UBND HUYỆN CỦ CHI
TRƯỜNG TRUNG CẤP NGHỀ CỦ CHI

GIÁO TRÌNH

MÔN HỌC/MÔ ĐUN: LẬP TRÌNH VI ĐIỀU KHIỂN
NGHỀ: ĐIỆN TỬ CÔNG NGHIỆP
TRÌNH ĐỘ: TRUNG CẤP NGHỀ

Ban hành kèm theo Quyết định số: 48/QĐ-TCNCC ngày 04 tháng 10 năm 2021 của Hiệu trưởng Trường Trung Cấp Nghề Củ Chi

Tp. Hồ Chí Minh, năm 2021

TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

Để thực hiện biên soạn giáo trình đào tạo nghề Điện tử công nghiệp ở trình độ TCN, giáo trình Mô đun Vi điều khiển là một trong những giáo trình mô học đào tạo được biên soạn theo nội dung chương trình khung được Sở Lao động - Thương binh và Xã hội TPHCM và Trường trung cấp nghề Củ Chi ban hành dành cho hệ Trung Cấp Nghề Điện tử công nghiệp.

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

Khi biên soạn, người biên soạn đã cố gắng cập nhật những kiến thức mới có liên quan đến nội dung chương trình đào tạo và phù hợp với mục tiêu đào tạo, nội dung lý thuyết và thực hành được biên soạn gắn với nhu cầu thực tế học tập đồng thời có tính thực tiễn cao. Nội dung giáo trình được biên soạn với dung lượng thời gian đào tạo 90 giờ gồm có:

Bài 1: Tổng quan về vi xử lý

Bài 2: Tập lệnh 89C51

Bài 3: Timer/Counter

Bài 4: Ngắt và chương trình phục vụ ngắt

Bài 5: Bài tập ứng dụng

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

Mặc dù đã cố gắng tổ chức biên soạn để đáp ứng được mục tiêu đào tạo nhưng không tránh được những khiếm khuyết. Rất mong nhận được đóng góp ý kiến của các thầy, cô giáo, bạn đọc để người biên soạn sẽ hiệu chỉnh hoàn thiện hơn.

Tp. HCM, ngày 30 tháng 10 năm 2022

Giáo viên biên soạn

Nguyễn Đoan Thùy Như Hồng Ngọc

MỤC LỤC

MÔN ĐUN VI ĐIỀU KHIỂN

Mã mô đun: MĐ 19

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

- Vị trí của mô đun:

+ Mô đun được bố trí dạy cuối chương trình sau khi học xong các môn học cơ bản như linh kiện điện tử, đo lường điện, kỹ thuật xung số, điện tử công suất.

+ Học song song các môn đun cơ sở ngành

- Tính chất của mô đun:

+ Là môn đun bắt buộc

- Ý nghĩa của mô đun:

Mô đun giúp người học có kiến thức về lập trình và xử dụng các phần mền để lập trình

- Vai trò của mô đun:

Là môn đun ngành giúp người học thiết kế được các sản phẩm ứng dụng vi điều khiển trong cuộc sống và trong công nghiệp

Mục tiêu của mô đun:

* Kiến thức

+ Giải thích được nguyên lý làm việc các hệ điều khiển ứng dụng vi xử lý.

* Kỹ năng

+ Cải tiến được chức năng của vi điều khiển theo yêu cầu.

+ Phát triển được các hệ điều khiển trên cơ sở khối trung tâm là vi xử lý.

* Năng lực tự chủ và trách nhiệm

+ Rèn luyện tính cẩn thận khoa học

+ Rèn luyện tính tỉ mỉ, cẩn thận, chính xác, khoa học và tác phong công nghiệp

BÀI 1: TỔNG QUAN VỀ VI XỬ LÝ

Giới thiệu:

Kỹ thuật vi xử lý là phần cứng chỉ đóng vai trò thứ yếu, phần mềm (chương trình) đóng vai trò chủ đạo đối với các chức năng cần thực hiện. Nhờ vậy vi xử lý có sự mềm dẻo hóa trong các chức năng của mình. Ngày nay vi xử lý có tốc độ tính toán rất cao và khả năng xử lý rất lớn.

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

Mục tiêu của bài:

- Trình bày được cấu trúc chung của vi xử lý, phân biệt vi xử lý dựa trên các đặc điểm cơ bản.
- Mô tả được cấu trúc cụ thể của bộ vi điều khiển 89C51.
- Phân tích nguyên lý hoạt động các khối chức năng của bộ vi điều khiển 89C51.
- Phân biệt các hệ thống đếm.
- Rèn luyện tính chính xác, nghiêm túc trong học tập và trong thực hiện công việc.

Nội dung chính:

1. Giới thiệu chung

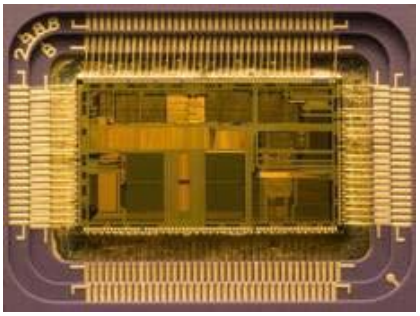
1.1. Cấu trúc chung.

1.1.1. Tổng quan

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

Trước khi xuất hiện các bộ vi xử lý, các CPU được xây dựng từ các mạch tích hợp cỡ nhỏ riêng biệt, mỗi mạch tích hợp chỉ chứa khoảng vào chục tranzito. Do đó, một CPU có thể là một bảng mạch gồm hàng ngàn hay hàng triệu vi mạch tích hợp.

Ngày nay, công nghệ tích hợp đã phát triển, một CPU có thể tích hợp lên một hoặc vài vi mạch tích hợp cỡ lớn, mỗi vi mạch tích hợp cỡ lớn chứa hàng ngàn hoặc hang triệu tranzito. Nhờ đó công suất tiêu thụ và giá thành của bộ vi xử lý đã giảm đáng kể.



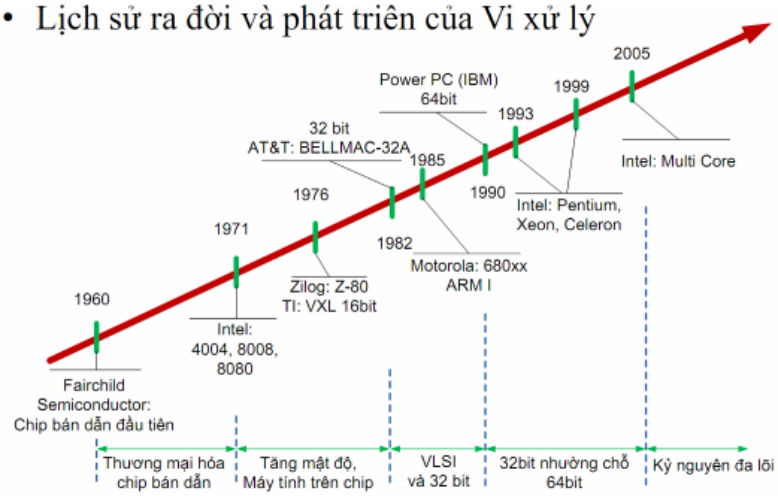
Hình 1.1.1: Bộ vi xử lý Intel 80486DX2

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

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

Hầu hết các vi điều khiển ngày nay được xây dựng dựa trên kiến trúc Harvard, kiến trúc này định nghĩa bốn thành phần cần thiết của một hệ thống nhúng. Những thành phần này là lõi CPU, bộ nhớ chương trình (thông thường là ROM hoặc bộ nhớ Flash), bộ nhớ dữ liệu (RAM), một hoặc vài bộ định thời và các cổng vào/ra để giao tiếp với các thiết bị ngoại vi và các môi trường bên ngoài - tất cả các khối này được thiết kế trong một vi mạch tích hợp. Vi điều khiển khác với các bộ vi xử lý đa năng ở chỗ là nó có thể hoạt động chỉ với vài vi mạch hỗ trợ bên ngoài.

1.1.2. Lịch sử phát triển của các bộ xử lý



Hình 1.1.2: Lịch sử phát triển của VXL

- Thế hệ 1 (1971 - 1973): vi xử lý 4 bit, đại diện là 4004, 4040, 8080 (Intel) hay IPM-16 (National Semiconductor).

+ Độ dài word thường là 4 bit (có thể lớn hơn).

+ Tốc độ 10 - 60 μs / lệnh với tần số xung nhịp 0.1 - 0.8 MHz.

+ Tập lệnh đơn giản và phải cần nhiều vi mạch phụ trợ.

- Thế hệ 2 (1974 - 1977): vi xử lý 8 bit, đại diện là 8080, 8085 (Intel) hay Z80.

+ Tập lệnh phong phú hơn.

+ Địa chỉ có thể đến 64 KB. Một số bộ vi xử lý có thể phân biệt 256 địa chỉ cho thiết bị ngoại vi.

+ Sử dụng công nghệ NMOS hay CMOS.

+ Tốc độ 1 - 8 μs / lệnh với tần số xung nhịp 1 - 5 MHz

- Thế hệ 3 (1978 - 1982): vi xử lý 16 bit, đại diện là 68000/68010 (Motorola) hay 8086/ 80286/ 80386 (Intel)

+ Tập lệnh đa dạng với các lệnh nhân, chia và xử lý chuỗi.

+ Địa chỉ bộ nhớ có thể từ 1 - 16 MB và có thể phân biệt tới 64KB địa chỉ cho ngoại vi

+ Sử dụng công nghệ HMOS.

+ Tốc độ 0.1 - 1 μs / lệnh với tần số xung nhịp 5 - 10 MHz.

- Thế hệ 4: vi xử lý 32 bit 68020/68030/68040/68060 (Motorola) hay 80386/80486 (Intel) và vi xử lý 32 bit Pentium (Intel)

+ Bus địa chỉ 32 bit, phân biệt 4 GB bộ nhớ. + Có thể dùng thêm các bộ đồng xử lý (coprocessor).

+ Có khả năng làm việc với bộ nhớ ảo.

+ Có các cơ chế pipeline, bộ nhớ cache.

+ Sử dụng công nghệ HCMOS.

- Thế hệ 5: vi xử lý 64 bit

1.1.3. Vi xử lý và vi điều khiển.

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

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

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

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

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

™Trong các sản phẩm dân dụng:

- Nhà thông minh:

f + Cửa tự động

f + Khóa số

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

ƒ + Điều khiển các thiết bị từ xa (qua điều khiển, qua tiếng vỗ tay...)
ƒ + Điều tiết hơi ẩm, điều tiết nhiệt độ, điều tiết không khí, gió
ƒ + Hệ thống vệ sinh thông minh...

- Trong quảng cáo:

ƒ + Các loại biển quảng cáo nháy chữ
ƒ + Quảng cáo ma trận LED (một màu, 3 màu, đa màu)
ƒ + Điều khiển máy cuốn bạt quảng cáo...

- Các máy móc dân dụng

ƒ + Máy điều tiết độ ẩm cho vườn cây
ƒ + Buồng ấp trứng gà/vịt
ƒ + Đồng hồ số, đồng hồ số có điều khiển theo thời gian

- Các sản phẩm giải trí

ƒ + Máy nghe nhạc
ƒ + Máy chơi game
ƒ + Đầu thu kỹ thuật số, đầu thu set-top-box...

™ - Trong các thiết bị y tế:

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

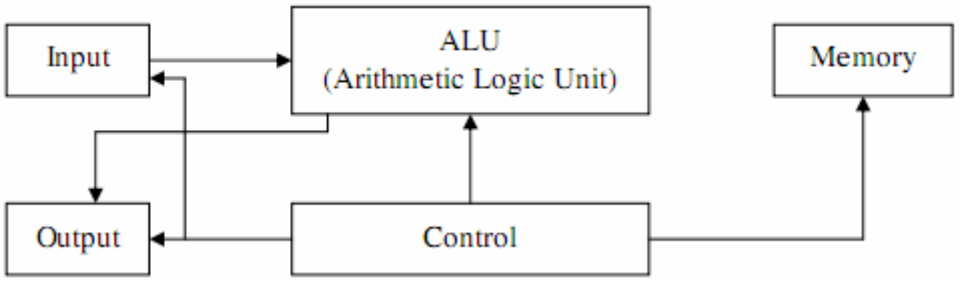
+ Máy cắt/mài kính

+ Máy chụp chiếu (city, X-quang...)

Các sản phẩm công nghiệp:

- Điều khiển động cơ
- Điều khiển số (PID, mờ...)
- Đo lường (đo điện áp, đo dòng điện, áp suất, nhiệt độ...)
- Cân băng tải, cân toa xe, cân ô tô...
- Máy cán thép: điều khiển động cơ máy cán, điều khiển máy quấn thép...
- Làm bộ điều khiển trung tâm cho RoBot
- Ổn định tốc độ động cơ
- Đếm sản phẩm của 1 nhà máy, xí nghiệp…
- Máy vận hành tự động (dạng CNC)

1.2. Đặc tính chung của vi xử lý.



Hình 1.1.3: Sơ đồ khối một máy tính cổ điển

- ALU (đơn vị logic số học): thực hiện các bài toán cho máy tính bao gồm: +, *, /,- phép toán logic …

- Control (điều khiển): điều khiển, kiểm soát các đường dữ liệu giữa các thành phần của máy tính.

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

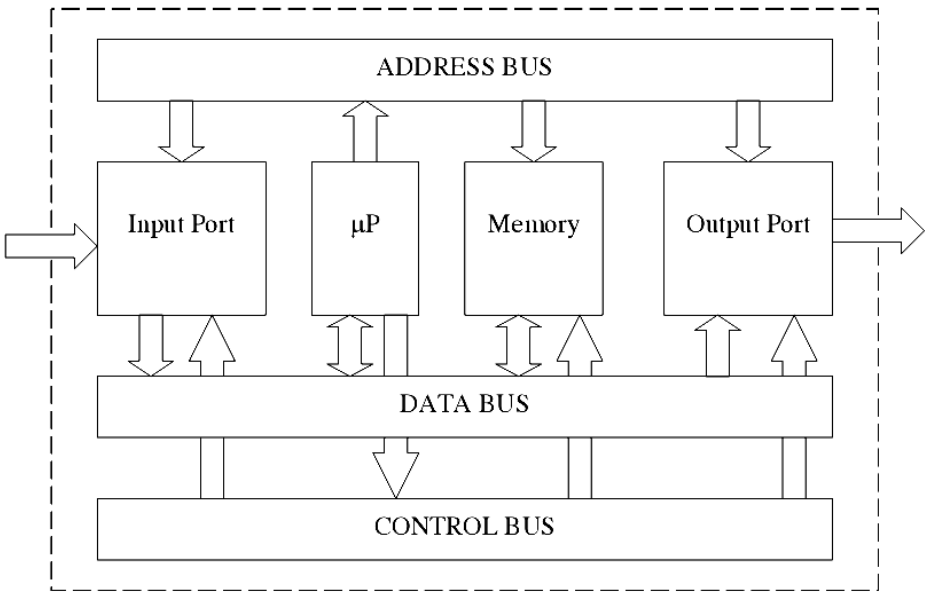
- Input (nhập), Output (Xuất): xuất nhập dữ liệu (còn gọi là thiết bị ngoại vi).

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

- Đơn vị xử lý trung tâm CPU (Central Processing Unit).
- Các bộ nhớ (Memories).
- Các cổng vào/ra (song song (Parallel I/O Ports), nối tiếp (Serial I/O Ports))
- Các bộ đếm/bộ định thời (Timers).
- Hệ thống BUS (Địa chỉ, dữ liệu, điều khiển)

Ngoài ra với mỗi loại vi điều khiển cụ thể còn có thể có thêm một số phần cứng khác như bộ biến đổi tương tự-số ADC, bộ biến đổi số-tương tự DAC, các mạch điều chế dạng sóng WG, điều chế độ rộng xung PWM…

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



Hình 1.1.4: Sơ đồ khối hệ vi xử lý

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

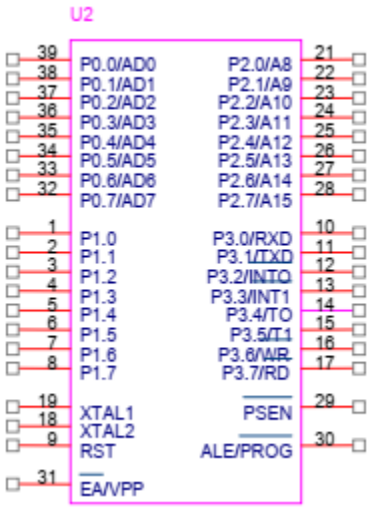
Tập hợp các đường tín hiệu có cùng chức năng gọi là các bus. Như vậy ta có các bus địa chỉ, bus dữ liệu và bus điều khiển.

2. Vi điều khiển 89C51.

2.1. Chức năng các chân ra.

Mặc dù các thành viên của họ MSC-51 có nhiều kiểu đóng vỏ khác nhau, chẳng hạn như hai hàng chân DIP (Dual In-Line Pakage) dạng vỏ dẹt vuông QFP (Quad Flat

Pakage) và dạng chíp không có chân đỡ LLC (Leadless Chip Carrier) và đều có 40 chân cho các chức năng khác nhau như vào ra I/0, đọc , ghi , địa chỉ, dữ liệu và ngắt. Tuy nhiên, vì hầu hết các nhà phát triển chính dụng chíp đóng vỏ 40 chân với hai hàng chân DIP, nên chúng ta cùng khảo sát Vi điều khiển với 40 chân dạng DIP.



Hình 1.2.1: Sơ đồ chân IC 8951

- Chân VCC: Chân số 40 là VCC cấp điện áp nguồn cho vi điều khiển. Nguồn điện cấp là +5V±0.5.

- Chân GND:Chân số 20 nối GND(hay nối Mass). Khi thiết kế cần sử dụng một mạch ổn áp để bảo vệ cho vi điều khiển, cách đơn giản là sử dụng IC ổn áp 7805.

- Port 0 (P0): Port 0 gồm 8 chân (từ chân 32 đến 39) có hai chức năng:

+ Chức năng xuất/nhập :các chân này được dùng để nhận tín hiệu từ bên ngoài vào để xử lí, hoặc dùng để xuất tín hiệu ra bên ngoài, chẳng hạn xuất tín hiệu để điều khiển led đơn sáng tắt.

+ Chức năng là bus dữ liệu và bus địa chỉ (AD7-AD0): 8 chân này (hoặc Port 0) còn làm nhiệm vụ lấy dữ liệu từ ROM hoặc RAM ngoại (nếu có kết nối với bộ nhớ ngoài), đồng thời Port 0 còn được dùng để định địa chỉ của bộ nhớ ngoài.

- Port 1 (P1): Port P1 gồm 8 chân (từ chân 1 đến chân 8), chỉ có chức năng làm các đường xuất/nhập, không có chức năng khác.

- Port 2 (P2): Port 2 gồm 8 chân (từ chân 21 đến chân 28) có hai chức năng:

+ Chức năng xuất/nhập

+ Chức năng là bus địa chỉ cao (A8-A15): khi kết nối với bộ nhớ ngoài có dung lượng lớn,cần 2 byte để định địa chỉ của bộ nhớ, byte thấp do P0 đảm nhận, byte cao do P2 này đảm nhận.

- Port 3 (P3): Port 3 gồm 8 chân (từ chân 10 đến 17):

Chức năng xuất/nhập. Với mỗi chân có một chức năng riêng thứ hai như trong bảng sau

BIT	TÊN	CHỨC NĂNG
P3.0	RxD	Ngõ vào nhận dữ liệu nối tiếp
P3.1	TxD	Ngõ xuất dữ liệu nối tiếp
P3.2	INT0	Ngõ vào ngắt cứng thứ 0
P3.3	INT1	Ngõ vào ngắt cứng thứ 1

P3.4	T0	Ngõ vào của Timer/Counter thứ 0
P3.5	T1	Ngõ vào của Timer/Counter thứ 1
P3.6	WR	Ngõ điều khiển ghi dữ liệu lên bộ nhớ ngoài
P3.7	RD	Ngõ điều khiển đọc dữ liệu từ bộ nhớ bên ngoài
P1.0	T2	Ngõ vào của Timer/Counter thứ 2
P1.1	T2X	Ngõ Nạp lại/thu nhận của Timer/Counter thứ 2

- Chân RESET (RST): Ngõ vào RST ở chân 9 là ngõ vào Reset dùng để thiết lập trạng thái ban đầu cho vi điều khiển. Hệ thống sẽ được thiết lập lại các giá trị ban đầu nếu ngõ này ở mức 1 tối thiểu 2 chu kì máy.

- Chân XTAL1 và XTAL2: Hai chân này có vị trí chân là 18 và 19 được sử dụng để nhận nguồn xung clock từ bên ngoài để hoạt động, thường được ghép nối với thạch anh và các tụ để tạo nguồn xung clock ổn định.

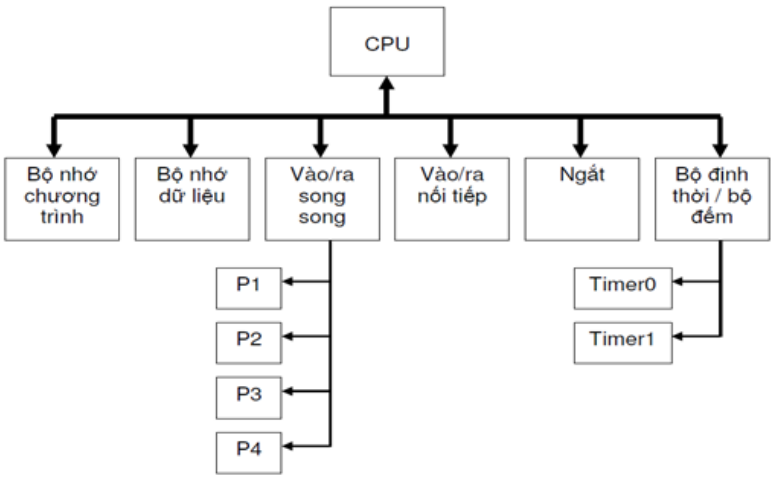
- Chân cho phép bộ nhớ chương trình PSEN: PSEN (program store enable) tín hiệu được xuất ra ở chân 29 dùng để truy xuất bộ nhớ chương trình ngoài. Chân này thường được nối với chân OE (output enable) của ROM ngoài.

Khi thực thi một chương trình ở ROM nội, chân này được duy trì ở mức logic không tích cực (logic 1) (Không cần kết nối chân này khi không sử dụng đến)

- Chân ALE (chân cho phép chốt địa chỉ-chân 30): Khi Vi điều khiển truy xuất bộ nhớ từ bên ngoài, port 0 vừa có chức năng là bus địa chỉ, vừa có chức năng là bus dữ liệu do đó phải tách các đường dữ liệu và địa chỉ. Tín hiệu ở chân ALE dùng làm tín hiệu điều khiển để giải đa hợp các đường địa chỉ và các đường dữ liệu khi kết nối chúng với IC chốt. Các xung tín hiệu ALE có tốc độ bằng 1/6 lần tần số dao động đưa vào Vi điều khiển, như vậy có thể dùng tín hiệu ở ngõ ra ALE làm xung clock cung cấp cho các phần khác của hệ thống. Khi không sử dụng có thể bỏ trống chân này

- Chân EA: Chân EA dùng để xác định chương trình thực hiện được lấy từ ROM nội hay ROM ngoại. Khi EA nối với logic 1(+5V) thì Vi điều khiển thực hiện chương trình lấy từ bộ nhớ nội. Khi EA nối với logic 0(0V) thì vi điều khiển thực hiện chương trình lấy từ bộ nhớ ngoại.

2.2. Mô tả nguyên lý hoạt động.



Hình 1.2.2: Cấu trúc sơ đồ khối tổng quát

*** 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 216 = 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

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 lien 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.

3. Các hệ thống đếm-các loại mã.

3.1. Các hệ thống đếm.

3.1.1. Giới thiệu.

Nhu cầu về định lượng nhất là trong những trao đổi thương mại, đã có từ khi xã hội hình thành. Đã có nhiều cố gắng trong việc tìm kiếm các vật dụng, các ký hiệu … dùng cho việc định lượng này như các que gỗ, vỏ sò, số La mã…

Hệ thập phân - Decimal

Hệ nhị phân - Binary

Hệ16 - Hexadecimal

Mã BCD (standard BCD, gray code): (Binary Coded Decimal)

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

Mã BCD gồm có 2 loại:

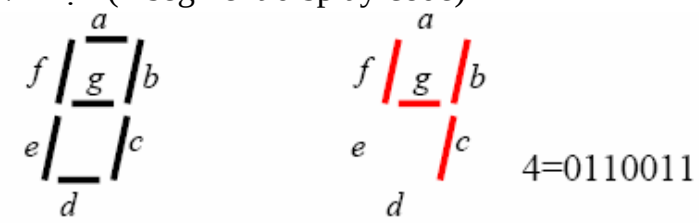
- Mã BCD không nén (unpacked): biểu diễn một số BCD bằng 8 bit nhị phân
- Mã BCD nén (packed): biểu diễn một số BCD bằng 4 bit nhị phân

VD: Số thập phân 5 2 9

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

Số BCD nén 0101b 0010b 1001b

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



Hình 1.3.1: LED 7 thanh và cách mã hóa

* Các mã hệ đếm thông dụng

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

Bảng 1.1: Giá trị tương ứng giữa các hệ số

3.1.2. Nguyên lý của việc viết số.

Một số được viết bằng cách đặt kề nhau các ký tự được chọn trong một tập hợp. Mỗi ký hiệu trong mỗi số được gọi là một số mã (số hạng – digit).

Ví dụ, trong hệ thống thập phân, tập hợp này gồm 10 ký hiệu rất quen thuộc, đó là các con số từ 0 đến 9.

S_{10} = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Khi một số gồm nhiều số mã được viết, giá trị của số mã tuỳ thuộc vị trí của nó trong số đó. Giá trị này được gọi là trọng số của số mã. Ví dụ, số 1998 trong hệ thập phân, số 9 đầu sau số 1 có trọng số là 900 trong khi số 9 thứ hai chỉ là 90.

Tổng quát, một hệ thống số được gọi là hệ b sẽ gồm b ký hiệu trong đó tập hợp:

$S_b = \{S_0, S_1, S_2, \ldots S_{b-1}\}$

Một số n trong hệ b được viết dưới dạng:

$N = (a_n a_{n-1} a_{n-2} \ldots a_i \ldots a_1 a_0, a_{-1} a_{-2} \ldots a_{-m})$ với $a_i \in S$.

Sẽ có giá trị:

$$N = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_i b^i + \ldots + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \ldots + a_{-m} b^{-m} = \sum_{i=-m}^{n} a_i b^i$$

3.1.3. Hệ thống số

Hệ thập phân – Decimal system – Cơ số 10

Hệ thập phân dùng 10 chữ số: 0 1 2 3 4 5 6 7 8 9 để biểu diễn các số.

Ví dụ: Tính giá trị của **1 234 567** trong hệ thập phân.

Biểu diễn theo công thức tổng quát:

$1\ 234\ 567 = 1*10^6 + 2*10^5 + 3*10^4 + 4*10^3 + 5*10^2 + 6*10^1 + 7*10^0$

1 234 567 = 1 000 000 + 200 000 + 30 000 + 4 000 + 500 + 60 + 7

Hệ nhị phân – Binary system – Cơ số 2

Hệ nhị phân dùng 2 chữ số : **0 1** để biểu diễn các số.

Ví dụ: Tính giá trị của số **100 111** trong hệ nhị phân.

Biểu diễn theo công thức tổng quát:

$100\ 111_{Bin} = 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0$

$100\ 111_{Bin} = 100\ 000_{Bin} + 00\ 000_{Bin} + 0\ 000_{Bin} + 100_{Bin} + 10_{Bin} + 1$

Nếu đổi sang cơ số 10 ta được:

$100\ 111_{Bin} \Leftrightarrow 32_{Dec} + 0_{Dec} + 0_{Dec} + 4_{Dec} + 2_{Dec} + 1_{Dec}$

$100\ 111_{Bin} \Leftrightarrow 39_{Dec}$

Hệ bát phân – Octal system – Cơ số 8

Hệ bát phân dùng 8 chữ số: **0 1 2 3 4 5 6 7** để biểu diễn các số.

Ví dụ: Tính giá trị của số **123 456** trong hệ bát phân.

Biểu diễn theo công thức tổng quát:

$123\ 456_{Oct} = 1*8^5 + 2*8^4 + 3*8^3 + 4*8^2 + 5*8^1 + 6*8^0$

$123\ 456_{Oct} = 100\ 000_{Oct} + 20\ 000_{Oct} + 3\ 000_{Oct} + 400_{Oct} + 50_{Oct} + 6_{Oct}$

Nếu đổi sang cơ số 10 ta được:

$123\ 456_{Oct} \Leftrightarrow 32768_{Dec} + 8192_{Dec} + 1536_{Dec} + 256_{Dec} + 40_{Dec} + 6_{Dec}$

$123\ 456_{Oct} \Leftrightarrow 42\ 798_{Dec}$

Hệ thập lục phân – Hexadecimal system – Cơ số 16

Hệ thập lục phân dùng 16 chữ số: **0 1 2 3 4 5 6 7 8 9 A B C D E F** để biểu diễn các số.

Ví dụ: Tính giá trị của số 4B trong hệ thập lục phân.

Biểu diễn theo công thức tổng quát:

$4B_{Hex} = 4*16^1 + B*16^0$

$4B_{Hex} = 40_{Hex} + B_{Hex}$

Nếu theo cơ số 10 ta có:

$4B_{Hex} \Leftrightarrow 64_{Dec} + 11_{Dec}$

$4B_{Hex} \Leftrightarrow 75_{Dec}$

3.2. Mã ASCII.

3.2.1. Biến đổi qua lại giữa các hệ thống số

*** Đổi một cơ số từ hệ b sang hệ 10**

Để đổi một cơ số từ hệ b sang hệ 10 ta khai triển trực tiếp đa thức của b.

Một số N trong hệ b được viết:

$N_b = a_n a_{n-1} ... a_i ... a_0 a_{-1} a_{-2} ... a_{-m}$ với $a_i \in S_b$

Có giá trị tương ứng với hệ cơ số 10 là:

$$N_{10} = a_n b^n + a_{n-1} b^{n-1} + ... + a_i b^i + ... + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + ... + a_{-m} b^{-m} = \sum_{i=-m}^{n} a_i b^i$$

Ví dụ 1: Đổi số 1010,11 ở cơ số 2 sang cơ số 10 ta làm như sau:

$1011,11_2 \Leftrightarrow 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 + 1.2^{-1} + 1.2^{-2}$

$1011,11_2 \Leftrightarrow 8 + 0 + 4 + 1 + 0,5 + 0,25$

$1011,11_2 \Leftrightarrow 13,75_{10}$

Ví dụ 2: Đổi giá trị của số 4B,8F trong hệ thập lục phân sang hệ thập phân.

$4B,8F_{16} \Leftrightarrow 4*16^1 + B*16^0 + 8*16^{-1} + 15*16^{-2}$

$4B,8F_{16} \Leftrightarrow 64 + 11 + 0,5 + 0.05859375$

$4B,8F_{16} \Leftrightarrow 75,55859375_{10}$

*** Đổi một cơ số từ hệ 10 sang hệ b**

Đây là bài toán tìm một dãy các ký hiệu cho số N viết trong hệ b. Một số N viết trong dạng cơ số 10 và viết trong cơ số b có dạng như sau:

$N = (a_n a_{n-1} \ldots a_0, a_{-1} a_{-2} \ldots a_{-m})_b = (a_n a_{n-1} \ldots a_0)_b + (0, a_{-1} a_{-2} \ldots a_{-m})_b$

Trong đó:

$(a_n a_{n-1} \ldots a_0)_b$ = PE(N) là phần nguyên của N.

$(0, a_{-1} a_{-2} \ldots a_{-m})_b$ = PF(N) là phần thập phân của N.

Có 2 cách biến đổi khác nhau cho phần nguyên và phần thập phân.

+ Phần nguyên – PE(N)

Phần nguyên có thể viết lại như sau:

$$PE(N) = (a_n b^{n-1} + a_{n-1} b^{n-2} + \ldots + a_1) b + a_0$$

Ta thấy rằng, nếu lấy PE(N) chia cho b thì ta sẽ có số dư là a_0, được thương là $PE'(N) = (a_n b^{n-1} + a_{n-1} b^{n-2} + \ldots + a_1) b$. Vậy số dư của lần thứ nhất này chính là bit có trọng số nhỏ nhất (bit LSB).

Tiếp tục cho đến khi được phép chia cuối cùng, đó chính là bit lớn nhất (MSB).

+ Phần thập phân – PF(N)

Phần thập phân có thể được viết lại như sau:

$$PF(N) = b^{-1}(a_{-1} + a_{-2} b^{-1} + \ldots + a_{-m} b^{-m+1})$$

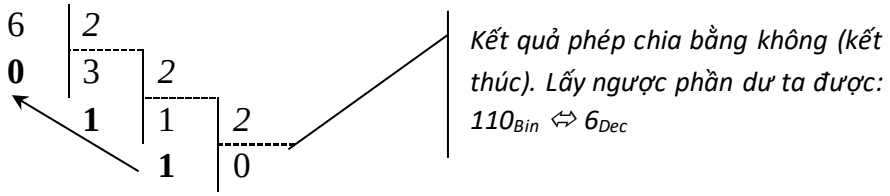
Ta thấy rằng nếu nhân PF(N) với b ta được $a_{-1} + a_{-2} b^{-1} + \ldots + a_{-m} b^{-m+1} = a_{-1} + PF'(N)$. Vậy a_{-1} chính là bit lẽ đầu tiên của phần thập phân.

Tiếp tục lặp lại bài toán nhân phần lẽ của kết quả có được của phép nhân trước đó với b cho tới khi kết quả phần lẽ bằng 0, ta tìm được dãy số $(a_{-1} a_{-2} a_{-3} \ldots a_{-m})$.

Chú ý: Phần thập phân của số N khi đổi sang hệ b có thể gồm vô số số hạng (do kết quả phần thập phân có được luôn khác 0), vậy tuỳ theo yêu cầu về độ chính xác của kết quả mà ta lấy một số số hạng nhất định.

Ví dụ: Đổi số 6,3 sang hệ nhị phân.

Phần nguyên ta thực hiện như sau:



Phần thập phân ta thực hiện như sau:

0,3*2 = 0,6	➔ $a_{-1} = 0$	Lấy phần chẵn là 0
0,6*2 = 1,2	➔ $a_{-2} = 1$	Lấy phần chẵn là 1
0,2*2 = 0,4	➔ $a_{-3} = 0$	
0,4*3 = 0,8	➔ $a_{-4} = 0$	
0,8*2 = 1,6	➔ $a_{-5} = 1$	
0,6*2 = 1,2	➔ $a_{-6} = 1$	
0,2*2 = 0,4	➔ $a_{-7} = 0$	(tiếp tục…)

Như vậy kết quả bài toán nhân luôn luôn khác 0, nếu kết quả bài toán chỉ cần 5 số lẽ thì ta lấy PF(N) = 0,01001.

Kết quả cuối cùng là: $6,3_{10} \Leftrightarrow 110,01111_2$

*** Đổi một cơ số từ hệ b sang hệ b^k**

Từ cách triển khai đa thức của số N trong hệ b, ta có thể nhóm thành từng k số hạng từ dấu phẩy về 2 phía và đặt thành thừa số chung.

$N = a_n b^n + \ldots + a_4 b^4 + a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + a_{-3} b^{-3} + \ldots + a_{-m} b^{-m}$

Giả sử k =3 số N được viết lại như sau:

$N = \ldots + (a_5 b^2 + a_4 b^1 + a_3 b^0) b^3 + (a_2 b^2 + a_1 b^1 + a_0 b^0) b^0 + (a_{-1} b^2 + a_{-2} b^1 + a_{-3} b^0) b^{-3} + \ldots$

Phần chứa trong mỗi dấu ngoặc luôn nhỏ hơn b^k (k=3), vậy số này chính là một số trong hệ b^k và được biểu diễn bởi các ký hiệu tương ứng trong hệ này.

Ví dụ 1: Đổi số 10011101010,10011 từ hệ cơ số 2 sang hệ cơ số 8 (k=3 vì $8 = 2^3$)

Từ dấu phẩy gom từng 3 số, ta có thể thêm số 0 vào bên trái của số hoặc bên phải sau dấu phẩy cho đủ nhóm 3 (k=3) số, ta được như sau:

010 **011** 101 **010**, 100 **110**$_{(2)} \Leftrightarrow 2352,46_{(8)}$

Ví dụ 2: Đổi số 10011101010,10011 từ hệ cơ số 2 sang hệ cơ số 16 (k=4 vì $16 = 2^4$)

Từ dấu phẩy gom từng 4 số, ta có thể thêm số 0 vào bên trái của số hoặc bên phải sau dấu phẩy cho đủ nhóm 4 (k=4) số, ta được như sau:

0100 1110 **1010**, 1001 **1000**$_{(2)} \Leftrightarrow 4EA,98_{(16)}$

Ngoài ra, ta cũng có thể biến đổi một số từ b^k sang b^p thực hiện trung gian qua hệ b. Điều này dễ dàng suy ra từ 2 ví dụ trên, đọc giả tự nghiên cứu.

Dưới đây là bảng kê các số đầu tiên trong 4 hệ số thường gặp:

Thập phân	Nhị phân	Bát phân	Thập lục phân
0	00000	0	0
1	00001	1	1
2	00010	2	2
3	00011	3	3
4	00100	4	4
5	00101	5	5
6	00110	6	6
7	00111	7	7
8	01000	10	8
9	01001	11	9
10	01010	12	A

Thập phân	Nhị phân	Bát phân	Thập lục phân
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15

3.2.3. Các phép tính trong hệ nhị phân

Các phép tính trong hệ nhị phân được thực hiện tương tự như hệ thập phân, tuy nhiên cũng có một số điểm cần lưu ý.

*** Phép cộng**

Là phép tính làm cơ sở cho các phép tính khác. Ta có các chú ý sau:

0 + 0 = 0

0 + 1 = 1 + 0 = 1

1 + 1 = 0, nhớ 1 (đem qua bit cao hơn).

Ngoài ra để thực hiện bài toán cộng nhiều số ta nên nhớ:

- Nếu số bit số 1 chẵn thì kết quả bằng 0.
- Nếu số bit số 1 lẽ thì kết quả bằng 1.
- Cứ 1 cặp số 1, cho 1 số nhớ.

Ví dụ: Tính 011 + 101 + 011 + 011

```
          11    ← số nhớ
         111    ← số nhớ
        011
        101
+       011
        011
       ----
       1110
```

*** Phép trừ**

Ta có các chú ý sau:

0 – 0 = 0

1 – 1 = 0

1 – 0 = 1

0 – 1 = 1, nhớ 1 cho bit cao hơn.

Ví dụ: Tính 1011 – 0101

```
          1    ← số nhớ
       1011
–      0101
       ----
       0110
```

*** Phép nhân**

Ta có các chú ý sau:

0 × 0 = 0

0 × 1 = 0

1 × 1 = 1

Ví dụ: Tính 110 × 101

```
        110
×       101
       ----
        110
+      000
       110
      -----
      11110
```

***Phép chia**

Tương tự như phép chia trong hệ cơ số 10.

Ví dụ: Tính 1001100100 : 11000

```
1001100100   | 11000
  –11000     | 11001,1
  0011100
  –11000
  00100100
–11000
001100 0        ← thêm 0 vào để
–1100 0         chia lấy phần lẽ.
0000 0
```

3.2.3. Mã ASCII.

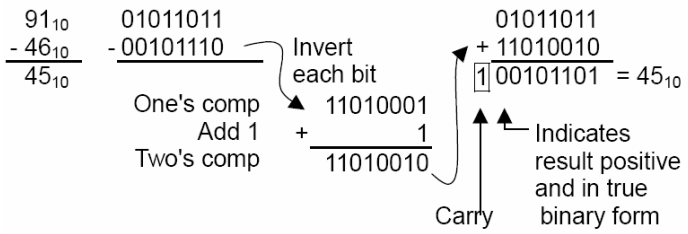
HEX	DEC	CHR	Ctrl	HEX	DEC	CHR	HEX	DEC	CHR	HEX	DEC	CHR
0	0	NUL	^@	20	32	SP	40	64	@	60	96	`
1	1	SOH	^A	21	33	!	41	65	A	61	97	a
2	2	STX	^B	22	34	"	42	66	B	62	98	b
3	3	ETX	^C	23	35	#	43	67	C	63	99	c
4	4	EOT	^D	24	36	$	44	68	D	64	100	d
5	5	ENQ	^E	25	37	%	45	69	E	65	101	e
6	6	ACK	^F	26	38	&	46	70	F	66	102	f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

Bảng 1.2: Bảng mã ASC II

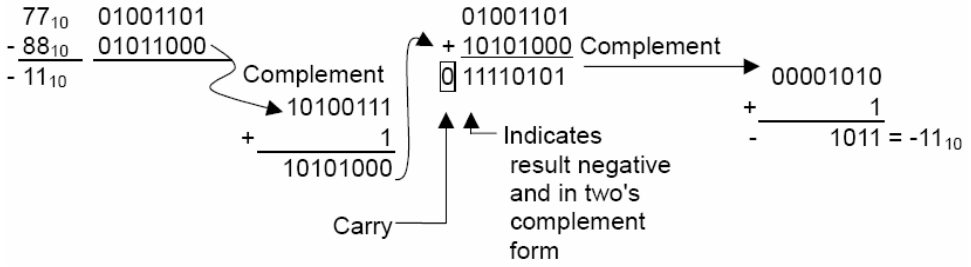
ASCII (American Standard Code for Information Interchange - Chuẩn mã trao đổi thông tin Hoa Kì) là bộ kí tự và bộ mã kí tự dựa trên bảng chữ cái La Tinh được dùng trong tiếng Anh hiện đại và các ngôn ngữ Tây Âu khác. Nó thường được dùng để hiển thị văn bản trong máy tính và các thiết bị thông tin khác. Nó cũng được dùng bởi các thiết bị điều khiển làm việc với văn bản.

Bảng mã ASCII chuẩn có 128 kí tự, gồm các kí tự điều khiển (0 - 31 và 127), các ký tự in được như bảng chữ cái, các dấu (32 - 126). Bảng mã ASCII mở rộng có 255 kí tự gồm 128 ký tự của bảng mã ASCII chuẩn và các chữ có dấu, ký tự trang trí, v.v...

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



Trường hợp kết quả âm



CÂU HỎI ÔN TẬP

1. Đặc tính chung của vi xử lý là gì? So sánh giữa vi xử lý và vi điều khiển?
2. Nêu các ứng dụng của vi xử lý và vi điều khiển?
3. Trình bày chức năng các chân của vi điều khiển 8951?
4. Có bao hệ đếm hiện nay trong vi điều khiển?
5. Mô tả nguyên lý hoạt động của IC 8951?

BÀI TẬP

Câu 1: 1 đĩa mềm có dung lượng 1,44MB lưu trữ được 400 trang văn bản.

Vậy nếu dùng một ổ đĩa cứng có dung lượng 12GB thì lưu giữ được bao nhiêu trang văn bản?

Câu 2: Chuyển xâu ký tự sau thành mã nhị phân: TIN HOC

Câu 3: Dãy bit 01100010 01111001 01110100 01100101 tương ứng là mã ASCII của dãy ký tự nào.

Câu4: Viết các số thực sau dưới dạng dấu phẩy động: 11005 ; 25,879 ; 0,000984

Câu 5: Đổi các số sau sang hệ nhị phân và hệ cơ số16: 7; 15; 22; 127; 97; 123.75

Câu 6: Đổi các số sau sang hệ cơ số 10: 5D1616 ; 7D71616; 11111122; 1011010122

Câu 7:

a. Đổi từ hệ hexa sang hệ nhị phân 5E; 2A; 4B; 6C

b. Đổi từ hệ nhị phân sang hệ hexa 1101011; 10001001; 1101001; 10110

BÀI 2: TẬP LỆNH 89C51

Giới thiệu:

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

Mục tiêu của bài:

- Giải thích tác dụng các lệnh điều khiển của họ vi điều khiển 89C51.
- Viết các chương trình theo yêu cầu.
- Phân tích các chương trình có sẵn.
- Thực hiện các ứng dụng thực tế thông qua các mô hình.
- Rèn luyện tính chính xác, nghiêm túc trong học tập và trong thực hiện công việc.

Nội dung chính:

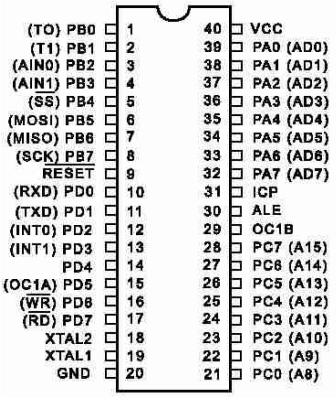
1. Giới thiệu.

IC 89C51 có khả năng điều khiển được các bóng Led rời chạy theo hiệu ứng như mạch led hào quang hay rất nhiều loại bóng Led khác IC 89C51 được sử dụng rất nhiều trong các biển quảng cáo cho một thương hiệu, một cái tên, một dịch vụ hay bảng thông

1.1. Đặc tính của IC 89C51.

- Có khả năng 1000 chu kì ghi xóa.
- Tần số hoạt động 0Hz-24MHz.
- 3 mức khóa bộ nhớ lập trình.
- 128 bytes Ram nội 4 port xuất nhập I/O 8 bit.
- 2 bộ time/counter 16 bit.
- Giao tiến nối tiến bằng phần cứng, 64 Kb vùng nhớ mã ngoài, 64 Kb vùng nhớ dữ liệu.
- 4 chu kì máy cho hoạt động nhân hoặc chia.
- Có các chế độ nghỉ và chế độ nguồn giảm.

1.2. Cấu trúc bên trong IC 89C51



Hình 2.1.1: Sơ đồ chân IC 89C51

Thành phần chính của vi điều khiển 89C51 chính là CPU bên trong CPU có :

+ Thanh ghi tích lũy A.

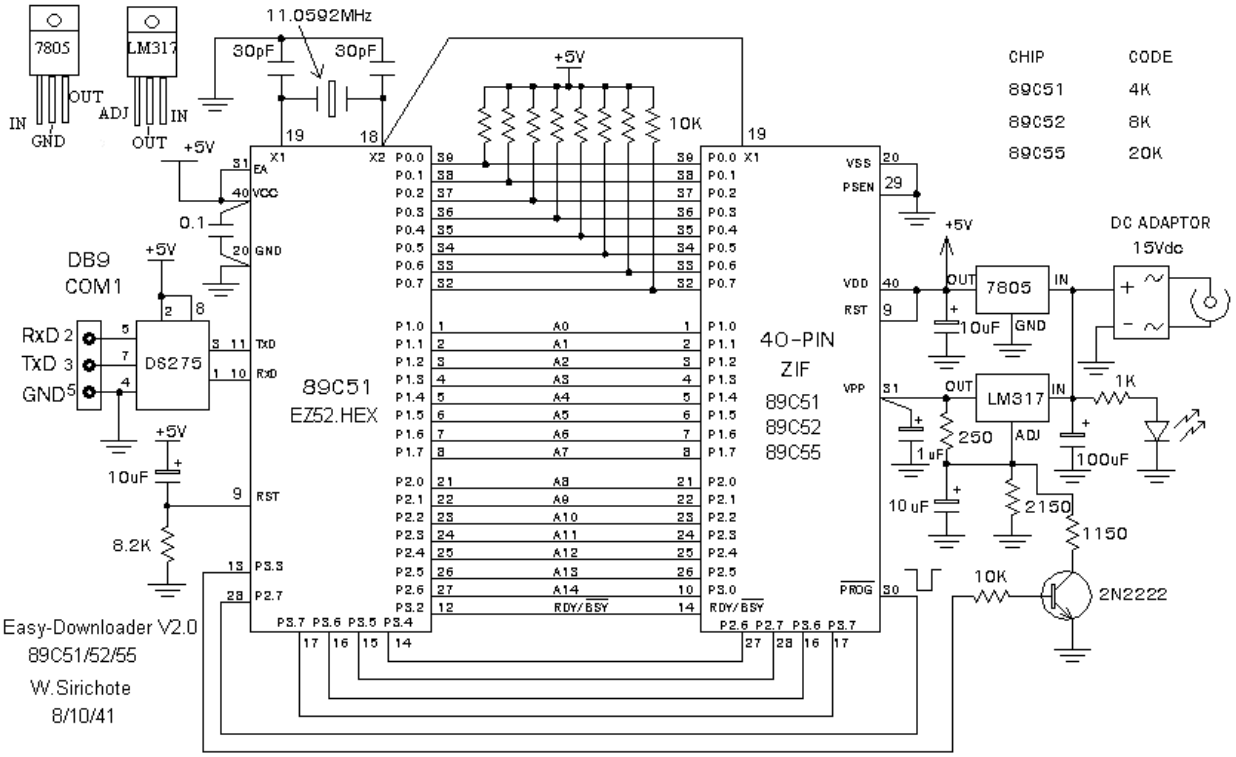
+ Thanh ghi tíc lũy B dùng cho phép nhân chia.

+ Đơn vị logic học ALU.

+ 4 bank thanh ghi.

+ Bộ nhớ chương trình, bộ giải mã lệnh, bộ điều khiển thời gian và logic.

+ Con trỏ ngăn xếp.



Hình 2.1.2: Sơ đồ mạch 89C51

1.3. Ứng dụng của IC 89C51.

- Dùng để điều khiển các bóng Led của biển quảng cáo.
- Ưu tiên sử dụng làm biển quảng cái ngoài trời, ở nơi khó tháo lắp.
- Dùng để điều khiển bảng hiển thị thông tin nơi công cộng.

Như vậy qua bài viết chắc các bạn cũng đã biết về cấu tạo cũng như ứng dụng của vi điều khiển IC 89C51, nói chung đây là vi điều khiển dùng để điều khiển các biển quảng cáo Led các chức năng cũng tương tự như một số mạch điều khiển Led trên thị trường như: mạch điều khiển Led vẫy Oeled v3, mạch điều khiển led 7 màu RGB...

2. Tập lệnh.

2.1. Nhóm truyền dữ liệu.

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

Cú pháp: Mov A, Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Chuyển dữ liệu của thanh ghi Rn vào thanh ghi A, dữ liệu trên thanh ghi Rn không đổi

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

Lệnh Mov A, R5

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

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

Cú pháp: Mov A, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: chuyển dữ liệu của ô nhớ có địa chỉ bằng direct vào thanh ghi A.

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

Lệnh Mov A,33H

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

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

Cú pháp: Mov A,@Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: chuyển dữ liệu của ô nhớ 'có địa chỉ bằng giá trị của thanh ghi Ri' vào thanh ghi A.

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

Lệnh Mov A,@R1

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

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

2.1.4. Lệnh đưa dữ liệu vào thanh ghi A

Cú pháp: Mov A, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thiết lập dữ liệu cho thanh ghi A

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

Mov A, #56H

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

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

Cú pháp: Mov Rn, A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: chuyển dữ liệu từ thanh ghi A vào thanh ghi Rn (n=0-7)

Ví dụ:

Mov A, #56H

Mov R1, A

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

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

Cú pháp: Mov Rn, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: chuyển dữ liệu của ô nhớ có địa chỉ direct vào thanh ghi Rn (n=0-7)

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

Mov R4,55H

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

2.1.7. Thiết đặt dữ liệu cho thanh ghi Rn

Cú pháp: Mov Rn, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thiết đặt dữ liệu cho thanh ghi Rn

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

Mov A, #37H

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

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

Cú pháp: Mov direct, A

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: chuyển dữ liệu từ thanh ghi A vào một ô nhớ có địa chỉ direct.

Ví dụ:

Mov A, #77H

Mov 69H, A

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

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

Cú pháp: Mov direct, Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: chuyển dữ liệu từ thanh ghi A vào một ô nhớ có địa chỉ direct

Ví dụ:

Mov Rn, #78H

Mov 7AH, Rn

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

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

Cú pháp: Mov direct, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: chuyển dữ liệu từ ô nhớ có địa chỉ direct này vào một ô nhớ có địa chỉ direct khác

Ví dụ: giả sử thanh ghi 20H mang dữ liệu có giá trị là FFH

Mov 22H, 20H

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

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

Cú pháp: Mov direct, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: thiết lập dữ liệu cho ô nhớ có địa chỉ direct

Ví dụ:

Mov 52H, #43H

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

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

Cú pháp: Mov direct, @Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Chuyển dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri vào ô nhớ có địa chỉ direct

Ví dụ:

Mov 30H, #46H

Mov R0, #30H

Mov 23H, @R0

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

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

Cú pháp: Mov @Ri, A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Chuyển dữ liệu của thanh ghi A vào ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri

Ví dụ:

Mov A, #33H

Mov R1, #22H

Mov @R0, A

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

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

Cú pháp: Mov @Ri, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Chuyển dữ liệu của ô nhớ có địa chỉ direct vào ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri

Ví dụ:

Mov 4BH, #2AH

Mov R0, #2AH

Mov @R0, 4BH

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

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

Cú pháp: Mov @Ri, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

Mov R0, #3BH

Mov @R0, #27H

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

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

Cú pháp: Mov DPTR,#data16

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

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

Ví dụ: Mov DPTR,#3A5FH

Sau khi lệnh được thực hiện DPTR mang giá trị là 3A5FH DPL mang giá trị 5FH và DPL mang giá trị 3AH

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

Cú pháp: XCH A, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Trao đổi dữ liệu của thanh ghi A với ô nhớ có địa chỉ direct, tức là sau khi thực hiện lệnh ô nhớ có địa chỉ direct mang dữ liệu của thanh ghi A trước đó và thanh ghi A mang dữ liệu của ô nhớ có địa chỉ direct.

Ví dụ: Mov A, #0FAH

Mov 50H, #60H

XCH A, 50H

Kết quả: A mang giá trị là 60H 50H mang giá trị là 0FAH

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

Cú pháp: XCH A, Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Trao đổi dữ liệu của thanh ghi A với thanh ghi Rn.

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

Cú pháp: XCH A, @Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Trao đổi dữ liệu của thanh ghi A với ô nhớ có địa chỉ bằng giá trị lưu giữ trong thanh ghi Ri

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

Cú pháp: XCHD A, @Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Trao đổi dữ liệu của 4 bit thấp ở thanh ghi A với dữ liệu của 4 bit thấp ở ô nhớ có địa chỉ bằng giá trị lưu giữ trong thanh ghi Ri

2.1.21. Lệnh truy xuất dữ liệu từ ROM nội

Cú pháp: MovC A,@A+DPTR

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Chuyển dữ liệu từ bộ nhớ ROM có địa chỉ bằng giá trị của A cộng với DPTR vào thanh ghi A

Các lệnh còn lại trong nhóm lệnh di chuyển

MovC A, @A+PC

MovC A, @i

MovX A, @DPTR

MovX A, @Ri

MovX @DPTR, A

PUSH direct

POP direct

*** GIẢI THUẬT VÀ LƯU ĐỒ**

Giải thuật là một trình tự thực hiện công việc nào đó.

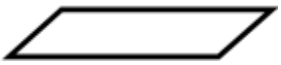
Lưu đồ là sự biểu diễn đồ hoạ của giải thuật.

Lưu đồ chứa các ký hiệu biểu diễn các bước của giải thuật.

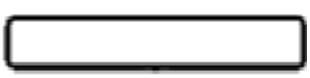
Mỗi ký hiệu biểu diễn một hoạt động.

Các ký hiệu được sử dụng trong lưu đồ:

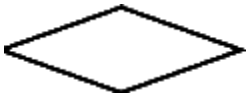
- Nhập - Input: tín hiệu vi điều khiển lấy vào để xử lí



- Xử lý – Process: quá trình xử lí tín hiệu

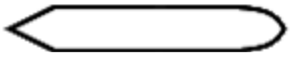


- Quyết định – Decision: chọn lựa hướng đi phù hợp



- Bắt đầu và Kết thúc – Start and Stop:

- Hiển thị - Display/Output: tín hiệu do vi điều khiển xuất ra để điều khiển thiết bị hiển thị



- Gọi chương trình con: gọi chương trình con

Khi chương trình con được gọi, chương trình chính dừng lại chờ cho chương trình con thực hiện xong thì chương trình chính mới tiếp tục thực hiện.

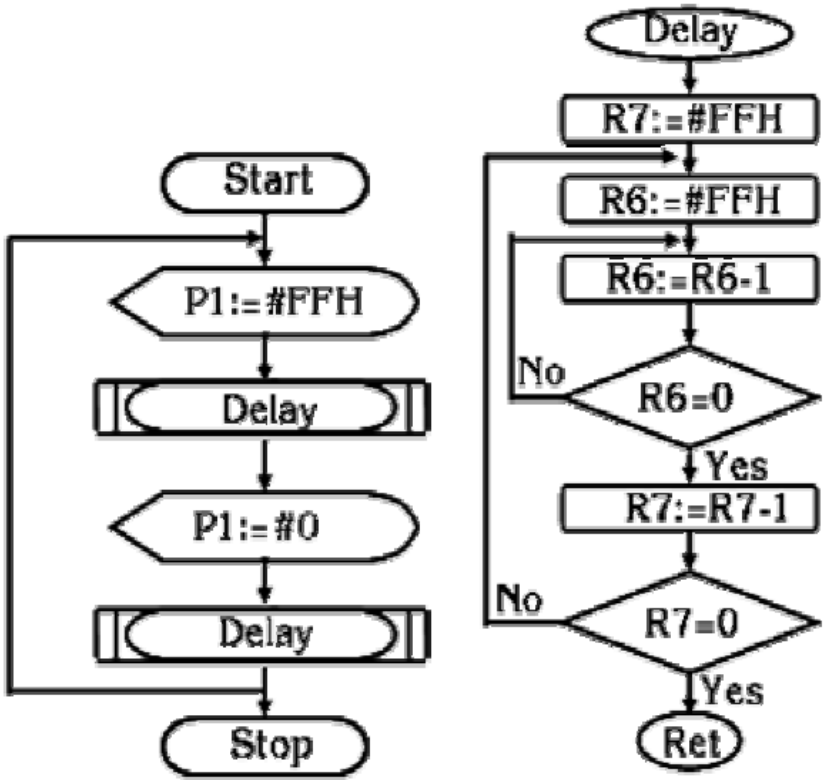
- Bắt đầu và Kết thúc chương trình con:

*** BÀI TẬP MẪU**

Bài tập 1:

Viết chương trình để các led nối với Port 1 sáng rồi tắt led. Biết led sáng khi tín hiệu xuất ra ở mức 1. Minh họa trong hình phía dưới

Phân tích: để led sáng rồi tắt, cần làm cho tín hiệu xuất ra mức 1 một khoảng thời gian để mắt có thể nhận biết được, sau đó làm tín hiệu xuất ra ở mức 0 một khoảng thời gian như trên. Cứ lặp đi lặp lại đoạn trên sẽ thấy dãy đèn sáng rồi tắt.



Khi bắt đầu chương trình, P1 được truyền giá trị là #FFH để làm các ngõ ra của P1 ở mức 1. Giả sử ta bỏ qua lệnh gọi chương trình con Delay, công việc cần thực hiện kế tiếp là làm tín hiệu ở P1 trở về mức 0 để làm led tắt, vì vậy P1 được truyền giá trị là #0H. Các lệnh của vi điều khiển chỉ thực hiện trong một vài chu kì máy, khoảng vài µs, do đó nếu bỏ qua lệnh gọi chương trình con Delay thì led sáng sau khoảng vài µs, rồi lại tắt khoảng vài µs, quá trình sáng tắt của led quá nhanh làm mắt người chỉ thấy led sang liên tục.

Muốn mắt người nhận ra led sáng lên rồi tắt đi cần làm tín hiệu xuất ra P1 lâu hơn. Chương trình con Delay thực hiện nhiệm vụ duy trì trạng thái ở ngõ ra lâu hơn bằng cách cho vi điều khiển thực hiện hàng ngàn lần các câu lệnh nào đó không ảnh hưởng đến trạng thái ngõ ra. Khi chương trình chính gặp lệnh gọi chương trình con Delay, chương trình chính sẽ dừng lại chờ cho chương trình con Delay thực hiện xong rồi mới thực hiện câu lệnh kế tiếp.

Chương trình:

```
;***************/////--- 8 LED SANG SANG TAT ---\\\\\\
;**////////////////___ lam 8 led sang roi tat
;**////////////////___ sang o muc 1 va tat o muc 0
;**/////////////_____ lap di lap lai khong gioi han
        ORG         000H ; khai bao dia chi de bat dau chuong trinh Tudau:
        Mov         P1, #11111111B ;===>>>> lam 8 led noi P1 sang
        LCall       Delay ;===>>>> goi chuong trinh con Delay
        Mov         P1, #00 ;===>>>> lam 8 led noi P1 tat
        LCall       Delay
        Sjmp        Tudau ;===>>>> nhay ve thuc hien tu dau
;**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  delay
        Delay:
        Mov         R7, #0FFH ;===>>>> ga'n R7=#0FFH, 1 chu ki may
        Kt2:  Mov       R6, #0FFH ;===>>>> ga'n R6=#0FFH, 1 chu ki may
        Kt1:  Djnz      R6, Kt1 ;===>>>> giam R6 xuong 1, R6 khac 0-
        Djnz        R7, Kt2 ;===>>>> giam R7 xuong 1, R6 khac 0-> Ret ;===>>>>
ket thuc chuong trinh con Delay
        End ;===>>>> ket thuc chuong trinh
```

Chương trình thực hiện, gặp câu lệnh Sjmp Tudau chương trình sẽ nhảy về nhãn Tudau ở đầu chương trình, và chương trình tiếp tục lại từ đầu, cứ thế chương trình lặp lại mãi mãi.

LCall Delay lệnh gọi chương trình con Delay. Chương trình con trong chương trình này được bắt đầu bằng nhãn Delay:, khi có lệnh gọi chương trình con, thì vi điều khiển sẽ chuyển sang thực hiện các câu lệnh của chương trình con, sau khi chương trình con hoàn thành, vi điều khiển trở về chương trình chính để thực hiện tiếp.

Lệnh Kt1: Djnz R6,Kt1

Lệnh Djnz R6,Kt1 này giảm R6 đi một đơn vị, nếu R6 chưa bằng 0, chương trình lại nhảy về nhãn Kt1, mà nhãn Kt1 lại gọi lại lệnh này, do đó toàn câu lệnh thực hiện giảm R6 cho đến khi R6 về 0 thì thực hiện lệnh kế tiếp. Câu lệnh này thực hiện, không ảnh hưởng hay thay đổi giá trị của các ô nhớ khác, vì vậy ngõ ra từ các Port cũng không thay đổi trạng thái.

Tổng thời gian của chương trình con Delay:

Nếu sử dụng thạch anh 12MHz thì mỗi chu kì máy là 1μs

Lệnh Mov R7,#0FFH thực hiện 1 lần và mất 1 chu kì máy

Lệnh Djnz R6,Kt1 thực hiện 255 lần và mỗi lần mất 2 chu kì máy

Đoạn lệnh

Kt2: Mov R6,#0FFH

Kt1: Djnz R6,Kt1 ;kí hiệu Kt thay cho kiểm tra

Djnz R7,Kt2

Đoạn lệnh này được thực hiện 255 lần với mỗi lần gồm (255×2)chu kì máy của lệnh Djnz R6,Kt1 và 2 chu kì máy của lệnh Djnz R7,Kt2 và 1 chu kì máy của lệnh Mov R6,#0FFH

Tổng cộng đoạn trên đã thực hiện (1+(255×2)+2)×255)=130815 chu kì máy

Vậy tổng cộng chương trình con Delay đã thực hiện 130815+1=130816 chu kì máy, tức là đã thực hiện trong 130816μs, khoảng thời gian gần bằng 0.13s

Như vậy trong 1s led sáng khoảng 4 lần đủ để mắt người có thể nhận ra được. Nếu mô phỏng với Pinnacle 52 chương trình delay phải chạy với số lần lặp lại nhiều hơn để thấy được các trạng thái ngõ ra của các Port, thay đoạn chương trình con Delay ở trên bằng đoạn chương trình con như dưới đây khi mô phỏng bằng Pinnacle 52

```
;**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  delay Pinnacle 52
    Delay:
    Mov 70H,#2H ;===>>>> ga'n
    70H=#0FFH, 2 chu ki may
    Kt3: Mov 71H,#0FFH ;===>>>> ga'n
    71H=#0FFH, 2 chu ki may
    Kt2: Mov 72H,#0FFH ;===>>>> ga'n
    72H=#0FFH, 2 chu ki may
    Kt1: Djnz 72H,Kt1 ;===>>>> giam 72H xuong 1, 72H khac 0-> giam tiep, 2ckm
    Djnz 71H,Kt2 ;===>>>> giam 71H xuong 1, khac 0-> ve Kt2
    Djnz 70H,Kt3 ;===>>>> giam 70H xuong 1, khac 0-> ve Kt3
    Ret ;===>>>> ket thuc chuong trinh con
```

Bài tập 2:"Sáng lần lượt"

Phần cứng: 8 led nối với Port 1, được định vị trí như sau: led 1 nối với P1.0, lần lượt cho đến led 8 (nối với P1.7). Biết led sáng khi tín hiệu xuất ở các chân Port 1 ở giá trị 1. Viết chương trình để các led nối với Port 1 sáng lần lượt từng led từ led 1 đến led 8. Các quá trình trên được lặp lại không ngừng.

Minh hoạ:

Cách 1:

Cách này đơn giản là làm cho các led sáng bằng cách thiết đặt các giá trị thích hợp cho các bit của Port để làm led sáng theo từng trạng thái, cách này đơn giản nhưng cần viết dài và tốn dung lượng bộ nhớ Rom.

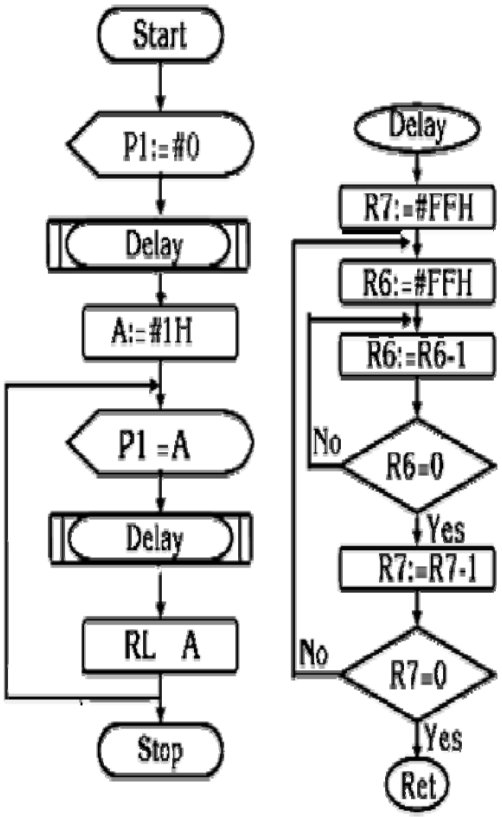
Chương trình

```
;***************/////--- LED SANG LAN LUOT -CACH DON GIAN
;**/////////////////___ led sang lan luot tu led 1 den led 8-P1
;**/////////////////___ sang o muc 1 va tat o muc 0
;**///////////////____ lap di lap lai khong gioi han
;** \\\\\\\\\\\\------- nap gia trị cho tung trang thai
        ORG             000H ; khai bao dia chi de bat dau chuong trinh tren Mov
        P1, #00000000B ;===>>>> lam 8 led noi P1 tat
        LCall           Delay ;===>>>>goi chuong trinh con Delay
        Xuat:  Mov              P1, #00000001B ;===>>>> lam led 1 sang
        LCall           Delay
        Mov             P1, #00000010B ;===>>>> lam led 2 sang
        LCall Delay
        Mov             P1, #00000100B ;===>>>> lam led 3 sang
        LCall Delay
        Mov             P1, #00001000B ;===>>>> lam led 4 sang
        LCall           Delay
        Mov             P1, #00010000B ;===>>>> lam led 5 sang
        LCall           Delay
        Mov             P1, #00100000B ;===>>>> lam led 6 sang
        LCall           Delay
        Mov             P1, #01000000B ;===>>>> lam led 7 sang
        LCall           Delay
        Mov             P1, #10000000B ;===>>>> lam led 8 sang
        LCall           Delay
        Sjmp            Xuat ;===>>>> nhay ve thuc hien tu dau
;**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  delay
        Delay:
        Mov             R7, #0FFH ;===>>>> gan R7=#0FFH, 1 chu ki may
        Kt2:   Mov              R6, #0FFH ;===>>>> gan R6=#0FFH, 1 chu ki may
        Kt1:   Djnz             R6, Kt1 ;===>>>> giam R6 xuong 1, R6 khac 0-> Djnz
               R7, Kt2 ;===>>>> giam R7 xuong 1, R6 khac 0->
        Ret ;===>>>> ket thuc chuong trinh con
;_____________________________________________________________
```

End ;===>>>> ket thuc chuong trinh

Cách 2:

Để led sáng lần lượt, cần làm cho tín hiệu xuất ra giá trị 1 một khoảng thời gian để có thể nhận biết được. Đầu tiên cho P1 tắt, sau đó làm cho bit A.0 lên 1, sử dụng lệnh xoay trái dữ liệu trên thanh Ram A, mỗi lần xoay giá trị 1 sẽ chuyển lần lượt qua A.1 - A.2 - A.3 -A.4 - A.5 -A.6 - A.7 - A.0, mỗi lần xoay xuất tín hiệu ra P1 sẽ thấy led sáng lần lượt từ led 1 đến led 8.



Chương trình:

```
;***************/////--- LED SANG LAN LUOT ---\\\\\\
;**////////////////___ led sang lan luot tu led 1 den led 8
;**////////////////___ sang o muc 1 va tat o muc 0
;**//////////////____ lap di lap lai khong gioi han
;** \\\\\\\\\\\\------- nap thanh Ram A=#00000001B
;**\\\\\\\\\\\\\--------ket hop xoay phai thanh Ram A va xuat du lieu
;*********************************************************************
        ORG     000H ; khai bao dia chi de bat dau chuong trinh tren Mov     P1,#0
;===>>>> lam 8 led noi P1 tat
        LCall   Delay ;===>>>>goi chuong trinh con Delay
        Mov     A, #00000001B ;===>>>> A co gia tri 00000001B
        Xuat:
        Mov     P1, A ;===>>>> xuat du lieu A ra P1 dieu khien led
        LCall   Delay ;===>>>>goi chuong trinh con Delay
        RL      A
```

```
        Sjmp        Xuat ;===>>>> nhay ve thuc hien tu dau
;**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  delay
        Delay:
        Mov         R7, #0FFH
        Kt2:  Mov       R6, #0FFH
        Kt1:  Djnz      R6, Kt1
        Djnz        R7, Kt2
        Ret
;_____________________________________________________________________
        End ;===>>>> ket thuc chuong trinh
```

Giải thích:

ORG 000H là lệnh chỉ dẫn: chương trình này được lưu vào ROM bắt đầu từ địa chỉ 000H. Khi mới bắt đầu chương trình, Mov P1,#0 làm tất cả các led đều tắt, LCall Delay làm cho thời gian tắt đủ để mắt người nhận ra. Mov A,#00000001B làm cho bit A.0 của thanh ghi A được set lên 1, các bit còn lại ở giá trị 0.

Tiếp theo trong chương trình là nhãn Xuat: vị trí trở về, trên lưu đồ giải thuật là điểm mũi tên trở về. Mov P1,A xuất tín hiệu ra P1, lúc này led 1 sáng lên. Tiếp theo là lệnh gọi chương trình con LCall Delay. Câu lệnh RL A làm trạng thái các bit dịch sang trái 1 bit, trước đó chỉ có A.0 ở giá trị 1, sau câu lệnh A.1 mang giá trị 1, còn bit A.0 bây giờ mang giá trị 0 do bit A.7 có giá trị 0 chuyển sang.

Sjmp Xuat chương trình nhảy đến đoạn chương trình được đánh dấu bằng nhãn Xuat, nên nhớ rằng chương lệnh gọi Sjmp chỉ nhảy được đến vị trí cách lệnh gọi không quá 128 byte, nếu vượt qua giới hạn này chương trình dịch sẽ báo lỗi, khi đó cần thay Sjmp bằng Ljmp vì Ljmp có thể nhảy đến vị trí bất kì trên ROM.

Lúc này chương trình bắt đầu từ nhãn Xuat: Sau nhãn Xuat: là câu lệnh Mov P1,A tín hiệu được xuất ra P1, lúc này P1.1 lên giá trị 1, led thứ hai sáng, các led còn lại không sáng. Câu lệnh RL A làm trạng thái các bit dịch sang trái 1 bit: A.2 lên 1, các bit khác mang giá trị 0.

Các quá trình trên được lặp lại liên tục (với câu lệnh Sjmp Xuat và nhãn Xuat:). Kết quả cuối cùng như trên hình minh họa phía trên.

2.2. Nhóm lệnh số học-logic.

2.2.1. Nhóm lệnh số học

2.2.2.1. Thanh ghi trạng thái chương trình PSW

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

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

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

Ví dụ: phép cộng không tràn

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

Phép cộng tràn

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

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

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

Số trừ	6CH	108	0 1 1 0 1 1 0 0 b
Số bị trừ	-9FH	159	1 0 0 1 1 1 1 1 b
Kết quả	CDH	-51	1 1 0 0 1 1 0 1 b
Cờ nhớ C	1		1 -phép trừ trên có số mượn

2.2.2.2. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu trên thanh ghi Rn

Cú pháp: Add A, Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên thanh ghi Rn, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW

Ví dụ 1:

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

Kết quả: A có giá trị là 28H. R1 vẫn giữ nguyên giá trị là 08H. Cờ C = 0

Ví dụ 2:

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

Kết quả: A = #0A3h. R6 = #0Bah. Cờ C = 1

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

Cú pháp: Add A, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên ô nhớ có địa chỉ direct, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW

Ví dụ:

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

Kết quả: A = #08H
50H = #20H
C = 1

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

Cú pháp: Add A,@Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

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

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

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

Cú pháp: Add A, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với một giá trị xác định, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW

Ví dụ:

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

Kết quả: A = #0Bh

C = 0

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

Cú pháp: AddC A, Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

C = 1

Mov A, #08h

Mov R1, #10h

Addc A, R1

Kết quả: A = #19h ; cộng cả cờ C

R1 = #10h

C =0

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

Cú pháp: AddC A, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

C = 0

Mov A, #0A5h

Mov 10h, #96h

Addc A, 10h

Kết quả: A = #3Bh

10h = #96h

C =1

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

Cú pháp: AddC A,@Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri và cộng thêm giá trị của số nhớ trên cờ C, sau khi

thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW

Ví dụ:

C = 1

Mov A, #05h

Mov 50h,#10h

Mov R2,#50h

Addc a,@R2

Kết quả: A = #16h

C = 0

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

Cú pháp: AddC A,#data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

C = 1

Mov A, #05h

Addc A, #16h

Kết quả: A = #1Ch

C = 0

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

Cú pháp: SubB A, Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên thanh ghi Rn và trừ cho giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW

Ví dụ:

C = 1

Mov A, #0E5h

Mov R3, #9Fh

Subb A, R3

Kết quả: A = 45h

C = 0

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

Cú pháp: SubB A, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ direct và trừ cho giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW

Ví dụ:

C = 0

Mov A, #0E5h

Mov 05h, #9Fh

Subb A, 05h

Kết quả: A = 46h

C = 0

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

Cú pháp: SubB A, @Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

C = 1

Mov A, #0E5h

Mov 4Fh, #50h

Mov R3, #4Fh

Subb A, @R3

Kết quả: A = 94h

C = 0

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

Cú pháp: SubB A,#data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị xác định và trừ thêm giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW

Ví dụ:

C = 0

Mov A, #05h

Subb A, #4Fh

Kết quả: A = 0B6h

C = 1

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

Cú pháp: Inc A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Tăng giá trị dữ liệu lưu giữ trên thanh ghi A lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW

Ví dụ:

Mov A, #05h

Inc A

Kết quả: A = #06h

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

Cú pháp: Inc Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Tăng giá trị dữ liệu lưu giữ trên thanh ghi Rn lên1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW

Ví dụ:

Mov R7, #0Fh

Inc R7

Kết quả: R7 = #10h

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

Cú pháp: Inc direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Tăng giá trị dữ liệu ở một ô nhớ có địa chỉ direct lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW

Ví dụ:

Mov 50h, #0FFh

Inc 50h

Kết quả: 50h = #00

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

Cú pháp: Inc @Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Tăng giá trị dữ liệu ở ô nhớ có địa chỉ bằng giá trị dữ liệu trên Ri lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW

Ví dụ:

Mov 0Fh, #05h

Mov R0, #0Fh

Inc @R0

Kết quả: R0 = #06h

0Fh = #05h

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

Cú pháp: Inc DPTR

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Tăng giá trị dữ liệu của thanh ghi con trỏ dữ liệu DPTR lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW

Ví dụ:

Mov DPTR, #5Fh

Inc DPTR

Kết quả: DPTR = #060h

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

Cú pháp: Dec A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Giảm giá trị dữ liệu lưu giữ trên thanh ghi A xuống 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW

Ví dụ:

Mov A, #05h

Dec A

Kết quả: A = #04h

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

Cú pháp: Dec Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Giảm giá trị dữ liệu lưu giữ trên thanh ghi Rn xuống 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW

Ví dụ:

Mov R6, #0Fh

Dec R6

Kết quả: R6 = #0Eh

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

Cú pháp: Dec direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Giảm giá trị dữ liệu ở ô nhớ có địa chỉ direct xuống 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW

Ví dụ:

Mov 7Fh, #0

Dec 7Fh

Kết quả: 7Fh = #0FFh

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

Cú pháp: Dec @Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: Giảm giá trị dữ liệu ở ô nhớ có địa chỉ bằng giá trị dữ liệu trên Ri xuống1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW

Ví dụ:

Mov 60h, #05h

Mov R1, #60h

Dec @R1

Kết quả: R1 = #04h

60h = #05h

2.2.2.23. Lệnh nhân thanh ghi A với thanh ghi B

Cú pháp: Mul AB

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 4 chu kì máy

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

Ví dụ:

Mov A, #0B9h

Mov B, #F7h

Mul AB

Kết quả: A = #7Fh

B = #0B2h

2.2.2.24. Lệnh chia thanh ghi A với thanh ghi B

Cú pháp: Div AB

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 4 chu kì máy

Công dụng: Chia hai dữ liệu là số nguyên không dấu ở thanh ghi A với thanh ghi B, dữ liệu ở thanh ghi A là số chia còn ở thanh ghi B là số bị chia, kết quả là một dữ liệu 8 bit được lưu ở thanh ghi A.số dư lưu trữ trong thanh ghi B Cờ nhớ C luôn ở giá trị 0. Cờ tràn OV được thiết lập giá trị 1 khi thanh ghi B mang giá trị là 00H-phép chia không thể thực hiện.

Ví dụ:

Mov A, #50h

Mov B, #10h

DIV AB

Kết quả: A = #5h

B = #0h

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

Cú pháp: DA A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 4 chu kì máy

Công dụng: hiệu chỉnh dữ liệu là giá trị lưu giữ ở thanh ghi A từ số Hex (số nhị phân) thành số BCD (số thập phân viết dưới dạng nhị phân). Lí do có lệnh hiệu chỉnh này vì khi cộng hai giá trị là số BCD bằng các lệnh cộng, vi điều khiển chỉ hiểu hai số cộng là số nhị phân bình thường, kết quả sau lệnh cộng là một số nhị phân bình thường, không phải là một số BCD, vì vậy kết quả cần được hiệu chỉnh để dữ liệu cuối là một số BCD. Khi thực hiện lệnh, cờ nhớ C được xác lập lên 1 nếu phép cộng có kết quả vượt qua 99 (số BCD). Kết quả cuối cùng, số BCD có hàng đơn vị nằm ở 4 bit thấp trên thanh ghi A, hàng chục ở 4 bit cao của thanh ghi A, hang trăm là 1 nếu cờ C mang giá trị 1, là 0 nếu cờ C mang giá trị 0.

Ví dụ 1:

Mov A, #10h

DA A

Kết quả: A = #10h

Ví dụ 2:

Mov A,#0Eh

DA A

Kết quả: A = #14h

2.2.2. Nhóm lệnh logic.

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

Cú pháp: ANL A, Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là: 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thực hiện phép logic AND dữ liệu ở thanh ghi A với dữ liệu ở thanh ghi Rn, kết quả được lưu trữ ở thanh ghi A

Ví dụ:

mov A, #0Fh

mov R1, #0F0h

ANL A, R1

Kết quả: A = #0H

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

Cú pháp: ANL A, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thực hiện phép logic AND dữ liệu ở thanh ghi A với dữ liệu ở ô nhớ có địa chỉ direct, kết quả được lưu trữ ở thanh ghi A

Ví dụ:

mov A, #0FFh
mov 10h, #010h
ANL A, 10h

Kết quả: A = #010h

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

Cú pháp: ANL A,@Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

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

Kết quả: A = #01h

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

Cú pháp: ANL A, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thực hiện phép logic AND dữ liệu ở thanh ghi A với dữ liệu cho trước, kết quả được lưu trữ ở thanh ghi A

Ví dụ:

mov A, #0Eh
ANL A, #11h

Kết quả: A = #00

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

Cú pháp: ANL direct, A

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thực hiện phép logic AND dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct, kết quả được lưu trữ ở ô nhớ có địa chỉ direct.

Ví dụ:

mov A, #08h
mov R1, #0F7h
ANL R1, A

Kết quả: R1 = #0

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

Cú pháp: ANL direct, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thực hiện phép logic AND dữ liệu của ô nhớ có địa chỉ direct với dữ liệu cho trước, kết quả được lưu trữ ở ô nhớ có địa chỉ direct.

Ví dụ:

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

Kết quả: R1 = #017h

2.2.2.7. Lệnh OR dữ liệu ở thanh ghi A với dữ liệu ở thanh ghi Rn

Cú pháp: ORL A, Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là: 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thực hiện phép logic OR dữ liệu ở thanh ghi A với dữ liệu ở thanh ghi Rn, kết quả được lưu trữ ở thanh ghi A

Ví dụ:

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

Kết quả: A = #0FFh

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

Cú pháp: ORL A, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thực hiện phép logic OR dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct, kết quả được lưu trữ ở thanh ghi A

Ví dụ:

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

Kết quả: A = #0FEh

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

Cú pháp: ORL A,@Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

```
mov A, #18h
mov 30h, #0F0h
mov R1, #30h
ORL A, @R1
```

Kết quả: A = #0F8h

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

Cú pháp: ORL A, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

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

Kết quả: A = #010h

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

Cú pháp: ORL direct, A

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thực hiện phép logic OR dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct, kết quả được lưu trữ ở ô nhớ có địa chỉ direct.

Ví dụ:

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

Kết quả: 5Fh = #0FFh

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

Cú pháp: ORL direct, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: thực hiện phép logic OR dữ liệu của ô nhớ có địa chỉ direct với dữ liệu cho trước, kết quả được lưu trữ ở ô nhớ có địa chỉ direct.

Ví dụ:

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

Kết quả: 60h = #0FFh

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

Cú pháp: XRL A, Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là: 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thực hiện phép logic EX-OR dữ liệu ở thanh ghi A với dữ liệu ở thanh ghi Rn, kết quả được lưu trữ ở thanh ghi A

Ví dụ:

```
mov A, #0F2h
mov R3, #0E0h
XRL A, R3
```

Kết quả: A = #12h

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

Cú pháp: XRL A, direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thực hiện phép logic EX-OR dữ liệu ở thanh ghi A với dữ liệu của ô nhớ có địa chỉ direct, kết quả được lưu trữ ở thanh ghi A

Ví dụ:

mov A, #012h

mov 10h, #0E0h

XRL A,1 0h

Kết quả: A = #0F2h

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

Cú pháp: XRL A,@Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

mov A, #08h

mov 10h, #0E9h

mov R0, #10h

XRL A, @R0

Kết quả: A = #0E1h

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

Cú pháp: XRL A, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

mov A, #12h

XRL A, #12h

Kết quả: A = #0

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

Cú pháp: XRL direct, A

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 1 chu kì máy

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

Ví dụ:

mov A, #0F2h

mov 50h, #0E0h

XRL 50h, A

Kết quả: 50h = #12h

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

Cú pháp: XRL direct, #data

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: thực hiện phép logic EX-OR dữ liệu của ô nhớ có địa chỉ direct với dữ liệu cho trước, kết quả được lưu trữ ở ô nhớ có địa chỉ direct.

Ví dụ:

mov 50h, #0E0h

XRL 50h, #01h

Kết quả: 50h = #0E1h

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

Cú pháp: CPL A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: lấy bù giá trị lưu giữ ở thanh ghi A, các bit có giá trị là 1 chuyển thành 0 và ngược lại các bit có giá trị là 0 chuyển thành 1.

Ví dụ:

mov A, #01100111b ;(tương đương 67h)

CPL A

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

2.2.2.20. Lệnh xóa dữ liệu trên thanh ghi A

Cú pháp: CLR A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: tất cả các bit của thanh ghi A đều được xác lập giá trị 0 .

Ví dụ:

mov A, #01100111b

CLR A

Kết quả: A = #0

2.2.2.21. Lệnh xoay trái dữ liệu trên thanh ghi A

Cú pháp: RL A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thanh ghi A gồm tám bit A7 A6 A5 A4 A3 A2 A1 A0. Khi thực hiện lệnh xoay trái RL A giá trị của các bit được chuyển trang bit ở bên trái nó, giá trị của bit

A0 chuyển sang bit A1, giá trị của bit A1 chuyển sang bit A2, tương tự với các bit còn lại, và giá trị của bit A7 chuyển sang bit A0.

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

Cú pháp: RLC A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thanh ghi A gồm tám bit A7 A6 A5 A4 A3 A2 A1 A0. Khi thực hiện lệnh xoay trái A với cờ nhớ RLC A giá trị của các bit được chuyển trang bit ở bên trái nó, giá trị của bit A0 chuyển sang bit A1, giá trị của bit A1 chuyển sang bit A2, tương tự với các bit còn lại, và giá trị của bit A7 chuyển sang cờ nhớ C, giá trị ở cờ nhớ C chuyển sang bit A0

2.2.2.23. Lệnh xoay phải dữ liệu trên thanh ghi A

Cú pháp: RR A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thanh ghi A gồm tám bit A7 A6 A5 A4 A3 A2 A1 A0. Khi thực hiện lệnh xoay phải RR A giá trị của các bit được chuyển trang bit ở bên phải nó, giá trị của bit A7 chuyển sang bit A6, giá trị của bit A6 chuyển sang bit A5, tương tự với các bit còn lại, và giá trị của bit A0 chuyển sang bit A7.

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

Cú pháp: RRC A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: thanh ghi A gồm tám bit A7 A6 A5 A4 A3 A2 A1 A0. Khi thực hiện lệnh xoay phải A với cờ nhớ -RRC A-giá trị của các bit được chuyển trang bit ở bên phải nó, giá trị của bit A7 chuyển sang bit A6, giá trị của bit A6 chuyển sang bit A5, tương tự với các bit còn lại, và giá trị của bit A0 chuyển sang cờ nhớ C, giá trị ở cờ nhớ C chuyển sang bit A7

2.2.2.25. Lệnh xoay 4 bit trên thanh ghi A

Cú pháp: SWAP A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: hoán chuyển dữ liệu ở 4 bit thấp lên 4 bit cao và 4 bit cao xuống 4 bit thấp

Ví dụ:

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

Kết quả: A = # 7Eh

*** BÀI TẬP MẪU**

Hãy viết chương trình điều khiển 16 led của 2 port: port0 và port1 sáng dồn

CHƯƠNG TRÌNH

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0, port 1 sang don va tat het
;ket noi port 0 va port 1 den 16 led bang 2 soi cap 8 soi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        org 0000h
        port01_6: mov r4,#00h ;luu trng thai ban dau
        mov r6,#00h
        mov p0,#00h ;luu trng thai ban dau
        mov p1,#00h
        lcall delay ;goi chuong trinh con delay
        mov 10h,#16 ;goi bien dem so lan dich chuyen cua led
        port01_6a: mov 11h,10h ;chuyen bien dem tung led
        mov r5,#00h ;nap 00 vao r5
        mov r7,#00h ;nap 00 vao r7
        setb c ;lam cho bit C = 1
        port01_6b:
        mov a,r7
        rrc a ;xoay noi dung thanh ghi A sang trai
        mov r7,a ;cat lai vao r7 de luu cho lan xu li ke
        orl a,r4 ;lay ket qua do or voi r4 roi goi
        ra p1
        mov p1,a
        mov a,r5
        rrc a ;xoay noi dung thanh ghi A sang trai
        mov r5,a ;cat lai vao r5 de luu cho lan xu li ke
        orl a,r6 ;lay ket qua do or voi r5 roi goi
        ra p0
        mov p0,a
        lcall delay
        clr c ;xoa Cy de chi dich 1 led di
        djnz 11h,port01_6b ;giam ndung o nho (11h)<>0 thi quay lai
        mov r4,p1
        mov r6,p0
        djnz 10h,port01_6a ;giam bien dem de xu li lan ke
        ljmp port01_6
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        delay: mov 7eh,#040h
        del: mov 7fh,#0ffh
```

djnz 7fh,$
djnz 7eh,del
ret
end

2.3. Nhóm lệnh so sánh.

STT	Cú pháp		Mô tả	Số byte	Số chu kỳ
	Mã lệnh	Toán hạng			
1	CLR	C	Xóa cờ C về 0	1	1
2	CLR	Bit	Xóa bit về 0	2	1
3	SETB	C	Đặt cờ C = 1	1	12
4	SETB	Bit	Đặt bit = 1	2	1
5	CPL	C	Đảo giá trị của cờ C	1	1
6	CPL	Bit	Đảo giá trị của bit	2	1
7	ANL	C,bit	C = (C)and(bit)	2	2
8	ANL	C,/bit	C = (C)and(đảo của bit)	2	2
9	ORL	C,bit	C = (C)or(bit)	2	2
10	ORL	C,/bit	C = (C)or(đảo của bit)	2	2
11	MOV	C,bit	C = bit	2	1

2.4. Nhóm lệnh nhảy.

2.4.1. Lệnh nhảy thuận với giá trị của bit nhớ

Cú pháp: JB bit, rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng:

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

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

2.4.2. Lệnh nhảy nghịch với giá trị của bit nhớ

Cú pháp: JNC bit, rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng:

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

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

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

Cú pháp: JBC bit, rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng:

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

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

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

Cú pháp: CJNE A, direct, rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Vi điều khiển nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt nếu giá trị của thanh ghi A khác giá trị của ô nhớ có địa chỉ direct, nếu bằng nhau Vi điều khiển không nhảy và thực hiện lệnh kế. Ảnh hưởng của lệnh đến cờ nhớ C: Nếu giá trị của thanh ghi A ≥ giá trị của ô nhớ có địa chỉ direct thì bit C có giá trị 0. Nếu giá trị của thanh ghi A < giá trị của ô nhớ có địa chỉ direct thì bit C có giá trị 1

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

Cú pháp: CJNE A, #data, rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

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

Ảnh hưởng của lệnh đến cờ nhớ C: Nếu giá trị của thanh ghi A ≥ giá trị dữ liệu cho trước thì bit C có giá trị 0. Nếu giá trị của thanh ghi A < giá trị dữ liệu cho trước thì bit C có giá trị 1

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

Cú pháp: CJNE Rn, #data, rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt nếu giá trị của thanh ghi Rn khác giá trị dữ liệu cho trước, nếu bằng nhau Vi điều khiển không nhảy và thực hiện lệnh kế. Ảnh hưởng của lệnh đến cờ nhớ C: Nếu giá trị của thanh ghi A ≥ giá trị dữ liệu cho trước thì bit C có giá trị 0. Nếu giá trị của thanh ghi A < giá trị dữ liệu cho trước thì bit C có giá trị 1.

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

Cú pháp:

CJNE @Ri, #data, rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Vi điều khiển nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt nếu giá trị của ô nhớ có địa chỉ bằng giá trị của Ri khác giá trị dữ liệu cho trước,

nếu bằng nhau Vi điều khiển không nhảy và thực hiện lệnh kế. Ảnh hưởng của lệnh đến cờ nhớ C: Nếu giá trị của ô nhớ có địa chỉ gián tiếp ≥ giá trị dữ liệu cho trước thì bit C có giá trị 0. Nếu giá trị của ô nhớ có địa chỉ gián tiếp < giá trị dữ liệu cho trước thì bit C có giá trị 1

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

Cú pháp:

DJNZ Rn, rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Giảm giá trị của thanh ghi Rn xuống 1 đơn vị, và

- Nếu giá trị trong thanh ghi Rn khác 0, Vi điều khiển nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt.

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

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

Cú pháp: DJNZ direct, rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Giảm giá trị của ô nhớ có địa chỉ direct xuống 1 đơn vị. Nếu giá trị trong ô nhớ có địa chỉ direct khác 0, Vi điều khiển nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt. Nếu giá trị trong ô nhớ có địa chỉ direct bằng 0, Vi điều khiển thực hiện lệnh kế tiếp

2.4.10. Lệnh delay 1 chu kì máy

Cú pháp: NOP

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 1 chu kì máy

Công dụng: delay trong 1 chu kì máy

2.5. Nhóm lệnh về ngăn xếp.

2.5.1. Lệnh PUSH direct

Chức năng: Cất vào ngăn xếp (stack).

Mô tả: Con trỏ stack được tăng bởi 1. Nội dung của toán hạng được chỉ ra trong lệnh sau đó được sao chép vào RAM nội tại địa chỉ được trỏ đến bởi con trỏ stack. Các cờ không bị ảnh hưởng.

Số byte	2
Số chu kỳ	2
Mã đối tượng	11000000 aaaaaaaa
Hoạt động	(SP) ←(SP) + 1
	((SP)) ←(direct)

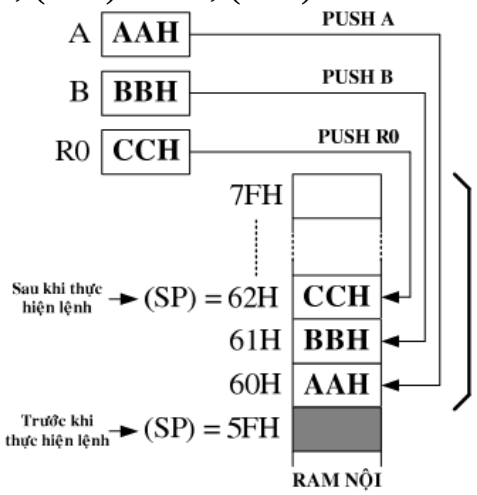
Ví dụ: Cất tạm thời nội dung của thanh ghi A, B và R0 vào ngăn xếp. Cho biết trước (A)=AAH, (B)=BBH, (R0)=CCH, (SP)=5FH.

Sau khi thực thi chuỗi lệnh: PUSH A

PUSH B

PUSH 00H

Thì: (SP)=62H, (60H)=AAH, (61H)=BBH, (62H)=CCH



2.5.2. Lệnh POP direct

Chức năng: Lấy ra từ ngăn xếp (stack).

Mô tả: Nội dung của vùng RAM nội được định địa chỉ bởi con trỏ stack SP được đọc và nội dung con trỏ stack được giảm bởi 1. Giá trị đọc được sau đó được chuyển đến byte được định địa chỉ trực tiếp chỉ ra trong lệnh. Các cờ không bị ảnh hưởng.

Số byte	2
Số chu kỳ	2
Mã đối tượng	11010000 aaaaaaaa
Hoạt động	(direct) ← ((SP))
	(SP) ← (SP) – 1

Ví dụ: Lấy lại nội dung của thanh ghi A, B và R0 đã cất vào ngăn xếp lúc đầu (ví dụ trên).

Sau khi thực thi chuỗi lệnh: POP 00H

POP B

POP A

Thì: (SP)=5FH, (R0)=CCH, (B)=BBH, (A)=AAH

2.5.3. Lệnh XCH A.

Chức năng: Tráo đổi nội dung của thanh ghi A với nội dung của một byte

Mô tả: XCH nạp cho thanh ghi A nội dung của byte chỉ ra trong lệnh, đồng thời ghi nội dung ban đầu của thanh ghi A cho byte vừa nêu trên. Toán hạng nguồn đồng thời là toán hạng đích và ngược lại.

Các dạng lệnh: XCH A, Rn

Số byte	1
Số chu kỳ	1
Mã đối tượng	11001rrr
Hoạt động	(A) ↔ (Rn)

XCH A, direct

Số byte	2

Số chu kỳ 1
Mã đối tượng 11000101 aaaaaaaa
Hoạt động (A) ↔ (direct)

XCH A, @Ri

Số byte 1
Số chu kỳ 1
Mã đối tượng 1100011i
Hoạt động (A) ↔((Ri))

2.5.4. Lệnh XCHD A, @Ri

Chức năng: Tráo ñổi digit (Exchange Digit).

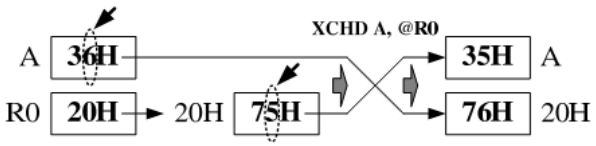
Mô tả: XCHD tráo đổi nội dung nửa thấp của thanh ghi A (biểu diễn một digit số hex hoặc BCD) với nội dung nửa thấp của một byte trong RAM nội, byte này được định địa chỉ gián tiếp bởi thanh ghi chỉ ra trong lệnh. Nửa cao của các thanh ghi vừa nêu trên không bị ảnh hưởng và các cờ cũng không bị ảnh hưởng.

Số byte 1
Số chu kỳ 1
Mã đối tượng 110101i
Hoạt động (A3 – A0) ↔(Ri3 – Ri0)

Ví dụ: Cho biết trước (A)=36H, (R0)=20H, (20H)=75H.

Sau khi thực thi lệnh XCHD A, @R0 thì: (A)=35H, (20H)=76H

Sau khi thực thi lệnh **XCHD A, @R0** thì: (A)=35H, (20H)=76H



2.6. Nhóm lệnh điều khiển.

2.6.1. Nhãn

Nhãn: Kí hiệu: rel

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

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

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

Ví dụ:

Các nhãn đúng X1: ;S_2: ;_5:s10: ;... Các nhãn sai 1X: ; S_2 ;S 5: ;DW: ,LPT:...

2.6.2. Chương trình con

Chương trình con: là những đoạn chương trình thực hiện một số lệnh nào đó và được viết ngoài chương trình chính, các chương trình con này được đặt tên bằng một nhãn và kết thúc bằng lệnh RET, chương trình con có thể gọi một chương trình con khác. Chương trình con được chương trình chính sử dụng khi cần thiết bằng các lệnh gọi

chương trình con; khi có lệnh gọi chương trình con, Vi điều khiển chuyển về thực hiện các đoạn chương trình của chương trình con, sau khi thực hiện chương trình con Vi điều khiển tiếp tục trở về thực hiện các câu lệnh trong chương trình chính.

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

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

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

Cú pháp:

ACall addr11

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Khi lệnh được thực hiện, Vi điều khiển chuyển về thực hiện các câu lệnh của chương trình con bắt đầu từ địa chỉ addr11 trên Rom, địa chỉ addr11 có thể thay bằng nhãn bắt đầu của một chương trình con. Câu lệnh được thực hiện khi địa chỉ addr11 cách lệnh gọi không quá 2 KByte .

Ví dụ:

ACall 45A6H

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

Cú pháp: ACall addr16

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

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

2.6.3. Lệnh kết thúc chương trình con

Cú pháp: Ret

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Lệnh này dùng kết thúc chương trình con, khi gặp lệnh này Vi điều khiển quay về thực hiện lệnh ở chương trình chính.

2.6.4. Lệnh kết thúc chương trình con phục vụ ngắt

Cú pháp: Reti

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Lệnh này dùng kết thúc chương trình con ngắt, khi gặp lệnh này Vi điều khiển quay về thực hiện lệnh ở chương trình chính.

2.6.5. Lệnh nhảy ngắn đến địa chỉ tuyệt đối

Cú pháp: AJMP addr11

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

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

2.6.6. Lệnh nhảy dài đến địa chỉ tuyệt đối

Cú pháp: LJMP addr16

Lệnh này chiếm dung lượng bộ nhớ ROM là 3 Byte

Thời gian thực hiện: 2 chu kì máy

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

2.6.7. Lệnh nhảy tương đối

Cú pháp:

SJMP rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

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

2.6.8. Lệnh nhảy gián tiếp

Cú pháp:

JMP @A+DPTR

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Khi lệnh được thực hiện, Vi điều khiển chuyển đến thực hiện các câu lệnh của chương trình có địa chỉ trên ROM bằng giá trị của A cộng với giá trị lưu giữ trên DPTR

2.6.9. Lệnh nhảy thuận với cờ Zero

Cú pháp: JZ rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

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

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

2.6.10. Lệnh nhảy nghịch với cờ Zero

Cú pháp: JNZ rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Nếu cờ Zero có giá trị 0(tức thanh ghi A có giá trị khác 0), Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt. Nếu cờ Zero có giá trị 1(tức thanh ghi A có giá trị 0), Vi điều khiển thực hiện lệnh kế tiếp (không thực hiện lệnh nhảy)

2.6.11. Lệnh nhảy thuận với cờ C

Cú pháp: JC rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Nếu cờ C có giá trị 1, Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt. Nếu cờ C có giá trị 0, Vi điều khiển thực hiện lệnh kế tiếp (không thực hiện lệnh nhảy)

2.6.12. Lệnh nhảy nghịch với cờ Zero

Cú pháp: JNC rel

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte

Thời gian thực hiện: 2 chu kì máy

Công dụng: Nếu cờ C có giá trị 0, Vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ mà nhãn được đặt. Nếu cờ C có giá trị 1, Vi điều khiển thực hiện lệnh kế tiếp (không thực hiện lệnh nhảy)

***Bài tập mẫu:**

Phần cứng: 8 led nối với Port 0 được định vị trí như sau:led 1 nối với P0.0, lầ n lượt cho đến led 8 (nối với P0.7). Biết led sáng khi tín hiệu xuất ở mức 1.Viết chương trình để các led nối với Port 0 sáng dồn từ led 1 đến led 8 sau đó tắt hết led và lặp lại. Các quá trình được lặp lại không giới hạn.

Cách 1:

Cách này đơn giản là làm cho các led sáng bằng cách thiết đặt các giá trị thích hợp cho các Port để làm led sáng theo từng trạng thái, cách này đơn giản nhưng cần viết dài và tốn dung lượng bộ nhớ Rom. Nếu cần thiết các bạn tự giải

Cách 2:

Ở cách này giải thuật phức tạp hơn: Chia quá trình sáng thành hai quá trình đơn: quá trình có 1 led sáng xoay vòng và quá trình các led sáng cố định có lưu giữ trạng thái led cuối chu trình, kết hợp hai chu trình trên sẽ ra kết quả cần thực hiện. Dùng thanh ghi R3 để làm cho bit mang giá trị 1 xoay vòng như ở bài 5.

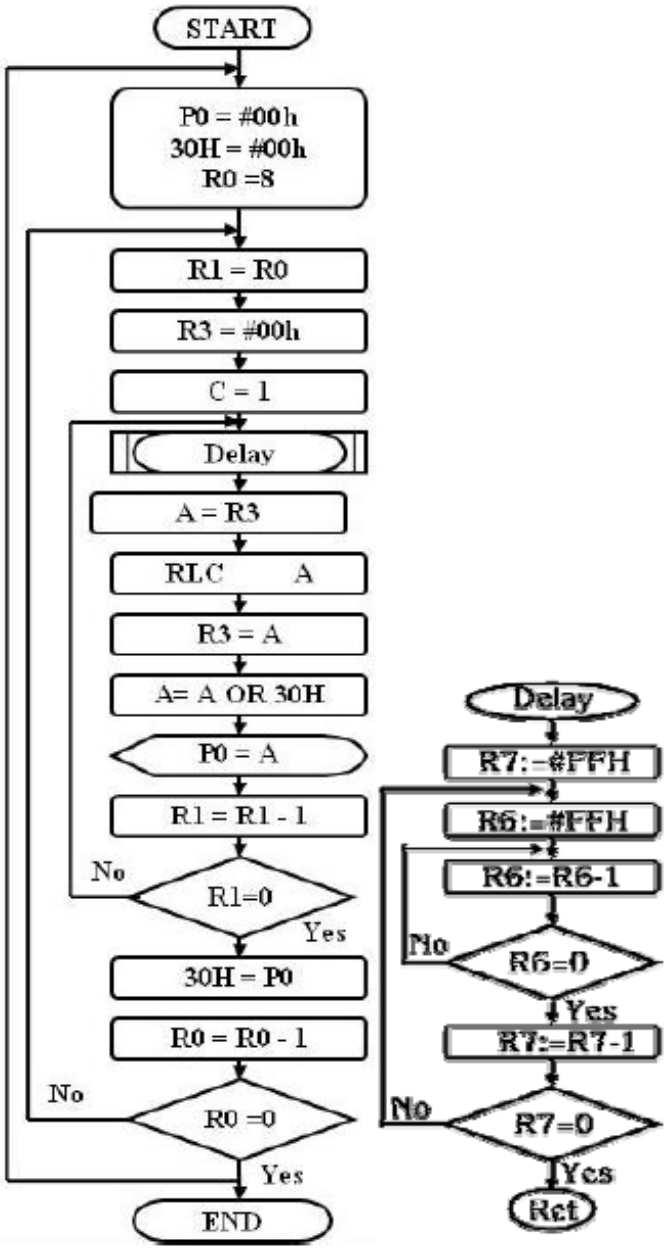
Dùng ô nhớ 30H dùng lưu giá trạng thái cuối của quá trình.(ban đầu 30H= #00000000B)

Thực hiện OR: R3 với 30H rồi xuất ra P0 sau mỗi lần R3 xoay 1 bit.

Trong quá trình đầu: một bit mang giá trị 1 trên R3 xoay từ vị trí R3.0 đến R3.7, vì 30H lúc này đang mang giá trị là #00000000B, nên khi OR R3 với 30H rồi xuất ra P0 sẽ thấy 1 led sáng di chuyển từ led 1 đến led 8. Khi vị trí sáng đến led thứ 8 vi điều khiển lưu lại giá trị của P0 vào 30H.

Trong quá trình hai: bit mang giá trị 1 trên R3 vẫn xoay, lúc này 30H có bit 30H.7 đang ở giá trị 1 tức là 30H đang mang giá trị #10000000B, nên khi OR R3 với 30H rồi xuất ra P0 sẽ thấy led 8 sáng cố định, trong lúc đó có 1 led sáng di chuyển từ led 1 đi vào. Đến led 7 vi điều khiển lưu lại giá trị P0 vào 30H (giá trị mới lưu là #11000000B).
Các quá trình tiếp tục như trên. Như vậy cần phải đếm số quá trình đã thực hiện, khi quá trình thực hiện đến lần thứ 8, lúc này các led đều sáng hết, quá trình phải được lặp lại từ đầu. Dùng thanh ghi R0 để lưu giữ giá trị này. Trong quá trình thực hiện, số bit tham gia vào xoay giá trị 1 sẽ giảm theo các quá trình, vì vậy cần kiểm soát số lần xoay trái trong mỗi quá trình. Dùng thanh ghi R1 lưu giữ giá trị này.

Lưu đồ:



CHƯƠNG TRÌNH

```
;************************************************************
;**************/////--- SANG DON 8 LED
;**------------------------------------------------------------------------------
;**/////////////////___ Sang dan tu led 1 den led 8
;**/////////////////___ sang o muc 1 va tat o muc 0
;**/////////////////____ lap di lap lai khong gioi han
;**<<<<<<<<<<<<<<<<<<<----------------------------------------------------
;************************************************************************
*******************************
        ORG 000H ; khai bao dia chi de bat dau chuong trinh tren Rom
TuDau:
        Mov P0, #00h
        Mov R0, #8 ;+++>>>so qua trinh thuc hien
        Mov 30H, #00h
SD2:
        Mov 01H, 00H ;+++>>> vi khong co lenh Mov R1,R0
        Mov R3, # 00h
        SetB C
SD1:
        LCall Delay
        Mov A, R3 ;@@@@
        RLC A ;@@@@===>>>xoay trai R3
        Mov R3, A ;@@@@
        ORL A, 30H ;+++>>> thuc hien phep OR giua A va 30H, ket qua o Mov P0,A
;===>>>xuat ket qua ra P0
        DJNZ R1, SD1
        Mov 30H, P0 ;===>>> luu giu trang thai cuoi
        DJNZ R0, SD2
Ljmp TuDau
;**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  delay
Delay:
        Mov R7,#0FFH ;===>>>> gan R7=#0FFH, 1 chu ki may
        Kt2:    Mov R6,#0FFH ;===>>>> gan R6=#0FFH, 1 chu ki may
        Kt1:    Djnz R6,Kt1 ;===>>>> giam R6 xuong 1, R6 khac 0-> Djnz R7,Kt2
;===>>>> giam R7 xuong 1, R6 khac 0-> Ret ;===>>>> ket thuc chuong trinh con
;_____________________________________________________________________
End
```

CÂU HỎI ÔN TẬP

1. Cấu trúc, chức năng của 89C51?
2. Nhóm lệnh truyền dữ liệu của 89C51? Công dụng của nhóm lệnh truyền dữ liệu trong chương trình vi điều khiển?
3. Nhóm lệnh số học-logic của 89C51?
4. Nhóm lệnh so sánh của 89C51? Cho ví dụ minh họa?
5. Nhóm lệnh nhảy của 89C51?
6. Nhóm lệnh về ngăn xếp của 89C51?
7. Nhóm lệnh điều khiển của 89C51? Công dụng của nhóm lệnh điều khiển trong chương trình vi điều khiển?
8. Khái niệm nhãn? Nguyên tắc đặt tên nhãn cho chương trình?
9. Chương trình con là gì? Ưu nhược điểm của chương trình con?
10. Lưu đồ là gì? Tại sao khi viết chương trình phải có lưu đồ trước?

BÀI TẬP

1. Viết chương trình xuất tín hiệu ở Port 0 và Port 2 để 8 đèn led sáng với các led được kết nối với Port 0 và Port 2 như sơ đồ
2. Cho các dãy đèn led có kết nối như trong sơ đồ dưới, các dãy led này được kết nối với Port 1 và Port 2. Viết chương trình để các led nối mỗi Port sáng xen kẽ: led 1,3,5,7 sáng; led 2,4,6,8 tắt, các led được đánh số như sau led 1 nối với Px.0, led 2 nối với Px.1, lần lượt với các led khác.
3. Phần cứng: 32 led nối với Port 0,1,2,3, được định vị trí như sau: led 1 nối với P0.0, lần lượt cho đến led 32 (nối với P3.7). Biết led sáng khi tín hiệu xuất ở giá trị 1.Viết chương trình để các led nối với Port 0,1,2,3 sáng lần lượt từng led từ led 1 đến led 32. Các quá trình được lặp lại không giới hạn.
4. Làm cho các led nối Port 2 sáng tắt xen kẽ nhau, (đèn 1,3,5,7 sáng, đèn 2,4,6,8 tắt , sau đó đèn 1,3,5,7 tắt, đèn 2,4,6,8 sáng. Lặp lại quá trình trên.). Biết led sáng khi tín hiệu xuất ở các chân Port 2 ở mức 1. Minh họa trong hình phía dưới

5. Hãy viết chương trình điều khiển 8 led của port 0 sáng dồn theo chiều ngược lại.
6. Hãy viết chương trình điều khiển 16 led của 2 port: port0 và port1 sáng dồn. [xem chương trình đã viết bên dưới và thực hiện các chương trình còn lại.
7. Hãy viết chương trình điều khiển 3 port: port0, port1, port2 sáng dồn.
8. Hãy viết chương trình điều khiển 4 port: port0, port1, port2 và port3 sáng dồn.
9. Hãy viết chương trình sáng dồn 2 port 0 và 1 từ ngoài vào trong và từ trong ra ngoài.
10. Hãy viết chương trình sáng dồn 4 port 0, 1, 2 và 3 từ ngoài vào trong và từ trong ra ngoài.
11. Phần cứng: 8 led nối với Port 2, được định vị trí như sau: led 1 nối với P2.0, lần lượt cho đến led 8 (nối với P2.7). Biết led sang khi tín hiệu xuất ở các chân Port 1 ở giá trị 1.

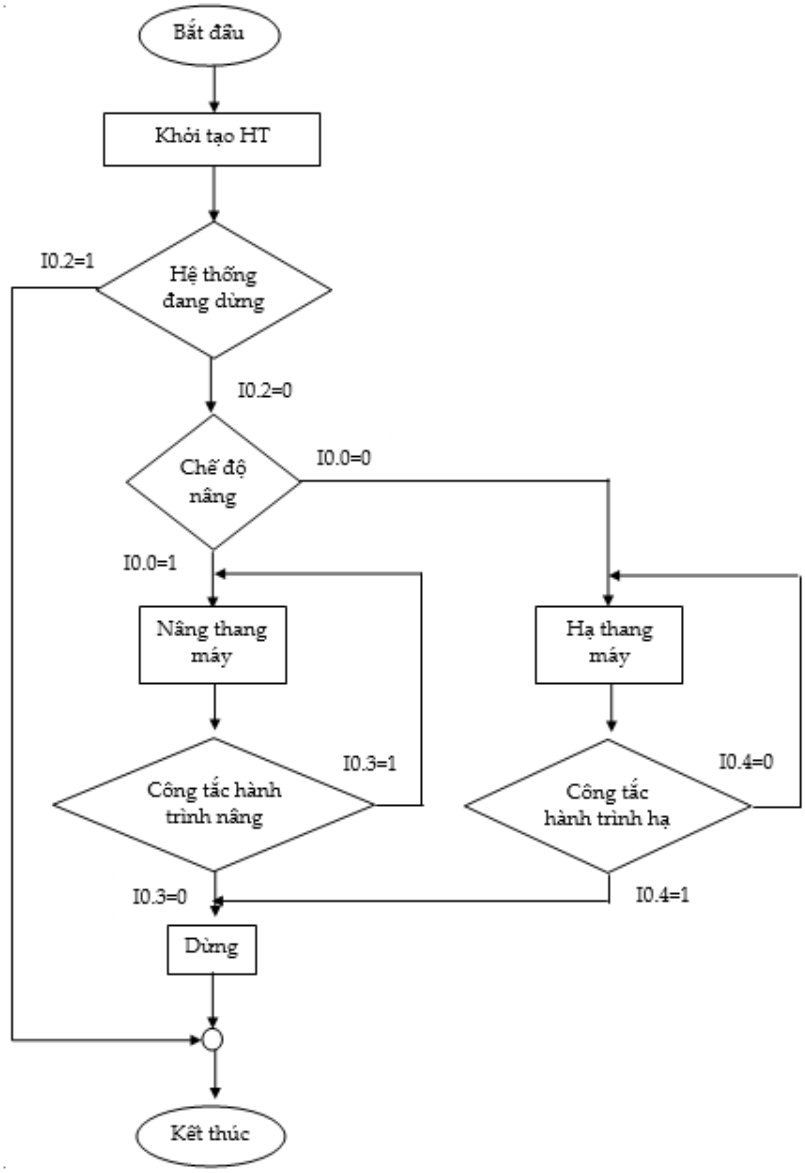
Viết chương trình để các led nối với Port 1 sáng theo kiểu tăng nhị phân. Các quá trình trên được lặp lại không giới hạn.

12. Phần cứng: 8 led nối với Port 0 được định vị trí như sau: led 1 nối với P0.0, lần lượt cho đến led 8 (nối với P0.7). Biết led sáng khi tín hiệu xuất ở mức 1.Viết chương trình để các led nối với Port 0 sáng dần từ led 1 đến led 8 sau đó tắt hết led và lặp lại. Các quá trình được lặp lại không giới hạn.

13. Phần cứng: 8 led nối với Port 0 được định vị trí như sau: led 1 nối với P0.0, lần lượt cho đến led 8 (nối với P0.7). Biết led sáng khi tín hiệu xuất ở mức 1.Viết chương trình để các led nối với Port 0 sáng dồn từ led 1 đến led 8 sau đó tắt hết led và lặp lại. Các quá trình được lặp lại không giới hạn.

14. Viết chương trình điều khiển tổng hợp với yêu cầu sau: 32 led chớp tắt 10 lần, 32 led tắt dần từ trái sang phải và ngược lại, 32 led sáng dần lên từ trái sang phải, một điểm led sáng chạy, chương trình quay lại từ đầu.

15. Kiểm tra và bổ sung lưu đồ thuật toán điều khiển Mô hình thang máy xây dựng sau: (Tự động hóa các QTSX)



BÀI 3: TIMER/COUNTER

Giới thiệu:

Bộ định thời/bộ đếm (Timer/Counter) là một trong những ngoại vi thông dụng mà bất cứ dòng vi điều khiển nào cũng có. Khi học tập, nghiên cứu về bất kì một dòng vi điều khiển nào đó, thì Timer/Counter là một trong những ngoại vi quan trọng mà người học chắc chắn phải tìm hiểu. Bởi vì Timer/Counter được sử dụng rất nhiều trong các ứng dụng như quét led, tạo xung pwm, đo tần số, định thời…

Trong vi điều khiển 8051 có 2 bộ Timer/Counter (T/C) có chức năng đếm xung.

Chế độ timer: Đếm xung nội lấy từ mạch dao động bên trong vi điều khiển có chu kỳ ổn định. Ứng dụng: thường dùng để xác định thời gian chính xác để điều khiển các thiết bị theo thời gian.

Chế độ counter: Đếm xung nhận từ bên ngoài đưa đến ngõ vào T0 với T/C0 và ngõ T1 đối với T/C1.

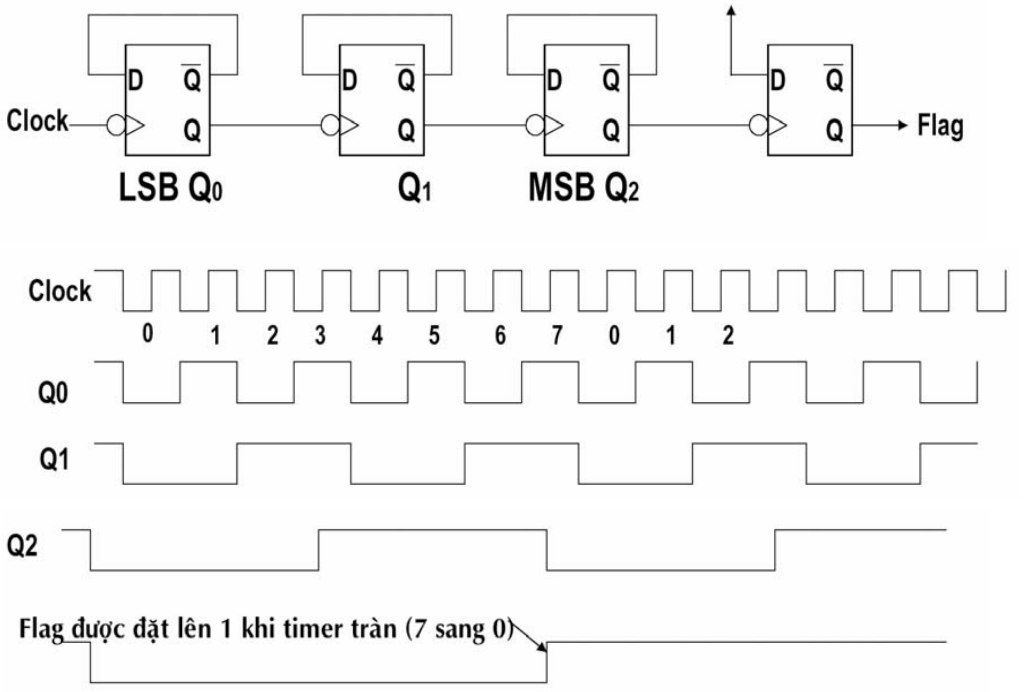
Mục tiêu của bài:

- Trình bày được nguyên lý hoạt động .
- Viết chương trình cho vi mạch xử lý đếm timer/counter dùng 89C51.
- Rèn luyện tính cẩn thận chính xác, nghiêm túc trong học tập và trong thực hiện công việc.

Nội dung chính:

1. Timer

1.1. Bộ định thì (Timer)



Hình 3.1.1: Timer 3 bit và giản đồ định thì

Tại mỗi thời điểm, ta chỉ sử dụng được 1 trong 2 hoặc là timer hoặc là counter.

Số lượng xung mà timer/counter có thể đếm được nhiều nhất từ 0 đến 65535 đếm theo giá trị thập phân, 0000H đến FFFFH đếm theo số thập lục phân.

Khi đến giá trị cực đại FFFFH và nếu có thêm 1 xung nữa thì bộ đếm sẽ bị tràn và giá trị đếm sẽ tự động nhảy về 0000H.

1.2. Các thanh ghi trong timer/counter

1.2.1. Thanh ghi TH0, TL0, TH1, TL1.

Dùng để lưu trữ các giá trị đếm được của bộ timer/counter.

TH0, TL0: lưu giá trị timer/counter 0.

TH1, TL1: lưu giá trị timer/counter 1.

Độ rộng mỗi thanh ghi là 8 bit.

1.2.2. Thanh ghi TMOD:

Thiết lập chế độ hoạt động cho chân T0 và T1.

Gồm 2 nhóm 4 bit:

4 bit thấp: thiết lập chế độ hoạt động cho T0.

4 bit cao: thiết lập chế độ hoạt động cho T1.

Timer SFR	Mục đích	Địa chỉ	Địa chỉ bit
TCON	Điều khiển	88H	Có
TMOD	Chế độ (hoạt động)	89H	Không
TL0	Byte thấp của Timer 0	8AH	Không
TL1	Byte thấp của Timer 1	8BH	Không
TH0	Byte cao của Timer 0	8CH	Không
TH1	Byte cao của Timer 1	8DH	Không
T2CON*	Điều khiển Timer 2	C8H	Có
RCAP2L*	Bắt byte thấp của Timer 2	CAH	Không
RCAP2H*	Bắt byte cao của Timer 2	CBH	Không
TL2*	Byte thấp của Timer 2	CCH	Không
TH2*	Byte cao của Timer 2	CDH	Không

* Với 8032/8052.

Bảng 3.1: Bảng các thanh ghi trong 8951

Trong đó, 2 bit M0 và M1 tạo ra 4 trạng thái tương ứng với 4 kiểu làm việc khác nhau của T0 và T1.

M1	M0	Kiểu	Chức năng
0	0	0	Mode Timer 13 bit (mode 8048)
0	1	1	Mode Timer 16 bit
1	0	2	Mode tự động nạp 8 bit
1	1	3	Mode tách Timer ra : *Timer0* : được tách ra làm 2 timer 8 bit gồm có: Timer 8 bit TL0 được điều khiển bởi các bit của T0. Timer 8 bit TH0 được điều khiển bởi các bit của T1. *Timer1* : không được hoạt động ở mode 3.

Bảng 1.2: Các trạng thái làm việc của T0 và T1

MODE 0: Mode timer 13 bit

- Sử dụng 8 bit cao của thanh ghi TH0 (TH1) và 5 bit thấp của thanh ghi TL0 (TL1) để lưu trữ các giá trị đếm của bộ timer/counter.
- Như vậy ta có thể đếm từ 0 đến 2^{13} = 8192 giá trị xung ở chế độ MODE 0.

MODE 1: Mode timer 16 bit

- Sử dụng 2 thanh ghi 8bit TH0 (TH1) và TL0 (TL1) để lưu trữ các giá trị đếm.
- Ở chế độ này, ta có thể đếm từ 0 đến 2^{16} = 65536 giá trị.

MODE 2: Mode timer tự động nạp 8 bit

- 8 bit của thanh ghi TL0 (TL1) dùng để đếm giá tri của bộ timer/counter.
- 8 bit của thanh ghi TH0 (TH1) dùng để lưu giá trị nạp lại vào thanh ghi TL0 (TL1).

Ví dụ:

- Khi thanh ghi TL0 = FFh.
- Thanh ghi TH0 = 89h.
- Khi đếm tiếp 1 lần giá trị nữa thì TL0 sẽ đếm bắt đầu từ 89h thay vì đếm lại từ 00h.

MODE 3: Mode tách timer

- Chỉ làm việc ở Timer 0.
- 2 thanh ghi 8 bit TL0 và TH0 hoạt động riêng lẻ và quay trở về giá trị 00h khi bị tràn.
- Sử dụng TF0 và TF1 làm các bit cờ tràn tương ứng.

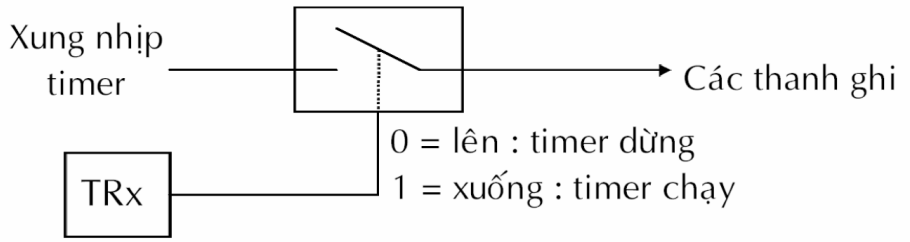
1.2.3. Thanh ghi TCON

Chứa các bit trạng thái và điều khiển cho T0 và T1.

Gồm 8 bit có chức năng như hình sau:

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Hình 1.2.2: 8 bit của thanh ghi TCON



Hình 1.2.3: Cho chạy và dừng các timer

- Bit TF1 (TF0): cờ tràn Timer 1 (0).

TF1 = 1 khi bộ đếm timer 1 (0) bị tràn.

- Bit TR1 (TR0): điều khiển Timer 1 (0) đếm hoặc ngừng.

TR1 (TR0) = 1: cho phép Timer 1 (0) đếm xung.

TR1 (TR0) = 0: cho phép Timer 1 (0) ngừng đếm xung.

- Bit IE1 (IE0): Cờ báo ngắt INT1 (INT0).

IE1 (IE0) = 1 khi có xảy ra ngắt xảy ra ở ngõ vào INT1 (INT0).

- Bit IT1 (IT0): Bit điều khiển cho phép ngắt IT1 (IT0) tác động bằng mức hay cạnh.

IT1 (IT0) = 1 ngắt tác động bằng cạnh xuống.

IT1 (IT0) = 0 ngắt tác động bằng mức.

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.

2. Counter.

Nguyên tắc hoạt độn của bộ timer/counter ở chế độ timer như sau:

- Khi bit TR của bộ T/C được set, bộ T/C sẽ được cho phép hoạt động, và cứ mỗi xung nhịp clock thì giá trị của thanh ghi chứa giá trị TH0, TL0 tăng lên 1(bắt đầu từ giá trị được cài đặt, mặc định là 0). Phụ thuộc vào mode hoạt động (8bit, 13bit, hay 16bit) mà giới hạn tràn của thanh ghi chứa giá trị là khác nhau.

- Với mode 8bit, chỉ có thanh ghi TL được sử dụng để định thời, thanh ghi TH được sử dụng để ghi giá trị khởi tạo, và mỗi khi thanh ghi TL bị tràn (vượt quá 255) thì giá trị trong thanh ghi TH được tự động load vào thanh ghi TL. Vậy nên chế đọ này được gọi là 8 bit tự nạp lại.

- Với mode 16 bit, thì cả 2 thanh ghi TL và TH đều sử dụng chứa giá trị của T/C. Và khi T/C tràn (giá trị TH,TL vượt quá 65535) thì sẽ được reset về 0 và đồng thời cờ báo tràn TF được bật. Vậy nên sau khi tràn, phải khởi tạo lại giá trị cho TH,TL để có được thời gian định thì mong muốn.

- Mode 13 bit tương tự mode 16 bit.

- Các bước cấu hình cho bộ T/C0 hoạt động cở chế độ timer (định thời) như sau.

+ Ghi vào thanh ghi TMOD để chọn chế độ timer, chọn mode: Ví dụ ở đây mình chọn chế độ timer (C/T = 0) và mode 16 bit (M1=0, M0=1), mình sẽ ghi giá trị 0x01 vào TMOD (TMOD = 0x01).

+ Ghi vào thanh ghi TL0, TH0 các giá trị để tạo ra giá trị định thời mong muốn. Ví dụ mình muốn định thời 1ms (với thạch anh 12MHz), mình sẽ ghi vào thanh ghi giá TH, TL giá trị (65536 - 1000 =64536). Vì với thạch anh 12MHz, xung nhịp của 8051 là 1/12 của tần số thạch anh là 1MHz. Tương ứng với một xung nhịp là 1us, do vậy để có thời gian 1ms = 1000us, mình cần cho bộ T/C đếm 1000 lần. Mình gán giá trị 64536 bởi vì sau 1000 lần tăng giá trị thì T/C sẽ tràn và khi đó cờ TF sẽ bật lên 1. Chỉ cần kiểm tra cờ TF là mình biết đã đủ 1ms hay chưa.

+ Set thanh ghi TR để cho phép bộ T/C hoạt động.

+ Kiểm tra cờ TF để biết bộ T/C đã tràn hay chưa.

+ Khởi tạo lại giá trị cho các thanh ghi TH, TL để chuẩn bị cho lần chạy tiếp theo.

Ta sắp xếp bộ nhớ và địa chỉ của các thiết bị ngoại vi giao tiếp với KIT trong bảng như sau:

Địa chỉ	*Thiết bị*	*Ghi chú*
0000h - 1FFFh	**EEPROM**	Chứa chương trình Monitor
4000h - 7FFFh	**RAM**	Chứa chương trình ứng dụng
8000h - 8003h	**8255** (1)	***8000h*** - PortA : Điều khiển LCD. ***8001h*** - PortB : Data bus của LCD. ***8002h*** - PortC : Nhận mã của bàn phím. ***8003h*** - CW : Thanh ghi điều khiển.
8004h - 8007h	**8255** (5)	***8004h*** - PortA: Điều khiển cấp nguồn cho các Anot chung của LED matrix (các hàng). ***8005h*** - PortB: Điều khiển đèn màu xanh của LED matrix (các cột xanh). ***8006h*** - PortC: Điều khiển đèn màu đỏ của LED matrix (các cột đỏ). ***8007h*** - CW: Thanh ghi điều khiển.
8008h - 800Bh	**8255** (2)	***8008h*** - PortA : Đường Data của ADC0809. ***8009h*** - PortB : Đường Data của DAC0808. ***800Ah*** - PortC : Điều khiển ADC0809. ***800Bh*** - CW : Thanh ghi điều khiển.
800Ch - 800Fh	**8255** (3)	***800Ch*** - PortA : Đầu vào Digital. ***800Dh*** - PortB : Điều khiển động cơ bước(***PB0-PB3***) và động cơ một chiều (***PB4 - PB5***). ***800Eh*** - PortC : 4 đầu vào xung dạng nút ấn (***PC4-PC7***), 4 đầu vào cho các thiết bị ngoài như encoder (***PC0 - PC3***). ***800Fh*** - CW : Thanh ghi điều khiển.
C000h - C003h	**8255** (4)	***C000h*** - PortA : Hiển thị 8 LED đơn. ***C001h*** - PortB : Hiển thị 2 LED 7 thanh (trái). ***C002h*** - PortC : Hiển thị 2 LED 7 thanh (phải). ***C003h*** - CW : Thanh ghi điều khiển.
C004h – C007h	**8255-EX**	BUS mở rộng dự trữ, có thể gắn thêm 1 module khác

Bảng 3.3: Giải mã địa chỉ của các thiết bị ngoại vi

Do yêu cầu thiết kế của bộ KIT, các chip nhớ RAM và EEPROM vừa phải có khả năng làm bộ nhớ dữ liệu và bộ nhớ chương trình nên các chân OE (Output Enable) của các chip nhớ này có logic như sau /OE = /PSEN * /RD.

Để rõ hơn ta tìm hiểu kỹ về chức năng của chân /PSEN và EA trong ứng dụng mở rộng bộ nhớ ngoài của họ 8051. PSEN (Program Store Enable) có nghĩa là cho phép cất chương trình. Đây là tín hiệu ra và được nối với chân OE của bộ nhớ chương trình ngoài. Khi chân EA được nối đất thì 8031/51 nạp mM lệnh từ bộ nhớ ngoài thông qua chân PSEN, ở đây bộ nhớ ngoài đóng vai trò là bộ nhớ chương trình.

Ngoài ra khi bộ nhớ ngoài làm chức năng là bộ nhớ dữ liệu thì tín hiệu RD được sử dụng để truy cập không gian dữ liệu ngoài (dùng lệnh MOVX), nên RD được nối đến OE của chip nhớ. Trong thiết kế phần mềm của KIT đòi hỏi cả 2 chức năng này của bộ nhớ, vừa là bộ nhớ chương trình, vừa là bộ nhớ dữ liệu do đó các chân OE của các chip nhớ phải có mức logic như trên /OE = /PSEN * /RD.

Dựa vào bảng trên ta có sơ đồ bộ nhớ cụ thể như bảng sau:

	Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EEPROM AT28C64 - 8Kbyte	0000h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1FFFh	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM HM62256 - 32Kbyte	4000h	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7FFFh	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8255-1 KEYPAD LCD	8000h	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8001h	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	8002h	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	8003h	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
8255-5 MATRIX LED	8004h	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	8005h	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
	8006h	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
	8007h	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
8255-2 ADC DAC	8008h	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	8009h	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
	800Ah	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
	800Bh	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
8255-3 XUNG SỐ STEPPER DC MOTOR	800Ch	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
	800Dh	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
	800Eh	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
	800Fh	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
8255-4 LED 7 ĐOẠN LED ĐƠN	C000h	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	C001h	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	C002h	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	C003h	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
8255-EX	C004h	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	C005h	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1
	C006h	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0
	C007h	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Nhìn vào các cột có màu xám trong bảng trên ta có được cách phân công giải mã như sau:

Ta sử dụng 3 chip giải mã chuyên dụng là vi mạch 74HC138 để phục vụ việc giải mã địa chỉ cho các chip nhớ cũng như các chip mở rộng I/O là 8255, đầu vào và ra của các chip giải mã được trình bày trong bảng sau:

Chip giải mã	A	B	C	G2A	G2B
74HC138 - 1 (U9)	A14	A15	0	0	0
74HC138 - 2 (U10)	A2	A3	A4	Y2(U9)	Y2(U9)
74HC138 - 3 (U11)	A2	0	0	Y3(U9)	Y3(U9)

Đầu vào chọn chip CS (Chip Select) của các chip EEPROM, RAM, và 8255 như bảng sau:

Chip	CS (Chip Select)	Ghi chú
EEPROM - 28C64	A13 + Y0(U9) (*)	U9,U10,U11 xem sơ đồ nguyên lý.
RAM - 62256	Y1(U9)	
8255(1) - Keypad, LCD	Y0(U10)	
8255(5) - LED ma trận	Y2(U10)	
8255(2) - ADC, DAC	Y3(U10)	
8255(3) - Xung số, stepper, DC motor	Y1(U10)	
8255(4) - LED 7 đoạn, LED đơn	Y0(U11)	
8255(EX)	Y1(U11)	

Ví dụ: Hiện thị chữ chạy trên LCD

```
ORG 0000h
MOV A,#38H ;
ACALL CSTROBE
MOV A,#0EH
ACALL CSTROBE
MOV A,#01H ;clear LCD
ACALL CSTROBE
MOV A,#06H
ACALL CSTROBE
MOV A,#80H
ACALL CSTROBE
MOV DPTR,#MYDATA
DONG1: CLR A
MOVC A,@A+DPTR
JZ THOAT
ACALL DSTROBE
INC DPTR
SJMP DONG1
THOAT: MOV A,#18H
ACALL CSTROBE
ACALL DELAY100MS
SJMP THOAT
HERE: SJMP HERE
CSTROBE:
ACALL READY
MOV P1,A
CLR P3.0 ;RS=0: le^.nh
CLR P3.1
SETB P3.2
```

```
CLR P3.2
DSTROBE: ;data strobe
ACALL READY ;
MOV P1,A ;
SETB P3.0
CLR P3.1
SETB P3.2 ;
CLR P3.2 ;E=0, cho^'t
RET
READY: SETB P1.7
CLR P3.0 ;
SETB P3.1
BACK: CLR P3.2
SETB P3.2 ;E=1
JB P1.7,BACK ;cho
RET
DELAY100ms: MOV R2,#100
DL5: MOV R1,#250
DL4: NOP
NOP
DJNZ R1,DL4
DJNZ R2,DL5
RET
ORG 250H
MYDATA:
DB "DAI HOC BACH KHOA HA NOI", 0
END
```

Ví dụ 2: CHƯƠNG TRÌNH DELAY SỬ DỤNG TIMER

Mục đích yêu cầu: biết cách tính toán các thông số delay của timer để viết các chương trình delay chính xác.

Trình tự thực hiện:

1. Kết nối mạch theo trình tự:
- Dùng bus dây kết nối port 1 với một trong bốn PINHD của dãy 32 led.
- Gắn vi điều khiển vào đế nạp 40 pin (socket) ở modul nạp của hệ thống 2 .
2. Khởi động phần mềm, mở File mới và đặt tên file. Vd: bai1_6
3. Viết chương trình với tên file vừa đặt:

CHƯƠNG TRÌNH

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh sang tat port1 su dung timer lam bo dinh thoi delay 65536 micro giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
        org 0000h
        b61: mov p1,#00h
        lcall delay ; delay 65536 micro giay
        mov p1,#0ffh
        lcall delay
        sjmp b61
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con delay 65535 micro giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        delay: clr tf0 ; xoa co ngat cua timer 0
        mov tl0,#0 ; nap 0 vao TL0
        mov th0,#0 ; nap 0 vao TH0
        mov tmod,#01 ; khoi tao timer T0 mode 1 dem 16 bit
        setb tr0 ; cho phep timer 0 bat dau dem xung
        del1 : jnb tf0,del1 ; kiem tra co tran
        ret
        end
```

Giải thích: bài sáng tắt port1 trên giống như bài đã làm trước đây chỉ khác là thay chương trình delay bằng một chương trình sử dụng timer để việc tính toán thời gian dễ dàng hơn. Với chương trình trên thì timer T0 sẽ đếm từ gía trị nạp ban đầu 0000H đến 100000H [tức cờ tràn bằng 1 – các số về 0000H -> kết thúc 1 chu kỳ đếm] kết quả số xung đếm được là 10000H – 0000H = 10000 (65536) xung và mỗi xung có chu kì 1 micro giây nên lượng thời gian mà timer T0 đếm được là 65536 micro giây.

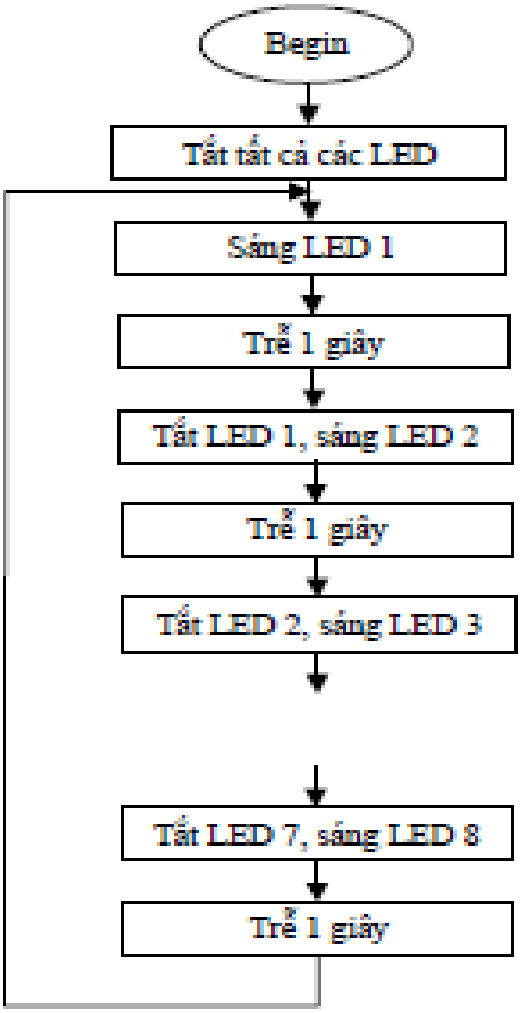
4. Thực hiện các bước giống như trên và xem kết quả.

CÂU HỎI ÔN TẬP

1. Chức năng của timer/ counter trong vi điều khiển?
2. Chức năng của thanh TCON và TMOD? Cho ví dụ minh họa?
3. Ứng dụng của timer/ counter trong lập trình vi điều khiển? Cho ví dụ minh họa?
4. Viết một lưu đồ sử dụng timer/counter trong sản xuất?

BÀI TẬP

1. Cho sáng lần lượt từng led mỗi led 1 giây



a. Hãy viết chương trình sáng tắt port 2 sử dụng timer làm bộ định thời delay 5 giây.

b. Hãy viết chương trình giống trên nhưng delay 10 giây.

c. Tương tự hãy viết chương trình delay 1 giờ.

2. Viết chương trình chạy chữ Hà – Nội từ phải sang trái trên 8 led

a. Kết nối mạch theo trình tự:

Dùng bus dây kết nối port 0 đến pinhd điều khiển các đoạn a,b,c,d,e,f,g,dp và kết nối port 2 đến pinhd điều khiển quét hàng.

Gắn vi điều khiển vào đế nạp 40 pin (socket) ở modul nạp của hệ thống 2.

b. Khởi động phần mềm, mở File mới và đặt tên file.

c. Viết chương trình với tên file vừa đặt

3. Kiểm tra lỗi trong chương trình kiểm tra phím 4x4

```
TESTPHIM:
KT_4PHIM_COT1:
MOV A,#0FFH
MOV P1,A
CLR P1.4 ; noi dat cot 1
MOV A,P1
ANL A,#0FH ; kiem tra phim nhan
```

```
CJNE A,#0FH,CO_PHIM_NHAN
KT_4PHIM_COT2:
MOV A,#0FFH
MOV P1,A
CLR P1.5 ; noi dat cot 2
MOV A,P1
ANL A,#0FH
CJNE A,#0FH,CO_PHIM_NHAN
KT_4PHIM_COT3:
MOV A,#0FFH
MOV P1,A
CLR P1.6 ; noi dat cot 3
MOV A,P1
ANL A,#0FH
CJNE A,#0FH,CO_PHIM_NHAN
KT_4PHIM_COT4:
MOV A,#0FFH
MOV P1,A
CLR P1.7 ; noi dat cot 4
MOV A,P1
ANL A,#0FH
CJNE A,#0FH,CO_PHIM_NHAN
LJMP KET_THUC ; khong co phim nao
CO_PHIM_NHAN:
MOV A,P1
P0_C1: ; hang 1 cot 1
CJNE A,#11101110B,P1_C1 ;
MOV A,#0 ;co phim C nhan
MOV 50H,A ;
LJMP KET_THUC
P1_C1: ;hang 2 cot 1
CJNE A,#11101101B,P2_C1 ;
MOV A,#1
MOV 50H,A
LJMP KET_THUC
P2_C1: ;hang 3 cot 1
CJNE A,#11101011B,P3_C1 ;
MOV A,#2
MOV 50H,A
LJMP KET_THUC
P3_C1: ;hang 4 cot 1
```

```
CJNE A,#11100111B,P0_C2 ;
MOV A,#3
MOV 50H,A
LJMP KET_THUC
P0_C2: ;hang 1 cot 2
CJNE A,#11011110B,P1_C2
MOV A,#4
MOV 50H,A
LJMP KET_THUC
P1_C2: ;hang 2 cot 2
CJNE A,#11011101B,P2_C2 ;
MOV A,#5
MOV 50H,A
LJMP KET_THUC
P2_C2: ;hang 3 cot 2
CJNE A,#11011011B,P3_C2 ;
MOV A,#6
MOV 50H,A
LJMP KET_THUC
P3_C2: ;hang 4 cot 2
CJNE A,#11010111B,P0_C3
MOV A,#7
MOV 50H,A
LJMP KET_THUC
P0_C3: ;hang 1 cot 3
CJNE A,#10111110B,P1_C3
MOV A,#8
MOV 50H,A
LJMP KET_THUC
P1_C3: ;hang 2 cot 3
CJNE A,#10111101B,P2_C3
MOV A,#9
MOV 50H,A
LJMP KET_THUC
P2_C3: ;hang 3 cot 3
CJNE A,#10111011B,P3_C3
MOV A,#10
MOV 50H,A
LJMP KET_THUC
P3_C3: ;hang 4 cot 3
CJNE A,#10110111B,P0_C4 ;
```

```
MOV A,#11
MOV 50H,A
LJMP KET_THUC
P0_C4: ;hang 1 cot 4
CJNE A,#01111110B,P1_C4
MOV A,#12
MOV 50H,A
LJMP KET_THUC
P1_C4: ;hang 2 cot 4
CJNE A,#01111101B,P2_C4
MOV A,#13
MOV 50H,A
LJMP KET_THUC
P2_C4: ;hang 3 cot 4
CJNE A,#01111011B,P3_C4 ;
MOV A,#14
MOV 50H,A
LJMP KET_THUC
P3_C4: ;hang 4 cot 4
CJNE A,#01110111B,KET_THUC
MOV A,#15
MOV 50H,A
LJMP KET_THUC
KET_THUC:
RET
```

4. Viết chương trình hiển thị chữ “TCN CỦ CHI” chạy từ trái sang phải và lập lại một các tuần tự hiển thị trên led 7 đoạn.

5. Viết chương trình 4 động cơ hoạt động tuần tự như sau

a. Động cơ 1 chạy 5 giây sau đó dừng, động cơ 2 chạy 20 rồi dừng, động cơ 3 chạy 15 giây rồi dừng

b. Cả 3 động cơ chạy 10 giây rồi dừng

c. Chương trình thực hiện một cách tuần tự

d. Chương trình lập lại 5 lần rồi dừng hẳn

BÀI 4: NGẮT VÀ CHƯƠNG TRÌNH PHỤC VỤ NGẮT

Giới thiệu:

Ngắt (Interrupt) là một số sự kiện khẩn cấp bên trong hoặc bên ngoài bộ vi điều khiển xảy ra, buộc vi điều khiển tạm dừng thực hiện chương trình hiện tại, phục vụ ngay lập tức nhiệm vụ mà ngắt yêu cầu – nhiệm vụ này gọi là trình phục vụ ngắt (ISR: Interrupt Service Routine).

Trong bài này tìm hiểu khái niệm ngắt và lập trình các ngắt trong bộ vi điều khiển 8051.

Mục tiêu của bài:

- Trình bày nguên lý hoạt động.
- Viết chương trình phục vụ ngắt trên kit 89C51.
- Trình bày cấu tạo vi mạch vi điều khiển chuyên dùng 89C51.
- Phát huy tính chủ động trong học tập và trong công việc.

Nội dung chính:

1. Trao đổi dữ liệu bằng ngắt.

1.1. Tổ chức ngắt.

8951 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 8951 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 8952, 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.

(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 đổiđược).

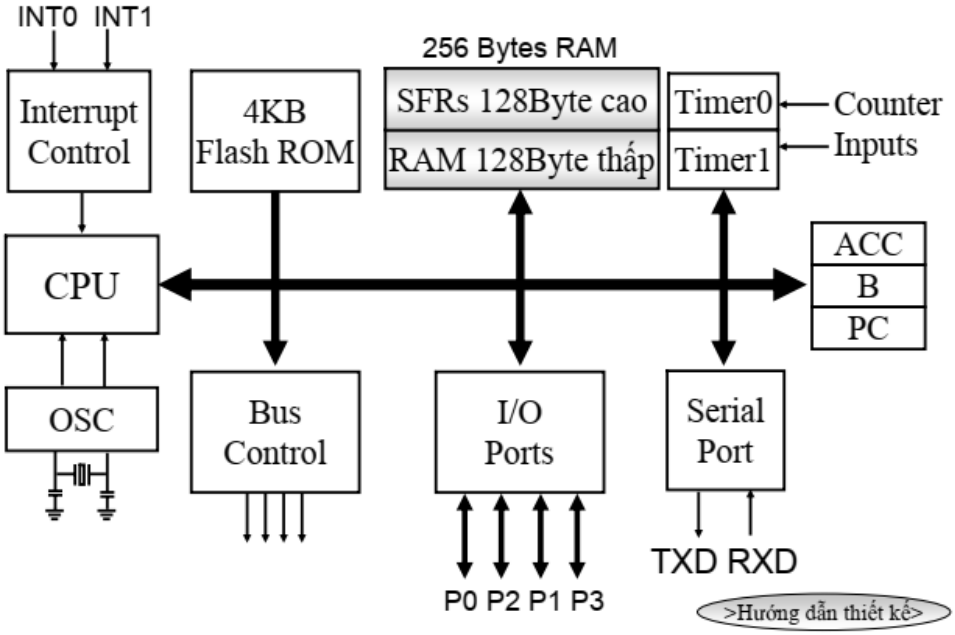
	Source	Priority Within Level
1	IE0	(highest)
2	TF0	
3	IE1	
4	TF1	
5	RI + TI	
6	TF2 + EXF2	(lowest)

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
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.



Hình 4.1.1: Tổ chức bên trong 89C51

*** Phân biệt cơ chế ngắt với hỏi vòng**

Ví dụ bộ vi điều khiển đóng vai trò như một vị bác sĩ, các thiết bị kiểm soát bởi vi điều khiển được coi như các bệnh nhân cần được bác sĩ phục vụ.

Bình thường, vị bác sĩ sẽ hỏi thăm lần lượt từng bệnh nhân, đến lượt bệnh nhân nào được hỏi thăm nếu có bệnh thì sẽ được bác sĩ phục vụ, xong lại đến lượt bệnh nhân khác, và tiếp tục đến hết. Điều này tương đương với phương pháp thăm dò - hỏi vòng(Polling) trong vi điều khiển.

Cứ như thế, nếu chúng ta có 10 bệnh nhân, thì bệnh nhân thứ 10 dù muốn hay không cũng phải xếp hàng chờ đợi 09 bệnh nhân trước đó. Giả sử trường hợp bệnh nhân thứ 10 cần cấp cứu thì sao? Anh ta sẽ gặp nguy cấp trước khi đến lượt hỏi thăm của bác sĩ mất! L Nhưng, nếu anh ta sử dụng phương pháp "ngắt" thì mọi chuyện sẽ ổn ngay. Lúc đó vị bác sĩ sẽ ngừng mọi công việc hiện tại của mình, và tiến hành phục vụ trường hợp khẩn cấp này ngay lập tức, xong việc bác sĩ lại trở về tiếp tục công việc đang dở. Điều này tương đương với phương pháp **ngắt** (**Interrupts**) trong vi điều khiển.

Trở lại với bộ vi điều khiển của chúng ta: 1 bộ vi điều khiển có thể phục vụ cho nhiều thiết bị, có 2 cách để thực hiện điều này đó là sử dụng các ngắt (Interrupts) và thăm dò (polling).

1.2. Qui trình xử lý yêu cầu ngắt.

Mỗi khi có một thiết bị bất kỳ cần được phục vụ thì nó báo cho bộ vi điều khiển bằng cách gửi một tín hiệu ngắt. Khi nhận được tín hiệu ngắt thì bộ vi điều khiển ngừng tất cả những gì nó đang thực hiện để chuyển sang phục vụ thiết bị gọi ngắt. Chương trình

ngắt được gọi là trình phục vụ ngắt ISR(Interrupt Service Routine) hay còn gọi là trình quản lý ngắt (Interrupt handler). Sau khi phục vụ ngắt xong, bộ vi xử lý lại quay trở lại điểm bị ngắt trước đó và tiếp tục thực hiện công việc.

Trong phương pháp thăm dò:

Bộ vi điều khiển kiểm tra liên tục tình trạng của tất cả các thiết bị, nếu thiết bị nào có yêu cầu thì nó dừng lại phục vụ thiết bị đó. Sau đó nó tiếp tục kiểm tra tình trạng của thiết bị kế tiếp cho đến hết. Phương pháp thăm dò rất đơn giản, nhưng nó lại rất lãng phí thời gian để kiểm tra các thiết bị kể cả khi thiết bị đó không cần phục vụ. Trong trường hợp có quá nhiều thiết bị thì phương án thăm dò tỏ ra không hiệu quả, gây ra chậm trễ cho các thiết bị cần phục vụ.

*** Điểm mạnh của phương pháp ngắt là:**

Bộ vi điều khiển có thể phục vụ được rất nhiều thiết bị (tất nhiên là không tại cùng một thời điểm). Mỗi thiết bị có thể nhận được sự chú ý của bộ vi điều khiển dựa trênmức ưu tiên được gán cho nó. Đối với phương pháp thăm dò thì không thể gán mức ưu tiên cho các thiết bị vì nó kiểm tra tất cả mọi thiết bị theo kiểu hỏi vòng.

Quan trọng hơn, trong phương pháp ngắt thì bộ vi điều khiển còn có thể che (làm lơ) một yêu cầu phục vụ của thiết bị. Điều này lại một lần nữa không thể thực hiện được trong phương pháp thăm dò.

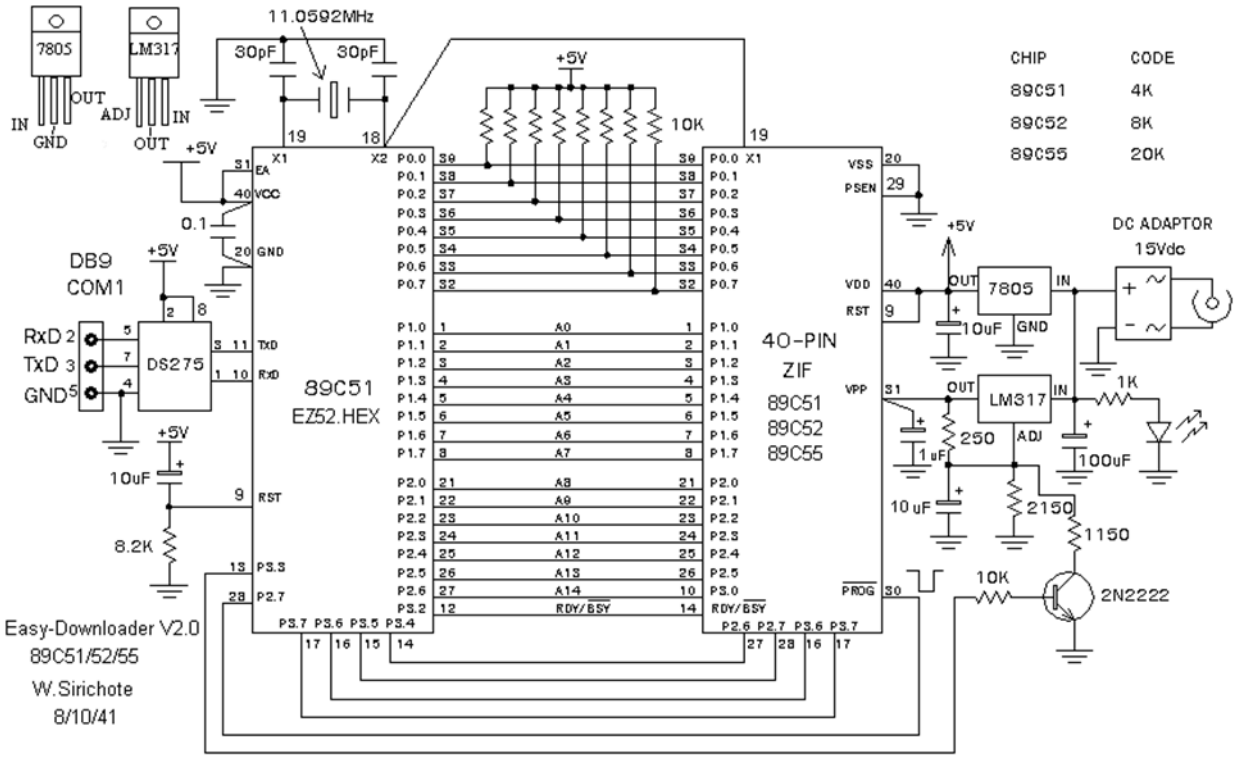
Lý do quan trọng nhất mà phương pháp ngắt được ưu chuộng là vì nó không lãng phí thời gian cho các thiết bị không cần phục vụ. Còn phương pháp thăm dò làm lãng phí thời gian của bộ vi điều khiển bằng cách hỏi dò từng thiết bị kể cả khi chúng không cần phục vụ.

Ví dụ trong các bộ định thời được bàn đến ở các bài trước ta đã dùng một vòng lặp kiểm tra và đợi cho đến khi bộ định thời quay trở về 0. Trong ví dụ đó, nếu sử dụng ngắt thì ta không cần bận tâm đến việc kiểm tra cờ bộ định thời, do vậy không lãng phí thời gian để chờ đợi, trong khi đó ta có thể làm việc khác có ích hơn.

2. Vi mạch xử lý ngắt 89C51.

2.1. Tóm tắt đặc tính 89C51.

Vi điều khiển AT89C52 Với thực tế thị trường của nước ta nói chung ta chọn AT89C52 là chip dễ dàng mua được, bộ nhớ 8Kbyte vừa đủ cho chương trình MONITOR điều hành KIT. AT89C52 có 8K Flash ROM làm bộ nhớ chương trình, 256 byte RAM, 32 đường xuất nhập, 3 bộ định thời, một cấu trúc ngắt 2 mức ưu tiên và 8 nguồn ngắt, một port nối tiếp song công (full duplex). Timer T2 của 89C52 có thể làm việc như Timer T0, T1 trong chế độ Reload ngay cả ở lúc làm Timer 16 bit. Vùng nhớ Flash ROM có thể nạp và xóa khoảng 1000 lần. Vi điều khiển AT89C52 hỗ trợ tần số làm việc đến 24 MHz. Có chế độ Power Down để tiết kiệm điện năng của hệ thống tuy nhiên vẫn duy trì nội dung RAM nhưng không cho mạch dao động cấp xung clock nhằm vô hiệu hóa các hoạt động khác cho chip cho đến khi có reset cứng tiếp theo. Chế độ Idle hay còn gọi là chế độ nghỉ dừng CPU trong khi vẫn cho phép RAM, các bộ định thời/ đếm, port nối tiếp và hệ thống ngắt tiếp tục hoạt động.



Hình 4.1.2: Sơ đồ chức năng 89C51

2.2. Sơ đồ khối.

Vi điều khiển 8051 Là IC đóng vỏ dạng DIP có 40 chân, mỗi chân có một kí hiệu tên và có các chức năng như sau:

Chân 40: nối với nguồn nuôI +5V.

Chân 20: nối với đất(Mass, GND).

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

Chân 31 (EA : Eternal Acess) được đưa xuống thấp cho phép chọn bộ nhớ mã ngoàI đối với 8051. Đối với 8051 thì : EA = 5V : Chọn ROM nội. EA = 0V : Chọn ROM ngoại.

32 chân còn lại chia làm 4 cổng vào ra:

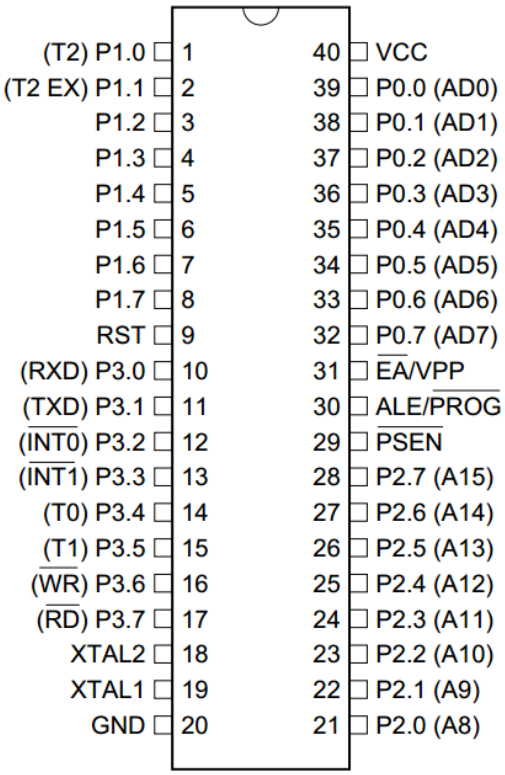
Vào ra tức là có thể dùng chân đó để đọc mức logic (0;1 tương ứng với 0V ; 5V)vào hay xuất mức logic ra(0;1)

P0 từ chân 39 32 tương ứng là các chân P0_0 P0_7

P1 từ chân 1 8 tương ứng là các chân P1_0 P1_7

P2 từ chân 21 28 tương ứng là các chân P2_0 P2_7

P3 từ chân 10 17 tương ứng là các chân P3_0 P3_7



Hình 4.1.3: Sơ đồ chân 89C51

Riêng cổng 3 có 2 chức năng ở mỗi chân như trên hình vẽ:

P3.0 – RxD : chân nhận dữ liệu nối tiếp khi giao tiếp RS232(Cổng COM).

P3.1 _ TxD : phân truyền dữ liệu nối tiếp khi giao tiếp RS232.

P3.2 _ INTO : interrupt 0 , ngắt ngoài 0.

P3.3 _ INT1: interrupt 1, ngắt ngoài 1.

P3.4 _T0 : Timer0 , đầu vào timer0.

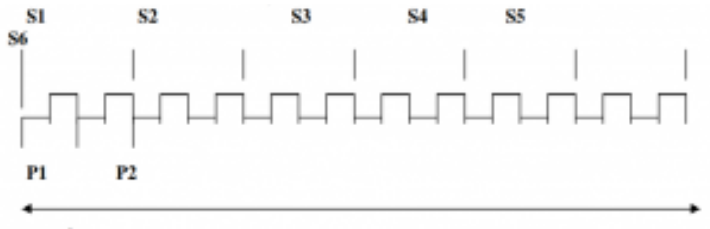
P3.5_T1 : Timer1, đầu vào timer 1.

P3.6_ WR: Write, điều khiển ghi dứ liệu.

P3.7 _RD: Read , điều khiển đọc dữ liệu.

Chân 18, 19 nối với thạch anh tạo thành mạch tạo dao động cho VĐK

Tần số thạch anh thường dùng trong các ứng dụng là : 11.0592Mhz(giao tiếp với cổng com máy tính) và 12Mhz Tần số tối đa 24Mhz. Tần số càng lớn VĐK xử lí càng nhanh.



Hình 4.1.4: Dao động của thạch anh

2.3. Lập trình 89C51.

Thực tế chỉ có 5 ngắt dành cho người dùng trong 8951 nhưng các nhà sản xuất nói rằng có 6 ngắt vì tính cả lệnh RESET. Sáu ngắt của 8951 được phân bố như sau:

- RESET: Khi chân RESET được kích hoạt từ 8051, bộ đếm chương trình nhảy về địa chỉ 0000H. Đây là địa chỉ bật lại nguồn.

- 2 ngắt dành cho các bộ định thời: 1 cho Timer0 và 1 cho Timer1. Địa chỉ tương ứng của các ngắt này là 000BH và 001BH.

- 2 ngắt dành cho các ngắt phần cứng bên ngoài: chân 12 (P3.2) và 13 (P3.3) của cổng P3 là các ngắt phần cứng bên ngoài INT0 và INT1 tương ứng. Địa chỉ tương ứng của các ngắt ngoài này là 0003H và 0013H.

- Truyền thông nối tiếp: có 1 ngắt chung cho cả nhận và truyền dữ liệu nối tiếp. Địa chỉ của ngắt này trong bảng vector ngắt là 0023H.

2.3.1. Trình phục vụ ngắt

Đối với mỗi ngắt thì phải có một trình phục vụ ngắt (ISR) hay trình quản lý ngắt để đưa ra nhiệm vụ cho bộ vi điều khiển khi được gọi ngắt. Khi một ngắt được gọi thì bộ vi điều khiển sẽ chạy trình phục vụ ngắt. Đối với mỗi ngắt thì có một vị trí cố định trong bộ nhớ để giữ địa chỉ ISR của nó. Nhóm vị trí bộ nhớ được dành riêng để lưu giữ địa chỉ của các ISR được gọi là bảng vector ngắt.

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

Bảng 4.1: Bảng vector ngắt của 8951.

Trong lập trình C trên Keil c cho 8051, chúng ta khai báo trình phục vụ ngắttheo cấu trúc sau:

```
Void  Name (void) interrupt X                //( X: là số thứ tự của ngắt )
{
      // chương trình phục vụ ngắt
}
```

Khi đó địa chỉ ngắt sẽ được tự động tính bằng:

Interrupt Address = (X * 8) + 3

2.3.2. Quy trình khi thực hiện một ngắt

Khi kích hoạt một ngắt bộ vi điều khiển thực hiện các bước sau:

- Hoàn thành nốt lệnh đang thực hiện và lưu địa chỉ của lệnh kế tiếp vào ngăn xếp.

- Lưu tình trạng hiện tại của tất cả các ngắt.

- Nhảy đến một vị trí cố định trong bộ nhớ được gọi là bảng vector ngắt, nơi lưu giữ địa chỉ của một trình phục vụ ngắt.

- Bộ vi điều khiển nhận địa chỉ ISR từ bảng vector ngắt và nhảy tới đó. Nó bắt đầu thực hiện trình phục vụ ngắt cho đến lệnh cuối cùng của ISR và trở về chương trình chính từ ngắt.

- Khi bộ vi điều khiển quay trở về nơi nó đã bị ngắt. Trước hết nó nhận địa chỉ của bộ đếm chương trình PC từ ngăn xếp bằng cách kéo 02 byte trên đỉnh của ngăn xếp vào PC. Sau đó bắt đầu thực hiện tiếp các lệnh từ địa chỉ đó.

2.3.3. Các bước cho phép và cấm ngắt.

Khi bật lại nguồn thì tất cả mọi ngắt đều bị cấm (bị che), có nghĩa là không có ngắt nào được bộ vi điều khiển đáp ứng trừ khi chúng được kích hoạt.

Các ngắt phải được kích hoạt bằng phần mềm để bộ vi điều khiển đáp ứng chúng. Có một thanh ghi được gọi là thanh ghi cho phép ngắt IE (Interrupt Enable) – ở địa chỉ A8H chịu trách nhiệm về việc cho phép và cấm các ngắt. Hình 2 trình bày chi tiết về thanh ghi IE.

Để cho phép một ngắt ta phải thực hiện các bước sau:

- Nếu EA = 0 thì không có ngắt nào được đáp ứng cho dù bit tương ứng của nó trong IEcó giá trị cao. Bit D7 - EA của thanh ghi IE phải được bật lên cao để cho phép các bit còn lại của thanh ghi hoạt động được.

- Nếu EA = 1 thì tất cả mọi ngắt đều được phép và sẽ được đáp ứng nếu các bit tương ứng của chúng trong IE có mức cao.

Ví dụ 1:

Hãy lập trình cho 8051:

a) cho phép **ngắt nối tiếp**, **ngắt Timer0** và **ngắt phần cứng ngoài 1** (EX1)

b) cấm **ngắt Timer0**

c) sau đó trình bày cách **cấm tất cả mọi ngắt** chỉ bằng một lệnh duy nhất.

Giải:

```
#include<at89x51.h>
main()
{
        //a)
        IE=0x96;        //1001 0110: lệnh này tương đương với 4 lệnh phía dưới
        EA=1;            //Cho phép sử dụng ngắt
        ES=1;           //Cho phép ngắt cổng nối tiếp
        ET0=1;          //Cho phép ngắt timer0
        EX1=1;          //Cho phép ngắt ngoài
        //b)
        ET0=0;          //Cấm ngắt timer0
        //c)
EA=0;           //Cấm tất cả các ngắt
        while(1)
        {
                //Chương trình chính
```

```
            //…
        }
}
```

2.3.4. Cờ quay về 0 của bộ định thời và ngắt

Bộ định thời TF được bật lên cao khi bộ định thời đạt giá trị cực đại và quay về 0 (Roll - over). Trong các bài trước chúng ta cũng chỉ ra cách kiểm tra cờ TF bằng một vòng lặp. Trong khi thăm dò cờ TF thì ta phải đợi cho đến khi cờ TFđược bật lên. Vấn đề với phương pháp này là bộ vi điều khiển bị trói buộc trong khi chờ cờ TF được bật và không thể làm được bất kỳ việc gì khác.

Sử dụng các ngắt sẽ giải quyết được vấn đề này và tránh được sự trói buộc bộ vi điều khiển. Nếu bộ ngắt định thời trong thanh ghi IE được phép thì mỗi khi nó quay trở về 0 bộ vi điều khiển sẽ bị ngắt, bất chấp nó đang thực hiện việc gì và nhảy tới bảng vector ngắt để phục vụ ISR. Bằng cách này thì bộ vi điều khiển có thể làm những công việc khác cho đến khi nó được thông báo rằng bộ định thời đã quay về 0.

2.3.5. Ngắt ngoài (External Interrupt)

8051 có 2 ngắt ngoài là INT0 và INT1. Ngắt ngoài được hiểu là ngắt được gây ra bởi sự kiện mức 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.

*** Bài tập ứng dụng**

1 Hãy viết chương trình điều khiển 8 led của port 0 sáng dồn theo chiều ngược lại.

2. Hãy viết chương trình điều khiển 16 led của 2 port: port0 và port1 sáng dồn.

3. Hãy viết chương trình điều khiển 3 port: port0, port1, port2 sáng dồn.

4. Hãy viết chương trình điều khiển 4 port: port0, port1, port2 và port3 sáng dồn.

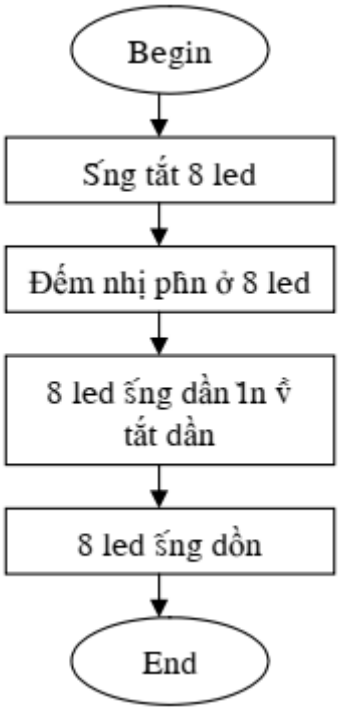
5. Hãy viết chương trình sáng dồn 2 port 0 và 1 từ ngoài vào trong và từ trong ra ngoài.
6. Hãy viết chương trình sáng dồn 4 port 0, 1, 2 và 3 từ ngoài vào trong và từ trong ra ngoài.

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0, port 1 sang don va tat het
;ket noi port 0 va port 1 den 16 led bang 2 soi cap 8 soi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
        org 0000h
        port01_6: mov r4,#00h ;luu trng thai ban dau
        mov r6,#00h
        mov p0,#00h ;luu trng thai ban dau
        mov p1,#00hlcall delay ;goi chuong trinh con delay
        mov 10h,#16 ;goi bien dem so lan dich
        chuyen cua led
        port01_6a: mov 11h,10h ;chuyen bien dem tung led
        mov r5,#00h ;nap 00 vao r5
        mov r7,#00h ;nap 00 vao r7
        setb c ;lam cho bit C = 1
        port01_6b:
        mov a,r7
        rrc a ;xoay noi dung thanh ghi A sang trai
        mov r7,a ;cat lai vao r7 de luu cho lan xu li ke
        orl a,r4 ;lay ket qua do or voi r4 roi goi
        ra p1
        mov p1,a
        mov a,r5
        rrc a ;xoay noi dung thanh ghi A sang trai
        mov r5,a ;cat lai vao r5 de luu cho lan xu li ke
        orl a,r6 ;lay ket qua do or voi r5 roi goi
        ra p0
        mov p0,a
        lcall delay
        clr c ;xoa Cy de chi dich 1 led di
        djnz 11h,port01_6b ;giam ndung o nho (11h)<>
        0 thi quay lai
        mov r4,p1mov r6,p0
        djnz 10h,port01_6a ;giam bien dem de xu li lan
        ke
        ljmp port01_6
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
        delay: mov 7eh,#040h
        del: mov 7fh,#0ffh
        djnz 7fh,$
        djnz 7eh,del
        ret
        end
```

7. Chương trình điều khiển led sang bằng cách tổ hợp các bài đã học

Bước 1: Giải thuật.



Bước 2: Kết nối mạch theo trình tự.

- Dùng bus dây kết nối port 0 với một trong bốn PINHD của dãy 32 led.
- Gắn vi điều khiển vào đế nạp 40 pin (socket) ở modul nạp của hệ thống 2.

Bước 3: Khởi động phần mềm, mở File mới và đặt tên file.

Bước 4: Viết chương trình với tên file vừa đặt.

```
;chuong trinh tong hop cac chuong trinh da viet dieu khien port 0
;cac chuong trinh bao gom:
;chop tat 5 lan
;sang dan tu tren xuong va tat het 3 lan
;sang dan tu duoi len va tat het 3 lan
;sang dan va tat dan tu tren xuong 3 lan
;sang dan va tat dan tu duoi len 3 lan
```

```
;sang don tu tren xuong 3 lan
;sang don tu duoi len 3 lan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
;chuong trinh chinh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
        org 0000h
        main: lcall choptat
        lcall sangdantx ;chtr sang dan tren xuong
        lcall choptat
        lcall sangdandl ;chtr sang dan duoi len
        lcall choptat
        lcall stdtx ;sang tat dan tren xuong
        lcall choptatlcall stddl ;sang tat dan duoi len
        lcall choptat
        lcall sangdontx ;sang don tren xuong
        lcall choptat
        lcall sangdondl
        lcall choptat
        lcall sangdontx ;sang don tren xuong
        lcall shtdtx
        lcall choptat
        lcall sangdondl
        lcall shtddl
        sjmp main
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
;chuong trinh chop tat port 0: 5 lan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
        choptat: mov r7,#5 ;5 lan
        loop1: mov p0,#0
        lcall delay
        mov p0,#0ffh
        lcall delay
        djnz r7,loop1
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
;chuong trinh sang dan port0 tu tren xuong: 3 lan
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxsangdantx: mov r7,#3 ;3 lan
        loop2: mov a,#0
        loop3: mov p0,a
        lcall delay
        setb c
        rlc a
        jnc loop3
        djnz r7,loop2
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
;chuong trinh sang dan port0 tu duoi len: 3 lan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
        sangdandl: mov r7,#3 ;3 lan
        loop5: mov a,#0
        loop4: mov p0,a
        lcall delay
        setb c
        rrc a
        jnc loop4
        djnz r7,loop5
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh sang tat dan port0 tu tren xuong: 3 lan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
stdtx:mov r7,#3 ;3 lan
        loop6: mov a,#0loop7: mov p0,a
        lcall delay
        setb c
        rlc a
        jnc loop7
        loop8: clr c
        rlc a
        mov p0,a
        lcall delay
        jnc loop8
        djnz r7,loop6
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
;chuong trinh sang tat dan port0 tu duoi len: 3 lan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        stddl: mov r7,#3 ;3 lan
        loop9: mov a,#0
        loop10: mov p0,a
        lcall delay
        setb c
        rrc a
        jnc loop10
        loop11: clr c
        rlc a
        mov p0,a
        lcall delay
        jnc loop11djnz r7,loop9
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang don va tat het tu tren xuong
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        sangdontx:mov r7,#3 ;bien dem 3 chu ky
        loop12: mov r2,#00h
        mov r4,#08
        loop13: mov r5, 04h
        mov r3,#00h
        mov a,r3
        orl a,r2
        mov p0,a
        lcall delay
        setb c
        loop14: mov a,r3
        rrc a
        mov r3,a
        orl a,r2
        mov p0,a
        lcall delay
        clr cdjnz r5,loop14
        mov r2,a
        djnz r4,loop13
        djnz r7,loop12
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang don va tat het tu tren xuong
```

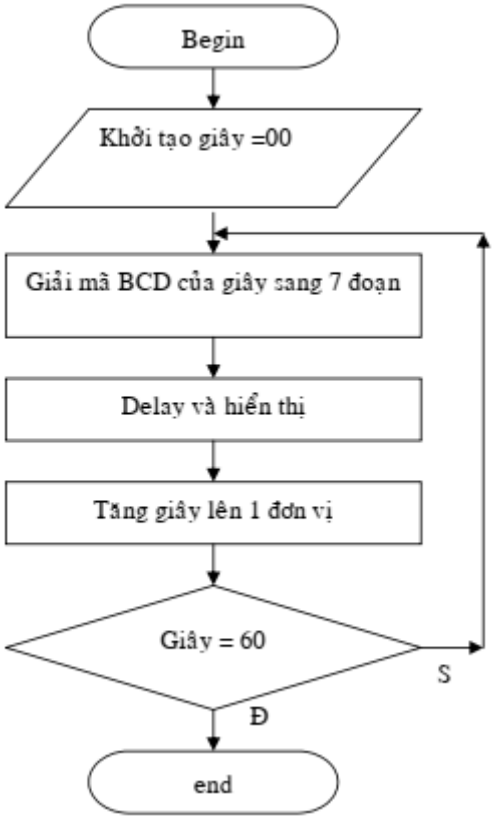
```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        sangdondl:mov r7,#3 ;bien dem 3 chu ky
        loop15: mov r2,#00h
        mov r4,#08
        loop16: mov r5, 04h
        mov r3,#00h
        mov a,r3
        orl a,r2
        mov p0,a
        lcall delay
        setb c
        loop17: mov a,r3
        rlc a
        mov r3,a
        orl a,r2mov p0,a
        lcall delay
        clr c
        djnz r5,loop17
        mov r2,a
        djnz r4,loop16
        djnz r7,loop15
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang het va tat dan tu phai sang trai
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        shtdtx: mov r7,#3 ;so lan thuc hien la 3
        loop20: mov p1,#0ffh
        lcall delay
        mov r0,#10000000b ;diem xuat phat
        mov r3,#11111111b ;luu tru trang thai ban dau
        mov 12h,#8 ;bien dem so led dich chuyen
        mov 10h,#1 ;bien dem so lan dich chuyen cua
        1 led
        loop22: mov 11h,10h ;luu tru so lan thuc hien
        mov 20h,r0 ;chuyen r0 sang o nho 20h
        loop21: lcall xnguoc09 ;xoay noi dung trong r1 sang phai
        ket qua luu trong 0 20h
        mov a,r0cpl a ;nghich dao a
        anl a,r3
        orl a,20h
        mov p0,a ;hien thi o p0
```

```
        lcall delay
        djnz 11h,loop21
        lcall xthuan09
        inc 10h
        djnz 12h,loop22 ;quay lai xu li led ke
        djnz r7,loop20
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;cac chuong trinh con
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        xthuan09: mov a,r0
        rr a
        mov r0,a
        clr c
        mov a,r3
        rrc a
        mov r3,a
        ret
        xnguoc09: mov a,20h
        clr c
        rlc amov 20h,a
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang het va tat dan tu phai sang trai
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        shtddl: mov r7,#3 ;so lan thuc hien la 3
        loop30: mov p1,#0ffh
        lcall delay
        mov r0,#00000001b ;diem xuat phat
        mov r3,#11111111b ;luu tru trang thai ban dau
        mov 12h,#8 ;bien dem so led dich chuyen
        mov 10h,#1 ;bien dem so lan dich chuyen cua
        1 led
        loop32: mov 11h,10h ;luu tru so lan thuc hien
        mov 20h,r0 ;chuyen r0 sang o nho 20h
        loop31: lcall xnguoc10 ;xoay noi dung trong r1 sang phai
        ket qua luu trong 0 20h
        mov a,r0
        cpl a ;nghich dao a
        anl a,r3
        orl a,20h
```

```
        mov p0,a ;hien thi o p0
        lcall delaydjnz 11h,loop31
        lcall xthuan10
        inc 10h
        djnz 12h,loop32 ;quay lai xu li led ke
        djnz r7,loop30
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;cac chuong trinh con
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        xthuan10: mov a,r0
        rl a
        mov r0,a
        clr c
        mov a,r3
        rlc a
        mov r3,a
        ret
        xnguoc10: mov a,20h
        clr c
        rrc a
        mov 20h,a
        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        delay: mov 7fh,#50h
        de: mov 7eh,#0
        djnz 7eh,$
        djnz 7fh,de
        ret
        end
```

6. Chương trình đếm từ 00 đến 60 hiển thị trên 2 led

Giải thuật:




```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dem len tu 00 den 60 hien thi tren 2 led cua 8 led
quet
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
giay equ r2 ;gan bien dem giay la R2
org 200h ;dia chi khai bao ma 7 doan tu so '0'
den so '9'
ma7doan: db
0C0h,0F9h,0A4h,0B0h,99h,92h,82h,0F8h,80h,90h
org 0000h ;bat dau chuong trinh
mov tmod,#01h ;timer0: mod 1 - dem 16 bit
mov dptr,#0200h ;dptr quan ly vung ma 7 doan
main: mov giay,#00h ;giay=00
main1: lcall gma
lcall delay_hthi ;goi chtr con delay co hien thi
mov a,giay ;chuyen giay sang Aadd a,#1 ;tang giay len 1
da a ;hieu chinh so BCD trong A
mov giay,a ;tra lai cho giay
cjne giay,#60h,main1 ;ss giay voi 60
sjmp main ;lam lai tu dau
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con giai ma
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
gma: mov a,giay
anl a,#0fh ;xoa 4 bit cao hang chuc giay
movc a,@a+dptr;lay ma 7 doan
mov 27h,a ;cat ma vao o nho 20h
mov a,giay
anl a,#0f0h ;xoa 4 bit thap hang dvi
swap a ;chuyen 4 bit cao xuong vi tri
thap
movc a,@a+dptr;lay ma 7 doan hang chuc
mov 26h,a
ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh delay co goi chuong trinh hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay_hthi:mov 7fh,#10h
del2: clr tr0
mov th0,#00
mov tl0,#00
setb tr0clr tf0
del1: lcall hthi
jnb tf0,del1
djnz 7fh,del2
ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi: mov a,#01111111b ;ma quet
mov r0,#27h
ht1: mov p0,@r0
mov p2,a
lcall delay1
mov p2,#0ffh
dec r0
rr a ;chuyen sang led ke
cjne r0,#25h,ht1
ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1: mov r7,#0fh
del11: djnz r7,del11
```

ret
end5. Thực hiện các bước giống như các bài chuẩn cho đến khi
mạch đếm đúng từ 00 đến 59.
8. Chương trình điều khiển đèn giao thông

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh  dieu khien den giao thong co hien thi thong so thoi gian tren 2 led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;port 3 dieu khien cac led xanh vang do, Xanh sang 15 giay, Vang 5 giay, Do 20 giay.
;ket noi port 0: p00 den p07  dieu khien cac doan a,b,c,d,e,f,g,p
;ket noi port 2: p20 den p27  dieu khien quet led0 den led7
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;du lieu dieu khien den
                X1_d2       EQU  11011110B  ;XANH 1, DO 2 SANG
                V1_d2       EQU  11011101B  ;VANG 1, DO 2 SANG
                d1_X2       EQU  11110011B  ;DO 1, XANH 2 SANG
                D1_V2       EQU  11101011B  ;DO 1, VANG 2 SANG
;               X1_d2       EQU  10000100B  ;XANH 1, DO 2 SANG
;               V1_d2       EQU  01000100B  ;VANG 1, DO 2 SANG
;               d1_X2       EQU  00100001B  ;DO 1, XANH 2 SANG
;               D1_V2       EQU  00100010B  ;DO 1, VANG 2 SANG
                tg_xanh     equ      14      ;24 dem xuong 0 tuc da dem 25
                tg_vang     equ      4       ;4 dem xuong 0 tuc da dem 5
                tg_do       equ      19      ;29 dem xuong 0 tuc da dem 30
                led7        equ      p3      ;dieu khien cac doan a,b,c,d,e,f,g,dp
                quet        equ      p2      ;dieu khien quet cac transistor T1 den T8
                leddonto    equ      p0      ;dieu khien led don loai to
;bat dau chuong trinh chinh
                org 0000h
                mov p1,#00
                mov leddonto,#0ffh
                mov tmod,#00000001b
                mov dptr,#ma7doan
                mov 20h,#0ffh
                mov 21h,#0ffh
                mov 23h,#0ffh
                mov 22h,#0ffh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xanh1 va do2 sang 15 giay
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
b222:       mov 16h,#tg_xanh        ;bien dem thoi gian cho ht1
            mov 17h,#tg_do              ;bien dem thoi gian cho ht2
            mov leddonto,#x1_d2         ;cho xanh1, do2 sang
b221:       lcall hextobcd
            lcall gma
            lcall delay
            dec 17h
            djnz 16h,b221
            lcall hextobcd
            lcall gma
            lcall delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;vang1 va do2 sang 5 giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            dec 17h
            mov 16h,#tg_vang        ;bien dem thoi gian cho ht2
            mov leddonto,#v1_d2         ;cho vang 1, do2 sang
b221a:      lcall hextobcd
            lcall gma
            lcall delay
            dec 16h
            djnz 17h,b221a
            lcall hextobcd
            lcall gma
            lcall delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;do1 va xanh2 sang 15 giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            mov 16h,#tg_do          ;bien dem thoi gian cho ht1
            mov 17h,#tg_xanh            ;bien dem thoi gian cho ht2
            mov leddonto,#d1_x2         ;do1 va xanh2 sang
b221b:          lcall hextobcd
            lcall gma
            lcall delay
            dec 16h
            djnz 17h,b221b
            lcall hextobcd
            lcall gma
            lcall delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
;do1 va vang2 sang 5 giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                dec 16h
                mov 17h,#tg_vang
                mov leddonto,#d1_V2             ;do1 va xanh2 sang
b221c:          lcall hextobcd
                lcall gma
                lcall delay
                dec 17h
                djnz 16h,b221c
                lcall hextobcd
                lcall gma
                lcall delay
                ljmp b222
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con chuyen so hex sang so bcd
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hextobcd:       mov a,17h
                mov b,#10
                div ab ;b luu hang don vi
                swap a;
                orl a,b
                mov 37h,a
                mov a,16h
                mov b,#10
                div ab ;b luu hang don vi
                swap a;
                orl a,b
                mov 36h,a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con giai ma
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
gma:            mov a,37h
                anl a,#0fh
                movc a,@a+dptr
                mov 27h,a
                mov a,37h
                anl a,#0f0h
                swap a
                movc a,@a+dptr
```

```
                mov 26h,a
                mov a,36h
                anl a,#0fh
                movc a,@a+dptr
                mov 25h,a
                mov a,36h
                anl a,#0f0h
                swap a
                movc a,@a+dptr
                mov 24h,a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh delay co goi chuong trinh hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov 7fh,#10h
del2:           clr tr0
                mov th0,#00
                mov tl0,#00
                setb tr0
                clr tf0
del1:           lcall hthi
                jnb tf0,del1
                djnz 7fh,del2
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           mov r1,#11111110b;ma quet
                mov r0,#27h         ;nap dia chi quan ly vung ma 7doan vao r0
                mov r5,#8
ht1:            mov led7,@r0
                mov quet,r1
                lcall delay1
                mov quet,#0ffh              ;chong lem
                dec r0
                mov a,r1
                rl a
                mov r1,a
                djnz r5,ht1         ;chi co 2 so nen so sanh voi 62H de ket thuc
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
;chuong trinh con delay1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1:         mov r7,#0fh
                djnz r7,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;vung ma 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:    db      0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h,080h,090h
end
```

CÂU HỎI ÔN TẬP

1. Trao đổi dữ liệu bằng ngắt là gì? Qui trình xử lý ngắt trong 89C51?
2. Hãy cho biết chương trình chính là gì? Chương trình con? Trong một chương trình có bao nhiêu chương trình con
3. Trình bày các thanh ghi sử dụng trong ngắt? Nhiệm vụ chức năng từng thanh ghi?
4. Tại sao phải chèn chương trình con hiển thị vào chương trình con delay? Hãy thử bỏ lệnh gọi chương trình con hiển thị trong chương trình con delay và cho nó vào chương trình chính sau lệnh gọi chương trình con delay rồi cho biết kết quả như thế nào?

Bài tập

1. Hãy viết chương trình cho 2 port, 3 port và 4 port theo 1 chiều từ trên xuống, từ dưới lên và từ ngoài vào trong và từ trong ra ngoài?
2. Hãy viết chương trình sáng tắt port 2 sử dụng timer làm bộ định thời delay 5 giây?
3. Hãy viết chương trình giống trên nhưng delay 10 giây?
4. Tương tự hãy viết chương trình delay 1 giờ?
5. Viết chương trình mạch đếm sản phẩm? Tiến hành thi công mạch và nạp chương trình vào mạch?
6. Viết chương trình mạch đèn giao thông?
7. Viết chương trình hiển thị Giờ-Phút-Giây trên led 7 đoạn?
8. Hãy viết chương trình đếm lên từ 00 đến 99 hiển thị trên 2 led 7 đoạn?
9. Hãy viết chương trình đếm xuống từ 60 về 00 hiển thị trên 2 led 7 đoạn?

TÀI LIỆU THAM KHẢO

[1] Nguyễn Trung Lập. Kỹ thuật xung số - Nhà xuất bản KHKT, 2000
[2] Nguyễn Thương Ngô. Kỹ thuật xung số - Nhà xuất bản Thống Kê, 2002
[3] TS Nguyễn Viết Nguyên. Giáo trình linh kiện điện tử và ứng dụng - Nhà xuất bản Giáo dục
[4] Nguyễn Hữu Phương. Mạch số - NXB khoa học kỹ thuật 2004
[5] Giáo trình kỹ thuật số - ĐH SPKT TP. HCM