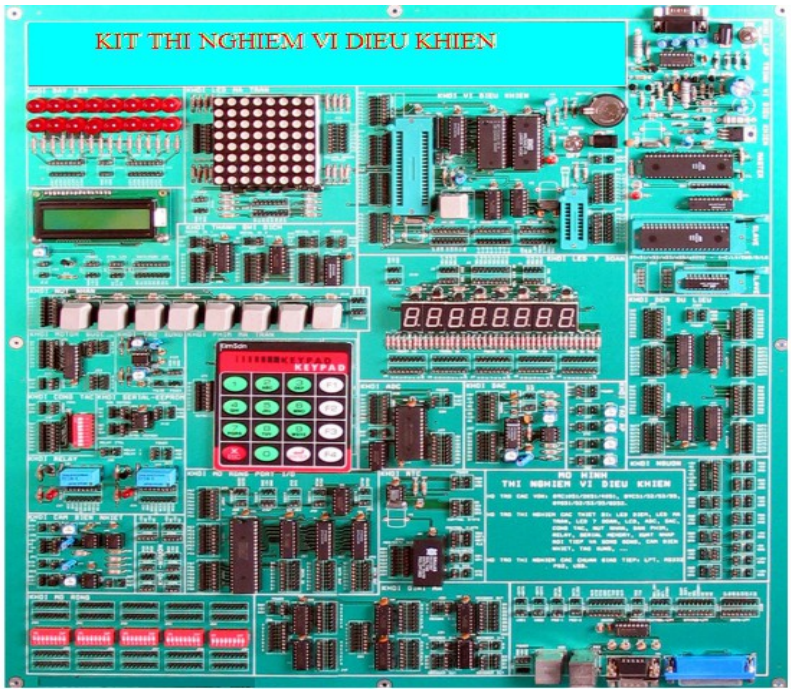
BỘ LAO ĐỘNG THƯƠNG BINH VÀ XÃ HỘI
TỔNG CỤC DẠY NGHỀ

GIÁO TRÌNH
Mô đun: VI ĐIỀU KHIỂN
NGHỀ: ĐIỆN TỬ CÔNG NGHIỆP
TRÌNH ĐỘ : TRUNG CẤP

Ban hành kèm theo Quyết định số:120/QĐ-TCDN ngày 25 tháng 02 năm 2013 của Tổng cục trưởng Tổng cục Dạy nghề



Năm 2013

TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

Để thực hiện biên soạn giáo trình đào tạo nghề Điện tử công nghiệp ở trình độ CĐN và TCN, giáo trình Mô đun Vi điều khiển là một trong những giáo trình mô đun đào tạo chuyên ngành được biên soạn theo nội dung chương trình khung được Bộ Lao động - Thương binh và Xã hội và Tổng cục Dạy nghề ban hành dành cho hệ Cao Đẳng Nghề và Trung Cấp Nghề Điện tử công nghiệp.

Nội dung biên soạn ngắn gọn, dễ hiểu, tích hợp kiến thức và kỹ năng chặt chẽ với nhau, logíc.

Khi biên soạn, nhóm biên soạn đã cố gắng cập nhật những kiến thức mới có liên quan đến nội dung chương trình đào tạo và phù hợp với mục tiêu đào tạo, nội dung lý thuyết và thực hành được biên soạn gắn với nhu cầu thực tế trong sản xuất đồng thời có tính thực tiễn cao. Nội dung giáo trình được biên soạn với dung lượng thời gian đào tạo 150 giờ gồm có:

Bài MĐ24-01: Sơ lược về lịch sử và hướng phát triển của vi điều khiển.

Bài MĐ24-02: Cấu trúc họ vi điều khiển 8051.

Bài MĐ24-03: Tập lệnh 8051.

Bài MĐ24-04: Bộ định thời.

Bài MĐ24-05: Cổng nối tiếp.

Bài MĐ24-06: Ngắt.

Bài MĐ24-07: Phần mềm hợp ngữ.

Trong quá trình sử dụng giáo trình, tuỳ theo yêu cầu cũng như khoa học và công nghệ phát triển có thể điều chỉnh thời gian và bổ sung những kiên thức mới cho phù hợp. Trong giáo trình, chúng tôi có đề ra nội dung thực tập của từng bài để người học cũng cố và áp dụng kiến thức phù hợp với kỹ năng.

Tuy nhiên, tuỳ theo điều kiện cơ sở vật chất và trang thiết bị, các trường có thể sử dụng cho phù hợp. Mặc dù đã cố gắng tổ chức biên soạn để đáp ứng được mục tiêu đào tạo nhưng không tránh được những khiếm khuyết. Rất mong nhận được đóng góp ý kiến của các thầy, cô giáo, bạn đọc để nhóm biên soạn sẽ hiệu chỉnh hoàn thiện hơn. Các ý kiến đóng góp xin gửi về Trường Cao đẳng nghề Lilama 2, Long Thành Đồng Nai

Đồng Nai, ngày 10 tháng 06 năm 2013

Tham gia biên soạn

1. Chủ biên :TS. Lê Văn Hiền

2. Kỹ sư Lê Văn Hùng

3. Kỹ sư Nguyễn Văn Tuấn

MỤC LỤC

TRANG

MÔ ĐUN VI ĐIỀU KHIỂN

Mã mô đun: MĐ 25

Vị trí, tính chất, ý nghĩa và vai trò của mô đun:

- Vị trí của mô đun: Mô đun được bố trí dạy sau khi học xong môn học mô đun: Kỹ thuật xung số điện tử cơ bản, điện tử nâng cao, điện tử công suất, và học trước môn vi mạch số lập trình..
- Tính chất của mô đun: Là mô đun chuyên môn nghề.
- Y´nghiã của mô đun: mô đun giuíp ngươìi học co´kiến thưć vê` điều khiển hệ thống va thiêt bị băng Vi đêu khiển.
- Vai tro`của mô đun: La`mô đun chuyên ngành giuíp ngươìi học điều Vi khiển hệ thống thông qua cać Vi xử ly´.

Mục tiêu của mô đun:

- Vận hành được các thiết bị và dây chuyền sản xuất dùng vi điều khiển.
- Xác định được các nguyên nhân gây ra hư hỏng xảy ra trong thực tế.
- Kiểm tra và viết được các chương trình điều kiển.

❖ *Về kiến thức:*

- Trình bày được cấu trúc, ứng dụng cả vi điều khiển trong công nghiệp.
- Kiểm tra và viết được các chương trình điều khiển.

❖ *Về kỹ năng:*

- Vận hành được các thiết bị và dây chuyền sản xuất dùng vi điều khiển.
- Xác định được các nguyên nhân gây ra hư hỏng xảy ra trong thực tế.

❖ *Về thái độ:*

- Rèn luyện cho học sinh thái độ nghiêm túc, cẩn thận, chính xác trong học tập và thực hiện công việc

Nội dung của mô đun:

Mã bài	Tên các bài trong mô đun	Thời gian			
		Tổng số	Lý thuyết	Thực Hành	Kiểm tra

MĐ24-01	Sơ lược về lịch sử và hướng phát triển của vi điều khiển	4	4		
1.	Lịch sử phát triển	1	1		
2.	Vi điều khiển	1	1		
3.	Lĩnh vực và ứng dụng	1	1		
4.	Hướng phát triển	1	1		
MĐ24-02	Cấu trúc họ vi điều khiển 8051	10	7	3	
1.	Tổng quan	1	1		
2.	Sơ đồ chân	1	1		

3.	Cấu trúc Port I/O	1	1		
4.	Tổ chức bộ nhớ	1	1		
5.	Các thanh ghi chức năng đặc biệt	1	1		
6.	Bộ nhớ ngoài	1	1		
7.	Các cải tiến của 8032/8052	0,5	0,5		
8.	Hoạt động Reset	0,5	0,5		
9.	Thực hành ứng dụng	3		3	
MĐ24-03	Tập lệnh 8051	30	7	22	1
1.	Mở đầu	1	1		

2.	Các cách định địa chỉ	4	2	2	
3.	Các nhóm lệnh	5	4		1
4.	Luyện tập	20		20	
MĐ24-04	Bộ định thời	30	9	20	1
1.	Mở đầu	1	1		
2.	Thanh ghi SFR của timer	1	1		
3.	Các chế độ làm việc	2	2		
4.	Nguồn cung cấp xung cho	2	2		

	Timer				
5.	Khởi động, dừng, điều khiển Timer	1	1		
6.	Khởi tạo và truy xuất thanh ghi Timer	2	1		1
7.	Timer 2 của 8052	1	1		
8.	Luyện tập	20		20	
MĐ24-05	Cổng nối tiếp	30	6	23	1
1.	Mở đầu	1	1		
2.	Thanh ghi điều khiển	1	1		

3.	Chế độ làm việc	2	1	1	
4.	Khởi tạo và truy suất thanh ghi PORT nối tiếp	2	1		1
5.	Truyền thông đa xử lý	2	1	1	
6.	Tốc độ BAUD	1	1	1	
7.	Luyện tập	20		20	
MĐ24-06	Ngắt	30	8	21	1
1.	Mở đầu	1	1		
2.	Tổ chức ngắt của 8051	2	2		
3.	Xử lý	1	1		

	ngắt				
4.	Thiết kế chương trình dùng ngắt	3	2	1	
5.	Ngắt cổng nối tiếp	1	1		
6.	Các cổng ngắt ngoài	1	1		
7.	Đồ thị thời gian của ngắt	1			1
8.	Luyện tập	20		20	
MĐ24-07	Phần mềm hợp ngữ	16	6	9	1
1.	Mở đầu	1	1		
2.	Hoạt	1	1		

	động của ASSEMBLER				
3.	Cấu trúc chương trình dữ liệu	1		1	
4.	Tính biểu thức trong khi hợp dịch	2	1	1	
5.	Các điều khiển của ASSEMBLER	2	0,5	1,5	
6.	Hoạt động liên kết	2	1	0,5	0,5
7.	MACRO	2	1,5		0,5
8.	Luyện tập	5		5	
Tổng cộng			150	45	100	5

BAI 1
SƠ LƯỢC VỀ LỊCH SỬ VÀ HƯỚNG PHÁT TRIỂN CỦA VI ĐIỀU KHIỂN
Mã bài: MĐ24-01

Giới thiệu:

Trong những thập niên cuối thế kỷ XX, từ sự ra đời của công nghệ bán dẫn, kỹ thuật điện tử đã có sự phát triển vượt bậc. Các thiết bị điện tử sau đó đã được tích hợp với mật độ cao và rất cao trong các diện tích nhỏ,nhờ vậy các thiết bị nhỏ hơn và nhiều chức năng hơn. Các thiết bị điện tử ngày càng nhiều chức năng trong khi giá thành ngày càng rẻ hơn, chính vì vậy điện tử có mặt khắp nơi. Bước đột phát mới trong kỹ thuật điện tử là tạo ra một thiết bị điện tử mới là Vi điều khiển.

Một bộ vi điều khiển (microcontroller) được xem như là "một máy tính trong một chip" – nó là một mạch điện tích hợp trên một chip, có thể lập trình được, dùng để điều khiển hoạt động của một hệ thống.

Vi điều khiển được ứng dụng rất rộng rãi hiện nay. Đa số các lĩnh vực đều có thể ứng dụng vi điều khiển. Và đối với nền cơ khí tự động hoá bây giờ thì có lẽ nó đã gắn liền với vi xử lý. Vi điều khiển là một câu trúc siêu nhỏ, gồm các linh kiện điện tử co´kích thước micro hoặc nano kết hợp với nhau, và được nối với các thiết bị bên ngoài qua các chân vi điều khiển. Vì vậy hiểu rõ cấu trúc của nó, ta sẽ hiểu được mình đang làm việc với cái gì? Và nó hoạt động như thế nào?

Mục tiêu:

- Trình bày được cấu trúc chung của vi điều khiển.
- Phát biểu được các ứng dụng của vi điều khiển và hướng phát triển của vi điều khiển.

Nội dung chính:

1. Lịch sử xuất hiện bộ vi điều khiển 8051.

Mục tiêu:

- Trình bay được lịch sử hình thành và qua´trình phát triển của họ vi điều khiển 8051.
- Trình bay được câú truć chung của vi điêù khiển.

Nội dung:

- 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. Chip 8051 chứa trên 60000 transistor bao gồm 4K byte ROM, 128 byte RAM, 32 đường xuất nhập, 1 port nối tiếp và 2 bộ định thời 16 bit. 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.

- Hiện nay có rất nhiều họ Vi điều khiển trên thị trường với nhiều ứng dụng khác nhau, trong đó họ Vi điều khiển họ MCS-51 được sử dụng rất rộng rãi trên thế giới và ở Việt Nam.
- Vào năm 1980 Intel công bố chíp 8051(80C51), bộ vi điều khiển đầu tiên của họ vi điều khiển MCS-51. Nó bao gồm 4KB ROM, 128 byte RAM, 32 đường xuất nhập, 1 port nối tiếp và 2 bộ định thời 16 bit. Tiếp theo sau đó là sự ra đời của chip 8052, 8053, 8055 với nhiều tính năng được cải tiến.
- Hiện nay Intel không còn cung cấp các loại Vi điều khiển họ MCS-51 nữa, thay vào đó các nhà sản xuất khác như Atmel, Philips/signetics, AMD, Siemens, Matra&Dallas, Semiconductors được cấp phép làm nhà cung cấp thứ hai cho các chip của họ MSC-51. Chip Vi điều khiển được sử dụng rộng rãi trên thế giới cũng như ở Việt Nam hiện nay là Vi điều khiển của hãng Atmel với nhiều chủng loại vi điều khiển khác nhau.
- Hãng Atmel có các chip Vi điều khiển có tính năng tương tự như chip Vi điều khiển MCS-51 của Intel, các mã số chip được thay đổi chút ít khi được Atmel sản xuất. Mã số 80 chuyển thành 89, chẳng hạn 80C52 của Intel khi sản xuất ở Atmel mã số thành 89C52 (Mã số đầy đủ: AT89C52) với tính năng chương trình tương tự như nhau. Tương tự 8051,8053,8055 có mã số tương đương ở Atmel là 89C51,89C53,89C55. Vi điều khiển Atmel sau này ngày càng được cải tiến và được bổ sung thêm nhiều chức năng tiện lợi hơn cho người dùng.

Bảng 1

	Dung lượng RAM	*Dung lượng ROM*	*Chế độ nạp*
89C51	128 byte	4 Kbyte	song song
89C52	128 byte	8 Kbyte	song song
89C53	128 byte	12 Kbyte	song song
89C55	128 byte	20 Kbyte	song song

- Sau khoảng thời gian cải tiến và phát triển, hãng Atmel tung ra thị trường dòng Vi điều khiển mang số hiệu 89Sxx với nhiều cải tiến và đặc

biệt là có thêm khả năng nạp chương trình theo chế độ nối tiếp rất đơn giản và tiện lợi cho người sử dụng.

Bảng 2

	Dung lượng RAM	*Dung lượng ROM*	*Chế độ nạp*
89S51	128 byte	4 Kbyte	nối tiếp
89S52	128 byte	8 Kbyte	nối tiếp
89S53	128 byte	12 Kbyte	nối tiếp
89S55	128 byte	20 Kbyte	nối tiếp

- Tất cả các Vi điều khiển trên đều có đặc tính cơ bản giống nhau về phần mềm (các tập lệnh lập trình như nhau), còn phần cứng được bổ sung với chip có mã số ở hai số cuối cao hơn, các Vi điều khiển sau này có nhiều tính năng vượt trội hơn Vi điều khiển thế hệ trước. Các Vi điều khiển 89Cxx như trong bảng 1 có cấu tạo ROM và RAM như 98Sxx trong bảng 2, tuy nhiên 98Sxx được bổ sung một số tính năng và có thêm chế độ nạp nối tiếp.
- 8051 là bộ vi điều khiển 8 bit tức là CPU chỉ có thể làm việc với 8 bit

dữ liệu. Dữ liệu lớn hơn 8 bit được chia thành các dữ liệu 8 bit để xử lý.

- 8051 đã trở nên phổ biến sau khi Intel cho phép các nhà sản xuất khác (Siemens, Atmel, Philips, AMD, Matra, Dallas, Semiconductor …) sản xuất và bán bất kỳ dạng biến thể nào của 8051 mà họ muốn với điều kiện họ phải để mã chương trình tương thích với 8051. Từ đó dẫn đến sự ra đời của nhiều phiên bản của 8051 với các tốc độ và dung lượng ROM trên chip khác nhau.
- Tuy nhiên, điều quan trọng là mặc dù có nhiều biến thể của 8051, cũng như khác nhau về tốc độ, dung lượng ROM nhưng tất cả các lệnh đều tương thích với 8051 ban đầu. Điều này có nghĩa là nếu chương trình được viết cho một phiên bản 8051 nào đó thì cũng sẽ chạy được với mọi phiên bản khác không phụ thuộc vào hãng sản xuất.
- Cać loại vi điều khiển khać: vi điều khiển AVR, vi điều khiển PIC, vi điều khiển MCUs của Philips,...Ngoaì ra, các loại vi điều khiển chuyên dụng của các hãng sản xuất khác: các loại vi điều khiển này được sử dụng chuyên dụng theo chức năng cần điều khiển.

Bảng 3: Địa chỉ của một sô hãng sản xuất cać thành viên vi điều khiển

Hãng	Địa chỉ Website
intel	www.intel.com/design/mcs51
Antel	www.atmel.com
Philips/Signetis	www.semiconductors.philips.com
Siemens	www.sci.siemens.com
Dallas Semiconductor	www.dalsemi.com

2. Vi điều khiển (microcontroller).

Mục tiêu : Hiểu được cấu trúc bên trong và nguyên lý hoạt động của bộ *vi điều khiển 8051.*

2.1. Nguyên lý, cấu tạo.

2.1.1. Cấu tạo vi điều khiển.

- *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 gồm một vi xử lý có hiệu suất đủ cao và giá thành thấp (so với các vi xử lý đa năng dùng trong máy tính) kết hợp với các thiết bị ngoại vi như các bộ nhớ, các mô đun vào/ra, các mô đun biến đổi từ số sang tương tự và từ tương tự sang số, mô đun điều chế độ rộng xung (PWM)...
- Vi điều khiển thường được dùng để xây dựng hệ thống nhúng. Nó xuất hiện nhiều trong các dụng cụ điện tử, thiết bị điện, máy giặt, lò vi sóng, điện thoại, dây truyền tự động...
- Hầu hết các loại vi điều khiển hiện nay có cấu trúc Harvard là loại cấu trúc mà bộ nhớ chương trình và bộ nhớ dữ liệu được phân biệt riêng.
- Cấu trúc của một vi điều khiển gồm CPU, bộ nhớ chương trình (thường là bộ nhớ ROM hoặc bộ nhớ Flash), bộ nhớ dữ liệu (RAM), các bộ định thời, các cổng vào/ra để giao tiếp với các thiết bị bên ngoài, tất cả các khối này được tích hợp trên một vi mạch.

Các loại vi điều khiển trên thị trường hiện nay:

- VDK MCS-51: 8031, 8032, 8051, 8052,...
- VDK ATMEL: 89Cxx, AT89Cxx51...
- VDK AVR AT90Sxxxx
- VDK PIC 16C5x, 17C43...

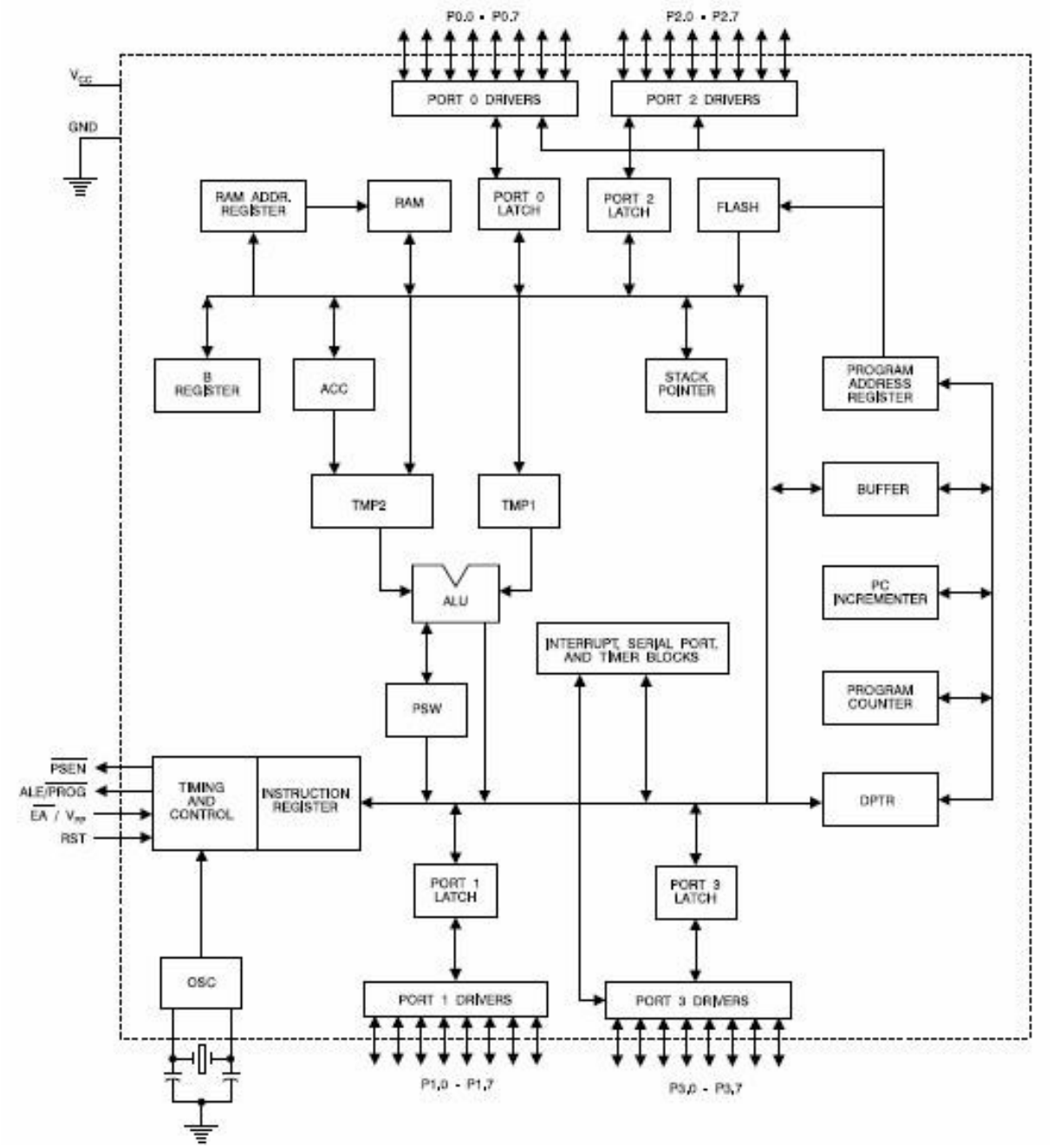
2.1.2. Nguyên lý hoạt động của Vi điều khiển

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:

- 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.
- 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 vaì mili giây đầu tiên tất cả đã sẵn sàng hoạt động.
- 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.
- 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.
- Giá trị trong thanh ghi PC được tăng lên 1 và toàn bộ quá trình được lặp lại vài … triệu lần trong một giây.

❖ Các kiểu cấu trúc bộ nhớ (*Hình 1.1)*



Hình 1.1. Cấu trúc bên trong của vi điều khiển.

- Memory (bộ nhớ): là ROM/RAM lưu trữ chương trình hay các kết quả trung gian.

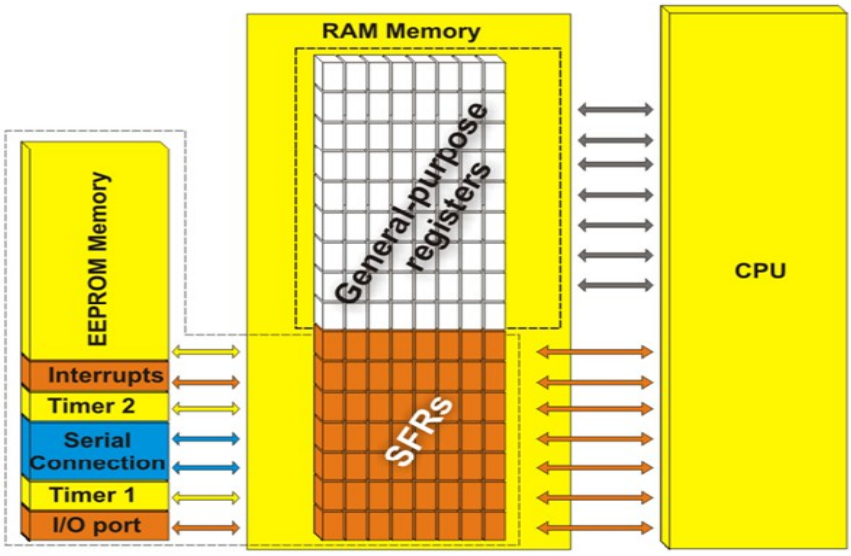
Read Only Memory (ROM): Read Only Memory (ROM) là một loại bộ nhớ được sử dụng để lưu vĩnh viễn các chương trình được thực thi. Kích cỡ của chương trình có thể được viết phụ thuộc vào kích cỡ của bộ nhớ này. ROM có thể được tích hợp trong vi điều khiển hay thêm vào như là một chip gắn bên ngoài, tùy 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. 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 co´thêm lá ghim sẵn để kết nối với môi trường ngoại vi. Kích thước của dãy ROM từ 512B đến 64KB.

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.

- Electrically Erasable Programmable ROM (EEPROM) (hình 1.2)

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 tương đối chậm.



Hình 1.2. Giao tiếp bộ nhớ

- 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. 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).

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

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.

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.

Arithmetical Logical Unit (ALU): Thực thi tất cả các thao tác tính toán số học và logic.

- 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.

- 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. Chúng có thể thay đổi chức năng, chiều vào/ra theo yêu cầu của người dùng.

Address bus(Bus địa chỉ): Là các đường tín hiệu song song 1 chiều nối từ CPU đến bộ nhớ, CPU gửi giá trị địa chỉ của ô nhớ cần truy nhập (đọc/ghi) trên các đường tín hiệu này. Độ rộng của bus địa chỉ là n (là số các đường tín hiệu, với n có thể là 8, 18, 20, 24, 32 hay 64), khi đó số ô nhớ mà CPU có thể địa chỉ hố được sẽ là 2^n.

- Data bus(Bus dữ liệu): 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. Độ rộng Bus dữ liệu là m(với m có thể là 4, 8, 16, 32 hay 64), khi đó số bit mà môi một chu kỳ đọc/ghi có thể truyền trên trên bus là m bits.

- Control bus(Bus điều khiển): CPU gửi tín hiệu thông qua bus này để điều khiển mọi hoạt động của hệ thống. Các tín hiệu điều khiển thường là: đọc/ ghi bộ nhớ, đọc/ ghi cổng vào/ra,…

3. **Lĩnh vực và ứng dụng.**

Mục tiêu: Biết được các ứng dụng của bộ vi điều khiển 8051.

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.
- Một phần mềm đơn giản có thể điều khiển được toàn bộ hoạt động của vi điều khiển và có thể dễ dàng cho người sử dụng nắm bắt.

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.

3.1. 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.

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, m áy chơi game, Đầu thu kỹ thuật số, đầu thu set-top-box,...

3.2. 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,...)

3.3. Các sản phẩm công nghiệp.

Điều khiển động cơ

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

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

Điều khiển các dây truyền sản xuất công nghiệp

Làm bộ điều khiển trung tâm cho Robot

4. **Hướng phát triển.**

- Kết hợp các bộ vi xử lý và vi điều khiển trong các sản phẩm hệ thống nhúng.
- Sử dụng tốt nhất các tính năng của vi điều khiển: tốc độ mà bộ vi điều khiển hỗ trợ, dung lượng bộ nhớ RAM và ROM trên chíp,...
- Tìm hiểu được khả năng phát triển các sản phẩm xung quanh.
- Nghiên cứu các bộ vi điều khiển 8051 từ các hãng khác nhau: 8751, AT89C51, DS500,...

CÁC BÀI TẬP MỞ RỘNG, NÂNG CAO VÀ GIẢI QUYẾT VẤN ĐỀ

Câu 1: Nêu câú truć của Vi điêù khiển họ 8051?

Gợi yí: Cấu trúc của một vi điều khiển gồm CPU, bộ nhớ chương trình (thường là bộ nhớ ROM hoặc bộ nhớ Flash), bộ nhớ dữ liệu (RAM), các bộ định thời, các cổng vào/ra để giao tiếp với các thiết bị bên ngoài, tất cả các khối này được tích hợp trên một vi mạch.

Câu 2: Chưć năng của thanh ghi ALU?

- Gợi yí:

Arithmetical Logical Unit (ALU): Thực thi tất cả các thao tác tính toán số học và logic.

Câu 3: Nêu 1 sô´liñh vực ứng của Vi điêù khiển.

- Gợi yí:

Một số ứng dụng cơ bản thành công có thể kể ra sau đây:

Sản phẩm dân dụng.

Nhà thông minh, các máy móc dân dụng, các sản phẩm giải trí, máy móc thiết bị hỗ trợ, maý cắt/mài kính, máy chụp chiếu (city, X-quang,...)

Các sản phẩm công nghiệp.

Điều khiển động cơ

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

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

Điều khiển các dây truyền sản xuất công nghiệp

Làm bộ điều khiển trung tâm cho Robot

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức: cấu tạo, đặc điểm,ứng dụng của các loại Viêù khiển được học

+ Về kỹ năng:
- Thực hiện viết các chương trình theo yêu cầu cho trướoc
+ Thái độ: Đánh giá phong cách, thái độ học tập

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết
+ Về kỹ năng: Đánh giá kỹ năng thực hành Mỗi sinh viên, hoặc mỗi nhóm học viên thực hiện công việc theo yêu cầu của giáo viên.Tiêu chí đánh giá theo các nội dung:
  - Độ chính xác của công việc
  - Thời gian thực hiện công việc
  - Độ chính xác theo yêu cầu kỹ thuật

+ Thái độ: Tỉ mỉ, cẩn thận, chính xác.

BAI 2
CẤU TRÚC CỦA HỌ VI ĐIỀU KHIỂN 8051
Mã bài: MĐ24-02

Giới thiệu:

Trong những thập niên cuối thế kỉ XX, từ sự ra đời của công nghệ bán dẫn, kỹ thuật điện tử đã có sự phát triển vượt bậc. Các thiết bị điện tử sau đó đã được tích hợp với mật độ cao và rất cao trong các diện tích nhỏ, nhờ vậy các thiết bị nhỏ hơn và nhiều chức năng hơn. Các thiết bị điện tử ngày càng nhiều chức năng trong khi giá thành ngày càng rẻ hơn, chính vì vậy điện tử có mặt khắp nơi. Bước đột phát mới trong kỹ thuật điện tử là tạo ra một bộ Vi điều khiển.

Vi điều khiển(Microcontroller) có 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, nhưng cấu trúc phần cứng dành cho người dùng đơn giản. 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. Vi`Vi điều khiển có giá thành rẻ hơn nhiều so với vi xử lý, việc sử dụng đơn giản nên nó được ứng dụng rộng rãi vào nhiều ứng dụng có chức năng đơn giản, không đòi hỏi tính toán phức tạp. Do đó, để nắm được hoạt động của các hệ thống dùng vi điều khiển ta phải tìm hiểu cấu trúc của họ vi điều khiển 8051.

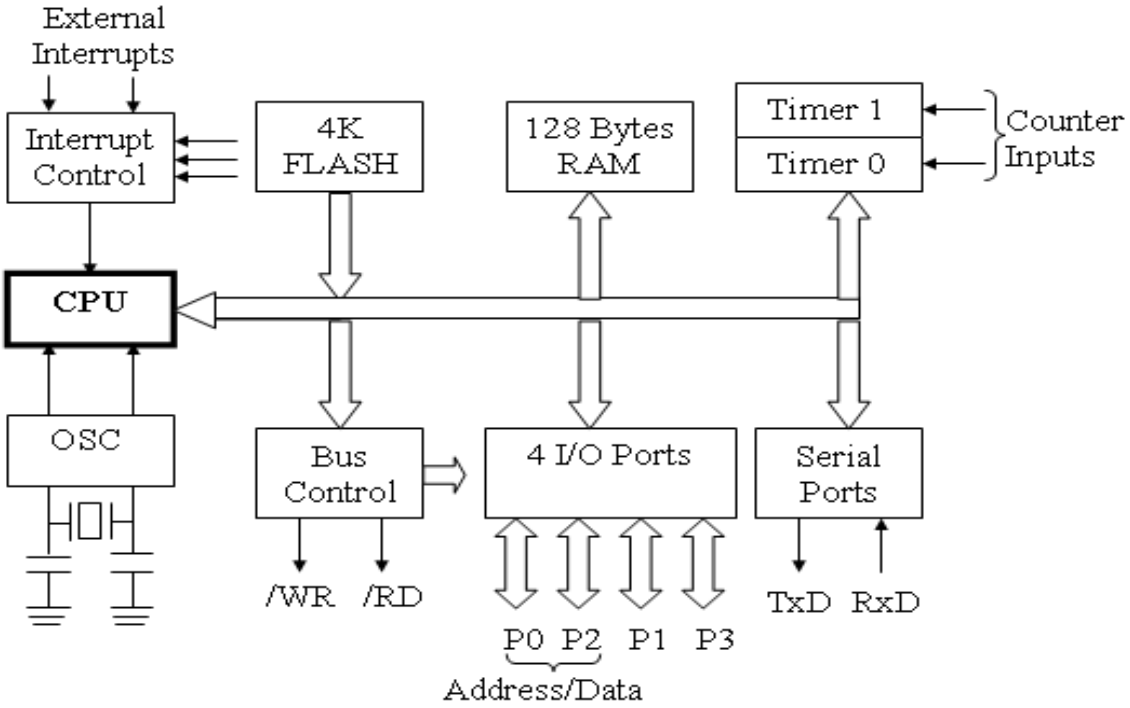
Mục tiêu của bài:

- Mô tả được cấu trúc họ vi điều khiển chuẩn công nghiệp.
- Thực hiện truy xuất bộ nhớ dữ liệu, bộ nhớ chương trình đúng qui trình kỹ thuật.
- Thực hiện đúng kỹ thuật phương pháp mở rộng bộ nhớ ngoài.
- Trình bày được nguyên lý hoạt động của mạch reset.

Nội dung chính:

1. Tổng quan

Mục tiêu: Mô tả được cấu trúc họ vi điều khiển chuẩn công nghiệp.



Hình 2.1. Sơ đồ khối vi điều khiển 8051.

Thuật ngữ "8051" được dùng để chỉ rộng rãi các chip của họ MSC-51.Vi mạch tổng quát của họ MSC-51 là chip 8051,linh kiện đầu tiên của họ này được hãng Intel đưa ra thị trường. 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. Hiện hay nhiều nhà sản xuất IC như Seimens, Advance Micro Devices (AMD), Fujitsu, Philips, Atmel … được cấp phép làm nhà cung cấp thứ hai cho các chip của họ MSC-51. Ở Việt Nam các chip và các biến thể họ MSC-51 của hãng Atmel và Philips được sử dụng rộng rãi như : AT89C2051, AT89C4051, AT89C51, AT8C52, AT89S52, AT89S8252, AT89S8253, P89C51RDxx, P89V51RDxx …

Vi điều khiển 8051có 40 chân, mỗi chân có một kí hiệu tên và có các đặc trưng như sau :

- 4KB ROM, 128 byte RAM.
- 4 port xuất nhập (I/O port) 8 bit.
- 2 bộ định thời 16 bit, m ạch giao tiếp nối tiếp.
- Không gian nhớ chương trình ngoài 64K, không gian nhớ dữ liệu ngoài 64K.
- Bộ xử lý bit 210 vị trí nhớ được định địa chỉ,mỗi vị trí 1 bit nhân/chia trong 4µs.

Tuy nhiên, tuỳ thuộc vào từng họ VĐK của từng hãng sản xuất khác nhau mà tính năng cũng như phạm vi ứng dụng của mỗi bộ VĐK là khác nhau, và chúng được thể hiện trong các bảng thống kê sau (bảng 2.1, 2.2):

Họ VĐK	ROM (bytes)	RAM (bytes)	Tốc độ (MHz)	Các chân I/O	Timer/ Counter	UART	Nguồn ngắt
8051							
8031AH	ROMLESS	128	12	32	2	1	5
8051AH	4K ROM	128	12	32	2	1	5
8051AHP	4K ROM	128	12	32	2	1	5
8751H	4K EPROM	128	12	32	2	1	5
8751BH	4K EPROM	128	12	32	2	1	5
8052							
8032AH	ROMLESS	256	12	32	3	1	6
8052AH	8K ROM	256	12	32	3	1	6
8752BH	8K EPROM	256	12	32	3	1	6
80C51				32			
80C31BH	ROMLESS	128	12,16	32	2	1	5
80C51BH	4K ROM	128	12,16	32	2	1	5
80C31BHP	4K ROM	128	12,16	32	2	1	5
87C51	4K EPROM	128	12,16,20,24	32	2	1	5
8xC52/54/58							
80C32	ROMLESS	256	12,16,20,24	32	3	1	6
80C52	8K ROM	256	12,16,20,24	32	3	1	6
87C52	8K EPROM	256	12,16,20,24	32	3	1	6
80C54	16K ROM	256	12,16,20,24	32	3	1	6
87C54	16K EPROM	256	12,16,20,24	32	3	1	6
Họ VĐK	ROM (bytes)	RAM (bytes)	Tốc độ (MHz)	Các chân I/O	Timer/ Counter	UART	Nguồn ngắt
80C58	32K ROM	256	12,16,20,24	32	3	1	6
87C58	32K EPROM	256	12,16,20,2	32	3	1	6

			4				
8xL52/54/58							
80L52	8K ROM	256	12,16,20	32	3	1	6
87L52	8K OTP ROM	256	12,16,20	32	3	1	6
80L54	16K ROM	256	12,16,20	32	3	1	6
87L54	16K OTP ROM	256	12,16,20	32	3	1	6
80L58	32K ROM	256	12,16,20	32	3	1	6
87L58	32K OTP ROM	256	12,16,20	32	3	1	6
…							

Bảng 2.1. Các thông số của các họ VĐK thuộc hãng Intel (MSC 51)

Họ VĐK	*Bộ nhớ chương trình (Bytes)*	*Bộ nhớ dữ liệu (Bytes)*	*Timer 16 bit*	*Công nghệ*
AT89C1051	1K Flash	64 RAM	1	CMOS
AT89C2051	2K Flash	128 RAM	2	CMOS
AT89C51	4K Flash	128 RAM	2	CMOS
AT89C52	8K Flash	256 RAM	3	CMOS
AT89C55	20K Flash	256 RAM	3	CMOS
AT89S8252	8K Flash	256 RAM + 2K EEPROM	3	CMOS
AT89S53	12K Flash	256 RAM	3	CMOS

Bảng 2.2. Các thông số của các họ VĐK thuộc hãng Atmel

2. Sơ đồ chân vi điều khiển 8051:

Mục tiêu: Hiểu chức năng các chân của vi điều khiển

Mặc dù các thành viên của họ MSC-51 có nhiều kiểu đóng vỏ khác nhau, chẳng hạn như: hai hàng chân DIP (Dual in-Line Pakage), dạng vỏ dẹp vuông và dạng chíp không có chân đỡ LLC (Leadless Chip Carrier). Họ MSC-51 có 40 chân thực hiện các chức năng khác nhau như: vào ra (I/O), đọc, ghi, địa chỉ, dữ liệu và ngắt. Tuy nhiên, trong khuôn khổ chương trình chỉ khảo sát Vi điều khiển 40 chân dạng DIP (hình 2.2).

Hình 2.2. Sơ đồ chân của AT89C51

Chip 8051 có 40 chân, mỗi chân có một kí hiệu tên và có các chức năng như sau:

- Chân 40: nối với nguồn nuôi +5V.
- Chân 20: nối với đất (Mass, GND).
- Chân 29 (PSEN)(program store enable) là tín hiệu điều khiển xuất ra của 8051, nó cho phép chọn bộ nhớ ngoài và được nối chung với chân của OE (Outout Enable) của EPROM ngoài để cho phép đọc các byte của chương trình. Các xung tín hiệu PSEN hạ thấp trong suốt thời gian thi hành lệnh. Những mã nhị phân của chương trình được đọc từ EPROM đi qua bus dữ liệu và được chốt vào thanh ghi lệnh của 8051 bởi mã lệnh. (chú ý việc đọc ở đây là đọc các lệnh (khác với đọc dữ liệu), khi đó VXL chỉ đọc các bit opcode của lệnh và đưa chúng vào hàng đợi lệnh thông qua các Bus địa chỉ và dữ liệu).
- Chân 30 (ALE : Adress Latch Enable) là tín hiệu điều khiển xuất ra của 8051, nó cho phép phân kênh bus địa chỉ và bus dữ liệu của Port 0.
- Chân 31 (EA : Eternal Acess) được đưa xuống thấp cho phép chọn bộ nhớ mã ngoài. Đối với 8051 thì:

EA = 5V : Chọn ROM nội.

EA = 0V : Chọn ROM ngoại.

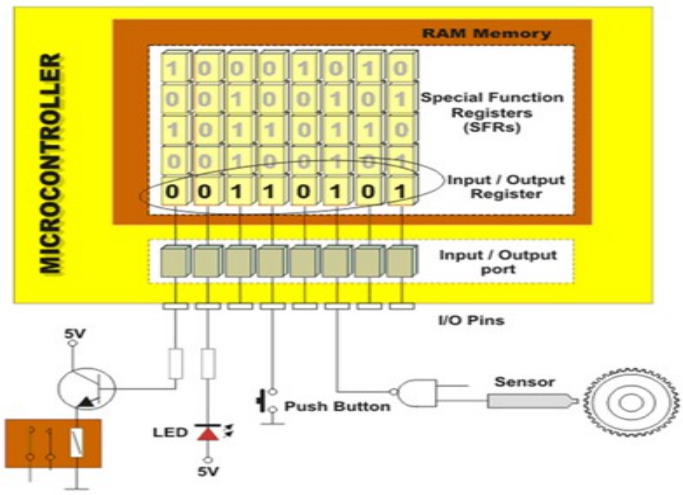
- 32 chân còn lại chia làm 4 cổng vào / ra: có thể dùng chân đó để để đọc mức logic (0;1 tương ứng với 0V; 5V)vào hay xuất mức logic ra (0;1), (hình 2.3)

Port 0 từ chân 39 - 32 tương ứng là các chân P0.0 đến P0.7.

Port 1 từ chân 1- 8 tương ứng là các chân P1.0 đến P1.7.

Port 2 từ chân 21- 28 tương ứng là các chân P2.0 đến P2.7.

Port 1 từ chân 10- 17 tương ứng là các chân P3.0 đến P3.7.



Hình 2.3. Sơ đồ kết nối các chân xuất nhập với thiết bị ngoại vi

- Chip 8051 có 32 chân xuất / nhập,tuy nhiên có 24 chân trong 32 chân này có 2 mục đích.Mỗi một chân này có thể hoạt động ở chế độ xuất/nhập, hoạt động điều khiển hoặc hoạt động như một đường địa chỉ / dữ liệu của bus địa chỉ / dữ liệu đa hợp.

2.1. Port 0

Port 0 (các chân từ 32 - 39) được ký hiệu là P0.0 – P0.7 có hai công dụng. Trong các thiết kế có tối thiểu thành phần, port 0 được sử dụng làm nhiệm vụ xuất nhập, với các thiết kế lớn hơn có bộ nhớ ngoài, port 0 trở thành bus địa chỉ và bus dữ liệu đa hợp (byte địa chỉ thấp).

2.2. Port 1

Port 1 (các chân từ 1 - 8) chỉ có một công dụng là xuất/nhập được ký hiệu là P1.0 – P1.7 và dùng để giao tiếp với thiết bị bên ngoài. Với chip 8052 ta có thể sử dụng P1.0 và P1.1 hoặc làm các đường xuất/nhập hoặc làm các ngõ vào cho mạch định thời thứ ba.

2.3. Port 2

Port 2 (các chân từ 21-28) được ký hiệu là P2.0 – P2.7 có hai công dụng,hoặc làm nhiệm vụ xuất/nhập hoặc là byte địa chỉ cao của bus địa chỉ

16 bit cho các thiết kế có bộ nhớ chương trình ngoài hoặc các thiết kế có nhiều hơn 256 byte bộ nhớ dữ liệu.

2.4. Port 3

Port 3 (các chân từ 10 - 17) được ký hiệu là P3.0 – P3.7 có hai công dụng. Khi không hoạt động xuất/nhập, các chân của port 3 có nhiều chức năng riêng.

Chức năng các chân của Port 3 và Port 1(bảng 2.3)

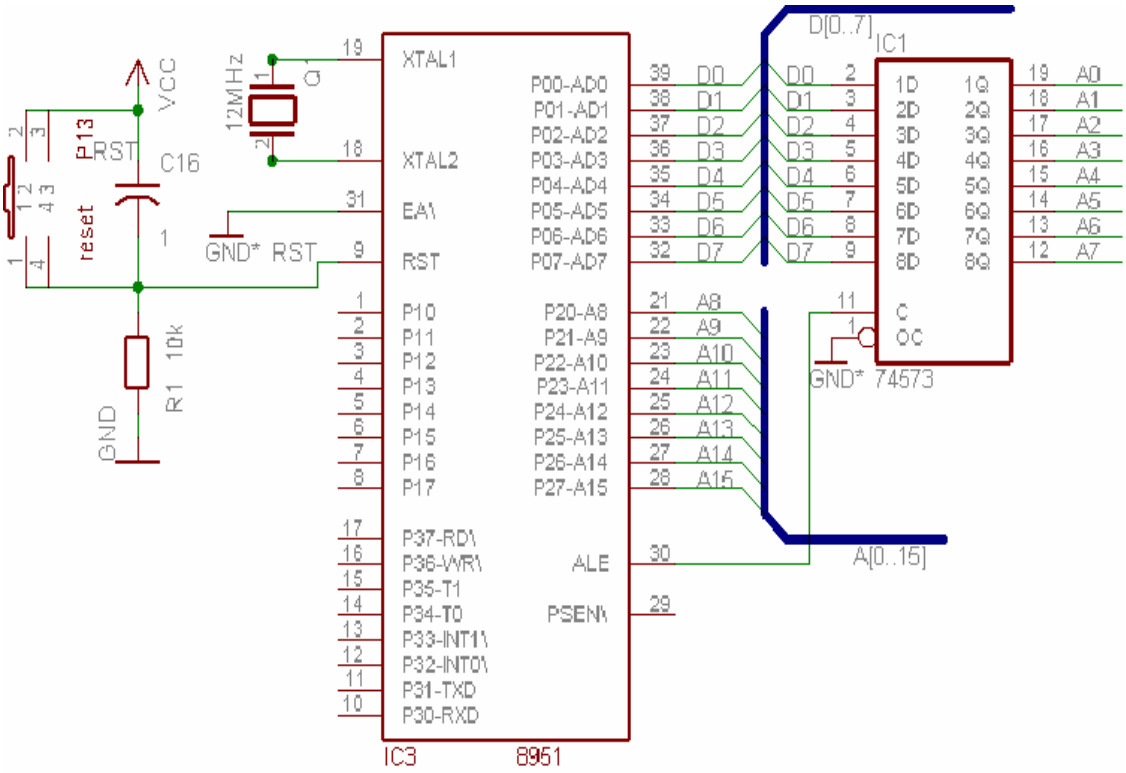
Bit	Tên chân	Địa chỉ bit	Chức năng
P3.0	RxD	B0H	Chân nhận dữ liệu của port nối tiếp
P3.1	TxD	B1H	Chân phát dữ liệu của port nối tiếp
P3.2	/INT0	B2H	Ngõ vào ngắt ngoài 0
P3.3	/INT1	B3H	Ngõ vào ngắt ngoài 1
P3.4	T0	B4H	Ngõ vào bộ định thời hoặc bộ đếm 0
P3.5	T1	B5H	Ngõ vào bộ định thời hoặc bộ đếm 1
P3.6	/WR	B6H	Điều khiển ghi bộ nhớ dữ liệu ngoài
P3.7	/RD	B7H	Điều khiển đọc bộ nhớ dữ liệu ngoài
P1.0	T2	90H	Ngõ vào bộ đinh thời hoặc bộ đếm 2
P1.1	T2EX	91H	Nạp lại hoặc thu nhận của bộ định thời 2

Bảng 2.3. Sơ đồ chân chức năng của Port 3 và Port 1

2.5. Chân cho phép bộ nhớ chương trình PSEN

Chân cho phép bộ nhớ chương trình /PSEN (Program store enable) là tín hiệu xuất trên chân 29. Đây là tín hiệu điều khiển cho phép ta truy xuấtbộ nhớ chương trình ngoài.Chân này thường nối với chân cho phép xuất /OE (Output enable) của EPROM hoặc ROM để cho phép đọc các byte lệnh.Tín hiệu /PSEN ở mức logic 0 trong suốt thời gian tìm nạp lệnh.

Các mã nhị phân của chương trình hay Opcode được đọc từ EPROM qua bus dữ liệu và được chốt vào thanh ghi lệnh IR của 8051 để được giải mã. Khi thực thi một chương trình chứa ở ROM nội, chân /PSEN được duy trì mức logic không tích cực (logic 1) (hình 2.4).



Hình 2.4. Ghép nối vi điều khiển 8951 với IC chốt, mạch Reset, tụ thạch anh.

2.6.Chân cho phép chốt địa chỉ ALE

Ngõ xuất tín hiệu cho phép chốt địa chỉ ALE (address latch enable) dùng để giải đa hợp (demultiplexing) bus dữ liệu và bus địa chỉ.Khi port 0 được sử dụng làm bus địa chỉ/dữ liệu đa hợp,chân ALE xuất tín hiệu để chốt địa chỉ (byte thấp của địa chỉ 16 bit) vào một thanh ghi ngoài trong suốt ½ đầu của chu kỳ bộ nhớ (memory cycle).Sau khi điều này đã được thực hiện, các chân của port 0 sẽ xuất/nhập dữ liệu hợp hệ trong suốt ½ thứ hai của chu kỳ bộ nhớ.Tín hiệu ALE có tần số bằng 1/6 tần số của mạch dao động bên trong chip vi điều khiển.

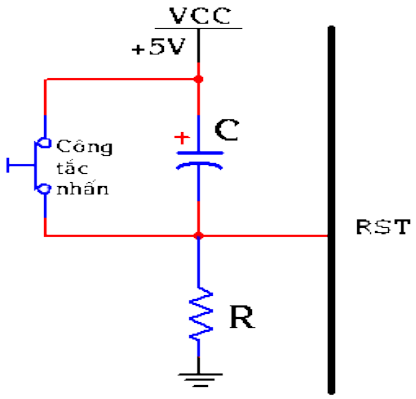
2.7. Chân truy xuất ROM ngoài EA

Ngõ vào /EA có thể được nối với 5V (logic 1) hoặc với GND (logic 0). Nếu chân này nối lên 5V chip 8051 thực thi chương trình trong ROM nội. Nếu chân này được nối với GND (và chân /PSEN cũng ở logic 0) thì chương trình cần được thực thi chứa ở bộ nhớ ngoài.

2.8. Chân RESET (RST)

- Ngõ vào RST là ngõ vào xóa chính (master reset) của 8051 (hình 2.5) dùng để thiết lập lại trạng thái ban đầu cho hệ thống hay gọi tắt là reset hệ thống. Khi ngõ vào này được treo ở mức logic 1 tối thiểu 2 chu kỳ máy, các thanh ghi bên trong của 8051 được nạp lại các giá trị thích

hợp cho việc khởi động lại hệ thống.



Hình 2.5 Mạch Reset

Việc kết nối chân RESET đảm bảo hệ thống bắt đầu làm việc khi Vi điều khiển được cấp điện, hoặc đang hoạt động mà hệ thống bị lỗi cần tác động cho Vi điều khiển hoạt động trở lại, hoặc do người sử dụng muốn quay về trạng thái hoạt động ban đầu. Vì vậy chân RESET được kết nối như sau:

- Với Vi điều khiển sử dụng thạch anh có tần số f_{zat} = 12MHz sử dụng C=10μF và R=10KΩ.Thanh ghi quan trọng nhất là thanh ghi bộ đếm chương trình PC = 0000H.

- Sau khi reset, vi điều khiển luôn bắt đầu thực hiện chương trình tại địa chỉ 0000H của bộ nhớ chương trình nên cácchương trình viết cho vi điều khiển luôn bắt đầu viết tại địa chỉ 0000H. Nội dung của RAM trong vi điều khiển không bị thay đổi bởi tác động của ngõ vào reset [có nghĩa là vi điều khiển đang sử dụng các thanh ghi để lưu trữ dữ liệu nhưng nếu vi điều khiển bị reset thì dữ liệu trong các thanh ghi vẫn không đổi].

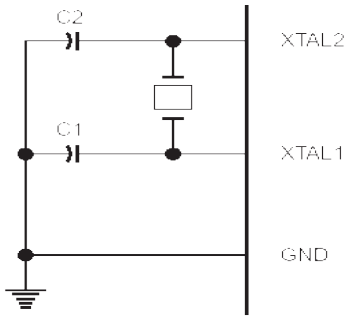
2.9. Các chân XTAL1, XTAL2

2.9.1. Kết nối chân XTAL1, XTAL2

- Mạch dao động trên chip được ghép nối với mạch thạch anh bên ngoài

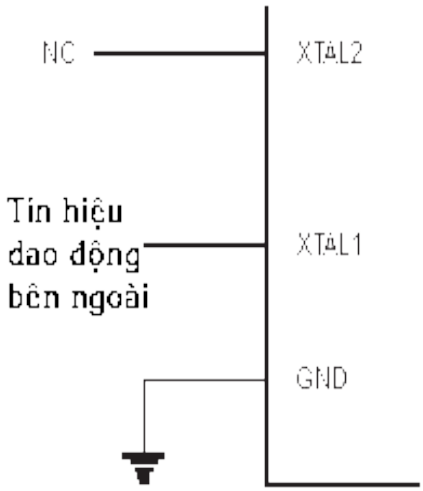
ở hai chân XTAL1 và XTAL2 (hình 2.6), các tụ ổn định cũng được yêu cầu kết nối, giá trị tụ do nhà sản xuất quy định (30p – 40p). Tần số thạch anh thường dùng trong các ứng dụng là: 11.0592Mhz (giao tiếp với cổng Com máy tính) và 12Mhz. Tần số tối đa 24Mhz. Tần số càng lớn VĐK xử lí càng nhanh.

- Mạch dao động được đưa vào hai chân này thông thường được kết nối với dao động thạch anh như sau:



Hình 2.6 Mạch dao động

Ghi chú: C1,C2= 30pF±10pF (thường được sử dụng với C1,C2 là tụ 33pF) dùng ổn định dao động cho thạch anh. Hoặc có thể cấp tín hiệu xung clock lấy từ một mạch tạo dao động nào đó và đưa vào Vi điều khiển theo cách sau (hình 2.7):



Hình 2.7 Lấy tín hiệu dao động bên ngoài

2.9.2. Chu kì máy.

Gọi f_{zat} là tần số dao động của *thạch anh.* Đối với 89Sxx có thể sử dụng thạch anh có tần số f_{zat} từ 2MHz đến 33MHz.

Chu kì máy là khoảng thời gian cần thiết được quy định để Vi điều khiển thực hiện hoàn thành một lệnh cơ bản. Một chu kì máy bằng 12 lần chu kì dao động của nguồn xung dao động cấp cho nó.

$$T_{ck} = 12.\frac{1}{f_{oc}}$$

$T_{ck} = 12.T_{oc}$

Với: T_{ck} là chu kì máy

T_{oc} là chu kì của nguồn xung dao động cấp cho Vi điều khiển

Như vậy:

Với: T_{ck} là chu kì máy

f_{oc} là tần số dao động cấp cho Vi điều khiển.

Ví dụ: Ta kết nối Vi điều khiển với thạch anh có tần số f_{zat} là 12MHz, thì chu kì máy

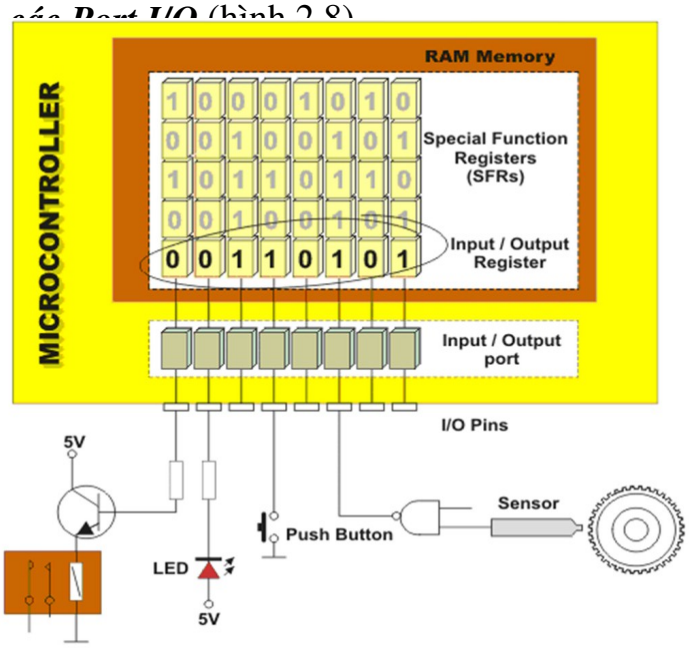
$T_{ck}=12/(12.10^{6})=10^{-6}s=1\mu s$

Chính vì lí do thạch anh có tần số f_{zat} là 12MHz tạo ra chu kì máy là $1\mu s$, thuận lợi cho việc tính toán thời gian khi lập trình do đó thạch anh có tần số f_{zat} là 12MHz thường được sử dụng trong thực tế. Khi giao tiếp truyền nối tiếp với máy vi tính dùng thạch anh có tần số f_{zat} là 11.0592MHz.

3. Cấu trúc Port I/O

Mục tiêu: Hiểu được cấu trúc các cổng vào ra của vi điều khiển

3.1. Chức năng các Port I/O (hình 2.8)



Hình 2.8. Vào ra với thiết bị ngoại vi

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

3.1.1. *Port 0:*

- Chức năng IO (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 ra, Port 0 có thể kéo được 8 ngõ TTL.

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 mà khi kiểm tra (quá trình kiểm tra đòi hỏi phải có điện trở kéo lên).

3.1.2. *Port 1:*

- Port1 (chân 1 – 8) chỉ có một chức năng là IO, 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.

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

- Chức năng IO (xuất/nhập): có khả năng kéo được 4 ngõ TTL.
- 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.
- Khi lập trình, Port 2 dùng làm 8 bit địa chỉ cao hay một số tín hiệu điều khiển.

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

- Chức năng IO: có khả năng kéo được 4 ngõ TTL.
- Khi dùng làm ngõ vào, Port 3 phải được set mức logic 1 trước đó.

3.2. Kết nối các Port với led.

- Các Port khi xuất tín hiệu ở mức logic 1 thường không đạt đến 5V mà dao động trong khoảng từ 3.5V đến 4.9V và dòng xuất ra rất nhỏ dưới 5mA (*P0,P2 dòng xuất khoảng 1mA; P1,P3 dòng xuất ra khoảng 1mA đến 5mA*) vì vậy dòng xuất này không đủ để có thể làm led sáng. Tuy nhiên khi các Port xuất tín hiệu ở mức logic 0 dòng điện cho phép đi qua lớn hơn rất nhiều:

Chân Vi điều khiển khi ở mức 0:

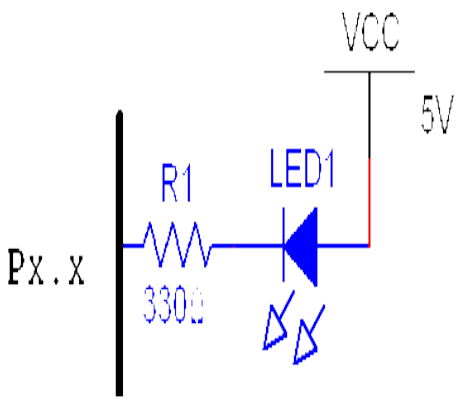
Dòng lớn nhất qua P0 : -25mA

Dòng lớn nhất qua P1,P2,P3 : -15mA

- Do đó khi kết nối với led hoặc các thiết bị khác Vi điều khiển sẽ

gặp trở ngại là nếu tác động làm led sáng khi Vi điều khiển xuất ở mức 1, lúc này dòng và áp ra không đủ để led có thể sáng rõ (led đỏ sáng ở điện áp 1.6V-2.2V và dòng trong khoảng 10mA). Khắc phục bằng các cách sau:

3.2.1. Cho led sáng khi Vi điều khiển ở mức 0 (hình 2.9):



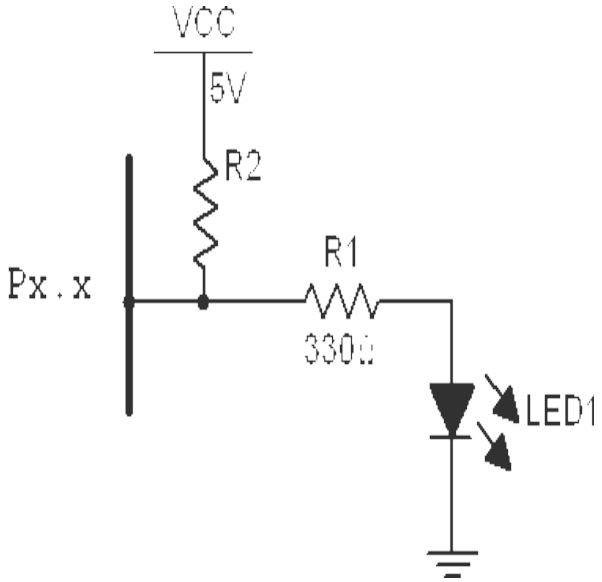
Px.x thay cho các chân xuất của các Port. Ví dụ: Chân P1.1, P2.0, v.v...

Khi Px.x ở mức 1 led không sáng

Khi Px.x ở mức 0 led sáng

Hình 2.9. Cho led sáng khi Vi điều khiển xuất ở mức 1

Như đã trình bày vì ngõ ra Vi điều khiển khi xuất ở mức 1 không đủ để cho led sáng, để led sáng được cần đặt thêm một điện trở kéo lên nguồn VCC (gọi là điện trở treo) hình 2.10.



Hình 2.10 Mạch dùng Trở kéo lên

Tuỳ từng trường hợp mà chọn R2 để dòng và áp phù hợp với thiết bị nhận.

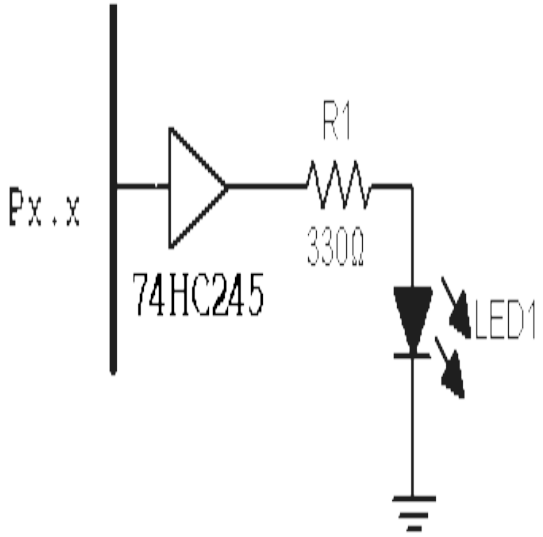
Khi Px.x ở mức 0, có sự chênh lệch áp giữa nguồn VCC và chân Px.x -dòng điện đi từ VCC qua R2 và Px.x về Mass, do đó hiệu điện thế giữa hai chân led gần như bằng 0, led không sáng.

Khi Px.x ở mức 1 (+5V),dòng điện không chạy qua chân Vi điều khiển để về mass được, có sự lệch áp giữa hai chân led, dòng điện trong trường hợp này qua led về Mass do đó led sáng.

R2 thường được sử dụng với giá trị từ 4.7KΩ đến 10KΩ. Nếu tất cả các chân trong 1 Port đều kết nối để tác động ở mức cao thì điện trở R2 có thể thay bằng điện trở thanh 9 chân vì nó có hình dáng và sử dụng dễ hơn khi làm mạch điện.

3.2.2. Ngoài cách sử dụng điện trở treo.

Việc sử dụng *cổng đệm* cũng có tác dụng thay đổi cường độ dòng điện xuất ra khi ngõ ra ở mức 1, cổng đệm xuất ra tín hiệu ở mức 1 với áp và dòng lớn khi có tín hiệu mức 1 đặt ở ngõ vào (hình 2.11). Tùy theo yêu cầu của người thiết kế về dòng và áp cần thiết mà chọn IC đệm cho phù hợp. Chẳng hạn từ một ngõ ra P0.0 làm nhiều led sáng cùng lúc thì việc sử dụng IC đệm được ưu tiên hơn. Có thể sử dụng 74HC244 hoặc 74HC245, tuy nhiên 74HC245 được cải tiến từ 74HC244 nên việc sử dụng 74HC245 dễ dàng hơn trong thiết kế mạch.

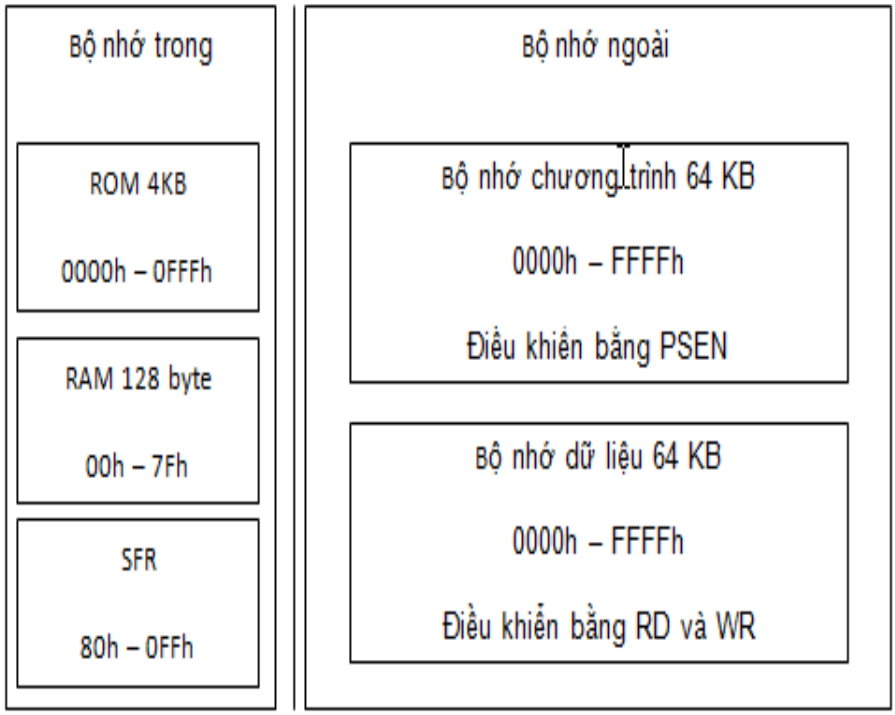


Hình 2.11. Mạch dùng cổng đệm

4. Tổ chức bộ nhớ.

Mục tiêu: Hiểu các chức năng và địa chỉ của bộ nhớ

4.1. Tổng quan tổ chức bộ nhớ (hình 2.[12])



Hình 2.12. Tổ chức bộ nhớ họ MCS-51

- 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.

- Các chip vi điều khiển được dùng làm thành phần trung tâm trong các thiết kế hướng điều khiển, trong đó bộ nhớ có dung lượng giới hạn, không có ổ khóa và hệ điều hành. Chương trình điều khiển phải thường trú trong ROM nên 8051 có không gian bộ nhớ riêng cho chương trình và dữ liệu, cả hai bộ nhớ chương trình và dữ liệu đều đặt trong chip, tuy nhiên ta có thể mở rộng bộ nhớ chương trình và bộ nhớ dữ liệu bằng cách sử dụng các chip nhớ bên ngoài với dung lượng tối đa là 64K.
- Bộ nhớ nội trong chip bao gồm ROM và RAM. RAM trên chip bao gồm vùng RAM đa chức năng (general purpose RAM : 30H-7FH), vùng RAM với từng bit được định địa chỉ (bit address locations gọi tắt là vùng RAM định địa chỉ bit: 20H-2FH), các dãy thanh ghi (bank : 00H-1FH) và các thanh ghi chức năng đặc biệt SFR (spectial function register : 80H-FFH).
- Bộ nhớ ROM:

Bộ nhớ ROM dùng để lưu chương trình do người viết chương trình viết ra. Chương trình là tập hợp các câu lệnh thể hiện các thuật toán

để giải quyết các công việc cụ thể, chương trình do người thiết kế viết trên máy vi tính, sau đó được đưa vào lưu trong ROM của vi điều khiển, khi hoạt động, vi điều khiển truy xuất từng câu lệnh trong ROM để thực hiện chương trình. ROM còn dùng để chứa số liệu các bảng, các tham số hệ thống, các số liệu cố định của hệ thống. Trong quá trình hoạt động nội dung ROM là cố định, không thể thay đổi, nội dung ROM chỉ thay đổi khi ROM ở chế độ xóa hoặc nạp chương trình (do các mạch điện riêng biệt thực hiện).

Bộ nhớ ROM được tích hợp trong chip Vi điều khiển với dung lượng tùy vào chủng loại cần dùng, chẳng hạn đối với 89S52 là 8KByte, với 89S53 là 12KByte.

Bộ nhớ bên trong Vi điều khiển 89Sxx là bộ nhớ Flash ROM cho phép xóa bộ nhớ ROM bằng điện và nạp vào chương trình mới cũng bằng điện và có thể nạp xóa nhiều lần.

Bộ nhớ ROM được định địa chỉ theo từng Byte, các byte được đánh địa chỉ theo số hex-số thập lục phân, bắt đầu từ địa chỉ 0000H, khi viết chương trình cần chú ý đến địa chỉ lớn nhất trên ROM, chương trình được lưu sẽ bị mất khi địa chỉ lưu vượt qua vùng này. Ví dụ: AT89S52 có 8KByte bộ nhớ ROM nội, địa chỉ lớn nhất là 1FFFH, nếu chương trình viết ra có dung lượng lớn hơn 8KByte các byte trong các địa chỉ lớn hơn 1FFFH sẽ bị mất.

Ngoài ra, Vi điều khiển còn có khả năng mở rộng bộ nhớ ROM với việc giao tiếp với bộ nhớ ROM bên ngoài lên đến 64Kbyte (địa chỉ từ 0000H đến FFFH).

- *Bộ nhớ RAM:*

  Bộ nhớ RAM dùng làm môi trường xử lý thông tin, lưu trữ các kết quả trung gian và kết quả cuối cùng của các phép toán, xử lí thông tin. Nó cũng dùng để tổ chức các vùng đệm dữ liệu, trong các thao tác thu phát, chuyển đổi dữ liệu.

  RAM nội trong Vi điều khiển được tổ chức như sau:

  - Các vị trí trên RAM được định địa chỉ theo từng Byte bằng các số thập lục phân (số Hex).
  - Các bank thanh ghi có địa chỉ 00H đến 1FH.
  - 210 vị trí được định địa chỉ bit.
  - Các vị trí RAM bình thường

  Các thanh ghi có chức năng đặc biệt có địa chỉ từ 80H đến FFH.

Các byte RAM 8 bit của vi điều khiển được gọi là "ô nhớ ", nếu các ô nhớ có chức năng đặc biệt thường được gọi là “thanh ghi”, nếu là bit

thì được gọi là “bit nhớ”.

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

RAM

Địa chỉ byte	Địa chỉ bit								
FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	D1	D0	PSW
B8	-	-	-	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	-	-	AC	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	Không được địa chỉ hóa bit								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	Không được địa chỉ hóa bit								TH1
8C	Không được địa chỉ hóa bit								TH0
8B	Không được địa chỉ hóa bit								TL1
8A	Không được địa chỉ hóa bit								TL0
89	Không được địa chỉ hóa bit								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	Không được địa chỉ hóa bit								PCON
83	Không được địa chỉ hóa bit								DPH
82	Không được địa chỉ hóa bit								DPL
81	Không được địa chỉ hóa bit								SP
80	87	86	85	84	83	82	81	80	P0

CÁC THANH GHI CHỨC NĂNG ĐẶC BIỆT

4.2. Vùng RAM đa năng.

Vùng RAM đa mục đích có 80 byte đặt ở địa chỉ từ 30H – 7FH. Bất kỳ vị trí nhớ nào trong vùng RAM đa mục đích đều có thể được truy xuất tự
do bằng cách sử dụng các kiểu định địa chỉ trực tiếp hoặc gián tiếp.

Ví dụ: MOV A,5FH
MOV R0,5FH
MOV A,@R0

4.3. Vùng RAM định địa chỉ bit.

8051 có 210 vị trí bit được định địa chỉ trong đó 128 bit chứa trong các byte ở địa chỉ từ 20H - 2FH và phần còn lại chứa trong các thanh ghi chức năng đặc biệt.

Các dãy thanh ghi:

32 vị trí thấp nhất của bộ nhớ nội chứa các dãy thanh ghi. Các lệnh của 8051 hỗ trợ 8 thanh ghi từ R0 – R7 thuộc dãy 0 (bank 0). Đây là dãy mặc định sau khi reset hệ thống. Các thanh ghi này ở các địa chỉ từ 00H–
07H.

Ví dụ: MOV A,R5 = MOV A,05H

Các lệnh sử dụng các thanh ghi từ R0 – R7 là các lệnh ngắn và thực hiện nhanh hơn so với các lệnh tương đương sử dụng kiểu định địa chỉ trực tiếp. Các giá trị thường được sử dụng nên chứa ở một trong các thanh ghi này. Dãy thanh ghi đang được sử dụng được gọi là dãy thanh ghi tích cực. Dãy thanh ghi tích cực có thể được thay đổi bằng cách thay đổi các bit chọn dãy trong thanh ghi PSW.

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

*Mục tiêu:*biết được chức năng của các thanh ghi.

Các thanh ghi nội của 8051 được cấu hình thành một phần của RAM trên chip, do vậy mỗi thanh ghi cũng có một địa chỉ. Cũng như các thanh ghi từ R0 – R7 ta có 21 thanh ghi chức năng đặc biệt SFR chiếm phần trên của RAM nội từ địa chỉ 80H – FFH. Lưu ý không phải tất cả 128 địa chỉ từ
80H – FFH đều được định nghĩa mà chỉ có 21 địa chỉ được định nghĩa.

5.1. Từ trạng thái chương trình PSW (program status word).

Thanh ghi PSW có địa chỉ là D0H chứa các bit trạng thái có chức năng được tóm tắt trong bảng sau:

BIT	KÝ	ĐỊA	MÔ TẢ
PSW.7	C hoặc	D7H	Cary Flag: Cờ nhớ
PSW.6	AC	D6H	Auxiliary Cary Flag: Cờ nhớ phụ
PSW.5	F0	D5H	Flag 0 còngọi là cờ Zero kí hiệu là Z
PSW4	RS1	D4H	Register Bank Select 1: bit lựa chọn bank thanh
PSW.3	RS0	D3H	Register Bank Select 0: bit lựa chọn bank thanh
			00 = Bank 0; ô nhớ có address 00H÷07H gán cho
			01 = Bank 1; ô nhớ có address 08H÷0FH gán cho
			10 = Bank 2; ô nhớ có address 10H÷17H gán cho
			11 = Bank 3; ô nhớ có address 18H÷1FH gán cho
PSW.2	OV	D2H	Overflow Flag: cờ tràn số nhị phân có dấu.
PSW.1	-	D1H	Reserved: chưa thiết kế nên chưa sử dụng được.
PSW.0	P	D0H	Even Parity Flag: cờ chẵn lẻ.

5.2. Thanh ghi B.

Thanh ghi B ở địa chỉ F0H được dùng chung với thanh chứa A trong các phép toán nhân (MUL), chia (DIV).Các bit của thanh ghi B được định địa chỉ từ F0H-F7H.

5.3. Con trỏ Stack.

Con trỏ Stack SP (stack pointer) là một thanh ghi 8 bit ở địa chỉ 81H. SP chứa địa chỉ của dữ liệu hiện đang ở đỉnh của Stack. Các lệnh liên quan đến Stack bao gồm lệnh cất dữ liệu vào Stack (PUSH) và lệnh lấy dữ liệu ra khỏi Stack (POP). Việc cất dữ liệu vào Satck làm tăng thanh ghi SP trước khi ghi dữ liệu và việc lấy dữ liệu ra Stack sẽ làm giảm thanh ghi SP.Nếu ta không khởi động SP, nội dung mặc định của thanh ghi này là 07H.

Vùng stack của 8051 được giữ trong RAM nội và được giới hạn đến

các địa chỉ truy xuất được bởi kiểu định địa chỉ gián tiếp. Các lệnh PUSH và POP sẽ cất dữ liệu vào stack và lấy dữ liệu từ stack, các lệnh gọi chương trình con (ACALL, LCALL) và lệnh trở về (RET, RETI) cũng cất và phục hồi nội dung của bộ đếm chương trình PC (Program counter).

5.4. Con trỏ dữ liệu DPTR.

Con trỏ dữ liệu DPTR (data pointer) được dùng để truy xuất bộ nhớ chương trình ngoài hoặc bộ nhớ dữ liệu ngoài. DPTR là thanh ghi 16 bit có địa chỉ là 82H (DPL, byte thấp) và 83H (DPH, byte cao).

```
Ex:   MOV A,#55H
      MOV DPTR,#1000H
      MOV @DPTR,A
```

5.5. Các thanh ghi Port nối tiếp.

Các port xuất nhập của 8051 bao gồm port 0 tại địa chỉ 80H, port 1 tại địa chỉ 90H, port 2 tại địa chỉ A0H và port 3 tại địa chỉ 0BH. Các port 0,2 và 3 không được dùng để xuất/nhập nếu ta sử dụng thêm bộ nhớ ngoài hoặc nếu có một số đặc tính của 8051 được sử dụng (như là ngắt, port nối tiếp).Tất cả các port đều được định địa chỉ từng bit nhằm cung cấp các khả năng giao tiếp mạnh.

5.6. Các thanh ghi định thời.

8051 có hai bộ đếm định thời (timer/counter) 16 bit để định các khoảng thời gian hoặc để đếm các sự kiện.Bộ định thời 0 có địa chỉ 8AH(TL0,byte thấp) và 8CH (TH0,byte cao), bộ định thời 1 có địa chỉ 8BH (TL1,byte thấp) và 8DH (TH1,byte cao).

Hoạt động của bộ định thời được thiết lập bởi thanh ghi chế độ định thời TMOD (timer mode register) ở địa chỉ 89H và thanh ghi điều khiển định thời TCON (timer control regsiter) ở địa chỉ 88H.

5.7. Các thanh ghi port nối tiếp (Serial Data Buffer).

- Bên trong 8051 có một port nối tiếp để truyền thông với các thiết bị nối tiếp như các thiết bị đầu cuối hoặc moderm, hoặc để giao tiếp với các IC khác. Một thanh ghi được gọi là bộ đệm dữ liệu nối tiếp SBUF (serial data buffer) ở địa chỉ 99H lưu trữ dữ liệu truyền đi và dữ liệu nhận về. Việc ghi lên SBUF sẽ nạp dữ liệu để truyền và việc đọc SBUF sẽ lấy dữ liệu đã nhận được. Khi dữ liệu được chuyển vào thanh ghi SBUF, dữ liệu sẽ được chuyển vào bộ đệm truyền dữ liệu và sẽ được

lưu giữ ở đó cho đến khi quá trình truyền dữ liệu qua truyền thông nối tiếp kết thúc. Khi thực hiện việc chuyển dữ liệu từ SBUF ra ngoài, dữ liệu sẽ được lấy từ bộ đệm nhận dữ liệu của truyền thông nối tiếp.

- Các chế độ hoạt động khác nhau được lập trình thông qua thanh ghi điều khiển port nối tiếp SCON (serial port control register) ở địa chỉ 98H. Chỉ có TCON được định địa chỉ từng bit.

5.8. Các thanh ghi ngắt.

8051 có một cấu trúc ngắt với hai mức ưu tiên và năm nguyên nhân ngắt. Các ngắt bị vơ hiệu hóa sau khi reset hệ thống và sau đó được cho phép ngắt bằng cách ghi vào thanh ghi cho phép ngắt IE (interrupt enable register) ở địa chỉ A8H. Mức ưu tiên ngắt được thiết lập qua thanh ghi ưu tiên ngắt IP (interrupt priority register) ở địa chỉ B8H.Cả hai thanh ghi này đều được định địa chỉ từng bit.

5.9.Thanh ghi điều khiển nguồn PCON.

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

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

- SMOD1 (Serial Mode 1): = 1 cho phép tăng gấp đơi tốc độ port nối tiếp trong chế độ 1, 2 và 3.
- SMOD0 (Serial Mode 0): cho phép chọn bit SM0 hay FE trong thanh ghi SCON (= 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 xóa bit POF trước đó.
- GF1, GF0 (General purpose Flag): các bit cờ dành cho người sử dụng.
- PD (Power Down): được xóa 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 thốt 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 xóa 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 thốt 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 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.

6. Tổ chức bộ nhớ ngoài.

Mục tiêu: Hiểu được cấu trúc giao tiếp với bộ nhớ ngoài của vi điều khiển.

- 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.

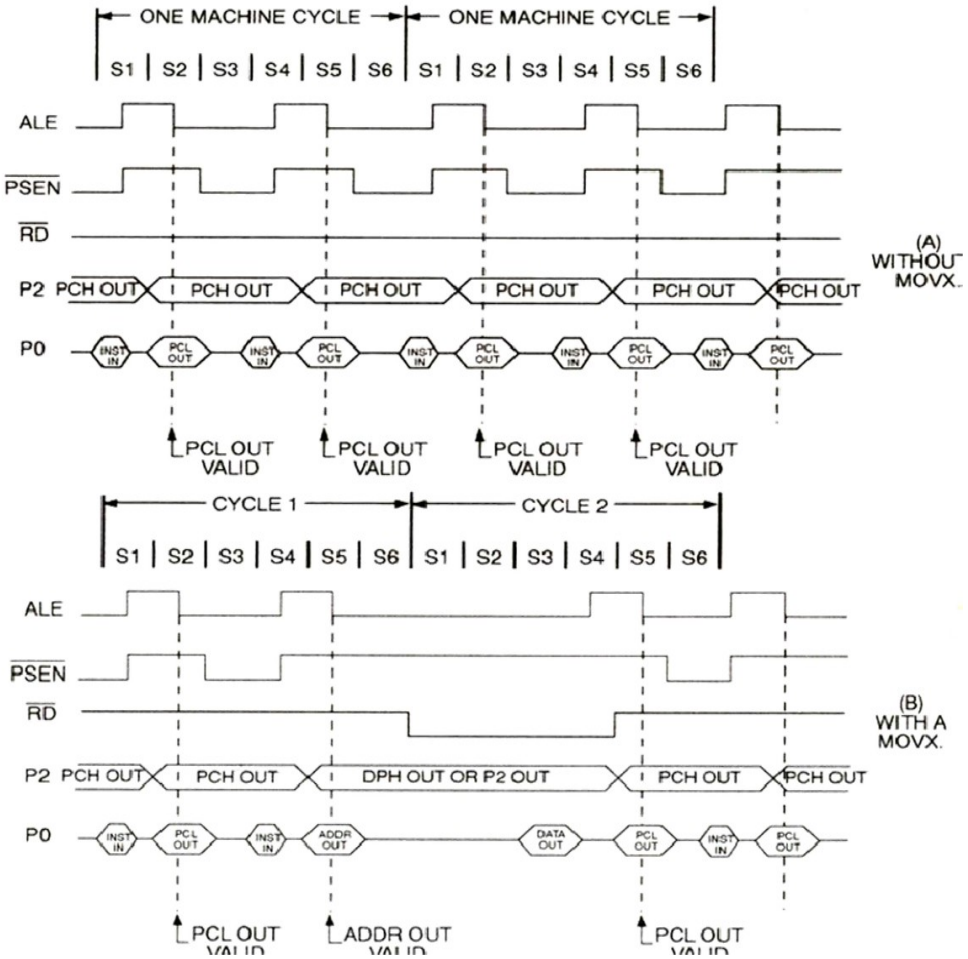
- Lưu ý rằng việc truy xuất bộ nhớ chương trình luôn luôn sử dụng địa chỉ 16 bit còn bộ nhớ dữ liệu có thể là 8 bit hay 16 bit tuỳ theo câu lệnh sử dụng. Khi dùng bộ nhớ dữ liệu 8 bit thì có thể dùng Port 2 như là Port I/O thông thường còn khi dùng ở chế độ 16 bit thì Port 2 chỉ dùng làm các bit địa 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.(hình 2.13)



Hình 2.13. Thực thi bộ nhớ chương trình ngoài

PCH: Program Counter High – PCL: Program Counter Low

DPH: Data Pointer High – DPL: Data Pointer Low

6.1. Truy xuất bộ nhớ chương trình ngoài.

- Quá trình thực thi lệnh khi dùng bộ nhớ chương trình ngoài có thể mô tả như h inh2.8: 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.

- 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.

6.2. Truy xuất 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).

6.3.Giải mã địa chỉ.

- Trong các ứng dụng dựa trên 8051, ngoài giao tiếp bộ nhớ dữ 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 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, 74138, 74154. Ngõ ra của các IC giải mã sẽ được đưa tới chân chọn chip hay bộ đệm khi điều khiển ngoại vi.

7. Các cải tiến của 8032/8052.

Các vi mạch 8052 (và các phiên bản CMOS) có hai cải tiến so với 8051. Một là có thêm 128 byte RAM trên chip từ địa chỉ 80H-FFH. Điều này không xung đột với các thanh ghi chức năng đặc biệt (có cùng địa chỉ) vì 128 byte Ram thêm vào chỉ có thể truy xuất bằng cách dùng kiểu định địa chỉ gián tiếp.

```
Ví dụ:    MOV A,#100
          MOV R0,#0F0H ( Trùng với địa chỉ của thanh ghi B )
          MOV A,@R0
```

Cải tiến thứ hai là có thêm bộ định thời 16 bit Timer 2.

8. Hoạt động Reset.

Mục tiêu: Hiểu được chức năng hoạt động của chân Reset.

8051 được Reset bằng cách giữ chân RST ở mức cao tối thiểu 2 chu kỳ máy và sau đó chuyển về mức thấp. RST có thể được tác động tay hoặc được tác động khi cấp nguồn bằng cách dùng một mạch RC.

Trạng thái của các thanh ghi sau khi reset như sau :

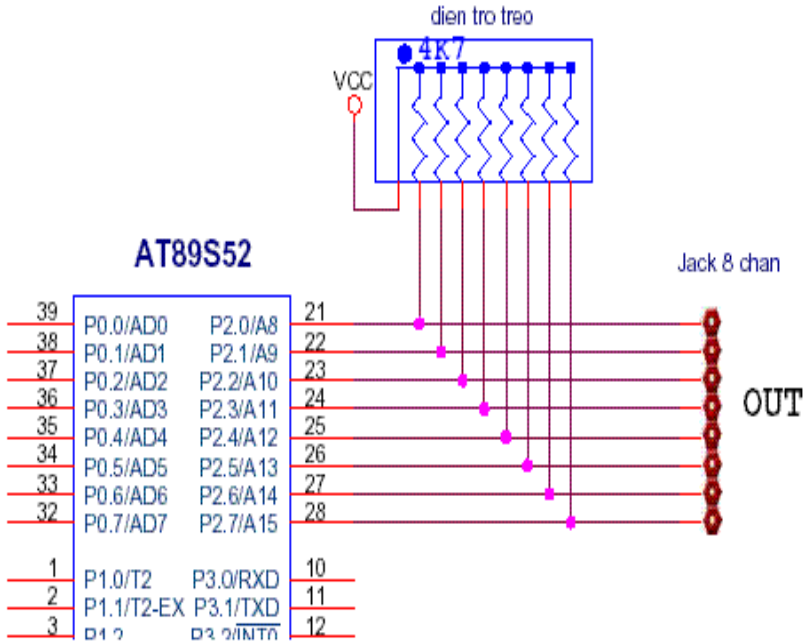
Thanh ghi	Nội dung
Bộ đếm chương trình	0000H
Thanh chứa A	00H
Thanh ghi B	00H
PSW	00H
SP	07H
DPTR	0000H
Port 0-3	FFH
IP	xxx00000B (8051)
IE	0xx00000B (8051)
Các thanh ghi định thời	00H
SCON	00H
SBUF	00H
PCON (HMOS)	0xxxxxxxB
PCON (CMOS)	0xxx0000B

CÁC BÀI TẬP MỞ RỘNG, NÂNG CAO VÀ GIẢI QUYẾT VẤN ĐỀ

Bài 1: Giải thích tại sao thường phải có điện trở kéo lên (pull-up) tại Port 0? Trường hợp nào không cần sử dụng điện trở này?

Gợi ý:

Do Port la`dạng cực maṅg hở dunàg mosfet do đo´phải dunàg điện trở keó lên khi sử dụng chưć năng I/O.



Bai 2: Ve~sơ đô`chân va`giải thích chức năng của cac chân trong Vi điêù khiển 8051?

Bai giải:

❖ Sơ đô`chân của Vi điêù khiển 8051

❖ Giải thích chức năng của cac chân trong vi điêù khiển 8051:

Chíp 8051 có 40 chân, mỗi chân có một kí hiệu tên và có các chức năng như sau:

- Chân 40: nối với nguồn nươi +5V.
- Chân 20: nối với đất(Mass, GND).
- Chân 29 (PSEN)(program store enable) là tín hiệu điều khiển xuất ra của 8051, nó cho phép chọn bộ nhớ ngoài và được nối chung với chân của OE (Outout Enable) của EPROM ngoài để cho phép đọc các byte của chương trình. Các xung tín hiệu PSEN hạ thấp trong suốt thời gian thi hành lệnh. Những mã nhị phân của chương trình được đọc từ EPROM đi qua bus dữ liệu và được chốt vào thanh ghi lệnh của 8051 bởi mã lệnh. (chú ý việc đọc ở đây là đọc các lệnh (khác với đọc dữ liệu), khi đó VXL chỉ đọc các bit opcode của lệnh và đưa chúng vào hàng đợi lệnh thông qua các Bus địa chỉ và dữ liệu).
- Chân 30 (ALE : Adress Latch Enable) là tín hiệu điều khiển xuất ra của 8051, nó cho phép phân kênh bus địa chỉ và bus dữ liệu của Port 0.
- Chân 31 (EA : Eternal Acess) được đưa xuống thấp cho phép chọn bộ nhớ mã ngoài. Đối với 8051 thì:

EA = 5V : Chọn ROM nội.

EA = 0V : Chọn ROM ngoại.

- 32 chân còn lại chia làm 4 cổng vào / ra: có thể dùng chân đó để để đọc mức logic (0;1 tương ứng với 0V; 5V)vào hay xuất mức logic ra (0;1), (hình 2.3)

Port 0 từ chân 39 - 32 tương ứng là các chân P0.0 đến P0.7.

Port 1 từ chân 1- 8 tương ứng là các chân P1.0 đến P1.7.

Port 2 từ chân 21- 28 tương ứng là các chân P2.0 đến P2.7.

Port 1 từ chân 10- 17 tương ứng là các chân P3.0 đến P3.7.

- Chip 8051 có 32 chân xuất / nhập,tuy nhiên có 24 chân trong 32 chân này có 2 mục đích.Mỗi một chân này có thể hoạt động ở chế độ xuất/nhập, hoạt động điều khiển hoặc hoạt động như một đường địa chỉ / dữ liệu của bus địa chỉ / dữ liệu đa hợp.

Bai 3: Nêu chức năng port I/O?

Bai giải:

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

Port 0:

- Chức năng IO (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 ra, Port 0 có thể kéo được 8 ngõ TTL.

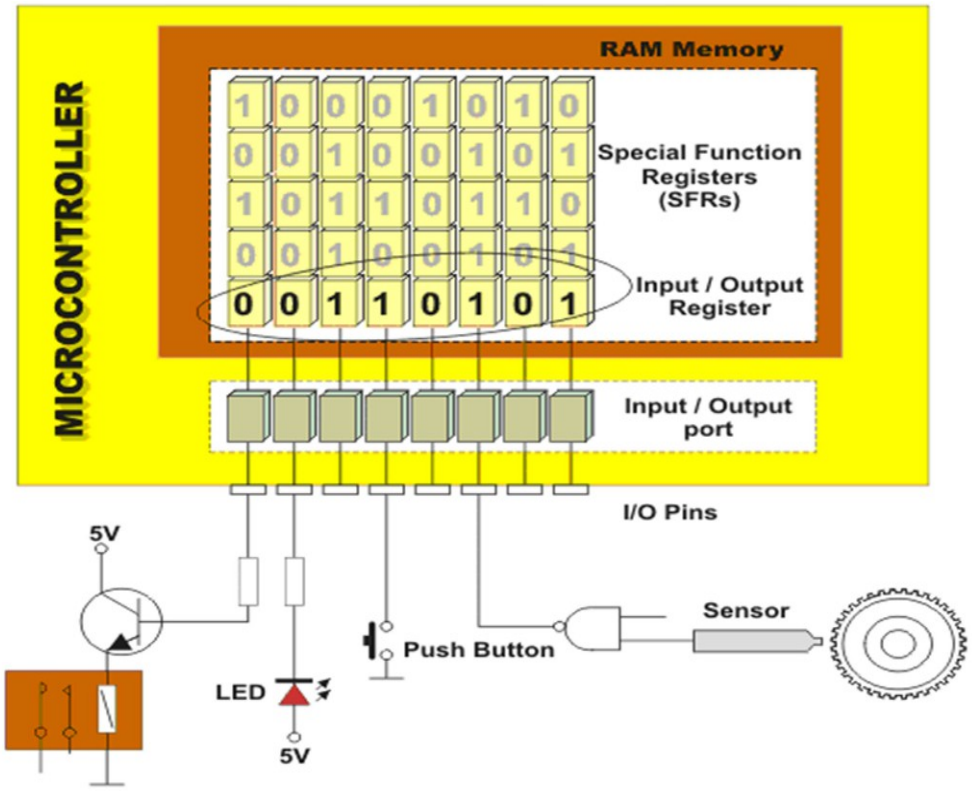
Khi dùng làm ngõ vào, Port 0 phải được set mức logic 1 trước đó.

- 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 mà 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 IO, 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.



Hình ảnh vào ra với thiết bị ngoại vi

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 IO (xuất/nhập): có khả năng kéo được 4 ngõ TTL.
- 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.
- Khi lập trình, Port 2 dùng làm 8 bit địa chỉ cao hay một số tín hiệu điều khiển.

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

- Chức năng IO: có khả năng kéo được 4 ngõ TTL.
- Khi dùng làm ngõ vào, Port 3 phải được set mức logic 1 trước đó.

Bai 3: Cać địa chỉ bit naò được lập sau chuoĩ lệnh sau:

```
MOV     R0,#26h
MOV     @R0,#7AH
```

Gợi yí: bit 53H

Bai 4: Mô tả chuoĩ lệnh ghi giaítrị 0ABH vaò địa chỉ 9A00H của RAM ngoaì trong hệ thống 8051.

Gợi y:

```
MOV     DPTR,9A00H
MOV     A,#0ABH
MOVX    @DPTR,A
```

Bai 5: Tín hiệu điều khiển nao của 8051 sử dụng để chọn cac bộ nhơ´ EPROM va`RAM ngoai.

Gợi y: PSEN

Bai 6: Viet lệnh lập bit thấp nhất của thanh ghi chưá ma`không ảnh hưởng tơí cac bit khac.

Gợi y: OR A,#01

Bai 7: Viet chuoĩ lệnh chep nội dung thanh ghi R7 vao địa chỉ 100H của bộ nhơ´RAM ngoai.

Gợi y:

```
MOV     R1,#50H
MOV     A,@R1
```

Bai 8: Giả sử lệnh đâu tiên thực hiện sau khi reset hệ thống la`một lệnh gọi chương trình con, thanh ghi PC se˜được chưá vao địa chỉ nao của RAM nội trươć khi được chuyển tơí chương trình con.

Gợi y:

Địa chỉ 08H va`09H

Bai 9: Lệnh nao chuyển 8051 qua chê´độ power-down.

Gợi y:

Lệnh ghi bit tương ưng trong thanh ghi PCON.

Bai 10: Viet chuoĩ lệnh gia´trị trong ô nhơ´co´địa chỉ 50H của RAM nội vao thanh ghi chưá sử dụng định vị địa chỉ giań tiep.

Gợi y:

```
MOV     R1,#50H
MOV     A,@R1
```

Bai 11: Tính gia´trị offset tương đôí cho lệnh SJMP AHEAD, giả sử lệnh nay nằm tại địa chỉ A050H va`A051H, va`nhañ AHEAD biểu diễn cho lệnh nằm tại địa chỉ 9FE0H.

Gợi y: 8FH

Điêù kiện thực hiện bai học:

IC họ 8051 - CMOS, TTL – 555.

Sơ đồ, IC họ 8051.

Tai liệu vi điều khiển, vi mạch số các loại.

Máy vi tính, mỏ hàn, kềm cắt, kềm nhọn.

Đồng hồ DVOM/VOM.

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức: cấu tạo, đặc điểm,ứng dụng của các loại Vi điều khiển

+ Về kỹ năng:

- Thực hiện viết các chương trình theo yêu cầu cho trước.

+ Thái độ: Đánh giá phong cách, thái độ học tập

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm

+ Về kỹ năng: Đánh giá kỹ năng thực hành môĩi sinh viên, hoặc mỗi nhóm học viên thực hiện công việc theo yêu cầu của giáo viên.Tiêu chí đánh giá theo các nội dung:

- Độ chính xác của công việc
- Tính thẩm mỹ của mạch điện
- Độ an toàn trên mạch điện
- Thời gian thực hiện công việc
- Độ chính xác theo yêu cầu kỹ thuật

+ Thái độ: Tỉ mỉ, cẩn thận, chính xác.

BAI 3
TẬP LỆNH 8051
Mã bài: MĐ24-03

Giới thiệu:

- Với kiến thức về cấu trúc bên trong của vi điều khiển và để điều khiển hoạt động khối mạch điện giao tiếp phức tạp của hệ thống. 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 điều khiển 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 điều khiển, bộ nhớ, các thiết bị ngoại vi. Sau khi kết nối vi điều khiển với các thiết bị ngoại vi, ta phải tìm hiểu về các tập lệnh của MSC-51 để điều khiển hoạt động hệ thống.

- Chương này giới thiệu cách thức lập trình trên MCS-51 cũng như giải thích hoạt động của các lệnh sử dụng cho họ MCS-51

Mục tiêu của bài:

- Phân biệt được các kiểu định địa chỉ và dữ liệu.
- Trình bày được đặc tính và công dụng của từng lệnh trong 8051.
- Xác định được độ lớn và thời gian thực hiện chương trình.
- Kết hợp được các lệnh riêng lẻ để thực hiện thao tác cho trước đuúng kỹ thuật.

Nội dung chính:

1. Mở đầu

Mục tiêu : Hiểu được các cú pháp lệnh và cách khai báo dữ liệu trong ngôn ngữ lập trình.

1.1. Cú pháp lệnh.

Một lệnh trong chương trình hợp ngữ có dạng như sau:

Nhãn	Lệnh	Toán	Chú thích
A:	MOV	A, #10h	; Đưa giá trị 10h vào thanh
LED	EQU	30h	; Định nghĩa ô nhớ chứa
On_Led	BIT	00h	; Cờ trạng thái led

- Trường nhãn định nghĩa các ký hiệu (có thể là địa chỉ trong chương trình, các hằng dữ liệu, tên đoạn hay các cấu trúc lập trình). Trường nhãn không bắt đầu bằng số và không trùng với các từ khóa có

sẵn.

- Trường lệnh chứa các từ gợi nhớ cho các lệnh của MCS-51 hay các lệnh giả dùng cho chương trình dịch.
- Trường toán hạng chứa các thông số liên quan đến lệnh đang sử dụng.
- Trường chú thích dùng để ghi chú trong chương trình hợp ngữ.
- Trường này phải được bắt đầu bằng dấu; và chương trình dịch sẽ bỏ qua các từ đặt sau dấu ;.

Lưu ý rằng các chương trình dịch không phân biệt chữ hoa và chữ thường.

1.2. Khai báo dữ liệu.

- Khi khai báo hằng số, chữ h cuối cùng xác định hằng số là số thập lục phân; chữ b cuối cùng xác định số nhị phân và chữ d cuối (hay không có) xác định số thập phân. Lưu ý rằng đối với số thập lục phân, khi bắt đầu bằng chữ A → F thì phải thêm số 0 vào phía trước.

Ví dụ:

1010b; Số nhị phân

1010h; Số thập lục phân

1010; Số thập phân

0F0h ; Số thập lục phân nhưng bắt đầu bằng chữ F nên phải thêm vào phía trước số 0.

- Khi dùng dấu # phía trước một con số, đó chính là dữ liệu tức thời cònnếu không dùng dấu # thì đó là địa chỉ của ô nhớ. Lưu ý rằng khi dùng RAM nội thì chỉ dùng địa chỉ từ 00 – 7Fh còn vùng địa chỉ từ 80h – 0FFh dùng cho các thanh ghi chức năng đặc biệt. Đối với họ 89x52, RAM nội có 256 byte thì các byte địa chỉ cao (từ 80h – 0FFh) không thể truy xuất trực tiếp mà phải truy xuất gián tiếp.

Ví dụ:

```
MOV A,30h       ; Chuyển nội dung ô nhớ 30h vào A
MOV A,#30h      ; Chuyển giá trị 30h vào A
MOV A,80h       ; Chuyển nội dung Port 0 vào A

MOV R0,#80h     ; Chuyển nội dung ô nhớ 80h vào A (chỉ
MOV A,@R0       ; dùng cho họ 89x52)
```

- Để định nghĩa trước một vùng nhớ trong bộ nhớ chương trình, có thể dùng các chỉ dẫn DB (define byte – định nghĩa 1 byte) hay DW (define word – định nghĩa 2 byte).

Ví dụ: Định nghĩa trước dữ liệu cho led như sau:

```
Led: DB   01h,02h,04h,08h,10h,20h,40h,80h
```

Đoạn chương trình này xác định tại nhãn Led có chứa các giá trị lần lượt từ 01h đến 80h. Nếu nhãn Led đặt tại địa chỉ 100h thì giá trị tương ứng như sau (bảng 3.1):

Địa chỉ	Giá trị
100h	01h
101h	02h
102h	04h
103h	08h
104h	10h
105h	20h
106h	40h
107h	80h

Bảng 3.1

- Để dễ nhớ và dễ hiểu khi lập trình, các chương trình dịch cho phép dùng các ký tự thay thế cho các ô nhớ bằng các lệnh giả EQU, BIT.

Ví dụ:

```
LED   EQU  30h
ON_LED    BIT   00h
```

Giả sử chương trình hợp ngữ có các lệnh sau:

```
MOV       A,LED
SETB ON_LED
```

Khi biên dịch, chương trình dịch sẽ tự động chuyển thành dạng lệnh sau:

```
MOV       A,30h
SETB 00h
```

Các ký hiệu cần chú ý:

Rn : các thanh ghi từ R0 – R7 (bank thanh ghi hiện hành).

Ri : các thanh ghi từ R0 – R1 (bank thanh ghi hiện hành).

@Rn : định địa chỉ gián tiếp 8 bit dùng thanh ghi Rn.

@DPTR : định địa chỉ gián tiếp 16 bit dùng thanh ghi DPTR.

direct : định địa chỉ trực tiếp RAM nội (00h – 7Fh) hay SFR (80h – FFh) (direct).

: nội dung của bộ nhớ tại địa chỉ direct

#data8 : giá trị tức thời 8 bit

#data16 : giá trị tức thời 16 bit

bit : địa chỉ bit của các ô nhớ có thể định địa chỉ bit (00h – 7Fh đối với địa chỉ bit và 20h – 2Fh đối với địa chỉ byte)

2. Các phương pháp định địa chỉ.

Mục tiêu : Biết được các cách định địa chỉ cho các bit, byte hoặc thanh ghi.

Các kiểu định địa chỉ cho phép định rõ nơi lấy dữ liệu hoặc nơi nhận dữ liệu tùy thuộc vào cách thức sử dụng lệnh của người lập trình. Vi điều khiển 8051 có 8 kiểu định địa chỉ như sau:

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

2.1. Định địa chỉ bằng thanh ghi (hình 3.1)

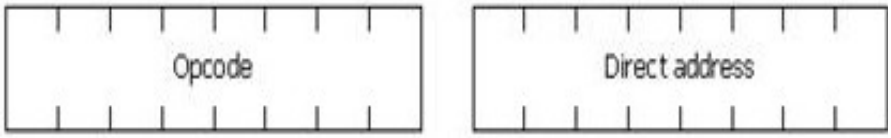
Hình 3.1

- Các thanh ghi từ R0 – R7 có thể truy xuất bằng cách định địa chỉ trực tiếp hay gián tiếp như trên. Ngoài ra, các thanh ghi này còn có thể truy xuất bằng cách dùng 3 bit trong mã lệnh để chọn 1 trong 8 thanh ghi (8 thanh ghi này có địa chỉ trực tiếp thay đổi tuỳ theo bank thanh ghi đang sử dụng).

- Các lệnh sử dụng kiểu định địa chỉ thanh ghi được mã hóa bằng các dùng 3 bit thấp nhất của opcode (của lệnh) để chỉ ra 1 thanh ghi bên trong không gian địa chỉ logic này. Vậy : 1 mã chức năng + địa chỉ toán hạng → 1 lệnh ngắn 1 byte.

- Kiểu này thường được dùng cho các lệnh xử lý dữ liệu mà dữ liệu luôn lưu trong các thanh ghi. Đối với vi điều khiển thì mã lệnh thuộc kiểu này chỉ có 1 byte.

2.2. Định địa chỉ trực tiếp(hình 3.2)



Hình 3.2

Định địa chỉ trực tiếp (hình 3.2) chỉ dùng cho các thanh ghi chức năng đặc biệt và RAM nội của 8951. Giá trị địa chỉ trực tiếp 8 bit được thêm vào phía sau mã lệnh. Nếu địa chỉ trực tiếp từ 00h – 7Fh thì đó là RAM nội của 8951 (128 byte), còndịa chỉ từ 80h – FFh là địa chỉ các thanh ghi

chức năng đặc biệt.

Các lệnh sau có kiểu định địa chỉ trực tiếp:

MOV A, P0

MOV A, 30h

- Trong 8051 có 128 byte bộ nhớ RAM. Bộ nhớ RAM được gán địa chỉ từ 00H đến FFH và được phân chia như sau:

Các ngăn nhớ từ 00H đến 1FH được gán cho các băng thanh ghi và ngăn xếp.

Các ngăn nhớ từ 20H đến 2FH được dành cho không gian định địa chỉ bít để lưu dữ liệu theo từng bit.

Các ngăn nhớ từ 30H đến 7FH là không gian để lưu dữ liệu có kích thước 1 byte.

Chế độ định địa chỉ trực tiếp có thể truy cập toàn bộ không gian của bộ nhớ RAM. Tuy nhiên, chế độ này thường được dùng để truy cập các ngăn nhớ RAM từ 30H đến 7FH, vì thực tế đối với không gian nhớ danh cho băng thanh ghi thì đã được truy cập bằng tên thanh ghi như R0- R7. ở chế độ định địa chỉ trực tiếp, địa chỉ ngăn nhớ RAM chứa dữ liệu là toán hạng của lệnh.

Ví dụ:

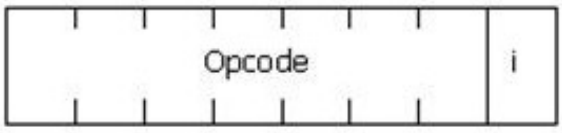
```
MOV  R0,  40     ; sao nội dung ngăn nhớ 40H của RAM vào R0
MOV  R4,  7FH    ; chuyển nội dung ngăn nhớ 7FH vào R4.
```

Một ứng dụng quan trọng của chế độ định địa chỉ trực tiếp là ngăn xếp. Trong họ 8051, chỉ có chế độ định địa chỉ trực tiếp là được phép cất và lấy dữ liệu từ ngăn xếp.

Lệnh đầu tiên chuyển nội dung từ Port 0 vào thanh ghi A. Khi biên dịch, chương trình sẽ thay thế từ gợi nhớ P0 bằng địa chỉ trực tiếp của Port 0 (80h) và đưa vào byte 2 của mã lệnh. Lệnh thứ hai chuyển nội dung của RAM nội có địa chỉ 30h vào thanh ghi A.

2.3. Định địa chỉ gián tiếp (Indirect Addressing) (hình 3.3).

Ở chế độ này, thanh ghi được dùng để trỏ đến dữ liệu có trong bộ nhớ.



Hình 3.3

Nếu dữ liệu có trên chip CPU thì chỉ các thanh ghi R0 và R1 mới được sử dụng, và như vậy cũng có nghĩa là không thể dùng các thanh ghi R2-R7 để trỏ đến địa chỉ của toán hạng ở chế độ định địa chỉ này.

Ví dụ:

MOV A, @R0 ; chuyển ngăn nhớ RAM có địa chỉ ở R0 vào A

MOV @R1, B ; chuyển B vào ngăn nhớ RAM có địa chỉ ở R1

Chú ý:

- Kiểu định địa chỉ gián tiếp được tượng trưng bởi ký hiệu @,được đặt trước các thanh ghi R0, R1,SP cho địa chỉ 8 bit (không sử dụng các thanh ghi R2 – R7 trong chế độ địa chỉ này) hay DPTR cho địa chỉ 16 bit. R0 và R1 có thể hoạt động như một thanh ghi con trỏ, nội dung của nó cho biết địa chỉ của một ô nhớ trong RAM nội mà dữ liệu sẽ ghi hoặc sẽ đọc. Còn DPTR dùng để truy xuất ô nhớ ngoại. Các lệnh thuộc dạng này chỉ có 1 byte. Tuy nhiên R0 và R1 là các thanh ghi 8 bit, nên chúng chỉ được phép truy cập đến các ngăn nhớ RAM trong, từ địa chỉ 30H đến 7FH và các thanh ghi SFR. Trong thực tế, có nhiều trường hợp cần truy cập dữ liệu được cất ở RAM ngoài hoặc không gian ROM trên chip. Trong những trường hợp đó chúng ta cần sử dụng thanh ghi 16 bit DPTR.

Ví dụ:

```
MOV A, @R1        ;copy noi dung o nho co dia chi dat
                  ;trong thanh ghi R1 vao thanh ghi A
```

2.4. Định địa chỉ tức thời (Immediate Addressing)

Khi toán hạng là một hằng số thay vì là một biến, hằng số này có thể đưa vào lệnh và đây là byte dữ liệu tức thời.

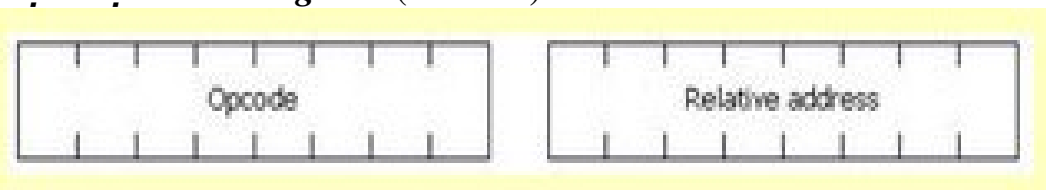
Trong hợp ngữ, các tồn hạng tức thời được nhận biết nhờ vào ký tự '#' đặt trước chúng. Toán hạng này có thể là một hằng số học, một biến hoặc một biểu thức số học sử dụng các hằng số, các ký hiệu và các toán tử. Trình dịch hợp ngữ tính giá trị và thay thế dữ liệu tức thời vào trong lệnh. Lệnh này thường dùng để nạp 1 giá trị là 1 hằng số ở byte thứ 2 (hoặc byte thứ 3) vào thanh ghi hoặc ô nhớ .

Ví dụ:

```
MOV  A,  #12  ;Nạp giá trị 12(OCH) vào thanh ghi A
MOV A, #30H       ;nap du lieu 30H vao thanh ghi A
```

Tất cả các lệnh sử dụng kiểu định địa chỉ tức thời đều sử dụng hằng dữ liệu 8 bit làm dữ liệu tức thời. Có một ngoại lệ khi ta khởi động con trỏ dữ liệu 16-bit DPTR, hằng địa chỉ 16 bit được cần đến.

2.5. Định địa chỉ tương đối (hình 3.4).



Hình 3.4

Kiểu định địa chỉ tương đối chỉ sử dụng với những lệnh nhảy. Nơi nhảy đến có địa chỉ bằng địa chỉ đang lưu trong thanh ghi PC cộng với 1 giá trị 8 bit [còn gọi là giá trị lệch tương đối: relative offset] có giá trị từ – 128 đến +127 nên vi điều khiển có thể nhảy lùi [nếu số cộng với số âm] và nhảy tới [nếu số cộng với số dương]. Lệnh này có mã lệnh 2 byte, byte thứ 2 chính là giá trị lệch tương đối:

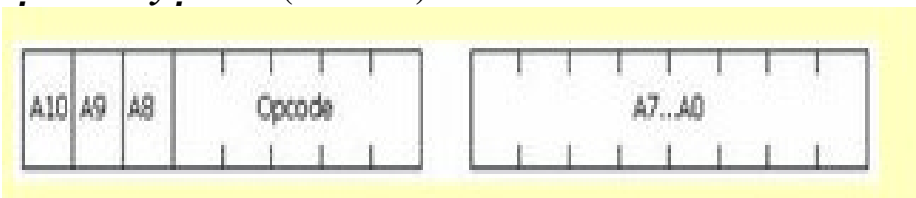
Nơi nhảy đến thường được xác định bởi nhãn (label) và trình biên dịch sẽ tính toán giá trị lệch.

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

Ví dụ:

```
SJMP   X1      ;nhay den nhan co tên X1 nằm trong
                ;tam vuc 256 byte
```

2.6. Định địa chỉ tuyệt đối (hình 3.5).



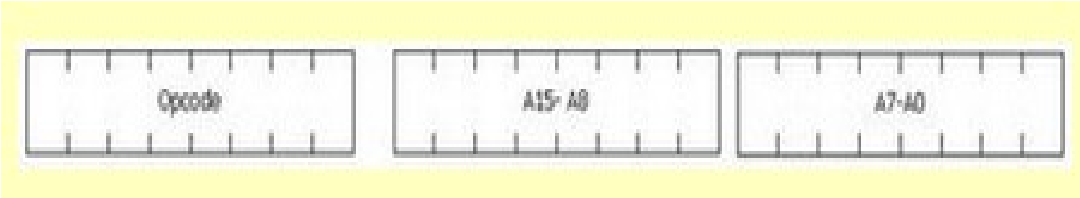
Hình 3.5

Kiểu định địa chỉ tuyệt đối (hình 3.5) được dùng với các lệnh ACALL và AJMP. Các lệnh này có mã lệnh 2 byte. Định địa chỉ tuyệt đối có ưu điểm là mã lệnh ngắn (2 byte), nhưng khuyết điểm là mã lệnh thay đổi và giới hạn phạm vi nơi nhảy đến, gọi đến không.

Ví dụ:

```
AJMP   X1;nhay den nhan co ten X1 nam trong tam vuc 2Kbyte
```

2.7. Định địa chỉ dài (Long Addressing) (hình 3.6)



Hình 3.6

Kiểu định địa chỉ dài được dùng với lệnh LCALL và LJMP. Các lệnh này có mã lệnh 3 byte – trong đó có 2 byte (16bit) là địa chỉ của nơi đến. Cấu trúc mã lệnh là 3 byte chứa địa chỉ đích 16 bit. Định địa chỉ dài là có thể gọi 1 chương trình con hoặc có thể nhảy đến bất kỳ vùng nhớ nào vùng nhớ 64KB. Lợi ích của kiểu định địa chỉ này là sử dụng hết toàn bộ không gian nhớ chương trình 64K, nhưng lại có điểm bất lợi là lệnh dài

đến 3-byte và phụ thuộc vào vị trí.

Ví dụ:

```
LJMP   X1   ;nhay den nhan co ten X1 nam trong
            ;tam vuc 64Kbyte
```

2.8. Định địa chỉ chỉ số (Index Addressing).

Chế độ định địa chỉ chỉ số được sử dụng rộng rãi khi truy cập các phần tử dữ liệu của bảng trong không gian ROM chương trình của 8051. Kiểu định địa chỉ chỉ số "dùng một thanh ghi cơ bản: là bộ đếm chương trình PC hoặc bộ đếm dữ liệu DPTR" kết hợp với "một giá trị lệch (offset) còn gọi là giá trị tương đối [thường lưu trong thanh ghi]" để tạo ra 1 địa chỉ của ô nhớ cần truy xuất hoặc là địa chỉ của nơi nhảy đến. Việc kết hợp được minh họa như sau:

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

Ví dụ:

```
MOVC A, @A + DPTR   ;lay du lieu trong o nho
                    ;DPTR+A de nap vao thanh ghi A
```

Ở lệnh này, nội dung của A được cộng với nội dung thanh ghi 16- bit DPTR để tạo ra địa chỉ 1.

3. Caác nhoám lệnh.

Mục tiêu : Hiểu được các nhóm lệnh sử dụng trong ngôn ngữ lập trình.

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

- Nhoám lệnh sô học
- Nhoám lệnh logic
- Nhoám lệnh vận truyên dữ liệu
- Nhoám lệnh Boolean (thao taác bit)
- Nhoám lệnh rẽ nhánh chương trình

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 nhãn LOOP). Có 3 toán hạng.

Các bạn cần nắm rõ phần cứng, đặc biệt là vùng nhớ Ram của vi điều khiển. Chú ý các thuật ngữ sau:

Các byte RAM 8 bit của vi điều khiển được gọi là "ô nhớ ", nếu các ô nhớ có chức năng đặc biệt thường được gọi là "thanh ghi", nếu là bit thì được gọi là

"bit nhớ". Dữ liệu của một ô nhớ là trạng thái (0 hoặc 1) cần thiết lập cho cać bit của ô nhớ (8 bit)

Ký hiệu	Mô tả
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).

Bảng 3.2: Các ký hiệu sử dụng mô tả lệnh (bảng 3.2)

3.1. Nhóm lệnh số học.

Cờ nhớ C:

C=1 nếu phép toán cộng xảy ra tràn hoặc phép trừ có mượn.

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

Phép cộng xảy ra tràn là phép cộng mà kết quả lớn hơn 255 (hay FFH hay 11111111b), lúc này C=1

Ví dụ:

- Phép cộng không tràn

Số cộng	38H	56	0 0 1 1 1 0 0 0 b
Số cộng	+3AH	58	0 0 1 1 1 0 1 0 b
Kết quả	72H	114	0 1 1 1 0 0 1 0 b
Cờ nhớ C	0		0

- Phép cộng tràn

Số cộng	6CH	108	0 1 1 0 1 1 0 0 b
Số cộng	+9FH	159	1 0 0 1 1 1 1 1 b
Kết quả	10BH	267	1 0 0 0 0 1 0 1 1 b
Cờ nhớ C	1		1

Phần bên traí là 8 bit của thanh ghi A sau khi kết quả được thực hiện, phần màu đỏ trong kết quả là giá trị bị tràn, giá trị này không lưu ở thanh ghi A mà lưu ở thanh ghi PSW, tại cờ C.

Số trừ	9FH	159	1 0 0 1 1 1 1 1 b

Số bị trừ	-6CH	108	0 1 1 0 1 1 0 0 b
Kết quả	33H	51	0 0 1 1 0 0 1 1 b
Cờ nhớ C	0		0
Số trừ	6CH	108	0 1 1 0 1 1 0 0 b
Số bị trừ	-9FH	159	1 0 0 1 1 1 1 1 b
Kết quả	CDH	-51	1 1 0 0 1 1 0 1 b
Cờ nhớ C	1		1

→ phép trừ trên có số mượn

3.1.1. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu trên thanh ghi Rn:

- Cú pháp: Add A,Rn
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên thanh ghi Rn, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.
- A (A) + (Rn)

Ví dụ:

```
Mov  A,#20H
Mov  R1,#08H
Add  A,R1
```

Kết quả : A có giá trị là 28H

R1 vẫn giữ nguyên giá trị là 08H

Cờ C = 0

Ví dụ 2:

```
Mov  A,#0E9H
Mov  R6,#0BAH
Add  A,R6
```

Kết quả : A = #0A3h

R6 = #0BAh

Cờ C = 1

3.1.2. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu ở ô nhớ có địa chỉ direct:

- Cú pháp: Add A,direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên ô nhớ có địa chỉ direct, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

- A (A) + (direct)

Ví dụ:

```
Mov   50h,#20H
Mov   A,#0E8H
Add   A,50H
```

Kết quả : A = #08H
50H = #20H
C = 1

3.1.3. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu của ô nhớ có địa chỉ gián tiếp:

- Cú pháp: Add A,@Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW. A (A) + (Ri)

Ví dụ:

```
AC = 1   ;cờ C đang mang giá trị 1
         Mov   50H,#60H
         Mov   R2,#50H
         Mov   A,#01H
         Add   A,@R2
```

Kết quả : A = #61H
R2 = #50H
C = 0 ;cờ C mang giá trị 0

3.1.4. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu xác định:

- Cú pháp: Add A,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với một giá trị xác định, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW. A (A) + # data

Ví dụ:

```
Mov   A,#05h
Add   A,#06h
```

Kết quả : A = #0Bh
C = 0

3.1.5. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu trên thanh ghi Rn có số nhớ ở cờ C:

- Cú pháp: AddC A,Rn
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên thanh ghi Rn và cộng thêm giá trị của số nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW. A (A) + (C) + (Rn)

Ví dụ: C = 1

```
Mov   A,#08h
Mov   R1,#10h
Addc  A,R1
```

Kết quả : A = #19h ;cộng cả cờ C

R1 = #10h

C =0

3.1.6. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu ở ô nhớ có địa chỉ direct và giá trị số nhớ ở cờ C.

- Cú pháp: AddC A,direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ direct và cộng thêm giá trị của số nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW. A (A) + (C) + (Rn).

Ví dụ: C = 0

```
Mov   A,#0A5h
Mov   10h,#96h
Addc  A,10h
```

Kết quả : A = #3Bh

10h = #96h

C =1

3.1.7. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu của ô nhớ có địa chỉ gián tiếp và số nhớ ở cờ C.

- Cú pháp: AddC A,@Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.

- Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri và cộng thêm giá trị của số nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

Ví dụ:

```
C = 1
Mov   A,#05h
Mov   50h,#10h
Mov   R2,#50h
Addc  a,@R2
```

Kết quả : A = #16h
C = 0

3.1.8. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu xác định và số nhớ ở cờ C.

- Cú pháp: AddC A,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị xác định và cộng thêm giá trị của số nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A (A) + (C) + #data

Ví dụ:

```
C = 1
Mov   A,#05h
Addc  A,#16h
```

Kết quả : A = #1Ch
C = 0

3.1.9. Lệnh trừ dữ liệu trên thanh ghi A với dữ liệu trên thanh ghi Rn và số nhớ ở cờ C.

- Cú pháp: SUBB A,Rn
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên thanh ghi Rn và trừ cho giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A (A) - (C) - (Rn)

Ví dụ: C = 1
Mov A,#0E5h
Mov R3,#9Fh
Subb A,R3
kết quả : A = 45h
C = 0

3.1.10. Lệnh trừ dữ liệu trên thanh ghi A với dữ liệu ở ô nhớ có địa chỉ direct và số nhớ ở cờ C.

- Cú pháp: SUBB A,direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ direct và trừ cho giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh trạng thái PSW. A (A) - (C) - (direct)

Ví dụ:
C = 0
Mov A,#0E5h
Mov 05h,#9Fh
Subb A,05h
Kết quả : A = 46h
C = 0

3.1.11. Lệnh trừ dữ liệu trên thanh ghi A với dữ liệu của ô nhớ có địa chỉ gián tiếp và số nhớ ở cờ C.

- Cú pháp: SUBB A,@Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri và trừ cho giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW. A (A) - (C) – ((Ri))

Ví dụ:
C = 1
Mov A,#0E5h
Mov 4Fh,#50h
Mov R3,#4Fh
Subb A,@R3
kết quả : A = 94h

C = 0

3.1.12. Lệnh trừ dữ liệu trên thanh ghi A với dữ liệu xác định và số nhớ ở cờ C.

- Cú pháp: SUBB A,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị xác định và trừ thêm giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A (A) - (C) - #data

Ví dụ:

C = 0
Mov A,#05h
Subb A,#4Fh

kết quả : A = 0B6h
C = 1

3.1.13.Lệnh tăng giá trị dữ liệu trên thanh ghi A lên 1 đơn vị.

- Cú pháp: INC A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Tăng giá trị dữ liệu lưu giữ trên thanh ghi A lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW. A (A) + 1

Ví dụ: Mov A,#05h
Inc A

Kết quả : A = #06h

3.1.14.Lệnh tăng giá trị dữ liệu trên thanh ghi Rn lên 1 đơn vị.

- Cú pháp: INC Rn
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Tăng giá trị dữ liệu lưu giữ trên thanh ghi Rn lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW. A (Rn) + 1

Ví dụ:

Mov R7,#0Fh
Inc R7

Kết quả : R7 = #10h

3.1.15.Lệnh tăng giá trị dữ liệu ở ô nhớ có địa chỉ direct lên 1 đơn vị.

Cú pháp: INC direct

- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.

- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Tăng giá trị dữ liệu ở một ô nhớ có địa chỉ direct lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   50h,#0FFh
Inc     50h
```

Kết quả : 50h = #00

3.1.16.Lệnh tăng giá trị dữ liệu ở ô nhớ có địa chỉ gián tiếp lên 1 đơn vị.

- Cú pháp: Inc @Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Tăng giá trị dữ liệu ở ô nhớ có địa chỉ bằng giá trị dữ liệu trên R_i lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   0Fh,#05h
Mov   R0,#0Fh
Inc     @R0
```

Kết quả : R0 = #06h

0Fh = #05h

3.1.17.Lệnh tăng giá trị của con trỏ dữ liệu DPTR lên 1 đơn vị.

- Cú pháp: Inc DPTR.
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Tăng giá trị dữ liệu của thanh ghi con trỏ dữ liệu DPTR lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   DPTR,#5Fh
Inc     DPTR
```

Kết quả : DPTR = #060h

3.1.18. Lệnh giảm giá trị dữ liệu trên thanh ghi A xuống 1 đơn vị.

- Cú pháp: DEC A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Giảm giá trị dữ liệu lưu giữ trên thanh ghi A xuống 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   A,#05h
Dec     A
```

Kết quả : A = #04h

3.1.19. Lệnh giảm giá trị dữ liệu trên thanh ghi Rn xuống 1 đơn vị:

- Cú pháp: Dec Rn
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Giảm giá trị dữ liệu lưu giữ trên thanh ghi Rn xuống 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   R6,#0Fh
Dec    R6
```

Kết quả : R6 = #0Eh

3.1.20.Lệnh giảm giá trị dữ liệu ở ô nhớ có địa chỉ direct xuống 1 đơn vị.

- Cú pháp: Dec direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Giảm giá trị dữ liệu ở ô nhớ có địa chỉ direct xuống 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   7Fh,#0
Dec    7Fh
```

Kết quả : 7Fh = #0FFh

3.1.21.Lệnh giảm giá trị dữ liệu ở ô nhớ có địa chỉ gián tiếp xuống 1 đơn vị.

- Cú pháp: Dec @Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Giảm giá trị dữ liệu ở ô nhớ có địa chỉ bằng giá trị dữ liệu trên Ri xuống1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   60h,#05h
Mov   R1,#60h
Dec    @R1
```

Kết quả : R1 = #04h

60h = #05h

3.1.22.Lệnh nhân thanh ghi A với thanh ghi B.

- Cú pháp: Mul AB
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 4 chu kì máy.

- Công dụng: Nhân hai dữ liệu là số nguyên không dấu ở thanh ghi A với thanh ghi B, kết quả là một dữ liệu 16 bit. Byte thấp của kết quả lưu ở thanh ghi A và byte cao của kết quả lưu ở thanh ghi B. Nếu tích số lớn hơn 255(0FFH), cờ tràn 0V ở thanh trạng thái PSW được thiết lập lên 1, ngược lại nếu tích số nhỏ hơn 255(0FFH), cờ tràn 0V được thiết lấp về 0. Cờ nhớ C luôn ở giá trị 0.

Ví dụ: Mov A,#0B9h
Mov B,#F7h
Mul AB

Kết quả : A = #7Fh , B = #0B2h

3.1.23.Lệnh chia thanh ghi A với thanh ghi B.

- Cú pháp: Div AB
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 4 chu kì máy.
- Công dụng: Chia hai dữ liệu là số nguyên không dấu ở thanh ghi A với thanh ghi B, dữ liệu ở thanh ghi A là số chia còn ở thanh ghi B là số bị chia, kết quả là một dữ liệu 8 bit được lưu ở thanh ghi A, số dư lưu trữ trong thanh ghi B Cờ nhớ C luôn ở giá trị 0.

Cờ tràn 0V được thiết lập giá trị 1 khi thanh ghi B mang giá trị là 00H-phép chia không thể thực hiện.

Ví dụ: Mov A,#50h
Mov B,#10h
DIV AB

Kết quả : A = #5h
B = #0h

3.1.24.Lệnh hiệu chỉnh thập phân nội dung của thanh ghi A đối với phép cộng.

- Cú pháp: DA A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 4 chu kì máy.
- Công dụng: hiệu chỉnh dữ liệu là giá trị lưu giữ ở thanh ghi A từ số Hex (số nhị phân) thành số BCD (số thập phân viết dưới dạng nhị phân). Lí do có lệnh hiệu chỉnh này vì khi cộng hai giá trị là số BCD bằng các lệnh cộng, vi điều khiển chỉ hiểu hai số cộng là số nhị phân bình thường, kết quả sau lệnh cộng là một số nhị phân bình thường, không phải là một số BCD, vì vậy kết quả cần được hiệu chỉnh để dữ liệu cuối là một số BCD. Khi thực hiện lệnh, cờ nhớ C được xác lập lên 1 nếu phép cộng có kết quả vượt qua 99 (số BCD). Kết quả cuối cùng, số BCD có hàng đơn vị nằm ở 4

bit thấp trên thanh ghi A, hàng chục ở 4 bit cao của thanh ghi A, hàng trăm là 1 nếu cờ C mang giá trị 1, là 0 nếu cờ C mang giá trị 0.

Ví dụ 1: Mov A,#10h
DA A
Kết quả : A = #10h

Ví dụ 2: Mov A,#0Eh
DA A
Kết quả : A = #14h

3.2. Nhóm lệnh logic.

3.2.1. Lệnh And dữ liệu ở thanh ghi A với dữ liệu ở thanh ghi Rn.

Cú pháp: ANL A,Rn

- Lệnh này chiếm dung lượng bộ nhớ ROM là: 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic AND dữ liệu ở thanh ghi A với dữ liệu ở thanh ghi Rn, kết quả được lưu trữ ở thanh ghi A.

Ví dụ:

```
MOV A,#0Fh
MOV R1,#0F0h
ANL A,R1
```

Kết quả : A = #0H

3.2.2. Lệnh And dữ liệu trên thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct:

- Cú pháp: ANL A,direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic AND dữ liệu ở thanh ghi A với dữ liệu ở ô nhớ có địa chỉ direct, kết quả được lưu trữ ở thanh ghi A.

Ví dụ:

```
MOV A,#0FFh
MOV 10h,#010h
ANL A,10h
```

Kết quả : A = #010h

3.2.3. Lệnh And dữ liệu trên thanh ghi A với dữ liệu của ô nhớ gián tiếp:

- Cú pháp: ANL A,@Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic AND dữ liệu ở thanh ghi A với

dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri, kết quả được lưu trữ ở thanh ghi A.

Ví dụ:

```
mov   A,#0Fh
mov   70h,#0E1h
mov   R1,#070h
ANL   A,@R1
```

Kết quả : A = #01h

3.2.4. Lệnh And dữ liệu trên thanh ghi A với dữ liệu xác định.

- Cú pháp: ANL A,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic AND dữ liệu ở thanh ghi A với dữ liệu cho trước, kết quả được lưu trữ ở thanh ghi A.

Ví dụ:

```
MOV   A,#0Eh
ANL   A,#11h
```

Kết quả : A = #00

3.2.5. Lệnh And dữ liệu của ô nhớ có địa chỉ direct với dữ liệu trên thanh ghi A.

- Cú pháp: ANL direct,A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic AND dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct, kết quả được lưu trữ ở ô nhớ có địa chỉ direct.

Ví dụ:

```
MOV   A,#08h
MOV   R1,#0F7h
ANL   R1,A
```

Kết quả : R1 = #0

3.2.6. Lệnh And dữ liệu trên ô nhớ có địa chỉ direct với dữ liệu xác định.

- Cú pháp: ANL direct,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic AND dữ liệu của ô nhớ có địa chỉ direct với dữ liệu cho trước, kết quả được lưu trữ ở ô nhớ có địa chỉ

direct.

Ví dụ:

```
MOV     R1,#0F7h
ANL     R1,#1Fh
```

Kết quả : R1 = #017h

3.2.7. Lệnh OR dữ liệu ở thanh ghi A với dữ liệu ở thanh ghi Rn:

- Cú pháp: ORL A,Rn
- Lệnh này chiếm dung lượng bộ nhớ ROM là: 1 Byte
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic OR dữ liệu ở thanh ghi A với dữ liệu ở thanh ghi Rn, kết quả được lưu trữ ở thanh ghi A.

Ví dụ:

```
MOV     A,#0Fh
MOV     R1,#0F0h
ORL     A,R1
```

Kết quả : A = #0FFh

3.2.8. Lệnh OR dữ liệu trên thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct:

- Cú pháp: ORL A,direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic OR dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct, kết quả được lưu trữ ở thanh ghi A.

Ví dụ:

```
MOV     A,#0Eh
MOV     50h,#0F0h
ORL     A,50h
```

Kết quả : A = #0FEh

3.2.9. Lệnh OR dữ liệu trên thanh ghi A với dữ liệu của ô nhớ gián tiếp.

- Cú pháp: ORL A,@Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic OR dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri, kết quả được lưu trữ ở thanh ghi A.

Ví dụ:

```
MOV     A,#18h
MOV     30h,#0F0h
```

```
MOV     R1,#30h
ORL     A,@R1
```

Kết quả : A = #0F8h

3.2.10. Lệnh OR dữ liệu trên thanh ghi A với dữ liệu xác định:

- Cú pháp: ORL A,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic OR dữ liệu ở thanh ghi A với dữ liệu cho trước, kết quả được lưu trữ ở thanh ghi A.

Ví dụ:

```
MOV     A,#00h
ORL     A,#10h
```

Kết quả : A = #010h

3.2.11. Lệnh OR dữ liệu của ô nhớ có địa chỉ direct với dữ liệu trên thanh ghi A:

- Cú pháp: ORL direct,A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic OR dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct, kết quả được lưu trữ ở ô nhớ có địa chỉ direct.

Ví dụ:

```
MOV     A,#0Fh
MOV     5Fh,#0F0h
ORL     5Fh,A
```

Kết quả : 5Fh = #0FFh

3.2.12. Lệnh OR dữ liệu trên ô nhớ có địa chỉ direct với dữ liệu xác định:

- Cú pháp: ORL direct,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Thực hiện phép logic OR dữ liệu của ô nhớ có địa chỉ direct với dữ liệu cho trước, kết quả được lưu trữ ở ô nhớ có địa chỉ direct.

Ví dụ:

```
MOV     60h,#0F0h
ORL     60h,#1Fh
```

Kết quả : 60h = #0FFh

3.2.13. Lệnh EX-OR dữ liệu ở thanh ghi A với dữ liệu ở thanh ghi Rn:

- Cú pháp: XRL A,Rn
- Lệnh này chiếm dung lượng bộ nhớ ROM là: 1 Byte
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic EX-OR dữ liệu ở thanh ghi A với dữ liệu ở thanh ghi Rn, kết quả được lưu trữ ở thanh ghi A.

Ví dụ: MOV A,#0F2h
MOV R3,#0E0h
XRL A,R3 Kết quả : A = #12h

3.2.14. Lệnh EX-OR dữ liệu trên thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct:

- Cú pháp: XRL A,direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic EX-OR dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct, kết quả được lưu trữ ở thanh ghi A.

Ví dụ:

MOV A,#012h
MOV 10h,#0E0h
XRL A,10h

Kết quả : A = #0F2h

3.2.15. Lệnh EX-OR dữ liệu trên thanh ghi A với dữ liệu của ô nhớ gián tiếp.

- Cú pháp: XRL A,@Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.

Công dụng: Thực hiện phép logic EX-OR dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri, kết quả được lưu trữ ở thanh ghi A.

Ví dụ:

MOV A,#08h
MOV 10h,#0E9h
MOV R0,#10h
XRL A,@R0

Kết quả : A = #0E1h

3.2.16. Lệnh EX-OR dữ liệu trên thanh ghi A với dữ liệu xác định:

- Cú pháp: XRL A,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.

- Công dụng: Thực hiện phép logic EX-OR dữ liệu ở thanh ghi A với dữ liệu cho trước, kết quả được lưu trữ ở thanh ghi A.

Ví dụ: MOV A,#12h
XRL A,#12h

Kết quả : A = #0

3.2.17. Lệnh EX-OR dữ liệu của ô nhớ có địa chỉ direct với dữ liệu trên thanh ghi A.

- Cú pháp: XRL direct,A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép logic EX-OR dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct, kết quả được lưu trữ ở ô nhớ có địa chỉ direct.

Ví dụ: MOV A,#0F2h
MOV 50h,#0E0h
XRL 50h,A

Kết quả : 50h = #12h

3.2.18. Lệnh EX-OR dữ liệu trên ô nhớ có địa chỉ direct với dữ liệu xác định.

- Cú pháp: XRL direct,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Thực hiện phép logic EX-OR dữ liệu của ô nhớ có địa chỉ direct với dữ liệu cho trước, kết quả được lưu trữ ở ô nhớ có địa chỉ direct.

Ví dụ:

Mov 50h,#0E0h
XRL 50h,#01h

Kết quả : 50h = #0E1h

3.2.19. Lệnh bù giá trị dữ liệu trên thanh ghi A.

- Cú pháp: CPL A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: lấy bù giá trị lưu giữ ở thanh ghi A, các bit có giá trị là 1 chuyển thành 0 và ngược lại các bit có giá trị là 0 chuyển thành 1.

Ví dụ: mov A,#01100111b ;(tương đương 67h)
CPL A

Kết quả : A = #10011000b (tương đương 98h)

3.2.20. Lệnh xóa dữ liệu trên thanh ghi A.

- Cú pháp: CLR A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Tất cả các bit của thanh ghi A đều được xác lập giá trị 0.

Ví dụ: MOV A,#01100111b
CLR A

Kết quả : A = #0

3.2.21. Lệnh xoay trái dữ liệu trên thanh ghi A:

- Cú pháp: RL A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thanh ghi A gồm tám bit A7 A6 A5 A4 A3 A2 A1 A0. Khi thực hiện lệnh xoay trái *RL A* giá trị của các bit được chuyển trang bit ở bên trái nó, giá trị của bit A0 chuyển sang bit A1, giá trị của bit A1 chuyển sang bit A2, tương tự với các bit còn lại, và giá trị của bit A7 chuyển sang bit A0. Minh họa các bit trong thanh ghi A khi thực hiện lệnh như trong hình dưới.

Các bit ở thanh ghi A —
Quá trình xoay dữ liệu từ A0 đến A6 —
Giá trị dữ liệu A7 chuyển sang bit A0 —

A7 A6 A5 A4 A3 A2 A1 A0
<— <— <— <— <— <— <—
A7 ——————————————————————>A0

Ví dụ: MOV A,#01001001b
RL A

Kết quả sau khi các lệnh được thực hiện A mang giá trị là 10010010b

Giá trị thanh ghi A
Trước khi thực hiện lệnh xoay trái 0 1 0 0 1 0 0 1
Sau khi thực hiện lệnh xoay trái 1 0 0 1 0 0 1 0

3.2.22. Lệnh xoay trái dữ liệu trên thanh ghi A cùng với cờ nhớ C:

- Cú pháp: RLC A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: thanh ghi A gồm tám bit A7 A6 A5 A4 A3 A2 A1 A0. Khi thực hiện lệnh xoay trái A với cờ nhớ RLC A giá trị của các bit được chuyển trang bit ở bên trái nó, giá trị của bit A0 chuyển sang bit A1, giá trị

của bit A1 chuyển sang bit A2, tương tự với các bit còn lại, và giá trị của bit A7 chuyển sang cờ nhớ C, giá trị ở cờ nhớ C chuyển sang bit A0

Các bit ở thanh ghi A cùng với cờ C —

Quá trình xoay dữ liệu từ A0 đến A6 —

Giá trị ở C chuyển sang bit A0 —

C A7 A6 A5 A4 A3 A2 A1 A0

<— <— <— <— <— <— <— <—

C ——————————————————————>A0

Ví dụ: giả sử cờ nhớ C đang mang giá trị 1

```
Mov   A,#11001001b
RLC   A
```

Kết quả sau khi các lệnh được thực hiện A mang giá trị là 10010011b và C mang giá trị 1.

	Cờ nhớ C	Giá trị thanh A
Trước khi thực hiện lệnh xoay trái với C	1	1 1 0 0 1 0 0 1
Sau khi thực hiện lệnh xoay trái với C	1	1 0 0 1 0 0 1 1

3.2.23. Lệnh xoay phải dữ liệu trên thanh ghi A:

- Cú pháp: RR A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thanh ghi A gồm tám bit A7 A6 A5 A4 A3 A2 A1 A0.

Khi thực hiện lệnh xoay phải *RR A* giá trị của các bit được chuyển trang bit ở bên phải nó, giá trị của bit A7 chuyển sang bit A6, giá trị của bit A6 chuyển sang bit A5, tương tự với các bit còn lại, và giá trị của bit A0 chuyển sang bit A7. Minh họa các bit trong thanh ghi A khi thực hiện lệnh như trong hình dưới.

Các bit ở thanh ghi A —

Quá trình xoay dữ liệu từ A7 đến A1 —

Giá trị dữ liệu A0 chuyển sang bit A7 —

A7 A6 A5 A4 A3 A2 A1 A0

—> —> —> —> —> —> —> —>

A7 <——————————————————————A0

Ví dụ:

```
Mov   A,#01001001b
RL    A
```

Kết quả sau khi các lệnh được thực hiện A mang giá trị là 10100100b

	Giá trị thanh A
Trước khi thực hiện lệnh xoay phải	0 1 0 0 1 0 0 1

Sau khi thực hiện lệnh xoay phải 1 0 1 0 0 1 0 0

3.2.24. Lệnh xoay phải dữ liệu trên thanh ghi A cùng với cờ nhớ C:

- Cú pháp: RRC A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: thanh ghi A gồm tám bit A7 A6 A5 A4 A3 A2 A1 A0. Khi thực hiện lệnh xoay phải A với cờ nhớ -RRC A -giá trị của các bit được chuyển trang bit ở bên phải nó, giá trị của bit A7 chuyển sang bit A6, giá trị của bit A6 chuyển sang bit A5, tương tự với các bit cònlại, và giá trị của bit A0 chuyển sang cờ nhớ C, giá trị ở cờ nhớ C chuyển sang bit A7.

Các bit ở thanh ghi A cùng với cờ C —

Quá trình xoay dữ liệu từ C đến A1 —

Giá trị ở A0 chuyển sang bit C —

C A7 A6 A5 A4 A3 A2 A1 A0

—> —> —> —> —> —> —> —>

C <——————————————————————A0

Ví dụ: giả sử cờ nhớ C đang mang giá trị 1

```
Mov A,#11001001b
RLC A
```

Kết quả sau khi các lệnh được thực hiện A mang giá trị là 11100100b và C mang giá trị 1.

Cờ nhớ C Giá trị thanh A

Trước khi thực hiện lệnh xoay trái với C 1 1 1 0 0 1 0 0 1

Sau khi thực hiện lệnh xoay trái với C 1 1 1 1 0 0 1 0 0

3.2.25. Lệnh xoay 4 bit trên thanh ghi A.

- Cú pháp: SWAP A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: hoán chuyển dữ liệu ở 4 bit thấp lên 4 bit cao và 4 bit cao xuống 4 bit thấp

Các bit ở thanh ghi A —

Dữ liệu trước khi thực hiên lệnh —

Dữ liệu sau khi thực hiên lệnh —

A7 A6 A5 A4 A3 A2 A1 A0

X7 X6 X5 X4 X3 X2 X1 X0

X3 X2 X1 X0 X7 X6 X5 X4

Ví dụ:

```
mov A,#0E7h
```

SWAP A

Kết quả : A = # 7Eh

3.3. Nhóm lệnh truyền dữ liệu.

3.3.1. Lệnh chuyển dữ liệu từ một thanh ghi Rn vào thanh ghi A

- Cú pháp: MOV A,Rn
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Chuyển dữ liệu của thanh ghi Rn vào thanh ghi A, dữ liệu trên thanh ghi Rn không đổi.

Ví dụ: Giả sử thanh ghi R5 mang dữ liệu với giá trị là 0A5H (10100101B)

Lệnh MOV A,R5

Sau khi lệnh được thực hiện A mang dữ liệu giá trị A5H, Rn không đổi giá trị thanh ghi A trước khi thực hiện lệnh không cần quan tâm

3.3.2. Lệnh chuyển dữ liệu từ ô nhớ có địa chỉ direct vào thanh ghi A

- Cú pháp: MOV A,direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: chuyển dữ liệu của ô nhớ có địa chỉ bằng direct vào thanh ghi A.

Ví dụ: Giả sử thanh ghi có địa chỉ 33H mang dữ liệu với giá trị là 09H (00001001B)

Lệnh Mov A,33H

Sau khi lệnh được thực hiện A mang dữ liệu giá trị 09H

3.3.3. Lệnh chuyển dữ liệu từ ô nhớ có địa chỉ gián tiếp vào thanh ghi A:

- Cú pháp: MOV A,@Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: chuyển dữ liệu của ô nhớ 'có địa chỉ bằng giá trị của thanh ghi Ri' vào thanh ghi A.

Ví dụ: Giả sử trước khi thực hiện lệnh ô nhớ có địa chỉ 33H mang dữ liệu với giá trị là 09H (00001001B) và thanh ghi R1 được thiết lập giá trị là 33H

Lệnh Mov A,@R1

Khi lệnh được thực hiện A nhận dữ liệu từ ô nhớ có vị trí bằng giá trị được thiết lập trong thanh ghi R1, tức là A nhận dữ liệu từ ô nhớ có địa chỉ là 33H, chú ý: trước đó ô nhớ 33H mang dữ liệu là 09H.

Sau khi lệnh được thực hiện A mang giá trị là 09H (00001001B)

3.3.4. Lệnh đưa dữ liệu vào thanh ghi A

- Cú pháp: MOV A,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: thiết lập dữ liệu cho thanh ghi A.

Ví dụ: Muốn thanh ghi A mang dữ liệu có giá trị là 56H ta thực hiện lệnh

```
MOV A,#56H
```

Sau khi lệnh được thực hiện A mang giá trị là 56H

3.3.5. Lệnh chuyển dữ liệu từ A vào thanh ghi Rn

- Cú pháp: MOV Rn,A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: chuyển dữ liệu từ thanh ghi A vào thanh ghi Rn (n=0-7)

Ví dụ:

```
MOV A,#56H
MOV R1,A
```

Sau khi các lệnh được thực hiện R1 mang giá trị là 56H

3.3.6. Lệnh chuyển dữ liệu từ một ô nhớ có địa chỉ direct vào thanh ghi Rn.

- Cú pháp: MOV Rn,direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: chuyển dữ liệu của ô nhớ có địa chỉ direct vào thanh ghi Rn (n=0-7)

Ví dụ: giả sử ô nhớ 55H mang dữ liệu có giá trị là A3H

```
MOV R4,55H
```

Sau khi các lệnh được thực hiện R4 mang giá trị là A3H

3.3.7. Thiết lập dữ liệu cho thanh ghi Rn

- Cú pháp: MOV Rn,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: thiết đặt dữ liệu cho thanh ghi Rn

Ví dụ:

Muốn thanh ghi Rn mang dữ liệu có giá trị là 37H ta thực hiện lệnh.

```
MOV A,#37H
```

Sau khi lệnh được thực hiện A mang giá trị là 37H

3.3.8. Lệnh chuyển dữ liệu từ thanh ghi A vào một ô nhớ có địa chỉ direct

- Cú pháp: MOV direct,A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: chuyển dữ liệu từ thanh ghi A vào một ô nhớ có địa chỉ direct.

Ví dụ: MOV A,#77H
MOV 69H,A

Sau khi các lệnh được thực hiện ô nhớ 69H mang giá trị là 77H (giá trị của các bit được thiết lập trong ô nhớ 69H là 01110111B)

3.3.9. Lệnh chuyển dữ liệu từ thanh ghi Rn vào một ô nhớ có địa chỉ direct

- Cú pháp: MOV direct,Rn
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: chuyển dữ liệu từ thanh ghi A vào một ô nhớ có địa chỉ direct.

Ví dụ: MOV Rn,#78H
MOV 7AH,Rn

Sau khi các lệnh được thực hiện ô nhớ 7AH mang giá trị là 78H

3.3.10. Lệnh chuyển dữ liệu từ một ô nhớ có địa chỉ direct này vào một ô nhớ có địa chỉ direct khác

- Cú pháp: MOV direct,direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: chuyển dữ liệu từ ô nhớ có địa chỉ direct này vào một ô nhớ có địa chỉ direct khác.

Ví dụ: Giả sử thanh ghi 20H mang dữ liệu có giá trị là FFH
MOV 22H,20H

Sau khi lệnh được thực hiện thanh ghi 22H mang giá trị là FFH

3.3.11. Lệnh đưa dữ liệu vào ô nhớ có địa chỉ direct

- Cú pháp: MOV direct,#data
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: thiết lập dữ liệu cho ô nhớ có địa chỉ direct

Ví dụ:

MOV 52H,#43H

Sau khi các lệnh được thực hiện ô nhớ 52H mang giá trị là 43H

3.3.12. Lệnh chuyển dữ liệu từ một ô nhớ có địa chỉ gián tiếp vào ô nhớ có địa chỉ direct

- Cú pháp: MOV direct,@Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Chuyển dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri vào ô nhớ có địa chỉ direct.

Ví dụ:

```
MOV 30H,#46H
MOV R0,#30H
MOV 23H, @R0
```

Sau khi các lệnh được thực hiện ô nhớ 23H mang giá trị là 46H

3.3.13. Lệnh chuyển dữ liệu từ thanh ghi A vào ô nhớ có địa chỉ gián tiếp.

- Cú pháp: Mov @Ri,A
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Chuyển dữ liệu của thanh ghi A vào ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri

Ví dụ:

```
MOV A,#33H
MOV R1,#22H
MOV @R0,A
```

Sau khi lệnh được thực hiện ô nhớ 22H mang giá trị là 33H

3.3.14. Lệnh chuyển dữ liệu từ một ô nhớ có địa chỉ direct vào ô nhớ có địa chỉ gián tiếp

- Cú pháp: MOV @Ri,direct
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Chuyển dữ liệu của ô nhớ có địa chỉ direct vào ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri.

Ví dụ:

```
MOV 4BH,#2AH
MOV R0,#2AH
MOV @R0,4BH
```

Sau khi lệnh được thực hiện ô nhớ 2AH mang giá trị là 2AH

3.3.15. Lệnh đưa dữ liệu vào ô nhớ có địa chỉ gián tiếp

- Cú pháp: MOV @Ri,#data.
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.

- Công dụng: Thiết đặt dữ liệu cho ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri.

Ví dụ:

```
MOV  R0,#3BH
MOV  @R0,#27H
```

Sau khi lệnh được thực hiện ô nhớ 3BH mang giá trị là 27H

3.3.16. Lệnh đưa dữ liệu vào con trỏ dữ liệu DPTR

- Cú pháp: MOV DPTR,#data16.
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.

- Công dụng: Thiết đặt dữ liệu cho con trỏ dữ liệu với dữ liệu 16 bit, thực chất dữ liệu được lưu ở hai thanh ghi DPL (byte thấp-địa chỉ byte 82H) và DPH (byte cao-địa chỉ byte 83H).

Ví dụ: MOV DPTR,#3A5FH

Sau khi lệnh được thực hiện DPTR mang giá trị là 3A5FH

DPL mang giá trị 5FH và DPL mang giá trị 3AH.

3.3.17. Lệnh trao đổi dữ liệu giữa ô nhớ có địa chỉ direct với thanh ghi A.

- Cú pháp: XCH A,direct.
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Trao đổi dữ liệu của thanh ghi A với ô nhớ có địa chỉ direct, tức là sau khi thực hiện lệnh ô nhớ có địa chỉ direct mang dữ liệu của thanh ghi A trước đó và thanh ghi A mang dữ liệu của ô nhớ có địa chỉ direct.

Ví dụ:

```
MOV  A,#0FAH
MOV  50H,#60H
XCH A,50H
```

Kết quả :A mang giá trị là 60H

50H mang giá trị là 0FAH

3.3.18. Lệnh trao đổi dữ liệu giữa thanh ghi Rn và thanh ghi A

- Cú pháp: XCH A,Rn.
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Trao đổi dữ liệu của thanh ghi A với thanh ghi Rn.

3.3.19. Lệnh trao đổi dữ liệu giữa thanh ghi có địa chỉ gián tiếp và thanh ghi A.

- Cú pháp: XCH A,@Ri.
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Trao đổi dữ liệu của thanh ghi A với ô nhớ có địa chỉ bằng giá trị lưu giữ trong thanh ghi Ri.

3.3.20. Lệnh trao đổi dữ liệu 4 bit giữa thanh ghi có địa chỉ gián tiếp và thanh ghi A.

- Cú pháp: XCHD A,@Ri
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte
- Thời gian thực hiện: 1 chu kì máy
- Công dụng: Trao đổi dữ liệu của 4 bit thấp ở thanh ghi A với dữ liệu của 4 bit thấp ở ô nhớ có địa chỉ bằng giá trị lưu giữ trong thanh ghi Ri.

3.3.21. Lệnh truy xuất dữ liệu từ ROM nội

- Cú pháp: MovC A,@A+DPTR
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte
- Thời gian thực hiện: 2 chu kì máy
- Công dụng: Chuyển dữ liệu từ bộ nhớ ROM có địa chỉ bằng giá trị của A cộng với DPTR vào thanh ghi A.

Các lệnh còn lại trong nhóm lệnh di chuyển

```
MOVC    A,@A+PC
MOVC    A,@i
MOVX    A,@DPTR
MOVX    A,@Ri
MOVX    @DPTR,A
PUSH    direct
POP     direct
```

3.4. Nhóm lệnh boolean.

Quy ước: trong câu lệnh "bit" đại diện cho một địa chỉ của bit nhớ

3.4.1. Lệnh xóa cờ nhớ C

- Cú pháp: CLR C
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.

- Công dụng: Xóa cờ nhớ C - tức là đưa giá trị của cờ nhớ C về 0

3.4.2. Lệnh xóa bit

- Cú pháp: CLR bit
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Xóa giá trị của bit nhớ có địa chỉ xác định - tức là đưa giá trị bit đó về 0.

3.4.3. Lệnh thiết đặt cờ nhớ C

- Cú pháp: SetB C
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: thiết đặt cờ nhớ C - tức là đưa giá trị của cờ nhớ C lên 1
- kit kit Cú pháp: SetB bit
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thiết đặt giá trị bit nhớ có địa chỉ xác định - tức là đưa giá trị bit đó lên 1.

3.4.5. Lệnh bù cờ nhớ C

- Cú pháp: CPL C
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: đổi giá trị của cờ nhớ C, nếu trước đó C có giá trị 0 chuyển thành 1, và ngược lại nếu trước đó C có giá trị 1 chuyển thành 0.

3.4.6. Lệnh bù bit

- Cú pháp: CPL bit
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: đổi giá trị của bit có địa chỉ xác định, nếu trước đó bit đó có giá trị 0 chuyển thành 1, và ngược lại nếu trước đó bit đó có giá trị 1 chuyển thành 0.

3.4.7. Lệnh And cờ nhớ C với bit

- Cú pháp: ANL C,bit
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép And cờ nhớ C và bit có địa chỉ xác định, kết quả lưu ở C.

3.4.8. Lệnh And cờ nhớ C với bit đã được lấy bù

- Cú pháp: ANL C,/bit
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện phép And cờ nhớ C và bit có địa chỉ xác định đã được lấy bù, kết quả lưu ở C.

3.4.9. Lệnh OR cờ nhớ C với bit

- Cú pháp: ORL C,bit
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Thực hiện phép And cờ nhớ C và bit có địa chỉ xác định, kết quả lưu ở C.

3.4.10. Lệnh OR cờ nhớ C với bit đã được lấy bù

- Cú pháp: ORL C,/bit
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Thực hiện phép And cờ nhớ C và bit có địa chỉ xác định đã được lấy bù, kết quả lưu ở C.

3.4.11. Lệnh chuyển giá trị bit có địa chỉ xác định vào cờ nhớ C.

- Cú pháp: MOV C,bit
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: Thực hiện chuyển giá trị của bit có địa chỉ xác định vào cờ nhớ C.

3.4.12. Lệnh chuyển giá trị cờ nhớ C vào bit có địa chỉ xác định.

- Cú pháp: MOV bit,C
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Thực hiện chuyển giá trị của cờ nhớ C vào bit có địa chỉ xác định.

3.5.Nhóm lệnh rẽ nhánh chương trình.

Phần này liên quan đến các câu lệnh được lưu giữ trên ROM, vì vậy cần xem lại phần bộ nhớ ROM trước khi xem phần này.

Phần phụ chú:

Nhãn: Kí hiệu: rel

Nhãn là một chuỗi kí tự do người dùng tự đặt dùng để đánh dấu các đoạn chương trình, nhãn này biểu thị địa chỉ của lệnh khi được lưu trên ROM.

Nhãn chỉ được bắt đầu bằng một kí tự chữ hoặc dấu "_",không được bắt đầu bằng số,không có khoảng trắng và kết thúc bằng dấu hai chấm ":"

Trong chương trình nhãn không được đặt trùng tên với nhau, và không được trùng với các từ khóa mà chương trình đã sử dụng.

Ví dụ: Các nhãn đúng X1: ;S_2: ;_5:s10: ;...

Các nhãn sai 1X: ; S_2 ;S 5: ;DW: ,LPT :...

Chương trình con: là những đoạn chương trình thực hiện một số lệnh nào đó và được viết ngoài chương trình chính, các chương trình con này được đặt tên bằng một nhãn và kết thúc bằng lệnh RET, chương trình con có thể gọi một chương trình con khác. Chương trình con được chương trình chính sử dụng khi cần thiết bằng các lệnh gọi chương trình con; khi có lệnh gọi chương trình con, Vi điều khiển chuyển về thực hiện các đoạn chương trình của chương trình con, sau khi thực hiện chương trình con Vi điều khiển tiếp tục trở về thực hiện các câu lệnh trong chương trình chính.

Chương trình con giúp cho chương trình mạch lạc, dễ hiểu hơn, nếu trong chương trình chính có các đoạn chương trình được lặp đi lặp lại nhiều lần thì các đoạn chương trình đó thường được viết thành một chương trình con và truy xuất bằng một câu lệnh gọi chương trình con. Việc sử dụng chương trình con giúp cho việc tìm lỗi và chỉnh sửa chương trình dễ hơn, nếu chương trình chính sử dụng nhiều lần chương trình con, khi cần sửa đổi chỉ cần thay đổi các câu lệnh trong chương trình con.

Chương trình con bắt đầu bằng một nhãn và kết thúc bằng lệnh Reti, chương trình con có thể đặt ở đầu hoặc cuối chương trình.

3.5.1. Lệnh gọi chương trình con dùng địa chỉ tuyệt đối

- Cú pháp: ACall addr11
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Khi lệnh được thực hiện, Vi điều khiển chuyển về thực hiện các câu lệnh của chương trình con bắt đầu từ địa chỉ addr11 trên Rom, địa chỉ addr11 có thể thay bằng nhãn bắt đầu của một chương trình con. Câu lệnh được thực hiện khi địa chỉ addr11 cách lệnh gọi không quá 2 KByte.

Ví dụ: ACall 45A6H

3.5.2. Lệnh gọi chương trình con dùng địa chỉ tuyệt đối.

- Cú pháp: ACall addr16
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.

Công dụng: Khi lệnh được thực hiện, Vi điều khiển chuyển về thực hiện các câu lệnh của chương trình con bắt đầu từ địa chỉ addr16 trên Rom, địa chỉ addr16 có thay bằng nhãn bắt đầu chương trình con. Câu lệnh có thể gọi chương trình con ở bất kì vị trí nào trên Rom vì khoảng cách từ lệnh gọi đến chương trình con là 64 KByte.

3.5.3. Lệnh kết thúc chương trình con.

- Cú pháp: Ret
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 2 chu kì máy
- Công dụng: Lệnh này dùng kết thúc chương trình con, khi gặp lệnh này Vi điều khiển quay về thực hiện lệnh ở chương trình chính.

3.5.4. Lệnh kết thúc chương trình con phục vụ ngắt

- Cú pháp: Reti
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Lệnh này dùng kết thúc chương trình con ngắt, khi gặp lệnh này Vi điều khiển quay về thực hiện lệnh ở chương trình chính.

3.5.5. Lệnh nhảy ngắn đến địa chỉ tuyệt đối

- Cú pháp: AJMP addr11
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.

Công dụng: Khi lệnh được thực hiện, Vi điều khiển chuyển về thực hiện các câu lệnh của chương trình bắt đầu tại địa chỉ addr11 trên Rom, địa chỉ addr11 có thể thay bằng nhãn. Câu lệnh chỉ được thực hiện khi vị trí lưu chương trình cần thực hiện cách lệnh gọi không quá 2 Kbyte.

3.5.6. Lệnh nhảy dài đến địa chỉ tuyệt đối

- Cú pháp: LJMP addr16
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.

Công dụng: Khi lệnh được thực hiện, Vi điều khiển chuyển về thực hiện các câu lệnh của chương trình bắt đầu tại địa chỉ addr11 trên Rom, địa chỉ addr11 có thể thay bằng nhãn. Câu lệnh có thể gọi chương trình ở bất kì vị trí nào trên Rom vì khoảng cách từ lệnh gọi đến chương trình con là 64 Kbyte.

3.5.7. Lệnh nhảy tương đối

- Cú pháp: SJMP rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.

Công dụng: Khi lệnh được thực hiện, Vi điều khiển chuyển đến thực hiện các câu lệnh của chương trình được đánh dấu bằng nhãn. Câu lệnh chỉ được thực hiện địa chỉ của nhãn cách lệnh gọi không quá 128 Byte(cả tới hoặc lùi).

3.5.8. Lệnh nhảy gián tiếp

- Cú pháp: JMP @A+DPTR
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng: Khi lệnh được thực hiện, Vi điều khiển chuyển đến thực hiện các câu lệnh của chương trình có địa chỉ trên ROM bằng giá trị của A cộng với giá trị lưu giữ trên DPTR

3.5.9. Lệnh nhảy thuận với cờ Zero

- Cú pháp: JZ rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

+ Nếu cờ Zero có giá trị 1(tức thanh ghi A có giá trị 0), Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt.

+ Nếu cờ Zero có giá trị 0(tức thanh ghi A có giá trị khác 0), Vi điều khiển thực hiện lệnh kế tiếp (không thực hiện lệnh nhảy).

3.5.10. Lệnh nhảy nghịch với cờ Zero

- Cú pháp: JNZ rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

+ Nếu cờ Zero có giá trị 0(tức thanh ghi A có giá trị khác 0), Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt.

+ Nếu cờ Zero có giá trị 1(tức thanh ghi A có giá trị 0), Vi điều khiển thực hiện lệnh kế tiếp (không thực hiện lệnh nhảy).

3.5.11. Lệnh nhảy thuận với cờ C

- Cú pháp: JC rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

+ Nếu cờ C có giá trị 1, Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt.

+ Nếu cờ C có giá trị 0, Vi điều khiển thực hiện lệnh kế tiếp (không thực hiện lệnh nhảy)

3.5.12. Lệnh nhảy nghịch với cờ Zero

- Cú pháp: JNC rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

+ Nếu cờ C có giá trị 0, Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt.

+ Nếu cờ C có giá trị 1, Vi điều khiển thực hiện lệnh kế tiếp (không thực hiện lệnh nhảy).

3.5.13. Lệnh nhảy thuận với giá trị của bit nhớ

- Cú pháp: JB bit,rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

+ Nếu bit nhớ có giá trị 1, Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt.

+ Nếu bit nhớ có giá trị 0, Vi điều khiển thực hiện lệnh kế tiếp(không thực hiện lệnh nhảy).

3.5.14. Lệnh nhảy nghịch với giá trị của bit nhớ

- Cú pháp: JNC bit,rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

+ Nếu bit nhớ có giá trị 0, Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt.

+ Nếu bit nhớ có giá trị 1, Vi điều khiển thực hiện lệnh kế tiếp (không thực hiện lệnh nhảy).

3.5.15. Lệnh nhảy thuận với giá trị của bit nhớ và xóa bit

- Cú pháp: JBC bit,rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

+ Nếu bit nhớ có giá trị 1, Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt, đồng thời xóa giá trị chứa trong bit nhớ đó tức là đưa bit nhớ đó về giá trị 0.

+ Nếu bit nhớ có giá trị 0, Vi điều khiển thực hiện lệnh kế tiếp

(không thực hiện lệnh nhảy).

3.5.16. Lệnh nhảy có điều kiện(so sánh giá trị của thanh ghi A và Rn)

- Cú pháp: CJNE A,direct,rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

+ Vi điều khiển nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt nếu giá trị của thanh ghi A khác giá trị của ô nhớ có địa chỉ direct, nếu bằng nhau Vi điều khiển không nhảy và thực hiện lệnh kế.

+ Ảnh hưởng của lệnh đến cờ nhớ C:

Nếu giá trị của thanh ghi A ≥ giá trị của ô nhớ có địa chỉ direct thì bit C có giá trị 0.

Nếu giá trị của thanh ghi A < giá trị của ô nhớ có địa chỉ direct thì bit C có giá trị 1.

3.5.17. Lệnh nhảy có điều kiện(so sánh giá trị của thanh ghi A và dữ liệu cho trước).

- ***Cú pháp:*** CJNE A,#data,rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

+ Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt, nếu giá trị của thanh ghi A khác giá trị dữ liệu cho trước, nếu bằng nhau Vi điều khiển không nhảy và thực hiện lệnh kế.

+ Ảnh hưởng của lệnh đến cờ nhớ C:

Nếu giá trị của thanh ghi A ≥ giá trị dữ liệu cho trước thì bit C có giá trị 0.

Nếu giá trị của thanh ghi A < giá trị dữ liệu cho trước thì bit C có giá trị 1.

3.5.18. Lệnh nhảy có điều kiện(so sánh giá trị của thanh ghi Rn và dữ liệu cho trước).

- Cú pháp: CJNE Rn,#data,rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

+ Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt nếu giá trị của thanh ghi Rn khác giá trị dữ liệu cho trước, nếu bằng nhau Vi điều khiển không nhảy và thực hiện lệnh kế.

+ Ảnh hưởng của lệnh đến cờ nhớ C:

Nếu giá trị của thanh ghi A ≥ giá trị dữ liệu cho trước thì bit C có giá trị 0.

Nếu giá trị của thanh ghi A < giá trị dữ liệu cho trước thì bit C có giá trị 1.

3.5.19. Lệnh nhảy có điều kiện(so sánh giá trị của ô nhớ có địa chỉ gián tiếp và dữ liệu cho trước)

- Cú pháp: CJNE @Ri,#data,rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

Vi điều khiển nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt nếu giá trị của ô nhớ có địa chỉ bằng giá trị của Ri khác giá trị dữ liệu cho trước, nếu bằng nhau Vi điều khiển không nhảy và thực hiện lệnh kế

Ảnh hưởng của lệnh đến cờ nhớ C:

Nếu giá trị của ô nhớ có địa chỉ gián tiếp ≥ giá trị dữ liệu cho trước thì bit C có giá trị 0.

Nếu giá trị của ô nhớ có địa chỉ gián tiếp < giá trị dữ liệu cho trước thì bit C có giá trị 1.

3.5.20. Lệnh nhảy có điều kiện kết hợp với lệnh giảm trên thanh ghi Rn

- Cú pháp: DJNZ Rn,rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

Giảm giá trị của thanh ghi Rn xuống 1 đơn vị, và nếu giá trị trong thanh ghi Rn khác 0, Vi điều khiển nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt.

Nếu giá trị trong thanh ghi Rn bằng 0, Vi điều khiển thực hiện lệnh kế tiếp.

3.5.21. Lệnh nhảy có điều kiện kết hợp với lệnh giảm trên ô nhớ có địa chỉ direct

- Cú pháp: DJNZ direct,rel
- Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte.
- Thời gian thực hiện: 2 chu kì máy.
- Công dụng:

Giảm giá trị của ô nhớ có địa chỉ direct xuống 1 đơn vị

Nếu giá trị trong ô nhớ có địa chỉ direct khác 0, Vi điều khiển

nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt.

Nếu giá trị trong ô nhớ có địa chỉ direct bằng 0, vi điều khiển thực hiện lệnh kế tiếp.

3.5.22. Lệnh delay 1 chu kì máy

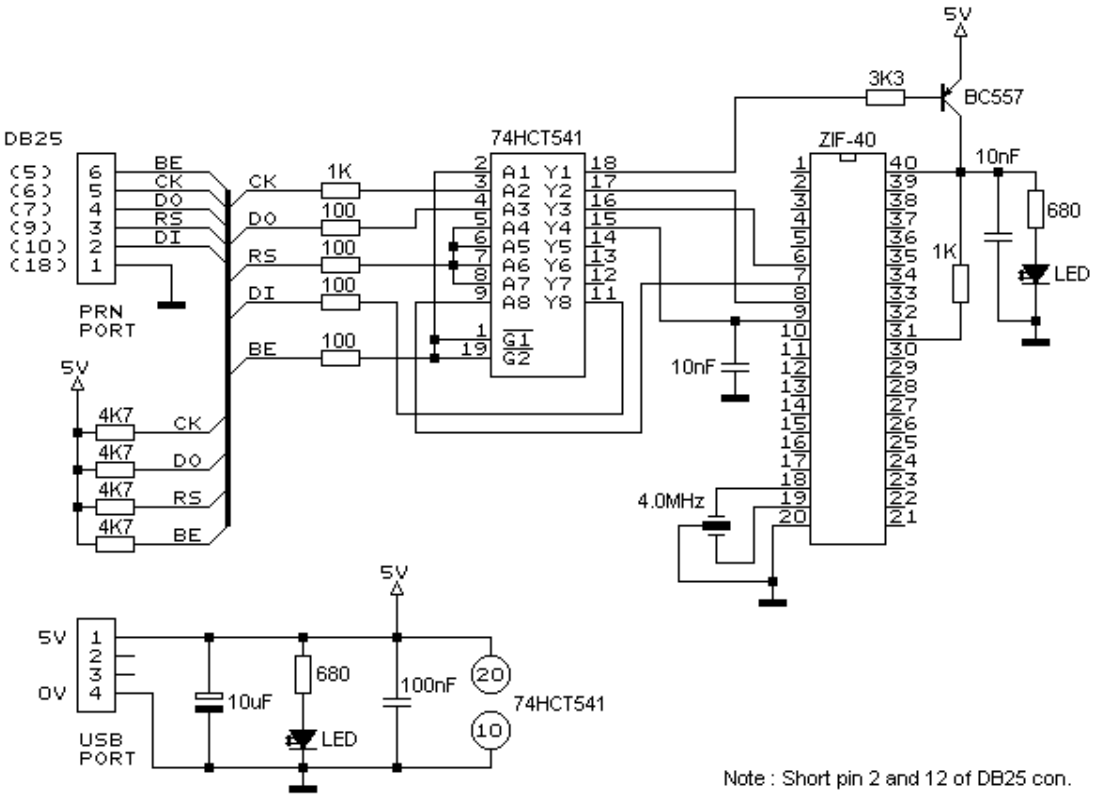
- Cú pháp: NOP
- Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.
- Thời gian thực hiện: 1 chu kì máy.
- Công dụng: delay trong 1 chu kì máy

CÁC BÀI TẬP MỞ RỘNG, NÂNG CAO VÀ GIẢI QUYẾT VẤN ĐỀ

Làm mạch nạp.

- Để đưa chương trình đã viết trên máy vi tính vào trong ROM của vi điều khiển, cần có các phần mềm riêng (hay còn gọi là phần mềm nạp) và các mạch giao tiếp tương ứng với phần mềm đó (hay còn gọi là mạch nạp). Có rất nhiều phần mềm nạp và các mạch nạp dành cho vi điều khiển, mỗi loại vi điều khiển đều có phần mềm nạp và mạch nạp dành riêng cho vi điều khiển đó.

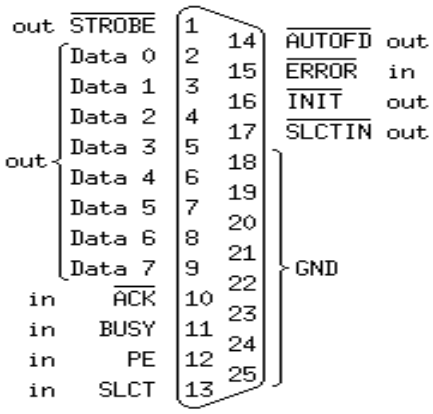
- Cũng có rất nhiều chương trình nạp cho vi điều khiển AT89Sxx (xx: hai số sau của mã vi điều khiển, ví dụ: AT89S52, AT89S53...), hiện nay phần mềm nạp ISP do "*Mohammad Asim Khan*" xây dựng được sử dụng rất phổ biến ở thế giới và Việt Nam. Mạch nạp kèm theo với phần mềm ISP rất đơn giản và được thiết kế theo kiểu nạp nối tiếp (các chương trình nạp trước đó thường là chương trình nạp song song, mạch nạp rất phức tạp,khó chế tạo). ISP có thể nạp chương trình cho vi điều khiển ngay trên board mạch hoạt động mà không cần phải chuyển vi điều khiển từ mạch hoạt động sang mạch khác để nạp như các chương trình nạp trước đây (hình 3.3).



Hình 3.3. Mạch nạp dùng 74HCT541

- PRN PORT: đường kết nối đến cổng máy in, các số tương ứng trong ngoặc là số của chân trên cổng máy in (hình 3.6).

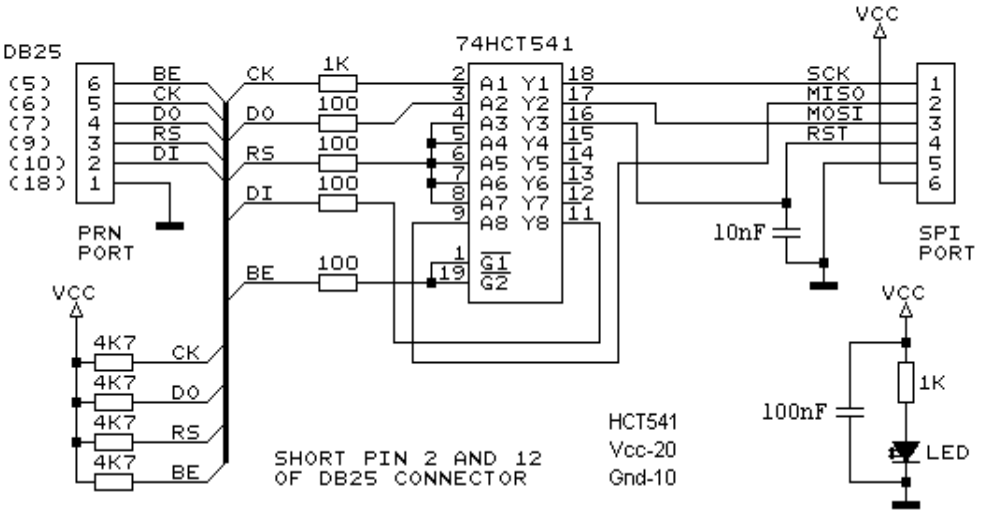
+ Nối ngắn mạch chân 2 và chân 12 của cổng máy in.
+ 74HCT245 được cấp nguồn 5V vào chân 10 và chân 20.



Hình 3.6: Sơ đồ chân của cổng máy in

Sơ đồ này chỉ dùng cho mạch chỉ thực hiện công việc duy nhất là nạp chương trình cho vi điều khiển.

Để nạp trực tiếp cho mạch đang hoạt động, dùng sơ đồ sau (hình 3.7):



Hình 3.7. Mạch nạp dùng 74HCT541

Nguồn cấp cho 74HTC541 được lấy từ mạch ổn áp trên board mạch vi điều khiển, điện thế dương 5V đưa vào chân 20, chân 10 nối với điện thế 0V (Ground). LED dùng báo hiệu trạng thái hoạt động của mạch.(hình 3.7)

Trên SPR PORT: chân số 1 nối với chân số 6 của vi điều khiển T89Sxx,

chân số 2 nối với chân số 7 của vi điều khiển AT89Sxx

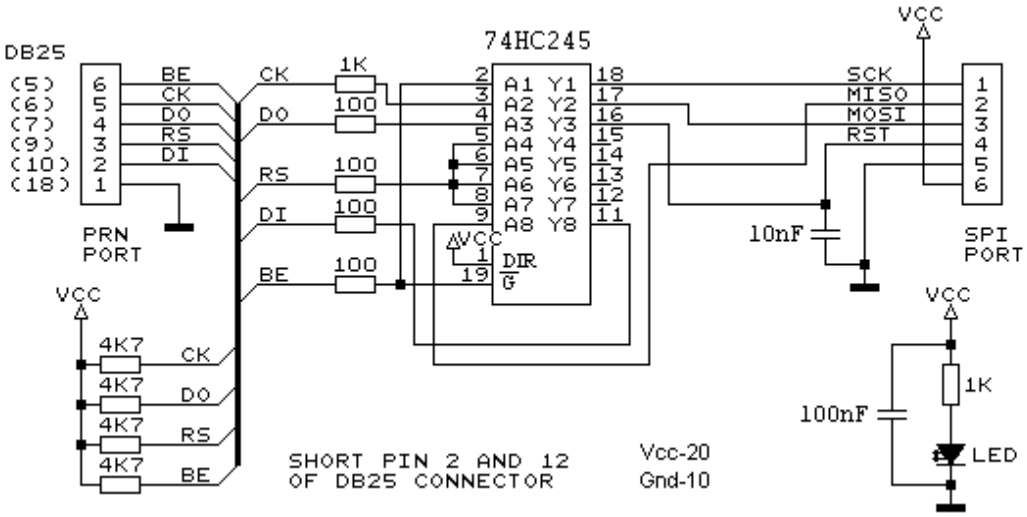
chân số 3 nối với chân số 8 của vi điều khiển AT89Sxx

chân số 4 nối với chân số 9 của vi điều khiển AT89Sxx

chân số 5 nối với chân - từ ngõ OUT của mạch ổn áp

chân số 6 nối với chân + từ ngõ OUT của mạch ổn áp

Ngoài ra có thể dùng 74HC245 theo sơ đồ dưới (hình 3.6):



Hình 3.6. Mạch nạp dùng 74HC245

Bài 1: Viết đoạn chương trình đọc nội dung của ô nhớ 30h. Nếu giá trị đọc lớn hơn hay bằng 10 thì xuất 10 ra P0, ngược lại thì xuất giá trị vừa đọc ra P0.

Gợi ý/bài giải:

```
                    ORG 0000H
MAIN:
        MOV     A,30H
        CJNE    A,#10,KT_C
KT_C:
        JC      NHO_10
        MOV     P0,A
NHO_10:
        MOV     P0,A ;
        JMP   $
        END
```

Bài 2: Viết đoạn chương trình xuất các giá trị trong ô nhớ 30h – 3Fh ra P1 (giữa các lần xuất có thời gian trì hoãn).

Gợi ý/bài giải:ORG 0000H

```
MAIN:
        MOV     P1,30H
        CALL    DELAY_1S
.....(Những ô nhớ sau viết tương
tự).....

DELAY_1S:
        MOV     R0,#20
DELAY:
        MOV     TMOD,#01H
        MOV     TH0,#HIGH(-50000)
        MOV     TL0,#LOW(-50000)
        SET     TR0
        JNB     TF0,$
        CLR     TR0
        CLR     TF0
        DJNZ    R0,DELAY
RET
END.
```

Bai 3: Tạo đồng thơi 2 sóng vuông, sử dụng ngăt́ timer.

Giải:

Chương trình mẫu

```
                ORG     0000H
               LJMP     MAIN
                ORG     0BH
                LJMP    T0ISR
                ORG     1BH
               L JMP    T1ISR

                ORG     30H
        MAIN:  MOV  TMOD,#12H
                MOV     TH0,#-71     ; 7kHz sử dụng ngăt́ Timer1
                SETB    TR0
                SETB    TF1
                MOV     IE,#8AH
                SJMP $
        ;
        ************************************************************
        ************
        ; TIMER 0 ISR, MODE 2 AUTORELOAD
T0ISR:          CPL P1.7
                RETI
        ;
        ************************************************************
        ************
        ; TIMER 1 ISR, MODE 1
T1ISR:          CLR     TR1
                MOV     TH1,#HIGH(-1000)
                MOV     TL1,#LOW(-1000)
                SETB    TR1
                CPL     P1.6
                RETI
                END
```

Bai 4: Viêt́ một đoạn chương trình sử dụng cać ngăt́ trạng thaí.

Gợi yí:

Viêt́ đoạn chương trình sử dung ngăt́.

```
        ORG 0H
 MAIN:
        LJMP MAIN          ; nhảy tới Main
        ORG 03H            ; external interrupt 0
        JMP EXINTER0
        ORG 0BH            ; timer 0 interrupt
        JMP COUNTER0
        ORG 13H            ; external interrupt 1
        JMP EXINTER1
```

```
        ORG 1BH          ;timer 1 interrupt
        JMP COUNTER1
MAIN:
        EXINTER0 ; chương trình phục vụ ngắt 0
        RETI
        EXINTER1 ; chương trình phục vụ ngắt 1
        RETI
        COUNTER0         ; chương trình phục vụ bộ đếm 0
        RETI
        COUNTER1         ; chương trình phục v ụ bộ đếm 1
        RETI
        END
```

Bai 5:

Giả sử có một cảm biến ở cổng P2, hãy viết chương trình đọc nhiệt độ và so sánh với giá trị 60. Dựa vào kết quả kiểm tra đặt cać gia´trị nhiệt độ vaò cać thanh ghi sau:

N ếu T=60 thì A=T
N ếu T>60 thì R1=T
N ếu T<60 thì R2=T

Gợi y´:

```
        MOV P1,#0FFH
        MOV A,P1
        CJNE A,#60,LON
        SJMP THOAT
LON:    JC NHO      ;C=1
        MOV R1,A
        SJMP THOAT
NHO:    MOV R2,A
THOAT:  END
```

Điều kiện thực hiện bai học:

- Máy tính cá nhân.
- Đồng hồ DVOM/VOM.
- Máy nạp chip vạn năng.
- - Vi điều khiển.
- Vi mạch số các loại.
- Điện trở.
- Tụ.
- Rơ le.

- Led các loại.
- Mạch in.
- Dây nối.
- Chì hàn.
- Sơ đồ, IC họ 8051.
- Panel chân cắm nhỏ.
- Tài liệu hướng dẫn sử dụng họ 8051
- Sơ đồ mạch.
- IC họ 8051 - CMOS, TTL – 555.
- Kit thực tập và mô hình kèm theo.
- Phần mềm chương trình Assembler.
- Sơ đồ- các bài tập ứng dụng trên kit thực hành.

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức:

- Phân biệt được các kiểu định địa chỉ và dữ liệu.
- Trình bày được đặc tính và công dụng của từng lệnh trong 8051.

+ Về kỹ năng:

- Thực hiện viết các chương trình theo yêu cầu cho trước

+ Thái độ: Đánh giá phong cách, thái độ học tập

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm

+ Về kỹ năng: Đánh giá kỹ năng thực hành Mỗi sinh viên, hoặc mỗi nhóm học viên thực hiện công việc theo yêu cầu của giáo viên.Tiêu chí đánh giá theo các nội dung:

- Độ chính xác của công việc
- Thời gian thực hiện công việc
- Độ chính xác theo yêu cầu kỹ thuật

+ Thái độ: Tỉ mỉ, cẩn thận, chính xác.

BAI 4
BỘ ĐỊNH THỜI
Mã bài: MĐ24-04

***Giới thiệu*:**

Trong quá trình viết chương trình dùng vi điều khiển để vận hành một hệ thống thì việc khởi tạo và đọc đúng bộ định thời là một vấn đề quan trọng.

Việc hiểu và sử dụng đúng chức năng của bộ timer là một yêu cầu cấp thiết trong việc lập trình cho vi điều khiển. Trong bài 4, chúng ta đi vào tìm hiểu cách thức khởi tạo và sử dụng bộ đinh thời của vi điều khiển.

Mục tiêu của bài:

- Trình bày được cấu tạo và các chế độ làm việc của bộ định thời 8051 theo nội dung đã học.
- Thực hiện khởi tạo bộ nhớ đúng yêu cầu kỹ thuật.
- Thực hiện đọc bộ định thời trong khi hoạt động đúng yêu cầu kỹ thuật.
- Thực hiện lập trình điều khiển dùng bộ định thời đúng yêu cầu kỹ thuật.

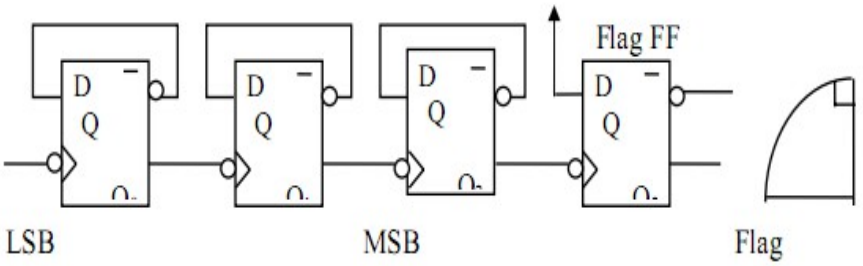
Nội dung chính:

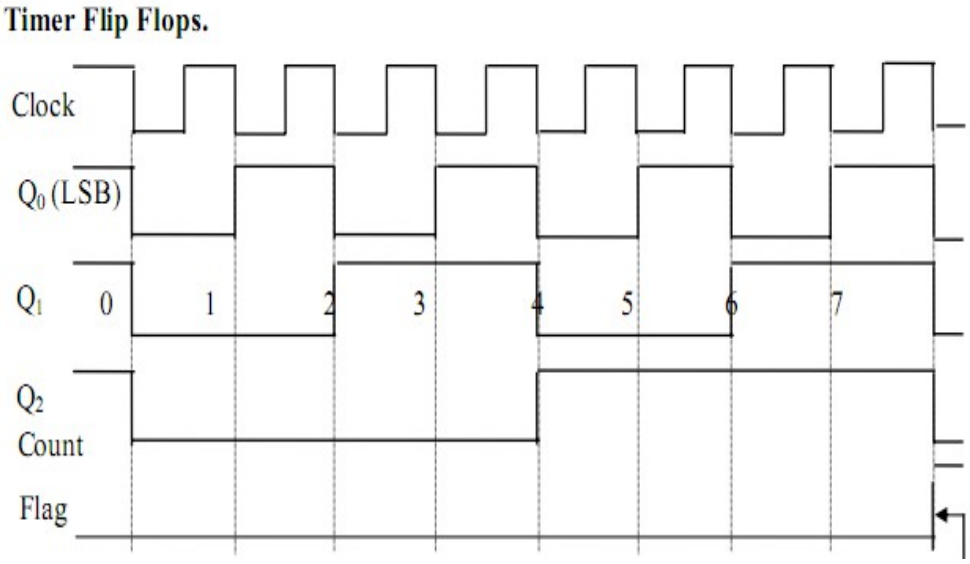
1. ***Mở đầu:***

Mục tiêu: Hiểu được chức năng của bộ định thời trong vi điều khiển.

- Một bộ định thời là một chuỗi các flipflop với mỗi flipflop là một mạch chia hai,chuỗi này nhận một tín hiệu ngõ vào làm nguồn xung clock.Xung clock đặt vào flipflop thứ nhất flipflop này chia đơi tần số xung clock. Ngõ ra của flipflop thứ nhất trở thành nguồn xung clock cho flipflop thứ hai,nguồn xung clock này cũng được chia cho 2,v.v..Vì mỗi một tần kế tiếp nhau đều chia cho 2 nên bộ định thời có n tầng sẽ chia tần số xung clock ở ngõ vào của bộ này cho 2.
- Ngõ ra của tần cuối cùng làm xung clock cho một flipflop báo tràn bộ định thời hay còngọi là cờ tràn (overflow flag),cờ tràn này được kiểm tra bởi phần mềm hoặc tạo ra một bộ ngắt.Giá trị nhị phân trong các flipflop của bộ định thời là số đếm của các xung clock từ bộ định thời bắt đầu đếm. Vi´dụ một bộ định thời 16bit sẽ đếm từ 0000H đến FFFH.Cờ tràn được set bằng 1 khi xảy ra tràn số đếm từ FFFFH xuống 0000H.

Hoạt động của Timer 3 bit đơn giản được minh họa như sau:





Hình 4.1

- Hoạt động của một bộ định thời đơn giản được minh họa trong hình bên trên (hình 4.1),bộ định thời 3 bit.Mỗi một tầng là một D-FF kích khởi cạnh âm hoạt động như một mạch chia cho 2 do ta nối ngõ ra Q đảo với ngõ vào D. Flipflop cờ đơn giản là một mạch chốt D được set bằng 1 bởi tầng cuối của bộ định thời. Giản đồ thời gian ở hình trên cho thấy tầng thứ nhất (Q_0) chia hai tần số xung clock, tầng thứ hai chia 4 tần số xung clock và v.v… Số đếm (count) được ghi ở dạng thập phân và được kiểm tra dễ dàng bằng cách khảo sát trạng thái của 3 flipflop.Ví dụ: Số đếm là 4 xuất hiện khi Q_2=1, Q_1=0 và Q_0=0(4_{10}=100_2).Các flipflop ở hình trên là các flipflop tác động cạnh âm (nghĩa là ngõ ra Q của các flipflop đổi trạng thái theo cạnh âm của xung clock).Khi số đếm tràn từ 111_2 xuống 000_2,ngõ ra Q2 có cạnh âm (1 hay 0) làm cho trạng thái của flipflop cờ đổi từ 0 lên 1(ngõ vào D của flipflop này luôn luôn ở logic 1).

- Bộ định thời được sử dụng trong hầu hết các ứng dụng hướng điều khiển và 8051 với các bộ định thời trên chip không phải là trường hợp ngoại lệ. 8051 có hai bộ định thời 16 bit, mỗi bộ có 4 chế độ hoạt động. Bộ định thời thứ 3 với ba chế độ hoạt động được thêm vào đối với chip 8051.Các bộ định thời được dùng để:

+ Định thời trong một khoảng thời gian.
+ Đếm sự kiện.
+ Tạo tốc độ baud cho port nối tiếp của chip 8051.

Với bộ định thời 16 bit, tầng cuối cùng (tầng thứ 16)chia tần số xung
clock của ngõ vào của bộ định thời cho 2^{16}=65536.

- Trong các ứng dụng định thời trong một khoảng thời gian, bộ định thời được lập trình sao cho sẽ tràn sau một khoảng thời gian quy định và set cờ tràn của bộ định thời bằng 1. Cờ tràn được sử dụng để đồng bộ chương trình nhằm thực hiện một công việc như là kiểm tra trạng thái của các ngõ nhập hoặc gửi dữ liệu đến các ngõ xuất. Các ứng dụng khác có thể sử dụng xung clock của bộ định thời để đo khoảng thời gian giữa hai sự kiện (Ví dụ: đo độ rộng xung).

- Việc đếm sự kiện được dùng để xác định số lần xuất hiện của một sự kiện hơn là đo thời gian của các sự kiện.Từ "sự kiện" là một kích thích bên ngoài cung cấp một chuyển trạng thái từ 1 xuống 0 tới một chân của chip 8051. Các bộ định thời cũng có thể cung cấp xung clock tốc độ baud cho port nối tiếp bên trong 8051.

- Các bộ định thời 8051 được truy xuất bằng cách sử dụng 6 thanh ghi chức năng đặc biệt.Với bộ định thời thứ 3 của chip 8052, ta có thêm 5 thanh ghi chức năng đặc biệt nữa để truy xuất bộ định thời này.

SFR của bit bộ định thời	Mục đích	Địa chỉ	Định địa chỉ
TCON	Điều khiển	88H	Có
TMOD	Chọn chế độ	89H	Không
TL0	Byte thấp của bộ định thời 0	8AH	Không
TL1	Byte thấp của bộ định thời 1	8BH	Không
TH0	Byte cao của bộ định thời 0	8CH	Không

TH1	Byte cao của bộ định thời 1	8DH	Không
T2CON	Điều khiển bộ định thời 2	C8H	Có
RCAP2H	Nhận byte thấp của bộ định thời	CAH	Không
RCAP2L	2	CBH	Không
TL2	Nhận byte cao của bộ định thời	CCH	Không
TH2	2	CDH	Không
	Byte thấp của bộ định thời 2		

2.Thanh ghi SFR của timer.

Mục tiêu: Hiểu được chức năng của từng thanh ghi của Timer.

- *Thanh ghi chế độ định thời (TMOD).*

Thanh ghi TMOD (timer mode register) chứa hai nhóm 4 bit dùng để thiết lập chế độ hoạt động cho bộ định thời 0 và bộ định thời 1.TMOD không được định địa chỉ từng bit. Một cách tổng quát,TMOD được nạp một lần bởi phần mềm ở thời điểm bắt đầu của một chương trình để khởi động chế độ hoạt động của bộ định thời.Sau đó bộ định thời có thể được dừng, được bắt đầu,v.v…bằng cách truy xuất các thanh ghi chức năng đặc biệt khác của bộ định thời.

❖ TMOD Register

MSB							LSB
GATE	C/T	M1	M	GATE	C/T	M1	M0
Timer 1				Timer 0			

Bảngh 4.1. Timer TMOD

❖ 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 của 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 (bảng 4.2, 4.3).

Bit	Tên	Bộ	Mô tả
7	GATE	1	Bit điều khiển cổng.Khi được set lên 1,bộ định
6	C/ *T*	1	Bit chọn chức năng đếm hoặc định thời:

5	M1	1	Bit chọn chế độ thứ nhất
4	M0	1	Bit chọn chế độ thứ hai
3	GATE	0	Bit điều khiển cổng cho bộ định thời 0
2	C/ *T*	0	Bit chọn chức năng đếm hoặc định thời cho bộ
1	M1	0	Bit chọn chế độ thứ nhất
0	M0	0	Bit chọn chế độ thứ hai

Bảng 4.2. Thanh ghi chọn chế độ định thời

M1	M0	Chế độ	Mô tả
0	0	0	Chế độ định thời 13 bit
0	1	1	Chế độ định thời 16 bit
1	0	2	Chế độ tự động nạp lại 8 bit
1	1	3	Chế độ định thời chia sẻ

Bảng 4.3. Các chế độ định thời

Thanh ghi điều khiển định thời (TCON).

Thanh ghi TCON chứa các bit điều khiển, bit trạng thái của bộ định thời 0 và bộ định thời 1.Bit cao trong TCON (TCON4 – TCON7) được dùng để điều khiển cho bộ định thời hoạt động, ngưng (TR0,TR1) hoặc để báo bộ định thời tràn (TF0,TF1). Bit thấp của TCON(TCON0- TCON3) không dùng để điều khiển các bộ định thời,chúng được dùng để phát hiện và khởi động các ngắt ngoài (bảng 4.4).

Bit	Ký hiệu	Địa chỉ bit	Mô tả
TCON.7	TF1	8FH	Cờ tràn của bộ định thời 1.Cờ này được set bởi phần cứng khi có tràn được xóa bởi phần
TCON.6	TR1	8EH	Bit điều khiển hoạt động của bộ định thời 1.Bit này được set hoặc được xóa bởi phần mềm
TCON.5	TF0	8DH	Cờ tràn của bộ định thời 0

TCON.4	TR0	8CH	Bit điều khiển hoạt động của bộ định thời 0
TCON.3	IE1	8BH	Cờ ngắt bên ngoài 1.Cờ này được set bởi phần cứng khi có cạnh âm(xuống) xuất hiện trên
TCON.2	IT1	8AH	Bit chọn ngắt ngoài 1 thuộc loại tác động cạnh hay tác động mức 0:mức;1:cạnh
TCON.1	IE0	89H	Cờ ngắt bên ngoài 0 (kích khởi cạnh)
TCON.0	IT0	88H	Bit chọn ngắt ngoài 0 thuộc loại tác động cạnh hay tác động mức 0:mức;1:cạnh

Bảng 4.4. Thanh ghi điều khiển định thời TCON

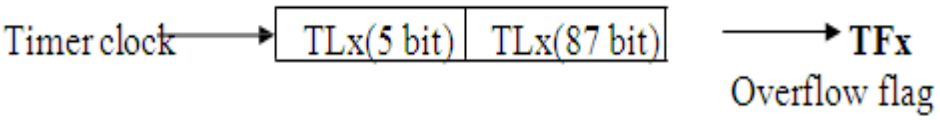
3. ***Các chế độ làm việc***

Mục tiêu: Hiểu và chọn được các chế độ làm việc của Timer.

Do ta có hai bộ định thời trên chip 8051,kí hiệu "x" được sử dụng để chỉ hoặc bộ định thời 0 hoặc bộ định thời 1.Thí dụ : THx có thể là TH0 hay TH1 tùy theo bộ định thời 0 hay 1.

3.1. ***Chế độ định thời 13 bit (chế độ 0).***

Chế độ định thời 0 là chế độ định thời 13 bit cung cấp khả năng tương thích với bộ vi điều khiển tiền nhiệm 8048.Chế độ này không được dùng cho các thiết kế mới.Byte cao của bộ định thời THx được ghép cascade với 5 bit thấp của byte thấp của bộ định thời THx để tạo thành một bộ định thời 13bit. Ba bit cao của TLx không sử dụng.



3.2. ***Chế độ định thời 16 bit (chế độ 1).***

Chế độ định thời 16 bit có cấu hình giống như chế độ định thời 13 bit. Khi có xung clock bộ định thời đếm lên: 0000H,0001H,0002H … FFFFH (65535). Một tràn sẽ xuất hiện khi có sự chuyển số đếm từ FFFFH xuống 0000H, sự kiện này sẽ set cờ tràn bằng 1 (TFx = 1) và bộ định thời tiếp tục đếm. Bit có ý nghĩa lớn nhất (MSB : Most significant bit) của giá trị trong các thanh ghi định thời là bit 7 của THx và bit có ý nghĩa thấp nhất (LSB : Least significan bit) là bit 0 của TLx. Các thanh ghi định

thời (THx,TLx) có thể được đọc hoặc ghi bằng phần mềm ở bất kỳ thời điểm nào.

Timer clock

TLx THx

3.3. *Chế độ tự nạp lại 8 bit (chế độ 2).*

Chế độ 2 là chế độ tự nạp lại 8bit. Byte thấp của bộ định thời (TLx) hoạt động định thời 8 bit trong khi byte cao của bộ định thời lưu giữ giá trị nạp lại. Khi số đếm tràn từ FFH xuống 00H không chỉ cờ tràn của bộ định thời set lên 1 mà giá trị trong THx còn được nạp vào TLx, việc đếm sẽ được tiếp tục từ giá trị này cho đến khi xảy ra lần tràn kế tiếp,v.v…

Timer clock TLx TFx

Reload

THx

3.4. *Chế độ định thời tách biệt timer (chế độ 3).*

Chế độ 3 là chế độ định thời chia xẻ và có hoạt động khác nhau cho

từng bộ định thời. Bộ định thời 0 của chế độ 3 được chia thành 2 bộ định thời 8 bit hoạt động riêng lẻ TL0 và TF1, mỗi bộ định thời sẽ Set các cờ tràn tương ứng TF0 và TF1 khi xảy ra tràn.

- Bộ định thời 1 không hoạt động ở chế độ 3 nhưng có thể được khởi động bằng cách chuyển bộ định thời này vào một trong các chế độ khác. Giới hạn duy nhất là cờ tràn TF1 của bộ định thời 1 không bị ảnh hưởng bởi bộ định thời 1 khi bộ này xảy ra tràn vì TF1 được nối với bộ định thời TH0.

- Chế độ chủ yếu cung cấp thêm một bộ định thời 8 bit nữa, nghĩa là

8051 có thêm bộ định thời thứ 3.Khi bộ định thời 0 của chế độ 3,bộ định thời 1 có thể hoạt động hoặc ngưng bằng cách chuyển bộ này ra khỏi chế độ 3 hoặc vào chế độ 3. Bộ định thời 1 có thể được sử dụng bởi port nối tiếp (lúc này bộ định thời 1 làm nhiệm vụ của bộ tạo xung clock tốc độ baud) hoặc được sử dụng theo một cách nào đó nhưng không yêu cầu ngắt (vì bộ định thời lúc này không còn nối với TF1).

***4.* Nguồn cung cấp xung cho Timer.**

Mục tiêu : Hiểu được chức năng hoạt động của bộ định thời .

Có 2 khả năng tạo ra nguồn xung clock này,việc lựa chọn khả năng

nào do ta thiết lập bit C/ *T* (counter/timer) của thanh ghi TMOD bằng 1 hay 0 khi bộ định thời được khởi động.Một nguồn xung clock được dùng để định thời trong một khoảng thời gian, nguồn xung clock còn lại được dùng để đếm sự kiện.

4.1. Chức năng định thời.

Nếu C/ *T* =0,hoạt động định thời được chọn và nguồn xung clock của bộ định thời do mạch dao động bên trong chip tạo ra. Một mạch chia 12 tầng được thêm vào để giảm tần số xung clock đến một giá trị thích hợp với hầu hết các ứng dụng. Lúc này bộ định thời được dùng để định thời trong một khoảng thời gian.Các thanh ghi định thời (TLx/THx) đếm lên với tần số xung clock bằng 1/12 tần số của mạch dao động trên chip (nghĩa là nếu thạch anh là 12MHz, tần số xung clock là 1MHz). Bộ định thời sẽ tràn sau một số xung clock cố định phụ thuộc vào giá trị ban đầu nạp cho các thanh ghi định thời(TLx/THx).

4.2. ***Đếm sự kiện.***

- Nếu C/ *T* = 1, bộ định thời được cung cấp xung clock từ 1 nguồn tạo xung bên ngoài. Trong đa số các ứng dụng,nguồn xung clock này cung cấp cho bộ định thời một xung dựa trên việc xảy ra một sự kiện bộ định thời bây giờ đếm sự kiện. Số các sự kiện được xác định trong phần mềm bằng cách đọc các thanh ghi định thời (TLx/THx), giá trị 16-bit trong các thanh ghi này tăng theo mỗi sự kiện. Hai chân của port 3(P3.4 và P3.5) bây giờ trở thành ngõ vào xung clock cho các bộ định thời. Chân P3.4 là ngõ vào xung clock cho bộ định thời 0 (ta còn gọi là chân T0 ở ngữ cảnh này), chân P3.5 là ngõ vào xung clock cho bộ định thời 1(T1).

- Trong các ứng dụng đếm sự kiện, các thanh ghi định thời tăng mỗi khi xảy ra chuyển trạng thái từ 1 xuống 0 ở ngõ vào Tx (T0 hoặc 1).Ngõ vào Tx được lấy mẫu trong suốt mỗi một chu kỳ máy, vậy thì khi ngõ vào ở mức cao trong một chu kỳ và mức thấp trong chu kỳ kế, số đếm được tăng. Phải mất 2 chu kỳ máy (2us) để nhận biết sự chuyển trạng thái từ 1 xuống 0, tần số cực đại của nguồn xung clock bên ngoài là 500KHz (với giả sử chip vi điều khiển hoạt động với thạch anh 12 MHz).

5. **Khởi động, dừng va`điều khiển Timer.**

Mục tiêu: Điều khiển được sự hoạt động của Timer.

- Cách đơn giản nhất để khởi động và dừng các bộ định thời là sử dụng bit điều khiển hoạt động TRx trong thanh ghi TCON. TRx được xóa

khi thiết lập hệ thống; nghĩa là các bộ định thời ngưng hoạt động. Ngược lại nếu ta set bit TRx nghĩa là cho phép bộ định thời hoạt động. Do thanh ghi TCON là thanh ghi được định địa chỉ từng bit, ta dễ dàng khởi động hoặc dừng các bộ định thời bằng chương trình.

- Một phương pháp khác để điều khiển các bộ định thời là sử dụng bit GATE trong thanh ghi TMOD và ngõ vào /INTx. Bằng cách set bit GATE lên 1 ta cho phép bộ định thời được điều khiển bởi /INTx.

6. Khởi tạo va`truy xuất thanh ghi Timer.

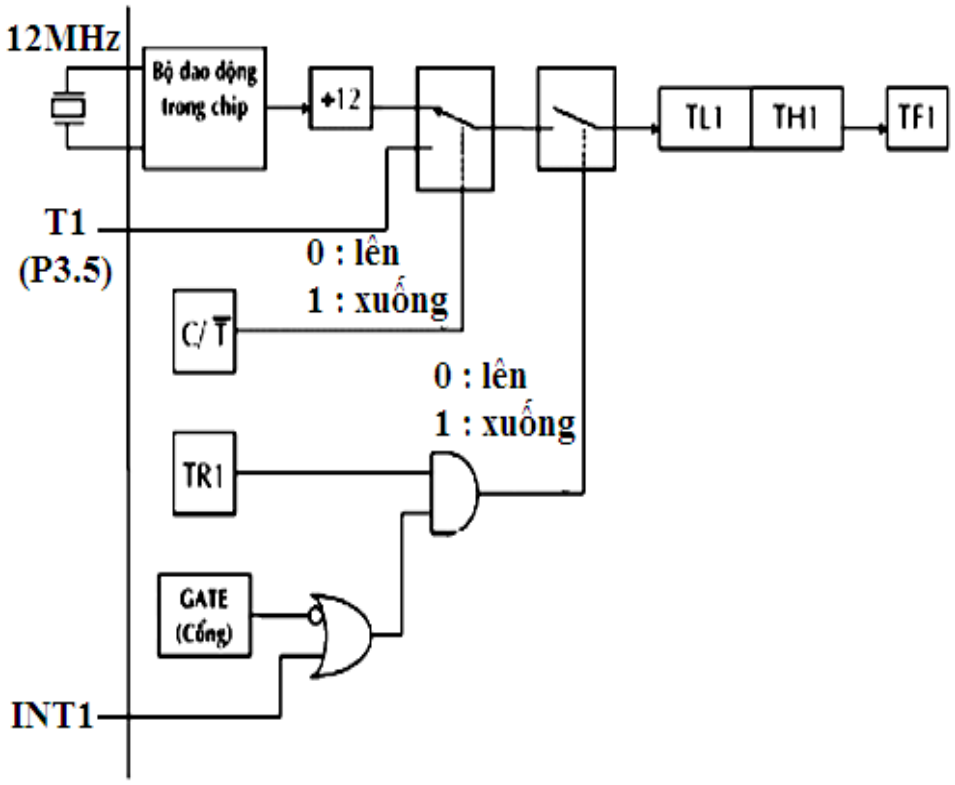
Mục tiêu: Đặt được các giá trị đếm của Timer và truy xuất giá trị của Timer khi cần thiết.

Các bộ định thời thường được khởi động một lần ở thời điểm bắt đầu chương trình để thiết lập chế độ hoạt động yêu cầu.Trong thân của chương trình,các bộ định thời được điều khiển hoạt động, dừng, kiểm tra các bit cờ và xóa, cá thanh ghi định thời được đọc hoặc cập nhật tuỳ theo yêu cầu ứng dụng. TMOD là thanh ghi được khởi động trước tiên vì đây là thanh ghi thiết lập chế độ hoạt động.

- Cać tać vụ:

  Đặt chê´độ laàm việc.
  Cho timer chạy.
  Dưng timer.
  Kiểm tra cơ`traàn.
  Xoá cơ`traàn.
  Đọc va`cập nhật cać thanh ghi timer.

Chu´y´: khi co´sử dụng ngắt ngoaì (TD: chê´độ 1)



Hình 4.2 Ngắt ngoài

Vi´dụ:

Đặt chê´độ laìm việc: Khởi động bộ định thời 1 hoạt động ở chế độ 16-bit (chế độ 1), xung clock được cấp từ mạch dao động trên chip(định thời một khỏang thời gian).

MOV TMOD,#00010000B

Timer 1: Gate = 0, C/T = 0, M1M0 = 01 (mode 1)

Timer 0 : Gate = 0, C/T = 0, M1M0 = 00(mode 0)

+ Trong trường hợp cần đến số đếm ban đầu, các thanh ghi định thời TL1/TH1 cũng phải được khởi động. Cần nhớ là các bộ định thời đếm lên và thiết lập cờ tràn bằng 1 khi xảy ra tràn số đếm từ FFFFH xuống 0000H, vậy thì một khoảng thời gian 100us có thể được định thời bằng cách khởi động TL1/TH1 chứa số đếm nhỏ hơn 0000H một lượng là 100 nghĩa là -100 hay FF9CH. Các lệnh sau thực hiện điều này.

MOV TL1,#9CH

MOV TH1,#0FFH

Cho timer chạy: Kế đến bộ định thời bắt đầu hoạt động bằng cách thiết lập bit điềukhiển hoạt động bằng 1 như sau :

SETB TR1

- Cờ tràn được tự động sau khoảng thời gian 100us. Phần mềm có

thể chứa 1 vịng lặp trì hỗn thời gian 100us bằng cách sử dụng một lệnh rẽ nhánh và lặp lại chính lệnh này trong khi cờ tràn chứa được set bằng 1.

```
WAIT : JNB TF1, WAIT
```

+ Dừng timer: Khi bộ định thời tràn ta cần dừng bộ định thời và xóa cờ tràn bằng phần mềm:

```
CLR TR1
CLR TF1
Kiểm tra cơ`va`xoá cơ:
WAIT:  JNB   TF1,WAIT
       CLR   TR1   ;dưng timer T1
       CLR   TF1   ; xoá cơ`báo tran
```

+ Cập nhật cać thanh ghi timer

```
MOV     TL1,#9CH
MOV     TH1,#0FFH
```

6.1. Đọc bộ định thời đang hoạt động.

- Trong một số ứng dụng ta cần phải đọc giá trị (nội dung) chứa trong các thanh ghi định thời đang hoạt động.Do ta phải đọc 2 thanh ghi định thời bằng 2 dòng lệnh liên tiếp (do không có lệnh đọc đồng thời cả hai thanh ghi định thời này), một số pha có thể xuất hiện nếu có tràn từ byte thấp chuyển sang byte cao giữa hai lần đọc và do vậy không thể đọc đúng được giá trị cần đọc. Giải pháp đưa ra là trước tiên ta phải đọc byte cao kế đến đọc byte thấp rồi đọc byte cao lần nữa.
- Nếu byte cao thay đổi giá trị, ta lặp lại các thao tác đọc vừa nêu. Các lênh sau đây đọc nội dung các thanh ghi định thời TL1/TH1, đưa vào các thanh ghi R6/R7 và giải quyết vấn đề vừa nêu:

```
AGAIN :   MOV A, TH1
          MOV R6,TL1
          CJNE A,TH1, AGAIN MOV R7,A
```

Ví dụ: Viết chương trình tạo sóng vuông 10KHz trên chân P0.0 bằng cách sử dụng bộ định thời 0. (Thạch anh 12MHz)

```
ORG       0030H          ; thiết lập gốc của chương trình
MOV TMOD,#00000010B      ;( #02H ) chế độ tự nạp lai
      MOV TH0,#206       ; TH0 chứa giá trị 206 =0CEH= -50
      SETB TR0           ; cho phép bộ định thời hoạt động
LOOP:                    JNB TF0,LOOP   ; chờ timer 0 tràn
            CLR TF0      ; xóa cờ tràn chuẩn bị cho lần sau
            CPL P0.0     ; đổi trạng thái bit P0.0

            SJMP LOOP    ; nhảy về nhãn LOOP, lăp lại
```

```
        END
```

Vi´dụ: Viết chương trình tạo sóng vuông 1KHz trên chân P1.0 sử dụng bộ định thời 0.

```
        ORG     0030H                   ; thiết lập gốc của chương trình
        MOV TMOD,#00000001B     ; chế độ định thời 16 bit
LOOP:
        MOV TH0,#0FEH           ; nạp trước cho TH0 byte cao của -500
        MOV TL0,#0CH            ; nạp trước cho TL0 byte thấp của -500
        SETB TR0                ; cho phép bộ định thời hoạt động
WAIT:
        JNB TF0,WAIT            ; chờ timer 0 tràn
        CLR TR0                 ; dừng bộ định
        thời CLR TF0            ; xóa cờ tràn
        CPL P1.0                ; lấy bù
        SJMP LOOP               ; lặp lại
        END
```

6.2. Thơi gian ngăń va`thơi gian dai.

Khoảng thơì gian định thơì cực đại (µs)

Khoảng thơì gian	Ky~thuật
≈ 10	Điêù chỉnh phâǹ mêm
256	Bộ định thơì 8 bit tự động nạp lại
65536	Bộ định thơì 16 bit
Không giơí hạn	Bộ định thơì 16 bit + cać voòng lặp

Vi´dụ: Tạo dạng xung trên chân P1.0

Viêt một chương trình tạo ra dạng soń g co´chu ky`trên P1.0 vơí tâǹ sô´cao nhât co´thể đươc. Tâǹ sô´va`chu ky`nhiệm vụ của dạng soń g naỳ la` bao nhiêu? (giả sử duùng XTAL = 12MHz→ 1MC = 1µs).

→ Vơí dạng soń g rât ngăń: không câǹ timer

```
        ORG         8000H
LOOP:   SETB        P1.0        ; 1MC
        CLR         P1.0        ; 1MC
SJMP LOOP       ; 2MC
```

Tâǹ sô´= 250 Khz (1/4 µs).

TON = 1µs, TOFF = 3µs → duty cycle = TON/(TON + TOFF) = 25Khz (1/4 µs).

Tạo soń g vuông 10KHz ở chân P1.0

Tạo soń g vuông 10KHz

Tần sô´= 10KHz chu ky`T=1/10000 = 10 us

TON = 50 US, TOFF = 50 us

Dung mode 2 (8 bit mode), vi`khoảng thơi gian <256 us

```
ORG     8000H
MOV     TMOD,#02H; chê´độ tự nạp lại 8 bit
MOV     TH0,#-50        ; 256-50=206
SETB    TR0             ; cho timer T0 chạy
LOOP: JNB     TF0,LOOP      ; đợi timer T0 tràn
      CLR     TF0           ; xoá cơ`báo tràn
      CPL     P1.0          ; đảo bit cổng
      SJMP    LOOP          ; lặp lại
      END
```

7. **Timer 2 của 8052.**

Mục tiêu : Hiểu được chức năng của Timer 2 trong vi điều khiển 8052

- Timer 2 là bộ định thời 16 bit (chỉ có trong họ 8x52). Giá trị đếm của timer 2 chứa trong các thanh ghi TH2 và TL2. Giống như timer 0 và timer1, timer 2 cũng hoạt động như bộ định thời (timer) hay đếm sự kiện (counter). Chế độ định thời đếm bằng dao động nội, chế độ đếm sự kiện đếm bằng xung ngoài tại chân T2 (P1.0) và chọn chế độ bằng bit C/ T 2 của thanh ghi T2CON. Các thanh ghi điều khiển timer 2 bao gồm: T2CON, T2MOD, RCAP2H, RCAP2L, TH2 và TL2.

- Timer 2 có 3 chế độ hoạt động: capture (giữ), autoreloaf (tự động nạp lại) và tạo tốc độ baud (chọn chế độ trong thanh ghi T2CON). Các bit chọn chế độ được mô tả như bảng 4.6. chọn chế độ trong Timer 2.

RCLK	TCLK	CP/RL2	TR2	Chế độ
0	0	0	1	Tự động nạp lại 16 bit
0	0	1	1	Giữ 16 bit
X	1	X	1	Tạo tốc độ baud
1	X	X	1	

X	X	X	0	Ngưng

7.1. Các thanh ghi điều khiển Timer 2.

❖ Thanh ghi T2CON:

Bảng 4.7. Nội dung thanh ghi T2CON

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

Tbit	Tên	Mô tả
7	TF2	Timer 2 overflow Flag TF2 không được tác động khi RCLK hay TCLK = 1. TF2 phải được xóa bằng phần mềm và được đặt bằng phần cứng khi Timer tràn.
6	EXF2	Timer 2 External Flag được đặt khi EXF2 = 1 và xảy ra chế độ nạp lại hay giữ do có cạnh âm tại chân T2EX(P1.1) (chuyển tù 1 xuống 0). Khi EXF2 = 1 và cho phép ngắt tại timer 2 thì chương trình sẽ chuyển đến chương trình phục vụ ngắt của Timer 2. EXF2 phải được xóa bằng phần mềm.
5	RCLK	Receive Clock Bit (chỉ dùng cho port nối tiếp ở chế độ 1 và 3): RCLK = 0: dùng timer 1 làm xung clock thu cho port nối tiếp RCLK = 1: dùng timer 2 làm xung clock thu cho port nối tiếp
4	TCLK	Transmit Clock Bit Giống như RCLK nhưng dùng cho xung clock phát

3	EXEN2	Timer 2 External Enable Bit: = 0: bỏ qua tác động tại chân T2EX (P1.1) = 1: xảy ra chế độ nạp lại hay giữ do có cạnh âm tại chân T2EX(P1.1) (chuyển từ 1 xuống 0).
2	TR2	Timer 2 Run Control Bit = 0: cấm timer 2 = 1: chạy timer 2
1	C/T2	Timer 2 capture / Reload Bit Nếu RCLK = 1 hay TCLK = 1: bỏ qua Nếu RCLK = 0 và TCLK = 0: chọn chế độ giữ (= 1) hay nạp lại (= 0) khi xuất hiện xung âm tại T2EX (P1.1) và EXEN2 = 1
0	CP/RL2	Timer 2 Capture / Reload Bit Nếu RCLK = 1 hay TCLK = 1: bỏ qua Nếu RCLK = 0 và TCLK = 0: chọn chế độ giữ (= 1) hay nạp lại (= 0) khi xuất hiện xung âm tại T2EX (P1.1) và EXEN2 = 1.

Giá trị khi reset: T2CON = 00h, T2CON cho phép định vị bit

- Thanh ghi T2MOD:

Bảng 4.8 Nội dung thanh ghi T2MOD

-	-	-	-	-	-	T2OE	DCEN

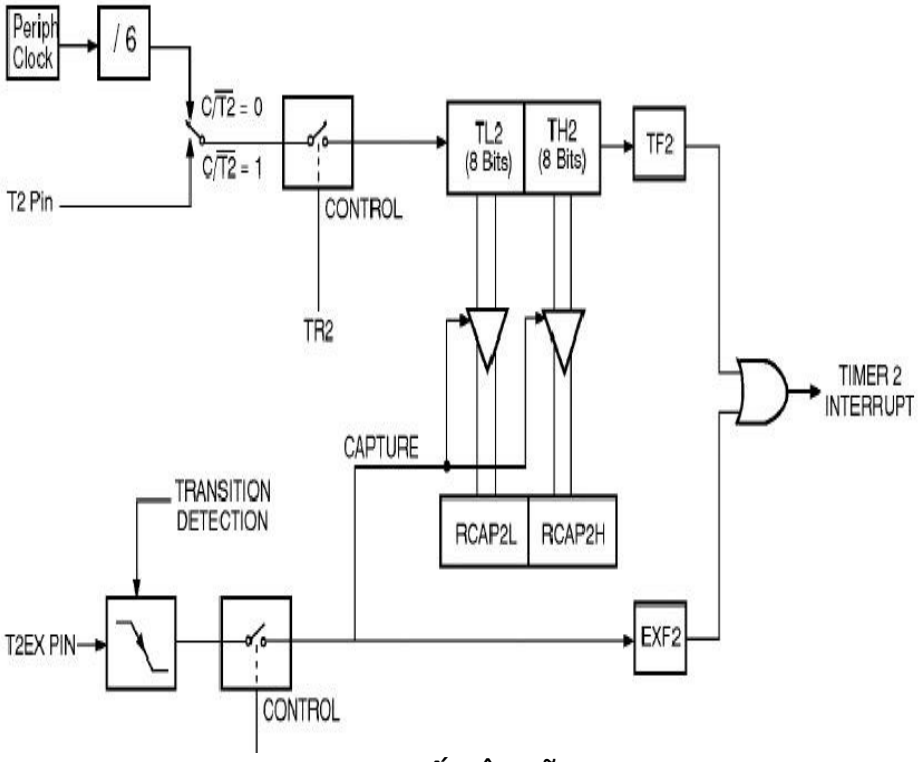
Bit	Tên	Mô tả
7	-	
6	-	
5	-	
4	-	
3	-	
2	-	

1	T2OE	Timer 2 Output Enable Bit = 0: T2 (P1.0) là ngõ vào clock hay I/O port = 1: T2 là ngõ ra clock
0	DCEN	Down Counter Enable Bit = 0: cấm timer 2 là bộ đếm lên / xuống = 1: cho phép timer 2 là bộ đếm lên / xuống

- Giá trị khi reset: T2MOD = xxxx xx00b, MOD không cho phép định vị bit.

- Các thanh ghi TH2, TL2, RCAP2H và RCAP2L không cho phép định vị bit và giá trị khi reset là 00h. Các chế độ hoạt động của Timer 2 mô tả trong phần sau.

7.2. *Chế độ capture.*



Hình 4.3. Chế độ giữ Timer 2

Chế độ giữ của Timer 2 có 2 trường hợp xảy ra:

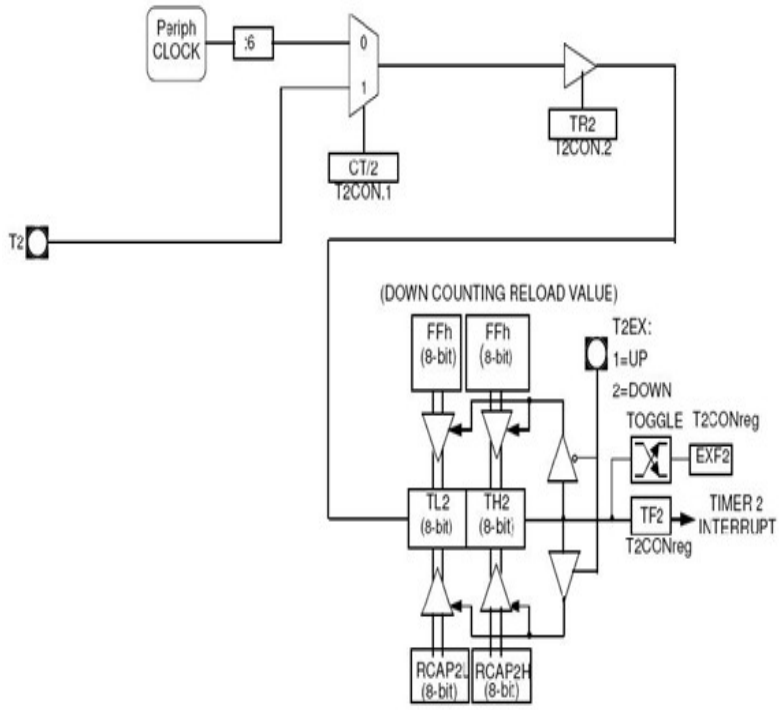
- Nếu EXEN2 = 0: Timer 2 hoạt động giống như Timer 0 và 1, nghĩa là khi giá trị đếm tràn (TH2_TL2 thay đổi từ FFFFh đến 00) thì cờ tràn TF2 được đặ lên mức 1 và tạo ngắt Timer 2(nếu cho phép ngắt).

- Nếu EXEN2= 1: vẫn hoạt động như trên nhưng them một tính chất nữa là: khi xuất hiện cạnh âm tại chân T2EX(P1.1), giá trị hiện tại của TH2 và TL2 được chuyển vòa cặp thanh ghi RCAP2H, RCAP2L (quá trinh2giu74 (capture) xảy ra); đồng thời, bit EXF2 = 1 (sẽ tạo ngắt nếu cho phép ngắt tại Timer 2) (hình 4.3).

7.3. *Chế độ tự động nạp lại.*

Chế độ tự động nạp lại cũng có 2 trường hợp giống như chế độ giữ:

- Nếu EXEN2 = 0: khi Timer tràn, cờ tràn TF2 đượcđặt lên 1 và nạp lại giá trị cho TH2, TL2 (từ cặp thanh ghi RCAP2H, RCAP2L)đồng thời tạo ngắt tại timer 2 nếu cho phép ngắt.
- Nếu EXEN2 = 1: hoạt động giống như trên nhưng khi có xung âm tại chân T2EX thì cũng nạp lại giá trị cho TH2, TL2 và đặt cờ EXF2 lên 1.
- Chế độ tự động nạp lại cũng cho phép thực hiện đếm lên hay xuống (điều khiển bằng bit DCEN trong thanh ghi T2MOD). Khi DCEN được đặt lên 1 và chân T2EX ở mức cao thì timer 2 sẽ đếm lên; còn nếu T2EX ở mức thấp thì timer 2 đếm xuống.
- Khi đếm lên, timer tràn tại giá trị đếm 0FFFFh. Khi tràn, cờ TF2 được đặt lên mức 1 và giá trị trong cặp thanh ghi RCAP2H, RCAP2L chuyển vào căp thanh ghi TH2, TL2.
- Khi đếm xuống, timer tràn khi giá trị trong cặp thanh ghi TH2, TL2 bằng giá trị trong cặp thanh ghi RCAP2H, RCAP2L. Khi tràn, cờ TF2 được đặt lên 1 và giá trị 0FFFFh được nạp vào cặp thanh ghi TH2, TL2.
- Trong chế độ này, khi timer tràn, giá trị trong cờ EXF2 sẽ chuyển mức và không tạo ngắt (có thể dùng thêm EXF2 để tạo giá trị đếm 17 bit).



Hình 4.4. Chế độ tự động lặp lại

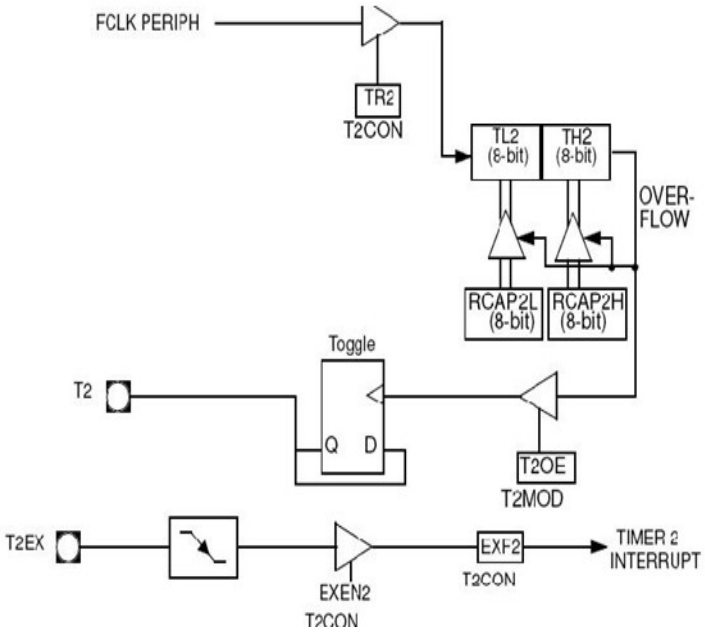
7.4. *Chế độ tạo xung clock.*

- Trong chế độ này, timer tạo ra một xung clock có chu kỳ bổn phận (duty cycle) 50%. Khi timer tràn, nội dung của thanh ghi CAP2H, RCAP2L

được nạp vào cặp thanh ghi TH2, TL2 và timer tiếp tục đếm. Tần số xung clock tại chân T2 được xác định theo công thức sau:

$$f = \frac{f_{osc} x 2^{X2}}{2\left(65536 - \frac{RCAP2H}{RCAP2L}\right)}$$

- X2: bit nằm trong thanh ghi CKCON. Trong chế độ X2: $f_{OSC} = f_{th}$ạch anh ngược lại thì $f_{OSC} = f_{th}$ạch anh/2.
- Để timer 2 hoạt động ở chế độ tạo xung clock, cần thực hiện các bước sau:
  + Đặt bit T2OE trong thanh ghi T2MOD = 1.
  + Xoá bit C/ T 2 trong thanh ghi T2CON = 0 (do chế độ này không cho phép đếm bằng dao động ngoài mà chỉ đếm bằng dao động nội).
  + Xác định giá trị của cặp thanh ghi RCAP2H và RCAP2L theo tần số xung clock cần tạo.
  + Khởi động giá trị cho cặp thanh ghi TH2, TL2 (có thể không cần thiết tuỳ theo ứng dụng).
  + Đặt bit TR2 trong thanh ghi T2CON = 1 để cho phép timer chạy.
  + Đặt bit TR2 trong thanh ghi T2CON = 1 để cho phép timer chạy.



Hình 4.5. Chế độ tạo xung clock

7.5 *Chế độ tạo tốc độ baud.*

Khi các bit TCLK và RCLK trong thanh ghi T2CON được đặt lên mức 1, timer 2 sẽ dùng để tạo tốc độ baud cho cổng nối tiếp. Chế độ này cùng hoạt động như timer 0 và timer 1 (giống như cổng nối tiếp).

CÁC BÀI TẬP MỞ RỘNG, NÂNG CAO VÀ GIẢI QUYẾT VẤN ĐỀ

Bai 1: Giả sử thanh ghi A chưá gia´trị %AH. Gia´trị của A se˜bằng bao nhiêu sau khi lệnh XRL A,#0FFH được thực hiện?

Gợi y´: A0H

Bai 2: Nêú PSW chưá gia´trị 0C0H va`A chưá gia´trị 50H, sau khi lệnh RLC A thực hiện thanh chưá se˜co´gia´trị la`bao nhiêu?

Gợi y´: A1H

Bài 3: Giả sử 8051 hoạt động với thạnh anh 12 MHz, viết đoạn chương trình tạo ra xung vuông83.3 kHz trên cổng P1.7.

Gợi y´:

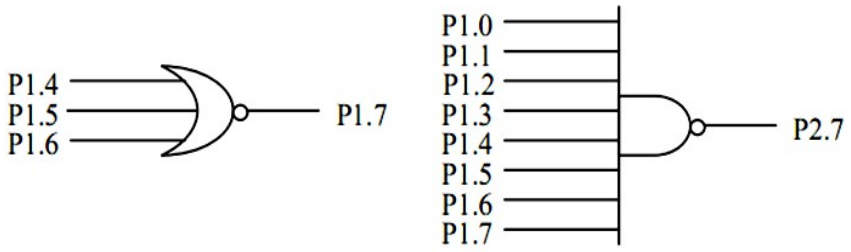
Cần sử dụng timer tạo ra khoảng thơì gian trê˜bằng ½ chu ky`xung vuông, sau đo´lật trạng thaí ngo˜ra P1.7 sau môĩ lần gọi trê˜

Bài 4: Viết chương trình tạo ra xung tác động mức cao trong 4 µsec trên cổng P1.7 sau mỗi 200 µsec.

Gợi y´:

Cần tạo ra hai chương trình trê˜4µsec va`200µsec, cư´bật trạng thaí P1.7 lên sau 4µsec lại xoá vê`200µsec.

Bài 5: Viết các đoạn chương trình thực hiện các hàm logic như trên hình vẽ sau:



Gợi y´: Đoạn chương trình thực hiện trên các hàm logic sau:

Thực hiện cać lệnh OR va`NOT cho hình a va`NAND rôi NOT cho hình b.

Bài 6: Xác định giá trị của thanh chứa sau khi các lệnh sau được thực hiện:

```
MOV A,#7FH
MOV 50H,#29H
MOV R0,#50H
XCHD A,@R0
```

Gợi y´: Giá trị của thanh chứa: A=29H

Bài 7: Viết chương trình phaí xung vuông 12KHz ở ngo˜ra P1.2 sử dụng Timer0.

Gợi ý: Chương trình sử dụng timer 0 chế độ định thơi cho phep tư`bên trong tạo ra khoảng thơi gian trê~ băng ½ chu ky`xung vuông 12KHz, lật trạng thaí P1.2 sau môĩ lân gọi chương trình trê~

Bai 8: Nêu cac kiểu định địa chỉ thanh ghi?

Gợi ý: Trình bay chức năng va`giải thích 8 kiểu định địa chỉ như sau:

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

Chúý: Môĩ kiểu định địa chỉ thanh ghi phải cómột vídụ.

Bai 9: Trình bay cac nhom lệnh được dung trong ngôn ngư~ lập trình(amsemly)?

Gợi ý:

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

- Nhom lệnh sốhọc
- Nhom lệnh logic
- Nhom lệnh vận truyên dữ liệu
- Nhom lệnh Boolean (thao tac bit)
- Nhom lệnh rẽ nhánh chương trình

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 hạng của lệnh, tùy thuộc vào mỗi lệnh số toán hạng có thể không có, có 1, 2 hoặc 3.

Chúý: Môĩ nhom lệnh phải nêu tom tăt́ cac lệnh trong nhom va`cach su dung chung.

❖ Điêù kiện thực hiện bai học

+ Vật liệu:

- Vi điều khiển.
- Vi mạch số các loại.

- Điện trở.
- Tụ.
- Rơ le.
- Led các loại.
- Mạch in, Dây nối, Chì hàn.

+ Dụng cụ, Trang thiết bị:

- Sơ đồ, IC họ 8051.
- Panel chân cắm nhỏ.
- Tài liệu hướng dẫn sử dụng họ 8051.
- Panel chân cắm các linh kiện điện tử IC CMOS – TTL.
- Sơ đồ mạch.
- IC họ 8051 - CMOS, TTL – 555.
- Led 7 đoạn.
- Sơ đồ- các bài tập ứng dụng trên kit thực hành.
- Phần mềm chương trình Assembler.
- Kít thực hành và iC họ TTL – CMOS.
- Dụng cụ tháo, ráp vi mạch.
- Kit thực tập và mô hình kèm theo.

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức:

- Phân biệt được các kiểu định địa chỉ và dữ liệu.
- Trình bày được đặc tính và công dụng của từng lệnh trong 8051.
- Xác định được độ lớn và thời gian thực hiện chương trình.

+ Về kỹ năng:

- Lắp ráp các mạch ứng dụng từng phần do giáo viên đề ra.
- Thực hiện viết các chương trình theo yêu cầu cho trướoc

+ Thái độ: Đánh giá phong cách, thái độ học tập

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm

+ Về kỹ năng: Đánh giá kỹ năng thực hành Mỗi sinh viên, hoặc mỗi nhóm học viên thực hiện công việc theo yêu cầu của giáo viên.Tiêu chí đánh giá theo các nội dung:

- Độ chính xác của công việc
- Tính thẩm mỹ của mạch điện
- Độ an toàn trên mạch điện
- Thời gian thực hiện công việc
- Độ chính xác theo yêu cầu kỹ thuật

+ Thái độ: Tỉ mỉ, cẩn thận, chính xác.

BAI 5

CỔNG NÔI TIÊP (SERIAL PORT)

Mã bài: MĐ24-05

Giới thiệu:

Để truyền thông với máy tính, hoặc với một vi điều khiển khác, trong kết nối này dữ liệu được chuyển 1 bit trong một đơn vị thời gian. Truyền thông nối tiếp chỉ yêu cầu một đường dây cho dữ liệu, đường thứ hai cho nối đất và có thể là đường thứ ba cho xung đồng hồ để đồng bộ. Đồng bộ và không đồng bộ là hai phương thức khác nhau trong việc truyền dữ liệu. Trong truyền đồng bộ thì bộ truyền và bộ thu được đồng bộ hóa qua một đồng hồ bên ngoài, trong khi với truyền không đồng bộ thì bộ truyền và bộ thu được đồng bộ hóa bởi một tín hiệu đặc biệt trên bộ truyền trung gian.

Mục tiêu của bài:

- Trình bày được cấu tạo và các chế độ làm việc của cổng truyền thông nối tiếp theo nội dung đã học.
- Thực hiện cổng truyền thông nối tiếp đúng yêu cầu kỹ thuật.
- Thực hiện thu phát dữ liệu nối tiếp bằng 8051 đạt yêu cầu kỹ thuật.

Nội dung chính:

1.Mở đâu.

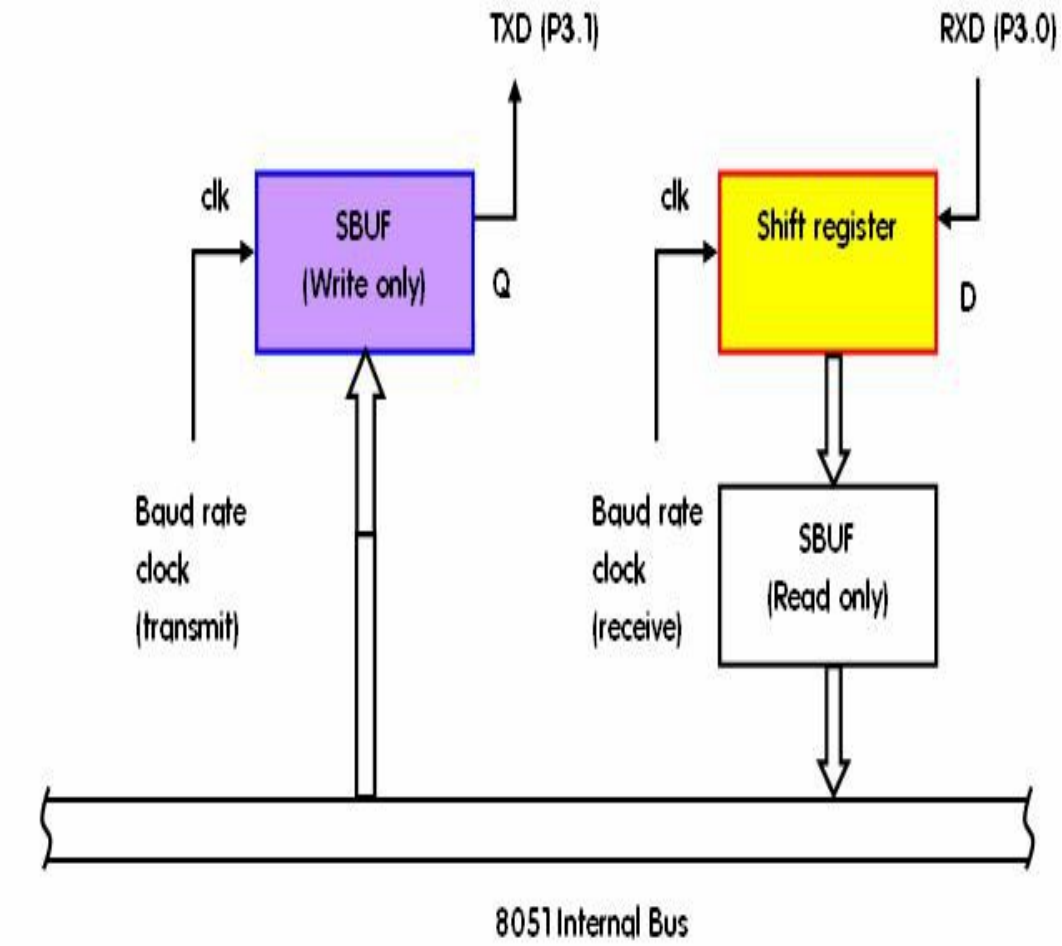
Mục tiêu: Trình bày được cấu tạo và các chế độ làm việc của cổng truyền thông nối tiếp.

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 đơi 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).

- Port nối tiếp hoạt động song công (full duplex), nghĩa là có khả năng thu và phát đồng thời.
- Sử dụng 2 thanh ghi chức năng đặc biệt SBUF (địa chỉ byte là 99H) và SCON (địa chỉ byte là 98H) để truy xuất port nối tiếp).
- Việc ghi lên SBUF sẽ nạp dữ liệu để phát, và việc đọc SBUF

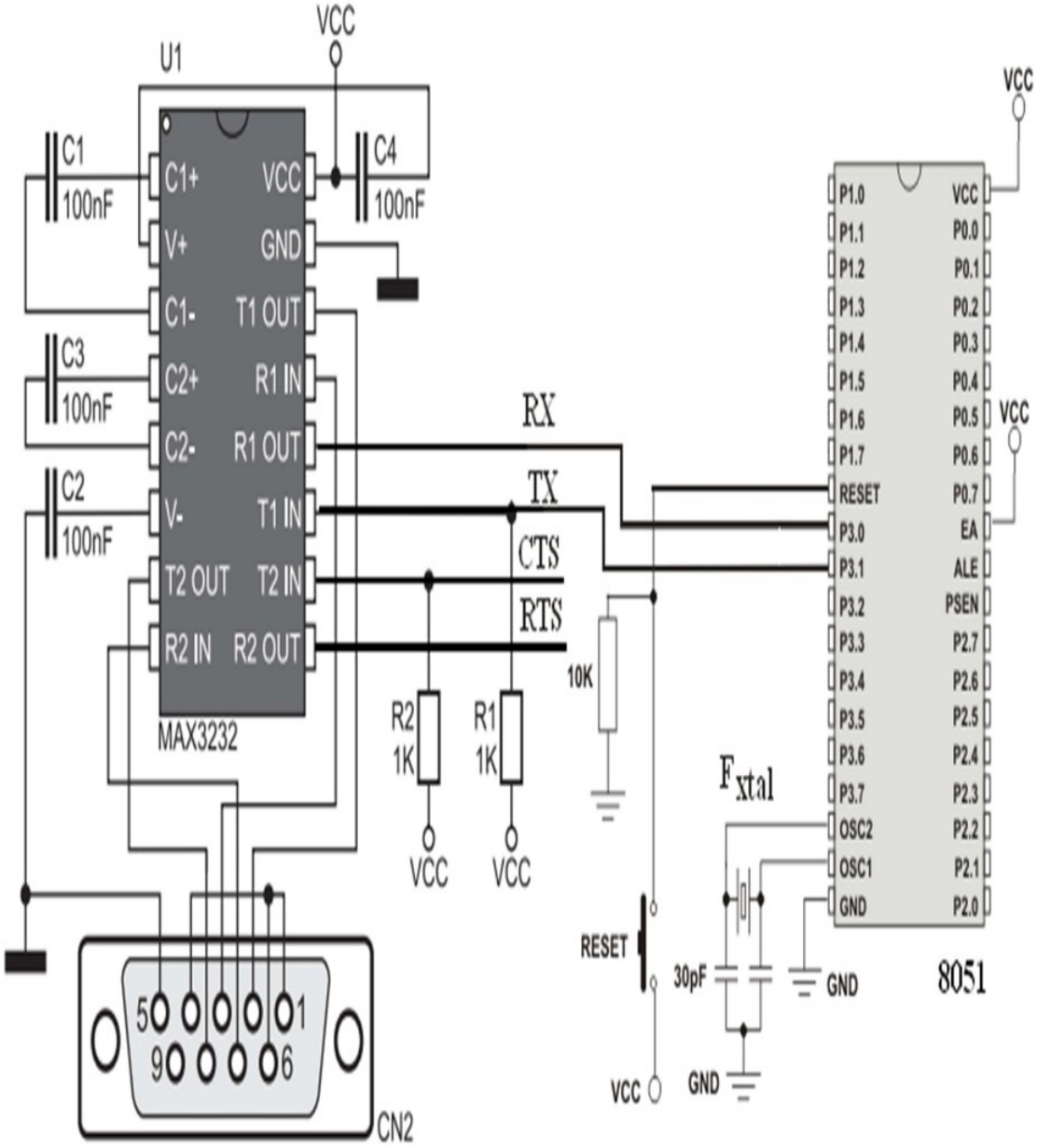
sẽ truy xuất dữ liệu đã nhận được thực ra có 2 SBUF riêng rẽ.

- SCON chứa các bit trạng thái và điều khiển, thanh này được định địa chỉ bit.
- Tần số hoạt động của port nối tiếp hay còn gọi là tốc độ baud (baud rate) có thể cố định hoặc thay đổi.
- Cổng nôí tiêp trong 8051 co´khả năng hoạt động ở chê´độ đôìng bộ va`bât đôìng bộ duìng 2 chân TxD (P3.1) va`RxD (P3.0). Chưć năng của port nôí tiêp la`thực hiện chuyển đổi song song sang nôí tiêp đôí vơí dư liệu xuât, va`chuyển đổi nôí tiêp sang song song đôí vơí dư liệu nhập.
- Khi hoạt động ở chê´độ truyêǹ/nhận bât đôìng bộ (UART – Universal Asynchronous Receiver / Transmiter), cổng nôí tiêp co´3 chê´ độ song công (1, 2 va`3). Qua´trình đọc/ghi cổng nôí tiêp duìng thanh ghi SBUF(Serial Buffer), thực chât la`2 thanh ghi khać nhau: một thanh ghi truyêǹ va`một thanh ghi nhận.
- 8051 có 1 cổng UART làm việc ở chuẩn TTL, mặc định sau khi khởi động tất các cổng của 8051 dều 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 5.1)



Hình 5.1. Ghép nối RS232 với 8051

- Hai thanh ghi chức năng đặc biệt phục vụ cho truyền dữ liệu là thanh ghi đệm SBUF và thanh ghi điều khiển SCON. Thanh ghi đệm SBUF nằm ở địa chỉ 99H có 2 chức năng: nếu vi điều khiển ghi dữ liệu lên thanh ghi sbuf thì dữ liệu đó sẽ được truyền đi, nếu hệ thống khác gởi dữ liệu đến thì sẽ được lưu vào thanh ghi đệm sbuf (hình 5.2)



Hình 5.2. Sơ đồ của khối truyền dữ liệu nối tiếp.

- Thanh ghi điều khiển truyền dữ liệu SCON nằm ở địa chỉ 98H là thanh ghi cho phép truy suất bit bao gồm các bit trạng thái và các bit điều khiển. Các bit điều khiển dùng để thiết lập nhiều kiểu hoạt động truyền dữ liệu khác nhau, còn các bit trạng thái cho biết thời điểm kết thúc khi truyền xong một kí tự hoặc nhận xong một kí tự. Các bit trạng thái có thể được kiểm tra trong chương trình hoặc có thể lập trình để sinh ra ngắt.
- Tần số hoạt động của truyền dữ liệu nối tiếp còn gọi tốc độ BAUD (số lượng bit dữ liệu được truyền đi trong một giây) có thể hoạt động cố định (sử dụng dao động trên chip) hoặc có thể thay đổi. Khi cần tốc độ Baud thay đổi thì phải sử dụng Timer 1 để tạo tốc độ baud.

2. Thanh ghi điều khiển.

Mục tiêu: Hiểu chức năng của các thanh ghi.

2.1. Thanh ghi SCON (Serial port controller).

Nội dung thanh ghi SCON

FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit	Ký hiệu	Địa chỉ	Mô tả
SCON.7	FE		Framing Error – kiểm tra lỗi khung Được đặt lên 1 khi phát hiện lỗi tại bit stop và phải xóa bằng phần mềm. Bit FE chỉ truy xuất
	SM0		Serial port Mode bit 0 – Xác định chế độ cho cổng nối tiếp
SCON.6	SM1	9Eh	Serial port Mode bit 1

SM0	SM1	Mô tả	Tốc độ baud
0	0	Thanh ghi dịch	$f_{OSC}/12$
0	1	UART 8 bit	Thay đổi
1	0	UART 9 bit	$f_{OSC}/32$ hay f /64

			1	1	UART 9 bit	Thay đổi
SCON.5	SM2	9Dh	Serial port Mode bit 2 – Chế độ đa xử lý = 0: bình thường			
SCON.4	REN	9Ch	Reception Enable bit – Cho phép thu = 0: cấm thu			
SCON.3	TB8	9Bh	Transmitter Bit – Bit truyền thứ 9 trong chế độ 2 và 3.			
SCON.2	RB8	9Ah	Receiver Bit – Bit nhận thứ 9 trong chế độ 2 và 3. Trong chế độ 1, nếu SM2 = 0 thì RB8 = stop			
SCON.1	TI	99h	Transmit Interrupt flag – Cờ ngắt phát Được đặt bằng 1 khi kết thúc quá trình truyền và			
SCON.0	RI	99h	Receive Interrupt flag – Cờ ngắt thu Được đặt bằng 1 khi nhận xong dữ liệu và xóa			

Giá trị khi reset: 00h, cho phep định địa chỉ bit

2.2. Thanh ghi BDRCON (Baud Rate Control Register).

Bảng 5.1. Nội dung thanh ghi BDRCON

-	-	-	BRR	TBCK	RBC	SPD	SRC

Bit	Ký hiệu	Mô tả

7	-	
6	-	
5	-	
4	BRR	Baud Rate Run control bit – Cho phép hoạt động = 0: cấm bộ tạo tốc độ baud nội (internal baud rate generator) hoạt động. = 1: cho phép bộ tạo tốc độ baud nội (internal baud rate
3	TBCK	Transmission Baud rate generator selection bit for UART – Chọn bộ tạo tốc độ baud truyền là bộ tạo tốc độ nội (= 1) hay bằng timer (= 0)
2	RBCK	Reception Baud rate generator selection bit for UART – Chọn bộ tạo tốc độ baud nhận là bộ tạo tốc độ nội (= 1) hay bằng timer (= 0)
1	SPD	Baud Rate Speed control bit for UART – Chọn tốc độ baud là nhanh (= 1) hay chậm (= 0)
0	SRC	Baud Rate Source select bit in Mode 0 for UART – Chọn tốc độ baud trong chế độ 0 từ dao động thạch anh (= 0) hay từ bộ tạo tốc độ baud nội (= 1)

Giá trị khi reset: 00h, không cho phép định địa chỉ bit

- Ngoài ra còncó các thanh ghi SBUF (Serial Buffer), BRL (Baud Rate Reload), SADEN (Slave Address Mark), SADDR (Slave Address).

❖ Lưu ý rằng các thanh ghi BDRCON, BRL, SADEN và SADDR chỉ có trong các phiên bản mới của MCS-51.

3. Chế độ làm việc.

Mục tiêu: Hiểu các chế độ làm việc của cổng nối tiếp.

Port nối tiếp của 8051 co´4 chế độ hoạt động, cać chê´độ được chọn băng cách ghi 1 hoặc 0 cho cać bit SM0 va`SM1 trong thanh ghi SCON. Trước khi truyền dữ liệu thì thanh ghi SCON phải được khởi tạo đúng kiểu. Ba trong sô´cać chê´độ hoạt động cho phep truyên không đông bộ (asynchronous), trong đo´môĩ ky´tự được thu hoặc được phaí se˜cuìng vơí một bit start va`một bit stop tạo thanh một khung (frame).

Ví dụ: để khởi tạo truyền dữ liệu kiểu 1 thì 2 bit: SM0 SM1 = 01, bit cho phép thu: REN =1, và cờ ngắt truyền TI = 1 để sẵn sàng truyền, ta dùng lệnh sau :

MOV SCON, #01010010b.

Truyền dữ liệu nối tiếp của MCS51 có 4 kiểu hoạt động tùy thuộc theo 4 trạng thái của 2 bit SM0, SM1 được liệt kê như sau:

SM0	SM1	Chế độ	Mô tả	Tốc độ baud
0	0	0	Thanh ghi dịch	Cố định (tần số dao động/12)
0	1	1	UART 8 bit	Thay đổi (thiết lập bởi bộ định thời)
1	0	2	UART 9 bit	Cố định (tần số dao động/12 hoặc /64)
1	1	3	UART 9 bit	Thay đổi (thiết lập bởi bộ định thời)

- Thanh ghi scon sẽ thiết lập các kiểu hoạt động truyền dữ liệu khác nhau cho MCS51. Cấu trúc của thanh ghi Scon như sau (bảng 5.2):

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit	Ký hiệu	Địa chỉ	Mô tả hoạt động
7	SM0	9FH	Bit chọn kiểu truyền nối tiếp: bit thứ 0.
6	SM1	9EH	Bit chọn kiểu truyền nối tiếp: bit thứ 1.
5	SM2	9DH	Bit cho phép truyền kết nối nhiều vi xử lý ở mode 2 và 3; RI sẽ không tích cực nếu bit thứ 9 đã thu vào là 0.
4	REN	9CH	Bit cho phép nhận kí tự, REN = 1 sẽ cho phép nhận kí tự.
3	TB8	9BH	Dùng để lưu bit thứ 9 để truyền đi khi hoạt động ở mode 2 và 3, TB8 bằng 0 hay là do người lập trình thiết lập.
2	RB8	9AH	Dùng để lưu bit 9 nhận về khi hoạt động ở mode 2 và 3.

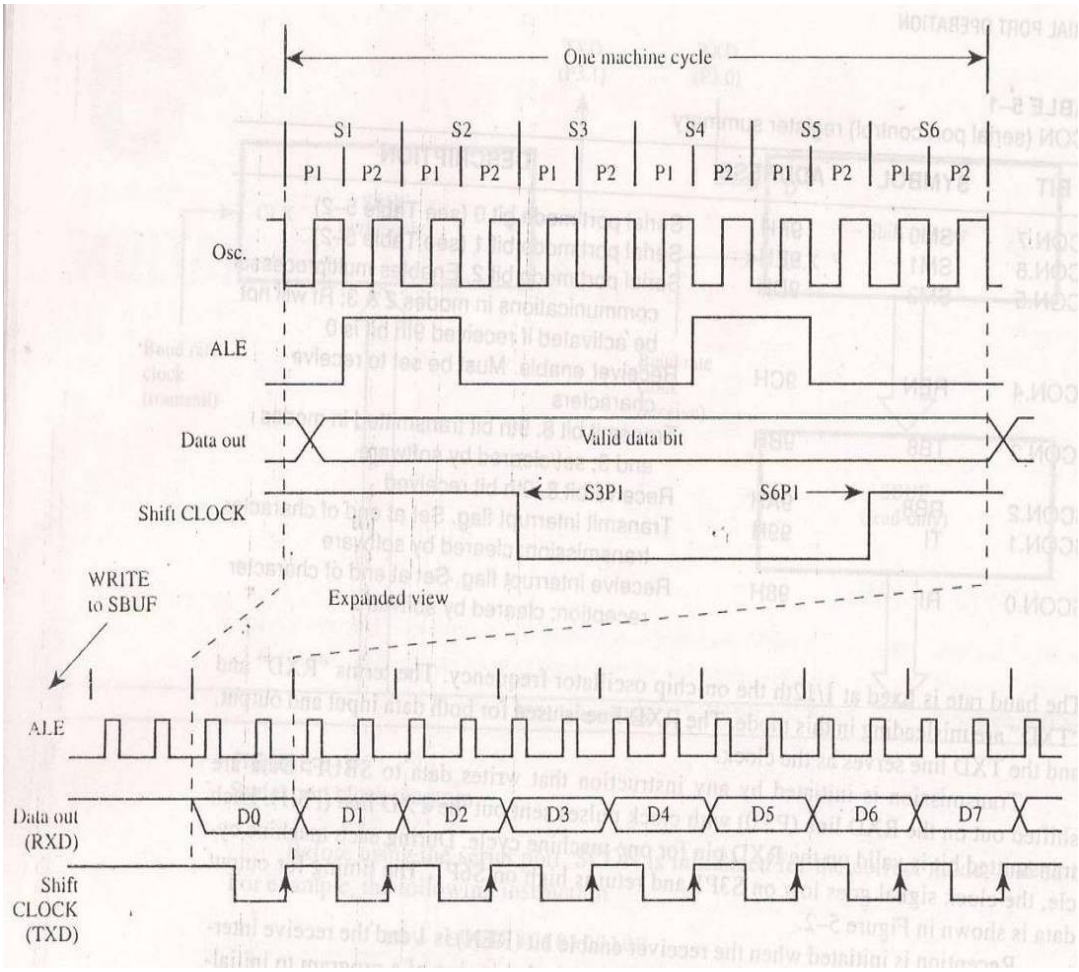
1	TI	99H	Cờ báo hiệu này lên mức 1 khi truyền xong 1 kí tự và xóa bởi người lập trình để sẵn sàng truyền kí tự tiếp theo.
0	RI	98H	Cờ báo hiệu này lên mức 1 khi nhận xong 1 kí tự và xóa bởi người lập trình để sẵn sàng nhận kí tự dữ liệu tiếp theo.

Bảng 5.2. Các bit trong thanh ghi điều khiển truyền dữ liệu.

3.1. Thanh ghi dịch 8 bit (chế độ 0).

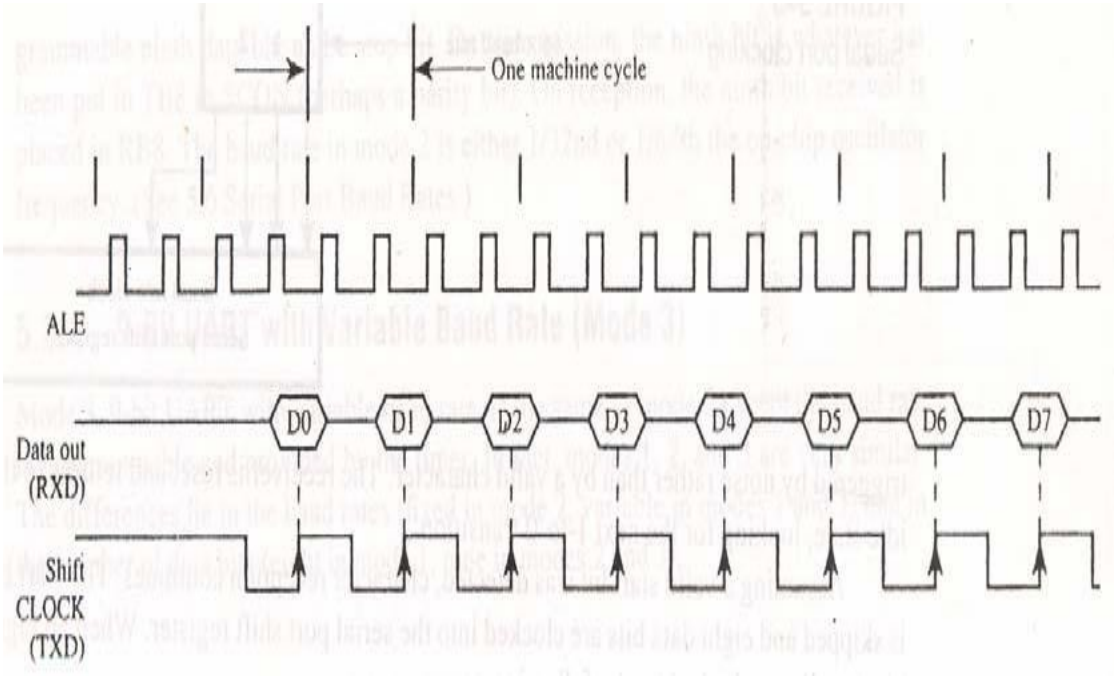
Để định cấu hình cho truyền dữ liệu nối tiếp ở kiểu 0 thì 2 bit SM1 SM0 = 00. Dữ liệu nối tiếp nhận vào và dữ liệu truyền đi đều thông qua chân RxD, còn chân TxD thì dùng để dịch chuyển xung clock. 8 bit dữ liệu để truyền đi hoặc nhận về thì luôn bắt đầu với bit có trọng số nhỏ nhất LSB. Tốc độ Baud được thiết lập cố định ở tần số bằng 1^{12} tần số dao động thạch anh trên Chip.

Khi thực hiện lệnh ghi dữ liệu lên thanh ghi sbuf thì quá trình truyền dữ liệu bắt đầu. Dữ liệu được dịch ra ngoài thông qua chân RxD cùng với các xung nhịp cũng được gởi ra ngoài thông qua chân TxD. Mỗi bit truyền đi chỉ có xuất hiện trên chân RxD trong khoảng thời gian một chu kỳ máy. Trong khoảng thời gian của mỗi chu kỳ máy, tín hiệu xung clock xuống mức thấp tại thời điểm S3P1 và lên mức cao tại thời điểm S6P1 trong giản đồ thời gian hình 5.3.



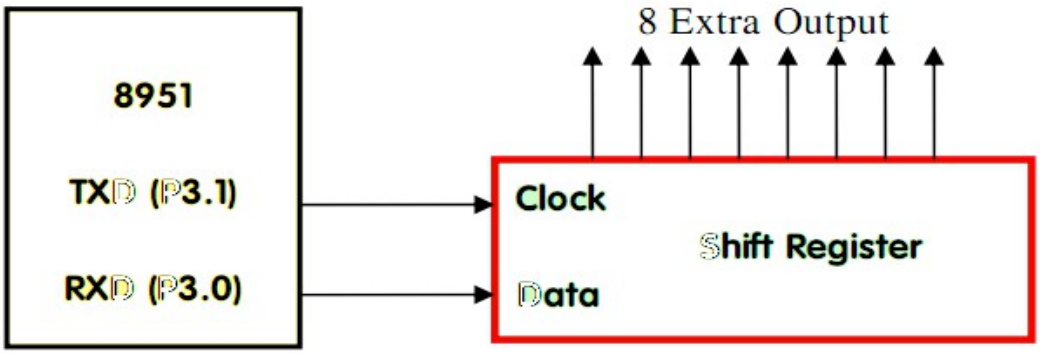
Hình 5.3. Giản đồ thời gian.

Biểu đồ thời gian của dữ liệu nối tiếp truyền vào vi điều khiển ở kiểu 0 như sau (hình 5.4):



Hình 5.4. Biểu đồ thời gian truyền dữ liệu mod 0

- Quá trình nhận được khởi động khi bit cho phép nhận REN = 1 và cờ nhận RI = 0. Nguyên tắc chung là khởi tạo bit REN = 1 ở đầu chương trình để khởi động truyền dữ liệu, và xóa bit RI để sẵn sàng nhận dữ liệu vào. Khi bit RI bị xóa, các xung clock sẽ xuất ra bên ngoài thông qua chân TxD, bắt đầu chu kỳ máy kế tiếp thì dữ liệu từ bên ngoài sẽ được dịch vào bên trong thông qua chân RxD.
- Một ứng dụng cụ thể sử dụng mode 0 là dùng để mở rộng thêm số lượng ngõ ra cho MCS51 với cách thức thực hiện như sau: một thanh ghi dịch từ nối tiếp thành song song được nối đến các đường TxD và RxD của MCS51 để mở rộng thêm 8 đường ra như hình 5.5. Nếu dùng thêm nhiều thanh ghi dịch mắc nối tiếp vào thanh ghi dịch đầu tiên sẽ mở rộng được nhiều ngõ ra.



Hình 5.5 Một ứng dụng kiểu 0 để tăng thêm ngõ ra bằng thanh ghi dịch.

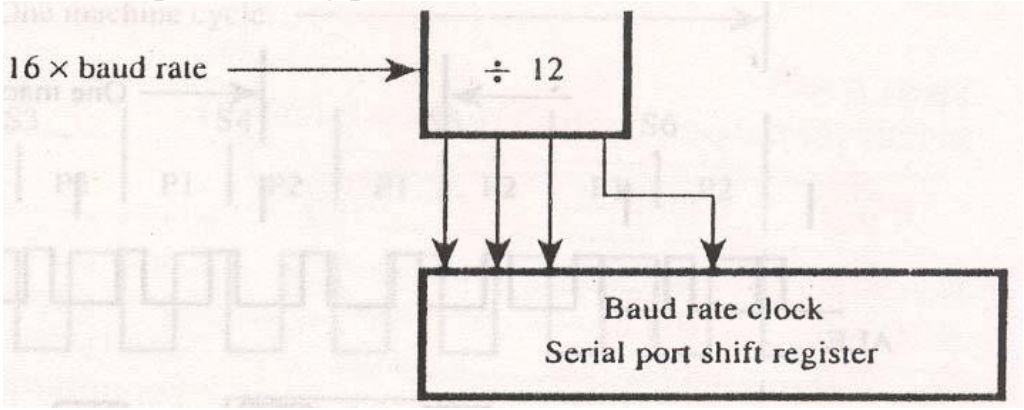
3.2. Chế độ UART 8 bit có tốc độ baud thay đổi (chế độ 1).

- Trong mode này, truyền dữ liệu nối tiếp hoạt động bất đồng bộ

UART 8 bit có tốc độ Baud thay đổi được. UART là bộ thu và phát dữ liệu nối tiếp với mỗi ký tự dữ liệu luôn bắt đầu bằng 1 bit Start (ở mức 0) và kết thúc bằng 1 bit Stop (ở mức 1), bit parity đôi khi được ghép vào giữa bit dữ liệu sau cùng và bit Stop.

- Trong kiểu này, 10 bit dữ liệu sẽ phát đi ở chân TxD và nếu nhận thì sẽ nhận ở chân RxD. 10 bit đó bao gồm: 1 bit start, 8 bit data (LSB là bit đầu tiên), và 1 bit stop. Đối với hoạt động nhận dữ liệu thì bit Stop được đưa vào bit RB8 trong thanh ghi SCON.
- Trong MCS51, tốc độ Baud được thiết lập bởi tốc độ tràn của Timer T1. Đối với họ 52 có 3 timer thì tốc độ baud có thể thiết lập bởi tốc độ tràn của timer T1 hoặc timer T2 hoặc cả 2 timer T1 và T2: một timer cho máy phát và 1 timer cho máy thu.

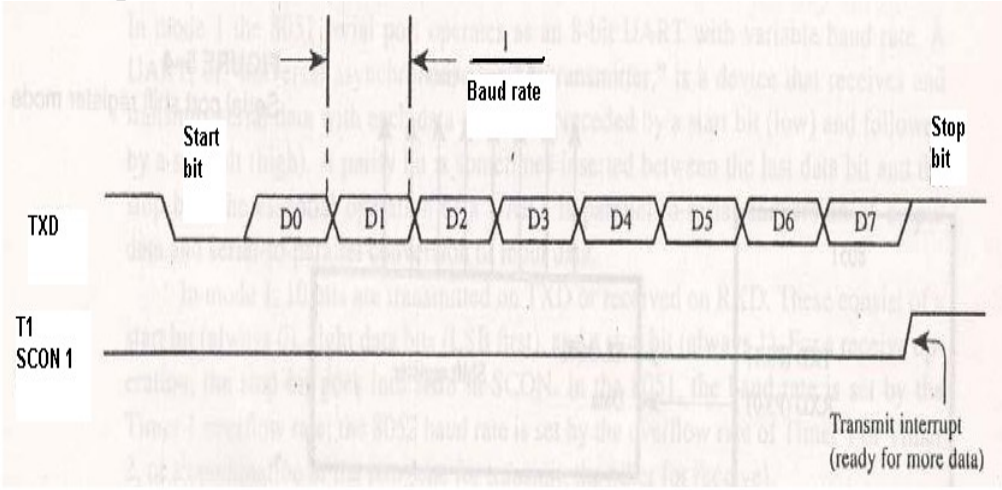
Nguồn cung cấp xung clock để đồng bộ các thanh ghi truyền dữ liệu nối tiếp hoạt động ở kiểu 1, 2, 3 được thiết lập bởi bộ đếm 16 như hình 5.6, ngõ ra của bộ đếm là xung clock tạo tốc độ baud. Xung ngõ vào của bộ đếm có thể lập trình bằng phần mềm.



Hình 5.6. Cung cấp xung cho truyền dữ liệu nối tiếp.

- Khi có một lệnh ghi dữ liệu lên thanh ghi sbuf thì quá trình truyền dữ liệu bắt đầu nhưng nó chưa truyền mà chờ cho đến khi bộ chia 16 (cung cấp tốc độ Baud cho truyền dữ liệu nối tiếp) bị tràn. Dữ liệu được xuất ra trên chân TxD bắt đầu với bit start theo sau là 8 bit data và sau cùng là bit stop. Các cờ phát TI được nâng lên mức 1 cùng lúc với thời điểm xuất hiện bit Stop trên chân TxD như hình 5.7.
- Quá trình nhận dữ liệu được khởi động khi có sự chuyển đổi từ mức 1 sang mức 0 ở ngõ vào RxD. Bộ đếm 4 bit được reset ngay lập tức để sắp xếp bit dữ liệu đang đến từ ngõ vào RxD. Mỗi bit dữ liệu đến được lấy mẫu ở trạng thái đếm thứ 8 trong một chu kỳ 16 trạng thái của bộ đếm 4 bit.
- Khi có sự chuyển trạng thái từ 1 xuống 0 ở ngõ vào RxD của bộ thu thì trạng thái 0 này phải tồn tại trong 8 trạng thái liên tục của bộ đếm 4

bit. Nếu trường hợp này không đúng thì bộ thu xem như bị tác động bởi tín hiệu nhiễu. Bộ thu sẽ reset và trở về trạng thái nghỉ và chờ sự chuyển trạng thái tiếp theo.



Hình 5.7. Cờ báo phát xong dữ liệu TI.

Giả sử việc kiểm tra bit Start là hợp lệ thì bit Start sẽ được bỏ qua và 8 bit data được nhận vào thanh ghi dịch nối tiếp.

- Khi tất cả 8 bit được ghi vào thanh ghi dịch thì 3 công việc sau sẽ được thực hiện tiếp theo:

Bit thứ 9 (bit Stop) được dịch vào bit RB8 trong SCON.

8 bit data được nạp vào thanh ghi SBUF.

Cờ ngắt nhận RI =1.

Tuy nhiên, 3 công việc trên chỉ xảy ra nếu hai điều kiện sau tồn tại:

RI = 0.

SM2 = 1 và bit Stop nhận được = 1 hoặc SM2 = 0.

3.3. Chế độ 2: UART 9 bit với tốc độ Baud cố định.

Khi SM1 SM0 = 10 thì truyền dữ liệu hoạt động ở kiểu 2 có tốc độ Baud cố định. Khi phát thìTruyền / nhận 11 bit: 1 bit start, 8 bit dữ liệu, bit thứ 9 và 1 bit stop. Khi truyền, bit 9 là bit TB8 và khi nhận, bit 9 là bit RB8 trong thanh ghi SCON. Tốc độ baud cố định là 1/32 hay 1/64 tần số dao động trên Chip.

3.4. Chế độ 3: UART 9 bit với tốc độ Baud thay đổi.

- Khi SM1 SM0 = 11 thì truyền dữ liệu hoạt động ở kiểu 3 là kiểu UART 9 bit có tốc độ Baud thay đổi. Kiểu 3 tương tự kiểu 2 ngoại trừ tốc độ Baud được lập trình và được cung cấp bởi Timer.
- Các kiểu 1, kiểu 2 và kiểu 3 rất giống nhau, những điểm khác nhau là ở tốc độ Baud (kiểu 2 cố định, kiểu 1 và kiểu 3 thay đổi) và số bit dữ liệu (kiểu 1 có 8 bit, kiểu 2 và kiểu 3 có 9 bit data).
- Trong 4 chế độ trên, thường sử dụng chế độ 1 hay 3 để truyền dữ

liệu. Trong trường hợp truyền dữ liệu giữa các vi điều khiển AT89C51 với nhau, có thể dùng chế độ 2. Ngoài ra, cổng nối tiếp còn có các chế độ nâng cao: kiểm tra lỗi khung và nhận dạng địa chỉ tự động.

4. Khởi tạo và truy xuất thanh ghi PORT nối tiếp.

Mục tiêu: Biết cách khởi tạo và truyền nhận dữ liệu của port nối tiếp.

4.1. Bit điều khiển cho phép nhận dữ liệu (Receive Enable).

Để cho phép thu dữ liệu thì chương trình phải làm cho bit REN = 1 và điều này được thực hiện ở đầu chương trình.

Ta có thể khởi tạo cho phép truyền dữ liệu bằng lệnh :

Setb ren hoặc MOV SCON, # Xxx1xxxxb

4.2. Bit dữ liệu thứ 9.

- Bit dữ liệu thứ 9 được phát trong kiểu 2 và kiểu 3 phải được nạp vào bit TB8 bằng phần mềm có nghĩa là người lập trình phải thực hiện công việc này trước khi truyền dữ liệu đi, còn bit dữ liệu thứ 9 của dữ liệu thu được thì tự động đặt vào trong bit RB8.
- Phần mềm có thể hoặc không đòi hỏi bit dữ liệu thứ 9 tham gia vào quá trình truyền dữ liệu tùy thuộc vào đặc tính của các thiết bị nối tiếp kết nối với nhau thiết lập ra qui định. Bit dữ liệu thứ 9 đóng 1 vai trị quan trọng trong truyền thông nhiều vi xử lý.

4.3. Thêm vào bit chẵn lẻ Parity

- Bit thứ 9 thường được dùng là bit kiểm tra chẵn lẻ. Ở mỗi chu kỳ máy, bit P trong thanh ghi trạng thái PSW bằng 1 hay bằng 0 tùy thuộc vào quá trình kiểm tra chẵn 8 bit dữ liệu chứa trong thanh ghi A.

Ví dụ 49: Nếu hệ thống truyền dữ liệu yêu cầu 8 bit data cộng thêm 1 bit kiểm tra chẵn, thì các lệnh sau đây sẽ phát 8 bit trong thanh ghi A cộng với bit kiểm tra chẵn được cộng vào bit thứ 9.

```
MOV   C,P          ;chuyển cờ chẵn lẻ P sang cờ C
MOV   TB8,C        ;chuyển cờ C sang bit TB8 để chuẩn bị truyền đi
MOV   SBUF,A       ;truyền dữ liệu 8 bit trong A và bit thứ 9 trong TB8

                                                    đi.
```

Nếu kiểm tra lẻ được yêu cầu thì các lệnh trên được sửa lại là:

```
MOV   C,P          ;chuyển cờ chẵn lẻ P sang cờ C
CPL   C            ;nghịch đảo chẵn thành lẻ
MOV   TB8,C        ;chuyển cờ C sang bit TB8 để chuẩn bị truyền đi
MOV   SBUF,A       ;truyền dữ liệu 8 bit trong A và bit thứ 9 trong TB8

                                                    đi
```

- Trong kiểu 1 ta vẫn có thể sử dụng bit kiểm tra chẵn lẻ như sau: 8 bit data được phát trong kiểu 1 có thể bao gồm 7 bit dữ liệu, và 1 bit kiểm tra chẵn lẻ. Để phát một mã ASCII 7 bit với 1 bit kiểm tra chẵn vào 8 bit, các lệnh sau đây được dùng:

```
MOV   C, P          ; Đưa Parity chẵn vào C
MOV   ACC.7, C      ; Đưa Parity chẵn vào bit MSB của A
MOV   SBUF, A       ; Gửi bit data cùng bit Parity chẵn
```

4.4. Các cờ ngắt.

- Cờ ngắt nhận RI và phát TI trong thanh ghi SCON đóng một vai trị quan trọng trong truyền dữ liệu của MCS-51. Cả hai bit đều được set bởi phần cứng nhưng phải xóa bởi phần mềm.

Điển hình là cờ RI được set ở mức 1 khi kết thúc quá trình nhận đầy đủ 1 kí tự và cho biết thanh ghi đệm thu đã đầy. Trạng thái của cờ RI có thể kiểm tra bằng phần mềm hoặc có thể lập trình để sinh ra ngắt. Nếu muốn nhận một ký tự từ một thiết bị đã được kết nối đến Port nối tiếp, thì chương trình phải chờ cho đến khi cờ RI = 1, sau đó xóa cờ RI và đọc ký tự từ thanh ghi SBUF.

Quá trình này được lập trình như sau :

```
WAIT :
      JNB   RI, WAIT  :Kiểm tra RI xem có bằng 1 hay không. Chờ nếu = 0
      CLR   RI       :khi cờ RI = 1 thì đã nhận xong dữ liệu và xóa cờ RI
      MOV A, SBUF    :đọc ký tự nhận được từ thanh ghi Sbuf
```

Cờ TI lên mức 1 cho biết đã phát xong ký tự và cho biết thanh ghi đệm sbuf đã rỗng. Nếu muốn gởi 1 ký tự đến một thiết bị đã được kết nối đến Port nối tiếp thì trước tiên phải kiểm tra xem Port nối tiếp đã sẵn sàng chưa. Nếu ký tự trước đang được gởi đi, thì phải chờ cho đến khi kết thúc quá trình gởi. Các lệnh sau đây dùng để phát một ký tự trong thanh ghi A ra :

```
WAIT :
      JNB   TI, WAIT    :Kiểm tra TI có bằng 1 hay không và chờ nếu
                          bằng 0.
       CLR  TI          :Xóa cờ ngắt thu TI
      MOV   SBUF,A      :Gởi nội dung trong thanh ghi A đi
Hoặc
        MOV   SBUF,A    :gởi nội dung trong thanh ghi A đi
WAIT :
      JNB     TI, WAIT :Kiểm tra TI có bằng 1 hay không và chờ nếu bằng
```

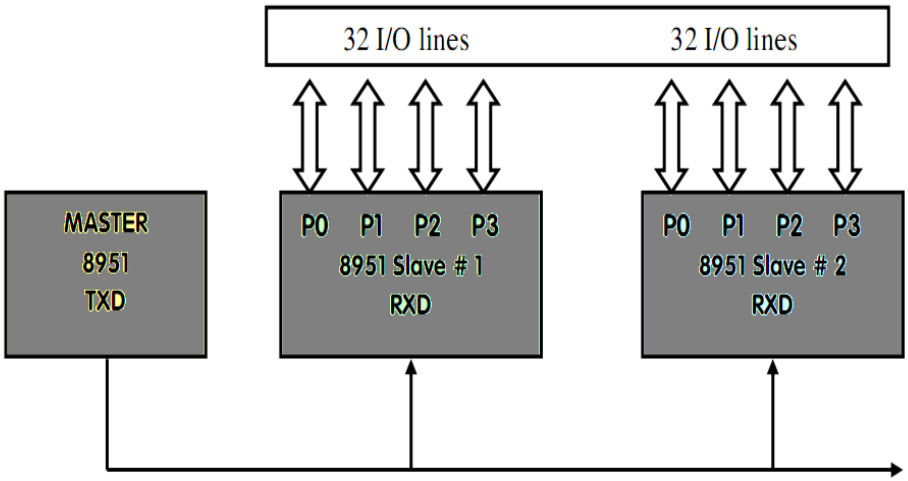
0.

CLR TI :Xóa cờ ngắt thu TI

5. Truyền thông đa xử lý (*Multiprocessor Communications*).

Mục tiêu: Hiểu được phương pháp truyền dữ liệu giữa Master với nhiều Slave.

- Chế độ 2 và 3 có một dự trù (chuẩn bị) đặc biệt cho có một chức năng đặc biệt cho việc truyền thông đa xử lý. Ở các mode 2 và 3, 9 bit dữ liệu được thu và bit thứ 9 được lưu vào bit RB8. Truyền dữ liệu có thể lập trình sao cho khi thu được bit Stop thì ngắt của truyền dữ liệu nối tiếp tác động chỉ khi bit RB8 =1. Cấu trúc này được phép bằng cách set bit SM2 = 1 trong thanh ghi SCON. Kiểu này được ứng dụng trong mạng sử dụng nhiều MCS51 được tổ chức theo cấu hình máy chủ và máy tớ như hình 5.8



Hình 5.8. Kết nối nhiều vi điều khiển

- Trong cấu hình kết nối ở trên thì mỗi một vi xử lý tớ (Slave) sẽ có một địa chỉ duy nhất do chúng ta qui định.
- Khi bộ xử lý chủ muốn phát một khối dữ liệu đến một trong các bộ xử lý tớ thì trước tiên vi xử lý chủ (Master) phải gởi ra 1 byte địa chỉ để nhận diện bộ xử lý tớ (Slave) muốn kết nối.
- Byte địa chỉ được phân biệt với byte dữ liệu bởi bit thứ 9: trong byte địa chỉ thì bit thứ 9 bằng 1 và trong byte dữ liệu thì bit thứ 9 bằng 0.
- Các vi xử lý tớ sau khi nhận được byte địa chỉ sẽ biết được vi xử lý chủ (Master) muốn giao tiếp tớ nào. Khi có vi xử lý tớ (Slave) được phép thì nó sẽ xóa bit SM2 để bắt đầu nhận các byte dữ liệu tiếp theo. Còn các vi xử lý không được phép thì vẫn giữ nguyên bit SM2=1 để không nhận các byte dữ liệu truyền giữa vi xử lý chủ và vi xử lý tớ đang được

phép. Vi xử lý tớ sau khi kết nối với vi xử lý chủ (Master) xong thì phải làm cho bit SM2=1 để sẵn sàng kết nối cho những lần tiếp theo.

- Sau khi thực hiện xong việc trao đổi dữ liệu thì vi xử lý muốn truy xuất một vi xử lý khác thì phải tạo ra một địa chỉ mới và vi xử lý tớ tương ứng với địa chỉ đó được phép và hoạt động giống như vừa trình bày.

❖ Ví dụ: Dưới đây cho thấy, cách thức sử dụng ngắt cổng truyền nối tiếp để tạo liên lạc đa xử lý.

Khi bộ xử lý chủ (Master) muốn truyền 1 khối dữ liệu tới một trong những bộ xử lý (Slave) khác, đầu tiên nó gửi đi 1 byte địa chỉ để xác định địa chỉ của bộ xử lý đích (Slave). Một byte địa chỉ khác với một byte dữ liệu ở chỗ: bit thứ 9 bằng 1 ở byte địa chỉ và bằng 0 ở byte dữ liệu. Với SM2=1, không có bộ xử lý (Slave) nào được ngắt bởi 1 byte dữ liệu. Tuy nhiên 1 byte địa chỉ sẽ ngắt tất cả các bộ xử lý (Slave) khác, để cho mỗi bộ xử lý(slave) khác có thể kiểm tra byte nhận được và để xem có phải nó đang được trỏ tới không. Bộ xử lý (slave) nào được trỏ tới sẽ xóa (clear) bit SM2 của nó và chuẩn bị nhận các byte dữ liệu sẽ đưa đến. Các bộ xử lý (Slave) khác nếu không được nhắc tới, thì sẽ thiết lập (set) bit SM2 của chúng và tiếp tục hoạt động của mình mà không cần quan tâm tới dữ liệu trên kênh.

6. Tốc độ baud.

Mục tiêu: Biết cách chọn tốc độ truyền dữ liệu của Port nối tiếp.

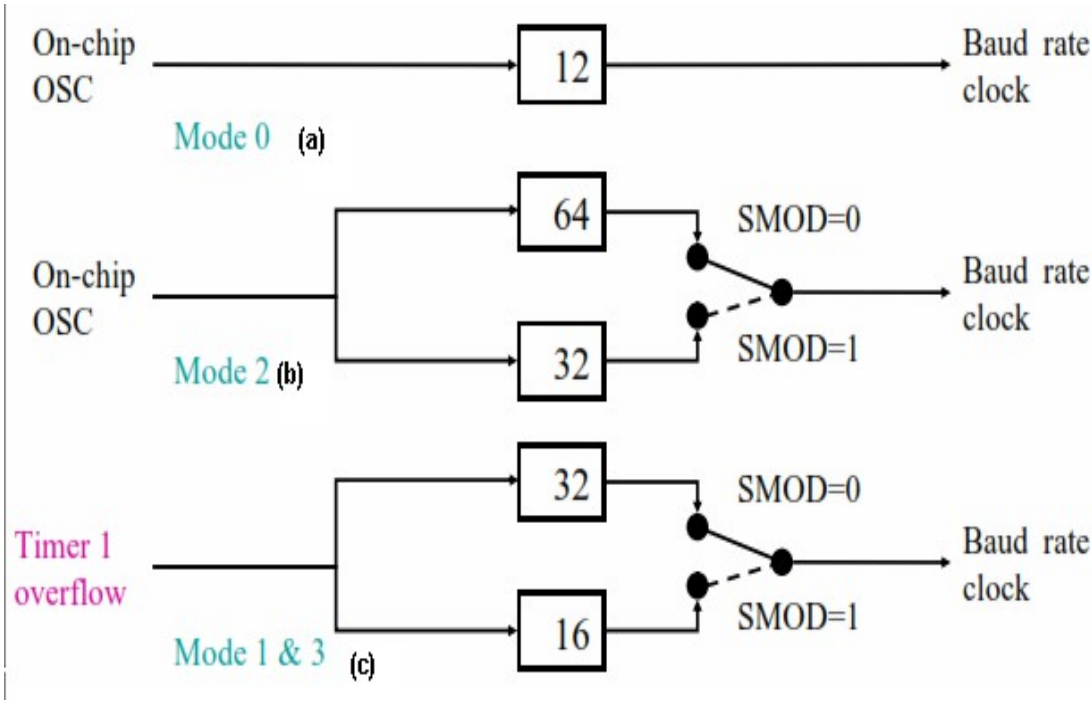
- Truyền dữ liệu nối tiếp nếu hoạt động ở kiểu 0 và kiểu 2 thì có tốc độ truyền cố định. Trong kiểu 0 thì tốc độ truyền bằng 112 tần số dao động trên Chip. Nếu sử dụng thạch anh 12 MHz thì tốc độ truyền của kiểu 0 là 1MHz như hình 5.9.
- Trong thanh ghi PCON có một bit SMOD có chức năng làm tăng gấp đôi tốc độ baud, mặc nhiên sau khi reset hệ thống thì bit SMOD = 0 thì các kiểu truyền dữ liệu hoạt động với tốc độ quy định, khi bit SMOD = 1 thì tốc độ tăng gấp đôi.

Ví dụ: Trong kiểu 2, tốc độ truyền có thể tăng gấp đơi từ giá trị mặc định 1/64 tần số dao động trên Chip (SMOD = 0) lên đến 1/32 tần số dao động trên Chip (ứng với SMOD =1) như hình 5.9.

- Do thanh ghi PCON không cho phép truy suất bit nên để set bit SMOD mà không thay đổi các bit khác của thanh ghi PCON thì phải thực hiện lệnh sau.

Lệnh sau đây set bit SMOD để tăng gấp đôi tốc độ truyền:

```
OR      PCON, #1000 0000b ;bit Smod ở vị trí thứ 7
```



Hình 5.9. Thiết lập tốc độ Baud.

❖ Các tốc độ Baud trong kiểu 1 và kiểu 3 của MCS-51 được xác định bởi tốc độ tràn của Timer:

+ Bởi vì Timer hoạt động ở tần số tương đối cao nên phải chia cho 32 khi bit SMOD = 0 và chia cho 16 nếu SMOD = 1 trước khi cung cấp xung clock để thiết lập tốc độ Baud cho Port nối tiếp.

+ Tốc độ Baud ở kiểu 1 và 3 của MCS51 được xác định bởi tốc độ tràn của Timer 1 hoặc Timer 2, hoặc cả 2 như hình 5.9.

6.1. Sử dụng bộ định thời 1 là xung clock tốc độ baud.

- Kỹ thuật thường dùng để tạo xung clock tốc độ baud là thiết lập timer 1 hoạt động ở chế độ 8bit tự nạp lại (chế độ định thời 2) và đặt giá trị nạp thích hợp vào thanh ghi TH1để có tốc độ tràn đúng, từ đó tạo ra tốc độ baud. Tốc độ baud = (Tốc độ tràn bộ định thời 1) / 32

Ví dụ : Nếu tốc độ baud là 9600

❖ Tốc độ tràn của bộ định thới = 9600*32 = 307200 Hz

❖ Nếu mạch dao động là 11.059 MHz, xung clock của bộ định thời = 11.059 M/12 = 921583 Hz,

giá trị nạp cho TH1= 921583/307200 = 3

Tốc độ baud	Tần số thạch anh	Giá trị nạp cho TH1
9600	11.059 MHz	-3(FDH)
2400	11.059 MHz	-12(F4H)
1200	11.059 MHz	-24(E8H)

Ví dụ: Viết một chuỗi lệnh để khởi động port nối tiếp sao cho port này hoạt động ở chế độ UART 8 bit với tốc độ baud 9600, sử dụng bộ định

thời 1 để cung cấp xung clock tốc độ baud. Dao động thạch anh là 11.059 MHz.

```
MOV     SCON,#01010010B
MOV     TMOD,#001000000B
MOV     TH1,# (-3)
SETB    TR1
```

- Chế độ 0: tốc độ baud cố định = 1/12 tần số thạch anh.
- Chế độ 2: tốc độ baud = 1/32 tần số thạch anh khi SMOD = 1 hay 1/64 khi SMOD = 0 (SMOD: nằm trong thanh ghi PCON).
- Chế độ 1 và 3: tốc độ baud xác định bằng tốc độ tràn của timer 1. Trong họ 89x52, có thể dùng timer 2 để tạo tốc độ baud còntrong các phiên bản mới, có thể dùng bộ tạo tốc độ nội (INT_BRG–Internal Baud Rate Generator). Việc xác định nguồn tạo tốc độ baud mô tả như hình 5.9 và bảng 5.3

Bảng 5.3 – Lựa chọn tốc độ baud

TCLK	RCLK	TBCK	RBCK	Clock phát	Clock thu
0	0	0	0	Timer 1	Timer 1
1	0	0	0	Timer 2	Timer 1
0	1	0	0	Timer 1	Timer 2
1	1	0	0	Timer 2	Timer 2
X	0	1	0	INT_BRG	Timer 1
X	1	1	0	INT_BRG	Timer 2
0	X	0	1	Timer 1	INT_BRG
1	X	0	1	Timer 2	INT_BRG
X	X	1	1	INT_BRG	INT_BRG

6.2. Tạo tốc độ baud bằng Timer 1.

Khi dùng timer 1 để tạo tốc độ baud, thông thường cần thiết lập timer 1 hoạt động ở chế độ 8 bit tự nạp lại và giá trị nạp ban đầu của

timer 1 (chứa trong thanh ghi TH1) phụ thuộc vào tốc độ baud cần tạo theo công thức sau:

$$\text{Baud\_Rate} = \frac{2^{SMOD} \times F_{Xtal}}{6^{(1-SPD)} \times 12 \times 32 \times [256 - (BRL)]}$$

$$(BRL) = 256 - \frac{2^{SMOD} \times F_{Xtal}}{6^{(1-SPD)} \times 12 \times 32 \times \text{Baud\_Rate}}$$

Đặt giá trị cho thanh ghi TH1 tùy thuộc vào tốc độ mong muốn theo bảng 5.4

Bảng 5.4 – Các giá trị nạp thông dụng trong truyền thông nối tiếp

Ví dụ:

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?

c) Hãy tìm tốc độ baud truyền dữ liệu.

Giải:

```
MOV     A, PCON      ; Chép 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           ; Khởi động 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 01000010)

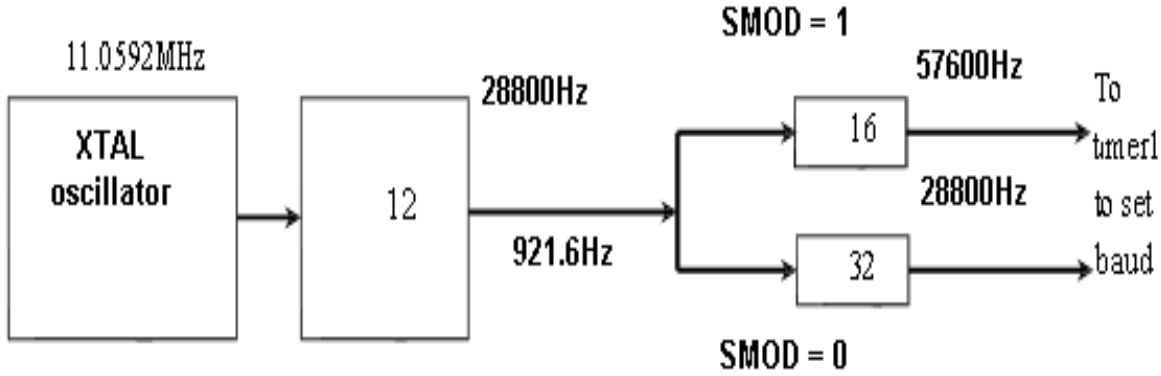
b) Với tần số XTAL = 11.0592MHz và SMOD = 1 trong chương trình trên ta có: 11.0592MHz/12 = 921.6kHz là tần số chu trình máy, 921.6kHz/16 = 57.6kHz là tần số được Timer1 sử dụng để đặt tốc độ baud

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.

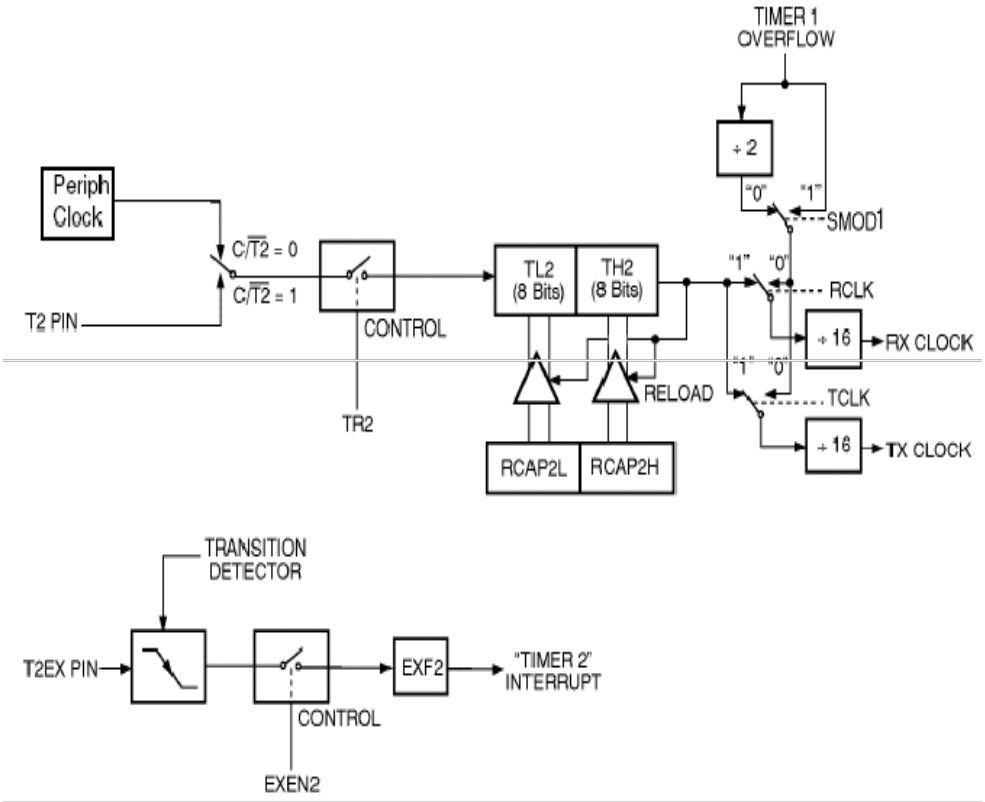
a) 57.600/9600 = 6 do vậy TH1 = - 6 hay TH1 = FAH

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?

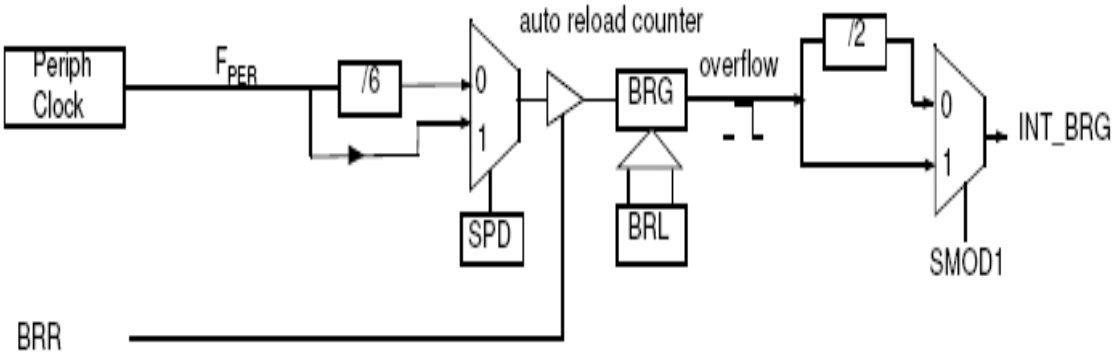
6.3. Tạo tốc độ baud bằng Timer 2.



Hình 5.10.– Tạo tốc độ baud bằng timer 2

Khi dùng Timer 2 để tạo tốc độ baud (hình 5.11), xung clock thu và phát có thể tách riêng bằng cách chỉ dùng TCLK hay RCLK. Lúc đó, xung clock còn lại được xác định theo Timer 1. Ngoài ra, cũng có thể tạo ngắt cho Timer 2 bằng cách đặt bit EXEN2 = 1 và ngắt tạo ra khi xuất hiện cạnh âm tại chân T2EX.

6.4. Bộ tạo tốc độ baud nội (INT_BRG – Internal Baud Rate Renerator).



Hình 5.11. Bộ tạo tốc độ baud nội

$$\text{Giá trị nạp} = -\frac{f_{OSC} \times 2^{SMOD1}}{2 \times 32 \times 6^{1-SPD} \times baud\_rate}$$

Giá trị nạp trong bộ tạo tốc độ nội chứa trong thanh ghi BRL và được xác định theo công thức sau:

Trong đó SMOD1 nằm trong thanh ghi PCON và SPD nằm trong thanh ghi BDRCON.

CÁC BÀI TẬP MỞ RỘNG, NÂNG CAO VÀ GIẢI QUYẾT VẤN ĐỀ ĐIỀU KHIỂN,TRUY XUẤT CÁC KHỐI CHỨC NĂNG

Câu 1: Nêu cać thanh ghi điều khiển trong Vi điều khiển.

Gợi ý:

Cać thanh ghi trong Vi điều khiển:

Thanh ghi SCON (Serial port controller).

Thanh ghi BDRCON (Baud Rate Control Register).

Ngoài ra còncó các thanh ghi SBUF (Serial Buffer), BRL (Baud Rate Reload), SADEN (Slave Address Mark), SADDR (Slave Address).

Lưu ý rằng các thanh ghi BDRCON, BRL, SADEN và SADDR chỉ có trong các phiên bản mới của MCS-51.

Câu 2: Nêu cać chế́ độ làm việc của Vi điều khiển 8051?

Gợi ý:

Cać chế́ độ làm việc của vi điều khiển:

Thanh ghi dịch 8 bit (chế độ 0).

Chế độ UART 8 bit có tốc độ baud thay đổi (chế độ 1).

Chế độ 2: UART 9 bit với tốc độ Baud cố định.

Chế độ 3: UART 9 bit với tốc độ Baud thay đổi.

Câu 3: Nêu cách khởi tạo và truy xuất thanh ghi PORT nối tiếp va`cách truyền thông đa xử lý?

Gợi ý:

Cać cách khởi tạo và truy xuất thanh ghi PORT nối tiếp

Bit điều khiển cho phép nhận dữ liệu (Receive Enable).

Bit dữ liệu thứ 9.

Thêm vào bit chẵn lẻ Parity.

Các cờ ngắt.

Truyền thông đa xử lý va`tôć độ BAUD

Bài 1. Điều khiển khối LED đơn :

1.1. Mục đích,yêu cầu :

Giúp cho sinh viên làm quen với việc điều khiển LED đơn như tắt, mở, dịch LED sang trái,dịch LED sang phải, sáng dồn, nhấp nháy. Sinh viên phải biết sử dụng Timer để tạo Delay và biết cách sử dụng các lệnh liên quan như CLR, CPL, RR,RL,R RC,RLC,CJNE, DJNZ, JNB, JB…

1.2. Chương trình tham khảo :

```
    ORG     000H
    LJMP    MAIN
    ORG     0030H
MAIN:
    LCALL   TURN_ON_OFF
    LCALL   FLASH_SINGLE_LED
    LCALL   FLASH_LED
    LCALL   SHIFT_LEFT_RIGHT
    LCALL   INCREMENT_LEFT_RIGHT
    SJMP $
;*************************************************
INCREMENT_LEFT_RIGHT:
    MOV     36H,#5
LOOP_INC_LR:
    MOV     P1,#11111111B
    INC_LEFT:
    CLR     C
    MOV     A,P1
    RLC     A
    MOV     P1,A
    LCALL   DELAY_50MS
    CJNE    A,#0,INC_LEFT
    MOV     P1,#11111111B
INC_RIGHT:
    CLR     C
    MOV     A,P1
    RRC     A
    MOV     P1,A
    LCALL   DELAY_50MS
    CJNE    A,#0,INC_RIGHT
    DJNZ    36H,LOOP_INC_LR
    MOV     P1,#0FFH
     RET
```

```
;*************************************************
SHIFT_LEFT_RIGHT:
   MOV      35H,#5
LOOP_SHIFT_LR:
   MOV      P1,#11111110B
LOOP_SHIFT_L:
   MOV      A,P1
   RL       A
   MOV      P1,A
   LCALL    DELAY_50MS
   CJNE A,#01111111B,LOOP_SHIFT_L
LOOP_SHIFT_R:
   MOV      A,P1
   RR       A
   MOV      P1,A
   LCALL    DELAY_50MS
   CJNE     A,#11111110B,LOOP_SHIFT_R
   DJNZ     35H,LOOP_SHIFT_LR
   MOV      P1,#0FFH
   RET
;*************************************************
FLASH_LED:
   MOV      34H,#100
LOOP_FL:


   MOV      A,P1
   CPL      A      ; COMPLEMENT A
   MOV      P1,A
   LCALL    DELAY_50MS
   DJNZ     34H,LOOP_FL
   RET
;*************************************************
FLASH_SINGLE_LED:
   MOV      33H,#100
LOOP_FLS:
   CPL      P1.0   ; COMPLEMENT BIT
   LCALL    DELAY_50MS
   DJNZ     33H,LOOP_FLS
```

```
    RET
;*************************************************
TURN_ON_OFF:
    CLR         P1.0   ; TURN ON
    LCALL       DELAY_5S
    SETB        P1.0   ; TURN OFF RET
;*************************************************
DELAY_5S:
    MOV         32H,#5
LOOP_DL5S:
    LCALL       DELAY_1S
    DJNZ        32H,LOOP_DL5S RET
;*************************************************
DELAY_1S:
    MOV         31H,#20
LOOP_DL1S:
    LCALL       DELAY_50MS
    DJNZ        31H,LOOP_DL1S RET
;*************************************************
DELAY_500MS:
    MOV         30H,#10
LOOP_DL500MS:
  LCALL         DELAY_50MS
    DJNZ        30H,LOOP_DL500MS
    RET


;*************************************************
DELAY_50MS:
    MOV         TMOD,#00000001B       ; TIMER 0 16 BIT SETB TR0
    JNB         TF0,$
    CLR         TF0
    CLR         TR0
    MOV         TH0,#HIGH(-50000)
    MOV         TL0,#LOW(-50000)
     RET
```

Bai 2: Chương trình đếm 000 đến 255 dùng ngắt ngoài.

Gợi y´giải bai:

Chương trình mẫu:

```
ORG    000H
```

```
   MOV      TMOD,#05H
   MOV      TH0,#0
   MOV      TL0,#0
   CLR      TF0
   SETB   TR0
MAIN:
   CALL    HEX_BCD
   CALL    BCD_7DOAN
   CALL    HIENTHI
   JMP    MAIN
;*********************************************
HEX_BCD:
   MOV    A,TL0
   MOV    B,#10
   DIV    AB
   MOV    10H,B       ;LUU SO HANG DV
   MOV    B,#10
   DIV    AB
   MOV    11H,B       ;LUU SO HANG CHUC
   MOV    12H,A       ;LUU SO HANG TRAM
RET
;*********************************************
BCD_7DOAN:
   MOV    DPTR,#MA7DOAN
   MOV    R0,#10H
   MOV    R1,#20H
GM:    MOV    A,@R0
   MOVC    A,@A+DPTR
   MOV    @R1,A
   INC    R0
   INC    R1
   CJNE    R0,#13H,GM
RET
;*********************************************
HIENTHI:
   MOV    R0,#20H
   MOV    A,#08H
HT:    MOV    P0,@R0
   MOV    P2,A
   CALL    DELAY
   MOV    P2,#00H       ;CHONG LEM
```

```
   INC   R0
   RR    A
   CJNE  A,#01H,HT
RET
;*********************************************
DELAY:
   MOV   R7,#0FFH
   DJNZ  R7,$
RET
;*********************************************
MA7DOAN:
DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
END
```

Bai 3: Viết chương trình điều khiển 8 LED chớp tắt.

Chương trình mẫu:

```
ORG    000H
MAIN:
   MOV    P0,#00H
   CALL    DELAY
   MOV    P0,#0FFH
   CALL   DELAY
   JMP   MAIN

;*****************************Chuong trinh con Delay
DELAY:
      MOV   R0,#0FFH
DEL: MOV    R1,#0FFH
      DJNZ   R1,$
      DJNZ   R0,DEL
RET
END
```

Bai 4: Viết chương trình điều khiển 16 LED sáng dần?

Chương trình mẫu:

```
ORG   000H
MAIN:
   MOV   P0,#0FFH       ;8 LED P0 TAT
   MOV   P2,#0FFH       ;8 LED P2 TAT
LOOP:   CALL   DELAY
   MOV   A,P0
   CLR   C
```

```
   RLC    A
   MOV    P0,A
   MOV    A,P2
   RLC    A
   MOV    P2,A
   JC    LOOP
   JMP    MAIN
;*************************************CHUONG TRINH CON
DELAY
DELAY:
      MOV    R0,#0FFH
DEL: MOV    R1,#0FFH
      DJNZ    R1,$
      DJNZ    R0,DEL
RET
END
```

Bai 5: Viết chương trình điều khiển động cơ bước?

Chương trình mẫu

```
ORG    000H
MAIN:
MOV    A,#0FEH
LOOP:
MOV    P2,A
CALL    DELAY
RL       A
CJNE    A,#0F7H,LOOP
JMP      MAIN
;*************************************
DELAY:
MOV    R6,#0FFH
DEL:
MOV    R7,#0FFH
DJNZ    R7,$
DJNZ    R6,DEL
RET
END
```

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức: Trình bày được cấu tạo và các chế độ làm việc của cổng truyền thông nối tiếp theo nội dung đã học
+ Về kỹ năng:
- Thực hiện cổng truyền thông nối tiếp đúng yêu cầu kỹ thuật.
- Thực hiện thu phát dữ liệu nối tiếp bằng 8051 đạt yêu cầu kỹ thuật.
- Thực hiện viết các chương trình theo yêu cầu cho trướoc
+ Thái độ: Đánh giá phong cách, thái độ học tập

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm
+ Về kỹ năng: Đánh giá kỹ năng thực hành Mỗi sinh viên, hoặc mỗi nhóm học viên thực hiện công việc theo yêu cầu của giáo viên.Tiêu chí đánh giá theo các nội dung:
- Độ chính xác của công việc
- Tính thẩm mỹ của mạch điện
- Độ an toàn trên mạch điện
- Thời gian thực hiện công việc
- Độ chính xác theo yêu cầu kỹ thuật

+ Thái độ: Tỉ mỉ, cẩn thận, chính xác.

BAI 6

NGẮT

Mã bài: MĐ24-06

Giới thiệu:

Lập trình cho vi điều khiển bằng cách 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. Việc dừng chương trình đang thực thi để phục vụ cho một chương trình khác khi xảy ra

một sự kiện. Chương trình xử lý sự kiện ngắt gọi là chương trình phục vụ ngắt (ISR- Interrupt Service Routine).

Mục tiêu của bài:

- Trình bày được tác dụng thực tế của một hệ thống được điều khiển bằng tín hiệu ngắt theo nội dung đã học.
- Thực hiện tổ chức ngắt và cơ chế thực hiện chương trình phục vụ ngắt của 8051 đúng yêu cầu kỹ thuật.
- Thực hiện tổ chức ngắt đạt yêu cầu kỹ thuật.

Nội dung chính:

1. Mở đầu.

Mục tiêu: Trình bày được tác dụng thực tế của một hệ thống được điều khiển bằng tín hiệu ngắt.

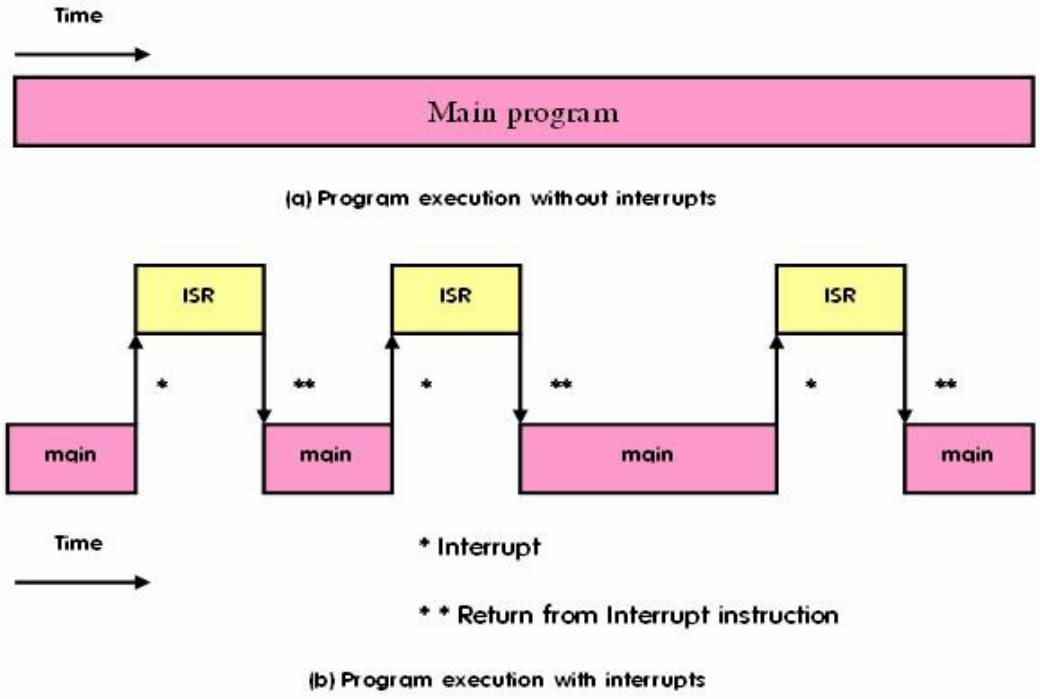
- Ngắt (interrupt) là sự sảy ra của một điều kiện làm cho trương trình hiện hành bị tạm ngưng trong khi điều kiên này được phục vụ bởi một chương trình khác. Các ngắt đóng vai trị quan trọng trong việc thiết kế và thực hiện các ứng dụng củ Bít dữ liệu thứ 9a vi điều khiển. Các ngắt cho phép hệ thống đáp ứng một sự kiện theo cách không đồng bộ và xử lý một sự kiện trong khi một chương trình khác đang thực thi. Một hệ thống được điều khiển bởi ngắt cho ta ảo tưởng đang làm nhiều công việc đồng thời.
- Có nhiều sự tác động làm ngừng chương trình chính gọi là các nguồn ngắt, trong vi điều khiển khi timer/counter đếm tràn sẽ tạo ra ngắt. Ngắt đóng một vai trò quan trọng trong lập trình điều khiển.
- CPU không thể thực hiện nhiều hơn một lệnh ở một thời điểm nhưng CPU có thể tạm ngưng việc thực thi một chương trình để thực thi một chương trình khác rồi sau đó quay trở về thực thi tiếp chương trình đang bị tạm ngưng. Điều này giống như CPU rời khỏi chương trình gọi để thực hiện chương trình con bị gọi để rồi sau đó quay về chương trình gọi.
- Chương trình xử lý một ngắt được gọi là chương trình phục vụ ngắt ISR (interrupt service routine). ISR được thực thi nhằm đáp ứng một ngắt và trong trường hợp tổng quát thực hiện việc xuất nhập với một thiết bị. Khi một ngắt xuất hiện, việc thực thi chương trình chính tạm thời bị dừng và CPU thực hiện rẽ nhánh đến trình phục vụ ngắt ISR. CPU thực thi ISR để thực hiện một công việc và kết thúc việc thực thi này khi gặp lệnh "quay về từ một trình phục vụ ngắt" RETI.Ta có thể nói chương trình chính được thực thi ở mức nền còn ISR được thực thi ở mức ngắt.
- Khi sử dụng ngắt sẽ cho phép vi xử lý hay vi điều khiển đáp ứng

nhiều sự kiện quan trọng và giải quyết sự kiện đó trong khi chương trình khác đang thực thi.

Ví dụ:

Trong vi điều khiển đang thực hiện chương trình chính thì có dữ liệu từ hệ thống khác gởi đến, khi đó vi điều khiển ngừng chương trình chính để thực hiện chương trình phục vụ ngắt nhận dữ liệu xong rồi trở lại tiếp tục thực hiện chương trình chính, hoặc có một tín hiệu báo ngắt từ bên ngoài thì vi điều khiển sẽ ngừng thực hiện chương trình chính để thực hiện chương trình ngắt rồi tiếp tục thực hiện chương trình chính.

- Ta có thể sử dụng ngắt để yêu cầu vi điều khiển thực hiện nhiều chương trình cùng một lúc có nghĩa là các chương trình được thực hiện xoay vòng.
- Ta có thể minh họa quá trình thực hiện 1 chương trình trong trường hợp có ngắt và không có ngắt như hình 6.1

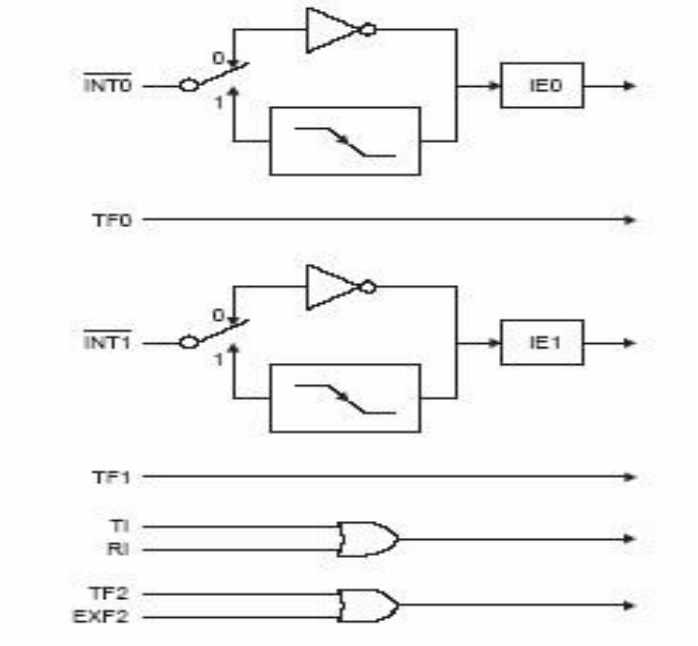


Hình 6.1. Vi điều khiển thực hiện chương trình chính trong 2 trường hợp không và có ngắt.

Trong đó : Ký hiệu * cho biết vi điều khiển ngừng chương trình chính để thực thi chương trình con phục vụ ngắt ISR. Còn ký hiệu ** cho biết vi điều khiển quay trở lại thực hiện tiếp chương trình chính sau khi thực hiện xong chương trình con phục vụ ngắt ISR.

2.Tổ chức ngắt của 8051 (hình 6.2):

Mục tiêu: Biết cách khởi tạo ngắt và chọn các nguồn ngắt.



Hình 6.2. Vi điều khiển 89C52 có 6 nguồn ngắt.

Có 5 nguyên nhân tạo ra ngắt đối với 8051: hai ngắt do bên ngoài, hai ngắt do bộ định thời và một ngắt do port nối tiếp. 8052 có thêm nguyên nhân ngắt thứ 6: do bộ định thời được thêm vào, bộ định thời thứ ba. Khi ta thiết lập trạng thái ban đầu cho hệ thống, tất cả các ngắt điều bị vơ hiệu hóa và sau đó chúng được cho phép riêng rẻ bằng phần mềm.

Khi xảy ra hai hay nhiều ngắt đồng thời hoặc xảy ra một ngắt trong khi một ngắt đang phục vụ, ta có hai sơ đồ xử lý ngắt: sơ đồ chuỗi vùng và sơ đồ hai mức ưu tiên.

Trong đó:

Bit	Mô tả
EA	Cho phép/cấm ngắt tổn cục – 0: Cấm tất cả các ngắt

2.1 Cho phép và không cho phép ngắt:

- Trước tiên chúng ta phải hiểu cho phép và không cho phép ngắt là như thế nào? Khi ta cho phép ngắt và khi ngắt tác động thì vi điều khiển sẽ ngừng chương trình chính để thực hiện chương trình con phục vụ ngắt, khi không cho phép thì dù có sự tác động đến ngắt vi điều khiển vẫn tiếp tục thực hiện chương trình chính, không thực hiện chương trình phục vụ ngắt.
- Trong vi điều khiển có 1 thanh ghi IE (Interrupt Enable) ở tại địa chỉ 0A8H có chức năng cho phép/cấm ngắt. Ta sử dụng thanh ghi này để cho phép hay không cho phép đối với từng nguồn ngắt và cho toàn bộ các nguồn ngắt. Tổ chức của thanh ghi như sau:

EA	-	ET2	ES	ET1	EX1	ET0	EX0

- Mỗi một nguyên nhân ngắt được cho phép hoặc không cho phép riêng rẽ thông qua thanh ghi chức năng đặc biệt định địa chỉ bit, thanh ghi cho phép ngắt IE (interrupt enable) có địa chỉ byte là 0A8H. Mỗi một bit của thanh ghi này cho phép hoặc không cho phép từng nguyên nhân ngắt riêng rẽ, thanh ghi IE đồng thời có một bit toàn cục(global) cho phép hoặc không cho phép tất cả các ngắt. Hoạt động của từng bit trong thanh ghi cho phép ngắt IE được tóm tắt trong bảng sau:

Thanh ghi cho phép ngắt IE

Bit	Ký hiệu	Địa chỉ bit	Mô tả (0: không cho phép, 1: cho phép)
IE.7	EA	AFH	Cho phép/không cho phép ngắt tổn cục
IE.6	-	AEH	Không sử dụng
IE.5	ET2	ADH	Cho phép ngắt do bộ định thời 2
IE.4	ES	ACH	Cho phép ngắt do port nối tiếp
IE.3	ET1	ABH	Cho phép ngắt do bộ định thời 1
IE.2	EX1	AAH	Cho phép ngắt từ bên ngoài (ngắt ngoài 1
IE.1	ET0	A9H	Cho phép ngắt do bộ định thời 0
IE.0	EX0	A8H	Cho phép ngắt từ bên ngoài (ngắt ngoài 0

- Trong thanh ghi IE có bit IE.6 chưa dùng đến, bit IE.7 là bit cho phép/cấm ngắt toàn bộ các nguồn ngắt. Khi bit IE.7= 0 thì cấm hết tất cả các nguồn ngắt, khi bit IE.7=1 thì cho phép tất cả các nguồn ngắt nhưng còn phụ thuộc vào từng bit điều khiển ngắt của từng nguồn ngắt.

Ví dụ: Ngắt do bộ định thời 1 được cho phép bằng cách dùng hai lệnh:

```
SETB    ET1     : Cho phép ngắt do bộ định thời một
SETB    EA      : set bit EA bằng 1 để cho phép ngắt tổn cục
```

Hoặc

```
MOV IE,#10001000B
```

- Đối với yêu cầu của ví dụ trên thì 2 cách thực hiện trên là xong nhưng ta hãy so sánh 2 cách thực hiện và chú ý một vài điều trong lập trình:Các lệnh của cách 1 không ảnh hưởng các bit còn lại trong thanh ghi IE.

- Cách thứ hai sẽ xóa các bit còn lại trong thanh ghi IE. Ở đầu chương trình ta nên khởi gán IE với lệnh MOV BYTE, nhưng khi điều khiển cho phép hay cấm trong chương trình thì ta sẽ dùng các lệnh SET BIT và CLR BIT để tránh làm ảnh hưởng đến các bit khác trong thanh ghi IE.

2.2 Ưu tiên ngắt.

- Khi có nhiều nguồn ngắt tác động cùng lúc thì ngắt nào quan trọng cần thực hiện trước và ngắt nào không quan trọng thì thực hiện sau giống như các công việc mà ta giải quyết hằng ngày.Ngắt cũng được thiết kế có sự sắp xếp thứ tự ưu tiên từ thấp đến cao để người lập trình sắp xếp các nguồn ngắt theo yêu cầu công việc mà mình xử lý.
- Mỗi một nguyên nhân ngắt được lập trình riêng rẻ để có một trong hai mức ưu tiên thông qua chức năng thanh ghi đặc biệt được định địa chỉ bit, thanh ghi ưu tiên ngắt IP (interrupt priority), thanh ghi này có địa chỉ byte là 0B8H.
- Thanh ghi có chức năng thiết lập chế độ ưu tiên trong vi điều khiển là thanh ghi IP (Interrupt Priority) tại địa chỉ 0B8H. Tổ chức của thanh ghi như sau:

-	-	PT2	PS	PT1	PX1	PT0	PX0

Hoạt động của từng bit trong thanh ghi IP được tóm tắt trong bảng sau:

Tha	Ký hiệu	Địa chỉ bit	Mô tả (1: mức cao, 0: mức thấp)
IP.7	-	-	Không sử dụng
IP.6	-	-	Không sử dụng
IP.5	PT2	0BDH	Ưu tiên ngắt do bộ định thời 2
IP.4	PS	0BCH	Ưu tiên ngắt do port nối tiếp
IP.3	PT1	0BBH	Ưu tiên ngắt do bộ định thời 1
IP.2	PX1	0BAH	Ưu tiên ngắt ngoài 1
IP.1	PT0	0B9H	Ưu tiên ngắt do bộ định thời 0
IP.0	PX0	0B8H	Ưu tiên ngắt ngoài 0

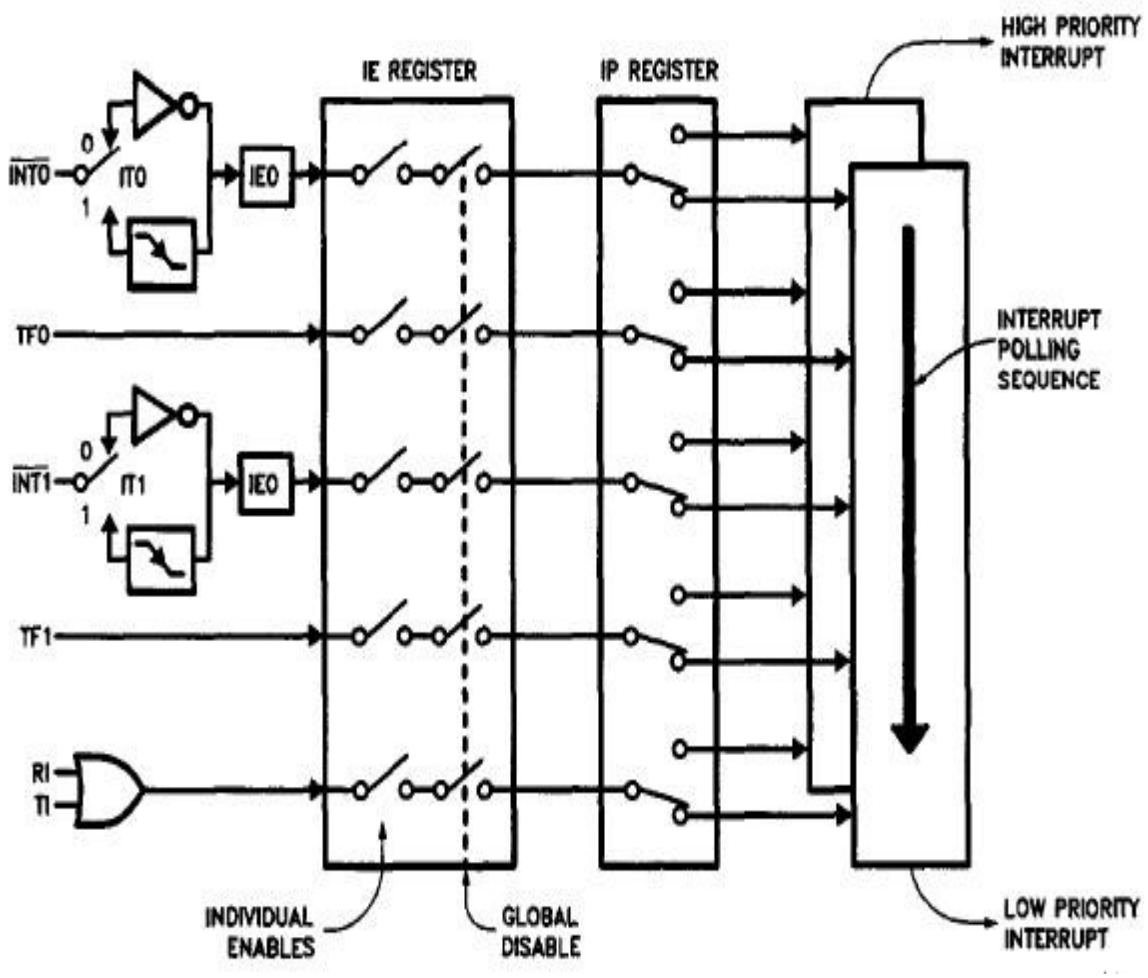
- Khi hệ thống được thiết lập lại trạng thái ban đầu, thanh ghi IP bị xóa và sẽ mặc định tất cả các ngắt ở mức ưu tiên thấp.
- Trong 89C51 có 2 mức ưu tiên thấp và 2 mức ưu tiên cao. Nếu vi điều khiển đang thực hiện chương trình con phục vụ ngắt có mức ưu tiên thấp và có một yêu cầu ngắt với mức ưu tiên cao hơn xuất hiện thì vi điều khiển phải ngừng thực hiện chương trình con phục vụ ngắt có mức ưu tiên thấp để thực hiện chương trình con phục vụ ngắt mới có ưu tiên cao hơn.
- Ngược lại nếu vi điều khiển đang thực hiện chương trình con phục vụ ngắt có mức ưu tiên cao hơn và có yêu cầu ngắt với mức ưu tiên thấp hơn xuất hiện thì vi điều khiển vẫn tiếp tục thực hiện cho đến khi thực hiện xong chương trình phục vụ ngắt có ưu tiên cao hơn rồi mới thực hiện chương trình phục vụ ngắt có ưu tiên thấp đang yêu cầu.
- Chương trình chính mà vi điều khiển luôn thực hiện trong một hệ thống thì ở mức thấp nhất, không có liên kết với yêu cầu ngắt nào, luôn

luôn bị ngắt bất chấp ngắt ở mức ưu tiên cao hay thấp. Nếu có 2 yêu cầu ngắt với các ưu tiên khác nhau xuất hiện đồng thời thì yêu cầu ngắt có mức ưu tiên cao hơn sẽ được phục vụ trước.

2.3. Chuỗi.

- Nếu có hai ngắt có cùng mức ưu tiên xuất hiện đồng thời, chuỗi vòng cố định sẽ xác định ngắt nào được phục vụ trước. Chuỗi vòng này sẽ là: ngắt ngoài 0, ngắt do bộ định thời 0, ngắt ngoài 1, ngắt do bộ định thời 1, ngắt do port nối tiếp, ngắt do bộ định thời 2.
- Các bit cờ của các nguồn ngắt được tóm tắt ở bảng sau:

Interrupt	Flag	SFR Register and Bit Position
External 0	IE0	TCON 1
External 1	IE1	TCON 3
Timer 1	TF1	TCON 7
Timer 0	TF0	TCON 5
Serial Port	TI	SCON 1
Serial Port	RI	CSON 0
Timer 2	TF2	T2CON 7(8052)
Timer 2	EXF2	T2CON 6(8052)



Hình 6.3. Cấu trúc ngắt của vi điều khiển.

- Hình 6.3 ta thấy tác dụng của các thanh ghi IE hoạt động như một contact On/Off còn thanh ghi IP hoạt động như một contact chuyển mạch giữa 2 vị trí để lựa chọn 1 trong 2.

- Ta hãy bắt đầu từ thanh ghi IE trước: bit cho phép ngắt toàn cục (Global Enable) nếu được phép sẽ đóng toàn bộ các contact và tùy thuộc vào bit cho phép của từng nguồn ngắt có được phép hay không và chúng hoạt động cũng giống như một contact: nếu được phép thì đóng mạch và tín hiệu yêu cầu ngắt sẽ đưa vào bên trong để xử lý, nếu không được phép thì contact hở mạch nên tín hiệu yêu cầu ngắt sẽ không đưa vào bên trong và không được xử lý.

- Tiếp theo là thanh ghi IP: tín hiệu sau khi ra khỏi thanh ghi IE thì đưa đến thanh ghi IP để sắp xếp ưu tiên cho các nguồn ngắt. Có 2 mức độ ưu tiên: mức ưu tiên cao và mức ưu tiên thấp. Nếu các nguồn nào có ưu tiên cao thì contact chuyển mạch sẽ đưa tín hiệu yêu cầu ngắt đó đến vùng kiểm tra có ưu tiên cao, nếu các nguồn nào có ưu tiên thấp thì contact chuyển mạch sẽ đưa tín hiệu yêu cầu ngắt đó đến vùng kiểm tra có ưu tiên

thấp.

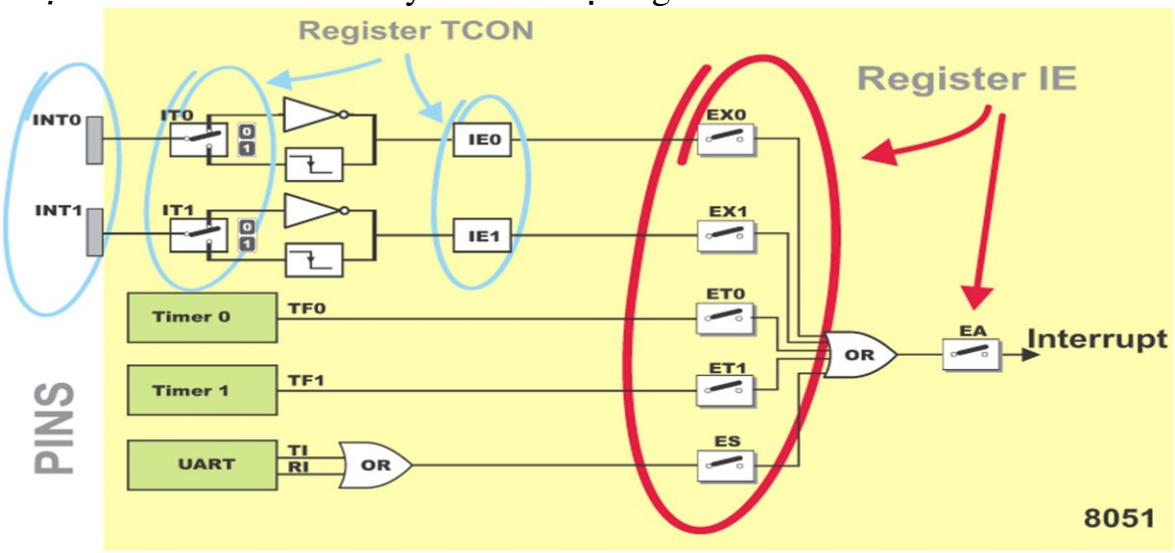
- Vùng kiểm tra ngắt ưu tiên cao sẽ được thực hiện trước và sẽ kiểm tra theo thứ tự từ trên xuống và khi gặp yêu cầu ngắt nào thì yêu cầu ngắt đó sẽ được thực hiện. Sau đó tiếp tục thực hiện cho vùng kiểm tra ưu tiên ngắt có mức ưu tiên thấp hơn.
- Trong hình còn cho chúng ta thấy yêu cầu ngắt truyền dữ liệu nối tiếp tạo ra từ tổ hợp OR của 2 cờ báo nhận RI và cờ báo phát TI. Khi ngắt truyền dữ liệu xảy ra và ta muốn biết là do cờ nhận hay cờ phát tạo ra ngắt để thực hiện 2 công việc khác nhau thì ta phải kiểm tra cờ RI và TI để biết thực hiện công việc nào tương ứng.

Ví dụ trong truyền dữ liệu: khi có báo ngắt truyền dữ liệu thì ta phải kiểm tra xem cờ RI = 1 hay không? Nếu đúng thì hệ thống khác đang gởi dữ liệu đến và ta phải chuyển hướng chương trình phục vụ ngắt sang hướng nhận dữ liệu, nếu không phải thì chắc chắn là cờ TI=1 báo cho chúng ta biết rằng dữ liệu đã truyền đi xong và sẵn sàng truyền kí tự tiếp theo và khi đó ta phải chuyển hướng chương trình phục vụ ngắt sang phát dữ liệu tiếp theo.

- Tương tự, các yêu cầu ngắt của Timer2 tạo ra từ tổ hợp OR của cờ tràn TF2 và cờ nhập ngoài EXF2.

3. Xử lý ngắt (hình 6.4).

Mục tiêu: Biết cách xử lý các tín hiệu ngắt.



Hình 6.4. 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. Khi có một

ngắt xuất hiện và được CPU chấp nhận, chương trình chính bị ngắt. Các thao tác sau đây xảy ra:

Hoàn tất việc thực thi lệnh hiện hành.
Bộ đếm chương trình PC được cất vào stack.
Trạng thái của ngắt hiện hành được lưu giữ lại.
Các ngắt được chận lại ở mức ngắt.
Bộ đếm chương trình PC được nạp địa chỉ vector của trình phục vụ ngắt ISR.

ISR được thực thi ISR được thực thi để đáp ứng công việc của ngắt. Việc thực thi ISR kết thúc khi gặp lệnh RETI. Lệnh này lấy lại giá trị cũ của bộ đếm chương trình PC từ stack và phục hồi trạng thái của ngắt cũ.

- Chú ý: chương trình con phục vụ ngắt không được làm mất hoặc làm sai địa chỉ của PC đã lưu trong ngăn xếp nếu điều này xảy ra thì khi trở lại chương trình chính CPU sẽ không thực hiện tiếp công việc của chương trình chính và chúng ta cũng không xác định CPU đang làm gì và ở đâu. Khi đó chúng ta mất quyền kiểm sốt vi xử lý.
- Trong "vi điều khiển" thì bộ nhớ ngăn xếp là bộ nhớ RAM nội nên chúng sẵn sàng hoạt động cho việc lưu trữ tạm, còn đối với "vi xử lý" thì bộ nhớ ngăn xếp sử dụng bộ nhớ ngoài nên bạn phải khởi tạo bộ nhớ ngăn xếp phải là vùng nhớ RAM để có thể ghi và đọc lại được, nếu bạn khởi tạo tại vùng nhớ EPROM hoặc khởi tạo tại nơi mà bộ nhớ không ghi vào được thì sẽ làm mất địa chỉ – dữ liệu lưu vào bộ nhớ ngăn xếp dẫn đến chương trình sẽ thực hiện sai.Một điều cần phải chú ý nữa là trong lập trình chúng ta không được nhảy từ chương trình con sang chương trình chính để thực hiện tiếp chương trình vì làm như vậy sau nhiều lần thực hiện thì bộ nhớ ngăn xếp sẽ bị tràn và ghi đè lên các dữ liệu khác làm sai chương trình.
- Trong trường hợp này chúng ta sẽ thấy rằng chương trình chúng ta thực hiện đúng một vài lần và sau đó thì sai.

3.1 Các vector ngắt.

Ngắt	Địa chỉ vector
Reset hệ thống	0000H
Ngắt ngoài 0	0003H
Ngắt do bộ định thời 0	000BH
Ngắt ngoài 1	0013H
Ngắt do bộ định thời 1	001BH

Ngắt do port nối tiếp	0023H
Ngắt do bộ định thời 2	002BH

- Khi một ngắt được chấp nhận, giá trị được nạp cho bộ đếm chương trình PC được gọi là vector ngắt. Vector ngắt là địa chỉ bắt đầu của trình phục vụ ngắt của nguyên nhân ngắt tương ứng. Vector reset hệ thống bắt đầu tại địa chỉ 0000H: khi reset vi điều khiển thì thanh ghi PC = 0000H và chương trình chính luôn bắt đầu tại địa chỉ này. Khi bạn sử dụng yêu cầu ngắt nào thì chương trình con phục vụ ngắt phải viết đúng tại địa chỉ tương ứng. Ví dụ bạn sử dụng ngắt timer T0 thì chương trình ngắt bạn phải viết tại địa chỉ 000BH.Do khoảng vùng nhớ giữa các vector địa chỉ của các nguồn ngắt chỉ có vài ô nhớ . Ví dụ như vector địa chỉ ngắt của ngắt INT0 tại 0003H và vector địa chỉ ngắt của ngắt T0 tại 000BH chỉ cách nhau có 9 ô nhớ . Nếu chương trình phục vụ ngắt của ngắt INT0 có kích thước lớn hơn 9 byte thì nó sẽ đụng đến vùng nhớ của ngắt T0. Cách giải quyết tốt nhất là ngay tại địa chỉ 0003H ta viết lệnh nhảy đến một vùng nhớ khác rộng hơn. Còn nếu các ngắt T0 và các ngắt khác không sử dụng thì ta có thể viết chương trình tại đó cũng được. Chương trình chính luôn bắt đầu tại địa chỉ 0000H sau khi reset hệ thống, nếu trong chương trình có sử dụng ngắt thì ta phải dùng lệnh nhảy tại địa chỉ 0000H để nhảy đến một vùng nhớ khác rộng hơn không bị giới hạn để viết tiếp.

3.2. Ngắt ngoài (External Interrupt).

- 8051 có 2 ngắt ngoài là INT0 và INT1. Ngắt ngoài được hiểu là ngắt được gây ra bởi sự kiện mức logic 0 (mức điện áp thấp, gần 0V) hoặc sườn xuống (sự chuyển mức điện áp từ mức cao về mức thấp) xảy ra ở chân ngắt tương ứng (P3.2 với ngắt ngoài 0 và P3.3 với ngắt ngoài 1). Việc lựa chọn kiểu ngắt được thực hiện bằng các bit IT (Interrupt Type) nằm trong thanh ghi TCON. Đây là thanh ghi điều khiển timer nhưng 4 bit LSB (bit0..3) được dùng cho các ngắt ngoài.Khi bit ITx = 1 thì ngắt ngoài tương ứng được chọn kiểu là ngắt theo sườn xuống, ngược lại nếu bit ITx = 0 thì ngắt ngoài tương ứng được sẽ có kiểu ngắt là ngắt theo mức thấp. Các bit IE là các bit cờ ngắt ngoài, chỉ có tác dụng trong trường hợp kiểu ngắt được chọn là ngắt theo sườn xuống.Khi kiểu ngắt theo sườn xuống được chọn thì ngắt sẽ xảy ra duy nhất một lần khi có sườn xuống của tín hiệu, sau đó khi tín hiệu ở mức thấp, hoặc có sườn lên, hoặc ở mức cao thì cũng không có ngắt xảy ra nữa cho đến khi có sườn xuống tiếp theo. Cờ ngắt IE sẽ dựng lên khi có sườn xuống và tự động bị xóa khi CPU bắt đầu xử lý ngắt.Khi kiểu ngắt theo mức thấp được chọn thì ngắt sẽ xảy ra bất cứ khi nào tín hiệu tại chân ngắt ở mức thấp. Nếu sau khi xử lý xong ngắt mà tín hiệu vẫn ở

mức thấp thì lại ngắt tiếp, cứ như vậy cho đến khi xử lý xong ngắt lần thứ n, tín hiệu đã lên mức cao rồi thì thôi không ngắt nữa. Cờ ngắt IE trong trường hợp này không có ý nghĩa gì cả.Thông thường kiểu ngắt hay được chọn là ngắt theo sườn xuống.

4. Thiết kế chương trình sử dụng ngắt.

Mục tiêu: Viết được chương trình sử dụng ngắt.

❖ Các bước khi thực hiện một ngắt.

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

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

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

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

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

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 trị 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.

❖ Thứ tự ưu tiên ngắt

Khi có hai hay nhiều ngắt cùng lúc xảy ra, hoặc một ngắt đang thực hiện, nếu mở 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 động 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 thời 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

Mẫu đề nghị cho một chương trình được thực thi độc lập có sử dụng ngắt như sau :

```
ORG         0000H       ; điểm nhập sau khi reset
LJMP MAIN
                        ; các điểm nhập của các ISR
ORG 0030H               ; điểm nhập chương trình chính
MAIN:                   ; chương trình chính bắt đầu
END
```

4.1 Các trình phục vụ ngắt kích thước nhỏ.

Các trình phục vụ ngắt phải được bắt đầu ở gần đáy của bộ nhớ chương trình tại các địa chỉ qui định. Mặc dù chỉ có 8 byte giữa các điểm nhập của các trình phục vụ ngắt, dung lượng này thường đủ để thực hiện các công việc được yêu cầu và quay trở về chương thình chính từ một trình phục vụ ngắt. Điều này có nghĩa là trình phục vụ ngắt cho các ngắt tương ứng thường không dài quá 8 byte.

Nếu có nhiều ngắt được dùng ta phải cẩn thận để đảm bảo các ISR được bắt đầu đúng vị trí mà không tràn sang ISR kế.

Ví dụ:

```
        ORG         0000H       ; điểm nhập reset
        LJMP MAIN
        ORG         000BH       ; điểm nhập ngắt bộ định thời 0
T0_ISR :                        ; bắt đầu ISR cho bộ định thời 0
        MOV P2,#00001111B
        RETI                    ; trở về chương trình chính
MAIN :                          ; chương trình chính
        -
        -
        END                     ; kết thúc chương trình
```

4.2 Các trình phục vụ ngắt kích thước lớn.

Nếu một trình phục vụ ngắt dài hơn 8 byte được cần đến, ta phải di chuyển phương trình này đến một nơi khác trong bộ nhớ chương trình hoặc ta có thể cho lấn qua điểm nhập của ISR kế. Điển hình là ISR bắt đầu với một lệnh nhảy đến một vùng khác của bộ nhớ chương trình, ở đó ISR được trải rộng nếu cần.

Ví dụ 1:

```
        ORG      0000H      ; điểm nhập reset
        LJMP     MAIN
        ORG      000BH      ; điểm nhập ngắt do timer 0
        LJMP     T0_ISR
                            ; điểm nhập các ngắt khác ( nếu có )
        ORG 0030H           ; địa chỉ phía trên các vector ngắt
MAIN:                       ; chương trình chính
  T0_ISR:                   ; chương trình con phụ vụ ngắt
                            ; do bộ định thời 0

        RETI                ; quay về chương trình chính
      END
```

Ví dụ 2:

Viết chương trình tạo sóng vuông có tần số 10KHz trên chân P1.0 sử dụng bộ định thời 0 và ngắt do bộ định thời 0.

```
        ORG    0000H        ; điểm nhập reset
        LJMP   MAIN
        ORG    000BH        ; điểm nhập ngắt timer 0
T0_ISR:                     ; ISR timer 0
        CPL    P1.0      ; đảo bit P1.0
        RETI                ; trở về chương trình chính
MAIN:
  MOV     TMOD,#00000010B; timer 0 hoạt động ở chế độ 2 (8
          bit)
        MOV    TH0,#-50     ; giá trị nạp lại tương ứng với 50us
        MOV    IE,#10000010B ; cho phép ngắt do bộ định thời 0
        SETB   TR0          ; bật timer 0

        SJMP   $            ; nhảy tại chổ
      END
```

Ví dụ 3:

Viết chương trình sử dụng các ngắt để tạo ra đồng thời các dạng sóng vuông có tần số là 10KHz trên chân P1.0 và 500Hz trên chân P1.1

```
        ORG     0000H
        LJMP    MAIN
        ORG     000BH
        LJMP    NGAT_T0
        ORG     001BH
        LJMP    NGAT_T1
        ORG     0030H
MAIN:
        MOV     TMOD,#00010010B; timer 1 chế độ 1, timer 0 chế độ
        2
        MOV     TH0,#-50
        MOV     TH1,#HIGH(-1000)
        MOV     TL1,#LOW(-1000)
        MOV     IE,#10001010B
        SETB    TR0
        SETB    TR1
        SJMP $
NGAT_T0:
        CPL     P1.0
        RETI
NGAT_T1:
        CLR     TR1
        MOV     TH1,#HIGH(-1000)
        MOV     TL1,#LOW(-1000)
        SETB    TR1
        CPL     P1.1
        RETI
        END
```

5. Ngắt cổng nối tiếp.

Mục tiêu: Hiểu cách ngắt cổng nối tiếp.

MCS-51 có 2 nguồn ngắt do cổng nối tiếp: ngắt phát và ngắt thu. Hai nguồn ngắt này xác định bằng các bit RI, TI và dùng chung một địa chỉ ISR nên khi chuyển đến ISR, các cờ ngắt không tự động xóa bằng phần cứng mà phải thực hiện bằng phần mềm: kiểm tra nguyên nhân ngắt (RI hay TI) và xóa bit cờ tương ứng.

Ví dụ:

Viết chương trình khởi động cổng nối tiếp ở chế độ UART 8 bit với tốc độ truyền 4800 bps. Viết ISR cho cổng nối tiếp theo yêu cầu: truyền tuần tự các ký tự từ 'A' đến 'Z' ra cổng nối tiếp đồng thời mỗi lần có ký tự đến cổng nối tiếp thì nhận về và xuất ký tự nhận ra P0 (giả sử tần số thạch anh là 11.0592 MHz).

Giải

Nội dung thanh ghi SCON:

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
0	1	0	1	0	0	0	0
UART 8 bit		Không ở chế độ đa xử lý	Cho phép thu			Không cho phép truyền	

SCON = 50h

- Nội dung thanh ghi TMOD:

GATE1	C/T1	M11	M10	GATE0	C/T0	M01	M00
0	0	1	0	0	0	0	0
Không dùng INT1	Đếm bằng dao động nội	Chế độ 8 bit		Timer 0 không dùng			

TMOD = 0010 0000b (20h)

- Giá trị đếm (theo bảng 4.2): TH1 = -6

- Nội dung thanh ghi IE:

EA	-	ET2	ES	ET1	EX1	ET0	EX0
1	0	0	1	0	0	0	0

IE = 1001 0000b (90h)

Chương trình thực hiện như sau:

```
        ORG 0000H
        LJMP   MAIN
        ORG    0023H        ;Địa chỉ ISR của cổng nối tiếp
        LJMP SERIAL_ISR
MAIN:
        MOV    TMOD,#20h
        MOV    TH1,#(-6)
        MOV    TL1,#(-6)    ; Tốc độ 4800 bps
        SETB   TR1
        MOV    R7,#'A'      ; Ký tự truyền đầu tiên
        MOV    IE,#90h      ; Cho phép ngắt tại công nối tiếp
        SETB   TI           ;Cho phép truyền
        SJMP   $
SERIAL_ISR:
        JNB    RI,TRANSMIT; Nếu không phải ngắt do nhận ký tự thì truyền
        CLR    RI
        MOV    A,SBUF       ; Nhận ký tự
```

```
    MOV   P0,A          ; Xuất ra Port 0
    SJMP  EXITE\SERIAL
TRANSMIT:
    CLR   TI
    MOV   A,R7
    MOV   SBUF,A        ; Truyền ký tự
    INC   R7            ; Qua ký tự kế
    CJNE  R7,#'Z'+1, EXITE\SERIAL     ; Nếu chưa truyền'Z' thì
                                      ; tiếp tục truyền, ngược lại thì
    MOV   R7,#'A'       ; bắt  đầu truyền từ ký tự 'A'
EXITE\SERIAL:
    RETI
    END
```

6. Các cổng ngắt ngoài.

Mục tiêu: Hiểu chức năng của các ngắt ngoài.

- Ngắt ngoài xảy ra khi có mức thấp hoặc có cạnh âm trên chân /INT0 hoặc /INT1. Việc chọn các ngắt thuộc tác động cạnh hay các ngắt loại tác động mức được lập trình thông qua các bit IT0 và IT1 của thanh ghi TCON.
- Vì các chân ngắt ngoài được lấy mẫu một lần ở mỗi chu kỳ máy các ngõ vào này phải được duy trì tối thiểu 12 chu kỳ dao động để đảm bảo rằng việc lấy mẫu là đúng.
- Nếu ngắt ngoài thuộc tác động cạnh nguyên nhân ngắt ngoài phải được duy trì tại chân yêu cầu ở mức cao tối thiểu một chu kỳ và sau đó ở mức thấp tối thiểu một chu kỳ nữa để đảm bảo rằng việc chuyển trạng thái được phát hiện. IE0 và IE1 tự động được xóa khi CPU trỏ tới trình phục vụ ngắt tương ứng.
- Nếu ngắt ngoài thuộc loại tác động mức nguyên nhân ngắt ngoài phải được duy trì trạng thái tích cực cho đến khi ngắt theo yêu cầu thực sự tạo ra. Sau đó nguyên nhân ngắt phải ở trạng thái thụ động trước khi trình phục vụ ngắt được thực thi xong hoặc trước khi có một ngắt khác được tạo ra.
- Thông thường, một công việc được thực thi bên trong trình phục vụ ngắt làm cho nguyên nhân ngắt trả tín hiệu yêu cầu ngắt trở về trạng thái không tích cực.

7. Đồ thị thời gian của ngắt.

Mục tiêu: Biết khoảng thời gian tiêu tốn cho 1 ngắt được kích hoạt.

Thực tế chỉ có 5 ngắt dành cho người dùng trong 8051 nhưng nhiều nhà sản xuất đưa ra các bảng dữ liệu nói rằng có sáu ngắt vì họ tính cả lệnh tái thiết lập lại RESET. Sáu ngắt của 8051 được phân bố như sau:

RESET: Khi chân RESET được kích hoạt từ 8051 nhảy về địa chỉ 0000. Đây là địa chỉ bật lại nguồn.

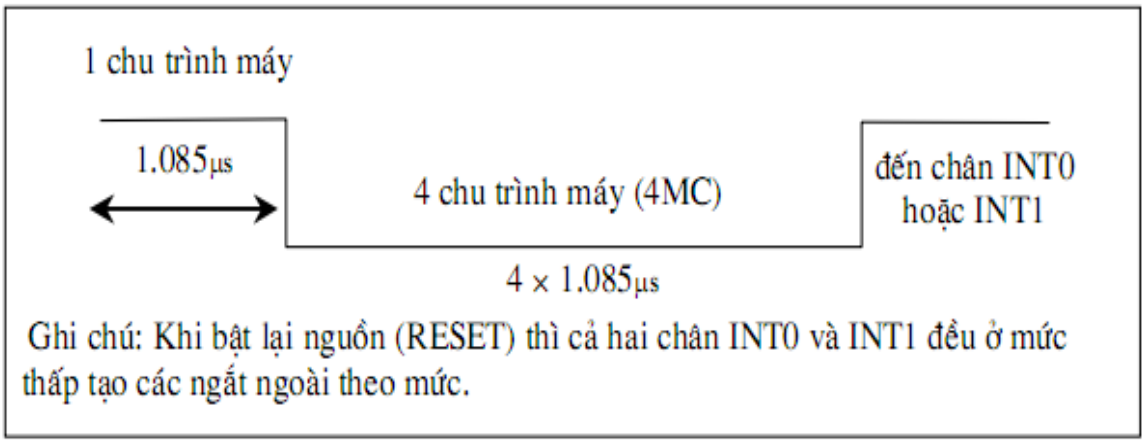
Gồm 2 ngắt dành cho các bộ điịnh thời: 1 cho timer0 và 1 cho Timer1. Địa chỉ của các ngắt này là 000B4 và 001B4 trong bảng vector ngắt dành cho Timer0 và Timer1 tương ứng.

Hai ngắt dành cho các ngắt phần cứng bên ngoài chân 12 (P3.2) và 13 (P3.3) của cổng P3 là các ngắt phần cứng bên ngoài INT0 và INT1 tương ứng. Các ngắt ngoài cũng còn được coi như EX1 và EX2 vị trí nhớ trong bảng vector ngắt của các ngắt ngoài này là 0003H và 0013H gán cho INT0 và INT1 tương ứng.

Truyền thông nối tiếp có một ngắt thuộc về cả thu và phát. Địa chỉ của ngắt này trong bảng vector ngắt là 0023H.

Đồ thị thời gian của ngắt.

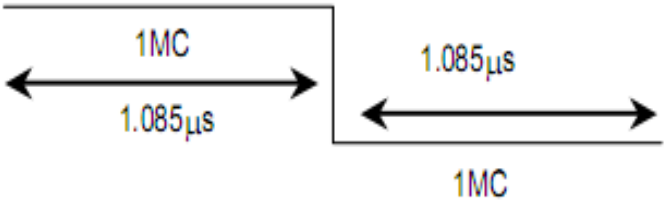
Các chân P3.2 và P3.3 bình thường được dùng cho vào-ra nếu các bít INT0 và INT1 trong thanh ghi IE không được kích hoạt. Sau khi các ngắt phần cứng trong thanh ghi IE được kích hoạt thì bộ vi điều khiển duy trì trích mẫu trên chân INTn đối với tín hiệu mức thấp một lần trong một chu kỳ máy. Theo bảng dữ liệu của nhà sản xuất của bộ vi điều khiển thì "chân ngắt phải được giữ ở mức thấp nhất cho đến khi bắt đầu thực hiện trình phục vụ ngắt ISR. Nếu chân INTn được đưa trở lại mức cao trước khi bắt đầu thực hiện ISR thì sẽ chẳng có ngắt nào xảy ra". Tuy nhiên trong quá trình kích hoạt ngắt theo mức thấp nên nó lại phải đưa lên mức cao trước khi thực hiện lệnh RETI và lại theo bảng dữ liệu của nhà sản xuất thì "nếu chân INTn vẫn ở mức thấp sau khi lệnh RETI của trình phục vụ ngắt thì một ngắt khác lại sẽ được kích hoạt sau lệnh RETI được thực hiện. Do vậy, để bảo đảm việc kích hoạt ngắt phần cứng tại các chân INTn phải khẳng định rằng thời gian tồn tại tín hiệu mức thấp là khoảng 4 chu trình máy và không được hơn.



Ghi chú: Khi bật lại nguồn (RESET) thì cả hai chân INT0 và INT1 đều ở mức thấp tạo các ngắt ngoài theo mức.

Hình 6.5. Thời gian tối thiểu của ngắt theo mức thấp (XTAL = 11,0592MHz)

Trong các ngắt sườn,nguồn ngoài phải giữ ở mức cao tối thiểu là một chu trình máy nữa để đảm bảo bộ vi điều khiển nhìn thấy được sự chuyển dịch từ cao xuống thấp của sườn xung.



Các mức ưu tiên các ngắt khi cấp lại nguồn

Mức ưu tiên cao xuống thấp	
Ngắt ngoài 0	INT0
Ngắt bộ định thời 0	TF0
Ngắt ngoài 1	INT1
Ngắt bộ định thời 1	TF1
Ngắt truyền thông nối tiếp	(RI + TI)

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 ; Xóa (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.*

CÁC BÀI TẬP MỞ RỘNG, NÂNG CAO VÀ GIẢI QUYẾT VẤN ĐỀ

❖ Mục đích:

- Thực hành lập trình ứng dụng trên máy tính, nạp vào vi điều khiển và sử dụng mô hình thí nghiệm để kiểm chứng.
- Điều khiển thiết bị ngoại vi bằng các port của vi điều khiển.
- Thiết kế các ứng dụng diều khiển thực tế có sử dụng ngắt

(Interrrupt).

- So sánh ưu và nhược điểm của các chương trình điều khiển có sử dụng ngắt và không sử dụng ngắt.

❖ Yêu cầu:

- Nắm vững tập lệnh của vi điều khiển MCS 51.
- Tham khảo trước hoạt động của ngắt (Interrrupt) ở các chế độ khác nhau.
- Nắm được phương pháp lập trình và điều khiển có sử dụng ngắt.

Bài 1: Chương trình điều khiển sóng vuông tuần hoàn có tần số 10Hz (sử dụng ngắt Timer) tại chân P0.0 và hiển thị mức logic tại chân này lên LED0 (LED0 được nối với P0.0).

Giải:

1. Trình tự tiến hành thí nghiệm:

1.1. Kết nối thiết bị thí nghiệm:

- Tắt nguồn cấp cho kít thí nghiệm.
- Dùng dây bus 8 nối
- Dùng dây bus

1.2. Vẽ lưu đồ giải thuật và viết chương trình điều khiển

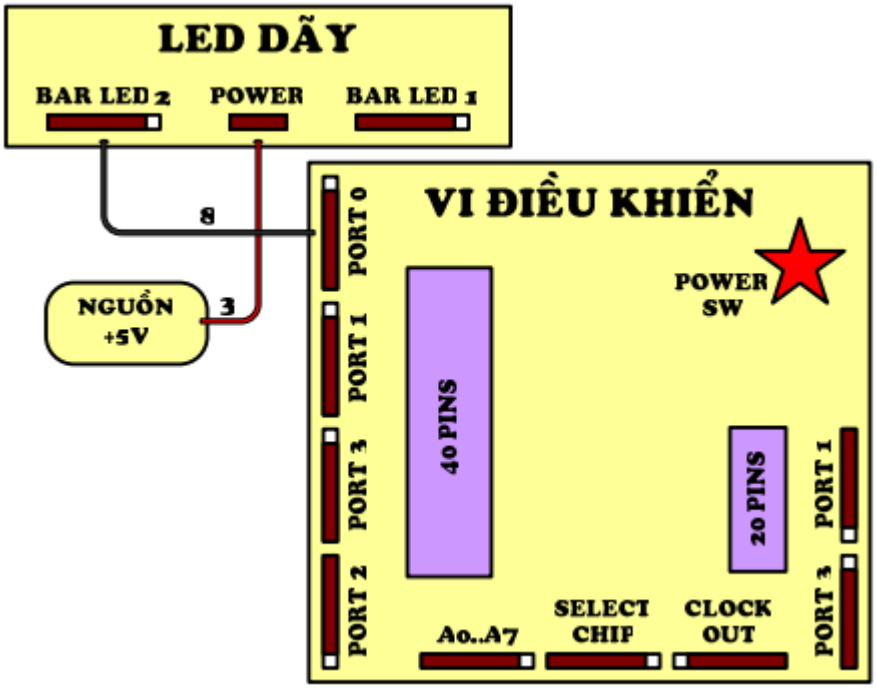
```
        ORG     0000H       ;DIEM NHAP RESET
        DJMP    MAIN
        ORG     0BH         ;DIEM NHAP ISR TIMER0
T0ISR:
        CPL     P0.0        ;DAO TRANG THAI P0.0 (TAO XUNG)
        RETI
        ORG     30H         ;DIEM NHAP CHUONG TRINH CHINH
MAIN:
        MOV     TMOD,#01H;TIMER0 LA TIMER 16 BIT
        MOV     TH0,#(-50000);THỜI GIAN TRE -50MS (THOI GIAN
                                XUNG O MUC THAP HOAC MUC
CAO)
        MOV     TL0,#CHU KY -2x50 =100 MS LA F = 10Hz
        SETB    TR0         ;CHO TIMER BAT DAU CHAY
```

```
        MOV     IE,#82H     ;CHO PHEP NGAT TIMER0 HOAT DONG
        SJMP    $           ;DUNG YEN
        END
```

1.3. Lưu chương trình và biên dịch chương trình.

1.4. Kiểm tra lỗi và hiệu chỉnh lỗi nếu có.

1.5. Gắn chip vi điều khiển thí nghiệm vào socket tương ứng trên khối nạp chip và bật nguồn cho khối nạp chip hoạt động.



1.6. Vẽ lưu đồ giải thuật và viết chương trình điều khiển

```
        ORG     0000H       ;DIEM NHAP RESET
        DJMP    MAIN
        ORG     0BH         ;DIEM NHAP ISR TIMER0
T0ISR:
        CPL     P0.0        ;DAO TRANG THAI P0.0 (TAO XUNG)
        RETI
        ORG     30H         ;DIEM NHAP CHUONG TRINH CHINH
MAIN:
        MOV     TMOD,#01H;TIMER0 LA TIMER 16 BIT
        MOV     TH0,#(-50000);THỜI GIAN TRE -50MS (THOI GIAN
                              XUNG O MUC THAP HOAC MUC
CAO)
        MOV     TL0,#CHU KY -2x50 =100 MS LA F = 10Hz
        SETB    TR0         ;CHO TIMER BAT DAU CHAY
        MOV     IE,#82H     ;CHO PHEP NGAT TIMER0 HOAT DONG
```

```
SJMP    $           ;DUNG YEN
END
```

1.7. Lưu chương trình và biên dịch chương trình.

1.8. Kiểm tra lỗi và hiệu chỉnh lỗi nếu có.

1.9. Gắn chip vi điều khiển thí nghiệm vào socket tương ứng trên khối nạp chip và bật nguồn cho khối nạp chip hoạt động.

1.10. Nạp chương trình vào vi điều khiển.

1.11. Sử dụng vi điều khiển vừa nạp gắn vào socket tương ứng tren khối vi điều khiển.

1.12. Bật nguồn cho mô hình thí nghiệm. Quan sát kết quả hoạt động, nếu kết quả hoạt động không đúng yêu cầu của đề bài thì phải quay lại kiểm tra việc kết nối mạch, hiệu chỉnh chương trình và làm lại các bước từ bước 3 đến bước 9.

Bài 2 (thực hành ở lớp): Chương trình điều khiển đếm số xung tại chân INT0 (sử dụng ngắt ngoài) và hiển thi số xung này (tối đa là 225 lần) lên ba LED 7 đoạn (LED7 – LED4 được nối với Port1, PULSE được nối với chân INT0).

1. Viết đoạn chương trình theo yêu cầu:
   - Khởi động cổng nối tiếp ở chế độ UART 8 bit với tốc độ truyền 4800 bps.
   - Định thời 1s thì đọc dữ liệu từ P1, lưu vào ô nhớ 30h và xuất dữ liệu vừa đọc ra cổng nối tiếp.
2. Viết đoạn chương trình theo yêu cầu:
   - Khởi động cổng nối tiếp ở chế độ UART 9 bit với tốc độ truyền 9600 bps.
   - Khi có ngắt xảy ra tại ngắt ngoài 0 thì xuất dữ liệu tại ô nhớ 30h ra cổng nối tiếp trong đó bit truyền thứ 9 là bit parity.

THỰC HANH VIÊT CHƯƠNG TRINH:

Bai 1: Viết chương trình điêù khiển ma trận 8x8?

Chương trình mẫu:

```
ORG   000H
  MOV     R2,#0FEH
MAIN:
  MOV     DPTR,#BITMAP
X1:
```

```
   MOV       A,#0
   MOVC      A,@A+DPTR
   MOV       P0,A
   MOV       P2,R2
   CALL      DELAY
   MOV       P2,#0FFH       ;CHONG LEM
   INC       DPTR
   MOV       A,R2
   RL        A
   MOV       R2,A
   CJNE      A,#0FEH,X1
   JMP       MAIN
;*****************************************
DELAY:
   MOV    R7,#200
   DJNZ   R7,$
RET
;*****************************************
BITMAP:
DB    0F8H,24H,22H,21H,21H,22H,24H,0F8H
END
```

Bai 2: Viêt chương trình điêù khiển đeìn giao thông ở nga˜tư?

Chương trình mâũ:

```
;*******************************************************
;Chuong trinh dieu khien den giao thong
;*******************************************************
BGIAY      EQU    R2
GIAY       EQU    R3
X1_D2      EQU    01111011B
V1_D2      EQU    10111011B
D1_X2      EQU    11011110B
D1_V2      EQU    11011101B
D1_D2      EQU    11011011B
;*******************************************************
   ORG    000H
   JMP    MAIN
```

```
   ORG   00BH
   JMP   NGAT_T0
MAIN:
   MOV   TMOD,#01H          ;T0 CHAY XUNG NOI, MODE1
   MOV   TL0,#LOW(-50000)
   MOV   TH0,#HIGH(-50000)
   CLR   TF0
   SETB  TR0
   MOV   IE,#82H            ;T0 DUOC PHEP NGAT KHI TRAN
   MOV   BGIAY,#0
;=====================================
TD: MOV   GIAY,#10
   MOV    P0,#X1_D2
   CJNE   GIAY,#0FFH,$        ;CHO DEN X1 + V2 SANG 10s.
   MOV    P0,#V1_D2
   MOV    GIAY,#3
   CJNE   GIAY,#0FFH,$        ;CHO DEN V1 + D2 SANG 3s.
   MOV    P0,#D1_D2
   MOV    GIAY,#5
   CJNE   GIAY,#0FFH,$        ;CHO DEN D1 + D2 SANG 5 GIAY
   MOV    P0,#D1_X2
   MOV    GIAY,#15
   CJNE   GIAY,#0FFH,$        ;CHO DEN D1 + X2 SANG 15s.
   MOV    P0,#D1_V2
   MOV    GIAY,#3
   CJNE   GIAY,#0FFH,$        ;CHO DEN D1 + V2 SANG 3s.
   MOV    P0,#D1_D2
   MOV    GIAY,#5
   CJNE   GIAY,#0FFH,$        ;CHO DEN D1 + D2 SANG 5 GIAY
   JMP    TD
;*************************************************
NGAT_T0:
   MOV    TL0,#LOW(-50000)
   MOV    TH0,#HIGH(-50000)
   CLR    TF0
   INC    BGIAY
   CJNE   BGIAY,#20,EXIT1     ;20  x 50ms = 1000ms = 1s
   MOV    BGIAY,#0
```

```
   DEC     GIAY
EXIT1:
RETI
END
```

Bai 3: Viết chương trình điều khiển phím ma trận phím 4x4?

Chương trình mẫu:

```
;*****************************************************************
;KEYPAD 4X4
;*****************************************************************
MAPHIM       EQU    30H
MACOT        EQU    31H
CHONGDOI     EQU    32H
TEMP         EQU    33H
;*****************************************************************
      ORG    000H
      MOV    A,#0                    ;GIA TRI HIEN THI TREN LED 7
DOAN BAN DAU
X1:   CALL   GIAIMA_HIENTHI
X2:   CALL   QUETPHIM
      CJNE   A,#0FFH,X1
      JMP    X2
;*****************************************************************
GIAIMA_HIENTHI:
   MOV     DPTR,#MA7DOAN
   MOVC    A,@A+DPTR
   MOV     P0,A
RET
;*****************************************************************
QUETPHIM:
       MOV    CHONGDOI,#50          ;CHONG DOI KHI NHAN PHIM
X3:    CALL   DOPHIM
       JC     EXIT                  ;C = 1 LA KO CO PHIM NHAN
       DJNZ   CHONGDOI,X3
       MOV    TEMP,A                ;LUU TAM MA PHIM

X4:    MOV    CHONGDOI,#50          ;CHONG DOI KHI NHA PHIM
X5:    CALL   DOPHIM
       JNC    X4                    ;C = 0 LA CO PHIM NHAN
       DJNZ   CHONGDOI,X5
EXIT:
```

```
      MOV    A,TEMP
RET
;*************************************************************
DOPHIM:
      MOV    MAPHIM,#0
      MOV    MACOT,#0EFH          ;MA COT 1
X6:   MOV    P3,MACOT
      NOP                         ;CHO NHAN PHIM
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      MOV    A,P3
      ANL    A,#0FH               ;LAY MA HANG
      CJNE   A,#0FH,LAPMA         ;KIEM TRA CO NHAN PHIM KO
;*****ko co phim nhan
      MOV    A,MAPHIM
      ADD    A,#4
      MOV    MAPHIM,A             ;MAPHIM = MAPHIM + 4
      MOV    A,MACOT
      RL     A                    ;DICH SANG COT KE
      MOV    MACOT,A
      CJNE   A,#0FEH,X6           ;KIEM TRA DA DICH DEN COT THU
4 CHUA ?
      SETB   C                    ;KO CO PHIM BAM
      MOV    A,#0FFH
RET
;*****co phim nhan
LAPMA:
     RRC    A
     JNC    X7
     INC    MAPHIM               ;MA PHIM = MA PHIM + 1
     JMP    LAPMA
X7:  MOV    A,MAPHIM             ;DA CO PHIM AN
     CLR    C
RET
;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
MA7DOAN:
```

```
DB  0F8H, 80H, 90H, 0B0H, 99H, 92H, 82H, 0C0H, 0F9H, 0A4H, 0B0H,
0FFH, 0FFH, 0C0H
END
```

Bai 4: Viết chương trình điều khiển đồng hồ số dùng timer tạo thời gian trễ?

Chương trình mẫu:

```
GIAY      EQU    R2
PHUT     EQU    R3
GIO       EQU    R4
BIEN_GIAY    EQU    R5
   ORG    000H
   JMP    MAIN
   ORG    00BH
   JMP    NGAT_T0
MAIN:
   MOV    TMOD,#01H
   MOV    TH0,#HIGH(-50000)
   MOV    TL0,#LOW(-50000)
   CLR    TF0
   SETB    TR0
   MOV    IE,#82H
LOOP:
   MOV    GIO,#0
X4:   MOV    PHUT,#0
X3:   MOV    GIAY,#0
X2:   MOV    BIEN_GIAY,#0
   CALL    HEX_BCD
   CALL    BCD_7DOAN
X1:   CALL    HIENTHI
   CJNE    BIEN_GIAY,#20,X1
   INC    GIAY
   CJNE    GIAY,#60,X2
   INC    PHUT
   CJNE    PHUT,#60,X3
   INC    GIO
   CJNE    GIO,#24,X4
   JMP    LOOP
;*********************************************
NGAT_T0:
   MOV    TL0,#LOW(-50000)
```

```
   MOV    TH0,#HIGH(-50000)
   INC    BIEN_GIAY
RETI
;*********************************************
HEX_BCD:
   MOV     A,GIAY
   MOV     B,#10
   DIV      AB
   MOV    10H,B          ;LUU SO HANG DV GIAY
   MOV    11H,A           ;LUU SO HANG CHUC GIAY
   MOV    A,PHUT
   MOV    B,#10
   DIV      AB
   MOV    12H,B          ;LUU SO HANG DV PHUT
   MOV    13H,A          ;LUU SO HANG CHUC PHUT
   MOV    A,GIO
   MOV    B,#10
   DIV      AB
   MOV    14H,B       ;LUU SO HANG DV GIO
   MOV    15H,A       ;LUU SO HANG CHUC GIO
RET
;*********************************************
BCD_7DOAN:
   MOV    DPTR,#MA7DOAN
   MOV    R0,#10H
   MOV    R1,#20H
GM1: MOV    R6,#2
GM2: MOV    A,@R0
   MOVC  A,@A+DPTR
   MOV    @R1,A
   INC     R0
   INC     R1
   DJNZ    R6,GM2
   MOV    @R1,#0BFH
   INC      R1
   CJNE    R0,#16H,GM1
RET
;*********************************************
HIENTHI:
   MOV     R0,#20H
   MOV     A,#80H
```

```
HT: MOV   P0,@R0
   MOV    P2,A
   CALL   DELAY
   MOV    P2,#00H       ;CHONG LEM
   INC    R0
   RR     A
   CJNE   A,#80H,HT
RET
;*********************************************
DELAY:
   MOV   R7,#0FFH
   DJNZ   R7,$
RET
;*********************************************
MA7DOAN:
DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
END
```

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

- + Về kiến thức: Trình bày được tác dụng thực tế của một hệ thống được điều khiển bằng tín hiệu ngắt theo nội dung đã học.

+ Về kỹ năng:

- Thực hiện tổ chức ngắt và cơ chế thực hiện chương trình phục vụ ngắt của 8051 đúng yêu cầu kỹ thuật.
- Thực hiện tổ chức ngắt đạt yêu cầu kỹ thuật.
- Lắp ráp các mạch ứng dụng từng phần do giáo viên đề ra.
- Thực hiện viết các chương trình theo yêu cầu cho trước

+ Thái độ: Đánh giá phong cách, thái độ học tập

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm

+ Về kỹ năng: Đánh giá kỹ năng thực hành Mỗi sinh viên, hoặc mỗi nhóm học viên thực hiện công việc theo yêu cầu của giáo viên.Tiêu chí đánh giá theo các nội dung:

- Độ chính xác của công việc
- Tính thẩm mỹ của mạch điện

- Độ an toàn trên mạch điện
- Thời gian thực hiện công việc
- Độ chính xác theo yêu cầu kỹ thuật

+ Thái độ: Tỉ mỉ, cẩn thận, chính xác.

BAI 7
PHÂN MÊM HỢP NGƯ˜
Mã bài: MĐ24-07

Giới thiệu:

Vi điều khiển là một IC lập trình, vì vậy Vi điều khiển cần được lập trình trước khi sử dụng. Mỗi phần cứng nhất định phải có chương trình phù hợp kèm theo, do đó trước khi viết chương trình đòi hỏi người viết phải nắm bắt được cấu tạo phần cứng và các yêu cầu mà mạch điện cần thực hiện.

Chương trình cho Vi điều khiển có thể viết bằng C++,C,Visual Basic, hoặc bằng các ngôn ngữ cấp cao khác. Tuy nhiên hợp ngữ Assembler được đa số người dùng Vi điều khiển sử dụng để lập trình, vì lí do này chúng tơi chọn Assembly để hướng dẫn viết chương trình cho Vi điều khiển. Assembly là một ngôn ngữ cấp thấp, trong đó mỗi câu lệnh chương trình tương ứng với một chỉ lệnh mà bộ xử lý có thể thực hiện được. Ưu điểm của hợp ngữ Assembly là: mã gọn, ít chiếm dung lượng bộ nhớ, hoạt động với tốc độ nhanh, và nó có hiệu suất tốt hơn so với các chương trình viết bằng ngôn ngữ bậc cao khác.

Mục tiêu của bài:

- Trình bày được sự cần thiết và cơ chế hoạt động của trình dịch hợp ngữ theo nội dung đã học.
- Trình bày được cấu trúc chung của chương trình hợp ngữ theo nội dung đã học.
- Thực hiện viết chương trình tổ chức lớn bằng cách phân chia thành các mô đun chương trình đúng qui trình kỹ thuật.
- Viết được chương trình điều khiển theo yêu cầu.

Nội dung chính:

1. Mở đầu

Mục tiêu: Biết được phần mềm hợp ngữ là gì.

1.1. Khái niệm.

- Vì các lệnh của Vi điều khiển (VĐK) có dạng số nhị phân quá dài và khó nhớ, hơn nữa việc gỡ lỗi khi chương trình phát sinh lỗi rất phức tạp và khó khăn. Khó khăn này được giải quyết với sự hỗ trợ của máy vi tính, người viết chương trình có thể viết chương trình cho vi điều khiển bằng các ngôn ngữ lập trình cấp cao, sau khi việc viết chương trình được hồn tất, các trình biên dịch sẽ chuyển các câu lệnh cấp cao thành mã máy một cách tự động. Các mã máy này sau đó được đưa (nạp) vào bộ nhớ ROM của VĐK, Vi điều khiển sẽ tìm đến đọc các lệnh từ ROM để thực hiện chương trình.

Bản thân máy tính không thể thực hiện các mã máy này vì chúng không phù hợp với phần cứng máy tính, muốn thực hiện phải có các chương trình mô phỏng dành riêng.

- Hợp ngữ (assembly language) thay thế những mã nhị phân bằng các từ gợi nhớ để lập trình dễ dàng hơn. Máy tính không hiểu hợp ngữ do đó trình biên dịch hợp ngữ Assembler và trình liên kết Linker có chức năng dịch những chương trình viết bằng hợp ngữ thành ngôn ngữ máy.

1.2. Một số khái niệm.

- *Chương trình hợp ngữ (Assembly Language Program):* Là chương trình được viết bằng cách dùng các nhãn, các từ gợi nhớ,…, trong đó mỗi phát biểu tương ứng với một lệnh của ngôn ngữ máy. Chương trình viết bằng hợp ngữ gọi là mã nguồn và chương trình này không thể thực thi mà nhằm giúp người lập trình đọc hiểu những gì vi điều khiển thực hiện và gỡ rối một cách dễ dàng. Assembly là một ngôn ngữ lập trình cấp thấp gần với ngôn ngữ máy, chương trình sau khi viết bằng assembly cần được chuyển đổi qua mã lệnh (hay còngọi là mã máy) của vi điều khiển, quá trình chuyển đổi được thực hiện bằng chương trình dịch Assembler. Các mã lệnh sau đó được nạp vào Rom của vi điều khiển để thực hiện chương trình. Chương trình dịch Assembler được dùng phổ biến hiện nay là chương trình Macro Assembler sử dụng trên Dos.
- *Chương trình ngôn ngữ máy (Machine Language Program):* Là chương trình gồm các mã nhị phân tương ứng với 1 lệnh của vi xử lý. Các chương trình viết bằng ngôn ngữ máy thường được gọi là mã đối tượng (object code) và thực thi được.
- *Chương trình Assembler:* Là chương trình dịch một chương trình viết bằng hợp ngữ sang chương trình ngôn ngữ máy. Chương trình ngôn ngữ máy có thể ở dạng tuyệt đối hoặc ở dạng tái định vị. *Chương trình Linker:* Là chương trình kết hợp các chương trình đối tượng tái định vị được để tạo ra chương trình đối tượng tuyệt đối để thực thi được.
- *Segment:* Là một đơn vị bộ nhớ chứa mã lệnh hoặc chứa dữ liệu. Một segment có thể ở dạng tuyệt đối hoặc tái định vị được. Segment tái định vị được sẽ có tên, kiểu và các thuộc tính cho phép chương trình linker kết hợp nó với các phần của các đoạn khác nếu cần để định vị đúng đoạn. Segment ở dạng tuyệt đối không có tên và không thể kết hợp được với các đoạn khác.
- *Module:* Chứa1 hay nhiều segment hoặc một phần segment. Một module có tên do người sử dụng đặt. Những định nghĩa module xác định tầm của các ký hiệu cục bộ. Một tập tin đối tượng chứa 1 hay nhiều

module. Một module được xem như là một tập tin trong nhiều tình huống.
- *Chương trình:* Gồm nhiều module tuyệt đối, trộn tất cả các đoạn tuyệt đối và tái định vị được từ tất cả các module nhập. Một chương trình chỉ chứa các mã nhị phân cho các chỉ thị mà máy tính hiểu. Vi điều khiển MSC-51 đều có chung một tập lệnh, các Vi điều khiển được cải tiến sau này thường ít thay đổi hoặc mở rộng tập lệnh mà chú trọng phát triển phần cứng.

- Để soạn thảo chương trình có thể sử dụng Notepal hoặc bất cứ chương trình soạn thảo có sử dụng bộ kí tự chuẩn ASCII và lưu tên đuơi như sau: "tên.asm". Ngoài ra có thể sử dụng các phần mềm hỗ trợ soạn thảo dành riêng cho vi điều khiển đã tích hợp sẵn chương trình dịch Assembler.

2. Hoạt động của trình biên dịch Assembler.

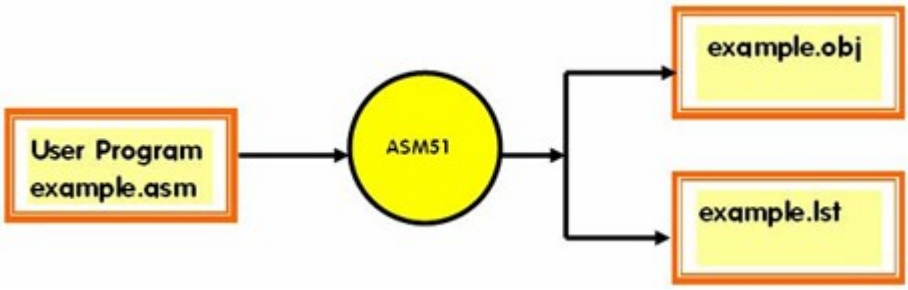
Mục tiêu:Hiểu được cách biên dịch của chương trình Assembler.

- ASM51 là assembler chạy trên máy tính do Intel cung cấp để biên dịch cho họ MCS51. Cách dùng ASM51 để biên dịch chương trình viết bằng hợp ngữ như sau: từ dấu nháy ở DOS hay ở Win commander ta thực hiện theo cú pháp:

ASM51 source_file[assembler_control]

Trong đó source_file là tên tập tin nguồn viết bằng hợp ngữ, assembler_control là những điều khiển assembler.

Assembler nhận tập tin nguồn (ví dụ "example.asm") sẽø tạo ra tập tin đối tượng ("example.Obj") và tập tin kiểu liệt kê ("example.lst") như hình 7.1



Hình 7.1. Biên dịch một chương trình nguồn

- Tất cả các chương trình biên dịch đều quét chương trình nguồn 2 lần để thực hiện dịch ra ngôn ngữ máy nên được gọi là Assembler hai bước. Assembler sử dụng bộ đếm vị trí làm địa chỉ của các lệnh và các giá trị cho các nhãn. Hoạt động của từng bước được mô tả như sau:

Bước 1: nhận diện các nhãn và các kí hiệu trong chương trình nguồn, tính toán các địa chỉ tương đối của chúng và cất vào bảng kí hiệu. Bảng kí hiệu chứa những vị trí nhãn, những kí hiệu và những giá trị tương ứng của chúng. Bộ đếm vị trí: lưu giữ địa chỉ của các lệnh và giá trị của nhãn chương trình.

Bước 2: tạo ra tập tin đối tượng và tập tin liệt kê:

- Các từ gợi nhớ.
- Các toán hạng được định vị và đặt sau các mã lệnh.
- Các giá trị kí hiệu được truy cập để tính đúng dữ liệu hoặc địa chỉ.
- Cho phép tham chiếu tới.

- Tập tin tái định vị chứa thông tin cần cho linker và định vị.
- Tập tin liệt kê chứa chương trình nguồn và mã lệnh.

3. Cấu trúc chung chương trình hợp ngữ cho 8051

Mục tiêu: Hiểu được cấu trúc chương trình hợp ngữ.

3.1. Các thành phần cơ bản của ngôn ngữ Assembly.

- Lables: Nhãn – đánh dấu cho một đoạn lệnh.
- Orders: Lệnh.
- Directives: Định hướng chương trình dịch
- Comments: Các lời chú thích

Một dòng lệnh trong chương trình hợp ngữ gồm có các trường sau:

Tên	Lệnh	Toán hạng	Chú thích
A:	Mov	AH, 10h	; Đưa giá trị 10h vào thanh ghi AH

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.

3.2. Cấu trúc chương trình dữ liệu.

- Những lệnh của vi xử lý.
- Những chỉ dẫn assembler (Assembler Directive).
- Những điều khiển Assembler.
- Các chú thích.

❖ Cú pháp lệnh của vi xử lý như sau:

[label:] mnemonic [operand] [,operand] […] [;comment]

Trong đó label là nhãn – theo sau bởi dấu hai chấm ":", mnemonic là từ gợi

nhớ của lệnh, operand là toán hạng tuỳ thuộc vào lệnh có một hoặc nhiều toán hạng hoặc không có toán hạng, cuối cùng là chú thích cho lệnh đó – đi sau dấu chấm phẩy ";". Kí hiệu là tên được định nghĩa để biểu diễn một giá trị, khối văn bản, địa chỉ hoặc tên thanh ghi và cũng có thể biểu diễn các hằng số và các biểu thức. Các tên của các kí hiệu cho phép tối đa 31 kí tự với kí tự đầu phải là chữ hoặc dấu "?" hoặc "-", và theo sau phải là các chữ, số, "?" hoặc "-". Các kí hiệu có thể sử dụng các kí tự in hoa hay thường không phân biệt. Chú ý các từ kí hiệu là các từ đã sử dụng nên người lập trình không được dùng chúng làm kí hiệu cho các mục đích khác.

Ví dụ 1: Bdn EQU R2

Nhãn là một loại kí hiệu dùng để định nghĩa vị trí trong chương trình:

- Tên nhãn tượng trưng cho một địa chỉ.
- Vùng văn bản thứ nhất trong dòng hợp ngữ
- Theo sau nhãn là dấu hai chấm ":"
- Trên một hàng chỉ có thể định nghĩa một nhãn.
- Không được đặt tên các nhãn trùng nhau

Ví dụ 2:

```
Label1:    MOV    R2,#35h
```

Mnemonic là tất cả các từ gợi nhớ cho tất cả các lệnh và các chỉ dẫn assembler:

- Mnemonic cho lệnh: ADD, SUB, MUL, DIV, MOV,…
- Mnemonic cho chỉ dẫn assembler: org, equ, db, bit,…

Toán hạng operand là đối số hoặc biểu thức được đặt tả cùng với lệnh hoặc chỉ dẫn assembler, toán hạng có thể là địa chỉ hoặc dữ liệu.

Bài tập: Phân tích 2 ví dụ sau:

```
Ví dụ 3:    Ngat    EQU     R2
Ví dụ 4:            MOV     R0,#75H
                    NOP
                    RET
```

Trong hợp ngữ ASM51 có các kiểu toán hạng bảng 7.1:

Kiểu toán hạng	Mô tả
Dữ liệu tức thời	Kí hiệu hoặc hằng được dùng làm giá trị số
Địa chỉ bit trực tiếp	Kí hiệu hoặc hằng tham chiếu địa chỉ bit
Địa chỉ chương trình	Kí hiệu hoặc hằng tham chiếu địa chỉ mã

Địa chỉ dữ liệu trực tiếp	Kí hiệu hoặc hằng tham chiếu địa chỉ dữ liệu
Địa chỉ gián tiếp	Tham chiếu gián tiếp đến bộ nhớ, có thể là offset
Kí hiệu assembler đặc biệt	Tên thanh ghi.

- Dữ liệu tức thời (immediate data): Là biểu thức số được mã hóa như một phần trong lệnh ngôn ngữ máy. Toán hạng này phải có kí hiệu "#" đi trước.

Ví dụ 5: MOV R0,#30. Trong ví dụ này 30 là dữ liệu tức thời.

Địa chỉ bit trực tiếp: (direct bit address): Kiểu này dùng để truy cập các bit của các ô nhớ cho phép truy xuất bit.

Có 3 cách để định địa chỉ bit:

Truy xuất trực tiếp địa chỉ bit.

Truy xuất toán tử chấm (byte.bit).

Kí hiệu assembler được định nghĩa trước.

```
Ví dụ 6:   SETB   00H        ;bit có địa chỉ 00H
           CLR    ACC.7      ;xóa bit thứ 7 của thanh ghi A
           CLR    EA         ;xóa bit ngắt tổn cục
```

Địa chỉ chương trình: (program address): Là toán hạng của lệnh nhảy. Lệnh nhảy tương đối: trong kiểu lệnh này toán hạng này có độ dài 8 bit được xem là offset sử dụng cho lệnh nhảy không điều kiện sjmp và lệnh nhảy có điều kiện. Lệnh nhảy và lệnh gọi tuyệt đối: trong kiểu lệnh này toán hạng này có độ dài 11 bit dùng để quản lý trang bộ nhớ cho lệnh AJMP và ACALL. Lệnh nhảy và lệnh gọi có địa chỉ dài: trong kiểu lệnh này toán hạng này có độ dài 16 bit dùng để quản lý toàn bộ bộ nhớ cho lệnh LJMP và LCALL.

- Nhảy và gọi generic:

Lệnh JMP có thể được dịch hợp thành lệnh SJMP, AJMP hoặc LJMP.

Lệnh Call có thể được dịch hợp thành lệnh ACALL hoặc LCALL.

Người lập trình không cần quan tâm đến địa chỉ thật khi nhảy hay gọi.

- Quy tắc chuyển thành tuỳ thuộc vào assembler:

Lệnh SJMP: không có tham chiếu tới và địa chỉ đích trong vùng

-128 byte so với địa chỉ của lệnh kế.

Lệnh AJMP/ACALL: không có tham chiếu tới và địa chỉ đích trong vùng nhớ cùng khối 2 KByte so với lệnh kế.

- Lệnh AJMP/ACALL: có tham chiếu tới địa chỉ đích trong vùng nhớ 64Kbyte.
- Địa chỉ dữ liệu trực tiếp (direct data address): Địa chỉ này dùng để truy xuất bộ nhớ dữ liệu nội từ có địa chỉ 00H đến 7FH và các vùng nhớ chứa các thanh ghi đặc biệt từ 80H đến FFH. Các kí hiệu được định nghĩa đều có thể sử dụng được cho các thanh ghi chức năng.

Ví dụ 7: Hai lệnh sau là tương đương:

```
MOV A,90H
MOV A,P1
```

- Địa chỉ dữ liệu gián tiếp (indirect data address): Kiểu này dùng các thanh ghi để chứa địa chỉ của các ô nhớ cần truy xuất dữ liệu. Các thanh ghi sử dụng cho kiểu này là thanh ghi R0, R1, DPTR và PC. Các kí hiệu đặc biệt của assembler: Các kí hiệu này dùng cho cách định địa chỉ dùng thanh ghi như A, DPTR, R0 đến R7, PC, cờ C và cặp thanh ghi AB. Kí hiệu dấu "$" dùng để tham chiếu đến giá trị hiện hành của bộ đếm vị trí.

Ví dụ 8: hai lệnh sau là tương đương:

```
WAIT:   JNB   RI,WAIT
        JNB   RI,$
```

Kí hiệu ";" đi sau nó là các chú thích

3.2.1. 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 2byte sử dụng DW để khai báo biến kiểu nguyên

```
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
```

3.2.2. 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

MO  R3, #count          ; Khi thực hiện lệnh "MOV R3,
                        ;thì thanh ghi R3 sẽ được nạp giá trị 25
```

3.2.3. Cấu trúc một chương trình hợp ngữ

```
        ORG 0000h; Đặt lệnh LJMP main tại địa chỉ
        LJMP main; 0000h (địa chỉ bắt đầu khi reset AT89C51)

        ORG 0030h; Vùng địa chỉ 0003h – 002Fh
Main:   ; dùng để chứa các chương trình phục vụ ngắt

        CALL Subname

        ;---------------
Subname:

        …
        …
        RET
            END ; kết thúc chương trình
```

Ví dụ 9:

```
    ORG 00H         ;(con trỏ chương trình bắt đầu từ 00h)
    LJMP MAIN       ; nhảy tới vị trí có nhãn là MAIN)
    ORG 0030H       ; (vị trí bắt đầu chương trình chính MAIN):
MAIN:
    MOV R1,#10      ;(nạp cho R1 giá trị là 10).
LAP1:
    DJNZ R1,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
```

3.2.4. Chương trình con.

```
Nhãn:
..... Các câu lệnh
.....

RET
```

Ví dụ 10:

```
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
```

4. Tính biểu thức trong khi hợp dịch.

Mục tiêu: Hiểu được các biểu thức và các toán tử trong chương trình.

Ký hiệu	Thực hiện	Ví dụ	Kết quả
+	Cộng	10+5	15
-	Trừ	28-17	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 7.2. Các toán tử

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	ACA	ADD	JZ	LCAL	LE	LJMP
ADD	AJM	AND	ANL	LOW	LT	MOD	MOV
AR0	AR1	AR2	AR3	MOV	MOV	MUL	NE
AR4	AR5	AR6	AR7	NOP	NOT	OR	ORG

BIT	BSE	C	CAL	ORL	PC	POP	PUSH
CJNE	CLR	COD	CPL	R0	R1	R2	R3
CSE	DA	DAT	DB	R4	R5	R6	R7
DBIT	DEC	DIV	DJN	RET	RETI	RL	RLC
DPT	DS	DSEG	DW	RR	RRC	SET	SETB
END	EQ	EQU	GE	SHL	SHR	SJMP	SUBB
GT	HIG	IDAT	INC	SWA	USIN	XCH	XCHD
ISEG	JB	JBC	JC	XDA	XOR	XRL	XSEG
JMP	JNB	JNC	JNZ	JZ	LCAL	LE	LJMP
LOW	LT	MOD	MOV				

4.1.Khái niệm các biểu thức và toán tử

- Toán tử được dùng để kết hợp và so sánh các toán hạng trong chương trình hợp ngữ.
- Biểu thức dùng để kết hợp các số, các chuỗi ký tự, các ký hiệu và các toán tử để tính toán ra số nhị phân 16 bit. Dùng biểu thức trong lập trình sẽ giúp cho chương trình dễ đọc hơn và uyển chuyển hơn.
- Các toán hạng gồm có: số, ký tự, chuỗi ký tự và bộ đếm vị trí.
- Các toán tử gồm có: toán tử số học, toán tử nhị phân, toán tử quan hệ và các toán tử khác.

Số: có thể được sử dụng là:

- Số thập lục phân (hexadecimal = hex, có cơ số 16): H, h.
- Số thập phân (decimal, có cơ số 10): D, d hoặc không cần ghi.
- Số bát phân (octal, có cơ số 8): O, o, Q, q.
- Số nhị phân (binary, có cơ số 2): B, b.

Chú ý: Với số hex nếu kí tự số hex đầu tiên bên trái là chữ (từ A đến F) thì phải có thêm kí tự số 0 ở trước.

Ví dụ 11: lệnh nạp dữ liệu F4H vào thanh ghi R0

MOV R0,#0F4H

❖ *Ký tự:* Cho phép tối đa 2 ký tự nằm giữa 2 dấu nháy (‘) có thể được dùng làm toán hạng trong biểu thức.

Ví dụ 12*:* ‘A’ có giá trị tương đương 0041H (bảng mã ASCII)

‘AB’ có giá trị tương đương 4142H

'a' có giá trị tương đương là 0061H
'ab' có giá trị tương đương 6162H

Chúng ta cũng có thể sử dụng ký tự làm toán hạng cho dữ liệu tức thời.

Ví dụ 13:

```
MOV     R0,#'0'
```

❖ Chuỗi ký tự (character string):

Có thể kết hợp với chỉ dẫn DB để định nghĩa các thông báo trong chương trình hợp ngữ.

Ví dụ 13: kytu DB 'a and b'

Chỉ dẫn trên sẽ tạo ra một vùng nhớ dữ liệu chứa các mã ASCII tương ứng là 50H (chữ P), 72H (chữ r), 65H (chữ e), …, lưu vào vùng nhớ bắt đầu từ địa chỉ kytu.

❖ Bộ đếm vị trí (location counter):

Dùng để xác định địa chỉ của từng mã lệnh trong chương trình biên dịch tuỳ thuộc vào chỉ dẫn ORG. Ký tự '$' sẽ trả về giá trị hiện hành của bộ đếm vị trí.

Ví dụ 14: LOC OBJ LINE SOURCE

Giải:

```
ORG   0000h
start:    INC     A
          JMP     start
ORG       start+200
          JMP     start
          JMP     finish
          CALL    delay
finish:   DEC     A
          JMP     start
delay:    MOV     r7,#0
          RET
          END
```

4.2. Các toán tử số học (arithetic operation).

- Toán tử cộng "+" expr + expr
- Toán tử trừ "-" expr - expr
- Toán tử nhân "×" expr * expr
- Toán tử chia "/" expr / expr
- Toán tử chia lấy phần dư "mod" expr MOD expr

Trong đó expr là biểu thức.

Ví dụ 15:

```
MOV   A,#10 + 10H    ;hai lệnh này tương đương
MOV       A,#1AH
MOV   A,#25 MOD 7  ;hai lệnh này tương đương
MOV       A,#4
```

4.3. Các toán tử logic.

- Toán tử NOT: NOT expr lấy bù đảo từng bit
- Toán tử SHR expr SHR n dịch sang phải n bit
- Toán tử SHL expr SHL n dịch sang trái n bit
- Toán tử AND expr AND expr and từng cặp bit tương ứng
- Toán tử OR expr OR expr or từng cặp bit tương ứng
- Toán tử XOR expr XOR expr xor từng cặp bit tương ứng

Trong đó expr là biểu thức và x là số vị trí cần dịch.

Ví dụ 16: 3 lệnh sau là tương đương

```
THREE   EQU 3
MINUS3  EQU  -3
MOV       A,#(NOT THREE) +1
MOV             A,#MINUS3
MOV             A,#11111101B
```

4.4. Các toán tử quan hệ (relation operators).

- EQ = bằng nhau (equal)
- NE <> không bằng nhau (not equal)
- LT < nhỏ hơn (less than)
- LE <= nhỏ hơn hoặc bằng (less than or equal)
- GT > lớn hơn (greater than)
- GE >= lớn hơn hay bằng (greater than or equal)

Kết quả luôn trả về đúng (FFFFH) hoặc sai (0000H

Ví dụ17: các lệnh sau là tương đương

```
MOV       A,#5=5
MOV       A,#5 EQ 5
MOV       A,#5 NE 4
MOV       A,#5 <> 4
MOV A,#0FFH
```

4.5. Các toán tử khác.

- Toán tử LOW expr có chức năng lấy kết quả byte thấp của expr.
- Toán tử HIGH expr có chức năng lấy kết quả byte cao của expr.

Ví dụ18:

```
MOV       DPH,#HIGH(1234H)    ;hai lệnh này tương đương
MOV       DPH,#12H
```

```
MOV     DPL,#LOW(1234H)     ;hai lệnh này tương đương
MOV     DPL,#34H
```

4.6. Thứ tự ưu tiên các toán tử.

Danh sách quyền ưu tiên của các toán tử được sắp theo thứ tự từ cao nhất đến thấp nhất như bảng 7.2

Thứ tự	Toán tử
1	()
2	HIGH LOW
3	* / MOD SHL SHR
4	+ -
5	EQ NE LT LE GT GE = <> < <= > >=
6	NOT
7	AND
8	OR
9	XOR

Khi các toán tử được sử dụng có quyền ưu tiên ngang nhau thì việc tính toán ra giá trị sẽ bắt đầu tính từ trái sang phải.

5. Các điều khiển của ASSEMBLER.

Mục tiêu: Biết các lệnh điều khiển trong chuong trình ASM.

Là những chỉ thị lệnh cho assembler và được chia ra làm các nhóm như sau:

- Điều khiển trạng thái Assembler: ORG, END, USING.
- Định nghĩa ký hiệu: segment, equ, set, data, Idata, Xdata, bit, code.
- Khởi tạo hay định nghĩa trong bộ nhớ: DS, DBIT, DB, DW.
- Liên kết chương trình: public, extrn, name.
- Chọn đoạn: Rseg, Cseg, Dseg, Iseg, Bseg, Xseg.

5.1. Điều khiển trạng thái ASSEMBLER.

❖ ORG: có chức năng thay đổi bộ đếm vị trí của segment hiện thời để đặt gốc chương trình mới cho các đoạn chương trình theo sau ORG.

Cách sử dụng: ORG expr

Ví dụ 19:

```
 ORG    2200h                ;khai báo địa chỉ bắt đầu 2200h
MOV    A,#35H
MOV    A,#35H
MOV    A,#35H

ORG  ($+1000H) and 0F000h        ;khai báo địa chỉ bằng địa chỉ hiện tại
cộng
MOV    A,#35H ;thêm 1000H và and với F000H để chuyển sang 4 kbyte kế
END
```

Nếu bỏ lệnh AND với F000H thì địa chỉ mới sẽ là 3206H = 2206H + 1000H. Ta có thể sử dụng khai báo ORG trong bất kỳ loại segment nào. Nếu segment hiện thời là tuyệt đối thì giá trị sẽ là địa chỉ tuyệt đối trong segment hiện thời. Nếu segment hiện thời là tái định vị được thì giá trị của biểu thức ORG được xử lý như offset của địa chỉ nền của segment hiện thời.

❖ USING: có chức năng báo cho assembler biết bank thanh ghi tích cực hiện thời, nhưng nó không chuyển băng thanh ghi, do đó để có thể sử dụng đúng thì ta phải sử dụng AR0 đến AR7 sau khai báo USING thay vì dùng R0 đến R7. Khi đó assembler sẽ tự động sử dụng đúng thanh ghi trong băng thanh ghi mong muốn đó và khi dịch assembler sẽ đổi ARN sang địa chỉ trực tiếp.

Cách sử dụng: USING expr

Ví dụ20:

```
USING 2
 MOV    AR3,#70H
MOV      R0,#22H
```

Trong lệnh thứ nhất, AR3 chính là thanh ghi R3 của bank thanh ghi 2 và sẽ được thay thế bằng địa chỉ trực tiếp là 13H. Trong lệnh thứ hai, R0 vẫn truy cập trong bank thanh ghi hiện tại là bank 0.

END: là phát biểu cuối cùng trong tập tin nguồn, những gì sau chỉ dẫn END sẽ không được xử lý.

5.2. Chỉ dẫn định nghĩa kí hiệu.

Những chỉ dẫn này tạo các ký hiệu để biểu diễn các segment, các thanh ghi, số và địa chỉ. Không được sử dụng nhãn cho chỉ dẫn. Những ký hiệu được định nghĩa bởi các chỉ dẫn này là duy nhất, ngoại trừ chỉ dẫn SET cho phép định nghĩa lại.

EQU hay SET: có chức năng gán 1 ký số hay ký hiệu thanh ghi cho tên ký hiệu được đặt tả.

Cách sử dụng:

symbol equ expr

symbol set expr

Trong đó symbol là ký hiệu do người dùng định nghĩa và expr là biểu thức.

Ví dụ21:

```
BDN     EQU     R2
GIAY    SET  40
```

Segment: có chức năng khai báo segment tái định vị được.

Cách sử dụng: symbol segment segment_type

Trong đó symbol là ký hiệu do người dùng định nghĩa và segment_type là kiểu segment. Có các kiểu segment như sau:

Code: segment mã chương trình.

Xdata: segment vùng dữ liệu chứa ở bộ nhớ bên ngoài.

Data: segment vùng dữ liệu nội có địa chỉ trực tiếp từ 00H÷7FH.

Idata: segment vùng dữ liệu nội có địa chỉ gián tiếp từ 00H÷7FH đối với 8051 và 00H÷FFH đối với 8052.

Bit: segment vùng nhớ bit nằm trong vùng nhớ cho phép truy xuất bit từ 20H÷2FH

Ví dụ 22:

```
EPROM   SEGMENT CODE
```

Khai báo ký hiệu eprom là segment kiểu code. Chú ý phát biểu này chỉ khai báo EPROM là kiểu code, để sử dụng segment này thì phải sử dụng chỉ dẫn RSEG.

CODE/DATA/IDATA/XDATA/BIT

Dùng để gán địa chỉ của kiểu tương ứng với ký hiệu, tuy nhiên nếu có sử dụng thì

assembler kiểm tra kiểu.

Ví dụ 23:

```
LOC   OBJ        LINE   SOURCE

0005             1      flag1   equ   05h
0005             2      flag2   bit   05h
 0000  D205      3      SETB   flag1
0002  D205       4      SETB   flag2
0004  750500     5      MOV    flag1,#0
```

```
0007   750500        6          MOV     flag2,#0
 *** ERROR #37, LINE #6 (0), DATA SEGMENT ADDRESS
EXPECTED
                      7          END
SYMBOL TABLE LISTING
------ ----- -------

 N A M E    T Y P E  V A L U E      A T T R I B U T E S

FLAG1...    NUMB    0005H   A
FLAG2...  B ADDR    0020H.5 A
REGISTER BANK(S) USED: 0
ASSEMBLY COMPLETE, 1 ERROR FOUND (6)
```

Trong chương trình ví dụ trên ta đã khai báo flag1 là ô nhớ có địa chỉ 05H, lag2 là bit có địa chỉ 05H. Hai lệnh setb khi biên dịch không có lỗi vì assembler xem chúng là các bit có địa chỉ 05H.

Lệnh thứ 5 khi biên dịch sẽ xem flag1 là ô nhớ có địa chỉ 05H, nhưng lệnh thứ 5 thì khi biên dịch sẽ báo lỗi vì lệnh MOV không thể thực hiện đối với ô nhớ bit.

5.3. Khởi tạo giá trị trong bộ nhớ

.DB (define byte): định nghĩa byte, có chức năng khởi tạo vùng nhớ mã với các giá trị kiểu byte.

Cách sử dụng: [label:] db expr [,expr][…]

Trong đó label là nhãn do người dùng định nghĩa và expr là biểu thức.

Ví dụ 24:

```
LOC    OBJ          LINE     SOURCE

----                  1          cseg     at      0100h
0100   C0             2          ma7d:   db      0c0h,0a4h
0101   A4
                      3
0102   48656C6C       4          msg:    db      'Hello'
0106   6F
                      5
                      6          end
SYMBOL TABLE LISTING
------ ----- -------
```

.DW (define word): định nghĩa từ, có chức năng khởi tạo vùng nhớ mã với

các giá trị kiểu word.

Cách sử dụng: [label:] dw expr [,expr][…]

Trong đó label là nhãn do người dùng định nghĩa và expr là biểu thức.

5.4. Định địa chỉ trong bộ nhớ.

.DS (define storage): định nghĩa vùng lưu trữ, có chức năng dành vùng nhớ theo byte. Chỉ dẫn này có thể được sử dụng trong bất kỳ loại segment nào ngoại trừ DBIT.

Cách sử dụng: [label:] ds expr

Trong đó label là nhãn do người dùng định nghĩa và expr là biểu thức không có tham chiếu tới.

Khi gặp chỉ dẫn DS trong chương trình thì bộ đếm vị trí của segment hiện tại được tăng thêm số byte là giá trị của expr.

Ví dụ 25: tạo vùng nhớ Ram nội 40 byte để lưu dữ liệu:

```
        DSEG    AT      10H     ;vùng nhớ dữ liệu nội
          LEN   EQU     40
   BUF: DS      LEN     ;dành 40 byte bắt đầu từ địa chỉ 10H
```

Ví dụ 26: tạo vùng nhớ Ram ngoại 1000 byte để lưu dữ liệu:

```
XSEG    AT      2000H   ;vùng nhớ dữ liệu ngoại
XLEN    EQU     1000
XBUF:   DS      LEN     ;dành 1000 byte bắt đầu từ địa chỉ
                        2000H
        CSEG    AT      0000H   ;bắt đầu mã chương trình
        MOV     DPTR,#XBUF ;nạp địa chỉ của vùng nhớ ngoại vào
                           dptr
LOOP:   CLR     A
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,DPL
        CJNE    A,#LOW(XBUF+XLEN+1),LOOP;so sánh địa chỉ
                                        byte thấp
        MOV     A,DPH
        CJNE    A,#HIGH(XBUF+XLEN),LOOP;so sánh địa chỉ byte
                                       cao để kết thúc
        SJMP $
        END
```

.Dbit (define bit): định nghĩa vùng lưu trữ dữ liệu bit, có chức năng dành vùng nhớ theo bit trong segment bit.

Cách sử dụng: [label:] dbit expr

Trong đó label là nhãn do người dùng định nghĩa và expr là biểu thức không có tham chiếu tới. Khi gặp chỉ dẫn DBIT trong chương trình thì bộ đếm vị trí của segment BIT hiện tại được tăng thêm với số bit là giá trị của expr.

5.5. Liên kết chương trình.

Cho phép các module (các tập tin) được hợp dịch riêng có thể liên lạc với nhau bằng cách cho phép tham chiếu giữa các module và đặt tên các module.

Public: Liệt kê các ký hiệu có thể được sử dụng trong các module đối tượng khác.

Cách sử dụng: public symbol [,symbol][,..]

Trong đó ký hiệu symbol được khai báo trong Public phải được định nghĩa trong module hiện hành.

Ví dụ 28: Public inchar, outchar, inline, outstr, extern

Extrn: Liệt kê các ký hiệu được tham chiếu trong module nguồn hiện hành nhưng chúng được khai báo trong các module khác.

Cách sử dụng: extrn segment_type(symbol [,symbol][,..])

Các segment_type là CODE, XDATA, DATA, IDATA, BIT và NUMBER (NUMBER là ký hiệu không có kiểu được định nghĩa bằng EQU).

Ví dụ 29: Có 2 tập tin MAIN.SRC và message.SRC

```
;main.src
Extrn   code (HELLO, GOOD_BYE)
…
CALL HELLO
…
CALL GOOD_BYE
…
END

;MESSAGE.SRC
 PUBLIC          HELLO, GOOD_BYE
…
HELLO;
…
RET
GOOD_BYE;
…
RET
```

Hai module trên không phải là chương trình đầy đủ: chúng được biên dịch riêng và liên kết với nhau để tạo chương trình khả thi. Trong khi liên kết, các tham chiếu ngoài được thay thế với địa chỉ đúng cho các lệnh CALL

Name: dùng để đặt tên của module đối tượng được sinh ra trong chương trình hiện hành.

Cách sử dụng: Name module_name

Trong đó module_name là tên module

5.6. Cách chọn segment.

- Segment là khối chứa mã lệnh hay vùng nhớ chứa dữ liệu mà assembler tạo ra từ mã hay dữ liệu trong tập tin nguồn hợp ngữ 8051. Khi Assembler gặp chỉ dẫn chọn segment thì nó sẽ chuyển hướng mã hoặc dữ liệu theo sau vào segment được chọn cho đến khi gặp chỉ dẫn chọn segment khác.
- .RSEG (relocatable segment – segment tái định vị được): cho phép chọn segment tái định được mà đã định nghĩa trước bằng segment.

Cách sử dụng: Rseg segment_name

Trong đó segment_name là tên segment đã định nghĩa trước đó.

- Chỉ dẫn này chuyển hướng mã và dữ liệu theo sau vào đoạn segment_name cho đến khi gặp chỉ dẫn chọn segment khác.
- Các kiểu chỉ dẫn chọn segment CSEG/ DSEG/ ISEG/ XSEG Cho phép chọn 1 segment tuyệt đối.

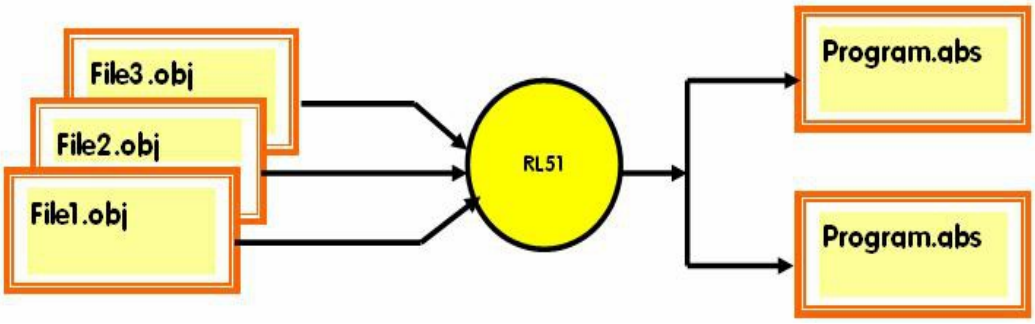
Cách sử dụng: aSEG [at address]

Trong đó a có thể là C, D, I, B hoặc X và address là địa chỉ.

6. Hoạt động liên kết (Linker).

Mục tiêu: hiểu được các hoạt động liên kết trong chương trình.

- Với những ứng dụng lớn người lập trình thường chia chương trình thành nhiều chương trình con hay các module và có thể tái định vị được.
- Ta cần chương trình liên kết và định vị để kết hợp các module thành một module đối tượng tuyệt đối mà có thể thực thi được. Tất cả các ký hiệu ngoài được thay thế bằng các giá trị đúng và được đặt vào trong các tập tin xuất được minh hoạ như hình 7.3:



Hình 7.3. Hoạt động của chương trình linker có tên là RL51.

7. Macro

Mục tiêu: biết cách ứng dụng các Macro vào chương trình ứng dụng.

- Phương tiện xử lý macro của ASM51 là phương tiện thay thế chuỗi ký tự. Macro cho phép các phần mã sử dụng thường xuyên sẽ được định nghĩa một lần bằng cách dùng từ gợi nhớ đơn giản và có thể sử dụng bất kỳ chỗ nào trong chương trình bằng cách chèn vào từ gợi nhớ đó.
- Ta có thể định nghĩa macro ở bất kỳ chỗ nào trong chương trình nguồn và sau đó sử dụng như các lệnh khác. Cú pháp của định nghĩa macro như sau:

%*define (call_pattern) (macro_body)

- Trong đó call_pattern là từ gợi nhớ do người dùng định nghĩa và macro_body là thân macro chính là đoạn chương trình thường lặp lại.
- Để phân biệt với các lệnh thật thì người ta đặt thêm ký hiệu "%" trước tên macro và khi hợp dịch thì tất cả các lệnh trong thân macro được thay thế vào nơi gọi chương trình macro.

Ví dụ 30: Nếu định nghĩa macro sau ở đầu tập tin nguồn

```
%*define (push_dptr)
(
   PUSH DPH
   PUSH    DPL
)
```

Thì khi gặp phát biểu %push_dptr trong chương trình nguồn thì trình biên dịch sẽ thay thế bằng 2 lệnh trên

```
PUSH DPH
PUSH    DPL
```

trong tập tin.lst

- Các tiện lợi khi sử dụng macro:

Chương trình nguồn có sử dụng macro thì dễ đọc hơn vì từ gợi nhớ của macro cho biết ý nghĩa của công việc phải thực hiện.

Chương trình ngắn gọn hơn nên ít đánh máy hơn.

Sử dụng macro sẽ làm giảm bớt lỗi.

Sử dụng macro giúp cho người lập trình không phải bận rộn với những chi tiết cấp thấp.

7.1. Truyền tham số cho Macro.

- Macro với các tham số được truyền từ chương trình chính có dạng như sau:

%*define (macro_name (parameter_list)) (macro_body)

Trong đó *macro_name* là tên macro, *parameter_list l*à danh sách các tham số và *macro_body* là thân macro.

Ví dụ 31: Định nghĩa macro có truyền tham số như sau

```
%*define (cmpa#(value))
( CJNE  A,#%value,$+3 )
```

Thì khi gọi phải truyền tham số như sau:

```
%Cmpa#(20H)
```

Khi biên dịch sẽ trở thành

```
CJNE  A,#20H,$+3
```

Chú ý: lệnh cjne là lệnh 3 byte, do đó $+3 chính là địa chỉ của lệnh kế nằm sau lệnh CJNE.

7.2. Macro với nhãn cục bộ.

- Ta cũng có thể sử dụng các nhãn cục bộ trong macro có dạng như sau:

*%*define (macro_name [(parameter_list)])*

[local list_of_labels](macro_body)

- Trong đó *macro_name* là tên macro, *parameter_list* là danh sách các tham số, *list_of_labels* là danh sách các nhãn cục bộ và *macro_body* là thân macro.

Ví dụ 32: Định nghĩa macro có nhãn cục bộ như sau

```
%*define (dec_dptr) local skip
( DEC   DPL
MOV A,DPL
CJNE  A,#0FFh,%skip
DEC DPH
%Skip:)
```

Khi macro được gọi

```
%dec_dptr
```

Thì trình biên dịch sẽ thay thế lệnh gọi trên bằng các lệnh đã định nghĩa trong macro ở file.lst như sau:

```
DEC   DPL
```

```
MOV A,DPL
CJNE  A,#0FFh,skip00
DEC DPH
      Skip00:
```

Nhãn cục bộ không quan hệ với nhãn có cùng tên trong chương trình chính vì trình biên dịch ASM51 đã tự động thêm vào mã số đi theo sau nhãn cục bộ khi biên dịch.

Nhưng nếu chúng ta định nghĩa macro như sau thì khi biên dịch ASM51 sẽ không đổi tên nhãn cục bộ:

```
%*define (dec_dptr) local skip
(
DEC   DPL
MOV A,DPL
CJNE  A,#0FFH,SKIP
DEC DPH
Skip:)
```

7.3. Tác động lặp lại (Repeat).

Là một trong các macro được xây dựng sẵn trong Assembler. Cú pháp: *%repeat (expression) (text)*

Trong đó expression là biểu thức và text là văn bản cần lặp lại.

Ví dụ 33: để thực hiện 100 lệnh NOP thì ta có thể sử dụng macro repeat như sau:

```
%repeat(100)
(
Nop
)
```

Khi biên dịch thì trong file.lst sẽ thay hàng lệnh trên bằng 100 lệnh NOP.

7.4. Các tác vụ điều khiển.

ASM51 cung cấp các định nghĩa macro luồng điều khiển để cho phép hợp dịch có điều kiện các phần mã. Dạng lệnh macro như sau:

Cú pháp:

%IF (expression) THEN (balaneced_text)

[ELSE (balaneced_text)]

Trong đó expression là biểu thức và balaneced_text là văn bản cần thay đổi theo điều kiện.

CÁC BÀI TẬP MỞ RỘNG, NÂNG CAO VÀ GIẢI QUYẾT VẤN ĐỀ

Luyện tập một số bài lập trình cho kít thi nghiệm

Bài 1. Điều khiển khối hiển thị LCD

1.1. Mục đích, yêu cầu:

Giúp sinh viên làm quen với việc điều khiển hiển thị dữ liệu trên màn hình LCD trong các ứng dụng như đếm sản phẩm,đồng hồ báo giờ,hiển thị chuỗi thông báo…

Sinh viên phải nắm được cấu tạo và cách thức điều khiển hiển thị trên màn hình LCD.

1.2. Chương trình tham khảo:

Chương trình điều khiển hiển thị hai dòng Text ra màn hình LCD sau đó di chuyển hai dòng text kết hợp với bật tắt đèn Back light.

```
;KHAI BAO BIEN
RS          BIT     P2.3
RW          BIT     P2.4
EN          BIT     P2.5
BACK_LIGHT  BIT     P2.6
DATA_LCD    EQU     P0.1
        ORG  0000H
    LJMP MAIN
        ORG 0030H
MAIN:
    MOV     A,#38H              ;TAO MA TRAN 2 DONG 5X7
    LCALL   OUT_INSTRUCTION
    MOV     A,#1H               ;XOA MAN HINH LCD
    LCALL   OUT_INSTRUCTION
    MOV     A,#0CH              ;BAT HIEN THI,TAT CON TRO
    LCALL   OUT_INSTRUCTION
    MOV     A,#80H  ;DUA CON TRO VE DAU DONG THU NHAT
    LCALL   OUT_INSTRUCTION
    MOV     DPTR,#BANG1
TEXT1:              ; HIEN THI DONG TEXT THU NHAT
    MOV     A,#0
    MOVC    A,@A+DPTR
    LCALL   OUT_DATA
```

```
        INC     DPTR
        CJNE    A,#99H,TEXT1
        MOV     A,#0C0H     ;DUA CON TRO VE DAU DONG THU
                            HAI

        LCALL   OUT_INSTRUCTION
        MOV     DPTR,#BANG2
TEXT2:                              ; HIEN THI DONG TEXT THU HAI
        MOV     A,#0
        MOVC    A,@A+DPTR
        LCALL   OUT_DATA
        INC     DPTR
        CJNE    A,#99H,TEXT2
        MOV     A,#0CH              ;TAT CON TRO
        LCALL   OUT_INSTRUCTION
        CLR     BACK_LIGHT
        LCALL   DELAY_5S LOOP:
        SETB    BACK_LIGHT          ; TAT DEN BACK LIGHT
        MOV     R0,#17
SHIFT_LEFT:
        MOV     A,#18H      ;DICH TOAN BO HIEN THI SANG TRAI
        LCALL   OUT_INSTRUCTION
        LCALL   DELAY_200MS
        DJNZ    R0,SHIFT_LEFT       ; KIEM TRA SO LAN DICH
        MOV     R0,#17
SHIFT_RIGHT:
        MOV     A,#1CH    ;DICH TOAN BO HIEN THI SANG TRAI
        LCALL   OUT_INSTRUCTION
        LCALL       DELAY_200MS
        DJNZ    R0,SHIFT_RIGHT      ; KIEM TRA SO LAN DICH
        CLR     BACK_LIGHT          ; BAT DEN BACK LIGHT
        LCALL   DELAY_1S
       LJMP LOOP
        ;
**********************************************************OUT
_INSTRUCTION:
    MOV DATA_LCD,A    ;DUA MA LENH RA PORT GIAO
                                    TIEP VOI LCD
    CLR     RS ;CHON THANH GHI LENH
```

```
        CLR     RW              ;CHON CHE DO GHI
        SETB  EN                ;CHO PHEP DUA LENH RA LCD
        LCALL DELAY
        CLR     EN
        LCALL DELAY
        RET
OUT_DATA:
        MOV     DATA_LCD,A ;DUA DU LIEU RA PORT GIAO TIEP
                                        VOI LCD
        SETB    RS                      ;CHON THANH GHI DU LIEU
        CLR R   W                               ;CHON CHE DO GHI
        SETB    EN ;CHO PHEP DUA DU LIEU RA LCD
        LCALL DELAY
        CLR     EN
        LCALL DELAY
        RET
        ;*****************************************************
DELAY_1S:
        MOV 33H,#20
        MOV TMOD,#00000001B
LOOP_DL1S:
        MOV TH0,#HIGH(-50000)
        MOV TL0,#LOW(-50000)
        SETB TR0
        JNB     TF0,$
        CLR     TR0
        CLR     TF0
        DJNZ 33H,LOOP_DL1S
                RET
        ;*****************************************************
DELAY_5S:
        MOV 32H,#100
        MOV TMOD,#00000001B
LOOP_DL5S:
        MOV TH0,#HIGH(-50000)
        MOV TL0,#LOW(-50000)
        SETB TR0
        JNB     TF0,$
```

```
    CLR    TR0
    CLR    TF0
    DJNZ 32H,LOOP_DL5S
    RET
        ;*************************************************
DELAY_200MS:
    MOV 31H,#4
    MOV TMOD,#00000001B
LOOP_DL200MS:
    MOV TH0,#HIGH(-50000)
    MOV TL0,#LOW(-50000)
    SETB TR0
    JNB    TF0,$
    CLR    TR0
    CLR    TF0
    DJNZ 31H,LOOP_DL200MS
RET
        ;
**************************************************DELAY
:
    MOV      30H,#255
    DJNZ 30H,$ RET
BANG1:
        DB     '   WELLCOME TO  ',99H BANG2:
        DB     'CONTROLLER LAB!',99H
        END
```

Bài 2: Truy xuất IC real time DS12887

1.1. Mục đích, yêu cầu :

Giúp sinh viên làm quen với việc truy xuất dữ liệu thời gian từ IC real time cho các ứng dụng yêu cầu thời gian thực như đồng hồ báo giờ…

1.2. Chương trình tham khảo :

Chương trình đồng hồ báo giờ,phút,giây,thứ ngày tháng năm bằng cách truy

xuất dữ liệu thời gian từ Real time,cho phép khởi tạo lại Real time.

```
;KHAI BAO BIEN
    RS              BIT             P2.3
    RW    BIT   P2.4
```

```
EN                BIT       P2.5
BACK_LIGHT        BIT       P2.6
ENABLE_REALTIME   BIT       P2.7
ENABLE_INT1       BIT       P3.3
DATA_LCD          EQU       P0
STEP_SHIFT        EQU       7FH

LCD_BUF           EQU       7EH


GIAY              EQU       7DH
PHUT              EQU       7CH

GIO               EQU       7BH


THU               EQU       7AH
NGAY              EQU       79H
THANG             EQU       78H
NAM               EQU       77H


DONVI_GIAY        EQU       76H
CHUC_GIAY         EQU       75H
DONVI_PHUT        EQU       74H




CHUC_PHUT         EQU       73H
DONVI_GIO         EQU       72H
CHUC_GIO          EQU       71H
DONVI_NGAY        EQU       70H
CHUC_NGAY         EQU       6FH
DONVI_THANG       EQU       6EH
CHUC_THANG        EQU       6DH
```

```
DONVI_NAM       EQU         6CH


CHUC_NAM        EQU         6BH
DATA_IN         EQU         6AH
DATA_OUT        EQU         69H


        ORG  0000H
    LJMP MAIN
        ORG 000BH
    LJMP ISR_T0
        ORG 0030H
  MAIN:
    MOV TMOD,#00010001B
    MOV TH0,#HIGH(-50000)
    MOV TL0,#LOW(-50000)
    MOV IE,#10000010B

    MOV     LCD_BUF,#38H   ;TAO MA TRAN 5X7
    LCALL   OUT_INSTRUCTION
    MOV     LCD_BUF,#00001111B  ;ON DINH LCD
    LCALL   OUT_INSTRUCTION
    MOV     LCD_BUF,#1H     ;XOA MAN HINH LCD
    LCALL   OUT_INSTRUCTION
    MOV     LCD_BUF,#0EH    ;BAT HIEN THI
    LCALL   OUT_INSTRUCTION
    LCALL   DELAY1
    JB      E_INT1,LOOP   ; CHO PHEP KHOI TAO LAI REAL TIME
    LCALL   DELAY
    JB      E_INT1,LOOP
    JN      B E_INT1,$
    MOV     A,#80H      ;DUA CON TRO VE DAU DONG THU
                            NHAT
    LCALL   OUT_INSTRUCTION
   MOV      DPTR,#BANG1
TEXT1:      ;HIEN THI DONG THONG BAO SET UP REAL TIME
    MOV     A,#0
    MOVC    A,@A+DPTR
    MOV     LCD_BUF,A
```

```
        LCALL   OUT_DATA
        INC     DPTR
        CJNE A,#99H,TEXT1     ;KIEM TRA MA KET THUC DONG
                              TEXT
        MOV   LCD_BUF,#0C0H   ;DUA CON TRO VE DAU DONG
                              THU HAI
        LCALL   OUT_INSTRUCTION
        MOV     R0,#16
LOOP_SETUP:
        MOV     LCD_BUF,#'*'  ; HIEN THI KHOANG TRONG
        LCALL   OUT_DATA
        LCALL   DELAY1
        DJNZ    R0,LOOP_SETUP

        LCALL   KHOI_TAO_REAL
        LCALL   DELAY1
        LCALL   DELAY1
        MOV     LCD_BUF,#1H           ;XOA MAN HINH LCD
        LCALL   OUT_INSTRUCTION
LOOP:
        SETB    TR0
        SJMP $



    ;
**********************************************************SHI
TF_RIGHT:
        MOV   LCD_BUF,#14H ;DICH CON TRO SANG PHAI
        LCALL OUT_INSTRUCTION
        DJNZ   STEP_SHIFT,SHITF_RIGHT
        RET
    ;**********************************************************
    TIME_DISPLAY:
        MOV   LCD_BUF,#80H    ;DUA CON TRO VE DAU HANG
                              THU NHAT
        LCALL OUT_INSTRUCTION
        MOV   LCD_BUF,#0CH    ;BAT HIEN THI TAT CON TRO
        LCALL OUT_INSTRUCTION

        MOV   LCD_BUF,#' '    ; HIEN THI KHOANG TRONG
        LCALL OUT_DATA
```

```
        MOV R0,THU                  ; HIEN THI DAY OF WEEK
SUN:
        CJNE R0,#1,MON
        MOV     LCD_BUF,#'S'        ;HIEN THI CHU "SUN"
        LCALL OUT_DATA
        MOV     LCD_BUF,#'U'
        LCALL OUT_DATA
         MOV    LCD_BUF,#'N'
         LCALL OUT_DATA
         LJMP NEXT_HT
MON:
        CJNE R0,#2,TUE
        MOV     LCD_BUF,#'M'        ;HIEN THI CHU "MON"
        LCALL OUT_DATA
        MOV     LCD_BUF,#'O'
         LCALL OUT_DATA MOV    LCD_BUF,#'N'
        LCALL OUT_DATA
        LJMP    NEXT_HT
    TUE:
        CJNE R0,#3,WEN
        MOV     LCD_BUF,#'T'        ;HIEN THI CHU "TUE"
        LCALL OUT_DATA
        MOV     LCD_BUF,#'U'
        LCALL OUT_DATA
        MOV     LCD_BUF,#'E'
        LCALL OUT_DATA
         LJMP NEXT_HT
WEN:
        CJNE R0,#4,THUR
        MOV     LCD_BUF,#'W'        ;HIEN THI CHU "WEN"
        LCALL OUT_DATA
        MOV     LCD_BUF,#'E'
        LCALL OUT_DATA
        MOV     LCD_BUF,#'N'
        LCALL OUT_DATA
         LJMP NEXT_HT
THUR:
        CJNE R0,#5,FRI
        MOV     LCD_BUF,#'T'        ;HIEN THI CHU "THU"
        LCALL OUT_DATA
```

```
        MOV    LCD_BUF,#'H'
        LCALL OUT_DATA
         MOV    LCD_BUF,#'U'
         LCALL OUT_DATA
        LJMP NEXT_HT
FRI:
        CJNE R0,#6,SAT
        MOV         LCD_BUF,#'F'      ;HIEN THI CHU "FRI"
        LCALL OUT_DATA
        MOV    LCD_BUF,#'R'
        LCALL OUT_DATA
        MOV    LCD_BUF,#'I'
        LCALL OUT_DATA
        LJMP NEXT_HT
SAT:
        CJNE R0,#7,ERROR ERROR:
        MOV         LCD_BUF,#'S'      ;HIEN THI CHU "SAT"
        LCALL OUT_DATA
        MOV    LCD_BUF,#'A'
        LCALL OUT_DATA
        MOV    LCD_BUF,#'T'
         LCALL OUT_DATA
         LJMP NEXT_HT

NEXT_HT:
        MOV    LCD_BUF,#' '                ; HIEN THI KHOANG
        TRONG LCALL OUT_DATA

        MOV    LCD_BUF,CHUC_NGAY    ; HIEN THI NGAY
        LCALL OUT_DATA
        MOV    LCD_BUF,DONVI_NGAY
        LCALL OUT_DATA
        MOV    LCD_BUF,#' '                ; HIEN THI KHOANG
                                           TRONG
        LCALL OUT_DATA
        MOV    LCD_BUF,CHUC_THANG  ; HIEN THI THANG
        LCALL OUT_DATA
        MOV    LCD_BUF,DONVI_THANG
        LCALL OUT_DATA
        MOV    LCD_BUF,#' '                ; HIEN THI KHOANG
                                            TRONG
```

```
LCALL OUT_DATA

MOV    LCD_BUF,#'2'              ; HIEN THI KY TU "2"
LCALL OUT_DATA

MOV    LCD_BUF,#'0'              ; HIEN THI KY TU "0"
LCALL OUT_DATA

MOV    LCD_BUF,CHUC_NAM          ; HIEN THI NAM
LCALL OUT_DATA

MOV    LCD_BUF,DONVI_NAM
LCALL OUT_DATA
MOV    LCD_BUF,#0C0H             ;DUA CON TRO VE DAU
                                 DONG THU HAI
LCALL OUT_INSTRUCTION

MOV STEP_SHIFT,#4
LCALL SHITF_RIGHT                ; DICH CON TRO SANG
                                 PHAI

MOV    LCD_BUF,CHUC_GIO          ; HIEN THI GIO
LCALL OUT_DATA

MOV    LCD_BUF,DONVI_GIO
LCALL OUT_DATA

MOV    LCD_BUF,#':'                    ; HIEN THI DAU ":"
LCALL OUT_DATA
MOV    LCD_BUF,CHUC_PHUT               ; HIEN THI PHUT
LCALL OUT_DATA

MOV    LCD_BUF,DONVI_PHUT
LCALL OUT_DATA

MOV    LCD_BUF,#':'                    ; HIEN THI DAU ":"
LCALL OUT_DATA
MOV    LCD_BUF,CHUC_GIAY               ; HIEN THI GIAY
LCALL OUT_DATA

MOV    LCD_BUF,DONVI_GIAY
LCALL OUT_DATA
```

```
        RET
;*************************************************************
GIAI_MA:
        MOV     A,DATA_IN
        MOV       DPTR,#MA_LCD
        MOVC    A,@A+DPTR
        MOV     DATA_OUT,A
        RET
;
*************************************************************
XU_LY:
        MOV A,GIAY MOV B,#10
        DIV AB
        MOV CHUC_GIAY,A
        MOV DONVI_GIAY,B

        MOV   DATA_IN,DONVI_GIAY
        LCALL GIAI_MA
        MOV   DONVI_GIAY,DATA_OUT

        MOV   DATA_IN,CHUC_GIAY
        LCALL GIAI_MA
        MOV   CHUC_GIAY,DATA_OUT
        MOV       A,PHUT
        MOV       B,#10
        DIV       AB
        MOV       CHUC_PHUT,A
        MOV       DONVI_PHUT,B
        MOV       DATA_IN,DONVI_PHUT
        LCALL     GIAI_MA
        MOV       DONVI_PHUT,DATA_OUT
        MOV       DATA_IN,CHUC_PHUT
        LCALL     GIAI_MA
        MOV       CHUC_PHUT,DATA_OUT
        MOV       A,GIO
        MOV       B,#10
        DIV       AB

        MOV       CHUC_GIO,A
        MOV       DONVI_GIO,B
        MOV       DATA_IN,DONVI_GIO
```

```
        LCALL   GIAI_MA
        MOV     DONVI_GIO,DATA_OUT

        MOV     DATA_IN,CHUC_GIO
        LCALL   GIAI_MA
        MOV     CHUC_GIO,DATA_OUT

        MOV     A,NGAY
        MOV     B,#10
        DIV     AB
        MOV     CHUC_NGAY,A
        MOV     DONVI_NGAY,B

        MOV     DATA_IN,DONVI_NGAY
        LCALL   GIAI_MA
        MOV     DONVI_NGAY,DATA_OUT

        MOV     DATA_IN,CHUC_NGAY
        LCALL   GIAI_MA
        MOV     CHUC_NGAY,DATA_OUT
        MOV     A,THANG
        MOV     B,#10
        DIV     AB
        MOV     CHUC_THANG,A
        MOV     DONVI_THANG,B
        MOV     DATA_IN,DONVI_THANG
        LCALL   GIAI_MA
        MOV     DONVI_THANG,DATA_OUT

        MOV     DATA_IN,CHUC_THANG
        LCALL   GIAI_MA
        MOV     CHUC_THANG,DATA_OUT

        MOV     A,NAM
        MOV     B,#10
        DIV     AB
        MOV     CHUC_NAM,A
        MOV     DONVI_NAM,B
```

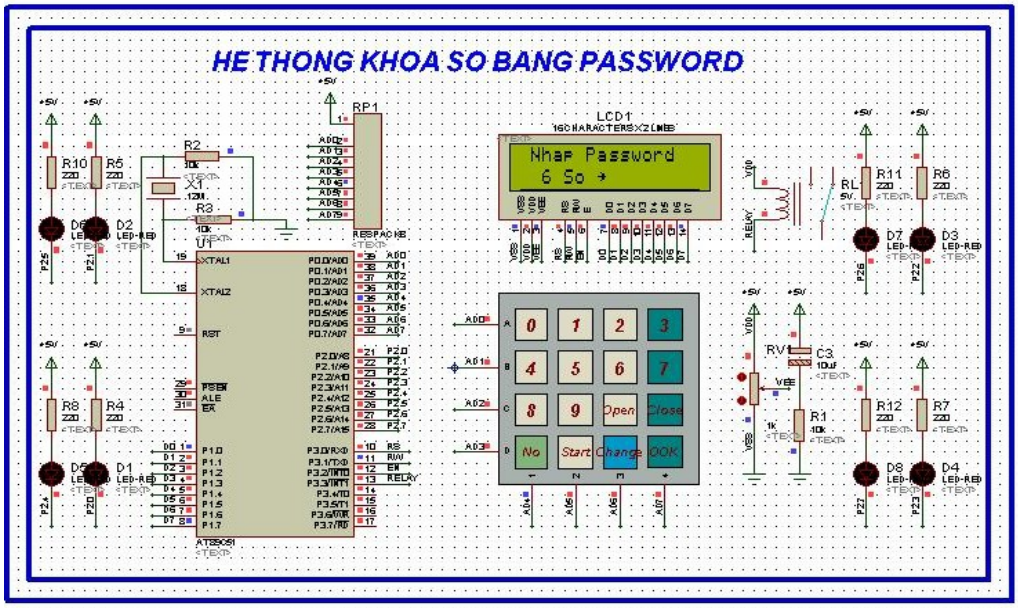
```
        MOV     DATA_IN,DONVI_NAM
        LCALL   GIAI_MA
        MOV     DONVI_NAM,DATA_OUT

        MOV     DATA_IN,CHUC_NAM
         LCALL  GIAI_MA
        MOV     CHUC_NAM,DATA_OUT
        RET
;*************************************************************
ISR_T0:
        CLR     TR0
        MOV     TH0,#HIGH(-50000)
        MOV     TL0,#LOW(-50000)
        LCALL   READ_TIME
        LCALL   XU_LY
        LCALL   TIME_DISPLAY
        SETB    TR0
        RETI
;
*************************************************************
READ_TIME:
        CLR     ENABLE_REALTIME   ; CHO PHEP DOC REAL TIME
        MOV     R0,#0
        MOVX    A,@R0
        MOV     GIAY,A

        MOV     R0,#2
        MOVX    A,@R0
        MOV     PHUT,A
        MOV     R0,#4
        MOVX    A,@R0
        MOV     GIO,A
        MOV     R0,#6
        MOVX    A,@R0
        MOV     THU,AMOV R0,#7

        MOVX      A,@R0
        MOV       NGAY,A

        MOV       R0,#8
```

```
        MOVX    A,@R0
        MOV     THANG,A
        MOV     R0,#9
        MOVX    A,@R0
        MOV     NAM,A
        SETB    ENABLE_REALTIME  ; NGUNG CHO PHEP DOC
                                   REAL TIME
        RET
;*****************************************************
KHOI_TAO_REAL:                   ; CHO PHEP DOC REAL TIME
        CLR     ENABLE_REALTIME
        MOV      R0,#0       ; O NHO GIAY
        MOV     A,#0  ; NAP GIA TRI 00 VAO O NHO GIAY
        MOVX    @R0,A

        MOV     R0,#02H     ; O NHO PHUT
        MOV     A,#0  ; NAP GIA TRI 00 VAO O NHO PHUT
        MOVX    @R0,A

        MOV     R0,#04H     ; O NHO GIO
        MOV     A,#0  ; NAP GIA TRI 00 VAO O NHO GIO
        MOVX    @R0,A

        MOV     R0,#06H     ; O NHO THU
        MOV     A,#7  ; NAP GIA TRI 1 VAO O NHO THU
        MOVX    @R0,A

        MOV     R0,#07H     ; O NHO NGAY
        MOV     A,#1
        MOVX    @R0,A

        MOV     R0,#08H     ; O NHO THANG
        MOV     A,#11
        MOVX    @R0,A

        MOV     R0,#09H     ; O NHO NAM
        MOV     A,#8
        MOVX    @R0,A

        MOV     R0,#0AH
        MOV     A,#00100000B     ;TURN ON THE OSCILATOR AND
                 ALLOW THE RTC TO KEEP TIME
        MOV     X @R0,A
```

```
        MOV     R0,#0BH
        MOV     A,#00000010B ;SIGNIFIES BINARY DATA AND 24H
        MODE
        MOVX    @R0,A
        SETB    ENABLE_REALTIME  ; NGUNG CHO PHEP DOC
                                          REAL TIME
        RET
    ;*********************************************************
OUT_INSTRUCTION:
        MOV   DATA_LCD,LCD_BUF     ;DUA MA LENH RA PORT
                                   GIAO TIEP VOI LCD
        CLR   RS                   ;CHON THANH GHI LENH
        CLR   RW                   ;CHON CHE DO GHI
        SETB  EN                   ;CHO PHEP DUA LENH RA LCD
        LCALL DELAY
        CLR   EN
        CALL DELAY
        RET

OUT_DATA:
        MOV   DATA_LCD,LCD_BUF     ;DUA DU LIEU RA PORT
                                   GIAO TIEP VOI LCD
        SETB   RS              ;CHON THANH GHI DU LIEU
        CLR   RW                   ;CHON CHE DO GHI
        SETB    EN     ;CHO PHEP DUA DU LIEU RA  LCD
        LCALL DELAY
        CLR   EN
        LCALL DELAY
        RET
;*********************************************************
DELAY1:
        MOV   33H,#255
LOOP_DL1:
        MOV   32H,#255
        DJNZ 32H,$
        DJNZ 33H,LOOP_DL1
        RET
    ;*********************************************************
DELAY:
        MOV   31H,#100
        DJNZ 31H,$
```

```
        RET
    ;*****************************************************
MA_LCD:
        DB      '0'
        DB      '1' DB    '2' DB      '3'
        DB      '4' DB    '5' DB      '6' DB  '7' DB  '8'  DB  '9'
    ;*****************************************************
BANG1:
        DB      'SET UP REAL TIME',99H
    ;*****************************************************
        LCD COMMAND
        MOV   LCD_BUF,#1H          ;XOA MAN HINH HIEN THI MOV
        MOV   LCD_BUF,#8H          ;TAT TOAN BO HIEN THI MOV
        MOV   LCD_BUF,#0CH         ;BAT HIEN THI TAT CON TRO
        MOV MOV                 LCD_BUF,#0DH        ;NHAP NHAY
        CON TRO
        MOV   LCD_BUF,#10H ;DICH CON TRO SANG TRAI
        MOV   LCD_BUF,#14H ;DICH CON TRO SANG PHAI
        MOV   LCD_BUF,#18H ;DICH TOAN BO HIEN THI SANG TRAI
        MOV   LCD_BUF,#1CH       ;DICH TOAN BO HIEN THI SANG
        PHAI
        MOV   LCD_BUF,#80H ;DUA CON TRO VE DAU DONG THU
                                   NHAT
        MOV   LCD_BUF,#0C0H ;DUA CON TRO VE DAU DONG THU
                                        HAI
        MOV   LCD_BUF,#15H ;ON DINH LCD
        END
```

Bai 3: Viết chương trình điều khiển mạch khóa số bằng Password có hiển thị bằng LCD co hình ve như sau:



Chương trình mẫu

```
$MOD51
ORG  0000H

        RS          EQU  P3.0
        RW          EQU  P3.1
        EN          EQU  P3.2
        PASS        EQU  40H
        DATABUS     EQU  P1

        LCALL CHAY_CHU
MAIN1:
        MOV R0,#70H
        MOV 70H,#0   ;PASS MAC DINH DUOC LUU TRU TU O NHO 70H->75H
        MOV 71H,#0
        MOV 72H,#0
        MOV 73H,#0
        MOV 74H,#0
        MOV 75H,#0
;WAITING ME
        MOV R3,#00H
        MOV R2,#30
        MOV R6,#0
                                ;BYTE CO DINH R0,R2,R3,R6,R5,R1
```

```
MAIN:
        SETB P3.3
        MOV R5,#0
        MOV R1,#60H

;**********************INNITIAL
LCD**************************************
;**********************STRING_LCD_FIRST
LINE*******************************
NHAP_PASS:

        CLR RS                  ;SEND COMMAND
        CLR RW                  ;RW=0-WRITE LCD MODE
        SETB EN                 ;E=1
        MOV DATABUS,#38H            ;CODE=38H-
8BIT,16CHAR/LINE,MATRIX 5X8
        LCALL GUI_LENH              ;SEND COMMAND TO LCD
        MOV TH0,#HIGH(-4100)
        MOV TL0,#LOW(-4100)
        LCALL DELAY_2

        MOV DATABUS,#38H
        LCALL GUI_LENH
        MOV TH0,#HIGH(-100)
        MOV TL0,#LOW(-100)
        LCALL DELAY_2

        MOV DATABUS,#38H
        LCALL GUI_LENH
        MOV DATABUS,#0CH            ;CODE=0CH-ENABLE DISPLAY
LCD
        LCALL GUI_LENH
        MOV DATABUS,#01H            ;CODE=01H-CLEAR LCD
        LCALL GUI_LENH
        MOV DATABUS,#81H            ; XUAT HIEN DONG DAU TIEN
        ACALL      GUI_LENH

        PUSH04H
        PUSHACC
```

```
        MOV R4,#00H
        MOV DPTR,#STRINGCODE
LOOP1:
        MOV A,R4
        MOVC    A,@A+DPTR

        LCALL   GUI_DATA
        INC   R4
        CJNE R4,#14,LOOP1     ; HIEN DONG THONG BAO THU NHAT
15 KI TU
        POP   ACC
        POP   4H
        MOV DATABUS,#0C0H               ;DISPALY DATA AT THE
SECOND LINE
        ACALL   GUI_LENH

        MOV A,#11111110B
        LCALL GUI_DATA              ; 2 KHOANG TRANG
        MOV A,#11111110B
        LCALL GUI_DATA

        MOV A,#00110110B
        LCALL GUI_DATA
        MOV A,#11111110B
        LCALL GUI_DATA

        MOV A,#01010011B
        LCALL GUI_DATA
        MOV A,#01101111B
        LCALL GUI_DATA

        MOV A,#11111110B
        LCALL GUI_DATA
        MOV A,#01111110B
        LCALL GUI_DATA              ; DAU       ->

        MOV R1,#60H
        MOV 50H,#00H

QUET_PHIM_FUNC:
```

```
        LCALL       QUET_PHIM                   ; GOI CHUONG TRINH QUET
PHIM
        CJNE A,#10,PHIM_CLOSE
        LJMP OPEN

PHIM_CLOSE:
        CJNE A,#11,PHIM_NO
        LJMP QUET_PHIM_FUNC

PHIM_NO:
        CJNE A,#12,START
        LJMP QUET_PHIM_FUNC

START:
        CJNE A,#13,PHIM_CHANGE_PASS
        LJMP MAIN

PHIM_CHANGE_PASS:
        CJNE A,#14,PHIM_OOK
        LJMP CHANGE_PASS

PHIM_OOK:
        CJNE A,#15,PHIMDATA
        LJMP QUET_PHIM_FUNC

PHIMDATA:

        PUSH ACC
        MOV A,50H
        CJNE A,#6,HIENTHI
        LJMP QUET_PHIM_FUNC
HIENTHI:
        INC 50H
        POP ACC
        MOV @R1,A
        ORL A,#30H
        LCALL DELAY
        LCALL       GUI_DATA
        INC R1
        LJMP QUET_PHIM_FUNC
```

```
        ;==============================SENDING
PROGRAM==================
GUI_LENH:
        CLR RS                  ;RS=0-SEND COMMAND
        SJMP PULSE_EN
GUI_DATA:
        MOV DATABUS,A
        SETB RS                 ;RS = 1-SEND DATA
        NOP
PULSE_EN:
        CLR RW                  ;RW = 0-WRITING MODE
        CLR EN                  ;MAKE AN ENABLE PULSE
        NOP
        SETB EN
        NOP

        MOV TH0,#HIGH(-1000)
        MOV TL0,#LOW(-1000)
        LCALL DELAY_2
        RET
; = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
DELAY_2:
        MOV TMOD,#01H
        SETB TR0
        JNB TF0,$
        CLR TR0
        CLR TF0
        RET
;_____________________________________

OPEN:
MOV A,@R0                       ; PASS TRONG 0 70H DUOC DEM RA SO
SANH VOI THANG PASS DUOC NHAP VAO THU BAN PHIM
        CJNE A,60H,PASS_SAI
        INC   R0
        MOV A,@R0
        CJNE A,61H,PASS_SAI
        INC   R0
        MOV A,@R0
        CJNE A,62H,PASS_SAI
```

```
        INC   R0
        MOV A,@R0
        CJNE A,63H,PASS_SAI
        INC   R0
        MOV A,@R0
        CJNE A,64H,PASS_SAI
        INC   R0
        MOV A,@R0
        CJNE A,65H,PASS_SAI
;------DUNG--------
THONGBAO:                               ;HIEN THI RA MAN HINH
NHAP DUNG PASS WORD
        MOV DATABUS,#0CH        ;CHO PHEP LCD HIEN THI
        LCALL       GUI_LENH
        MOV DATABUS,#01H        ;XOA MANG HINH LCD
        LCALL       GUI_LENH
        MOV DATABUS,#84H        ;DU LIEU HIEN THI O DONG DAU
        LCALL       GUI_LENH

        MOV A,#11111110B                ;HIEN THI DAU CACH
        LCALL GUI_DATA
        MOV A,#11111110B
        LCALL GUI_DATA
        MOV A,#11111110B
        LCALL GUI_DATA

        MOV R4,#62
        MOV DPTR,#STRINGCODE
LOVE1:
        MOV A,R4
        MOVC        A,@A+DPTR
        LCALL       GUI_DATA
        INC   R4
        CJNE R4,#70,LOVE1     ; CORRECT

        LCALL       DELAY
        LCALL       DELAY
        CLR  P3.3
        MOV DATABUS,#01H
        LCALL       GUI_LENH
```

```
        PUSH ACC
        PUSH 04H
        LCALL DELAY
        MOV R4,#46
        MOV DPTR,#STRINGCODE
LOVE2:
        MOV A,R4
        MOVC    A,@A+DPTR
        LCALL   GUI_DATA
        INC   R4
        CJNE R4,#62,LOVE2           ;XUAT HIEN CAU "CUA DUOC MO RUI"
        LCALL AAAA


EXIT_OPEN:
        LCALL    QUET_PHIM
        CJNE A,#11,EXIT_OPEN    ; TRO VE LAN NHAP GIA TRI BAN DAU
        POP   04H
        POP   ACC
        LJMP MAIN
;------SAI------
PASS_SAI:
            INC   R6            ;R3 CHO PHEP SO LAN NHAP PASS SAI
        CJNE R6,#3,HOME
        LCALL   ALARM
        MOV R3,#00H
        LCALL DELAY50MS
        LJMP MAIN1
HOME:

        MOV DATABUS,#02H
        ACALL   GUI_LENH
        MOV DATABUS,#01H
        ACALL   GUI_LENH
        MOV DATABUS,#0CH        ;CHO PHEP LCD HIEN THI
        ACALL   GUI_LENH
        PUSH4H
        PUSHACC
        MOV R4,#15
        MOV DPTR,#STRINGCODE
STING:
```

```
        MOV A,R4                    ;HIEN THI PASS SAI ROI
        MOVC    A,@A+DPTR
        LCALL   GUI_DATA
        INC   R4
        CJNE R4,#30,STING
        POP   ACC
        POP   04H
        LCALL   DELAY
        LCALL   DELAY
        LCALL   DELAY
        LJMP MAIN
; DEN SANG CHOP NHAY KHI R3 >= 3 LAN NHAP
ALARM:
        MOV DATABUS,#2
        LCALL   GUI_LENH
        MOV DATABUS,#1
        LCALL   GUI_LENH
        MOV DATABUS,#0CH        ;lcd on
        LCALL   GUI_LENH
        MOV DATABUS,#80H
        LCALL   GUI_LENH
        PUSH04H
        PUSH    01H
        PUSHACC
        MOV R4,#31
        MOV DPTR,#STRINGCODE
LOOP4:
        MOV A,R4
        MOVC    A,@A+DPTR
        LCALL   GUI_DATA        ;DISPALY LINE:KEYPAD LOCKED
        INC   R4
        CJNE R4,#46,LOOP4

; ONE MINUTE
        MOV DATABUS,#0C0H           ; XUAT HIEN DONG DAU
TIEN
        ACALL   GUI_LENH
        MOV A,#11111110B            ;HIEN THI DAU CACH
        LCALL GUI_DATA
        MOV A,#11111110B
        LCALL GUI_DATA
```

```
        MOV A,#11111110B
        LCALL GUI_DATA
        MOV A,#11111110B
        LCALL GUI_DATA

        MOV A,#01001111B
        LCALL GUI_DATA
        MOV A,#01101110B
        LCALL GUI_DATA
        MOV A,#01100101B
        LCALL GUI_DATA

        MOV A,#11111110B                ;HIEN THI DAU CACH
        LCALL GUI_DATA
        MOV A,#01101101B
        LCALL GUI_DATA
        MOV A,#01101001B
        LCALL GUI_DATA
        MOV A,#01101110B
        LCALL GUI_DATA
        MOV A,#01110101B
        LCALL GUI_DATA
        MOV A,#01110100B
        LCALL GUI_DATA
        MOV A,#01100101B
        LCALL GUI_DATA
; DEN CHOP NHAY
        MOV   R1,#0
ANH:
        MOV         P2,#00H
        LCALL DELAY50MS
        MOV   P2,#0FFH
        LCALL DELAY50MS
        INC   R1
        CJNE        R1,#20,ANH
        LCALL THONG_BAO
                MOV P2,#0FFH
        POP ACC
        POP 01H
        POP 04H
        RET
```

```
; = = = = = = = = = = = = = = = = SET PASS = = = = = = = = = = = = = = = =
CHANGE_PASS:

        ;HIEN THI NHAP PASSWORD CU
        MOV DATABUS,#01H          ;clear lcd
        ACALL     GUI_LENH
        MOV DATABUS,#81H          ; First following character will appear
on first
        LCALL     GUI_LENH
        PUSHACC
        PUSH04H
        MOV R4,#0
        MOV DPTR,#DATACODE1
PING:
        MOV A,R4
        MOVC      A,@A+DPTR
        LCALL     GUI_DATA
        INC   R4
        CJNE R4,#14,PING
        POP   04H
        POP   ACC
        MOV DATABUS,#0C0H               ; First following character will
appear on second
        ACALL     GUI_LENH
        MOV A,#11111110B
        LCALL GUI_DATA            ; 2 KHOANG TRANG
        MOV A,#11111110B
        LCALL GUI_DATA

        MOV A,#00110110B
        LCALL GUI_DATA
        MOV A,#11111110B
        LCALL GUI_DATA

        MOV A,#01010011B
        LCALL GUI_DATA
        MOV A,#01101111B
        LCALL GUI_DATA
```

```
        MOV A,#11111110B
        LCALL GUI_DATA
        MOV A,#01111110B
        LCALL GUI_DATA              ; DAU        - >
        MOV 60H,00H
        MOV 61H,00H
        MOV 62H,00H
        MOV 63H,00H
        MOV 64H,00H
        MOV 65H,00H

DOI_PASS:                           ; CHI CO TAC DUNG KHI DOI
PASSWORD    PHIM START
        MOV R1,#60H
        MOV 51H,#00H
        MOV R0,#70H
KT_OPEN:
        LCALL       QUET_PHIM
        CJNE A,#10,KT_CLOSE
        SJMP KT_OPEN
KT_CLOSE:
        CJNE A,#11,KT_NO
        SJMP KT_OPEN
KT_NO:                      ; BUTTON NO CO TAC DUNG TRO VE MAN
HINH BAN DAU KHI KO MUON THAY DOI PASSWORD
        CJNE A,#12,KT_START
        SJMP CHANGE_PASS
KT_START:
        CJNE A,#13,KT_CHANGE_PASS
        JMP   MAIN
KT_CHANGE_PASS:
        CJNE A,#14,KT_OOK
        JMP   KT_OPEN
        KT_OOK:
        CJNE A,#15,PHIMDATA_1
        SJMP START_CHANGE
PHIMDATA_1:

        PUSH ACC
        MOV A,51H
        CJNE A,#6,HIENTHI_1
```

```
        LJMP KT_OPEN
HIENTHI_1:
        INC 51H
        POP ACC
        MOV @R1,A
        ORL A,#30H
        LCALL DELAY
        LCALL       GUI_DATA
        INC R1
        LJMP KT_OPEN
START_CHANGE:
        MOV A,R5
        CJNE A,#1,NHAP_LAI_PASS_CU
        MOV         R5,#00H
        MOV 70H,60H
        MOV 71H,61H
        MOV 72H,62H
        MOV 73H,63H
        MOV 74H,64H
        MOV 75H,65H
        LJMP MAIN
NHAP_LAI_PASS_CU:
                MOV A,@R0
        CJNE A,60H,LOI_DOI_PASS
        INC   R0
        MOV A,@R0
        CJNE A,61H,LOI_DOI_PASS
        INC   R0
        MOV A,@R0
        CJNE A,62H,LOI_DOI_PASS
        INC   R0
        MOV A,@R0
        CJNE A,63H,LOI_DOI_PASS
        INC   R0
        MOV A,@R0
        CJNE A,64H,LOI_DOI_PASS
        INC   R0
        MOV A,@R0
        CJNE A,65H,LOI_DOI_PASS
; HIEN THI NHAP PASS MOI
;*******************
```

```
        MOV DATABUS,#01H          ;clear lcd
        ACALL     GUI_LENH
        MOV DATABUS,#81H          ; First following character will appear
on first
        LCALL     GUI_LENH
        PUSH04H
        PUSH      ACC
        MOV R4,#0
        MOV DPTR,#DATACODE2
PINGMOI:
        MOV A,R4
        MOVC      A,@A+DPTR
        LCALL     GUI_DATA
        INC   R4
        CJNE R4,#15,PINGMOI
        POP   ACC
        POP   04H
        MOV DATABUS,#0C0H               ; First following character will
appear on second
        ACALL     GUI_LENH

        MOV A,#11111110B
        LCALL GUI_DATA            ; 2 KHOANG TRANG
        MOV A,#11111110B
        LCALL GUI_DATA

        MOV A,#00110110B
        LCALL GUI_DATA
        MOV A,#11111110B
        LCALL GUI_DATA

        MOV A,#01010011B
        LCALL GUI_DATA
        MOV A,#01101111B
        LCALL GUI_DATA

        MOV A,#11111110B
        LCALL GUI_DATA
        MOV A,#01111110B
        LCALL GUI_DATA            ; DAU       - >
```

```
EXIT_DUNG:
        INC   R5
        LJMP DOI_PASS

LOI_DOI_PASS:
        INC   R3
        CJNE R3,#3,HOMESET_1                    ; HIEN CAU NHAP
PASS CU
        LCALL       ALARM
        MOV R3,#00H
        LJMP CHANGE_PASS
HOMESET_1:

        MOV DATABUS,#2
        ACALL       GUI_LENH
        MOV DATABUS,#1
        ACALL       GUI_LENH
        MOV DATABUS,#0CH          ;CHO PHEP LCD HIEN THI
        ACALL       GUI_LENH
        PUSH04H
        PUSHACC
        MOV R4,#15
        MOV DPTR,#STRINGCODE          ;HIEN THI PASS SAI ROI
STINGSET:
        MOV A,R4
        MOVC        A,@A+DPTR
        ACALL       GUI_DATA
        INC   R4
        CJNE R4,#30,STINGSET
        POP   ACC
        POP   4H
        LCALL       DELAY
        LCALL       DELAY
        LCALL       DELAY
        LCALL       DELAY
        LJMP CHANGE_PASS

;*******************
; LAP TRINH QUET BAN PHIM NAY QUA HAY KHONG CHE VAO
DAU DUOC
```

```
QUET_PHIM:
        PUSH07H                 ; R7 DUOC DUNG DE LUU TRU CAC GIA
TRI COT
SCAN:
        MOV A,#0EFH             ;QUET COLUMN0
        MOV R7,#0               ;R7 = i
CONT:
        MOV P0,A                ;PORT2 NOI VOI BAN PHIM
        MOV A,P0                ;read row
        JNB   ACC.0,ROW_0
        JNB   ACC.1,ROW_1
        JNB   ACC.2,ROW_2
        JNB   ACC.3,ROW_3
        RL    A                 ;prepare to conect next col to ground
        INC   R7
        CJNE R7,#4,CONT                 ;4 col is conected to ground one after
another
        SJMP SCAN
ROW_0:                          ;row = 0,col = 7
        MOV A,R7
        ADD  A,#0               ;A = 0+R7
        SJMP EXIT
ROW_1:                          ;row = 1,col = R7
        MOV A,R7
        ADD  A,#4               ;A = 4+R7
        SJMP EXIT
ROW_2:
        MOV A,R7                ;row = 2,col = R7
        ADD  A,#8               ;A = 8+R7
        SJMP EXIT
ROW_3:
        MOV A,R7                ;row = 3,col = R7
        ADD  A,#12              ;A = 12+R7
        SJMP          EXIT
EXIT:
        MOV PASS,A

LOVE3:
        MOV A,P0
        ANL  A,#0FH
        CJNE A,#0FH,LOVE3
```

```
        MOV A,PASS
        ;MOV A,#00101010B
        POP   07H
        RET
;
****************************DELAY*************************
**************
DELAY:
        PUSH4H
        PUSH2H
        MOV R2,#0FFH
LOOPDE1:

        MOV R4,#0FFH
        DJNZ R4,$
        DJNZ R2,LOOPDE1
        POP   2H
        POP   4H
        RET
DELAY50MS:
        PUSH00H
        MOV R0,#50
        MOV TMOD,#01H
STINGA:
        MOV TH0,#0ECH
        MOV TL0,#78H
        SETB TR0
        JNB   TF0,$
        CLR   TR0
        CLR   TF0
        DJNZ R0,STINGA
        POP   00H
        RET


CHAY_CHU:

        ACALL      LCDINIT            ;KHOI TAO LCD
        ACALL      DISP_SLINE               ;HIEN THI DONG 2
MP1:          ;XU LY DICH CHUYEN DU LIEU HIEN THI CHO DONG 1
```

```
        MOV DPTR,#FLINE_DATA    ;NAP DIA CHI VUNG DU LIEU
DONG 1 CUA LCD
SHIFT:
        ACALL       DISP_FLINE              ;HIEN THI DONG 1
        MOV R1,#10              ;DELAY 500MS
DEL500:
        MOV TH0,#HIGH(-10000)
        MOV TL0,#LOW(-10000)
        ACALL       DELAYA
        DJNZ R1,DEL500          ;THOI GIAN DUNG YEN CUA MOT
TRANG THAI MAN HINH LCD
        INC   DPTR              ;TANG GIA TRI DPTR DE DICH CHUYEN
DONG CHU
        MOV A,DPL                       ;KIEM TRA DA DICH XONG DONG
CHU RA MAN HINH LCD
        CJNE A,#LOW(FLINE_DATA+80),SHIFT
        MOV A,DPH
        CJNE A,#HIGH(FLINE_DATA+80),SHIFT
        LJMP MAIN1

;************************************************
LCDINIT:   ;CTC KHOI TAO LCD
        CLR   RS                ;RS = 0 - GUI LENH
        CLR   RW                ;RW = 0 - WRITE LCD MODE
        SETB EN                 ;E = 1 - ENABLE
        MOV DATABUS,#38H  ;CODE = 38H - 8 BIT, 16 CHAR/LINE,
MATRIX 5x7
        ACALL       SENDCOMMAND             ;GUI LENH RA LCD
        MOV TH0,#HIGH(-4100)
        MOV TL0,#LOW(-4100)
        ACALL       DELAYA                  ;DELAY 4.1MS
        MOV DATABUS,#38H  ;CODE = 38H - 8 BIT, 16 CHAR/LINE,
MATRIX 5x7
        ACALL       SENDCOMMAND             ;GUI LENH RA LCD
        MOV TH0,#HIGH(-100)
        MOV TL0,#LOW(-100)
        ACALL       DELAYA                  ;DELAY 100US
        MOV DATABUS,#38H  ;CODE = 38H - 8 BIT, 16 CHAR/LINE,
MATRIX 5x7
        ACALL       SENDCOMMAND             ;GUI LENH RA LCD
        MOV DATABUS,#0CH  ;CODE = 0CH - CHO PHEP LCD HIEN THI
```

```
        ACALL   SENDCOMMAND             ;GUI LENH RA LCD
        MOV DATABUS,#01H  ;CODE = 01H - XOA LCD
        ACALL   SENDCOMMAND             ;GUI LENH RA LCD
        MOV DATABUS,#06H  ;CODE = 06H - TU TANG DIA CHI HIEN
THI, TAT DICH CHUYEN HIEN THI
        ACALL   SENDCOMMAND             ;GUI LENH RA LCD
        RET
;************************************************
SENDCOMMAND:    ;CTC GUI LENH (SENDCOMMAND) VA GUI
DU LIEU (SENDDATA) RA LCD
        CLR  RS                 ;RS = 0 - GUI LENH
        SJMP PULSE_ENA
SENDDATA:
        SETB RS                 ;RS = 1 - GUI DU LIEU
        NOP
PULSE_ENA:                      ;TAO XUNG ENABLE DE CHUYEN
THONG TIN (COMMAND/DATA) VAO LCD
        CLR  RW                 ;RW = 0 - WRITE LCD MODE
        CLR  EN                 ;EN = 0
        NOP
        SETB EN                 ;EN = 1 - XUNG ENABLE
        NOP
;KIEM TRA CO BAO BAN (BUSY FLAG) DE DAM BAO HOAN TAT
VIEC LCD GHI NHAN THONG TIN GUI DEN
        ;************************************************
****
        MOV TH0,#HIGH(-1000)    ;LENH NAY DUOC THAY THE
CHO DOAN MA KIEM TRA DUOI DAY KHI
        MOV TL0,#LOW(-1000);CHAY CHUONG TRINH NAY TRONG
PHAN MEM MO PHONG TOPVIEW
        ACALL DELAYA
        RET

;************************************************
DISP_FLINE:     ;CTC DAT DIA CHI BAT DAU DONG 1 VA NAP DU
LIEU DONG 1 VAO DDRAM
        MOV DATABUS,#80H  ;CODE = 80H - DAT DDRAM DIA CHI
BAT DAU CUA DONG 1 - 00H
        ACALL   SENDCOMMAND             ;GUI LENH RA LCD
        ACALL   WRITE                   ;GUI VUNG DU LIEU SANG
LCD
```

```
        RET
;*************************************************
DISP_SLINE:     ;CTC DAT DIA CHI BAT DAU DONG 2 VA NAP DU
LIEU DONG 2 VAO DDRAM
        MOV DATABUS,#0C0H       ;CODE = C0H - DAT DDRAM DIA
CHI BAT DAU CUA DONG 2 - 40H
        ACALL   SENDCOMMAND             ;GUI LENH RA LCD
        MOV DPTR,#SLINE_DATA    ;NAP DIA CHI VUNG DU LIEU
DONG 2 CUA LCD
        ACALL   WRITE                   ;GUI VUNG DU LIEU SANG
LCD
        RET
;*************************************************
WRITE:      ;CTC GUI DU LIEU SANG LCD, KET THUC GUI DU LIEU
KHI DU LIEU GUI DI LA 99H
        MOV R0,#0               ;OFFSET DAU TIEN TRONG VUNG DU
LIEU CUA DPTR
WR1:
        MOV A,R0                ;NAP OFFSET
        MOVC    A,@A+DPTR               ;LAY DU LIEU TU VUNG DU
LIEU
        MOV DATABUS,A           ;CHUYEN DU LIEU CAN GUI RA
PORT DEN LCD
        ACALL   SENDDATA                ;GUI DU LIEU RA LCD
        INC R0                  ;CHUYEN SANG DU LIEU KE TIEP
        CJNE R0,#16,WR1                 ;KIEM TRA NAP DAY DU DU LIEU
CHO MOT DONG MAN HINH - 16 CHU
        RET
;*************************************************
DELAYA:
        MOV TMOD,#01H
    SETB    TR0
        JNB TF0,$
    CLR     TR0
    CLR     TF0
        RET
FLINE_DATA:     ;DU LIEU HIEN THI DONG 1
        DB  '           '
        DB  'CHAO MUNG BAN DEN VOI HE THONG KHOA SO BANG
PASSWORD CUA NHOM 3 '
        DB  '           '
```

```
SLINE_DATA:     ;DU LIEU HIEN THI DONG 2
        DB    '    Welcome    '
THONG_BAO:
        ACALL       LCDINIT1            ;KHOI TAO LCD
        ACALL       DISP_SLINE1                 ;HIEN THI DONG 2
MP11:                   ;XU LY DICH CHUYEN DU LIEU HIEN THI CHO
DONG 1
        MOV DPTR,#FLINE_DATA1 ;NAP DIA CHI VUNG DU LIEU
DONG 1 CUA LCD
SHIFT1:
        ACALL       DISP_FLINE1                 ;HIEN THI DONG 1
        MOV R1,#5           ;DELAY 500MS
DEL5001:
        MOV TH0,#HIGH(-10000)
        MOV TL0,#LOW(-10000)
        ACALL       DELAY1
        DJNZ R1,DEL5001                 ;THOI GIAN DUNG YEN CUA MOT
TRANG THAI MAN HINH LCD
        INC    DPTR                 ;TANG GIA TRI DPTR DE DICH CHUYEN
DONG CHU
        MOV A,DPL                       ;KIEM TRA DA DICH XONG DONG
CHU RA MAN HINH LCD
        CJNE A,#LOW(FLINE_DATA1+102),SHIFT1
        MOV A,DPH
        CJNE A,#HIGH(FLINE_DATA1+102),SHIFT1
        RET
;*************************************************
LCDINIT1:  ;CTC KHOI TAO LCD
        CLR   RS                    ;RS = 0 - GUI LENH
        CLR   RW                    ;RW = 0 - WRITE LCD MODE
        SETB EN                     ;E = 1 - ENABLE
        MOV DATABUS,#38H  ;CODE = 38H - 8 BIT, 16 CHAR/LINE,
MATRIX 5x7
        ACALL       SENDCOMMAND1                ;GUI LENH RA LCD
        MOV TH0,#HIGH(-4100)
        MOV TL0,#LOW(-4100)
        ACALL       DELAY1                      ;DELAY 4.1MS
        MOV DATABUS,#38H  ;CODE = 38H - 8 BIT, 16 CHAR/LINE,
MATRIX 5x7
        ACALL       SENDCOMMAND1                ;GUI LENH RA LCD
        MOV TH0,#HIGH(-100)
```

```
        MOV TL0,#LOW(-100)
        ACALL   DELAY1                  ;DELAY 100US
        MOV DATABUS,#38H  ;CODE = 38H - 8 BIT, 16 CHAR/LINE,
MATRIX 5x7
        ACALL   SENDCOMMAND1            ;GUI LENH RA LCD
        MOV DATABUS,#0CH  ;CODE = 0CH - CHO PHEP LCD HIEN THI
        ACALL   SENDCOMMAND1            ;GUI LENH RA LCD
        MOV DATABUS,#01H  ;CODE = 01H - XOA LCD
        ACALL   SENDCOMMAND1            ;GUI LENH RA LCD
        MOV DATABUS,#06H  ;CODE = 06H - TU TANG DIA CHI HIEN
THI, TAT DICH CHUYEN HIEN THI
        ACALL   SENDCOMMAND1            ;GUI LENH RA LCD
        RET
;************************************************
SENDCOMMAND1:     ;CTC GUI LENH (SENDCOMMAND) VA GUI
DU LIEU (SENDDATA) RA LCD
        CLR  RS                 ;RS = 0 - GUI LENH
        SJMP PULSE_EN1
SENDDATA1:
        SETB RS                 ;RS = 1 - GUI DU LIEU
        NOP
PULSE_EN1:                      ;TAO XUNG ENABLE DE CHUYEN
THONG TIN (COMMAND/DATA) VAO LCD
        CLR  RW                 ;RW = 0 - WRITE LCD MODE
        CLR  EN                 ;EN = 0
        NOP
        SETB EN                 ;EN = 1 - XUNG ENABLE
        NOP
;KIEM TRA CO BAO BAN (BUSY FLAG) DE DAM BAO HOAN TAT
VIEC LCD GHI NHAN THONG TIN GUI DEN
        ;************************************************
****
        MOV TH0,#HIGH(-1000)        ;LENH NAY DUOC THAY THE
CHO DOAN MA KIEM TRA DUOI DAY KHI
        MOV TL0,#LOW(-1000);CHAY CHUONG TRINH NAY TRONG
PHAN MEM MO PHONG TOPVIEW
        ACALL DELAY1
        RET

;************************************************
```

```
DISP_FLINE1:     ;CTC DAT DIA CHI BAT DAU DONG 1 VA NAP DU
LIEU DONG 1 VAO DDRAM
        MOV DATABUS,#80H  ;CODE = 80H - DAT DDRAM DIA CHI
BAT DAU CUA DONG 1 - 00H
        ACALL     SENDCOMMAND1          ;GUI LENH RA LCD
        ACALL     WRITE1                ;GUI VUNG DU LIEU SANG
LCD
        RET
;*************************************************
DISP_SLINE1:     ;CTC DAT DIA CHI BAT DAU DONG 2 VA NAP DU
LIEU DONG 2 VAO DDRAM
        MOV DATABUS,#0C0H       ;CODE = C0H - DAT DDRAM DIA
CHI BAT DAU CUA DONG 2 - 40H
        ACALL     SENDCOMMAND1          ;GUI LENH RA LCD
        MOV DPTR,#SLINE_DATA1 ;NAP DIA CHI VUNG DU LIEU
DONG 2 CUA LCD
        ACALL     WRITE1                ;GUI VUNG DU LIEU SANG
LCD
        RET
;*************************************************
WRITE1:    ;CTC GUI DU LIEU SANG LCD, KET THUC GUI DU LIEU
KHI DU LIEU GUI DI LA 99H
        MOV R0,#0               ;OFFSET DAU TIEN TRONG VUNG DU
LIEU CUA DPTR
WR11:
        MOV A,R0                ;NAP OFFSET
        MOVC      A,@A+DPTR             ;LAY DU LIEU TU VUNG DU
LIEU
        MOV DATABUS,A           ;CHUYEN DU LIEU CAN GUI RA
PORT DEN LCD
        ACALL     SENDDATA1             ;GUI DU LIEU RA LCD
        INC   R0                ;CHUYEN SANG DU LIEU KE TIEP
        CJNE R0,#16,WR11        ;KIEM TRA NAP DAY DU DU LIEU
CHO MOT DONG MAN HINH - 16 CHU
        RET
;*************************************************
DELAY1:
        MOV TMOD,#01H
    SETB    TR0
    JNB     TF0,$
    CLR     TR0
```

```
    CLR     TF0
      RET
;************************************************
FLINE_DATA1:   ;DU LIEU HIEN THI DONG 1
      DB    '              '
      DB    ' Ban Da Quen Pass . Vui Long Dung Ngoai Cho . Neu La Trom
Thi Vui Long Pha Khoa . Hi'
      DB    '              '
SLINE_DATA1:   ;DU LIEU HIEN THI DONG 2
      DB    '  Keypad Locked '
AAAA:
      MOV DATABUS,#0C0H
      LCALL GUI_LENH

      MOV A,#11111110B                    ;HIEN THI DAU CACH
      LCALL GUI_DATA
      MOV A,#11111110B
      LCALL GUI_DATA
      MOV A,#11111110B
      LCALL GUI_DATA
      MOV A,#11111110B
      LCALL GUI_DATA


      MOV A,#01010111B
      LCALL GUI_DATA
      MOV A,#01100101B
      LCALL GUI_DATA
      MOV A,#01101100B
      LCALL GUI_DATA
      MOV A,#01100011B
      LCALL GUI_DATA
      MOV A,#01101111B
      LCALL GUI_DATA
      MOV A,#01101101B
      LCALL GUI_DATA
      MOV A,#01100101B
      LCALL GUI_DATA
RET
STRINGCODE:
      DB    ' Nhap Password '
```

```
        DB    '  Pass Sai Rui  '
        DB    '  Keypad Locked '
        DB   'Open The Door '
        DB    ' Correct '


DATACODE1:
        DB    ' Nhap Pass Cu '
DATACODE2:
        DB    ' Nhap Pass Moi '

END
```

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức:

Trình bày cấu tạo, đặc điểm,ứng dụng của các loại Vi điều khiển được học

Trình bày được sự cần thiết và cơ chế hoạt động của trình dịch hợp ngữ theo nội dung đã học.

Trình bày được cấu trúc chung của chương trình hợp ngữ theo nội dung đã học.

+ Về kỹ năng:

Lắp ráp các mạch ứng dụng từng phần do giáo viên đề ra.

Thực hiện viết các chương trình theo yêu cầu cho trước

+ Thái độ: Đánh giá phong cách, thái độ học tập

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm

+ Về kỹ năng: Đánh giá kỹ năng thực hành Mỗi sinh viên, hoặc mỗi nhóm học viên thực hiện công việc theo yêu cầu của giáo viên.Tiêu chí đánh giá theo các nội dung:

- Độ chính xác của công việc
- Tính thẩm mỹ của mạch điện
- Độ an toàn trên mạch điện
- Thời gian thực hiện công việc
- Độ chính xác theo yêu cầu kỹ thuật

+ Thái độ: Tỉ mỉ, cẩn thận, chính xác.

TÀI LIỆU THAM KHẢO:

[1] Đề cương môđun/môn học nghề Sửa chữa thiết bị điện tử công nghiệp", Dự án Giáo dục kỹ thuật và Dạy nghề (VTEP), Tổng cục Dạy Nghề, Hà Nội, 2003

[2] Microprocessor and IC families - Walter H. Buchbaum. Sc.D

[3] Mikrocompute Lehrbuch - HPI Fachbuchreihen Pflaum Verlag Munchen

[4] 8051 Development Boad, Rev 5 - Paul Stoffregen

[5] Họ vi điều khiển - Tống văn On - Đại học Bách khoa TP.HCM - 2005