BÀI MỞ ĐẦU

TỔNG QUAN CẤU TRÚC PHẦN CỨNG

MÃ BÀI: MH30-01

Mục tiêu:

- Mô tả được cấu trúc bên trong của vi điều khiển
- Giải thích được chức năng từng chân của vi điều khiển

Nội dung chính:

1. Giới thiệu họ vi điều khiển 89C51

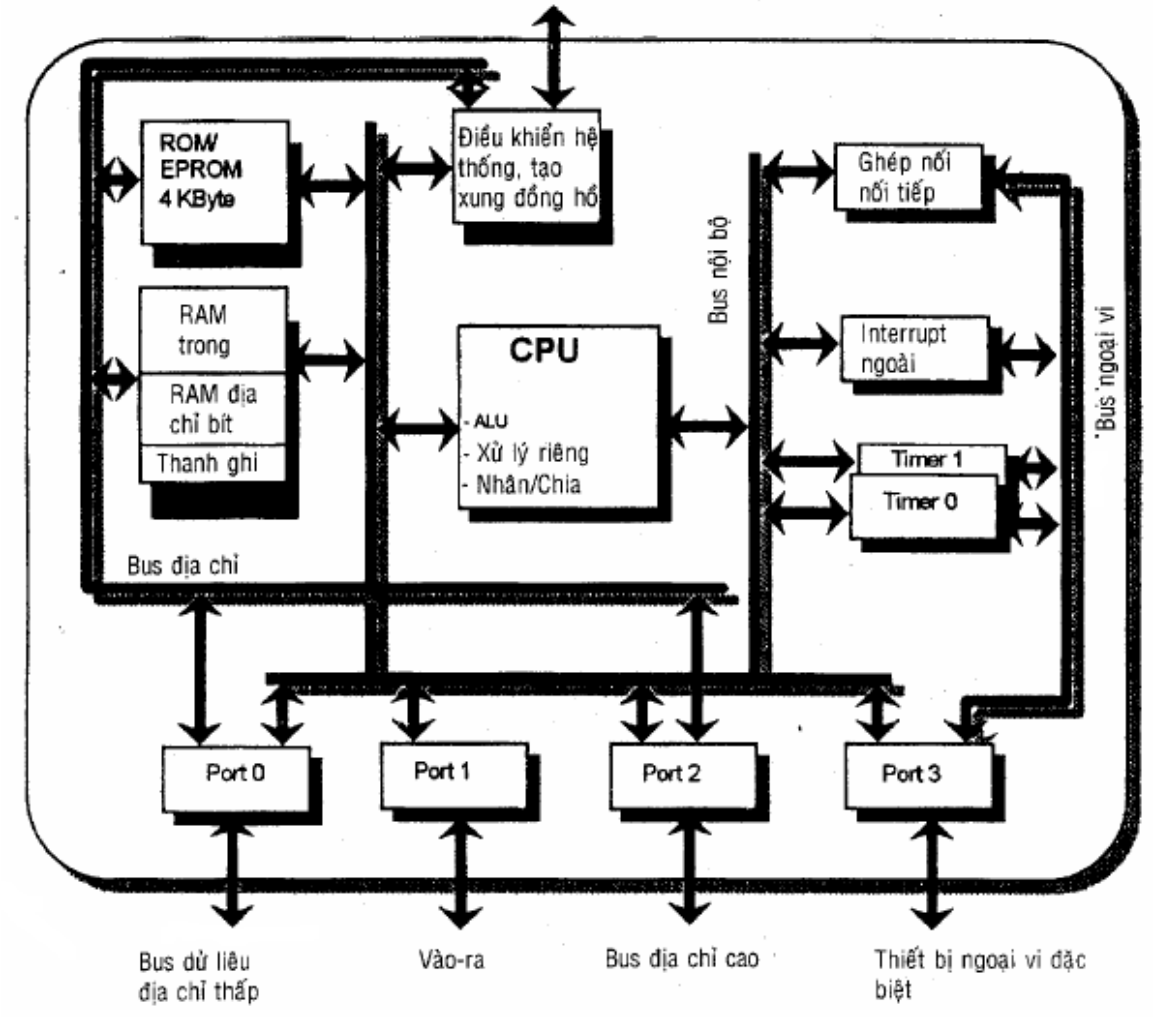
Mục tiêu:

-Mô tả được cấu trúc bên trong của vi điều khiển.

MCS-51 là họ vi điều khiển của Intel. Các nhà sản xuất IC khác như Siemens, Advanced Micro Device, Fujitsu và Philips được cấp phép là các nhà cung cấp thứ hai cho các vi mạch họ MCS-51.

Vi mạch tổng quát 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 được tóm tắt như sau:

- 4 Kbyte ROM nội
- 128 byte RAM nội
- 4 port I/O 8 bít
- 2 bộ định thời 16 bít (timer)
- Cổng giao tiếp nối tiếp
- Không gian chương trình ngoài 64 K
- Không gian nhớ dữ liệu ngoài 64 K
- Có khả năng xử lý bít
- 210 địa chỉ bít
- Nhân/chia trong 4 µS



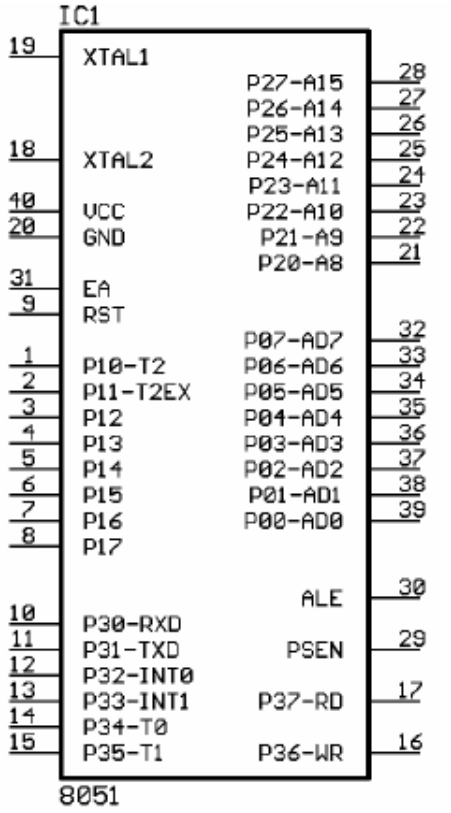
Hình 1.1 Sơ đồ khối 8051

Các thành viên khác của họ MCS-51 có dung lượng ROM (EPROM), RAM trên chip khác nhau hoặc có thêm bộ định thời thứ ba (bảng 2.2).

BẢNG 1.1 Cac chip họ MCS-51

Chip	ROM nội (Kbyte)	RAM nội (byte)	Số timer
8031	0	128	2
8051	4 ROM	128	2
8751	4 EPROM	128	2
8032	0	256	3
8052	8 ROM	256	3
8752	8 EPROM	256	3

Thuật ngữ „8051“ được dùng để chỉ chung có các chip họ MCS-51 với các chip cải tiến từ 8051 sẻ được đề cập đến một cách rõ ràng khi cần thiết, cấu trúc cơ bản được trình bày trong sơ đồ khối hình 1.1



Hình 1.2 Sơ đồ chân của 8051

2. Sơ lược về các chân của 89C51

Mục tiêu:

- Giải thích được chức năng từng chân của vi điều khiển

Chức năng từng chân của 8051 được mô tả tóm tắt như sau:

Như trong hình 2.2, 32 trong số 40 chân của 8051 có công dụng xuất/nhập. Tuy nhiên, 24 trong 32 chân này có hai chức năng (26/32 đối với 8032 và 8052). Mỗi một chân ngoài chức năng xuất nhập còn có thể hoạt động như một đường điều khiển, đường dữ liệu hoặc đường địa chỉ. 32 chân nói trên hình thành 4 port 8 bít, với các thiết kế yêu cầu tối thiểu các thành phần bên ngoài có thể sử dụng các port này làm nhiệm vụ xuất/nhập. 8 đường cho mỗi port có thể được xử lý như một đơn vị giao tiếp với các thiết bị song song như máy in, bộ biến đổi D-A.....hoặc mỗi đường có thể hoạt động độc lập giao tiếp với một thiết bị đơn bít như chuyển mạch, LED, BJT, FET, cuộn dây, động cơ, loa...

2.1 Port 0

Port gồm các chân từ 32 đến 39 đối với 8051 có 2 công dụng. Trong các thiết kế cần tối thiểu hóa thành phần, port 0 được sử dụng làm nhiệm vụ xuất/nhập. Trong các thiết kế lớn hơn có bộ nhớ ngoài, port 0 trở thành bus địa chỉ/dữ liệu đa hợp (byte thấp địa chỉ).

2.2 Port 1

Port 1 chỉ có một công dụng là xuất/nhập (các chân 1…8 trên 8051). Các chân của

port 1 được ký hiệu là P1.0, P1.1...P1.7 và được dùng để giao tiếp với thiết bị bên ngoài khi có yêu cầu. Không có chức năng nào khác nữa gán cho các chân của port 1, nghĩa là chúng chỉ được sử dụng để giao tiếp với các thiết bị ngoại vi (ngoại lệ với 8032, 8052 có thể dùng P1.0 và P1.1 làm ngõ vào cho mạch định thời thứ ba).

2.3 Port 2

Port 2 gồm các chân từ 21…28 trên 8051 có hai công dụng hoặc làm nhiệm vụ xuất/nhập hoặc là byte địa chỉ cao của bus địa chỉ 16 bít cho các thiết kế có bộ nhớ chương trình ngoài hoặc các thiết kế có nhiều hơn 256 byte bộ nhớ dữ liệu ngoài.

2.4 Port 3

Port 3 gồm các chân từ 10…17 trên 8051 có hai công dụng, ngoài chức năng xuất/nhập các chân của port 3 còn có chức năng riêng như trình bày trong bảng 2.2

BẢNG 1.2 CHỨC NĂNG CÁC CHÂN PORT 3 VÀ PORT 1

Bít	Tên	Địa chỉ bít	Chức năng
P3.0	RxD	B0H	Nhận dữ liệu port nối tiếp
P3.1	TxD	B1H	Phát dữ liệu port nối tiếp
P3.2	INT0	B2H	Ngõ vào ngắt 0 ngoài
P3.3	INT1	B3H	Ngõ vào ngắt 1 ngoài
P3.4	T0	B4H	Ngõ vào timer 0
P3.5	T1	B5H	Ngõ vào timer 1
P3.6	WR	B6H	Ghi RAM ngoài
P3.7	RD	B7H	Đọc RAM ngoài
P1.0	T2	90H	Ngõ vào timer 2
P1.1	T2EX	91H	Nạp/nhận timer 2

2.5 Chân cho phép bộ nhớ chương trình $\overline{\text{PSEN}}$

8051 cung cấp 4 tín hiệu điều khiển bus. Tín hiệu cho phép bộ nhớ chương trình $\overline{\text{PSEN}}$ (program store enable) là tín hiệu xuất trên chân 29, đây là tín hiệu điều khiển cho phép truy xuất bộ nhớ chương trình ngoài, chân này thường nối với chân cho phép xuất. $\overline{\text{OE}}$ (output enable) của EPROM hoặc ROM để cho phép đọc các byte mã lệnh.

Tín hiệu $\overline{\text{PSEN}}$ ở logic 0 trong suốt thời gian tìm-nạp lênh, các mã nhị phân của chương trình được đọc từ EPROM qua bus dữ liệu và được chốt vào thanh ghi lệnh IR của 8051 để được giải mã.

Khi thực hiện một chương trình chứa trong ROM nội, $\overline{\text{PSEN}}$ được duy trì ở mức logic 1.

2.6 Chân cho phép chốt địa chỉ ALE

Đây là tín hiệu ra cho phép chốt địa chỉ ALE (address latch enable) để giải đa hợp bus địa chi/dữ liệu khi port 0 được dùng làm bus đa hợp địa chỉ/dữ liệu, chân ALE xuất tín hiệu để chốt địa chỉ (byte thấp của địa chỉ 16 bít) vào một thanh ghi ngoài trong suốt nửa đầu của chu kỳ bộ nhớ. Sau khi thực hiện xong các chân của port 0 sẽ xuất/nhập dữ liệu hợp lệ trong nửa thứ hai của chu kỳ bộ nhớ.

Tín hiệu ALE có tần số bằng 1/6 tần số của mạch dao động bên trong chip 8051 và
có thể được dùng làm xung đồng hồ cho phần còn lại của hệ thống. Nếu mạch dao động có tần số 12 MHz thì tần số ALE là 2 MHz. Ngoại lệ duy nhất là trong thời gian thực hiện lệnh MOVX, một xung ALE sẽ bị bỏ qua (hình 2.10). Chân ALE còn được dùng để nhận xung ngõ vào lập trình cho EPROM trên chip.

2.7 Chân truy xuất ROM ngoài $\overline{\text{EA}}$

Đây là ngõ vào (chân 31) có thể được nối với 5 V (logic 1) hoặc với GND (logic 0). Nếu chân này nối lên 5 V, 8051 và 8052 sẽ thực hiện chương trình chứa trong ROM nội (chương trình nhỏ hơn 4K/8K). Nếu chân này nối với GND và chân $\overline{\text{PSEN}}$ cũng ở logic 0 thì chương trình thực hiện được chứa ở ROM ngoài. Đối với 8031/8032 chân $\overline{\text{EA}}$ phải ở logic 0 vì chúng không có bộ nhớ chương trình trên chip. Nếu chân $\overline{\text{EA}}$ ở logic 0 đối với 8051/8052, ROM nội bên trong chip được vô hiệu hóa và chương trình cần thực hiện chứa ở ROM ngoài.

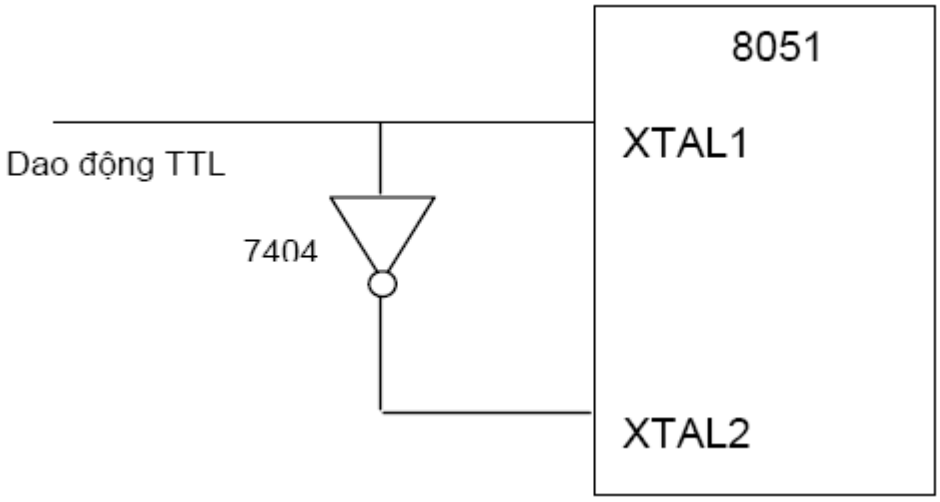
Các phiên bản EPROM của 8051 còn sử dụng chân $\overline{\text{EA}}$ làm chân nhận điện áp cấp điện 21 V (VPP) cho việc lập trình EPROM nội.

2.8 Chân RESET

Ngõ vào RST (chân 9) là ngõ vào xóa của 8051 dùng để thiết lập lại trạng thái ban
đầu cho hệ thống hay thường gọi là reset hệ thống. Khi ngõ vào này được treo ở logic 1 ít nhất 2 chu kỳ máy, các thanh ghi bên trong của 8051 được nạp các gía trị thích hợp cho việc khởi động lại hệ thống.

2.9 Các chân XTAL1 và XTAL2

Như trên hình 1.2, mạch dao động bên trong chip 8051 được ghép với thạch anh bên ngoài ở hai chân XTAL1 và XTAL2 (chân 18 và 19). Tần số danh định là 12 MHz cho hầu hết các chip của họ MCS-51, trong hình 1.3 cho thấy một nguồn xung clock TTL có thể được nối vào các chân XTAL1 và XTAL2.



Hình 1.3: 8051 với mạch dao động bên ngoài

3. Tổ chức bộ nhớ

Mục tiêu:

- Trình bày được tổ chức bộ nhớ của vi điều khiển.

Hầu hết các bộ vi xử lý đều có không gian nhớ chung cho dữ liệu và chương trình. Điều này cũng hợp lý vì các chương trình thường được lưu trên đĩa và được nạp vào RAM để thực hiện, vậy thì cả hai dữ liệu và chương trình đều lưu trú trên RAM.

Các chip vi điều khiển hiếm khi được xử dụng giống như các vi xử lý trong máy tính, thay vào đó chúng được dùng làm thành phần trung tâm trong các thiết kế hướng điều khiển. Trong đó bộ nhớ có dung lượng giới hạn không có ổ đĩa và hệ điều hành. Chương trình điều khiển phải thường trú trong ROM.

Do lý do trên, 8051 có không gian bộ nhớ riêng cho chương trình và dữ liệu. Như đã thấy trong bảng 2.1 cả 2 bộ nhớ chương trình và dữ liệu đều đặt bên trong chip, tuy nhiên có thể mở rộng bộ nhớ chương trình và dữ liệu bằng cách dùng thêm các chip nhớ bên ngoài với dung lượng tối đa là 64 K cho bộ nhớ chương trình và 64 K cho bộ nhớ dữ liệu.

Bộ nhớ nội trong chip bao gồm ROM (chỉ có trong 8051/8052) và RAM. RAM trên chíp bao gồm vùng RAM đa chức năng, vùng RAM địa chỉ bít, các dãy thanh ghi và các thanh ghi chức năng đặc biệt SFR (special funtion registers), hai đặc tính đáng lưu ý là:

a) Các thanh ghi và các port xuất/nhập được định địa chỉ theo kiểu ánh xạ bộ

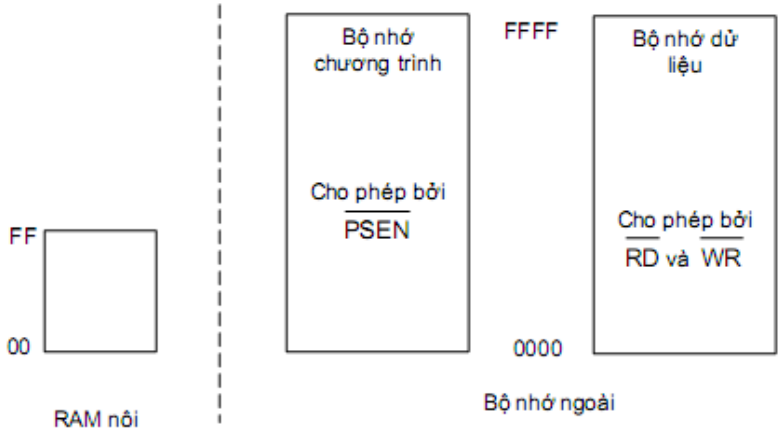
nhớ và được truy xuất như một vị trí nhớ trong bộ nhớ.

b) Vùng ngăn xếp thường trú trong RAM trên chíp thay vì trong RAM ngoài

như đối với các bộ vi xử lý.

Hình 1.4 tóm tắt các không gian nhớ của chíp 8031 không có ROM nội, không trình bày chi tiết về bộ nhớ dữ liệu trên chíp

Hình 1.5 trình bày chi tiết bộ nhớ dử liệu trên chip, không gian nhớ nội này được chia thành: Các dãy thanh ghi (00H..1FH) vùng RAM địa chỉ bít (20H..2FH), vùng RAM đa năng (30H..7FH) và các thanh ghi chức năng đặc biệt (80H..FFH)



Hình 1.4 Tổ chức bộ nhớ của 8051

3.1 Vung RAM đa năng

Vùng RAM đa năng có 80 byte đặt ở địa chỉ từ 30H đến 7FH, bên dưới vùng này từ địa chỉ 00H đến 2FH là vùng nhớ có thể được xử dụng tương tự (mặc dù vùng này có công dụng khác). Bất kỳ vị trí nhớ nào trong vùng RAM đa năng đều có thể được truy xuất tự do bằng cách xử dụng cách định địa chỉ trực tiếp hoặc gián tiếp. Ví dụ để đọc nội dung tại 5FH của RAM nội vào thanh ghi A có thể dùng lệnh sau :

MOV A, 5FH

Lệnh trên di chuyển một byte dử liệu bằng cách dùng kiểu định địa chỉ trực tiếp để xác định vị trí nguồn (nghĩa là địa chỉ 5FH). Đích của dử liệu được xác định rỏ ràng trong mã lệnh là thanh ghi A

Vùng RAM đa năng còn có thể được truy xuất bằng cách dùng kiểu định địa chỉ

gián tiếp qua các thanh ghi R0, R1. Ví dụ hai lệnh sau thực hiện cùng công việc như ở ví dụ trên.

MOV R0,#5FH

MOV A,@R0

Lệnh đầu tiên xử dụng kiểu định địa chỉ tức thời di chuyển giá trị 5FH vào thanh ghi R0, lệnh tiếp theo xử dụng cách định địa chỉ gián tiếp di chuyển dử liệu trỏ bởi R0 vào thanh ghi A

7F	Vùng RAM đa năng								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	-	D0	PSW
30									B8	-	-	-	BC	BB	BA	B9	B8	IP
2F	7F	7E	7D	7C	7B	7A	79	78										
2E	77	76	75	74	73	72	71	70	B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
2D	6F	6E	6D	6C	6B	6A	69	68										
2C	67	66	65	64	63	62	61	60	A8	AF	-	-	AC	AB	AA	A9	A8	IE
2B	5F	5E	5D	5C	5B	5A	59	58										
2A	57	56	55	54	53	52	51	50	A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
29	4F	4E	4D	4C	4B	4A	49	48										
28	47	46	45	44	43	42	41	40	99	Địa chỉ byte								SBUF
27	3F	3E	3D	3C	3B	3A	39	38	98	9F	9E	9D	9C	9B	9A	99	98	SCON
26	37	36	35	34	33	32	31	30										
25	2F	2E	2D	2C	2B	2A	29	28	90	97	96	95	94	93	92	91	90	P1
24	27	26	25	24	23	22	21	20										
23	1F	1E	1D	1C	1B	1A	19	18	8D	Địa chỉ byte								TH1
22	17	16	15	14	13	12	11	10	8C	Địa chỉ byte								TH0
21	0F	0E	0D	0C	0B	0A	09	08	8B	Địa chỉ byte								TL1
20	07	06	05	04	03	02	01	00	8A	Địa chỉ byte								TL0
1F	Bank 3								89	Địa chỉ byte								TMOD
18									88	8F	8E	8D	8C	8B	8A	89	88	TCON
17	Bank 2								87	Địa chỉ byte								PCON
10																		
0F	Bank 1								83	Địa chỉ byte								DPH
08									82	Địa chỉ byte								DPL
07	Bank 0 (mặc định) R0-R7								81	Địa chỉ byte								SP
00									80	87	86	85	84	83	82	81	80	P0

Hình 1.5 Cấu trúc RAM nội của 8051

3.2 Vùng RAM địa chỉ bít

8051 chứa 210 vị trí bít được định địa chỉ trong đó 128 bít chưa trong các byte ở địa chỉ từ 20H đến 2FH (16 byte x 8 bít = 128 bít) và phần còn lại chứa trong các thanh ghi chức năng đặc biệt.

Ý tưởng truy xuất các bít riêng rẽ thông qua phần mềm là một đặc trưng mạnh của

hầu hết các bộ vi điều khiển. Các bít có thể được set, xóa, AND, OR…bằng một lệnh.

Hầu hết các bộ vi xử lý yêu cầu một chuỗi lệnh đọc-sửa-ghi để nhận được cùng một kết quả. Ngoài ra, 8051 còn có port xuất/nhập có thể định địa chỉ từng bít, điều này làm đơn giản việc giao tiếp bằng phần mềm với các thiết bị xuất/nhập đơn bít.

Như vừa đề cập ở trên, 8051 có 128 vị trí bít được định địa chỉ và có nhiều mục đích ở các byte có địa chỉ từ 20H đến 2FH. Các địa chỉ này được

truy xuất như là các byte hay các bít tùy vào lệnh cụ thể. Ví dụ để set bít 67H bằng 1 ta dùng lệnh sau:

SETB 67H

Theo hình 2.6 cho thấy bít ở địa chỉ 67H là bít có ý nghĩa lớn nhất của byte ở địa chỉ 2CH. Lệnh vừa nêu trên không ảnh hưởng đến các bít khác trong byte này. Hầu hết các bộ vi xử lý muốn thực hiện công việc như trên phải dùng các lệnh có dạng tương tự như sau:

```
MOV A, 2CH              ; đọc cả byte
MOV A, #10000000B       ; set bít có ý nghĩa lớn nhất
MOV 2CH, A              ; ghi trở lạI cả byte
```

3.3 Các dãy thanh ghi

32 vị trí thấp nhất của bộ nhớ nội chứa các dãy thanh ghi. Các lệnh của 8051 hỗ trợ 8 thanh ghi từ R0 đến R7 thuộc dãy 0 (bank 0). Đây là dãy mặc định sau khi rết hệ thống.

Các thanh ghi này ở các địa chỉ từ 00H đến 07H. Lệnh sau đây đọc nội dung tại địa chỉ 05H vào thanh ghi A

MOV A, R5

Lệnh này là lệnh 1 byte dùng kiểu định địa chỉ thanh ghi. Dĩ nhiên thao tác tương tự có thể thực hiện với một lệnh 2 byte bằng cách dùng kiểu định địa chỉ trực tiếp.

MOV A, 05H

Các lệnh sử dụng các thanh ghi từ R0 đến R7 là các lệnh ngắn và thực hiện nhanh

hơn so với các lệnh tương đương sử dụng cách định địa chỉ trực tiếp. Các giá trị dữ liệu thường được sử dụng nên chứa ở một trong các thanh ghi này. Dãy thanh ghi đang được sử dụng được gọi là dãy thanh ghi tích cực. Dãy thanh ghi tích cực có thể được thay đổi bằng cách thay đổi các bít chọn dãy trong từ trạng thái chương trình PSW. Giả sử rằng dãy thanh ghi 3 (bank 3) tích cực, Lệnh sau đây ghi nội dung thanh ghi A vào vị trí 18H

MOV R0, A

Ý tưởng "các dãy thanh ghi" cho phép chuyển đổi ngữ cảnh nhanh và có hiệu quả ở những nơi mà các phần riêng rẽ của phần mềm sử dụng một tập thanh ghi riêng, độc lập với các phần khác của phần mềm.

4. Các thanh ghi chức năng đặc biệt

Các thanh ghi nội của hầu hết các bộ vi xử lý đều được truy xuất rõ ràng bởi một tập lệnh. Việc truy xuất các thanh ghi cũng được sử dụng trên 8051

ví dụ lệnh **INC A** tăng nội dung A lên 1.

Các thanh ghi nội của 8051 được cấu hình thành một phần của RAM trên chip, do

vậy mỗi một thanh ghi cũng có một địa chỉ. Điều này hợp lý với 8051 vì chip này có rất nhiều thanh ghi, cũng như các thanh ghi từ R0 đến R7, trong 8051

còn có 21 thanh ghi chức năng đặc biệt (SFR) chiếm phần trên của RAM nội từ địa chỉ 80H đến FFH (hình 2.6)

Lưu ý là không phải tất cả địa chỉ từ 80H đến FFH đều được định nghĩa mà chỉ có
21 địa chỉ được định nghĩa (26 trên 8032/8052).

Thanh ghi A có thể được truy xuất rõ ràng như được minh họa trong các ví dụ ở các phần trên. Hầu hết các thanh ghi chức năng đặc biệt được truy xuất bằng kiểu định địa chỉ trực tiếp. Cần lưu ý trong hình 2.6 là một số thanh ghi chức năng đặc biệt được định địa chỉ từng bít. Ví dụ lệnh sau

SETB 0E0H

Sẽ set bít 0 của thanh ghi A lên 1, các bít khác không thay đổi. Một nhận xét là tại
địa chỉ E0H có thể là địa chỉ byte cho cả thanh ghi A và địa chỉ bít của bít có ý nghĩa thấp nhất trong A. Vì lệnh SETB thao tác trên các bít và không thao tác trên các byte nên chỉ có bít được định địa chỉ bị ảnh hưởng. Lưu ý là các bít được định địa chỉ trong một thanh ghi chức năng đặc biệt có 5 bít cao của địa chỉ giống nhau cho tất cả các bít của thanh ghi này. Ví dụ port 1 có địa chỉ byte là 90H (10010000B) và các bít trong port này có các địa chỉ từ 90H đến 97H hay 10010xxxB.

Từ trạng thái chương trình PSW (program status word) sẽ được thảo luận chi tiết trong phần sau. Các thanh ghi chức năng đặc biệt khác cũng được giới thiệu vắn tắt.

4.1 Từ trạng thái chương trình PSW

BẢNG 1.3 THANH GHI PSW

Bít	Ký hiệu	Địa chỉ	Mô tả
PSW.7	CY	D7H	Cờ nhớ
PSW.6	AC	D6H	Cờ nhớ phụ
PSW.5	F0	D5H	Cờ 0
PSW.4	RS1	D4H	Chọn dãy thanh ghi (bít 1)
PSW.3	RS0	D3H	Chọn dãy thanh ghi (bít 0)
			00 = bank 0 01 = bank 1 10 = bank2 11 = bank 3
PSW.2	OV	D2H	Cờ tràn
PSW.1	-	D1H	Dự trữ
PSW.0	P	D0H	Cờ chẵn lẻ

PSW có địa chỉ là D0H chứa các bít trạng thái có chức năng được tóm tắt trong bảng 2.3

Cờ nhớ

Cờ nhớ CY (carrier flag) có hai công dụng: Công dụng truyền thống trong các phép toán số học là được set bằng 1 nếu có số nhớ từ phép cộng bít 7 hoặc có số mượn mang đến bít 7. Ví dụ nếu thanh ghi A có nội dung là FFH

ADD A,#1

Sẽ làm cho A có nội dung là 00H và cờ CY trong PSW được set bằng 1. Cờ nhớ CY còn là một bộ tích lũy logic được dùng như một thanh ghi 1 bít đối với các lệnh logic thao tác trên các bít. VD: Lệnh sau đây sẽ AND bít 25H với cờ CY và đặt kết quả vào cờ CY

ANL C, 25H ; AND bít ở địa chỉ 25H với cờ nhớ

Cờ nhớ phụ

Khi cộng các giá trị BCD, cờ nhớ phụ AC (auxiliary carry flag) được set bằng 1 nếu có 1 số nhớ được tạo ra từ bít 3 chuyển sang bít 4 hoặc nếu kết quả trong decade thấp nằm trong khoảng từ 0AH đến 0FH. Nếu các giá trị được cộng là giá trị BCD, lệnh cộng phải được tiếp theo bởi lệnh DA A (hiệu chỉnh thập phân thanh ghi A) để đưa các kết quả lớn hơn 9 về giá trị đúng.

Cờ 0

Đây là cờ có nhiều mục đích dành cho các ứng dụng của người lập trình.

Các bít chọn dãy thanh ghi

Các bít chọn dãy thanh ghi RS0, RS1 dùng để xác định dãy thanh ghi tích cực. Các

bít này được xóa sau khi có thao tác reset hệ thống và đổi mức logic bởi phần mềm khi cần. VD: Ba lệnh sau cho phép dãy thanh ghi 3 (bank 3) tích cực, sau đó di chuyển nội dung của R7 (địa chỉ byte 1FH) vào thanh ghi A.

SETB RS1

SETB RS0

MOV A, R7

Khi đoạn chương trình trên được dịch, các địa chỉ bít sẽ thay thế cho các ký hiệu RS0 và RS1, vậy thì lệnh SETB RS1 tương đương với lệnh SETB 0D4H.

Cờ tràn

Cờ tràn OV (over flag) được set bằng 1 sau phép toán cộng hoặc trừ nếu có xuất hiện một tràn số học. Khi các số có dấu được cộng hoặc được trừ, phần mềm có thể kiểm tra bít tràn OV để xác định xem kết quả có nằm trong phạm vi hợp lệ hay không.

Với phép cộng các số không dấu, cờ tràn OV được bỏ qua. Kết quả lớn hơn +128 hoặc nhỏ hơn -127 sẽ set cờ OV bằng 1, ví dụ phép cộng sau đây gây ra một tràn và set cờ OV trong PSW

Số hex	0F	Số thập phân 15
	+7F	+127
	8E	+142

8EH biểu diễn số âm -116, như vậy không đúng với kết quả mong muốn là 142 nên cờ OV được set bằng 1.

Cờ chẵn lẻ

Bít chẵn lẻ P tự động được set bằng 1 hay xóa bằng 0 ở mỗi chu kỳ máy để thiết lập kiểm tra chẵn cho thanh ghi A. Số các bít 1 trong A cộng với bít P luôn luôn là số chẵn.

Ví dụ nếu thanh ghi A có nội dung là 10101101B, bít P sẽ là 1 để có số bít 1 là 6. Bít chẵn lẻ được sử dụng nhiều để kết hợp với các chương trình xuất/nhập nối tiếp trước khi truyền dữ liệu hoặc để kiểm tra chẵn lẻ sau khi nhận dữ liệu.

4.2 Thanh ghi B

Thanh ghi B ở địa chỉ F0H được dùng chung với thanh ghi A trong các phép toán

nhân, chia. Lệnh MUL AB nhân hai số 8 bít không dấu chứa trong A và B và chứa kết quả 16 bít vào cặp thanh ghi B:A (thanh ghi A cất byte thấp và thanh ghi B cất byte cao)

Lệnh chia DIV AB chia A cho B, thương số cất trong A và dư số cất trong B. Thanh ghi B còn được xử lý như một thanh ghi nháp, các bít được định địa chỉ của thanh ghi B có địa chỉ từ F0H đến F7H.

4.3 Con trỏ stack

Con trỏ stack SP (stack pointer) là một thanh ghi 8 bít ở địa chỉ 81H. SP chứa địa chỉ của dữ liệu hiện đang ở đỉnh của stack. Các lệnh liên quan đến stack bao gồm lệnh cất dữ liệu vào stack và lệnh lấy dữ liệu ra khỏi stack. Việc cất vào stack làm tăng SP trước khi ghi dữ liệu và việc lấy dữ liệu ra khỏi stack sẽ giảm SP. Vùng stack của 8051 được giữ trong RAM nội và được giới hạn đến các địa chỉ truy xuất được bởi kiểu định địa chỉ gián tiếp. Vùng RAM nội có 128 byte trên 8031/8051 hoặc 256 byte trên 8032/8052, lệnh sau khởi động SP để bắt đầu stack tại địa chỉ 60H

MOV SP,#5FH

Vùng stack được giới hạn là 32 byte trên 8031/8051 vì địa chỉ vì địa chỉ cao nhất của RAM trên chíp là 7FH. Giá trị 5FH được dùng ở đây vì SP tăng lên 60H trước khi thực hiện cất vào stack

Nếu không khởi động SP, nội dung mặc định của thanh ghi này là 07H nhằm duy trì sự tương thích với 8048, bộ vi điều khiển đời trước của 8051. Kết quả là thao tác cất vào stack đầu tiên sẽ lưu dữ liệu vào vị trí nhớ có địa chỉ 08H. Như vậy, nếu phần mềm ứng dụng không khởi động SP, dãy thanh

ghi 1 và có thể 2 và 3 không còn hợp lệ vì vùng này được sử dụng làm stack. Các lệnh PUSH và POP sẽ cất dữ liệu vào stack và lấy dữ liệu từ stack, các lệnh gọi chương trình con (ACALL, LCALL) và lệnh trở về RET, RETI cũng cất và phục hồi nội dung của bộ đếm chương trình PC (program counter).

4.4 Con trỏ dữ liệu DPTR

Con trỏ dữ liệu DPTR (data pointer) được dùng để truy xuất bộ nhớ chương trình ngoài hoặc bộ nhớ dữ liệu ngoài. DPTR là một thanh ghi 16 bít có địa chỉ là 82H (DPL byte thấp) và 83H (DPH byte cao), 3 lệnh sau đây ghi 55H vào RAM ngoài ở địa chỉ 1000H

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

Lệnh đầu tiên sử dụng kiểu định địa chỉ tức thời để nạp hằng dữ liệu 55H vào thanh ghi A, lệnh thứ hai cũng sử dụng kiểu định địa chỉ tức thời, lần này nạp hằng địa chỉ 16 bít 1000H cho con trỏ dữ liệu DPTR. Lệnh thứ ba sử dụng kiểu định địa chỉ gián tiếp di chuyển giá trị 55H chứa trong A đến RAM ngoài tại địa chỉ chứa trong DPTR (1000H).

4.5 Các thanh ghi port

Các port xuất/nhập của 8051 bao gồm port 0 tại địa chỉ 80H, port 1 tại 90H, port 2 tại A0H và port 3 tại B0H. Các port 0, 2 và 3 không được dùng để xuất nhập nếu dùng thêm bộ nhớ ngoài hoặc nếu có một số đặc tính đặc biệt của 8051 được sử dụng (như là ngắt, cổng nối tiếp...) Ngược lại, P1.2 đến P1.7 luôn luôn là các đường xuất/nhập đa mục đích hợp lệ.

Tất cả các port đều được định địa chỉ từng bít nhằm cung cấp các khả năng giao tiếp mạnh, ví dụ một động cơ nối qua một cuộn dây và một mạch kích dùng transistor nối đến bít 7 của port 1, động cơ có thể dừng hay chạy chỉ nhờ vào một lệnh đơn của 8051

```
SETB P1.7
```

Làm động cơ chạy, và

```
CLR P1.7
```

Sẽ làm dừng động cơ

Các lệnh trên sử dụng toán tử . (dot) để định địa chỉ 1 bít trong 1 byte, cho phép
định địa chỉ từng bít.

Trình dịch hợp ngữ thực hiện việc biến đổi dạng ký hiệu thành địa chỉ thực tế, nghĩa là hai lệnh sau tương đương nhau

```
CLR P1.7
CLR 97H
```

Việc sử dụng các ký hiệu được định nghĩa trước của trình dịch hợp ngữ sẽ được thảo luận chi tiết ở các bài sau.

Ví dụ sau đây khảo sát việc giao tiếp với một thiết bị có bít trạng thái gọi là BUSY, bít này được set bằng 1 khi thiết bị đang bận và được xóa khi thiết bị đã sẵn sàng. Nếu BUSY được nối với bít 5 của port 1, vòng lặp sau đây được dùng để chờ cho đến khi thiết bị sẵn sàng.

WAIT: JB P1.5, WAIT

Lệnh trên có nghĩa là nếu bít P1.5 được set thì nhảy đến nhãn WAIT cũng có nghĩa là nhảy về và kiểm tra lần nữa.

4.6 Các thanh ghi định thời

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

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

4.7 Các thanh ghi của port nối tiếp

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

Các chế độ hoạt động khác nhau được lập trình thông qua thanh ghi điều khiển port nối tiếp SCON (serial port control register) ở địa chỉ 98H, thanh ghi này được định địa chỉ từng bít

4.8 Các thanh ghi ngắt

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

4.9 Thanh ghi điều khiển nguồn

Thanh ghi điều khiển nguồn PCON (power control register) có địa chỉ 87H chứa các bít điều khiển được tóm tắt trong bảng 2.4

Bít SMOD tăng gấp đôi tốc độ baud của port nối tiếp khi port này hoạt động ở các

chế độ 1, 2 hoặc 3. các bít 4, 5 và 6 của PCON không được định nghĩa, các bít 2 và 3 là các bít cờ đa mục đích dành cho các ứng dụng của người sử dụng.

Các bít điều khiển nguồn, nguồn giãm PD và nghỉ IDL, hợp lệ trong tất cả các chip họ MCS-51 nhưng chỉ được hiện thực trong các phiên bản CMOS của MCS-51, PCON không được định địa chỉ bít.

Chế độ nguồn giảm

Lệnh thiết lập bít PD bằng 1 sẽ là lệnh sau cùng được thực hiện trước khi đi vào chế độ nguồn giảm. Ở chế độ nguồn giảm:

1) Mạch dao động trên chip ngừng hoạt động
2) Mọi chức năng ngừng hoạt động
3) Nội dung của RAM trên chíp được duy trì
4) Các chân port duy trì mức logic của chúng
5) ALE và PSEN được giữ ở mức thấp, chỉ ra khỏi chế độ này bằng cách reset hệ thống.

Trong suốt thời gian ở chế độ nguồn giảm, VCC có điện áp là 2 V, cần phải giữ cho VCC không thấp hơn sau khi đạt được chế độ nguồn giảm và cần phục hồi VCC = 5 V tối thiểu 10 chu kỳ dao động trước khi chân reset đạt mức thấp lần nữa

BẢNG 1.4 Thanh ghi PCON

Bít	Ký hiệu	Mô tả
7	SMOD	Bít tăng đôi tốc độ baud ở chế độ 1, 2 và 3 của port nối tiếp
6	-	
5	-	
4	-	
3	Bít cờ đa mục đích 1	
2	Bít cờ đa mục đich 2	
1	PD	Nguồn giãm Chế độ nghỉ, thoát ra bằng 1 ngắt hoặc reset
0	IDL	

Chế độ nghỉ

Lệnh thiết lập bít IDL bằng 1 sẽ là lệnh sau cùng được thực hiện trước khi đi vào chế độ nghỉ. Ở chế độ nghỉ tín hiệu clock nội được khóa không cho đến CPU nhưng không khóa đối với các chức năng ngắt, định thời và port nối tiếp. Trạng thái của CPU được duy trì và nội dung của tất cả các thanh ghi cũng được giữ không đổi.

Các chân port cũng được duy trì các mức logic của chúng. ALE và PSEN được giữ ở mức cao.

Chế độ nghỉ kết thúc bằng cách cho phép ngắt hoặc bằng cách reset hệ thống, cả
hai cách vừa nêu trên đều xóa bít IDL.

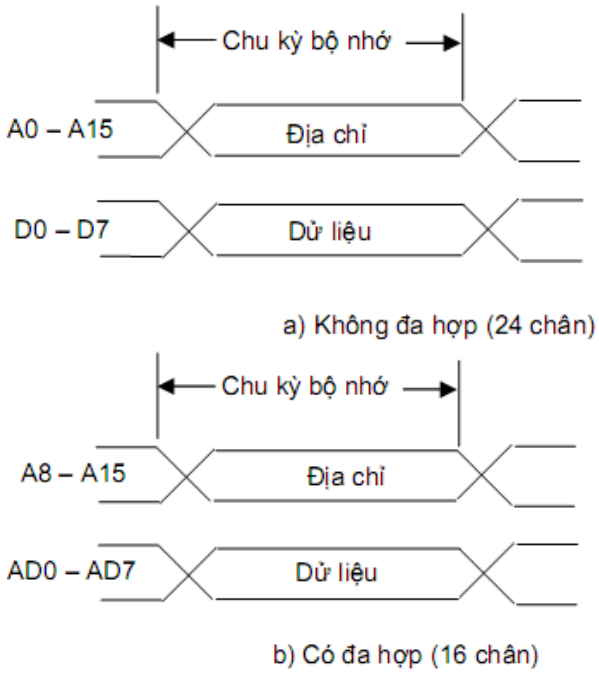
5. Bộ nhớ ngoài

Mục tiêu:

- Trình bày được bộ nhớ ngoài của vi điều khiển

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 xử 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. Ngỏ 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.



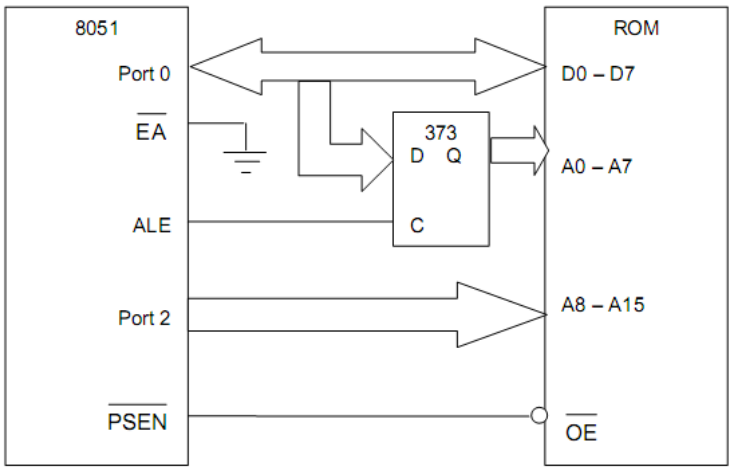
Hình 1.6 Bus đa hợp địa chỉ/dữ liệu

Sắp xếp không đa hợp xử 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 xử dụng làm bus dữ liệu và dữ liệu sẻ được đọc hay ghi.

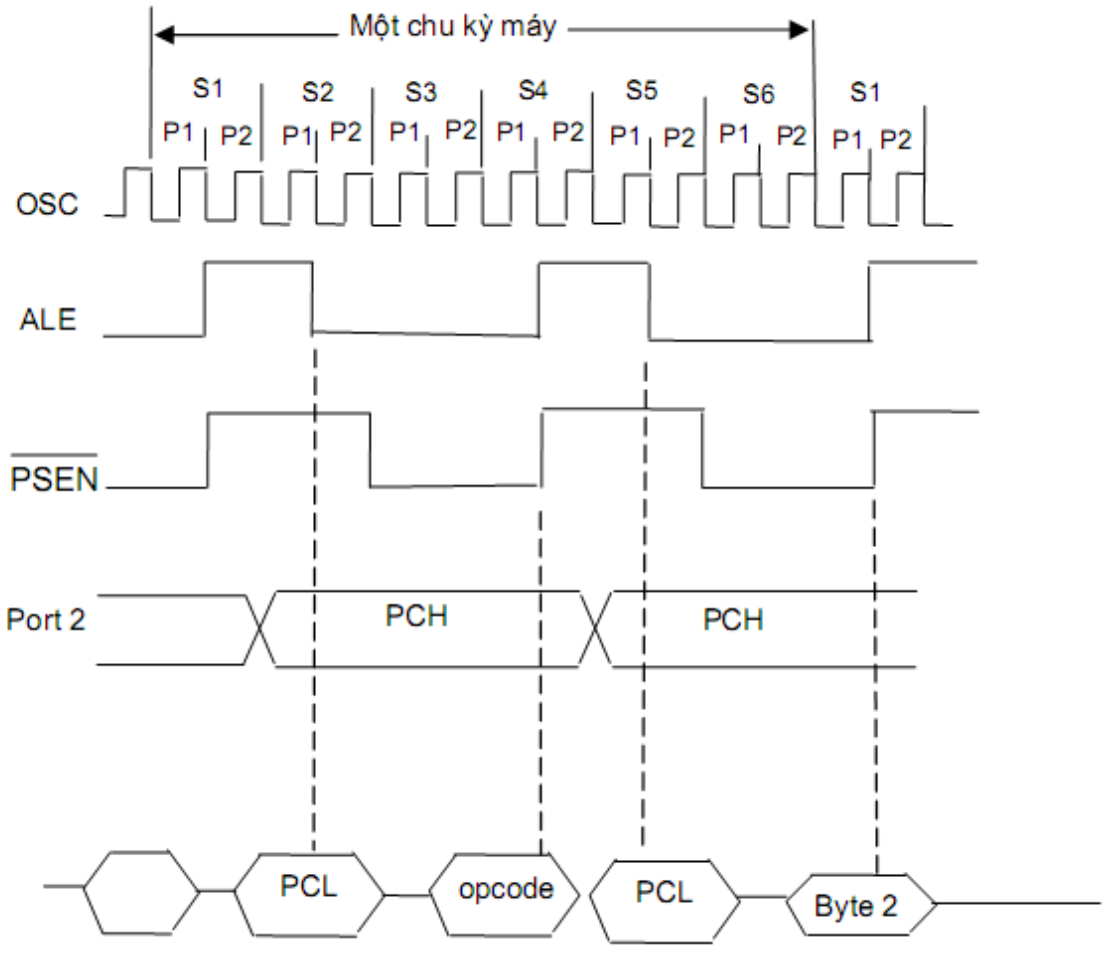
5.1 Truy xuất bộ nhớ chương trình ngoài

Bộ nhớ chương trình ngoài là bộ nhớ chỉ đọc, được cho phép bởi tín hiệu $\overline{PSEN}$



Hình 1.7 Truy xuất ROM ngoài

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 1.7 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 1 µS. 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 1.8.



Hình 1.8 Chu kỳ tìm nạp lệnh ROM ngoài

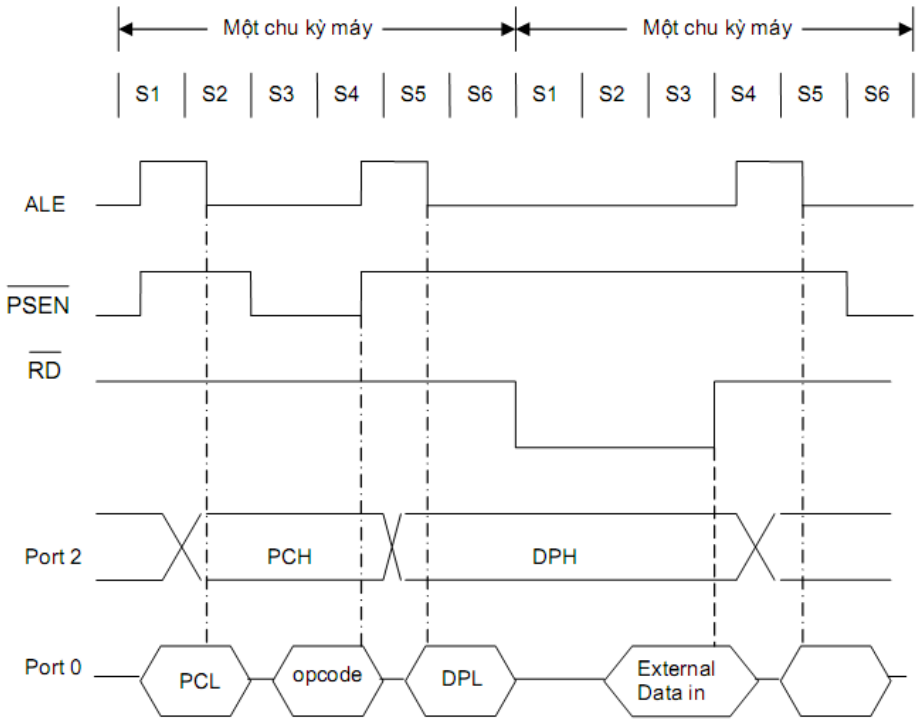
5.2 Truy xuất bộ nhớ dữ liệu ngoài

Bộ nhớ dữ liệu ngoài là bộ nhớ đọc-ghi được cho phép bởi các tín hiệu $\overline{RD}$ và $\overline{WR}$ 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 bít DPTR hoặc R0, R1 làm thanh ghi chứa địa chỉ.

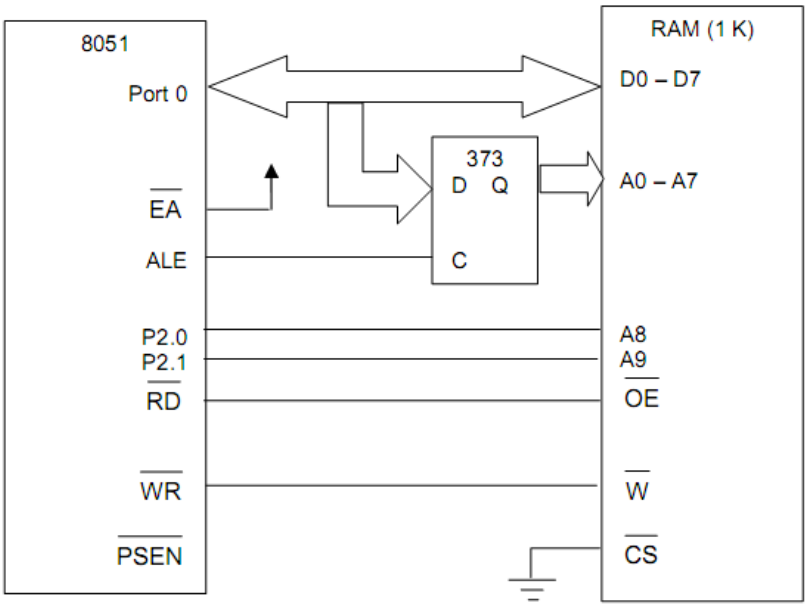
RAM có thể giao tiếp với 8051 theo cùng cách như ROM ngoại trừ đường $\overline{RD}$ nối với đường cho phép xuất $\overline{OE}$ của RAM và đường $\overline{WR}$ nối với đường $\overline{W}$ của RAM. Các kết nối với bus dữ liệu và bus địa chỉ giống như ROM bằng cách sử dụng port 0 và port 2 như ở phần trên dung lượng của RAM lên đến 64 K.

Giản đồ thời gian của thao tác đọc dữ liệu ở bộ nhớ dữ liệu ngoài được trình bày ở hình 2.10 cho lệnh MOVX A,@DPTR. Lưu ý là cả hai xung ALE và $\overline{PSEN}$ được bỏ qua ở nơi mà xung $\overline{RD}$ cho phép đọc RAM, nếu lệnh MOVX và RAM ngoài không bao giờ được dùng, các xung ALE luôn có tần số bằng 1/6 tần số của mạch dao động

Giản đồ thời gian của chu kỳ ghi (lệnh MOVX @DPTR, A) cũng tương tự ngoại trừ các xung $\overline{WR}$ ở mức thấp và dữ liệu được xuất ra ở port 0 ($\overline{RD}$ vẫn ở mức cao)



Hình 1.9 Giản đồ thời gian lệnh MOVX



Hình 1.10 Giao tiếp với 1K 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. VD: Với một RAM 1 KB (4 trang 256) có thể được kết nối với 8051 như ở hình 1.10.

Các bít 0 và 1 của port 2 phải được khởi động để chọn một trang, sau đó dùng lệnh MOVX để đọc hoặc ghi trên trang này. Giả sử P2.0 = P2.1 = 0, các lệnh sau có thể dùng để đọc nội dung của RAM ngoài tại địa chỉ 0050H vào thanh ghi A

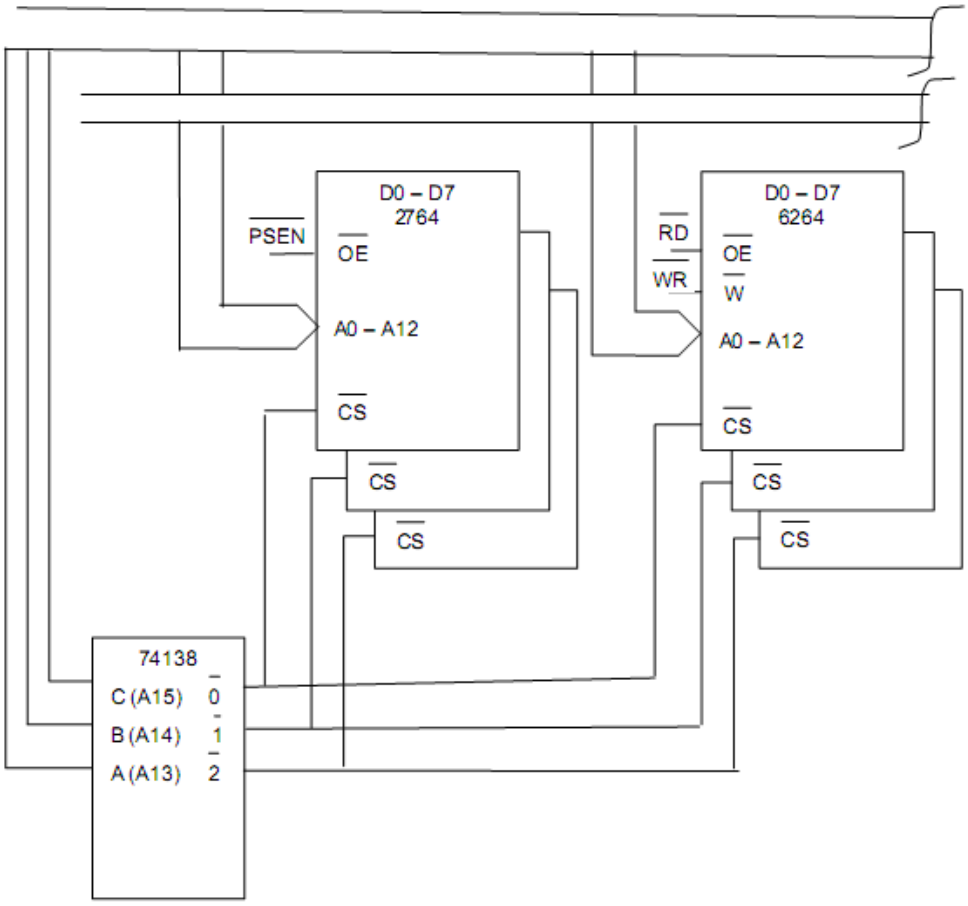
MOV R0,#50H
MOVX A, @R0

Để đọc ở địa chỉ cuối cùng của RAM là 03FFH thì phải chọn trang 3, nghĩa là phải set các bít P2.0 và P2.1 băng 1 như chuỗi lệnh sau

SETB P2.0
SETB P2.1
MOV R0,#0FFH
MOVX A, @R0

Một đặc trưng của thiết kế này là các bít từ 2 đến 7 của port 2 không còn cần làm bít địa chỉ nữa, các bít còn lại này có thể xử dụng cho mục đích xuất/nhập.

5.3 Giãi mã địa chỉ



Hình 1.11: Giãi mã địa chỉ

Nếu có nhiều ROM hoặc RAM giao tiếp với 8051 thì cần phải giãi mã địa chỉ. Việc giãi mã này cũng cần cho hầu hết các bộ vi xử lý.

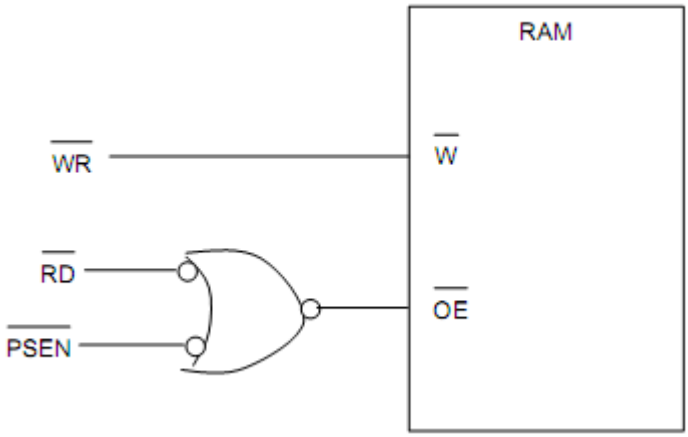
Ví dụ nếu các ROM và RAM 8 KB được xử dụng, địa chỉ phải được giải mã để chọn các IC nhớ này trên các giới hạn 8K (0000H..1FFFH, 2000H..3FFFH....) Một IC giải mã điển hình là 74138 được dùng với các ngõ ra được nối với các ngỏ vào chọn chíp $\overline{CS}$ của các IC nhớ như mô tả ở hình 1.11 cho một bộ nhớ có nhiều EPROM 2764 và RAM 6264. Cần lưu ý là do các đường cho phép riêng rẻ ($\overline{PSEN}$ cho bộ nhớ chương trình, $\overline{RD}$ và $\overline{WR}$ cho bộ nhớ dữ liệu) 8051 có thể quản lý không gian nhớ 64 K cho bộ nhớ ROM và 64 K cho bộ nhớ RAM.

5.4 Các không gian chương trình và dữ liệu gối nhau

Vì bộ nhớ chương trình là bộ nhớ chỉ đọc, một tình huống khó xử được phát sinh trong quá trình phát triển phần mềm cho 8051. Làm thế nào phần mềm được viết cho một hệ thống đích để gỡ rối nếu phần mềm chỉ có thể được thực hiện từ không gian bộ nhớ chương trình chỉ đọc.

Giải pháp tổng quát là cho các không gian bộ nhớ chương trình và dữ liệu ngoài gối lên nhau. Vì $\overline{PSEN}$ được dùng để đọc bộ nhớ chương trình và $\overline{RD}$ được dùng để đọc bộ nhớ dữ liệu, một RAM có thể chiếm không gian nhớ chương trình và dữ liệu bằng cách nối chân $\overline{OE}$ tới ngõ ra cổng AND có các ngõ vào là $\overline{PSEN}$ và $\overline{RD}$.

Mạch trình bày ở hình 1.12 cho phép IC RAM được ghi như là bộ nhớ dữ liệu và được đọc như là bộ nhớ chương trình hoặc dữ liệu. Vậy thì một chương trình có thể được nạp vào RAM bằng cách ghi vào RAM như là bộ nhớ dữ liệu và được thực hiện bằng cách truy xuất như là bộ đếm chương trình.



Hình 1.12 Gối không gian nhớ ROM và RAM

Thực hành ứng dụng

1. Viết lệnh set bit có giá trị thấp nhát tại địa chỉ byte 25H ?
2. Viết chuỗi lệnh thực hiện phép OR nội dung tai địa chỉ bít 00H với nội dung tại
01H, kết quả đưa vào địa chỉ bít 02H.
3. Một chương trình con xử dụng các thanh ghi R0 – R7. Chương trình sẻ chuyển
sang dảy thanh ghi 3 khi bắt đầu và phục hồi trở lại dãy thanh ghi ban đầu trước
khi thoát. Hãy mô tả chương trình con này
4. Vi điều khiển 80C31BH-1 hoạt động với thạch anh 16 MHz. Tần số xung ALE là
bao nhiêu nếu không dùng lệnh MOVX ?
5. Địa chỉ của bít giá trị cao nhất tại byte RAM nội 25H trong 8051 là bao nhiêu ?
6. Viết lệnh set bít cao nhất trong thanh ghi A
7. Cho biét trạng thái của bít P trong thanh ghi PSW sau khi thực hiện lênh sau

MOV A,#55H

8. Viét chuỗi lệnh copy nội dung thanh ghi R7 đên địa chỉ 100H trong RAM ngoài
9. Giả sử lệnh đầu tiên sau khi reset hệ thống là lệnh gọi chương trình con. Hãy
cho biết nội dung của PC được lưu vào đâu trước khi chương trình con này được thực hiện ?
10. Trình bày điễm khác nhau giữa chế độ nghỉ với chế độ giãm nguồn ?
11. Viết lệnh thực hiện chế độ giãm nguồn
12. Trình bày sơ đồ kết nối thêm 2 RAM ngoài mỗi RAM có dung lượng là 32 KB với
8051

CHƯƠNG 1

TẬP LỆNH CỦA 89C51

MÃ BÀI: MH30-02

Mục tiêu:

- Trình bày được các chế độ đánh địa chỉ của 89C51
- Sử dụng được tập lênh của 89C51.
- Tự tin trong thao tác, tiếp cận tập lệnh 89C51.
- Tính chính xác, tỉ mỉ, cẩn thận.

Nội dung chính:

1. Các chế độ đánh địa chỉ

Mục tiêu:

- Trình bày được các chế độ đánh địa chỉ của 89C51

Khi thực hiện một lệnh với dữ liệu, một câu hỏi được đặt ra là ``dữ liệu cần xử lý này nằm ở đâu ?`` Câu trả lời nằm trong các cách định địa chỉ của MCs 8051, dữ liệu cần tìm có thể là byte thứ hai trong câu lệnh, trong thanh ghi VD: R4, trong ô nhớ có địa chỉ
VD: 35H hoặc là trong ô nhớ ngoài có địa chỉ chứa trong con trỏ dữ liệu. Họ MCs 8051 có tất cả 8 cách định địa chỉ

1.1 Bằng 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 mả 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:

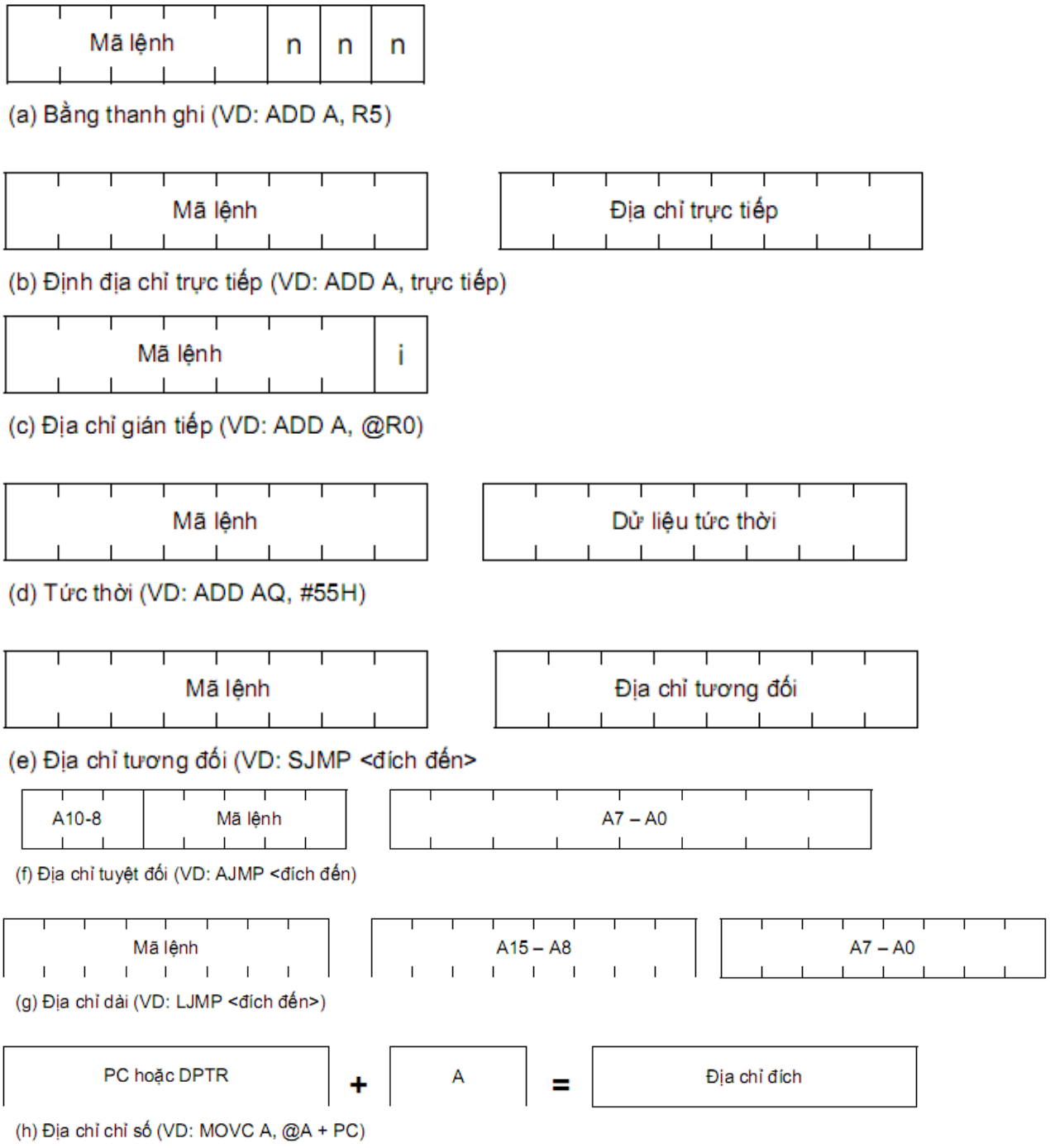
MOV PSW, #00011000B

Lúc này dãy thanh ghi hoạt động là dãy 3, bít 4 của PSW là bít RS1 và bít 3 của SW 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-thannh ghi B là AB. VD:

INC DPTR

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



Hình 2.1 Các cách định địa chỉ của 8051

1.2 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 lệnh sau:

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 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 dữ liệu, hai thanh ghi này được xác định bởi bít thấp nhất trong mã lệnh (hình 2.1c).

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 VD các lệnh sai 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 đảm ô nhớ 7FH bị xóa trước khi kết thúc chương trình.

1.4 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 dữ liệu (hình 2.1d).

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 sẽ nạp hằng số 16 bít có giá trị 8000H vào con trỏ dữ liệu

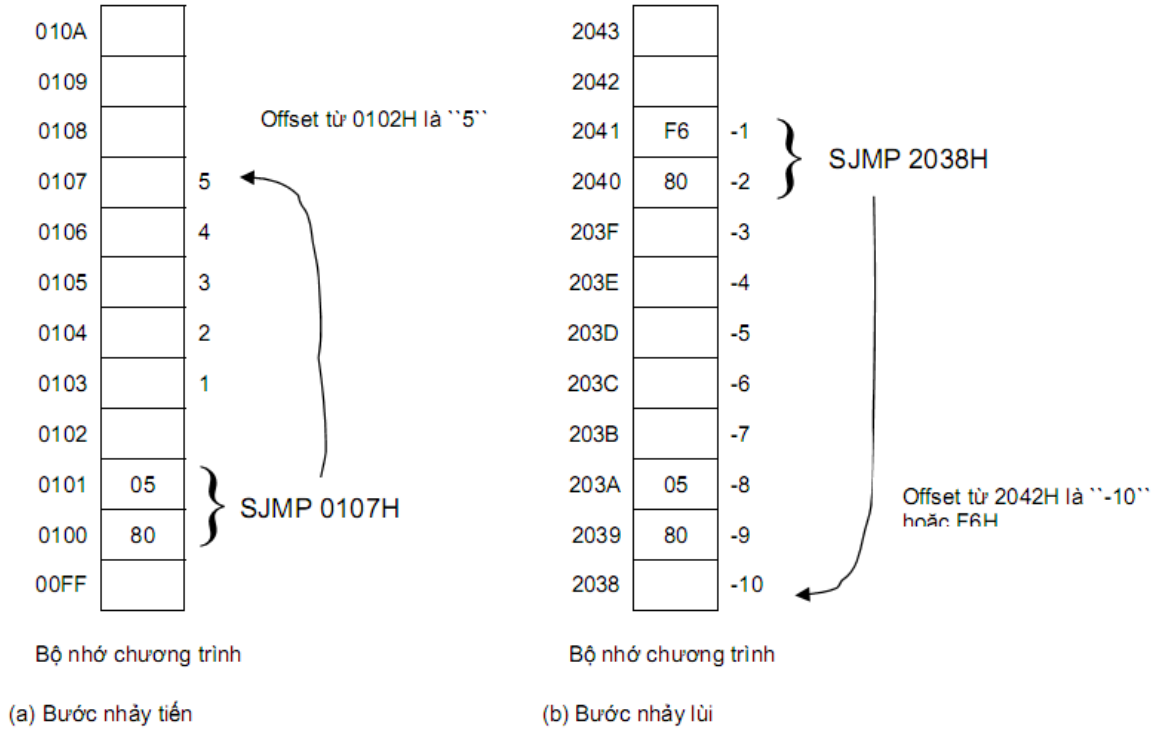
1.5 Tương đối

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 ssó 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 (hình 3.1e).

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 (hình 2.2).

Thông thường chi tiết này không quan trọng đối với người lập trình bởi vì địa chỉ đích thường được chỉ ra bởi một nhãn và phần mềm hợp ngữ sẽ tự xác định offset tương ứng. VD: Nếu nhãn 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 điểm 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.



Hình 2.2 Cách tính địa chỉ offset

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 2.1f).

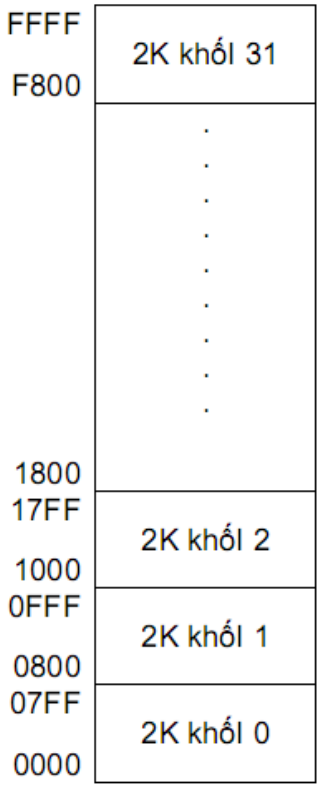
Năm bít cao của địa chỉ đích chính là 5 bít cao hiện đang chứa trong bộ đếm chương trình, do đó lệnh theo sau lệnh nhảy và đích đến của lệnh nhảy phải có vị trí trong cùng một trang 2K, bởi vì A15-A11 không thay đổi (hình 3.3). Ưu điểm của phương pháp này là mã lệnh ngắn (2 byte) nhưng khuyết điểm là 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 mả lệnh (hình 2.1g). 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 lênh 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 theo 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 2.1h). 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.



Hình 2.3 Tổ chức các khối 2K trong bộ nhớ chương trình 64K

2. Các lệnh của 89C51

Mục tiêu:

- Sử dụng được tập lênh của 89C51.

2.1 Nhóm lệnh số học

Nhóm lệnh này sử dụng 4 cách định địa chỉ. VD: Lệnh ADD A có thể viết theo 4 cách sau:

ADD A, 7FH (trực tiếp)

ADD A, @ R0 (gián tiếp)

ADD A, R7 (thanh ghi)

ADD A, # 35H (tức thời)

Tất cả các lệnh số học đều được thực hiện trong vòng một chu kỳ đồng hồ ngoại trừ các lệnh INC DPTR (2 chu kỳ) lệnh MUL AB và DIV AB (4 chu kỳ). Nếu tần số xung đồng hồ là 12 MHz thì một chu kỳ là 1 µS.

Họ 8051 cung cấp cách định địa chỉ rất mạnh để truy xuất bộ nhớ trong, nội dung của một ô nhớ bất kỳ có thể được tăng lên hoặc giảm xuống bằng cách định địa chỉ trực tiếp mà không cần đến trung gian là bộ tích lũy. VD: Tại địa chỉ RAM trong là 7FH có nội dung là 40H, lệnh sau đây.

INC 7FH

Sẽ tăng nội dung ô nhớ này lên 41H. Một trong các lệnh INC có khả năng tác động lên con trỏ dữ liệu 16 bít, vì nội dung con trỏ là địa chỉ 16 bít

của bộ nhớ ngoài nên lệnh này rất thường được sử dụng, nhưng 8051 lại không có lệnh giảm con trỏ dữ liệu vì vậy để thực hiện yêu cầu này cần phải kết hợp một số lệnh như sau:

```
DEC DPL
 MOV R7, DPL
 CJNE R7, #0FFH, SKIP
DEC DPH
SKIP: ……..
 …….
```

Phải giảm riêng byte cao và byte thấp của DPTR. Tuy nhiên, chỉ giảm byte cao khi khi byte thấp bị tràn từ 00H đến FFH.

Lệnh **MUL AB** nhân nội dung của bộ tích lũy với dữ liệu chứa trong thanh ghi B và đưa kết quả vào cặp thanh ghi này: Byte cao của kết quả vào B và byte thấp vào A. Lệnh DIV AB lấy nội dung trong A chia cho dữ liệu trong B, thương số được chứa trong A và số dư chứa trong B. VD: Nội dung của A là 25 (19H) và B là 6 (06H), lệnh

```
DIV AB
```

Sẽ lấy 25 chia cho 6, kết quả là 4 chứa trong A và số dư 1 được chứa trong B (25 chia 6 bằng 4 dư 1)

Đối với các phép tính số BCD, sau lệnh ADD và ADDC phải có lệnh DA A (decimal adjust) để bảo đảm là kết quả nằm trong vùng cho phép của số BCD, chú ý là lệnh này không biến đổi số nhị phân sang số BCD mà nó chỉ tạo ra một kết quả có nghĩa. VD: Giả sử A chúa số BCD là 59 (59H)

```
ADD A, # 1
DA A
```

Lệnh thứ nhất cộng A với số 1 và được kết quả là 5AH, kết quả này sẻ được sửa lại thành số BCD có giá trị là 60 (60H) vì 59 + 1 = 60.

2.2 Nhóm lệnh logic

Các lệnh logic của 8051 thực hiện các phép toán logic (AND, OR, EXOR và NOT) giữa các byte dữ liệu theo từng bít, nếu nội dung của A là 00110101B, thì lệnh AND sau đây

```
ANL A, # 01010011B
```

Sẽ làm nội dung của A trở thành 00010001B, kết quả này được mô tả như sau

```
    01010011  (dữ liệu tức thời)
AND 00110101  (nội dung ban đầu của A)
    ________
    00010001  (kết quả chứa trong A)
```

Nhóm lệnh logic áp dụng cùng các cách định địa chỉ giống như nhóm lệnh số học,
do đó lệnh AND có thể viết như sau

ANL A, 55H (trực tiếp)
ANL A, @ R0 (gián tiếp)
ANL A, R6 (thanh ghi)
ANL A, # 33H (tức thời)

Tất cả các lệnh logic dùng A làm một trong các toán hạng có thời gian thực hiện là 1 chu kỳ máy còn các lệnh khác thì cần đến 2 chu kỳ.

Các lệnh logic có thể tác động lên 1 byte bất kỳ trong vùng nhớ trong mà không cần qua trung gian thanh ghi A, lệnh: XRL, addr, # data là cách dể dàng và nhanh chóng để đảo các bít của một port.

XRL P1, # 0FFH

Lệnh này thực hiện công việc đọc-sửa-ghi. Tám bít của port 1 được đọc sau đó từng bít được EXOR với từng bít tương ứng của dữ liệu tức thời, vì 8 bít dữ liệu đều có giá trị là 1 nên kết quả sẽ là đảo của các bít vừa đọc (A 1 = $\overline{A}$), sau đó kết quả sẽ được ghi trở lại vào port 1.

Các lệnh quay (RL A và RR A) sẽ dịch chuyển nội dung của A về phía trái hoặc phải 1 bít. Đối với lệnh quay trái bít MSB sẽ dịch về vị trí của bít LSB và đối với lệnh quay phải thì LSB lại dịch về vị trí của MSB. Lệnh RLC A và RRC A là lệnh quay 9 bít gồm nội dung của A và bít carry trong thanh ghi trạng thái PSW. VD: Bít carry là 1 và nội dung của A là 00H, lệnh

RRC A

Sẽ xóa carry và nội dung của A bây giờ là 80H, bít carry dịch vào bít **ACC.7** và bít **ACC.0** đi vào carry.

Lệnh SWAP A hoán chuyển vị trí giữa 4 bít cao với 4 bít thấp của bộ tích lũy, lệnh này thường dùng trong các phép tính với số BCD, VD: Nội dung của A là một số nhị phân có giá trị nhỏ hơn 100_{10}, giá trị này được chuyển thành số BCD như sau

MOV B, # 10
DIV AB
SWAP A
ADD A, B

Hai lệnh đầu tiên chia một số cho 10, hàng chục được chứa ở 4 bít thấp của A và hàng đơn vị trong thanh ghi B, lệnh SWAP và ADD sẽ chuyển hàng chục lên 4 bít cao của A và hàng đơn vị vào 4 bít thấp của A.

2.3 Nhóm lệnh truyền dữ liệu

RAM trong

Các lệnh truyền dữ liệu trong vùng nhớ bên trong được thực hiện trong 1 hoặc 2 chu kỳ máy, cú pháp như sau:

MOV <destination>, <source>

Dữ liệu được truyền đến vị trí bất kỳ trong phạm vi RAM trong và các thanh ghi chức năng đặc biệt mà không cần qua trung gian bộ tích lũy. Một điểm cần nhớ là 128 byte trên của RAM trong (8032/8052) chỉ được truy xuất bằng địa chỉ gián tiếp và các thanh ghi SFR chỉ được truy xuất bằng địa chỉ trực tiếp.

Một đặc điểm khác về cấu trúc giữa 8051 với hầu hết các bộ vi xử lý là vị trí của ngăn xếp ở bên trong RAM trong và kích thước của nó tăng dần lên vùng nhớ cao (có nghĩa là lên địa chỉ cao trong RAM). Lệnh PUSH trước tiên sẽ tăng con trỏ ngăn xếp SP, sau đó mới chuyển byte dữ liệu vào ngăn xếp. PUSH và POP chỉ dùng địa chỉ trực tiếp để xác định byte dữ liệu cần ghi hoặc lấy ra khỏi ngăn xếp, nhưng bản thân ngăn xếp thì lại được truy xuất bằng địa chỉ gián tiếp thông qua con trỏ ngăn xếp SP. Điều này có nghĩa là ngăn xếp có thể sử dụng 128 byte trên RAM trong của 8032/8052, 128 byte RAM trên này không có trong 8031/8051, với các vi điều khiển này nếu nội dung SP vượt quá 7FH (127) thì byte cần lưu trử sẽ bị mất và byte lấy ra từ ngăn xếp sẻ không xác định.

Các lệnh truyền dữ liệu còn có lênh MOV 16 bít dùng để khởi tạo con trỏ dữ liệu DPTR cho các bảng tìm kiếm trong bộ nhớ chương trình hoặc để truy xuất bộ nhớ dữ liệu 16 bít bên ngoài. Cú pháp lệnh

XCH A, <source>

Sẽ hoán chuyển giữa nội dung của bộ tích lũy với byte có địa chỉ là <source>, lệnh

hoán chuyển một ``digital`` có dạng như sau:

XCHD A, @R1

Lệnh này chỉ hoán chuyển 4 bít thấp. VD: Nội dung của A là F3H và R1 là 40H, tại ô nhớ RAM có địa chỉ 40H chứa giá trị 5BH thì sau khi lệnh được thực hiện kết quả nhận được nội dung của A sẽ là FBH và giá trị tại địa chỉ 40H là 53H.

RAM ngoài Các lệnh truyền dữ liệu giữa bộ nhớ trong với bộ nhớ ngoài áp dụng cách định địa chỉ gián tiếp. Địa chỉ gián tiếp được xác định bằng cách dùng địa chỉ một byte (@ Ri, trong đó Ri có thể là R0 hoặc R1 của dãy thanh ghi đâng hoạt động) hoặc địa chỉ 2 byte (@ DPTR). Khuyết điểm của địa chỉ 2 byte là phải dùng hết 8 bít của port 2 làm byte địa chỉ cao của bus địa chỉ và không thể dùng port 2 cho các yêu cầu xuất nhập khác. Ngược lại địa chỉ 1 byte cho phép truy xuất vài Kbyte RAM mà không cần dùng đến port 2.

Tất cả các lệnh truyền dữ liệu với bộ nhớ ngoài được thực hiện trong 2 chu kỳ máy và phải dùng bộ tích lũy là một trong hai toán hạng nguồn hoặc đích.

Các tín hiệu đọc, ghi bộ nhớ ngoài ($\overline{RD}$ và $\overline{WR}$) chỉ được tác động trong khi thực hiện lệnh MOVX, thông thường các tín hiệu này không tác động (mức HIGH). Nếu không dùng bộ nhớ ngoài thì các đường tín hiệu này được dùng như các đường I/O.

Bãng tìm kiếm

Có 2 lệnh truyền dữ liệu có khả năng đọc các bảng tìm kiếm trong bộ nhớ chương trình, vì các lệnh này chỉ truy xuất bộ nhớ chương trình nên các bảng tìm kiếm chỉ được phép đọc và không thể cập nhật. MOVC là dạng gợi nhớ ``move constant`` có thể dùng một trong hai bộ đếm chương trình hoặc con trỏ dữ liệu làm thanh ghi cơ sở và bộ tích lũy là địa chỉ offset. VD:

MOVC A, @ A + DPTR

Cho phép truy xuất một bảng có 256 điểm nhập được đánh số từ 0 đến 255. Số điểm nhập cần truy xuất được nạp vào bộ tích lũy và con trỏ dữ liệu được khởi tạo tại vị trí đầu bảng, lệnh

MOVC A, @ A + PC

Cũng hoạt động tương tự như lệnh trên, điểm khác là bộ nhớ chương trình được dùng làm địa chỉ cơ sở và bảng được truy xuất bằng một chương trình con. Trước tiên, số của điểm nhập cần thiết được nạp vào bộ tích lũy sau đó gọi chương trình con. Việc cài đặt và gọi được viết như sau:

```
    MOV A, ENTRY_NUMBER
    CALL LOOK_UP
    ……..
LOOK_UP: INC A
    MOVC A, @ A+ PC
   RET
TABLE:  DB data, data,…..
```

Bảng dữ liệu được đặt ngay sau lệnh RET trong bộ nhớ chương trình, lệnh INC là cần thiết vì thanh ghi sẽ trỏ đến lệnh RET khi thực hiện lệnh MOVC, việc tăng nội dung bộ tích lũy sẽ nhảy qua lệnh RET và đến vị trí của bảng.

2.4 Nhóm lệnh Boolean

Họ 8051 có đầy đủ các thao tác Boolean phục vụ các yêu cầu xử lý từng bít đơn lẻ. RAM trong có một vùng gồm 128 bít nhớ và trong vùng các thanh ghi chức năng đặc biệt cũng hỗ trợ 128 bít nhớ khác. Tất cả các đường vào – ra đều được định địa chỉ bít và mỗi đường được xem như một bít đơn riêng biệt. Các lệnh xử lý bít không chỉ là các lệnh nhảy có điều kiện mà còn bao gồm các lệnh truyền bít, đặt bít, xóa bít, đảo bít, OR và AND.

Chính các lệnh này là điểm mạnh của họ MCs 8051 mà không dễ dàng có được trong các cấu trúc khác với các lệnh xử lý byte.

Tất cả các bít đều được truy xuất bằng địa chỉ trực tiếp từ 00H-7FH trong 128 vị trí thấp và các địa chỉ bít từ 80H-FFH trong vùng các thanh ghi SFR, 128 vị trí thấp nằm trong các byte có địa chỉ từ 20H-2FH và được đanh số theo thứ tự từ bít 0 của byte 20H (bít 00H) đến bít 7 của byte 2FH (bít 7FH).

Các bít có thể được set hoặc clear bằng một lệnh đơn. Phương pháp điều khiển bít đơn rất thông dụng cho các thiết bị I/O. VD: Xuất ra một rơ le, động cơ, cuộn nam châm, LED, chuông, loa…hoặc nhập vào từ các loại công tắc hoặc các cảm biến trạng thái. Nếu một thiết bị báo động được nối đến bít 7 của port 1 thì thiết bị này có thể được tác động bằng lệnh set bít như sau:

SETB P1.7

Và tắt bằng lệnh xóa bít

CLR P1.7

Phần mềm hợp ngữ sẽ dịch ký hiệu P1.7 thành địa chỉ bít 97H, lệnh sau di chuyển một cờ vào chân của port một cách dễ dàng.

```
MOV C, FLAG
MOV P1.0, C
```

Trong ví dụ trên, FLAG là tên của một bít nhớ bất kỳ trong phạm vi 128 vị trí thấp của RAM trong hoặc trong vùng SFR, một đường I/O (trong trường hợp này là bít LSB của port 1) được set hay clear là tùy thuộc vào bít cờ bằng 1 hoặc 0.

Bít Cy trong thanh ghi trạng thái PSW được dùng như một bộ tích lũy bít trong các thao tác Boolean. Các lệnh xử lý bít dùng ký hiệu ``C`` để chỉ bít carry VD: CLR C, bít carry cũng có địa chỉ trực tiếp vì nó nằm trong thanh ghi PSW mà thanh ghi này được truy xuất từng bít giống như các địa chỉ bít khác trong các thanh ghi SFR, các bít trong PSW đều có dạng gợi nhớ thay cho địa chỉ trực tiếp, dạng gợi nhớ của carry là ``Cy`` được dùng thay cho địa chỉ bít 0D7H. Xét 2 lệnh sau đây:

```
CLR C
CLR Cy
```

Kết quả của hai lệnh giống nhau. Tuy nhiên, lệnh trên dài 1 byte và lệnh dưới dài 2 byte trong đó byte thứ hai là địa chỉ trực tiếp của bít chỉ định (bít carry).

Lưu ý là các lệnh Boolean bao gồm ANL (And bít) và ORL (OR bít) nhưng không có XRL (EXOR bít). VD để thực hiện phép EXOR 2 bít với nhau (BIT1 và BIT2) và chứa kết quả trong carry, cách thực hiện như sau:

```
MOV C, BIT1
JNB BIT2, SKIP
```

CPL C

SKIP: (continue)

Trước tiên, BIT1 được chuyển vào carry. Nếu bít 2 bằng 0 thì kết quả chứa trong C là đúng, đó là BIT1 BIT2 = BIT1 khi BIT2 = 0, nếu BIT2 = 1 thì đảo nội dung của C là kết quả đúng, việc đảo C sẽ hoàn tất phép EXOR.

Kiểm tra bít

Đoạn chương trình trong ví dụ trên sử dụng lệnh JNB là một trong các lệnh kiểm tra bít, các lệnh này sẽ nhảy nếu bít được định địa chỉ bằng 1 (JC, JB, JBC) hoặc nhảy nếu bít đang xét bằng 0 (JNC, JNB). Trong trường hợp trên nếu BIT2 = 0 thì lệnh CPL bị bỏ qua. JBC (jump if bit set then clear bit) sẽ nhảy nếu bít bằng 1 và sau đó xóa bít. Do đó, một cờ có thể được kiểm tra và xóa bằng một lệnh.

Tất cả các bít trong PSW đều được định địa chỉ trực tiếp nên bít parity hoặc các cờ công dụng chung khác đều chịu tác động của lênh kiểm tra bít.

2.5 Nhóm lệnh rẻ nhánh chương trình

Có nhiều lệnh cho phép rẽ nhánh chương trình bao gồm lệnh CALL và RETURN từ chương trình con hoặc các lệnh nhảy có hoặc không có điều kiện. Các khả năng này được mở rộng bằng 3 cách định địa chỉ áp dụng cho các lệnh rẽ nhánh chương trình. Có 3 loại lệnh JMP: SJMP, LJMP và AJMP tương ứng với 3 cách định địa chỉ theo thứ tự: Tương đối, dài và tuyệt đối. Phần mềm hợp ngữ ASM51 của Intel cho phép dùng dạng gợI nhớ tổng quát là JMP nếu người lập trình không quan tâm đến loại lệnh nào được sử dụng. Assembler của các hãng khác có thể không có đặc tính này, lệnh JMP sẽ được dịch ra là AJMP nếu đích đến không chứa tham chiếu thuận và phải ở trong phạm vi một khối 2K. Ngược lại sẽ được dịch là LJMP, lệnh CALL tổng quát cũng tương tự như vậy.

Lệnh SJMP xác định đích đến nhờ địa chỉ offset, vì vậy độ dài của lệnh là 2 byte (1 byte mã lệnh và 1 byte offset). Khoảng cách nhảy bị giới hạn trong phạm vi –128 byte đến +127 byte tính từ lệnh sau lệnh SJMP.

Lệnh LJMP có địa chỉ đích là một hằng số 16 bít, vì vậy lệnh dài 3 byte (1 byte mã lệnh + 2 byte địa chỉ) địa chỉ đích có vị trí bất kỳ trong bộ nhớ chương trình 64K.

Lệnh AJMP có địa chỉ đích là hằng số 11 bít, giống như SJMP lệnh này dài 2 byte nhưng được mã hóa khác, trong mã lệnh chứa 3 bít địa chỉ cao và byte thứ hai là địa chỉ thấp trong PC và 5 bít địa chỉ cao trong PC vẫn giữ nguyên. Do đó, đích đến phải ở trong cùng một khối 2K với lệnh sau lệnh AJMP. Vì bộ nhớ chương trình có 64K nên sẽ được chia thành 32 khối 2K với địa chỉ bắt đầu của mỗi khối là: 0000H, 0800H, 1000H, 1800H,…cho đến F800H.

Trong mọi trường hợp, người lập trình thường xác định địa chỉ nhảy bằng 1 nhãn hoặc một số 16 bít, phần mềm hợp ngữ sẽ đặt địa chỉ đích đúng

đúng theo yêu cầu của lệnh, nếu dạng thức cần thiết của lệnh không được hỗ trợ khoảng cách dùng để xác định địa chỉ đích thì sẽ xuất hiện thông báo lổi ``destination out of range``.

Bảng nhảy

Lệnh **JMP @ A + DPTR** hổ trợ lệnh nhảy phụ thuộc trường hợp cụ thể đối với các bảng nhảy. Địa chỉ đích được tính ra trong khi chạy chương trình và là tổng của thanh ghi DPTR với Thanh ghi tích lũy. Thanh ghi DPTR được nạp địa chỉ của bảng nhảy và bộ tích lũy là một chỉ số. VD: Nếu có 5 trường hợp được yêu cầu thì một giá trị từ 0 đến 4 sẽ được nạp vào A và một bước nhảy được thực hiện như sau:

```
MOV DPTR, # JUMP_TABLE
MOV A, # INDEX_NUMBER
RL A
JMP @ A + DPTR
```

Lệnh RL A biến chỉ số (0 đến 4) thành các số chẳn từ 0 đến 8 bởi vì mỗi một điểm nhập trong bảng nhảy là địa chỉ 2 byte

```
JUMP_TABLE: AJMP CASE0
AJMP CASE1
AJMP CASE2
AJMP CASE3
```

Chương trình con và ngắt

Có 2 loạI lệnh CALL: ACALL và LCALL sử dụng địa chỉ tuyệt đối và địa chỉ dài, cũng giống như JMP, dạng tổng quát CALL trong phần mềm hợp ngữ Intel thường được dùng khi người lập trình không cần quan tâm đến cách mã hóa địa chỉ. Các lệnh này sẽ cất nội dung của PC vào ngăn xếp trước khi nạp địa chỉ được chỉ ra trong câu lệnh vào bộ đếm chương trình PC, nội dung của PC chính là địa chỉ của lệnh tiếp theo sau lệnh CALL, byte thấp của địa chỉ này được cất trước và tiếp theo là byte cao, khi lấy ra thì thứ tự sẻ ngược lại. VD nếu lệnh CALL trong bộ nhớ chương trình được đặt tại địa chỉ 1000H-1002H và nội dung SP là 20H thì lênh CALL sẽ: (a) cất địa chỉ trở về 1003H vào ngăn xếp: Giá trị 03H vào vị trí 21H và 10H vào 22H, (b) tăng SP lên 22H và (c) nhảy đến vị trí của chương trình con bằng cách nạp địa chỉ trong byte 2 và 3 của lệnh vào PC.

Lệnh LCALL và ACALL có cùng giới hạn địa chỉ đích giống như AJMP và LJMP, các chương trình con được kết thúc bằng lệnh RET để quay về thực hiện lệnh tiếp theo sau lệnh CALL, không có điều gì huyền bí về cách lệnh RET trở về chương trình chính, chỉ đơn giản là lệnh này ``pop`` 2 byte cuối cùng trong ngăn xếp và nạp vào PC, một quy tắc cần nhớ khi lập trình với chương trình con là chỉ nhảy vào chương trình con bằng lênh CALL và thoát ra bằng lệnh RET, việc vào và ra chương trình con bằng phương pháp khác thường chỉ dẫn đến rối loạn ngăn xếp và làm treo máy.

Lệnh RETI được dùng để trở về từ chương trình phục vụ ngắt (ISR) điểm khác nhau giữa RET và RETI là RETI sẽ báo cho hệ thống điều khiển ngắt biết là yêu cầu ngắt đã được thực hiện xong nếu không có một ngắt nào tồn tại trong khi RETI đang thực hiện, ngoài ra các chức năng khác cũng giống như RET

Nhảy có điều kiện 8051 cung cấp một số lệnh nhảy có điều kiện. Tất cả các lệnh này xác định địa chỉ đích bằng địa chỉ tương đối và khoảng cách nhảy trong phạm vi –128 đến +127 byte tính từ lênh tiếp theo sau lệnh nhảy có điều kiện. Tuy nhiên, người lập trình thường chỉ cần xác định địa chỉ đích bằng nhãn hoặc hằng số 16 bít và phần mềm assembler sẽ thực hiện các công việc còn lại.

Trong thanh ghi PSW không có bít zero , lệnh JZ và JNZ sẽ dùng nội dung của bộ tích lũy để làm điều kiện nhảy.

Lệnh DJNZ (decrement and jump if not zero) được dùng trong cấu trúc vòng lặp, để thực hiện một vòng lặp N lần, số N này sẽ được nạp vào một byte đếm và kết thúc vòng lặp bằng DJNZ chỉ đến nơi bắt đầu vòng lặp. VD: N = 10.

```
MOV R7, # 10
LOOP: (bắt đầ vòng lặp)
…….
(kết thúc vòng lặp)
DJNZ R7, LOOP
(tiếp tục)
```

Lệnh CJNE (compare and jump if not equal) cũng được dùng cho cấu trúc vòng lặp, bước nhảy chỉ được thực hiện nếu 2 byte trong vùng toán hạng của lệnh bằng nhau.

VD: Một ký tự được nhập vào A từ cổng nối tiếp và yêu cầu là chương trình sẽ nhảy về nhãn TERMINATE nếu ký tự nhập vào là CONTROL-C (03H), cách thực hiện như sau:

```
CJNE A, #30H, SKIP
SJMP TERMINATE
SKIP: (tiếp tục)
……..
```

Vì việc nhảy chỉ xảy ra nếu A khác CONTROL-C nên nhãn SKIP được dùng để bỏ qua lệnh SJMP trừ khi ký tự đọc vào đúng yêu cầu

Một ứng dụng khác của lệnh này là yêu cầu so sánh lớn hơn và nhỏ hơn, 2 byte trong vùng toán hạng là các số nguyên không dấu. Nếu số thứ nhất nhỏ hơn số thứ hai thì Cy = 1 và ngược lại thì Cy = 0. VD: Chương trình sẻ nhảy đến nhãn BIG nếu nội dung của A lớn hơn hoặc bằng 20H

CJNE A, #20H, $ + 3
JNC BIG

Đích nhảy của CJNE là $ + 3, phần mềm hợp ngữ dùng ký hiệu $ để biểu diễn địa chỉ của lệnh hiện hành. Do CJNE là lệnh 3 byte nên $ + 3 là địa chỉ của lệnh kế tiếp JNC.

Nói cách khác lệnh CJNE theo sau bởi lệnh JNC không quan tâm đến kết quả của phép so sánh, mục đích của phép so sánh là để set hoặc clear Cy và Cy là điều kiện quyết định của JNC.

THỰC HÀNH LUYỆN TẬP

1. Trình bày cách truyền nội dung tại địa chỉ RAM trong 50H vào thanh ghi A bằng địa chỉ gián tiếp.
2. Mã lệnh nào không được định nghĩa trong họ 8051 ?
3. Cho lệnh sau đây

MOV 50H, #0FFH

a) Mã lệnh của lệnh này là gì ?
b) Lệnh này dài bao nhiêu byte ?
c) Giải thích ý nghĩa của từng byte trong mã lệnh.
d) Lệnh này được thực hiện bằng bao nhiêu chu kỳ máy ?
e) Cho biết thời gian thực hiện lệnh này, giả sử tần số thạch anh là 12 MHz
4. Xem lệnh sau
AJMP AHEAD
Lệnh này đặt tại địa chỉ 2FF0H và 2FF1H và nhãn AHEAD đặt tại địa chỉ 2F96H. Mã lệnh dang hexa của lệnh này là gì ?
5. Viết đoạn chương trình thực hiện yêu cầu nhảy đến đoạn EXIT khi nội dung
của A là mã ASCII carrige return.
6. Cho lệnh sau:
SJMP BACK
Đặt tại địa chỉ 0100H, nhãn BACK tại địa chỉ 00AEH, cho biết mã hexa của lệnh này ?
7. Lệnh sau đây làm gì ?

SETB 0D7H

Có cách nào thực hiện yêu cầu này tốt hơn ? Tại sao ?

8. Có gì khác nhau giữa hai lệnh sau:

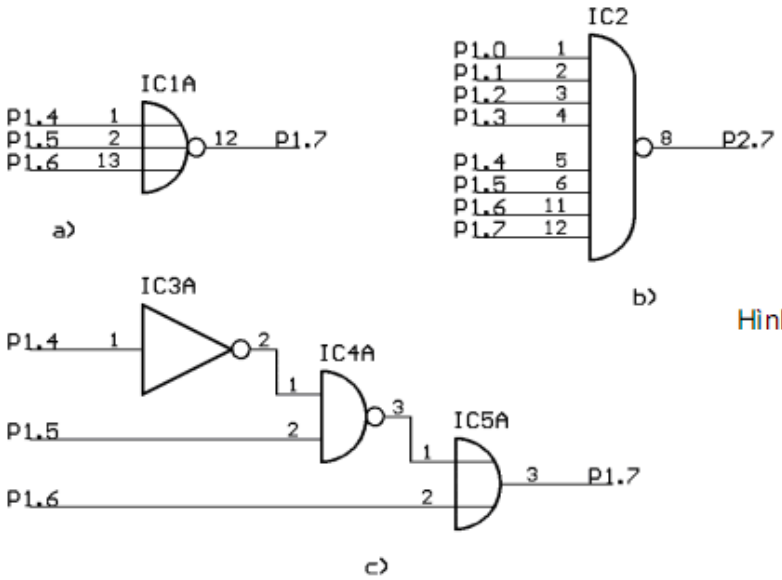
INC A
INC ACC

9. Viết lệnh tạo một mức logic thấp trong vòng 5 µS tại chân P1.7. Giả sử P1.7 đang ở mức cao và tần số thạch anh là 12 MHz

10. Viết chương trình tạo xung vuông tần số 83,3 KHz tại P1.0, giả sử tần số thạch anh là 12 MHz

11. Viết chương trình tạo mức cao 4 µS trong mỗi 200 µS tại P1.7

12. Viết chương trình thực hiện các thao tác logic vẽ ở hình 2.4



Hình 2.4 Các mạch logic
a) NOR 3 ngỏ vào
b) NAND 7 ngỏ vào
c) Mạch tổ hợp

13. Trong hình (a) của câu trên, cho biết thời gian trì hoãn tính từ lúc tín hiệu vào chuyển trạng thái đến lúc tín hiệu ra chuyển trạng thái.

14. Giả sử bộ nhớ trong của 8051 trước khi thực hiện lệnh RET như sau:

Địa chỉ	Nội dung	SFR	Nội dung
0B	9A	SP	0B
0A	78	PC	0200
09	56	A	55
08	34		
07	12		

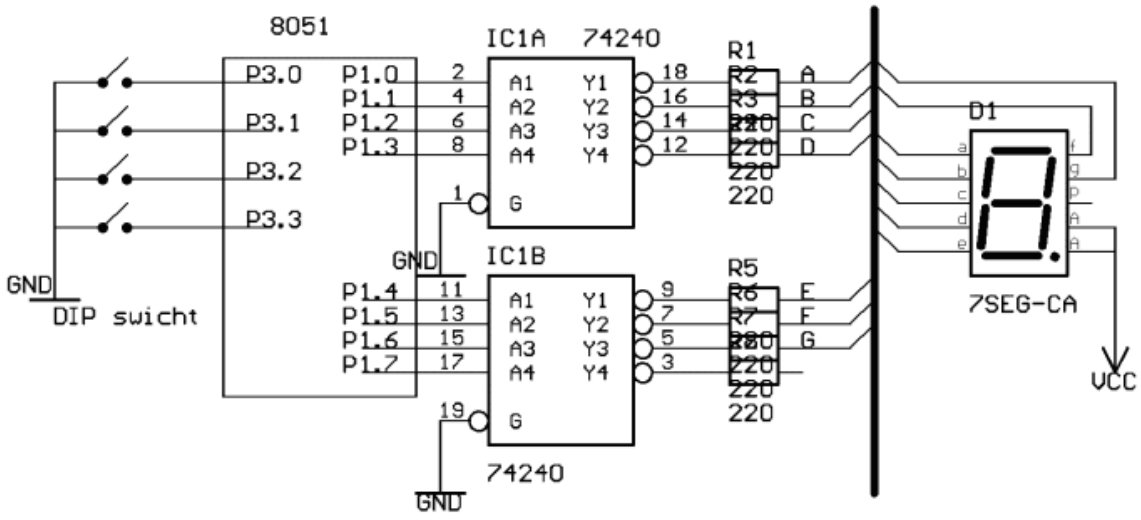
Nội dung PC là bao nhiêu sau khi thực hiện lệnh RET ?

15. Xét chương trình con sau đây

SUB: MOV R0, #20H
LOOP: MOV @R0, #0
INC R0
CJNE R0, #80H, LOOP
RET

a) Cho biết chức năng của chương trình con này ?
b) Mỗi lệnh cần bao nhiêu chu kỳ máy ?
c) Độ dài mỗi lệnh là bao nhiêu byte ?
d) Viết lại chương trình này bằng mã máy.
e) Thời gian thực hiện chương trình là bao nhiêu ?

16. Một công tắc DIP điều khiển LED 7 đoạn bằng 8051 như sơ đồ ở hình 2.5.
Viết chương trình giải mã số nhị phân 4 bít từ công tắc DIP sang số hexa hiển thị trên LED 7 đoạn. VD: Dữ liệu đọc từ DIP là 1100B thì LED sẽ hiển thị ký hiệu ``C``



Hình 2.5 Giải mả BIN → LED 7 đoạn

CHƯƠNG 2
HOẠT ĐỘNG CỦA BỘ ĐỊNH THỜI

MÃ BÀI: MH30-03

Mục tiêu:

- Sử dụng được các lệnh của thanh ghi Timer định thời gian chính xác
- Sử dụng được các lệnh của thanh ghi Counter đếm sự kiện bên ngoài
- Tự tin trong thao tác sử dụng bộ định thời.
- Tính chính xác, tỉ mỉ, cẩn thận.

Nội dung chính:

1. Giới thiệu bộ định thời

Mục tiêu:

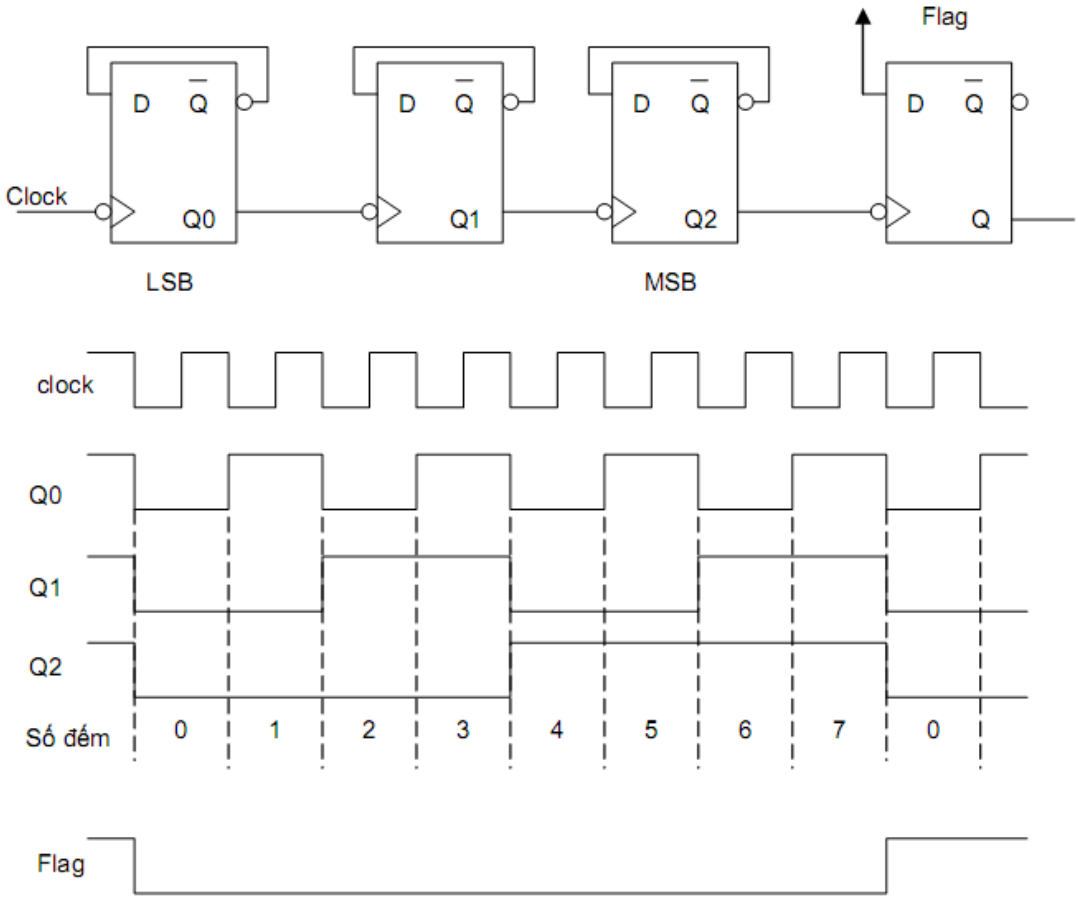
- Trình bày được các bộ định thời của vi điều khiển.

Bộ định thời (timer) là một mạch chia gồm nhiều FF ghép lại với nhau với tín hiệu vào là xung đồng hồ, xung này được chia 2 ở tầng FF đầu tiên, sau đó lại được đưa đến các tầng tiếp theo, với n tầng FF số chia sẻ là 2^n, ngõ ra của tầng cuối cùng là FF tràn của bộ định thời còn gọi là cờ tràn, cờ này được kiểm tra bởi phần mềm và có thể tạo ra một ngắt. Trị nhị phân của các

FF chính là số xung đồng hồ đếm được (hoặc số sự kiện) kể từ khi bộ định thời được khởi động. VD một timer 16 bít có thể đếm được từ 0000H đến FFFFH, cờ tràn sẽ bị set khi có tràn từ FFFFH đến 0000H.

Hoạt động của một bộ định thời đơn giản được mô tả ở hình 4.1 đây là một bộ định thời 3 bít, mỗi tầng là 1 D-FF tác động cạnh âm với chức năng chia 2 (ngõ ra $\overline{Q}$ nối đến ngõ vào D), FF cò là một chốt loại D, cờ này được set bởi tầng cuối cùng của bộ định thời. Giản đồ thời gian ở hình 4.1b cho thấy tầng đầu tiên Q0 chia 2 tần số xung đồng hồ, tầng thứ nhì chia 4 và cứ thế tiếp tục. Số đếm được biểu diễn bằng số thập phân để dễ thay đổi bằng cách kiểm tra trạng thái của 3 FF. VD số đếm là 4 khi Q2 = 1; Q1 = 0; Q0 = 0 ($4_{10} = 100_2$)

Bộ định thời được áp dụng trong hầu hết các ứng dụng hướng điều khiển và timer trong 8051 cũng không ngoại lệ. 8051 có 2 bộ định thời 16 bít với 4 chế độ làm việc khác nhau. Trong 8052 còn có thêm timer thứ ba 16 bít với 3 chế độ làm việc. Công dụng của timer là (a) đếm thời gian (định thời) và (b) đếm sự kiện hoặc (c) tạo tốc độ Baud cho cổng nối tiếp bên trong 8051. Vì mỗi bộ định thời gồm 16 bít nên tầng cuối cùng thứ 16 sẽ chia tần số xung đồng hồ cho $2^{16} = 65.536$.



Hình 3.1 Bộ đếm 3 bít

Trong ứng dụng định thời, một bộ định thời được lập trình để set cờ tràn của nó tương ứng với khoảng thời gian đặt trước, cờ này được dùng để đồng bộ hóa chương trình nhằm thực hiện một thao tác nào đó như là: Kiểm tra trạng thái các ngõ vào hoặc gởi dữ liệu đến các ngõ ra, các ứng dụng khóa có thể dùng xung đồng hồ chuẩn của timer để đo thời gian giữa hai thời điểm VD: Đo độ rộng xung.

Khả năng đếm sự kiện được dùng để xác định số lần xảy ra của một sự kiện thay vì đo khoảng thời gian giữa các sự kiện. Một sự kiện là một tác nhân kích thích bên ngoài nào đó tạo ra một sự chuyển tiếp từ 1 xuống 0 tại 1 chân của 8051. Các bộ định thời cũng được dùng để tạo ra một đồng hồ tốc độ baud cho cổng nối tiếp tích hợp bên trong 8051.

2. Thanh ghi chế độ Timer (TMOD)

Mục tiêu:

- Sử dụng được thanh ghi chế độ Timer TMOD .

Thanh ghi TMOD gồm hai nhóm 4 bít có chức năng chọn chế độ làm việc cho hai timer 0 và 1 (bảng 3.2 và 3.3).

BẢNG 3.1
Các thanh ghi SFR của timer

SFR của timer	Chức năng	Địa chỉ	Định địa chỉ bít
TCON	Điều khiển	88H	Có
TMOD	Chế độ	89H	Không
TL0	Byte thấp timer 0	8AH	Không
TL1	Byte thấp timer 1	8BH	Không
TH0	Byte cao timer 0	8CH	Không
TH1	Byte cao timer 1	8DH	Không
T2CON*	Điều khiển timer 2	C8H	Có
RCAP2L*	Capture byte thấp	CAH	Không
RCAP2H*	Capture byte cao	CBH	Không
TL2*	Byte thấp timer 2	CCH	Không
TH2*	Byte cao timer 2	CDH	Không

* Chỉ có trong 8032 và 8052

BẢNG 4.2 Thanh ghi TMOD

Bít	Tên	Timer	Mô tả
7	GATE	1	GATE = 1, timer chỉ hoạt động trong khi $\overline{INT1} = 1$
6	C/$\overline{T}$	1	Bít chọn nguồn xung kích 0 = định thời 1 = đếm sự kiện
5	M1	1	Mode bit 1 (bảng 4.3)
4	M0	1	Mode bít 0 (bảng 4.3)
3	GATE	0	Timer 0 GATE bít
2	C/$\overline{T}$	0	C/$\overline{T}$ bít timer 0
1	M1	0	M1 bít timer 0
0	M0	0	M0 bít timer 0

BẢNG 3.3 Các chế độ làm việc

M1	M0	Chế độ	Mô tả
0	0	0	Timer 13 bít (8048 mode)
0	1	1	Timer 16 bít
1	0	2	Timer 8 bít tự động nạp lại
1	1	3	Timer phân biệt *Timer 0:* a) TL0 là timer 8 bít được điều khiển bởi các mode bít timer 0 b) TH0 là timer 8 bít được điều khiển bởi các mode bít timer 1 *Timer 1*: Dừng

TMOD không được định địa chỉ bít mà thực ra điều này cũng không cần thiết.

Thông thường thanh ghi này được khởi tạo ngay khi bắt đầu chương trình. Sau đó timer có thể dừng, hoặc hoạt động thông qua các thanh ghi SFR khác.

3. Thanh ghi điều khiển Timer (TCON)

Mục tiêu:

- Sử dụng được thanh ghi điều khiển Timer TCON vào lập trình ứng dụng.

Thanh ghi TCON gồm các bít trạng thái và các bít điều khiển cho hai bộ định thời 0 và 1 (bảng 4.4). Trong đó 4 bít interrupt (TCON.0-TCON.3) sẽ được bàn đến trong bài về Interrupt.

BẢNG 3.4 Thanh ghi điều khiển TCON

Bít	Ký hiệu	Địa chỉ bít	Mô tả
TCON.7	TF1	8FH	Cờ tràn timer 1. Set bằng phần cứng khi tràn số đếm, Clear bằng phần mềm hoặc bằng phần cứng khi xử lý ngắt
TCON.6	TR1	8EH	Bít khởI động timer 1. Set và Clear bằng phần mềm để khởI động hoặc dừng timer
TCON.5	TF0	8DH	Cờ tràn timer 0
TCON.4	TR0	8CH	Bít khởI động timer 0
TCON.3	IE1	8BH	Cờ tác động cạnh ngắt 1 bên ngoài. Set bằng phần cứng tạI cạnh xuống của ngỏ $\overline{INT1}$. Clear bằng phần mềm hoặc bằng phần cứng khi xử lý ngắt
TCON.2	IT1	8AH	Ngắt 1 bên ngoài. Set và Clear bằng phần mềm tác động cạnh xuống hoặc mức thấp
TCON.1	IE0	89H	Cờ cho phép ngắt 0 bên ngoài
TCON.0	IT0	88H	Cờ chọn kiểu ngắt 0 bên ngoài

4. Các chế độ Timer

Mục tiêu:

- Trình bày được các chế độ Timer.

Vì 8051 có hai bộ định thời nên ký hiệu ``x`` được dùng để ám chỉ một trong hai bộ định thời này, do đó THx có thể là TH1 hoặc TH0 tùy theo timer Thứ tự sắp xếp các thanh ghi TLx, THx và cờ tràn TFx được mô tả ở hình 4.2 theo
từng chế độ làm việc.

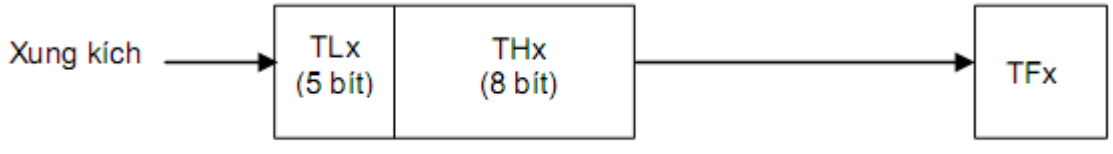
4.1 Chế độ timer 13 bít (mode 0)

Chế độ này nhằm tương thích với họ vi điều khiển trước của 8051 là 8048 (hình 4.2a). Trong chế độ này thanh ghi định thời byte cao THx được nối tiếp với 5 bít thấp của thanh ghi TLx để tạo thành bộ định thời 13 bít, 3 bít cao của TLx không dùng.

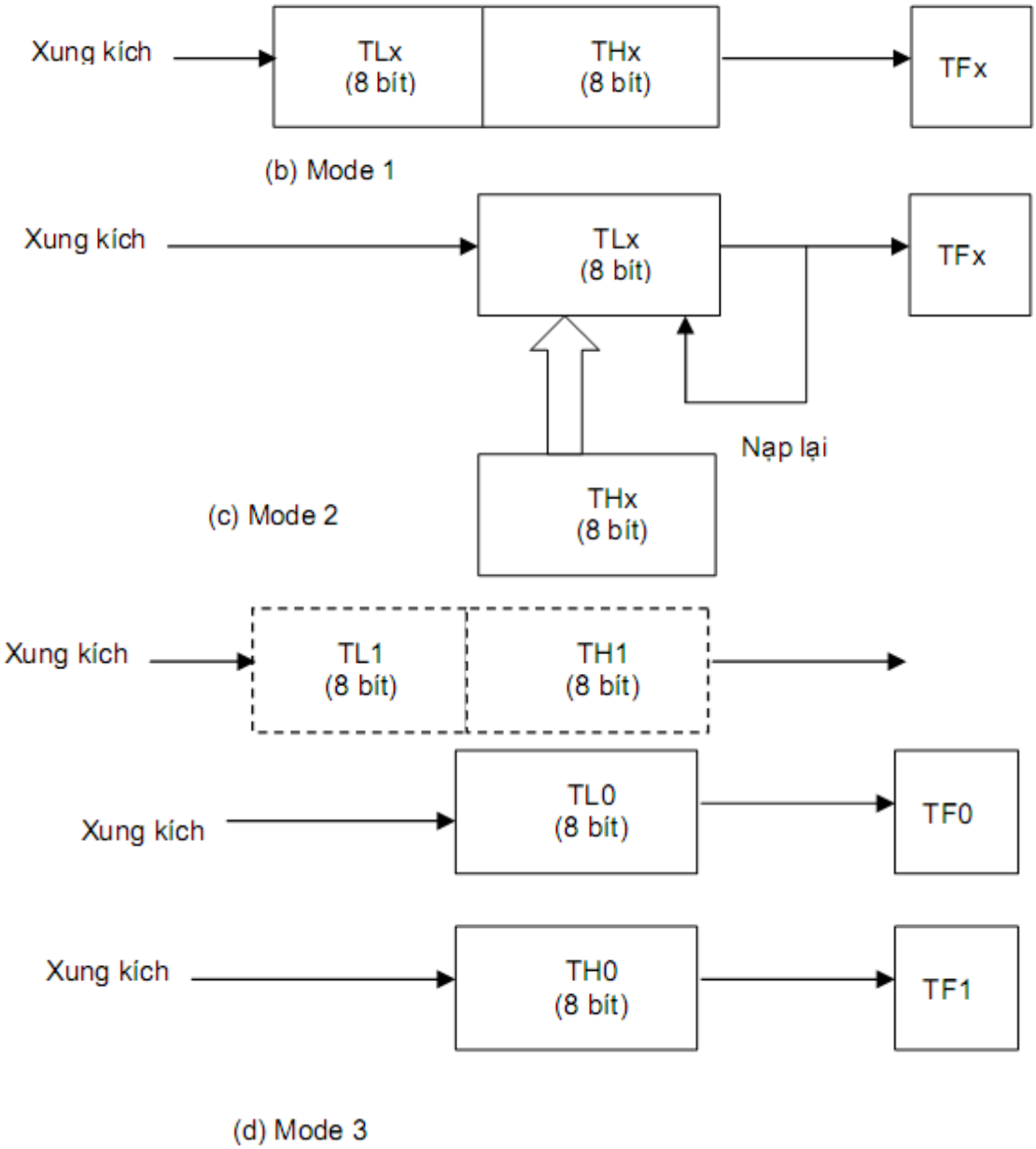
4.2 Chế độ timer 16 bít (mode 1)

Cũng giống như mode 0, xung đồng hồ được đưa vào cặp thanh ghi định thời TLx/THx, khi có xung đồng hồ timer sẽ đếm lên từ 0000H, 0001H, 0002H…Hiện tượng tràn sẽ xảy ra khi số đếm từ giá trị FFFFH chuyển sang 0000H và sẽ làm cờ tràn bị set và timer tiếp tục đếm.

Cờ tràn là biến TFx trong thanh ghi điều khiển TCON, cờ này được đọc và ghi bằng phần mềm (hình 4.2b). Bít có giá trị cao nhất của timer là bít 7 trong thanh ghi THx và bít thấp nhất là bít 0 trong thanh ghi TLx, cặp thanh ghi định thời TLx/THx có thể được đọc hoặc ghi bằng phần mềm tại bất kỳ lúc nào.



(a) Mode 0



Hình 3.2 Các chế độ làm việc

4.3 Chế độ tự nạp lại 8 bít (mode 2)

Trong chế đọ này thanh ghi TLx là một bộ định thờI 8 bít trong khi đó thanh ghi THx chứa giá trị cần nạp lại. Khi số đếm tràn từ FFH đến 00H, lúc này không chỉ cờ tràn bị set mà giá trị của THx sẻ được nạp lại vào TLx và quá trình đếm vẫn tiếp tục cho đến lần tràn tiếp theo. Chế độ này rất tiện lợi do việc tràn xảy ra sau một khoảng thời gian xác định lặp lại theo chu kỳ mỗi khi TMOD và THx được khởi tạo (hình 4.2c)

4.4 Chế độ tách biệt timer (mode 3)

Ảnh hưởng của chế độ này lên hai bộ định thời không giống nhau. Timer 0 thì được tách ra làm hai timer 8 bít đó là TL0 và TH0 hoạt động độc lập với nhau với hai cờ tràn tương ứng là TF0 cho TL0 và TF1 cho TH0.

Timer 1 dừng trong chế độ này nhưng có thể được khởi động bằng cách chuyển sang các chế độ khác, chỉ có một hạn chế là cờ tràn TF1 không bị set khi timer 1 tràn vì cờ này đã được nối đến TH0.

Mode 3 được dùng chủ yếu để tạo ra thêm một timer 8 bít thứ ba trong 8051. Timer 1 có thể được điều khiển ON/OFF bằng cách chuyển qua lại giữa mode 3 và các mode khác và có thể được dùng để tạo tốc độ baud cho cổng nối tiếp của 8051 hoặc những yêu cầu không cần đến ngắt.

5. Nguồn tạo xung nhịp

Mục tiêu:

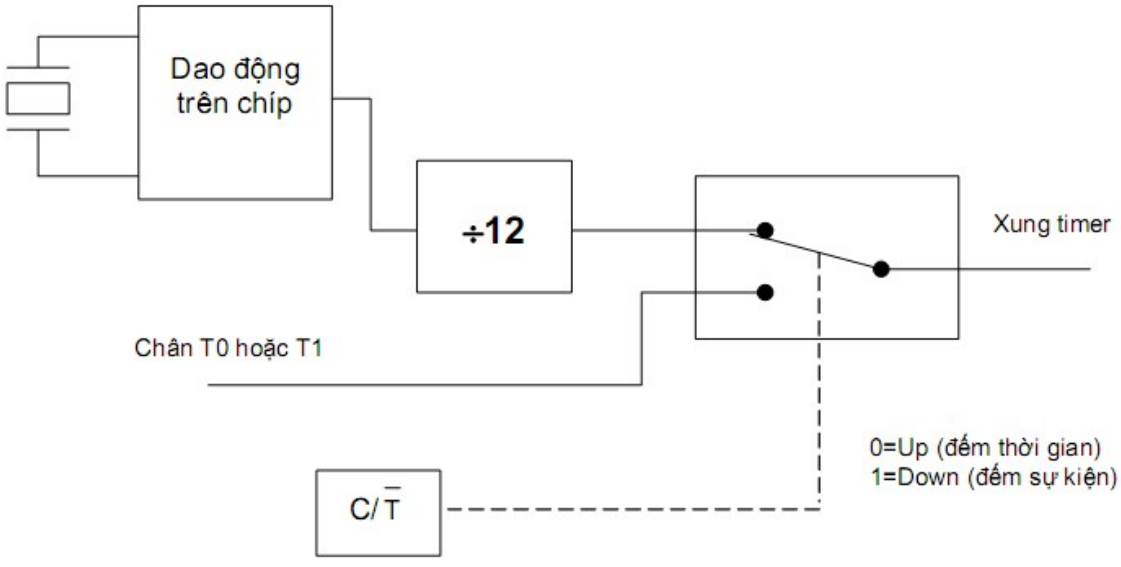
- Trình bày được các chức năng của bộ định thời.

Hình 4.2 không cho biết bộ định thời được kích như thế nào, có hai khả năng kích cho timer các khả năng này được chọn bằng cách viết vào bít $C/\overline{T}$ trong thanh ghi TMOD khi khởi tạo timer. Một nguồn xung đồng hồ dùng cho chức năng định thời và một nguồn xung khác phục vụ cho chức năng đếm sự kiện.

5.1 Chức năng định thời (đếm thời gian)

Nếu bít $C/\overline{T} = 0$ thì bộ định thời hoạt động như một bộ đếm thời gian với xung vào là xung đồng hồ trong chip, một mạch chia 12 được thêm vào để giảm tần số xung đồng hồ đến giá trị phù hợp với hầu hết các ứng dụng.

Ở chế độ này bộ định thời được dùng để đếm thời gian, cặp thanh ghi TLx/THx tăng dần với tốc độ là 1/12 tần số xung dao động trên chip, nếu tần số thạch anh là 12 MHz thì tốc độ xung đếm là 1 MHz, hiện tượng tràn xảy ra sau một số xung nhất định phụ thuộc vào giá trị khởi tạo của TLx/THx.



Hình 3.3 Nguồn xung đếm

5.2 Chức năng đếm sự kiện

Nếu bít C/$\overline{T}$ = 1 thì bộ định thời sẽ được kích từ bên ngoài bởi một xung xảy ra từ một sự kiện, số sự kiện được xác định bởi phần mềm bằng cách đọc nội dung cặp thanh ghi TLx/THx vì nội dung này tăng lên theo từng sự kiện.

Xung kích bên ngoài được đưa vào chân đa chức năng của port 3 đó là bít 4 của port 3 (P3.4) nhận xung kích cho timer 0 và được ký hiệu là T0, chân P3.5 được ký hiệu là T1 là nơi nhận xung kích của timer 1 (hình 4.3)

Ở chế độ đếm sự kiện, nộI dung thanh ghi định thời tăng lên tại cạnh xuống của xung kích ngoài Tx, xung kích ngoài được lấy mẫu trong khoảng thời gian S5P2 của mỗi chu kỳ máy. Do đó, khi ngõ vào ở mức cao trong một chu kỳ và ở mức thấp trong chu kỳ kế tiếp thì số đếm được sẽ tăng lên, giá trị mới của bộ đếm xuất hiện trong khoảng thời gian S3P1 của chu kỳ kế tiếp chu kỳ nhận ra sự chuyển tiếp, vì vậy phải cần đến 2 chu kỳ (2 μS) để nhận dạng quá trình chuyển từ 1 xuống 0, suy ra tần số xung kích ngoài cao nhất là 500 KHz tương ứng với tần số thạch anh 12 MHz.

6. Khởi động điều khiển và truy xuất thanh ghi Timer

Các timer thường được khởi tạo một lần mỗi khi bắt đầu chương trình để thiết lập đúng chế độ làm việc. Sau đó, trong phần thân của chương trình các timer được khởi động, dừng, kiểm tra và xóa cờ tràn, đọc và cập nhật các thanh ghi…theo yêu cầu của ứng dụng.

TMOD là thanh ghi đầu tiên được khởi tạo vì nó xác định chế độ làm việc. VD: Các lệnh sau đây sẽ thiết lập timer 1 là timer 16 bít (mode 1), và xung đếm là xung dao động hệ thống (định thời)

MOV TMOD, #00010000B

Kết quả của lệnh này làm cho bít M1 = 0 và M0 = 1 (mode 1), C/$\overline{T}$ = 0 và GATE = 0 (xung kích bên trong) và xóa mode bít của timer 0 (bảng 4.2)

Dĩ nhiên, timer chỉ thực sự hoạt động khi bít TR1 = 1, nếu cần một số đếm ban đầu thì phải khởi tạo cặp thanh ghi TL1/TH1, cần nhớ là timer luôn đếm lên và cờ tràn sẽ bằng 1 khi số đếm chuyển từ FFFFH xuống 0000H, để tạo thời gian định thời là 100 μS. thì phải khởi tạo số đếm nhỏ hơn 0000H một trị là +100, giá trị đúng là –100 hoặc FF9CH, các lệnh thực hiện như sau:

MOV TL1, #9CH

MOV TH1, #0FFH

Sau đó khởi động timer

SETB TR1

Cờ tràn tự động bằng 1 sau 100 μS, phần mềm có thể đợi trong một vòng lặp 100 μS bằng lệnh nhảy tại chổ có điều kiện khi cờ tràn vẫn chưa được set

WAIT: JNB TF1, WAIT

Khi timer tràn thì cần thiết phải dừng timer và xóa cờ tràn bằng phần mềm

CLR TR1

```
CLR  TF1
```

6.1 Đọc thời gian đang hoạt động

Trong một vài trường hợp cần phải đọc các thanh ghi định thời của timer trong khi timer đang hoạt động, việc làm này có thể gặp một sai số nếu như giữa hai lần đọc nội dung byte thấp TLx và byte cao THx lại xảy ra hiện tượng tràn số từ byte thấp lên byte cao. Cách giải quyết là trước tiên đọc byte cao sau đến byte thấp và tiếp theo lại đọc byte cao một lần nữa, nếu byte cao bị thay đổi thì lặp lại quá trình đọc từ đầu, đoạn mã sau đây đọc nội dung TLx/THx và đưa vào R6/R7

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

6.2 Thời gian ngắn và thời gian dài

Một câu hỏi là 8051 có thể định thời trong khoảng thời gian bao lâu ? Giả sử tần số thạch anh là 12 MHz do đó xung kích cho bộ định thời là 1 MHz

Khoảng thời gian định thời ngắn nhất không phụ thuộc tần số xung kích timer mà phụ thuộc vào phần mềm, có nghĩa là do thời gian thực hiện các lệnh, lệnh có thời gian thực hiện nhanh nhất của 8051 là một chu kỳ máy hoặc 1 µS. Bảng 4.5 tóm tắt phương pháp tạo các thời gian định thời khác nhau.

BẢNG 3.5

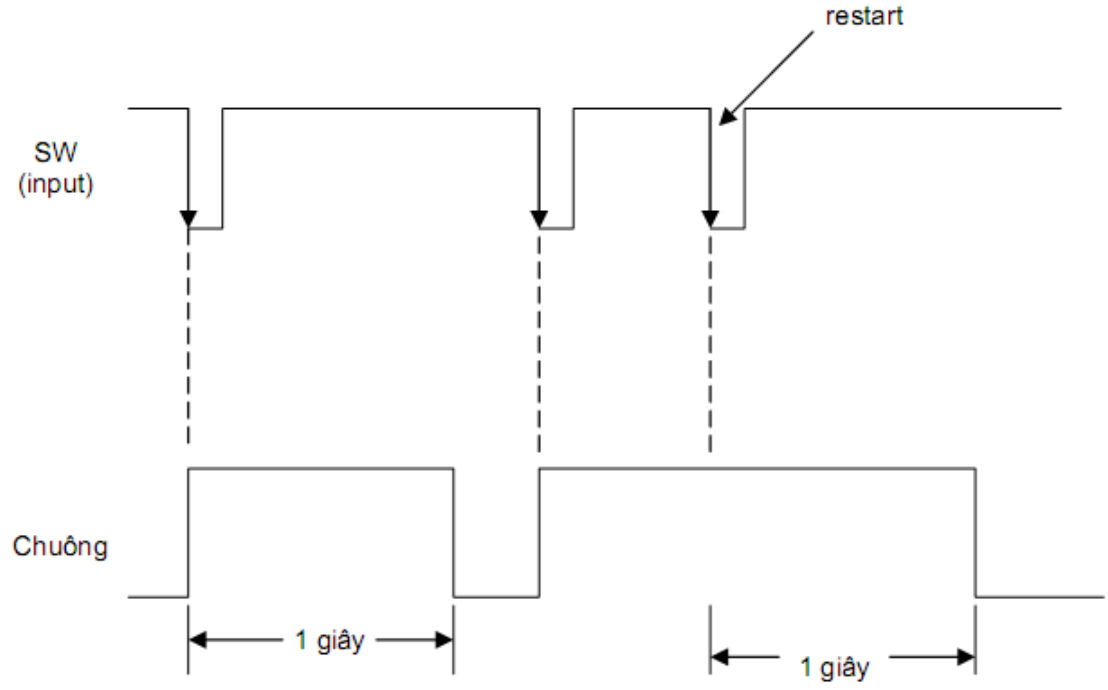
Cách tạo thờI gian (tần số XTAL 12 MHz)

Trị tốI đa (µS)	Phương pháp
≈ 10	Phần mềm
256	Timer 8 bít tự nạp lạI
65.536	Timer 16 bít
Không giớI hạn	Timer 16 bít kết hợp vòng lặp

THỰC HÀNH LUYỆN TẬP

1. Viết chương trình điều khiển 8051 tạo xung vuông tại P1.5 có tần số 100 KHz
(lưu ý: Không dùng timer).

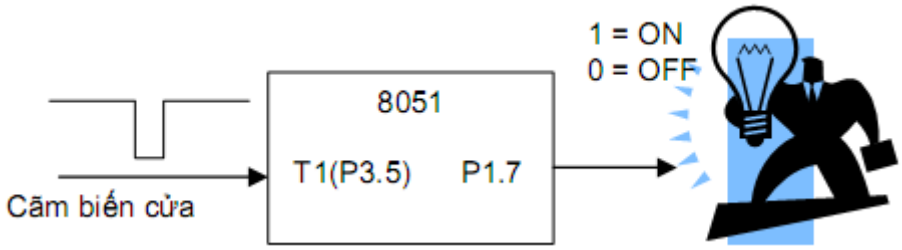
2. Viết lại lời giải ở ví dụ 4-4 bằng cách thêm chế độ restart, nếu ở ngõ vào lại xuất hiện sườn xuống trong khi chuông đang reo, thì vòng lặp định thời sẽ restart lại từ đầu để kéo dài thêm 1S khác như mô tả ở hình 3.10.



Hình 3.10 Mạch chuông cải tiến

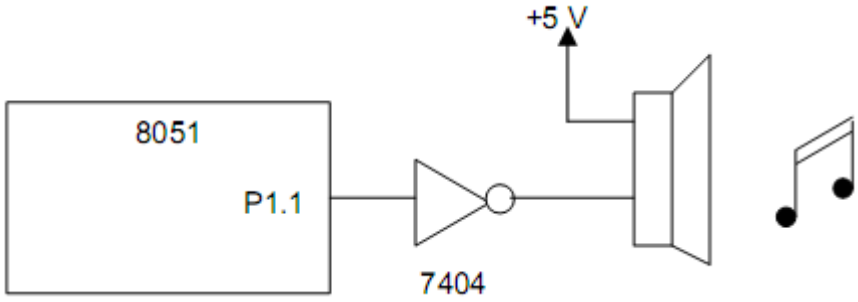
3. Viết chương trình tạo xung vuông 12 KHz tại P1.2 dùng timer 0.

4. Thiết kế ứng dụng ``cửa xoay`` dùng timer 1 để khách hàng thứ 10.000 đã đi vào một khu chợ trời. Giả sử (a) cãm biến cửa xoay được nối đến ngõ vào T1 và sẽ tạo ra một xung mỗi khi có 1 người đi vào và (b) một đèn báo được nối đến P1.7 và sẽ sáng khi P1.7 = 1 (hình 3.11)



Hình 3.11 Mạch cảnh báo cửa

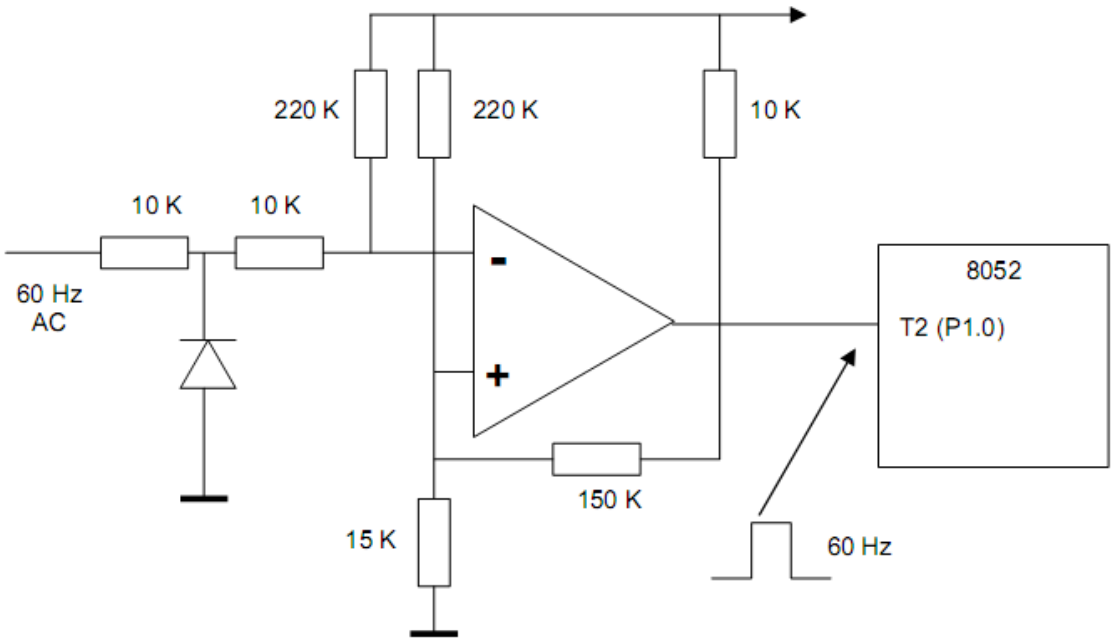
5. Theo chuẩn thế giới tần số chuẩn để điều chỉnh để điều chỉnh các nhạc cụ là nốt ``A`` có tần số 440 Hz. Viết chương trình tạo âm thanh này cung cấp cho loa đặt tại chân P1.1 (hình 3.12). Do việc làm tròn của TL1/TH1 nên sẽ xuất hiện sai số. Hãy cho biết tần số thực sự nhận được tại ngõ ra là bao nhiêu ? Sai số tính theo phần trăm ? Với chương trình đã viết để có được tần số chính xác 440 Hz thì tần số thạch anh phải là bao nhiêu ?



Hình 3.12 Mạch tạo âm thanh

6. Viết chương trình tạo xung 500 Hz tạI P1.0 dùng timer 0, dạng sóng ra có duty cycle là 30%.

7. Mạch điện ở hình 3.13 cung cấp một xung 60 Hz rất chính xác đến chân T2. hãy khởi tạo timer 2 sao cho xảy ra tràn mỗi giây một lần, xung tràn này sẽ cập nhật thời gian chứa trong RAM nội của 8052 tại địa chỉ 50H (giờ), 51H (phút), và 52H (giây)



Hình 3.13 Mạch tạo thời gian chuẩn 60 Hz

CHƯƠNG 3

HOẠT ĐỘNG CỦA PORT NỐI TIẾP MÃ BÀI: MH30-04

Mục tiêu:

- Trình bày được các thanh ghi điều khiển cổng nối tiếp (Serial Port)
- Sử dụng các cổng nối tiếp trong khi viết chương trình truyền dữ liệu
- Tự tin trong thao tác sử dụng Port nối tiếp.
- Tính chính xác, tỉ mỉ, cẩn thận.

Nội dung chính:

1. Giới thiệu các Port của vi xử lý 89C51

Mục tiêu:

- Trình bày được các Port của vi xử lý 89C51.

Cổng nối tiếp tích hợp trong họ 8051 có vài chế độ hoạt động trong một phạm vi 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.

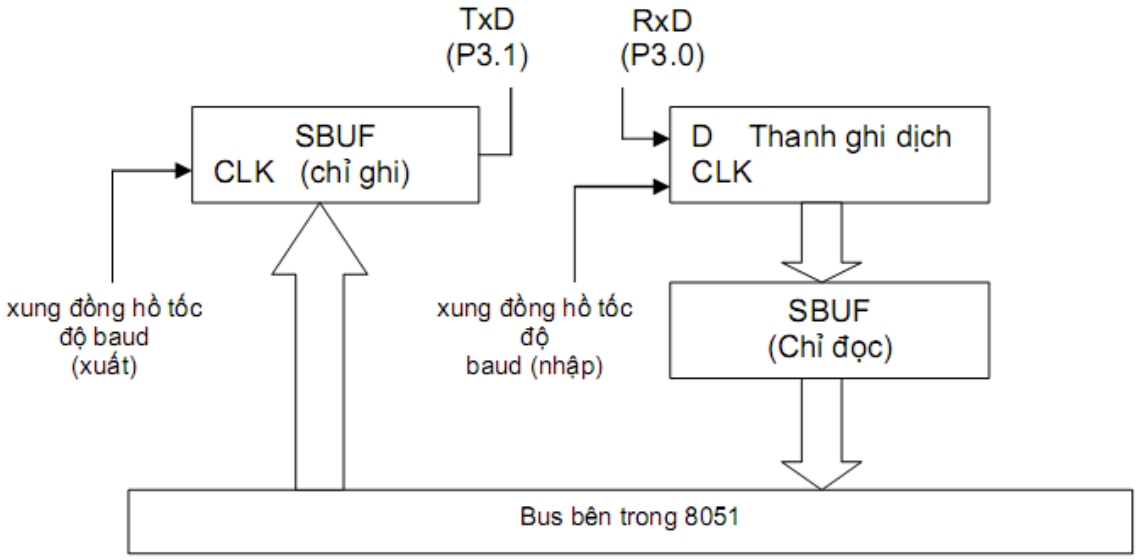
Thiết bị ngoại vi giao tiếp với port nối tiếp qua các chân TXD và RXD như giới thiệu ở bài 2. Các chân này là các chân đa chức năng của port 3, bít P3.1 tại chân 11 (TXD) và P3.0 tại chân 10 (RXD).

Đặc điểm của port nối tiếp là truyền sóng công toàn phần (thu phát đồng thời) và đặc tính đệm dữ liệu cho phép lưu giữ ký tự 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 liệu 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 ngoài và đọc bộ đệm tức là nhận dữ liệu từ ngoài vào trong bộ đệm. Điều này có nghĩa là có 2 thanh ghi phân biệt: Thanh ghi xuất chỉ cho phép ghi và thanh ghi nhận chỉ cho phép đọc. (Hình 5.1)

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.)



Hình 4.1 Sơ đồ khối cổng nối tiếp

2. **Thanh ghi điều khiển cổng nối tiếp**

Mục tiêu:

- Trình bày được các thanh ghi điều khiển cổng nối tiếp (Serial Port).

Chế độ làm việc của cổng nối tiếp được thiết lập bằng cách ghi vào thanh ghi điều khiển SCON tại địa chỉ 99H (bảng 4.1 và bảng 4.2).

Trước khi sử dụng cổng nối tiếp, SCON phải được khởi tạo đúng chế độ. Ví dụ: mã lệnh sau:

MOV SCON; # 01010010B

Sẽ khởi tạo cổng nối tiếp làm việc ở mode 1 (SMO/SM1 = 0/1) cho phép thu (REN=1) và set cờ ngắt phát (Ti = 1) để báo cho biết là cổng nối tiếp đã sẵn sàng để xuất dữ liệu.

3. **Các chế độ hoạt động**

Mục tiêu:

- Trình bày được các chế độ hoạt động của cổng nối tiếp (Serial Port).

Cổng nối tiếp trong 8051 có 4 chế độ làm việc và được chọn bằng cách ghi các giá trị 1 hoặc 0 vào các bít SMO và SM1 trong thanh ghi SCON. Ba chế độ cho phép truyền không đồng bộ trong đó mỗi ký tự thu hoặc phát được báo lại bởi một Start bít và một Stop bít. Chế độ này tương tự như chế độ làm việc của cổng nối tiếp RS232C của máy vi tính. Ở chế độ thứ tư, cổng nối tiếp hoạt động như một thanh ghi dịch đơn giản.

BẢNG 4.1 Thanh ghi SCON

Bít	Ký hiệu	Địa chỉ	Mô tả
SCON.7	SM0	9FH	mode bít ∅ (bảng 5-2)
SCON.6	SM1	9EH	mode bít 1 (bảng 5-2)
SCON.5	SM2	9DH	mode bít 2. Cho phép truyền thông đa xử lý trong mode 2 và 3; Ri sẽ không tác động nếu bít nhận thứ 9 bằng 0.
SCON.4	REN	9CH	bít cho phép thu, phải set bít này để thu ký tự
SCON.3	TB8	9BH	phát bít 8. Bít thứ 9 được phút ở mode 2 và 3, được set và clear bằng phần mềm.
SCON.2	RB8	9AH	thu bít 8. Bít thứ 9 được thu.
SCON.1	Ti	99H	Cờ ngắt phát. Set khi kết thúc việc phát ký tự, được clear bằng phần mềm.
SCON.0	Ri	98H	Cờ ngắt thu. Set khi kết thúc việc thu ký tự, được clear bằng phần mềm.

3.1 Thanh ghi dịch 8 bít (mode 0)

Mode o được chọn bằng cách ghi giá trị 0 vào bít SM1 và SM0 trong thanh ghi điều khiển SCON, lúc này cổng nối tiếp hoạt động như một thanh ghi dịch 8 bít. Dữ liệu nối tiếp vào và ra thông qua chân RxD và chân TxD tạo xung dịch chuyển, 8 bít thu-phát được bắt đầu với bít thấp nhất LSB (least significant bit).

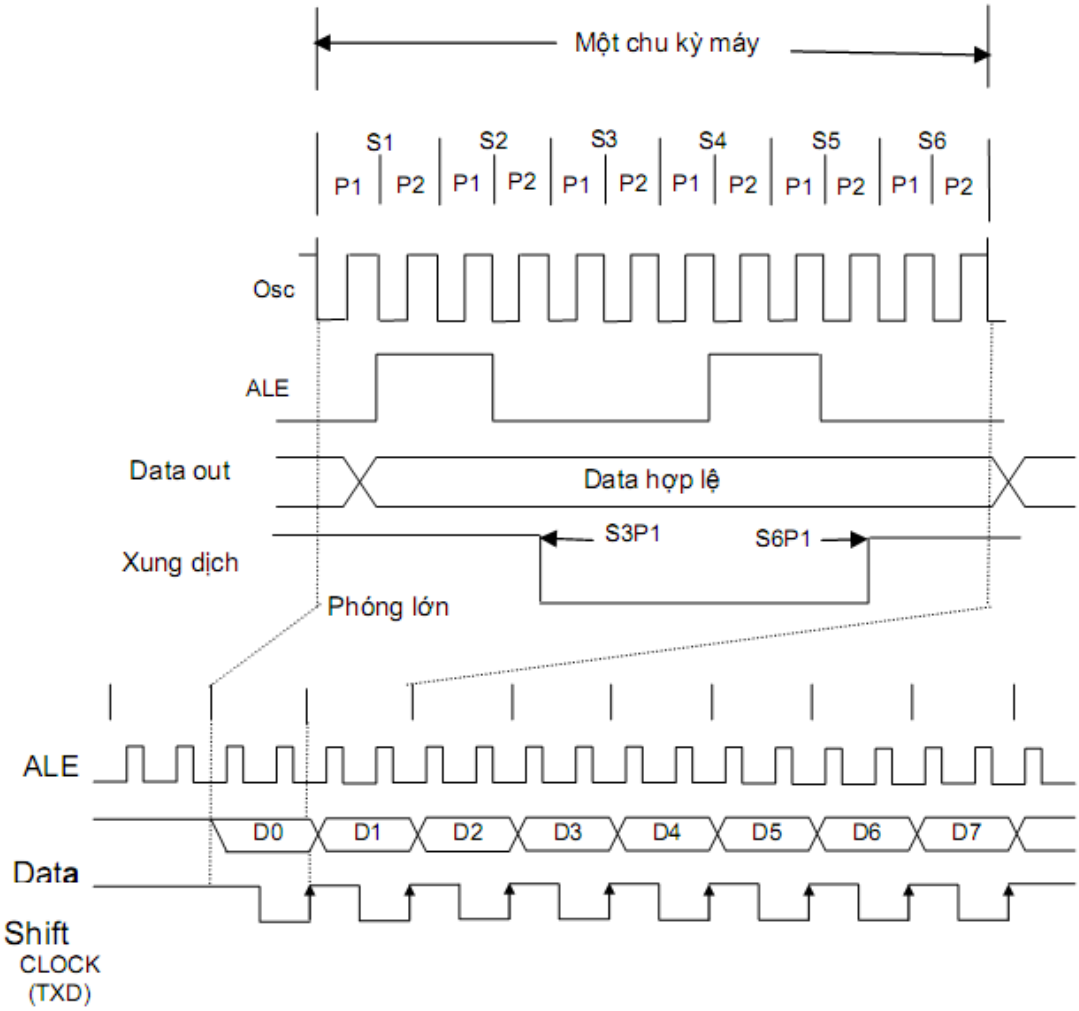
Tốc độ baud cố định tại giá trị 1/12 tần số dao động trên chip 8051 (lưu ý là ký hiệu RxD và TxD không còn đúng ý nghĩa trong chế độ này. Chân RxD được dùng để thu và phát dữ liệu trong khi chân TxD thì cung cấp xung dịch).

Quá trình truyền được khởi động bằng một lệnh bất kỳ viết dữ liệu vào SBUF, dữ liệu được dịch ra chân RxD (P3.0) theo nhịp xung đồng hồ ở chân TxD (P3.1). Mỗi bít được phát hợp lệ tại chân RxD trong một chu kỳ máy. Trong suốt một chu kỳ máy, xung đồng hồ ở mức thấp trong khoảng thời gian S3P1 và lên mức cao trong khoảng S6P1, giản đồ thời gian dữ liệu được vẽ ở hình 5.2.

Quá trình thu được bắt đầu khi bít cho phép thu REN =1 và cờ ngắt thu Ri = 0, quy tắc chung là set bít REN khi bắt đầu chương trình để khởi tạo cổng nối tiếp và sau đó xóa Ri để bắt đầu thu dữ liệu.

BẢNG 4.2 Chế độ làm việc

SMO	SM1	MODE	Mô tả	Tốc độ baud
0	0	0	Thanh ghi	Cố định (tần số dao động chia 12)
0	1	1	8 bít UART	Thay đổi (thiết lập bởi timer)
1	0	2	9 bít UART	Cố định (tần số dao động chia 12 hoặc 64)
1	1	3	9 bít UART	Thay đổi (thiết lập bởi timer)



Hình 4.2 Giản đồ thời gian phát mode 0

Khi bít Ri bị xóa, xung đồng hồ được cung cấp từ đường TxD và bắt đầu chu kỳ máy theo sau, dữ liệu được dịch vào chân RxD. Đương nhiên thiết bị ngoại vi cung cấp dữ liệu vào chân RxD được đồng bộ hóa với xung dịch ở đường TxD, việc đồng bộ dữ liệu vào cổng nối tiếp xảy ra tại cạnh lên của xung TxD (hình 5.3).

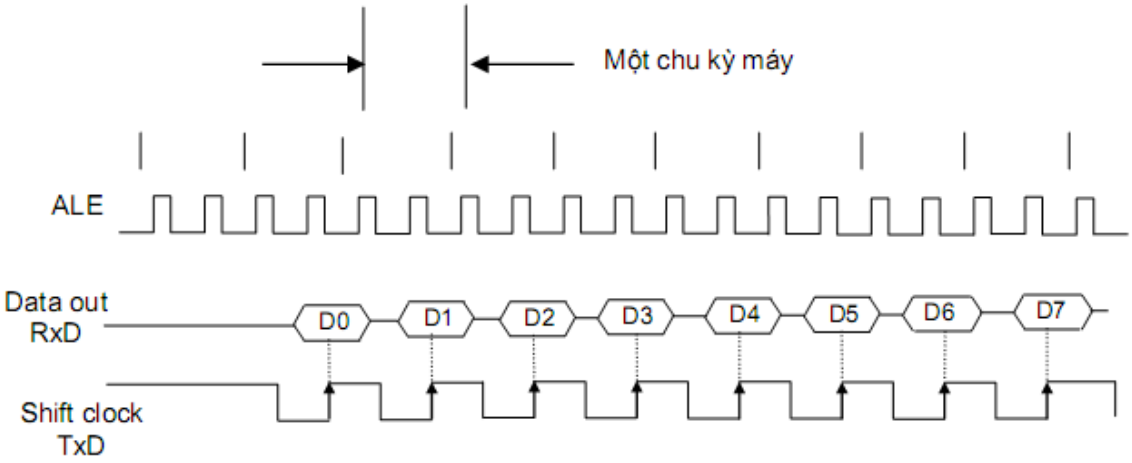
Một ứng dụng dùng mode thanh ghi dịch là mở rộng ngỏ ra của 8051. Một vi mạch ghi dịch từ nối tiếp sang song song được nối đến TxD và RxD

của 8051 để tạo ra thêm 8 đường xuất dữ liệu (hình 5.4), để mở rộng thêm có thể nối tầng (cascade) các vi mạch ghi dịch khác với vi mạch đầu tiên.

3.2 Chế độ UART 8 bít có tốc độ baud thay đổi (Mode 1)

Trong chế độ này cổng nối tiếp 8051 hoạt động như một bộ thu phát không đồng bộ vạn năng có tốc độ baud thay đổi viết tắt là UART. Đây là một mạch truyền thông dữ liệu nối tiếp với mỗi ký tự được bắt đầu bởi một start bit (mức thấp) và theo sau là một stop bít (mức cao), 1 bít chẵn - lẻ đôi khi được chèn vào giữa bít dữ liệu cuối cùng và stop bít.

Chức năng chính của UART là biến đổi dữ liệu ra từ song song sang nối tiếp và biến đổi dữ liệu vào từ nối tiếp sang song song.

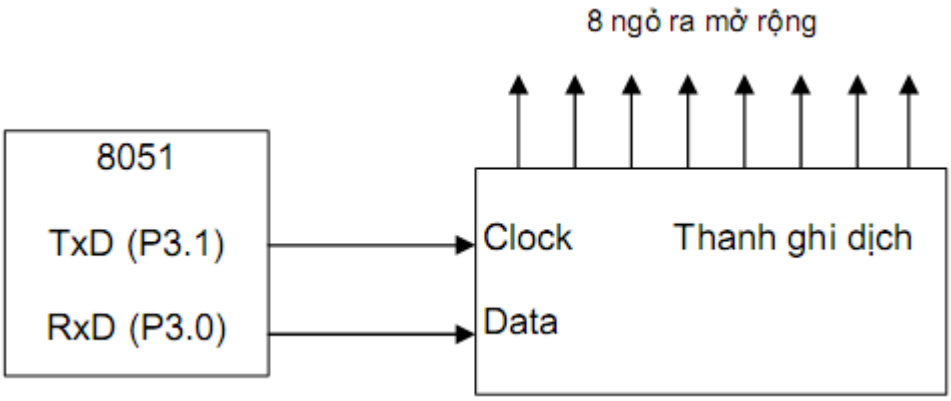


Hình 4.3 Giản đồ thời gian thu mode 0

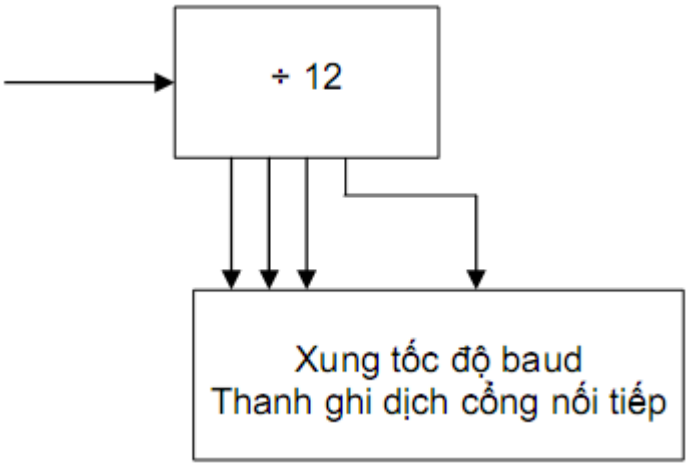
Trong mode 1 mười bít dữ liệu được phát đi từ TxD hoặc được nhận vào từ RxD, những bít này bao gồm 1 start bít (luôn ở mức thấp), 8 bít dữ liệu (bít đầu tiên là LSB) và 1 stop bít (luôn ở mức cao) khi thu, bít stop được đưa vào RB8 trong SCON. Đối với 8051 tốc độ baud được xác định bởi timer 1, đối với 8052 bởi timer 1 hoặc timer 2 hoặc cả hai (một phát và một thu).

Việc dịch chuyển và đồng bộ các thanh ghi dịch cổng nối tiếp trong mode 1,2 và 3 được thiết lập bởi một bộ đếm 4 bít chia 16, ngõ ra của bộ đếm này là xung tốc độ baud (Hình 4.5) ngõ vào bộ đếm được chọn bằng phần mềm.

Quá trình phát được khởi động bằng cách ghi vào SBUF nhưng việc phát cũng không thực sự bắt đầu cho đến lần tràn kế tiếp của bộ đếm cung cấp xung tốc độ baud cho cổng nối tiếp.



Hình 4.4 Mode thanh ghi dịch 16x tốc độ baud



Hình 4.5 Xung dịch cổng nối tiếp

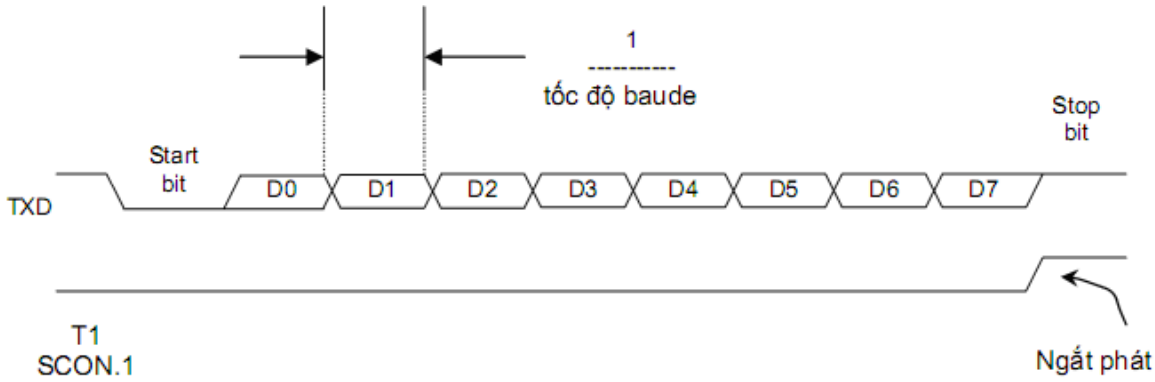
Dữ liệu dịch đưa ra tại chân TxD được bắt đầu bằng start bit, theo sau là 8 bít dữ liệu và kế tiếp là stop bít, thời gian của mỗi bít là nghịch đảo của tốc độ baud được lập trình trong timer. Cờ ngắt phát (Ti) được set ngay khi stop bít xuất hiện ở TxD (Hình 4.6).

Quá trình thu được khởi động bởi sườn xuống tại RxD, ngay lập tức bộ đếm 10 bít xóa để gán số đếm với chuỗi bít nhận vào (bít kế tiếp đến khi bộ đếm tràn và cứ thế tiếp tục). Dòng bít vào được được lấy mẫu ở giữa 16 số đếm.

Quá trình thu bao gồm việc nhận dạng sự sai sót của bít bằng cách yêu cầu một trạng thái 0 (start bít) của 8 số đếm sau sườn xuống đầu tiên. Nếu điều này không xảy ra giả sử bộ thu bị kích thích bởi nhiều thay đổi vì một ký tự thì mạch thu sẽ reset và trở về trạng thái nghỉ cho đến khi gặp sườn xuống tiếp theo.

Giả sử một bít start hợp lệ được nhận ra, quá trình nhận ký tự sẽ tiếp tục, bít start được bỏ qua và 8 bít dữ liệu được dịch vào thanh ghi nối tiếp, khi việc nhận 8 bít hoàn tất thì sẽ xảy ra những điều sau đây:

1. Bít thứ 9 (stop bít) được chuyển vào RB8 trong SCON
2. Tám bít dữ liệu được nạp vào SBUF
3. Cờ ngắt thu (Ri) bị set.



Hình 4.6 Thiết lập cờ Ti

Tuy nhiên, các điều nói trên chỉ xảy ra nếu thỏa mãn các điều kiện sau:

1. Ri = 0
2. SM 2 = 1 và stop bít nhận vào bằng 1, hoặc SM 2 = 0

Yêu cầu Ri = 0 để bảo đảm rằng phần mềm đã đọc xong ký tự trước đó (và Ri bị xóa) yêu cầu thứ hai phức tạp hơn nhưng chỉ áp dụng trong chế độ truyền thông đa xử lý, điều này có nghĩa là không set bít Ri trong chế độ truyền thông đa xử lý khi bít dữ liệu thứ 9 bằng 0.

3.3 UART 9 bít với tốc độ baud cố định (Mode 2)

Khi SM1 = 1 và SM0 = 0, cổng nới tiếp hoạt động như một UART 9 bít có tốc độ baud cố định, số bít thu và phát gồm 11 bít, 1 start bít, 8 data bít, 1 bít dữ liệu thứ 9 lập trình được và 1 stop bít, khi phát bít thứ 9 là bít bất kỳ trong TB8 của SCON (có thể là bít parity). Khi thu, bít thứ 9 nhận vào sẽ được đưa vào RB8. Tốc độ baud của mode 2 bằng 1/32 hoặc 1/64 tần số dao động trên chip.

3.4 Chế độ UART 9 bít với tốc độ baud thay đổi (Mode 3)

Mode này giống như mode 2 ngoại trừ tốc độ baud được lập trình và cung cấp bởi timer. Trên thực tế mode 1, 2 và 3 rất giống nhau. Điểm khác nhau là tốc độ baud (cố định trong mode 2, thay đổi trong mode 1 và 3) và về số lượng bít dữ liệu (8 trong mode 1, 9 trong mode 2 và 3).

4. Khởi động và truy xuất thanh ghi cổng nối tiếp

Mục tiêu:

- Trình bày được phương pháp khởi động và truy xuất thanh ghi cổng nối tiếp cổng nối tiếp (Serial Port).

4.1 Cho phép nhận

Phải set bít cho phép nhận REN trong thanh ghi SCON bằng phần mềm để cho phép quá trình thu các ký tự. Việc này thường được thực hiện ở đầu chương trình. Khi cổng nối tiếp, bộ đình thời ..... được khởi động, có 2 cách thực hiện.

```
SET B  REN
```

Sẽ làm REN = 1 hoặc dùng lệnh sau

```
MOV  SCON , # xxx 1 xxxx B
```

Sẽ set bít REN và các bít còn lại của SCON có thể là 1 hoặc 0.

4.2 Bít dữ liệu thứ 9

Bít dữ liệu thứ 9 được phát ở mode 2 và 3 phải được nạp vào TB8 bằng phần mềm.

Bít dữ liệu thứ 9 đã nhận thì được thay vào RB8. Phần mềm có thể hoặc không cần đến bít dữ liệu thứ 9 phụ thuộc vào sự thiết lập truyền thông của thiết bị nối tiếp. (bít dữ liệu thứ 9 còn đóng một vai trò quan trọng trong truyền thông đa xử lý).

4.3 Thêm vào bít chẵn - lẻ (parity)

Một ứng dụng thường dùng của bít dữ liệu thứ 9 là để chèn bít chẵn - lẽ vào trong ký tự, như đã biết bít parity trong thanh ghi PSW được set và clear trong từng chu kỳ máy để tạo bít chẵn tương ứng với dữ liệu 8 bít trong bộ tích lũy.

VD: Nếu yêu cầu truyền thông cần truyền đi 8 bít dữ liệu kèm theo 1 bít chẵn, đoạn mã lệnh sau sẽ phải đi 8 bít trong bộ tích lũy và bít chẵn là bít thứ 9

```
MOV   C, P     ; đưa bít chẵn vào TB8
MOV    TB8, C   ; bít này trở thành bít dữ liệu thứ 9
MOV   SBUF, A  ; chuyển 8 bít từ A vào SBUF
```

Nếu cần truyền bít lẻ thì đoạn mã lệnh được sửa lại như sau:

```
MOV C, P  ; đưa bít chẵn vào cờ C
CPL C  ; đổi thành bít lẻ
MOV  TB8, C
MOV  SBUF, A
```

Đương nhiên việc sử dụng bít parity không bị giới hạn ở mode 2 và 3. Trong mode 1, tám bít được phát đi có thể bao gồm 7 bít dữ liệu cộng thêm một parity bít. Để phát mã ASCII 7 bít và bít chẵn là bít thứ 8, có thể viết như sau:

```
CLR  ACC-7   ; xóa bít MSB, bít parity là bít P
MOV  C, P
MOV  ACC-7, C  ; đưa bít chẵn vào MSB
MOV  SBUF, A  ; gởi ký tự 7 bít và một bít chẵn
```

4.4. Các cờ ngắt

Các cờ ngắt thu và phát (Ri và Ti) trong SCON có vai trò quan trọng trong yêu cầu truyền thông nối tiếp của 8051. Cả hai bít này được set bằng phần cứng nhưng phải xóa bằng phần mềm.

Cụ thể là Ri được set khi kết thúc quá trình thu ký tự và cho biết là bộ đệm thu đã đầy. Điều này được kiểm tra bằng phần mềm hoặc có thể được lập trình để tạo ra một ngắt. Nếu chương trình cần thu mốt ký tự từ thiết bị ngoại vi được kết nối với cổng nối tiếp (thì nó phải đợi cho đến khi Ri được set, sau đó xóa Ri và đọc ký tự trong SBUF, đoạn mã thực hiện như sau:

WAIT : JNB Ri, WAIT ; Kiểm tra Ri cho đến khi được set
CLR Ri ; xóa Ri
MOV A, SBUF ; đọc ký tự

Ti được set khi kết thúc quá trình phát ký tự và cho biết là bộ đếm phát đã trống, để phát mốt ký tự đến thiết bị ngoại vi kết nối với cổng nối tiếp. Trước tiên phải chắc chắn là cổng nối tiếp đã sẵn sàng, nói cách khác nếu một ký tự trước đó đã được phát đi thì chương trình phải chờ cho đến khi kết thúc quá trình phát trước khi phát ký tự tiếp theo.

Đoạn mã sau sẽ phát một ký tự chứa trong A.

WAIT: JNB Ti, WAIT ; kiểm tra Ti cho đến khi được set
CLR Ti ; xóa Ti
MOV SBUF, A ; phát ký tự

5. Tốc độ truyền (Baud Rate) nối tiếp

Mục tiêu:

- Trình bày được các chế độ truyền (Baud Rate) nối tiếp.

Tốc độ baud được cố định ở mode 0 và mode 2. Ở mode 0 tốc độ này luôn bằng tần số dao động của 8051 chia cho 12 thường là dao động thạch anh, nếu tần số dao động này là 12MHz thì tốc độ baud ở mode 0 là 1MHz (hình 5.8a)

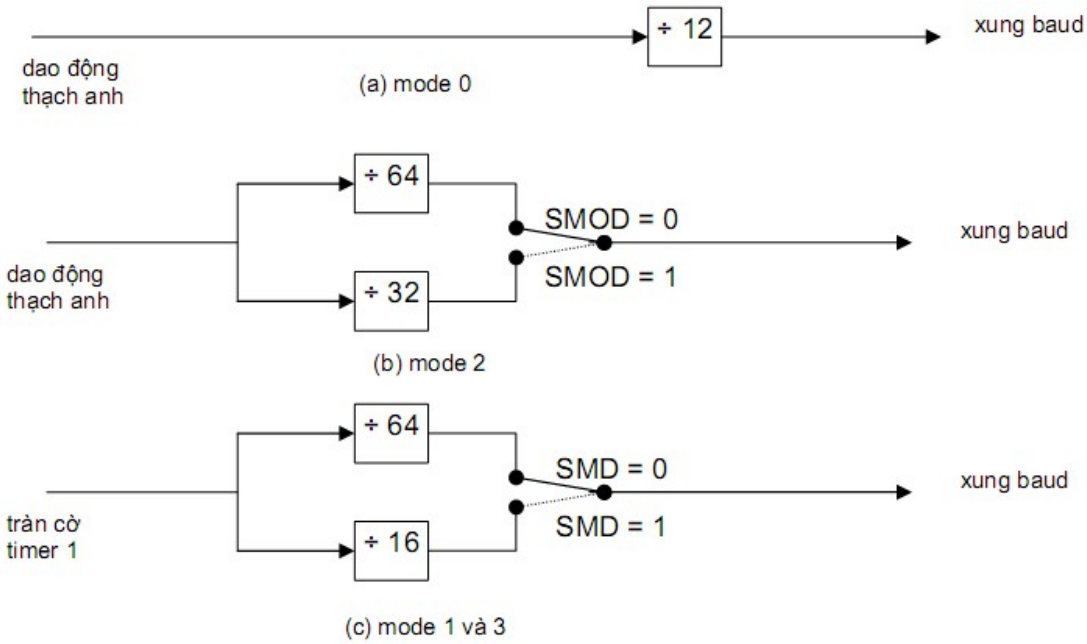
Khi reset hệ thống gía trị mặc định của tốc độ baud ở mode 2 bằng tần số dao động chia cho 64. Tốc độ baud cũng bị ảnh hưởng bởi một bít trong thanh ghi PCON, bít của PCON là bít SMOD, set SMOD bít sẽ làm tăng đôi tốc độ baud ở mode 1, 2 và 3.

Trong mode 2 tốc độ baud có thể được tăng đôi từ giá trị mặc định là 1/64 tần số dao động (SMOD = 0) đến 1/32 tần số dao động (SMOD = 1). (Xem hình 5.8b)

Vì thanh ghi PCON không được định địa chỉ bít, nên để set bít SMOD mà không làm thay đổi các bít khác phải cần một thao tác đọc - sửa - ghi như sau:

MOV A, PCON ; đọc giá trị hiện hành của PCON
SETB ACC.7 ; set bít 7 (SMOD)
MOV PCON, A ; ghi giá trị mới vào PCON

Các tốc độ baud của 8051 trong chế độ 1 và 3 được xác định bởi tốc độ tràn của timer 1. Vì timer hoạt động với tần số tương đối cao, nên phải chia thêm 32 (16 nếu SMOD = 1) trước khi cung cấp xung tốc độ baud cho cổng nối tiếp. Tốc độ baud của 8052 ở chế độ 1 và 3 được xác định bởi tốc độ tràn của timer 1 hoặc timer 2 hoặc cả hai.



Hình 4.8 Tạo xung baud cho cổng nối tiếp

Dùng timer 1 để tạo xung baud

Phương pháp tạo xung baud là khởi tạo TMOD ở chế độ tự nạp lại 8 bít (mode 2 của timer) và đặt đúng giá trị nạp lại trong TH1 tương ứng với tốc độ baud cần thiết lệnh cách khởi tạo TMOD như sau:

MOV TMOD, # 0010 xxxx B

Các ký hiệu "x" có thể là 1 hoặc 0 theo yêu cầu của timer 0.

Có thể tạo xung baud tốc độ chậm bằng cách dùng chế độ 16 bít, timer mode 1 với TMOD = 0001 xxxx B. Tuy nhiên, có một lỗi nhỏ xảy ra do cặp thanh ghi dịnh thời TH1/TL1 phải được khởi tạo lại sau mỗi lần tràn. Nên thực hiện việc này bằng một chương trình phục vụ ngắt, một lựa chọn khác là dùng xung ngoài kích vào ngõ T1 (P3.5). Dù áp dụng cách nào thì tốc độ baud luôn là tốc độ tràn của timer 1 chia cho 32 (hoặc chia cho 16 nếu SMOD = 1)

Công thức tổng quát để tính tốc độ baud ở chế độ 1 và 3 như sau:

Tốc độ baud = Tốc độ tràn timer 1 ÷ 32

VD: Để có tốc độ 1200 baud, suy ra tốc độ tràn của timer bằng 32 lần là 38,4 KHz.

Nếu tần số thạch anh là 12MHz thì xung kích cho timer 1 là 1MHz = 1.000 KHz, vì timer phải tràn với tốc độ là 38,4KHz nên sau mỗi 1.000/38,4 = 26,04 xung (làm tròn là 26) kích thì timer phải tự tràn. Do timer đếm lên và chỉ tràn khi số đếm chuyển từ FFH xuống 00H nên giá trị nạp lại phải nhỏ hơn 0 một lượng là +26, giá trị đúng là –26, cách dễ nhất để đặt giá trị nạp lại vào TH1 là:

MOV TH1, # -26

Phần mềm hợp ngữ sẽ dịch –26 thành 0E6H, do đó lệnh tương đương lệnh trên là:

MOV TH1, 0E6H

Do làm tràn nên sẽ làm sai chút ít kết quả tốc độ baud, phương pháp truyền không đồng bộ (start/ stop) cho phép sai số đến 5%. Có thể tạo tốc độ baud chính xác bằng cách dùng thạch anh 11,059 MHz. Bảng 5.3 tóm tắt các giá trị nạp lại cho TH1 tương ứng với hầu hết các tốc độ baud dùng thạch anh 12MHz và 11,059MHz.

Bảng 4.3 Tóm tắt tốc độ baud

Tốc độ baud	Tần số thạch anh	SMOD	Giá trị nạp lại TH1	Tốc độ baud thực sự	Sai số
9600	12,000 MHz	1	-7(F9H)	8923	7%
2400	12,000 MHz	0	-13(F3H)	2404	0,16%
1200	12,000 MHz	0	-26(E6H)	1202	0,16%
19200	11,059 MHz	1	-3(FDH)	19200	0
9600	11,059 MHz	0	-3(FDH)	9600	0
2400	11,059 MHz	0	-12(F4H)	2400	0
1200	11,059 MHz	0	-24(E8H)	1200	0

Ví dụ 3-1: Khởi tạo cổng nối tiếp

Viết đoạn mã lệnh khởi tạo cổng nối tiếp hoạt động như một UART với tốc độ 2400 baud, dùng timer 1 để cung cấp xung baud.

Trong ví dụ này phải khởi tạo 4 thanh ghi SMOD, TMOD, TCON và TH1, các gía trị cần thiết được tóm tắt như sau:

	SMO	SM1	SM2	REN	TB8	RB8	Ti	Ri
SCON:	0	1	0	1	0	0	1	0
	GTE	C/T	M1	M0	GTE	C/T	M1	M0
TMOD:	0	0	1	0	0	0	0	0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
TCON:	0	1	0	0	0	0	0	0
TH1:	1	1	1	1	0	0	1	1

Thiết lập SMO/SM1 = 0/1 để đặt cổng nối tiếp vào chế độ UART 8 bít, REN = 1 cho phép cổng nối tiếp nhận các ký tự, đặt Ti = 1 cho phép phát ký tự đầu tiên bằng cách báo rằng bộ đệm phát trống. Đối với TMOD việc thiết lập M1/M0 = 1/0 sẽ đặt timer 1 ở chế độ tự nạp lại 8 bít, đặt TR1 = 1 để khởi động timer 1, các bít khác bằng 0 vì chúng điều khiển các đặc tính và chế độ không cần dùng trong ví dụ này.

Tốc độ tràn của timer 1 là 2.400 baud x 32 = 76,8KHz. Giả sử tần số thạch anh của 8051 là 12MHz nên xung kích timer 1 là 1MHz = 1.000KHz do đó xung cho mỗi lần tràn là 1.000KHz ÷ 76,8KHz = 13,02 (làm tròn 13), vậy giá trị nạp lại là –13 hoặc 0F3H, đoạn lệnh khởi tạo như sau:

```
INIT:  MOV  SCON, # 52 H   ; cổng nối tiếp ở mode 1
MOV  TMOD, # 20 H   ; timer 1, mode 2
MOV TH1, # -13  ; trị nạp lại tương ứng 2.400 baud
SET TR1   ; khởi động timer 1
```

Ví dụ 3-2: Chương trình con xuất ký tự

Viết một chương trình con trên là OUTCHR để phát mã ASCII 7 bít trong thanh ghi A ra cổng nối tiếp của 8051 với bít lẻ là bít thứ 8, khi trở về nội dung của A vẫn không bị thay đổi.

Ví dụ này và mô tả tiếp theo là 2 trong hầu hết các chương trình con dùng cho máy vi tính có ngõ kết nối RS232; xuất ký tự (OUTCHR) và nhập ký tự (INCHAR)

```
OUTCHR: MOV C, P  ; đưa parity bít bào cờ C
CPL  C  ; đổi thành bit lẻ
MOV ACC.7, C ; cộng vào ký tự
AGAIN: JNB Ti, AGAIN ; kiểm tra bộ đếm phát trống?
CLR Ti  ; xóa cờ
MOV SBUF, A ; xuất ký tự
CLR  ACC.7  ; xóa bít 7 của A
RET   ; trở về
```

Ba lệnh đầu tiên đưa bít lẻ vào bít 7 của thanh ghi A, vì bít P torng PSW là 0 khi nội dung thanh ghi A là chẵn nên nó phải được đảo lại trước khi đưa vào ACC.7, lệnh JNB tạo một vòng lặp chờ cho đến khi cờ ngắt phát Ti = 1 (do ký tự được phát đi trước đó kết thúc) sau đó nó được xóa và ký tự trong A được ghi vào bộ đệm cổng nối tiếp, quá trình phát bắt đầu ở lần tràn kế tiếp của bộ đêm 16 tạo xung kích cho cổng nối tiếp (hình 5.5). Cuối cùng, bít ACC.7 được xóa để nội dung của A cũng giống như khi gọi chương trình con OUTCHR có thể được gọi để phát ký tự đơn hoặc một chuỗi ký tự.

VD: Các lệnh sau đây sẽ phát mã ASCII của ký tự "z" ra thiết bị nối tiếp kết nối với cổng nối tiếp của 8051.

```
MOV A, # 'z'
CALL OUTCHR
(tiếp tục)
```

Ví dụ 3-3: Chương trình con nhập ký tự

Viết chương trình con có tên là INCHR để nhập mốt ký tự từ cổng nối tiếp của 8051 và trở về với mã ASCII 7 bít trong thanh ghi A. Sử dụng bít kiểm tra lẻ trong bít nhập thứ 8 và set cờ carry nếu có lỗi parity.

```
INCHAR: JNB Ri, $  ; chờ ký tự
CLR Ri  ; xóa cờ
MOV A, SBUF  ; đọc ký từ vào A
MOV C, P        ; P set nếu A lẻ
CPL C           ; C = 1 nếu có lỗi parity
CLR ACC.7       ; xóa parity
RET
```

Chương trình bắt đầu bằng cách chờ cờ ngắt thu Ri được set, cho biết rằng ký tự đang ở trong SBUF chờ đọc. Khi Ri = 1, lệnh kế tiếp JNB được thực hiện, Ri bị xóa và nội dung trong SBUF được đọc vào A.

Bít P trong PSW thiết lập kiểm tra chẵn cho thanh ghi A, vì vậy bít này cần được set bằng 1 nếu bản thân thanh ghi A chứa bít kiểm tra lẻ ở bít thứ 7 của thanh ghi này.

Việc di chuyển bít P vào cờ nhớ làm cho CY = 0 nếu không có lỗi. Mặt khác nếu thanh ghi A chứa một lỗi chẵn – lẻ, cờ CY sẻ bằng 1. Cuối cùng bít ACC.7 được xóa để bảo đảm rằng chỉ có mã 7 bít được trả về cho chương trình gọi.

BÀI TẬP THỰC HÀNH

Những bài tập sau đây là các chương trình điển hình trong yêu cầu giao tiếp giữa các thiết bị đầu cuối (hoặc các thiết bị nối tiếp khác) với máy tính. Giả sử cổng nối tiếp của 8051 được khởi tạo ở chế độ UART 8 bít và tốc độ buade được cung cấp bởi timer 1.

1. Viết chương trình con có tên là OUTSTR để gửi một chuỗi mã ASCII kết thúc bằng ký tự null đến thiết bị kết nối với cổng nối tiếp của 8051. Giả sử chuỗi mã ASCII được chứa ở bộ nhớ chương trình ngoài và chương trình gọi đặt địa chỉ của chuỗi vào con trỏ dữ liệu trước khi gọi OUTSTR. Chuỗi này là các byte ASCII nối tiếp và kết thúc bằng byte 00H.
2. Viết chương trình con có tên INLINE để nhập một dòng mã ASCII từ một thiết bị nối với cổng nối tiếp của 8051 và lưu vào RAM nội tại địa chỉ 50H. Giả sử dòng dữ liệu được kết thúc bằng ký tự xuống dòng. Đặt mã của ký tự xuống dòng trong bộ đệm dòng theo sau các mã khác và sau đó kết thúc bộ đệm dòng bằng byte 00H.
3. Viết chương trình gửi liên tục các chữ cái (chữ nhỏ) đến thiết bị kết nối với port
nối tiếp 8051. Dùng chương trình con OUTCHAR ở phần trên.
4. Dựa trên chương trình con OUTCHAR viết chương trình gửi liên tục các mã
ASCII hiển thị được (20H – 7EH) đến thiết bị kết nối với cổng nối tiếp của 8051.
5. Sửa lại lời giải của bài tập trên để đưa ngõ ra đến màn hình bằng cách dùng
mã XOFF và XON nhập vào từ bàn phím. Bỏ qua các mã nhập vào khác (lưu ý:
XOFF = CONTROL-S; XON = CONTROL-Q)
6. Dựa trên các chương trình con INCHAR và OUTCHR, viết chương trình nhập
các ký tự từ bàn phím và hiển thị chúng trên màn hình. Biến đổi chữ nhỏ thành

chữ in.

7. Dựa trên các chương trình con INCHAR và OUTCHR, viết chương trình nhập
các ký tự từ một thiết bị kết nối với port nối tiếp của 8051 và hiển thị trên màn
hình bằng cách thay thế bằng dấu chấm (.) đối với các ký tự điều khiển (có mã
ASCII 00H đến 1FH và 7FH).

8. Dùng chương trình con OUTCHR viết chương trình xóa màn hình kết nối với
port nối tiếp 8051 sau đó gửi tên của bạn ra màn hình 10 lần trên 10 dòng khác nhau. Chức năng xóa màn hình được thực hiện bằng cách gửi mã CONTROL-Z
đối với đa số thiết bị hoặc <ESC> đối với các thiết bị hỗ trợ chuẩn ANSI (american national standarts institute).

9. Trong hình 5.1 cho thấy một kỹ thuật mở rộng ngõ ra của 8051. Giả sử với cấu
hình như thế viết một chương trình khởi tạo port nối tiếp 8051 ở chế độ thanh
ghi dịch và sau đó ánh xạ nội dung của RAM nội tại địa chỉ 20H đến 8 ngõ ra mở rộng, mỗi giây 10 lần

CHƯƠNG 4
HOẠT ĐỘNG NGẮT

MÃ BÀI: MH30-05

Mục tiêu:

- Trình bày được các ngắt của vi điều khiển
- Xử lý và vận dụng được các ngắt trong khi viết chương trình
- Tự tin trong thao tác sử dụng ngắt.
- Tính chính xác, tỉ mỉ, cẩn thận.

Nội dung chính:

1. Giới thiệu các ngắt

Mục tiêu:

-Trình bày được các ngắt của vi điều khiển

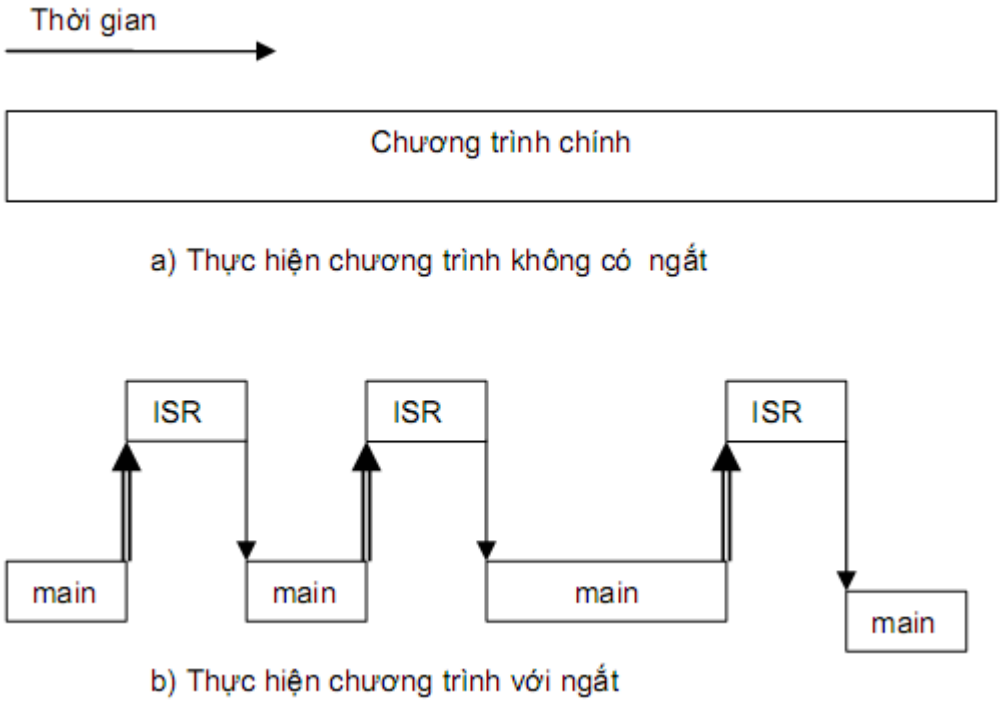
Ngắt là sự xuất hiện của một điều kiện, một sự kiện làm tạm dừng chương trình trong khi điều kiện này được phục vụ bởi một chương trình khác. Ngắt có một vai trò quan trọng trong thiết kế và thực hiện các ứng dụng của vi điều khiển. Chúng cho phép hệ thống đáp ứng không đồng bộ với một sự kiện và xử 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, trong hình 6.1a trình bày hoạt động của một chương trình không có ngắt và 5.1b là hoạt động của chương trình chính ở mức cơ sở có ngắt và các ngắt hoạt động ở mức ngắt.

Một ví dụ điển hình về ngắt là việc nhập dữ liệu bằng tay dùng bàn phím. Hãy khảo sát một 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 giảm 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



Hình 5.1 Thực hiên chương trình

2. Tổ chức các ngắt

Mục tiêu:

- Trình bày được quá trình tổ chức ngắt của vi điều khiển.

8051 có năm nguồn tín hiệu ngắt: 2 ngắt ngoài, 2 ngắt định thời và 1 ngắt cổng nối tiếp. 8052 có thêm ngắt thứ sáu của timer thứ ba. Trạng thái

mặc định của các ngắt là không hoạt động sau khi reset hệ thống và chuyển sang hoạt động từng ngắt riêng rẽ bằng phần mềm.

Trong trường hợp có hai hoặc nhiều ngắt xuất hiện đồng thời hoặc một ngắt xảy ra trong khi một ngắt khác đang được phục vụ. Có hai sơ đồ sắp xếp ưu tiên các ngắt đó là: Chuỗi pooling và ưu tiên hai cấp, thứ tự theo chuỗi pooling thì cố định nhưng sơ đồ ưu tiên hai cấp thì lập trình được. Sau đây là phương pháp cho phép và không cho phép sự hoạt động của các ngắt.

Cho phép và không cho phép các ngắt

Mỗi một tín hiệu ngắt được cho phép hoặc không cho phép bởi địa chỉ bít trong thanh ghi chức năng đặc biệt IE (Interrupt enable) tại địa chỉ 0A8H, có một bít cho phép toàn cục, bít này khi bị xóa sẻ ngăn tất cả các ngắt. (bảng 6.1).

Để cho phép một ngắt cần phải set hai bít: Một bít cho phép riêng và bít cho phép toàn cục. VD Ngắt timer 1 được cho phép như sau:

```
SETB ET1
SETB EA
```

Hoặc

```
MOV  IE,#10001000B
```

Mặc dù hai cách trên có cùng kết quả sau khi reset hệ thống nhưng kết quả sẽ khác nhau nếu IE được ghi ở giữa chương trình trong khi đang chạy. Cách thứ nhất không ảnh hưởng đến 5 bít còn lại trong thanh ghi IE còn cách thứ hai sẽ xóa các bít còn lại khác.

Tốt nhất nên dùng cách thứ hai tại vị trí bắt đầu chương trình (nghĩa là khi bắt đầu mở máy hoặc reset hệ thống) nên dùng lệnh SETB và CLR trong khi chương trình đang chạy để tránh ảnh hưởng các bít khác trong thanh ghi IE.

Mức ưu tiên

Mỗi ngắt được lập trình ở một trong hai mức ưu tiên bằng thanh ghi IP (Interrupt priority) tại địa chỉ oB8H (bảng 6.2) Thanh ghi IP tự động xóa sau khi reset hệ thống để đặt các ngắt ở mức ưu tiên thấp

BẢNG 5.1 Thanh ghi IE

BÍT	Ký hiệu	Địa chỉ bít	Mô tả (1=cho phép, 0=không cho phép)
IE.7	EA	AFH	Cho phép toàn cục
IE.6	-	AEH	Không dùng
IE.5	ET2	ADH	Cho phép ngắt timer 2 (8052)
IE.4	ES	ACH	Cho phép ngắt cổng nối tiếp
IE.3	ET1	ABH	Cho phép ngắt timer 1
IE.2	EX1	AAH	Cho phép ngắt 1 ngoài
IE.1	ET0	A9H	Cho phép ngắt timer 0
IE.0	EX0	A8H	Cho phép ngắt 0 ngoài

BẢNG 6.2 Thanh ghi IP

BÍT	Ký hiệu	Địa chỉ bít	Mô tả (1=mức cao, 0=mức thấp)
IP.7	-	-	Không dùng
IP.6	-	-	Không dùng
IP.5	PT2	0BDH	Ưu tiên ngắt timer 2 (8052)
IP.4	PS	0BCH	Ưu tiên ngắt cổng nối tiếp
IP.3	PT1	0BBH	Ưu tiên ngắt timer 1
IP.2	PX1	0BAH	Ưu tiên ngắt 1 ngoài
IP.1	PT0	0B9H	Ưu tiên ngắt timer 0
IP.0	PX0	0B8H	Ưu tiên ngắt 0 ngoài

Khái niệm "ưu tiên" cho phép một ISR bị dừng bởi một ngắt khác nếu ngắt mới xuất hiện này có mức ưu tiên cao hơn ngắt đang được phục vụ, điều này phù hợp với 8051 vì chỉ có hai mức ưu tiên, nếu một ISR ưu tiên thấp đang chạy nhưng lại xảy ra một ngắt ưu tiên cao thì ISR sẻ bị dừng. Một ISR ưu tiên không thể bị dừng.

Chương trình chính hoạt động ở mức ưu tiên cơ sở và không liên hệ với một ngắt bất kỳ nào nên luôn bị dừng khi xảy ra ngắt. Nếu hai ngắt có mức ưu tiên khác nhau cùng xảy ra thì ngắt có mức ưu tiên cao sẽ được phục vụ trước.

Chuỗi pooling

Nếu đồng thời xuất hiện hai ngắt có cùng mức ưu tiên thì ngắt được phục vụ trước được xác định theo thứ tự chuỗi pooling: Ngắt 0 ngoài, ngắt timer 0, ngắt 1 ngoài, ngắt timer 1, ngắt cổng nối tiếp, ngắt timer 2.

Hình 5.2 trình bày năm nguồn tín hiệu ngắt cùng cơ chế cho phép toàn cục và riêng rẽ, chuỗi pooling và các mức ưu tiên, trạng thái của tất cả các nguồn tín hiệu ngắt có thể thông qua các bít cờ trong thanh ghi chức năng đặc biệt. Dĩ nhiên, nếu một ngắt nào đó không được cho phép thì ngắt tương ứng không được tạo ra nhưng phần mềm vẫn có thể kiểm tra cờ ngắt. Các ví dụ về timer và cổng nối tiếp trong hai bài trước đã sử dụng các cờ ngắt mà thực tế không dùng các ngắt.

Một ngắt cổng nối tiếp là kết quả từ phép OR của ngắt thu (RI) với ngắt phát (TI). Tương tự, ngắt timer 2 được tạo ra bởi cờ tràn TF2 hoặc với cờ nhập bên ngoài EXF2. Khả năng tạo ngắt của các bít cờ được tóm tắt trong bảng 6.3.

3. Xử lý ngắt

Mục tiêu:

- Trình bày được quá trình xử lý ngắt trong vi điều khiển.

Khi một ngắt xuất hiện được CPU nhận ra, chương trình chính sẽ dừng lại và kế tiếp là các thao tác như sau:

• Thực hiện hoàn tất lệnh hiện hành

• Lưu nội dung thanh ghi PC vào ngăn xếp

• Lưu trạng thái ngắt hiện hành
• Các ngắt được chặn lại tại mức ngắt
• Nạp địa chỉ vectơ của ISR vào PC
• Thực hiện ISR

Chương trình ISR hoạt động và thực hiện các thao tác tương ứng với ngắt. Sau đó, kết thúc khi gặp lệnh RETI (return from interrupt), lệnh này lấy lại giá trị của PC từ ngăn xếp và phục hồi trạng thái ngắt cũ, chương trình tiếp tục chạy từ nơi tạm dừng.

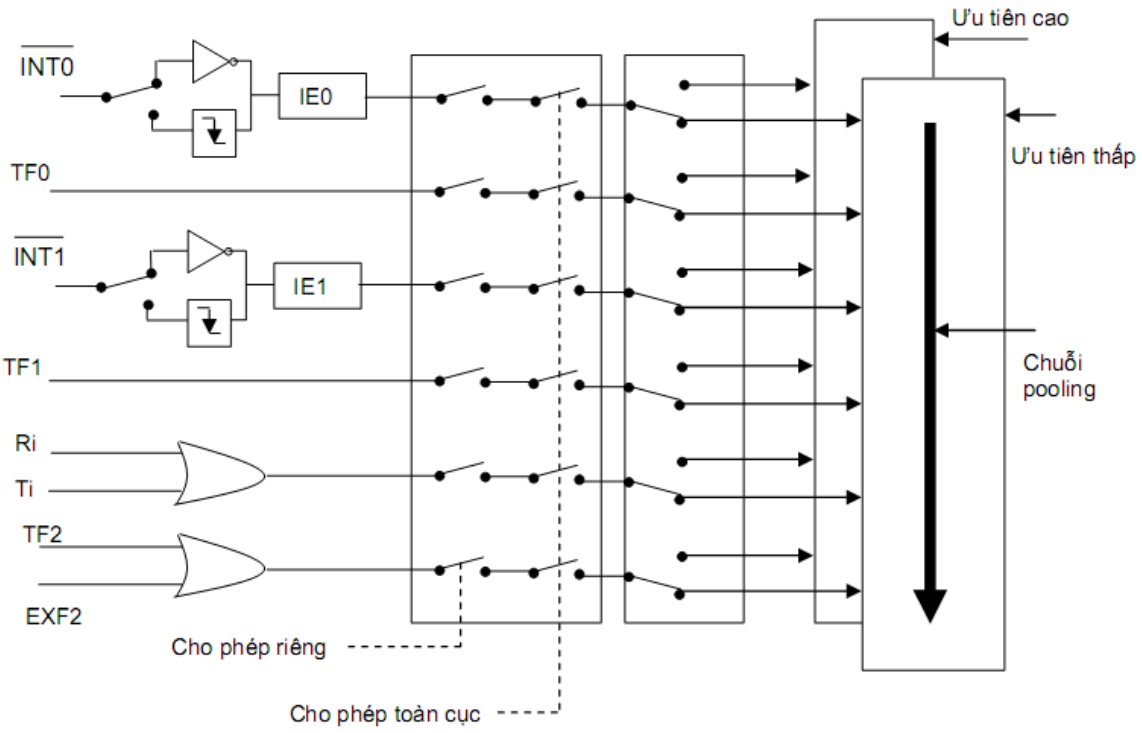
4. Các ngắt của 89C51

Mục tiêu:

- Trình bày được các ngắt của vi điều khiển 89C51.

BẢNG 6.3 Các bít cờ ngắt

Ngắt	Cờ	Thanh ghi SFR và vị trí bít
0 ngoài	IE0	TCON.1
1 ngoài	IE1	TCON.3
timer 1	TF1	TCON.7
timer 0	TF0	TCON.5
cổng nối tiếp	Ti	SCON.1
cổng nối tiếp	Ri	SCON.0
timer 2	TF2	T2CON.7 (8052)
timer 2	EXF2	T2CON.6 (8052)



Hình 5.2 Cấu trúc ngắt 8051

Các vec tơ ngắt

Khi một ngắt được chấp nhận, giá trị nạp vào bộ đếm chương trình PC được gọi là véc tơ ngắt. Đây là địa chỉ bắt đầu của ISR đối với ngắt tương ứng. Các véc tơ ngắt được cho trong bảng 6.4.

Véc tơ reset hệ thống (RST tại địa chỉ 0000H) cũng được cho trong bảng nên nó cũng giống như một ngắt, nó dừng chương trình chính và nạp vào PC một giá trị mới. Khi trỏ đến một ngắt, cờ gây ra ngắt sẽ bị xóa tự động bị xóa bởi phần cứng ngoại trừ Ri và Ti đối với ngắt cổng nối tiếp và TF2, EXF2 đối với ngắt timer 2. Do có hai nguyên nhân tạo ngắt cho các ngắt này nên thật là không thực tế nếu CPU xóa cờ ngắt. Các bít này phải được kiểm tra trong ISR để xác định nguyên nhân tạo ngắt và sau đó cờ ngắt được xóa bằng phần mềm, thường có một sự rẽ nhánh đến các thao tác tương ứng phụ thuộc vào nguồn tạo ra ngắt.

Vì các véc tơ ngắt được đặt phía dưới đáy của bộ nhớ chương trình nên lệnh đầu tiên của chương trình chính thường là lệnh nhảy qua vùng này VD lệnh LJMP 0030H.

Các ngắt ngoài xảy ra tại mức thấp hoặc cạnh âm ở chân $\overline{\text{INT0}}$ hoặc $\overline{\text{INT1}}$ của 8051, đây là các chân đa năng của port 3: Bít P3.2 (chân 12) và bít P3.3 (chân 13).

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 sẽ 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 $\overline{\text{INT1}}$ và nếu IT1 = 1 thì ngắt này được kích bằng cạnh âm.

Trong chế độ này nếu các mẫu tại chân $\overline{\text{INT1}}$ 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 sẽ yêu cầu một ngắt.

Vì các chân ngắt ngoài được lấy mẫu mỗi chu kỳ máy một lần nên ngõ vào này phải được duy trì ít nhất trong 12 chu kỳ dao động để bảo đảm 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.

5. Thiết kế chương trình dùng ngắt

Mục tiêu:

- Thiết kế được chương trình ngắt cho vi điều khiển.

Trong các bài trước đây đã không dùng đến ngắt mà dùng nhiều các vòng lặp để kiểm tra cờ tràn của timer TF0, TF1 hoặc TF2 hoặc các cờ thu phát của cổng nối tiếp Ti hoặc Ri, vấn đề của phương pháp này là thời gian hoạt động của CPU hoàn toàn được dùng vào việc chờ các cờ này được set, điều này hoàn toàn không thích hợp với các ứng dụng hướng điều khiển mà trong đó yêu cầu vi điều khiển phải đồng thời tương tác với nhiều thiết bị vào ra.

Một ví dụ trong phần này sẽ minh họa các phương pháp thực tế viết phần mềm cho các ứng dụng hướng điều khiển, thành phần chính là các ngắt. Mặc dù các ví dụ này không cần thiết lớn hơn nhưng chúng sẽ phức tạp hơn, điều này được nhận ra bằng cách tiến hành từng bước. Một số rắc rối xảy ra trong khi thiết kế hệ thống là do các ngắt.

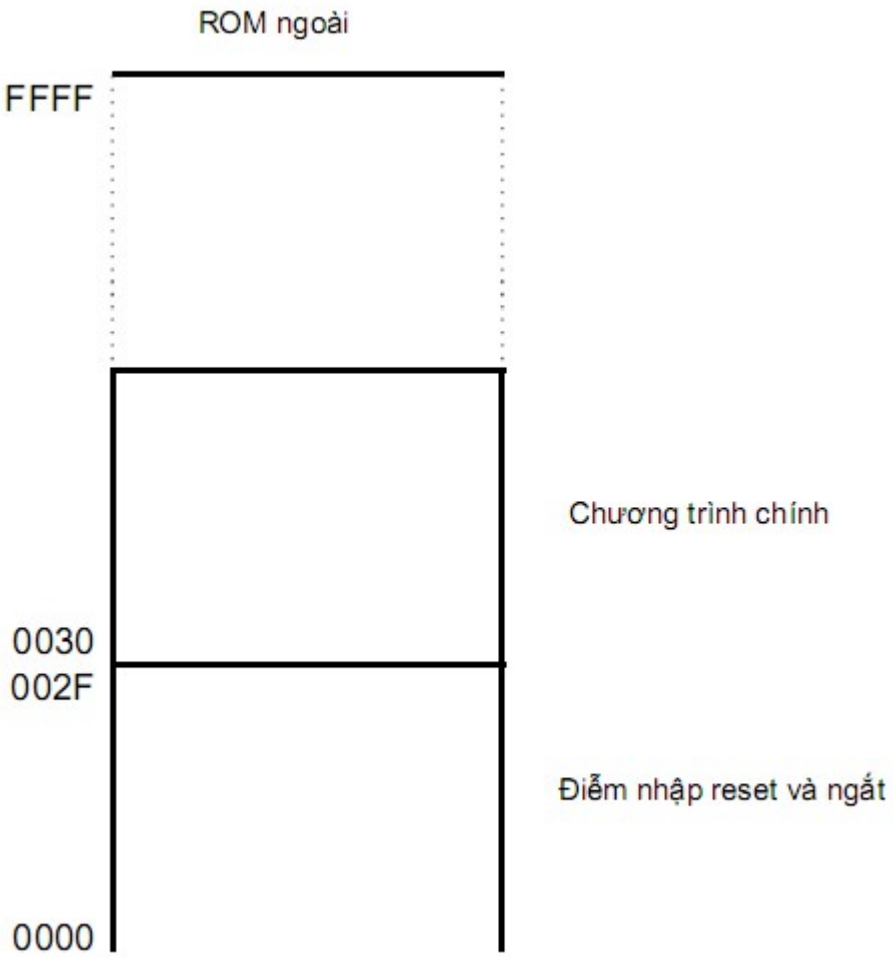
Các chương trình ví dụ sau đây được bắt đầu tại địa chỉ 0000H và chương trình sẽ bắt đầu thực hiện khi reset hệ thống, các chương trình này phát triển cho các ứng dụng thực tế và được lưu vào ROM hoặc EPROM. Khuôn mẫu đề nghị cho một chương trình có sử dụng ngắt như sau:

```
 ORG 0000H ; reset điểm nhập
LJMP MAIN
  .
  . ; các điểm nhập ISR
  .
ORG 0030H ; điểm nhập chương trình chính
MAIN: …….. ; bắt đầu chương trình chính
```

Lệnh đầu tiên nhảy đến địa chỉ 0030H ngay phía trên các vị trí của véc tơ ngắt và cũng là nơi bắt đầu các ISR như ở hình 6.3, chương trình chính bắt đầu tại địa chỉ 0030H

BẢNG 5.4 Các véc tơ ngắt

Ngắt	Cờ	Địa chỉ véc tơ
Reset hệ thống	RST	0000H
0 bên ngoài	IE0	0003H
Timer 0	TF0	000BH
1 bên ngoài	IE1	0013H
Timer 1	TF1	001BH
Cổng nối tiếp	Ri hoặc Ti	0023H
Timer 2	TF2 hoặc EXF2	002BH



Hình 6.3 Tổ chức ngắt trong bộ nhớ

Chương trình phục vụ ngắt kích thước nhỏ

Các chương trình phục vụ ngắt phải bắt đầu gần phía dưới đáy của bộ nhớ chương trình tại các địa chỉ cho trong bảng 6.4. Mặc dù giữa các điểm nhập chỉ có 8 byte nhưng thường cũng đủ để thực hiện các thao tác cần thiết và trở lại chương trình chính từ ISR

Nếu chỉ dùng một tín hiệu ngắt ví dụ timer 0 thì có thể áp dụng chương trình sau đây

```
  ORG  0000H ; reset
  LJMP MAIN
  ORG  000BH ; điểm nhập timer 0
T0ISR: ……   ; bắt đầu ISR của timer 0
  ……..
  ……
 RETI  ; trở về chương trình chính
MAIN: ……..  ; chương trình chính
  ……..
```

Nếu dùng nhiều ngắt thì phải cẩn thận để bảo đảm rằng chúng bắt đầu tại các vị trí đúng (bảng 6.4) và không đè lên các ISR kế tiếp vì trong ví

dụ trên chỉ dùng có một ngắt nên chương trình chính có thể bắt đầu ngay phía dưới lệnh RETI

Chương trình phục vụ ngắt kích thước lớn

Nếu một ISR có kích thước nhiều hơn 8 byte thì có thể chuyển nó đến vị trí khác trong bộ nhớ chương trình hoặc cho lấn sang điểm nhập của ngắt kế tiếp. Điển hình là ISR bắt đầu bằng một lệnh nhảy đến nơi khác trong bộ nhớ chương trình và tại đó ISR có thể trải rộng ra. Hãy xem ví dụ sau chỉ dùng ngắt timer 0

```
    ORG  0000H ; reset
    LJMP MAIN
    ORG  000BH ; điểm nhập timer 0
    LJMP T0ISR
     ORG  0030H; ; trên vùng véc tơ ngắt
MAIN: ……
    ……
T0ISR:  ……..  ; ISR của timer 0
    ……..
    ……..
   RETI  ; trở về chương trình chính
```

Để cho đơn giản các chương trình chỉ làm một việc tại thời điểm bắt đầu, chương trình chính sẽ khởi tạo timer, cổng nối tiếp và các thanh ghi ngắt tương ứng và sau đó không làm gì hết. Toàn bộ công việc được thực hiện trong ISR, sau các dòng lệnh khởi tạo chương trình chính chỉ còn dòng lệnh sau đây:

HERE: SJMP HERE

Khi một ngắt xảy ra, chương trình chính tạm thời dừng lại trong khi ISR đang hoạt động, lệnh RETI ở cuối ISR trả quyền điều khiển về cho chương trình chính và chương trình lại tiếp tục không làm gì cả.

Điều này cũng không có gì lạ, trong nhiều ứng dụng hướng điều khiển phần lớn công việc thường được thực hiện trong chương trình phục vụ ngắt

Ví dụ 4-1: Tạo xung vuông bằng ngắt timer

Viết chương trình dùng timer 0 và các ngắt để tạo xung vuông 1 KHz tại chân P1.0

Các ngắt timer xảy ra khi các thanh ghi THx/TLx tràn và cờ TFx được set, chương trình như sau

```
  ORG 0  ; reset
  LJMP MAIN  ; nhảy qua vùng véc tơ ngắt
  ORG 000BH  ; véc tơ ngắt timer 0
T0ISR: CPL  P1.0    ; đảo port bít
```

```
    RETI
    ORG  0030H
MAIN:  MOV  TMOD,#02H  ; timer 0 mode 2
    MOV  TH0,#-50  ; delay 50µS
   SETB TR0  ; khởi động timer
    MOV  IE,#82H  ; cho phép ngắt timer 0
    SJMP  $    ; không làm gì cả
```

Đây là một chương trình đầy đủ có thể nạp vào EPROM và cài đặt vào UNIKIT để chạy thử. Ngay sau khi reset bộ đếm chương trình sẽ được nạp giá trị 0000H, lệnh đầu tiên được thực hiện là LJMP MAIN và chương trình nhảy qua ISR của timer đến địa chỉ 0030H trong bộ nhớ chương trình, ba lệnh tiếp theo khởi tạo timer 0 ở chế độ tự động nạp lại 8 bít và tràn sau mỗi 50 µS. Lệnh MOV IE,#82H cho phép ngắt timer 0 có nghĩa là mỗi lần tràn của timer sẽ tạo ra một ngắt.

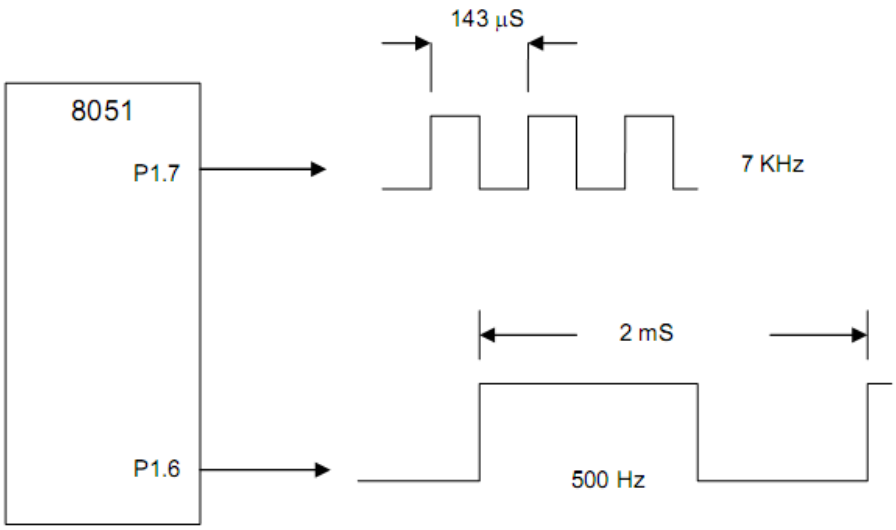
Dĩ nhiên lần tràn đầu tiên sẽ không xảy ra sau 50 µS do chương trình chính rơi vào vòng lặp "không làm gì cả". Cứ sau mỗi 50 µS một ngắt sẽ xuất hiện, chương trình chính bị dừng và chương trình T0ISR hoạt động đảo trạng thái của port bít và sau đó trở về chương trình chính thực hiện tiếp vòng lặp "không làm gì cả" và chờ một ngắt khác sau 50 µS tiếp theo.

Lưu ý là cờ timer TF0 không bị xóa bằng phần mềm, khi các ngắt được cho phép TF0 sẽ bị xóa tự động bởi phần cứng ngay khi CPU trỏ đến ngắt. Địa chỉ trở về trong chương trình chính là vị trí của lệnh SJMP, địa chỉ này được CPU cất vào trong ngăn xếp trước khi trỏ đến véc tơ ngắt và được lấy lại từ ngăn xếp khi thực hiện lệnh RETI ở cuối chương trình ngắt. Do thanh ghi SP đã không được khởi tạo nên địa chỉ mặc định sau khi reset của ngăn xếp là 07H, lệnh PUSH sẽ cất địa chỉ trở về trong RAM nội tại 08H (PCL) và 09H (PCH)

Ví dụ 4-2: Tạo hai xung bằng ngắt

Viết chương trình xử dụng ngắt tạo đồng thời hai xung vuông 7 KHz và 500 Hz tại port P1.7 và P1.6

Cấu hình phần cứng và dạng sóng ra được trình bày ở hình 6.4



Hình 5.4 Dạng sóng

Việc kết hợp các ngõ ra là cực kỳ khó khăn đối với các hệ thống không được điều khiển bằng ngắt. Timer 0 hoạt động ở mode 2 cung cấp xung 7 KHz và timer 1 hoạt động ở mode 1 cung cấp xung 500 Hz. Vì xung 500 Hz có thời gian mức cao là 1 mS và mức thấp cũng là 1 mS nên không thể dùng mode 2, chương trình như sau:

```
    ORG  0
    LJMP MAIN
   ORG 0BH  ; địa chỉ véc tơ timer 0
    JMP  T0ISR
   ORG 1BH  ; địa chỉ véc tơ timer 1
    LJMP T1ISR
    ORG  30H
MAIN:  MOV  TMOD,#12H  ; timer 1=mode 1; timer 0=mode 2
    MOV  TH0,#-71  ; tạo xung 7 KHz
    SETB TR0
   SETB TF1  ; tạo ngắt timer 1
     MOV  IE,#8AH  ; cho phép cả hai ngắt timer
    SJMP $
T0ISR: CPL  P1.7
    RETI
T1ISR: CLR  TR1
     MOV  TH1,#HIGH (-1000)  ; 1 mS mức cao
     MOV  TL1,#LOW (-1000)  ; và 1 mS mức thấp
     SETB TR1
    CPL  P1.6
    RETI
```

Chương trình chính và các ISR được đặt phía trên vùng dành cho các véc tơ ngắt và reset hệ thống, cả hai dạng sóng được tạo ra bởi lệnh CPL. Tuy nhiên, cách tạo thời gian trì hoãn thì có khác nhau.

Vì cặp thanh ghi TL1/TH1 phải được nạp lại sau mỗi lần tràn (có nghĩa là sau mỗi ngắt) nên ISR của timer 1 hoạt động như sau: (a) dừng timer, (b) nạp lại TL1/TH1, (c) khởi động timer và sau đó (d) đảo port bít. Một điểm lưu ý là TL1/TH1 không được khởi tạo tại vị trí bắt đầu của chương trình chính giống như TH0 vì TL1/TH1 phải được khởi tạo lại sau mỗi lần tràn. TF1 được set trong chương trình chính bằng phần mềm để tạo ra ngắt ban đầu ngay khi các ngắt được cho phép để bắt đầu dạng sóng 500 Hz.

ISR của timer 0 chỉ có nhiệm vụ đơn giản là đảo port bít và sau đó trở về chương trình chính, SJMP $ là dạng viết tắt của HERE: SJMP HERE

Ví dụ 4-3: Điều khiển lò sưởi

Xử dụng ngắt thiết kế chương trình điều khiển lò sưởi ổn định tại nhiệt độ $20^0C \pm 1^0C$

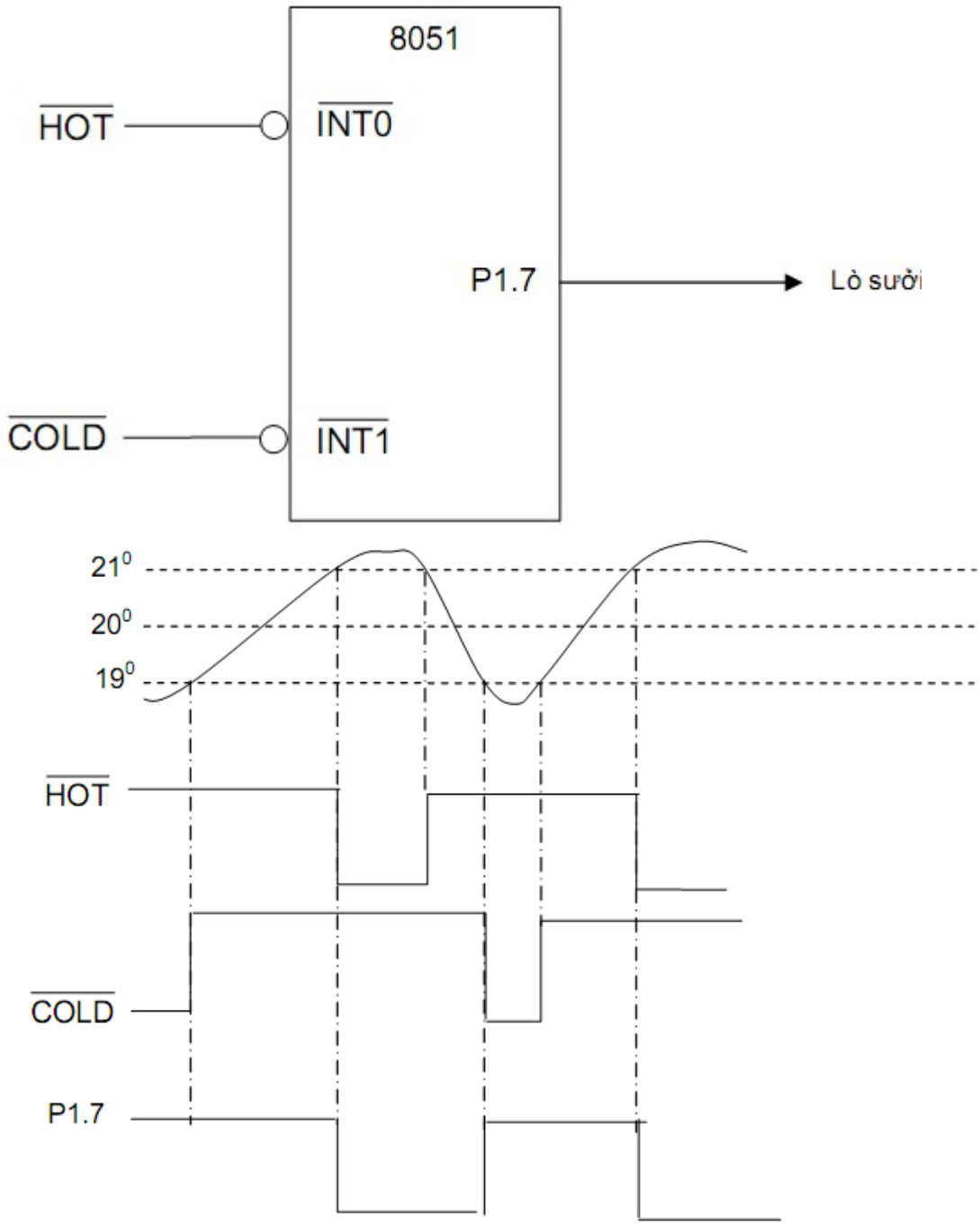
Giả sử rờ le tắt/mở lò sưởi được đặt P1.7

P1.7 = 1 (lò sưởi mở)
P1.7 = 0 (lò sưởi tắt)

Các cảm biến nhiệt độ được nối đến $\overline{INT0}$ và $\overline{INT1}$ và tạo ra các tín hiệu $\overline{HOT}$ và $\overline{COLD}$ theo thứ tự

$\overline{HOT}$ = 0 nếu T> 21^0 C
$\overline{COLD}$ = 0 nếu T< 19^0C

Lò sưởi được mở nếu T< 19^0C và tắt khi T> 21^0C, cấu trúc phần cứng và đồ thị thời gian được trình bày ở hình 4.5



Hình 4.5 Sơ đồ lò sưởi và đồ thị thời gian

```
 ORG  0
 LJMP MAIN
 EX0ISR: CLR  P1.7    ; véc tơ ngắt 0 ngoài tại 0003H; lò sưởi tắt
 RETI
ORG 13H  ; địa chỉ véc tơ 1 ngoài, lò sưởi mở
EX1ISR: SETB  P1.7
 RETI
 ORG  30H
MAIN:  MOV  IE,#85H  ; cho phép các ngắt ngoài
```

```
    SETB IT0  ; tác động cạnh âm
     SETB IT1
    SETB P1.7  ; mở lò sưởi
     JB  P3.2,SKIP  ; nếu T> 21
    CLR P1.7  ; thì tắt lò sưởi
SKIP:  SJMP  $    ; không làm gì cả
```

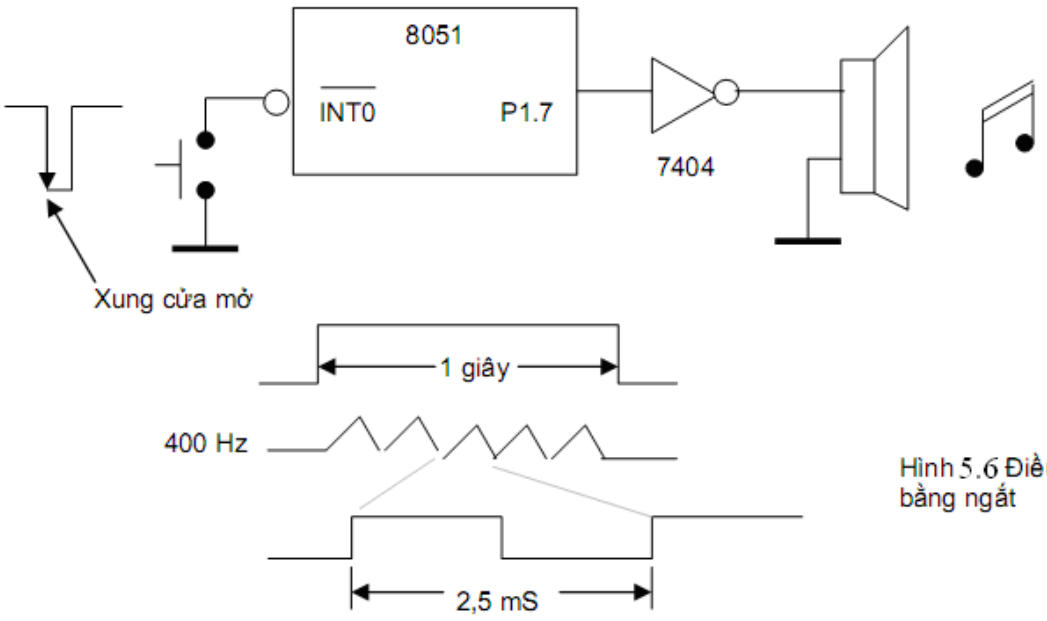
Ba dòng lệnh đầu tiên trong chương trình chính cho phép các ngắt ngoài và làm cho cả hai $\overline{\text{INT0}}$ và $\overline{\text{INT1}}$ trở thành loại tác động cạnh âm. Vì trạng thái hiện tại của các ngõ vào $\overline{\text{HOT}}$ (P3.2) và $\overline{\text{COLD}}$ (P3.1) không biết trước nên cần ba dòng lệnh tiếp theo để tắt hoặc mở lò sưởi tùy theo nhiệt độ. Trước tiên, lò được mở (SETB P1.7) và ngõ vào $\overline{\text{HOT}}$ được lấy mẫu (JB P3.2,SKIP) nếu ngỏ vào này ở mức cao, nhiệt độ chưa cao hơn 21^0C nên lệnh kế tiếp bị bỏ qua và lò vẫn tiếp tục được mở.

Ngược lai, nếu ngõ vào $\overline{\text{HOT}}$ ở mức thấp (T> 21^0C) thì lệnh kế tiếp CLR P1.7 được thực hiện để tắt lò trước khi đi vào vòng lặp "không làm gì cả".

Lưu ý là chỉ dẫn ORG 0003H không cần thiết phải hiện diện ngay trên nhãn EX0ISR do lệnh LJMP MAIN dài 3 byte nên EX0ISR chắc chắn được bắt đầu tại địa chỉ 0003H, điểm nhập của ngắt 0 ngoài.

Ví dụ 4-4: Hệ thống báo động

Dùng ngắt để thiết kế một hệ thống báo động tạo ra âm thanh 400 Hz trong 1 giây (nhờ 1 loa nối vào chân P1.7) mỗi khi cảm biến đặt ở cửa (nỗi đến chân $\overline{\text{INT0}}$) tạo ra một sườn xuống.



Hình 5.6 Điều khiển loa bằng ngắt

Hướng giải quyết là dùng 3 ngắt: Ngắt 0 ngoài (cảm biến cửa), ngắt timer 0 (âm thanh 400 Hz) và ngắt timer 1 (định thời 1 giây). Sơ đồ mạch và đồ thị thời gian trình bày ở hình 6.6

```
        ORG  0
        LJMP MAIN
        LJMP EX0ISR
       ORG 000BH  ; véc tơ timer 0
        LJMP T0ISR
       ORG 001BH  ; véc tơ timer 1
        LJMP T1ISR
        ORG  0030H
MAIN: SETB IT0    ; tác động cạnh âm
        MOV  TMOD,#11H  ; chế độ định thời 16 bít
         MOV  IE,#81H  ; cho phép EX0
        SJMP $

EX0ISR: MOV  R7,#20   ;  20x5000 µS = 1 S
      SETB TF0  ; tạo ngắt timer 0
      SETB TF1  ; tạo ngắt timer 1
      SETB ET0  ; tạo âm thanh trong 1 S
        SETB ET1
        RETI
T0ISR: CLR  TR0
        DJNZ R7,SKIP  ; nếu chưa đủ 20 lần, thoát
       CLR ET0  ; nếu đủ, kết thúc âm thanh
        CLR  ET1
        LJMP EXIT
SKIP:  MOV  TH0,#HIGH(-50000)  ; trì hoãn 0,05 S
        MOV  TL0,#LOW(-50000)
        SETB TR0
EXIT: RETI

T1ISR: CLR  TR1
        MOV  TH1,#HIGH(1250)
        MOV  Tl1,#LOW(-1250)
        CPL  P1.7
        SETB TR1
        RETI

        END
```

Chương trình trên gồm 5 phần phân biệt: Vị trí các véc tơ ngắt, chương trình chính và 3 chương trình phục vụ ngắt. Các vị trí véc tơ ngắt chứa các lệnh LJMP để chuyển điều khiển đến các ISR tương ứng. Chương trình chính bắt đầu tại địa chi3 0030H chỉ gồm 4 lệnh. Lệnh SETB IT0 cho phép ngõ vào ngắt nối với cảm biến cửa được kích bởi cạnh âm.

Lệnh MOV TMOD,#11H xác định chế độ hoạt động của cả hai timer là chế độ định thời 16 bít, chỉ có ngắt 0 ngoài được bắt đầu (MOV IE,#81H) khi cửa mở là điều kiện cần phải có trước khi một ngắt nào đó được chấp nhận. Lệnh cuối cùng SJMP $ đưa chương trình chính vào vòng lặp "không làm gì cả" khi trạng thái mở cửa được phát hiện (cạnh âm tại $\overline{\text{INT0}}$) thì ngắt 0 ngoài sẽ được tạo ra.

Chương trình phục vụ ngắt EX0ISR bắt đầu bằng việc nạp hằng số 20 cho R7 rồi set cờ tràn của cả hai timer để buộc các ngắt định thời xuất hiện. Tuy nhiên, các ngắt timer chỉ xuất hiện khi các bít tương ứng trong thanh ghi IE cho phép. Hai lệnh kế tiếp SETB ET0 và SETB ET1 cho phép các ngắt bộ định thời, cuối cung EX0ISR trở về chương trình chính bằng lệnh RETI.

Timer tạo ra khoảng thời gian trì hoãn 1 S và timer 1 tạo ra âm thanh 400 Hz. Sau khi chương trình EX0ISR kết thúc, các ngắt timer được lập tức tạo ra và được thực hiện sau khi thực hiện 1 lệnh SJMP $. Do tác dụng của chuỗi pooling nên ngắt của timer 0 được phục vụ trước tiên. Khoảng thời gian 1 S được tạo ra bằng cách lập trình để lặp lại 20 lần khoảng thời gian định thời 50000 µS, thanh ghi R7 hoạt động như một bộ đếm.

Chương trình phục vụ ngắt T0ISR hoạt động như sau: Trước tiên, timer 0 được điều khiển dừng và thanh ghi R7 bị giãm 1, tiếp theo TH0/TL0 được nạp lại giá trị -50000, timer 0 chạy trở lại và ngắt được kết thúc. Ở lần ngắt thứ 20, R7 được giảm xuống 0 (đã trôi qua 1 S), các ngắt của cả hai timer bị ngăn (CLR ET0, CLR ET1) và ngắt kết thúc.

Không còn ngắt do bộ định thời tạo ra nữa cho đến khi phát hiện cửa mở một lần nữa. Âm thanh 400 Hz được lập trình bằng cách sử dụng ngắt timer 1, tần số 400 Hz tương đương chu kỳ là 2500 µS với 1250 µS mức cao và 1250 µS mức thấp. Chương trình T1ISR chỉ đơn giản nạp -1250 cho TH1/TL1, đảo port bít để điều khiển loa và kết thúc.

BÀI TẬP THỰC HÀNH

1. Viết lại chương trình ở ví dụ 1 để ngăn các ngắt và kết thúc khi có một phím bất kỳ được ấn.

2. Viết chương trình dùng ngắt tạo xung vuông 1 KHz tại chân P1.7

3. Dùng ngắt viết chương trình tạo xung vuông 7 KHz với chu kỳ làm việc là 30% tại chân P1.6

4. Kết hợp ví dụ 4.1 và 4.3 thành một chương trình duy nhất

5. Viết lại ví dụ 4.3 để gửi 1 ký tự trong 1 S

TÀI LIỆU THAM KHẢO

- ThS. Nguyễn Đình Phú. *Giáo trình Vi điều khiển*. Nhà xuất bản KHKT. 2006
- Th.S Lê Xứng – Nguyễn Bá Hội. *Kỹ thuật vi điều khiển.* Trường ĐHBK tpHCM, 2008