THx không thay đổi giá trị)
- Xảy ra tràn (TFx = 1) khi số đếm chuyển từ FFH sang 00H, đồng thời giá trị trong THx sẽ được nạp vào TLx và việc đếm sẽ tiếp tục đếm lên từ giá trị chứa trong thanh ghi TLx (giá trị này bằng với giá trị của THx)

**4.2.3.4 Chế độ 3:**




- Chế độ định thời chia xẻ
- Bộ định thời 0 được chia ra:

Bộ định thời 8 bit thứ 1:

- Sử dụng thanh ghi TL0 để tạo ra bộ định thời.
- Giá trị đếm được là từ 00→FFH nghĩa là từ 0 đến $2^8$ -1 = 255. Thời gian định thời: từ $1.T_{Timer}$ đến $2^8$ nghiã là từ 1 đến $256T_{timer}$.
- Thanh ghi TL0 chứa giá trị của bộ định thời
- Khi có xung clock, bộ định thời bắt đầu đếm lên từ giá trị chứa trong TL0
- Xảy ra tràn (TF0 = 1) khi số đếm chuyển từ FFH sang 00H, và việc đếm sẽ tiếp tục đếm lên từ giá 00H.

Bộ định thời 8 bit thứ 2:

- Sử dụng thanh ghi TH0 để tạo ra bộ định thời.
- Giá trị đếm được là từ 00→FFH nghĩa là từ 0 đến $2^8$ -1 = 255. Thời gian định thời: từ $1.T_{Timer}$ đến $2^8$ nghiã là từ 1 đến $256T_{timer}$.

- Thanh ghi TH0 chứa giá trị của bộ định thời
- Khi có xung clock, bộ định thời bắt đầu đếm lên từ giá trị chứa trong TH0
- Xảy ra tràn (TF1 = 1) khi số đếm chuyển từ FFH sang 00H, và việc đếm sẽ tiếp tục đếm lên từ giá 00H

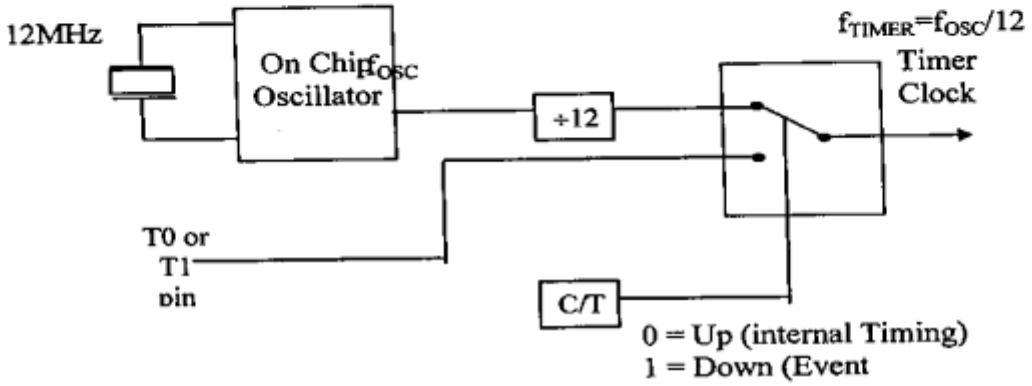
Bộ định thời 8 bit thứ 1:

- Là bộ định thời 16 bit.
- Không hoạt động ở chế độ 3 nhưng có thể hoạt động các chế độ khác (chế độ 0, 1, 2)
- Không có cờ báo tràn như các bộ định thời khác.

**4.2.4 Nguồn tạo xung nhịp cho bộ định thời**

Có 2 nguồn xung nhịp có thể được sử dụng và nó được chọn bằng bit C/T trong thanh ghi TMOD khi khởi động timer.

**4.2.4.1 Xung nhịp định khoảng thời gian:**

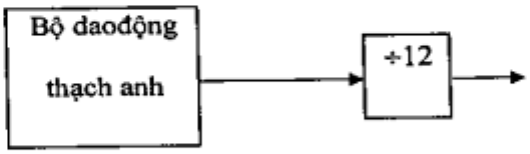



- Khi C/T =0 thì timer được dùng để định khoảng thời gian.
- Nguồn xung nhịp được lấy từ bộ dao động trên chip đưa qua một bộ chia 12 để giảm đến tần số thích hợp cho phần lớn các ứng dụng.
- $f_{timer} = f_{osc}/12$.
- Thời gian định thời là khoảng thời gian được tính từ lúc bộ định thời bắt đầu đếm lên ( từ giá trị chứa trong các thanh ghi THx/TLx) cho đến lúc bộ định thời bắt đầu tràn( thời gian này phụ thuộc vào giá trị ban đầu được nạp cho các thanh ghi THx và TLx)

**Ví dụ :**

Hãy tìm tần số đồng bộ và chu kỳ của bộ định thời cho các hệ dựa trên 8051 với các tần số thạch anh sau:

a) 12MHz
b) 16MHz
c) 11,0592MHz




**Lời giải:**

a) $f_{Timer} = \frac{1}{12} \times 12MHz = 1MHz$ và $T_{Timer} = \frac{1}{1/1MHz} = 1\mu s$

b) $f_{Timer} = \frac{1}{12} \times 16MHz = 1,111Mz$ và $T_{Timer} = \frac{1}{1,333MHz} = 0,75\mu s$

c) $f_{Timer} = \frac{1}{12} \times 11,0592MHz = 921,6kHz$ và $T_{Timer} = \frac{1}{0,9216MHz} = 1,085\mu s$

**4.2.4.2 Xung nhịp đếm sự kiện:**

- Khi C/T =1 thì bộ định thời dùng để đếm sự kiện
- Xung nhịp được lấy từ xung kích thích bên ngoài thông qua ngõ T0 hoặc T1.
- Tần số kích thích tối đa cho phép tại chân $T_0$ và $T_1$

$$f_{T0,T1(MAX)} = \frac{f_{TIMER}}{2}$$

- Số lượng sự kiện (số xung) mà bộ định thời đếm được sẽ được chứa trong các thanh ghi THx/TLx, giá trị trong các thanh ghi này sẽ tăng theo mỗi xung kích bên ngoài tại T0 và T1.
- Một kích thích được gọi là 1 xung khi xảy ra sự chuyển trạng thái từ 1 xuống 0 ở chân T0 hoặc T1.

Ví dụ: tính tần số kích thích tối đa cho phép tại chân T0 và T1 trên chip 8051 với tần số thạch anh như sau: 11,0592MHz, 12MHz và 16MHz.

$$f_{T0,T1(MAX)} = \frac{f_{TIMER}}{2} = \frac{921,6KHz}{2} = 460,8KHz$$

$$f_{T0,T1(MAX)} = \frac{f_{TIMER}}{2} = \frac{921,6KHz}{2} = 460,8KHz$$

$$f_{T0,T1(MAX)} = \frac{f_{TIMER}}{2} = \frac{1,333MHz}{2} = 666,5KHz$$

**4.3 Khởi động, dừng và điều khiển Timer**

Bit TRx trong thanh ghi TCON được điều khiển bởi phần mềm để bắt đầu

hoặc kết thúc các Timer.

Cách 1:

Để bắt đầu các Timer ta set bit TRx và để kết thúc Timer ta Clear TRx.

Ví dụ Timer 0 được bắt đầu bởi lệnh SETB TR0 và được kết thúc bởi lệnh

CLR TR0 (bit Gate= 0).

Bit TRx bị xóa sau sự reset hệ thống, do đó các Timer bị cấm bằng sự mặc

định.

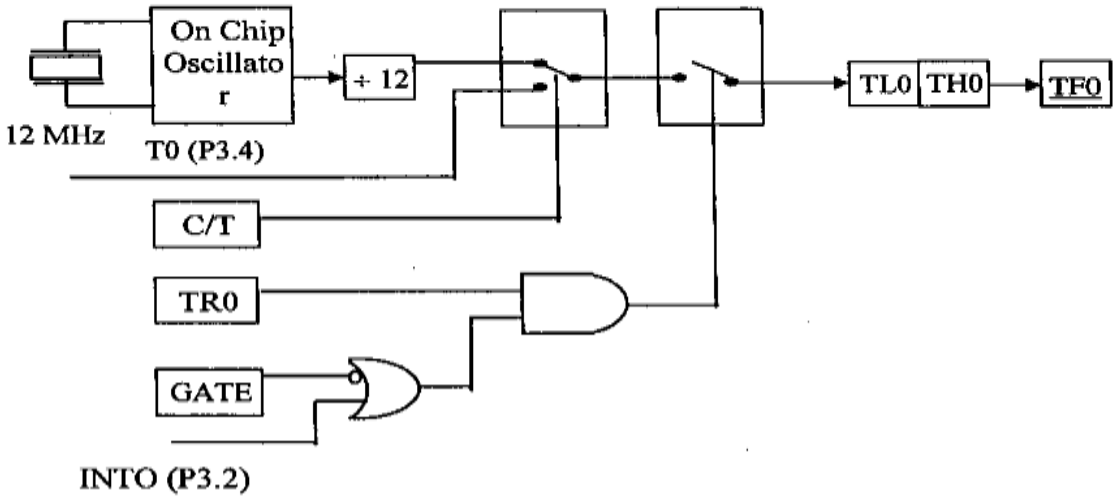
Cách 2:

Để điều khiển các Timer là dùng bit GATE trong thanh ghi TMOD và ngõ nhập bên ngoài INTx. Điều này được dùng để đo các độ rộng xung.

Giả sử xung đưa vào chân INT0 ta khởi động Timer 0 cho mode 1 là mode Timer 16 bit với TL0/TH0 = 0000H, GATE = 1, TR0 = 1.

Như vậy khi INT0 = 1 thì Timer "được mở cổng" và ghi giờ với tốc độ của tần số 1MHz.

Khi INT0 xuống thấp thì Timer "đóng cổng" và khoảng thời gian của xung tính bằng μs là sự đếm được trong thanh ghi TL0/TH0.




Hoạt động ở chế độ 1 của Timer 1

**4.3.1 Sự khởi động và truy xuất các thanh ghi timer:**

- Các Timer được khởi động 1 lần ở đầu chương trình để đặt chế độ hoạt động cho chúng. Sau đó trong chương trình các Timer được bắt đầu, được xóa, các thanh ghi Timer được đọc và cập nhật ... theo yêu cầu của từng ứng dụng cụ thể.
- TMOD là thanh ghi đầu tiên được khởi gán, bởi vì đặt chế độ hoạt động cho các Timer.

**Ví dụ** khởi động cho Timer 1 hoạt động ở chế độ 1 (chế độ định thời 16bit) và được ghi giờ bằng dao động trên Chip

Ta dùng lệnh : MOV TMOD, # 00001000B.

Trong lệnh này M1 = 0, M0 = 1 để vào mode 1 và C/T = 0, GATE = 0 để cho phép ghi giờ bên trong đồng thời xóa các bit mode của Timer 0. Sau lệnh trên Timer vẫn chưa đếm giờ, nó chỉ bắt đầu đếm giờ khi set bit điều khiển chạy TR1 của nó.

- Nếu ta không khởi gán giá trị đầu cho các thanh ghi TLx/THx thì Timer sẽ bắt đầu đếm từ 0000H lên và khi tràn từ FFFFH sang 0000H nó sẽ bắt đầu tràn TFx rồi tiếp tục đếm từ 0000H lên tiếp . . .
- Nếu ta khởi gán giá trị đầu cho TLx/THx, thì Timer sẽ bắt đầu đếm từ giá trị khởi gán đó lên nhưng khi tràn từ FFFFH sang 0000H lại đếm từ 0000H lên.

– Chú ý rằng cờ tràn TFx tự động được set bởi phần cứng sau mỗi sự tràn và sẽ được xóa bởi phần mềm. Chính vì vậy ta có thể lập trình chờ sau mỗi lần tràn ta sẽ xóa cờ TFx và quay vòng lặp khởi gán cho TLx/THx để Timer luôn luôn bắt đầu đếm từ giá trị khởi gán lên theo ý ta mong muốn.
– Đặc biệt những sự khởi gán nhỏ hơn 256 μs, ta sẽ gọi mode Timer tự động nạp 8 bit của mode 2. Sau khi khởi gán giá trị đầu vào THx, khi set bit TRx thì Timer sẽ bắt đầu đếm giá trị khởi gán và khi tràn từ FFH sang 00H trong TLx, cờ TFx tự động được set đồng thời giá trị khởi gán mà ta khởi gán cho Thx được nạp tự động vào TLx và Timer lại được đếm từ giá trị khởi gán này lên. Nói cách khác, sau mỗi tràn ta không cần khởi gán lại cho các thanh ghi Timer mà chúng vẫn đếm được lại từ giá trị ban đầu.

**4.3.2 Các khoảng thời gian định thời:**

❖ Khoảng thời gian định thời ngắn nhất: $1.T_{Timer}$
❖ Khoảng thời gian định thời dài nhất:
  ➢ $\approx 10. T_{Timer} \Rightarrow$ Dùng các lệnh
  ➢ $\leq 256.T_{Timer} \Rightarrow$ Dùng bộ định thời 8 bit tự động nạp lại.
  ➢ $\leq 65536. T_{Timer} \Rightarrow$ Dùng bộ định thời 16 bit.
  ➢ Không giới hạn $\Rightarrow$ Dùng bộ định thời 16 bit + các vòng lặp
  Với $T_{Timer} = 12/f_{OSC}$

**Ví dụ:**
MOV TH1,#(-255)→ định thời 255. $T_{TIMER}$
Chế độ 13 bit: gía trị phải nằm trong khoảng từ -8191 đến -1 ( tương ứng từ 8191. $T_{TIMER}$ đến $1.T_{TIMER}$)
Ví dụ: MOV TL1, #LOW(-8000) → định thời 8000. $T_{TIMER}$
MOV TH1,#HIGH(-8000)
Chế độ 16bit: gía trị phải nằm trong khoảng từ -65535 đến -1 ( tương ứng từ 65535 $T_{TIMER}$ đến $1.T_{TIMER}$)
Ví dụ: MOV TL1, #LOW(-10000) → định thời 8000. $T_{TIMER}$
MOV TH1,#HIGH(-10000)

**4.3.3 Các bước cơ bản để khởi động**

– Chọn chế độ hoạt động cho timer, cho bit Gate=0, C/T=0
  MOV TMOD, # ....
– Nạp các giá trị thích hợp cho các thanh ghi THx và TLx
  MOV THx, # ...
  MOV TLx, # ...
– Cho timer chạy : SETB TRx
– Chờ cờ báo tràn: có 2 cách:
  – JNB TFx, $
  – Wait: JNB TFx, Wait
– Xóa cờ báo tràn:
  – CLR TFx
– Dừng timer:
  – CLR TRx

**4.4 Bài tập áp dụng: viết chương trình điều khiển:**

**4.4.1 Phương pháp tính thời gian $t_{Delay}$ có độ chính xác tương đối:**

Phương pháp thực hiện là sử dụng lệnh vòng lặp DJNZ để tạo thời gian trễ $t_{Delay}$ như mong muốn. Phương pháp dùng lệnh , không dùng Timer thì việc định thời gian thường có 1 sai số nhất định. Vì để đơn giản trong việc tính toán ta bỏ qua không tính đến thời gian cần thiết để thực hiện từng lệnh trong chương trình, chỉ quan tâm đến thời gian thực hiện của lệnh DJNZ và số lần thực hiện của nó.
Tổng quát công thức tính thời gian trễ $t_{Delay}$ như sau:

$$t_{Delay} = [R_n] \times [R_m] \times \ldots \times [R_v] \times 2 \times \frac{12}{f_{OSC}}$$

$t_{Delay}$ (µs): thời gian trễ.
$f_{OSC}$(MHz): tần số thạch anh
$T_{Timer}$(µs): chu kỳ Timer ($T_{Timer} = 12/f_{OSC}$)
$[R_n]$, $[R_m]$,..., $[R_v]$: giá trị của các vòng lặp (số lần lặp lại lệnh DJNZ)
Lưu ý: giá trị của các vòng lặp phải thỏa điều kiện $0 \le R_n \le 255$.

```
      DELAY:
                  MOV R0,#10
      AAA: MOV R1,#20
      BBB:        MOV R2, #30
                  DJNZ R2, $          ; lệnh 2. TTimer
                  DJNZ R1,BBB
                  DJNZ R0, AAA
                  RET
```

$T_{Delay} = 10 \times 20 \times 30 \times 2. T_{Timer}$
**Ví dụ 1:** Tính toán giá trị cần nạp cho các thanh ghi vòng lặp biết
a. $t_{Delay} = 100\mu s$ , $f_{OSC} = 12MHz$
Ta chọn $[R_n] = 50$
$t_{Delay} = 50 \times 2 \times 12/12 = 100\mu s$
Chương trình:

```
      DELAY:
                  MOV R0, #50         ; lệnh 1. TTimer
                  DJNZ R2, $          ; lệnh 2. TTimer
                  RET                 ; lệnh 2. TTimer
```

b. $t_{Delay} = 10ms$ , $f_{OSC} = 12MHz$
Ta chọn $[R_n] = 20$, $[R_m] = 250$
$t_{Delay} = 20 \times 250 \times 2x\ 12/12 = 10000\mu s$

Chương trình:

```
      DELAY:
                  MOV R0, #20         ; lệnh 1. TTimer
      AAA:        MOV R1, #250        ; lệnh 2. TTimer
                  DJNZ R1, $          ; lệnh 2. TTimer
                  DJNZ R0, AAA        ; lệnh 2. TTimer
                  RET
```

c. $t_{Delay} = 1s$ , $f_{OSC} = 12MHz$
Ta chọn $[R_n] = 10$, $[R_m] = 200$, $[R_o] = 250$

$t_{Delay}$ = 10 x 200 x 250x 2x12/12 = 1000000µs
Chương trình:

```
DELAY:
        MOV R0, #10      ; lệnh 1. TTimer
BBB:    MOV R1, #200     ; lệnh 1. TTimer
AAA:    MOV R2, #250     ; lệnh 1. TTimer
        DJNZ R2, $       ; lệnh 2. TTimer
        DJNZ R1, AAA     ; lệnh 2. TTimer
        DJNZ R0, BBB     ; lệnh 2. TTimer
        RET              ; lệnh 2. TTimer
```

**4.4.2 Phương pháp tính thời gian $t_{Delay}$ có độ chính xác**

Với $t_{Delay}$ = 100µs , $f_{OSC}$ = 12MHz thì $t_{Delay}$ chính xác là:
$t_{Delay}$ = 1. $T_{Timer}$ + 2. $T_{Timer}$ x 50 + 2 . $T_{Timer}$ = 103 $T_{Timer}$
Với $t_{Delay}$ = 10ms , $f_{OSC}$ = 12MHz thì $t_{Delay}$ chính xác là:
$t_{Delay}$ = 1. $T_{Timer}$ + (1. $T_{Timer}$ + 2 . $T_{Timer}$ x 250 + 2. $T_{Timer}$ ) x 20 + 2. $T_{Timer}$ = 10065 $T_{Timer}$
$t_{Delay}$ = 1s , $f_{OSC}$ = 12MHz thì $t_{Delay}$ chính xác là:
$t_{Delay}$ = 1. $T_{Timer}$ + (1. $T_{Timer}$ + (1 . $T_{Timer}$ + 2. $T_{Timer}$ x 250 + 2. $T_{Timer}$ ) x 200 + 2 . $T_{Timer}$ )x 10 + 2. $T_{Timer}$ = 1006033 $T_{Timer}$

**4.4.3 Tính toán giá trị cần nạp cho bộ định thời và chế độ hoạt động cho bộ định thời :**

$$N = -\frac{f_{osc}}{12} \times t_{Delay}$$

**Ví dụ 1:** Cho thời gan trễ $t_{Delay}$ = 100µs , $f_{OSC}$ = 12MHz. Tính giá trị cần nạp cho bộ định thời và chế độ hoạt động cho bộ định thời.

$$N = -\frac{f_{osc}}{12} \times t_{Delay} = \frac{12 \times 10^6}{12} \times 100 \times 10^{-6} = -100$$

Vậy N = -100 hoặc N = 9CH

$$T_{Timer} = \frac{1}{f_{Timer}} = \frac{12}{f_{OSC}} = \frac{12}{12.10^6} = 10^{-6}(s) = 1\mu s$$

Vì $t_{Delay}$ = 100µs = 100 $T_{Timer} \leq$ 256. $T_{Timer}$
Nên ta có thể chọn Timer ở chế độ 1 ( chế độ 16 bit) hoặc chế độ 2(chế độ 8 bit tự động nạp lại.

**Ví dụ 2:** Cho thời gan trễ $t_{Delay}$ = 10ms , $f_{OSC}$ = 12MHz. Tính giá trị cần nạp cho bộ định thời và chế độ hoạt động cho bộ định thời.

$$N = -\frac{f_{osc}}{12} \times t_{Delay} = \frac{12 \times 10^6}{12} \times 100 \times 10^{-6} = -10000$$

Vậy N = -10000 hoặc N = D8F0H

$$T_{Timer} = \frac{1}{f_{Timer}} = \frac{12}{f_{OSC}} = \frac{12}{12.10^6} = 10^{-6}(s) = 1\mu s$$

Vì $t_{Delay}$ = 10000µs = 10000 $T_{Timer} \leq$ 65536. $T_{Timer}$
Nên ta có thể chọn Timer ở chế độ 1 ( chế độ 16 bit)

**Ví dụ 3:** Cho thời gan trễ $t_{Delay}$ = 1s , $f_{OSC}$ = 12MHz. Tính giá trị cần nạp cho bộ định thời và chế độ hoạt động cho bộ định thời.

$$N = -\frac{f_{osc}}{12} \times t_{Delay} = \frac{12 \times 10^6}{12} \times 100 \times 10^{-6} = -1000000$$

Vậy N = -1000000 (giá trị quá lớn không thể nạp trực tiếp vào các thanh ghi định thời THx/TLx)

Vì $t_{Delay}$ = 1000000µs = 1000000 $T_{Timer}$ > 65536. $T_{Timer}$

Nên ta phải chọn Timer ở chế độ 1 ( chế độ 16 bit) kết hợp với các thanh ghi để tạo vòng lặp.

Gọi N' là giá trị cần nạp cho các thanh ghi định thời.

$[R_n]$ là gía trị cần nạp cho thanh ghi kết hợp (vòng lặp 1)

$[R_m]$ là gía trị cần nạp cho thanh ghi kết hợp (vòng lặp 2)

$N = [R_n] \times [R_m] \times N'$

Ta tự chọn N' = -10000 $\rightarrow [R_n]$=100

**Lưu ý:**

N' được chọn sao cho phù hợp với qui định chọn giá trị cần nạp cho các thanh ghi định thời ở từng chế độ

$[R_n]$, $[R_m] \leq 255$, đặc biệt nếu chọn $[R_n]$, $[R_m]$ = 0 thì tương ứng với trường hợp ta chọn $[R_n] = 256 \Rightarrow$ Giá trị cần nạp cho các thanh ghi định thời là -10000 và giá trị cần nạp cho thanh ghi kết hợp là 100.

**Ví dụ 4:** Cho thời gan trễ $t_{Delay}$ = 60s , $f_{OSC}$ = 12MHz. Tính giá trị cần nạp cho bộ định thời và chế độ hoạt động cho bộ định thời.

$$N = -\frac{f_{osc}}{12} \times t_{Delay} = -\frac{12 \times 10^6}{12} \times 60 = -60000000$$

Vậy N = -60000000 (giá trị quá lớn không thể nạp trực tiếp vào các thanh ghi định thời THx/TLx)

Vì $t_{Delay}$ = 60000000µs = 60000000 $T_{Timer}$ > 65536. $T_{Timer}$

Nên ta phải chọn Timer ở chế độ 1( chế độ 16 bit) kết hợp với các thanh ghi để tạo vòng lặp.

Gọi N' là giá trị cần nạp cho các thanh ghi định thời.

$[R_n]$ là gía trị cần nạp cho thanh ghi kết hợp (vòng lặp 1)

$[R_m]$ là gía trị cần nạp cho thanh ghi kết hợp (vòng lặp 2)

$N = [R_n] \times [R_m] \times N'$

Ta tự chọn N' = -10000 $\rightarrow [R_m]$=100$\rightarrow [R_n]$=60

**4.4.4 Phương pháp tính thời gian $t_{Delay}$ dùng Timer:**

Phương pháp dùng dùng Timer thì việc định thời gian cũng xuất hiện 1 sai số . Vì để đơn giản trong việc tính toán ta bỏ qua không tính đến thời gian cần thiết để thực hiện từng lệnh trong chương trình, chỉ quan tâm đến giá trị cần nạp cho bộ định thời sao cho Timer định được khoảng thời gian mà ta yêu cầu.

Độ chính xác của chương trình định thời dùng Timer không phụ thuộc vào giá trị cần nạp cho Timer mà nó chỉ phụ thuộc vào số lượng lệnh sử dụng trong chương trình:

Ví dụ :

$t_{Delay}$ = 1s , $f_{OSC}$ = 12MHz

Chương trình:

DELAY:

```
        MOV R0, #10      ; lệnh 1. TTimer
BBB:    MOV R1, #200     ; lệnh 1. TTimer
AAA:    MOV R2, #250     ; lệnh 1. TTimer
        DJNZ R2, $       ; lệnh 2. TTimer
        DJNZ R1, AAA     ; lệnh 2. TTimer
        DJNZ R0, BBB     ; lệnh 2. TTimer
        RET              ; lệnh 2. TTimer
```

Như vậy: $t_{Delay(cx)} = 11.T_{Timer} + t_{Delay} = 111(\mu s)$

**4.4.5 Bài tập áp dụng:**

**Ví dụ 1:**

Viết chương trình tạo trễ (delay)100µs dùng Timer 0, biết rằng bộ dao động trên chip sử dụng thạch anh 12MHz.

Bước 1: Tìm tần số dao động và chu kỳ xung nhịp timer:

- Ta có fosc =12MHz
- Suy ra: ftimer = fosc/12 =12MHz/12 =1MHz
- Ttimer = 1/ ftimer = 1/106 = 1µs

Bước 2: Xác định giá trị cần nạp cho các thanh ghi timer

- Ta có $t_{delay}$ =100µs
- Mà Ttimer = 1µs
- Suy ra : $t_{delay}$ =100Ttimer
- Do timer hoạt động như là một bộ đếm lên và khi bộ đếm bị tràn thì nó sẽ quay lại về giá trị 0 nên giá trị phải nạp cho các thanh ghi timer là -100

Bước 3: Xác định giá trị cần nạp cho thanh ghi chế độ timer TMOD

- Do $t_{delay}$ =100Ttimer , giá trị này <255 nên chúng ta có thể chọn timer ở chế độ 1(16bit) hay chế độ 2 (8bit tự động nạp lại)

Bước 4: Cho timer chạy

Bước 5: Chờ cờ báo tràn

Bước 6: Xóa cờ báo tràn

Bước 7: Dừng timer

Chương trình hoàn chỉnh khi sử dụng Timer 0 ở chế độ 2:

```
MOV TMOD,# 0000 0010 B
MOV TH0,# -100
SETB TR0
JNB TF0,$
CLR TF0
CLR TR0
END
```

Chương trình hoàn chỉnh khi sử dụng Timer 0 ở chế độ 1:

```
MOV TMOD,# 0000 0001 B
MOV TH0,# high(-100) ; lệnh này sẽ chuyển giá trị -100 sang mã nhị phân và sau
đó lấy 8bit cao của giá trị này gán vào thanh ghi TH0
MOV TL0,# low(-100) ; lệnh này sẽ chuyển giá trị -100 sang mã nhị phân và sau đó
lấy 8bit thấp của giá trị này gán vào thanh ghi TL0
```

```
SETB TR0
JNB TF0,$
CLR TF0
CLR TR0
END
```

**Ví dụ 2:** Viết chương trình tạo trễ 10ms, sử dụng timer 1, tần số dao động trên chip là 12MHz

Bước 1: Tìm tần số dao động và chu kỳ xung nhịp timer:

- Ta có fosc =12MHz
- Suy ra: $f_{timer}$ = fosc/12 =12MHz/12 =1MHz
- Ttimer = 1/ $f_{timer}$ = 1/106 = 1µs

Bước 2: Xác định giá trị cần nạp cho các thanh ghi timer

- Ta có tdelay =10ms
- Mà $T_{timer}$ = 1µs
- Suy ra : $t_{delay}$ =10000Ttimer

Bước 3: Xác định giá trị cần nạp cho thanh ghi chế độ timer TMOD

- Do $t_{delay}$ =10000$T_{timer}$ , nên ta chọn timer ở chế độ 1(16bit)

Chương trình hoàn chỉnh khi sử dụng Timer 1 ở chế độ 1:

```
MOV TMOD,#0001 0000 B
MOV TH1,#high(-10000)
MOV TL1,#low(-10000)
SETB TR1
JNB TF1, $
CLR TF1
CLR TR1
END
```

Ví dụ 3: Viết chương trình tạo sóng vuông 50% có tần số 10KHz ở ngõ ra P1.0, biết tần số dao động trên chip là 12MHz

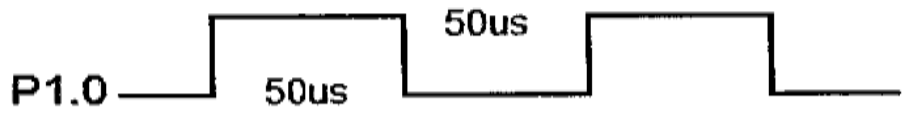
Bước 1: Tìm tần số dao động và chu kỳ xung nhịp timer:

- Ta có $f_{osc}$ =12MHz
- Suy ra: $f_{timer}$ = fosc/12 =12MHz/12 =1MHz
- Ttimer = 1/ $f_{timer}$ = 1/106 = 1µs

Bước 2: Xác định giá trị cần nạp cho các thanh ghi timer

- Ta có tần số xung vuông là 10KHz nên chu kỳ của xung vuông là T=1/(10.$10^3$)=$10^{-4}$s=100µs
- Do xung vuông 50% nên thời gian xung ở mức cao bằng thời gian xung ở mức thấp :

$t_H = t_L = T/2 = 50\mu s$



– Ở đây ta có thể sử dụng 1 timer để tạo thời gian trễ là 50µs

Cách 1:

```
MOV TMOD, # 0000 0010B
MOV TH0,# -50
SETB TR0
LOOP: JNB TF0,$
CLR TF0
CPL P1.0
SJMP LOOP
END
```

Cách 2:

```
LOOP: SETB P1.0
ACALL Delay_50us
CLR P1.0
ACALL Delay_50us
SJMP LOOP
Delay_50us:
MOV TMOD, # 0000 0010B
MOV TH0,# -50
SETB TR0
JNB TF0,$
CLR TF0
RET
END
```

**Ví dụ 4:** Giả sử tần số XTAL = 11.0592MHz. Hãy tìm

a) Tần số của sóng vuông được tạo ra trên chân P1.0 trong chương trình sau

b) Tần số nhỏ nhất có thể có được bằng chương trình này và giá trị TH để đạt được điều đó.

```
        MOV  TMOD, #20H; Chọn Timer1/ chế độ 2/ 8 bit/ tự nạp lại.

        MOV  TH1, #5            ; TH1 = 5
        SETB TR1                ; Khởi động Timer1
BACK:   JNB  TF1, BACK
        CPL  P1.0
        CLR  TF1                ; Xoá cờ bộ định thời TF1

        SJMP BACK       ; Chế độ 2 tự động nạp lại.
```

**Lời giải:**

a) Trước hết để ý đến đích của lệnh SJMP. Trong chế độ 2 ta không cần phải nạp lại TH vì nó là chế độ tự nạp. Bây giờ ta lấy (256 - 05).1.085μs = 251×1.085μs = 272.33μs là phần cao của xung. Cả chu kỳ của xung là T = 544.66μs và tần số là $\frac{1}{T}$ = 1,83597kHz.

b) Để nhận tần số nhỏ nhất có thể ta cần tạo T chu kỳ lớn nhất có thể có nghĩa là TH = 00. Trong trường hợp này ta có T = 2 × 256 × 1.085μs = 555.52μs và tần số nhỏ nhất sẽ là $\frac{1}{T}$ = 1,8kHz.

**Ví dụ 5:**

Hãy tìm tần số của xung vuông được tạo ra trên P1.0.

Lời giải:

```
            MOV  TMOD, #2H  ; Chọn Timer0, chế độ 1 (8 bít tự nạp lại)

AGAIN:      MOV  TH0, #0    ; Nạp TH0 = 00
            MOV  R5, #250   ; Đếm cho độ trễ lớn
            ACALL DELAY
            CPL  P1.0
            SJMP AGAIN
DELAY:      SETB TR0        ; Khởi động Timer0
BACK:       JNB  TF0, BACK
            CLR  TR0        ; Dừng Timer0.
            CLR  TF0        ; Xoá cờ TF0 cho vòng sau.

            DJNZ R5, DELAY

            RET
```

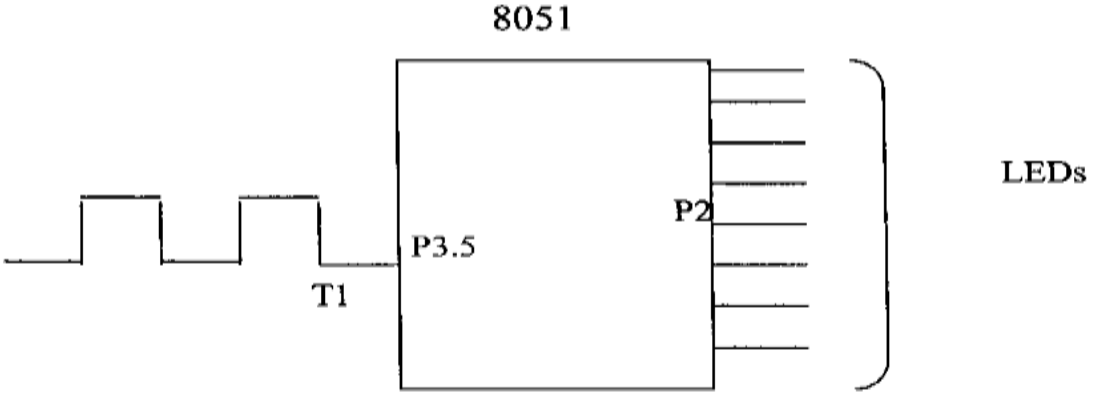
T = 2 × (250 × 256 × 1.085μs) = 1.38.88ms và f = 72Hz.

**Ví dụ 6:** Giả sử ta đang lập trình chế độ 2 hãy tìm các giá trị (dạng Hex) cần nạp vào TH cho các trường hợp sau:

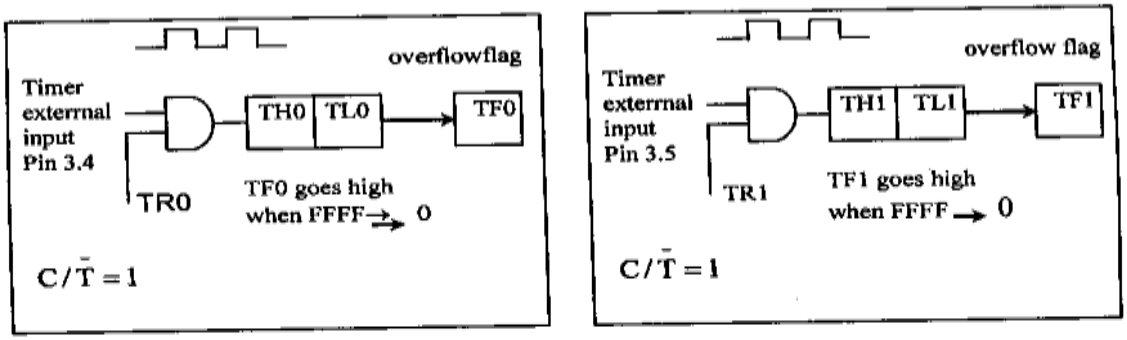
a) MOV TH1, #200

```
MOV  P2, A          ; Đưa TL1 hiển thị ra cổng P2.
JNB  TF1, Back      ; Duy trì nó nếu TF = 0
CLR  TR1            ; Dừng bộ đếm
CLR  TF1            ; Xoá cờ TF
SJMP AGAIN          ; Tiếp tục thực hiện
```

Để ý trong chương trình trên về vai trò của lệnh "SETB P3.5" vì các cổng được thiết lập dành cho đầu ra khi 8051 được cấp nguồn nên ta muốn P3.5 trở thành đầu vào thì phải bật nó lên cao. Hay nói cách khác là ta phải cấu hình (đưa lên cao) chân T1 (P3.5) để cho phép các xung được cấp vào nó.




Trong ví dụ 7 chúng ta sử dụng bộ Timer1 như bộ đếm sự kiện để nó đếm lên mỗi khi các xung đồng hồ được cấp đến chân P3.5. Các xung đồng hồ này có thể biểu diễn số người đi qua cổng hoặc số vòng quay hoặc bất kỳ sự kiện nào khác mà có thể chuyển đổi thành các xung.




a) Bộ Timer0 với đầu vào ngoài (chế độ 1)

# Chương 5: Hoạt động của port nối tiếp ( SERIAL PORT)

## 5.1 Giới thiệu

Các máy tính truyền dữ liệu theo hai cách: Song song và nối tiếp. Trong truyền dữ liệu song song thường cần 8 hoặc nhiều đường dây dẫn để truyền dữ liệu đến một thiết bị chỉ cách xa vài bước. Ví dụ của truyền dữ liệu song song là các máy in và các ổ cứng, mỗi thiết bị sử dụng một đường cáp với nhiều dây dẫn. Mặc dù trong các trường hợp như vậy thì nhiều dữ liệu được truyền đi trong một khoảng thời gian ngắn bằng cách dùng nhiều dây dẫn song song nhưng khoảng cách thì không thể lớn được. Để truyền dữ liệu đi xa thì phải sử dụng phương pháp truyền nối tiếp. Trong truyền thông nối tiếp dữ liệu được gửi đi từng bít một so với truyền song song thì một hoặc nhiều byte được truyền đi cùng một lúc.

89C51 có một port nối tiếp hoạt động ở nhiều chế độ trên một dãi tần số rộng

Chức năng chủ yếu là thực hiện chuyển đổi song song sang nối tiếp với dữ liệu xuất và chuyển đổi nối tiếp sang song song với dữ liệu nhập.

Port nối tiếp được truy xuất qua các chân TXD và RXD. Dữ liệu được thu (nhập) qua chân RXD (P3.0) và dữ liệu được phát (xuất) qua chân TXD

(P3.1)

Port nối tiếp cho hoạt động song công (full duplex: thu và phát đồng thời) và đệm thu (receiver buffering) cho phép một ký tự sẽ được thu và được giữ trong khi ký tự thứ hai được nhận. Nếu CPU đọc ký tự thứ nhất trước khi ký tự thứ hai được thu đầy đủ thì dữ liệu sẽ không bị mất.

Hai thanh ghi chức năng đặc biệt cho phép phần mềm truy xuất đến port nối tiếp là: SBUF và SCON.

Bộ đệm port nối tiếp (SBUF) ở địa chỉ 99H nhận dữ liệu để thu hoặc phát.

Thanh ghi điều khiển port nối tiếp (SCON) ở địa chỉ 98H là thanh ghi có địa chỉ bit chứa các bit trạng thái và các bit điều khiển. Các bit điều khiển đặt chế độ hoạt động cho port nối tiếp, và các bit trạng thái báo kết thúc việc phát hoặc thu ký tự . Các bit trạng thái có thể được kiểm tra bằng phần mềm hoặc có thể lập trình để tạo ngắt.

## 5.2 Các thanh ghi PORT

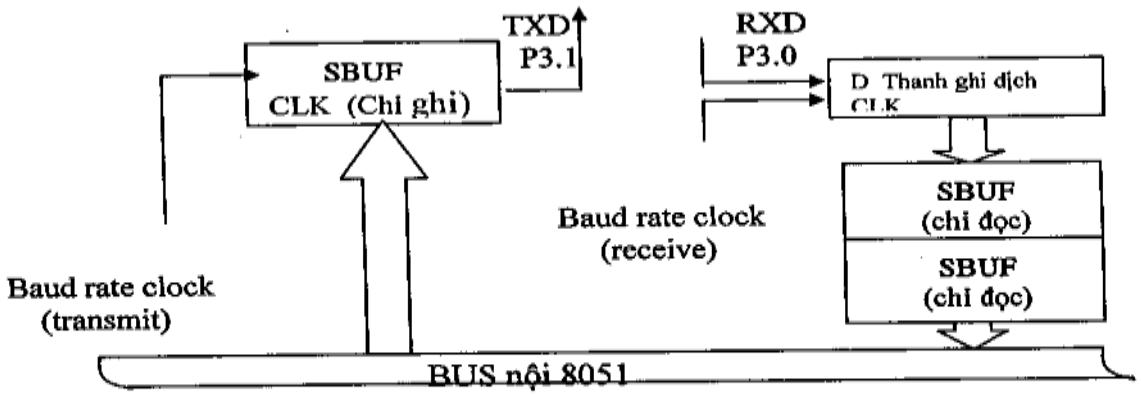
### 5.2.1 Thanh ghi đệm nối tiếp SBUF(Serial Buffer Register)

Thanh ghi SBUF đóng vai trò vừa là bộ đệm phát vừa là bộ đệm thu:

- Dữ liệu cần phát đi sẽ được ghi vào SBUF và được phát qua ngõ TXD, trong trường hợp này nó là bộ đệm phát
- Dữ liệu thu sẽ được nạp vào SBUF thông qua ngõ RXD và đọc dữ liệu từ thanh ghi SBUF để truy xuất dữ liệu thu được, trong trường hợp này nó là bộ đệm thu

Tốc độ baud của port nối tiếp có thể lấy từ bộ dao động trên chip hoặc sử dụng timer.

Cấu trúc của thanh ghi SBUF:




**Ví dụ:** các lệnh ghi dữ liệu vào SBUF và đọc dữ liệu từ SBUF

```
MOV  SBUF, # 40H; phát giá trị 40H qua port nối tiếp
MOV  SBUF, A          ; phát nội dung của A qua port nối tiếp
MOV A, SBUF           ; đọc dữ liệu thu được từ port nối tiếp
```

Khi muốn phát-ra các dữ liệu 8 bit chúng ta phải di chuyển các dữ liệu này vào thanh ghi SBUF

Khi muốn nhận-vào, dữ liệu 8 bit sẽ lần lượt nằm trong thanh ghi SBUF, chúng ta phải cho chuyển dữ liệu này ra ngoài để nó có thể tiếp tục nhận vào các dữ liệu 8 bit khác.

### 5.2.2 Thanh ghi điều khiển port nối tiếp SCON

Thanh ghi này được dùng để điều khiển chế độ hoạt động của port nối tiếp

| Bit | Ký hiệu | Địa chỉ |
|---|---|---|
| SCON.7 | SM0 | 9FH |
| SCON.6 | SM1 | 9EH |
| SCON.5 | SM3 | 9DH |
| SCON.4 | REN | 9CH |
| SCON.3 | TB8 | 9BH |
| SCON.2 | RB8 | 9AH |
| SCON.1 | TI | 99H |
| SCON.0 | RI | 98H |

Thanh ghi này được địa chỉ hóa từng bit

| SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

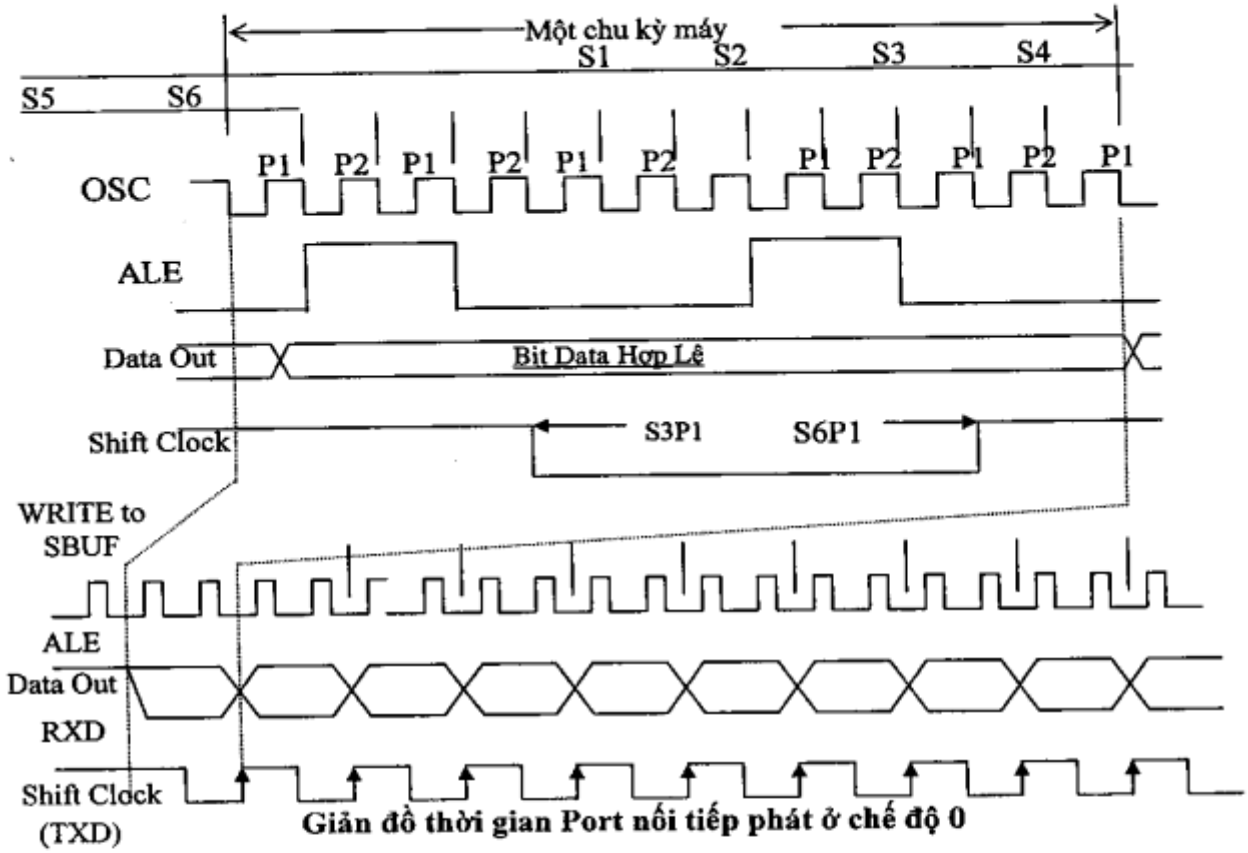
Trong đó :
- REN = 1: Cho phép thu
- REN = 0: cấm
- TI : cờ ngắt phát
  - ➢ TI được đặt lên mức 1 khi bộ đệm phát đã rỗng (kết thúc việc phát 1 ký tự).
  - ➢ Được xóa bằng phần mềm
- RI : cờ ngắt thu
  - ➢ RI được đặt lên mức 1 khi bộ đệm thu đã đầy (đã nhận đủ 1 ký tự)
  - ➢ Được xóa bằng phần mềm
- TB8: bit thứ 9 được phát trong chế độ UART 9bit, bit này được đặt hoặc xóa bằng phần mềm
- RB8: bit thứ 9 thu được trong chế độ UART 9bit
- SM2:
  - ➢ SM2 =1: Port nối tiếp hoạt động ở chế độ truyền thông đa xử lý trong các chế độ 2 và 3; bit cờ RI sẽ không được đặt lên 1 nếu bit thứ 9 thu được là 0
- SM0, SM1: chọn chế độ hoạt động:

| SM0 | SM1 | Chế độ | Mô tả | Tốc độ baud |
|---|---|---|---|---|
| 0 | 0 | 0 | Thanh ghi dịch | Cố định (Fosc /12 ) |
| 0 | 1 | 1 | UART 8 bit | Thay đổi ( đặt bằng timer ) |
| 1 | 0 | 2 | UART 9 bit | Cố định (Fosc /12 hoặc Fosc/64 ) |
| 1 | 1 | 3 | UART 9 bit | Thay đổi ( đặt bằng timer ) |

### 5.3 Các chế độ hoạt động:

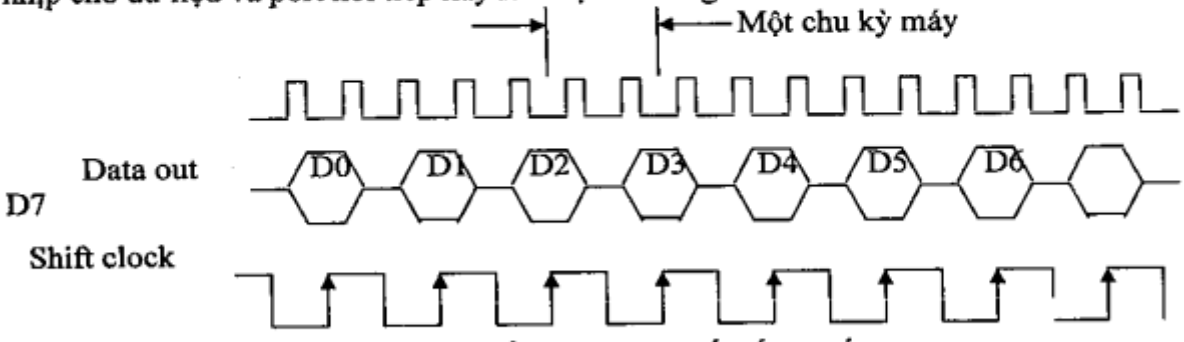
#### 5.3.1 Chế độ thanh ghi dịch 8 bit (Mode 0)

- Chọn các bit SM0 = 0 và SM1 = 0
- Dữ liệu nối tiếp được vào và ra qua ngõ RXD.
- Ngõ TXD là ngõ xuất xung nhịp dịch, với 1 chu kỳ xung nhịp dịch thì 1 bit sẽ được phát hoặc thu
- Dữ liệu được phát hoặc thu với bit đầu tiên là bit LSB
- Tốc độ baud cố định ở 1/12 tần số dao động trên chip
- Việc phát ký tự được khởi động bằng bất cứ lệnh nào ghi dữ liệu vào thanh ghi SBUF, trước khi phát phải kiểm tra bộ đệm phát đã rỗng. Dữ liệu dịch ra ngòai trên đường RXD (P3.0) với các xung nhịp được gửi ra đường TXD (P3.1). Mỗi bit phát đi hợp lệ (trên RXD) trong một chu kỳ máy, tín hiệu xung nhập xuống thấp ở S3P1 và trở về cao ở S6P1.




**Giản đồ thời gian Port nối tiếp phát ở chế độ 0**

Việc thu được khởi động khi cho phép bộ thu (REN) = l và ngắt thu (RI)= 0. Quy tắc tổng quát đặt REN khi bắt đầu chương trình để khởi động port nối tiếp, rồi xóa RI để bắt đầu nhận dữ liệu. Khi RI bị xóa, các xung nhịp được đưa ra đường TXD, bắt đầu chu kỳ máy kế tiếp và dữ liệu theo xung nhịp ở đường RXD. Lấy xung nhịp cho dữ liệu và port nối tiếp xảy ra ở cạnh đường của TXD.




**Giản đồ thời gian thu nối tiếp ở chế độ 0**

- Chương trình con để phát 1byte dữ liệu

```
PHAT: JNB TI,$
CLR TI
MOV SBUF, A
RET
```

- Việc thu ký tự được khởi động khi bit cho phép REN ở mức 1 và cờ ngắt thu RI ở mức 0
- Chương trình con dùng để thu 1byte dữ liệu:

```
THU:  JNB RI,$
CLR RI
MOV A, SBUF
RET
```

**5.3.2 Chế độ 1 (UART 8 bit với tốc độ baud thay đổi được):**

Trong chế độ 1, port nối tiếp của 89c51 hoạt động như 1 bộ thu phát không đồng bộ 8 bit có tốc độ baud thay đổi (UART – Universal Asynchronous Receiver Transmitter).

Bộ UART là một dụng cụ thu phát dữ liệu nối tiếp với mỗi ký tự dữ liệu đi trước là bit start ở mức thấp và theo sau bit stop ở mức cao. Đôi khi xen thêm bit kiểm tra chẵn lẻ giữa bit dữ liệu cuối cùng và bit stop. Hoạt động chủ yếu của UART là chuyển đổi dữ liệu phát từ song song sang nối tiếp và biến đổi dữ liệu thu từ nối tiếp thành song song.

- Chọn các bit SM0 = 0 và SM1 = 1
- Khuôn dạng dữ liệu gồm 10 bit (1 bit start luôn luôn là 0, 8 bit dữ liệu bit LSB luôn luôn được thu phát đầu tiên và 1 bit stop luôn luôn là 1) được phát trên TXD hoặc thu trên RXD.
- Trong 8951 tốc độ baud được đặt bằng tốc độ báo tràn của timer 1.

Quá trình phát dữ liệu:

- Ghi dữ liệu cần phát vào SBUF .
- Dữ liệu từ SBUF được dịch ra ngoài trên đường TXD bắt đầu bằng bit Start, theo sau là 8bit dữ liệu và sau cùng là bit Stop
- Cờ ngắt phát TI sẽ được đặt lên 1 khi xuất hiện bit Stop trên được TXD.
- Tốc độ baud: do người lập trình thiết lập và được qui định bởi tốc độ tràn của Timer1.
- Thời gian của 1 bit trên đường truyền: bằng nghịch đảo của tốc độ baud.

Quá trình thu dữ liệu:

- Được khởi động bằng một sự chuyển trạng thái từ mức 1 xuống mức 0 trên đường RXD.
- Việc thu dữ liệu bắt đầu bằng 8 bit dữ liệu được dịch vào trong SBUF → Stop bit ( bit thứ 9) được đưa vào bit RB8 thuộc thanh ghi SCON → cờ RI = 1

**5.3.3 Chế độ 2 - UART 9 bit tốc độ baud cố định**

- Chọn các bit SM0 = 1 và SM1 = 0
- Trong chế độ này dữ liệu thu/phát có 11bit bao gồm 1bit Start, 8 bit dữ liệu, 1

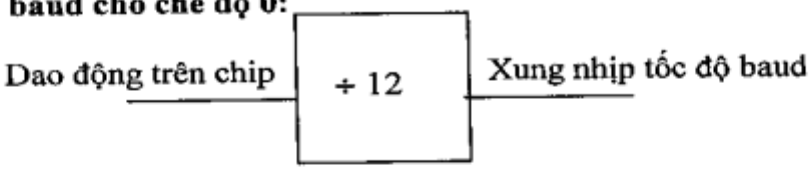
bit dữ liệu thứ 9 (có thể lập trình được) và cuối cùng là 1bit Stop
- Khi phát thì bit dữ liệu thứ 9 chính là dữ liệu được đưa vào bit TB8 trong thanh ghi SCON (có thể là parity chẵn hoặc lẻ)
- Khi thu thì bit dữ liệu thứ 9 được đưa vào bit RB8 trong thanh ghi SCON
- Tốc độ baud không đổi và bằng $f_{OSC}/32$ (hoặc 64)

### 5.3.4 Chế độ 3 - UART 9 bit tốc độ baud thay đổi được
- Chế độ này giống như ở chế độ 2 ngoại trừ tốc độ baud có thể lập trình được và được cung cấp bởi Timer.Thật ra các chế độ 1, 2, 3 rất giống nhau. Cái khác biệt là ở tốc độ baud (cố định trong chế độ 2, thay đổi trong chế độ 1 và 3) và ở số bit data (8 bit trong chế độ 1,9 trong chế độ 2 và 3).

## 5.4 Tốc độ baud của port nối tiếp
- Tốc độ baud là số bit dữ liệu được truyền trong 1 giây. Đơn vị tính là bit/giây (bps)
- Để tạo tốc độ baud thì khởi động cho Timer 1 tràn sau một khoảng thời gian tương ứng với tốc độ baud.
- Tốc độ baud cũng bị ảnh hưởng bởi 1 bit trong thanh ghi điều khiển nguồn cung cấp (PCON) bit 7 của PCON là bit SMOD. Đặt bit SMOD lên 1 làm gấp đôi tốc độ baud trong các chế độ 1, 2 và 3.
- Vì PCON không được định địa chỉ theo bit, nên để đặt bit SMOD lên 1 cần phải theo các lệnh sau:

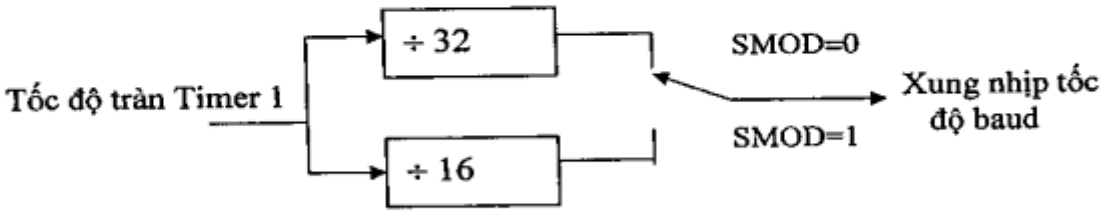
```
MOV A,PCON ; lấy giá trị hiện thời của PCON
SETB ACC.7 ; đặt bit SMOD lên 1
MOV PCON,A ; ghi giá trị ngược về PCON
```

### 5.4.1 Tốc độ baud cho chế độ 0:




$$Baudrate = \frac{f_{osc}}{12}$$

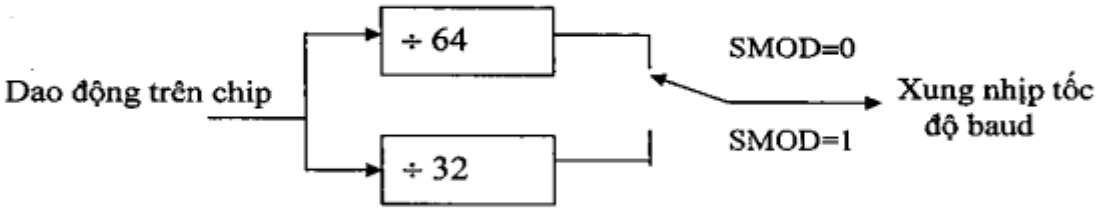
### 5.4.2 Tốc độ baud cho chế độ 1,3:



$$Baudrate = \frac{Timer1overflowrate}{16}; SMOD = 1$$

$$Baudrate = \frac{Timer1overflowrate}{32}; SMOD = 0$$

### 5.4.3 Tốc độ baud cho chế độ 2:




$$Baudrate = \frac{f_{OSC}}{32}; SMOD = 1$$

$$Baudrate = \frac{f_{OSC}}{64}; SMOD = 0$$

### 5.5 Sử dụng Timer 1 để tạo xung nhịp tốc độ baud

Các bước tạo xung nhịp tốc độ baud bằng Timer 1:

- Chọn chế độ cho Timer 1:
  - Chế độ 13 bit (chế độ 0)
  - Chế độ 16 bit (chế độ 1)
  - Chế độ 8 bit tự nạp lại (chế độ 2)
  - Chế độ chia xẻ (chế độ 3)
- Nạp giá trị thích hợp vào thanh ghi TH1 để có tốc độ tràn đúng, tạo ra tốc độ baud cho port nối tiếp.
- Chọn tốc độ baud:

Gọi M là giá trị cần nạp cho thanh ghi TH1 để có tốc độ Baud theo yêu cầu

$$M = -\frac{f_{Timer}}{Timer1overflowrate};$$

$$f_{Timer} = \frac{f_{OSC}}{12}$$

$$Baudrate = \frac{Timer1overflowrate}{16}; SMOD = 1$$

$$Baudrate = \frac{Timer1overflowrate}{32}; SMOD = 0$$

**Ví dụ 1:**

Tìm giá trị TH1 (ở dạng thập phân và hex) để đạt tốc độ baud cho các trường hợp sau.
a) 9600 b) 4800 nếu SMOD = 1 với tần số XTAL = 11.0592MHz

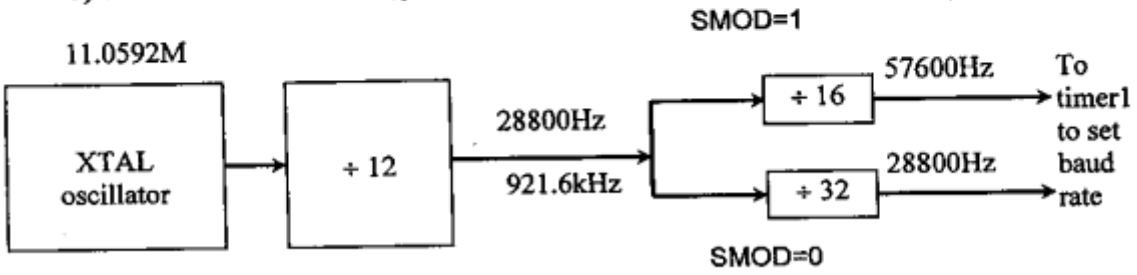
**Lời giải:**

Với tần số XTAL = 11.0592MHz và SMOD = 1 ta có tần số cấp cho Timer 1 là

$f_{Timer} = f_{OSC}/12 = 11.0592MHz / 12 = 921600Hz$

Baudrate = 921600 /16 = 57600 với SMOD =1

a) 57600/9600 = 6 do vậy TH1 = - 6 hay TH1 = FAH
b) 57600/4800 = 12 do vậy TH1 = - 12 hay TH1 = F4H




**Ví dụ 2:**

Hãy tìm tốc độ baud nếu TH1 = -2, SMOD = 1 và tần số XTAL = 11.0592MHz.

**Lời giải:**

Với tần số XTAL = 11.0592MHz và SMOD = 1 ta có tần số cấp cho Timer 1 là 57.6kHz. Tốc độ baud là 57.600kHz/2 = 28.800.

### 5.6 Các bước cơ bản lập trình port nối tiếp:

#### 5.6.1 Lập trình phát dữ liệu nối tiếp:

- Chọn chế độ hoạt động cho port nối tiếp:
  MOV SCON, #.....
- Chọn chế độ hoạt động Timer 1, cho bit Gate = 0 và C/T = 0
  MOV TMOD, #.....
- Chọn giá trị thích hợp căn cứ vào tốc độ baud cho Timer 1:
  MOV TH1,#....
- Cho Timer 1 chạy:
  SETB TR1
- Kiểm tra xem đã phát xong toàn bộ dữ liệu trước đó hay chưa?
  JNB TI, $

Hoặc

  WAIT : JNB TI, WAIT

- Xóa cờ ngắt TI chuẩn bị cho lần phát dữ liệu tiếp theo:
  CLR TI
- Ghi dữ liệu cần phát vào port nối tiếp để phát đi
  MOV SBUF, …

- Quay trở lại bước kiểm tra cờ phát TI để phát một dữ liệu tiếp.

**5.6.2 Lập trình nhận (thu) dữ liệu nối tiếp:**

- Chọn chế độ hoạt động cho port nối tiếp:
  MOV SCON, #.....
- Chọn chế độ hoạt động Timer 1, cho bit Gate = 0 và C/T = 0
  MOV TMOD, #.....
- Chọn giá trị thích hợp căn cứ vào tốc độ baud cho Timer 1:
  MOV TH1,#....
- Cho Timer 1 chạy:
  SETB TR1
- Kiểm tra xem đã thu xong toàn bộ dữ liệu trước đó hay chưa?
  JNB RI, $

Hoặc

  WAIT : JNB RI, WAIT

- Xóa cờ ngắt TI chuẩn bị cho lần phát dữ liệu tiếp theo:
  CLR RI
- Cất dữ liệu vừa thu được vào
  MOV …, SBUF
- Quay trở lại bước kiểm tra cờ thu RI để thu dữ liệu tiếp theo.

**5.6.3 Các ví dụ minh họa**

Ví dụ1: Tạo tốc độ baud 1200bps biết rằng bộ dao động trên chip sử dụng thạch anh 12MHz

- Ta có fosc =12MHz
- Suy ra: $f_{timer}$ = fosc/12 =12MHz/12 =1MHz
- $T_{timer} = 1/ f_{timer} = 1/106 = 1\mu s$
- Tốc độ tràn timer 1 là ( giả sử bit SMOD trong thanh ghi PCON có giá trị bằng 0):
  - Ta có f baud = f tràn T1 / 32
  - Suy ra $f_{tràn\ T1} = f_{baud} \times 32 = 1200x32 = 38400Hz$
  - Suy ra $T_{tràn\ T1} = 1/ f_{tràn\ T1} = 1/38400 = 26\mu s$
- $T_{tràn\ T1} = 26T_{timer}$

Như vậy giá trị cần nạp cho thanh ghi timer là -26

**Đoạn chương trình tạo tốc độ baud:**

```
MOV TMOD,#0010 0000B
MOV TH1,# -26
SETB TR1
```

**Ví dụ 2:**

Cho chương trình dưới đây với tần số XTAL = 11.0592MHz, hãy cho biết

a) chương trình này làm gì?

b) Tính toán tần số được Timer1 sử dụng để đặt tốc độ baud?

c) Tìm tốc độ baud truyền dữ liệu.

```
        MOV  A, PCON      ; Sao nội dung thanh ghi PCON vào thanh ghi ACC
```

```
        SETB ACC.7        ; Đặt D7 = 0
        MOV  PCON, A      ; Đặt SMOD = 1 để tăng gấp đôi tần số baud
        MOV  TMOD, #20H; Chọn bộ Timer1, chế độ 2, tự động nạp lại
        MOV  TH1, - 3     ; Chọn tốc độ baud 19200 vì SMOD = 1
        MOV  SCON, #50H; Đóng khung dữ liệu gồm 8 bít dữ liệu,1 Stop.
        SETB TR1          ; Khởi động Timer1
        MOV  A, #"B"      ; Truyền ký tự B
AA:     CLR  TI           ; Khẳng định TI = 0
        MOV  SBUF, A      ; Truyền nó
BB:     JNB  TI, BB       ; Chờ ở đây cho đến khi bít cuối được gửi đi
        SJMP AA           ; Tiếp tục gửi "B"
```

**Lời giải:**

a) Chương trình này truyền liên tục mã ASCII của chữ B (ở dạng nhị phân là 0100 0010)

b) Với tần số XTAL = 11.0592MHz và SMOD = 1 trong chương trình trên ta có:

11.0592MHz/12 = 921.6kHz là tần số bộ định thời

921.6kHz/16 = 57.6kHz là tần số được Timer1 sử dụng để đặt tốc độ baud

c) 57.6kHz/3 = 19.200 là tốc độ cần tìm

Bài tập 1:

Viết chương trình con để truyền dữ liệu trong thanh ghi A ra port nối tiếp. Biết rằng port nối tiếp hoạt động ở chế độ UART 8bit với tốc độ baud là 1200bps, tần số thạch anh là 12MHz

- Sử dụng Timer1 để tạo tốc độ baud (giống ví dụ ở trên)
- Khởi động thanh ghi điều khiển port nối tiếp:
  - ➢UART 8bit : Mode 1 (SM0 SM1= 0 1)
  - ➢SM2=0
  - ➢Set bit cờ TI =1 để phát ký tự lần đầu tiên ngay khi khởi động

Chương trình:

```
        MOV TMOD,#20H
        MOV TH1,#-26
        SETB TR1
        MOV SCON,#0100 0010B
        CALL PHAT
PHAT:   JNB TI,$
        CLR TI
        MOV SBUF,A
        RET
    END
```

**Bài tập 2:**

Viết chương trình con để truyền ký tự mã ASCII trong thanh ghi A ra port nối tiếp với bit parity chẵn được thêm vào ở bit thứ 8. Khi ra khỏi chương trình con phải

trả lại ký tự mã ASCII ban đầu . Biết rằng port nối tiếp hoạt động ở chế độ UART 8bit với tốc độ baud là 1200bps, tần số thạch anh là 12MHz

- Mã ASCII là các ký tự 7bit
- Bit P trong thanh ghi PSW là bit parity chẵn
- Sử dụng Timer1 để tạo tốc độ baud (giống ví dụ ở trên)
- Khởi động thanh ghi điều khiển port nối tiếp:
  - ➢ UART 8bit : Mode 1 (SM0 SM1= 0 1)
  - ➢ SM2=0
  - ➢ Set bit cờ TI =1 để phát ký tự lần đầu tiên ngay khi khởi động

Chương trình:

```
        MOV TMOD,#20H
        MOV TH1,#-26
        SETB TR1
        MOV SCON,#0100 0010B
        CALL PHAT
PHAT:  MOV C, P
        MOV ACC.7, C
        JNB TI,$
        CLR TI
        MOV SBUF,A
        CLR ACC.7
        RET
        END
```

**Bài tập 3:**

Viết chương trình nhận 1 byte dữ liệu từ port nối tiếp. Biết rằng port nối tiếp hoạt động ở chế độ UART 8 bit với tốc độ baud là 1200bps, tần số thạch anh là 12MHz.

```
        MOV TMOD,#20H
        MOV TH1,#-26
        SETB TR1
        MOV SCON,#0101 0000B
        JNB RI,$
        CLR RI
        MOV A, SBUF
        END
```

# Chương 6: Ngắt (Interupts)

## 6.1 Giới thiệu

- Ngắt là một sự kiện mà nó gây ra treo tạm thời chương trình chính để thực thi chương trình khác .
- Ngắt cho phép hệ thống đáp ứng bất đồng bộ với một sự kiện và giải quyết sự kiện đó trong khi một chương trình khác đang thực thi
- Ngắt làm cho một hệ thống có ảo giác là đang làm nhiều việc đồng thời (dĩ nhiên là CPU không thể thực thi mỗi lần hơn 1 lệnh)
- Chương trình giải quyết các ngắt gọi là chương trình con phục vụ ngắt (hay bộ xử lý ngắt) ISR – Interrupt Service Routine.

## 6.2 Tổ chức ngắt

- 89C51 có 6 nguồn ngắt: 2 ngắt ngoài, 3 ngắt timer, 1 ngắt port nối tiếp
- Các nguồn ngắt bị cấm sau khi reset hệ thống và cho phép bằng phần mềm.
- Khi xảy ra hai hay nhiều ngắt đồng thời hoặc xảy ra một ngắt trong khi một ngắt khác đang được phục vụ, ta có 2 sơ đồ xử lý các ngắt sơ đồ chuỗi vòng và sơ đồ hai mức ưu tiên. Sơ đồ chuỗi vòng là sơ đồ cố định còn sơ đồ ưu tiên ngắt được lập trình bởi người sử dụng.

### 6.2.1 Cho Phép và Không Cho Phép Ngắt

Mỗi nguồn ngắt được cho phép hoặc không cho phép thông qua thanh ghi chức năng đặc biệt có các bit được địa chỉ hóa IE (Interrupt Enable) tại địa chỉ 0A8H.

| BIT | Ký hiệu | BIT địa chỉ | Mô tả (1: cho phép, 0: Cấm) |
|---|---|---|---|
| IE.7 | EA | AFH | Cho phép/cấm toàn bộc các ngắt |
| IE.6 | - | AEH | Không xác định |
| IE.5 | ET2 | ADH | Cho phép ngắt Timer 2 (8052) |
| IE.4 | ES | ACH | Cho phép ngắt port nối tiếp |
| IE.3 | ET1 | ABH | Cho phép ngắt Timer 1 |
| IE.2 | EX1 | AAH | Cho phép ngắt ngoài 1 |
| IE.1 | ET0 | A9H | Cho phép ngắt Timer 0 |
| IE.0 | EX0 | A8H | Cho phép ngắt ngoài 0 |

### 6.2.2 Ưu tiên ngắt.

Mỗi nguồn ngắt được lập trình g vào một trong hai mức ưu tiên qua thanh ghi chức năng đặc biệt có các bit được địa chỉ bit IP (Interrupt priority : ưu tiên ngắt) ở địa chỉ B8H.

| Bit | Ký hiệu | Địa chỉ bit | Mô tả (1=mức cao hơn,0=mức thấp) |
|---|---|---|---|
| IP.7 | - | BFH | Không được định nghĩa |
| IP.6 | - | BEH | Không được định nghĩa |
| IP.5 | PT2 | BDH | Ưu tiên cho ngắt từ timer 2 (8052) |
| IP.4 | PS | BCH | Ưu tiên cho ngắt Port nối tiếp |
| IP.3 | PT1 | BBH | Ưu tiên cho ngắt từ timer 1 |
| IP.2 | PX1 | BAH | Ưu tiên cho ngắt ngoài |
| IP.1 | PT0 | B9H | Ưu tiên cho ngắt từ timer 0 |
| IP.0 | PX0 | B8H | Ưu tiên cho ngắt ngoài 0 |

- Thanh ghi IP bị xóa sau khi reset hệ thống. Ý tưởng ưu tiên cho phép một IST sẽ bị ngắt bởi một ngắt khác có mức ưu tiên cao hơn
- Nếu hai ngắt có cùng độ ưu tiên xảy ra đồng thời thì sự hỏi vòng tuần tự sẽ xác định ngắt nào được phục vụ trước tiên
- Việc hỏi vòng tuần tự sẽ theo thứ tự sau:

Ngắt ngòai 0 → ngắt timer 0 → ngắt ngoài 1→ ngắt timer 1→ ngắt port nối tiếp → ngắt timer 2

Nghĩa là ngắt nào được hỏi vòng trước thì được thực hiện trước nếu có cùng độ ưu tiên

### 6.2.3 Vectơ ngắt

Các chương trình con phục vụ ngắt (ISR) phải được viết đúng vùng địa chỉ dành cho nó để khi có yêu cầu ngắt xảy ra thì giá trị của PC sẽ được trỏ đến đúng địa chỉ vùng ngắt đó gọi là các vectơ ngắt.

| INTERRUPT | FLAG | VECTOR ADDRESS |
|---|---|---|
| Reset hệ thống | RST | 0000 H |
| Ngắt ngoài 0 | IE0 | 0003 H |
| Ngắt Timer 0 | TF0 | 000B H |
| Ngắt ngoài 1 | IE1 | 0013 H |
| Ngắt Timer 1 | TF1 | 001B H |
| Ngắt Port nối tiếp | RI OR TI | 0023 H |
| Ngắt Timer 2 | TF2 OR EXF2 | 002B H |

### 6.2.4 Xử lý ngắt:

Khi một ngắt xảy ra và được CPU chấp nhận, chương trình chính bị ngắt quãng trong khi truy xuất đến ISR.

**Các bước khi thực hiện một ngắt.**

Khi kích hoạt một ngắt bộ vi điều khiển đi qua các bước sau:

- Nó kết thúc lệnh đang thực hiện và lưu địa chỉ của lệnh kế tiếp (PC) vào ngăn xếp.
- Nó cũng lưu tình trạng hiện tại của tất cả các ngắt vào bên trong (nghĩa là không lưu vào ngăn xếp).

- Nó nhảy đến một vị trí cố định trong bộ nhớ được gọi là bảng véc tơ ngắt nơi lưu giữ địa chỉ của một trình phục vụ ngắt.
- Bộ vi điều khiển nhận địa chỉ ISR từ bảng véctơ ngắt và nhảy tới đó. Nó bắt đầu thực hiện trình phục vụ ngắt cho đến lệnh cuối cùng của ISR là RETI (trở về từ ngắt).
- Khi thực hiện lệnh RETI bộ vi điều khiển quay trở về nơi nó đã bị ngắt. Trước hết nó nhận địa chỉ của bộ đếm chương trình PC từ ngăn xếp bằng cách kéo hai byte trên đỉnh của ngăn xếp vào PC. Sau đó bắt đầu thực hiện các lệnh từ địa chỉ đó.

### 6.2.5 Thiết kế chương trình dùng các ngắt

#### 6.2.5.1 Đối với các ISR có kích thước nhỏ (nhỏ hơn 8byte)

```
ORG   0000H; địa chỉ bắt đầu chương trình
LJMP  MAIN ; Nhảy đến chương trình chính, ngòai tất cả các ngắt
ORG   000BH; Địa chỉ vectơ ngắt timer 0
ISR-T0: ……..; Chương trình con phục vụ ngắt timer 0
……
RETI ; Kết thúc phục vụ ngắt timer 0, quay trở về chương trình chính
ORG   001BH; địa chỉ vectơ ngắt timer 1
ISR-T1: ……; Chương trình con phục vụ ngắt timer 1
……
RETI; Kết thúc phục vụ ngắt timer 1, quay trở về chương trình chính
      ORG   0030H; ngòai tất cả các ngắt
MAIN: ……..;Chương trình chính
  ……..
  ……..
  END
```

#### 6.2.5.2 Đối với ISR có kích thước lớn

Khi ISR có kích thước lớn hơn 8byte thì ISR được bắt đầu với một lệnh nhảy sang vùng nhớ khác của bộ nhớ chương trình để có thể mở rộng chiều dài của ISR:

```
ORG 0000H; địa chỉ bắt đầu chương trình
LJMP MAIN
ORG 000BH; Địa chỉ vectơ ngắt timer 0
LJMP ISR-T0
ORG 001BH; địa chỉ vectơ ngắt timer 1
LJMP IST-T1
ORG 0030H; Ngoài tất cả các vectơ ngắt
MAIN:  ……..
……..
……..
ISR-T0: ……..
……..
……..
RETI
```

ISR-T1: ……..
……..
……..
RETI
END

**6.2.6 Ngắt timer:**

- Các ngắt timer có địa chỉ Vector ngắt là 000BH (timer 0) và 001BH (timer 1).
- Ngắt timer xảy ra khi các thanh ghi timer (THx / TLx) bị tràn và cờ báo tràn (TFx) lên mức 1. Các cờ timer (TFx) không bị xóa bằng phần mềm. Khi cho phép các ngắt, TFx tự động bị xóa bằng phần cứng khi CPU chuyển đến ngắt.

**Ví dụ:**

Viết chương trình tạo sóng vuông 50% có tần số 10KHz ở ngõ P1.0 sử dụng ngắt timer 0. Biết rằng bộ dao động trên chip sử dụng thạch anh có tần số 12MHz.

- Phân tích yêu cầu:
- Bộ dao động sử dụng thạch anh 12MHz nên ta có chu kỳ của xung nhịp timer Ttimer =1µs
- Sóng vuông có tần số 10KHz nên chu kỳ sóng là T=10-4 s =100µs
- Do sóng vuông 50% nên ta có tlow = thigh =50µs
- Vậy ta sẽ sử dụng timer 0 để tạo thời gian trì hoãn là 50µs, và chương trình con phục vụ ngắt timer 0 sẽ chỉ cần làm công việc lấy bù bit P1.0. Chương trình chính sẽ nạp các giá trị thích hợp để khởi động cho timer 0 chạy và khởi động ngắt
    - Chương trình hòan chỉnh:

```
ORG 0000H; địa chỉ bắt đầu của chương trình
LJMP MAIN
ORG 000BH; Địa chỉ vectơ ngắt timer 0
ISR-T0: CPL P1.0 ; chương trình con phục vụ ngắt timer 0
RETI
ORG 0030H ; Bắt đầu chương trình chính tại điểm ngoài tất cả các ngắt
MAIN: MOV TMOD,#02H ; Timer 0 mode 2
MOV TH0,#-50 ; delay 50us
SETB TR0 ; cho timer 0 chạy
MOV IE,#82H ; cho phép ngắt timer 0
SJMP $ ; không làm gì cả
END
```

**6.2.7 Ngắt port nối tiếp**

- Ngắt port nối tiếp xảy ra khi cờ ngắt phát TI hoặc cờ ngắt thu RI được đặt lên mức 1.
- Ngắt phát xảy ra khi truyền xong 1 ký tự vừa được ghi vào thanh ghi SBUF
- Ngắt thu xảy ra khi một ký tự đã được nhận xong và đang đợi trong thanh ghi SBUF để được đọc
- Khác với ngắt timer, cờ gây ra ngắt port nối tiếp không được xóa đi bằng phần cứng khi CPU chuyển đến ngắt.

- Nguồn ngắt được xác định trong chương trình con phục vụ ngắt và phải xóa cờ tạo ngắt bằng phần mềm

**Ví dụ:**

Viết chương trình dùng ngắt port nối tiếp để thu liên tục các ký tự và cất vào 10byte ô nhớ RAM nội có địa chỉ đầu là 20H. Tốc độ baud là 1200bps, tần số thạch anh là 12MHz

```
        ORG 0000H; địa chỉ bắt đầu của chương trình
        LJMP MAIN
        ORG 0023H; Địa chỉ vectơ ngắt port nối tiếp
        LJMP ISR-SP; nhảy đếnchương trình con phục vụ ngắt port nối tiếp
        ORG 0030H ; Bắt đầu chương trình chính tại điểm ngoài tất cả các ngắt
MAIN:MOV SCON,#50H; Port nối tiếp nhận dữ liệu UART 8bit
        MOV TMOD,#20H ; Timer 1 mode 2
        MOV TH1,#-26  ; tạo tốc độ baud 1200bps
        SETB TR1 ; cho timer 1 chạy
        MOV IE,#90H ; cho phép ngắt port nối tiếp
        MOV R0, #20H; gánđịa chỉ đầu vào R0
        SJMP $ ; không làm gì cả
ISR-SP : CLR RI;xóa cờ ngắt thu
        MOV A, SBUF;đọc dữ liệu thu được vào A
        MOV @R0,A; chuyển dữ liệu vào ô nhớ ram nội
        INC R0; tăng địa chỉ ô nhớ
        CJNE R0,#2AH,EXIT; kiểm tra xem đã đủ 10byte dữ liệu hay chưa, nếu đã
đủ thì thóat
EXIT: RETI
END
```

### 6.2.8 Ngắt ngoài

- MCS-51 có 2 nguồn ngắt ngoài khác nhau: ngắt ngoài 0 và ngắt ngoài 1.
- Ngắt ngoài xảy ra khi bit IEx chuyển lên mức 1, quá trình chuyển mức của bit IEx xảy ra khi:
  - Bit ITx = 0 và xuất hiện mức logic 0 tại chân INTx tương ứng (P3.2 cho ngắt ngoài 0 hay P3.3 cho ngắt ngoài 1).
  - Bit ITx = 1 và xuất hiện cạnh âm tại chân INTx.

Khi có ngắt xảy ra và cho phép ngắt (dùng thanh ghi IE), chương trình sẽ được chuyển đến địa chỉ của ISR tương ứng (0003h cho ngắt ngoài 0 và 0013h cho ngắt ngoài 1) và xoá cờ ngắt TFx.

Lưu ý rằng các cờ ngắt được lấy mẫu trong mỗi chu kỳ nên để phát hiện ngắt, yêu cầu phải:

- Ở mức thấp tối thiểu 1 chu kỳ nếu tác động bằng mức logic (ITx = 0).
- Ở mức cao tối thiểu 1 chu kỳ trước khi chuyển xuống mức thấp và mức thấp cũng phải tồn tại tối thiểu 1 chu kỳ (ITx = 1).

**Quá trình điều khiển ngắt ngoài:**

- Xác định yêu cầu ngắt bằng cạnh âm hay bằng mức logic.

- Cho phép ngắt tại ngắt ngoài tương ứng (dùng thanh ghi IE).
- Xác định mức ưu tiên (thanh ghi IP).
- Viết ISR cho các ngắt.

**Ví dụ**: Viết chương trình sao cho mỗi khi có mức logic 0 xuất hiện tại P3.2 (ngắt ngoài 0) thì tạo xung 1 KHz tại P1.0. Quá trình tạo xung chỉ dừng khi có mức logic 0 xuất hiện tại P3.3 (ngắt ngoài 1). Biết rằng bộ dao động trên chip sử dụng thạch anh có tần số 12MHz.

**Giải**

Chương trình thực hiện có 3 ngắt xảy ra: ngắt ngoài 0 cho phép timer chạy để tạo xung tại P1.0, ngắt ngoài 1 cấm timer để ngừng quá trình tạo xung và ngắt timrer để tạo xung.

Giải:

- f = 1KHz → T = 1ms (1000 chu kỳ) → giá trị đếm là 500 ( chọn chế độ 16 bit)
- Nạp giá trị thanh ghi TMOD = 10H
- Thanh ghi IE = 8DH

Chương trình thực hiện như sau:

```
ORG 0000h
LJMP MAIN
        ORG 0003h                       ; Địa chỉ ISR ngắt ngoài 0
                SETB TR1                ; Timer 1 chạy
        RETI
        ORG 0013h                       ; Địa chỉ ISR của ngắt ngoài 1
                CLR TR1                 ; Cấm timer 1
        RETI
        ORG 001Bh                       ; Địa chỉ ISR timer 1
                MOV TH1,#HIGH(-500)     ; Chế độ 16 bit nên mỗi lần tràn
                MOV TL1,#LOW(-500)      ; phải nạp lại giá trị
                CPL P1.0                ; Đảo bit P1.0 để tạo xung
        RETI
MAIN:
MOV TMOD,#10h
MOV TH1,#HIGH(-500)
MOV TL1,#LOW(-500)
MOV IE,#8Dh                                     ; Cho phép ngắt tại ngắt ngoài 0,
1 và
SJMP $                                  ; timer 1
END
```

### 6.2.9 Bài tập áp dụng: viết chương trình điều khiển

Bài tập 1:

Hãy viết chương trình nhận liên tục dữ liệu 8 bít ở cổng P0 và gửi nó đến cổng P1 trong khi nó cùng lúc tạo ra một sóng vuông chu kỳ 200μs trên chân P2.1. Hãy sử dụng bộ Timer 0 để tạo ra sóng vuông, tần số của 8051 là XTAL = 11.0592MHz.

**Lời giải:**

Ta sử dụng bộ Timer0 ở chế độ 2 (tự động nạp lại) giá trị nạp cho TH0 là 100/1.085μs = 92.

```
            ORG  0000H
            ORG  0030H
            CPL P2.1            ;
MAIN:       TMOD, #02H          ; Chọn bộ Timer0, chế độ 2 tự nạp lại
            MOV  P0, #0FFH      ; Lấy P0 làm cổng vào nhận dữ liệu
            MOV  TH0, # - 92    ; Đặt TH0 = A4H cho – 92
            MOV  IE, #82H       ; IE = 1000 0010 cho phép Timer0
            SETB TR0            ; Khởi động bộ Timer0
BACK:       MOV  A, P0          ; Nhận dữ liệu vào từ cổng P0
            MOV  P1, A          ; Chuyển dữ liệu đến cổng P1
            SJMP BACK           ; Tiếp tục nhận và chuyển dữ liệu
            END
```

Bài tập 2:

Hãy viết lại chương trình ở bài tập 2 để tạo sóng vuông với mức cao kéo dài 1085μs và mức thấp dài 15μs với giả thiết tần số XTAL = 11.0592MHz. Hãy sử dụng bộ định thời Timer1.

**Lời giải:**

Vì 1085μs là 1000 × 1085μs nên ta cần sử dụng chế độ 1 của bộ định thời Timer1.

```
; - - Khi khởi tạo tránh sử dụng không gian dành cho bảng véc tơ ngắt.
            ORG  0000H
            LJMP MAIN           ; Chuyển đến bảng véc tơ ngắt.
;
; - - Trình ISR đối với Timer1 để tạo ra xung vuông
            ORG  001BH          ; Địa chỉ ngắt của Timer1 trong
bảng véc tơ ngắt
            LJMP ISR-T1         ; Nhảy đến ISR
;
; - - Bắt đầu các chương trình chính MAIN.
            ORG  0030H          ; Sau bảng véc tơ ngắt
MAIN:       MOV  TMOD, #10H     ; Chọn Timer1 chế độ 1
            MOV  P0, #0FFH      ; Chọn cổng P0 làm đầu vào nhận dữ liệu
            MOV  TL1, #018H     ; Đặt TL1 = 18 byte thấp của - 1000
            MOV  TH1, #0FCH     ; Đặt TH1 = FC byte cao của - 1000
            MOV  IE, #88H       ; IE = 10001000 cho phép ngắt Timer1
            SETB TR1            ; Khởi động bộ Timer1
BACK:       MOV  A, P0          ; Nhận dữ liệu đầu vào ở cổng P0
            MOV  P1, A          ; Chuyển dữ liệu đến P1
            SJMP BACK           ; Tiếp tục nhận và chuyển dữ liệu
```

```
;
; - - Trình ISR của Timer1 phải được nạp lại vì ở chế độ 1
ISR-T1:   CLR  TR1            ; Dừng bộ Timer1
          CLR  P2.1           ; P2.1 = 0 bắt đầu xung mức thấp
          MOV  R2, #4         ; 2 chu kỳ máy MC (Machine Cycle)
HERE:     DJNZ R2, HERE       ; 4 × 2 MC = 8 MC
          MOV  TL1, #18H      ; Nạp lại byte thấp giá trị   2 MC
          MOV  TH1, #0FCH     ; Nạp lại byte cao giá trị    2 MC
          SETB TR1            ; Khởi động Timer1 1 MC
          SETB P2.1           ; P2.1 = 1 bật P2.1 trở lại cao
          RETI                ; Trở về chương trình chính
          END
```

Lưu ý rằng phần xung mức thấp được tạo ra bởi 14 chu kỳ mức MC và mỗi MC = 1.085µs và 14 × 1.085µs = 15.19µs.

Bài tập 3: Viết chương trình tạo xung vuông tần số f = 10KHz tại P1.0 dùng ngắt timer 0 và xung vuông tần số f = 1 KHz tại P1.1 dùng ngắt timer 1.

**Giải**

Chương trình thực hiện như sau:

```
ORG 0000h
LJMP main
ORG 000Bh
        CPL P1.0
RETI
ORG 001Bh
        MOV TH1,#HIGH(-500) ; 2 byte
        MOV TL1,#LOW(-500) ; 2 byte
        CPL P1.1 ; 2 byte
RETI ; 1 byte
Main:
        MOV TMOD,#12h
        MOV IE,#8Ah
        SETB TR0
        SETB TR1
        MOV TH1,#HIGH(-500)
        MOV TL1,#LOW(-500)
        MOV TH0,#(-50)
        MOV TL0,#(-50)
        SJMP $
END
```

Trong ví dụ này, do timer 1 hoạt động ở chế độ 16 bit nên mỗi lần timer tràn phải thực hiện nạp lại giá trị cho timer 1.

**Bài tập 4:** Viết chương trình khởi động cổng nối tiếp ở chế độ UART 8 bit với tốc

độ truyền 4800 bps. Viết ISR cho cổng nối tiếp theo yêu cầu: truyền tuần tự các ký tự
từ 'A' đến 'Z' ra cổng nối tiếp đồng thời mỗi lần có ký tự đến cổng nối tiếp thì nhận
về và xuất ký tự nhận ra P0 (giả sử tần số thạch anh là 11.0592 MHz).
Chương trình thực hiện như sau:

```
ORG 0000h
LJMP main
ORG 0023h ; Địa chỉ ISR của cổng nối tiếp
LJMP Serial_ISR
Main:
        TMOD,#20h
        MOV TH1,#(-6)
        MOV TL1,#(-6) ; Tốc độ 4800 bps
        SETB TR1
        MOV R7,#'A' ; Ký tự truyền đầu tiên
        MOV IE,#90h ; Cho phép ngắt tại công nối tiếp
        SETB TI ;Cho phép truyền
        SJMP $
Serial_ISR:
        JNB RI, Transmit ; Nếu không phải ngắt do nhận ký tự thì truyền
        CLR RI
        MOV A,SBUF ; Nhận ký tự
        MOV P0,A ; Xuất ra Port 0
        SJMP exitSerial
Transmit:     ; Truyền ký tự
        CLR TI
        MOV A,R7
        MOV SBUF,A ; Truyền ký tự
        INC R7 ; Qua ký tự kế
        CJNE R7,#'Z'+1,exitSerial ; Nếu chưa truyền'Z' thì
        ; tiếp tục truyền, ngược lại thì
        MOV R7,#'A' ; bắt đầu truyền từ ký tự 'A'
exitSerial:
        RETI
END
```