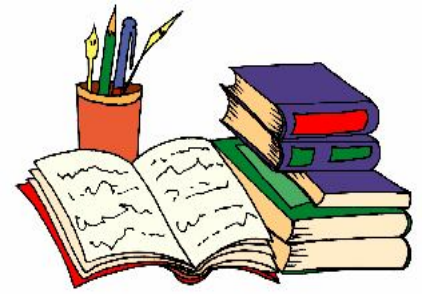


Giáo trình
Vi xử lý - Vi điều khiển

MỤC LỤC

Danh mục hình vẽ 4
Danh mục bảng biểu 6
CHƯƠNG 1 TỔNG QUAN VỀ VI XỬ LÝ – VI ĐIỀU KHIỂN 7
1.1 GIỚI THIỆU CHUNG VỀ VI XỬ LÝ – VI ĐIỀU KHIỂN 7
1.1.1. Tổng quan 7
1.1.2. Vi xử lý và vi điều khiển 8
1.1.3. Ứng dụng của Vi xử lý – vi điều khiển 10
1.2. CẤU TRÚC CHUNG CỦA MỘT HỆ VI XỬ LÝ 11
1.2.1 Khối xử lý trung tâm (CPU) 12
1.2.2. Hệ thống bus 13
1.3. ĐỊNH DẠNG DỮ LIỆU VÀ BIỂU DIỄN THÔNG TIN TRONG HỆ VI XỬ LÝ – VI ĐIỀU KHIỂN 14
1.3.1. Các hệ đếm 14
1.3.2. Mã ký tự - Alphanumeric CODE (ASCII, EBCDIC) 16
1.3.3. Các phép toán số học trên hệ đếm nhị phân 17
CHƯƠNG 2 VI ĐIỀU KHIỂN 19
2.2. ỨNG DỤNG CỦA VI ĐIỀU KHIỂN 20
2.3. HOẠT ĐỘNG CỦA VI ĐIỀU KHIỂN 20
2.4. CẤU TRÚC CHUNG CỦA VI ĐIỀU KHIỂN 21
2.4.1.Read Only Memory (ROM) 21
2.4.2. Random Access Memory (RAM) 22
2.4.3. Electrically Erasable Programmable ROM (EEPROM) 22
2.4.4. Các thanh ghi chức năng đặc biệt (SFR) 23
2.4.5. Bộ đếm chương trình (PC:Program Counter) 23
2.4.6. Central Processor Unit (CPU) 23
2.4.7. Các cổng vào/ra (I/O Ports) 23
2.4.8. Bộ dao động (Oscillator) 24
2.4.9. Bộ định thời/đếm (Timers/Counters) 25
2.4.10. Truyền thông nối tiếp 25
CHƯƠNG 3 KIẾN TRÚC VI ĐIỀU KHIỂN 8051 27
3.1. CHUẨN 8051 27
3.2. CHÂN VI ĐIỀU KHIỂN 8051 28
3.3. CỔNG VÀO/ RA 29
3.4 . TỔ CHỨC BỘ NHỚ 34
3.4.1 Tổ chức bộ nhớ trong 35
3. 4.2. Tổ chức bộ nhớ ngoài 37
3.5. CÁC THANH GHI CHỨC NĂNG ĐẶC BIỆT (SFRs - Special Function Registers) 39
3.6. BỘ ĐẾM / BỘ ĐỊNH THỜI 43
3.7. TRUYỀN THÔNG KHÔNG ĐỒNG BỘ (UART) 44
3.8. NGẮT VI ĐIỀU KHIỂN 8051 44

CHƯƠNG 4 LẬP TRÌNH HỢP NGỮ CHO 8051 45
4.1 CÁC CHẾ ĐỘ ĐỊA CHỈ 45
4.1.1. Địa chỉ tức thời 45
4.1.2. Địa chỉ theo thanh ghi 45
4.1.3. Địa chỉ trực tiếp 46
4.1.4. Địa chỉ gián tiếp. 47
4.1.5. Địa chỉ theo chỉ số 48
4.2. TẬP LỆNH TRONG 8051 48
4.2.1. Phân loại tập lệnh 48
4.2.2. Cấu trúc chung của mỗi lệnh 48
4.2.3. Các lệnh toán học 49
4.2.4. Các lệnh logic 52
4.2.5. Các lệnh vận chuyển dữ liệu 55
4.2.6. Các lệnh thao tác bit 55
4.2.7. Lệnh đọc cổng 56
4.2.8. Các lệnh điều khiển chương trình (rẽ nhánh) 56
4.3 CẤU TRÚC CHUNG CHƯƠNG TRÌNH HỢP NGỮ CHO 8051 61
4.3.1. Các thành phần cơ bản của ngôn ngữ Assembly 61
4.3.2. Khai báo trong lập trình hợp ngữ cho 8051 62
4.3.3. Cấu trúc một chương trình hợp ngữ 64
CHƯƠNG 5 BỘ ĐỊNH THỜI, BỘ ĐẾM 66
5.1. CÁC THANH GHI CƠ SỞ CỦA BỘ ĐỊNH THỜI 66
5.1.1. Các thanh ghi của bộ Timer 0. 66
5.1.2. Các thanh ghi của bộ Timer 1. 66
5.1.3. Thanh ghi TMOD (chế độ của bộ định thời). 66
5.2. CÁC CHẾ ĐỘ CỦA BỘ ĐẾM / ĐỊNH THỜI (Timer Mode) 69
5.3. NGẮT TIMER 72
CHƯƠNG 6 TRUYỀN THÔNG NỐI TIẾP 73
6.1. CÁC CƠ SỞ CỦA TRUYỀN THÔNG NỐI TIẾP 73
6.2. CÁC THANH GHI ĐIỀU KHIỂN TRUYỀN THỒN 75
6.2.1. SBUF 75
6.2.2. SCON 75
6.3. LỰA CHỌN CHẾ ĐỘ TRUYỀN THÔNG 76
6.3.1. Mode 0 76
6.3.2. Mode 1 78
6.3.3. Mode 2 78
6.3.4. Mode 3 79
6.4. MỘT SỐ VÍ DỤ VÀ BÀI TẬP 80
CHƯƠNG 7 XỬ LÝ NGẮT 82
7.1. TRÌNH PHỤC VỤ NGẮT 82
7.2. CÁC BƯỚC KHI THỰC HIỆN MỘT NGẮT 84
7.3. MỘT SỐ VÍ DỤ VÀ BÀI TẬP 85
7.4. THỨ TỰ ƯU TIÊN NGẮT 88

CHƯƠNG 8 PHỐI GHÉP 8051 VỚI THẾ GIỚI THỰC 89
8.1. PHỐI GHÉP VỚI LCD 89
8.1.1. Hoạt động của LCD. 89
8.1.2. Mô tả các chân của LCD. 89
8.1.3 Gửi các lệnh và dữ liệu đến LCD với một độ trễ. 92
8.1.4. Gửi mã lệnh hoặc dữ liệu đến LCD có kiểm tra cờ bận. 93
8.2. PHỐI GHÉP VỚI ADC. 95
8.2.1. Các thiết bị ADC. 95
8.2.2. Chíp ADC 0804. 95
8.2.3. Ghép nối 8051 với ADC 0804. 99
PHỤ LỤC 102
Phụ lục A: Các ký hiệu sử dụng mô tả lệnh 102
Phụ lục B: Chi tiết các thanh ghi chức năng trong 8051 108

Danh mục hình vẽ

Hình 1-1. Bộ vi xử lý Intel 80486DX2 7
Hình 2-1. Cấu trúc chung họ VĐK 21
Hình 2-2. Giao tiếp bộ nhớ 23
Hình 2-3. Vào ra với thiết bị ngoại vi 24
Hình 2-4. Ghép nối bộ dao động 24
Hình 2-5. Bộ định thời/đếm 25
Hình 2-6. Truyền nhận nối tiếp 25
Hình 3-1.Kiến trúc vi điều khiển 8051 27
Hình 3-2. Sơ đồ chân VĐK AT89C51 28
Hình 3-3. Cổng vào/ra 30
Hình 3-4. Xuất mức 0 31
Hình 3-5. Trở treo nội tại chân 31
Hình 3-6. xuất mức 1 31
Hình 3-7 . Sơ đồ kết nối thạch anh 34
Hình 3-8. Các vùng nhớ trong AT89C51 34
Hình 3-9. Thực thi bộ nhớ chương trình ngoài 38
Hình 3-10. Thanh ghi PSW 40
Hình 3-11. Chọn bank thanh ghi 40
Hình 3-12. Thanh ghi PCON 42
Hình 3-13 - Ghép nối RS232 với 8051 44
Hình 5-1. Các thanh ghi của bộ Timer 0 66
Hình 5-2. Các thanh ghi của bộ Timer 1 66
Hình 5-3. Timer TMOD 67
Hình 5-4. Timer 0 – Mode 0 69
Hình 5-5. Timer 0 – Mode 1 70
Hình 5-6. Timer 0 – Mode 2 70
Hình 5-7. Timer 0 – Mode 3 71
Hình 6-1. Truyền thông 73
Hình 6-2. Ghép nối RS232 với 8051 75
Hình 6-3. Thanh ghi SBUF 75
Hình 6-4. Thanh ghi SCON 75
Hình 6-5. Truyền thông nối tiếp – Mode 0 77
Hình 6-6. Giản đồ thời gian truyền nối tiếp – Mode 0 77
Hình 6-7. Giản đồ thời gian nhận nối tiếp – Mode 0 77
Hình 6-8. Truyền nhận nối tiếp – Mode 1 78
Hình 6-9. Giản đồ thời gian truyền nối tiếp – Mode 1 78
Hình 6-10. Giản đồ thời gian nhận nối tiếp – Mode 1 78
Hình 6-11. Giản đồ thời gian truyền nối tiếp – Mode 2 79
Hình 6-12. Giản đồ thời gian nhận nối tiếp – Mode 2 79
Hình 7-1. Các tín hiệu điều khiển ngắt 83
Hình 7-2. Thanh ghi điều khiển ngắt 83
Hình 7-3. Thanh ghi IP 88
Hình 8-1. Ghép Nối LCD 93

Hình 8-2. Kiểm tra ADC 0804 ở chế độ chạy tự do.....97
Hình 8-3. Phân chia thời gian đọc và ghi của ADC 804. 99
Hình 8-4. Nối ghép ADC 0804 100
Hình 8-5. Nối ghép ADC 804 với đồng hồ từ XTAL2 của 8051. 101

Danh mục bảng biểu

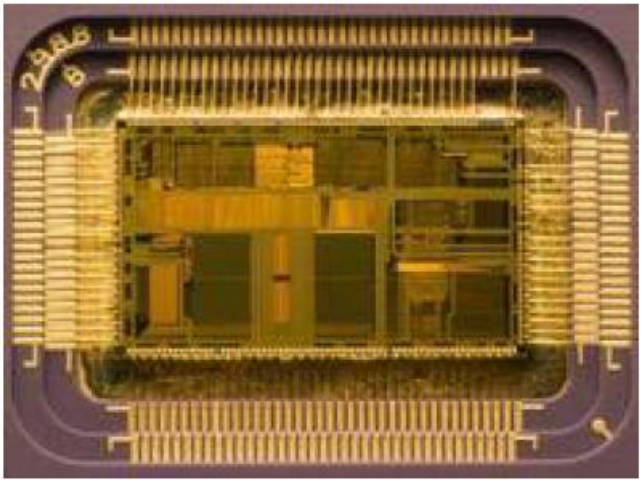
Bảng 1-1. Giá trị tương ứng giữa các hệ số 15
Bảng 1-2. Bảng mã ASCII 16
Bảng 1-3. Bảng mã ASCII có cả ký tự trong phần mở rộng 17
Bảng 1-4 Phép cộng nhị phân và phép trừ nhị phân 17
Bảng 3-1. Chức năng các chân của Port 3 33
Bảng 3-2. Các thanh ghi chức năng đặc biệt 35
Bảng 3-3. Địa chỉ RAM nội 8051 36
Bảng 4-1: Tóm tắt phép nhân hai số không dấu (MUL AB) 51
Bảng 4-2. Tóm tắt phép chia không dấu (DIV AB). 52
Bảng 4-3. Lệnh đọc cổng 56
Bảng 4-4. Lệnh đọc cổng ra 56
Bảng 4-5. Các lệnh nhảy có điều kiện 58
Bảng 4-6. Các toán tử 63
Bảng 4-7. Một số từ khóa của Assembly 64
Bảng 5-1. Chế độ hoạt động của Timer/Counter 67
Bảng 6-1. Các bít của thanh SCON 76
Bảng 6-2. Lựa chọn chế độ làm việc 76
Bảng 6-3. Một số giá trị thường dùng trong truyền thông nối tiếp 80
Bảng 7-1. Các bít của thanh ghi diều khiển ngắt 84
Bảng 7-2. Bảng vector ngắt và ví dụ 85
Bảng 8-1. Mô tả các chân của LCD. 91
Bảng 8-2. Các mã lệnh LCD. 91
Bảng 8-3. Điện áp $Vref_{/2}$ liên hệ với dải V_{in}. 98

CHƯƠNG 1
TỔNG QUAN VỀ VI XỬ LÝ – VI ĐIỀU KHIỂN

1.1 GIỚI THIỆU CHUNG VỀ VI XỬ LÝ – VI ĐIỀU KHIỂN

1.1.1. Tổng quan

Vi xử lý (viết tắt là **µP** hay **uP**), đôi khi còn được gọi là **bộ vi xử lý**, là một linh kiện điện tử được chế tạo từ các tranzito thu nhỏ tích hợp lên trên một vi mạch tích hợp hơn. Khối xử lý trung tâm (CPU) là một bộ vi xử lý được nhiều người biết đến nhưng ngoài ra nhiều thành phần khác trong máy tính cũng có bộ vi xử lý riêng của nó, ví dụ trên card màn hình (*video card*) chúng ta cũng có một bộ vi xử lý



Hình 1-1. Bộ vi xử lý Intel 80486DX2

Trước khi xuất hiện các bộ vi xử lý, các CPU được xây dựng từ các mạch tích hợp cỡ nhỏ riêng biệt, mỗi mạch tích hợp chỉ chứa khoảng vào chục tranzito. Do đó, một CPU có thể là một bảng mạch gồm hàng ngàn hay hàng triệu vi mạch tích hợp. ngày nay, công nghệ tích hợp đã phát triển, một CPU có thể tích hợp lên một hoặc vài vi mạch tích hợp cỡ lớn, mỗi vi mạch tích hợp cỡ lớn chứa hàng ngàn hoặc hàng triệu tranzito. Nhờ đó công suất tiêu thụ và giá thành của bộ vi xử lý đã giảm đáng kể.

Vi điều khiển là một máy tính được tích hợp trên một chíp, nó thường được sử dụng để điều khiển các thiết bị điện tử. Vi điều khiển, thực chất, là một hệ thống bao gồm một vi xử lý có hiệu suất đủ dùng và giá thành thấp (khác với các bộ vi xử lý đa năng dùng trong máy tính) kết hợp với các khối ngoại vi như bộ nhớ, các mô đun vào/ra, các mô đun biến đổi số sang tương tự và tương tự sang số,... Ở máy tính thì các mô đun thường được xây dựng bởi các chíp và mạch ngoài.

Vi điều khiển thường được dùng để xây dựng các hệ thống nhúng. Nó xuất hiện khá nhiều trong các dụng cụ điện tử, thiết bị điện, máy giặt, lò vi sóng, điện thoại, đầu đọc DVD, thiết bị đa phương tiện, dây chuyền tự động, v.v.

Hầu hết các vi điều khiển ngày nay được xây dựng dựa trên kiến trúc Harvard, kiến trúc này định nghĩa bốn thành phần cần thiết của một hệ thống nhúng. Những thành phần này là lõi CPU, bộ nhớ chương trình (thông thường là ROM hoặc bộ nhớ

Flash), bộ nhớ dữ liệu (RAM), một hoặc vài bộ định thời và các cổng vào/ra để giao tiếp với các thiết bị ngoại vi và các môi trường bên ngoài - tất cả các khối này được thiết kế trong một vi mạch tích hợp. Vi điều khiển khác với các bộ vi xử lý đa năng ở chỗ là nó có thể hoạt động chỉ với vài vi mạch hỗ trợ bên ngoài.

1.1.2. Vi xử lý và vi điều khiển

Khái niệm "vi xử lý" (microprocessor) và "vi điều khiển" (microcontroller).

Về cơ bản hai khái niệm này không khác nhau nhiều, "vi xử lý" là thuật ngữ chung dùng để đề cập đến kỹ thuật ứng dụng các công nghệ vi điện tử, công nghệ tích hợp và khả năng xử lý theo chương trình vào các lĩnh vực khác nhau. Vào những giai đoạn đầu trong quá trình phát triển của công nghệ vi xử lý, các chip (hay các vi xử lý) được chế tạo chỉ tích hợp những phần cứng thiết yếu như CPU cùng các mạch giao tiếp giữa CPU và các phần cứng khác. Trong giai đoạn này, các phần cứng khác (kể cả bộ nhớ) thường không được tích hợp trên chip mà phải ghép nối thêm bên ngoài. Các phần cứng này được gọi là các ngoại vi (Peripherals). Về sau, nhờ sự phát triển vượt bậc của công nghệ tích hợp, các ngoại vi cũng được tích hợp vào bên trong IC và người ta gọi các vi xử lý đã được tích hợp thêm các ngoại vi là các "vi điều khiển".

Vi xử lý có các khối chức năng cần thiết để lấy dữ liệu, xử lý dữ liệu và xuất dữ liệu ra ngoài sau khi đã xử lý. Và chức năng chính của Vi xử lý chính là xử lý dữ liệu, chẳng hạn như cộng, trừ, nhân, chia, so sánh.v.v... Vi xử lý không có khả năng giao tiếp trực tiếp với các thiết bị ngoại vi, nó chỉ có khả năng nhận và xử lý dữ liệu mà thôi.

Để vi xử lý hoạt động cần có chương trình kèm theo, các chương trình này điều khiển các mạch logic và từ đó vi xử lý xử lý các dữ liệu cần thiết theo yêu cầu. Chương trình là tập hợp các lệnh để xử lý dữ liệu thực hiện từng lệnh được lưu trữ trong bộ nhớ, công việc thực hành lệnh bao gồm: nhận lệnh từ bộ nhớ, giải mã lệnh và thực hiện lệnh sau khi đã giải mã. Để thực hiện các công việc với các thiết bị cuối cùng, chẳng hạn điều khiển động cơ, hiển thị kí tự trên màn hình .... đòi hỏi phải kết hợp vi xử lý với các mạch điện giao tiếp với bên ngoài được gọi là các thiết bị I/O (nhập/xuất) hay còn gọi là các thiết bị ngoại vi. Bản thân các vi xử lý khi đứng một mình không có nhiều hiệu quả sử dụng, nhưng khi là một phần của một máy tính, thì hiệu quả ứng dụng của Vi xử lý là rất lớn. Vi xử lý kết hợp với các thiết bị khác được sử trong các hệ thống lớn, phức tạp đòi hỏi phải xử lý một lượng lớn các phép tính phức tạp, có tốc độ nhanh. Chẳng hạn như các hệ thống sản xuất tự động trong công nghiệp, các tổng đài điện thoại, hoặc ở các robot có khả năng hoạt động phức tạp v.v...

Bộ Vi xử lý có khả năng vượt bậc so với các hệ thống khác về khả năng tính

toán, xử lý, và thay đổi chương trình linh hoạt theo mục đích người dùng, đặc biệt hiệu quả đối với các bài toán và hệ thống lớn. Tuy nhiên đối với các ứng dụng nhỏ, tầm tính toán không đòi hỏi khả năng tính toán lớn thì việc ứng dụng vi xử lý cần cân nhắc. Bởi vì hệ thống dù lớn hay nhỏ, nếu dùng vi xử lý thì cũng đòi hỏi các khối mạch điện giao tiếp phức tạp như nhau. Các khối này bao gồm bộ nhớ để chứa dữ liệu và chương trình thực hiện, các mạch điện giao tiếp ngoại vi để xuất nhập và điều khiển trở lại, các khối này cùng liên kết với vi xử lý thì mới thực hiện được công việc. Để kết nối các khối này đòi hỏi người thiết kế phải hiểu biết tinh tường về các thành phần vi xử lý, bộ nhớ, các thiết bị ngoại vi. Hệ thống được tạo ra khá phức tạp, chiếm nhiều không gian, mạch in phức tạp và vấn đề chính là trình độ người thiết kế. Kết quả là giá thành sản phẩm cuối cùng rất cao, không phù hợp để áp dụng cho các hệ thống nhỏ. Vì một số nhược điểm trên nên các nhà chế tạo tích hợp một ít bộ nhớ và một số mạch giao tiếp ngoại vi cùng với vi xử lý vào một IC duy nhất được gọi là Microcontroller-Vi điều khiển. Vi điều khiển có khả năng tương tự như khả năng của vi xử lý, nhưng cấu trúc phần cứng dành cho người dùng đơn giản hơn nhiều.

Vi điều khiển ra đời mang lại sự tiện lợi đối với người dùng, họ không cần nắm vững một khối lượng kiến thức quá lớn như người dùng vi xử lý, kết cấu mạch điện dành cho người dùng cũng trở nên đơn giản hơn nhiều và có khả năng giao tiếp trực tiếp với các thiết bị bên ngoài. Vi điều khiển tuy được xây dựng với phần cứng dành cho người sử dụng đơn giản hơn, nhưng thay vào lợi điểm này là khả năng xử lý bị giới hạn (tốc độ xử lý chậm hơn và khả năng tính toán ít hơn, dung lượng chương trình bị giới hạn). Thay vào đó, Vi điều khiển có giá thành rẻ hơn nhiều so với vi xử lý, việc sử dụng đơn giản, do đó nó được ứng dụng rộng rãi vào nhiều ứng dụng có chức năng đơn giản, không đòi hỏi tính toán phức tạp.

Vi điều khiển được ứng dụng trong các dây chuyền tự động loại nhỏ, các robot có chức năng đơn giản, trong máy giặt, ôtô v.v...

Năm 1976 Intel giới thiệu bộ vi điều khiển (microcontroller) 8748, một chip tương tự như các bộ vi xử lý và là chip đầu tiên trong họ MCS-48. Độ phức tạp, kích thước và khả năng của Vi điều khiển tăng thêm một bậc quan trọng vào năm 1980 khi intel tung ra chip 8051, bộ Vi điều khiển đầu tiên của họ MCS-51 và là chuẩn công nghệ cho nhiều họ Vi điều khiển được sản xuất sau này. Sau đó rất nhiều họ Vi điều khiển của nhiều nhà chế tạo khác nhau lần lượt được đưa ra thị trường với tính năng được cải tiến ngày càng mạnh.

Trong tài liệu này, ranh giới giữa hai khái niệm “vi xử lý” và “vi điều khiển” thực sự không cần phải phân biệt rõ ràng. Chúng tôi sẽ dùng thuật ngữ “vi xử lý” khi

đề cập đến các khái niệm cơ bản của kỹ thuật vi xử lý nói chung và sẽ dùng thuật ngữ "vi điều khiển" khi đi sâu nghiên cứu một họ chip cụ thể.

1.1.3. Ứng dụng của Vi xử lý – vi điều khiển

Vi xử lý, chính là chip của các loại máy tính ngày nay, nên hẳn các bạn đã biết rất rõ nó có những ứng dụng gì. Ở đây, tôi chỉ nói đên ứng dụng của vi điều khiển. Vi điều khiển có thể dùng trong thiết kế các loại máy tính nhúng. Máy tính nhúng có trong hầu hết các thiết bị tự động, thông minh ngày nay. Chúng ta có thể dùng vi điều khiển để thiết kế bộ điều khiển cho các sản phẩm như:

Trong các sản phẩm dân dụng:

Nhà thông minh:

Cửa tự động

Khóa số

Tự động điều tiết ánh sáng thông minh (bật/tắt đèn theo thời gian, theo cường độ ánh sáng,...)

Điều khiển các thiết bị từ xa (qua điều khiển, qua tiếng vỗ tay,...)

Điều tiết hơi ẩm, điều tiết nhiệt độ, điều tiết không khí, gió

Hệ thống vệ sinh thông minh,...

Trong quảng cáo:

Các loại biển quảng cáo nháy chữ

Quảng cáo ma trận LED (một màu, 3 màu, đa màu)

Điều khiển máy cuốn bạt quảng cáo,...

Các máy móc dân dụng

Máy điều tiết độ ẩm cho vườn cây

Buồng ấp trứng gà/vịt

Đồng hồ số, đồng hồ số có điều khiển theo thời gian

Các sản phẩm giải trí

Máy nghe nhạc

Máy chơi game

Đầu thu kỹ thuật số, đầu thu set-top-box,...

Trong các thiết bị y tế:

Máy móc thiết bị hỗ trợ: máy đo nhịp tim, máy đo đường huyết, máy đo huyết áp, điện tim đồ, điện não đồ,…

Máy cắt/mài kính

Máy chụp chiếu (city, X-quang,...)

Các sản phẩm công nghiệp

Điều khiển động cơ

Điều khiển số (PID, mờ,...)

Đo lường (đo điện áp, đo dòng điện, áp suất, nhiệt độ,...)

Cân băng tải, cân toa xe, cân ô tô,...

Máy cán thép: điều khiển động cơ máy cán, điều khiển máy quấn thép,..

Làm bộ điều khiển trung tâm cho RoBot

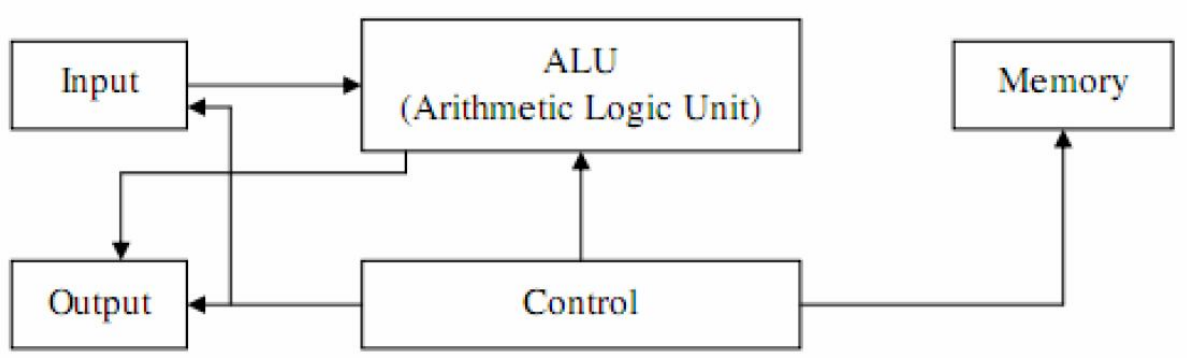
Ổn định tốc độ động cơ

Đếm sản phẩm của 1 nhà máy, xí nghiệp,…

Máy vận hành tự động (dạng CNC)

1.2. CẤU TRÚC CHUNG CỦA MỘT HỆ VI XỬ LÝ

Sơ đồ khối một máy tính cổ điển



Hình 1-2. Sơ đồ khối một máy tính cổ điển

- ALU (đơn vị logic số học): thực hiện các bài toán cho máy tính bao gồm: +, *, /,-, phép toán logic, …
- Control (điều khiển): điều khiển, kiểm soát các đường dữ liệu giữa các thành phần của máy tính.
- Memory (bộ nhớ): lưu trữ chương trình hay các kết quả trung gian.
- Input (nhập), Output (Xuất): xuất nhập dữ liệu (còn gọi là thiết bị ngoại vi).

Về cơ bản kiến trúc của một vi xử lý gồm những phần cứng sau:

Đơn vị xử lý trung tâm CPU (Central Processing Unit).

Các bộ nhớ (Memories).

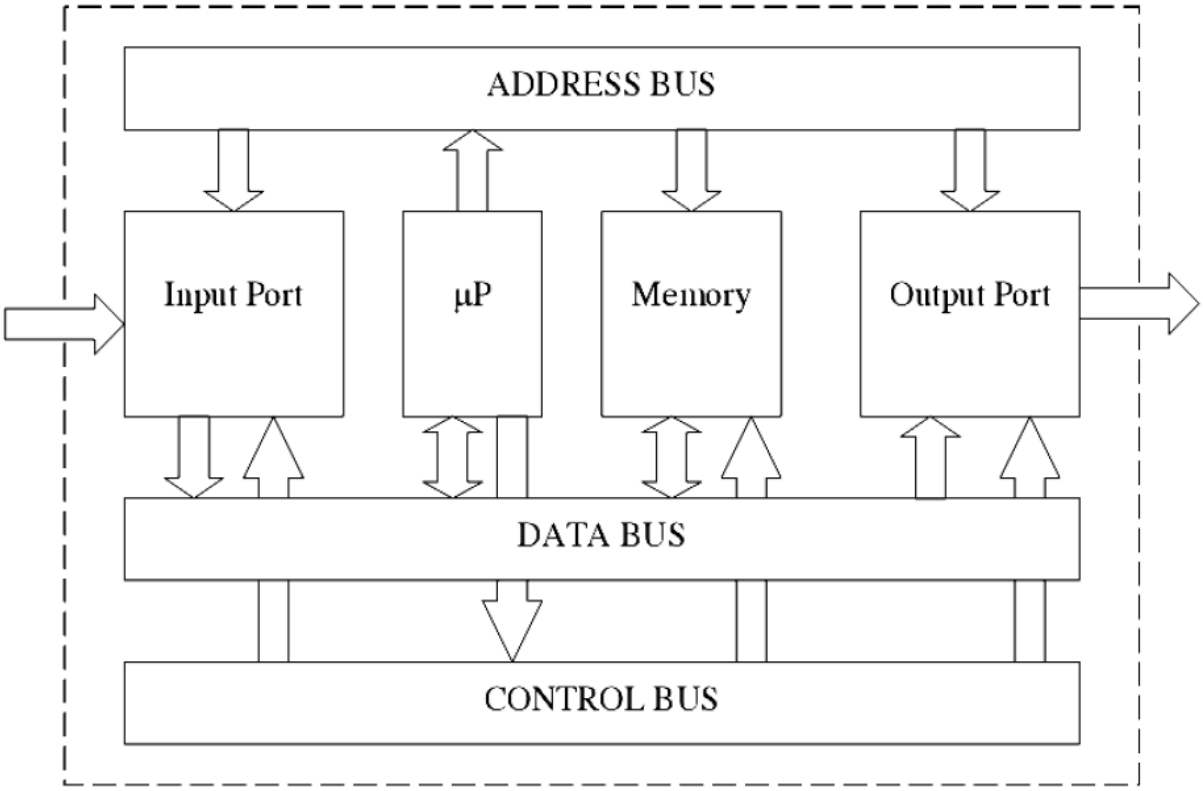
Các cổng vào/ra (song song (Parallel I/O Ports), nối tiếp (Serial I/O Ports))

Các bộ đếm/bộ định thời (Timers).

Hệ thống BUS (Địa chỉ, dữ liệu, điều khiển)

Ngoài ra với mỗi loại vi điều khiển cụ thể còn có thể có thêm một số phần cứng khác như bộ biến đổi tương tự-số ADC, bộ biến đổi số-tương tự DAC, các mạch điều

chế dạng sóng WG, điều chế độ rộng xung PWM…Bộ não của mỗi vi xử lý chính là CPU, các phần cứng khác chỉ là các cơ quan chấp hành dưới quyền của CPU. Mỗi cơ quan này đều có một cơ chế hoạt động nhất định mà CPU phải tuân theo khi giao tiếp với chúng.

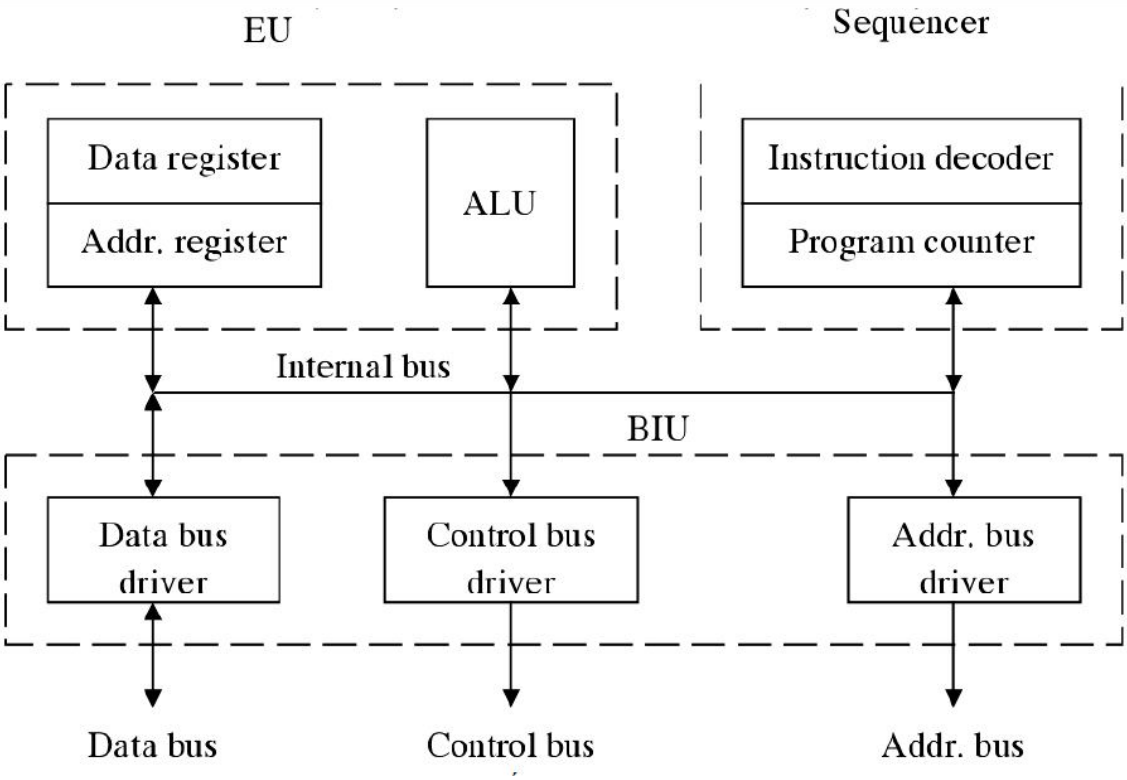


Hình 1-3. Sơ đồ khối hệ vi xử lý

Để có thể giao tiếp và điều khiển các cơ quan chấp hành (các ngoại vi), CPU sử dụng 03 loại tín hiệu cơ bản là tín hiệu địa chỉ (Address), tín hiệu dữ liệu (Data) và tín hiệu điều khiển (Control). Về mặt vật lý thì các tín hiệu này là các đường nhỏ dẫn điện nối từ CPU đến các ngoại vi hoặc thậm chí là giữa các ngoại vi với nhau. Tập hợp các đường tín hiệu có cùng chức năng gọi là các bus. Như vậy ta có các bus địa chỉ, bus dữ liệu và bus điều khiển.

1.2.1 Khối xử lý trung tâm (CPU)

CPU có cấu tạo gồm có đơn vị xử lý số học và lôgic (ALU), các thanh ghi, các khối lôgic và các mạch giao tiếp. Chức năng của CPU là tiến hành các thao tác tính toán xử lý, đưa ra các tín hiệu địa chỉ, dữ liệu và điều khiển nhằm thực hiện một nhiệm vụ nào đó do người lập trình đưa ra thông qua các lệnh (Instructions).



Hình 1-4. Khối xử lý trung tâm

1.2.2. Hệ thống bus

Là các đường tín hiệu song song 1 chiều nối từ CPU đến bộ nhớ, bao gồm:

- Bus địa chỉ
- Address bus

Độ rộng bus: là số các đường tín hiệu, có thể là 8, 18, 20, 24, 32 hay 64.

CPU gửi giá trị địa chỉ của ô nhớ cần truy nhập (đọc/ghi) trên các đường tín hiệu này.

1 CPU với n đường địa chỉ sẽ có thể địa chỉ hoá được 2n ô nhớ. Ví dụ, 1 Cpu có 16 đường địa chỉ có thể địa chỉ hoá được 216 hay 65,536 (64K) ô nhớ.

a. Bus dữ liệu - Data bus

Là các đường tín hiệu song song 2 chiều, nhiều thiết bị khác nhau có thể được nối với bus dữ liệu; nhưng tại một thời điểm, chỉ có 1 thiết bị duy nhất có thể được phép đưa dữ liệu lên bus dữ liệu.

Độ rộng Bus: 4, 8, 16, 32 hay 64 bits

Bất kỳ thiết bị nào được kết nối đến bus dữ liệu phải có đầu ra ở dạng 3 trạng thái, sao cho nó có thể ở trạng thái treo (trở kháng cao) nếu không được sử dụng.

b. Bus điều khiển - Control bus

Bao gồm 4 đến 10 đường tín hiệu song song.

CPU gửi tín hiệu ra bus điều khiển để cho phép các đầu ra của ô nhớ hay các

cổng I/O đã được địa chỉ hoá. Các tín hiệu điều khiển thường là: đọc/ ghi bộ nhớ - memory read, memory write, đọc/ ghi cổng vào/ra - I/O read, I/O write.

Ví dụ, để đọc 1 byte dữ liệu từ ô nhớ sẽ cần đến các hoạt động sau:

CPU đưa ra địa chỉ của ô nhớ cần đọc lên bus địa chỉ.

CPU đưa ra tín hiệu đọc bộ nhớ - Memory Read trên bus điều khiển.

Tín hiệu điều khiển này sẽ cho phép thiết bị nhớ đã được địa chỉ hoá đưa byte dữ liệu lên bus dữ liệu. Byte dữ liệu từ ô nhớ sẽ được truyền tải qua bus dữ liệu đến CPU.

1.3. ĐỊNH DẠNG DỮ LIỆU VÀ BIỂU DIỄN THÔNG TIN TRONG HỆ VI XỬ LÝ – VI ĐIỀU KHIỂN

1.3.1. Các hệ đếm

- Hệ thập phân - Decimal
- Hệ nhị phân - Binary
- Hệ16 - Hexadecimal
- Mã BCD (standard BCD, gray code): (Binary Coded Decimal)

Trong thực tế, đối với một số ứng dụng như đếm tần, đo điện áp, … ngõ ra ở dạng số thập phân, ta dùng mã BCD. Mã BCD dùng 4 bit nhị phân để mã hoá cho một số thập phân 0..9. Như vậy, các số hex A..F không tồn tại trong mã BCD.

Mã BCD gồm có 2 loại:

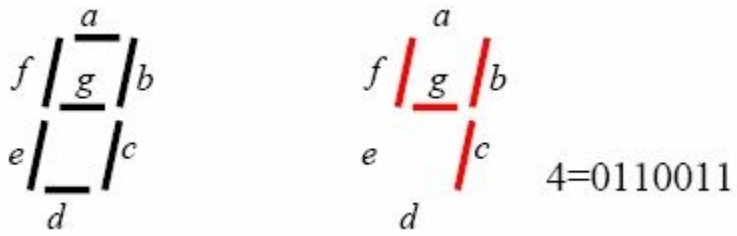
- Mã BCD không nén (unpacked): biểu diễn một số BCD bằng 8 bit nhị phân
- Mã BCD nén (packed): biểu diễn một số BCD bằng 4 bit nhị phân

VD: Số thập phân 5 2 9

Số BCD không nén 0000 0101b 0000 0010b 0000 1001b

Số BCD nén 0101b 0010b 1001b

- Mã hiển thị 7 đoạn (7-segment display code)



Hình 1-5.LED 7 thanh và cách mã hóa

- **Các mã hệ đếm thông dụng**

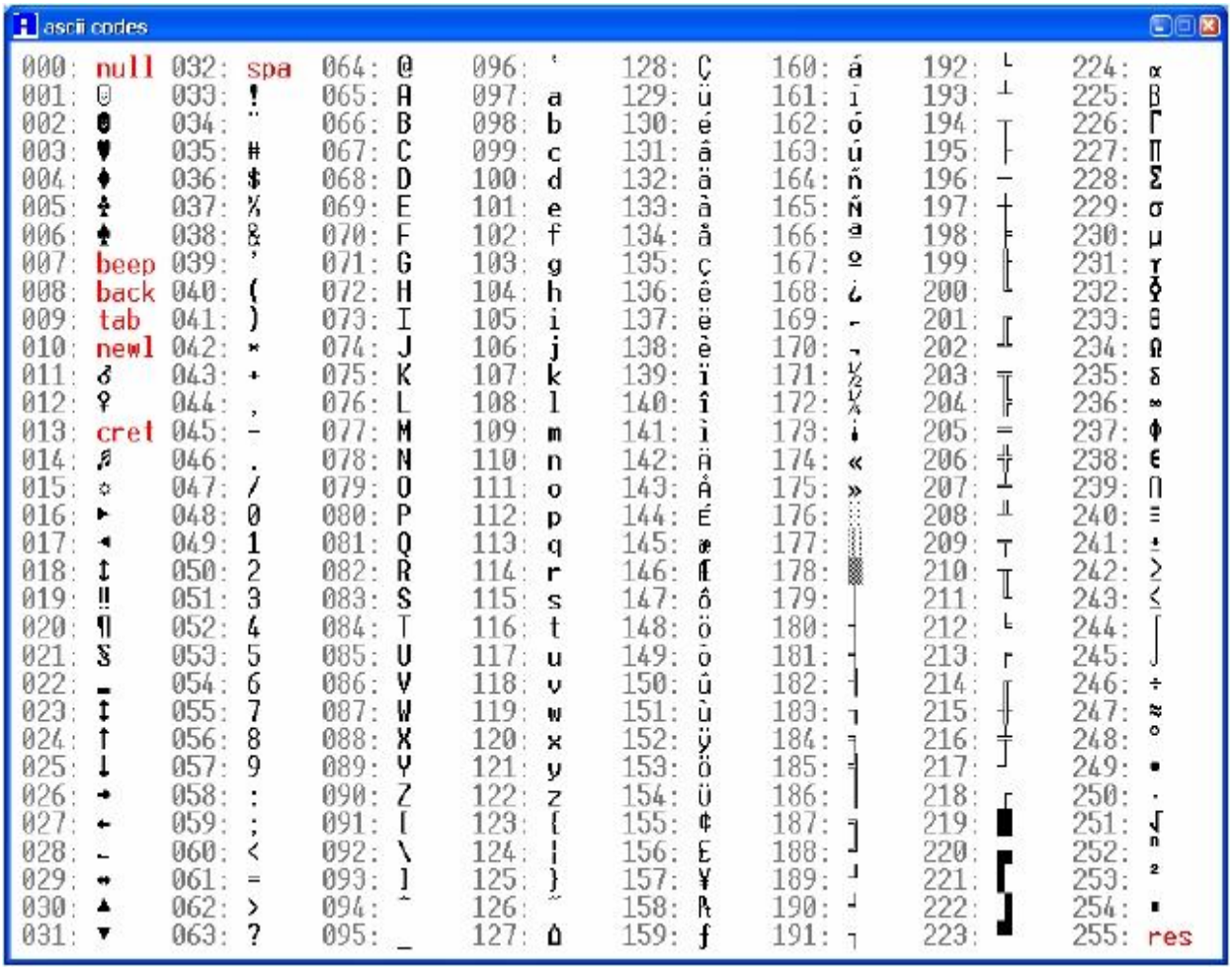
Hệ 10	Hệ 2	Hệ 8	Hệ 16	Binary-Coded Decimal		Gray Code	7-Segment	
				8421 BCD	EXCESS-3		abcdefg	Display
0	0000	0	0	0000	0011 0011	0000	111111	0
1	0001	1	1	0001	0011 0100	0001	011000	1
2	0010	2	2	0010	0011 0101	0011	110110	2
3	0011	3	3	0011	0011 0110	0010	111100	3
4	0100	4	4	0100	0011 0111	0110	011001	4
5	0101	5	5	0101	0011 1000	0111	101101	5
6	0110	6	6	0110	0011 1001	0101	101111	6
7	0111	7	7	0111	0011 1010	0100	111000	7
8	1000	10	8	1000	0011 1011	1100	111111	8
9	1001	11	9	1001	0011 1100	1101	111001	9
10	1010	12	A	0001 0000	0100 0011	1111	111110	A
11	1011	13	B	0001 0001	0100 0100	1110	001111	B
12	1100	14	C	0001 0010	0100 0101	1010	000110	C
13	1101	15	D	0001 0011	0100 0110	1011	011110	D
14	1110	16	E	0001 0100	0100 0111	1001	110111	E
15	1111	17	F	0001 0101	0100 1000	1000	100011	F

Bảng 1-1. Giá trị tương ứng giữa các hệ số

1.3.2. Mã ký tự - Alphanumeric CODE (ASCII, EBCDIC)

HEX	DEC	CHR	Ctrl	HEX	DEC	CHR	HEX	DEC	CHR	HEX	DEC	CHR
0	0	NUL	^@	20	32	SP	40	64	@	60	96	`
1	1	SOH	^A	21	33	!	41	65	A	61	97	a
2	2	STX	^B	22	34	"	42	66	B	62	98	b
3	3	ETX	^C	23	35	#	43	67	C	63	99	c
4	4	EOT	^D	24	36	$	44	68	D	64	100	d
5	5	ENQ	^E	25	37	%	45	69	E	65	101	e
6	6	ACK	^F	26	38	&	46	70	F	66	102	f
7	7	BEL	^G	27	39	'	47	71	G	67	103	g
8	8	BS	^H	28	40	(	48	72	H	68	104	h
9	9	HT	^I	29	41	)	49	73	I	69	105	i
0A	10	LF	^J	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	^K	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	^L	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	^M	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	^N	2E	46	.	4E	78	N	6E	100	n
0F	15	SI	^O	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	^P	30	48	0	50	80	P	70	112	p
11	17	DC1	^Q	31	49	1	51	81	Q	71	113	q
12	18	DC2	^R	32	50	2	52	82	R	72	114	r
13	19	DC3	^S	33	51	3	53	83	S	73	115	s
14	20	DC4	^T	34	52	4	54	84	T	74	116	t
15	21	NAK	^U	35	53	5	55	85	U	75	117	u
16	22	SYN	^V	36	54	6	56	86	V	76	118	v
17	23	ETB	^W	37	55	7	57	87	W	77	119	w
18	24	CAN	^X	38	56	8	58	88	X	78	120	x
19	25	EM	^Y	39	57	9	59	89	Y	79	121	y
1A	26	SUB	^Z	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC		3B	59	;	5B	91	[	7B	123	{
1C	28	FS		3C	60	<	5C	92	\	7C	124	|
1D	29	GS		3D	61	=	5D	93	]	7D	125	}
1E	30	RS		3E	62	>	5E	94	^	7E	126	~
1F	31	US		3F	63	?	5F	95	_	7F	127	DEL

Bảng 1-2. Bảng mã ASCII



Bảng 1-3. Bảng mã ASCII có cả ký tự trong phần mở rộng

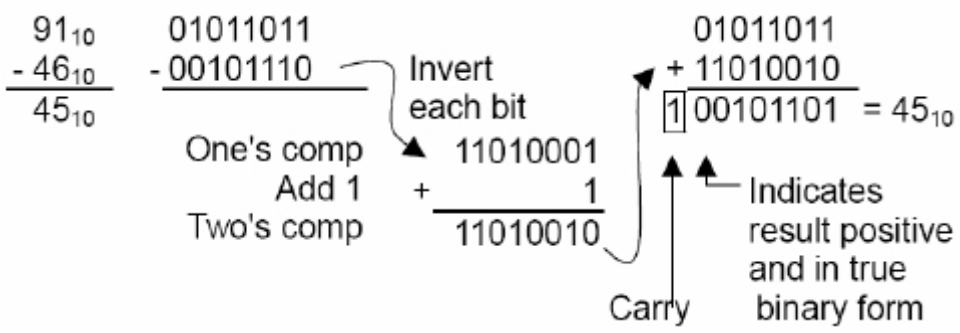
1.3.3. Các phép toán số học trên hệ đếm nhị phân

Vào			Ra	
A	B	BIN	D	BOUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

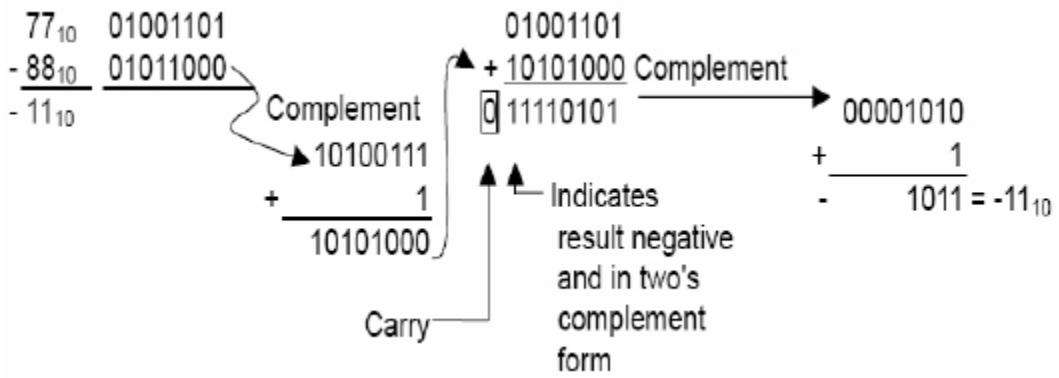
Vào			Ra	
A	B	BIN	D	BOUT
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Bảng 1-4 Phép cộng nhị phân và phép trừ nhị phân

Phép trừ nhị phân, chính là phép cộng nhị phân với số bù 2 của số trừ, trường hợp kết quả dương:



Trường hợp kết quả âm:



Phép nhân, phép chia, đề nghị sinh viên tự nghiên cứu.

CHƯƠNG 2
VI ĐIỀU KHIỂN

2.1. ĐẶT VẤN ĐỀ

Bộ Vi xử lý có khả năng vượt bậc so với các hệ thống khác về khả năng tính toán, xử lý, và thay đổi chương trình linh hoạt theo mục đích người dùng, đặc biệt hiệu quả đối với các bài toán và hệ thống lớn.Tuy nhiên đối với các ứng dụng nhỏ, tầm tính toán không đòi hỏi khả năng tính toán lớn thì việc ứng dụng vi xử lý cần cân nhắc. Bởi vì hệ thống dù lớn hay nhỏ, nếu dùng vi xử lý thì cũng đòi hỏi các khối mạch điện giao tiếp phức tạp như nhau. Các khối này bao gồm bộ nhớ để chứa dữ liệu và chương trình thực hiện, các mạch điện giao tiếp ngoại vi để xuất nhập và điều khiển trở lại, các khối này cùng liên kết với vi xử lý thì mới thực hiện được công việc. Để kết nối các khối này đòi hỏi người thiết kế phải hiểu biết tinh tường về các thành phần vi xử lý, bộ nhớ, các thiết bị ngoại vi. Hệ thống được tạo ra khá phức tạp, chiếm nhiều không gian, mạch in phức tạp và vấn đề chính là trình độ người thiết kế. Kết quả là giá thành sản phẩm cuối cùng rất cao, không phù hợp để áp dụng cho các hệ thống nhỏ.

Vì một số nhược điểm trên nên các nhà chế tạo tích hợp một ít bộ nhớ và một số mạch giao tiếp ngoại vi cùng với vi xử lý vào một IC duy nhất được gọi là Microcontroller-Vi điều khiển. Vi điều khiển có khả năng tương tự như khả năng của vi xử lý, nhưng cấu trúc phần cứng dành cho người dùng đơn giản hơn nhiều. Vi điều khiển ra đời mang lại sự tiện lợi đối với người dùng, họ không cần nắm vững một khối lượng kiến thức quá lớn như người dùng vi xử lý, kết cấu mạch điện dành cho người dùng cũng trở nên đơn giản hơn nhiều và có khả năng giao tiếp trực tiếp với các thiết bị bên ngoài. Vi điều khiển tuy được xây dựng với phần cứng dành cho người sử dụng đơn giản hơn, nhưng thay vào lợi điểm này là khả năng xử lý bị giới hạn (tốc độ xử lý chậm hơn và khả năng tính toán ít hơn,dung lượng chương trình bị giới hạn). Thay vào đó, Vi điều khiển có giá thành rẻ hơn nhiều so với vi xử lý, việc sử dụng đơn giản, do đó nó được ứng dụng rộng rãi vào nhiều ứng dụng có chức năng đơn giản, không đòi hỏi tính toán phức tạp.

Vi điều khiển được ứng dụng trong các dây chuyền tự động loại nhỏ, các robot có chức năng đơn giản, trong máy giặt, ôtô v.v...

Năm 1976 Intel giới thiệu bộ vi điều khiển (microcontroller) 8748, một chip tương tự như các bộ vi xử lý và là chip đầu tiên trong họ MCS-48. Độ phức tạp, kích thước và khả năng của Vi điều khiển tăng thêm một bậc quan trọng vào năm 1980 khi intel tung ra chip 8051, bộ Vi điều khiển đầu tiên của họ MCS-51 và là chuẩn công nghệ cho nhiều họ Vi điều khiển được sản xuất sau này. Sau đó rất nhiều họ Vi điều

khiển của nhiều nhà chế tạo khác nhau lần lượt được đưa ra thị trường với tính năng được cải tiến ngày càng mạnh.

2.2. ỨNG DỤNG CỦA VI ĐIỀU KHIỂN

Về cơ bản, vi điều khiển rất đơn giản. Chúng chỉ bao gồm tối thiểu một số thành phần sau:

- Một bộ vi xử lý tối giản được sử dụng như bộ não của hệ thống
- Tùy theo công nghệ của mỗi hãng sản xuất, có thể có thêm bộ nhớ, các chân nhập/xuất tín hiệu, bộ đếm, bộ định thời, các bộ chuyển đổi tương tự/số (A/D), …
- Tất cả chúng được đặt trong một vỏ chíp tiêu chuẩn.

Dựa trên nguyên tắc cơ bản trên, rất nhiều họ vi điều khiển đã được phát triển và ứng dụng một cách thầm lặng nhưng mạnh mẽ vào mọi mặt của đời sống của con người. Một số ứng dụng cơ bản thành công có thể kể ra sau đây:

- Những thành phần điện tử được nhúng vào vi điều khiển có thể trực tiếp hoặc qua các thiết bị vào ra (công tắc, nút bấm, cảm biến, LCD, rơ le, …) điều khiển rất nhiều thiết bị và hệ thống như thiết bị tự động trong công nghiệp, điều khiển nhiệt độ, dòng điện, động cơ, …
- Giá thành rất thấp khiến cho chúng được nhúng vào rất nhiều thiết bị thông minh trong đời sống con người như ti vi, máy giặt, điều hòa nhiệt độ, máy nghe nhạc,…

2.3. HOẠT ĐỘNG CỦA VI ĐIỀU KHIỂN

Mặc dù đã có rất nhiều họ vi điều khiển được phát triển cũng như nhiều chương trình điều khiển tạo ra cho chúng, nhưng tất cả chúng vẫn có một số điểm chung cơ bản. Do đó nếu ta hiểu cặn kẽ một họ thì việc tìm hiểu thêm một họ vi điều khiển mới là hoàn toàn đơn giản. Một kịch bản chung cho hoạt động của một vi điều khiển như sau:

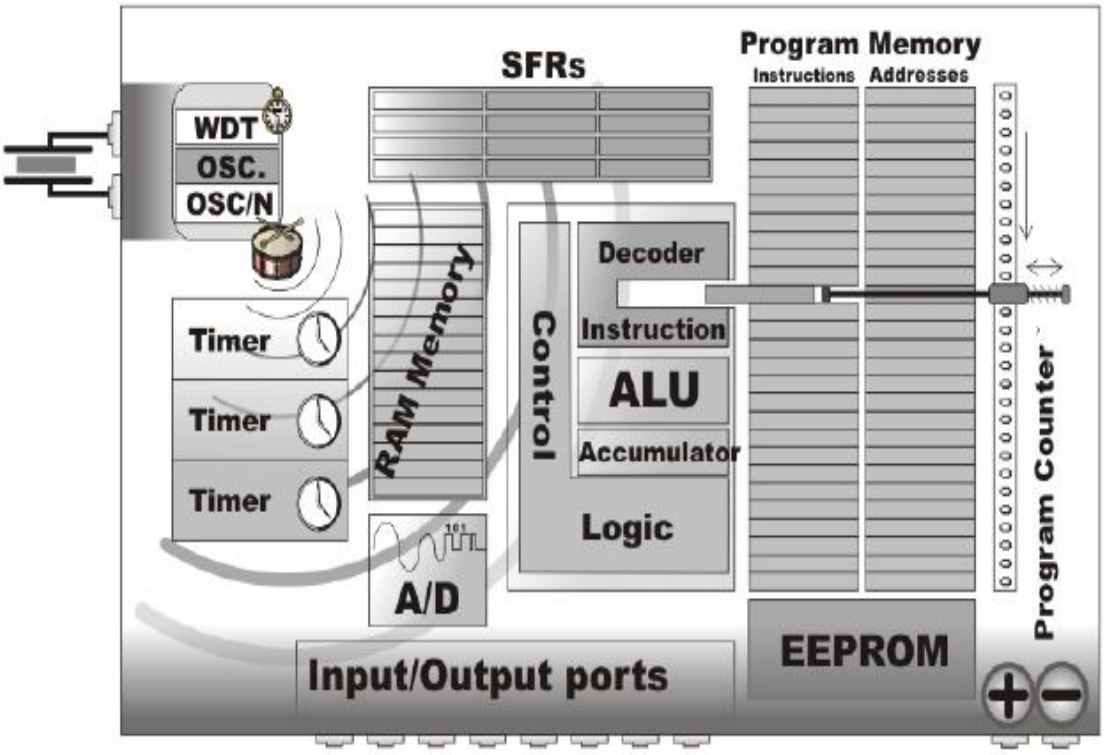
1. Khi không có nguồn điện cung cấp, vi điều khiển chỉ là một con chip có chương trình nạp sẵn vào trong đó và không có hoạt động gì xảy ra.

2. Khi có nguồn điện, mọi hoạt động bắt đầu được xảy ra với tốc độ cao. Đơn vị điều khiển logic có nhiệm vụ điều khiển tất cả mọi hoạt động. Nó khóa tất cả các mạch khác, trừ mạch giao động thạch anh. Sau mini giây đầu tiên tất cả đã sẵn sàng hoạt động.

3. Điện áp nguồn nuôi đạt đến giá trị tối đa của nó và tần số giao động trở nên ổn định. Các bit của các thanh ghi SFR cho biết trạng thái của tất cả các mạch trong vi điều khiển. Toàn bộ vi điều khiển hoạt động theo chu kỳ của chuỗi xung chính.

4. Thanh ghi bộ đếm chương trình (Program Counter) được xóa về 0. Câu lệnh từ địa chỉ này được gửi tới bộ giải mã lệnh sau đó được thực thi ngay lập tức.

5. Giá trị trong thanh ghi PC được tăng lên 1 và toàn bộ quá trình được lặp lại vài triệu lần trong một giây.



Hình 2-1. Cấu trúc chung họ VĐK

2.4. CẤU TRÚC CHUNG CỦA VI ĐIỀU KHIỂN

Như ta thấy, tất cả các hoạt động trong các vi điều khiển được thực hiện ở tốc độ cao và khá đơn giản, nhưng vi điều khiển chính nó sẽ không được thật sự hữu ích nếu không có mạch đặc biệt làm cho nó hoàn thiện. Có một số mạch cụ thể sau đây.

2.4.1.Read Only Memory (ROM)

Read Only Memory (ROM) là một loại bộ nhớ được sử dụng để lưu vĩnh viễn các chương trình được thực thi. Kích cỡ của chương trình có thể được viết phụ thuộc vào kích cỡ của bộ nhớ này. ROM có thể được tích hợp trong vi điều khiển hay thêm vào như là một chip gắn bên ngoài, tùy thuộc vào loại vi điều khiển. Cả hai tùy chọn có một số nhược điểm. Nếu ROM được thêm vào như là một chip bên ngoài, các vi điều khiển là rẻ hơn và các chương trình có thể tồn tại lâu hơn đáng kể. Nhưng đồng thời, làm giảm số lượng các chân vào/ra để vi điều khiển sử dụng với mục đích khác.

ROM nội thường là nhỏ hơn và đắt tiền hơn, nhưng lá ghim thêm có sẵn để kết

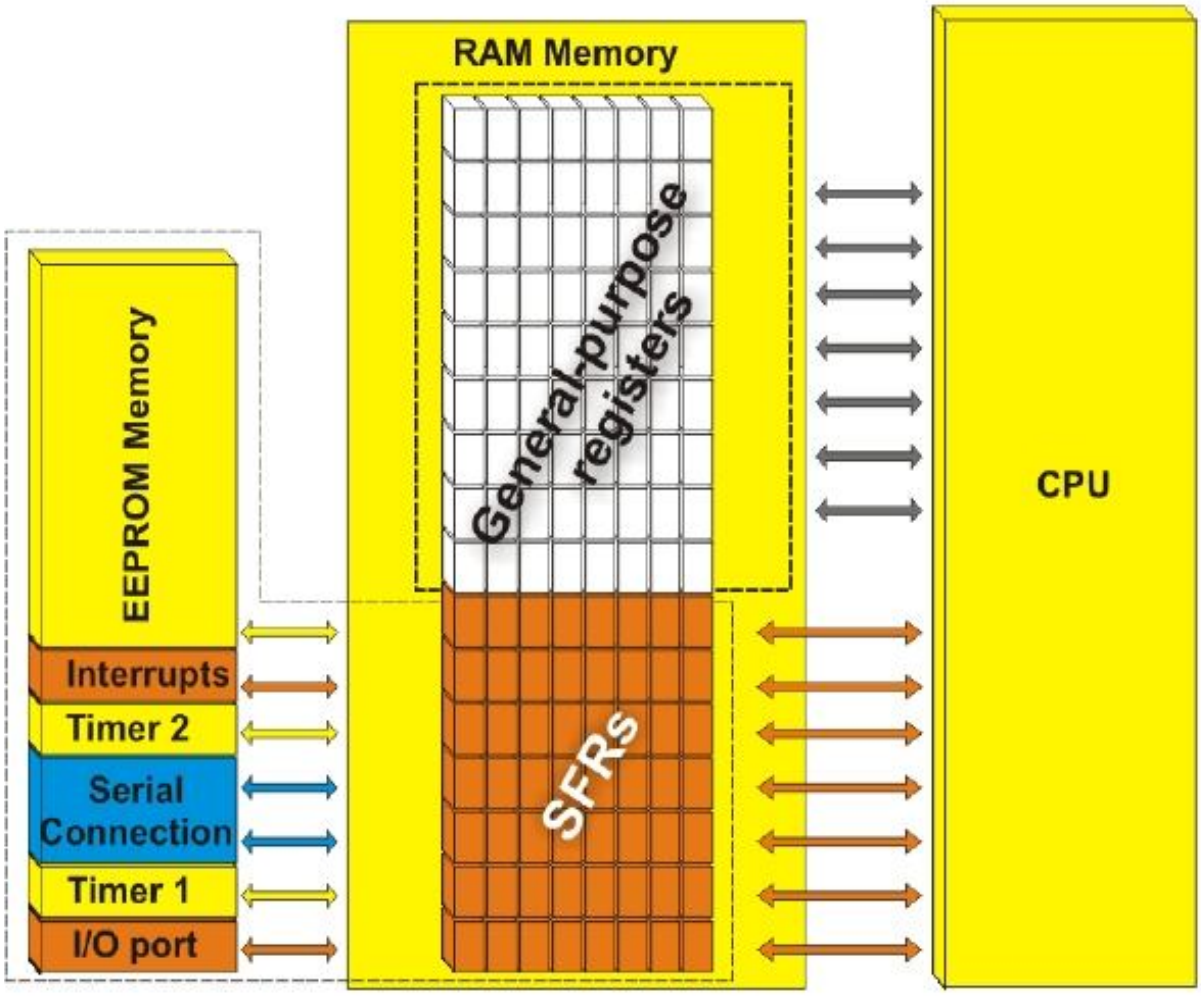
nối với môi trường ngoại vi. Kích thước của dãy ROM từ 512B đến 64KB

2.4.2. Random Access Memory (RAM)

Random Access Memory (RAM) là một loại bộ nhớ sử dụng cho các dữ liệu lưu trữ tạm thời và kết quả trung gian được tạo ra và được sử dụng trong quá trình hoạt động của bộ vi điều khiển. Nội dung của bộ nhớ này bị xóa một khi nguồn cung cấp bị tắt.

2.4.3. Electrically Erasable Programmable ROM (EEPROM)

EEPROM là một kiểu đặc biệt của bộ nhớ chỉ có ở một số loại vi điều khiển. Nội dung của nó có thể được thay đổi trong quá trình thực hiện chương trình (tương tự như RAM), nhưng vẫn còn lưu giữ vĩnh viễn, ngay cả sau khi mất điện (tương tự như ROM). Nó thường được dùng để lưu trữ các giá trị được tạo ra và được sử dụng trong quá trình hoạt động (như các giá trị hiệu chuẩn, mã, các giá trị để đếm, v.v..), mà cần phải được lưu sau khi nguồn cung cấp ngắt. Một bất lợi của bộ nhớ này là quá trình ghi vào là tương đối chậm.



Hình 2-2. Giao tiếp bộ nhớ

2.4.4. Các thanh ghi chức năng đặc biệt (SFR)

Thanh ghi chức năng đặc biệt (Special Function Registers) là một phần của bộ nhớ RAM. Mục đích của chúng được định trước bởi nhà sản xuất và không thể thay đổi được. Các bit của chúng được liên kết vật lý tới các mạch trong vi điều khiển như bộ chuyển đổi A/D, modul truyền thông nối tiếp,… Mỗi sự thay đổi trạng thái của các bit sẽ tác động tới hoạt động của vi điều khiển hoặc các vi mạch.

2.4.5. Bộ đếm chương trình (PC:Program Counter)

Bộ đếm chương trình chứa địa chỉ chỉ đến ô nhớ chứa câu lệnh tiếp theo sẽ được kích hoạt. Sau mỗi khi thực hiện lệnh, giá trị của bộ đếm được tăng lên 1. Vì lý do đó nên chương trình chỉ thực hiện được được từng lệnh trong một thời điểm.

2.4.6. Central Processor Unit (CPU)

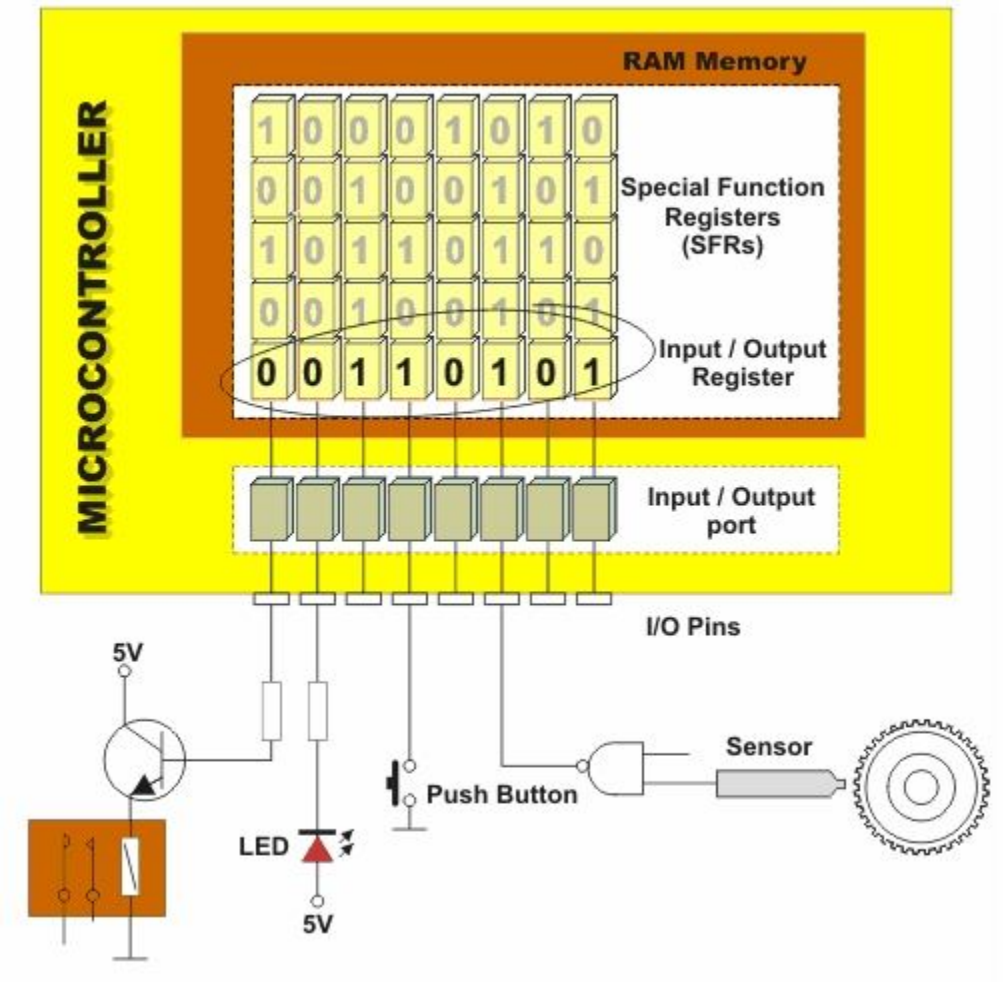
Đây là một đơn vị có nhiệm vụ điều khiển và giám sát tất cả các hoạt động bên trong vi điều khiển và người sử dụng không thể tác động vào hoạt động của nó. Nó bao gồm một số đơn vị con nhỏ hơn, trong đó quan trọng nhất là:

Bộ giải mã lệnh có nhiệm vụ nhận dạng câu lệnh và điều khiển các mạch khác theo lệnh đã giải mã. Việc giải mã được thực hiện nhờ có tập lệnh "instruction set". Mỗi họ vi điều khiển thường có các tập lệnh khác nhau.
Arithmetical Logical Unit (ALU) Thực thi tất cả các thao tác tính toán số học và logic.

Thanh ghi tích lũy (Accumulator) là một thanh ghi SFR liên quan mật thiết với hoạt động của ALU. Nó lưu trữ tất cả các dữ liệu cho quá trình tính toán và lưu giá trị kết quả để chuẩn bị cho các tính toán tiếp theo. Một trong các thanh ghi SFR khác được gọi là thanh ghi trạng thái (Status Register) cho biết trạng thái của các giá trị lưu trong thanh ghi tích lũy.

2.4.7. Các cổng vào/ra (I/O Ports)

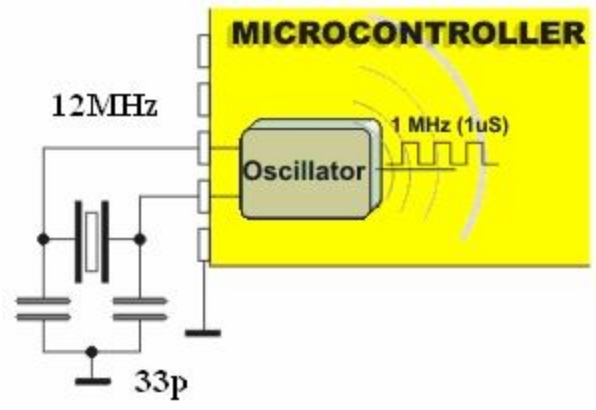
Để vi điều khiển có thể hoạt động hữu ích, nó cần có sự kết nối với các thiết bị ngoại vi. Mỗi vi điều khiển sẽ có một hoặc một số thanh ghi (được gọi là cổng) được kết nối với các chân của vi điều khiển.



Hình 2-3. Vào ra với thiết bị ngoại vi

Chúng được gọi là cổng vào/ra (I/O port) bởi vì chúng có thể thay đổi chức năng, chiều vào/ra theo yêu cầu của người dùng.

2.4.8. Bộ dao động (Oscillator)

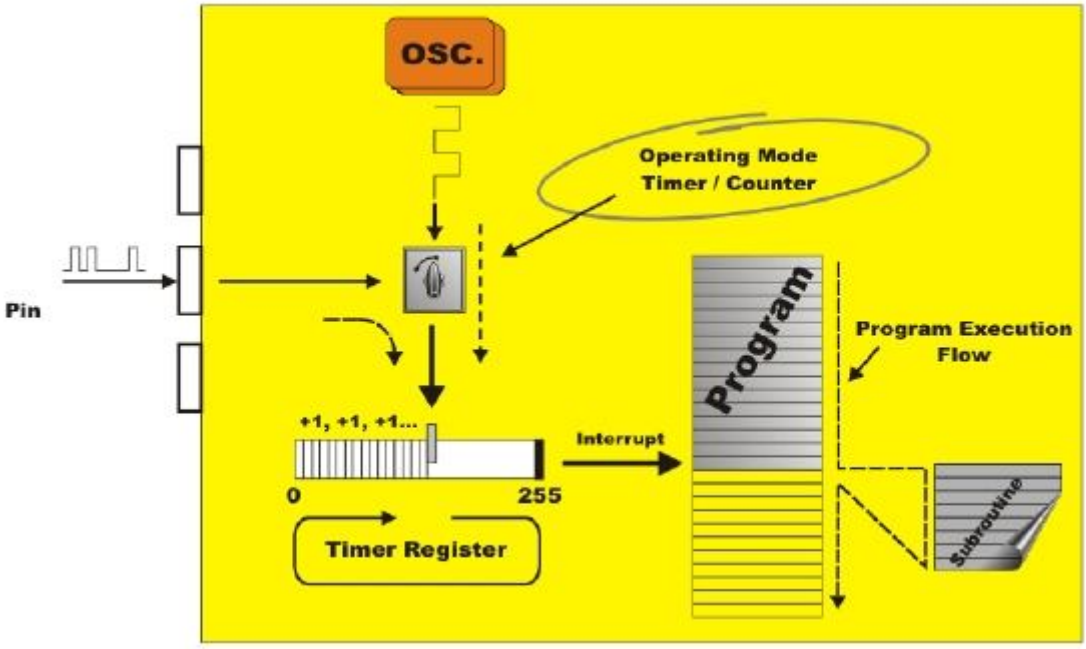


Hình 2-4. Ghép nối bộ dao động

Bộ dao động đóng vai trò nhạc trưởng làm nhiệm vụ đồng bộ hóa hoạt động của tất cả các mạch bên trong vi điều khiển. Nó thường được tạo bởi thạch anh hoặc gốm để ổn định tần số. Các lệnh không được thực thi theo tốc độ của bộ dao động mà thường chậm hơn, bởi vì mỗi câu lệnh được thực hiện qua nhiều bước. Mỗi loại vi điều khiển cần số chu kỳ khác nhau để thực hiện lệnh.

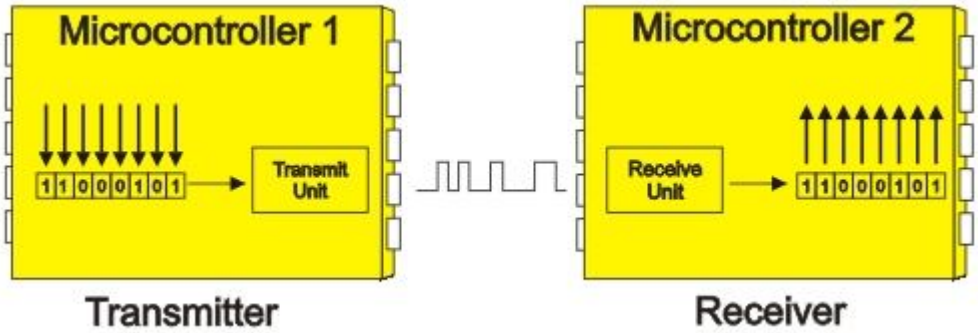
2.4.9. Bộ định thời/đếm (Timers/Counters)

Hầu hết các chương trình sử dụng các bộ định thời trong hoạt động của mình. Chúng thường là các thanh ghi SFR 8 hoặc 16 bit, sau mỗi xung dao động clock, giá trị của chúng được tăng lên. Ngay khi thanh ghi tràn, một ngắt sẽ được phát sinh.



Hình 2-5. Bộ định thời/đếm

2.4.10. Truyền thông nối tiếp



Hình 2-6. Truyền nhận nối tiếp

Kết nối song song giữa vi điều khiển và thiết bị ngoại vi được thực hiện qua các cổng vào/ra là giải pháp lý tưởng với khoảng cách ngắn trong vài mét. Tuy nhiên khi cần truyền thông giữa các thiết bị ở khoảng cách xa thì không thể dùng kết nối song song, vì vậy truyền thông nối tiếp là giải pháp tốt nhất.

Ngày nay, hầu hết các vi điều khiển có một số bộ điều khiển truyền thông nối tiếp như một trang bị tiêu chuẩn. Chúng được sử dụng phụ thuộc vào nhiều yếu tố khác nhau như:

- Bao nhiêu thiết bị vi điều khiển muốn trao đổi dữ liệu
- Tốc độ trao đổi dữ liệu
- Khoảng cách truyền
- Truyền/nhận dữ liệu đồng thời hay không?

Chương trình

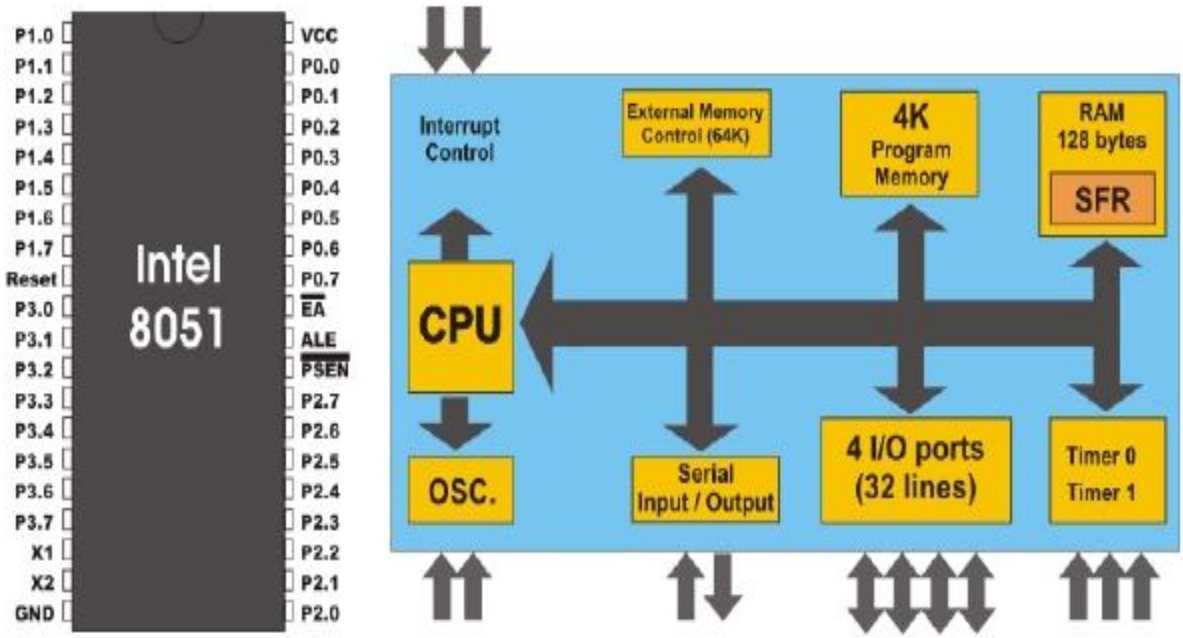
Không giống như các mạch tích hợp, chỉ cần kết nối các thành phần với nhau và bật nguồn, vi điều khiển cần phải lập trình trước. Để viết một chương trình cho vi điều khiển, có một vài ngôn ngữ lập trình bậc thấp có thể sử dụng như Assembly, C hay Basic. Viết một chương trình bao gồm việc viết các câu lệnh đơn giản theo một thứ tự để chúng có thể thực thi. Có rất nhiều phần mềm chạy trên môi trường Windows cho phép xây dựng các chương trình hoàn chỉnh cho các họ vi điều khiển.

CHƯƠNG 3
KIẾN TRÚC VI ĐIỀU KHỂN 8051

3.1. CHUẨN 8051

Họ vi điều khiển MCS-51 do Intel sản xuất đầu tiên vào năm 1980 là các IC thiết kế cho các ứng dụng hướng điều khiển. Các IC này chính là một hệ thống vi xử lý hoàn chỉnh bao gồm các các thành phần của hệ vi xử lý: CPU, bộ nhớ, các mạch giao tiếp, điều khiển ngắt.

MCS-51 là họ vi điều khiển sử dụng cơ chế CISC (Complex Instruction Set Computer), có độ dài và thời gian thực thi của các lệnh khác nhau. Tập lệnh cung cấp cho MCS-51 có các lệnh dùng cho điều khiển xuất/nhập tác động đến từng bit. MCS 51 bao gồm nhiều vi điều khiển khác nhau, bộ vi điều khiển đầu tiên là 8051 có 4KB ROM, 128 byte RAM và 8031, không có ROM nội, phải sử dụng bộ nhớ ngoài. Sau này, các nhà sản xuất khác như Siemens, Fujitsu, … cũng được cấp phép làm nhà cung cấp thứ hai. MCS-51 bao gồm nhiều phiên bản khác nhau, mỗi phiên bản sau tăng thêm một số thanh ghi điều khiển hoạt động của MCS-51.



Hình 3-1.Kiến trúc vi điều khiển 8051

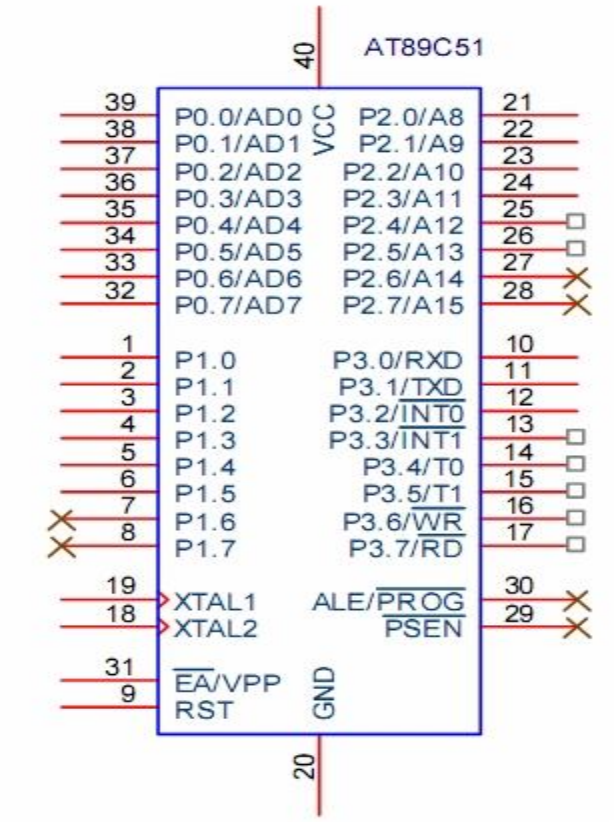
AT89C51 là vi điều khiển do Atmel sản xuất, chế tạo theo công nghệ CMOS có các đặc tính như sau:

+ 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.
+ Có các chế độ nghỉ (Low-power Idle) và chế độ nguồn giảm (Power-down).
+ Ngoài ra, một số IC khác của họ MCS-51 có thêm bộ định thời thứ 3 và 256 byte RAM nội.

3.2. CHÂN VI ĐIỀU KHIỂN 8051



Hình 3-2. Sơ đồ chân VĐK AT89C51

Chip AT89C51 có các tín hiệu điều khiển cần phải lưu ý như sau:

Tín hiệu vào **/EA** trên chân 31 thường đặt lên mức cao (+5V) hoặc mức thấp (GND). Nếu ở mức cao, 8951 thi hành chương trình từ ROM nội trong khoảng địa chỉ thấp (4K hoặc tối đa 8k đối với 89C52). Nếu ở mức thấp, chương trình được thi hành từ bộ nhớ mở rộng (tối đa đến 64Kbyte). Ngoài ra người ta còn dùng /EA làm chân cấp điện áp 12V khi lập trình EEPROM trong 8051.

Chân PSEN (Program store enable):

PSEN là chân tín hiệu ra trên chân 29. Nó là tín hiệu điều khiển cho phép chương trình mở rộng, PSEN thường được nối đến chân /OE (Output Enable) của một EPROM hoặc ROM để cho phép đọc các bytes mã lệnh.

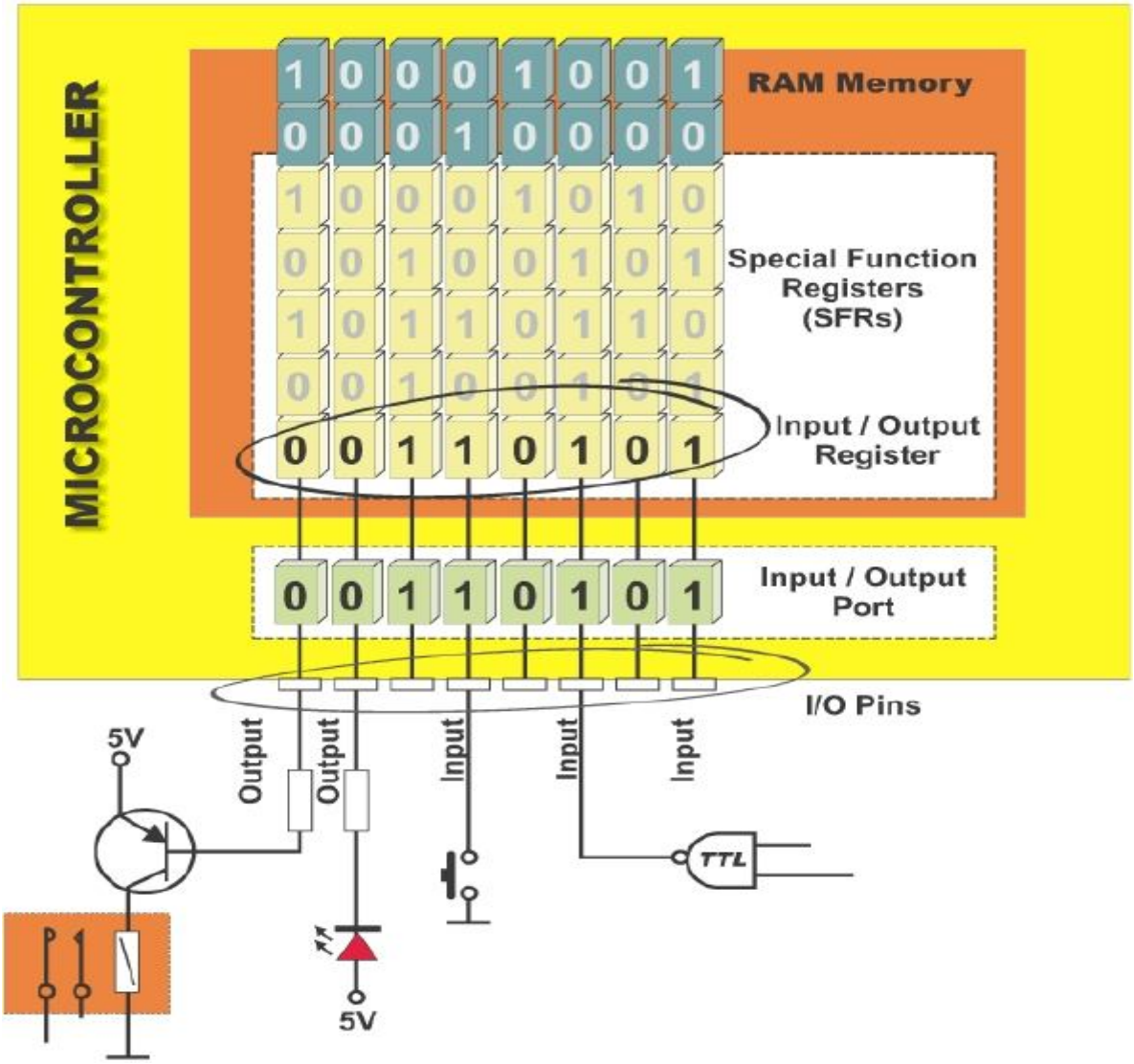
Hãy nhớ rằng : bình thường chân /PSEN sẽ được thả trống (No Connect).Chỉ khi nào cho /EA ở mức thấp thì lúc đó: /PSEN sẽ ở mức thấp trong thời gian lấy lệnh. Các mã nhị phân của chương trình được lấy từ EPROM qua bus dữ liệu và được chốt vào thanh ghi lệnh của 8951 để giải mã lệnh. /PSEN ở mức thụ động (mức cao) nếu thi hành chương trình trong ROM nội của 8951.

Các chân nguồn:

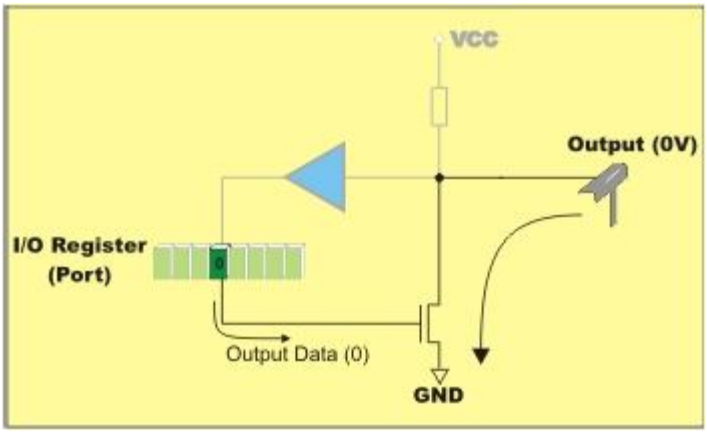
AT89C51 hoạt động ở nguồn đơn +5V. Vcc được nối vào chân 40, và Vss (GND) được nối vào chân 20.

3.3. CỔNG VÀO/ RA

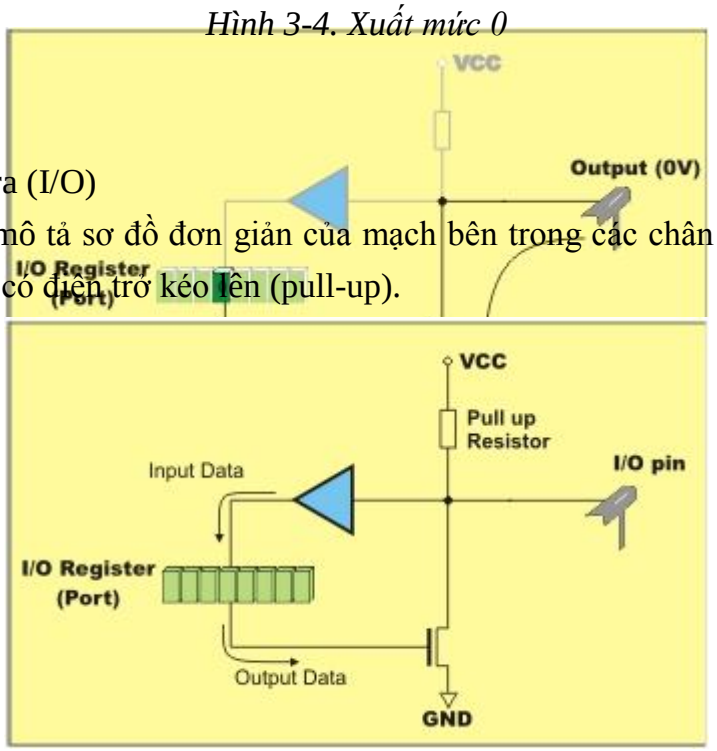
Tất cả các vi điều khiển 8051 đều có 4 cổng vào/ra 8 bit có thể thiết lập như cổng vào hoặc ra. Như vậy có tất cả 32 chân I/O cho phép vi điều khiển có thể kết nối với các thiết bị ngoại vi.



Hình 3-3. Cổng vào/ra



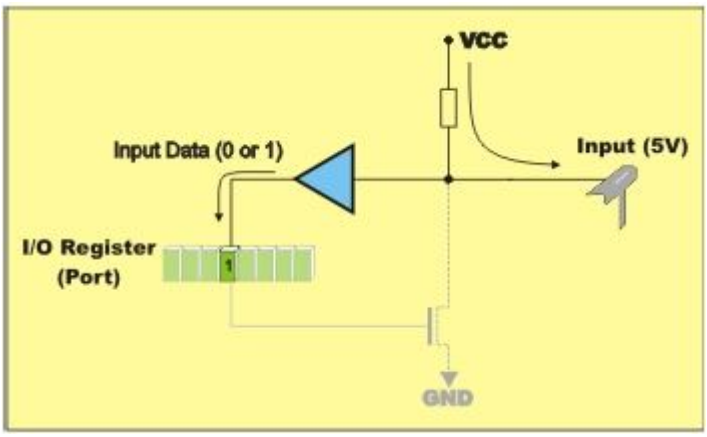
Hình 3-4. Xuất mức 0



Hình 3-5. Trở treo nội tại chân

Chân ra

Một mức logic 0 đặt vào bit của thanh ghi P làm cho transistor mở, nối chân tương ứng với đất.



Hình 3-6. xuất mức 1

Chân vào

Một bit 1 đặt vào một bit của thanh ghi cổng, transistor đóng và chân tương ứng được nối với nguồn Vcc qua trở kéo lên.

Port 0

Port 0 là port có 2 chức năng ở các chân 32 – 39 của AT89C51:

- Chức năng I/O (xuất/nhập): dùng cho các thiết kế nhỏ. Tuy nhiên, khi dùng chức năng này thì Port 0 phải dùng thêm các điện trở kéo lên (pull-up), giá trị của điện trở phụ thuộc vào thành phần kết nối với Port.
- Khi dùng làm ngõ vào, Port 0 phải được set mức logic 1 trước đó.
- Chức năng địa chỉ / dữ liệu đa hợp: khi dùng các thiết kế lớn, đòi hỏi phải sử dụng bộ nhớ ngoài thì Port 0 vừa là bus dữ liệu (8 bit) vừa là bus địa chỉ (8 bit thấp).

Ngoài ra khi lập trình cho AT89C51, Port 0 còn dùng để nhận mã khi lập trình và xuất mã khi kiểm tra (quá trình kiểm tra đòi hỏi phải có điện trở kéo lên).

Port 1:

Port1 (chân 1 – 8) chỉ có một chức năng là I/O, không dùng cho mục đích khác (chỉ trong 8032/8052/8952 thì dùng thêm P1.0 và P1.1 cho bộ định thời thứ 3). Tại Port 1 đã có điện trở kéo lên nên không cần thêm điện trở ngoài.

Port 1 có khả năng kéo được 4 ngõ TTL và còn dùng làm 8 bit địa chỉ thấp trong quá trình lập trình hay kiểm tra.

Khi dùng làm ngõ vào, Port 1 phải được set mức logic 1 trước đó.

Port 2:

Port 2 (chân 21 – 28) là port có 2 chức năng:

+ Chức năng I/O (xuất / nhập)
+ Chức năng địa chỉ: dùng làm 8 bit địa chỉ cao khi cần bộ nhớ ngoài có địa chỉ 16 bit. Khi đó, Port 2 không được dùng cho mục đích I/O.
+ Khi dùng làm ngõ vào, Port 2 phải được set mức logic 1 trước đó.

Port 3:

Port 3 (chân 10 – 17) là port có 2 chức năng:

+ Chức năng I/O. Khi dùng làm ngõ vào, Port 3 phải được set mức logic 1 trước đó.
+ Chức năng khác: mô tả như sau:

Bit	Tên	Chức năng
P3.0	RxD	Ngõ vào port nối tiếp
P3.1	TxD	Ngõ ra port nối tiếp
P3.2	INT0	Ngắt ngoài 0
P3.3	INT1	Ngắt ngoài 1
P3.4	T0	Ngõ vào của bộ định thời 0
P3.5	T1	Ngõ vào của bộ định thời 1
P3.6	WR	Tín hiệu điều khiển ghi dữ liệu lên bộ nhớ ngoài.
P3.7	RD	Tín hiệu điều khiển đọc từ bộ nhớ dữ liệu ngoài.

Bảng 3-1. Chức năng các chân của Port 3

Các chân nguồn:

+ Chân 40: VCC = 5V± 20%

+ Chân 20: GND

/PSEN (Program Store Enable):

/PSEN (chân 29) cho phép đọc bộ nhớ chương trình mở rộng đối với các ứng dụng sử dụng ROM ngoài, thường được nối đến chân /OC (Output Control) của ROM để đọc các byte mã lệnh. /PSEN sẽ ở mức logic 0 trong thời gian AT89C51 lấy lệnh.Trong quá trình này, / PSEN sẽ tích cực 2 lần trong 1 chu kỳ máy.

Mã lệnh của chương trình được đọc từ ROM thông qua bus dữ liệu (Port0) và bus địa chỉ (Port0 + Port2).

Khi 8051 thi hành chương trình trong ROM nội, PSEN sẽ ở mức logic 1.

ALE/ PROG (Address Latch Enable / Program):

ALE/ PROG (chân 30) cho phép tách các đường địa chỉ và dữ liệu tại Port 0 khi truy xuất bộ nhớ ngoài. ALE thường nối với chân Clock của IC chốt (74373, 74573). Các xung tín hiệu ALE có tốc độ bằng 1/6 lần tần số dao động trên chip và có thể được dùng làm tín hiệu clock cho các phần khác của hệ thống. Xung này có thể cấm bằng cách set bit 0 của SFR tại địa chỉ 8Eh lên 1. Khi đó, ALE chỉ có tác dụng khi dùng lệnh MOVX hay MOVC. Ngoài ra, chân này còn được dùng làm ngõ vào xung lập trình cho ROM nội (/PROG **)**.

EA /VPP (External Access) :

EA (chân 31) dùng để cho phép thực thi chương trình từ ROM ngoài. Khi nối chân 31 với Vcc, AT89C51 sẽ thực thi chương trình từ ROM nội (tối đa 8KB), ngược lại thì thực thi từ ROM ngoài (tối đa 64KB).

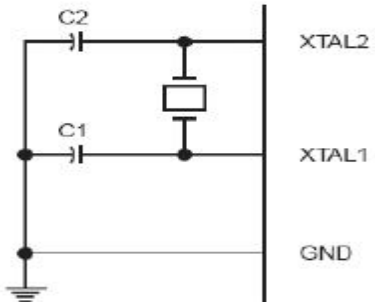
Ngoài ra, chân /EA được lấy làm chân cấp nguồn 12V khi lập trình cho ROM.

RST (Reset):

RST (chân 9) cho phép reset AT89C51 khi ngõ vào tín hiệu đưa lên mức 1 trong ít nhất là 2 chu kỳ máy.

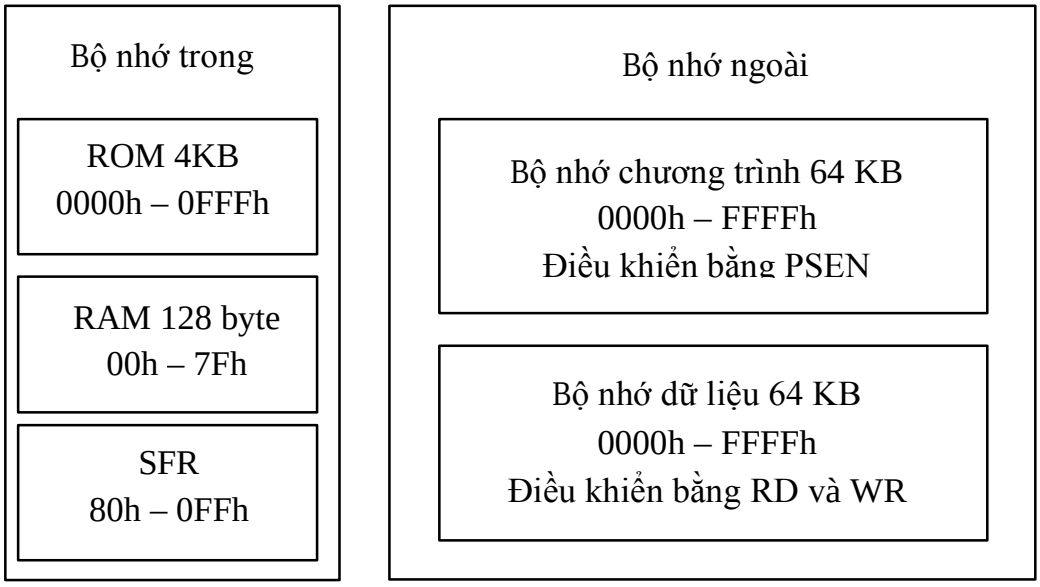
X1, X2:

Ngõ vào và ngõ ra bộ dao động, khi sử dụng có thể chỉ cần kết nối thêm thạch anh và các tụ như hình vẽ trong sơ đồ. Tần số thạch anh thường sử dụng cho AT89C51 là 12Mhz.



Hình 3-7 . Sơ đồ kết nối thạch anh

3.4 . TỔ CHỨC BỘ NHỚ



Hình 3-8. Các vùng nhớ trong AT89C51

Bộ nhớ của họ MCS-51 có thể chia thành 2 phần: bộ nhớ trong và bộ nhớ ngoài. Bộ nhớ trong bao gồm 4 KB ROM và 128 byte RAM (256 byte trong 8052). Các byte RAM có địa chỉ từ 00h – 7Fh và các thanh ghi chức năng đặc biệt (SFR) có

địa chỉ từ 80h – 0FFh có thể truy xuất trực tiếp. Đối với 8052, 128 byte RAM cao (địa chỉ từ 80h – 0FFh) không thể truy xuất trực tiếp mà chỉ có thể truy xuất gián tiếp (xem thêm trong phần tập lệnh).

Bộ nhớ ngoài bao gồm bộ nhớ chương trình (điều khiển đọc bằng tín hiệu PSEN **)** và bộ nhớ dữ liệu (điều khiển bằng tín hiệu RD hay WR để cho phép đọc hay ghi dữ liệu). Do số đường địa chỉ của MCS-51 là 16 bit (Port 0 chứa 8 bit thấp và Port 2 chứa 8 bit cao) nên bộ nhớ ngoài có thể giải mã tối đa là 64KB.

Địa chỉ byte	**Có thể định địa chỉ bit**	**Không định địa chỉ bit**						
F8h								
F0h	B							
E8h								
E0h	ACC							
D8h								
D0h	PSW							
C8h	(T2CON)		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)		
C0h								
B8h	IP	SADEN						
B0h	P3							
A8h	IE	SADDR						
A0h	P2							
98h	SCON	SBUF	BRL	BDRCON				
90h	P1							
88h	TCON	TMOD	TL0	TH0	TL1	TH1	AUXR	CKCON
80h	P0	SP	DPL	DPH				PCON

3.4.1 Tổ chức bộ nhớ trong

Bộ nhớ trong của MCS-51 gồm ROM và RAM. RAM bao gồm nhiều vùng có mục đích khác nhau: vùng RAM đa dụng (địa chỉ byte từ 30h – 7Fh và có thêm vùng 80h – 0FFh ứng với 8052), vùng có thể địa chỉ hóa từng bit (địa chỉ byte từ 20h – 2Fh, gồm 128 bit được định địa chỉ bit từ 00h – 7Fh), các bank thanh ghi (từ 00h – 1Fh) và các thanh ghi chức năng đặc biệt (từ 80h – 0FFh).

Các thanh ghi chức năng đặc biệt (SFR – Special Function Registers):

Bảng 3-2. Các thanh ghi chức năng đặc biệt

Địa chỉ byte	Địa chỉ bit								Chức năng
7F 30									Vùng RAM đa dụng
2F	7F	7E	7D	7C	7B	7A	79	78	Vùng có thể định địa chỉ bit
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 18	Bank 3								Các bank thanh ghi
17 10	Bank 2								
1F 08	Bank 1								
07 00	Bank thanh ghi 0 (mặc định cho R0-R7)								

Các thanh ghi có thể định địa chỉ bit sẽ có địa chỉ bit bắt đầu và địa chỉ byte trùng nhau. Ví dụ như: thanh ghi P0 có địa chỉ byte là 80h và có địa chỉ bit bắt đầu từ 80h (ứng với P0.0) đến 87h (ứng với P0.7). Chức năng các thanh ghi này sẽ mô tả trong phần sau.

a. RAM nội:

chia thành các vùng phân biệt: vùng RAM đa dụng (30h – 7Fh), vùng RAM có thể định địa chỉ bit (20h – 2Fh) và các bank thanh ghi (00h – 1Fh).

Bảng 3-3. Địa chỉ RAM nội 8051

b. RAM đa dụng:

RAM đa dụng có 80 byte từ địa chỉ 30h – 7Fh có thể truy xuất mỗi lần 8 bit bằng cách dùng chế độ địa chỉ trực tiếp hay gián tiếp.

Các vùng địa chỉ thấp từ 00h – 2Fh cũng có thể sử dụng cho mục đích như trên ngoài các chức năng đề cập như phần sau.

c. RAM có thể định địa chỉ bit:

Vùng địa chỉ từ 20h – 2Fh gồm 16 byte (= 128 bit) có thể thực hiện giống như vùng RAM đa dụng (mỗi lần 8 bit) hay thực hiện truy xuất mỗi lần 1 bit bằng các lệnh xử lý bit. Vùng RAM này có các địa chỉ bit bắt đầu tại giá trị 00h và kết thúc tại 7Fh.

Như vậy, địa chỉ bắt đầu 20h (gồm 8 bit) có địa chỉ bit từ 00h – 07h; địa chỉ kết thúc 2Fh có địa chỉ bit từ 78h – Fh.

d. Các bank thanh ghi:

Vùng địa chỉ từ 00h – 1Fh được chia thành 4 bank thanh ghi: bank 0 từ 00h-07h, bank 1 từ 08h – 0Fh, bank 2 từ 10h – 17h và bank 3 từ 18h – 1Fh. Các bank thanh ghi này được đại diện bằng các thanh ghi từ R0 đến R7. Sau khi khởi động hệ thống thì bank thanh ghi được sử dụng là bank 0.

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. Việc thay đổi bank thanh ghi có thể thực hiện thông qua thanh ghi từ trạng thái chương trình (PSW). Các bank thanh ghi này cũng có thể truy xuất bình thường như vùng RAM đa dụng đã nói ở trên.

3. 4.2. Tổ chức bộ nhớ ngoài

MCS-51 có bộ nhớ theo cấu trúc Harvard: phân biệt bộ nhớ chương trình và dữ liệu. Chương trình và dữ liệu có thể chứa bên trong nhưng vẫn có thể kết nối với 64KB chương trình và 64KB dữ liệu. Bộ nhớ chương trình được truy xuất thông qua chân PSEN còn bộ nhớ dữ liệu đươc truy xuất thông qua chân WR hay RD .

Lưu ý rằng việc truy xuất bộ nhớ chương trình luôn luôn sử dụng địa chỉ 16 bit còn bộ nhớ dữ liệu có thể là 8 bit hay 16 bit tuỳ theo câu lệnh sử dụng. Khi dùng bộ nhớ dữ liệu 8 bit thì có thể dùng Port 2 như là Port I/O thông thường còn khi dùng ở chế độ 16 bit thì Port 2 chỉ dùng làm các bit địa chỉ cao.

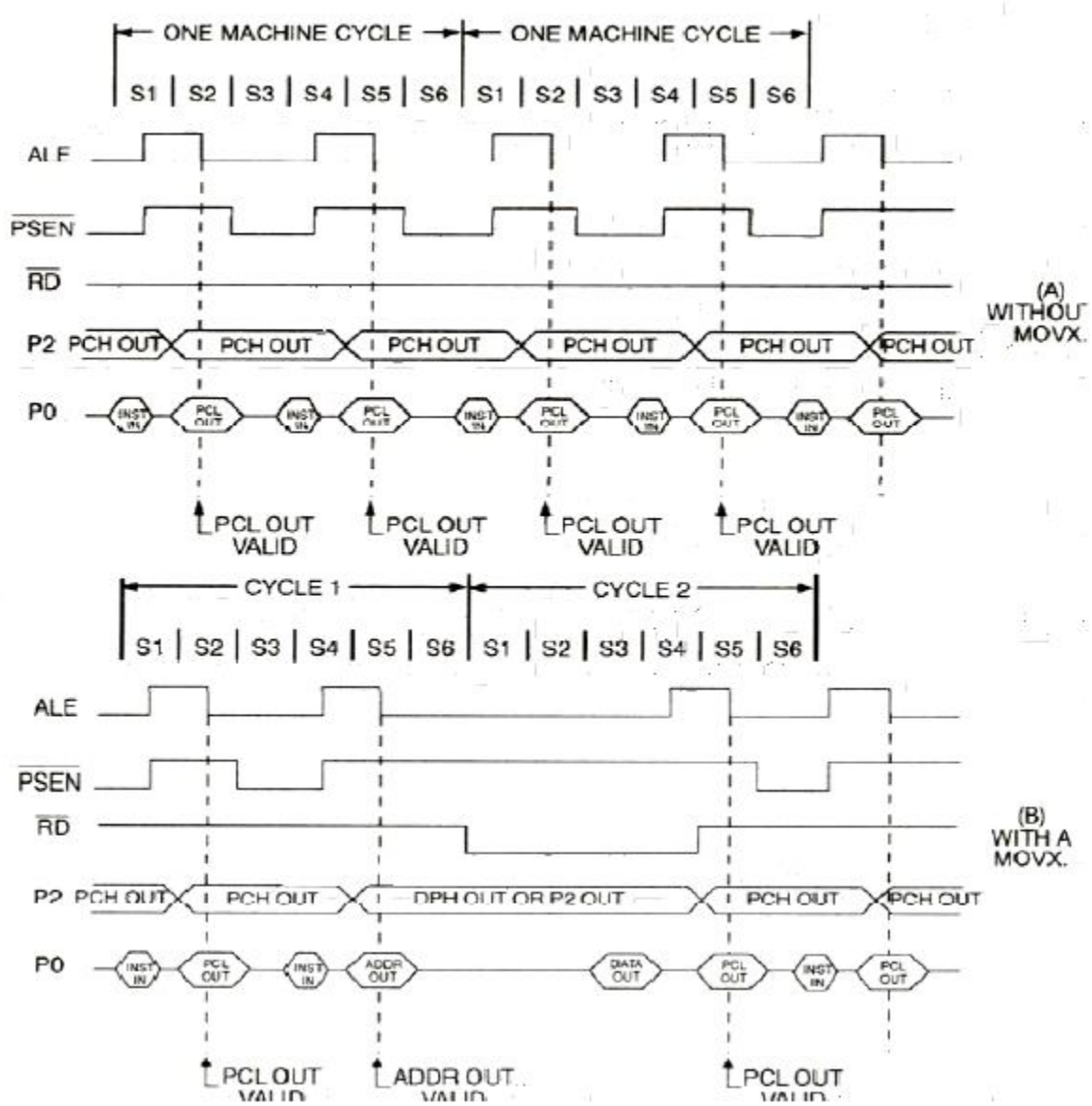
Port 0 được dùng làm địa chỉ thấp/ dữ liệu đa hợp. Tín hiệu /ALE để tách byte địa chỉ và đưa vào bộ chốt ngoài.

Trong chu kỳ ghi, byte dữ liệu sẽ tồn tại ở Port 0 vừa trước khi /WR tích cực và được giữ cho đến khi /WR không tích cực.Trong chu kỳ đọc, byte nhận được chấp nhận vừa trước khi /RD không tích cực.

Bộ nhớ chương trình ngoài được xử lý 1 trong 2 điều kiện sau:

+ Tín hiệu /EA tích cực (= 0).

+ Giá trị của bộ đếm chương trình (PC – Program Counter) lớn hơn kích thước bộ nhớ.



PCH: Program Counter High – PCL: Program Counter Low

DPH: Data Pointer High – DPL: Data Pointer Low

Hình 3-9. Thực thi bộ nhớ chương trình ngoài

a. Bộ nhớ chương trình ngoài:

Quá trình thực thi lệnh khi dùng bộ nhớ chương trình ngoài có thể mô tả như "Hình 3-9. Thực thi bộ nhớ chương trình ngoài". Trong quá trình này, Port 0 và Port 2 không còn là các Port xuất nhập mà chứa địa chỉ và dữ liệu. Sơ đồ kết nối với bộ nhớ chương trình ngoài mô tả như "Hình 3-8. Các vùng nhớ trong AT89C51".

Trong một chu kỳ máy, tín hiệu ALE tích cực 2 lần. Lần thứ nhất cho phép 74HC573 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 ROM chưa xuất vì PSEN chưa tích cực, khi tín hiệu ALE lên 1 trở lại thì Port 0 đã có dữ liệu là mã lệnh. ALE tích cực lần thứ hai

được giải thích tương tự và byte 2 được đọc từ bộ nhớ chương trình. Nếu lệnh đang thực thi là lệnh 1 byte thì CPU chỉ đọc Opcode, còn byte thứ hai bỏ qua.

b. Bộ nhớ dữ liệu ngoài:

Bộ nhớ dữ liệu ngoài được truy xuất bằng lệnh MOVX thông qua các thanh ghi xác định địa chỉ DPTR (16 bit) hay R0, R1 (8 bit).

Quá trình thực hiện đọc hay ghi dữ liệu được cho phép bằng tín hiệu RD hay WR (chân P3.7 và P3.6).

c. Bộ nhớ chương trình và dữ liệu dùng chung:

Trong các ứng dụng phát triển phần mềm xây dựng dựa trên AT89C51, ROM sẽ được lập trình nhiều lần nên dễ làm hư hỏng ROM. Một giải pháp đặt ra là sử dụng RAM để chứa các chương trình tạm thời. Khi đó, RAM vừa là bộ nhớ chương trình vừa là bộ nhớ dữ liệu. Yêu cầu này có thể thực hiện bằng cách kết hợp chân RD và chân PSEN thông qua cổng AND. Khi thực hiện đọc mà lệnh, chân /PSEN tích cực cho phép đọc từ RAM và khi đọc dữ liệu, chân RD sẽ tích cực.

d. Giải mã địa chỉ

Trong các ứng dụng dựa trên AT89C51, ngoài giao tiếp bộ nhớ dỡ liệu, vi điều khiển còn thực hiện giao tiếp với các thiết bị khác như bàn phím, led, động cơ, … Các thiết bị này có thể giao tiếp trực tiếp thông qua các Port. Tuy nhiên, khi số lượng các thiết bị lớn, các Port sẽ không đủ để thực hiện điều khiển. Giải pháp đưa ra là xem các thiết bị này giống như bộ nhớ dữ liệu. Khi đó, cần phải thực hiện quá trình giải mã địa chỉ để phân biệt các thiết bị ngoại vi khác nhau. Quá trình giải mã địa chỉ thường được thực hiện thông qua các IC giải mã như 74139 (2 -> 4), 74138 (3 -> 8), 74154 (4 -> 16). Ngõ ra của các IC giải mã sẽ được đưa tới chân chọn chip của RAM hay bộ đệm khi điều khiển ngoại vi.

3.5. CÁC THANH GHI CHỨC NĂNG ĐẶC BIỆT (SFRs - Special Function Registers)

- Thanh ghi tích luỹ (Accumulator)

Thanh ghi tích luỹ là thanh ghi sử dụng nhiều nhất trong AT89C51, được ký hiệu trong câu lệnh là A. Ngoài ra, trong các lệnh xử lý bit, thanh ghi tích luỹ được ký hiệu là ACC.

Thanh ghi tích luỹ có thể truy xuất trực tiếp thông qua địa chỉ E0h (byte) hay truy xuất từng bit thông qua địa chỉ bit từ E0h đến E7h.

VD: Câu lệnh:

```
MOV   A,#1
MOV   0E0h,#1
```

có cùng kết quả. Hay:

```
SETB ACC.4
SETB 0E4h
```

- Thanh ghi B dùng cho các phép toán nhân, chia và có thể dùng như một thanh ghi tạm, chứa các kết quả trung gian.

Thanh ghi B có địa chỉ byte F0h và địa chỉ bit từ F0h – F7h có thể truy xuất giống như thanh ghi A.

- Thanh ghi từ trạng thái chương trình (PSW - Program Status Word)

Thanh ghi từ trạng thái chương trình PSW nằm tại địa chỉ D0h và có các địa chỉ bit từ D0h – D7h, bao gồm 7 bit (1 bit không sử dụng) có các chức năng như sau:

Bit	7	6	5	4	3	2	1	0
Chức năng	CY	AC	F0	RS1	RS0	OV	F1	P

Hình 3-10. Thanh ghi PSW

CY (Carry): cờ nhớ, thường được dùng cho các lệnh toán học không dấu (C = 1 khi có nhớ trong phép cộng hay mượn trong phép trừ)

AC (Auxiliary Carry): cờ nhớ phụ (thường dùng cho các phép toán BCD).

F0 (Flag 0): được sử dụng tuỳ theo yêu cầu của người sử dụng.

RS1, RS0: dùng để chọn bank thanh ghi sử dụng. Khi reset hệ thống, bank 0 sẽ được sử dụng.

RS1	RS0	Bank thanh ghi
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

Hình 3-11. Chọn bank thanh ghi

OV (Overflow): cờ tràn. Cờ OV = 1 khi có hiện tượng tràn số học xảy ra (dùng cho số nguyên có dấu).

F1 (Flag 1): được sử dụng tuỳ theo yêu cầu của người sử dụng.

P (Parity): kiểm tra parity (lẻ). Cờ P = 1 khi tổng số bit 1 trong thanh ghi A là số lẻ (nghĩa là tổng số bit 1 của thanh ghi A cộng thêm cờ P là số chẵn). Ví dụ như: A = 10101010b có tổng cộng 4 bit 1 nên P = 0. Cờ P thường được dùng để

kiểm tra lỗi truyền dữ liệu.

- Thanh ghi con trỏ stack (SP – Stack Pointer)

Con trỏ stack SP nằm tại địa chỉ 81h và không cho phép định địa chỉ bit. SP dùng để chỉ đến đỉnh của stack. Stack là một dạng bộ nhớ lưu trữ dạng LIFO (Last In First Out) thường dùng lưu trữ địa chỉ trả về khi gọi một chương trình con. Ngoài ra, stack còn dùng như bộ nhớ tạm để lưu lại và khôi phục các giá trị cần thiết.

Đối với AT89C51, stack được chứa trong RAM nội (128 byte đối với 8031/8051 hay 256 byte đối với 8032/8052). Mặc định khi khởi động, giá trị của SP là 07h, nghĩa là stack bắt đầu từ địa chỉ 08h (do hoạt động lưu giá trị vào stack yêu cầu phải tăng nội dung thanh ghi SP trước khi lưu). Như vậy, nếu không gán giá trị cho thanh ghi SP thì không được sử dụng các bank thanh ghi 1, 2, 3 vì có thể làm sai dữ liệu. Đối với các ứng dụng thông thường không cần dùng nhiều đến stack, có thể không cần khởi động SP mà dùng giá trị mặc định là 07h. Tuy nhiên, nếu cần, ta có thể xác định lại vùng stack cho MCS-51.

- Con trỏ dữ liệu DPTR (Data Pointer)

Con trỏ dữ liệu DPTR là thanh ghi 16 bit bao gồm 2 thanh ghi 8 bit: DPH (High) nằm tại địa chỉ 83h và DPL (Low) nằm tại địa chỉ 82h. Các thanh ghi này không cho phép định địa chỉ bit. DPTR được dùng khi truy xuất đến bộ nhớ có địa chỉ 16 bit.

- Các thanh ghi port

Các thanh ghi P0 tại địa chỉ 80h, P1 tại địa chỉ 90h, P2, tại địa chỉ A0h, P3 tại địa chỉ B0h là các thanh ghi chốt cho 4 port xuất / nhập (Port 0, 1, 2, 3). Tất cả các thanh ghi này đều cho phép định địa chỉ bit trong đó địa chỉ bit của P0 từ 80h – 87h, P1 từ 90h – 97h, P2 từ A0h – A7h, P3 từ B0h – B7h. Các địa chỉ bit này có thể thay thế bằng toán tử địa chỉ.

Ví dụ: 2 lệnh sau là tương đương:

```
SETB P0.0
SETB 80h
```

Thanh ghi port nối tiếp (SBUF - Serial Data Buffer)

Thanh ghi port nối tiếp tại địa chỉ 99h thực chất bao gồm 2 thanh ghi: thanh ghi nhận và thanh ghi truyền. Nếu dữ liệu đưa tới SBUF thì đó là thanh ghi truyền, nếu dữ liệu đươc đọc từ SBUF thì đó là thanh ghi nhận. Các thanh ghi này không cho phép định địa chỉ bit.

- Các thanh ghi định thời (Timer Register)

Các cặp thanh ghi (TH0, TL0), (TH1, TL1) và (TH2, TL2) là các thanh ghi dùng cho các bộ định thời 0, 1 và 2 trong đó bộ định thời 2 chỉ có trong 8032/8052. Ngoài ra, đối với họ 8032/8052 còn có thêm cặp thanh ghi (RCAP2L, RCAP2H) sử dụng cho bộ định thời 2 (sẽ thảo luận trong phần hoạt động định thời).

- Các thanh ghi điều khiển:

Bao gồm các thanh ghi IP (Interrupt Priority), IE (Interrupt Enable), TMOD (Timer Mode), TCON (Timer Control), T2CON (Timer 2 Control), SCON (Serial port control) và PCON (Power control).

+ Thanh ghi IP tại địa chỉ B8h cho phép chọn mức ưu tiên ngắt khi có 2 ngắt xảy ra đông thời. IP cho phép định địa chỉ bit từ B8h – BFh.

+ Thanh ghi IE tại địa chỉ A8h cho phép hay cấm các ngắt. IE có địa chỉ bit từ A8h – AFh.

+ Thanh ghi TMOD tại địa chỉ 89h dùng để chọn chế độ hoạt động cho các bộ định thời (0, 1) và không cho phép định địa chỉ bit.

+ Thanh ghi TCON tại địa chỉ 88h điều khiển hoạt động của bộ định thời và ngắt. TCON có địa chỉ bit từ 88h – 8Fh.

+ Thanh ghi T2CON tại địa chỉ C8h điều khiển hoạt động của bộ định thời 2. T2CON có địa chỉ bit từ C8h – CFh.

+ Thanh ghi SCON tại địa chỉ 98h điều khiển hoạt động của port nối tiếp. SCON có địa chỉ bit từ 98h – 9Fh.

Các thanh ghi đã nói ở trên sẽ được thảo luận thêm ở các phần sau.

- Thanh ghi điều khiển nguồn PCON

Thanh ghi PCON tại địa chỉ 87h không cho phép định địa chỉ bit bao gồm các bit như sau:

Bit	7	6	5	4	3	2	1	0
Chức năng	SMOD1	SMOD0	-	POF	GF1	GF0	PD	IDL

Hình 3-12. Thanh ghi PCON

SMOD1 (Serial Mode 1): = 1 cho phép tăng gấp đôi tốc độ port nối tiếp trong chế độ 1, 2 và 3.

SMOD0 (Serial Mode 0 = 0): cho phép chọn bit SM0 hay FE trong thanh ghi SCON (Serial Mode 0 = 1 chọn bit FE).

POF (Power-off Flag): dùng để nhận dạng loại reset. POF = 1 khi mở nguồn. Do đó, để xác định loại reset, cần phải xoá bit POF trước đó.

GF1, GF0 (General purpose Flag): các bit cờ dành cho người sử dụng.

PD (Power Down): được xoá bằng phần cứng khi hoạt động reset xảy ra.

Khi bit PD = 1 thì vi điều khiển sẽ chuyển sang chế độ nguồn giảm. Trong chế độ này:

- Chỉ có thể thoát khỏi chế độ nguồn giảm bằng cách reset.
- Nội dung RAM và mức logic trên các port được duy trì.
- Mạch dao động bên trong và các chức năng khác ngừng hoạt động.
- Chân ALE và PSEN ở mức thấp.
- Yêu cầu Vcc phải có điện áp ít nhất là 2V và phục hồi Vcc = 5V ít nhất 10 chu kỳ trước khi chân RESET xuống mức thấp lần nữa.

IDL (Idle): được xoá bằng phần cứng khi hoạt động reset hay có ngắt xảy ra. Khi bit IDL = 1 thì vi điều khiển sẽ chuyển sang chế độ nghỉ. Trong chế độ này:

- Chỉ có thể thoát khỏi chế độ nguồn giảm bằng cách reset hay có ngắt xảy ra.
- Trạng thái hiện hành của vi điều khiển được duy trì và nội dung các thanh ghi không đổi.
- Mạch dao động bên trong không gởi được tín hiệu đến CPU.
- Chân ALE và PSEN ở mức cao.

Lưu ý rằng các bit điều khiển PD và IDL có tác dụng chính trong tất cả các IC họ MSC-51 nhưng chỉ có thể thực hiện được trong các phiên bản CMOS.

3.6. BỘ ĐẾM / BỘ ĐỊNH THỜI

Định thời là sự hoạt động để kiểm soát thời gian thực thi các câu lệnh trong quá trình xử lý của vi điều khiển.

8051 có hai bộ định thời/ bộ đếm. Chúng có thể được dùng như các bộ định thời để tạo một bộ trễ thời gian hoặc như các bộ đếm để đếm các sự kiện xảy ra bên ngoài bộ VĐK. Các timer này đều là timer 16bit, giá trị đếm được tính từ 0 đến 216 (đếm từ 0 đến 65535).

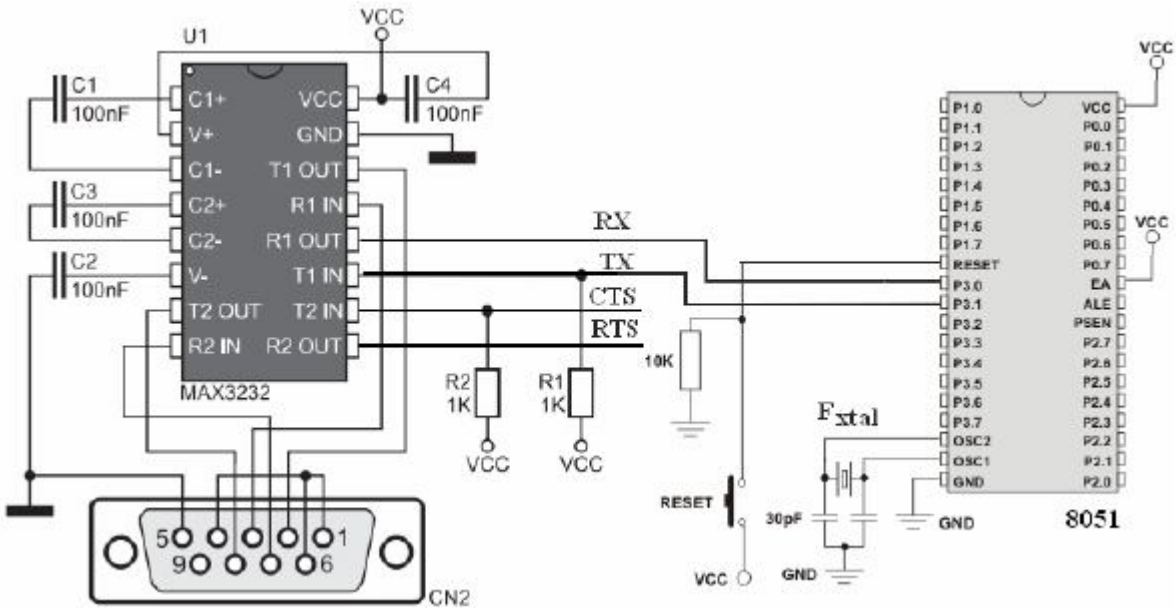
Hai timer có nguyên lý hoạt động hoàn toàn giống nhau và độc lập. Sau khi cho phép chạy, mỗi khi có thêm một xung tại đầu vào đếm, giá trị của timer sẽ tự động được tăng lên 1 đơn vị, cứ như vậy cho đến khi giá trị tăng lên vượt quá giá trị 65535 mà thanh ghi đếm có thể biểu diễn thì giá trị đếm lại được đưa trở về giá trị 0

Việc cho timer chạy/dừng được thực hiện bởi các bit TR trong thanh ghi TCON (đánh địa chỉ đến từng bit).

Các timer có thể hoạt động theo nhiều chế độ, được quy định bởi các bit trong thanh ghi TMOD.

3.7. TRUYỀN THÔNG KHÔNG ĐỒNG BỘ (UART)

8051 có 1 cổng UART làm việc ở chuẩn TTL, mặc định sau khi khởi động tất các cổng của 8051 đều làm việc ở chế độ vào ra số, vì thế để có thể sử dụng UART cần phải cấu hình cho cổng này làm việc thông qua các thanh ghi điều khiển và ghép nối tương thích với chuẩn RS 232.



Hình 3-13 - Ghép nối RS232 với 8051

Cổng nối tiếp trong 8051 chủ yếu được dùng trong các ứng dụng có yêu cầu truyền thông với máy tính, hoặc với một vi điều khiển khác. Liên quan đến cổng nối tiếp chủ yếu có 2 thanh ghi: SCON và SBUF. Ngoài ra, một thanh ghi khác là thanh ghi PCON (không đánh địa chỉ bit) có bit 7 tên là SMOD quy định tốc độ truyền của cổng nối tiếp có gấp đôi lên (SMOD = 1) hay không (SMOD = 0).

Dữ liệu được truyền nhận nối tiếp thông qua hai chân cổng P3.0(RxD) và P3.1(TxD).

3.8. NGẮT VI ĐIỀU KHIỂN 8051

8051 hỗ trợ 5 loại ngắt, mỗi ngắt có một vector ngắt riêng, đó là một địa chỉ cố định nằm trong bộ nhớ chương trình. Khi xảy ra ngắt CPU sẽ tự động nhảy đến thực hiện lệnh thuộc địa chỉ này.

Liên quan đến ngắt chủ yếu có hai thanh ghi là thanh ghi IE và thanh ghi IP. Thanh ghi IE là thanh ghi đánh địa chỉ bit, do đó có thể dùng các lệnh tác động bit để tác động riêng rẽ lên từng bit mà không làm ảnh hưởng đến giá trị các bit khác. Để cho phép một ngắt, bit tương ứng với ngắt đó và bit EA phải được đặt bằng 1.

CHƯƠNG 4
LẬP TRÌNH HỢP NGỮ CHO 8051

Lập trình cho vi điều khiển cũng tương tự như lập trình cho máy tính, bản chất là ta gia lệnh cho vi điều khiển thực hiện 1 danh sách các lệnh cơ bản được sắp xếp theo một trình tự nào đó để có thể hoàn thành một nhiệm vụ đề ra. Và tất cả những lệnh mà vi điều khiển có thể hiểu được gọ là tập lệnh. Các vi điều khiển tương thích với 8051 có 255 lệnh.

4.1 CÁC CHẾ ĐỘ ĐỊA CHỈ

Có thể truy cập dữ liệu theo nhiều cách khác nhau. Dữ liệu có thể ở trong một thanh ghi hoặc trong bộ nhớ hoặc được cho như một giá trị tức thời các cách truy cập dữ liệu khác nhau được gọi là các chế độ đánh địa chỉ. Chương này chúng ta bàn luận về các chế độ đánh địa chỉ của 8051 trong phạm vi một số ví dụ.

Các chế độ đánh địa chỉ khác nhau của bộ vi xử lý được xác định như nó được thiết kế và do vậy người lập trình không thể đánh địa chỉ khác nhau là:

1. Tức thời 2. Theo thanh ghi 3. Trực tiếp
4. Gián tiếp 5. Theo chỉ số

4.1.1. Địa chỉ tức thời

Trong chế độ đánh địa chỉ này toán hạng nguồn là một hằng số. Và như tên gọi của nó thì khi một lệnh được hợp dịch toán hạng đi tức thi ngay sau mã lệnh. Lưu ý rằng trước dữ liệu tức thời phải được đặt dấu (#) chế độ đánh địa chỉ này có thể được dùng để nạp thông tin vào bất kỳ thanh ghi nào kể cả thanh ghi con trỏ dữ liệu DPTR.

Ví dụ:

```
MOV  DPTR, #MYDATA
MOV  A, # 25H              ; Nạp giá trị 25H vào thanh ghi A
MOV  R4, #62               ; Nạp giá trị  62 thập phân vào R4
MOV  B, #40H               ; Nạp giá trị 40 H vào thanh ghi B
MOV  DPTR, #4521H          ; Nạp 4512H vào con trỏ dữ liệu DPTR
```

Mặc dù thanh ghi DPTR là 16 bit nó cũng có thể được truy cập như 2 thanh ghi 8 bit DPH và DPL trong đó DPH là byte cao và DPL là byte thấp.

Cũng lưu ý rằng lệnh dưới đây có thể tạo ra lỗi vì giá trị nạp vào DPTR lớn hơn16 bit:

```
MOV   DPTR, # 68975      ; Giá tri không hợp lệ  > 65535 (FFFFH)
```

4.1.2. Địa chỉ theo thanh ghi

Chế độ đánh địa chỉ theo thanh ghi liên quan đến việc sử dụng các thanh ghi

để lưu dữ liệu cần được thao tác và các các toán hạng là 1 trong các thanh ghi Ri của các bank được chọn.

Ví dụ :

```
MOV  R2, A        ; Sao chép nội  dung thanh ghi  A vào thanh ghi R2
ADD  A, R5        ; Cộng nội dung  thanh ghi R5 vào thanh ghi A
```

Ta có thể chuyển dữ liệu giữa thanh ghi tích luỹ A và thanh ghi Rn (n từ 0 đến 7) nhưng việc chuyển dữ liệu giữa các thanh ghi Rn thì không được phép.

Ví dụ: MOV R4, R7 là không hợp lệ.

Lưu ý rằng các thanh ghi nguồn và đích phải phù hợp về kích thước. Hay nói cách khác, nếu viết " MOV DPTR, A" sẽ cho một lỗi vì nguồn là thanh ghi 8 bit và đích lại là thanh ghi 16 bit.

4.1.3. Địa chỉ trực tiếp

8051 có 128 byte bộ nhớ RAM. Bộ nhớ RAM được gán các địa chỉ từ 00 đến 7FH và được phân chia như sau:

1. Các ngăn nhớ từ 00 đến 1FH được gán cho các băng thanh ghi và ngăn xếp.
2. Các ngăn nhớ từ 20H đến 2FH được dành cho không gian đánh địa chỉ theo bit để lưu các dữ liệu 1 bit.
3. Các ngăn nhớ từ 30H đến 7FH là không gian để lưu dữ liệu có kích thước 1byte.

Mặc dù toàn bộ byte của bộ nhớ RAM có thể được truy cập bằng chế độ đánh địa chỉ trực tiếp, nhưng chế độ này thường được sử dụng nhất để truy cập các ngăn nhớ RAM từ 30H đến 7FH. Đây là do một thực tế là các ngăn nhớ dành cho băng ghi được truy cập bằng thanh ghi theo các tên gọi của chúng là R0 - R7 còn các ngăn nhớ khác của RAM thì không có tên như vậy. Trong chế độ đánh địa chỉ trực tiếp thì dữ liệu ở trong một ngăn nhớ RAM mà địa chỉ của nó được biết và địa chỉ này được cho như là một phần của lệnh. Khác với chế độ đánh địa chỉ tức thì mà toán hạng tự nó được cấp với lệnh. Dấu (# 0 là sự phân biệt giữa hai chế độ đánh địa chỉ. Xét các ví dụ dưới đây và lưu ý rằng các lệnh không có dấu (#):

```
MOV  R0, 40H      ; Lưu nội dung của ngăn nhớ 40H của RAM vào R0
MOV  56H, A       ; Lưu nội dung thanh ghi A vào ngăn nhớ 56H của RAM
MOV  R4, 7FH      ; Chuyển nội dung ngăn nhớ 7FH của RAM  vào R4
```

Các thanh ghi R0 - R7 có thể được truy cập theo 2 cách như sau:

```
MOV  A, 4    ; Hai lệnh này giống nhau đều sao nội dung thanh ghi  R4 vào A
MOV  A, R4  ;
```

4.1.4. Địa chỉ gián tiếp.

Trong chế độ này, một thanh ghi được sử dụng như một con trỏ đến dữ liệu. Nếu dữ liệu ở bên trong CPU thì chỉ các thanh ghi R0 và R1 được sử dụng cho mục đích này. Hay nói cách khác các thanh ghi R2 - R7 không thể dùng được để giữ địa chỉ của toán hạng nằm trong RAM khi sử dụng chế độ đanh địa chỉ này khi R0 và R1 được dùng như cac con trỏ, nghĩa là khi chúng giữ các địa chỉ của các ngăn nhớ RAM thì trước chúng phải đặt dấu (@) như chỉ ra dưới đây.

MOV A, @ R0 ; Chuyển nội dung của ngăn nhớ RAM có địa chỉ trong R0 vào A

MOV @ R1, B ; Chuyển nội dung của B vào ngăn nhớ RAM có địa chỉ ở R1

Lưu ý rằng R0 cũng như R1 luôn có dấu "@" đứng trước. Khi không có dấu này thì đó là lệnh chuyển nội dung các thanh ghi R0 và R1 chứ không phải dữ liệu ngăn nhớ mà địa chỉ có trong R0 và R1.

* Ưu điểm của chế độ đánh địa chỉ gián tiếp thanh ghi.

Một trong những ưu điểm của chế độ đánh địa chỉ gián tiếp thanh ghi là nó làm cho việc truy cập dữ liệu năng động hơn so với chế độ đánh địa chỉ trực tiếp.

Ví dụ:

Viết chương trình để sao chép giá trị 55H vào ngăn nhớ RAM tại địa chỉ 40H đến 44H sử dụng:

- Chế độ định địa chỉ trực tiếp
- Chế độ đánh địa chỉ gián tiếp thanh ghi không dùng vòng lặp

Lời giải:

- Chế độ địa chỉ trực tiếp.

```
MOV  A, #55H          ; Nạp A giá trị  55H
MOV  40H, A           ; Sao chép A vào ngăn nhớ RAM  40H
MOV  41H, A           ; Sao chép A vào ngăn nhớ RAM  41H
MOV  42H, A           ; Sao chép A vào ngăn nhớ RAM  42H
MOV  43H, A           ; Sao chép A vào ngăn nhớ RAM  43H
MOV  44H, A           ; Sao chép A vào ngăn nhớ RAM  44H
```

- Chế độ đánh địa chỉ gián tiếp thanh ghi không dùng vòng lặp

```
MOV  A, # 55H      ; Nạp vào A giỏ trị 55H
MOV  R0, #40H      ; Nạp con trỏ  R0 = 40 H
MOV  @R0, A      ; Sao chép A vào vị trí ngăn nhớ RAM do R0 chỉ đến
INC    R0          ; Tăng con trỏ. Bây giờ R0 = 41H
MOV  @R0, A        ; Sao chép A vào vị trí ngăn nhớ RAM do R0 chỉ
```

```
INC   R0          ; Tăng con trỏ. Bây giờ R0 = 42H
MOV  @R0,A        ; Sao chép A vào vị trí ngăn nhớ RAM do R0 chỉ
INC   R0          ; Tăng con trỏ. Bây giờ R0 = 43H
MOV  @R0, A       ; Sao chép A vào vị trí ngăn nhớ RAM do R0 chỉ
INC   R0          ;Tăng con trỏ. Bây giờ R0 = 44H
MOV  @R0, A
```

4.1.5. Địa chỉ theo chỉ số

Chế độ đánh địa chỉ theo chỉ số được sử dụng rộng rãi trongviệc truy cập các phân tử dữ liệu của bảng trong không gian ROM/RAM chương trình của 8051 trong dải 64KB. Lệnh được dùng cho mục đích này là

MOVC A, @ A + DPTR

Thanh ghi 16 bit DPTR là thanh ghi A được dùng để tạo ra địa chỉ của phần tử dữ liệu được lưu cất trong ROM trên chíp. Do các phần tử dữ liệu được cất trong không gian mã (chương trình) của ROM trên chip của 8051, nó phải dùng lệnh MOVC thay cho lệnh MOV (chủ C ở cuối lệnh là chỉ mà lệnh Code). Trong lệnh này thì nội dung của A được bổ xung vào thanh ghi 16 bit DPTR để tạo ra địa chỉ 16 bit của dữ liệu cần thiết.

4.2. TẬP LỆNH TRONG 8051

4.2.1. Phân loại tập lệnh

Tùy thuộc vào cách và chức năng của mỗi lệnh, có thể chia ra thành 5 nhóm lệnh như sau:

Các lệnh toán học

Các lệnh điều khiển chương trình

Các lệnh vận chuyển dữ liệu

Các lệnh logic

Các lệnh thao tác bit

4.2.2. Cấu trúc chung của mỗi lệnh

Mã_lệnh Toán_hạng1, Toán_hạng2, Toán_hạng3

Trong đó:

Mã_lệnh: Tên gợi nhớ cho chức năng của lệnh. (VD như add cho addition)

Toán_hạng1, Toán_hạng2, Toán_hạng3: Là các toán hạnh của lệnh, tùy thuộc vào mỗi lệnh số toán hạng có thể không có, có 1, 2 hoặc 3.

Ví dụ:

RET (Kết thúc chương trình con) Lệnh này không có toán hạng

JZ TEMP (Chuyển con trỏ chương trình đến vị trí TEMP) Chỉ có 1 toán hạng

ADD A, R3; (A = A + R3) Có 2 toán hạng

CJNE A, #20, LOOP (So sánh A với 20, nếu không bằng thì chuyển con trỏ chương trình đến nhã LOOP) Có 3 toán hạng

4.2.3. Các lệnh toán học

Thực hiện các phép tính cơ bản như +, -, *, /, … Kết quả sau khi thực hiện lệnh được lưu vào toán hạng đầu tiên trong lệnh

Các lệnh toán học như: ADD, ADDC, SUBB, INC, DEC, MUL, DA.

a. Phép cộng

ADD A, nguồn ; A = A + nguồn

Lệnh ADD được dùng để cộng hai toán hạng. Toán hạng đích luôn là thanh ghi A trong khi đó toán hạng nguồn có thể là một thanh ghi dữ liệu trực tiếp hoặc là ở trong bộ nhớ. Hãy nhớ rằng các phép toán số học từ bộ nhớ đến bộ nhớ không bao giờ được phép trong hợp ngữ. Lệnh này có thể thay đổi một trong các bit AF, CF hoặc PF của thanh ghi cờ phụ thuộc vào các toán hạng liên quan.

Ví dụ:

Hãy biểu diễn xem các lệnh dưới đây tác động đến thanh ghi cờ như thế nào?

MOV A, # 0F5H ; A = F5H

MOV A, # 0BH ; A = F5 + 0B = 00

Lời giải:

F5H		1111 0101
+0BH	+	0000 1011
100H		0000 0000

Sau phép cộng, thanh ghi A (đích) chứa 00 và các cờ sẽ như sau:

CY = 1 vì có phép nhớ từ D7

PF = 1 vì số các số 1 là 0 (một số chẵn) cờ PF được đặt lên 1.

AC = 1 vì có phép nhớ từ D3 sang D4

b. Phép trừ các số không dấu.

Cú pháp: SUBB A, nguồn ; A = A - nguồn - CY.

Trong rất nhiều các bộ xử lý có hai lệnh khác nhau cho phép trừ đó là SUB và SUBB (trừ có mượn). Trong 8051 ta chỉ có một lệnh SUBB duy nhất. Để thực hiện SUB từ SUBB, do vậy có hai trường hợp cho lệnh SUBB là: với CY = 0 và với CY = 1. Lưu ý ở đây ta dùng cờ CY để mượn.

- Lệnh SUBB với CY = 0.

Trong phép trừ thì các bộ vi xử lý 8051 (thực tế là tất cả mọi CPU hiện đại) đều sử dụng phương pháp bù 2. Mặc dù mỗi CPU đều có mạch cộng, nó có thể quá cồng kềnh (và cần nhiều bóng bán dẫn) để thiết kế mạch trừ riêng biệt. Vì ly do đó mà 8051 sử dụng mạch cộng để thực hiện lệnh trừ. Giả sử 8051 sử dụng mạch cộng để thực hiện lệnh trừ và rằng CY - 0 trước khi thực hiện lệnh thì ta có thể tóm tắt các bước mà phần cứng CPU thực hiện lệnh SUBB đối với các số không dấu như sau:

1. Thực hiện lấy bù 2 của số trừ (toán hạng nguồn)
2. Cộng nó vào số bị trừ (A)
3. Đảo nhớ

Đây là 3 bước thực hiện bởi phần cứng bên trong của CPU 8051 đối với mỗi lệnh trừ SUBB bất kể đến nguồn của các toán hạng được cấp có được hỗ trợ chế độ đánh địa chỉ hay không? Sau ba bước này thì kết quả có được và các cờ được bật.

Ví dụ :

Phân tích chương trình sau:

```
        CLR  C
        MOV  A, #4CH        ; Nạp A giá trị 4CH (A = 4CH)
        SUBB A, #6EH        ; Trừ A cho 6EH
        JNC  NEXT           ; Nếu CY = 0 nhảy đến đích NEXT
        CPL  A              ; Nếu CY = 1 thực hiện bù 1
        INC  A              ; Tăng 1 để có bù 2
NEXT:   MOV  R1, A          ; Lưu A vào R1
```

- Lệnh SUBB khi CY = 1.

Lệnh này được dùng đối với các số nhiều byte và sẽ theo dõi việc mượn của toán hạng thấp. Nếu CY = 1 trước khi xem thực hiện SUBB thì nó cũng trừ 1 từ kết quả.

Ví dụ:

Phân tích chương trình sau:

```
CLR  C          ; CY = 0
MOV  A, #62     ; A = 62H
SUBB A, #96H    ; 62H - 96H = CCH with CY = 1
MOV  R7, A      ; Save the result
MOV  A, #27H    ; A = 27H
SUBB A, #12H    ; 27H - 12H - 1 = 14H
MOV  R6, A      ;
```

Sau khi SUBB thì A = 62H - 96H = CCH và cờ nhớ được lập báo rằng có mượn. Vì CY = 1 nên khi SUBB được thực hiện lần thứ 2 thì a = 27H - 12H - 1 = 14H. Do vậy, ta có 2762H - 1296H = 14CCH.

c. Nhân hai số không dấu.

MUL AB ; Là phép nhân A × B và kết quả 16 bit được đặt trong A và B.

Khi nhân byte với byte thì một trong các toán hạng phải trong thanh ghi A và toán hạng thứ hai phải ở trong thanh ghi B. Sau khi nhân kết quả ở trong các thanh ghi A và B. Phần tiếp thấp ở trong A, còn phần cao ở trong B. Ví dụ dưới đây trình bày phép nhân 25H với 65H. Kết quả là dữ liệu 16 bit được đặt trong A và B.

```
MOV  A, #25H          ; Nạp vào A giá trị 25H
MOV  B, #65H          ; Nạp vào B giá trị 65H
MUL  AB               ; 25H*65H = E99 với B = 0EH và A = 99H
```

Nhân	Toán hạng 1	Toán hạng 2	Kết quả
Byte*Byte	A	B	A = byte thấp, B = byte cao

Bảng 4-1: Tóm tắt phép nhân hai số không dấu (MUL AB)

d. Chia hai số không dấu.

8051 cùng chỉ hỗ trợ phép chia hai số không dấu byte cho byte với cú pháp:

DIV AB ; Chia A cho B

Khi chia một byte cho một byte thì tử số (số bị chia) phải ở trong thanh ghi A và mẫu số (số chia) phải ở trong thanh ghi B. Sau khi lệnh chia DIV được thực hiện thì thương số được đặt trong A, còn số dư được đặt trong B. Xét ví dụ dưới đây:

```
MOV     A, #95      ; Nạp số bị chia vào A = 95
MOV     B, #10      ; Nạp số chia vào B = 10

DIV     AB          ; A = 09 (thương số); B = 05 (số dư)
```

Lưu ý các điểm sau khi thực hiện "DIV AB"

Lệnh này luôn bắt CY = 0 và OV = 0 nếu tử số không phải là số 0

Nếu tử số là số 0 (B = 0) thì OV =1 báo lỗi và CY = 0. Thực tế chuẩn trong tất cả mọi bộ vi xử lý khi chia một số cho 0 là bằng cách nào đó báo có kết quả không xác định. Trong 8051 thì cờ OV được thiết lập lên 1.

Phép chia	Tử số	Mẫu số	Thương số	Số dư
Byte cho Byte	A	B	A	B

Bảng 4-2. Tóm tắt phép chia không dấu (DIV AB).

4.2.4. Các lệnh logic

Thực hiện các phép toán logic, các lệnh bao gồm:

ANL: phép toán "Và" logic
ORL: phép toán "Hoặc " logic
XRL: phép toán "XOR " logic
CPL: phép toán bù
RL: phép quay bit sang trái
RR: phép quay bit sang phải
RLC: : phép quay trái có nhớ
RRC: phép quay phải có nhớ
SWAP: lệnh trao đổi thanh ghi

a. Lệnh Và (ANL).

Cú pháp: ANL đích, nguồn ; đích = đích Và nguồn

Lệnh này sẽ thực hiện một phép Và lô-gíc trên hai toán hạng đích và nguồn và đặt kết quả vào đích. Đích thường là thanh ghi tổng (tích luỹ). Toán hạng nguồn có thể là thanh ghi trong bộ nhớ hoặc giá trị cho sẵn. Lệnh AND đối với toán hạng theo byte không có tác động lên các cờ. Nó thường được dùng để che (đặt về 0) những bit nhất định của một toán hạng.

Ví dụ:

Trình bày kết quả của các lệnh sau:

```
MOV  A, #35H          ; Gán A = 35H
ANL  A, #0FH          ; Thực hiện Và logic A và 0FH (Bây giờ A = 05)
```

Lời giải:

```
35H   0  0  1  1  0  1  0  1
0FH   0  0  0  0  1  1  1  1
05H   0  0  0  0  0  1  0  1          35H và 0FH = 05H
```

b. Lệnh Hoặc (ORL).

Cú pháp ORL đích, nguồn ; đích = đích hoặc nguồn.

Các toán hạng đích và nguồn được Hoặc với nhau và kết quả được đặt vào đích. Phép Hoặc có thể được dùng để thiết lập những bit nhất định của một toán hạng 1. Đích thường là thanh ghi tổng, toán hạng nguồn có thể là một thanh ghi trong bộ nhớ hoặc giá trị cho sẵn. Lệnh ORL đối với các toán hạng đánh địa chỉ theo byte sẽ không có tác động đến bất kỳ cờ nào.

Ví dụ: Trình bày kết quả của đoạn mã sau:

```
MOV  A, #04          ; A = 04H
ORL   A, #68H        ; A = 6CH
```

Lời giải:

```
04H   0000  0100
68H   0110  1000
6CH   0110  1100
 04  OR  68 = 6CH
```

c. Lệnh XOR (hoặc loại trừ).

Cú pháp: XRL đích, nguồn ; đích = đích Hoặc loại trừ nguồn.

Lệnh này sẽ thực hiện phép XRL trên hai toán hạng và đặt kết quả vào đích. Đích thường là thanh ghi tổng. Toán hạng nguồn có thể là một thanh ghi trong bộ nhớ hoặc giá trị cho sẵn. Lệnh XRL đối với các toán hạng đánh địa chỉ theo byte sẽ không có tác động đến bất kỳ cờ nào.

Ví dụ: Trình bày kết quả của đoạn mã sau:

```
MOV  A, #54H
XRL   A, #78H
```

Lời giải:

```
54H   0  1  0  1  0  1  0  0
78H   0  1  1  1  1  0  0  0
2CH   0  0  1  0  1  1  0  1
54H  XRL  78H = 2CH
```

Lệnh XRL có thể được dùng để xoá nội dung của một thanh ghi bằng cách XOR nó với chính nó.

d. Lệnh bù thanh ghi tổng CPL A.

Lệnh này bù nội dung của thanh ghi tổng A. Phép bù là phép biến đổi các số 0 thành các số 1 và đổi các số 1 sang số 0. Đây cũng còn được gọi là phép bù 1.

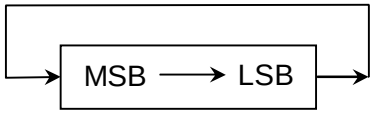
```
MOV          A, #55H
CPL          A            ; Bây giờ nội dung của thanh ghi A là AAH
```

e. Quay phải: RR

Cú pháp: **RR A** ; Quay các bit thanh ghi A sang phải.

Trong phép quay phải, 8 bit của thanh ghi tổng được quay sang phải một bit và bit D0 rời từ vị trí bit thấp nhất và chuyển sang bit cao nhất D7. Xem đoạn mã dưới đây.

```
MOV   A, #36H     ; A = 0011 0110
RR    A           ; A = 0001 1011
RR    A           ; A = 1000 1101
RR    A           ; A = 1100 0110
RR    A           ; A = 0110 0011
```

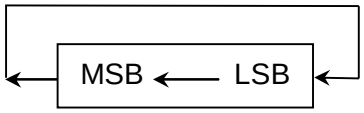


f. Quay trái: RL

Cú pháp: RL A ; Quay trái các bit của thanh ghi A.

Trong phép quay trái thì 8 bit của thanh ghi A được quay sang trái 1 bit và bit D7 rời khỏi vị trí bit cao nhất chuyển sang vị trí bit thấp nhất D0. Xem biểu đồ mã dưới đây.

```
MOV  A, #72H      ; A = 0111 0010
RL   A            ; A = 1110 0100
RL   A            ; A = 1100 1001
```



Lưu ý rằng trong các lệnh RR và RL thì không có cờ nào bị tác động.

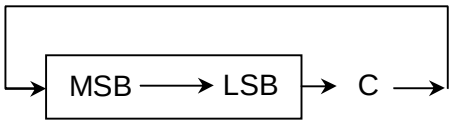
g. Quay có nhớ.

Cú pháp: RRC A và RLC A

* Quay phải có nhớ: RRC A

Trong quay phải có nhớ thì các bit của thanh ghi A được quay từ trái sang phải 1 bit và bit thấp nhất được đưa vào cờ nhớ CY và sau đó cờ CY được đưa vào vị trí bit cao nhất. Hay nói cách khác, trong phép RRC A thì LSB được chuyển vào CY và CY được chuyển vào MSB. Trong thực tế thì cờ nhớ CY tác động như là một bit bộ phận của thanh ghi A làm nó trở thành thanh ghi 9 bit.

```
CLR  C            ; make CY = 0
MOV  A #26H       ; A = 0010 0110
RRC  A            ; A = 0001 0011  CY = 0
RRC  A            ; A = 0000 1001  CY = 1
RCC  A            ; A = 1000 0100  CY = 1
```



* Quay trái có nhớ: RLC A.

Trong RLC A thì các bit được dịch phải một bit và đẩy bit MSB vào cờ nhớ CY, sau đó CY được chuyển vào bit LSB. Hay nói cách khác, trong RLC thì bit MSB được chuyển vào CY và CY được chuyển vào LSB. Hãy xem đoạn mã sau.

```
SETB C          ; Make CY = 1
MOV  A, #15H    ; A = 0001 0101
RRC  A          ; A = 0101 1011  CY = 0
RRC  A          ; A = 0101 0110  CY = 0
RCC  A          ; A = 1010 1100  CY = 0
RCC  A          ; A = 1000 1000  CY = 1
```

4.2.5. Các lệnh vận chuyển dữ liệu

Di chuyển dữ liệu từ ô nhớ này đến ô nhứ khác, hoặc giữa hai thanh ghi, thanh ghi ô nhớ.

Các lệnh vận chuyển dữ liệu bao gồm:

MOV: chuyển dữ liệu giữa thanh ghi với thanh ghi, thanh ghi với ô nhớ, một hằng số đến thanh ghi, một hằng số đến ô nhớ, và ngược lại

MOVC: Sao chép mã nguồn (dữ liệu đã được đặt trong vùng mã nguồn)

4.2.6. Các lệnh thao tác bit

SETB bit	Thiết lập bit (bit bằng 1)
CLR bit	Xoá bit về không (bit = 0)
CPL bit	Bù bit (bit = NOT bit)
JB bit, đích	Nhảy về đích nếu bit = 1
JNB bit,	đích Nhảy về đích nếu bit = 0
JNC đích	Nhảy tới đích nếu CY = 0
CLR C	Xoá bit nhớ CY = 0
JC đích	Nhảy tới đích nếu CY = 1

Ví dụ:

Hãy viết chương trình thực hiện các công việc sau:

a) Duy trì hiển thị bit P1.2 cho đến khi nó lên cao
b) Khi P1.2 lên cao, hãy ghi giá trị 45H vào cổng P0
c) Gửi một xung cao xuống thấp (H-to-L) tới P2.3

Lời giải:

```
        SETB P1.2          ; Tạo bit P1.2 là đầu vào
        MOV  A, #45H       ; Gán A = 45H
AGAIN:  JNB  P1.2, AGAIN   ; Thoát khi P1.2 = 1
```

```
MOV  P0, A          ; Xuất A tới cổng P0
SETB P2.3           ; Đưa P2.3 lên cao
CLR  P2.3           ; Tạo P2.3 xuống thấp để xung H-T0-L
```

Trong chương trình này lệnh "JNB P1.2, AGCN" (JNB có nghĩa là nhảy nếu không bit) ở lại vòng lặp cho đến khi P1.2 chưa lên cao. Khi P1.2 lên cao nó thoát ra khỏi vòng lặp ghi giá trị 45H tới cổng P0 và tạo ra xung H-to-L bằng chuỗi các lệnh SETB và CLR.

4.2.7. Lệnh đọc cổng

Trong việc đọc cổng thì một số lệnh đọc trạng thái của các chân cổng, còn một số lệnh khác thì đọc một số trạng thái của chốt cổng trong. Do vậy, khi đọc các cổng thì có hai khả năng:

1. Đọc trạng thái của cổng vào.

Lệnh	Ví dụ	Mô tả
MOV A, PX	MOV A, P2	Chuyển dữ liêuj ở chân P2 vào ACC
JNB PX.Y, ...	JNB P2.1, đích	Nhảy tới đích nếu, chân P2.1 = 0
JB PX.Y,	JB P1.3, đích	Nhảy đích nếu, chân P1.3 = 1
MOV C, PX.Y	MOV C, P2.4	Sao trạng thái chân P2.4 vào CY

Bảng 4-3. Lệnh đọc cổng

2. Đọc chốt trong của cổng ra.

Lệnh	Ví dụ
ANL PX	ANL P1, A
ORL PX	ORL P2, A
XRL PX	XRL P0, A
JBC PX.Y, đích	JBC P1.1, đích
CPL PX	CPL P1.2
INC PX	INC P1
DEC PX	DEC P2
DJN2 PX.Y, đích	DJN2 P1, đích
MOV PX.Y, C	MOV P1.2, C
CLR PX.Y	CLR P2.3
SETB PX.Y	SETB P2.3

Bảng 4-4. Lệnh đọc cổng ra

4.2.8. Các lệnh điều khiển chương trình (rẽ nhánh)

Nhóm lệnh điều khiển chương trình có thể chia thành 2 loại:

1. Nhảy vô điều kiện
2. Nhảy có điều kiện:

a. Các lệnh nhảy có điều kiện.

Các lệnh nhảy có điều kiện đối với 8051 được tổng hợp trong bảng 2.2. Các chi tiết về mỗi lệnh được cho trong phụ lục AppendixA. Trong bảng 2.2 lưu ý rằng một số lệnh như JZ (nhảy nếu A = 0) và JC (nhảy nếu có nhớ) chỉ nhảy nếu một điều kiện nhất định được thoả mãn. Kế tiếp ta xét một số lệnh nhảy có điều kiện với các Ví dụ minh hoạ sau.

Lệnh JZ: (nhảy nếu A = 0). Trong lệnh này nội dung của thanh ghi A được kiểm tra. Nếu nó bằng không thì nó nhảy đến địa chỉ đích. Ví dụ xét đoạn mã sau:

```
MOV     A, R0     ; Nạp giá trị của R0 vào A
JZ      OVER      ; Nhảy đến OVER nếu A = 0
MOV     A, R1     ; Nạp giá trị của R1 vào A
JZ      OVER      ; Nhảy đến OVER nếu A = 0
OVER ...
```

Trong chương trình này nếu R0 hoặc R1 có giá trị bằng 0 thì nó nhảy đến địa chỉ có nhãn OVER. Lưu ý rằng lệnh JZ chỉ có thể được sử dụng đối với thanh ghi A. Nó chỉ có thể kiểm tra xem thanh ghi A có bằng không không và nó không áp dụng cho bất kỳ thanh ghi nào khác. Quan trọng hơn là ta không phải thực hiện một lệnh số học nào như đếm giảm để sử dụng lệnh JNZ như ở ví dụ dưới đây.

Ví dụ :

Viết một chương trình để xác định xem R5 có chứa giá trị 0 không? Nếu nạp thì nó cho giá trị 55H.

Lời giải:

```
        MOV     A, R5     ; Sao nội dung R5 vào A
        JNZ     NEXT      ; Nhảy đến NEXT nếu A không bằng 0
        MOV     R5, #55H  ;
NEXT:           ...
```

Lệnh JNC: (nhảy nếu không có nhớ, cờ CY = 0).

Trong lệnh này thì bit cờ nhớ trong thanh ghi cờ PSW được dùng để thực hiện quyết định nhảy. Khi thực hiện lệnh "JNC nhãn" thì bộ xử lý kiểm tra cờ nhớ xem nó có được bật không (CY = 1). Nếu nó không bật thì CPU bắt đầu nạp và thực hiện các lệnh từ địa chỉ của nhãn. Nếu cờ CY = 1 thì nó sẽ không nhảy và thực hiện lệnh kế tiếp dưới JNC.

Cần phải lưu ý rằng cũng có lệnh "JC nhãn". Trong lệnh JC thì nếu CY = 1 nó nhảy đến địa chỉ đích là nhãn. Ta sẽ xét các ví dụ về các lệnh này trong các ứng dụng ở các chương sau.

Ngoài ra lệnh JB (nhảy nếu bit có mức cao), JNB (nhảy nếu bit có mức thấp).

Lệnh	**Hoạt động**
JZ	Nhảy nếu A = 0
JNZ	Nhảy nếu A ≠ 0
DJNZ	Giảm và nhảy nếu A = 0
CJNE A, byte	Nhảy nếu A ≠ byte
CJNE re, # data	Nhảy nếu Byte ≠ data
JC	Nhảy nếu CY = 1
JNC	Nhảy nếu CY = 0
JB	Nhảy nếu bit = 1
JNB	Nhảy nếu bit = 0
JBC	Nhảy nếu bit = 1 và xoá nó

Bảng 4-5. Các lệnh nhảy có điều kiện.

Ví dụ :

Hãy tìm tổng của các giá trị 79H, F5H và E2H. Đặt vào trong các thanh ghi R0 (byte thấp) và R5 (byte cao).

Lời giải:

```
        MOV  A, #0        ; Xoá thanh ghi A = 0
        MOV  R5, A        ; Xoá R5
        ADD  A #79H       ; Cộng 79H vào A (A = 0 + 79H = 79H)
        JNC  N-1          ; Nếu không có nhớ cộng kế tiếp
        INC  R5           ; Nếu CY = 1, tăng R5
N-1:    ADD  A, #0F5H     ; Cộng F5H vào A (A = 79H + F5H = 6EH) và CY = 1
        JNC  N-2          ; Nhảy nếu CY = 0
        INC  R5           ; Nếu CY = 1 tăng R5 (R5 = 1)
```

```
N-2:   ADD   A, #0E2H     ; Cộng E2H vào A (A = GE + E2 = 50) và CY = 1
       JNC   OVER         ; Nhảy nếu CY = 0
       INC   R5           ; Nếu CY = 1 tăng R5
OVER:  MOV   R0, A        ; Bây giờ R0 = 50H và R5 = 02
```

Tất cả các lệnh nhảy có điều kiện đều là những phép nhảy ngắn.

Cần phải lưu ý rằng tất cả các lệnh nhảy có điều kiện đều là các phép nhảy ngắn, có nghĩa là địa chỉ của đích đều phải nằm trong khoảng -127 đến +127 byte của nội dung bộ đếm chương trình PC.

b. Các lệnh nhảy không điều kiện.

Lệnh nhảy không điều kiện là một phép nhảy trong đó điều khiển được truyền không điều kiện đến địa chỉ đích. Trong 8051 có hai lệnh nhảy không điều kiện đó là: LJMP - nhảy xa và SJMP - nhảy gần.

Nhảy xa LJMP:

Nhảy xa LJMP là một lệnh 3 byte trong đó byte đầu tiên là mã lệnh còn hai byte còn lại là địa chỉ 16 bit của đích. Địa chỉ đích 02 byte có phép một phép nhảy đến bất kỳ vị trí nhớ nào trong khoảng 0000 - FFFFH.

Hãy nhớ rằng, mặc dù bộ đếm chương trình trong 8051 là 16 bit, do vậy cho không gian địa chỉ là 64k byte, nhưng bộ nhớ chương trình ROM trên chíp lớn như vậy. 8051 đầu tiên chỉ có 4k byte ROM trên chíp cho không gian chương trình, do vậy mỗi byte đều rất quý giá. Vì lý do đó mà có cả lệnh nhảy gần SJMP chỉ có 2 byte so với lệnh nhảy xa LJMP dài 3 byte. Điều này có thể tiết kiệm được một số byte bộ nhớ trong rất nhiều ứng dụng mà không gian bộ nhớ có hạn hẹp.

Lệnh nhảy gần SJMP.

Trong 2 byte này thì byte đầu tiên là mã lệnh và byte thứ hai là chỉ tương đối của địa chỉ đích. Đích chỉ tương đối trong phạm vi 00 - FFH được chia thành các lệnh nhảy tới và nhảy lùi: Nghĩa là -128 đến +127 byte của bộ nhớ tương đối so với địa chỉ hiện thời của bộ đếm chương trình. Nếu là lệnh nhảy tới thì địa chỉ đích có thể nằm trong khoảng 127 byte từ giá trị hiện thời của bộ đếm chương trình. Nếu địa chỉ đích ở phía sau thì nó có thể nằm trong khoảng -128 byte từ giá trị hiện hành của PC.

Lệnh gọi xa LCALL

Trong lệnh 3 byte này thì byte đầu tiên là mã lệnh, còn hai byte sau được dùng cho địa chỉ của chương trình con đích. Do vậy LCALL có thể được dùng để gọi các chương trình con ở bất kỳ vị trí nào trong phạm vi 64k byte, không gian địa chỉ của 8051. Để đảm bảo rằng sau khi thực hiện một chương trình được gọi để 8051 biết

được chỗ quay trở về thì nó tự động cất vào ngăn xếp địa chỉ của lệnh đứng ngay sau lệnh gọi LCALL. Khi một chương trình con được gọi, điều khiển được chuyển đến chương trình con đó và bộ xử lý cất bộ đếm chương trình PC vào ngăn xếp và bắt đầu nạp lệnh vào vị trí mới. Sau khi kết thúc thực hiện chương trình con thì lệnh trở về RET chuyển điều khiển về cho nguồn gọi. Mỗi chương trình con cần lệnh RET như là lệnh cuối cùng.

Cần phải lưu ý các điểm sau đây:

1. Chương trình con DELAY khi thực hiện lệnh "LCALL DELAY" đầu tiên thì địa chỉ của lệnh ngay kế nó là "MOV A, #0AAH" được đẩy vào ngăn xếp và 8051 bắt đầu thực hiện các lệnh ở địa chỉ 300H.
2. Trong chương trình con DELAY, lúc đầu bộ đếm R5 được đặt về giá trị 255 (R5 = FFH). Do vậy, vòng lặp được lặp lại 256 lần. Khi R5 trở về 0 điều khiển rơi xuống lệnh quay trở về RET mà nó kéo địa chỉ từ ngăn xếp vào bộ đếm chương trình và tiếp tục thực hiện lệnh sau lệnh gọi CALL.

Ví dụ :

Hãy viết một chương trình để chốt tất cả các bit của cổng P1 bằng cách gửi đến nó giá trị 55H và AAH liên tục. Hãy đặt một độ trễ thời gian giữa mỗi lần xuất dữ liệu tới cổng P1. Chương trình này sẽ được sử dụng để kiểm tra các cổng của 8051 trong chương tiếp theo.

Lời giải:

```
            ORG   0000
BACK:       MOV   A, #55H          ; Nạp A với giá trị 55H
            MOV   P1, A            ; Gửi 55H đến cổng P1
            LCALL       DELAY      ; Tạo trễ thời gian
            MOV   A, #0AAH         ; Nạp A với giá trị  AAH
            MOV   P1, A            ; Gửi AAH đến cổng P1
            LCALL       DELAY      ; Giữ chậm
            SJMP  BACK             ; Lặp lại vô tận
; ------------------ - Đây là chương trình con tạo độ trễ thời gian
            ORG   300H             ; Đặt trễ thời gian ở địa chỉ 300H
DELAY:      MOV   R5, #00H         ; Nạp bộ đếm R5 = 255H (hay FFH)
AGAIN:      DJNZ  R5, AGAIN        ; Tiếp tục cho đến khi R5 = 0
            RET                    ; Trả điều khiển về nguồn gọi (khi R5 = 0)
            END                    ; Kết thúc tệp tin của hợp ngữ
```

4.3 CẤU TRÚC CHUNG CHƯƠNG TRÌNH HỢP NGỮ CHO 8051

4.3.1. Các thành phần cơ bản của ngôn ngữ Assembly

Cấu trúc của một lệnh hợp ngữ có 4 trường như sau:

[nhãn:] [từ gợi nhớ] [các toán hạng] [; chú giải]

Các trường trong dấu ngoặc vuông là tuỳ chọn và không phải dòng lệnh nào cũng có chúng. Các dấu ngoặc vuông không được viết vào. Với dạng thức trên đây cần lưu ý các điểm sau:

a. Trường nhãn cho phép chương trình tham chiếu đến một dòng lệnh bằng tên. Nó không được viết quá một số ký tự nhất định. Hãy kiểm tra quy định này của hợp ngữ mà ta sử dụng.

b. Từ gợi nhớ (lệnh) và các toán hạng là các trường kết hợp với nhau thực thi công việc thực tế của chương trình và hoàn thiện các nhiệm vụ mà chương trình được viết cho chúng. Trong hợp ngữ các câu lệnh như:

“ ADD A, B”

“MOV A, #67H”

thì ADD và MOV là những từ gợi nhớ tạo ra mã lệnh, còn “A, B” và “A, #67H” là những toán hạng thì hai trường có thể chứa các lệnh giả hoặc chỉ lệnh của hợp ngữ. Hãy nhớ rằng các chỉ lệnh không tạo ra mã lệnh nào (mã máy) và chúng chỉ dùng bởi hợp ngữ, ngược lại đối với các lệnh là chúng được dịch ra mã máy (mã lênh) cho CPU thực hiện.

c. Trường chú giải luôn phải bắt đầu bằng dấu chấm phẩy (;). Các chú giải có thể bắt đầu ở đầu dòng hoặc cuối dòng. Hợp ngữ bỏ qua các chú giải nhưng chúng lại rất cần thiết đối với lập trình viên. Mặc dù các chú giải là tuỳ chọn, không bắt buộc nhưng ta nên dùng chúng để mô tả chương trình để giúp cho người khác đọc và hiểu chương trình dễ dàng hơn.

d. Lưu ý đến nhãn HERE trong trường nhãn của ví dụ mẫu, một nhãn bất kỳ tham chiếu đến một lệnh phải có dấu hai chấm (:) đứng ở sau. Trong câu lệnh nhảy ngắn SJMP thì 8051 được ra lệnh ở lại trong vòng lặp này vô hạn. Nếu hệ thống của chúng ta có một chương trình giám sát thì takhông cần dòng lệnh này và nó có thể được xoá đi ra khỏi chương trình.

Để có thể dịch thành file mã máy dạng HEX-Code trước khi download vào Chip thì một chương trình assembly phải tuân thủ các nguyên tắc sau:

- Mỗi dòng lệnh không vượt quá 255 ký tự
- Mỗi dòng lệnh phải bắt đầu bằng 1 ký tự, nhãn, lệnh hoặc chỉ thị định hướng

- chương trình dịch
- Mọi thứ sau dấu ";" được xem là lời giải thích và chương trình dịch sẽ bỏ qua.
- Các thành phần của mỗi dòng lệnh cách biệt nhau ít nhất bằng một dấu cách.

4.3.2. Khai báo trong lập trình hợp ngữ cho 8051

- Khai báo biến

Ten_bien **DB** Gia_Tri_Khoi_Tao

DB là một chỉ lệnh dữ liệu được sử dụng rộng rãi nhất trong hợp ngữ. Nó được dùng để định nghĩa dữ liệu 8 bit. Khi DB được dùng để định nghĩa byte dữ liệu thì các số có thể ở dạng thập phân, nhị phân, Hex hoặc ở dạng thức ASCII. Đối với dữ liệu thập phân thì cần đặt chữ "D" sau số thập phân, đối với số nhị phân thì đặt chữ "B" và đối với dữ liệu dạng Hex thì cần đặt chữ "H".

Khi dữ liệu có kích thước là 2 byte sử dụng: **DW** để khai báo biến kiểu nguyên

Ví dụ:

```
DATA1:      DB 2D              ; Số thập phân
DATA2:      DB 00110101B       ; Số nhị phân (35 ở dạng Hex)
DATA3:      DB 39H             ; Số dạng Hex
DATA4 DB    "Ky thuat may tinh"  ; Các ký tự ASCII
```

- **Khai báo hằng**

Ten_Hang **EQU** Gia_tri

Được dùng để định nghĩa một hằng số mà không chiếm ngăn nhớ nào. Chỉ lệnh EQU không dành chỗ cất cho dữ liệu nhưng nó gắn một giá trị hằng số với nhãn dữ liệu sao cho khi nhãn xuất hiện trong chương trình giá trị hằng số của nó sẽ được thay thế đối với nhãn

Ví dụ:

```
COUNT      EQU 25
MOV  R3, # COUNT
```

Khi thực hiện lện "MOV R3, #COUNT" thì thanh ghi R3 sẽ được nạp giá trị 25 (chú ý đến dấu #). Vậy ưu điểm của việc sử dụng EQU là gì? Giả sử có một hằng số (một giá trị cố định) được dùng trong nhiều chỗ khác nhau trong chương trình và lập trình viên muốn thay đổi giá trị của nó trong cả chương trình. Bằng việc sử dụng chỉ lệnh EQU ta có thể thay đổi một lần và hợp ngữ sẽ thay đổi tất cả mọi lần xuất hiện của nó.

- **Các toán tử**

Ký hiệu	Thực hiện	Ví dụ	Kết quả
+	Cộng	10+5	15
-	Trừ	25-17	8
*	Nhân	7*4	28
/	Chia nguyên	7/4	1
MOD	Chia lấy dư	7 MOD 4	3
SHR	Dịch phải	1000B SHR 2	0010B
SHL	Dịch trái	1010B SHL 2	101000B
NOT	Đảo	NOT 1	1111111111111110B
AND	And bit	1101B AND 0101B	0101B
OR	Or bit	1101B OR 0101B	1101B
XOR	Xor	1101B XOR 0101B	1000B
LOW	Lấy byte thấp	LOW(0AADDH)	0DDH
HIGH	Lấy byte cao	HIGH(0AADDH)	0AAH
EQ, =	So sánh bằng	7 EQ 4 or 7=4	0 (false)
NE,<>	SS Không bằng	7 NE 4 or 7<>4	0FFFFH (true)
GT, >	SS lớn hơn	7 GT 4 or 7>4	0FFFFH (true)
GE, >=	SS nhỏ hơn hoặc bằng	7 GE 4 or 7>=4	0FFFFH (true)
LT, <	SS nhỏ hơn	7 LT 4 or 7<4	0 (false)
LE,<=	SS nhỏ hơn hoặc bằng	7 LE 4 or 7<=4	0 (false)

Bảng 4-6. Các toán tử

- **Tên**

Thay vì phải nhớ tên từng thanh ghi, hay từng bit, ta có thể gán cho nó một cái nhã gợi nhớ tương ưngs với chức năng của nó, assembly hỗ trợ việc đặt tên theo quy tắc sau:

Tên được tổ hợp từ các ký tự (A-Z, a-z), các số (0-9), các ký tự đặc biệt (“?” Và “_”) và không phâ biệt chữ cái và chữ thường.

Độ dài tên tối đa là 255 ký tự, nhưng chỉ 32 ký tự đầu được dùng để phân biệt

Tên phải bắt đầu bằng ký tự.

Không được trùng với các từ khóa sau:

A	AB	ACALL	ADD	JZ	LCALL	LE	LJMP
ADDC	AJMP	AND	ANL	LOW	LT	MOD	MOV
AR0	AR1	AR2	AR3	MOVC	MOVX	MUL	NE
AR4	AR5	AR6	AR7	NOP	NOT	OR	ORG
BIT	BSEG	C	CALL	ORL	PC	POP	PUSH
CJNE	CLR	CODE	CPL	R0	R1	R2	R3
CSEG	DA	DATA	DB	R4	R5	R6	R7
DBIT	DEC	DIV	DJNZ	RET	RETI	RL	RLC
DPTR	DS	DSEG	DW	RR	RRC	SET	SETB
END	EQ	EQU	GE	SHL	SHR	SJMP	SUBB
GT	HIGH	IDATA	INC	SWAP	USING	XCH	XCHD
ISEG	JB	JBC	JC	XDATA	XOR	XRL	XSEG
JMP	JNB	JNC	JNZ	JZ	LCALL	LE	LJMP
LOW	LT	MOD	MOV				

Bảng 4-7. Một số từ khóa của Assembly

4.3.3. Cấu trúc một chương trình hợp ngữ

ORG (Vị trí bắt đầu con trỏ chương trình)

<đoạn chương trình chính>

<các chương trình con>

END .(Kết thúc chương trình)

Ví dụ:

```
ORG 00H          ;(con trỏ chương trình bắt đầu từ 00h)
LJMP  MAIN       ; nhảy tới vị trí có nhãn là MAIN)
; (vị trí bắt đầu chương trình chính MAIN):
ORG  0030H       ; Nhảy qua vùng ngắt
MAIN:
MOV R1,#10       ;(nạp cho R1 giá trị là 10).
LAP1:
DJNZ  R1, LAP1   ; Nếu R1 khác 0 thì giảm R1 và nhảy tới nhãn LAP1
END              ; (Kết thúc chương trình.)
```

Con trỏ: vị trí mà vi điều khiển bắt đầu thực thi tại đó. Thường khi bắt đầu con trỏ có địa chỉ thấp nhất là 00h, tuy nhiên người lập trình cũng có thể quy định cho nó làm việc tại một vị trí bất kỳ

Ví dụ:

```
ORG   00H          ; Bắt đầu tại vị trí 00h
ORG   0030H        ; Bắt đầu tại vị trí 0030h
```

Chương trình con:

```
Nhãn:
  .................
  Các câu lệnh
  .....
RET
```

Ví dụ:

```
ORG  00H
LJMP  MAIN
ORG  0030H
MAIN:
MOV  R1,#10
LCALL  LAP1          ;gọi chương trình con
LAP1:
DJNZ  R1,LAP1
RET                  ; kết thúc chương trình con
END
```

CHƯƠNG 5
BỘ ĐỊNH THỜI, BỘ ĐẾM

5.1. CÁC THANH GHI CƠ SỞ CỦA BỘ ĐỊNH THỜI

Cả hai bộ định thời Timer 0 và Timer 1 đều có độ dài 16 bít được truy cập như hai thanh ghi tách biệt byte thấp và byte cao. Chúng ta sẽ bàn riêng về từng thanh ghi.

5.1.1. Các thanh ghi của bộ Timer 0.

Thanh ghi 16 bít của bộ Timer 0 được truy cập như byte thấp và byte cao. Thanh ghi byte thấp được gọi là TL0 (Timer 0 bow byte) và thanh ghi byte cao là TH0 (Timer 0 High byte). Các thanh ghi này có thể được truy cập như mọi thanh ghi khác chẳng hạn như A, B, R0, R1, R2 v.v... Ví dụ, lệnh "MOV TL0, #4FH" là chuyển giá trị 4FH vào TL0, byte thấp của bộ định thời 0. Các thanh ghi này cũng có thể được đọc như các thanh ghi khác. Ví dụ "MOV R5, TH0" là lưu byte cao TH0 của Timer 0 vào R5.

TH0								TL0							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hình 5-1. Các thanh ghi của bộ Timer 0

5.1.2. Các thanh ghi của bộ Timer 1.

Bộ định thời gian Timer 1 cũng dài 16 bít và thanh ghi 16 bít của nó được chia ra thành hai byte là TL1 và TH1. Các thanh ghi này được truy cập và đọc giống như các thanh ghi của bộ Timer 0 ở trên.

TH1								TL1							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hình 5-2. Các thanh ghi của bộ Timer 1

5.1.3. Thanh ghi TMOD (chế độ của bộ định thời).

Cả hai bộ định thời Timer 0 và Timer 1 đều dùng chung một thanh ghi được gọi là IMOD để thiết lập các chế độ làm việc khác nhau của bộ định thời. 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. Các phép toán này sẽ được bàn dưới đây.

MSB							LSB
GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer1				Timer0			

Hình 5-3. Timer TMOD

a. Các bít M1, M0

Là các bít chế độ của các bộ Timer 0 và Timer 1. Chúng chọn chế độ của các bộ định thời: 0, 1, 2 và 3. Chế độ 0 là một bộ định thời 13, chế độ 1 là một bộ định thời 16 bít và chế độ 2 là bộ định thời 8 bít. Chúng ta chỉ tập chung vào các chế độ thường được sử dụng rộng rãi nhất là chế độ 1 và 2. Chúng ta sẽ sớm khám phá ra các đặc tính củ các chế độ này sau khi khám phần còn lại của thanh ghi TMOD. Các chế độ được thiết lập theo trạng thái của M1 và M0 như sau:

M1	M0	Chế độ	Chế độ hoạt động
0	0	0	Bộ định thời 13 bít gồm 8 bít là bộ định thời/ bộ đếm 5 bít đặt trước
0	1	1	Bộ định thời 16 bít (không có đặt trước)
1	0	2	Bộ định thời 8 bít tự nạp lại
1	1	3	Chế độ bộ định thời chia tách

Bảng 5-1. Chế độ hoạt động của Timer/Counter

b. C/ T (Bộ đếm / Bộ định thời).

Bít này trong thanh ghi TMOD được dùng để quyết định xem bộ định thời được dùng như một máy tạo độ trễ hay bộ đếm sự kiện. Nếu bít C/T = 0 thì nó được dùng như một bộ định thời tạo độ chễ thời gian. Nguồn đồng hồ cho chế độ trễ thời gian là tần số thạch anh của 8051. ở phần này chỉ bàn về lựa chọn này, công dụng của bộ định thời như bộ đếm sự kiện thì sẽ được bàn ở phần kế tiếp.

Ví dụ :

Hãy cho biết chế độ nào và bộ định thời nào đối với các trường hợp sau:

a) MOV TMOD, #01H b) MOV TMOD, #20H c) MOV TMOD, #12H

Lời giải:

Chúng ta chuyển đổi giá trị từ số Hex sang nhị phân và đối chiếu với từng bít trong thanh ghi TMOD 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.

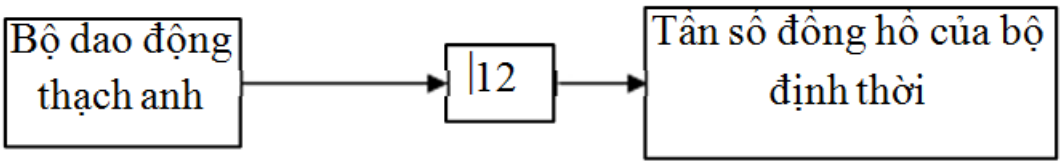
c. Nguồn xung đồng hồ cho bộ định thời

Như chúng ta biết, mỗi bộ định thời cần một xung đồng hồ để giữ nhịp. Vậy nguồn xung đồng hồ cho các bộ định thời trên 8051 lấy ở đâu? Nếu C/T = 0 thì tần số thạch anh đi liền với 8051 được làm nguồn cho đồng hồ của bộ định thời. Điều đó có nghĩa là độ lớn của tần số thạch anh đi kèm với 8051 quyết định tốc độ nhịp của các bộ định thời trên 8051. Tần số của bộ định thời luôn bằng 1/12 tần số của thạch anh gắn với 8051.

Ví dụ:

Đoạn mã dưới đây trình bày tần số thạch anh cho 3 hệ thống dựa trên 8051 khác nhau. Hãy tìm chu kỳ máy của mỗi trường hợp: a) 11.0592MHz b) 16MHz và c) 20MHz.

Lời giải:



a) 11.0592/12 = 921.6kHz; Chu kỳ máy là 1/921.6kHz = 1.085μs (micro giây)

b) 16MHz/12 = 1.333MHz; Chu kỳ máy MC = 1/1.333MHz = 0.75μs

c) 20MHz/12 = 1.66MHz ⇒ MC = 1/1.66MHz = 0.60μs

Mặc dù các hệ thống dựa trên 8051 khác với tần số thạch anh từ 10 đến 40MHz, song ta chỉ tập chung vào tần số thạch anh 11,0592MHz. Lý do đằng sau một số lẻ như vậy là phải làm việc với tần suất baud đối với truyền thông nối tiếp của 8051. Tần số XTAL = 11,0592MHz cho phép hệ 8051 truyền thông với IBM PC mà không có lỗi.

d. Bít cổng GATE.

Một bít khác của thanh ghi TMOD là bít cổng GATE. Để ý trên thanh ghi TMOD ta thấy cả hai bộ định thời Timer0 và Timer1 đều có bít GATE. Vậy bít GATE dùng để làm gì? Mỗi bộ định thời thực hiện điểm khởi động và dừng. Một số bộ định thời thực hiện điều này bằng phần mềm, một số khác bằng phần cứng và một số khác vừa bằng phần cứng vừa bằng phần mềm. Các bộ định thời trên 8051 có cả hai. Việc khởi động và dừng bộ định thời được khởi động bằng phần mềm bởi các bít khởi động bộ định thời TR là TR0 và TR1. Điều này có được nhờ các lệnh "SETB TR1" và "CLR TR1" đối với bộ Timer1 và "SETB TR0" và "CLR TR0" đối với bộ Timer0. Lệnh SETB khởi động bộ định thời và lệnh CLR dùng để dừng nó. Các lệnh này khởi động và dừng các bộ định thời khi bít GATE = 0 trong thanh ghi TMOD. Khởi động và ngừng bộ định thời bằng phần cứng từ nguồn ngoài bằng cách đặt bít GATE = 1 trong

thanh ghi TMOD. Tuy nhiên, để tránh sự lẫn lộn ngay từ bây giờ ta đặt GATE = 0 có nghĩa là không cần khởi động và dừng các bộ định thời bằng phần cứng từ bên ngoài. Để sử dụng phần mềm để khởi động và dừng các bộ định thời phần mềm để khởi động và dừng các bộ định thời khi GATE = 0. Chúng ta chỉ cần các lệnh "SETB TRx" và CLR TRx".

Ví dụ:

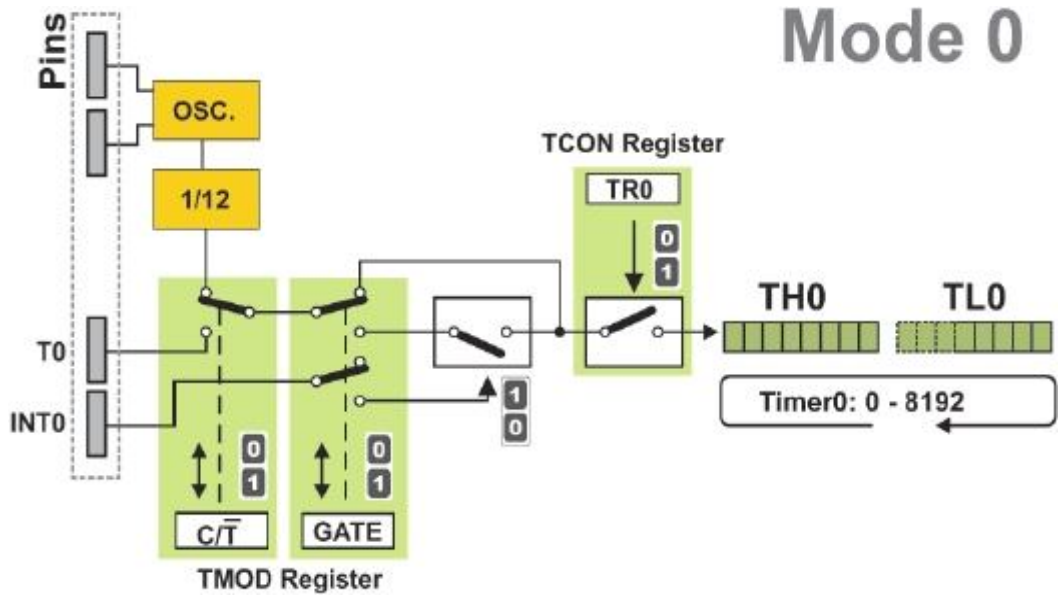
Tìm giá trị cho TMOD nếu ta muốn lập trình bộ Timer0 ở chế độ 2 sử dụng thạch anh XTAL 8051 làm nguồn đồng hồ và sử dụng các lệnh để khởi động và dừng bộ định thời.

Lời giải:

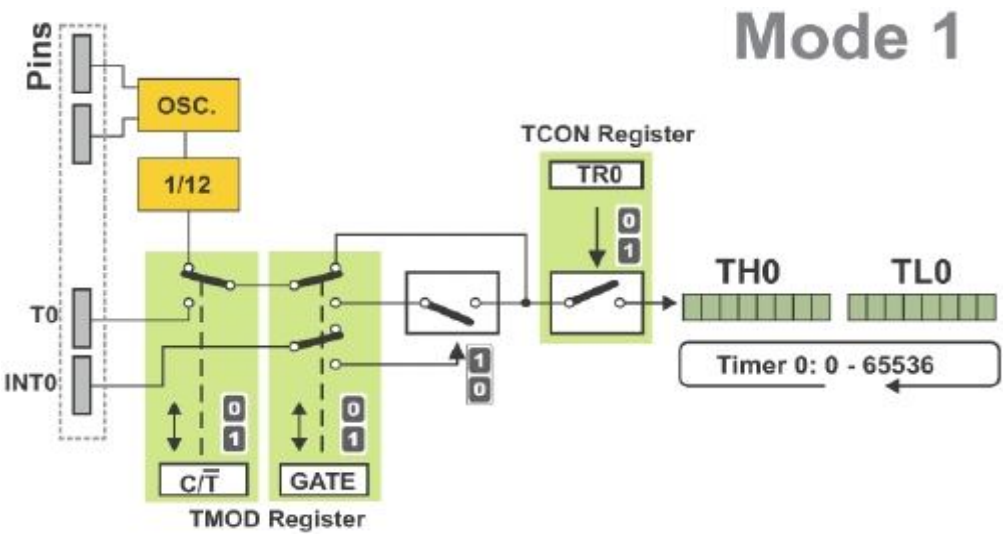
TMOD = 0000 0010: Bộ định thời Timer0, chế độ 2 C/T = 0 dùng nguồn XTAL GATE = 0 để dùng phần mềm trong để khởi động và dừng bộ định thời.

5.2. CÁC CHẾ ĐỘ CỦA BỘ ĐẾM / ĐỊNH THỜI (Timer Mode)

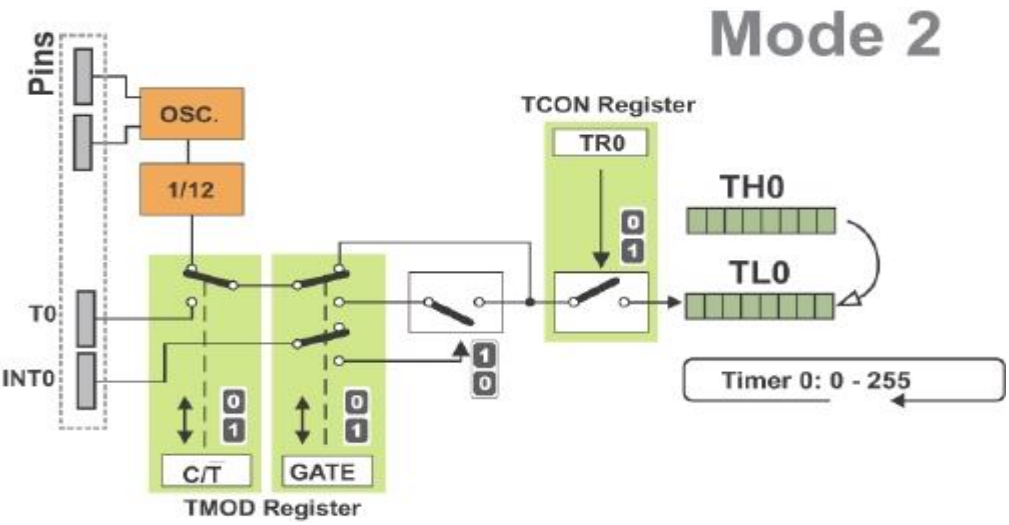
Như vậy, bây giờ chúng ta đã có hiểu biết cơ bản về vai trò của thanh ghi TMOD, chúng ta sẽ xét chế độ của bộ định thời và cách chúng được lập trình như thế nào để tạo ra một độ trễ thời gian. Do chế độ 1 và chế độ 2 được sử dụng rộng rãi nên ta đi xét chi tiết từng chế độ một.



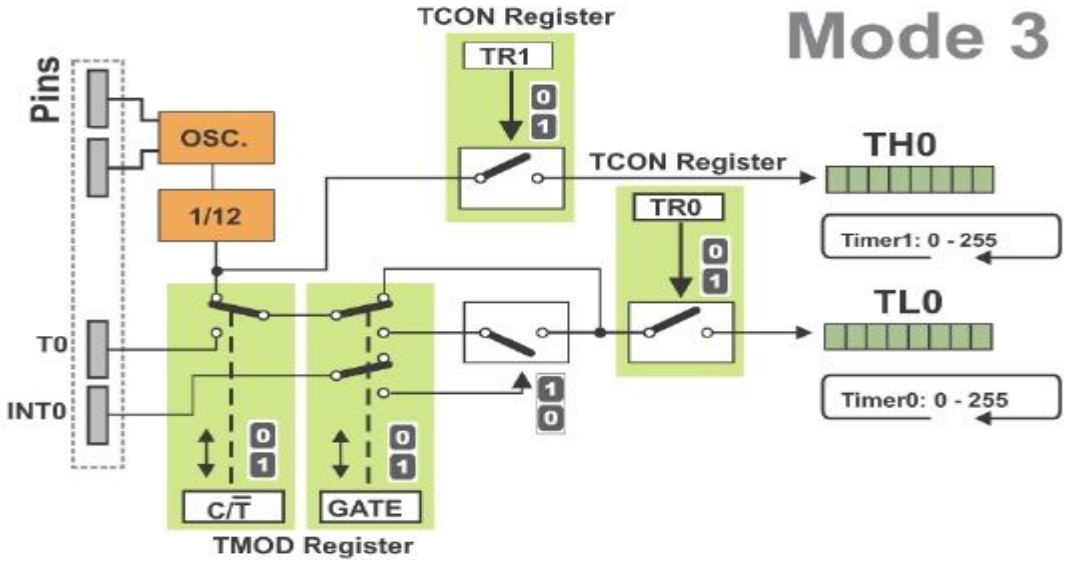
Hình 5-4. Timer 0 – Mode 0



Hình 5-5. Timer 0 – Mode 1



Hình 5-6. Timer 0 – Mode 2



Hình 5-7. Timer 0 – Mode 3

5.3. 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 (TLx ITHx) tràn và set cờ báo tràn (TFx) lên 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:

Trong chương trình dưới đây ta tạo ra một sóng vuông với độ đầy xung 50% (cùng tỷ lệ giữa phần cao và phần thấp) trên chân P1.5. Bộ định thời Timer0 được dùng để tạo độ trễ thời gian. Hãy phân tích chương trình này.

```
      MOV   TMOD, #01         ; Sử dụng Timer0 và chế độ 1(16 bít)
 HERE:      MOV   TL0, #0F2H        ; TL0 = F2H, byte thấp
            MOV   TH0, #0FFH        ; TH0 = FFH, byte cao
            CPL   P1.5              ; Sử dụng chân P1.5
            ACALL DELAY
            SJMP  HERE              ; Nạp lại TH, TL
 ;_____                               delay using timer0.
 DELAY:
SETB  TR0               ; Khởi động bộ định thời Timer0
 AGAIN:     JNB   TF0, AGAIN        ; Hiển thị cờ bộ định thời cho đến
                                    ; khi nó vượt  qua FFFFH.
            CLR   TR0               ; Dừng bộ Timer
            CLR   TF0               ; Xoá cờ bộ định thời 0
            RET
```

Lời giải:

Trong chương trình trên đây ta lưu ý đến đích của SJMP. Ở chế độ 1 chương trình phải nạp lại thanh ghi. TH và TL mỗi lần nếu ta muốn có sóng dạng liên tục.

Dưới đây là kết quả tính toán:

Vì FFFFH - 7634H = 89CBH + 1 = 89CCH và 90CCH = 35276 là số lần đếm xung đồng hồ, độ trễ là 35276 · 1.085μs = 38274ms

Cũng để ý rằng phần cao và phần thấp của xung sóng vuông là bằng nhau.

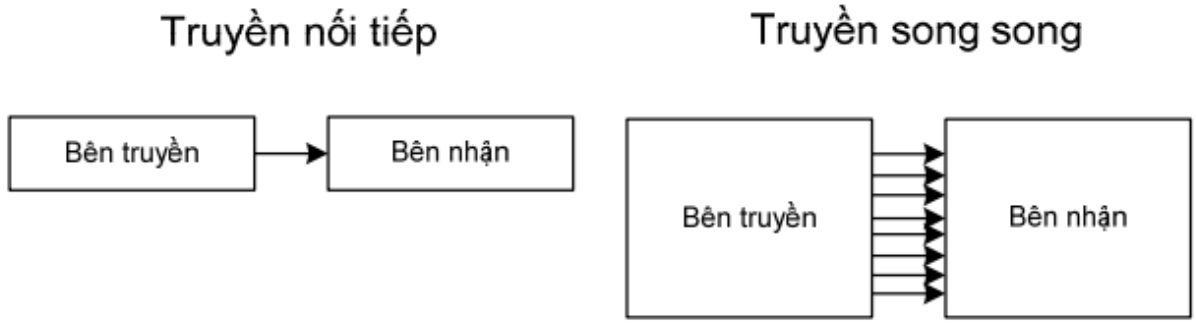
Trong tính toán trên đây là chưa kể đến tổng phí các lệnh vòng lặp.

CHƯƠNG 6
TRUYỀN THÔNG NỐI TIẾP

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 dudx 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. Truyền thông nối tiếp của 8051 là chủ đề của chương này. 8051 đã được cài sẵn khả năng truyền thông nối tiếp, do vậy có thể truyền nhánh dữ liệu với chỉ một số ít dây dẫn.

6.1. CÁC CƠ SỞ CỦA TRUYỀN THÔNG NỐI TIẾP

Khi một bộ vi xử lý truyền thông với thế giới bên ngoài thì nó cấp dữ liệu dưới dạng từng khúc 8 bít (byte) một. Trong một số trường hợp chẳng hạn như các máy in thì thông tin đơn giản được lấy từ đường bus dữ liệu 8 bít và được gửi đi tới bus dữ liệu 8 bít của máy in. Điều này có thể làm việc chỉ khi đường cáp bus không quá dài vì các đường cáp dài làm suy giảm thậm chí làm méo tín hiệu. Ngoài ra, đường dữ liệu 8 bít giá thường đắt. Vì những lý do này, việc truyền thông nối tiếp được dùng để truyền dữ liệu giữa hai hệ thống ở cách xa nhau hàng trăm đến hàng triệu dặm. “Hình 3-24. Truyền thông” là sơ đồ truyền nối tiếp so với sơ đồ truyền song song.



Hình 6-1. Truyền thông

Thực tế là trong truyền thông nối tiếp là một đường dữ liệu duy nhất được dùng thay cho một đường dữ liệu 8 bít của truyền thông song song làm cho nó không chỉ rẻ hơn rất nhiều mà nó còn mở ra khả năng để hai máy tính ở cách xa nhau có truyền

thông qua đường thoại.

Đối với truyền thông nối tiếp thì để làm được các byte dữ liệu phải được chuyển đổi thành các bít nối tiếp sử dụng thanh ghi giao dịch vào - song song - ra - nối tiếp. Sau đó nó có thể được truyền quan một đường dữ liệu đơn. Điều này cũng có nghĩa là ở đầu thu cũng phải có một thanh ghi vào - nối tiếp - ra - song song để nhận dữ liệu nối tiếp và sau đó gói chúng thành từng byte một. Tất nhiên, nếu dữ liệu được truyền qua đường thoại thì nó phải được chuyển đổi từ các số 0 và 1 sang âm thanh ở dạng sóng hình sin. Việc chuyển đổi này thực thi bởi một thiết bị có tên gọi là Modem là chữ viết tắt của "Modulator/ demodulator" (điều chế/ giải điều chế).

Khi cự ly truyền ngắn thì tín hiệu số có thể được truyền như nói ở trên, một dây dẫn đơn giản và không cần điều chế. Đây là cách các bàn PC và IBM truyền dữ liệu đến bo mạch mẹ. Tuy nhiên, để truyền dữ liệu đi xa dùng các đường truyền chẳng hạn như đường thoại thì việc truyền thông dữ liệu nối tiếp yêu cầu một modem để điều chế (chuyển các số 0 và 1 về tín hiệu âm thanh) và sau đó giải điều chế

Trong RS232 thì mức 1 được biểu diễn bởi - 3v đến 25v trong khi đó mức 0 thì ứng với điện áp + 3v đến +25v làm cho điện áp - 3v đến + 3v là không xác định. Vì lý do này để kết nối một RS232 bất kỳ đến một hệ vi điều khiển thì ta phải sử dụng các bộ biến đổi điện áp như MAX232 để chuyển đổi các mức lô-gíc TTL về mức điện áp RS232 và ngược lại.

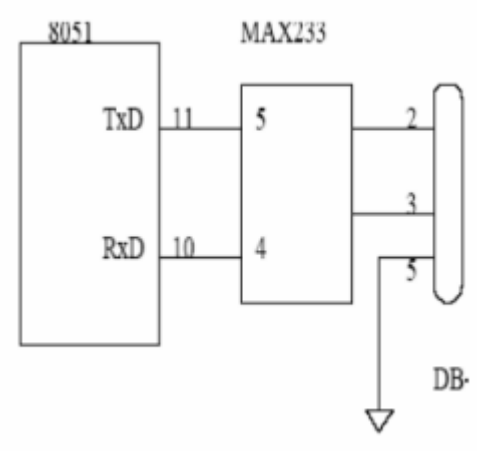
8051 có hai chân được dùng chuyên cho truyền và nhận dữ liệu nối tiếp. Hai chân này được gọi là TxD và RxD và là một phần của cổng P3 (đó là P3.0 và P3.1). chân 11 của 8051 là P3.1 được gán cho TxD và chân 10 (P3.0) được dùng cho RxD. Các chân này tương thích với mức lô-gích TTL. Do vậy chúng đòi hỏi một bộ điều khiển đường truyền để chúng tương thích với RS232. Một bộ điều khiển như vậy là chíp MAX232.

Trong phần này chúng ta sẽ nghiên cứu về các thanh ghi truyền thông nối tiếp của 8051 và cách lập trình chúng để truyền và nhận dữ liệu nối tiếp. Vì các máy tính IBM PC và tương thích được sử dụng rất rộng rãi để truyền thông với các hệ dựa trên 8051, do vậy ta chủ yếu tập trung vào truyền thông nối tiếp của 8051 với cổng COM của PC. Để cho phép truyền dữ liệu giữa máy tín PC và hệ thống 8051 mà không có bất kỳ lỗi nào thì chúng ta phải biết chắc rằng tốc độ baud của hệ 8051 phải phù hợp với tốc độ baud của công COM máy tính PC.

Các thanh ghi cần nghiên cứu: **SCON, SBUF, TMOD, TH1, TL1, ...**

8051 có 1 cổng UART làm việc ở chuẩn TTL, mặc định sau khi khởi động tất các cổng của 8051 đều làm việc ở chế độ vào ra số, vì thế để có thể sử dụng UART

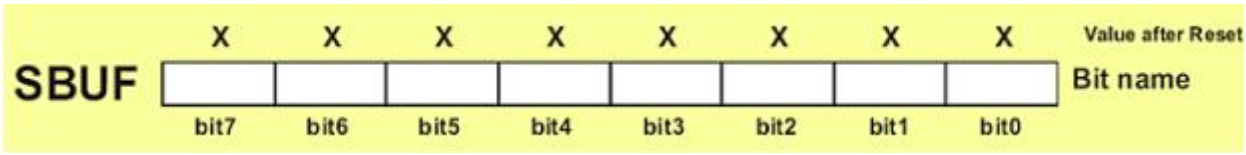
cần phải cấu hình cho cổng này làm việc thông qua các thanh ghi điều khiển và ghép nối tương thích với chuẩn rs232.



Hình 6-2. Ghép nối RS232 với 8051

6.2. CÁC THANH GHI ĐIỀU KHIỂN TRUYỀN THÔN

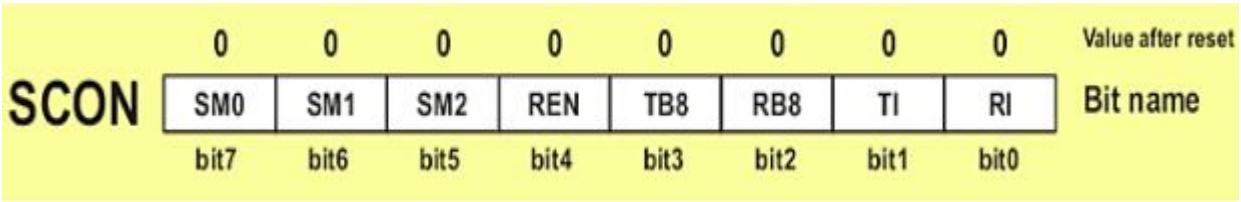
6.2.1. SBUF: Vùng đệm truyền thông dữ liệu ra/vào cổng nối tiếp.



Hình 6-3. Thanh ghi SBUF

- Việc truyền dữ liệu tương ứng với việc nạp cho SBUF một giá trị
- Dữ liệu nhận từ RxD cũng được lưu vào SBUF

6.2.2. SCON: Thanh ghi điều khiển hoạt động cổng nối tiếp



Hình 6-4. Thanh ghi SCON

Trong đó:

Bit	Mô tả
SM0	**Lựa chọn mode làm việc**
SM1	
SM2	
REN	= 1: Cho phép nhận = 0: Chỉ truyền
TB8	(=1) Bit truyền thông thứ 8, được sử dụng khi truyền thông ở chế độ 9 bit
RB8	(=1) Bit truyền thông thứ 8, hệ thống sẽ tự đặt nó =1 nếu phát hiện khung truyền là 9bit
TI	Cờ ngắt truyền. Khi một byte trong SBUF được truyền thành công thì TI=1. Trước khi truyền byte khác bit này cần phải được xóa bằng phần mềm
RI	Cờ ngắt nhận, Khi nhần thành công 1 byte vào SBUF thì RI=1. Sau khi đọc SUBF, RI cần phaiđược xóa bằng phần mềm

Bảng 6-1. Các bít của thanh SCON

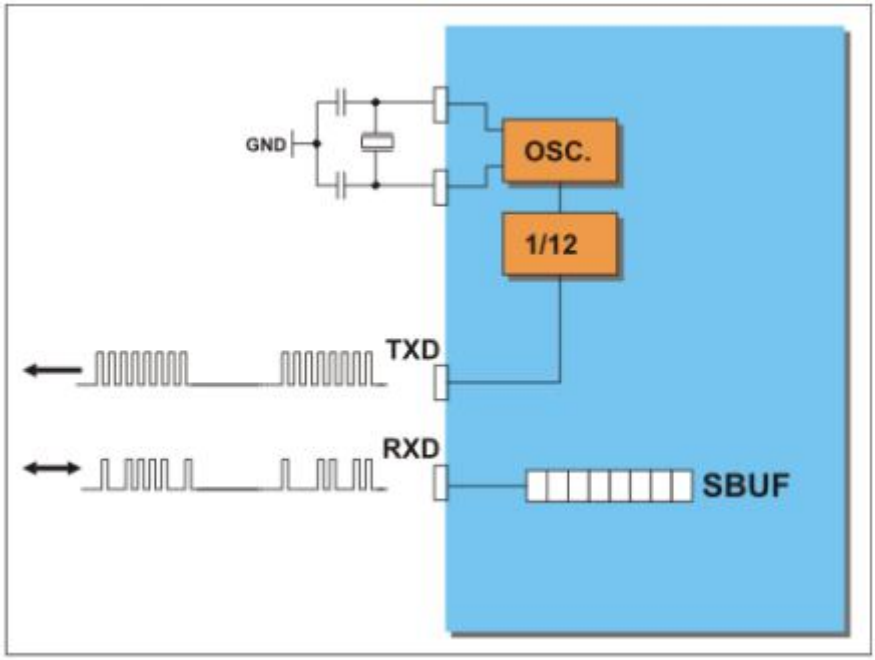
6.3. LỰA CHỌN CHẾ ĐỘ TRUYỀN THÔNG

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Thanh ghi dịch 8 bit	1/12 tần số clock
0	1	1	8-bit UART	Cấu hình qua timer1
1	0	2	9-bit UART	1/32 tần số clock (hoặc 1/64)
1	1	3	9-bit UART	Cấu hình qua timer 1

Bảng 6-2. Lựa chọn chế độ làm việc

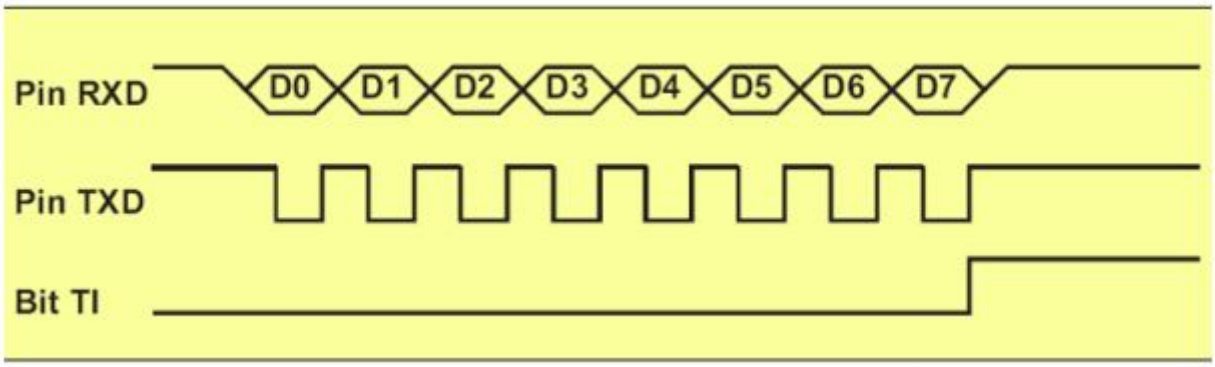
6.3.1. Mode 0

Đây là chế độ thanh ghi dịch 8 bit, không có bit start/stop, ở chế độ này RxD là chân truyền nhận, còn TxD phát xung đồng bộ.



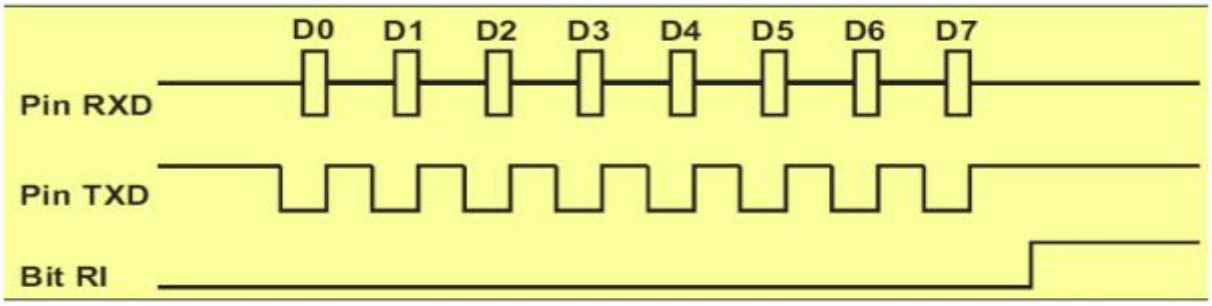
Hình 6-5. Truyền thông nối tiếp – Mode 0

- Quá trình truyền bắt đầu khi ghi giá trị vào SBUF, kết thúc được báo qua TI



Hình 6-6. Giản đồ thời gian truyền nối tiếp – Mode 0

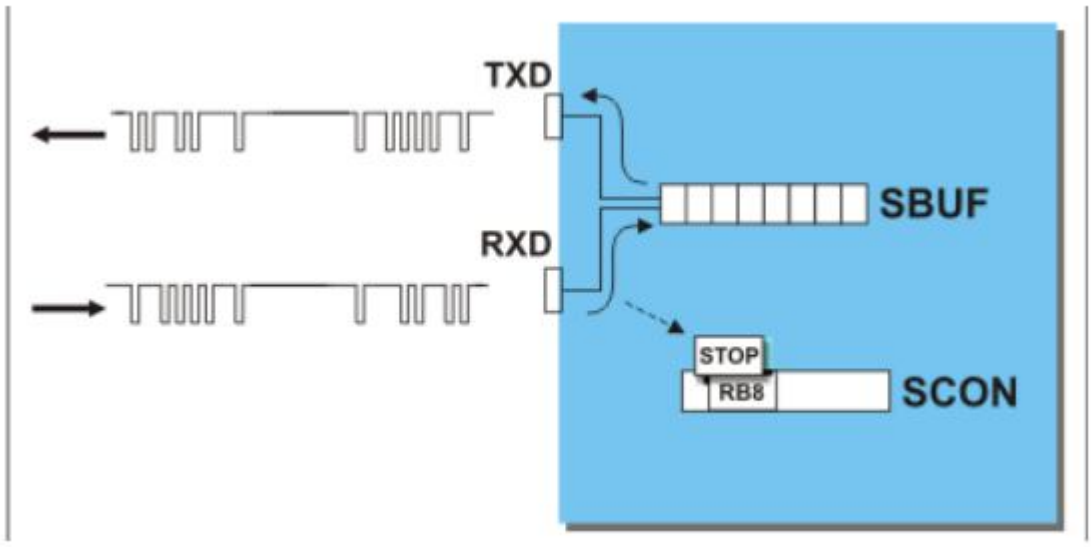
- Quá trình nhận tự động bởi hệ thống và kết thúc khi RI=1



Hình 6-7. Giản đồ thời gian nhận nối tiếp – Mode 0

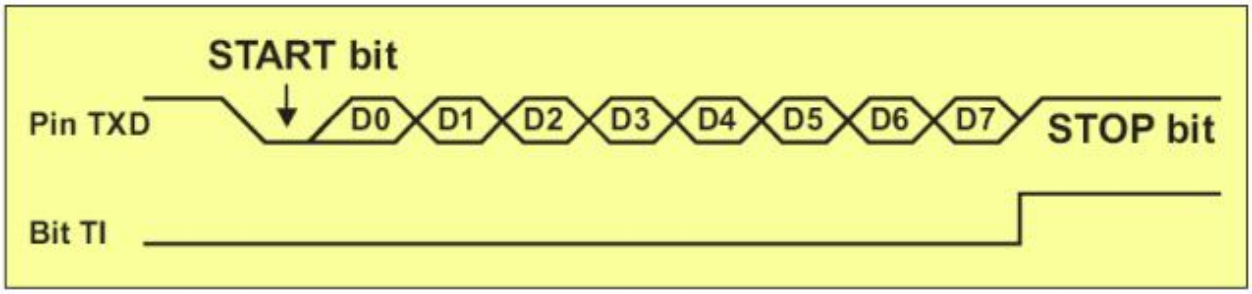
6.3.2. Mode 1

Truyền thông bất đồng bộ với frame truyền 10 bit, gồm 1 start, 8 bit dữ liệu và 1 stop. TxD thực hiện truyền, RxD nhận dữ liệu, tốc độ truyền cài đặt qua Timer 1



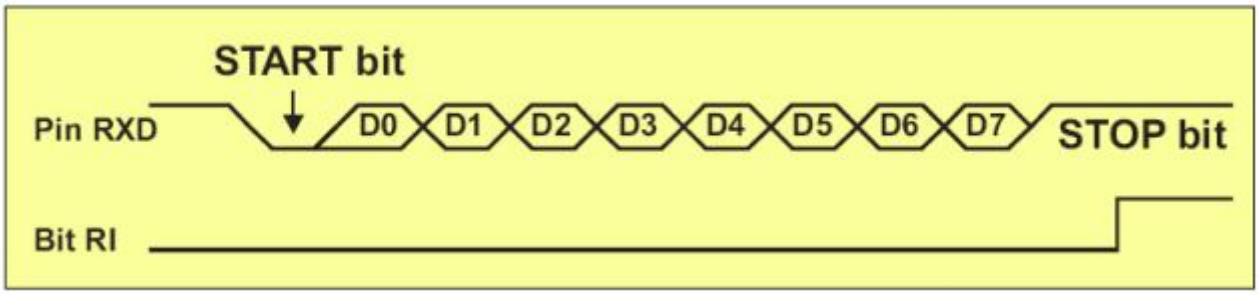
Hình 6-8. Truyền nhận nối tiếp – Mode 1

- Quá trình truyền:



Hình 6-9. Giản đồ thời gian truyền nối tiếp – Mode 1

- Quá trình nhận



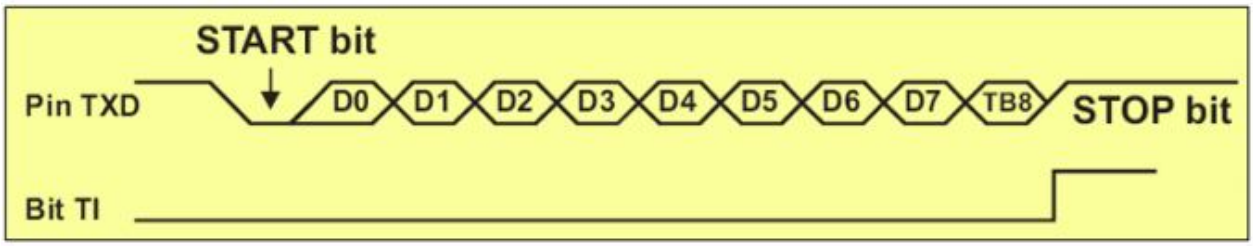
Hình 6-10. Giản đồ thời gian nhận nối tiếp – Mode 1

6.3.3. Mode 2

Truyền thông bất đồng bộ với frame truyền 11 bit, gồm 1 start, 8 bit dữ liệu, 1 bit lập trình được(nêu truyền là TB8, nhận là RB8) và 1 bit stop. TxD thực hiện

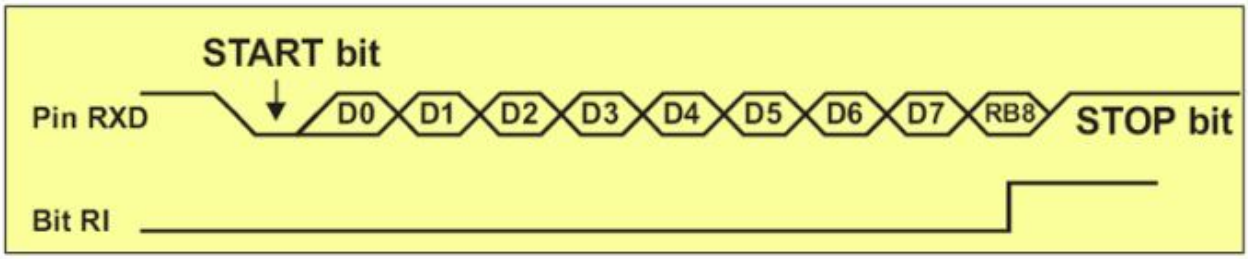
truyền, RxD nhận dữ liệu, tốc độ truyền cài đặt qua Timer 1. Bit thứ 9 thường được dùng là bit phát hiện lỗi party.

- Quá trình truyền



Hình 6-11. Giản đồ thời gian truyền nối tiếp – Mode 2

- Quá trình nhận:



Hình 6-12. Giản đồ thời gian nhận nối tiếp – Mode 2

6.3.4. Mode 3

Mode 3 tương tự mode 2 về mọi mặt ngoại trừ tốc độ baud

Tốc độ Baud

Trong một số hoạt mode động của cổng nối tiếp thì tốc độ baud phụ thuộc vào timer 1. Để cài đặt cần qua các bước sau:

- Cho phép timer 1 hoặt động và cho phép ngắt tràn timer 1
- Cấu hình cho timer 1 làm việc ở chế độ tự nạp lại

Công thức tính:

$$Baud\_Rate = \frac{2^{SMOD}.F_{xtal}}{6^{(1-SPD)}.12.32.\left[256-(BRL)\right]}$$

$$(BRL) = 256 - \frac{2^{SMOD}.F_{xtal}}{6^{(1-SPD)}.12.32.Baud\_Rate}$$

Đặt giá trị cho thanh ghi TH1 tùy thuộc vào tốc độ mong muốn theo bảng dưới :

Baud Rate	Tần số thạch anh					Bit SMOD
	11.0592	12	14.7456	16	20	
150	40 h	30 h	00 h			0
300	A0 h	98 h	80 h	75 h	52 h	0
600	D0 h	CC h	C0 h	BB h	A9 h	0
1200	E8 h	E6 h	E0 h	DE h	D5 h	0
2400	F4 h	F3 h	F0 h	EF h	EA h	0
4800		F3 h	EF h	EF h		1
4800	FA h		F8 h		F5 h	0
9600	FD h		FC h			0
9600					F5 h	1
19200	FD h		FC h			1
38400			FE h			1
76800			FF h			1

Bảng 6-3. Một số giá trị thường dùng trong truyền thông nối tiếp

6.4. MỘT SỐ VÍ DỤ VÀ BÀI TẬP

Ví dụ 1:

Giả sử tần số XTAL = 11.0592MHz cho chương trình dưới đây, hãy phát biểu a) chương trình này làm gì? b) hãy tính toán tần số được Timer1 sử dụng để đặt tốc độ baud? và c) hãy 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 với tần số XTAL cố định
MOV   TMOD, #20H ; Chọn bộ Timer1, chế độ 2, tự động nạp lại
      MOV   TH1, - 3          ; Chọn tốc độ baud 19200
```

```
                              ;(57600/3=19200) vì SMOD = 1
      MOV   SCON, #50H ; Đóng khung dữ liệu gồm 8 bít
                       ;dữ liệu, 1 Stop và cho phép RI.
      SETB  TR1        ; Khởi động Timer1
      MOV   A, #'B'    ; Truyền ký tự B
A_1:  CLR   TI         ; Khẳng định TI = 0
      MOV   SBUF, A    ;Truyền nó
 H_1: JNB   TI, H_1    ; Chờ ở đây cho đến khi bít cuối được gửi đi
      SJMP  A_1        ; 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 chu trình máy, 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

Ví dụ 2:

Tìm giá trị TH1 (ở dạng thập phân và hex) để đạt tốc độ baud cho các trường hợp sau.

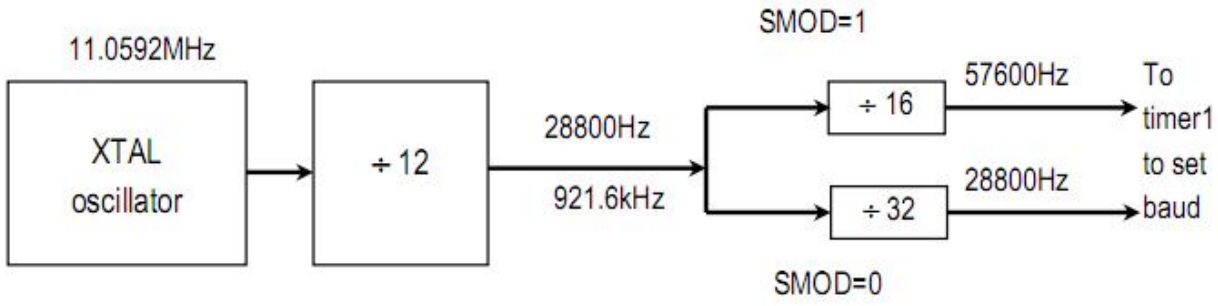
a) 9600 b) 4800 nếu SMOD = 1 và tần số XTAL = 11.0592MHz

Lời giải:

Với tần số XTAL = 11.0592MHz và SMOD = 1 ta có tần số cấp cho Timer1 là 57.6kHz.

a) 57.600/9600 = 6 do vậy TH1 = - 6 hay TH1 = FAH

b) 57.600/4800 = 12 do vậy TH1 = - 12 hay TH1 = F4H



Bài tập:

Hãy tìm tốc độ baud nếu TH1 = -2, SMOD = 1 và tần số XTAL = 11.0592MHz. Tốc độ này có được hỗ trợ bởi các máy tính IBM PC và tương thích không?

CHƯƠNG 7
XỬ LÝ NGẮT

Một ngắt là một sự kiện bên trong hoặc bên ngoài làm ngắt bộ vi điều khiển để báo cho nó biết rằng thiết bị cần dịch vụ của nó. Trong chương này ta tìm hiểu khái niệm ngắt và lập trình ngắt.

Một bộ vi điều khiển có thể phục vụ một vài thiết bị, có hai cách để thực hiện điều này đó là sử dụng các ngắt và thăm dò (polling). Trong phương pháp sử dụng các ngắt thì mỗi khi có một thiết bị bất kỳ cần đến dịch vụ của nó thì nó bao cho bộ vi điều khiển bằng cách gửi một tín hiệu ngắt. Khi nhận được tín hiệu ngắt thì bộ vi điều khiển ngắt tất cả những gì nó đang thực hiện để chuyển sang phục vụ thiết bị.

Chương trình đi cùng với ngắt được gọi là trình dịch vụ ngắt ISR (Interrupt Service Routine) hay còn gọi là trình quản lý ngắt (Interrupt handler). Còn trong phương pháp thăm dò thì bộ vi điều khiển hiển thị liên tục tình trạng của một thiết bị đã cho và điều kiện thoả mãn thì nó phục vụ thiết bị. Sau đó nó chuyển sang hiển thị tình trạng của thiết bị kế tiếp cho đến khi tất cả đều được phục vụ. Mặc dù phương pháp thăm dò có thể hiển thị tình trạng của một vài thiết bị và phục vụ mỗi thiết bị khi các điều kiện nhất định được thoả mãn nhưng nó không tận dụng hết cộng dụng của bộ vi điều khiển. Điểm mạnh của phương pháp ngắt là bộ vi điều khiển có thể phục vụ được rất nhiều thiết bị (tất nhiên là không tại cùng một thời điểm). Mỗi thiết bị có thể nhận được sự chú ý của bộ vi điều khiển dựa trên mức ưu tiên được gán cho nó.

Đối với phương pháp thăm dò thì không thể gán mức ưu tiên cho các thiết bị vì nó kiểm tra tất cả mọi thiết bị theo kiểu hơi vòng. Quan trọng hơn là trong phương pháp ngắt thì bộ vi điều khiển cũng còn có thể che hoặc làm lơ một yêu cầu dịch vụ của thiết bị. Điều này lại một lần nữa không thể thực hiện được trong phương pháp thăm dò. Lý do quan trọng nhất là phương pháp ngắt được ưu chuộng nhất là vì phương pháp thăm dò làm lãng phí thời gian của bộ vi điều khiển bằng cách hỏi dò từng thiết bị kể cả khi chúng không cần đến dịch vụ.

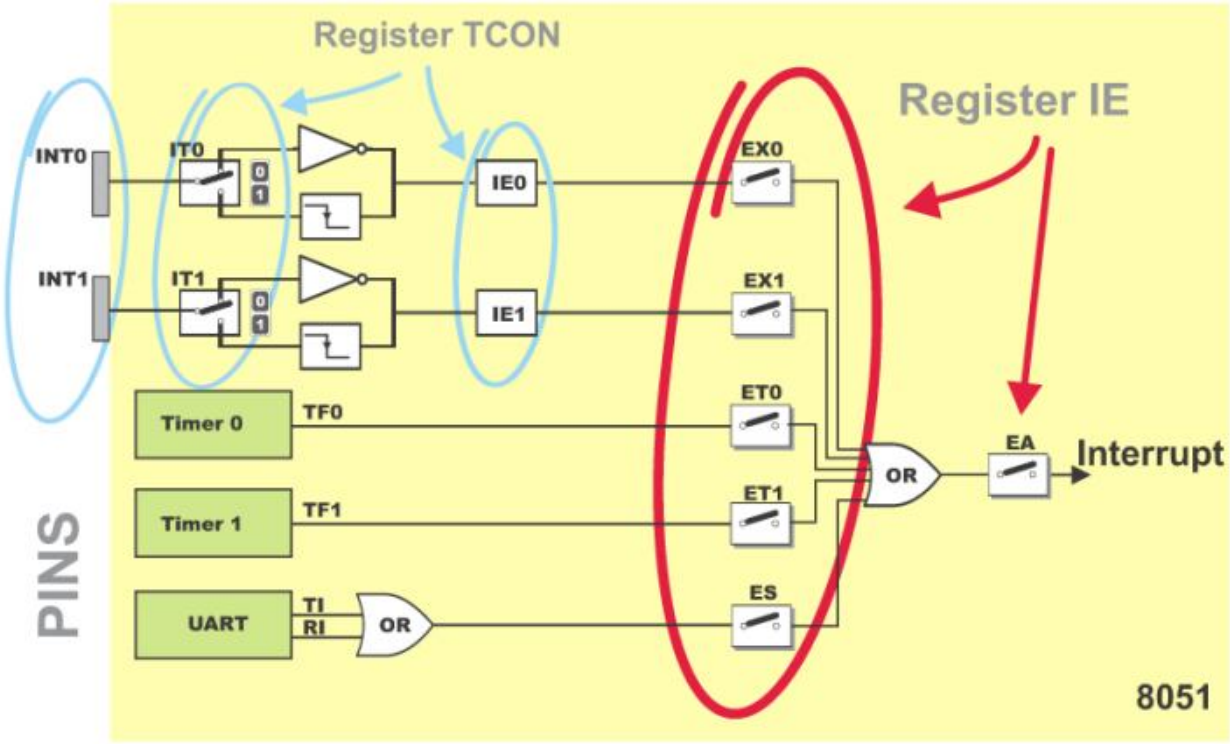
Ví dụ trong các bộ định thời, ta đã dùng lệnh "JNB TF, đích" và đợi cho đến khi bộ định thời quay trở về 0. Trong ví dụ đó, trong khi chờ đợi thì ta có thể làm việc được gì khác có ích hơn, chẳng hạn như khi sử dụng phương pháp ngắt thì bộ vi điều khiển có thể đi làm các việc khác và khi cờ TF bật lên nó sẽ ngắt bộ vi điều khiển cho dù nó đang làm bất kỳ điều gì.

7.1. TRÌNH PHỤC VỤ NGẮT

Đối với mỗi ngắt thì phải có một trình phục vụ ngắt ISR hay trình quản lý ngắt. Khi một ngắt được gọi thì bộ vi điều khiển phục vụ ngắt. Khi một ngắt được gọi thì bộ

vi điều khiển chạy trình phục vụ ngắt. Đối với mỗi ngắt thì có một vị trí cố định trong bộ nhớ để giữ địa chỉ ISR của nó. Nhóm các vị trí nhớ được dành riêng để gửi các địa chỉ của các ISR được gọi là bảng véc tơ ngắt, xem "Hình 3-35. Bảng vector ngắt và ví dụ" 8051 hỗ trợ 5 loại ngắt, có thể cho phép hoặc cấm ngắt với từng loại thông qua thanh ghi điều khiển ngắt IE, hoặc có thể cấm tất cả các ngắt thông qua bit EA.

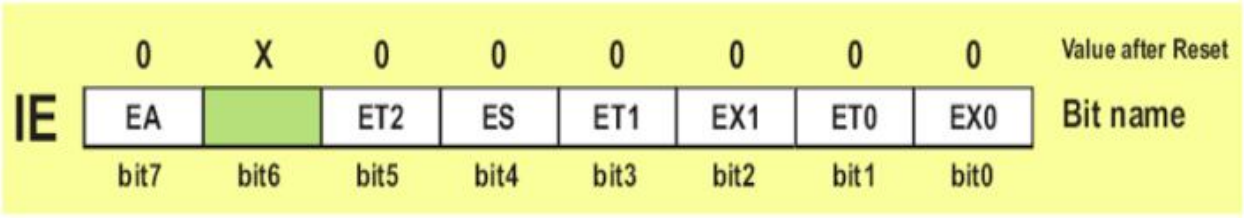
Các tín hiệu điều khiển ngắt có thể được mô tả như hình dưới



Hình 7-1. Các tín hiệu điều khiển ngắt

Ở hình trên chỉ có 1 điểm chú ý đó là hai tín hiệu IT0 và IT1, hai bit này lựa chọn nguyên nhân ngắt cho 2 ngắt ngoài INTR0 và INTR1. Nếu =1 thì ngắt tại sườn âm, =0 ngắt tại sườn dương

Thanh ghi điều khiển ngắt IE



Hình 7-2. Thanh ghi điều khiển ngắt

Trong đó:

Bit	Mô tả
EA	Cho phép/cấm ngắt toàn cục = 0: Cấm tất cả các ngắt = 1: Cho phép các ngắt
ES	= 0: Cấm ngắt truyền thông nối tiếp = 1: Cho phép ngắt truyền thông nối tiếp
ET1	= 0: Cấm ngắt Timer 1 = 1: Cho phép ngắt Timer 1
EX1	= 0: Cấm ngắt ngoại vi INT0 = 1: Cho phép ngắt ngoại vị INT0
ET0	= 0: Cấm ngắt Timer 0 = 1: Cho phép ngắt timer 0
EX0	= 0: Cấm ngắt ngoại vi INT1 = 1: Cho phép

Bảng 7-1. Các bít của thanh ghi diều khiển ngắt

7.2. CÁC BƯỚC KHI THỰC HIỆN MỘT NGẮT

Khi kích hoạt một ngắt bộ vi điều khiển đi qua các bước sau:

1. Nó kết thúc lệnh đang thực hiện và lưu địa chỉ của lệnh kế tiếp (PC) vào ngăn xếp.

2. Nó cũng lưu tình trạng hiện tại của tất cả các ngắt vào bên trong (nghĩa là không lưu vào ngăn xếp).

3. Nó nhảy đến một vị trí cố định trong bộ nhớ được gọi là bảng véc tơ ngắt nới lưu giữ địa chỉ của một trình phục vụ ngắt.

4. Bộ vi điều khiển nhận địa chỉ ISR từ bảng véc tơ 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).

5. 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.

Lưu ý ở bước 5 đến vai trò nhạy cảm của ngăn xếp, vì lý do này mà chúng ta phải cẩn thận khi thao tác các nội dung của ngăn xếp trong ISR. Đặc biệt trong ISR cũng như bất kỳ chương trình con CALL nào số lần đẩy vào ngăn xếp (Push) và số lần lấy ra từ nó (Pop) phải bằng nhau.

Lập trình ngắt

Khi có một ngắt, chương trình chính sẽ bị dừng, con trỏ chương trình ngay lập tức được chuyển đến một địa chỉ quy định sẵn trong bản vector ngắt như hình dưới:

```
ORG 0
rom_start: LJMP
main_code
ORG 13H
int1_vec: LJMP
int1_isr
ORG 30H
main_code:     ;bla bla
; ....
int1_isr:      ;bla bla
```

Interrupt	ROM Location	Pin
Reset	0000H	9
INT0	0003H	P3.2
TF0	000BH	
INT1	0013H	P3.3
TF1	001BH	
S0	0023H	

Bảng 7-2. Bảng vector ngắt và ví dụ

7.3. MỘT SỐ VÍ DỤ VÀ BÀI TẬP

Ví dụ 1:

Hãy chỉ ra những lệnh để a) cho phép ngắt nối tiếp ngắt Timer0 và ngắt phần cứng ngoài 1 (EX1) và b) cấm (che) ngắt Timer0 sau đó c) trình bày cách cấm tất cả mọi ngắt chỉ bằng một lệnh duy nhất.

Lời giải:

a) MOV IE, #10010110B ; Cho phép ngắt nối tiếp, cho phép ngắt Timer0 và cho phép ngắt phần cứng ngoài.

Vì IE là thanh ghi có thể đánh địa chỉ theo bít nên ta có thể sử dụng các lệnh sau đây để truy cập đến các bít riêng rẽ của thanh ghi:

```
SETB IE.7   ; EA = 1, Cho phép tất cả mọi ngắt
SETB IE.4   ; Cho phép ngắt nối tiếp
SETB IE.1   ; Cho phép ngắt Timer1
SETB IE.2   ; Cho phép ngắt phần cứng ngoài 1
```

(tất cả những lệnh này tương đương với lệnh "MOV IE, #10010110B" trên đây).

```
b) CLR  IE.1   ; Xoá (che) ngắt Timer0
c) CLR  IE.7   ; Cấm tất cả mọi ngắt.
```

Ví dụ 2:

Hãy viết chương trình nhân 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 200us trên chân P2.1. Hãy sử

dụng bộ Timer0 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 100/1.085us = 92.

```
; - - Khi khởi tạo vào chương trình main tránh dùng không gian.
; Địa chỉ dành cho bảng véc tơ ngắt.
          ORG   0000H
          CPL   P2.1              ; Nhảy đến bảng véc tơ ngắt.
; - - Trình ISR dành cho Timer0 để tạo ra sóng vuông.
          ORG   0030H       ; Ngay sau địa chỉ bảng véc-tơ ngắt
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
                                  ; Chừng nào bị ngắt bởi TF0
          END
```

Trong ví dụ 2 trình phục vụ ngắt ISR ngắn nên nó có thể đặt vừa vào không gian địa chỉ dành cho ngắt Timer0 trong bảng véc tơ ngắt. Tất nhiên không phải lúc nào cũng làm được như vậy. Xét ví dụ 3 dưới đây.

Ví dụ 3:

Hãy viết lại chương trình ở ví dụ 2 để tạo sóng vuông với mức cao kéo dài 1085us và mức thấp dài 15us với giả thiết tần số XTAL = 11.0592MHz. Hãy sử dụng bộ định thời Timer1.

Lời giải:

Vì 1085us là 1000x1085us 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
          OR6   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 = 1.085us và 14 x 1.085us = 15.19us.

Bài tập:

Viết một chương trình để tạo ra một sóng vuông tần số 50Hz trên chân P1.2. Giả sử XTAL = 11.0592MHz.

7.4. THỨ TỰ ƯU TIÊN NGẮT

Khi có hai hay nhiều ngắt cùng lúc xảy ra, hoặc một ngắt đang thực hiện thì mô ngắt khác yêu cầu thì ngắt nào có độ ưu tiên hơn sẽ được ưu tiên xử lý.

Có 3 cấp độ ưu tiên ngắt trong 8051

- Ngắt reset là ngắt có mức ưu tiên cao nhất, khi reset xảy ra tất cả các ngắt khác và chương trình đều bị dừng và vi điều khiển trở về chế độ khởi dộng ban đầu.
- Ngắt mức 1, chỉ có reset mới có thể cấm ngắt này
- Ngắt mức 0, các ngắt mức 1 và reset có thể cấm ngắt này.

Việc đăt chọn mức ưu tiên ngắt là 1 hoặc 0 thông qua thanh ghi IP. Việc xử lý ưu tiên ngắt của 8051 như sau:

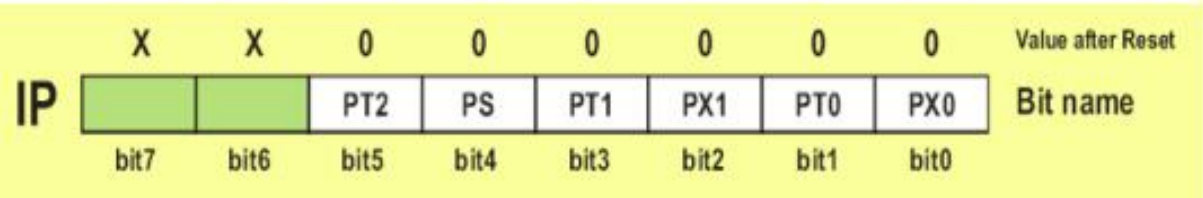
+ Nếu 1 có độ ưu tiên cao hơn một ngắt đang được xử lý xuất hiện thì, ngắt có ưu tiên thấp ngay lập tức bị dừng để ngắt kia được thực hiện

+ Nếu 2 ngắt cùng yêu cầu vào 1 hời điểm thì ngắt có mức ưu tiên hơn sẽ được xử lý trước

+ Nếu 2 ngắt có cùng mức ưu tiên cùng yêu cầu vào 1 thời điểm thì thứ tự được chọn như sau:

- INTR 0
- Timer 0
- INTR 1
- Timer 1
- UART

Thanh ghi IP



Hình 7-3. Thanh ghi IP

Trong đó: Các bit từ 0 đến 5 đặt mức ngắt là 0 hoặc 1 cho các ngắt tương ứng như sau:

PS: UART
PT1: Timer 1
PX1: INTR 1
PT0: Timer 0
PX0: INTR 0

CHƯƠNG 8
PHỐI GHÉP 8051 VỚI THẾ GIỚI THỰC

Chương này khám phá một số ứng dụng của 8051 với thế giới thực. Chúng ta giải thích làm cách nào phối ghép 8051 với các thiết bị như là LCD, ADC và các cảm biến.

8.1. PHỐI GHÉP VỚI LCD

Ở phần này ta sẽ mô tả các chế độ hoạt động của các LCD và sau đó mô tả cách lập trình và phối ghép một LCD tới 8051.

8.1.1. Hoạt động của LCD.

Trong những năm gần đây LCD đang ngày càng được sử dụng rộng rãi thay thế dần cho các đèn LED (các đèn LED 7 đoạn hay nhiều đoạn). Đó là vì các nguyên nhân sau:

1. Các LCD có giá thành hạ.
2. Khả năng hiển thị các số, các ký tự và đồ hoạ tốt hơn nhiều so với các đèn LED (vì các đèn LED chỉ hiển thị được các số và một số ký tự).
3. Nhờ kết hợp một bộ điều khiển làm tươi vào LCD làm giải phóng cho CPU công việc làm tươi LCD. Trong khi đèn LED phải được làm tươi bằng CPU (hoặc bằng cách nào đó) để duy trì việc hiển thị dữ liệu.
4. Dễ dàng lập trình cho các ký tự và đồ hoạ.

8.1.2. Mô tả các chân của LCD.

LCD được nói trong mục này có 14 chân, chức năng của các chân được cho trong bảng 12.1. Vị trí của các chân được mô tả trên bảng 6.1 cho nhiều LCD khác nhau.

1. Chân V_{CC}, V_{SS} và V_{EE}: Các chân V_{CC}, V_{SS} và V_{EE}: Cấp dương nguồn - 5v và đất tương ứng thì V_{EE} được dùng để điều khiển độ tương phản của LCD.
2. Chân chọn thanh ghi RS (Register Select).

Có hai thanh ghi rất quan trọng bên trong LCD, chân RS được dùng để chọn các thanh ghi này như sau: Nếu RS = 0 thì thanh ghi mà lệnh được chọn để cho phép người dùng gửi một lệnh chẳng hạn như xoá màn hình, đưa con trỏ về đầu dòng v.v… Nếu RS = 1 thì thanh ghi dữ liệu được chọn cho phép người dùng gửi dữ liệu cần hiển thị trên LCD.

3. Chân đọc/ ghi (R/W).

Đầu vào đọc/ ghi cho phép người dùng ghi thông tin lên LCD khi R/W = 0 hoặc đọc thông tin từ nó khi R/W = 1.

4. Chân cho phép E (Enable).

Chân cho phép E được sử dụng bởi LCD để chốt thông tin hiện hữu trên chân dữ liệu của nó. Khi dữ liệu được cấp đến chân dữ liệu thì một xung mức cao xuống thấp phải được áp đến chân này để LCD chốt dữ liệu trên các chân dữ liêu. Xung này phải rộng tối thiểu là 450ns.

5. Chân D0 - D7.

Đây là 8 chân dữ liệu 8 bít, được dùng để gửi thông tin lên LCD hoặc đọc nội dung của các thanh ghi trong LCD.

Để hiển thị các chữ cái và các con số, chúng ta gửi các mã ASCII của các chữ cái từ A đến Z, a đến f và các con số từ 0 - 9 đến các chân này khi bật RS = 1.

Cũng có các mã lệnh mà có thể được gửi đến LCD để xoá màn hình hoặc đưa con trỏ về đầu dòng hoặc nhấp nháy con trỏ. Bảng 6.2 liệt kê các mã lênh.

Chúng ta cũng sử dụng RS = 0 để kiểm tra bít cờ bận để xem LCD có sẵn sàng nhân thông tin. Cờ bận là D7 và có thể đượcđọc khi R/W = 1 và RS = 0 như sau:

Nếu R/W = 1, RS = 0 khi D7 = 1 (cờ bận 1) thì LCD bận bởi các công việc bên trong và sẽ không nhận bất kỳ thông tin mới nào. Khi D7 = 0 thì LCD sẵn sàng nhận thông tin mới. Lưu ý chúng ta nên kiểm tra cờ bận trước khi ghi bất kỳ dữ liệu nào lên LCD.

Chân	Ký hiệu	I/O	Mô tả
1	V_{SS}	-	Đất
2	V_{CC}	-	Dương nguồn 5v
3	V_{EE}	-	Cấp nguồn điều khiển phản
4	RS	I	RS = 0 chọn thanh ghi lệnh. RS = 1 chọn thanh dữ liệu
5	R/W	I	R/W = 1 đọc dữ liệu. R/W = 0 ghi
6	E	I/O	Cho phép
7	DB0	I/O	Các bít dữ liệu
8	DB1	I/O	Các bít dữ liệu
9	DB2	I/O	Các bít dữ liệu

10	DB3	I/O	Các bít dữ liệu
11	DB4	I/O	Các bít dữ liệu
12	DB5	I/O	Các bít dữ liệu
13	DB6	I/O	Các bít dữ liệu
14	DB7	I/O	Các bít dữ liệu

Bảng 8-1. Mô tả các chân của LCD.

Mã (Hex)	**Lệnh đến thanh ghi của LCD**
1	Xoá màn hình hiển thị
2	Trở về đầu dòng
4	Giả con trỏ (dịch con trỏ sang trái)
6	Tăng con trỏ (dịch con trỏ sang phải)
5	Dịch hiển thị sang phải
7	Dịch hiển thị sang trái
8	Tắt con trỏ, tắt hiển thị
A	Tắt hiển thị, bật con trỏ
C	Bật hiển thị, tắt con trỏ
E	Bật hiển thị, nhấp nháy con trỏ
F	Tắt con trỏ, nhấp nháy con trỏ
10	Dịch vị trí con trỏ sang trái
14	Dịch vị trí con trỏ sang phải
18	Dịch toàn bộ hiển thị sang trái
1C	Dịch toàn bộ hiển thị sang phải
80	Ép con trỏ Vũ đầu dòng thứ nhất
C0	Ép con trỏ Vũ đầu dòng thứ hai
38	Hai dòng và ma trận 5 × 7

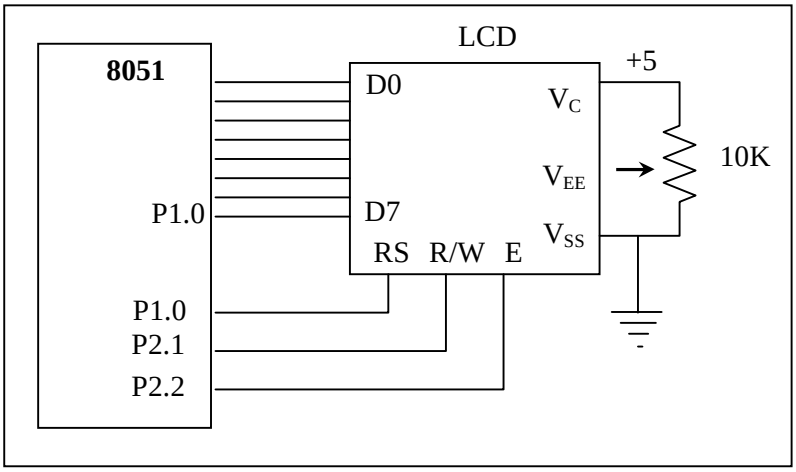
Bảng 8-2. Các mã lệnh LCD.

8.1.3 Gửi các lệnh và dữ liệu đến LCD với một độ trễ.

Để gửi một lệnh bất kỳ từ bảng 6.2 đến LCD ta phải đưa chân RS về 0. Đối với dữ liệu thì bật RS = 1 sau đó gửi một sườn xung cao xuống thấp đến chân E để cho phép chốt dữ liệu trong LCD. Điều này được chỉ ra trong đoạn mã chương trình dưới đây.

```
; gọi độ thời gian trễ trước khi gửi dữ liệu/ lệnh kế tiếp.
; chân P1.0 đến P1.7 được nối tới chân dữ dữ liệu D0 - D7 của  LCD.
; Chân P2.0 được nối tới chân RS của LCD.
; Chân P2.1 được nối tới chân R/W của LCD.
; Chân P2.2 được nối đến chân E của LCD.
        ORG
        MOV  A, # 38H          ; Khởi tạo LCD hai dòng với ma trận 5 × 7
        ACALL COMNWRT          ; Gọi chương trình con lệnh
        ACALL DELAY            ; Cho LCD một độ trễ
        MOV  A, # 0EH          ; Hiển thị màn hình và con trỏ
        ACALL COMNWRT          ; Gọi chương trình con lênh
        ACALL  DELAY           ; Cấp một độ trễ cho LCD
        MOV  AM # 01           ; Xoá LCD
        ACALL COMNWRT          ; Gọi chương trình con lệnh
        ACALL DELAY            ; Tạo độ trễ cho LCD
        MOV  A, # 06H          ; Dịch con trỏ sang phải
        ACALL COMNWRT          ; Gọi chương trình con lệnh
        ACALL DELAY            ; Tạo độ trễ cho LCD
        MOV  AM # 48H          ; Đưa con trỏ về dòng 1 cột 4
        ACALL COMNWRT          ; Gọi chương trình con lệnh
        ACALL DELAY            ; Tạo độ trễ cho LCD
        MOV  A, # “N”          ; Hiển thị chữ N
        ACALL DATAWRT          ; Gọi chương trình con hiển thị DISPLAY
        ACALL DELAY            ; Tạo độ trễ cho LCD
        MOV  AM # “0”          ; Hiển thị chữ 0
        ACALL DATAWRT          ; Gọi DISPLAY
AGAIN:  SJMP  AGAIN            ; Chờ ở đây
COMNWRT:                       ; Gửi lệnh đến LCD
        MOV     P1, A          ; Sao chép thanh ghi A đến cổng P1
        CLR     P2.0           ; Đặt RS = 0 để gửi lệnh
```

```
                CLR     P2.1        ; Đặt R/W = 0 để ghi dữ liệu
                SETB    P2.2        ; Đặt E = 1 cho xung cao
                CLR     P2.2        ; Đặt E = 0 cho xung cao xuống thấp
                RET
DATAWRT:                            ; Ghi dữ liệu ra LCD
                MOV     P1, A       ; Sao chép thanh ghi A đến cổng P1
                SETB    P2.0        ; Đặt RS = 1 để gửi dữ liệu
                CLR     P2.1        ; Đặt R/W = 0 để ghi
                SETB    P2.2        ; Đặt E = 1 cho xung cao
                CLR     P2.2        ; Đặt E = 0 cho xung cao xuống thấp
                RET
DELAY:  MOV  R3, # 50               ; Đặt độ trễ 50µs hoặc cao hơn cho CPU
HERE2:  MOV  R4, # 255              ; Đặt R4 = 255
HERE:   DJNZ R4, HERE               ; Đợi ở đây cho đến khi R4 = 0
        DJNZ R3, HERE2
        RET
        END
```



Hình 8-1. Ghép Nối LCD.

8.1.4. Gửi mã lệnh hoặc dữ liệu đến LCD có kiểm tra cờ bận.

Đoạn chương trình trên đây đã chỉ ra cách gửi các lệnh đến LCD mà không có kiểm tra cờ bận (Busy Flag). Lưu ý rằng chúng ta phải đặt một độ trễ lớn trong quá ảtình xuất dữ liệu hoặc lệnh ra LCD. Tuy nhiên, một cách tốt hơn nhiều là hiển thị cờ bận trước khi xuất một lệnh hoặc dữ liệu tới LCD. Dưới đây là một chương trình như vậy.

```
        ; Kiểm tra cờ bận trước khi gửi dữ liệu, lệnh ra LCD
        ; Đặt P1 là cổng dữ liệu
        ; Đặt P2.0 nối tới cổng RS
        ; Đặt P2.1 nối tới chân R/W
        ; Đặt P2.2 nối tới chân E
              ORG
              MOV  A, # 38H          ; Khởi tạo LCD hai dòng với ma trận 5 × 7
              ACALL COMMAND          ; Xuất lệnh
              MOV  A, # 0EH          ; Dịch con trỏ sang phải
              ACALL COMMAND          ; Xuất lệnh
              MOV  A, # 01H          ; Xoá lệnh LCD
              ACALL COMMAND          ; Xuất lệnh
              MOV  A, # 86H          ; Dịch con trỏ sang phải
              ACALL COMMAND          ; Đưa con trỏ về dòng 1 lệnh 6
              MOV  A, # “N”          ; Hiển thị chữ N
              ACALL DATA DISPLAY
              MOV  A, # “0”          ; Hiển thị chữ 0
              ACALL DATA DISPLAY
HERE:         SJMP  HERE             ; Chờ ở đây
COMMAND:ACALL READY                  ; LCD đã sẵn sàng chưa?
              MOV  P1, A             ; Xuất mã lệnh
              CLR   P2.0             ; Đặt RS = 0 cho xuất lệnh
              CLR   P2.1             ; Đặt R/W = 0 để ghi dữ liệu tới LCD
              SETB P2.2              ; Đặt E = 1 đối với xung cao xuống thấp
              CLR   P2.2             ; Đặt E = 0 chốt dữ liệu
              RET
DATA-DISPLAY:
              ACALL READY            ; LCD đã sẵn sàng chưa?
              MOV  P1, A             ; Xuất dữ liệu
              SETB P2.0              ; Đặt RS = 1 cho xuất dữ liệu
              CLR   P2.1             ; Đặt R/W = 0 để ghi dữ liệu ra LCD
              SETB P2.2              ; Đặt E = 1 đối với xung cao xuống thấp
              CLR   P2.2             ; Đặt E = 0 chốt dữ liệu
              RET
DELAY:
```

```
        SETB P1.7           ; Lấy P1.7 làm cổng vào
        CLR  P2.0           ; Đặt RS = 0 để truy cập thanh ghi lệnh
        SETB P2.1           ; Đặt R/W = 1 đọc thanh ghi lệnh
; Đọc thanh ghi lệnh và kiểm tra cờ lệnh
BACK:   CLR  P2.2           ; E = 1 đối với xung cao xuống thấp
        SETB P2.2           ; E = 0 cho xung cao xuống thấp?
        JB P1.7, BACK       ; Đợi ở đây cho đến khi cờ bận = 0
        RET
        END
```

Lưu ý: Trong chương trình cờ bận D7 của thanh ghi lệnh. Để đọc thanh ghi lệnh ta phải đặt RS = 0, R/W = 1 và xung cao - xuống - thấp cho bít E để cấp thanh ghi lệnh cho chúng ta. Sau khi đọc thanh ghi lệnh, nếu bít D7 (cờ bận) ở mức cao thì LCD bận và không có thông tin (lệnh) nào được xuất đến nó chỉ khi nào D7 = 0 mới có thể gửi dữ liệu hoặc lệnh đến LCD. Lưu ý trong phương phát này không sử dụng độ trễ thời gian nào vì ta đang kiểm tra cờ bận trước khi xuất lệnh hoặc dữ liệu lên LCD.

8.2. PHỐI GHÉP VỚI ADC.

Phần này sẽ khám phá ghép các chíp ADC (bộ chuyển đổi tương tự số) và các cảm biến nhiệt với 8051.

8.2.1. Các thiết bị ADC.

Các bộ chuyển đổi ADC thuộc trong những thiết bị được sử dụng rộng rãi nhất để thu dữ liệu. Các máy tính số sử dụng các giá trị nhị phân, nhưng trong thế giới vật lý thì mọi đại lượng ở dạng tương tự (liên tục). Nhiệt độ, áp suất (khí hoặc chất lỏng), độ ẩm và vận tốc và một số ít trọng những đại lượng vật lý của thế giới thực mà ta gặp hàng ngày. Một đại lượng vật lý được chuyển về dòng điện hoặc điện áp qua một thiết bị được gọi là các bộ biến đổi. Các bộ biến đổi cũng có thể được coi như các bộ cảm biến. Mặc dù chỉ có các bộ cảm biến nhiệt, tốc độ, áp suất, ánh sáng và nhiều đại lượng tự nhiên khác nhưng chúng đều cho ra các tín hiệu dạng dòng điện hoặc điện áp ở dạng liên tục. Do vậy, ta cần một bộ chuyển đổi tương tự số sao cho bộ vi điều khiển có thể đọc được chúng. Một chíp ADC được sử dụng rộng rãi là ADC 804.

8.2.2. Chíp ADC 0804.

Chíp ADC 804 là bộ chuyển đổi tương tự số trong họ các loạt ADC 800 từ hãng National Semiconductor. Nó cũng được nhiều hãng khác sản xuất, nó làm việc với +5v và có độ phân giải là 8 bít. Ngoài độ phân giải thì thời gian chuyển đổi cũng là một yếu tố quan trọng khác khi đánh giá một bộ ADC. Thời gian chuyển đổi được định

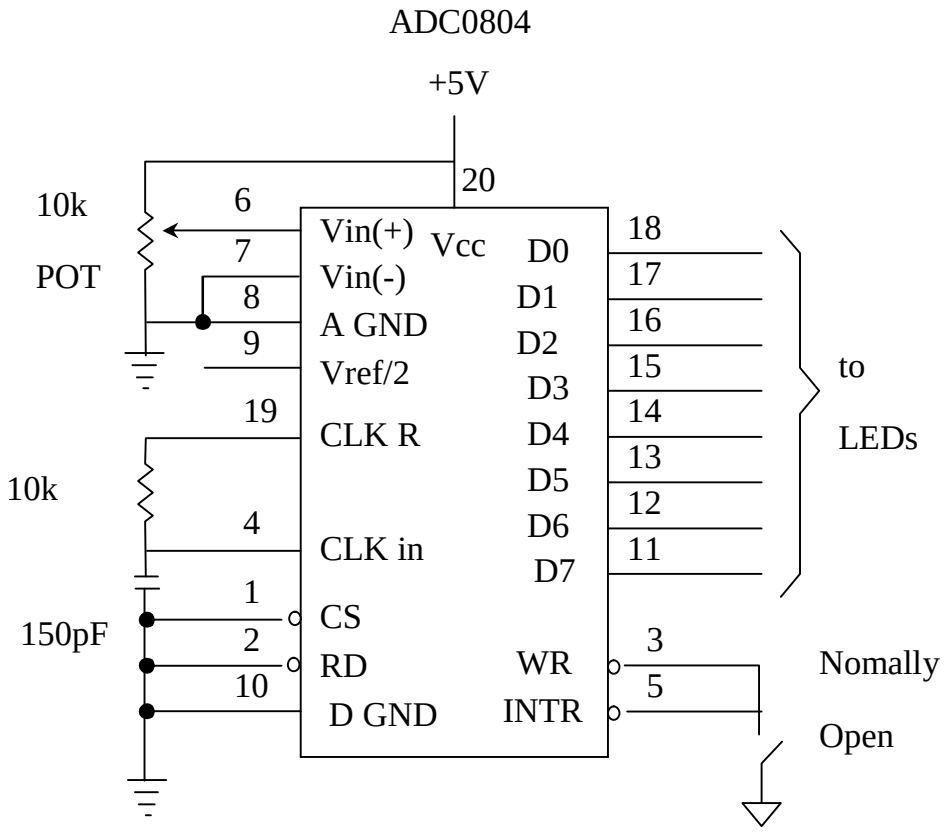
nghĩa như là thời gian mà bộ ADC cần để chuyển một đầu vào tương tự thành một số nhị phân. Trong ADC 804 thời gian chuyển đổi thay đổi phụ thuộc vào tần số đồng hồ được cấp tới chân CLK và CLK IN nhưng không thể nhanh hơn 110µs. Các chân của ADC 804 được mô tả như sau:

1. Chân $\overline{CS}$ - chọn chíp: Là một đầu vào tích cực mức thấp được sử dụng để kích hoạt chíp ADC 804. Để truy cập ADC 804 thì chân này phải ở mức thấp.
2. Chân $\overline{RD}$ (đọc): Đây là một tín hiệu đầu vào được tích cực mức thấp. Các bộ ADC chuyển đổi đầu vào tương tự thành số nhị phân tương đương với nó và giữ nó trong một thanh ghi trong. $\overline{RD}$ được sử dụng để nhận dữ liệu được chuyển đổi ở đầu ra của ADC 804. Khi CS = 0 nếu một xung cao - xuống - thấp được áp đến chân $\overline{RD}$ thì đầu ra số 8 bít được hiển diện ở các chân dữ liệu D0 - D7. Chân $\overline{RD}$ cũng được coi như cho phép đầu ra.
3. Chân ghi $\overline{WR}$ (thực ra tên chính xác là "Bắt đầu chuyển đổi"). Đây là chân đầu vào tích cực mức thấp được dùng để báo cho ADC 804 bắt đầu quá trình chuyển đổi. Nếu CS = 0 khi $\overline{WR}$ tạo ra xung cao - xuống - thấp thì bộ ADC 804 bắt đầu chuyển đổi giá trị đầu vào tương tự V_{in} về số nhị phân 8 bít. Lượng thời gian cần thiết để chuyển đổi thay đổi phụ thuộc vào tần số đưa đến chân CLK IN và CLK R. Khi việc chuyển đổi dữ liệu được hoàn tất thì chân INTR được ép xuống thấp bởi ADC 804.
4. Chân CLK IN và CLK R.

Chân CLK IN là một chân đầu vào được nối tới một nguồn đồng hồ ngoài khi đồng hồ ngoài được sử dụng để tạo ra thời gian. Tuy nhiên 804 cũng có một máy tạo xung đồng hồ. Để sử dụng máy tạo xung đồng hồ trong (cũng còn được gọi là máy tạo đồng hồ riêng) của 804 thì các chân CLK IN và CLK R được nối tới một tụ điện và một điện trở như chỉ ra trên hình 6.4. Trong trường hợp này tần số đồng hồ được xác định bằng biểu thức:

$$f = \frac{1}{1{,}1RC}$$

Giá trị tiêu biểu của các đại lượng trên là R = 10kΩ và C= 150pF và tần số nhận được là f = 606kHz và thời gian chuyển đổi sẽ mất là 110µs.



Hình 8-2. Kiểm tra ADC 0804 ở chế độ chạy tự do.

5. Chân ngắt $\overline{\text{INTR}}$ (ngắt hay gọi chính xác hơn là "kết thúc chuyển đổi').

Đây là chân đầu ra tích cực mức thấp. Bình thường nó ở trạng thái cao và khi việc chuyển đổi hoàn tất thì nó xuống thấp để báo cho CPU biết là dữ liệu được chuyển đổi sẵn sàng để lấy đi. Sau khi $\overline{\text{INTR}}$ xuống thấp, ta đặt CS = 0 và gửi một xung cao 0 xuống - thấp tới chân $\overline{\text{RD}}$ lấy dữ liệu ra của 804.

6. Chân V_{in} (+) và V_{in} (-).

Đây là các đầu vào tương tự vi sai mà $V_{in} = V_{in}$ (+) - V_{in} (-). Thông thường V_{in} (-) được nối xuống đất và V_{in} (+) được dùng như đầu vào tương tự được chuyển đổi về dạng số.

7. Chân V_{CC}.

Đây là chân nguồn nuôi +5v, nó cũng được dùng như điện áp tham chiếu khi đầu vào $V_{ref/2}$ (chân 9) để hở.

8. Chân $V_{ref/2}$.

Chân 9 là một điện áp đầu vào được dùng cho điện áp tham chiếu. Nếu chân này hở (không được nối) thì điện áp đầu vào tương tự cho ADC 804 nằm trong dải 0 đến +5v (giống như chân V_{CC}). Tuy nhiên, có nhiều ứng dụng mà đầu vào tương tự áp

đến V_{in} cần phải khác ngoài dải 0 đến 5v. Chân $V_{ref/2}$ đượcdùng để thực thi các điện áp đầu vào khác ngoài dải 0 - 5v. Ví dụ, nếu dải đầu vào tương tự cần phải là 0 đến 4v thì $V_{ref/2}$ được nối với +2v.

V_{ref}/ 2(V)	V_{in}(V)	Step Size (mV)
Hở *	0 đến 5	5/256 = 19.53
2.0	0 đến 4	4/255 = 15.62
1.5	0 đến 3	3/256 = 11.71
1.28	0 đến 2.56	2.56/256 = 10
1.0	0 đến 2	2/256 = 7.81
0.5	0 đến 1	1/256 = 3.90

Bảng 8-3. Điện áp $Vref_{/2}$ liên hệ với dải V_{in}.

Ghi chú: - V_{CC} = 5V

- Khi $V_{ref}/2$ hở thì đo được ở đó khoảng 2,5V
- Kích thước bước (độ phân dải) là sự thay đổi nhỏ nhất mà ADC có thể phân biệt được.

9. Các chân dữ liệu D0 - D7.

Các chân dữ liệu D0 - D7 (D7 là bít cao nhất MSB và D0 là bít thấp nhất LSB) là các chân đầu ra dữ liệu số. Đây là những chân được đệm ba trạng thái và dữ liệu được chuyển đổi chỉ được truy cập khi chân CS = 0 và chân $\overline{RD}$ bị đưa xuống thấp. Để tính điện áp đầu ra ta có thể sử dụng công thức sau:

$$D_{out} = \frac{V_{in}}{\text{kich thuoc buoc}}$$ Với D_{out} là đầu ra dữ liệu số (dạng thập phân). V_{in} là điện áp đầu vào tương tự và độ phân dải là sự thay đổi nhỏ nhất được tính như là $(2 \times V_{ref}/2)$ chia cho 256 đối với ADC 8 bít.

10. Chân đất tương tự và chân đất số.

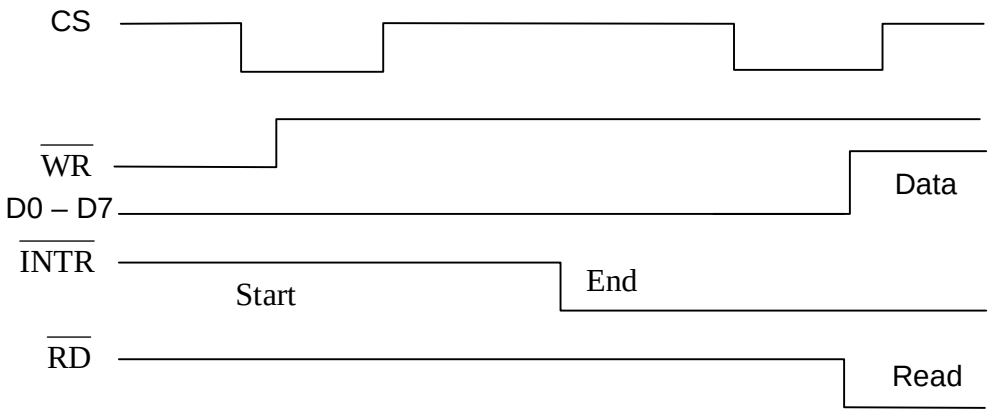
Đây là những chân đầu vào cấp đất chung cho cả tín hiệu số và tương tự. Đất tương tự được nối tới đất của chân V_{in} tương tự, còn đất số được nối tới đất của chân V_{cc}. Lý do mà ta phải có hai đất là để cách ly tín hiệu tương tự V_{in} từ các điện áp ký sinh tạo ra việc chuyển mạch số được chính xác. Trong phần trình bày của chúng ta thì các chân này được nối chung với một đất. Tuy nhiên, trong thực tế thu đo dữ liệu các chân đất này được nối tách biệt.

Từ những điều trên ta kết luận rằng các bước cần phải thực hiện khi chuyển đổi dữ liệu bởi ADC 804 là:

a) Bật CS = 0 và gửi một xung thấp lên cao tới chân $\overline{WR}$ để bắt đầu chuyển đổi.

b) Duy trì hiển thị chân $\overline{INTR}$. Nếu $\overline{INTR}$ xuống thấp thì việc chuyển đổi được hoàn tất và ta có thể sang bước kế tiếp. Nếu $\overline{INTR}$ cao tiếp tục thăm dò cho đến khi nó xuống thấp.

c) Sau khi chân $\overline{INTR}$ xuống thấp, ta bật CS = 0 và gửi một xung cao - xuống - thấp đến chân $\overline{RD}$ để lấy dữ liệu ra khỏi chíp ADC 804. Phân chia thời gian cho quá trình này được trình bày trên hình 6.6

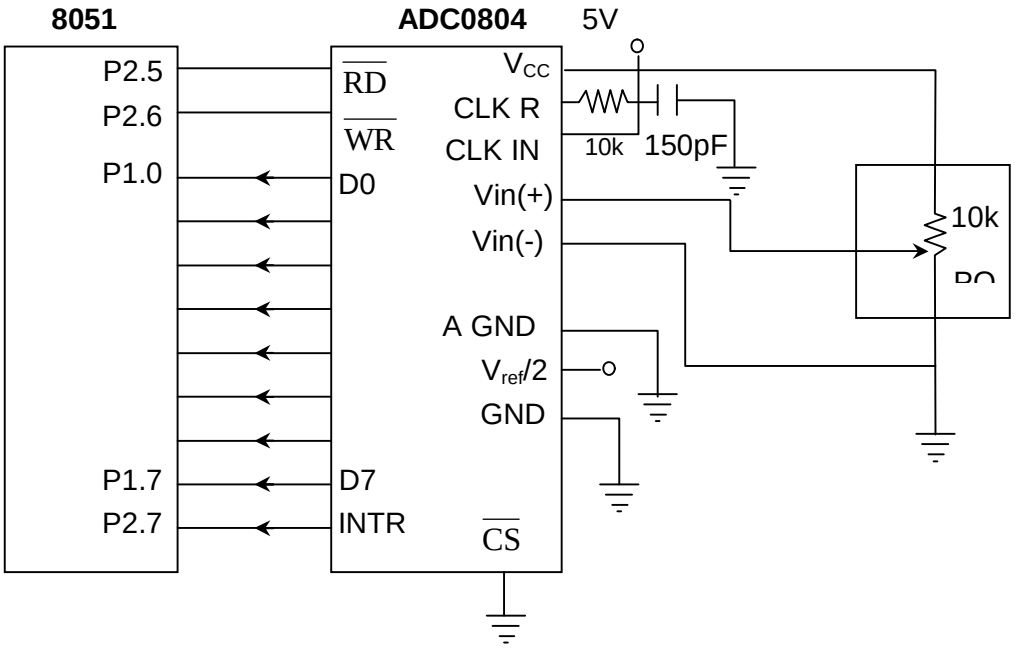


Hình 8-3. Phân chia thời gian đọc và ghi của ADC 804.

8.2.3. Ghép nối 8051 với ADC 0804.

Chúng ta có thể kiểm tra ADC 804 bằng cách sử dụng sơ đồ mạch trên hình 7-6. thiết lập này được gọi là chế độ kiểm tra chạy tự do và được nhà sản xuất khuyến cao nên sử dụng. Hình 6.4 trình bày một biến trở được dùng để cấp một điện áp tương tự từ 0 đến 5V tới chân đầu vào.

$V_{in}(+)$ của ADC 804 các đầu ra nhị phân được hiển thị trên các đèn LED của bảng huấn luyện số. Cần phải lưu ý rằng trong chế độ kiểm tra chạy tự do thì đầu vào CS được nối tới đất và đầu vào $\overline{WR}$ được nối tới đầu ra $\overline{INTR}$. Tuy nhiên, theo tài liệu của hãng National Semiconductor "nút WR và INTR phải được tạm thời đưa xuống thấp kế sau chu trình cấp nguồn để bảo đảm hoạt động".



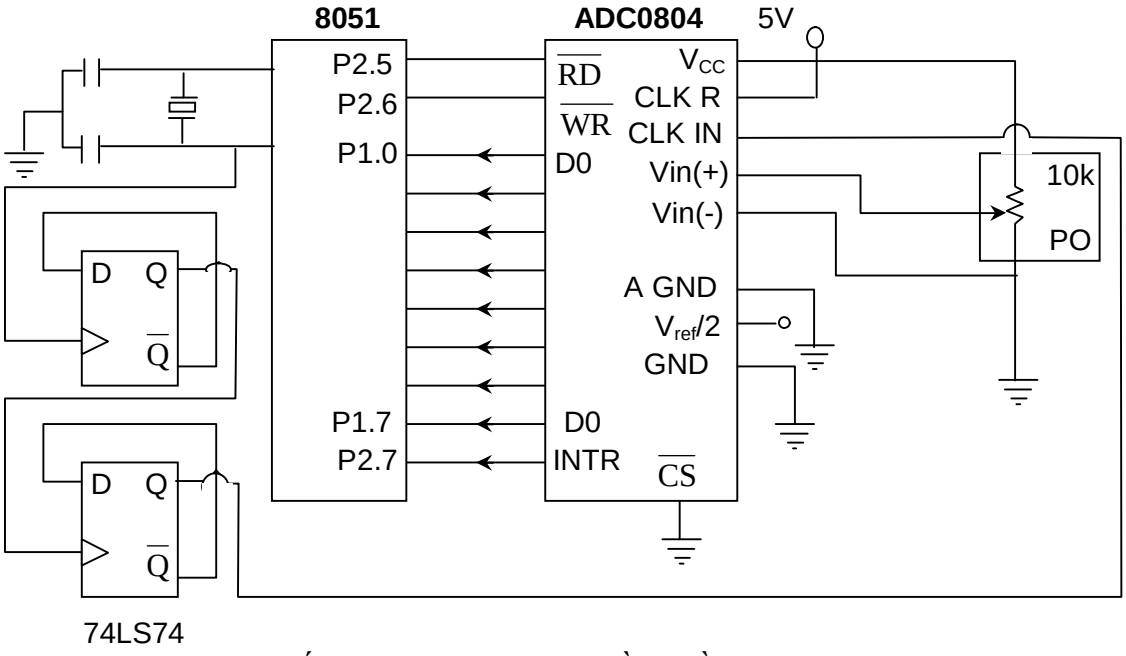
Hình 8-4. Nối ghép ADC 0804

Ví dụ:

Hãy thử nối ghép ADC 804 với 8051 theo sơ đồ 6.7. Viết một chương trình để hiển thị chân INTR và lấy đầu vào tương tự vào thanh ghi A. Sau đó gọi một chương trình chuyển đổi mã Hex ra ASCII và một chương trình hiển thị dữ liệu. Thực hiện điều này liên tục.

Lời giải:

```
; Đặt P2.6 = WR (bắt đầu chuyển đổi cần 1 xung thấp lên cao)
; Đặt chân P2.7 = 0 khi kết thúc chuyển đổi
; Đặt P2.5 = RD (xung cao - xuống - thấp sẽ đọc dữ liệu từ ADC)
; P1.0 – P1.7 của ADC 804
          MOV  P1, # 0FFH          ; Chọn P1 là cổng đầu vào
BACK:     CLR   P2.6              ; Đặt WR = 0
          SETB P2.6               ; Đặt WR = 1 để bắt đầu chuyển đổi
HERE:     JB P2.7, HERE           ; Chờ cho P2.7 to để kết thúc chuyển đổi
          CLR   P2.5              ; Kết thúc chuyển đổi, cho phép đọc RD
          MOV  A,  P1             ; Đọc dữ liệu vào thanh ghi A
          ACALL CONVERSION              ; Chuyển đổi số Hex ra mã ASCII
          ACALL DATA-DISPLAY            ; Hiển thị dữ liệu
          SETB P2.5                     ; Đưa RD = 1 để cho lần đọc sau.
          SJMP  BACK
```



Hình 8-5. Nối ghép ADC 804 với đồng hồ từ XTAL2 của 8051.

Trên hình 7-7 ta có thể thấy rằng tín hiệu đồng hồ đi vào ADC 804 là từ tần số thạch anh của 8051. Vì tần số này quá cao nên ta sử dụng hai mạch lật Rlip - Flop kiểu D (74LS74) để chia tần số này cho 4. Một mạch lật chia tần số cho 2 nếu ta nối đầu $\overline{Q}$ tới đầu vào D. Đối với tần số cao hơn thì ta cần sử dụng nhiều mạch Flip - Plop hơn.

PHỤ LỤC

Phụ lục A: Các ký hiệu sử dụng mô tả lệnh

Ký hiệu	Mô
A:	Thanh ghi chứa (Accumulator).
B:	Thanh ghi B.
Ri:	Thanh ghi R0 hoặc R1 của bất kỳ băng thanh ghi nào trong 4 băng thanh ghi trong RAM.
Rn:	Rn: bất kỳ thanh ghi nào của bất kỳ băng thanh ghi nào trong 4 băng thanh ghi trong RAM.
Dptr:	thanh ghi con trỏ dữ liệu (có độ rộng 16bit được kết hợp từ 2 thanh ghi 8 bit là DPH và DPL).
Direct:	Direct: là một biến 8 bit(hay chính là ô nhớ) bất kỳ trong RAM (trừ 32 thanh ghi Rn ở đầu RAM).
#data:	một hằng số 8 bit bất kỳ.
#data16:	một hằng số 16 bit bất kỳ
<rel>:	địa chỉ bất kỳ nằm trong khoảng [PC-128 ; PC+127]
<addr11>:	địa chỉ bất kỳ nằm trong khoảng 0 – 2Kbyte tính từ địa chỉ của lệnh tiếp theo.
<addr16>:	địa chỉ bất kỳ trong không gian 64K (áp dụng cho cả không gian nhớ chương trình và không gian nhớ dữ liệu).
<bit>:	bit bất kỳ có thể đánh địa chỉ được (không dùng cho các bit không đánh được địa chỉ).

Phụ lục B: Tập lệnh 8051

Lệnh số học

STT	Cú pháp		Mô tả	Số byte	Số chu kỳ
	Mã lệnh	Toán hạng			
1	ADD	A, Rn	A = A + Rn	1	1
2	ADD	A,direct	A = A + direct	2	1
3	ADD	A,@Ri	A = A + @Ri	1	1
4	ADD	A,#data	A = A + #data	2	1
5	ADDC	A,Rn	A = A + Rn + C	1	1
6	ADDC	A,direct	A = A + direct + C	2	1
7	ADDC	A,@Ri	A = A + @Ri + C	1	1
8	ADDC	A,#data	A = A + #data + C	2	1
9	SUBB	A,Rn	A = A – Rn – C	1	1
10	SUBB	A,direct	A = A – direct – C	2	1
11	SUBB	A,@Ri	A = A – @Ri – C	1	1
12	SUBB	A,#data	A = A – #data – C	2	1
13	INC	A	A=A+1	1	1
14	INC	Rn	Rn = Rn + 1	1	1
15	INC	Direct	direct = direct + 1	2	1
16	INC	@Ri	@Ri = @Ri + 1	1	1
17	DEC	A	A=A–1	1	1
18	DEC	Rn	Rn = Rn – 1	1	1
19	DEC	Direct	direct = direct – 1	2	1
20	DEC	@Ri	@Ri = @Ri – 1	1	1
21	INC	Dptr	dptr = dptr + 1	1	2
22	MUL	AB	B:A = A*B	1	4
23	DIV	AB	A/B = A(thương) + B (dư)	1	4
24					
25	DA	A	Hiệu chỉnh thập phân số liệu trong thanh ghi A	1	1

Lệnh logic

STT	Cú pháp		Mô tả	Số byte	Số chu kỳ
	Mã lệnh	Toán hạng			
1	ANL	A,Rn	A = (A)and(Rn)	1	1
2		A,direct	A = (A)and(direct)	2	1
3		A,@Ri	A = (A)and(@Ri)	1	1
4		A,#data	A = (A)and(#data)	2	1
5		direct,A	direct = (direct)and(A)	2	1
6		Direct,#data	direct = (direct)and(#data)	3	2
7	ORL	A,Rn	A = (A)or(Rn)	1	1
8		A,direct	A = (A)or(direct)	2	1
9		A,@Ri	A = (A)or(@Ri)	1	1
10		A,#data	A = (A)or(#data)	2	1
11		direct,A	direct = (direct)or(A)	2	1
12		Direct,#data	direct = (direct)or(#data)	3	2
13	XRL	A,Rn	A = (A)xor(Rn)	1	1
14		A,direct	A = (A)xor(direct)	2	1
15		A,@Ri	A = (A)xor(@Ri)	1	1
16		A,#data	A = (A)xor(#data)	2	1
17		direct,A	direct = (direct)xor(A)	2	1
18		Direct,#data	direct = (direct)xor(#data)	3	2
19	CLR	A	A = 0	1	1
20	CPL	A	A = not(A)	1	1
21	RL	A	Quay trái A	1	1
22	RLC	A	Quay trái A qua cờ C	1	1
23	RR	A	Quay phải A	1	1
24	RRC	A	Quay phải A qua cờ C	1	1
25	SWAP	A	Hoán đổi 2 nửa của A	1	1

Các lệnh bít

STT	Cú pháp		Mô tả	Số byte	Số chu kỳ
	Mã lệnh	Toán hạng			
1	CLR	C	Xóa cờ C về 0	1	1
2	CLR	Bit	Xóa bit về 0	2	1
3	SETB	C	Đặt cờ C = 1	1	12
4	SETB	Bit	Đặt bit = 1	2	1
5	CPL	C	Đảo giá trị của cờ C	1	1
6	CPL	Bit	Đảo giá trị của bit	2	1
7	ANL	C,bit	C = (C)and(bit)	2	2
8	ANL	C,/bit	C = (C)and(đảo của bit)	2	2
9	ORL	C,bit	C = (C)or(bit)	2	2
10	ORL	C,/bit	C = (C)or(đảo của bit)	2	2
11	MOV	C,bit	C = bit	2	1
12	MOV	Bit,C	Bit = C	2	2
13	JC	<rel>	nhảy đến nhãn <rel> nếu C = 1	2	2
14	JNC	Bit, <rel>	nhảy đến nhãn <rel> nếu bit= 1	3	2
15	JB	Bit, <rel>	nhảy đến nhãn <rel> nếu bit= 1	3	2
16	JNB	Bit, <rel>	nhảy đến nhãn <rel> nếu bit= 0	3	2
17	JBC	Bit, <rel>	nhảy đến nhãn <rel> nếu bit = 1 và sau đó xóa luôn bit về 0	3	2

Lệnh dich chuyển dữ liệu

STT	Cú pháp		Mô tả	Số byte	Số chu kỳ
	Mã lệnh	Toán hạng			
1	MOV	A,Rn	Copy giá trị của toán hạng bên phải cho vào toán hạng bên trái (các toán hạng đều là 8bit)	1	1
2	MOV	A,direct		2	1
3	MOV	A,@Ri		1	1
4	MOV	A,#data		2	1
5	MOV	Rn,A		1	1
6	MOV	Rn,direct		2	2
7	MOV	Rn,#data		2	1
8	MOV	Direct,A		2	1
9	MOV	Direct,Rn		2	2
10	MOV	Direct,direct		3	2
11	MOV	Direct,@Ri		2	2
12	MOV	Direct,#data		3	2
13	MOV	@Ri,A		1	1
14	MOV	@Ri,direct		2	1
15	MOV	@Ri,#data		2	1
16	MOV	Dptr,#data16	Đưa giá trị 16bit vào thanh ghi DPTR	3	2
17	MOVC	A,@A+dptr	Đọc giá trị bộ nhớ chương trình tại địa chỉ = A + DPTR, cất kết quả	1	2
18	MOVC	A,@A+PC	Đọc giá trị bộ nhớ chương trình tại địa chỉ = A + PC, cất kết quả vào A	1	2
19	MOVX	A,@Ri	Đọc vào A giá trị của bộ nhớ ngoài tại địa chỉ = Ri	1	2
20	MOVX	A,@dptr	Đọc vào A giá trị của bộ nhớ ngoài tại địa chỉ = DPTR	1	2
21	MOVX	@dptr,A	Ghi giá trị của A vào bộ nhớ ngoài tại địa chỉ = DPTR	1	2
22	MOVX	@dptr,A	Ghi giá trị của A vào bộ nhớ ngoài tại địa chỉ = DPTR	2	2
23	PUSH	Direct	Cất nội dung của biến trong RAM vào đỉnh ngăn xếp	2	2
24	POP	Direct	Lấy byte ở đỉnh ngăn xếp cho vào biến trong RAM	2	2
25	XCH	A,Rn	Hoán đổi giá trị của A và giá trị còn lại	1	1
26	XCH	A,direct		2	1

Các lệnh điều khiển chương trình

STT	Cú pháp		Mô tả	Số byte	Số chu kỳ
	Mã lệnh	Toán hạng			
1	ACALL	<addr11>	gọi chương trình con (nằm trong phạm vi 2k mem)	3	2
2	LCALL	<addr16>	gọi chương trình con (trong pham vi 64k mem)	3	2
3	RET		trở về từ chương trình con	1	2
4	RETI		trở về từ chương trình phục vụ ngắt	1	2
5	AJMP	<addr11>	nhảy đến nhãn (trong phạm vi 2k mem)	2	2
6	LJMP	<addr16>	nhảy đến nhãn (trong phạm vi 64 mem)	3	2
7	SJMP	<rel>	nhảy đến nhãn	2	2
8	JMP	@A+DPTR	nhảy đến địa chỉ = A+DPTR	1	2
9	JZ	<rel>	nhảy đến nhãn nếu A = 0	2	2
10	JNZ	<rel>	nhảy đến nhãn nếu A #0	2	2
11	CJNE	A,direct,<rel>	So sánh và nhảy đến nhãn nếu A # direct	3	2
12	CJNE	A,#data,<rel>	So sánh và nhảy đến nhãn nếu A#data	3	2
13		Rn,#data,<rel>	So sánh và nhảy đến nhãn nếu Rn#data	3	2
14		@Ri,#data,<rel>	So sánh và nhảy đến nhãn nếu byte có địa chỉ = Ri có nội dung khác với data	3	2
15	DJNZ	Rn,<rel>	Giảm Rn đi 1 và nhảy đến nhãn nếu chưa giảm về 0	2	2
16	DJNZ	direct,<rel>	Giảm direct đi 1 và nhảy đến nhãn nếu chưa giảm về 0	3	2
17	NOP		Không làm gì cả	1	1

Phụ lục B: Chi tiết các thanh ghi chức năng trong 8051

1. Thanh ghi IE:

IE: Interrupt Enable, cho phép ngắt: thanh ghi này cho phép/cấm các ngắt hoạt động

EA	--	ET2	ES	ET1	EX1	ET0	EX0

2. Thanh ghi TCON: TCON Register - TCON (S:88h)

TCON: Timer/Counter Control Register: thanh ghi điều khiển bộ đếm/bộ định thời

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

3. Thanh ghi TMOD: TMOD Register - TMOD (S: 89h)

TMOD: Timer/Counter 0 and 1 Modes: thanh ghi đặt chế độ cho Timer/Counter 0 và 1

GATE1	C/T1#	M11	M01	GATE0	C/T0#	M10	M00

4. Thanh ghi T2CON: T2CON Register - T2CON (S:C8h)

Thanh ghi điều khiển Timer/Counter 2

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#

5. Thanh ghi T2MOD: T2MOD Register - T2MOD (S:C9h)

Thanh ghi điều khiển Timer/Counter 2

-	-	-	-	-	-	T2OE	DCEN

6. Thanh ghi SCON

SCON: Serial Controller: thanh ghi cấu hình truyền thông nối tiếp.

FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI

7. Thanh ghi PCON: PCON Register

PCON - Power Control Register (87h): Thanh ghi điều khiển nguồn

SMOD1	SMOD0	-	POF	GF1	GF0	PD	IDL