Cấu trúc bus

Bus địa chỉ của họ vi điều khiển 8051 gồm 16 đường tín hiệu (thường gọi là bus địa chỉ 16 bit). Với số lượng bit địa chỉ như trên, không gian nhớ của chip được mở rộng tối đa là 2^{16} = 65536 địa chỉ, tương đương 64K.

Bus dữ liệu của họ vi điều khiển 8051 gồm 8 đường tín hiệu (thường gọi là bus dữ liệu 8 bit), đó là lý do tại sao nói 8051 là họ vi điều khiển 8 bit. Với độ rộng của bus dữ liệu như vậy, các chip họ 8051 có thể xử lý các toán hạng 8 bit trong một chu kỳ lệnh.

Bộ nhớ chương trình

Vi điều khiển họ 8051 có không gian bộ nhớ chương trình là 64K địa chỉ, đó cũng là dung lượng bộ nhớ chương trình lớn nhất mà mỗi chip thuộc họ này có thể có được. Bộ nhớ chương trình của các chip họ 8051 có thể thuộc một trong các loại: ROM, EPROM, Flash, hoặc không có bộ nhớ chương trình bên trong chip. Tên của từng chip thể hiện chính loại bộ nhớ chương trình mà nó mang bên trong, cụ thể là vài ví dụ sau:

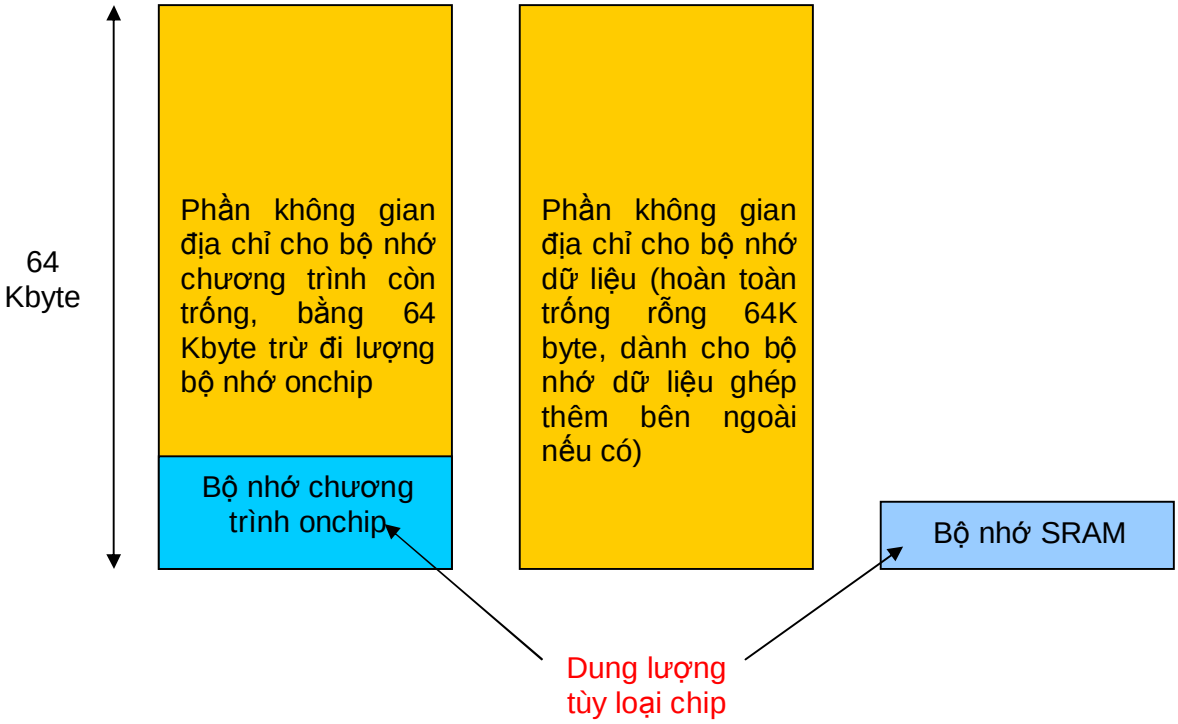
STT	Tên chip	ROM	EPROM	Flash
1	8051	4 Kbyte	x	x
2	8052	8 Kbyte	x	x
3	8031	x	x	x
4	8032	x	x	x
5	87C51	x	4 Kbyte	x
6	87C52	x	8 Kbyte	x
7	AT89C51 / AT89S51	x	x	4 Kbyte
8	AT89C52 / AT89S52	x	x	8 Kbyte

Bộ nhớ dữ liệu

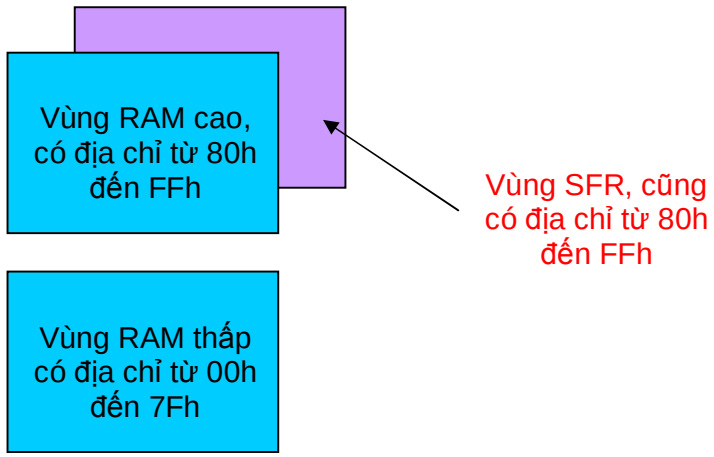
Vi điều khiển họ 8051 có không gian bộ nhớ dữ liệu là 64K địa chỉ, đó cũng là dung lượng bộ nhớ dữ liệu lớn nhất mà mỗi chip thuộc họ này có thể có được (nếu phối ghép một cách chính tắc, sử dụng các đường tín hiệu của bus địa chỉ và dữ liệu). Bộ nhớ dữ liệu của các chip họ 8051 có thể thuộc một hay hai loại: SRAM hoặc EEPROM. Bộ nhớ dữ liệu SRAM được tích hợp bên trong mọi chip thuộc họ vi điều khiển này, có dung lượng khác nhau tùy loại chip, nhưng thường chỉ khoảng vài trăm byte. Đây chính là nơi chứa các biến trung gian trong quá trình hoạt động của chip. khi mất điện, do bản chất của SRAM mà giá trị của các biến này cũng bị mất theo. Khi có điện trở lại, nội dung của các ô nhớ chứa các biến này cũng là bất kỳ, không thể xác định trước. Bên cạnh bộ nhớ loại SRAM, một số chip thuộc họ 8051 còn có thêm bộ nhớ dữ liệu loại EEPROM với dung lượng tối đa vài Kbyte, tùy từng loại chip cụ thể. Dưới đây là một vài ví dụ về bộ nhớ chương trình của một số loại chip thông dụng thuộc họ 8051.

STT	Tên chip	Bộ nhớ SRAM	Bộ nhớ EEPROM
1	AT89C51	128 byte	0
2	AT89C52	256 byte	0
3	AT89C2051	128 byte	0
4	AT89S51	128 byte	0
5	AT89S52	256 byte	0
6	AT89S8252	256 byte	2048 byte

Tổng quát về bộ nhớ của 8051, ta có thể thấy mỗi chip 8051 gồm có những bộ nhớ sau:



Đối với các chip có bộ nhớ SRAM 128 byte thì địa chỉ của các byte SRAM này được đánh số từ 00h đến 7Fh. Đối với các chip có bộ nhớ SRAM 256 byte thì địa chỉ của các byte SRAM được đánh số từ 00h đến FFh. Ở cả hai loại chip, SRAM có địa chỉ từ 00h đến 7Fh được gọi là vùng RAM thấp, phần có địa chỉ từ 80h đến FFh (nếu có) được gọi là vùng RAM cao.
Bên cạnh các bộ nhớ, bên trong mỗi chip 8051 còn có một tập hợp các thanh ghi chức năng đặc biệt (SFR – Special Function Register). Các thanh ghi này liên quan đến hoạt động của các ngoại vi onchip (các cổng vào ra, timer, ngắt ...). Địa chỉ của chúng trùng với dải địa chỉ của vùng SRAM cao, tức là cũng có địa chỉ từ 80h đến FFh.



Vậy khi truy cập vào một địa chỉ thuộc dải từ 00h đến 7Fh thì sẽ truy cập đến ô nhớ thuộc vùng RAM thấp. Tuy nhiên khi truy cập đến một địa chỉ x thuộc dải từ 80h đến FFh thì xảy ra vấn đề cần giải quyết: sẽ truy cập đến thanh ghi SFR ở địa chỉ x hay truy cập đến ô nhớ ở địa chỉ x của vùng RAM cao? Nhà sản xuất quy định rằng, trong trường hợp này, nếu kiểu truy cập sử dụng chế độ địa chỉ trực tiếp thì sẽ truy cập vào vùng SFR, ngược lại nếu kiểu truy cập sử dụng chế độ địa chỉ gián tiếp thì sẽ truy cập vào vùng RAM cao.

Bản đồ các thanh ghi SFR

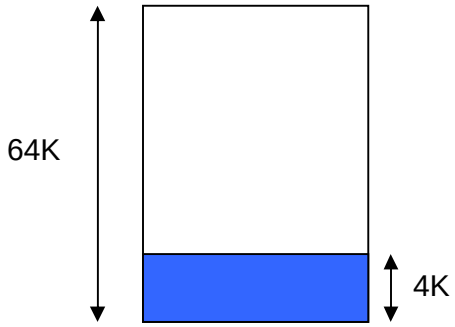
Table 5-1. AT89S52 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								0AFH
0A0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX		0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0		8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXX0000	87H

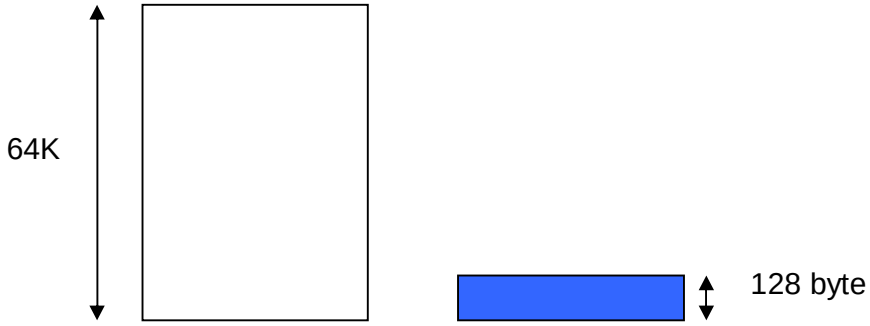
Nhắc lại về bộ nhớ của 8501

8051 có 2 không gian địa chỉ dành cho bộ nhớ chương trình và bộ nhớ dữ liệu riêng biệt. Cả 2 không gian này đều có 16bit địa chỉ, do đó có thể chứa được tối đa 2^{16} = 64K = 65536 ô nhớ mỗi loại.

Bộ nhớ onchip của 8051 gồm có 4 Kbyte bộ nhớ chương trình (ROM, EPROM, EEPROM hoặc Flash tùy loại biến thể) và 128byte bộ nhớ dữ liệu (RAM). 4 Kbyte bộ nhớ chương trình onchip nằm trong không gian địa chỉ 64 Kbyte dành cho bộ nhớ chương trình (thuộc dải địa chỉ từ 0x0000 đến 0x0FFF).



Ngược lại, 128 byte RAM onchip lại không nằm trong không gian địa chỉ 64 Kbyte dành cho bộ nhớ dữ liệu. Ta có thể tưởng tượng không gian bộ nhớ chương trình là một chiếc thùng được lấp đầy 1/16 dung tích bởi 4 Kbyte bộ nhớ onchip, còn không gian bộ nhớ dữ liệu là một thùng to dung tích 64 Kbyte rỗng hoàn toàn và một hộp nhỏ dung tích 128 byte (địa chỉ từ 0x00 đến 0x7F) nằm riêng rẽ bên cạnh.



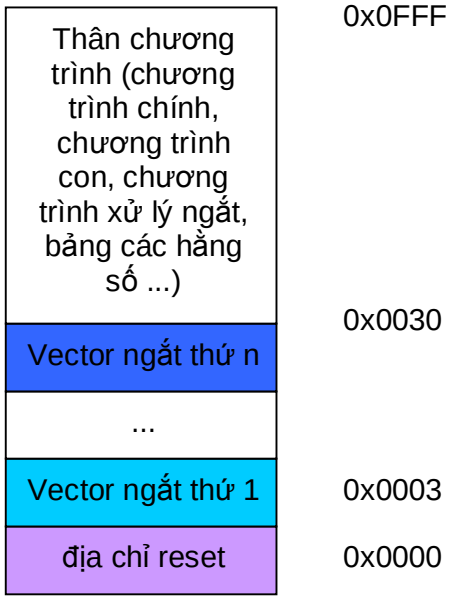
Bộ nhớ chương trình dùng để chứa mã của chương trình nạp vào chip. Mỗi lệnh được mã hóa bởi 1 hay vài byte, dung lượng của bộ nhớ chương trình phản ánh số lượng lệnh mà bộ nhớ có thể chứa được. Địa chỉ đầu tiên của bộ nhớ chương trình (0x0000) chính là địa chỉ Reset của 8051. Ngay sau khi reset (do tắt bật nguồn, do mức điện áp tại chân RESET bị kéo lên 5V...), CPU sẽ nhảy đến thực hiện lệnh đặt tại địa chỉ này trước tiên, luôn luôn là như vậy. Phần còn trống trong không gian chương trình không dùng để làm gì cả. Nếu muốn mở rộng bộ nhớ chương trình, ta phải dùng bộ nhớ chương trình bên ngoài có dung lượng như ý muốn. Tuy nhiên khi dùng bộ nhớ chương trình ngoài, bộ nhớ chương trình onchip không dùng được nữa, bộ nhớ chương trình ngoài sẽ chiếm dải địa chỉ ngay từ địa chỉ 0x0000.

Bộ nhớ dữ liệu RAM onchip thường dùng để chứa các biến tạm thời trong quá trình vi điều khiển hoạt động, đó cũng là nơi dành cho ngăn xếp hoạt động.

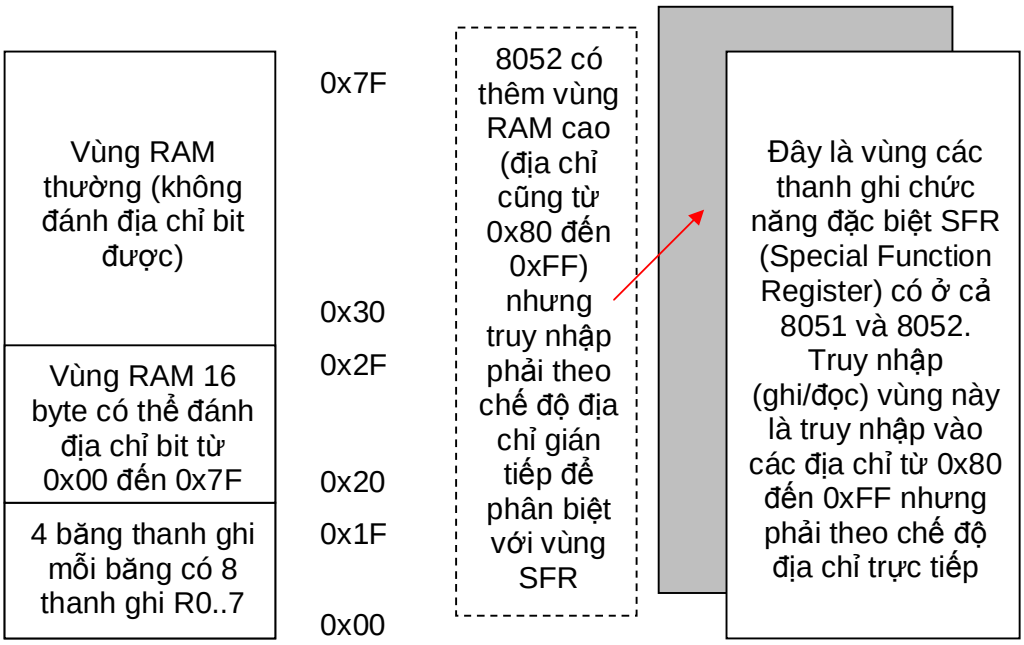
Không gian dữ liệu 64Kbyte được để trống hoàn toàn và chỉ dùng được khi ghép nối với bộ nhớ dữ liệu bên ngoài. Khi ghép nối thêm bộ nhớ dữ liệu bên ngoài, dung lượng của các bộ nhớ này sẽ chiếm dần các vị trí trong không gian, tuy nhiên không hề ảnh hưởng đến 128byte RAM onchip.

Ngăn xếp trong 8051 liên quan đến một thanh ghi tên là con trỏ ngăn xếp SP (Stack Pointer). Thanh ghi này luôn trỏ vào đỉnh của ngăn xếp, tức là nó chứa địa chỉ của vị trí ngay sát vị trí có thể lưu địa chỉ/dữ liệu tiếp theo vào. Khi cất 1 byte địa chỉ/dữ liệu vào ngăn xếp, SP tự động tăng lên 1 đơn vị sau đó mới cất địa chỉ/dữ liệu vào ô nhớ có địa chỉ bằng với giá trị của SP sau khi đã tăng. Khi lấy 1 byte địa chỉ/dữ liệu ra khỏi ngăn xếp, giá trị sẽ được lấy ra sau đó SP mới tự động trừ đi 1 đơn vị. Giá trị sau khi reset của SP là 0x07, do đó quy định ngăn xếp sẽ cất dữ liệu từ địa chỉ 0x08 trở đi. Tuy nhiên do đặc tính hoạt động bành trướng theo chiều tăng địa chỉ mà ngăn xếp thường được bố trí lên vùng trên cùng của bộ nhớ RAM onchip để tránh tranh chấp với các biến lưu trong RAM.

Mô tả bộ nhớ chương trình của 8051:



Mô tả bộ nhớ dữ liệu RAM của 8051:



Cổng vào/ra song song (Parrallel I/O Port) trong 8051

8051 có 4 cổng vào ra song song, có tên lần lượt là P0, P1, P2 và P3. Tất cả các cổng này đều là cổng vào ra hai chiều 8bit. Các bit của mỗi cổng là một chân trên chip, như vậy mỗi cổng sẽ có 8 chân trên chip.

Hướng dữ liệu (dùng cổng đó làm cổng ra hay cổng vào) là độc lập giữa các cổng và giữa các chân (các bit) trong cùng một cổng. Ví dụ, ta có thể định nghĩa cổng P0 là cổng ra, P1 là cổng vào hoặc ngược lại một cách tùy ý, với cả 2 cổng P2 và P3 còn lại cũng vậy. Trong cùng một cổng P0, ta cũng có thể định nghĩa chân P0.0 là cổng vào, P0.1 lại là cổng ra tùy ý.

Liên quan đến mỗi cổng vào/ra song song của 8051 chỉ có một thanh ghi SFR (thanh ghi chức năng đặc biệt) có tên trùng với tên của cổng. Ta có các thanh ghi P0 dùng cho cổng P0, thanh ghi P1 dùng cho cổng P1 … Đây là các thanh ghi đánh địa chỉ đến từng bit (bit addressable), do đó ta có thể dùng các lệnh tác động bit đối với các bit của các thanh ghi này. Mỗi thanh ghi này gồm 8 bit tương ứng với các chân (bit) của cổng đó. Khi một chân (bit) cổng nào đó được dùng làm cổng vào thì trước đó bit tương ứng trong thanh ghi SFR phải được đặt ở mức 1. Nếu một chân (bit) cổng nào đó được dùng làm cổng ra thì giá trị của bit tương ứng trong thanh ghi SFR sẽ là giá trị lôgic muốn đưa ra chân cổng đó. Nếu muốn đưa ra mức lôgic cao (điện áp gần 5V), bit tương ứng trong thanh ghi phải được đặt bằng 1, hiển nhiên nếu muốn đưa ra mức lôgic thấp (điện áp gần 0V) thì bit tương ứng trong thanh ghi phải được đặt bằng 0. Như đã nói ở trên, các bit trong thanh ghi cổng có thể được đặt bằng 1/0 mà không làm ảnh hưởng đến các bit còn lại trong cổng đó bằng cách dùng các lệnh **setb** (đặt lên 1) hay **clr** (đặt về 0).

Sau khi đặt một chân cổng làm cổng vào, ta có thể dùng các lệnh kiểm tra bit để đọc vào và kiểm tra các mức lôgic của mạch ngoài đang áp vào là mức 0 hay mức 1. Các lệnh này là **jb** (nhảy nếu bit bằng 1), **jnb** (nhảy nếu bit bằng 0).

Mỗi cổng có cấu trúc gồm một latch (chính là các bit của thanh ghi cổng), mạch lái đầu ra (output driver) và mạch đệm đầu vào (input buffer).

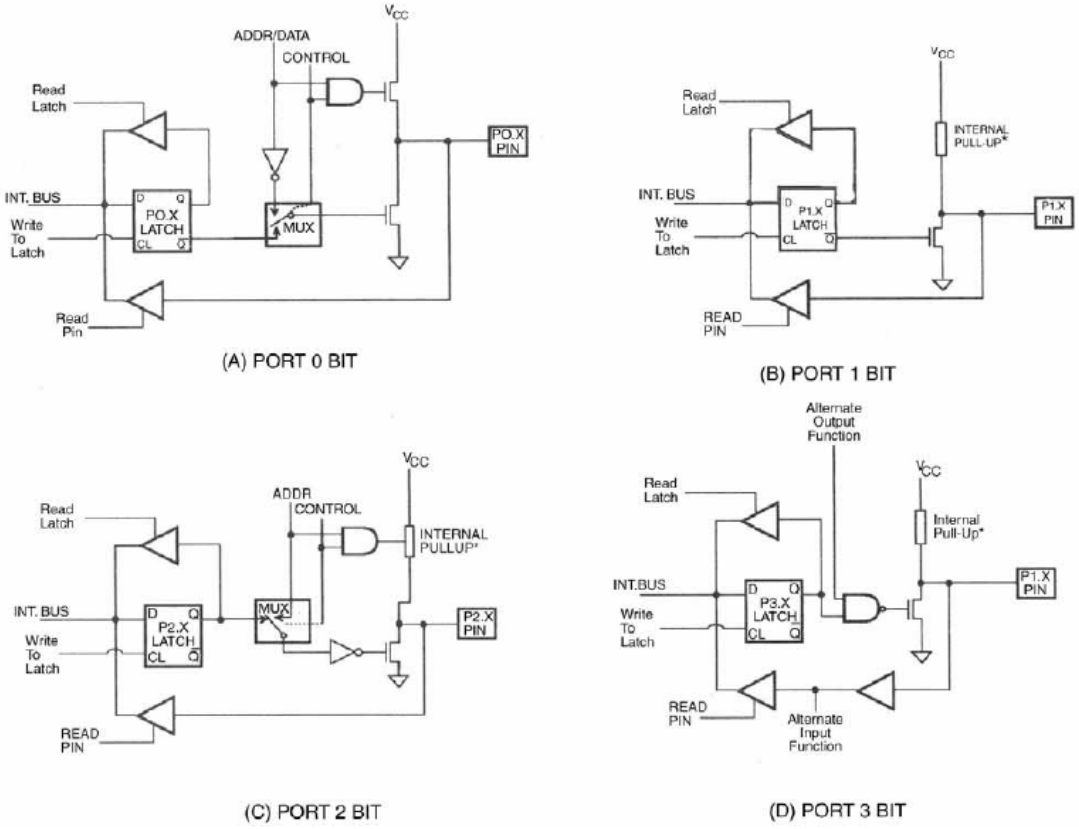
Ngoài chức năng vào/ra thông thường, một số cổng còn được tích hợp thêm chức năng của một số ngoại vi khác. Xem bảng liệt kê sau:

Port Pin	Alternate Function
(1)P1.0	T2 (Timer/Counter 2 external input) (If Timer 2 available)
(1)P1.1	T2EX (Timer/Counter 2 capture/reload trigger) (If Timer 2 available)
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt)
P3.3	$\overline{INT1}$ (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	$\overline{WR}$ (external Data memory write strobe)
P3.7	$\overline{RD}$ (external Data memory read strobe)

Các chân cổng P1.0 và P1.1 được tích hợp với các tín hiệu của timer2 trong trường hợp chip là 8052.

Khi dùng với các chức năng của các ngoại vi, chân cổng tương ứng phải được đặt lên 1. Nếu không các tín hiệu sẽ luôn bị ghim ở mức 0.

Sơ đồ của mạch của một chân cổng:



Cổng P0 không có điện trở treo cao (pullup resistor) bên trong, mạch lái tạo mức cao chỉ có khi sử dụng cổng này với tính năng là bus dồn kênh địa chỉ/dữ liệu. Như vậy với chức năng ra thông thường, P0 là cổng ra open drain, với chức năng vào, P0 là cổng vào cao trở (high impedance). Nếu muốn sử dụng cổng P0 làm cổng vào/ra thông thường, ta phải thêm điện trở pullup bên ngoài. Giá trị điện trở pullup bên ngoài thường từ 4K7 đến 10K.

Các cổng P1, P2 và P3 đều có điện trở pullup bên trong, do đó có thể dùng với chức năng cổng vào/ra thông thường mà không cần có thêm điện trở pullup bên ngoài. Thực chất, điện trở pullup bên trong là các FET, không phải điện trở tuyến tính thông thường, tuy vậy nhưng khả năng phun dòng ra của mạch lái khi đầu ra ở mức cao (hoặc khi là đầu vào) rất nhỏ, chỉ khoảng 100 micro Ampe. Trong datasheet của AT89S5x (một trong những biến thể của họ 8051 do Atmel sản xuất) có thống kê số liệu như sau:

Symbol	Parameter	Condition	Min	Max	Units
V_{OH}	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V

Theo đó, nếu ta thiết kế để các cổng phải cung cấp cho tải ở đầu ra mức cao một lượng dòng điện I_{OH} = 60 micro Ampe thì mức điện áp ở đầu ra V_{OH} sẽ bị kéo sụt xuống, chỉ có thể đảm bảo từ 2.4V trở lên bởi nhà sản xuất, không thể cao sát với 5V như lý thuyết.

Trong khi đó, khả năng nuốt dòng của mạch lái khi đầu ra ở mức thấp lại cao hơn rất nhiều, có thể đạt từ vài đến hàng chục mili Ampe.

Symbol	Parameter	Condition	Min	Max	Units
V_{OL}	Output Low Voltage[1] (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
V_{OL1}	Output Low Voltage[1] (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V

Như vậy, khi thiết kế với các phần tử bên ngoài, ta nên để ý đến đặc tính vào/ra của các chân cổng. Ví dụ khi dùng để ghép nối với LED đơn hoặc LED 7 thanh, ta nên thiết kế chân cổng nuốt dòng từ LED để làm LED sáng (cổng nối với Cathode của LED), không nên thiết kế chân cổng phun dòng cho LED để làm LED sáng (cổng nối với Anode của LED).

Cơ chế ngắt của 8051

8051 chỉ có một số lượng khá ít các nguồn ngắt (interrupt source) hoặc có thể gọi là các nguyên nhân ngắt. Mỗi ngắt có một vector ngắt riêng, đó là một địa chỉ cố định nằm trong bộ nhớ chương trình, khi ngắt xảy ra, **CPU sẽ tự động nhảy đến thực hiện lệnh nằm tại địa chỉ này**. Bảng tóm tắt các ngắt trong 8051 như sau:

STT	Tên ngắt	Mô tả	Cờ ngắt	Thanh ghi chứa cờ	Vector ngắt
1	INT0	Ngắt ngoài 0 khi có tín hiệu tích cực theo kiểu đã chọn ở chân P3.2	IE0	TCON	0x0003
2	Timer0	Ngắt tràn timer0 khi giá trị timer0 tràn từ giá trị max về giá trị min	TF0	TCON	0x000B
3	INT1	Ngắt ngoài 1 khi có tín hiệu tích cực theo kiểu đã chọn ở chân P3.3	IE1	TCON	0x0013
4	Timer1	Ngắt tràn timer1 khi giá trị timer1 tràn từ giá trị max về giá trị min	TF1	TCON	0x001B
5	Serial Port	Ngắt cổng nối tiếp khi vi điều khiển nhận hoặc truyền xong một byte bằng cổng nối tiếp	TI, RI	SCON	0x0023

Với 8052, ngoài các ngắt trên còn có thêm ngắt của timer2 (do vi điều khiển này có thêm timer2 trong số các ngoại vi onchip).

Mỗi ngắt được dành cho một vector ngắt kéo dài 8byte. Về mặt lý thuyết, nếu chương trình đủ ngắn, mã tạo ra chứa đủ trong 8 byte, người lập trình hoàn toàn có thể đặt phần chương trình xử lý ngắt ngay tại vector ngắt. Tuy nhiên trong hầu hết các trường hợp, chương trình xử lý ngắt có dung lượng mã tạo ra lớn hơn 8byte nên tại vector ngắt, ta chỉ đặt lệnh nhảy tới chương trình xử lý ngắt nằm ở vùng nhớ khác. Nếu không làm vậy, mã chương trình xử lý ngắt này sẽ lấn sang, đè vào vector ngắt kế cận.

Liên quan đến ngắt chủ yếu có hai thanh ghi là thanh ghi IE và thanh ghi IP.

Table 13-1. Interrupt Enable (IE) Register

(MSB)							(LSB)
EA	–	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	Serial Port interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.
User software should never write 1s to reserved bits, because they may be used in future AT89 products.		

Để cho phép một ngắt, bit tương ứng với ngắt đó và bit EA phải được đặt bằng 1. Thanh ghi IE là thanh ghi đánh địa chỉ bit, do đó có thể dùng các lệnh tác động bit để tác động riêng rẽ lên từng bit mà không làm ảnh hưởng đến giá trị các bit khác. Cờ ngắt hoạt động độc lập với việc cho phép ngắt, điều đó có nghĩa là cờ ngắt sẽ tự động đặt lên bằng 1 khi có sự kiện gây ngắt xảy ra, bất kể sự kiện đó có được cho phép ngắt hay không. Do vậy, trước khi cho phép một ngắt, ta nên xóa cờ của ngắt đó để đảm bảo sau khi cho phép, các sự kiện gây ngắt trong quá khứ không thể gây ngắt nữa. Ví dụ trước khi cho phép ngắt timer0 mà timer 0 đã chạy và tràn (dù là tràn một hay nhiều lần) thì cờ TF0 sẽ bằng 1, nếu sau đó ta cho phép ngắt timer0 thì sẽ gây ra ngắt ngay do cờ tràn đang bằng 1 (sự kiện tràn gây ngắt trong trường hợp này là tràn trong quá khứ, không phải sự kiện ta quan tâm đến). Vì vậy hãy xóa cờ TF0 trước khi cho phép ngắt tràn timer0.

Ngoại trừ cờ của của ngắt nối tiếp (và cờ của ngắt timer2 trong 8052), các cờ ngắt khác đều tự động được xóa khi CPU thực hiện chương trình phục vụ ngắt. Lý do là ngắt cổng nối tiếp (và ngắt timer2 trong 8052) được gây ra bởi 2 nguyên nhân (có 2 cờ cho mỗi ngắt), khi xảy ra ngắt, người lập trình cần phải kiểm tra xem cờ nào được đặt bằng 1 để phân biệt nguyên nhân gây ra ngắt đó là nguyên nhân nào để xử lý thích hợp. Ví dụ ngắt cổng nối tiếp là ngắt được gây ra bởi 1 trong 2 nguyên nhân: vi điều khiển nhận xong hoặc truyền xong một byte dữ liệu qua cổng nối tiếp. Xảy ra sự kiện nào thì cờ ngắt tương ứng sẽ tự động được đặt lên bằng 1, nếu nhận xong thì cờ RI bằng 1, nếu truyền xong thì cờ TI bằng 1. Trong chương trình xử lý ngắt, người lập trình phải kiểm tra cờ TI hay cờ RI bằng 1 để quyết định xử lý ngắt truyền hay xử lý ngắt nhận. Sau khi kiểm tra,

người lập trình phải viết lệnh xóa cờ đó vì việc này không được CPU thực hiện tự động như các cờ ngắt khác.

Nói đến ngắt không thể không nói đến mức ưu tiên của ngắt. Mức ưu tiên của ngắt ở đây có thể được hiểu là sự phân bậc, quyết định xử lý ngắt nào khi hai hay nhiều ngắt xảy ra. Có 2 cơ chế phân bậc ưu tiên. Thứ nhất là cơ chế phân bậc dành cho các ngắt xảy ra đồng thời, hai ngắt A và B xảy ra cùng một thời điểm nhìn từ phía vi điều khiển. Thứ hai là cơ chế phân bậc dành cho các ngắt xảy ra xen kẽ nhau, trong khi đang xử lý ngắt A thì ngắt B xảy ra, vậy thì trong từng trường hợp, CPU sẽ xử lý ra sao? Hãy xem dưới đây.

Với trường hợp các ngắt xảy ra đồng thời, CPU sẽ xem xét mức ưu tiên của các ngắt đó, từ đó quyết định xử lý ngắt có mức ưu tiên cao hơn trước. Mức ưu tiên trong trường hợp này là mức ưu tiên cứng (được quy định bởi nhà sản xuất, bởi cấu trúc sẵn có của 8051 và người lập trình **không thể thay đổi được**).

Table 2-27. Interrupt Priority Level

	Source	Priority Within Level
1	IE0	(highest)
2	TF0	
3	IE1	
4	TF1	
5	RI + TI	
6	TF2 + EXF2	(lowest)

Nhìn vào bảng trên ta thấy ngắt INT0 là ngắt có mức ưu tiên cao nhất và ngắt timer2 là ngắt có mức ưu tiên thấp nhất trong số các ngắt. Như vậy nếu ngắt ngoài 1 và ngắt timer0 cùng xảy ra một lúc, ngắt ngoài 1 sẽ được CPU xử lý trước, sau đó mới xử lý ngắt timer0.

Với trường hợp xảy ra ngắt xen kẽ, khi CPU đang xử lý ngắt A mà ngắt B xảy ra, CPU sẽ giải quyết theo 2 hướng: tiếp tục xử lý ngắt A nếu mức ưu tiên của ngắt B **không cao hơn** mức ưu tiên của ngắt A, hoặc sẽ dừng việc xử lý ngắt A lại, chuyển sang xử lý ngắt B nếu mức ưu tiên của ngắt B **cao hơn** mức ưu tiên của ngắt A. Mức ưu tiên cho các ngắt trong trường hợp này không phải là mức ưu tiên cứng do nhà sản xuất quy định (tức là không căn cứ vào bảng trên) mà là do người lập trình đặt. Người lập trình có thể dùng thanh ghi IP để quy định mức ưu tiên cho các ngắt ở một trong hai mức: mức cao và mức thấp. Để đặt mức ưu tiên của một ngắt (trong trường hợp xảy ra xen kẽ) ở mức cao, ta đặt bit tương ứng với ngắt đó trong thanh ghi IP bằng 1, mức thấp ứng với giá trị bit = 0.

Thanh ghi IP (Interrupt Priority)

-	-	PT2	PS	PT1	PX1	PT0	PX0

Các bit trong thanh ghi IP tương ứng với các ngắt đúng như trong thanh ghi IE (bit PX0 dành cho ngắt ngoài 0, bit PT0 dành cho ngắt timer 0…)

Một điều dễ nhận ra là nếu một ngắt được đặt mức ưu tiên cao (bit tương ứng trong thanh ghi IP bằng 1) thì sẽ chẳng có ngắt nào có thể xen vào quá trình xử lý nó được nữa.

Nói về mức ưu tiên ngắt, có thể dùng một ví dụ tổng quát sau, giả sử hai ngắt timer0 và ngắt cổng nối tiếp cùng được cho phép (các bit tương ứng và bit EA trong thanh ghi IE được đặt bằng 1), bit PT0 = 0, bit PS = 1 thì:

- Nếu hai ngắt cùng xảy ra, ngắt timer0 sẽ thắng thế và được phục vụ trước.
- Nếu ngắt cổng nối tiếp xảy ra trước và đang được xử lý thì ngắt timer0 nếu có xảy ra cũng không thể chen vào, làm dừng quá trình xử lý ngắt cổng nối tiếp được.
- Nếu ngắt timer0 xảy ra trước và đang được xử lý mà ngắt cổng nối tiếp xảy ra thì CPU sẽ phải dừng việc xử lý ngắt timer0 lại, chuyển sang xử lý ngắt cổng nối tiếp, xử lý xong mới quay lại xử lý tiếp ngắt timer0.

Ngắt ngoài (External Interrupt)

Như đã nói ở trên, 8051 có 2 ngắt ngoài là INT0 và INT1. Ngắt ngoài được hiểu là ngắt được gây ra bởi sự kiện mức lôgic 0 (mức điện áp thấp, gần 0V) hoặc sườn xuống (sự chuyển mức điện áp từ mức cao về mức thấp) xảy ra ở chân ngắt tương ứng (P3.2 với ngắt ngoài 0 và P3.3 với ngắt ngoài 1). Việc lựa chọn kiểu ngắt được thực hiện bằng các bit IT (Interrupt Type) nằm trong thanh ghi TCON. Đây là thanh ghi điều khiển timer nhưng 4 bit LSB (bit0..3) được dùng cho các ngắt ngoài.

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Khi bit ITx = 1 thì ngắt ngoài tương ứng được chọn kiểu là ngắt theo sườn xuống, ngược lại nếu bit ITx = 0 thì ngắt ngoài tương ứng được sẽ có kiểu ngắt là ngắt theo mức thấp. Các bit IE là các bit cờ ngắt ngoài, chỉ có tác dụng trong trường hợp kiểu ngắt được chọn là ngắt theo sườn xuống.

Khi kiểu ngắt theo sườn xuống được chọn thì ngắt sẽ xảy ra duy nhất một lần khi có sườn xuống của tín hiệu, sau đó khi tín hiệu ở mức thấp, hoặc có sườn lên, hoặc ở mức cao thì cũng không có ngắt xảy ra nữa cho đến khi có sườn xuống tiếp theo. Cờ ngắt IE sẽ dựng lên khi có sườn xuống và tự động bị xóa khi CPU bắt đầu xử lý ngắt.

Khi kiểu ngắt theo mức thấp được chọn thì ngắt sẽ xảy ra bất cứ khi nào tín hiệu tại chân ngắt ở mức thấp. Nếu sau khi xử lý xong ngắt mà tín hiệu vẫn ở mức thấp thì lại ngắt tiếp, cứ như vậy cho đến khi xử lý xong ngắt lần thứ n , tín hiệu đã lên mức cao rồi thì thôi không ngắt nữa. Cờ ngắt IE trong trường hợp này không có ý nghĩa gì cả.

Thông thường kiểu ngắt hay được chọn là ngắt theo sườn xuống.

Các timer/counter trong 8051

8051 có 2 timer tên là timer0 và timer1. Các timer này đều là timer 16bit, giá trị đếm max do đó bằng 2^{16} = 65536 (đếm từ 0 đến 65535).

Hai timer có nguyên lý hoạt động hoàn toàn giống nhau và độc lập. Sau khi cho phép chạy, mỗi khi có thêm một xung tại đầu vào đếm, giá trị của timer

sẽ tự động được tăng lên 1 đơn vị, cứ như vậy cho đến khi giá trị tăng lên vượt quá giá trị max mà thanh ghi đếm có thể biểu diễn thì giá trị đếm lại được đưa trở về giá trị min (thông thường min = 0). Sự kiện này được hiểu là sự kiện tràn timer (overflow) và có thể gây ra ngắt nếu ngắt tràn timer được cho phép (bit ETx trong thanh ghi IE = 1).

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

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Khi bit TRx = 1, timerx sẽ đếm, ngược lại khi TRx = 0, timerx sẽ không đếm mặc dù vẫn có xung đưa vào. Khi dừng không đếm, giá trị của timer được giữ nguyên.

Các bit TFx là các cờ báo tràn timer, khi sự kiện tràn timer xảy ra, cờ sẽ được tự động đặt lên bằng 1 và nếu ngắt tràn timer được cho phép, ngắt sẽ xảy ra. Khi CPU xử lý ngắt tràn timerx, cờ ngắt TFx tương ứng sẽ tự động được xóa về 0.

Giá trị đếm 16bit của timerx được lưu trong hai thanh ghi THx (byte cao) và TLx (byte thấp). Hai thanh ghi này có thể ghi/đọc được bất kỳ lúc nào. Tuy nhiên nhà sản xuất khuyến cáo rằng nên dừng timer (cho bit TRx = 0) trước khi ghi/đọc các thanh ghi chứa giá trị đếm.

Các timer có thể hoạt động theo nhiều chế độ, được quy định bởi các bit trong thanh ghi TMOD (không đánh địa chỉ đến từng bit).

7	6	5	4	3	2	1	0
GATE1	C/T1#	M11	M01	GATE0	C/T0#	M10	M00

Bit Number	Bit Mnemonic	Description
7	GATE1	**Timer 1 Gating Control Bit** Clear to enable timer 1 whenever the TR1 bit is set. Set to enable timer 1 only while the INT1# pin is high and TR1 bit is set.
6	C/T1#	**Timer 1 Counter/Timer Select Bit** Clear for timer operation: timer 1 counts the divided-down system clock. Set for Counter operation: timer 1 counts negative transitions on external pin T1.
5	M11	**Timer 1 Mode Select Bits** M11 M01 Operating mode 0 0 Mode 0: 8-bit timer/counter (TH1) with 5-bit prescaler (TL1). 0 1 Mode 1: 16-bit timer/counter. 1 0 Mode 2: 8-bit auto-reload timer/counter (TL1). Reloaded from TH1 at overflow. 1 1 Mode 3: timer 1 halted. Retains count.
4	M01	
3	GATE0	**Timer 0 Gating Control Bit** Clear to enable timer 0 whenever the TR0 bit is set. Set to enable timer/counter 0 only while the INT0# pin is high and the TR0 bit is set.
2	C/T0#	**Timer 0 Counter/Timer Select Bit** Clear for timer operation: timer 0 counts the divided-down system clock. Set for counter operation: timer 0 counts negative transitions on external pin T0.
1	M10	**Timer 0 Mode Select Bit** M10 M00 Operating mode 0 0 Mode 0: 8-bit timer/counter (TH0) with 5-bit prescaler (TL0). 0 1 Mode 1: 16-bit timer/counter. 1 0 Mode 2: 8-bit auto-reload timer/counter (TL0). Reloaded from TH0 at overflow. 1 1 Mode 3: TL0 is an 8-bit timer/counter. TH0 is an 8-bit timer using timer 1's TR0 and TF0 bits.
0	M00	

Để xác định thời gian, người ta chọn nguồn xung nhịp (clock) đưa vào đếm trong timer là xung nhịp bên trong (dành cho CPU). Nguồn xung nhịp này

thường rất đều đặn (có tần số ổn định), do đó từ số đếm của timer người ta có thể nhân với chu kỳ xung nhịp để tính ra thời gian trôi qua. Timer lúc này được gọi chính xác với cái tên "timer", tức bộ định thời.

Để đếm các sự kiện bên ngoài, người ta chọn nguồn xung nhịp đưa vào đếm trong timer là tín hiệu từ bên ngoài (đã được chuẩn hóa về dạng xung vuông 0V/5V). Các tín hiệu này sẽ được nối với các bit cổng có dồn kênh thêm các tính năng T0/T1/T2. Khi có sự kiện bên ngoài gây ra thay đổi mức xung ở đầu vào đếm, timer sẽ tự động tăng lên 1 đơn vị giống như trường hợp đếm xung nhịp bên trong. Lúc này, timer được gọi chính xác với cái tên khác: "counter", tức bộ đếm (sự kiện).

Nhìn vào bảng mô tả thanh ghi TMOD bên trên, ta có thể nhận thấy có 2 bộ 4 bit giống nhau (gồm GATEx, C/Tx, Mx0 và Mx1) dành cho 2 timer0 và 1. Ý nghĩa các bit là như nhau đối với mỗi timer.

Bit **GATEx** quy định việc cho phép timer đếm (run timer). Nếu GATEx = 0, timerx sẽ đếm khi bit TRx bằng 1, dừng khi bit TRx bằng 0. Nếu GATEx = 1, timerx sẽ chỉ đếm khi bit TRx = 1 và tín hiệu tại chân INTx = 1, dừng khi một trong hai điều kiện trên không còn thỏa mãn. Thông thường người ta dùng timer với GATE = 0, chỉ dùng timer với GATE = 1 trong trường hợp muốn đo độ rộng xung vì lúc đó timer sẽ chỉ đếm thời gian khi xung đưa vào chân INTx ở mức cao.

Bit **C/Tx** quy định nguồn clock đưa vào đếm trong timer. Nếu C/Tx = 0, timer sẽ được cấu hình là bộ định thời, nếu C/Tx = 1, timer sẽ được cấu hình là bộ đếm sự kiện.

Hai bit còn lại (Mx0 và Mx1) tạo ra 4 tổ hợp các giá trị (00,01,10 và 11) ứng với 4 chế độ hoạt động khác nhau của timerx. Trong 4 chế độ đó thường chỉ dùng chế độ timer/counter 16bit (Mx1 = 0, Mx0 = 1) và chế độ Auto Reload 8bit timer/counter (Mx1 = 1, Mx0 = 0).

Trong chế độ timer/counter 16bit, giá trị đếm (chứa trong hai thanh ghi THx và TLx) tự động được tăng lên 1 đơn vị mỗi lần nhận được thêm một xung nhịp. Khi giá trị đếm tăng vượt quá giá trị max = 65535 thì sẽ tràn về 0, cờ ngắt TFx được tự động đặt = 1. Chế độ này được dùng trong các ứng dụng đếm thời gian và đếm sự kiện.

Trong chế độ Auto Reload 8bit, giá trị đếm sẽ chỉ được chứa trong thanh ghi TLx, còn giá trị của thanh ghi THx bằng một số n (từ 0 đến 255) do người lập trình đưa vào. Khi có thêm 1 xung nhịp, giá trị đếm trong TLx đương nhiên cũng tăng lên 1 đơn vị như bình thường. Tuy nhiên trong trường hợp này, giá trị đếm lớn nhất là 255 chứ không phải 65535 như trường hợp trên vì timer/counter chỉ còn 8bit. Do vậy sự kiện tràn lúc này xảy ra nhanh hơn, chỉ cần vượt quá 255 là giá trị đếm sẽ tràn. Cờ ngắt TFx vẫn được tự động đặt = 1 như trong trường hợp tràn 16bit. Điểm khác biệt là thay vì tràn về 0, giá trị THx sẽ được tự động nạp lại (Auto Reload) vào thanh ghi TLx, do đó timer/counter sau khi tràn sẽ có giá trị bằng n (giá trị chứa trong THx) và sẽ đếm từ giá trị n trở đi. Chế độ này được dùng trong việc tạo Baud rate cho truyền thông qua cổng nối tiếp.

Để sử dụng timer của 8051, hãy thực hiện các bước sau:

- Quy định chế độ hoạt động cho timer bằng cách tính toán và ghi giá trị cho các bit trong thanh ghi TMOD.
- Ghi giá trị đếm khởi đầu mong muốn vào 2 thanh ghi đếm THx và TLx. Đôi khi ta không muốn timer/counter bắt đầu đếm từ 0 mà từ một giá trị nào đó để thời điểm tràn gần hơn, hoặc chẵn hơn trong tính toán sau này. Ví dụ nếu cho timer đếm từ 15535 thì sau 50000 xung nhịp (tức 50000 micro

giây với thạch anh 12MHz) timer sẽ tràn, và thời gian một giây có thể dễ dàng tính ra khá chính xác = 20 lần tràn của timer (đương nhiên mỗi lần tràn lại phải nạp lại giá trị 15535).
- Đặt mức ưu tiên ngắt và cho phép ngắt tràn timer (nếu muốn).
- Dùng bit TRx trong thanh ghi TCON để cho timer chạy hay dừng theo ý muốn.

Cổng nối tiếp (Serial Port) của 8051

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

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

Thanh ghi SBUF là thanh ghi 8bit chứa dữ liệu truyền hoặc nhận. Về thực chất có hai thanh ghi dữ liệu khác nhau, một dành để chứa dữ liệu truyền đi, một để chứa dữ liệ nhận được. Cả hai thanh ghi này đều có chung một tên là SBUF, tuy nhiên CPU hoàn toàn phân biệt được một cách dễ dàng. Khi ta muốn truyền dữ liệu đi, ta phải ghi vào thanh ghi SBUF (ví dụ viết lệnh **mov SBUF,a**), còn khi muốn đọc kiểm tra dữ liệu nhận về ta phải đọc thanh ghi SBUF (ví dụ viết lệnh **mov a,SBUF**). CPU sẽ căn cứ vào việc thanh ghi SBUF nằm ở vị trí toán hạng đích (toán hạng bên trái) hay toán hạng nguồn (toán hạng bên phải) để quyết định sẽ truy nhập (đọc/ghi) thanh ghi SBUF nào. Người lập trình không cần phải quan tâm xử lý vấn đề này.

Thanh ghi quy định chế độ hoạt động và điều khiển cổng nối tiếp là thanh ghi SCON (đánh địa chỉ bit).

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit SM0, SM1, SM2 quy định chế độ hoạt động của cổng nối tiếp. Thông thường để truyền thông giữa 2 vi điều khiển hoặc giữa 1 vi điều khiển và 1 máy tính, giá trị của bit SM2 được đặt bằng 0. Khi truyền thông theo kiểu mạng đa vi xử lý (multiprocessor communication), SM2 được đặt bằng 1. Hai bit SM0 và SM1 thực sự là các bit quy định chế độ hoạt động của cổng nối tiếp, chúng tạo ra 4 tổ hợp (00,01,10 và 11) ứng với 4 chế độ hoạt động mô tả trong bảng sau.

SM0	SM1	Chế độ	Khung dữ liệu	Baud rate
0	0	0 - Đồng bộ	8 bit SBUF	Fosc/12
0	1	1 - Dị bộ	8 bit SBUF	Thay đổi được
1	0	2 - Dị bộ	8bit SBUF + RB8/TB8	Fosc/32 hoặc Fosc/64
1	1	3 - Dị bộ	8bit SBUF + RB8/TB8	Thay đổi được

Chế độ 0: là chế độ truyền đồng bộ duy nhất. Chân RxD sẽ là tín hiệu truyền/nhận dữ liệu, chân TxD là tín hiệu xung nhịp. Bit LSB (bit 0) của dữ liệu được truyền đi trước tiên. Tốc độ truyền cố định và bằng 1/12 giá trị thạch anh.

Chế độ 1: là chế độ truyền dị bộ 8 bit. Dữ liệu 8 bit được đóng khung bởi một bit Start (= 0) ở đầu và một bit Stop (=1) ở cuối trước khi được truyền đi. Tốc độ truyền thay đổi được theo ý người lập trình.
Chế độ 2: là chế độ truyền dị bộ 9 bit. Dữ liệu 9 bit được ghép thành bởi 8bit trong thanh ghi SBUF và bit RB8 (trường hợp nhận về) hoặc TB8 (trường hợp truyền đi) trong thanh ghi SCON. Ngoài ra các bit Start và Stop vẫn được gắn bình ở đầu và cuối khung truyền. Trong chế độ này, tốc độ truyền chỉ có thể chọn được ở 1 trong 2 mức: 1/32 hoặc 1/64 giá trị của thạch anh (tùy thuộc vào giá trị của bit SMOD trong thanh ghi PCON đã nói ở trên).
Chế độ 3: cũng là chế độ truyền dị bộ 9 bit, khác với chế độ 2 ở chỗ tốc độ truyền có thể thay đổi được theo ý người lập trình như trong chế độ 1.

Bit **REN** trong thanh ghi SCON là bit cho phép nhận dữ liệu. Dữ liệu chỉ được nhận qua cổng nối tiếp khi bit này = 1.

Bit **TB8** là bit dữ liệu thứ 9 trong trường hợp truyền đi 9 bit (8 bit kia trong thanh ghi SBUF).

Bit **RB8** là bit dữ liệu thứ 9 trong trường hợp nhận về 9 bit (8 bit kia trong thanh ghi SBUF).

Bit **TI** là cờ ngắt truyền, báo hiệu việc truyền 1 khung dữ liệu đã hoàn tất.

Bit **RI** là cờ ngắt nhận, báo hiệu việc nhận 1 khung dữ liệu đã hoàn tất.

Để tạo ra tốc độ truyền (Baud rate) của cổng nối tiếp trong 8051, phải dùng đến timer1 ở chế độ Auto Reload 8bit. Giá trị nạp lại chứa trong thanh ghi TH1 được tính toán theo công thức sau (phụ thuộc vào Baud rate mong muốn và giá trị của thạch anh).

$$\text{Modes 1, 3 Baud Rate} = \frac{2^{\text{SMOD}}}{32} \times \frac{\text{Oscillator Frequency}}{12x\,[256 - (\text{TH1})]}$$

Tóm lại để sử dụng cổng nối tiếp của 8051, hãy thực hiện các bước sau:

- Chọn chế độ cho cổng nối tiếp (đồng bộ/dị bộ, 8bit/9bit...), từ đó chọn được giá trị cho các bit trong thanh ghi SCON. Lưu ý xóa các bit TI và RI.
- Chọn tốc độ truyền mong muốn, từ đó tính ra giá trị của thanh ghi TH1. Cho timer1 chạy ở chế độ Auto Reload 8bit (không dùng ngắt tràn timer1).
- Đặt mức ưu tiên ngắt và cho phép ngắt cổng nối tiếp nếu muốn.
- Bắt đầu quá trình truyền dữ liệu bằng một lệnh ghi dữ liệu muốn truyền vào thanh ghi SBUF. Quá trình truyền kết thúc thì cờ TI sẽ tự động đặt lên 1.
- Khi một khung dữ liệu đã được nhận đầy đủ, cờ RI sẽ tự động đặt lên 1 và người lập trình lúc này có thể dùng lệnh đọc thanh ghi SBUF để lấy dữ liệu nhận được ra xử lý.

Tập lệnh của 8051

Trước khi nói về tập lệnh của 8051 phải nhắc tới thanh ghi PSW, là thanh ghi có các bit phản ánh trạng thái hiện thời của CPU.

Table 1-1. PSW: Program Status Word Register

(MSB)							(LSB)
CY	AC	F0	RS1	RS0	OV	-	P

Symbol	Position	Name and Significance
CY	PSW.7	Carry flag
AC	PSW.6	Auxiliary Carry flag. (For BCD operations.)
F0	PSW.5	Flag 0 (Available to the user for general purposes.)
RS1	PSW.4	Register bank Select control bits 1 & 0. Set/cleared by software to determine working register bank (see Note).
RS0	PSW.3	
OV	PSW.2	Overflow flag.
-	PSW.1	(reserved)
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate and odd/even number of "one" bits in the accumulator, i.e., even parity.

Note:
The contents of (RS1, RS0) enable the working register banks as follows:
(0.0)-Bank 0(00H-07H)
(0.1)-Bank 1(08H-0FH)
(1.0)-Bank 2(10H-17H)
(1.1)-Bank 3(18H-1FH)

Các bit cờ trong thanh ghi này được tự động cập nhật thường xuyên ngay sau từng lệnh được CPU thực hiện.

Ngoài các bit cờ, các bit RS0 và RS1 cho phép người lập trình chọn bank thanh ghi R dùng hiện tại là bank 0, 1, 2 hay 3.

Các chế độ địa chỉ

Chế độ địa chỉ trực tiếp: chỉ dùng cho các toán hạng nằm trong vùng RAM thấp và vùng thanh ghi chức năng đặt biệt SFR.

Chế độ địa chỉ gián tiếp: dùng cho các toán hạng nằm trong RAM (cả vùng cao, vùng thấp và RAM ngoài), không dùng cho vùng SFR. Địa chỉ của toán hạng sẽ được chứa trong một thanh ghi con trỏ (R0 hoặc R1 đối với RAM trong, DPTR đối với RAM ngoài). Thay vì xuất hiện trực tiếp ngay trong câu lệnh như chế độ trực tiếp, toán hạng không xuất hiện mà chỉ có thanh ghi con trỏ đại diện đứng ra. Đặc điểm dễ nhận ra là các thanh ghi này xuất hiện luôn kèm theo ký tự "@" phía trước.

Chế độ địa chỉ thanh ghi: dùng cho trường hợp toán hạng là 1 trong 8 thanh ghi Ri trong bank thanh ghi được chọn. Các thanh ghi R trong trường hợp này không có ký tự "@" phía trước.

Chế độ địa chỉ thanh ghi cụ thể: là chế độ địa chỉ áp dụng cho những lệnh chỉ tác động lên một thanh ghi duy nhất nào đó.

Chế độ địa chỉ tức thời: là chế độ địa chỉ khi mà giá trị của toán hạng được nêu ra rõ ràng ngay trong câu lệnh. Đặc điểm dễ dàng nhận ra là các toán hạng này luôn kèm theo ký tự "#" phía trước.
Chế độ địa chỉ chỉ số: chỉ dành cho lệnh movc, là lệnh đọc bộ nhớ chương trình, thường dùng cho việc tra bảng. Trong câu lệnh này cũng xuất hiện ký tự "@" nhưng sau đó là một toán hạng tạo thành bởi phép cộng một thanh ghi 16bit (PC hoặc DPTR) với thanh ghi Acc. Thanh ghi 16bit chứa địa chỉ của đầu mảng, còn thanh ghi A chứa độ lệch của ô nhớ cần đọc so với đầu bảng. Giá trị đọc ra sẽ được ghi đè vào thanh ghi A (xem mô tả tập lệnh để biết chi tiết hơn).

Khi lập trình hợp ngữ cho 8051, lưu ý các điều sau:

- *viết đúng mã lệnh mà nhà sản xuất quy định, đừng bao giờ nghĩ đến chuyện sáng tác mã lệnh. Trình hợp ngữ sẽ không chấp nhận bất kỳ một biến tấu nào, dù là nhỏ nhất..*
- *chỉ sử dụng một trong các chế độ địa chỉ dành cho lệnh đó. Không phải lệnh nào cũng cho phép sử dụng với tất cả 6 chế độ địa chỉ kể trên, thậm chí có những lệnh chỉ cho phép sử dụng với 1 chế độ địa chỉ duy nhất.*
- *tuân theo các cú pháp mà chế độ địa chỉ đã chọn yêu cầu.*
- *đặc biệt lưu ý các lệnh có liên quan đến các cờ như các lệnh cộng có nhớ (ADDC), lệnh trừ (SUBB), các lệnh nhảy có điều kiện (JZ, JNZ, JC, JNC, CJNE ...) Các cờ luôn được cập nhật giá trị mới một cách tự động sau mỗi lệnh được thực hiện, do đó cần nắm được các tình huống của giá trị các cờ trước khi viết các lệnh trên.*
- *viết đúng thứ tự toán hạng. Toán hạng nguồn nằm bên phải, toán hạng đích nằm bên trái, giữa các toán hạng ngăn cách nhau bởi dấu ",".*

Các ký hiệu dùng trong việc mô tả tập lệnh

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

Các lệnh tác động đến các cờ trong thanh ghi trạng thái

STT	*Lệnh*	*Cờ CY*	*Cờ OV*	*Cờ AC*
0	ADD	X	X	X
1	ADDC	X	X	X
2	SUBB	X	X	X
3	MUL	0	X	
4	DIV	0	X	
5	DA	X		
6	RRC	X		
7	RLC	X		
8	SETB C	1		
9	CLR C	0		
10	CPL C	X		
11	ANL C,<bit>	X		
12	ORL C,<bit>	X		
13	MOV C,<bit>	X		
14	CJNE	X		

0: cờ bị xóa về "0".
1: cờ bị set lên "1".
X: cờ bị thay đổi tùy theo kết quả của việc thực hiện lệnh.

Các lệnh tính toán số học

STT	*Cú pháp lệnh*		*Mô tả*	*Số byte mã hóa*	*Số chu kỳ clock*
	Mã lệnh	*Toán hạng*			
1	ADD	A,Rn	A = A + Rn	1	12
2	ADD	A,direct	A = A + direct	2	12
3	ADD	A,@Ri	A = A + @Ri	1	12
4	ADD	A,#data	A = A + #data	2	12
5	ADDC	A,Rn	A = A + Rn + C	1	12
6	ADDC	A,direct	A = A + direct + C	2	12
7	ADDC	A,@Ri	A = A + @Ri + C	1	12
8	ADDC	A,#data	A = A + #data + C	2	12
9	SUBB	A,Rn	A = A – Rn – C	1	12
10	SUBB	A,direct	A = A – direct – C	2	12
11	SUBB	A,@Ri	A = A – @Ri – C	1	12
12	SUBB	A,#data	A = A – #data – C	2	12
13	INC	A	A = A + 1	1	12
14	INC	Rn	Rn = Rn + 1	1	12
15	INC	Direct	direct = direct + 1	2	12
16	INC	@Ri	@Ri = @Ri + 1	1	12
17	DEC	A	A = A – 1	1	12
18	DEC	Rn	Rn = Rn – 1	1	12

19	DEC	Direct	direct = direct – 1	2	12
20	DEC	@Ri	@Ri = @Ri – 1	1	12
21	INC	Dptr	dptr = dptr + 1	1	24
22	MUL	AB	B:A = A*B	1	48
23	DIV	AB	A/B = A(thương) + B (dư)	1	48
24	DA	A	Hiệu chỉnh thập phân số liệu trong thanh ghi A	1	12

Các lệnh thực hiện các phép toán lôgic

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

Các lệnh trao đổi dữ liệu

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

26	XCH	A,direct	toán hạng còn lại	2	12
27	XCH	A,@Ri		1	12
28	XCHD	A,@Ri	Hoán đổi 4 bit thấp giữa A và một ô nhớ trong Ram tại địa chỉ = Ri	1	12

Các lệnh thao tác xử lý đại số Boolean

STT	***Cú pháp lệnh***		***Mô tả***	***Số byte mã hóa***	***Số chu kỳ clock***
	Mã lệnh	***Toán hạng***			
1	CLR	C	Xóa cờ C về 0	1	12
2	CLR	Bit	Xóa bit về 0	2	12
3	SETB	C	Đặt cờ C = 1	1	12
4	SETB	Bit	Đặt bit = 1	2	12
5	CPL	C	Đảo giá trị của cờ C	1	12
6	CPL	Bit	Đảo giá trị của bit	2	12
7	ANL	C,bit	C = (C)and(bit)	2	24
8	ANL	C,/bit	C = (C)and(đảo của bit)	2	24
9	ORL	C,bit	C = (C)or(bit)	2	24
10	ORL	C,/bit	C = (C)or(đảo của bit)	2	24
11	MOV	C,bit	C = bit	2	12
12	MOV	Bit,C	Bit = C	2	24
13	JC	<rel>	nhảy đến nhãn <rel> nếu C = 1	2	24
14	JNC	<rel>	nhảy đến nhãn <rel> nếu C = 0	2	24
15	JB	Bit, <rel>	nhảy đến nhãn <rel> nếu bit= 1	3	24
16	JNB	Bit, <rel>	nhảy đến nhãn <rel> nếu bit= 0	3	24
17	JBC	Bit, <rel>	nhảy đến nhãn <rel> nếu bit = 1 và sau đó xóa luôn bit về 0	3	24

Các lệnh rẽ nhánh chương trình

STT	Cú pháp lệnh		Mô tả	Số byte mã hóa	Số chu kỳ clock
	Mã lệnh	Toán hạng			
1	ACALL	<addr11>	gọi chương trình con	3	24
2	LCALL	<addr16>	gọi chương trình con	3	24
3	RET		trở về từ chương trình con	1	24
4	RETI		trở về từ chương trình phục vụ ngắt	1	24
5	AJMP	<addr11>	nhảy đến nhãn	2	24
6	LJMP	<addr16>	nhảy đến nhãn	3	24
7	SJMP	<rel>	nhảy đến nhãn	2	24
8	JMP	@A+DPTR	nhảy đến địa chỉ = A+DPTR	1	24
9	JZ	<rel>	nhảy đến nhãn nếu A = 0	2	24
10	JNZ	<rel>	nhảy đến nhãn nếu A ≠ 0	2	24
11	CJNE	A,direct,<rel>	So sánh và nhảy đến nhãn nếu A ≠ direct	3	24
12	CJNE	A,#data,<rel>	So sánh và nhảy đến nhãn nếu A ≠ data	3	24
13	CJNE	Rn,#data,<rel>	So sánh và nhảy đến nhãn nếu Rn ≠ data	3	24
14	CJNE	@Ri,#data,<rel>	So sánh và nhảy đến nhãn nếu byte có địa chỉ = Ri có nội dung khác với data	3	24
15	DJNZ	Rn,<rel>	Giảm Rn đi 1 và nhảy đến nhãn nếu chưa giảm về 0	2	24
16	DJNZ	direct,<rel>	Giảm direct đi 1 và nhảy đến nhãn nếu chưa giảm về 0	3	24
17	NOP		Không làm gì cả	1	12

Cấu trúc một chương trình hợp ngữ cho 8051 (sử dụng trình hợp ngữ Reads51)

```
; đầu chương trình, khai báo file chứa địa chỉ của các thanh ghi SFR
#include      <sfr51.inc>
; định nghĩa tên gọi cho các chân cổng vào/ra (nếu muốn)
#define       led1          P1.0
#define       led2          P1.1
...
; khai báo các biến dạng byte (nếu có)
var1          data          0x30
var2          data          0x31
...
; khai báo các biến dạng bit (nếu có)
flag1         bit           0x00
flag2         bit           0x01
...
; định nghĩa các hằng số (nếu có)
constant1     equ           123
constant2     equ           456
...
; tạo mã đặt tại địa chỉ reset
              org           0x0000
              ajmp          main
; tạo mã đặt tại các vector ngắt (nếu sử dụng ngắt)
              org           0x0003
              ljmp          ChuongTrinhXuLyNgatNgoai0
              org           0x000B
              ljmp          ChuongTrinhXuLyNgatTimer0
...
; đặt địa chỉ đầu cho chương trình chính
              org           0x0030
main:
              ; bắt đầu viết các lệnh cho chương trình chính từ đây
              mov           SP,#0x6F
              ; viết các thủ tục khởi tạo hệ thống
              ...
              ; viết thân chương trình chính (vòng lặp chính)
      main_loop:
              ...
              sjmp          main_loop
; viết các chương trình con và các chương trình xử lý ngắt (nếu có)
ChuongTrinhCon1:
              ; các lệnh xử lý của chương trình con 1
              ...
              ; kết thúc bằng lệnh ret
              ret
ChuongTrinhCon2:
```

```
                ; các lệnh xử lý của chương trình con 2
                ...
                ; kết thúc bằng lệnh ret
                ret
...
ChuongTrinhXuLyNgatNgoai0:
                ; các lệnh xử lý của chương trình xử lý ngắt ngoài 0
                ...
                ; kết thúc bằng lệnh reti
                reti
ChuongTrinhXuLyNgatTimer0:
                ; các lệnh xử lý của chương trình xử lý ngắt timer 0
                ...
                ; kết thúc bằng lệnh reti
                reti
...
; định nghĩa các bảng hằng số lưu sẵn trong bộ nhớ chương trình
Bang1:
db              0,1,0x02,0x86
Bang2:
db              156,235,8,9
...
; chỉ dẫn báo hiệu kết thúc toàn bộ đoạn chương trình
end
```

Chú ý:

- Các chữ viết sát lề, kết thúc bằng dấu ":" là các nhãn, đó có thể là đầu một chương trình con hoặc đơn giản chỉ là một nhãn phụ trong thân một chương trình nào đó.
- các chữ màu đỏ là chỉ dẫn hoặc mã lệnh nên phải viết đúng theo.
- Các chữ màu xanh là các chữ được đặt sau dấu ";", sẽ được coi là các câu chú thích và trình hợp ngữ sẽ bỏ qua, nội dung tùy ý nhưng phải trên một dòng, nếu kéo dài xuống dòng khác thì phải thêm dấu ";" khác vào trước phần chú thích của dòng đó.
- Các chữ màu đen hầu hết là các tên nhãn hay tên biến và người lập trình tùy ý đặt.
- Giữa các tên (nhãn) phải sự thống nhất khi khai báo và lúc sử dụng trong câu lệnh, không được khai báo một kiểu, dùng trong lệnh lại kiểu khác đi.

Chúc học tốt!