BỘ XÂY DỰNG
TRƯỜNG CAO ĐẲNG XÂY DỰNG THÀNH PHỐ HỒ CHÍ MINH

GIÁO TRÌNH
LƯU HÀNH NỘI BỘ

LẬP TRINH VI ĐIỀU KHIỂN

TP. HỒ CHÍ MINH 2020

BÀI 1
SƠ LƯỢC VỀ LỊCH SỬ VÀ HƯỚNG PHÁT TRIỂN CỦA VI ĐIỀU KHIỂN

Giới thiệu:

Ứng dụng vi điều khiển để giải quyết các bài toán điều khiển cỡ nhỏ và cỡ trung hiện nay là khá phổ biến trong mọi lĩnh vực đời sống. Việc giới thiệu lịch sử ra đời và quá trình phát triển của vi điều khiển nhằm cung cấp cho người học tổng quan về vi điều khiển cũng như hướng phát triển của nó trong tương lai.

Mục tiêu:

- Hiểu lịch sử phát triển của vi điều khiển.
- Hiểu được cấu trúc chung của vi điều khiển.
- Biết được các lĩnh vực ứng dụng và hướng phát triển trong tương lai của vi điều khiển.

Nội dung chính:

1. Lịch sử phát triển

Mục tiêu:

- Biết được lịch sử ra đời của vi điều khiển

- Hiểu được quá trình phát triển của vi điều khiển

Phát minh ra transistor vào năm 1948 là thời điểm bắt đầu cho quá trình phát triển của máy tính với tính năng ngày càng cao và kích thước ngày càng nhỏ, linh kiện hội đủ 2 ưu điểm trên chính là vi xử lý. Máy tính điện tử đầu tiên của mỹ năm 1946 tên gọi ENIAC đã sử dụng 18.000 bóng đèn điện tử và sau đó năm 1960 được IBM thay thế bằng model 1410 với toàn bộ linh kiện là transistor. Vì chức năng phức tạp nên việc lắp ráp hệ thống cũng rất khó khăn và tốn kém, do đó đã phát sinh ý tưởng phải tìm cách thu nhỏ kích thước của các linh kiện rời như: transistor, diode, điện trở...và kết quả là sự ra đời của công nghệ vi mạch. Theo yêu cầu của các chuyên viên về tên lửa của cơ quan NASA luôn đòi hỏi tính ổn định và kích thước thật nhỏ nên vào năm 1958 Jack Kilby của hãng Texas instrument đã thiết kế được vi mạch đầu tiên và năm 1963 công ty Rockwell đã cho ra đời tên lửa Minerva II được chế tạo toàn bộ bằng vi mạch. Trong lĩnh vực dân sự vào năm 1961 công ty Fairchild lần đầu tiên giới thiệu một FF không dùng 2 hoặc 4 transistor rời mà được tích hợp trong một vi mạch đơn tinh thể. Các thế hệ vi mạch đầu tiên chỉ được sản xuất theo công nghệ

lưỡng cực, trong trường hợp cần nhiều lớp khuếch tán, nhiều lổ tiếp xúc và đường dẫn...giá thành có thể lên đến 10 - 20 đô la một mạch. Nhờ kỹ thuật MOS mật độ tích hợp được tăng cao hơn hẳn kỹ thuật lưỡng cực. Hướng phát triển tiếp theo sau đó là công nghệ CMOS bao gồm 2 transistor trường bổ túc làm giãm công suất tiêu thụ vì tại cùng một thời điểm luôn có 1 transistor bị khóa. Với yêu cầu ngày càng phức tạp và đa dạng làm cho việc sản xuất vi mạch với số lượng lớn cũng khó khăn, điều này dẫn đến một suy nghĩ mới về một vi mạch có khả năng lập trình, các vi mạch này có cấu tạo giống nhau và chức năng sẽ thay đổi sau khi lập trình V.D: Bằng phương pháp làm chảy các đường dẫn điện. Không bao lâu vào năm 1974 hãng INTEL đã sản xuất được chip vi xử lý đầu tiên lập trình theo yêu cầu của khách hàng mở đầu cho kỹ nguyên vi xử lý cũng còn được gọi là cuộc cách mạng công nghiệp lần thứ II.

2. Vi điều khiển

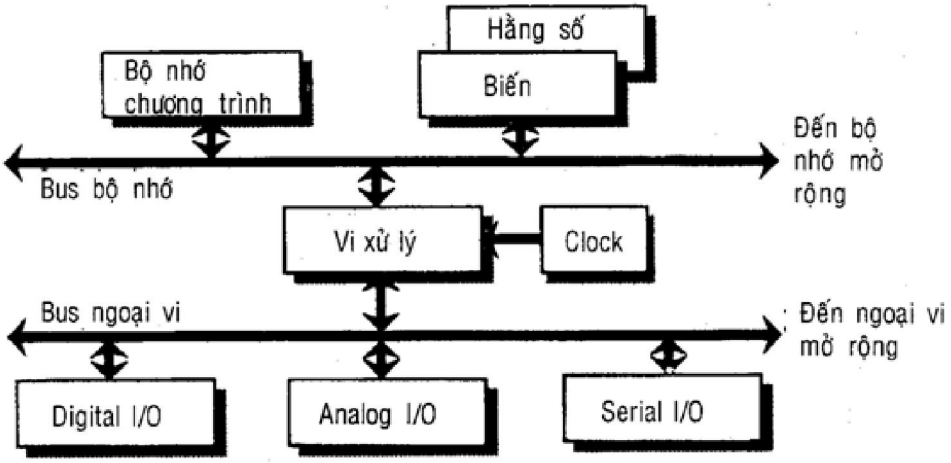
Mục tiêu:

- Hiểu được nguyên lý cấu tạo của vi điều khiển

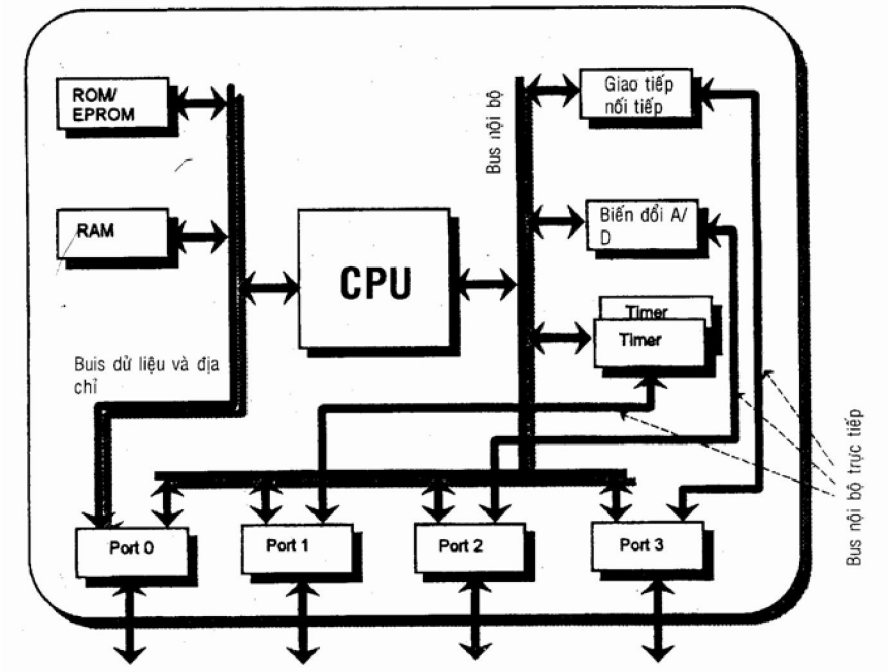
- Hiểu được các cấu trúc bộ nhớ của vi điều khiển

2.1. Nguyên lý cấu tạo

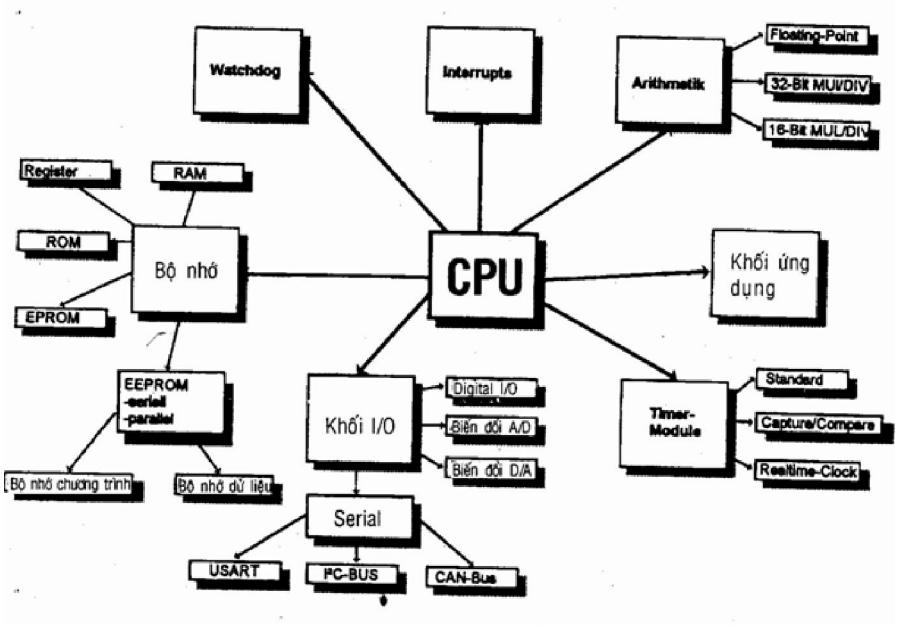
Điểm lưu ý về vi điều khiển là sơ đồ khối cấu tạo của nó. Cấu tạo một họ microcontroller chủ yếu dựa trên một kiểu tiêu chuẩn bao gồm các tính năng quan trọng nhất, nhiều chủng loại phù hợp với các lĩnh vực ứng dụng đặc biệt khác nhau, có thể kết hợp thêm thiết bị ngoại vi để tăng khả năng hoặc giảm nhỏ kích thước đến mức tối thiểu trong các ứng dụng chuyên biệt như: Kết nối bus, kết nối video hoặc điều khiển trực tiếp các cơ cấu hiển thị LCD...Với kiểu tiêu chuẩn cũng đủ dùng cho hầu hết các ứng dụng.



Hình 32-01-1 Cấu trúc máy tính



Hình 32-01-2 Cấu trúc vi điều khiển

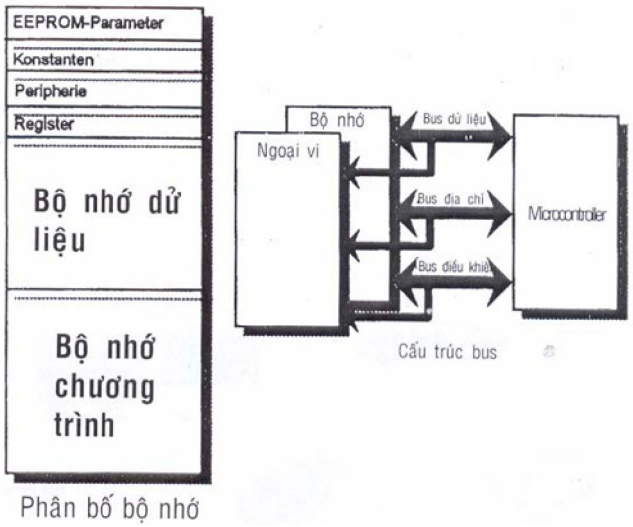


Hình 32-01-3 Sơ đồ khối vi điều khiển

2.2. Các kiểu cấu trúc bộ nhớ

2.2.1. Cấu trúc Von Neumann

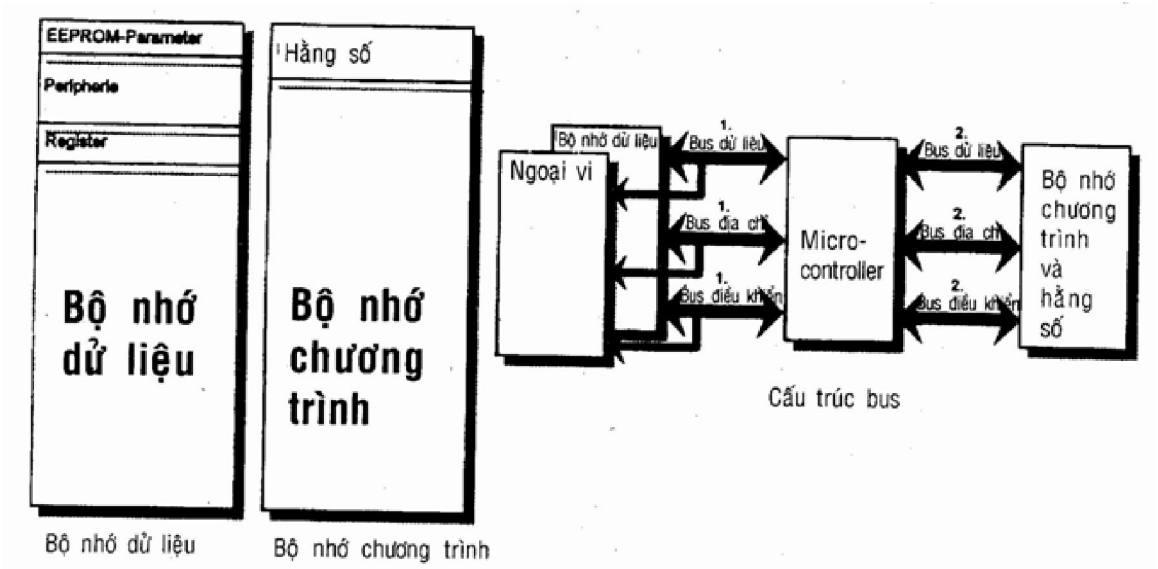
Trong cấu trúc Von Neumann chỉ có một vùng địa chỉ tuyến tính bao gồm tất cả dữ liệu và lệnh điều khiển, độ lớn của vùng địa chỉ phụ thuộc vào chiều dài của bộ đếm chương trình, nếu không trang bị thêm linh kiện phụ thì việc định địa chỉ bộ nhớ chương trình và bộ nhớ dữ liệu không độc lập với nhau. Trong cấu trúc này chỉ tồn tại một bus dữ liệu và một bus địa chỉ để đọc-ghi dữ liệu và đọc lệnh điều khiển chương trình và không có khả năng thực song song (truy xuất đồng thời bộ nhớ dữ liệu và bộ nhớ chương trình).



Hình 32-01-4. Cấu trúc Von Neumann

2.2.2. Cấu trúc Harvard

Gồm hai vùng địa chỉ riêng biệt cho bộ nhớ dữ liệu và bộ nhớ chương trình nên có thể truy xuất song song dữ liệu và lệnh điều khiển, cấu trúc này đặc biệt thích hợp với các vi điều khiển 16 và 32 bít vì làm tăng tốc độ làm việc. Nếu chỉ có một hệ thống bus như thường thấy ở vi điều khiển 8 bít thì việc truy xuất bộ nhớ dữ liệu hoặc bộ nhớ chương trình sẽ được thực hiện thông qua các tín hiệu điều khiển, nếu không có yêu cầu ghi vào bộ nhớ chương trình thì cấu trúc này còn cho phép tăng tính an toàn của chương trình.



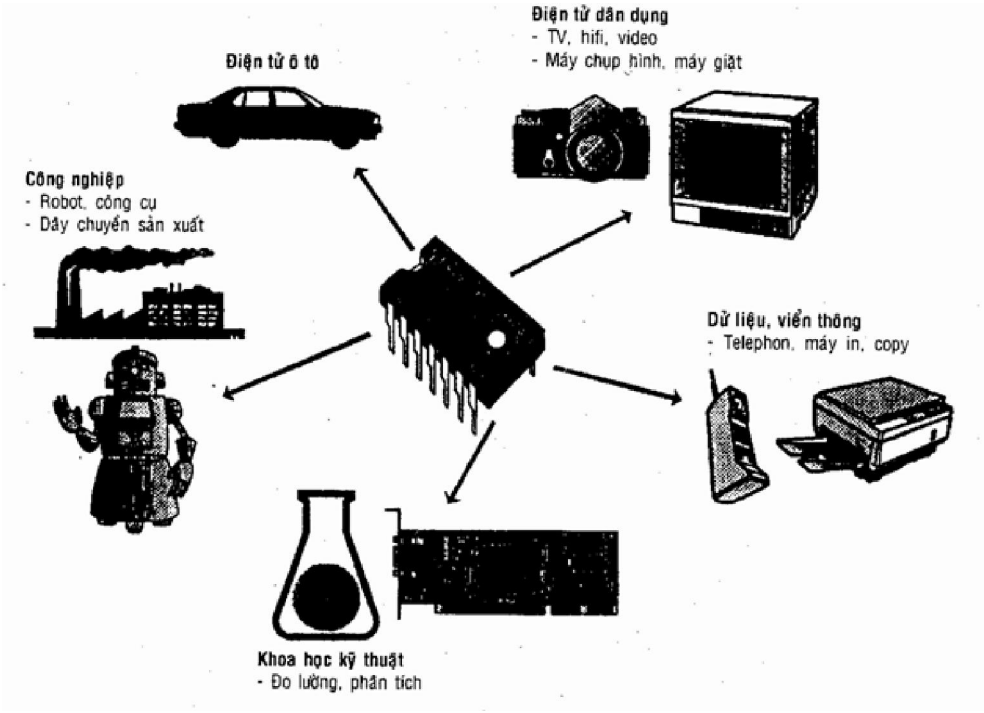
Hình 32-01-5. Cấu trúc Harvard

3. Lĩnh vực ứng dụng

Mục tiêu:

- Biết được lĩnh vực ứng dụng của vi điều khiển

Vi điều khiển hiện nay được ứng dụng trong nhiều lĩnh vực như: TV, thiết bị HiFi, máy giặt, điện thoại và trong ôtô... góp phần làm đơn giản hóa quá trình sử dụng với nhiều tính năng và độ an toàn cao hơn. Ngoài ra vi điều khiển còn được áp dụng trong lĩnh vực khoa học kỹ thuật như: các thiết bị phân tích và đo lường, trong công nghiệp như các dây chuyền sản xuất tự động, trong lĩnh vực máy công cụ như CNC và điều khiển chất lượng sản phẩm.



Hình 32-01-6. Lĩnh vực ứng dụng

4. Hướng phát triển

Mục tiêu: Nắm được hướng phát triển của vi điều khiển trong tương lai

Yêu cầu đặt ra cho vi điều khiển hiện nay là tăng lĩnh ứng dụng với tốc độ xử lý ngày càng nhanh và kích thước nhỏ gọn, công suất tiêu thụ thấp. Vấn đề đặt ra là liệu với vi điều khiển 8 bít có còn phù hợp hay không? hoặc trong tương lai phải thay bằng các vi điều khiển 16/32 bít. Khác với vi xử lý việc phát triển luôn kèm theo việc nâng cao khả năng tính toán bằng cách mở rộng hệ thống bus. Đối với vi điều khiển không nhất thiết phải như thế, một vi điều khiển 8 bít cũng đủ cho rất nhiều ứng dụng và vi điều khiển 16 bít là hoàn toàn quá dư thừa, trong trường hợp cần giảm giá thành, kích thước và công suất tiêu thụ thì vi điều khiển 4 bít là giải pháp tối ưu. Một vài ứng dụng cần vi điều khiển có nhiều khối ngoại vi, có ứng dụng lại cần ngoại vi tốc độ cao, hướng phát triển tương lai là tăng khả năng của CPU và khối ngoại vi.

Một hướng đơn giản là tăng tần số xung đồng hồ để rút ngắn thời gian thực hiện chương trình, giảm thời gian biến đổi A/D và tăng tần số giới hạn của timer. Tuy nhiên các linh kiện bên ngoài cũng phải có khả năng làm việc ở tần số cao, khi tăng tần số đồng cũng làm tăng công suất tiêu thụ của vi điều khiển. Việc tối ưu hóa cấu trúc chương trình và bộ nhớ cũng góp phần nâng cao khả năng hệ thống. Trong các

ứng dụng đa nhiệm, phương pháp phân đoạn và phân dãy hóa có một ý nghĩa rất lớn. Với công nghệ sản xuất mới có thể đồng thời tăng tần số làm việc và giảm công suất tiêu thụ và cả điện áp nuôi điều này sẽ mở ra các lĩnh vực ứng dụng mới trong đó mạch điện rất đơn giản và năng lượng tiêu thụ rất thấp, bằng cách thay đổi cú pháp tập lệnh thích hợp cho phép biên dịch dễ dàng từ các ngôn ngữ cấp cao như "C" hoặc "FORTH" sang mã lệnh của vi điều khiển.

BÀI 2
CẤU TRÚC VI ĐIỀU KHIỂN 8051

Giới thiệu: Vi điều khiển 8051 là một trong những họ vi điều khiển khá cơ bản và thông dụng. Việc nắm bắt cấu trúc phần cứng và các đặc điểm riêng của vi điều khiển loại này là tiền đề để người học hiểu rõ và thực hành tốt các kỹ năng lập trình ở các nội dung tiếp theo.

Mục tiêu:

- Hiểu được cấu trúc phần cứng vi điều khiển 8051.
- Hiểu được cấu trúc bộ nhớ, biết được cách truy xuất bộ nhớ dữ liệu và bộ nhớ chương trình.
- Hiểu được đặc tính của các thanh ghi đặc biệt.
- Biết cách mở rộng thêm bộ nhớ ngoài.
- Hiểu nguyên lý hoạt động của mạch reset.

Nội dung chính:

1. Cấu trúc phần cứng vi điều khiển 8051

Mục tiêu:

- Hiểu được đặc điểm chung của vi điều khiển

- Hiểu được sơ đồ khối của vi điều khiển

- Biết được chức năng các chân tín hiệu của vi điều khiển

1.1. Đặc điểm chung

Vi mạch tổng quát chung của họ MCS-51 là chip 8051, linh kiện đầu tiên của họ này được đưa ra thị trường. Chip 8051 có các đặc điểm như sau:

4 KB FLASH ROM, 128 Byte RAM nội.

4 Port xuất /nhập (8 bit.)

2 bộ định thời 16 bit.

Mạch giao tiếp nối tiếp.

Không gian nhớ chương trình ngoài 64KB.

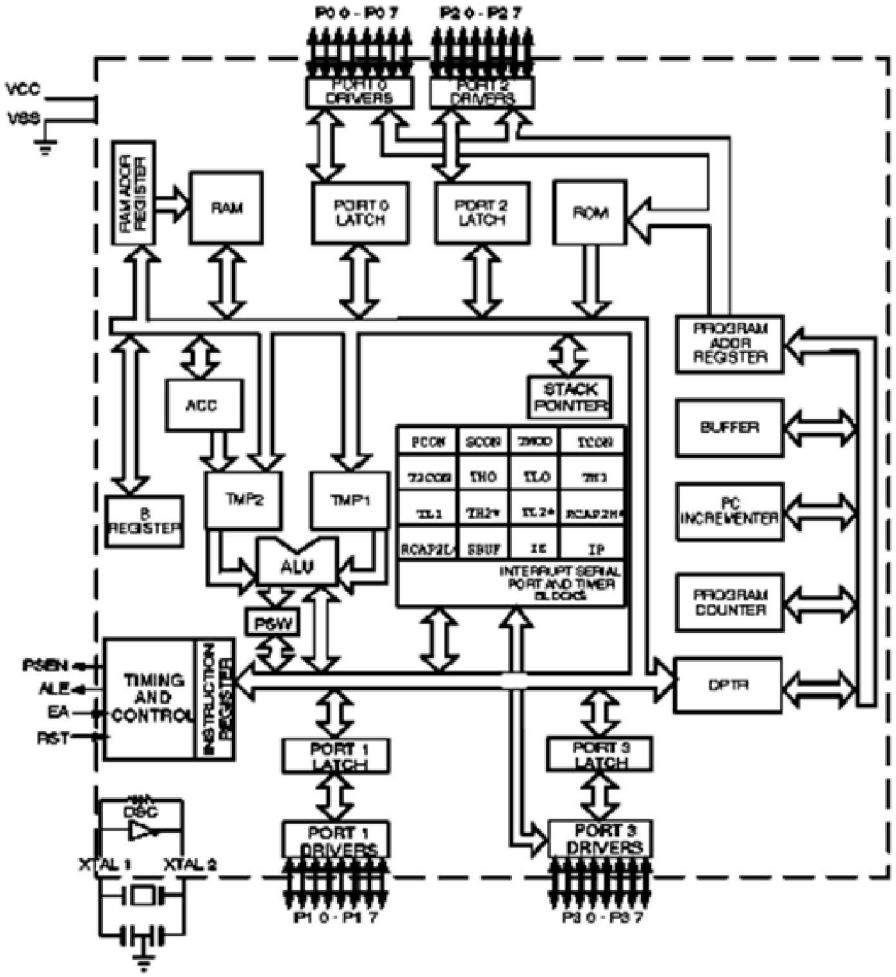
Không gian nhớ dữ liệu ngoài 64KB.

Bộ xử lý bit.

210 vị trí nhớ được định địa chỉ, mỗi vị trí 1 bit.

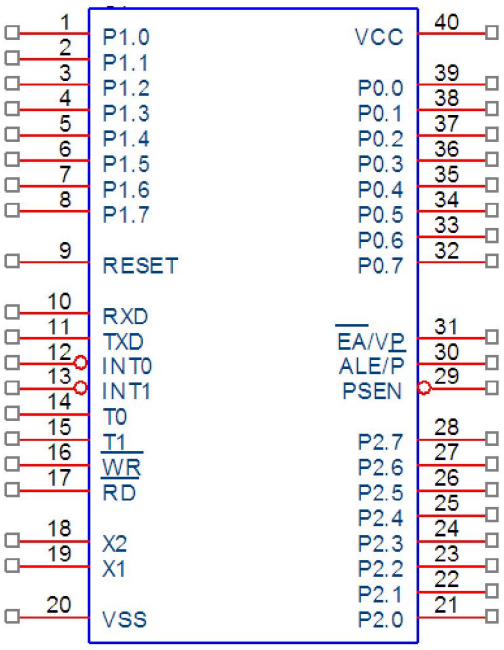
Các thành viên khác của họ MCS-51 có các tổ hợp ROM, RAM trên chip khác nhau hoặc có thêm bộ định thời thứ ba

1.2. Sơ đồ khối



Hình 32-02-1 Sơ đồ khối 8051

1.3. Sơ đồ chân



Hình 32-02-2 Sơ đồ chân 8051

*** Port 0**

Port 0 là port có 2 chức năng ở các chân 32 – 39 của 8051. Trong các thiết kế cỡ nhỏ không dùng bộ nhớ mở rộng nó có chức năng như các đường IO. Đối với các thiết kế cỡ lớn có bộ nhớ mở rộng, P0 là port đa hợp địa chỉ và dữ liệu.

*** Port 1**

Port 1 là port IO trên các chân 1-8. Các chân được ký hiệu P1.0, P1.1, , ….P1.7. Port 1 được dùng cho giao tiếp và điều khiển với các thiết bị bên ngoài.

*** Port 2**

Port 2 là port có tác dụng kép trên các chân 21 - 28 được dùng như các đường xuất nhập hoặc là byte cao của bus địa chỉ đối với thiết kế lớn có mở rộng port và bộ nhơ mở rộng.

*** Port 3**

Port 3 là port có tác dụng kép trên các chân 10 - 17. Các chân của port này có nhiều chức năng, các công dụng chuyển đổi có liên hệ với các đặc tính đặc biệt của 8051 như ở bảng sau:

Bit	Tên	Chức năng chuyển đổi
P3.0	RXT	Ngõ vào dữ liệu nối tiếp.
P3.1	TXD	Ngõ xuất dữ liệu nối tiếp.
P3.2	INT0\	Ngõ vào ngắt cứng thứ 0.
P3.3	INT1\	Ngõ vào ngắt cứng thứ 1.
P3.4	T0	Ngõ vào của timer/counter thứ 0.
P3.5	T1	Ngõ vào của timer/counter thứ 1.
P3.6	WR\	Tín hiệu ghi dữ liệu lên bộ nhớ ngoài.
P3.7	RD\	Tín hiệu đọc bộ nhớ dữ liệu ngoài.

*** Ngõ tín hiệu PSEN (Program store enable)**

PSEN là tín hiệu ngõ ra ở chân 29 có tác dụng cho phép đọc bộ nhớ chương trình mở rộng thường được nói đến chân 0E\ (output enable) của Eprom cho phép đọc các byte mã lệnh. PSEN ở mức thấp trong thời gian Microcontroller 8051 lấy lệnh. Các mã lệnh của chương trình được đọc từ Eprom qua bus dữ liệu và được chốt vào thanh ghi lệnh bên trong 8051 để giải mã lệnh. Khi 8951 thi hành chương trình trong ROM nội PSEN sẽ ở mức logic 1.

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

Khi 8051 truy xuất bộ nhớ bên ngoài, port 0 có chức năng là bus địa chỉ và bus dữ liệu do đó phải tách các đường dữ liệu và địa chỉ. Tín hiệu ra ALE ở chân thứ 30 dùng làm tín hiệu điều khiển để giải đa hợp các đường địa chỉ và dữ liệu khi kết nối chúng với IC chốt.

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

Các xung tín hiệu ALE có tốc độ bằng 1/6 lần tần số dao động trên chip và có thể được dùng làm tín hiệu clock cho các phần khác của hệ thống. Chân ALE được dùng làm ngõ vào xung lập trình cho Eprom trong 8051.

*** Ngõ tín hiệu EA\(External Access)**

Tín hiệu vào EA\ ở chân 31 nối nguồn 5VDC (mức 1) hoặc nối GND (mức 0). Nếu ở mức 1, 8051 thi hành chương trình từ ROM nội trong khoảng địa chỉ thấp 8 Kbyte. Nếu ở mức 0, 8051 sẽ thi hành chương trình từ bộ nhớ mở rộng.

Chân EA\ được lấy làm chân cấp nguồn 21V khi lập trình cho Eprom trong 8051.

*** Ngõ tín hiệu RST (Reset)**

Ngõ vào RST ở chân 9 là ngõ vào Reset của 8051. Khi ngõ vào tín hiệu này đưa lên cao ít nhất là 2 chu kỳ máy, các thanh ghi bên trong được nạp những giá trị thích hợp để khởi động hệ thống. Khi cấp điện mạch tự động Reset.

*** Các ngõ vào bộ dao động X1, X2**

Bộ dao động được tích hợp bên trong 8051, khi sử dụng 8051 người thiết kế chỉ cần kết nối thêm thạch anh và các tụ như hình vẽ trong sơ đồ. Tần số thạch anh thường sử dụng cho 8051 là 12MHz.

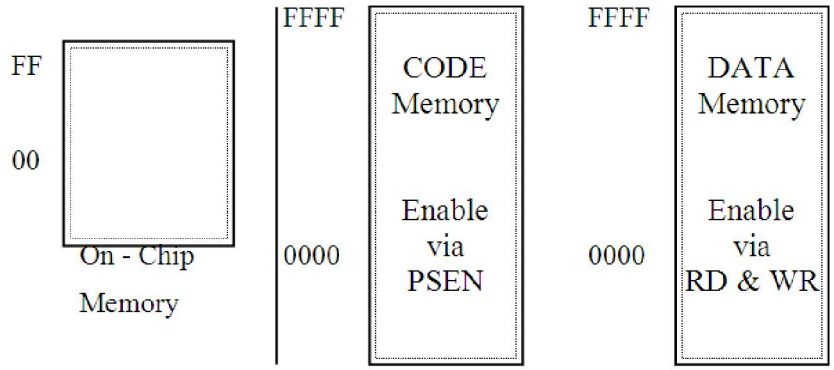
Chân 40 (Vcc) được nối lên nguồn 5V.

2. Cấu trúc bộ nhớ vi điều khiển 8051

Mục tiêu: - Biết được tổ chức các bộ nhớ trong vi điều khiển

- Biết được địa chỉ của RAM đa dụng và các thanh ghi

2.1. Tổ chức bộ nhớ



Hình 32-02-3 Tổ chức bộ nhớ 8051

Địa chỉ byte	Địa chỉ bit							
7F	RAM đa dụng							
.								
.								
.								
.								
30								
2F	7F	7E	7D	7C	7B	7A	79	78

Địa chỉ byte	Địa chỉ bit								Tên
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

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	Bank 2							
10								
0F	Bank 1							
08								
07	Bank thanh ghi 0 (R0..R7)							
00								
Vùng Ram								

B0	B7	B6	B5	B4	B3	B2	B1	B0	P.3
A8	AF			AC	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	Không được địa chỉ hoá bit								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	không được địa chỉ hoá bit								TH1
8C	không được địa chỉ hoá bit								TH0
8B	không được địa chỉ hoá bit								TL1
8A	không được địa chỉ hoá bit								TL0
89	không được địa chỉ hoá bit								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	không được địa chỉ hoá bit								PCON
83	không được địa chỉ hoá bit								DPH
82	không được địa chỉ hoá bit								DPL
81	không được địa chỉ hoá bit								SP
80	87	86	85	84	83	82	81	80	P0
Thanh ghi đặc biệt									

Hình 32-02-4 Cấu trúc RAM nội của 8051

Bộ nhớ trong 8051 bao gồm Rom và Ram. Ram trong 8051 bao gồm nhiều thành phần: phần lưu trữ đa dụng, phần lưu trữ định địa chỉ byte và bit, các băng thanh ghi và vùng các thanh ghi chức năng đặc biệt.

8051 có 4KByte bộ nhớ Rom nội. Với những thiết kế đòi hỏi dung lượng bộ nhớ, 8051 cho phép kết nối với 64K byte bộ nhớ chương trình và 64K byte dữ liệu ngoài.

2.2. RAM đa dụng

Trong vùng Ram từ địa chỉ 30 đến 7F đều có thể dùng chung mục đích sao chép và lưu trữ dữ liệu tạm thời. truy xuất dữ liệu tự do bằng các lệnh trực tiếp hoặc gián tiếp.

2.3. RAM có thể truy xuất từng bit

Vi điều khiển 8051 chứa 210 bit được địa chỉ hóa, trong đó có 128 bit có chứa các byte có chứa các địa chỉ từ 20F đến 2FH và các bit còn lại chứa trong nhóm thanh ghi có chức năng đặc biệt. Ý tưởng truy xuất từng bit bằng phần mềm là các đặc tính mạnh của microcontroller xử lý chung. Các bit có thể được đặt, xóa, AND, OR, . . . , với 1 lệnh đơn. Đa số các microcontroller xử lý đòi hỏi một chuỗi lệnh đọc – sửa - ghi để đạt được mục đích tương tự. Ngoài ra các port cũng có thể truy xuất được từng bit, 128 bit truy xuất từng bit này cũng có thể truy xuất như các byte hoặc như các bit phụ thuộc vào lệnh được dùng.

2.4. Các bank thanh ghi

32 byte thấp của bộ nhớ nội được dành cho các bank thanh ghi. Bộ lệnh 8051 hổ trợ 8 thanh ghi có tên là R0 đến R7 và theo mặc định sau khi reset hệ thống, các thanh ghi này có các địa chỉ từ 00H đến 07H.

Các lệnh dùng các thanh ghi R0 đến R7 sẽ ngắn hơn và nhanh hơn so với các lệnh có chức năng tương ứng dùng kiểu địa chỉ trực tiếp. Các dữ liệu được dùng thường xuyên nên dùng một trong các thanh ghi này.

Có 4 bank thanh ghi có tên R0 đến R7 (khác nhau địa chỉ trực tiếp). Để chuyển đổi việc truy xuất các bank thanh ghi ta phải thay đổi các bit chọn bank trong thanh ghi trạng thái.

3. Các thanh ghi chức năng đặc biệt

Mục tiêu:

- Biết được chức năng các thanh ghi đặc biệt

- Biết được địa chỉ, ký hiệu các bit trong các thanh ghi đặc biệt

3.1. Thanh ghi trạng thái chương trình

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

BIT	SYMBOL	ADDRESS	DESCRIPTION
PSW.7	CY	D7H	Cary Flag
PSW.6	AC	D6H	Auxiliary Cary Flag
PSW.5	F0	D5H	Flag 0
PSW4	RS1	D4H	Register Bank Select 1
PSW.3	RS0	D3H	Register Bank Select 0
			00=Bank 0; address 00H÷07H
			01=Bank 1; address 08H÷0FH
			10=Bank 2; address 10H÷17H
			11=Bank 3; address 18H÷1FH
PSW.2	OV	D2H	Overlow Flag
PSW.1	-	D1H	Reserved
PSW.0	P	DOH	Even Parity Flag

Chức năng từng bit trạng thái chương trình.

Cờ Carry CY (Carry Flag): Cờ nhớ có tác dụng kép. Thông thường nó được dùng cho các lệnh toán học: C=1 nếu phép toán cộng có sự tràn hoặc phép trừ có mượn và ngược lại C= 0 nếu phép toán cộng không tràn và phép trừ không có mượn.

Cờ Carry phụ AC (Auxiliary Carry Flag): Khi cộng những giá trị BCD (Binary Code Decimal), cờ nhớ phụ AC được set nếu kết quả 4 bit thấp nằm trong phạm vi điều khiển 0AH÷ 0FH. Ngược lại AC= 0.

Cờ 0 (Flag 0):Cờ 0 (F0) là 1 bit cờ đa dụng dùng cho các ứng dụng của người dùng.

BIT RS0 & RS1: Những bit chọn bank thanh ghi truy xuất RS1 và RS0 quyết định dãy thanh ghi tích cực. Chúng được xóa sau khi reset hệ thống và được thay đổi bởi phần mềm khi cần thiết. Tùy theo RS1, RS0 = 00, 01, 10, 11 sẽ được chọn Bank tích cực tương ứng là Bank 0, Bank1, Bank2, Bank3.

RS1	RS0	BANK
0	0	0
0	1	1
1	0	2
1	1	3

Cờ tràn OV (Over Flag): Cờ tràn được set sau một hoạt động cộng hoặc trừ nếu có sự tràn toán học. Khi các số có dấu được cộng hoặc trừ với nhau, phần mềm có thể kiểm tra bit này để xác định xem kết quả có nằm trong tầm xác định không. Khi các số không có dấu được cộng bit OV được bỏ qua. Các kết quả lớn hơn +127 hoặc nhỏ hơn –128 thì bit OV = 1.

Bit Parity (P): Bit tự động được set hay Clear ở mỗi chu kỳ máy để lập Parity chẵn với thanh ghi A. Sự đếm các bit 1 trong thanh ghi A cộng với bit Parity luôn luôn

chẵn. Ví dụ A chứa 10101101B thì bit P set lên một để tổng số bit 1 trong A và P tạo thành số chẵn. Bit Parity thường được dùng trong sự kết hợp với những thủ tục của Port nối tiếp để tạo ra bit Parity trước khi phát đi hoặc kiểm tra bit Parity sau khi thu.

3.2. Thanh ghi B

Thanh ghi B ở địa chỉ F0H được dùng cùng với thanh ghi A cho các phép toán nhân chia. Lệnh MUL AB sẽ nhân những giá trị không dấu 8 bit trong hai thanh ghi A và B, rồi trả về kết quả 16 bit trong A (byte cao) và B(byte thấp). Lệnh DIV AB lấy A chia B, kết quả nguyên đặt vào A, số dư đặt vào B. Thanh ghi B có thể được dùng như một thanh ghi đệm trung gian đa mục đích. Nó là những bit định vị thông qua những địa chỉ từ F0H÷F7H.

3.3. Con trỏ Ngăn xếp SP (Stack Pointer)

Con trỏ ngăn xếp là một thanh ghi 8 bit ở địa chỉ 81H. Nó chứa địa chỉ của byte dữ liệu hiện hành trên đỉnh ngăn xếp. Các lệnh trên ngăn xếp bao gồm các lệnh cất dữ liệu vào ngăn xếp (PUSH) và lấy dữ liệu ra khỏi Ngăn xếp (POP). Lệnh cất dữ liệu vào ngăn xếp sẽ làm tăng SP trước khi ghi dữ liệu và lệnh lấy ra khỏi ngăn xếp sẽ làm giảm SP. Ngăn xếp của 8031/8051 được giữ trong RAM nội và giới hạn các địa chỉ có thể truy xuất bằng địa chỉ gián tiếp, chúng là 128 byte đầu của 8951.

3.4. Con trỏ dữ liệu DPTR (Data Pointer)

Con trỏ dữ liệu (DPTR) được dùng để truy xuất bộ nhớ ngoài là một thanh ghi 16 bit ở địa chỉ 82H (DPL: byte thấp) và 83H (DPH: byte cao)

3.5. Các thanh ghi Port (Port Register)

Các Port của 8051 bao gồm Port0 ở địa chỉ 80H, Port1 ở địa chỉ 90H, Port2 ở địa chỉ A0H, và Port3 ở địa chỉ B0H. Tất cả các Port này đều có thể truy xuất từng bit nên rất thuận tiện trong khả năng giao tiếp.

3.6. Các thanh ghi Timer (Timer Register)

Vi điều khiển 8051 có chứa hai bộ định thời/ bộ đếm 16 bit được dùng cho việc định thời được đếm sự kiện. Timer0 ở địa chỉ 8AH (TLO: byte thấp) và 8CH (THO: byte cao). Timer1 ở địa chỉ 8BH (TL1: byte thấp) và 8DH (TH1: byte cao). Việc khởi

động timer được SET bởi Timer Mode (TMOD) ở địa chỉ 89H và thanh ghi điều khiển Timer (TCON) ở địa chỉ 88H. Chỉ có TCON được địa chỉ hóa từng bit.

3.7. Các thanh ghi Port nối tiếp (Serial Port Register)

Vi điều khiển 8051 chứa một Port nối tiếp cho việc trao đổi thông tin với các thiết bị nối tiếp như máy tính, modem hoặc giao tiếp nối tiếp với các IC khác. Một thanh ghi đệm dữ liệu nối tiếp (SBUF) ở địa chỉ 99H sẽ giữ cả hai dữ liệu truyền và dữ liệu nhập. Khi truyền dữ liệu ghi lên SBUF, khi nhận dữ liệu thì đọc SBUF. Các mode vận khác nhau được lập trình qua thanh ghi điều khiển Port nối tiếp (SCON) được địa chỉ hóa từng bit ở địa chỉ 98H.

3.8. Các thanh ghi ngắt (Interrupt Register)

Vi điều khiển 8051 có cấu trúc 5 nguồn ngắt, 2 mức ưu tiên. Các ngắt bị cấm sau khi bị reset hệ thống và sẽ được cho phép bằng việc ghi thanh ghi cho phép ngắt (IE) ở địa chỉ A8H. B8H là địa của thanh ghi ưu tiên ngắt. Cả hai được địa chỉ hóa từng bit.

3.9. Thanh ghi điều khiển nguồn PCON (Power Control Register)

Thanh ghi PCON không có bit định vị. Nó ở địa chỉ 87H chứa nhiều bit điều khiển. Thanh ghi PCON được tóm tắt như sau:

Bit 7 (SMOD) : Bit có tốc độ Baud ở mode 1, 2, 3 ở Port nối tiếp khi set.

Bit 6, 5, 4 : Không có địa chỉ.

Bit 3 (GF1) : Bit cờ đa năng 1.

Bit 2 (GF0) : Bit cờ đa năng 2 .

Bit 1 (PD) : Set để khởi động mode Power Down và thoát để reset.

Bit 0 (IDL) : Set để khởi động mode Idle và thoát khi ngắt mạch hoặc reset.

Các bit điều khiển Power Down và Idle có tác dụng chính trong tất cả các

IC họ MSC-51 nhưng chỉ được thi hành trong sự biên dịch của CMOS.

4. Bộ nhớ ngoài

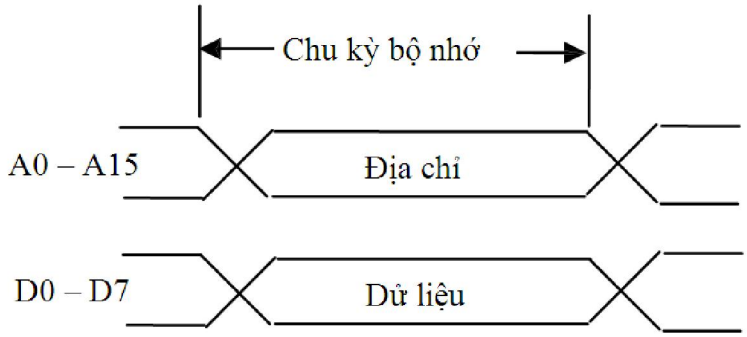
Mục tiêu:

- Biết cách truy xuất bộ nhớ chương trình ngoài

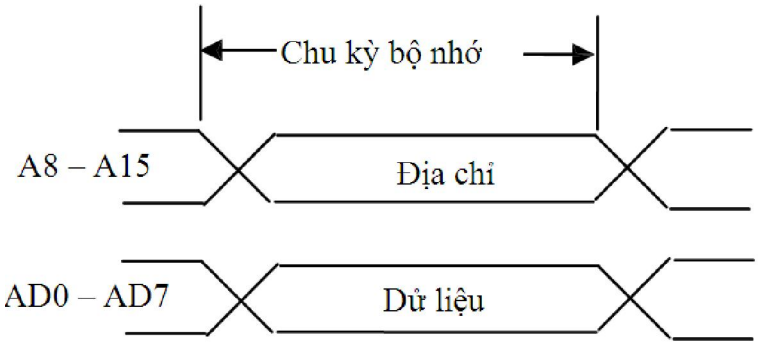
- Biết cách truy xuất bộ nhớ dữ liệu ngoài

Các bộ vi điều khiển cần có khả năng mở rộng các tài nguyên trên chip (bộ nhớ, I/O...) để tránh hiện tượng cổ chai trong thiết kế. Cấu trúc của MCS-51 cho phép khả năng mở rộng không gian bộ nhớ chương trình đến 64 K và không gian bộ nhớ dữ liệu đến 64 K ROM và RAM ngoài được thêm vào khi cần. Các IC giao tiếp ngoại vi cũng có thể được thêm vào để mở rộng khả năng xuất/nhập. Chúng trở thành một phần của không gian bộ nhớ dữ liệu ngoài bằng cách sử dụng cách định địa chỉ kiểu I/O ánh xạ bộ nhớ. Khi bộ nhớ ngoài được sử dụng, port 0 không làm nhiệm vụ của port xuất/nhập, port này trở thành bus địa chỉ (A0..A7) và bus dữ liệu (D0..D7) đa hợp. Ngõ ra ALE chốt một byte thấp của địa chỉ ở thời điểm bắt đầu một chu kỳ bộ nhớ ngoài. Port 2 thường (nhưng không phải luôn luôn) được dùng làm byte cao của bus địa chỉ.

Trước khi nghiên cứu cụ thể về bus địa chỉ và dữ liệu đa hợp ý tưởng tổng quat được trình bày ở hình 2.5



a. Không đa hợp 24 chân



b. Có đa hợp 16 chân

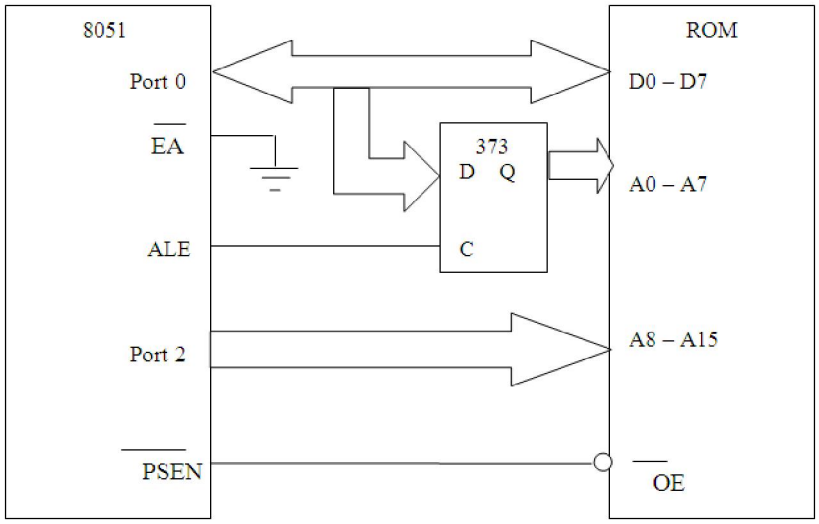
Hình 32-02-5 Bus đa hợp địa chỉ/dữ liệu

Sắp xếp không đa hợp sử dụng 16 đường địa chỉ và 8 đường dữ liệu tổng cộng 24 đường. Sắp xếp đa hợp kết hợp 8 đường của bus dữ liệu và byte thấp của bus địa chỉ thì chỉ cần 16 đường. Việc tiết kiệm các chân cho phép đóng gói họ MCS-51 trong một vỏ 40 chân.

Sắp xếp đa hợp hoạt động như sau: Trong ½ chu kỳ đầu của chu kỳ bộ nhớ, byte thấp của địa chỉ được cung cấp bởi port 0 và được chốt nhờ tín hiệu ALE. Mạch chốt 74373 giữ cho byte thấp của địa chỉ ổn định trong cả chu kỳ bộ nhớ. Trong ½ sau của chu kỳ bộ nhớ, Port 0 được sử dụng làm bus dữ liệu và dữ liệu được đọc hay ghi.

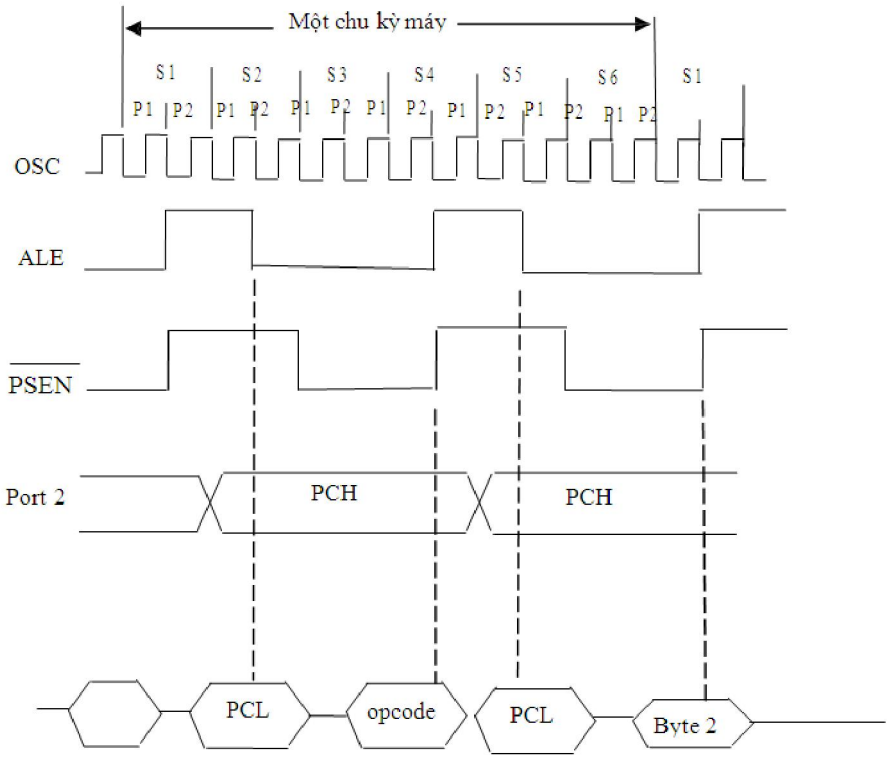
4.1. Truy xuất bộ nhớ chương trình ngoài

Bộ nhớ chương trình ngoài là bộ nhớ chỉ đọc, được cho phép bởi tín hiệu PSEN. Khi có một ROM ngoài được xử dụng, cả hai port 0 và port 2 đều không còn là các port xuất/nhập. Kết nối phần cứng với bộ ngoài được trình bày ở hình 32-02-6



Hình 32-02-6 Truy xuất ROM ngoài

Một chu kỳ máy của 8051 có 12 chu kỳ dao động. Nếu bộ dao động trên chíp có tần số 12 MHz thì một chu kỳ máy dài 1us. Trong 1 chu kỳ máy điển hình, ALE có hai xung và 2 byte của lệnh được đọc từ bộ nhớ chương trình (nếu lệnh chỉ có 1 byte, byte thứ hai bị loại bỏ). Giản đồ thời gian của chu kỳ máy này được gọi là chu kỳ tìm-nạp lệnh được trình bày ở hình 2.7.

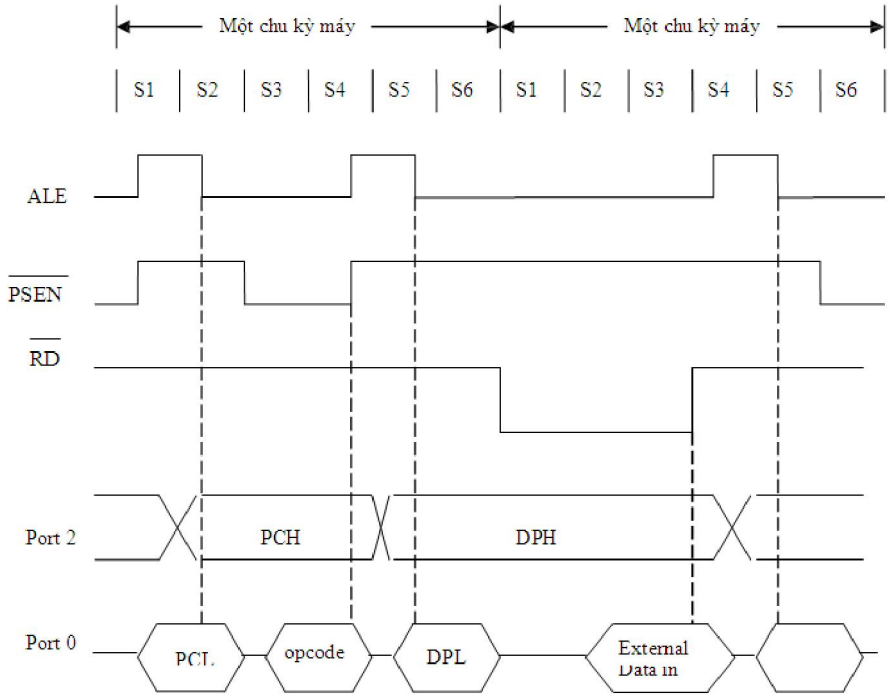


Hình 32-02-7 Chu kỳ tìm nạp lệnh ROM ngoài

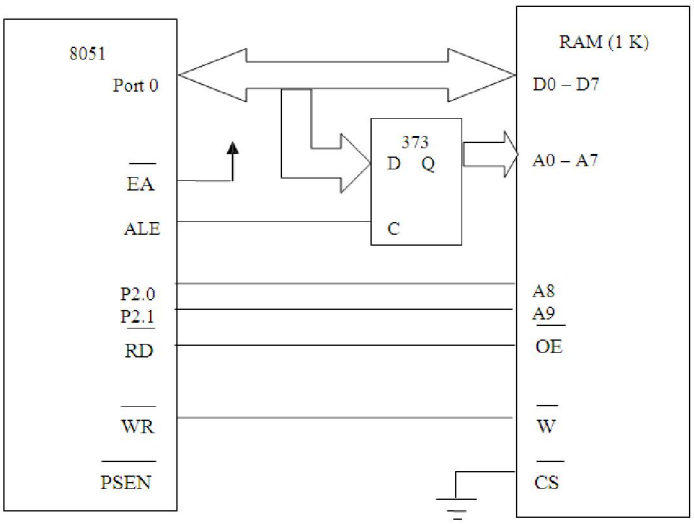
4.2. Truy xuất bộ nhớ dữ liệu ngoài

Bộ nhớ dữ liệu ngoài là bộ nhớ đọc/ghi được cho phép bởi các tín hiệu /RD và /WR ở các chân P3.7 và P3.6. Lệnh dùng để truy xuất bộ nhớ dữ liệu ngoài là MOVX, sử dụng hoặc con trỏ dữ liệu 16 bit DPTR hoặc R0, R1 làm thanh ghi chứa địa chỉ.

RAM có thể giao tiếp với 8051 theo cách như EPROM ngoại trừ đường /RD nối với đường cho phép xuất (/OE) của RAM và /WR nối với đường ghi (/W) của RAM. Các kết nối với bus dữ liệu và bus địa chỉ giống như EPROM. Bằng cách sử dụng các port 0 và port 2 ta có 1 dung lượng RAM ngoài lên tới 64K được kết nối với 8051.
Giản đồ thời gian của thao tác đọc dữ liệu ở bộ nhớ dữ liệu ngoài được trình bày ở hình 2.8 cho lệnh MOVX



Hình 32-02-8 Giản đồ thời gian lệnh MOVX



Hình 32-02-9 Giao tiếp với 1kB RAM

Port 2 giảm bớt được chức năng làm nhiệm vụ cung cấp byte cao của địa chỉ trong các hệ thống tối thiểu hóa thành phần, hệ thống không dùng bộ nhớ chương trình ngoài và chỉ có một dung lượng nhỏ bộ nhớ dữ liệu ngoài. Các địa chỉ 8 bít có thể truy xuất bộ nhớ dữ liệu ngoài với cấu hình bộ nhớ nhỏ hướng trang. Nếu có nhiều hơn một trang 256 byte RAM, một vài bít từ port 2 hoặc một port khác có thể chọn một trang. Thí dụ với 1RAM 1KB (nghĩa là 4 trang 256 byte), ta có thể kết nối RAM này với 8051 như hinh 2.9.

5. Hoạt động reset

Mục tiêu:

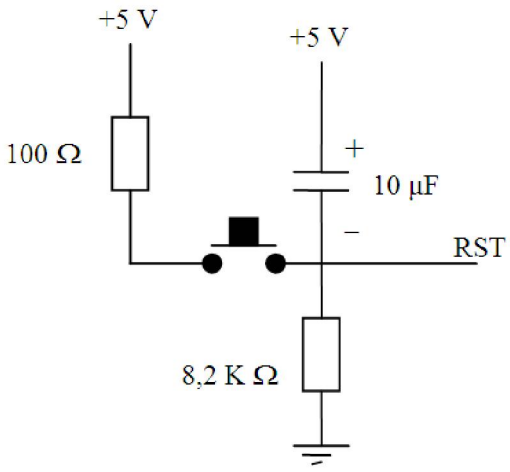
- Hiểu và vẽ được các cách reset ở vi điều khiển

- Biết được giá trị các thanh ghi sau khi reset

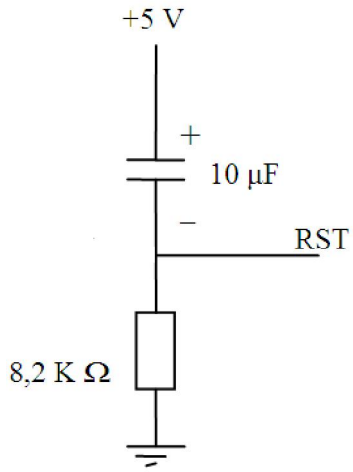
8051 được reset bằng cách giữ chân RST ở mức cao tối thiểu 2 chu kỳ máy và sau đó chuyển về mức thấp. RST có thể được tác động bằng tay hoặc được tác động khi cấp nguồn bằng cách dùng một mạch RC như trình bày ở hình 2.10. Trạng thái của tất cả các thanh ghi sau khi reset hệ thống được tóm tắt ở bảng sau. Quan trọng nhất trong các thanh ghi này có lẻ là thanh ghi PC (bộ đếm chương trình) được nạp 0000H. Khi RST trở lại mức thấp, việc thực hiện chương trình luôn luôn bắt đầu ở vị trí đầu tiên trong bộ nhớ chương trình đó chính là địa chỉ 0000H, nội dung của RAM trên chíp không bị ảnh hưởng bởi reset

Bảng Giá trị của các thanh ghi sau khi reset hệ thống

Thanh ghi	Nội dung
Bộ đếm chương trinh	0000H
Thanh ghi A	00H
Thanh ghi B	00H
PSW	00H
SP	07H
DPTR	0000H
Port 0..3	F
IP	F H
IE	xxx00000B (8031/8051) xx000000B (8032/8052)
Thanh ghi định thời	0xx00000B (8031/8051)
SCON	0x000000B (8032/8052)
SBUF	00H
PCON (HMOS)	00H
PCON (CMOS)	00H 0xxxxxxxB



a. Reset bằng tay



b. Reset khi cấp nguồn

Hình 32-02-10 Sơ đồ mạch RESET

TẬP LỆNH VI ĐIỀU KHIỂN 8051

Mục tiêu:

- Hiểu và phân biệt được các kiểu định địa chỉ dữ liệu
- Biết được đặc tính và công dụng của từng lệnh trong 8051
- Xác định được độ lớn cũng như thời gian thực hiện của chương trình
- Kết hợp được các lệnh riêng lẻ để viết chương trình.

Nội dung chính:

1. Các cách định địa chỉ

Mục tiêu:

- Biết được các cách định địa chỉ của vi điều khiển

- Ứng các cách định địa chỉ vào trong câu lệnh

1.1. Định địa chỉ thanh ghi

MCS 8051 có 8 thanh ghi làm việc được đánh số từ R0 đến R7, trong các lệnh áp dụng cách định địa chỉ này, thanh ghi được mã hóa bởi 3 bít trong byte mã lệnh như trình bày ở hình 3.1a. Trong cú pháp hợp ngữ của 8051 các thanh ghi được ký hiệu là Rn (n = 0..7) VD lệnh sau đây sẽ cộng nội dung bộ tích lũy với nội dung của thanh ghi R7

ADD A, R7

Mã lệnh tương ứng là 00101111B, 5 bít cao 00101 là mã lệnh cộng và 3 bít thấp 111 là mã của R7

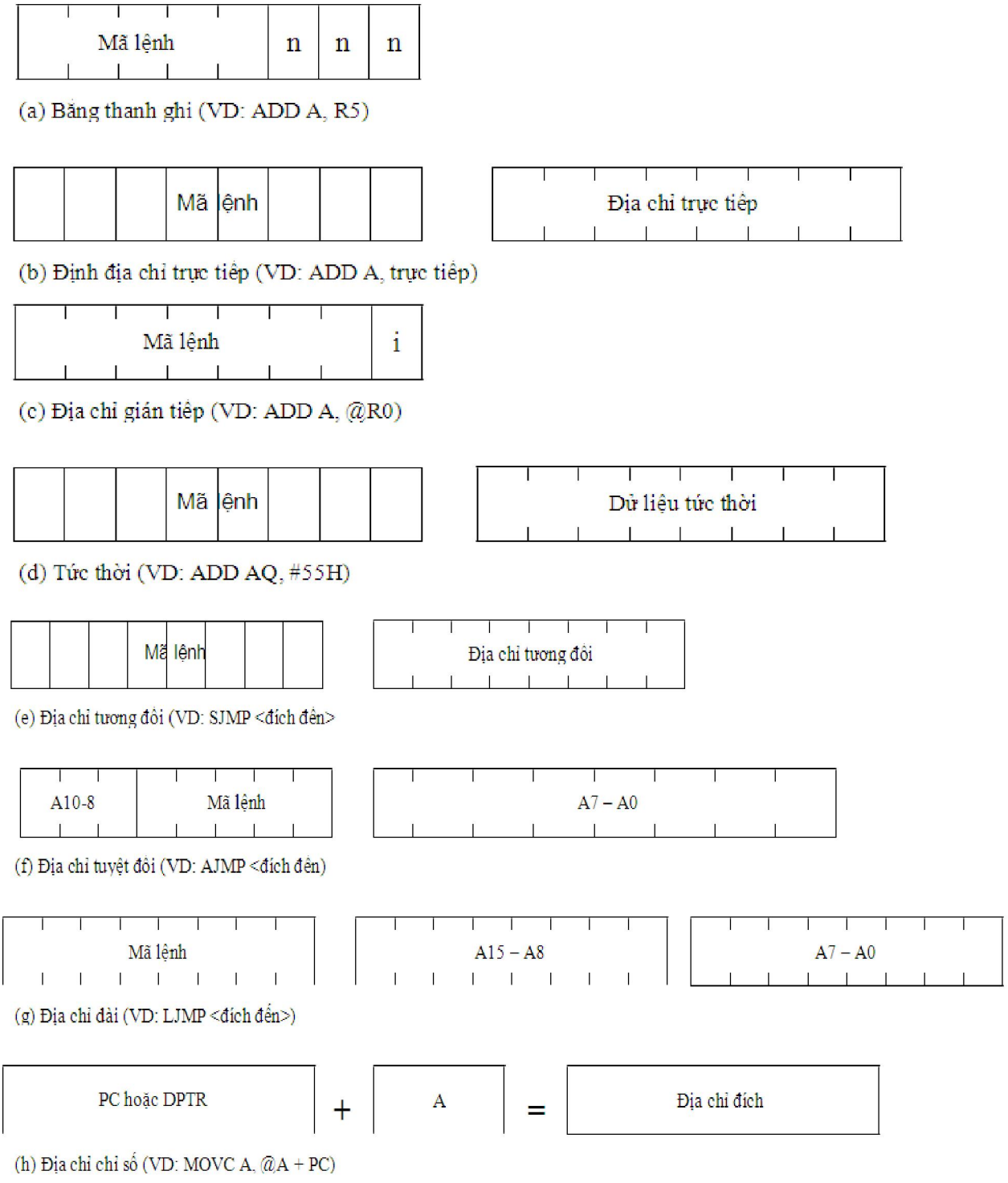
Cấu tạo 8051 có 4 dãy thanh ghi nhưng tại mỗi thời điểm chỉ có một dãy duy nhất hoạt động, bốn dãy thanh ghi này là vùng nhớ 32 byte đầu tiên của RAM trong chiếm địa chỉ từ 00H đến 1FH, bít 4 và bít 3 trong thanh ghi PSW sẽ xác định dãy thanh ghi đang hoạt động, khi reset hệ thống thì dãy thanh ghi hoạt động mặc định là dãy 0, nhưng cũng có thể chọn dãy thanh ghi khác bằng cách thay đổi giá trị bít 4 và bít 3 của thanh ghi PSW.

VD: MOV PSW, #00011000B

Lúc này dãy thanh ghi hoạt động là dãy 3, bít 4 của PSW là bít RS1 và bít 3 của PSW là RS0. Có một số lệnh đặc biệt tác động đến một thanh ghi mặc định như: Bộ tích lũy, con trỏ dữ liệu …Do đó, không cần đến các bít địa chỉ, trong trường hợp này bộ tích lũy được ký hiệu là A, con trỏ dữ liệu là DPTR, bộ đếm chương trình là PC, thanh ghi cờ là C, và cặp bộ tích lũy-thanh ghi B là AB.

VD: INC DPTR

Đây là lệnh có độ dài 1 byte, kết quả là con trỏ dữ liệu DPTR sẽ tăng lên 1 sau khi lệnh này được thực hiện



Hình 32-03-1 Các cách định địa chỉ của 8051

1.2. Định địa chỉ trực tiếp

Phương pháp này cho phép truy xuất đến mọi ô nhớ hoặc thanh ghi, mã đối tượng cần thêm 1 byte để xác định địa chỉ của dữ liệu (hình 3.1b). Tùy thuộc vào giá trị của bít

cao trong byte địa chỉ mà một trong hai vùng nhớ của 8051 sẽ được chọn. Khi bít 7 bằng 0 thì địa chỉ trực tiếp có giá trị từ 0 đến 127 (00H..7FH) và lúc này 128 ô nhớ ở vùng thấp của RAM trong sẽ được chọn. Tất cả các cổng I/O, thanh ghi đặc biệt, thanh ghi điều khiển hoặc thanh ghi trạng thái có địa chỉ từ 128 đến 255 (80H..FFH). Do đó, khi bít 7 trong byte địa chỉ bằng 1 thì thanh ghi đặc biệt tương ứng sẽ được chọn. VD: port 0 và port 1 có địa chỉ trực tiếp là 80H và 90H. Thực ra cũng không cần thiết phải nhớ các địa chỉ này vì hợp ngữ 8051 cho phép dùng ký hiệu P0 để chỉ port 0, TMOD để chỉ thanh ghi chọn chế độ của timer ...

VD: MOV P1, A

Sẽ chuyển nội dung của bộ tích lũy vào port 1, địa chỉ trực tiếp của port 1 sẽ được phần mề hợp ngữ tự động điền vào byte thứ hai trong mã lệnh.

1.3. Định địa chỉ gián tiếp

Làm thế nào để xác định một ô nhớ khi mà địa chỉ của nó là kết quả của một phép tính, hoặc địa chỉ này bị thay đổi khi đang chạy chương trình, điều này thường xảy ra khi cần thâm nhập bảng chữ số hoặc bảng ký tự có địa chỉ liên tiếp nhau. Yêu cầu này được giải quyết bằng cách dùng hai thanh ghi R0 và R1 làm con trỏ, nội dung của chúng chính là địa chỉ của dử liệu, hai thanh ghi này được xác định bởi bít thấp nhất trong mả lệnh (hình 3.1c). Cú pháp hợp ngữ 8051 dùng ký hiệu @ đặt trước ký hiệu thanh ghi R0 hoặc R1 trong cách định địa chỉ gián tiếp .

VD: Nếu nội dung thanh ghi R1 là 40H và nội dung của ô nhớ trong tại địa chỉ 40H là 55H thì lệnh sau

MOV A, @ R1

Sẽ chuyển giá trị 55H vào bộ tích lũy. Cách định địa chỉ gián tiếp rất tiện lợi khi tác động lên vùng nhớ có địa chỉ liên tiếp

Ví dụ: Các lệnh sau đây sẽ xóa một vùng RAM trong có địa chỉ từ 60H đến 7FH.

MOV R0, # 60H Loop: MOV @ R0, # 0

INC R0

CJNE R0, # 80H, Loop

Lệnh đầu tiên đặt địa chỉ đầu tiên của vùng nhớ vào R0, lệnh thứ hai áp dụng cách định địa chỉ gián tiếp để chuyển giá trị 00H vào ô nhớ được trỏ đến bởi R0, lệnh thứ

ba tăng con trỏ R0 đến địa chỉ tiếp theo và lệnh cuối cùng kiểm tra xem R0 đã trỏ đến ô nhớ cuối cùng chưa, ở đây địa chỉ so sánh là 80H thay vì 7FH để bảo đãm ô nhớ 7FH bị xóa trước khi kết thúc chương trình.

1.4. Định địa chỉ tức thời

Trong trường hợp toán hạng nguồn là một hằng số thay vì là một biến thì hằng số này có thể được ghép vào mã lệnh như là một byte dữ liệu "tức thời", byte thêm vào chính là giá trị của dử liệu (hình 3.1d).

Trong hợp ngữ của 8051, byte dữ liệu tức thời được viết theo sau ký hiệu # toán hạng này có thể là một số, một biến ký hiệu hoặc một biểu thức số học, ký hiệu và toán tử. Phần mềm hợp ngữ sẽ tính ra giá trị và cho dữ liệu tức thời này vào mã lệnh.

VD: MOV A, # 12

Lệnh này nạp giá trị 12 (0CH) vào bộ tích lũy (hằng số 12 là số 12 nên không có chữ H theo sau).

Tất cả các lệnh áp dụng cách định địa chỉ tức thời đều dùng dữ liệu tức thời có độ dài là 8 bít ngoại trừ trường hợp khởi tạo con trỏ dữ liệu.

VD: MOV DPTR, # 8000H

Lệnh này dài 3 byte và sẻ nạp hằng số 16 bít có giá trị 8000H vào con trỏ dữ liệu

1.5. Định địa chỉ tương đối

Phương pháp này chỉ dùng cho các lệnh nhảy xác định, địa chỉ tương đối còn gọi là địa chỉ offset là một giá trị 8 bít có dấu được cộng thêm vào bộ đếm chương trình để tạo thành địa chỉ của lệnh sẽ được thực hiện tiếp theo, vì địa chỉ tương đối là 1 số 8 bít có dấu nên phạm vi nhảy chỉ trong giới hạn – 128 byte đến + 127 byte, địa chỉ tương đối là byte thêm vào trong mã lệnh. Trước khi cộng, bộ đếm chương trình được tăng lên đến địa chỉ sau lệnh nhảy, do đó địa chỉ đích có quan hệ tương đối với địa chỉ này chú không liên hệ gì đến địa chỉ của lệnh nhảy.

Thông thường chi tiết này không quan trọng đối với người lập trình bởi vì địa chỉ đích thường được chỉ ra bởi một nhản và phần mềm hợp ngữ sẽ tự xác định offset tương ứng. VD: Nếu nhản THERE đặt tại lệnh có địa chỉ 1040H, và lệnh nhảy SJMP THERE được đặt tại địa chỉ 1000H và 1001H, hợp ngữ sẽ tính ra offset là 3EH ở byte thứ hai của mã lệnh (1002H + 3EH = 1040H).

Ưu điêm của cách định địa chỉ tương đối là cung cấp các mã lệnh không phụ thuộc vị trí (do không dùng địa chỉ tuyệt đốI) nhưng khuyết điểm là phạm vi nhảy bị hạn chế.

1.6. Định địa chỉ tuyệt đối

Cách định địa chỉ này chỉ áp dụng với lệnh ACALL và AJMP. Đây là các lệnh 2 byte cho phép chương trình nhảy trong phạm vi một trang 2K trong bộ nhớ chương trình bằng cách cung cấp 11 bít thấp của địa chỉ đích trong mã lệnh (A10-A8) và byte thứ 2 của mã lệnh (A7-A0) (hình 3.1f).

Năm bít cao của địa chỉ đích chính là 5 bít cao hiện đang chứa trong bộ đếm chương trình, do đó lệnh theo sau lệnh nhảy và đích đến của lệnh nhảy phải có vị trí trong cùng một trang 2K, bởi vì A15-A11 không thay đổi (hình 3.3). Ưu điểm của phương pháp này là mã lệnh ngắn (2 byte) nhưng khuyết điểm là phạm vi nhảy có giới hạn và mã lệnh phụ thuộc vào vị trí.

1.7. Định địa chỉ dài

Cách định địa chỉ này chỉ dùng cho lệnh LCALL và LJMP, đây là các lệnh 3 byte bao gồm địa chỉ đích đầy đủ 16 bít, đó là byte 2 và byte 3 trong mã lệnh (hình 3.1g). Ưu điểm là có thể nhảy đến một vị trí bất kỳ trong phạm vi 64 K bộ nhớ chương trình, khuyết điểm là lệnh dài (3 byte) và phụ thuộc vị trí, điều này làm cho chương trình không thể được thực hiện tại địa chỉ khác. VD: Một chương trình có địa chỉ bắt đầu tại 2000H và trong chương trình có lệnh nhảy LJMP 2040H, thì chương trình không thể nào di chuyển đến 4000H vì lệnh nhảy này luôn nhảy đến 2040H, đây sẽ là một vị trí sai nếu di chuyển chương trình đến 4000H.

1.8. Định địa chỉ chỉ số

Cách định địa chỉ này sử dụng một thanh ghi cơ sở (có thể là bộ đếm chương trình hoặc con trỏ dữ liệu) và một offset (bộ tích lũy) để tạo thành một địa chỉ cho lệnh JMP hoặc MOVC (hình 3.1h). Các bảng nhảy hoặc các bảng tìm kiếm được tạo ra một cách dễ dàng nhờ vào cách định địa chỉ này thông qua các lệnh MOVC A, @ A + <base-reg> và JMP @ A + DPTR.

2. Các nhóm lệnh

Mục tiêu:

- Biết được tập lệnh của vi điều khiển được chia làm mấy loại

- Hiểu và vận dụng được các lệnh của vi điều khiển để viết chương trình

8951 chia ra 5 nhóm lệnh chính:

Các lệnh số học.

Lệnh logic.

Dịch chuyển dữ liệu.

Xử lý bít

Rẽ nhánh chương trình.

2.1. Nhóm lệnh số học

ADD A, <src, byte>

ADD A, Rn : (A) ← (A) + (Rn)

ADD A, direct : (A) ← (A) + (direct)

ADD A, @ Ri : (A) ← (A) + ((Ri))

ADD A, # data : (A) ← (A) + # data

ADDC A, Rn : (A) ← (A) + (C) + (Rn)

ADDC A, direct : (A) ← (A) + (C) + (direct)

ADDC A, @ Ri : (A) ← (A) + (C) + ((Ri))

ADDC A, # data : (A) ← (A) + (C) + # data

SUBB A, <src, byte>

SUBB A, Rn : (A) ← (A) - (C) - (Rn)

SUBB A, direct : (A) ← (A) - (C) - (direct)

SUBB A, @ Ri : (A) ← (A) - (C) - ((Ri))

SUBB A, # data : (A) ← (A) - (C) - # data

INC <byte>

INC A : (A) ← (A) + 1

INC direct : (direct) ← (direct) + 1

INC Ri : ((Ri)) ← ((Ri)) + 1

INC Rn : (Rn) ← (Rn) + 1

INC DPTR : (DPTR) ← (DPTR) + 1

DEC <byte>

DEC A : (A) ← (A) - 1

DEC direct : (direct) ← (direct) - 1

DEC @Ri : ((Ri)) ← ((Ri)) - 1

DEC Rn : (Rn) ← (Rn) - 1

MULL AB : (A)= LOW [(A) x (B)];có ảnh hưởng cờ OV

: (B)= HIGH [(A) x (B)];cờ Cary được xóa.

DIV AB : (A)= Integer Result of [(A)/(B)]; cờ OV

: (B)= Remainder of [(A)/(B)]; cờ Carry xóa

DA A :Điều chỉnh thanh ghi A thành số BCD đúng

; trong phép cộng BCD (thường DA A đi kèm

; với ADD, ADDC)

;Nếu [(A3-A0)>9] và [(AC)=1] (A3A0) = (A3A0) + 6.

;Nếu [(A7-A4)>9] và [(C)=1] ;(A7A4) = (A7A4) + 6.

2.2. Nhóm lệnh logic

Tất cả các lệnh logic sử dụng thanh ghi A như là một trong những toán hạng thực thi một chu kỳ máy, ngoài A ra mất 2 chu kỳ máy. Những hoạt động logic có thể được thực hiện trên bất kỳ byte nào trong vị trí nhớ dữ liệu nội mà không qua thanh ghi A.

Các hoạt động logic được tóm tắt như sau:

ANL <dest>, <src>

ANL A, Rn *; (A)=(A) AND (Rn).*

ANL A, direct *; (A)=(A) AND (direct).*

ANL A,@ Ri *; (A)=(A) AND ((Ri)).*

ANL A, # data *; (A)=(A) AND (# data).*

ANL direct, A *; (direct)=(direct) AND (A).*

ANL direct, # data *; (direct)=(direct) AND # data.*

ORL <dest>,<src>

ORL A, Rn *; (A)=(A) OR (Rn).*

ORL A, direct *; (A)=(A) OR (direct).*

ORL A,@ Ri *; (A)=(A) OR ((Ri)).*

ORL A, # data *; (A)=(A) OR # data.*

ORL direct, A *; (direct)=(direct) OR (A).*

ORL direct, # data *; (direct)=(direct) OR # data.*

XRL <dest>,<src>

XRL A, Rn *: (A)=(A)Xor(Rn).*

XRL A, direct *: (A)= (A)Xor(direct).*

XRL A,@ Ri *; (A)= (A)Xor((Ri)).*

XRL A, # data *; (A)= (A) # data.*

XRL direct, A *; (direct)=(direct)Xor(A).*

XRL direct, # data *; (direct)=(direct)Xor # data.*

CLR A *; (A)= 0*

CLR C *; (C)= 0*

CLR Bit *; (Bit)= 0*

RL A *; Quay vòng thanh ghi A qua trái 1 bit*

;(An + 1)=(An); n = 06

;(A0)=(A7)

RLC A ; Quay vòng thanh ghi A qua trái 1

;bit có cờ Carry

;bit (An + 1)=(An); n = 06

;(A0)=(C)

;(C)=(A7)

RR A ; Quay vòng thanh ghi A qua phải 1 bit

(An + 1)=(An); n = 06

(A0)=(A7)

RRC A ; Quay vòng thanh ghi A qua phải 1

;bit có cờ Carry

(An + 1)=(An); n = 06

(A7)=(C)

(C)=(A0)

SWAPA ; Đổi chổ 4 bit thấp và 4 bit cao của A cho

; nhau (A3A0)(A7A4).

2.3. Nhóm lệnh di chuyển dữ liệu

Các lệnh dịch chuyển dữ liệu trong những vùng nhớ nội thực thi 1 hoặc 2 chu kỳ máy. Mẫu lệnh MOV <destination>, <source> cho phép di chuyển dữ liệu bất kỳ 2 vùng nhớ nào của RAM nội hoặc các vùng nhớ của các thanh ghi chức năng đặc biệt mà không thông qua thanh ghi A.

Vùng Ngăn xếp của 8951 chỉ chứa 128 byte RAM nội, nếu con trỏ Ngăn xếp SP được tăng quá địa chỉ 7FH thì các byte được PUSH vào sẽ mất đi và các byte POP ra thì không biết rõ.

Các lệnh dịch chuyển bộ nhớ nội và bộ nhớ ngoại dùng sự định vị gián tiếp. Địa chỉ gián tiếp có thể dùng địa chỉ 1 byte (@ Ri) hoặc địa chỉ 2 byte (@ DPTR). Tất cả các lệnh dịch chuyển hoạt động trên toàn bộ nhớ ngoài thực thi trong 2 chu kỳ máy và dùng thanh ghi A làm toán hạng DESTINATION.

Việc đọc và ghi RAM ngoài (RD và WR) chỉ tích cực trong suốt quá trình thực thi của lệnh MOVX, còn bình thường RD và WR không tích cực (mức 1).

Tất cả các lệnh dịch chuyển đều không ảnh hưởng đến cờ. Hoạt động của từng lệnh được tóm tắt như sau:

MOV A,Rn ; *(A)= (Rn)*

MOV A, direct ; *(A)= (direct)*

MOV A, @ Ri ; *(A)= ((Ri))*

MOV A, # data ; *(A):= data*

MOV Rn, A ; *(Rn)= (A)*

MOV Rn, direct ; *(Rn)= (direct)*

MOV Rn, # data ; *(Rn)= data*

MOV direct, A ; *(direct)= (A)*

MOV direct, Rn ; *(direct)= (Rn)*

MOV direct, direct ; *(direct)= (direct)*

MOV direct, @ Ri ; *(direct)= ((Ri))*

MOV direct, # data ; *(direct)= data*

MOV @ Ri, A ; *((Ri)) = (A)*

MOV @ Ri, direct ; *((Ri)) = (direct)*

MOV @ Ri, # data ; *((Ri))= data*

MOV DPTR, # data16 ; *(DPTR) # data16*

MOV A, @ A + DPTR ; *(A) (A) + (DPTR)*

MOV @ A + PC *; (PC)= (PC) + 1*

;(A)= (A) + (PC)

MOVX A, @ Ri *;(A)= ((Ri))*

MOVX A, @ DPTR *; (A)= ((DPTR))*

MOVX @ Ri, A *; ((Ri)) = (A)*

MOVX @ DPTR, A *; ((DPTR)) = (A)*

PUSH direct *; Cất dữ liệu vào Ngăn xếp*

; (SP)= (SP) + 1

; (SP)= (Drirect)

POP direct *; Lấy từ Ngăn xếp ra direct*

;(direct) ((SP))

;(SP)= (SP) - 1

XCH A, Rn *;Đổi chổ nội dung của A với Rn*

;(A)= (Rn)

XCH A, direct *;(A)= (direct)*

XCH A, @ Ri *;(A)= ((Ri))*

XCHD A, @ Ri *;Đổi chổ 4 bit thấp của (A) với ((Ri))*

;(A3A0) ((Ri3Ri0))

2.4. Nhóm lệnh xử lý bit

8951 chứa một bộ xử lí luận lý đầy đủ cho các hoạt động bit đơn, đây là một điểm mạnh của họ vi điều khiển MSC-51 mà các họ vi điều khiển khác không có.

RAM nội chứa 128 bit đơn vị và các vùng nhớ các thanh ghi chức năng đặc biệt cấp lên đến 128 đơn vị khác. Tất cả các đường Port là bit định vị, mỗi đường có thể được xử lí như Port đơn vị riêng biệt. Cách truy xuất các bit này không chỉ các lệnh rẽ nhánh không, mà là một danh mục đầy đủ các lệnh MOVE, SET, CLEAR, COMPLEMENT, OR, AND.

Toàn bộ sự truy xuất của bit dùng sự định vị trực tiếp với những địa chỉ từ 00H - 7FH trong 128 vùng nhớ thấp và 80H - FFH ở các vùng thanh ghi chức năng đặc biệt.

Bit Carry C trong thanh ghi PSW\ của từ trạng thái chương trình và được dùng như một sự tích lũy đơn của bộ xử lí luận lý. Bit Carry cũng là bit định vị và có địa chỉ trực tiếp vì nó nằm trong PSW. Hai lệnh CLR C và CLR CY đều có cùng tác dụng là xóa bit cờ Carry nhưng lệnh này mất 1 byte còn lệnh sau mất 2 byte.

Hoạt động của các lệnh luận lý được tóm tắt như sau.

CLR C ; Xóa cờ Carry xuống 0. Có ảnh hưởng cờ Carry.

CLR BIT ; Xóa bit xuống 0. Không ảnh hưởng cờ Carry

SET C ; Set cờ Carry lên 1. Có ảnh hưởng cờ Carry.

SET BIT ; Set bit lên 1. Không ảnh hưởng cờ Carry.

CPL C ; Đảo bit cờ Carry. Có ảnh hưởng cờ Carry.

CPL BIT ; Đảo bit. Không ảnh hưởng cờ Carry.

ANL C, BIT ; (C) = (C) AND (BIT) : Có ảnh hưởng cờ Carry.

ANL C, /BIT ; (C) = (C) AND NOT (BIT)

;Không ảnh hưởng cờ Carry.

ORL C, BIT ; (C) = (C) OR (BIT) : Tác động cờ Carry.

ORL C, /BIT ; (C) = (C) OR NOT (BIT) : Tác động cờ Carry.

MOV C, BIT ; (C) = (BIT) : Cờ Carry bị tác động.

MOV BIT, C ; (BIT) = (C) : Không ảnh hưởng cờ Carry.

2.5. Nhóm lệnh rẽ nhánh chương trình

Có nhiều lệnh để điều khiển lên chương trình bao gồm việc gọi hoặc trả lại từ chương trình con hoặc chia nhánh có điều kiện hay không có điều kiện.

Tất cả các lệnh rẽ nhánh đều không ảnh hưởng đến cờ. Ta có thể định nhãn cần nhảy tới mà không cần rõ địa chỉ, trình biên dịch sẽ đặt địa chỉ nơi cần nhảy tới vào đúng khẩu lệnh đã đưa ra.

Sau đây là sự tóm tắt từng hoạt động của lệnh nhảy.

JC rel : Nhảy đến "rel" nếu cờ Carry C = 1.

JNC rel : Nhảy đến "rel" nếu cờ Carry C = 0.

JB bit, rel : Nhảy đến "rel" nếu (bit) = 1.

JNB bit, rel : Nhảy đến "rel" nếu (bit) = 0.

JBC bit, rel : Nhảy đến "rel" nếu bit = 1 và xóa bit.

ACALL addr11 : Lệnh gọi tuyệt đối trong page 2K.

(PC) = (PC) + 2

(SP) = (SP) + 1

((SP))= (PC7PC0)

(SP) = (SP) + 1

((SP)) = (PC15PC8)

(PC10PC0) page Address.

LCALL addr16 ;Lệnh gọi dài chương trình con trong 64K.

(PC)= (PC) + 3

(SP) = (SP) + 1

((SP)) = (PC7PC0)

(SP) = (SP) + 1

((SP)) = (PC15PC8)

(PC) Addr15Addr0.

RET ;Kết thúc chương trình con.

(PC15PC8) (SP)

(SP)= (SP) - 1

(PC7PC0) = ((SP))

(SP)= (SP) -1.

RETI ;Kết thúc thủ tục phục vụ ngắt quay về chương trình

;chính hoạt động tương tự như RET.

AJMP Addr11 ;Nhảy tuyệt đối không điều kiện trong 2K.

;(PC)= (PC) + 2

;(PC10PC0) page Address.

LJMP Addr16 : Nhảy dài không điều kiện trong 64K

;Hoạt động tương tự lệnh LCALL.

SJMP rel ;Nhảy ngắn không điều kiện trong

;(-128127) byte

(PC)= (PC) + 2

(PC)= (PC) + byte 2

JMP @ A + DPTR ;Nhảy không điều kiện đến địa chỉ

;(A) + (DPTR)

;(PC)= (A) + (DPTR)

JZ rel : Nhảy đến A = 0. Thực hành lệnh kế nếu A <> 0.

;(PC)= (PC) + 2

;(A) = 0 (PC)= (PC) + byte 2

JNZ rel ; Nhảy đến A <> 0. Thực hành lệnh kế nếu A = 0.

;(PC)= (PC) + 2

;(A) < > 0 (PC) (PC) + byte 2

CJNE A, direct, rel ;So sánh và nhảy đến (A) <> (direct)

;(PC)= (PC) + 3

;(A) < > (direct) (PC)= (PC) + Relative ;Address.

;(A) < (direct) C = 1

;(A) > (direct) C = 0

;(A) = (direct). Thực hành lệnh kế tiếp

CJNE A, # data, rel : Tương tự lệnh CJNE A, ;direct, rel.

CJNE Rn,#data, rel ;Tương tự lệnh CJNE A, direct, rel.

CJNE @Ri,#data,rel ;Tương tự lệnh CJNE A, direct, rel.

DJNZ Rn,rel ;Giảm Rn và nhảy nếu Rn <> 0.

;(PC)= (PC) + 2

;(Rn)= (Rn) -1

;(Rn) < > 0, (PC)= (PC) + byte 2.

DJNZ direct, rel ; Tương tự lệnh DJNZ Rn, rel.

NOP ; Không hoạt động gì cả, chỉ tốn 1 byte và 1 chu kỳ máy

THỰC HÀNH VỚI TẬP LỆNH 8051

I. MỤC TIÊU

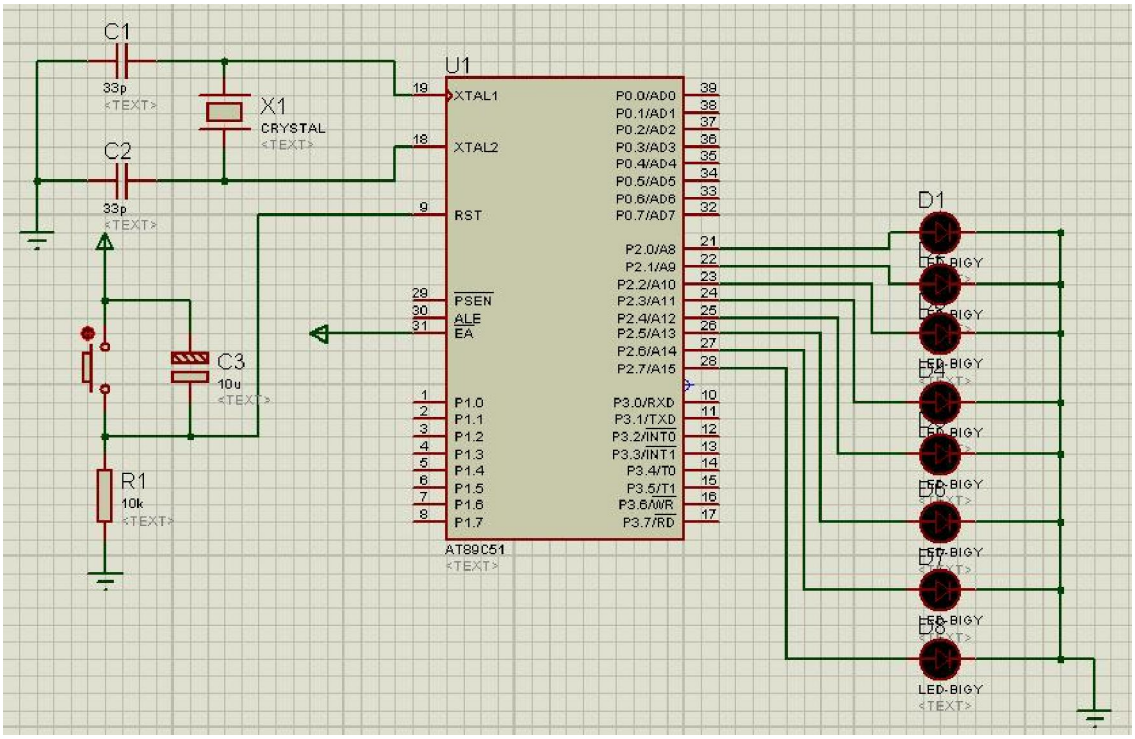
- Giúp sinh viên làm quen với thiết bị và tìm hiểu một số nhóm lệnh quan trọng trong tập lệnh của 8051
- Hiểu rõ hơn về tập lệnh của vi điều khiển MCS-51.
- Biết cách viết các chương trình điều khiển LED đơn ở các chế độ khác nhau.
- Hiểu được sơ đồ và nguyên lý hoạt động của khối LED đơn trên mô hình thí nghiệm.

II. NỘI DUNG THÍ NGHIỆM

1. Nhóm lệnh di chuyển dữ liệu

a. Nối mạch thí nghiệm:

Các led tương ứng từ led1 đến led8 sẽ nối với các bit P2.0 đến P2.7 theo hình vẽ sau:



b. Viết chương các trình ứng dụng:

Chương trình 1.1: Chớp tắt 8 led vô hạn lần

```
 MAIN:
        MOV   P2,#0FFH     ; P1 <- 11111111B, các led đều sáng
        CALL  DELAY        ; gọi chương trình trì hoãn DELAY
        MOV   P2,#00H      ; P1  <- 00000000B, các led đều tắt
        CALL  DELAY
        LJMP   MAIN        ; nhảy đến MAIN để lập lại quá trình vô hạn  DELAY:
        MOV  R1,#250
LAP:    MOV R2,#200
LAP1:   DJNZ R2,LAP1
        DJNZ R1,LAP
            RET
      END
      NHẮC LẠI:
```

DJNZ Ri, <NHÃN>: Ri là một trong các thanh ghi từ R0 -> R7, lệnh giảm nội dung thanh ghi Ri một đơn vị, nếu Ri > 0 thì nhảy đến NHÃN chỉ định, ngược lại sẽ chuyển sang thực hiện kế tiếp.

Chương trình 1.2: giống nội dung chương trình 01 nhưng lặp lại quá trình 10 lần. Thời gian sáng tắt là 1s

- Hướng dẫn:

```
        MOV   Ri, # <SỐ LẦN LẶP (1->255)>
    NHÃN:
        - LỆNH 1
        - LỆNH 2

        - LỆNH n
        DJNZ   Ri, <NHÃN>
```

- Chương trình:

```
    MOV   R7,#10
MAIN:
    MOV   P2,#0FFH     ; P1 <- 11111111B, các led đều sáng
    CALL  DELAY        ; gọi chương trình trì hoãn DELAY
    MOV   P2,#0FFH       ; P1  <-00000000B, các led đều tắt
    CALL  DELAY
     DJNZ   R7,MAIN      ;
     SJMP   $              ; "dừng chương trình"
DELAY:
    MOV  R1,#250
LAP:
    MOV R2,#200
LAP1: MOV R3,#20
LAP2: DJNZ R3,LAP2
    DJNZ R2,LAP1
```

```
        DJNZ  R1,LAP
              RET
        END
```

BÀI TẬP SV

Chương trình 1.3: Hiển thị 1 led sáng, dịch dần từ D1 đến D8 vô hạn lần.

Chương trình 1.4: Hiển thị 1 led sáng, dịch dần từ D8 đến D1 với số lần lặp là 6.

2. Nhóm lệnh quay vòng

a. Nối mạch thí nghiệm:

Các led tương ứng từ led1 đến led8 sẽ nối với các bit P3.0 đến P3.7, các led đều tác động ở mức cao.

b. Viết các chương trình ứng dụng:

Chương trình 1.5: viết chương trình dịch một led sáng từ D1 đến D8

```
 MAIN:
          MOV   A,#01H
    BEGIN:
          MOV   P2, A
           RL      A
           CALL  DELAY
          LJMP   BEGIN
    DELAY:
           MOV  R1,#250
    LAP:     MOV R2,#200
    LAP1:   DJNZ R2,LAP1
            DJNZ R1,LAP
              RET
        END
```

Chương trình 1.6: viết chương trình dịch một led sáng từ D8 đến D1

```
    MAIN:
          MOV   A,#80H
    BEGIN:
```

```
        MOV   P1, A
 RR      A
        CALL  DELAY
        LJMP   BEGIN

DELAY:
        PUSH  06              ; cất nội dung R6 vào ngăn xếp
        PUSH  07              ; cất nội dung R7 vào ngăn xếp
         MOV   R6,#255
LAP:
        MOV   R7,#255
        DJNZ   R7,$           ;
        DJNZ   R6, LAP
        POP    07             ; lấy lại giá trị cũ của R7 trong ngăn xếp
        POP    06             ; lấy lại giá trị cũ của R0 trong ngăn xếp
        RET                   ; kết thúc chương trình con.
         END
```

Chương trình 1.7: viết chương trình sáng dần các led từ D1 đến D8

```
MAIN:
        MOV   A,#01H
BEGIN:
        SETB   C
        MOV   P2, A
 RLC    A
        CALL  DELAY
        JNC     BEGIN
        LJMP   MAIN

DELAY:
```

```
        PUSH  06            ; cất nội dung R6 vào ngăn xếp
        PUSH  07            ; cất nội dung R7 vào ngăn xếp
        MOV   R6,#255
LAP:
        MOV   R7,#255
        DJNZ   R7,$         ;
        DJNZ   R6, LAP
        POP    07           ; lấy lại giá trị cũ của R7 trong ngăn xếp
        POP    06           ; lấy lại giá trị cũ của R0 trong ngăn xếp
        RET                 ; kết thúc chương trình con.
END
```

Chương trình 1.8: viết chương trình sáng dần các led từ D8 đến D1

```
MAIN:
        MOV   A,#80H
BEGIN:
        SETB   C
        MOV   P2, A
 RRC    A
        CALL  DELAY
        JNC     BEGIN
        LJMP   MAIN

DELAY:
        PUSH  06            ; cất nội dung R6 vào ngăn xếp
        PUSH  07            ; cất nội dung R7 vào ngăn xếp
        MOV   R6,#255
LAP:
        MOV   R7,#255
        DJNZ   R7,$         ; Ù X: DJNZ R7,X
```

```
        DJNZ   R6, LAP
        POP    07          ; lấy lại giá trị cũ của R7 trong ngăn xếp
        POP    06          ; lấy lại giá trị cũ của R0 trong ngăn xếp
        RET                ; kết thúc chương trình con.
    END
```

BÀI TẬP

Chương trình 1.9: viết chương trình tắt dần các led từ D1 đếnD8

Chương trình 1.10: viết chương trình tắt dần các led từ D8 đến D1

Chương trình 1.11: viết chương trình tắt dần các led từ D8 đến D1, thực hiện lặp lại 20 lần.

3. Nhóm lệnh toán học

a. Nối mạch thí nghiệm

Các led tương ứng từ led1 đến led8 sẽ nối với các bit P1.0 đến P1.7,

các led đều tác động ở mức cao

b. Viết các chương trình ứng dụng

Chương trình 1.12: Viết chương trình cộng 2 số sau: 12 + 34, hiển thị kết quả trên các led.

```
 MAIN:
        MOV    A,#12
        ADD    A,#34
        MOV    P2,A
        SJMP   $
END
```

Chương trình 1.13: Viết chương trình cộng 2 số sau: 12 + 34, hiển thị kết quả trên các led.

```
MAIN:
        MOV    A,#12
SETB   C
        ADDC   A,#34
        MOV    P2,A
```

```
    SJMP    $
END
```

So sánh kết quả của chương trình 1 và chương trình 2.

Chương trình 1.14: Viết chương trình chia 2 số 17 cho 3, hiển thị kết quả (phần nguyên) trên các led.

```
MAIN:
    MOV    A,#17
    MOV    B,#3
    DIV    AB
    MOV    P2,A
    SJMP   $
END
```

Chương trình 1.15: Viết chương trình chia 2 số 17 cho 3, hiển thị kết quả (phần dư) trên các led.

```
MAIN:
    MOV    A,#17
    MOV    B,#3
    DIV    AB
    MOV    P2,B
    SJMP   $
END
```

Chương trình 1.16: Viết chương trình nhân 2 số 23 cho 14, hiển thị kết quả (byte thấp) trên các led.

```
MAIN:
    MOV    A,#7
    MOV    B,#14
    MUL    AB
    MOV    P2,A
    SJMP   $
END
```

Chương trình 1.17: Viết chương trình nhân 2 số 23 cho 14, hiển thị kết quả (byte cao) trên các led.

```
MAIN:
        MOV    A,#7
MOV    B,#14
        MUL    AB
        MOV    P2,B
        SJMP    $
END
```

BÀI 4

BỘ ĐỊNH THỜI (TIMER)

Mã bài: MĐ 25-04

Mục tiêu:

- Hiểu được đặc điểm các thanh ghi định thời
- Hiểu được sơ đồ và nguyên lý hoạt động các chế độ làm việc của bộ định thời 8051
- Ứng dụng được các bộ định thời trong các bài tập điều khiển

Nội dung chính:

1. Hoạt động của các bộ định thời

Mục tiêu: Nêu được cấu trúc và cơ chế làm việc của bộ định thời

Bộ định thời của Vi Điều Khiển là một chuỗi các Flip Flop, nó nhận tín hiệu vào là một nguồn xung clock, xung clock được đưa vào Flip Flop thứ nhất là xung clock của Flip Flop thứ hai mà nó cũng chia tần số clock này cho 2 và cứ tiếp tục.

Vì mỗi tầng kế tiếp chia cho 2, nên Timer n tầng phải chia tần số clock ngõ vào cho 2^n. Ngõ ra của tầng cuối cùng là clock của Flip Flop báo tràn Timer. Cờ này được kiểm tra bởi phần mềm hoặc sinh ra ngắt. Giá trị nhị phân trong các FF của bộ Timer

chính là số xung clock (tính từ giá trị bắt đầu cho đến khi tràn). Ví dụ bộ định thời 16-bit sẽ đếm từ 0000h đến FFFFh. Cờ tràn được setb bằng 1 khi xảy ra tràn từ FFFFH về 0000h.

Vi điều khiển 8951 có 2 bộ định thời. Mỗi bộ định thời là một chuỗi 16 Flip-Flop (16 bit) được chia làm 2 byte chứa ở TL0,TH0 (Timer 0) và TL1,TH1 (Timer 1) Trong các ứng dụng định thời, 1 Timer được lập trình để tràn ở một khoảng thời gian đều đặn và được set cờ tràn Timer

2. Các thanh ghi định thời

Mục tiêu:

- Biết được cấu tạo và chức năng các thanh ghi của bộ định thời

- Biết khai báo các thanh ghi để viết chương trình

2.1. Thanh ghi điều khiển chế độ timer TMOD

Thanh ghi mode gồm hai nhóm 4 bit: 4 bit thấp đặt chế độ hoạt động cho Timer 0 và 4 bit cao đặt chế độ hoạt động cho Timer 1.Thanh ghi TMOD được tóm tắt như sau:

Bit	Name	Timer	Description
7	GATE	1	Khi GATE = 1, Bit điều khiển cổng 1. Timer1 chỉ làm việc khi INT1=1, GATE=0 sự hoạt dộng của Timer1 không bị ảnh hưởng của INT1 (P3.3).
6	C/T	1	Bit cho đếm sự kiện hay định thời
			C/T = 1 : Đếm sự kiện
			C/T = 0 : Ghi giờ đều đặn
5	M1	1	Bit chọn mode của Timer 1
4	M0	1	Bit chọn mode của Timer 1

3	GATE	0	Khi GATE = 1, Bit điều khiển cổng 0. Timer0 chỉ làm việc khi INT1=0, GATE=0 sự hoạt động của Timer0 không bị ảnh hưởng của INT0 (P3.2).
2	C/T	0	Bit chọn Counter/Timer của Timer 0
1	M1	0	Bit chọn mode của Timer 0
0	M0	0	Bit chọn mode của Timer 0

Hai bit M0 và M1 của TMOD để chọn mode cho Timer 0 hoặc Timer 1.

M1	M0	MODE	DESCRIPTION
0	0	0	Chế độ Timer 13 bit (không còn được sử dụng cho các thiết kế mới)
0	1	1	Chế độ 16 bit
1	0	2	Chế độ tự động nạp 8 bit
1	1	3	Chế độ chia xẻ: Timer 0 : TL0 là Timer 8 bit nhận TF0 làm cờ tràn. TL0 là Timer 8 bit mượn TF1 làm cờ tràn Timer 1 : Không có cờ báo trán

2.2. Thanh ghi điều khiển timer TCON

Thanh ghi điều khiển bao gồm các bit trạng thái và các bit điều khiển bởi Timer 0 và Timer 1. Thanh ghi TCON có bit định vị. Hoạt động của từng bit được tóm tắt như sau:

Bit	Symbol	Bit Address	Description
TCON.7	TF1	8FH	Cờ tràn Timer 1 được set bởi phần cứng ở sự tràn, được xóa bởi phần mềm hoặc bởi phần cứng khi chương trình trỏ đến phục vụ ngắt ISR_T1
TCON.6	TR1	8EH	Bit điều khiển chạy Timer 1 (được set hoặc xóa bởi phần mềm để chạy hoặc ngưng chạy Timer)
TCON.5	TF0	8DH	Cờ tràn Timer 0 (hoạt động tương tự TF1)
TCON.4	TR0	8CH	Bit điều khiển chạy Timer 0 (giống TR1)
TCON.3	IE1	8BH	Cờ ngắt ngoài 1 (INT1).
TCON.2	IT1	8AH	Bit cho phép ngắt ngoài 1 tác động cạnh âm. IT1 = 1 cho phép ngắt tại thời điểm xảy ra cạnh âm.
TCON.1	IE0	89H	Cờ cạnh ngắt ngoài 0 (INT0)
TCON.0	IT0	88H	Bit cho phép ngắt ngoài 0 tác động cạnh âm hoặc mức. IT0 = 1 cho phép ngắt tại thời điểm xảy ra cạnh âm.

3. Các chế độ làm việc

Mục tiêu:

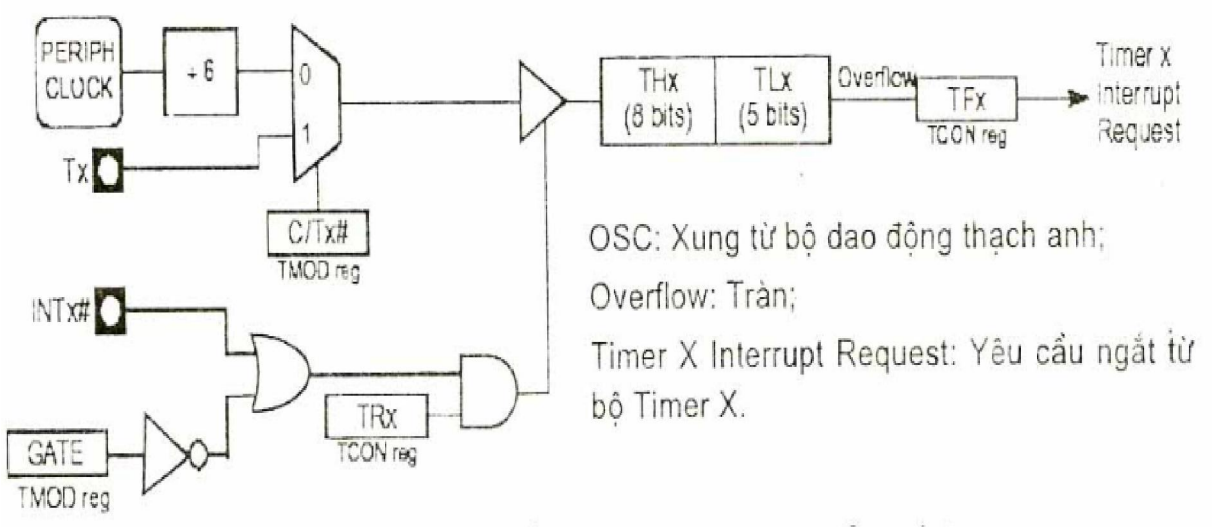
- Biết được sơ đồ cấu tạo và nguyên lý hoạt động các chế độ hoạt động của bộ định thời

- Biết được các bước lập trình ở các chế độ hoạt động

- Viết được các chương trình điều khiển sử dụng bộ định thời ở các chế độ khác nhau

3.1. Chế độ 13-bit (chế độ 0)

Chế độ 0 là chế độ định thời 13 bit, chế độ này tương thích với các bộ vi điều khiển trước đó, trong các ứng dụng hiện nay chế độ này không còn thích hợp.



Hình 32-04-1 Hoạt động của Timer 0 và Timer 1 ở chế độ 0

Trong chế độ này, bộ định thời dùng 13 bit (8bit của TH và 5 bit thấp của TL) để chứa giá trị đếm, 3 bit cao của TL không được sử dụng

Hình 4.1 mô tả hoạt động của các Timer ở chế độ 0: Nguồn xung clock được đưa tới Timer từ một trong cách phụ thuộc vào bit C/T trong thanh ghi TMOD

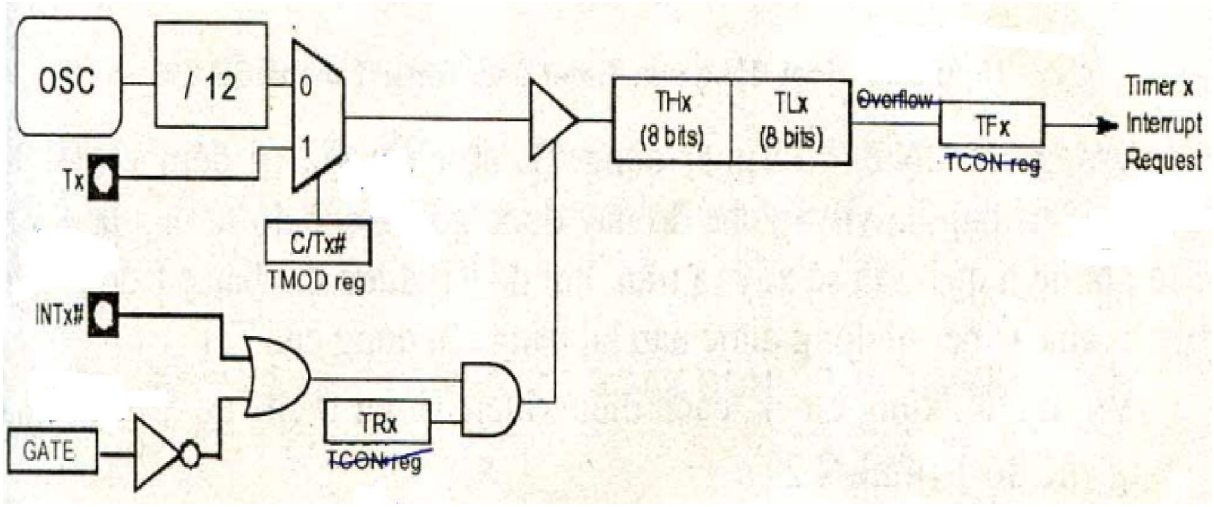
- Nếu C/T =1, xung clock sẽ được lấy từ bên ngoài qua chân Tx (T0 hoặc T1)
- Nếu C/T = 0, xung clock sẽ được lấy từ bộ chia tần trong chip

Nguồn xung clock nói trên sẽ được điều khiển để đưa tới các Timer bằng các bit: TR, GATE và mức logic trên các chân INTx.

- Nếu TRx = 0, các Timer sẽ bị cấm mà không cần quan tâm tới GATE và mức logic trên các chân INTx
- Nếu TRx = 1, các Timer sẽ hoạt động khi hoặc là bit GATE = 0 hoặc là bit GATE = 1 và trên chân INTx có mức logic 1.

3.2. Chế độ 16-bit (chế độ 1)

Trong chế độ 1, bộ Timer dùng cả 2 thanh ghi TH và TL để chứa giá trị đếm, vì vậy chế độ này còn được gọi là chế độ định thời 16 bit.



Hình 32-04-2 Hoạt động của Timer 0 và Timer 1 ở chế độ 1

Hình 4.2 mô tả hoạt động của các Timer ở chế độ 1: Nguồn xung clock được đưa tới Timer từ một trong cách phụ thuộc vào bit C/T trong thanh ghi TMOD.

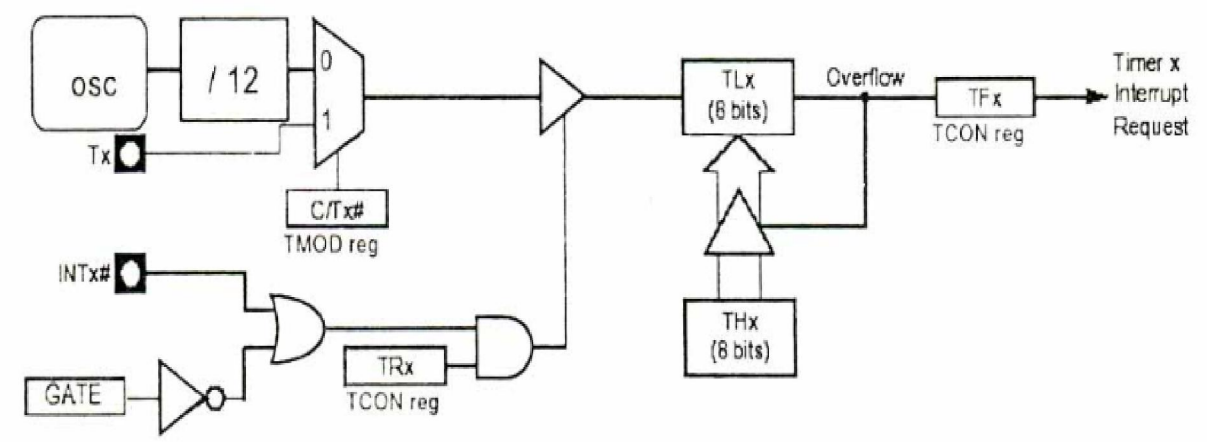
- Nếu C/T =1, xung clock sẽ được lấy từ bên ngoài qua chân Tx (T0 hoặc T1)
- Nếu C/T = 0, xung clock sẽ được lấy từ bộ chia tần trong chip

Nguồn xung clock nói trên sẽ được điều khiển để đưa tới các Timer bằng các bit: TR, GATE và mức logic trên các chân INTx.

- Nếu TRx = 0, các Timer sẽ bị cấm mà không cần quan tâm tới GATE và mức logic trên các chân INTx
- Nếu TRx = 1, các Timer sẽ hoạt động khi hoặc là bit GATE = 0 hoặc là bit GATE = 1 và trên chân INTx có mức logic 1.

Với chế độ 1, giá trị lớn nhất mà các Timer chứa được là 65535 khi đếm quá giá trị này sẽ xảy ra tràn, khi cờ tràn TF sẽ được thiết lập bằng 1. Sau khi xảy ra tràn, nếu muốn Timer tiếp tục đếm chương trình phải có câu lệnh nạp lại giá trị khởi tạo sau khi đã dừng Timer bằng cách xóa bit TR.

3.3. Chế độ tự nạp 8-bit (chế độ 2)



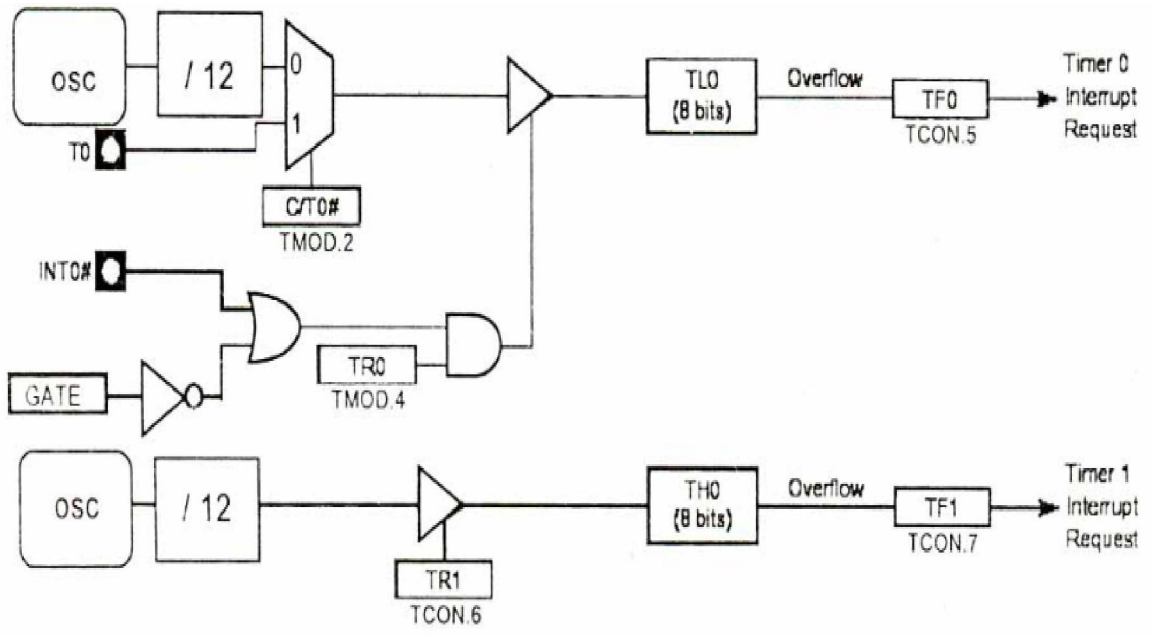
Hình 32-04-3 Hoạt động của Timer 0 và Timer 1 ở chế độ 2

Trong chế độ 2, bộ Timer dùng TL để chứa giá trị đếm và TH để chứa giá trị nạp lại vì vậy chế độ này được gọi là chế độ tự nạp lại 8 bit. Sau khi đếm quá 255 sẽ xảy ra tràn, khi đó TF được đặt bằng 1 đồng thời giá trị của Timer tự động được nạp lại bằng nội dung của TH

Với nguồn xung clock cách điều khiển Timer ở chế độ 2 hoàn toàn giống chế độ 1.

3.4. Chế độ chia xẻ (chế độ 3)

Trong chế độ 3, Timer 0 được tách thành 2 bộ Timer hoạt động độc lập, chế độ này sẽ cung cấp cho bộ vi điều khiển thêm một Timer nữa



Hình 32-04-4 Hoạt động của Timer 0 và Timer 1 ở chế độ 3

Bộ Timer thứ nhất với nguồn xung clock được lấy từ bộ chia tần trên chip hoặc từ bộ tạo xung bên ngoài qua chân T0 tùy thuộc vào giá trị của bít C/T0 (hoạt động giống chế độ 0, 1, 2)

Giá trị đếm của Timer được chứa trong TL0, khi xảy ra tràn, cờ TF0 được đặt bằng 1 và gây ra ngắt do Timer 0

Bộ Timer thứ hai với nguồn xung clock được lấy từ bộ chia tần trên chip. Việc điều khiển hoạt động của bộ thứ hai chỉ là việc đặt giá trị của bit TR0. Giá trị đếm của Timer được chứa trong TH0, khi xảy ra tràn, cờ TF1 được đặt bằng 1 và gây ra ngắt do Timer 1

Khi Timer 0 được tách thành 2 Timer 8 bit thì Timer 1 vẫn có thể hoạt động bình thường ở các chế độ 0, 1, 2 tuy nhiên khi xảy ra tràn cờ TF1 không được thiết lập bằng 1.

4. Bộ định thời 2 của 8052

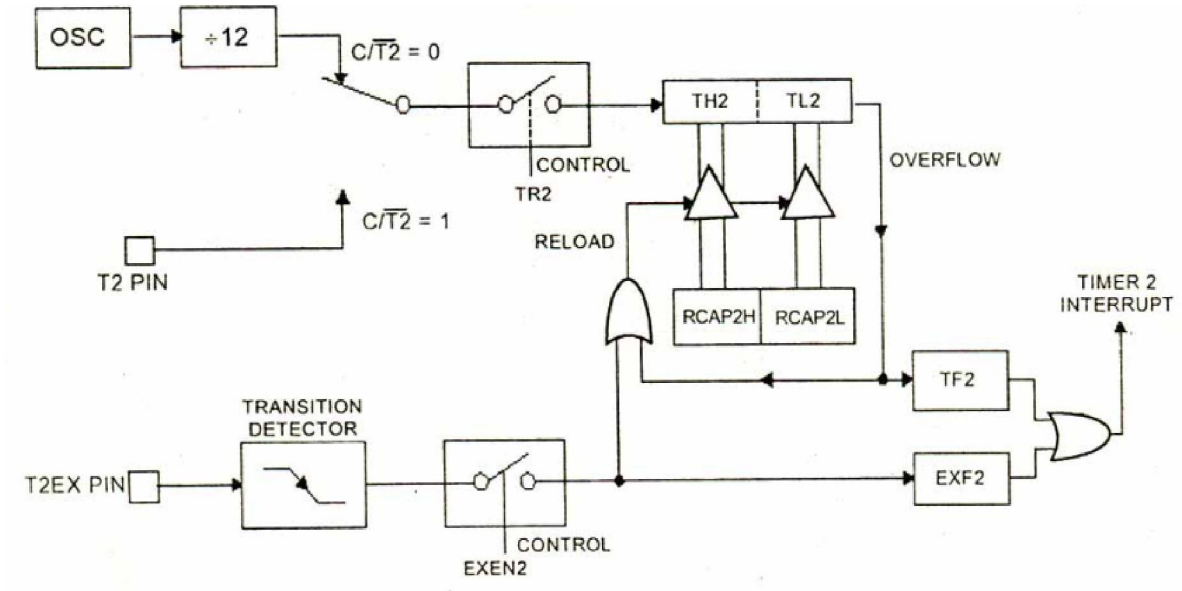
Mục tiêu:

- Hiểu được sơ đồ và nguyên lý hoạt động các chế độ hoạt động bộ định thời 2 của 8052

- Biết được các bước lập trình ở các chế độ hoạt động

- Viết được các chương trình điều khiển sử dụng bộ định thời 2 ở các chế độ khác nhau

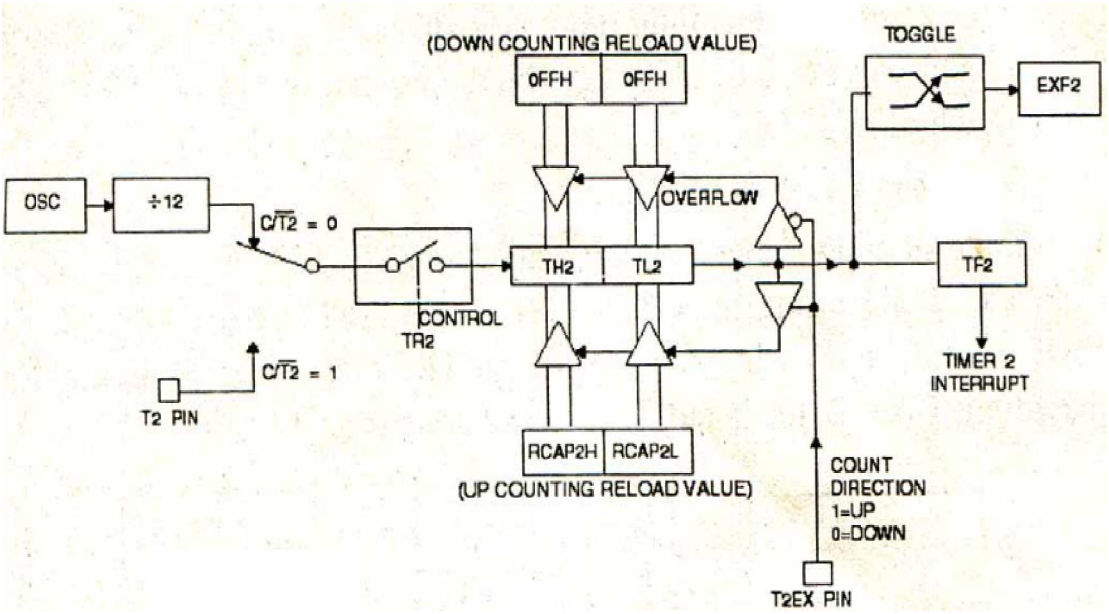
4.1. Chế độ tự nạp lại



Hình 32-04-5 Timer 2 ở chế độ tự nạp lại (DCEN = 0)

Trong chế độ này, khi bit DCEN = 0 Timer 2 hoạt động như một Timer 16 bit tự nạp lại. Giá trị nạp lại được chứa trong RCAP2H và RCAP2L, sự kiện nạp lại xảy ra khi: Hoặc là xảy ra tràn tức là có sự chuyển số đếm từ FFFFH sang 0

Hoặc là có sự chuyển mức từ 1 xuống 0 trên chân T2EX khi EXEN2 đã được đặt bằng 1. Sự chuyển mức này cũng đồng thời đặt EXF2 = 1.



Hình 32-04-6 Timer 2 ở chế độ tự nạp lại (DCEN = 1)

Khi bit DCEN = 1, Timer vẫn hoạt động như một Timer 16 bit tự nạp lại có 2 cách tự nạp lại

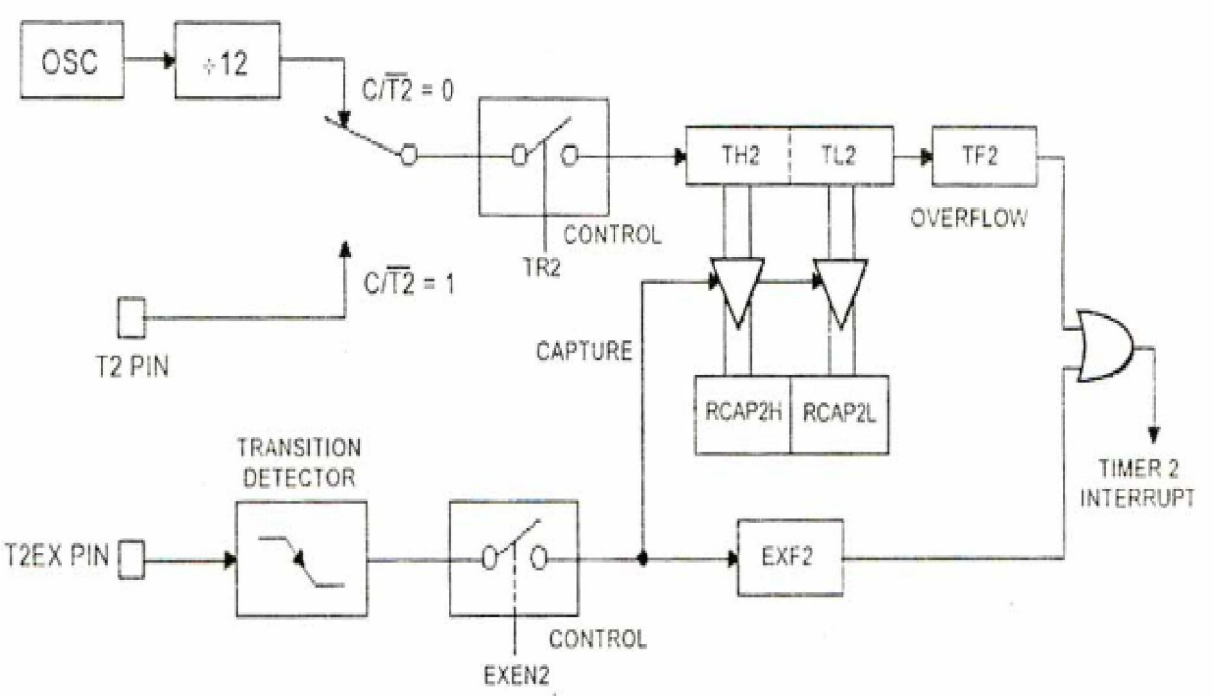
Cách thứ nhất: Khi chân T2EX được đặt ở mức logic 1, Timer 2 sẽ đếm tiến từ giá trị xuất phát cho đến khi có sự chuyển số đếm từ FFFFH sang 0 thì xảy ra tràn

Cách thứ hai: Khi chân T2EX được đặt ở mức logic 0 Timer 2 sẽ đếm lùi từ giá trị xuất phát cho đến giá trị được đặt trong RCAP2H và RCAP2L thì xảy ra tràn.

4.2. Chế độ thu nhận

Khi CP/RL2 = 1, chế độ thu nhận của Timer 2 được chọn bởi bit EXEN2. Xung clock cấp cho Timer 2 cũng được lấy từ một trong hai nguồn và được điều khiển bởi C/T2, điều khiển hoạt động của Timer 2 cũng là bit TR2. Giá trị đếm của Timer được chứa trong TH2 và TL2, khi xảy ra tràn cờ tràn TF2 được đặt bằng 1

Giá trị hiện thời của Timer 2 nằm trong TH2 và TL2 sẽ được chuyển tương ứng vào RCAP2H và RCAP2L khi bit EXEN2 được đặt bằng 1 và có sự chuyển mức từ 1 xuống 0 trên chân T2EX



Hình 32-04-7 Timer 2 ở chế độ thu nhận

THỰC HÀNH VỀ BỘ ĐỊNH THỜI

BÀI 1: ĐIỀU KHIỂN CÁC LED ĐƠN

I. MỤC TIÊU

- Giúp sinh viên tìm hiểu về timer trong 8051
- Biết cách tính toán các thông số delay của Timer trong vi điều khiển.
- Biết cách viết các chương trình tạo thời gian trễ với các khoảng thời gian bất kỳ.

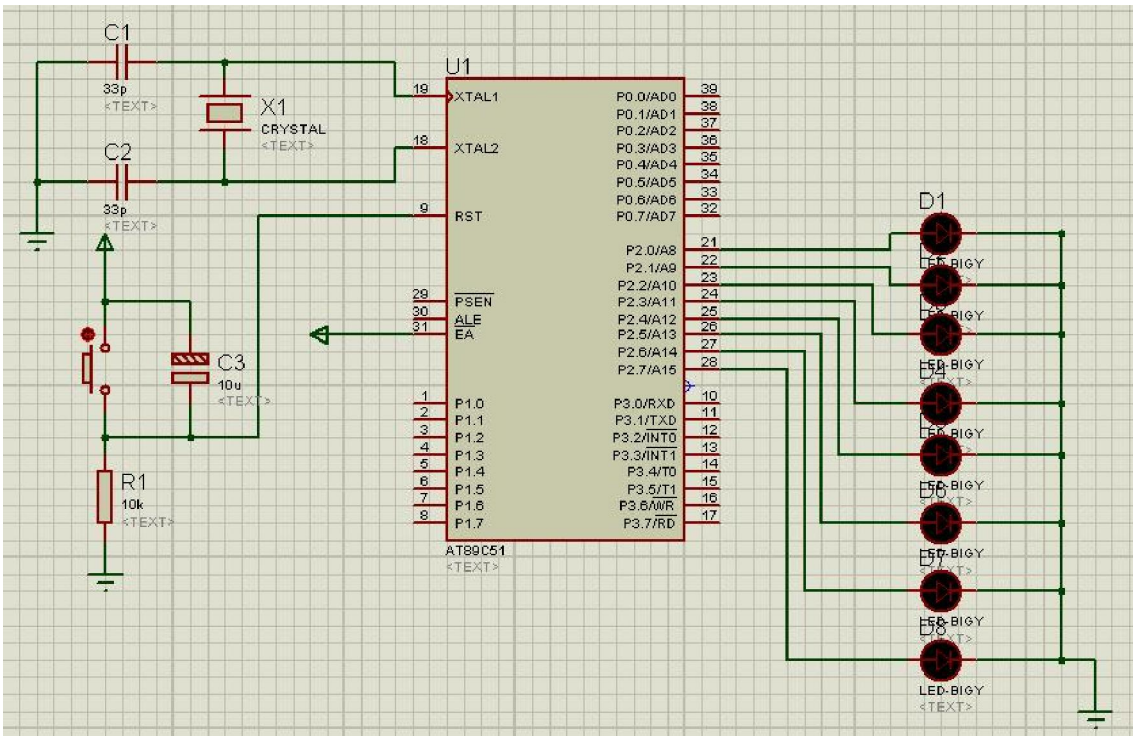
II. NỘI DUNG THÍ NGHIỆM

1. Nối mạch thí nghiệm

Các led tương ứng từ led1 đến led8 sẽ nối với các bit P1.0 đến P1.7, các led đều tác động ở mức thấp

2. Viết chương các trình ứng dụng

Sơ đồ kết nối như hình vẽ



Chương trình 1.1: Viết chương trình hiển thị các led sáng dần từ D1 đến D8, với thời gian Delay 200ms, sử dụng timer0, mode 1

```
 MAIN:    MOV      A,#0
BEGIN:  MOV      P2,A
  CALL     DL_200MS
  SETB     C
  RLC      A              ; dịch bit 1 vào A để sáng dần các led
  JNC      BEGIN
  SJMP     MAIN
DL_200MS:
  PUSH     02
  MOV      TMOD,#01H
  MOV      R2,#20         ; lặp lại 20 lần
X1:
  MOV      TH0,#HIGH(-10000)
  MOV      TL0,#LOW(-10000)
  SETB     TR0
CLR       TF0
  JNB      TF0,$
```

```
        DJNZ    R2,X1
        POP     02
        RET
```

Chương trình 1.2: Viết chương trình hiển thị các led sáng tắt xen kẻ, với thời gian Delay 500ms, sử dụng timer1, mode 1

```
MAIN:   MOV     A,#0FFH
        MOV     P2,A
        CALL    DL_200MS
        MOV     A,#00H
        CALL    DL_200MS
        SJMP    MAIN
DL_200MS:
  PUSH    02
  MOV     TMOD,#10H
  MOV     R2,#50        ; lặp lại 20 lần
X1:
  MOV     TH0,#HIGH(-10000)
  MOV     TL0,#LOW(-10000)
  SETB    TR0
CLR     TF0
  JNB     TF0,$
  DJNZ    R2,X1
  POP     02
  RET
```

Chương trình 1.3: Hãy tạo chương trình con điều khiển tạo thời gian trễ 200µs, 20ms, 2s sử dụng Timer.

```
;CHUONG TRINH DIEU KHIEN 8 LED SANG TAT
;************************************************
;KET NOI: 8 LED -> PORT0 (CO DEM DAO).
;************************************************
```

```
  ORG      00H
CHOPTAT:
MOV      P0,#00H    ;LED TAT
  LCALL   DELAY200US
            MOV      P0,#0FFH  ;LED SANG
            LCALL   DELAY200US
            SJMP     CHOPTAT  ;QUAY LAI
;*************************************************
;CHUONG TRINH CON TAO THOI GIAN TRE
;*************************************************
;TIME DELAY: 200US
;*************************************************
;TUONG DOI:       TIME = [R0]*[R1]*T
;CHINH XAC:       TIME = 2T+2T+1T+(1T+2T*[R1]+2T)*[R0]+2T+2T+2T
;VOI T LA CHU KY MAY
;*************************************************
DELAY200US:
  PUSH      00H
  PUSH      01H
  MOV      R0,#20
DEL:
  MOV      R1,#10
  DJNZ      R1,$
  DJNZ      R0,DEL
  POP       01H
  POP       00H
            RET
  END
```

- Ứng dụng chương trình tạo thời gian trễ 20ms:

```
;*************************************************
```

```
;CHUONG TRINH DIEU KHIEN 8 LED SANG TAT
;*************************************************
;KET NOI: 8 LED -> PORT0 (CO DEM DAO).
;*************************************************
 ORG      00H
CHOPTAT:
MOV     P1,#00H    ;LED TAT
 LCALL   DELAY20MS
          MOV     P1,#0FFH  ;LED SANG
          LCALL   DELAY20MS
          SJMP    CHOPTAT  ;QUAY LAI
;*************************************************
;CHUONG TRINH CON TAO THOI GIAN TRE
;*************************************************
;TIME DELAY: 20MS
;*************************************************
;TUONG DOI:      TIME = (10000H-[TH0,TL0]H)*T
;CHINH XAC:      TIME = 2T+2T+2T+1T+(10000H-[TH0,TL0]H)*T+1T+1T++2T
;VOI T LA CHU KY MAY
;*************************************************
DELAY20MS:
          MOV     TMOD,#01H
          MOV     TH0,#0B1H
          MOV     TL0,#0E0H
          SETB    TR0
          JNB     TF0,$
   CLR      TR0
          CLR     TF0
          RET
END
```

- Ứng dụng chương trình tạo thời gian trễ 2s:

```
;*************************************************
;CHUONG TRINH DIEU KHIEN 8 LED SANG TAT
;*************************************************
;KET NOI: 8 LED -> PORT0 (CO DEM DAO).
;*************************************************
 ORG      00H
CHOPTAT:
MOV     P0,#00H    ;LED TAT
 LCALL   DELAY2S
         MOV     P0,#0FFH  ;LED SANG
         LCALL   DELAY2S
         SJMP    CHOPTAT  ;QUAY LAI
;*************************************************
;CHUONG TRINH CON TAO THOI GIAN TRE
;*************************************************
;TIME DELAY: 2S
;*************************************************
;TUONG DOI:      TIME = (10000H-[TH0,TL0]H)*[R0]*T
;CHINH   XAC:                    TIME   =   2T+1T+2T+(2T+2T+1T+(10000H-
[TH0,TL0]H)*T+1T+1T+2T)*[R0]+2T+2T
;VOI T LA CHU KY MAY
;*************************************************
DELAY2S:
 PUSH     00H
         MOV     R0,#200
         MOV     TMOD,#01H
DEL:
         MOV     TH0,#0D8H
         MOV     TL0,#0F0H
```

```
        SETB    TR0
        JNB     TF0,$
        CLR     TR0
        CLR     TF0
        DJNZ    R0,DEL
POP     00H
        RET
END
```

3. Bài tập

- Bài 1: Hãy viết chương trình con điều khiển tạo thời gian trễ 250µs sử dụng Timer.
- Bài 2: Hãy viết chương trình con điều khiển tạo thời gian trễ 1ms sử dụng Timer.
- Bài 3: Hãy viết chương trình con điều khiển tạo thời gian trễ 100ms sử dụng Timer.
- Bài 4: Hãy viết chương trình con điều khiển tạo thời gian trễ 1s sử dụng Timer.
- Bài 5: Hãy viết chương trình con điều khiển tạo thời gian trễ 3s sử dụng Timer.
- Bài 6: Hãy viết chương trình con điều khiển tạo thời gian trễ 10s sử dụng Timer.
- Bài 7: Hãy viết chương trình con điều khiển tạo thời gian trễ 1 phút sử dụng Timer.

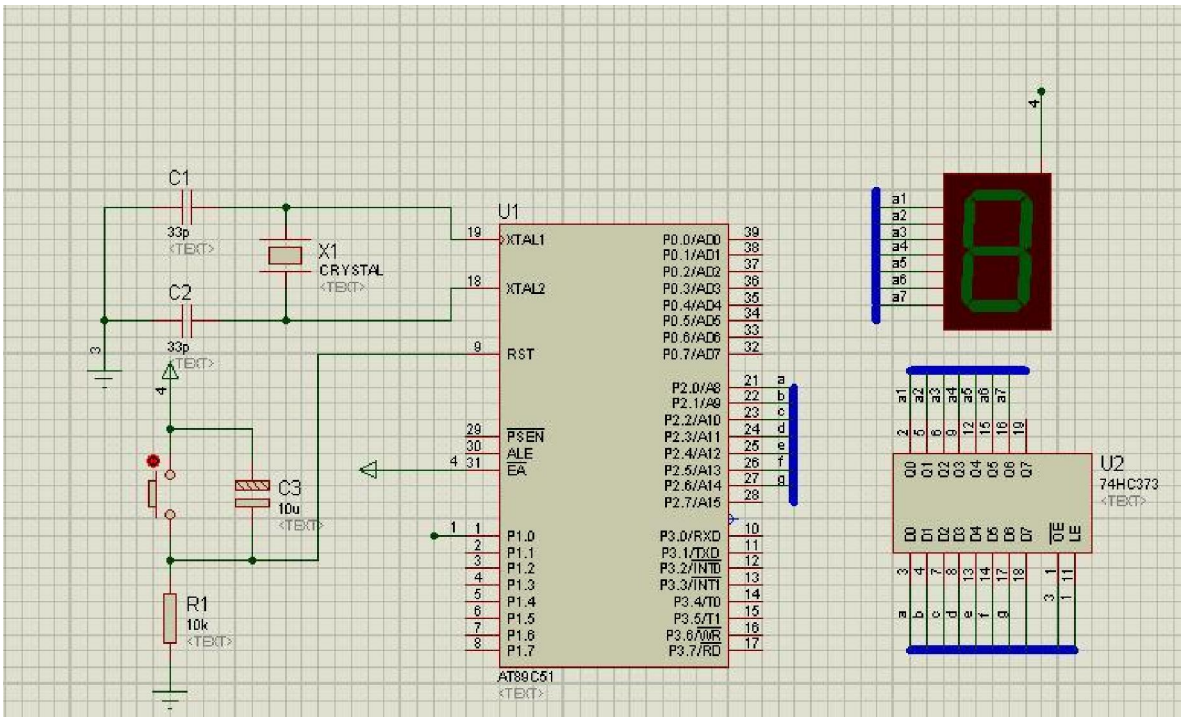
BÀI 2 : LED 7 ĐOẠN VÀ BÀN PHÍM

I. MỤC TIÊU

- Hiểu rõ hơn về tập lệnh của vi điều khiển MCS-51.
- Biết cách viết các chương trình điều khiển LED 7 đoạn ở chế độ
    + Quét led đoạn
    + Chốt dữ liệu
- Biết cách viết các chương trình điều khiển để hiển thị các thông tin và số liệu trên các bộ hiển thị dùng LED 7 đoạn.

II. NỘI DUNG THỰC HÀNH

1. Nối mạch thí nghiệm



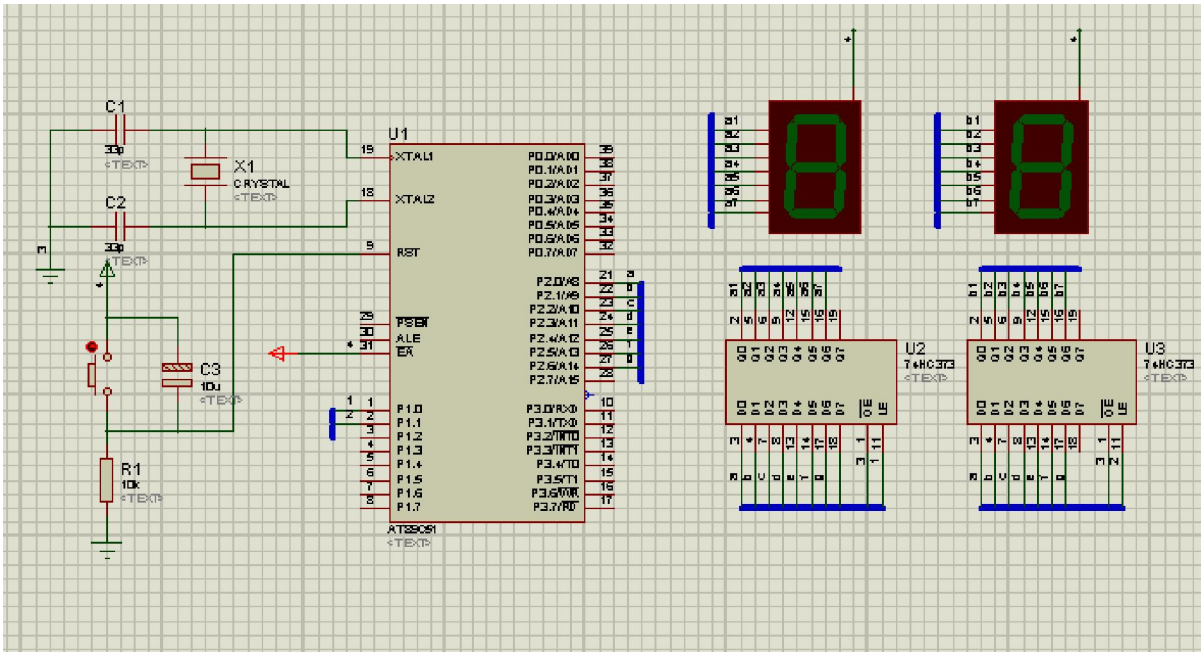
2. Viết chương trình điều khiển

Chương trinh 2.1 : Chương trình điều khiển hiển thị đếm số BCD từ 0 lên 9 trên LED3 (LED3 được nối với Port1).

```
ORG     00H
MAIN:
  MOV     DPTR,#CODEDISP  ;NAP DIA CHI VUNG MA HIEN THI
  MOV     R0,#00H     ;VI TRI DU LIEU HIEN THI TRONG VUNG MA
DISP:
  MOV     A,R0          ;NAP VI TRI DU LIEU
  MOVC    A,@A+DPTR          ;LAY MA HIEN THI
  MOV     P1,A          ;XUAT HIEN THI
LCALL   DELAY500MS
  INC     R0
  CJNE    R0,#10,DISP          ;KIEM TRA VI TRI DU LIEU > 9 (DEM XONG)
  SJMP    MAIN
;*************************************************
DELAY500MS:     ;CHUONG TRINH CON TAO THOI GIAN TRE 500MS
  PUSH    00H
          MOV     R0,#100
```

```
        MOV     TMOD,#01H
LOOP2:
        MOV     TH0,#0ECH
        MOV     TL0,#78H
        SETB    TR0
        JNB     TF0,$
        CLR     TR0
        CLR     TF0
        DJNZ    R0,LOOP2
  POP     00H
  RET
;*************************************************
CODEDISP:           ;VUNG DU LIEU HIEN THI
  DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
  END
```

Chương trinh 2.2: Chương trình điều khiển hiển thị số 54 trên 2 LED7 đoạn (LED 7 đoạn được nối với Port1).




```
MAIN:
        MOV         P1,#00H     ; tắt các led
    BEGIN:
```

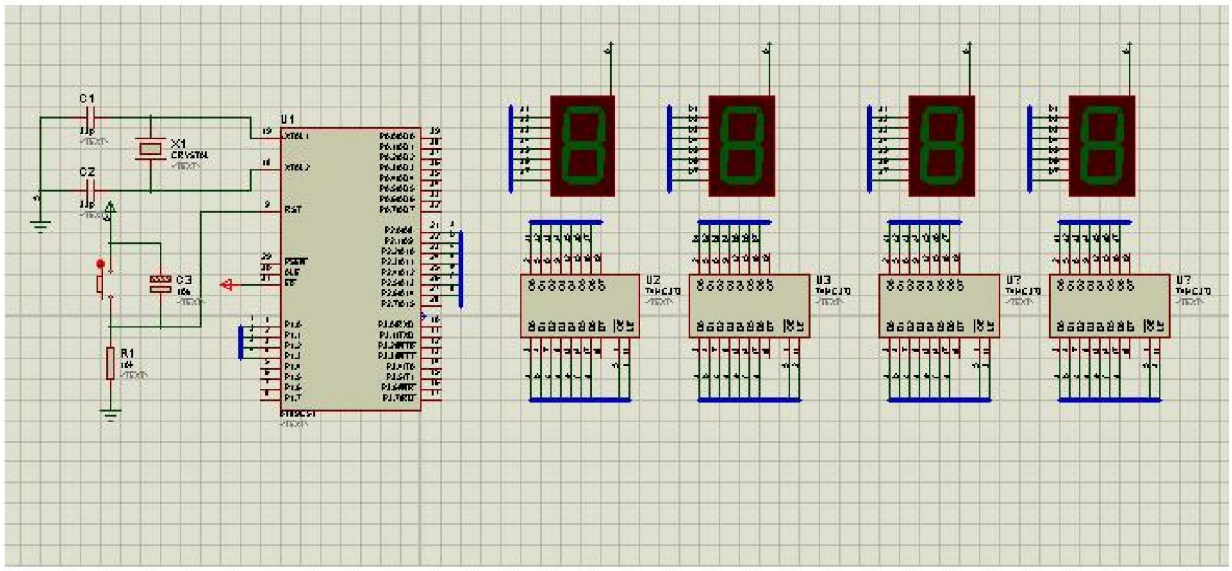
```
        MOV     P2,#6DH     ; đặt DATA 5 ra P1
        SETB    P1.0        ; mở nguồn led 7
        CALL    DELAY       ; trì hoãn để mắt lưu ảnh thông tin
        CLR     P1.0        ; tắt nguồn led 7
        MOV     P2,#66H     ; đặt thông tin DATA 4 ra P1
        SETB    P1.1        ; mở nguồn led 8
        CALL    DELAY       ; trì hoãn để mắt lưu ảnh
        CLR     P1.1        ; tắt nguồn led 8
        LJMP    BEGIN       ; lặp lại quá trình vô hạn
DELAY:                      ; chương trình DELAY ~ 1ms
        PUSH    07H
        MOV     R7,#100
        DJNZ    $
        POP     07H
        RET
        END
```

Chương trình 2.3: Chương trình điều khiển hiển thị đếm số BCD từ 00 lên 99 trên hai LED (LED 7 đoạn được nối với Port1).

```
ORG     00H
MAIN:
  MOV   DPTR,#CODEDISP  ;NAP DIA CHI VUNG MA HIEN THI
  MOV   R0,#00H     ;VI TRI DU LIEU HIEN THI TRONG VUNG MA
  MOV   R1,#00H
DISP:
  MOV   A,R0        ;NAP VI TRI DU LIEU
  MOVC  A,@A+DPTR       ;LAY MA HIEN THI
  MOV   P2,A        ;XUAT HIEN THI (DON VI)
  MOV   A,R1        ;NAP VI TRI DU LIEU
  MOVC  A,@A+DPTR       ;LAY MA HIEN THI
  MOV   P1,A        ;XUAT HIEN THI (CHUC)
```

```
 LCALL   DELAY500MS
 INC     R0     ;TANG DON VI
 CJNE    R0,#10,DISP          ;KIEM TRA DON VI > 9 (DEM XONG)
 MOV     R0,#00H     ;XOA DON VI (DON VI = 0)
 INC     R1     ;TANG CHUC
 CJNE    R1,#10,DISP          ;KIEM TRA CHUC > 9 (DEM XONG)
 SJMP    MAIN
;*************************************************
DELAY500MS:     ;CHUONG TRINH CON TAO THOI GIAN TRE 500MS
 PUSH    00H
         MOV     R0,#100
         MOV     TMOD,#01H
LOOP2:
         MOV     TH0,#0ECH
         MOV     TL0,#78H
         SETB    TR0
         JNB     TF0,$
         CLR     TR0
         CLR     TF0
         DJNZ    R0,LOOP2
 POP     00H
 RET
;*************************************************
CODEDISP:          ;VUNG DU LIEU HIEN THI
 DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
 END
```

Chương trình 2.4 :điều khiển hiển thị số 1234 trên bốn LED (LED7, LED6, LED5 và LED4 được nối với Port2



```
ORG      00H
MAIN:
  MOV     P1,#0FEH  ;CHO PHEP LED 7 SANG
  MOV     P2,#04H    ;HIEN THI SO 4
  LCALL   DELAYLED
  MOV     P1,#0FDH  ;CHO PHEP LED 6 SANG
  MOV     P2,#03H    ;HIEN THI SO 3
  LCALL   DELAYLED
  MOV     P1,#0FBH  ;CHO PHEP LED 5 SANG
  MOV     P2,#02H    ;HIEN THI SO 2
  LCALL   DELAYLED
  MOV     P1,#0F7H  ;CHO PHEP LED 4 SANG
  MOV     P2,#01H    ;HIEN THI SO 1
  LCALL   DELAYLED
  SJMP     MAIN
;*************************************************
DELAYLED:        ;CHUONG TRINH CON TAO THOI GIAN TRE 2.5MS
  PUSH     00H
  PUSH     01H
```

```
  MOV     R1,#5
DEL:    MOV     R0,#250
        DJNZ    R0,$
        DJNZ    R1,DEL
  POP     01H
  POP     00H
  RET
  END
```

Chương trình 2.5 : điều khiển hiển thị đếm số BCD từ 0000 lên 9999 trên bốn LED (LED1, LED2, LED3 và LED4 được nối với Port2)

```
        ORG     00H
MAIN:
  MOV     63H,#00H  ;HANG NGAN
  MOV     62H,#00H  ;HANG TRAM
  MOV     61H,#00H  ;HANG CHUC
  MOV     60H,#00H  ;HANG DON VI
MP1:
  LCALL   DELAY_SCAN      ;DELAY VA QUET HIEN THI
  INC     60H         ;TANG HANG DON VI
  MOV     A,60H
  CJNE    A,#10,MP1  ;KIEM TRA HANG DON VI > 9
  MOV     60H,#00H  ;XOA HANG DON VI (DON VI = 0)
  INC     61H         ;TANG HANG CHUC
  MOV     A,61H
  CJNE    A,#10,MP1  ;KIEM TRA HANG CHUC > 9
  MOV     61H,#00H  ;XOA HANG CHUC (CHUC = 0)
  INC     62H         ;TANG HANG TRAM
  MOV     A,62H
  CJNE    A,#10,MP1  ;KIEM TRA HANG TRAM > 9
```

```
  MOV      62H,#00H  ;XOA HANG TRAM (TRAM = 0)
  INC      63H       ;TANG HANG NGAN
  MOV      A,63H
  CJNE     A,#10,MP1  ;KIEM TRA HANG NGAN > 9
  SJMP     MAIN
DISP7SEGMUL4:
  PUSH     ACC       ;CAT TAM THOI GIA TRI CAC THANH GHI
  PUSH     00H
  MOV      A,#0F7H   ;MA QUET
  MOV      R0,#63H   ;DIA CHI VUNG MA HIEN THI
DISP:
  MOV      P2,@R0    ;XUAT MA HIEN THI
  MOV      P1,A      ;XUAT MA QUET
  LCALL    DELAYLED
  MOV      P1,#0FFH  ;CHONG LAM
  DEC      R0        ;LAY MA HIEN THI KE TIEP
  RR       A    ;CHUYEN SANG LED KE TIEP
  CJNE     R0,#5FH,DISP      ;KIEM TRA DA QUET XONG CHUA
  POP      00H       ;PHUC HOI GIA TRI CHO CAC THANH GHI
  POP      ACC
  RET
;************************************************
DELAYLED:       ;CHUONG TRINH CON TAO THOI GIAN TRE 2.5MS
  PUSH     00H
  PUSH     01H
  MOV      R1,#5
DEL:      MOV      R0,#250
   DJNZ     R0,$
            DJNZ     R1,DEL
  POP      01H
```

```
 POP       00H
 RET
;*************************************************
DELAY_SCAN:     ;CHUONG TRINH CON TAO THOI GIAN TRE 250MS CO GOI CTC QUET HIEN THI
 PUSH     00H
          MOV      R0,#50
          MOV      TMOD,#01H
LOOP2:
          MOV      TH0,#0ECH
          MOV      TL0,#78H
          SETB     TR0
SCAN:      ;LIEN TUC GOI CTC QUET HIEN THI TRONG LUC TIMER DANG CHAY
 LCALL   DISP7SEGMUL4   ;GOI CTC QUET HIEN THI
          JNB       TF0,SCAN
          CLR       TR0
          CLR       TF0
          DJNZ      R0,LOOP2
 POP       00H
 RET
BCD4TO7SEG:
 PUSH     DPH        ;CAT TAM THOI GIA TRI CAC THANH GHI
 PUSH     DPL
 PUSH     ACC
  MOV     DPTR,#CODE7SEG  ;DIA CHI VUNG MA 7 DOAN

 MOV      A,R6        ;LAY SO BCD CAN GIAI MA
 ANL       A,#0FH     ;XOA 4 BIT CAO
 MOVC    A,@A+DPTR        ;LAY MA 7 DOAN TUONG UNG
```

```
  MOV     60H,A      ;CAT BCD HANG DON VI VAO O NHO

  MOV     A,R6       ;LAY SO BCD CAN GIAI MA
  ANL      A,#0F0H   ;XOA 4 BIT THAP
  SWAP     A    ;HOAN CHUYEN CAO - THAP
  MOVC    A,@A+DPTR        ;LAY MA 7 DOAN TUONG UNG
  MOV     61H,A      ;CAT BCD HANG CHUC VAO O NHO

  MOV     A,R7       ;LAY SO BCD CAN GIAI MA
  ANL      A,#0FH     ;XOA 4 BIT CAO
  MOVC    A,@A+DPTR        ;LAY MA 7 DOAN TUONG UNG
  MOV     62H,A      ;CAT BCD HANG TRAM VAO O NHO

  MOV     A,R7       ;LAY SO BCD CAN GIAI MA
  ANL      A,#0F0H   ;XOA 4 BIT THAP
  SWAP     A    ;HOAN CHUYEN CAO - THAP
  MOVC    A,@A+DPTR        ;LAY MA 7 DOAN TUONG UNG
  MOV     63H,A      ;CAT BCD HANG NGAN VAO O NHO

  POP      ACC        ;PHUC HOI GIA TRI CHO CAC THANH GHI
  POP      DPL
POP       DPH
  RET
;*************************************************
CODE7SEG:         ;VUNG CHUA MA 7 DOAN (0 -> 9)
  DB       0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
  END
```

Trang 28

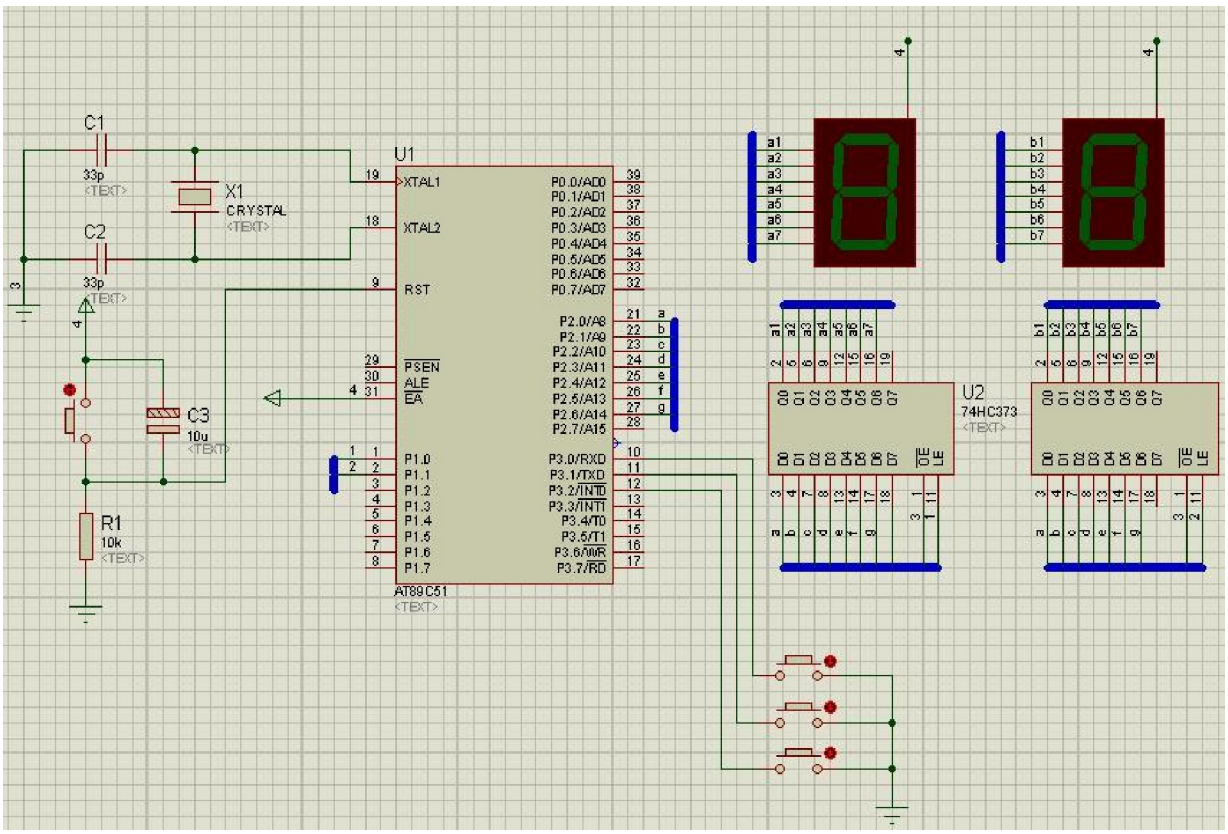
```
            POP    06              ; lấy lại giá trị cũ của R6 trong ngăn xếp
            RET                    ; kết thúc chương trình con.
```

MA_7S: DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH

END

. **Thực hành điều khiển các phím**: phím SW1 nối chân P1.0; SW2 nối chân P1.1; SW3 nối chân P1.2;

Sơ đồ kết nối



Chương trình 2.6: Viết chương trình thực hiện:

- Nhấn SW1: hiển thị số 3 trên led 7 đoạn 1
- Nhấn SW2: hiển thị số 7 trên led 7 đoạn 2
- Nhấn SW3: tắt các thông tin đang hiển thị

HƯỚNG DẪN: dùng lệnh JNB bit, <NHÃN> để kiểm tra phím nhấn; ví dụ:

JNB P3.0, HIENTHI3 ; nếu SW1 nhấn thì logic tại P3.0 là 0, nên lệnh này thỏa điều kiện nhảy đến nhãn HIENTHI3, ngược lại sẽ thực hiện lệnh kế tiếp.

SW1 BIT P1.0

SW2 BIT P1.1

```
SW3             BIT             P1.2
MAIN:
        MOV   P2,#00H               ; tắt các led
        SETB  P2.0                  ; mở nguồn led1
BEGIN:
        JNB     SW1,HT3             ;
        JNB     SW2,HT7
        JNB     SW3,TAT
        LJMP    BEGIN
SW1:
        MOV   P1,#4FH               ;
        LJMP    BEGIN               ; trở về tiếp tục kiểm tra phím
SW2:
        MOV   P1,#07H
        LJMP    BEGIN
SW3:
        MOV   P2,#00H
        LJMP    BEGIN
END
```

Chương trình 2.7: Viết chương trình thực hiện:

- Nhấn SW1: tăng nội dung hiển thị một đơn vị trên led1 (0-1-2-3-4-5…9-0)
- Nhấn SW2: giảm nội dung hiển thị một đơn vị trên led1 (9-8-7…2-1-0-9)

HƯỚNG DẪN: dùng một ô nhớ chứa số đếm, khi nhấn SW1, tăng nội dung ô nhớ, sau đó trì hoãn một thời gian (chờ nhấc tay khỏi phím, thời gian này thường chọn từ 100ms -> 500ms); khi nhấn SW2, giảm nội dung ô nhớ 1 đơn vị, sau đó trì hoãn; nếu không phím nào được nhấn thì đổi nội dung ô nhớ sang mã led 7 đoạn và hiển thị.

```
        CNT     EQU             127
```

```
SW1     BIT         P1.0
SW2     BIT         P1.1

MAIN:
        MOV         P2,#00H
        SETB        P2.0
        MOV         CNT,#0
BEGIN:
        JNB         SW1,TANG
        JNB         SW2,GIAM
        MOV         A,CNT
        MOVC        A,@A+DPTR
        MOV         P1,A
        LJMP        BEGIN
TANG:
INC         CNT
        MOV         A,CNT
        CJNE        A,#10,TROVE
        MOV         CNT,#0
TROVE:
        CALL        DELAY
        LJMP        BEGIN
GIAM:
        DEC         CNT
        MOV         A,CNT
        CJNE        A,#255,TROVE
        MOV         CNT,#0
        LJMP        TROVE

DELAY:
```

```
        PUSH  05            ; cất nội dung R5 vào ngăn xếp
        PUSH  06            ; cất nội dung R6 vào ngăn xếp
        PUSH  07            ; cất nội dung R7 vào ngăn xếp
        MOV   R5,#2
LAP1:
        MOV   R6,#255
LAP:
        MOV   R7,#255
        DJNZ   R7,$         ; Ù X: DJNZ R7,X
        DJNZ   R6, LAP
        DJNZ   R5,LAP1
        POP    07           ; lấy lại giá trị cũ của R7 trong ngăn xếp
        POP    06           ; lấy lại giá trị cũ của R6 trong ngăn xếp
        POP    05           ; lấy lại giá trị cũ của R5 trong ngăn xếp
        RET                 ; kết thúc chương trình con.
MA_7S:      DB    3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH
END
```

BÀI 3. ĐIỀU KHIỂN MA TRẬN LED

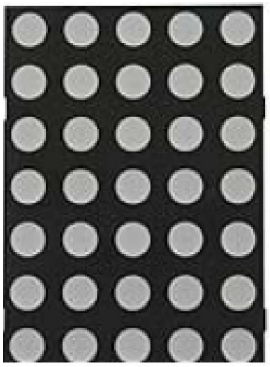
I. MỤC ĐÍCH

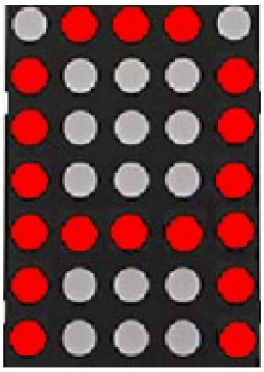
- Hiểu rõ hơn tập lệnh của vi điều khiển MCS-51.
- Biết cách viết các chương trình điều khiển LED ma trận ở các chế độ khác nhau.
- Hiểu được sơ đồ và nguyên lý hoạt động của khối LED ma trận trên mô hình thí nghiệm.
- Hiểu được nguyên lý điều khiển LED ma trận ở các chế độ khác nhau.
- Biết cách viết các chương trình quang báo để hiển thị các thông tin được yêu cầu (thông tin dạng tĩnh và dạng động).

II. NỘI DUNG THỰC HÀNH

Ma trận LED bao gồm nhiều LED cùng nằm trong một vỏ chia thành nhiều cột và hàng, mỗi giao điểm giữa hàng và cột có thể có 1 LED (ma trận LED một màu) hay nhiều LED (2 LED tại một vị trí tạo thành ma trận LED 3 màu). Để LED tại một vị

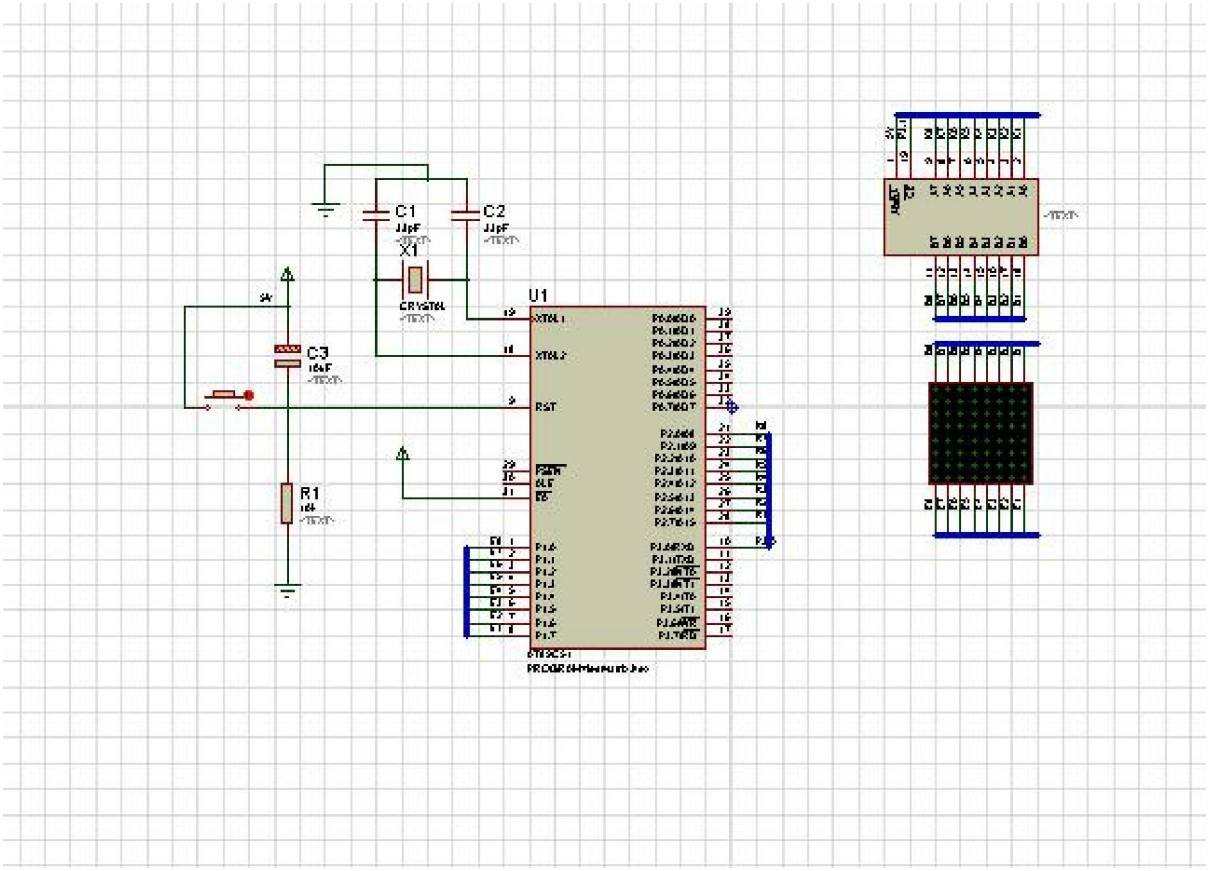
trí nào đó sáng thì phải cấp hiệu điện thế dương giữa Anode và Cathode. Trên cơ sở cấu trúc như vậy, ta có thể mở rộng hàng và cột của ma trận LED để tạo thành các bảng quang báo.





1. Nối mạch thí nghiệm

Chương trình 3.1 Viết chương trình điều khiển hiển thị chữ A màu đỏ trên LED ma trận



2. Viết chương trình điều khiển

```
  ORG     00H
MAIN:
  MOV     P2,#07H    ;XUAT MA HIEN THI RA HANG
```

```
MOV     P1,#01H    ;XUAT MA QUET RA COT 1
LCALL   DELAYLED
MOV     P1,#00H    ;CHONG LEM

MOV     P2,#0DBH  ;XUAT MA HIEN THI RA HANG
MOV     P1,#02H    ;XUAT MA QUET RA COT 2
LCALL   DELAYLED
MOV     P1,#00H    ;CHONG LEM

MOV     P2,#0DDH  ;XUAT MA HIEN THI RA HANG
MOV     P1,#04H    ;XUAT MA QUET RA COT 3
LCALL   DELAYLED
MOV     P1,#00H    ;CHONG LEM

MOV     P2,#0DBH  ;XUAT MA HIEN THI RA HANG
MOV     P1,#08H    ;XUAT MA QUET RA COT 4
LCALL   DELAYLED
MOV     P1,#00H    ;CHONG LEM

MOV     P2,#07H    ;XUAT MA HIEN THI RA HANG
MOV     P1,#10H    ;XUAT MA QUET RA COT 5
LCALL   DELAYLED
MOV     P1,#00H    ;CHONG LEM

MOV     P2,#0FFH  ;XUAT MA HIEN THI RA HANG
MOV     P1,#20H    ;XUAT MA QUET RA COT 6
LCALL   DELAYLED
MOV     P1,#00H    ;CHONG LEM

MOV     P2,#0FFH  ;XUAT MA HIEN THI RA HANG
```

```
  MOV      P1,#40H    ;XUAT MA QUET RA COT 7
  LCALL   DELAYLED
  MOV      P1,#00H    ;CHONG LEM

  MOV      P2,#0FFH  ;XUAT MA HIEN THI RA HANG
  MOV      P1,#80H    ;XUAT MA QUET RA COT 8
  LCALL   DELAYLED
  MOV      P1,#00H    ;CHONG LEM

  LJMP      MAIN
DELAYLED:        ;CHUONG TRINH CON TAO THOI GIAN TRE 2.5MS
  PUSH      00H
  PUSH      01H
  MOV      R1,#5
DEL:
  MOV      R0,#250
            DJNZ      R0,$
            DJNZ      R1,DEL
  POP       01H
  POP       00H
  RET
  END
```

Chương trình 3.2: Chương trình điều khiển hiển thị lần lượt các chữ A, B, C, a, b, c màu đỏ trên LED ma trận.

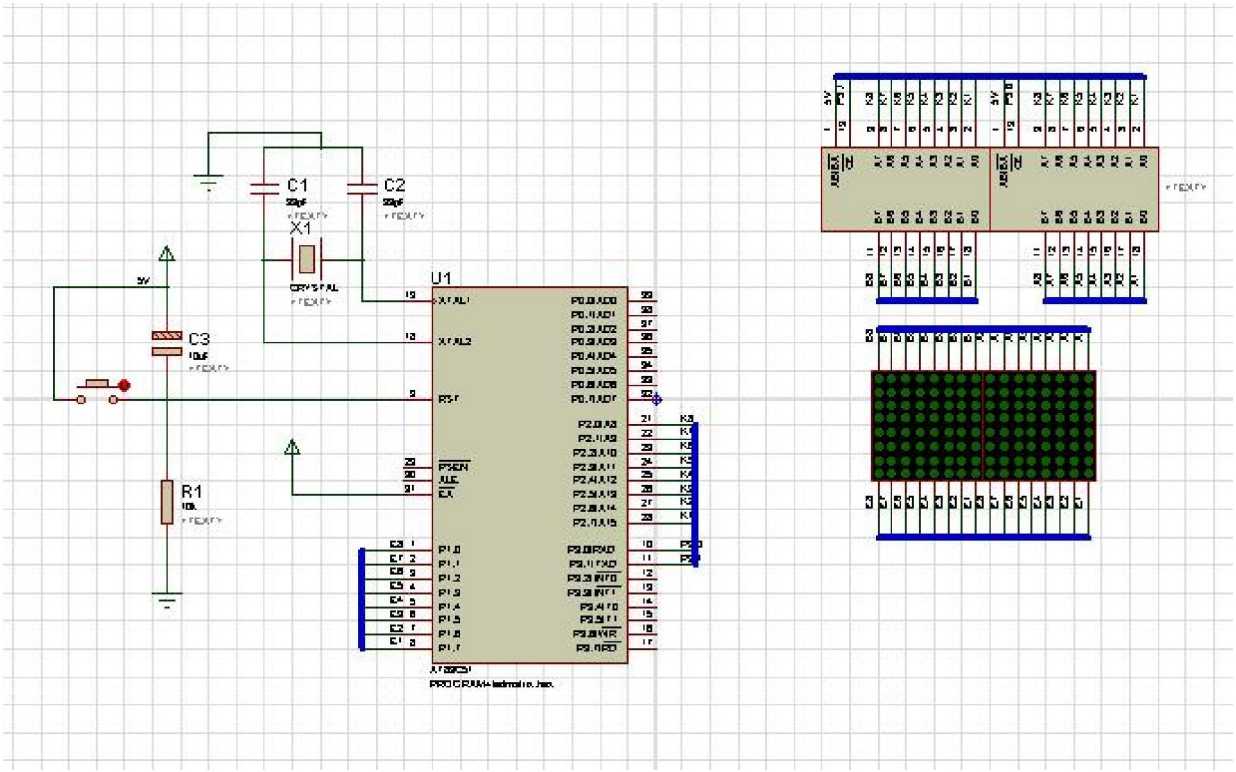
```
BATDAU:
MOV DPTR,#MALED
MAIN:       MOV R0,#30
 LAP:       MOV R1,#0
       MOV R2,#07FH
```

```
LAP1:   MOV A,R2
        RL A
        MOV R2,A
        MOV P3,A
        MOV A,R1
        MOVC A,@A+DPTR
        CLR P1.0
        MOV P2,A
        CALL DELAY
        INC R1
        CJNE R2,#07FH,LAP1
        DJNZ R0,LAP
        INC DPTR
        MOV R3,DPL
        CJNE R3,#48,MAIN
        JMP BATDAU
DELAY:
MOV R4,#8
N1:MOV R5,#8
N2:MOV R6,#4
N3:DJNZ R6,N3
   DJNZ R5,N2
   DJNZ R4,N1

RET
        CODEDISP:
          DB        07H,0DBH,0DDH,0DBH,07H,0FFH,0FFH,0FFH  ;CHU A
          DB         01H,6DH,6DH,6DH,93H,0FFH,0FFH,0FFH     ;CHU B
          DB         83H,7DH,7DH,7DH,0BBH,0FFH,0FFH,0FFH    ;CHU C
          DB         0BFH,57H,57H,57H,0FH,0FFH,0FFH,0FFH    ;CHU a
```

```
        DB      01H,6FH,77H,77H,8FH,0FFH,0FFH,0FFH      ;CHU b
        DB      8FH,77H,77H,77H,0BFH,0FFH,0FFH,0FFH     ;CHU c
END
```

Chương trình 3.3 Viết chương trình cho chuỗi ‘CAO DANG NGHE CONG NGHIEP HAI PHONG’ di chuyển từ trái sang phải trên 2 ma trận Led




```
ORG 000H
BATDAU:
MOV DPTR,#MALED
MAIN:
MOV R7,#30
LAP:
MOV R0,#8
MOV R6,#0
MOV R5,#07FH

MAIN1:
MOV A,R5
```

```
RL A
MOV R5,A
MOV P1,A
MOV A,R0
MOVC A,@A+DPTR
CLR P3.0
SETB P3.1
MOV P2,A
INC R0
ACALL DELAY
CJNE R5,#07FH,MAIN1

MAIN2:

MOV A,R5
RL A
MOV R5,A
MOV P1,A
MOV A,R6
MOVC A,@A+DPTR
CLR P3.1
SETB P3.0
MOV P2,A
INC R6
ACALL DELAY
CJNE R5,#07FH,MAIN2

DJNZ R7,LAP
INC DPTR
MOV R4,DPL
```

```
CJNE R4,#142,MAIN
SJMP BATDAU

DELAY:
MOV R1,#8
N1:MOV R2,#8
N2:MOV R3,#4
N3:DJNZ R3,N3
   DJNZ R2,N2
   DJNZ R1,N1

RET
ORG 0800H
MALED:
DB     0H,0H,0H,0H,0H,0H,0H,0H
DB     0H,7CH,82H,82H,44H,0H,0H,0F8H
DB     0CH,0AH,0CH,0F8H,0H,0H,7CH,82H
DB     82H,82H,7CH,0H,0H,0H,0H,0H
DB     10H,0FEH,92H,82H,7CH,0H,0F8H,0CH
DB     0AH,0CH,0F8H,0H,0FEH,4H,8H,10H
DB     0FEH,0H,7CH,82H,0A2H,0E2H,20H,0H
DB      1H,0FBH,0F7H,0EFH,1H,0FFH,83H,7DH
DB      7DH,5DH,9BH,0DFH,0FFH,1H,0EFH,0EFH
DB      7DH,5DH,9BH,0DFH,0FFH,1H,0EFH,0EFH
DB      0EFH,1H,0FFH,1H,6DH,6DH,6DH,0FFH
DB     83H,7DH,7DH,7DH,0BBH,0FFH,83H,7DH
DB      7DH,7DH,83H,0FFH,1H,0FBH,0F7H,0EFH
DB      1H,0FFH,83H,7DH,7DH,5DH,9BH,0DFH
DB     1H,0FBH,0F7H,0EFH,1H,0FFH,83H,7DH
DB      7DH,5DH,9BH,0DFH,0FFH,1H,0FFH,1H
```

DB 6DH,6DH,7DH,0FFH,1H,0EDH,0EDH,0F3H

DB 0FFH,1H,0F7H,0F7H,1H,0FFH,7H,0F3H

DB 0F5H,0F3H,7H,0FFH,1H,0FFH,1H,0EDH

DB 0EDH,0F3H,0FFH,1H,0EFH,0EFH,1H,0FFH

DB 83H,7DH,7DH,0FDH,83H,0FFH,1H,0FBH

BÀI 4. ĐỘNG CƠ BƯỚC

I. MỤC TIÊU

- Hiểu rõ hơn về tập lệnh của vi điều khiển MCS-51.

- Biết cách viết các chương trình điều khiển động cơ bước quay thuận/ngược, quayliên tục/từng bước, ….

-Hiểu được sơ đồ và nguyên lý hoạt động của khối động cơ bước trên mô hình thí nghiệm.

- Hiểu được nguyên lý hoạt động và nguyên lý điều khiển động cơ bước.

- Biết cách viết các chương trình ứng dụng điều khiển động cơ bước hoạt động theo các chế độ khác nhau.

II. NỘI DUNG THỰC HÀNH

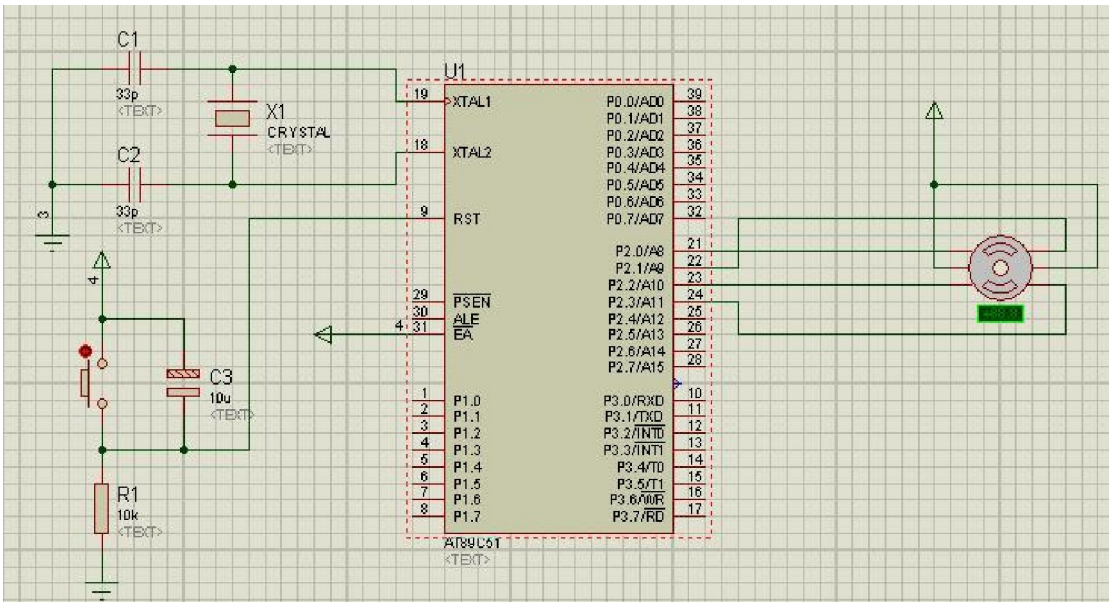
1. Lắp mạch thí nghiệm

Chương trình 4.1 Viết chương trình điều khiển động cơ quay thuận mỗi lần một bước với tốc độ 50vòng/phút (giả sử động cơ có góc quay là 7.20/bước).

Góc quay 7.20/bước → 1 vòng quay cần 3600/7.20 = 50 bước → 50 vòng quay cần thực hiện 2500 bước. Tốc độ 50 vòng / phút → 1 phút (60s) thực hiện 2500 bước → mỗi bước cần 60/2500 = 0.024s = 24,,000 μs.

Thứ tự kích xung như bảng sau

Ngược				Thuận			
1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
1	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1
0	0	1	0	0	0	1	0
0	0	0	1	0	1	0	0
1	0	0	0	1	0	0	0



2. Viết chương trình điều khiển

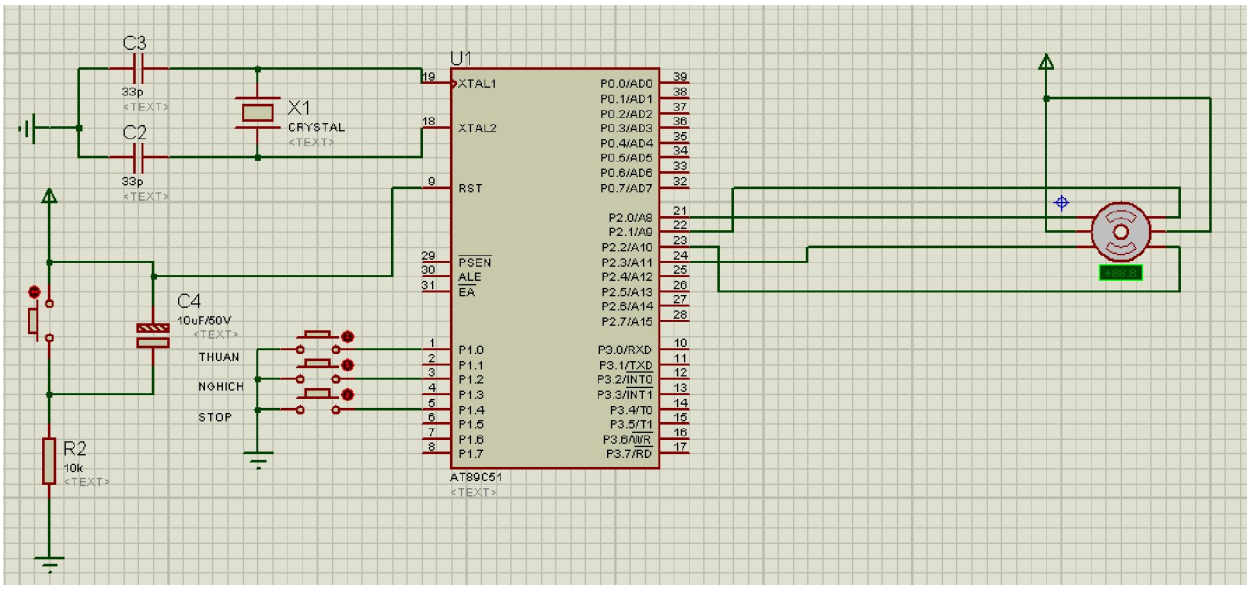
```
main:
MOV R0,#0
 MOV DPTR,#thuan1buoc
begin:
 MOV A,R0
 MOVC A,@A+DPTR
MOV P2,A     ;
CALL Delay
INC R0
CJNE R0,#4,begin
SJMP main
;-----------------
Delay:
MOV TMOD,#01h
MOV TH0,#HIGH(-24000)
MOV TL0,#LOW(-24000)
SETB TR0
JNB TF0,$
CLR TF0
CLR TR0
```

RET

thuan1buoc: DB 08h,04h,02h,01h

END

Chương trình 4.2 Điều khiển động cơ bước quay tuận, ngược, dừng.



Chương trình:

```
        thuan bit     p1.0
        nghich        bit     p1.2
        stop   bit    p1.4
org 00h
        mov r0,#0
main:
        jb thuan,x1
        mov r0,#1
x1:
        jb nghich,x2
        mov r0,#2
x2:
        jb stop,x3
        mov r0,#0
x3:
        cjne r0,#0,x4
```

```
        jmp main
x4:
        cjne r0,#1,x5
        call quaythuan
        jmp main
x5:
        cjne r0,#2,x6
        call quaynghich
x6:     jmp main
quaythuan:
        mov p2,#11111110b
        call delayms
        mov p2,#11111101b
        call delayms
        mov p2,#11111011b
        call delayms
        mov p2,#11110111b
        call delayms
        ret
quaynghich:
        mov p2,#11110111b
        call delayms
        mov p2,#11111011b
        call delayms
        mov p2,#11111101b
        call delayms
        mov p2,#11111110b
        call delayms
        ret
delayms:
```

```
        mov   70h,#255
dl:     mov   71h,#255
        djnz  71h,$
        djnz  70h,dl

        ret
end
```

BÀI 5: CỔNG NỐI TIẾP

Mã bài: MĐ 25-05

Mục tiêu:

- Biết được đặc điểm các thanh ghi của cổng nối tiếp
- Hiểu được cấu tạo và các chế độ làm việc của cổng nối tiếp
- Biết cách khởi tạo cổng nối tiếp
- Biết cách thu, phát dữ liệu nối tiếp

Nội dung chính:

1. Khái quát chung

Mục tiêu:

- Hiểu được chức năng của cổng nối tiếp

- Biết được hoạt động của cổng nối tiếp

Cổng nối tiếp tích hợp trong họ 8051 có vài chế độ hoạt động trong một phạm vị tần số rộng, chức năng cơ bản của cổng nối tiếp là biến đổi tín hiệu xuất từ song song sang nối tiếp và tín hiệu nhập từ nối tiếp sang song song.

Thiết bị ngoại vi giao tiếp với port nối tiếp qua các chân TXD và RXD, các chân này là các chân đa chức năng của port 3, bít P3.1 tại chân 11 (TXD) và P3.0 tại chân 10 (RXD).

Đặc điểm của port nối tiếp là truyền song công toàn phần (thu phát đồng thời) và đặc tính đệm dữ liệu cho phép lưu giữ ký tử đã nhận trong bộ đệm trong khi nhận ký tự thứ hai, nếu CPU đọc ký tự thứ nhất trước khi hoàn tất việc nhận ký tự thứ hai thì dữ iệu cũng không bị mất.

Có 2 thanh ghi đặc biệt phục vụ cho cổng nối tiếp đó là ghanh ghi đệm SBUF và thanh ghi điều khiển SCON, bộ đệm port nối tiếp có địa chỉ là 99H thực chất gồm có 2 bộ đệm. Ghi vào bộ đệm tức là nạp dữ liệu để xuất ra ngòai và đọc bộ đệm tức là nhận dữ liều từ ngòai vào trong bộ đệm.

Thanh ghi điều khiển SCON có địa chỉ là 98H được định địa chỉ theo bít bao gồm các bít trạng thái và các bít điều khiển. Các bít điều khiển sẽ xác lập chế độ làm việc của port nối tiếp còn các bít trạng thái cho biết sự kết thúc của việc xuất và nhập một ký tự, các bít trạng thái có thể được kiểm tra bằng phần mềm hoặc có thể được lập trình để tạo ra mốt ngắt.

Tần số hoạt động của cổng nối tiếp còn gọi là tốc độ baud (tạo ra từ dao động trên chip 8051) có thể được cố định hoặc thay đổi. Nếu một tốc độ baud thay đổi được sử dụng thì timer 1 sẽ cung cấp xung đồng hồ tốc độ baud và phải được lập trình thích hợp. (Timer 2 trong 8032 và 8052 có thể được lập trình để cung

cấp xung đồng hồ tốc độ baud.)

2. Các thanh ghi của cổng nối tiếp

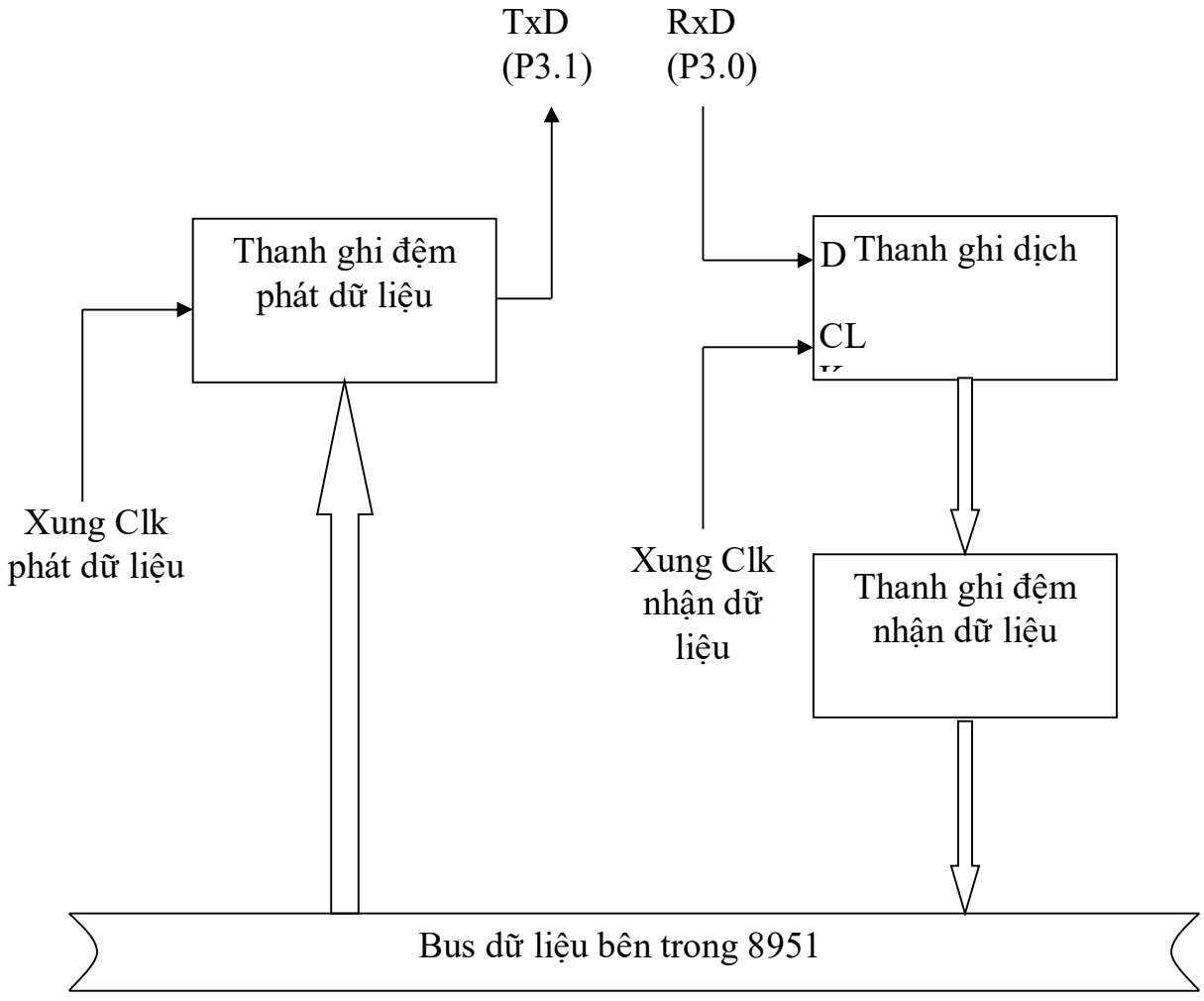
Mục tiêu:

- Hiểu được cấu tạo và hoạt động các thanh ghi cổng nối tiếp

- Biết được chức năng của các thanh ghi trong quá trình truyền và nhận dữ liệu qua cổng nối tiếp

2.1. Thanh ghi đệm cổng nối tiếp (SBUF)

Bộ đệm sbuf trên thực tế gồm 2 bộ đệm. Việc ghi lên sbuf sẽ nạp dữ liệu để phát, việc đọc dữ liệu từ sbuf sẽ truy xuất dữ liệu đã nhận được. Điều này nghĩa là có hai thanh ghi riêng biệt: thanh ghi phát (chỉ ghi) và thanh ghi thu (chỉ đọc). Hình vẽ sau đây mô tả cấu tạo và hoạt động của bộ đệm SBUF.



Hình 32-05-1 Cấu tạo và hoạt động của bộ đệm SBUF

2.2. Thanh ghi điều khiển cổng nối tiếp (SCON)

Các hoạt động của port nối tiếp điều được khai báo trong thanh ghi SCON. Thanh ghi này được mô tả như sau:

BIT	Ký hiệu	Địa chỉ	Mô tả
SCON.7	SM0	9FH	Bit 0 chọ chế độ port nối tiếp
SCON.6	SM1	9EH	Bit 1 chọ chế độ port nối tiếp
SCON.5	SM2	9DH	Bit 2 chọ chế độ port nối tiếp. Bit này cho phép truyền thông đa xử lý ở chế độ 2 và 3. Bit RI sẽ không được tích cực nếu bit thứ 9 nhận được là 0
SCON.4	**REN**	9CH	Cho phép thu. Bit này phải được set đề nhận các ký tự
SCON.3	TB8	9BH	Bit phát thứ 8. Bit phát trong chế độ 2 và 3; tác động bởi phần mềm.
SCON.2	**RB8**	9AH	Bit thu thứ 8.
SCON.1	TI	98H	Cờ ngắt phát. Cờ này được set ngay khi phát xong 1 ký tự.
SCON.0	RI	98H	Cờ ngắt thu. Cờ này được set ngay khi thu xong 1 ký tự.

3. Khởi động và truy xuất các thanh ghi

Mục tiêu:

- Biết cách khởi động các thanh ghi

- Biết cách truy xuất các thanh ghi

3.1. Cho phép thu, phát dữ liệu

Trong thanh ghi SCON, bit-REN =1 để cho phép thu dữ liệu. Lệnh này được thực thi ở đầu chương trình.

setb ren

hoặc:

mov scon,#xxx1xxxxb

Trong thanh ghi SCON, bit-TI =1 để cho phép port nối tiếp sẵn sàng phát dữ liệu. Lệnh này được thực thi ở đầu chương trình.

setb TI

3.2. Bit dữ liệu thứ 9

Trong qua trình phát dữ liệu, bit thứ 9 dùng để truyền bit kiểm tra chẵn lẻ. Trong truyền thông đa xử lý, bit dữ liệu thứ 9 dùng để truyền bit '1' hoặc bit '0' để phân biệt byte định địa chỉ và byte dữ liệu. Khi phát, bit thứ 9 được đưa vào TB8 và khi thu, bit này được nhận về RB8. Bit này còn chứa bit 'STOP' khi truyền dữ liệu ở chế độ 8 bit.

3.3. Thêm bit chẵn lẻ

Bit P trong thanh ghi PSW dùng để thiết lập kiểm tra chẵn cho dữ liệu 8 bit chứa trong thanh ghi A.

Ví dụ. Nếu việc truyền thông yêu cầu 8 bit dữ liệu cộng với một bit kiểm tra chẵn, các lệnh sau được dùng để phát đi 8 bit và bit kiểm tra chẵn chứa trong bit thứ 9:

```
Mov     c,p
Mov     tb8,c
Mov     sbuf,a
```

3.4. Các cờ ngắt

TI và RI là hai cờ ngắt phát, thu chứa trong thanh ghi SCON. Hai cờ này được set lên 1 và xoá bằng phần mềm.

Đoạn lệnh chờ đọc một ký tự:

```
Wait:
    Jnb   RI,$   ;khi RI = '1' bộ đệm hoàn tất việc nhận dữ liệu
    Clr   RI     ;xoá cờ RI cho việc nhận lần sau
```

Mov a,sbuf;đọc dữ liệu về thanh ghi A

Đoạn lệnh chờ phát một ký tự:

Wait:

Jnb RI,$;khi TRI = '1' bộ đệm hoàn tất việc phát dữ liệu

Clr RI ;xoá cờ TI cho việc phát lần sau

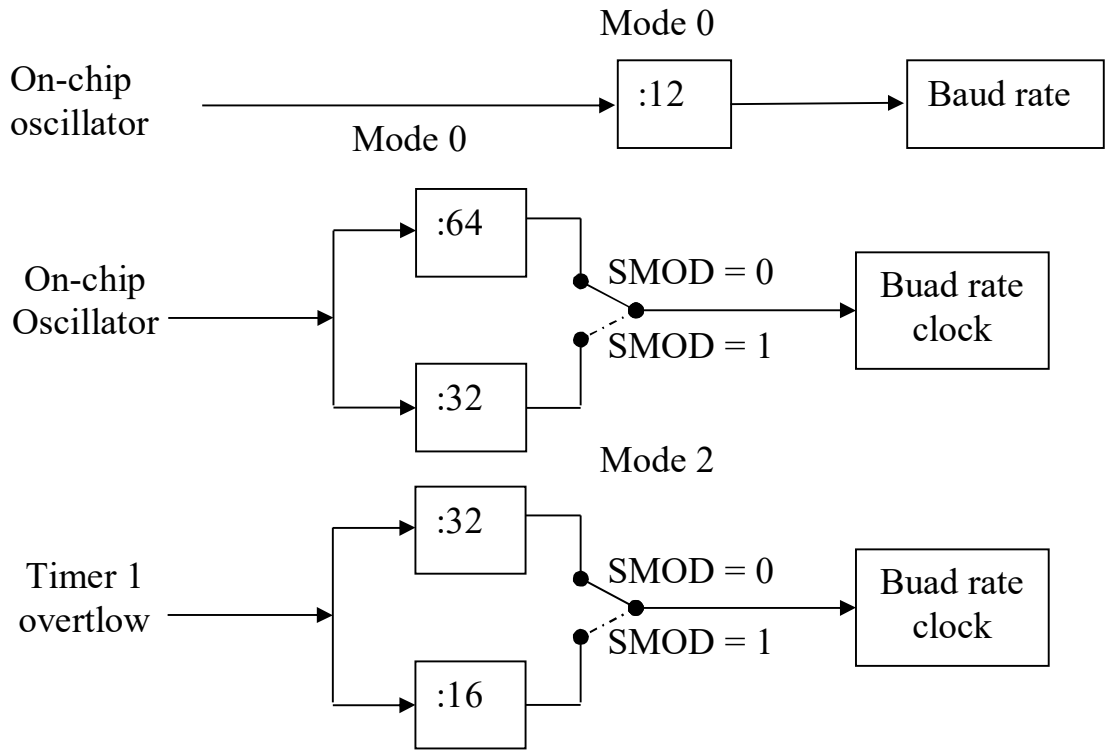
Mov a,sbuf; đưa dữ liệu vào thanh ghi đệm Sbuf

4. Tốc độ baud

Mục tiêu:

- Biết được các loại tốc độ baud

- Biết Cách tính tốc độ baud



Hình 32-05-2 Tạo xung baud cho cổng nối tiếp

Tốc độ baud cố định.

Trong chế độ 0, tốc độ baud luôn bằng tần số của mạch dao động chia cho 12. Nếu thạch anh 12MHz thì tốc độ baud là 1MHz

Trong chế độ 2, khi reset hệ thống, tốc độ baud sẽ bằng dao động trên chip chia cho 64. Khi bit SMOD = 1 (trong thanh ghi PCON) sẽ làm cho tốc độ Baud tăng lên gấp đôi.

Tốc độ baud thay đổi.

Trong chế độ truyền nối tiếp có tốc độ baud thay đổi, ta thường dùng timer 1 chế độ 2 (chế độ tự động nạp lại) và cài đặt giá trị tự nạp lại vào TH1.

Tốc độ baud là tần số xung clock tạo ra để dịch bit và trong mỗi lần tràn của bộ định thời (t), nó tự tạo ra một xung clock. Tần số này qua bộ chia 32 (hoặc 16 nếu SMOD=1)

5. Các chế độ làm việc của cổng nối tiếp

Mục tiêu:

- Hiểu được nguyên lý hoạt động các chế độ làm việc của cổng nối tiếp

- Ứng dụng được các chế độ hoạt động để viết chương trình điều khiển

Để chọn chế độ cho port nối tiếp ta khai báo ‘0’ hoặc ‘1’ vào hai bit SM0 và SM1 trong thnah ghi SCON.

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)

5.1. Chế độ 0_ Thanh ghi dịch 8 bit

Chế độ 0 đặt port nối tiếp vào chế độ thanh ghi dịch 8 bit. Dữ liệu thu và phát thông qua chân RxD và chân TxD sẽ xuất xung Clock dịch bit. Khi phát và thu dữ liệu 8 bit, bit có trọng số thấp ngất được phát đi trước. Thuật ngữ “RxD” và “TxD”

không đúng trong trường hợp này vì chân RxD dùng thu và phát dữ liệu còn chân TxD dùng làm chân xuất xung clock dịch bit.

Việc phát dữ liệu được thực hiện bằng một lệnh ghi dữ liệu vào SBUF. Việc thu dữ liệu được khởi động khi bit REN=1 và cờ RI=0 để bắt đầu nhận xung clock thu dữ liệu. Ở chế độ 0 còn thích hợp cho việc mở rộng thêm các ngõ ra cho port nối tiếp.

5.2. Chế độ 1_UART 8 bit tốc độ baud thay đổi

UART (Universal asynchronous receiver transmitter_thu phát không đồng bộ) là một bộ thu phát dữ liệu với mỗi ký tự dữ liệu được kèm bởi một bit *Start(0)* ở đầu và một bit *Stop(1)* ở cuối cùng. Nếu trong trường hợp có kiểm tra chẵn lẻ, bit chẵn (lẻ) được đi kèm trúoc bit Stop.

Vậy trong chế độ 1 có tất cả 10 bit được truyền đi (1 bit Start=0, 8 bit dữ liệu và 1 bit Stop=1). Khi thu bit thứ 8 chứa trong RB8. Với 8951, tốc độ baud được thiết lập bởi tốc độ tràn của timer1 còn ở 8952 tốc độ tràn được thiết bởi timer1 hoặc 2 hoặc cà 2.

5.3. Chế độ 2_UART 9 bit tốc độ baud cố định

Trong chế độ này có tất cả 11 bit được thu hoặc phát (bit Start, 8 bit data, bit 9, Stop). Khi phát bit thứ 9 chứa trong TB8 và khi thu bit thứ 9 chứa trong RB8. Tốc độ baud cố định bằng 1/32 hoặc 1/64 dao động trên chip.

5.4. Chế độ 3_UART 9 bit tốc độ baud thay đổi

Trong chế độ này có tất cả 11 bit được thu hoặc phát (bit Start, 8 bit data, bit 9, Stop). Khi phát bit thứ 9 chứa trong TB8 và khi thu bit thứ 9 chứa trong RB8. Tốc độ baud thay đổi bằng bộ định thời.

THỰC HÀNH

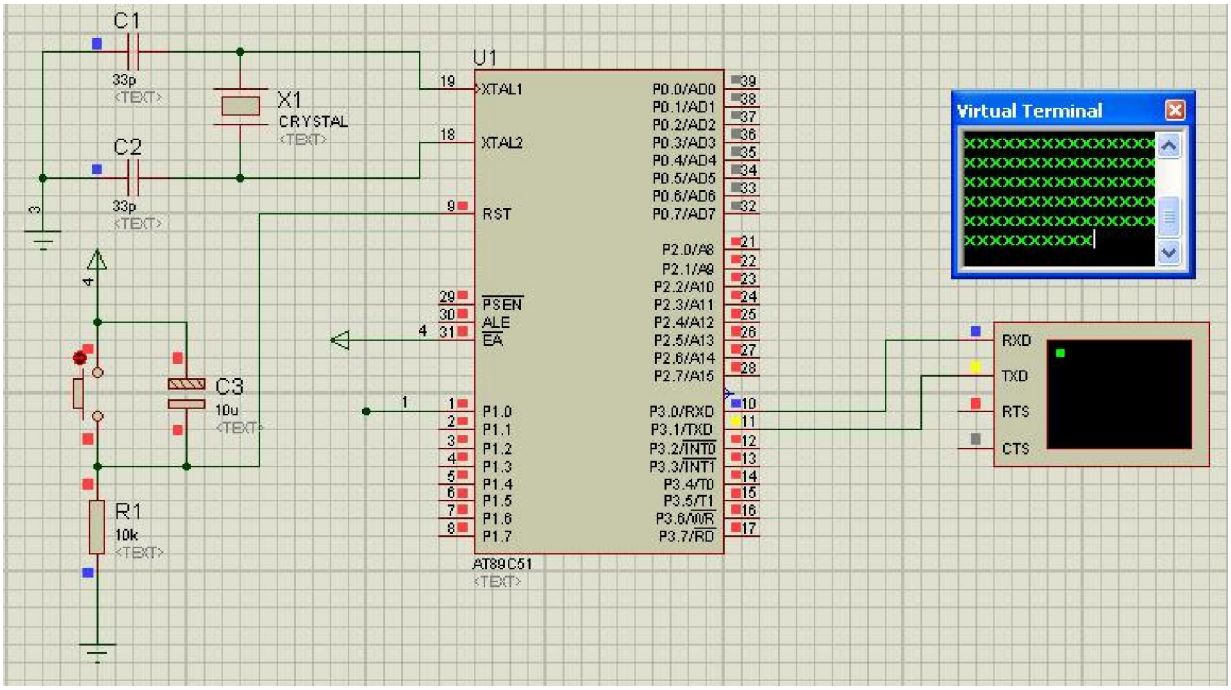
I. MỤC TIÊU

- Hiểu các chế độ làm việc của cổng nối tiếp
- Hiểu công dụng của cổng nối tiếp
- Lập trình cho cổng nối tiếp làm việc ở các chế độ khác nhau

II. NỘI DUNG THÍ NGHIỆM

1. Lắp mạch thí nghiệm

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).



2. Viết chương trình điều khiển

```
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 exitSerial
Transmit:             ; Truyền ký tự
CLR TI
MOV A,R7
MOV SBUF,A        ; Truyền ký tự
INC R7            ; Qua ký tự kế
CJNE R7,#'Z'+1,exitSerial  ; Nếu chưa truyền'Z' thì
            ; tiếp tục truyền, ngược lại thì
MOV R7,#'A'       ; bắt đầu truyền từ ký tự 'A'
exitSerial:
RETI
END
```

BAI 6: NGẮT

Mã bài: MĐ 25-06

Mục tiêu:

- Hiểu được tác dụng thực tế của một hệ thống sử dụng ngắt
- Biết được tổ chức ngắt và các thanh ghi ngắt
- Biết cách thiết kế một chương trình sử dụng ngắt

Nội dung chính:

1. Mở đầu

Mục tiêu:

- Hiểu được ngắt là gi

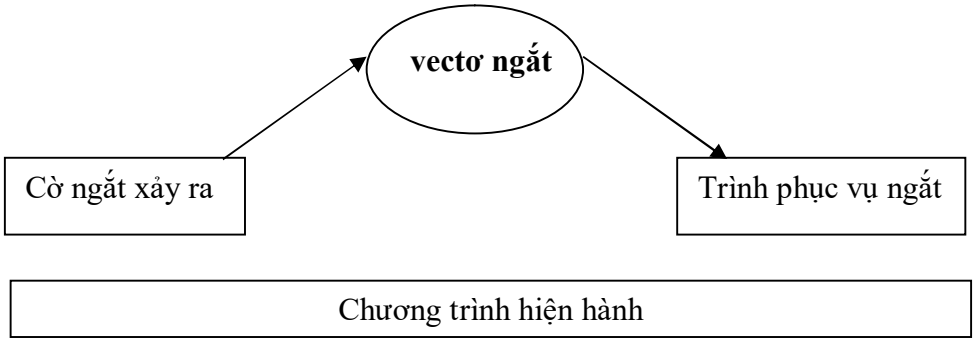
- Biết so sánh một chương trình sử dụng ngắt và một chương trình không sử dụng ngắt.

Ngắt là sự xuất hiện của một điều kiện, một sự kiện làm tạm dừng chương trình trong khi điều kiện này được phục vụ bởi một chương trình khác. Ngắt có một vai trò quan trọng trong thiết kế và thực hiện các ứng dụng của vi điều khiển. Chúng cho phép hệ thống đáp ứng không đồng bộ với một sự kiện và xử lý sự kiện trong khi một chương trình khác đang hoạt động. Một hệ thống được điều khiển bằng ngắt tạo một ảo giác thực hiện đồng thời nhiều công việc cùng một lúc. Dĩ nhiên, tại một thời điểm CPU không thể thực hiện nhiều hơn một lệnh nhưng nó có thể tạm dừng chương trình để thực hiện một chương trình khác và sau đó trở lại chương trình đầu tiên. Điểm khác là trong một hệ thống điều khiển bằng ngắt, các ngắt không xảy ra như là kết quả của một lệnh (như lệnh gọi chương trình con) mà là đáp ứng với một sự kiện xảy ra một cách không đồng bộ với chương trình chính có nghĩa là không biết trước chương trình chính sẽ bị ngắt lúc nào.

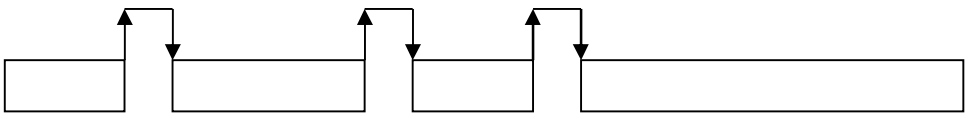
Chương trình xử lý ngắt được gọi là chương trình phục vụ ngắt (Interrupt service routine) viết tắt là ISR hay quản lý ngắt. ISR hoạt động để đáp ứng một ngắt và thường thực hiện một thao tác vào hoặc ra đến một thiết bị. Khi xảy ra một ngắt thì chương trình chính tạm thời dừng lại và rẽ nhánh đến ISR. ISR thực hiện

các thao tác cần thiết và kết thúc với lệnh trở về từ ngắt và chương trình chính lại tiếp tục từ nơi tạm dừng. Như vậy có thể nói chương trình chính hoạt động ở mức cơ sở và các ISR hoạt động ở mức ngắt cũng có dùng các thuật ngữ: "phía trước" (foreground) để chỉ mức cơ sở và "phía sau" (backgrround) để chỉ mức ngắt.

Một ví dụ điển hình về ngắt trong ứng dụng về lò vi sóng: Chương trình chính điều khiển phần tử tạo năng lượng vi sóng để nấu ăn, nhưng trong khi đang nấu hệ thống cần phải đáp ứng việc nhập bằng tay trên cửa lò ví dụ tăng hoặc giãm thời gian nấu. Khi người sử dụng thả nút nhấn, một ngắt được tạo ra (có thể là một tín hiệu chuyển từ mức cao xuống mức thấp) và chương trình chính bị dừng lại, chương trình ISR hoạt động đọc các mã của bàn phím và thay đổi quá trình nấu tương ứng sau đó chấm dứt bằng cách chuyển điều khiển về cho chương trình chính, chương trình chính lại tiếp tục từ nơi bị ngắt. Một điểm quan trọng trong ví dụ này là việc nhập bằng tay xảy ra một cách không đồng bộ có nghĩa là không biết trước hoặc không được điều khiển bằng phần mềm đang chạy trong hệ thống. Đó chính là đặc điểm của ngắt.



Cách nhìn ngắn gọn của một chương trình thực thi không có ngắt



Cách nhìn ngắn gọn của một chương trình thực thi có ngắt

↑ Bắt đầu trình phục vụ ngắt ↓ Kết thúc trình phục vụ ngắt

Hình 32-06-1 Thực hiện chương trình

2. Tổ chức ngắt

Mục tiêu:

- Hiểu được cấu tạo các thanh ghi của ngắt

- Biết được chuỗi vòng hoạt động của các ngắt

- Biết được địa chỉ các thanh ghi ngắt

- Ứng dụng các thanh ghi để viết chương trình điều khiển

2.1. Thanh ghi cho phép và không cho phép ngắt

Mỗi nguồn ngắt được cho phép hoặc không cho phép thông qua thanh ghi chức năng đặc biệt có các bit được địa chỉ hóa IE (Interrupt Enable) tại địa chỉ 0A8H.

Bit	**Symbol**	**Address**	**Mô tả** **(1: cho phép, 0: không cho phép)**
IE.7	EA	AFH	Cho phép, không cho phép toàn cục
IE.6	-	AEH	Không xác định
IE.5	ET2	ADH	Cho phép ngắt do timer 2(có ở 8952)
IE.4	ES	ACH	Cho phép ngắt nối tiếp
IE.3	ET1	ABH	Cho phép ngắt do timer1
IE.2	EX1	AAH	Cho phép ngắt do bên ngoài (INT 1)
IE.1	ET0	A9H	Cho phép ngắt do timer0
IE.0	EX0	A8H	Cho phép ngắt do bên ngoài (INT 0)

2.2. Thanh ghi ưu tiên ngắt

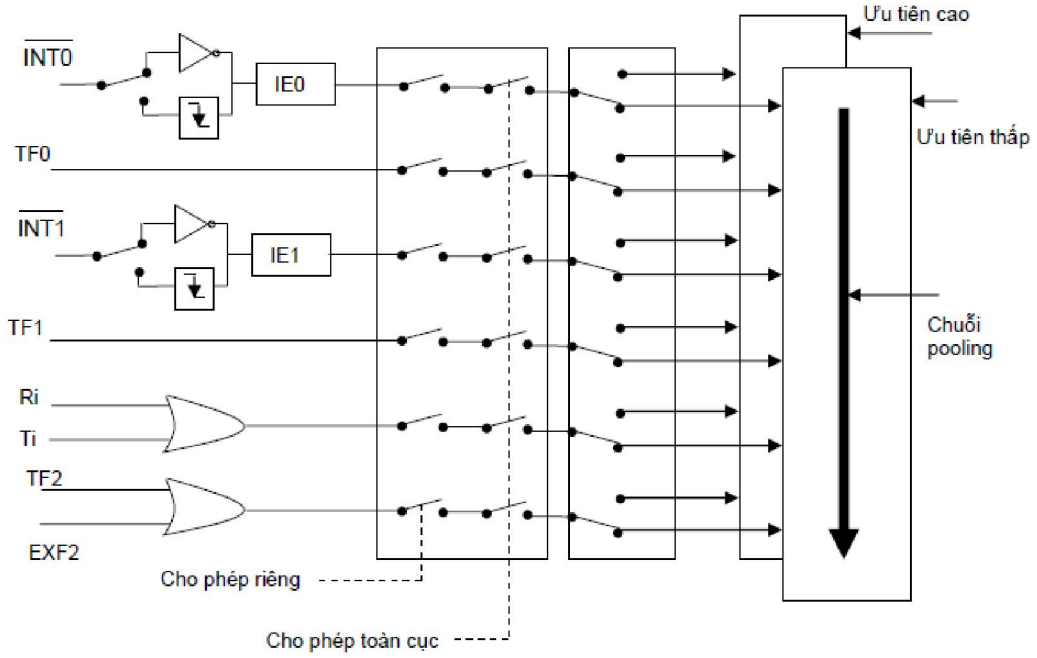
Khi có nhiều ngắt được cho phép, một trong số các ngắt được chọn ưu tiên cao hơn. Nếu một ngắt được chọn ưu tiên nó sẽ làm cho ngắt khác đang thực thi sẽ tạm dừng cho đến khi trình phục vụ ngắt ưu tiên kết thúc . Trong một tình huống khác, khi hai ngắt xảy ra cùng lúc, ngắt nào có mức ưu tiên cao hơn sẽ làm trước. Ở vi điều khiển 89xxx, một chương trình có sử dụng ngắt được chia làm 3 mức: mức nền (thường cho chương trình chính), mức ngắt và ưu tiên ngắt và chỉ có một ngắt được ưu tiên cao nhất nếu có nhiều ngắt được cho phép. Mỗi một nguyên nhân ngắt riêng

rẽ được lập trình để có một trong hai mức ưu tiên thông qua thanh ghi chức năng đặt biệt IP (interrupt priority).

Bit	Symbol	Bit Address	Description (1: ưu tiên, 0: không ưu tiên)
IP.7	-	-	Không sử dụng
IP.6	-	-	Không sử dụng
IP.5	ET2	ADH	Ưu tiên do ngắt do bộ định thời 2
IP.4	ES	ACH	Ưu tiên do ngắt do port nối tiếp
IP.3	ET1	ABH	Ưu tiên do ngắt do bộ định thời 1
IP.2	EX1	AAH	Ưu tiên do ngắt do bên ngoài (INT 1)
IP.1	ET0	A9H	Ưu tiên do ngắt do bộ định thời 0
IP.0	EX0	A8H	Ưu tiên do ngắt do bên ngoài (INT 0)

2.3. Chuỗi Pooling

Nếu như có nhiều ngắt cùng xuất hiện đồng thời, chuỗi vòng sẽ xác định ngắt nào thực hiện trước. Chuỗi vòng này 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.



Hình 32-06-2 Cấu trúc ngắt 8051

2.4. Vector ngắt

Khi có sự kiện ngắt (Cờ ngắt) xuất hiện, chương trình ngắt bắt đầu một địa chỉ cố định – gọi là vectơ ngắt.

Interrupt	Flag	Interrupt Vector
System Reset	RST	0000h
External 0	IE0	0003H
Timer 0	TF0	000BH
External 1	IE1	0013H
Timer 1	TF1	001BH
Serial Port	RI, TI	0023H
Timer2	TF2 & EXF2 (vđk 8952)	002BH

3. Thiết kế ngắt

Mục tiêu:Viết chương trình sử dụng ngắt

Dựa vào kích thước của chương trình, ta có 2 mẫu thiết kế như sau:

Mẫu thiết *kế trình phục vụ ngắt có kích thước nhỏ*

```
Org 0000h
Ljmp main  ; Nhảy qua khỏi đoạn các vectơ ngắt
Org 00xxh
ISR_xx:      ;Trình phục vụ ngắt
…
Reti
Org 0030h
 Main:
…
End
```

Mẫu thiết *kế trình phục vụ ngắt có kích thước lớn*

```
Org 0000h
Ljmp main  ; Nhảy qua khỏi đoạn các vectơ ngắt
Org 00xxh
Ljmp ISR_xx        ;vùng khai báo các vecto ngắt và
;dùng l ệnh nhảy đến trình phục vụ ngắt
Org 0030h
 Main:
…
ISR_xx:              ; Các trình phục vụ ngắt
…
Reti
End
```

4. Ngắt Timer

Mục tiêu:

- Hiểu được hoạt động của ngắt Timer

- Viết chương trình điều khiển sử dụng các ngắt Timer

Các ngắt do các bộ Timer xảy ra do sự kiện tràn ở các Timer, khi đó các cờ tràn TF0 hoặc TF1 sẽ được đặt bằng 1. Khi ISR được đáp ứng, các cờ TF0 hoặc TF1 sẽ tự động được xóa bởi phần mềm.

5. Ngắt ngoài

Mục tiêu:

- Hiểu được hoạt động của ngắt ngoài

- Viết chương trình điều khiển sử dụng các ngắt ngoài

Các ngắt ngoài xảy ra khi có một mức thấp hoặc cạnh xuống trên chân /INT0 hoặc /INT1 của bộ vi điều khiển. Các cờ tạo ra các ngắt này là bít IE.0 và IE.1 trong thanh ghi TCON, cờ tạo ra ngắt bị xóa bởi phần cứng khi CPU trỏ đến ISR nếu ngắt là loại tác động cạnh, còn đối với ngắt tác động bằng mức thì nguồn tạo ngắt bên ngoài sẻ điều khiển mức của cờ ngắt.

Việc chọn lựa ngắt tác động mức thấp hoặc tác động cạnh âm được được lập trình thông qua bít IT0 và IT1 trong thanh ghi TCON. VD: Nếu IT1 = 0 thì ngắt 1 ngoài được kích bởi mức thấp tại chân INT1 và nếu IT1 = 1 thì ngắt này được kích bằng cạnh âm. Trong chế độ này nếu các mẫu tại chân INT1 ở mức cao trong một chu kỳ và ở mức thấp trong các chu kỳ kế tiếp thì cờ IE1 trong TCON được set và sau đó cờ này sẻ yêu cầu một ngắt. Vì các chân ngắt ngoài được lấy mẫu mỗi chu kỳ máy một lần nên ngỏ vào này phải được duy trì ít nhất trong 12 chu kỳ dao động để bảo đãm việc lấy mẫu là thích hợp. Nếu là loại tác động cạnh thì nguồn ngoài phải giữ ở mức cao ít nhất một chu kỳ và ở mức thấp ít nhất một chu kỳ hoặc hơn để bảo đảm nhận ra được sự chuyển mức. IE0 và IE1 được xóa tự động khi CPU trỏ đến ngắt. Nếu ngắt ngoài là loại tác động mức thì nguồn ngoài phải duy trì mức tác động cho đến khi ngắt yêu cầu thực sự được tạo ra. Sau đó phải trở về mức không tác động trước khi ISR hoàn tất hoặc trước khi một ngắt khác được tạo ra. Thông thường một

thao tác trong ISR làm cho nguồn tạo ngắt trả tín hiệu ngắt trở về trạng thái không tác động.

6. Ngắt cổng nối tiếp

Mục tiêu:

- Hiểu được hoạt động của ngắt do cổng nối tiếp

- Viết chương trình điều khiển sử dụng ngắt do cổng nối tiếp

Các ngắt của cổng nối tiếp xảy ra khi một trong hai cờ ngắt phát TI hoặc cờ ngắt thu RI được set. Ngắt phát xuất hiện khi quá trình phát của ký tự trước đó được viết vào SBUF hoàn tất, một ngắt thu xuất hiện khi một ký tự đã được nhận đầy đủ và đang chờ đọc trong SBUF.

Các ngắt cổng nối tiếp khác với các ngắt của timer, cờ tạo ra ngắt cổng nối tiếp không được xóa bằng phần cứng khi CPU trỏ đến véc tơ ngắt lý do là có hai nguồn tạo ra một ngắt cổng nối tiếp đó là TI hoặc RI. Nguồn tạo ra ngắt phải được xác định trong ISR và cờ ngắt được xóa bằng phần mềm. Trở lại với các ngắt timer, cờ ngắt được xóa bằng phần cứng khi CPU trỏ đến ISR

THỰC HÀNH VỚI NGẮT

I. MỤC TIÊU

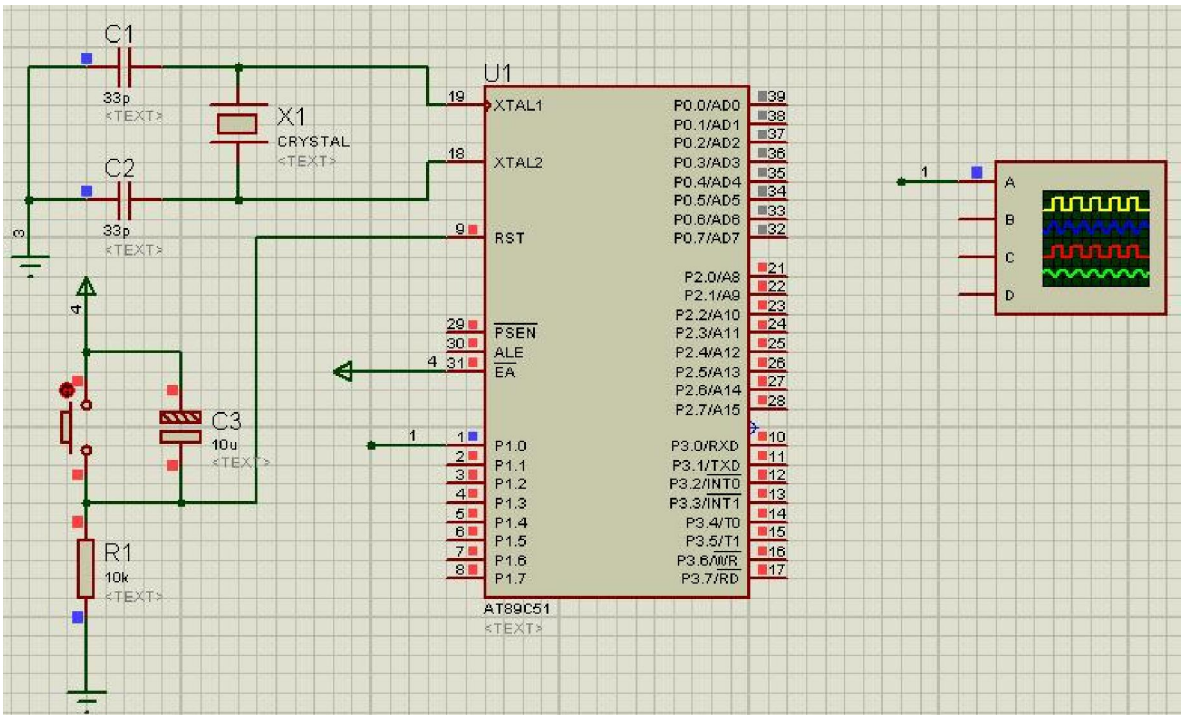
- Hiểu rõ hơn về tập lệnh của vi điều khiển MCS-51.
- Hiểu được hoạt động của ngắt (Interrupt) ở các chế độ khác nhau.
- Hiểu được phương pháp lập trình và điều khiển có sử dụng các ngắt.

II. NỘI DUNG THÍ NGHIỆM

1. Nối mạch thí nghiệm

Ngắt do Timer

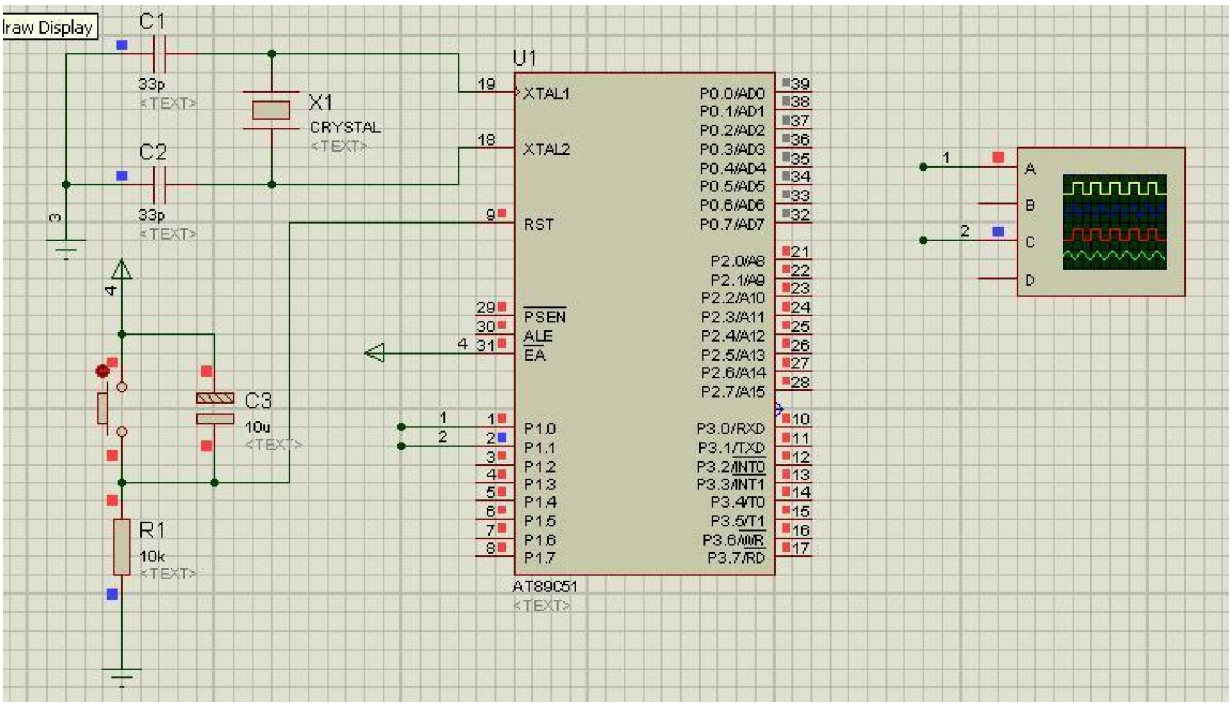
Chương trình 1 : Viết chương trình tạo sóng vuông tần số f = 5 KHz tại P1.0 dùng ngắt timer 1 (giả sử tần số thạch anh là 12 MHz).



2. Viết chương trình điều khiển

```
ORG 0000h
LJMP main
ORG 001Bh
CPL P1.0     ; đảo bit
RETI         ; trở về chương trình chính từ ISR
Main:
MOV TMOD,#20h
MOV IE,#88h   ; Có thể thay thế bằng 2 lệnh sau:
          ; SETB EA
          ; SETB ET1
MOV TH1,#(-100)
MOV TL1,#(-100)
SETB TR1
SJMP $
END
```

Chương trình 2 : Viết chương trình tạo xung vuông tần số f =10KHz tại P1.0 dùng ngắt timer 0 và xung vuông tần số f = 1 KHz tại P1.1 dùng ngắt timer 1.



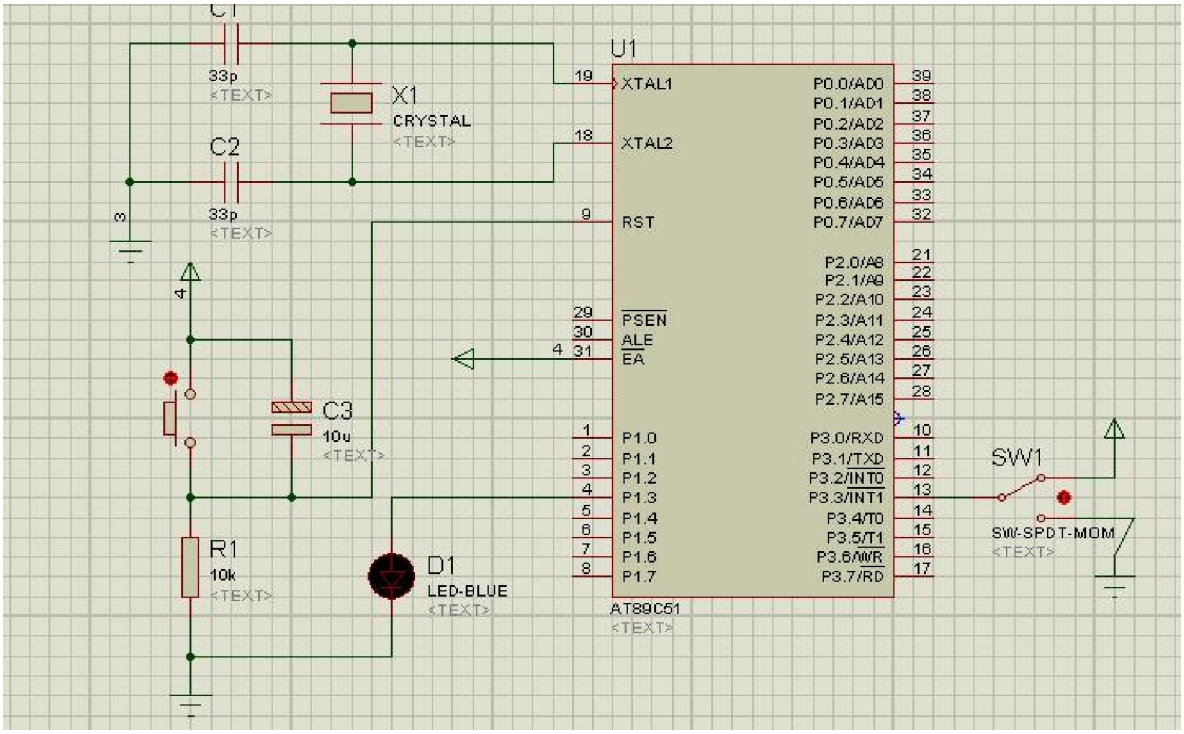

```
ORG 0000h
LJMP main
ORG 000Bh
CPL P1.0
RETI
ORG 001Bh
MOV TH1,#HIGH(-500)    ; 2 byte
MOV TL1,#LOW(-500)     ; 2 byte
CPL P1.1               ; 2 byte
RETI                   ; 1 byte
Main:
MOV TMOD,#12h
MOV IE,#8Ah
SETB TR0
SETB TR1
MOV TH1,#HIGH(-500)
MOV TL1,#LOW(-500)
MOV TH0,#(-50)
MOV TL0,#(-50)
```

SJMP $

END

Ngắt ngoài

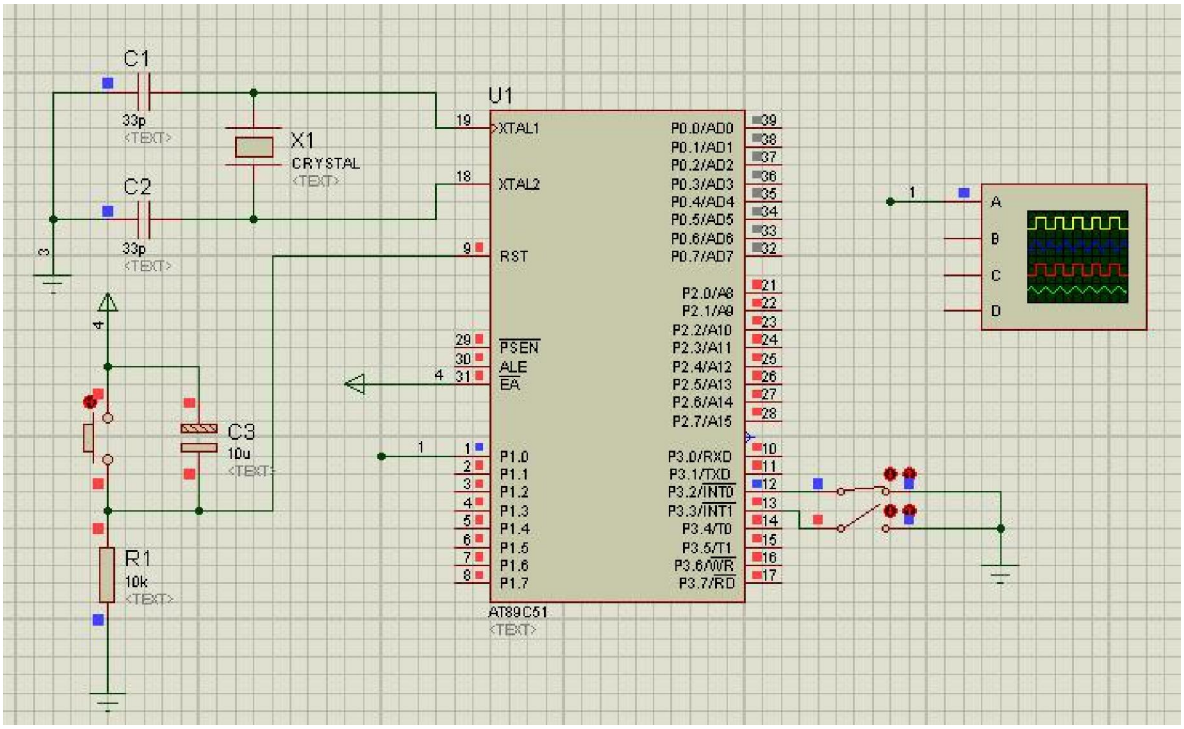
Chương trình 3: Giả sử chân INT1 được nối đến công tắc bình thường ở mức cao. Mỗi khi nó xuống thấp phải bật một đèn LED. Đèn LED được nối đến chân P1.3 và bình thường ở chế độ tắt. Khi nó được bật lên nó phải sáng vài phần trăm giây. Chừng nào công tắc được ấn xuống thấp đèn LED phải sáng liên tục.




```
    ORG     0000H
   LJMP     MAIN        ; Nhảy đến bảng véc tơ ngắt
; - - Chương trình con ISR cho ngắt cứng INT1 để bật đèn LED.
   ORG      0013H       ; Trình phục vụ ngắt ISR cho INT1
   SETB     P1.3        ; Bật đèn LED
   MOV      R3, # 255   ;
BACK:  DJNZ     R3, BACK    ; Giữ đèn LED sáng một lúc
   CLR      P1.3        ; Tắt đèn LED
   RETI         ; Trở về từ ISR
; - - Bắt đầu chương trình chính Main.
   ORG      30H
```

```
MAIN:   MOV    IE, #10000100B   ; Cho phép ngắt dài
  SJMP   HERE      ; Chờ ở đây cho đến khi được ngắt
 END
```

Chương trình 4 Viết chương trình sao cho mỗi khi có mức logic 0 xuất hiện tại P3.2 (ngắt ngoài 0) thì tạo xung 1 KHz tại P1.0. Quá trình tạo xung chỉ dừng khi có mức logic 0 xuất hiện tại P3.3 (ngắt ngoài 1).




```
ORG 0000h
LJMP main

ORG 0003h        ; Địa chỉ ISR ngắt ngoài 0
SETB TR1         ; Timer 1 chạy
RETI

ORG 0013h        ; Địa chỉ ISR của ngắt ngoài 1
CLR TR1          ; Cấm timer 1
RETI

ORG 001Bh        ; Địa chỉ ISR timer 1
```

MOV TH1,#HIGH(-500); Chế độ 16 bit nên mỗi lần tràn

MOV TL1,#LOW(-500); phải nạp lại giá trị

CPL P1.0 ; Đảo bit P1.0 để tạo xung

RETI

Main:

MOV TMOD,#10h

MOV TH1,#HIGH(-500)

MOV TL1,#LOW(-500)

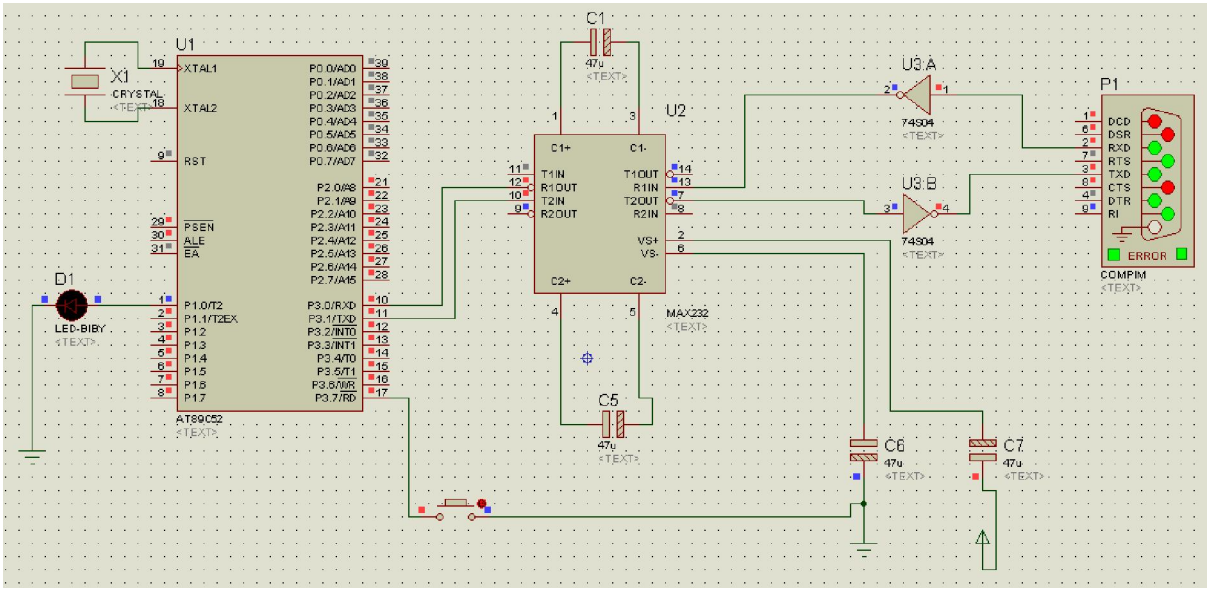
MOV IE,#8Dh ; Cho phép ngắt tại ngắt ngoài 0, 1 và

SJMP $; timer 1

END

Ngắt cổng nối tiếp

Chương trình 6 Viết chương trình đếm sản phẩm bằng vi điều khiển 8051 và truyền dữ liệu qua cổng nối tiếp.



Chương trình:

org 0000h

mov tmod,#20h

mov scon,#50h

mov th1,#0fdh

setb tr1

```
        clr     tf1
        clr     ti
        clr     ri
        mov     r7,#0
lap:
        call    delay
        cpl     p1.0
        jb      p3.7,lap
        jnb     p3.7,$
        inc     r7
        call    truyensanpham
        sjmp    lap
truyensanpham:
        mov     a,r7
        mov     b,#10
        div     ab
        mov     30h,b
        mov     b,#10
        div     ab
        mov     31h,b
        mov     32h,a
        mov     dptr,#ma
        movc    a,@a+dptr
        clr     ti
        mov     sbuf,a
        jnb     ti,$
        mov     a,31h
        movc    a,@a+dptr
        clr     ti
        mov     sbuf,a
```

```
        jnb     ti,$
        mov     a,30h
        movc    a,@a+dptr
        clr     ti
        mov     sbuf,a
        jnb     ti,$
        clr     ti
        mov     sbuf,#0dh
        jnb     ti,$
        ret
delay:
        mov     70h,#2
dl1:    mov     71h,#255
dl2:    mov     72h,#255
        djnz    72h,$
        djnz    71h,dl2
        djnz    70h,dl1
ret
ma: db '0123456789',00h
end
```

ĐIỀU KHIỂN LCD

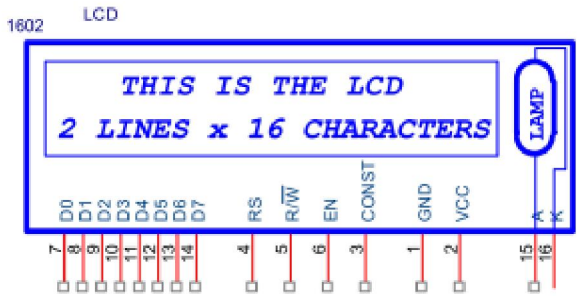
I. MỤC TIÊU

- Hiểu hơn về tập lệnh của vi điều khiển MCS-51.
- Biết cách viết các chương trình điều khiển LCD.
- Hiểu được sơ đồ và nguyên lý hoạt động của khối LCD trên mô hình thí nghiệm.
- Hiểu được nguyên lý và kỹ thuật điều khiển để hiển thị các thông tin trên LCD.
- Biết cách viết các chương trình ứng dụng để hiển thị các dạng thông tin khác nhau trên LCD tùy theo nhu cầu sử dụng.

II. NỘI DUNG THÍ NGHIỆM

1. Nối mạch thí nghiệm

Sơ đồ của LCD1602A:



CONST (contrast): chỉnh độ tương phản (độ sáng của hình ảnh trên LCD).

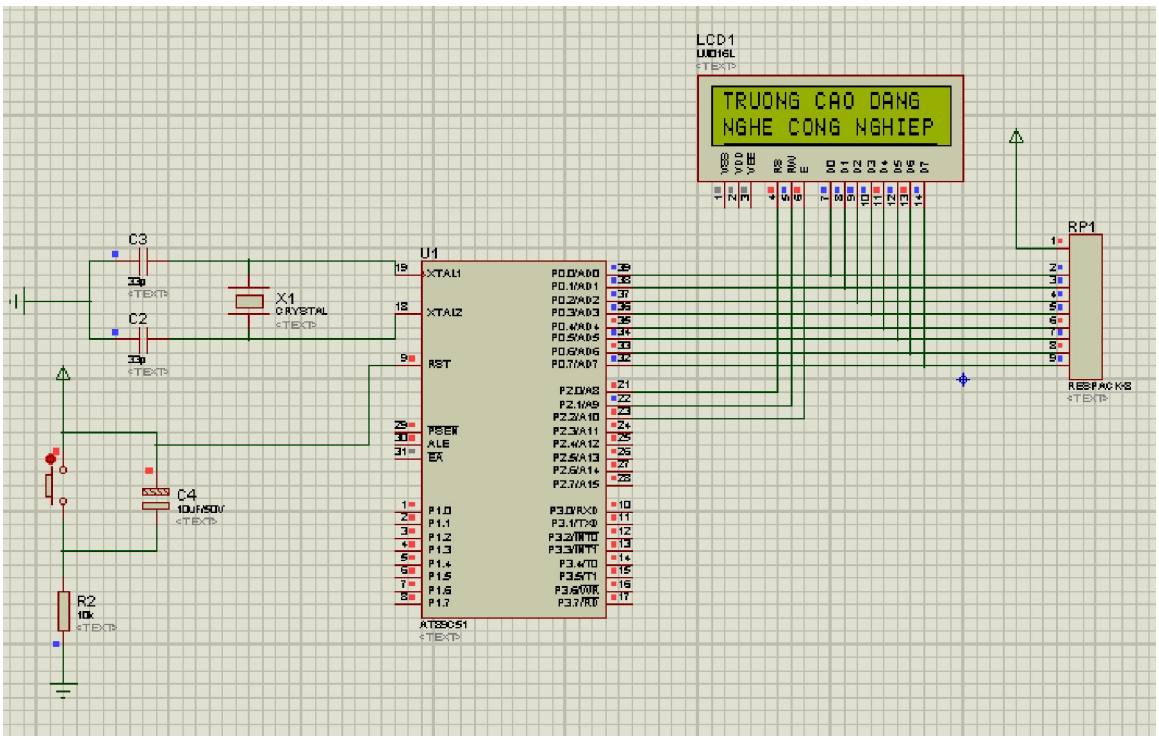
- EN (Enable): cho phép đọc/ghi dữ liệu. Trong chế độ đọc, EN tác động bằng xung dương (cạnh lên) và trong chế độ ghi, EN tác động bằng xung âm (cạnh xuống).
- RS (register selection): chọn thanh ghi lệnh (RS = 0) hoặc thanh ghi dữ liệu (RS = 1)
- R/W: đọc (R/W = 1) hay ghi (R/W = 0)
- D7 – D4: bus dữ liệu (chế độ 8 bit: 4 bit cao, chế độ 4 bit: dùng cho truyền 4 bit cao và 4 bit thấp). Ngoài ra, bit D7 còn dùng làm ngõ ra cho cờ Busy.
- D3 – D0: 4 bit thấp trong chế độ 8 bit hay bỏ trống trong chế độ 4 bit.
- A, K: anode và cathode đèn nền của LCD.

Các thành phần chức năng của LCD1602A:

- Cờ Busy (BF – Busy flag): Nếu BF = 1, LCD đang trong quá trình thực thi một lệnh. Khi đó, các lệnh gởi tiếp theo sẽ bị bỏ qua. BF được đọc tại chân D7 khi RS = 0 và R/W = 1. Do đó, trước khi thực hiện một lệnh, cần kiểm tra BF trước, nếu BF = 0 thì mới gởi lệnh.
- DDRAM (Display Data RAM): chứa các ký tự sẽ hiển thị trên LCD, tối đa là 80x8 bit (80 ký tự). Khi hiển thị ở chế độ 1 dòng, địa chỉ của DDRAM có phạm vi từ 00h ÷ 4Fh còn khi ở chế độ 2 dòng, địa chỉ DDRAM từ 00h ÷
27h cho dòng 1 và 40h ÷ 67h cho dòng 2.
- Bộ đếm địa chỉ (AC - Address Counter): dùng để lưu địa chỉ hiện hành của DDRAM và CGRAM, có thể thực hiện đọc AC khi RS = 0 và R/W = 1.

- CGROM (Character Genaration ROM): chứa các mô hình ký tự sẽ hiển thị trên LCD, bao gồm 192 ký tự 5x7 theo bảng mã ASCII (nghĩa là khi DDRAM chứa giá trị 41h tương ứng với mã ASCII của ký tự ‘A’ thì trên LCD sẽ hiện ‘A’), trong đó chỉ có các mã từ 00h – 0Fh sẽ không lấy theo mã ASCII mà lấy theo các ký tự đã định nghĩa trong CGRAM.

Chương trình 1 Cho sơ đồ kết nối LCD 1602A với AT89C51 như hình vẽ. Viết chương trình hiển thị chuỗi “TRUONG CAO DANG” trên dòng 1 hiển thị chuỗi “NGHE CONG NGHIEP” trên dòng 2.



2. Viết chương trình điều khiển

```
DATA_PORT      EQU  P0
RS     BIT    P2.0
RW_ BIT    P2.1
E      BIT    P2.2
DB7    BIT    P0.7
ORG  00h
       CALL       KHOITAO
LOOP:
       MOV        DPTR,#CHUOI1
```

```
        CALL    WRITE_STRING
        MOV     A,#0C0H
        CALL    GHI_LENH
        MOV DPTR,#CHUOI2
        CALL    WRITE_STRING

        JMP   $
KHOITAO:
        MOV     A,#038H
        CALL    GHI_LENH
        MOV     A,#0dH
        CALL    GHI_LENH
        MOV A,#01
        CALL    GHI_LENH
RET
WRITE_STRING:
        MOV     R0,#0
WR_LOOP:
        MOV     A,R0
        MOVC    A,@A+DPTR
        CJNE A,#00H,NEXT
        JMP   EXIT
NEXT:
        CALL    WRITE
        INC   R0
        JMP   WR_LOOP
EXIT:
RET
WRITE:
        CALL    READY
```

```
        MOV     DATA_PORT,A
        SETB    RS
        CLR     RW_
        CLR     E
        SETB    E
RET
Ghi_lenh:
        CALL    READY
        mov   DATA_PORT,A
        clr     rw_
        clr     rs
        clr     e
        setb    e
ret
READY:
        MOV     DATA_PORT,#0FFH
         CLR  RS
         SETB    RW_
recheck:
        CLR E
        SETB E
        JB DB7,recheck
RET
CHUOI1:
DB    'TRUONG CAO DANG',00H
CHUOI2:
DB    'NGHE CONG NGHIEP ',00H
END
```

Yêu cầu về đánh giá kết quả học tập

* Về kiến thức:

- Hiểu cấu tạo vi điều khiển 8051.

- Ứng dụng bộ định thời để tạo thời gian trễ.

- Biết công dụng và cách sử dụng ngắt.

- Truyền dữ liệu kiểu nối tiếp.

- Giải thích sơ đồ mạch.

- Viết chương trình ứng dụng theo yêu cầu.

*Về kỹ năng:

- Lắp ráp được mạch điều khiển theo sơ đồ có sẵn.

- Phân tích được hiện tượng và phán đoán nguyên nhân gây hư hỏng trong mạch điều khiển bằng các thiết bị đo.

- Sửa được chương trình điều khiển.

* Về thái độ

- Nghiêm túc, tích cực, chủ động trong học tập.

- Chấp hành nghiêm chỉnh nội quy của xưởng và phòng thực hành vi điều khiển.

TÀI LIỆU THAM KHẢO

[1]- Tống Văn On, Hoàng Đức Hải, *Họ vi điều khiển 8051*, NXB Lao đông xã hội, Hà Nội 2005.

[2]- Ngô Diên Tập, *Lập trình bằng hợp ngữ*, NXB Khoa học kỹ thuật, Hà Nội 1998.

[3]- Ngô Diên Tập, *Vi xử lý trong đo lường và điều khiển*, NXB Khoa học kỹ thuật, Hà Nội 1999.

[4]- Đỗ Xuân Thụ, Hồ Khánh Lâm, *Kỹ thuật vi xử lý và máy tính*, NXB Giáo dục, Hà Nội 2000.

[5]- Nguyễn Tăng Cường, Phan Quốc Thắng, *Cấu trúc và lập trình vi điều khiển*, NXB Khoa học kỹ thuật, Hà Nội 2004.

[6]- Ngô Diên Tập, Vũ Trung Kiên, Phạm Xuân Khánh, Kiều Xuân Thực, *Giáo trình vi xử lý và cấu trúc máy tính*, NXB Giáo dục, Hà Nội 2007.