Chương 1

LỊCH SỬ PHÁT triển của vi xử lý

I. Lịch sử phát triển của vi xử lý:

1. Lịch sử phát triển của vi xử lý.
2. Chức năng của Vi xử lý.
3. Máy vi tính.
4. Năng lực của máy vi tính.

II. Các khái niệm cơ bản về cấu trúc của vi xử lý:

1. Chiều dài từ dữ liệu.
2. Khả năng truy xuất bộ nhớ.
3. Tốc độ làm việc của vi xử lý.
4. Các thanh ghi của vi xử lý.
5. Các lệnh của vi xử lý.
6. Các kiểu truy xuất bộ nhớ.
7. Các mạch điện giao tiếp của vi xử lý.

I. LỊCH SỬ PHÁT TRIỂN VI XỬ LÝ:

1. Lịch sử phát triển vi xử lý:

Vi xử lý là sự kết hợp của 2 kỹ thuật công nghệ quan trọng nhất: đó là máy tính dùng kỹ thuật số (Digital computer) và các mạch vi điện tử. Hai công nghệ này kết hợp lại với nhau vào năm 1970, sau đó các nhà nghiên cứu đã chế tạo ra vi xử lý (Microprocessor).

Máy tính số là các mạch điện xử lý tín hiệu dạng số được điều khiển bởi chương trình, có thể làm những công việc mà con người mong muốn. Chương trình sẽ điều khiển các mạch điện số cách di chuyển và xử lý dữ liệu (data) bằng cách điều khiển các mạch logic số học, các bộ nhớ (memory), các thiết bị xuất / nhập (Input/output). Cách thức các mạch điện logic của máy tính số kết hợp lại với nhau tạo thành các mạch logic số học, các vi mạch nhớ và các thiết bị xuất / nhập được gọi là cấu trúc.

Vi xử lý có cấu trúc giống như máy tính số và có thể xem nó là máy tính số vì cả hai đều tính toán dưới sự điều khiển của chương trình.

Lịch sử phát triển của vi xử lý gắn liền với sự phát triển của các vi mạch điện tử vì vi xử lý là vi mạch điện tử chế tạo theo công nghệ LSI (large scale integrated) cho đến VLSI (very large scale integrated).

Với sự khám phá ra transistor và phát triển của công nghệ chế tạo vi mạch SSI, MSI, máy tính vẫn còn là một nhóm gồm nhiều IC kết hợp lại với nhau, cho đến thập niên 70, với sự phát triển của công nghệ LSI, cấu trúc máy tính được rút gọn bởi các nhà thiết kế và được chế tạo thành một IC duy nhất được gọi là vi xử lý (microprocessor).

Vi xử lý kết hợp với các thiết bị khác tạo ra các máy tính có khả năng tính toán rất lớn như máy vi tính và có thể tạo ra các sản phẩm khác các máy điện thoại, các tổng đài điện thoại, các hệ thống điều khiển tự động...

Vi xử lý đầu tiên có khả năng xử lý 4 bit dữ liệu, các vi xử lý này có tốc độ xử lý rất chậm, các nhà thiết kế cải tiến thành vi xử lý 8bit, sau đó là vi xử lý 16 bit và 32 bit. Sự phát triển về dung lượng các bit của vi xử lý làm tăng thêm số lượng các lệnh điều khiển và các lệnh tính toán phức tạp.

2. Chức năng của xử lý

Vi xử lý dùng các cổng logic giống như các cổng logic được sử dụng trong đơn vị xử lý trung tâm (central procecessing unit) của máy tính số. Do cấu trúc giống như CPU và được xây dựng từ các mạch vi điện tử nên có tên là vi xử lý: microprocessor. Giống như CPU, microprocessor có các mạch điện tử cho việc điều khiển dữ liệu (data) và tính toán dữ liệu dưới sự điều khiển của chương trình. Ngoài ra microprocessor là một đơn vị xử lý dữ liệu.

Công việc xử lý dữ liệu là chức năng chính của vi xử lý. Việc xử lý dữ liệu bao gồm tính toán và điều khiển dữ liệu. Việc tính toán được thực hiện bởi các mạch điện logic được gọi là đơn vị xử logic số học (arithmetic logic unit: ALU) có thể thực hiện các phép toán như Add, Subtract, And, Or, Compare, Increment, Decrement.

ALU không thể thực hiện một phép toán mà không có dữ liệu, ví dụ ALU cộng 2 dữ liệu với nhau thì 2 dữ liệu phải đặt đúng vị trí trước khi cộng. ALU không thể thực hiện việc chuyển dữ liệu từ nơi này đến nơi khác. Để ALU có dữ liệu cho việc xử lý thì ngoài mạch điện ALU, vi

xử lý còn có các mạch điện logic khác để điều khiển dữ liệu. Các mạch điện logic điều khiển dữ liệu sẽ di chuyển dữ liệu vào đúng vị trí để khối ALU xử lý dữ liệu. Sau khi thực hiện xong, khối điều khiển sẽ di chuyển dữ liệu đến bất cứ nơi nào mong muốn.

Để xử lý dữ liệu, vi xử lý phải điều khiển các mạch logic, để vi xử lý điều khiển các mạch logic thì cần phải có chương trình. Chương trình là tập hợp các lệnh để xử lý dữ liệu thực hiện từng lệnh đã được lưu trữ trong bộ nhớ, công việc thực hiện lệnh bao gồm các bước như sau: đón lệnh từ bộ nhớ, sau đó các mạch logic điều khiển sẽ giải mã lệnh và sau cùng thì các mạch logic điều khiển sẽ thực hiện lệnh sau khi mã giải mã.

Do các lệnh lưu trữ trong bộ nhớ nên có thể thay đổi các lệnh nếu cần. Khi thay đổi các lệnh của vi xử lý tức là thay đổi cách thức xử lý dữ liệu. Các lệnh lưu trữ trong bộ nhớ sẽ quyết định công việc mà vi xử lý sẽ làm.

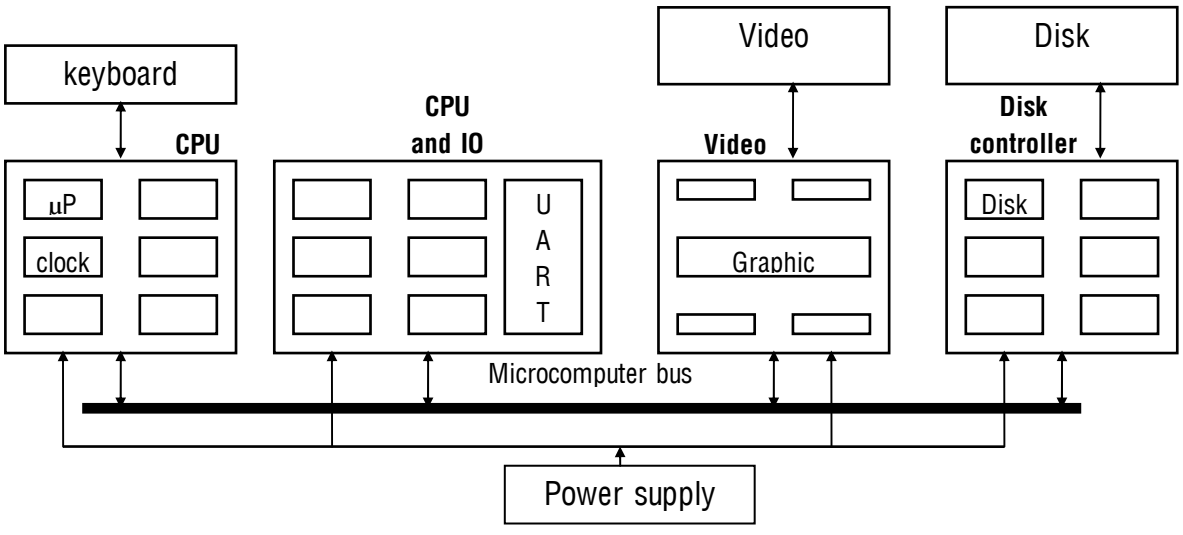
Tóm tắt: Chức năng chính của vi xử lý là xử lý dữ liệu. Để làm được điều này vi xử lý phải có các mạch logic cho việc xử lý và điều khiển dữ liệu và các mạch logic điều khiển. Các mạch logic xử lý sẽ di chuyển dữ liệu từ nơi này sang nơi khác và thực hiện các phép toán trên dữ liệu, mạch logic điều khiển sẽ quyết định mạch điện nào cho việc xử lý dữ liệu. vi xử lý thực hiện một lệnh với trình tự như sau: đón lệnh từ bộ nhớ, tiếp theo mạch logic điều khiển sẽ giải mã lệnh để xem lệnh đó yêu cầu vi xử lý thực hiện công việc gì, sau đó vi xử lý sẽ thực hiện đúng công việc của lệnh đã yêu cầu, quá trình này được gọi là chu kỳ đón - và - thực hiện lệnh (fetch / execute cycle).

Ngoài chức năng đón và thực hiện lệnh, các mạch logic điều khiển còn điều khiển các mạch điện giao tiếp bên ngoài kết nối với vi xử lý. vi xử lý cần phải có sự trợ giúp của các mạch điện bên ngoài. Các mạch điện dùng để lưu trữ lệnh để vi xử lý xử lý được gọi là bộ nhớ, các mạch điện giao tiếp để di chuyển dữ liệu từ bên ngoài vào bên trong vi xử lý và xuất dữ liệu từ bên trong vi xử lý ra ngoài được gọi là các thiết bị I/O hay các thiết bị ngoại vi.

3. **Máy vi tính (Microcomputer)**:

Vi xử lý là một IC chuyên về xử lý data và điều khiển còn máy vi tính là một hệ thống máy tính hoàn chỉnh được xây dựng từ một vi xử lý. Máy vi tính hoàn chỉnh bao gồm một vi xử lý, bộ nhớ và các cổng I/O.

Sơ đồ khối của một hệ thống máy vi tính như hình 1-1:



Hình 1-1. Cấu trúc của một máy vi tính.

Máy vi tính tổ chức theo card bao gồm: CPU card, card bộ nhớ RAM, card điều khiển đĩa, card điều khiển màn hình, ngoài ra máy vi tính còn có màn hình video, bàn phím...

Tất cả các card trong máy vi tính được kết nối với vi xử lý thông qua bus, bus bao gồm nhiều đường tín hiệu để phân biệt và xử lý các card khác nhau.

Trong card CPU có mạch tạo xung Clock dùng để tạo ra tín hiệu clock cho vi xử lý. Card CPU còn có các IC giao tiếp để nâng cao khả năng giao tiếp của CPU.

Bộ nhớ ROM dùng để lưu trữ các lệnh của chương trình để cho phép nạp các chương trình từ đĩa mềm.., card bộ nhớ RAM bao gồm các IC RAM để vi xử lý lưu trữ chương trình và dữ liệu khi xử lý. Trong card bộ nhớ có phần xuất nhập data nối tiếp UART (Universal asynchronous reciever - transmitter), hai khối này có thể tách rời. UART dùng để chuyển đổi dữ liệu song song thành nối tiếp để máy vi tính có thể giao tiếp với máy in, các modem, và các thiết bị điều khiển khác.

Để giao tiếp với màn hình video cần phải có card video, bên cạch các IC giao tiếp với bus của vi xử lý còn có các IC điều khiển màn hình Video. Màn hình Video dùng để hiển thị nội dung của một vùng nhớ đặc biệt trong bộ nhớ RAM do đó Card video có các IC RAM.

Khối nguồn cung cấp điện cho tất cả các hệ thống.

4. **Năng lực của vi xử lý**:

Khi nói đến năng lực của vi xử lý có nghĩa là nói đến khả xử lý data, có 3 thông số để đánh năng lực của vi xử lý:

- Chiều dài của từ data của vi xử lý.
- Số lượng các ô nhớ mà vi xử lý có thể truy xuất được.
- Tốc độ mà vi xử lý có thể thực hiện một lệnh.

Các vi xử lý thường được so sánh với nhau thông qua độ dài các từ data, mỗi một vi xử lý làm việc với các từ dữ liệu có độ dài cố định. Độ dài từ dữ liệu của các vi xử lý bao gồm: 4 bit, 8 bit, 16 bit, 32 bit và sắp tới là 64 bit. Từ dữ liệu thường được sử dụng là 8 bit tương với một byte.

Một thông số thường để đánh giá năng lực của vi xử lý là dung lượng bộ nhớ dữ liệu mà vi xử lý có thể truy xuất, trong trường hợp này độ dài của từ dữ liệu cũng đóng một vai trò quan trọng khi so sánh. Độ dài của từ dữ liệu trong bộ nhớ bằng với độ dài của từ dữ liệu trong vi xử lý, ví dụ vi xử lý 4 bit thì từ dữ liệu lưu trữ trong bộ nhớ sẽ có độ dài là 4 bit.

Mỗi một từ dữ liệu trong bộ nhớ được chỉ định bởi một địa chỉ (Address), vi xử lý muốn truy xuất dữ liệu của ô nhớ nào phải tạo ra địa chỉ của ô nhớ đó. Vi xử lý có khả năng giao tiếp với bộ nhớ có dung lượng càng lớn thì vi xử lý đó càng mạnh.

Thông số thứ 3 là tốc độ xử lý một lệnh của vi xử lý được đánh giá qua tần số xung clock cung cấp cho vi xử lý làm việc. Vi xử lý có tần số làm việc càng lớn thì tốc độ xử lý càng nhanh.

II. CÁC KHÁI NIỆM CƠ BẢN VỀ CẤU TRÚC CỦA VI XỬ LÝ:

1. **Chiều dài từ dữ liệu:**

Vi xử lý đầu tiên có chiều dài từ dữ liệu là 4 bit, tiếp theo là các vi xử lý 8 bit, 16 bit, 32 bit và 64 bit. Mỗi vi xử lý có chiều dài từ dữ liệu khác nhau sẽ có một khả năng ứng dụng khác nhau, các vi xử lý có chiều dài từ dữ liệu lớn, tốc độ làm việc nhanh, khả năng truy xuất bộ nhớ lớn được dùng trong các công việc xử lý dữ liệu, điều khiển phức tạp, các vi xử lý có chiều dài từ dữ liệu nhỏ hơn, khả năng truy xuất bộ nhớ nhỏ hơn, tốc độ làm việc thấp hơn được sử dụng trong các công việc điều khiển và xử lý đơn giản, chính vì thế các vi xử lý này vẫn tồn tại.

Các vi xử lý 16 bit, 32 bit được sử dụng rất nhiều trong máy tính. Máy vi tính đầu tiên của IBM sử dụng vi xử lý 8088 vào năm 1981. Cấu trúc bên trong của vi xử lý 8088 có thể xử lý các từ dữ liệu 16 bit, nhưng bus dữ liệu giao tiếp bên ngoài chỉ có 8 bit. Do cấu trúc bên trong 16 bit nên các máy tính PC sử dụng bộ vi xử lý 8088 có thể tương thích với các máy tính mới sử dụng các vi xử lý 16 bit: 286, hoặc các vi xử lý 32 bit: 386, 486 và bộ vi xử lý Pentium.

Hầu hết các ứng dụng được điều khiển bởi máy tính tốt hơn nhiều so với vi xử lý và tùy theo yêu cầu điều khiển mà chọn điều khiển bằng máy tính hay điều khiển bằng vi xử lý. Các lĩnh vực điều khiển bằng vi xử lý như: công nghiệp, khoa học, y học... Một lĩnh vực điều khiển phức tạp là robot khi đó các bộ vi xử lý 16 bit và 32 bit là thích hợp. Tùy theo yêu cầu độ phức tạp mà chọn bộ vi xử lý thích hợp.

Vi xử lý 32 bit là sự phát triển của vi xử lý 16 bit và ứng dụng đầu tiên của các vi xử lý 32 bit là các máy tính 32 bit. Các vi xử lý 32 bit có khả năng làm việc nhanh hơn vì mỗi lần lấy dữ liệu từ bộ nhớ vi xử lý có thể lấy một lần 4 byte, trong khi đó các vi xử lý 8 bit thì phải làm 4 lần, với vi xử lý 16 bit phải thực hiện 2 lần. Vậy nếu so với vi xử lý 8 bit thì vi xử lý 32 bit có tốc độ tăng gấp 4, với vi xử lý 16 bit thì tốc độ vi xử lý 32 bit tăng gấp đôi. Để tăng tốc độ làm việc của vi xử lý là mục tiêu hàng đầu của các nhà chế tạo vi xử lý.

2. **Khả năng truy xuất bộ nhớ**:

Dung lượng bộ nhớ mà vi xử lý có thể truy xuất là một phần trong cấu trúc của vi xử lý. Các vi xử lý đầu tiên bị giới hạn về khả năng truy xuất bộ nhớ: vi xử 4004 có 14 đường địa chỉ nên có thể truy xuất được 2^{14} = 16.384 ô nhớ, vi xử lý 8 bit có 16 đừng địa chỉ nên có thể truy xuất được 2^{16} = 65.536 ô nhớ, vi xử lý 16 bit có 20 đường địa chỉ nên có thể truy xuất 2^{20} = 1.024.000 ô nhớ, vi xử lý 32 bit như 386 hay 68020 có thể truy xuất 4 G ô nhớ. Vi xử lý có khả năng truy xuất bộ nhớ càng lớn nên có thể xử lý các chương trình lớn. Tùy theo ứng dụng cụ thể mà chọn một vi xử lý thích hợp.

3. **Tốc độ làm việc của vi xử lý**:

Tần số xung clock cung cấp cho vi xử lý làm việc quyết định đến tốc độ làm việc của vi xử lý, vi xử lý có tốc độ làm việc càng lớn thì khả năng xử lý lệnh càng nhanh. Tần số xung clock làm việc của các vi xử lý được cho bởi các nhà chế tạo:

Vi xử lý	Tần số xung clock	chiều dài từ dữ liệu
8051	12MHz	8-bit
Z80A	4MHz	8-bit
Z80B	6MHz	8-bit
286	16MHz	16-bit
486DX2-66	66Mhz	32-bit
Pentium	66MHz	32-bit

4. **Các thanh ghi của vi xử lý**:

Các thanh ghi là một phần quan trọng trong cấu trúc của vi xử lý.

Các thanh ghi bên trong của vi xử lý dùng để xử lý dữ liệu, có nhiều loại thanh ghi khác nhau cho các chức năng khác nhau trong vi xử lý, số lượng các thanh ghi đóng một vai trò rất quan trọng đối với vi xử lý và người lập trình.

Các vi xử lý khác nhau sẽ có số lượng và chức năng của các thanh cũng khác nhau.

Nếu vi xử lý có số lượng thanh ghi nhiều thì người lập trình có thể viết các chương trình điều khiển vi xử lý đơn giản hơn, làm tăng tốc độ xử lý chương trình. Nếu vi xử lý có số lượng thanh ghi ít thì chương trình sẽ phức tạp hơn, tốc độ xử lý chương trình chậm hơn.

Để hiểu rõ các thanh ghi bên trong của một vi xử lý cần phải khảo sát một vi xử lý cụ thể. Vậy số lượng các thanh ghi bên trong vi xử lý cũng ảnh hưởng đến tốc độ và khả năng xử lý chương trình.

5. **Các lệnh của vi xử lý**:

Tập lệnh của vi xử lý là một trong những yếu tố cơ bản để đánh giá tốc độ làm việc của vi xử lý. Nếu vi xử lý có nhiều mạch điện logic bên trong để thực hiện thì số lệnh điều khiển của vi xử lý càng nhiều, khi đó vi xử lý càng lớn và độ phức tạp càng lớn. Ví dụ so sánh 2 tập lệnh của 2 vi xử lý 8 bit là 80C51 và Z80 thì 80X51 có 111 lệnh khác nhau còn Z80 có 178 lệnh. Tập lệnh của một vi xử lý càng nhiều rất có ích khi lập trình hay viết chương trình cho vi xử lý.

6. **Cấu trúc truy xuất bộ nhớ**:

Một yếu tố quyết định sự mềm dẻo trong lập trình là số lượng các kiểu truy xuất bộ nhớ khác nhau của vi xử lý, vi xử lý có nhiều kiểu truy xuất bộ nhớ sẽ có khả năng xử lý càng nhanh và cấu trúc các mạch điện bên trong càng phức tạp. Các kiểu truy xuất bộ nhớ của các vi xử lý 8 bit và 16 bit:

Kiểu truy xuất bộ nhớ (Addressing mode)	Vi xử lý 6800	Vi xử lý Z80	Vi xử lý 8088
Implied - hiểu ngầm.	x	x	x
8-bit	x	x	x
16-direct	x	x	x
8-bit immediate	x	x	x
16-bit immediate	x	x	x
8-bit relative	x	x	x
8-bit index	x	x	x
16-bit index			x
Bit		x	x
8-bit indirect			x
16-bit idirect		x	x
16-bit computed			x
8-bit I/O		x	x
16-bit I/O			x

Vi xử lý 16 bit và 32 bit có số lượng các kiểu truy xuất bộ nhớ rất lớn, tuỳ thuộc vào yêu cầu điều khiển mà chọn vi xử lý thích hợp.

7. **Các mạch điện giao tiếp bên ngoài của vi xử lý**:

Ngoài giao tiếp với bộ nhớ, vi xử lý có các mạch điện giao tiếp với các mạch điện bên ngoài để điều khiển hay mở rộng khả năng điều khiển. Các mạch điện bên ngoài là các IC và được gọi là IC ngoại vi. Mỗi IC ngoại vi có một chức năng riêng, tùy thuộc vào yêu cầu điều khiển mà chọn các IC ngoại vi.

Bảng danh sách sau đây trình bày các IC ngoại vi có thể giao tiếp với Z80:

Mã số IC	Chức năng	dạng võ
8410	Direct memory access controller	40 pin -DIP
8420	Parallel input/output controller	40 pin -DIP
8430	Counter timer circuit	28 pin -DIP
8440	Serial input/output controller	40 pin -DIP
8470	Dual channel asynchronous receiver transmitter	40 pin -DIP
8530	Serial communications controller	40 pin -DIP

Bảng danh sách sau đây trình bày các IC ngoại vi có thể giao tiếp với 8088/80286:

Mã số IC	Chức năng	dạng võ
8087/80287	Arithmetic coprocessor	40 pin -DIP
8116	Dual baud rate clock generator (programmable)	18 pin -DIP
8202	Dynamic RAM controller	40 pin -DIP
8224	Clock generator/driver	16 pin -DIP
8250	Asynchronous communications element	40 pin -DIP
8253	Programmable interval timer	24 pin -DIP
8272	Floppy disk controller	40 pin -DIP

Chương 2

Cấu trúc bên trong và lệnh của vi xử lý

I. Cấu trúc bên trong của vi xử lý:

1. Sơ đồ khối cấu trúc bên trong của vi xử lý.
2. Chức năng của khối ALU.
3. Các thanh ghi bên trong của vi xử lý.
4. Thanh ghi Accumulator.
5. Thanh ghi bộ đếm chương trình PC.
6. Thanh ghi trạng thái.
7. Thanh ghi con trỏ ngăn xếp.
8. Thanh ghi địa chỉ bộ nhớ.
9. Thanh ghi lệnh.
10. Thanh ghi chứa dữ liệu tạm thời.
11. Khối điều khiển logic và khối giải mã lệnh.
12. Bus dữ liệu bên trong của vi xử lý.

II. Giới thiệu các lệnh của vi xử lý:

1. Giới thiệu về tập lệnh của vi xử lý.
2. Từ gợi nhớ.
3. Các nhóm lệnh cơ bản của vi xử lý
4. Các kiểu truy xuất địa chỉ của vi xử lý.

III. Tóm tắt – câu hỏi ôn tập – bài tập:

1. Tóm tắt.
2. Câu hỏi ôn tập và bài tập.

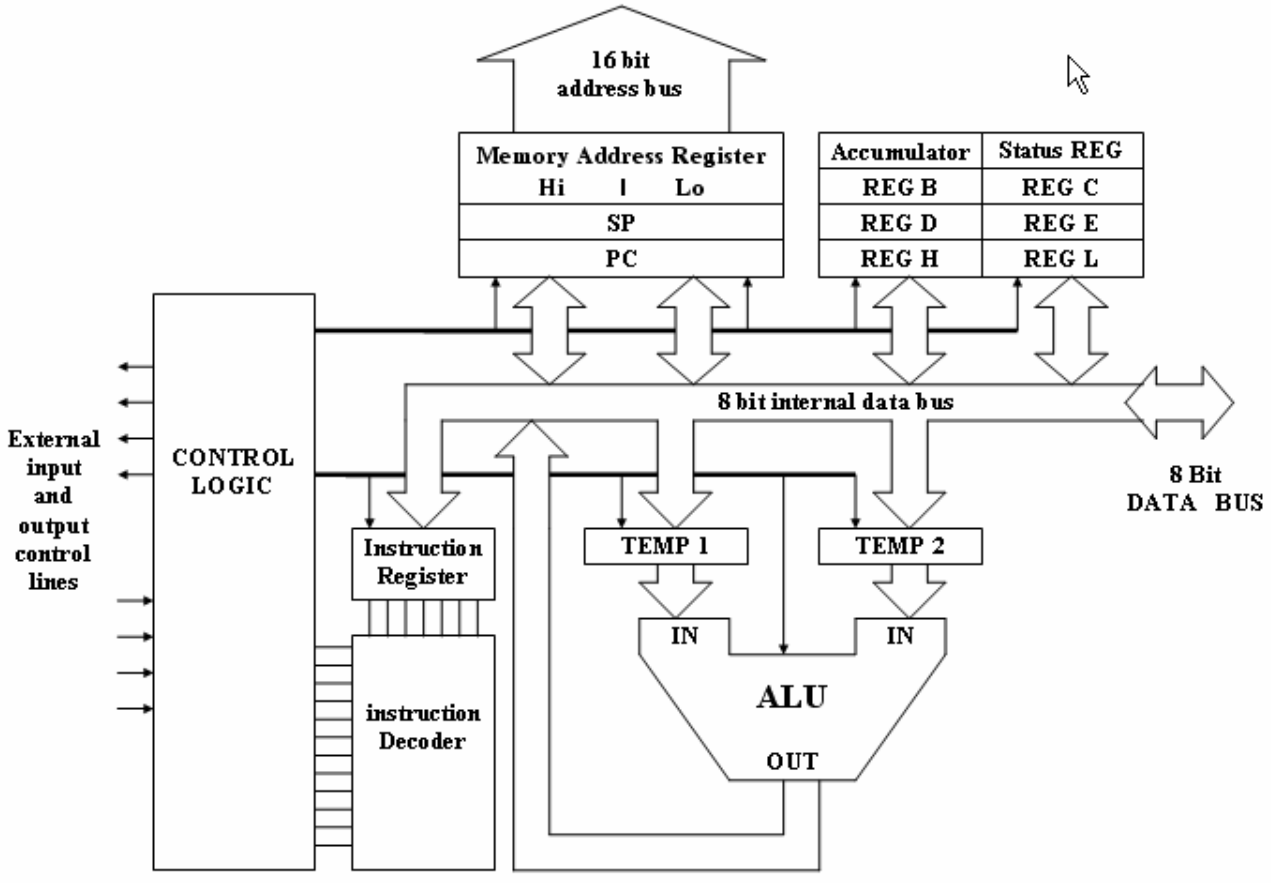
I. CẤU TRÚC BÊN TRONG CỦA VI XỬ LÝ:

1. Sơ đồ khối của vi xử lý:

Cấu trúc của tất cả các vi xử lý đều có các khối cơ bản giống nhau như ALU, các thanh ghi, khối điều khiển là các mạch logic. Để nắm rõ nguyên lý làm việc của vi xử lý cần phải khảo sát nguyên lý kết hợp các khối với nhau để xử lý một chương trình.

Sơ đồ khối của vi xử lý sẽ trình bày cấu trúc của một vi xử lý. Mỗi một vi xử lý khác nhau sẽ có cấu trúc khác nhau. Ví dụ vi xử lý 8 bit sẽ có cấu trúc khác với vi xử lý 16 bit...

Với mỗi vi xử lý đều có một sơ đồ cấu trúc bên trong và được cho trong các sổ tay của nhà chế tạo. Sơ đồ cấu trúc ở dạng khối rất tiện lợi và dễ trình bày nguyên lý hoạt động của vi xử lý. Hình 2-1 trình bày sơ đồ khối của vi xử lý 8 bit:



Hình 2-1. Sơ đồ cấu trúc bên trong của vi xử lý.

Trong sơ đồ khối của vi xử lý bao gồm các khối chính như sau: khối ALU, các thanh ghi và khối control logic. Ngoài ra sơ đồ khối còn trình bày các đường truyền tải tín hiệu từ nơi này đến nơi khác bên trong và bên ngoài hệ thống.

2. Chức năng của khối ALU:

ALU là khối quan trọng nhất của vi xử lý, khối ALU chứa các mạch điện logic chuyên về xử lý dữ liệu. Khối ALU có 2 ngõ vào có tên là "IN" chính là các ngõ vào dữ liệu cho ALU xử lý và 1 ngõ ra có tên là "OUT" chính là ngõ ra kết quả dữ liệu sau khi ALU xử lý xong.

Dữ liệu trước khi vào ALU được chứa ở thanh ghi tạm thời (Temporarily Register) có tên là **TEMP 1** và **TEMP 2**. Bus dữ liệu bên trong vi xử lý được kết nối với 2 ngõ vào "IN" của ALU thông qua 2 thanh ghi tạm thời. Việc kết nối này cho phép ALU có thể lấy bất kỳ dữ liệu nào trên bus dữ liệu bên trong vi xử lý.

Thường thì ALU luôn lấy dữ liệu từ một thanh ghi đặc biệt có tên là accumulator (A). Ngõ ra OUT của ALU cho phép ALU có thể gởi kết dữ liệu sau khi xử lý xong lên bus dữ liệu bên trong vi xử lý, do đó thiết bị nào kết nối với bus bên trong đều có thể nhận dữ liệu này. Thường thì ALU gởi dữ liệu sau khi xử lý xong tới thanh ghi Accumulator.

Ví dụ khi ALU cộng 2 dữ liệu thì một trong 2 dữ liệu được chứa trong thanh ghi Accumulator, sau khi phép cộng được thực hiện bởi ALU thì kết quả sẽ gởi trở lại thanh ghi Accumulator và lưu trữ ở thanh ghi này.

ALU xử lý một dữ liệu hay 2 dữ liệu tùy thuộc vào lệnh hay yêu cầu điều khiển, ví dụ khi cộng 2 dữ liệu thì ALU sẽ xử lý 2 dữ liệu và dùng 2 ngõ vào "IN" để nhập dữ liệu, khi tăng một dữ liệu nào đó lên 1 đơn vị hay lấy bù một một dữ liệu, khi đó ALU chỉ xử lý 1 dữ liệu và chỉ cần một ngõ vào "IN".

Khối ALU có thể thực hiện các phép toán xử lý như sau:

Add	Complement	OR	Exclusive OR
Subtract	Shift right	Increment	
AND	Shift left	Decrement	

Tóm Tắt: Chức năng chính của khối ALU là làm thay đổi dữ liệu hay chuyên về xử lý dữ liệu nhưng không lưu trữ dữ liệu. Để hiểu rõ thêm chức năng đặc biệt của ALU cần phải khảo sát một vi xử lý cụ thể.

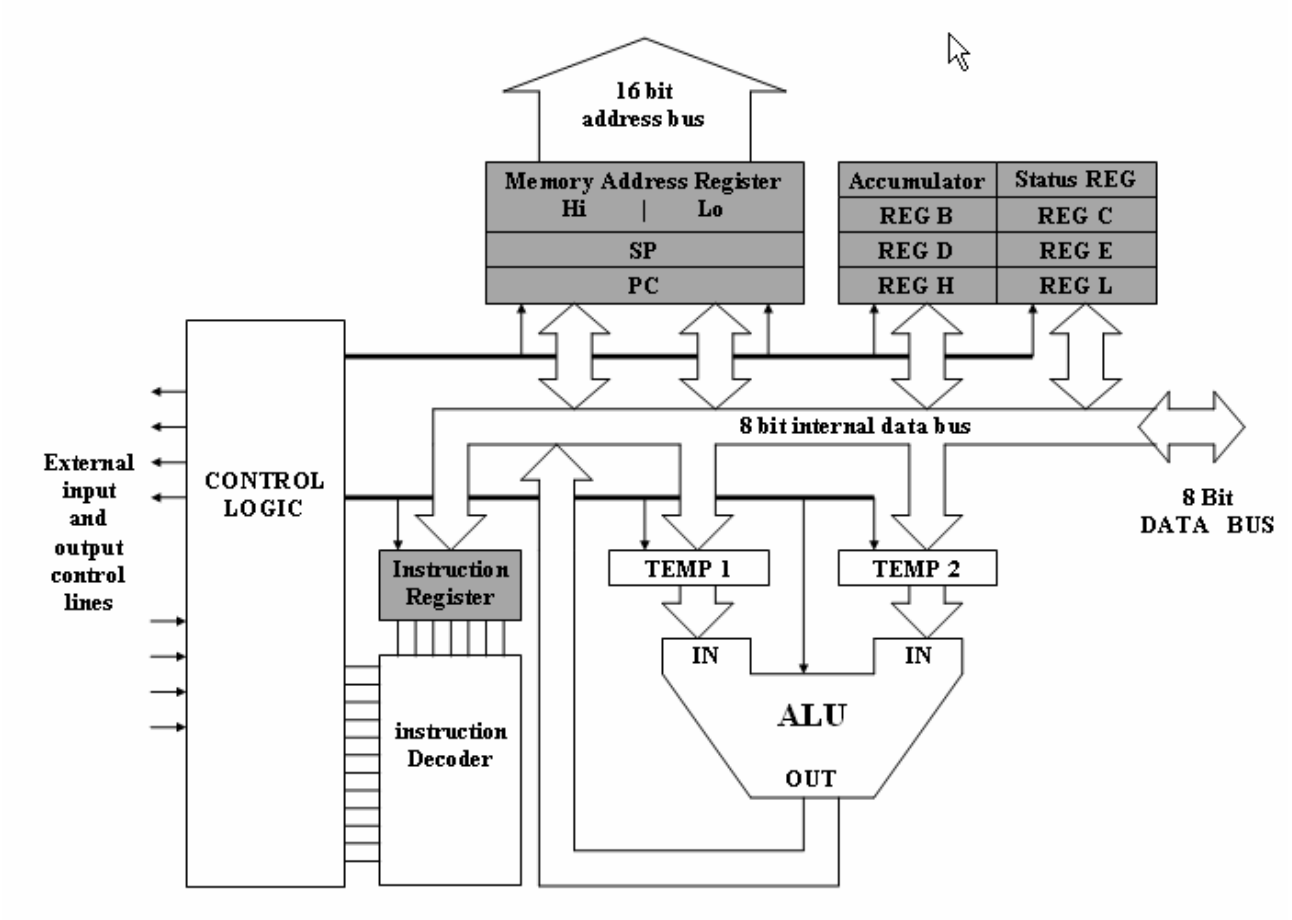
3. Các thanh ghi bên trong của vi xử lý:

Các thanh ghi bên trong có chức năng lưu trữ tạm thời các dữ liệu khi xử lý. Trong số các thanh ghi có một vài thanh ghi đặc biệt khi thực hiện các lệnh đặc biệt, các thanh ghi còn lại gọi là các thanh ghi thông dụng. Với sơ đồ khối minh họa ở trên, các thanh ghi thông dụng có tên Reg B, Reg C, Reg D, Reg E.

Các thanh ghi thông dụng rất hữu dụng cho người lập trình dùng để lưu trữ dự liệu phục vụ cho công việc xử lý dữ liệu và điều khiển, khi viết chương trình chúng ta luôn sử dụng các thanh ghi này. Số lượng các thanh ghi thông dụng thay đổi tùy thuộc vào từng vi xử lý.

Số lượng và cách sử dụng các thanh ghi thông dụng tùy thuộc vào cấu trúc của từng vi xử lý, nhưng chúng có một vài điểm cơ bản giống nhau. Càng nhiều thanh ghi thông dụng thì vấn đề lập trình đơn giản hơn.

Các thanh ghi cơ bản luôn có trong một vi xử lý là thanh ghi A (Accumulator register), thanh ghi bộ đếm chương trình PC (Program Counter register), thanh ghi con trỏ ngăn xếp SP (Stack pointer register), thanh ghi trạng thái F (Status register –Flag register), các thanh ghi thông dụng, thanh ghi lệnh IR (Instruction register), thanh ghi địa chỉ AR (address register).



Hình 2-2. Sơ đồ minh họa các thanh ghi bên trong của Microprocessor được tô đậm.

4. Thanh ghi Accumulator:

Thanh ghi A là một thanh ghi quan trọng của vi xử lý có chức năng lưu trữ dữ liệu khi tính toán. Hầu hết các phép toán số học và các phép toán logic đều xảy ra giữa ALU và Accumulator.

Ví dụ khi thực hiện một lệnh cộng 1 dữ liệu A với một dữ liệu B, thì một dữ liệu phải chứa trong thanh ghi Accumulator giả sử là A, sau đó sẽ thực hiện lệnh cộng dữ liệu A (chứa trong Accumulator) với dữ liệu B (có thể chứa trong ô nhớ hoặc trong một thanh ghi thông dụng), kết quả của lệnh cộng là dữ liệu C sẽ được đặt trong thanh ghi A thay thế cho dữ liệu A trước đó.

Chú ý: Kết quả sau khi thực hiện ALU thường gởi vào thanh ghi Accumulator làm cho dữ liệu trước đó chứa trong Accumulator sẽ mất.

Một chức năng quan trọng khác của thanh ghi Accumulator là để truyền dữ liệu từ bộ nhớ hoặc từ các thanh ghi bên trong của vi xử lý ra các thiết bị điều khiển bên ngoài thì dữ liệu đó phải chứa trong thanh ghi Accumulator.

Thanh ghi Accumulator còn nhiều chức năng quan trọng khác sẽ được thấy rõ qua tập lệnh của một vi xử lý cụ thể, số bit của thanh ghi Accumulator chính là đơn vị đo của vi xử lý, vi xử lý 8 bit thì thanh ghi Accumulator có độ dài 8 bit.

5. Thanh ghi bộ đếm chương trình PC:

Thanh ghi PC là một thanh ghi có vai trò quan trọng nhất của vi xử lý. Chương trình là một chuỗi các lệnh nối tiếp nhau trong bộ nhớ của vi xử lý, các lệnh này sẽ yêu cầu vi xử lý thực hiện chính xác các công việc để giải quyết một vấn đề.

Từng lệnh phải đơn giản và chính xác và các lệnh phải theo đúng một trình tự để chương trình thực hiện đúng. Chức năng của thanh ghi PC là quản lý lệnh đang thực hiện và lệnh sẽ được thực hiện tiếp theo.

Thanh ghi PC trong vi xử lý có chiều dài từ dữ liệu lớn hơn chiều dài từ dữ liệu của vi xử lý. Ví dụ đối với các vi xử lý 8 bit có thể giao tiếp với 65536 ô nhớ thì thanh ghi PC phải có chiều dài là 16 bit để có thể truy xuất từng ô nhớ bắt đầu từ ô nhớ thứ 0 đến ô nhớ thứ 65535.

Chú ý nội dung chứa trong thanh ghi PC cũng chính là nội dung chứa trong thanh ghi địa chỉ.

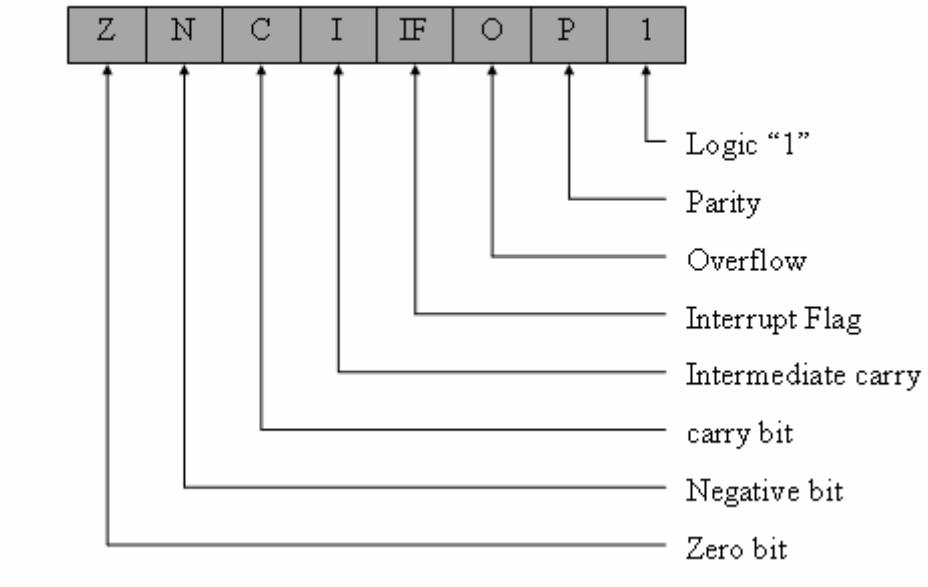
Trước khi vi xử lý thực hiện một chương trình thì thanh ghi PC phải được nạp một con số : "***Đó chính là địa chỉ của ô nhớ chứa lệnh đầu tiên của chương trình***".

Địa chỉ của lệnh đầu tiên được gởi đến IC nhớ thông qua bus địa chỉ 16 bit. Sau đó bộ nhớ sẽ đặt nội dung của ô nhớ lên bus dữ liệu, nội dung này chính là mã lệnh, quá trình này gọi là đón lệnh từ bộ nhớ.

Tiếp theo vi xử lý tự động tăng nội dung của thanh ghi PC để chuẩn bị đón lệnh kế. PC chỉ được tăng khi vi xử lý bắt đầu thực hiện lệnh được đón trước đó. Lệnh đang thực hiện có chiều dài bao nhiêu byte thì thanh ghi PC tăng lên đúng bấy nhiêu byte.

Một vài lệnh trong chương trình có thể nạp vào thanh ghi PC một giá trị mới, khi lệnh làm thay đổi thanh ghi PC sang giá trị mới được thực hiện thì lệnh kế có thể xảy ra ở một địa chỉ mới – đối với các lệnh nhảy hoặc lệnh gọi chương trình con.

6. Thanh ghi trạng thái (Status Register):



Hình 2-3. Cấu trúc của một thanh ghi trạng thái.

Thanh ghi trạng thái còn được gọi là thanh ghi cờ (Flag register) dùng để lưu trữ kết quả của một số lệnh kiểm tra. Việc lưu trữ các kết quả kiểm tra cho phép người lập trình thực hiện việc rẽ nhánh trong chương trình. Khi rẽ nhánh, chương trình sẽ bắt đầu tại một vị trí

mới. Trong trường hợp rẽ nhánh có điều kiện thì chương trình rẽ nhánh chỉ được thực hiện khi kết quả kiểm tra đúng điều kiện. Thanh ghi trạng thái sẽ lưu trữ các kết quả kiểm tra này.

Các bit thường có trong một thanh ghi trạng thái được trình bày ở hình 2-3.

Các lệnh xảy ra trong khối ALU thường ảnh hưởng đến thanh ghi trạng thái, ví dụ khi thực hiện một lệnh cộng 2 dữ liệu 8 bit, nếu kết quả lớn hơn 11111111_2 thì bit carry sẽ mang giá trị là 1. Ngược lại nếu kết quả của phép cộng nhỏ hơn 11111111_2 thì bit carry bằng 0. Ví dụ lệnh tăng hay giảm giá trị của một thanh ghi, nếu kết quả trong thanh ghi khác 0 thì bit Z luôn bằng 0, ngược lại nếu kết quả bằng 0 thì bit Z bằng 1.

Ví dụ về rẽ nhánh khi kiểm tra bit trong thanh ghi trạng thái: hãy viết một chương trình giảm giá trị của một thanh ghi có giá trị là 10.

1. Nạp vào thanh ghi một số nhị phân có giá trị là 10.
2. Giảm nội dung của thanh ghi đi 1.
3. Kiểm tra bit Zero của thanh ghi trạng thái có bằng 1 hay không ?
4. Nếu không nhảy đến thực hiện tiếp lệnh ở bước 2
5. Nếu đúng kết thúc chương trình.

Ý nghĩa của các bit trong thanh ghi trạng thái:

[a]. Carry/borrow (cờ tràn/mượn): là bit carry khi thực hiện một phép cộng có giá trị tùy thuộc vào kết quả của phép cộng. Kết quả tràn thì bit carry =1, ngược lại bit carry = 0. Là bit borrow khi thực hiện một phép trừ: nếu số bị trừ lớn hơn số trừ thì bit borrow = 0, ngược lại bit borrow =1. Bit carry hay bit borrow là 1 bit chỉ được phân biệt khi thực hiện lệnh cụ thể.

[b]. Zero: bit Z bằng một khi kết quả của phép toán bằng 0, ngược lại bit Z=1.

[c]. Negative (cờ số âm): bit N = 1 khi bit MSB của thanh ghi có giá trị là 1, ngược lại N=0.

[d]. Intermediate carry (cờ tràn phụ): giống như bit Carry nhưng chỉ có tác dụng đối với phép cộng hay trừ 4 bit thấp.

[e]. Interrupt Flag (cờ báo ngắt): Bit IF có giá trị là 1 khi người lập trình muốn cho phép ngắt, ngược lại thì không cho phép ngắt.

[f]. Overflow (cờ tràn số có dấu): bit này bằng 1 khi bit tràn của phép toán cộng với bit dấu của dữ liệu.

[g]. Parity (cờ chẵn lẻ): bit này có giá trị là 1 khi kết quả của phép toán là số chẵn, ngược lại là số lẻ thì bit P = 0.

Số lượng các bit có trong thanh ghi trạng thái tùy thuộc vào từng vi xử lý. Trong một số vi xử lý có thể xóa hoặc đặt các bit của thanh ghi trạng thái.

7. Thanh ghi con trỏ ngăn xếp (Stack pointer SP):

Thanh ghi con trỏ ngăn xếp là một thanh ghi quan trọng của vi xử lý, độ dài từ dữ liệu của thanh ghi SP bằng thanh ghi PC, chức năng của thanh ghi SP gần giống như thanh ghi PC nhưng nó dùng để quản lý bộ nhớ ngăn xếp khi muốn lưu trữ tạm thời dữ liệu vào ngăn xếp.

Giống như thanh ghi PC, thanh ghi SP cũng tự động chỉ đến ô nhớ kế. Trong hầu hết các vi xử lý, thanh ghi SP giảm (để chỉ đến ô nhớ tiếp theo trong ngăn xếp) sau khi thực hiện lệnh cất dữ liệu vào ngăn xếp. Do đó khi thiết lập giá trị cho thanh ghi SP là địa chỉ cuối cùng của bộ nhớ.

Thanh ghi SP phải chỉ đến một ô nhớ do người lập trình thiết lập, quá trình này gọi là khởi tạo con trỏ ngăn xếp. Nếu không khởi tạo, con trỏ ngăn xếp sẽ chỉ đến một ô nhớ ngẫu nhiên. Khi đó dữ liệu cất vào ngăn xếp có thể ghi đè lên dữ liệu quan trọng khác làm chương trình xử lý sai hoặc thanh ghi SP chỉ đến vùng nhớ không phải là bộ nhớ RAM làm chương trình thực hiện không đúng vì không lưu trữ được dữ liệu cần cất tạm vào bộ nhớ ngăn xếp. Tổ chức của ngăn xếp là vào sau ra trước **(LAST IN FIRST OUT : LIFO).**

8. Thanh ghi địa chỉ bộ nhớ:

Mỗi khi vi xử lý truy xuất bộ nhớ thì thanh ghi địa chỉ phải tạo ra đúng địa chỉ mà vi xử lý muốn. Ngõ ra của thanh ghi địa chỉ được đặt lên bus địa chỉ 16 bit. Bus địa chỉ dùng để lựa chọn một ô nhớ hay lựa chọn 1 port Input/Output.

Nội dung của thanh ghi địa chỉ ô nhớ và nội dung của thanh ghi PC là giống nhau khi vi xử lý truy xuất bộ nhớ để đón lệnh, khi lệnh đang được giải mã thì thanh ghi PC tăng lên để chuẩn bị đón lệnh tiếp theo, trong khi đó nội dung của thanh ghi địa chỉ bộ nhớ không tăng, trong suốt chu kỳ thực hiện lệnh, nội dung của thanh ghi địa chỉ phụ thuộc vào lệnh đang được thực hiện, nếu lệnh yêu cầu vi xử lý truy xuất bộ nhớ thì thanh ghi địa chỉ bộ nhớ được dùng lần thứ 2 trong khi thực hiện một lệnh.

Trong tất cả các vi xử lý, thanh ghi địa chỉ bộ nhớ có chiều dài bằng với thanh ghi PC.

9. Thanh ghi lệnh (Instruction Register):

Thanh ghi lệnh dùng để chứa lệnh vi xử lý đang thực hiện.

Một chu kỳ lệnh bao gồm đón lệnh từ bộ nhớ và thực hiện lệnh.

Đầu tiên là lệnh được đón từ bộ nhớ, sau đó PC chỉ đến lệnh kế trong bộ nhớ. Khi một lệnh được đón có nghĩa là dữ liệu trong ô nhớ đó được copy vào vi xử lý thông qua bus dữ liệu đến thanh ghi lệnh. Tiếp theo lệnh sẽ được thực hiện, trong khi thực hiện lệnh bộ giải mã lệnh đọc nội dung của thanh ghi lệnh. Bộ giải mã sẽ giải mã lệnh để báo cho vi xử lý thực hiện chính xác công việc mà lệnh yêu cầu.

Chiều dài từ dữ liệu của thanh ghi lệnh tùy thuộc vào từng vi xử lý.

Thanh ghi lệnh do vi xử lý sử dụng người lập trình không được sử dụng thanh ghi này.

10. Thanh ghi chứa dữ liệu tạm thời (Temporary data register):

Thanh ghi lưu trữ dữ liệu tạm thời dùng để ALU thực hiện các phép toán xử lý dữ liệu. Do ALU chỉ xử lý dữ liệu không có chức năng lưu trữ dữ liệu, bất kỳ dữ liệu nào đưa đến ngõ vào của ALU, lập tức sẽ xuất hiện ở ngõ ra.

Dữ liệu xuất hiện tại ngõ ra của ALU được quyết định bởi lệnh trong chương trình yêu cầu ALU thực hiện. ALU lấy dữ liệu từ bus dữ liệu bên trong vi xử lý, xử lý dữ liệu, sau đó đặt dữ liệu vừa xử lý xong trở lại thanh ghi Accumulator, do đó cần phải có thanh ghi lưu trữ

dữ liệu tạm thời để ALU thực hiện. Người lập trình không được phép xử dụng các thanh ghi tạm thời. Số lượng các thanh ghi này tùy thuộc vào từng vi xử lý cụ thể.

11. Khối điều khiển logic (control logic) và khối giải mã lệnh (Instruction decoder):

Chức năng của khối giải mã lệnh là nhận lệnh từ thanh ghi lệnh sau đó giải mã để gởi tín hiệu điều khiển đến cho khối điều khiển logic.

Chức năng của khối điều khiển logic (control logic) là nhận lệnh hay tín hiệu điều khiển từ bộ giải mã lệnh, sau đó sẽ thực hiện đúng các yêu cầu của lệnh. Khối điều khiển logic được xem là một vi xử lý nhỏ nằm trong một vi xử lý.

Các tín hiệu điều khiển của khối điều khiển logic là các tín hiệu điều khiển bộ nhớ, điều khiển các thiết bị ngoại vi, các đường tín hiệu đọc-ghi...và các tín hiệu điều khiển vi xử lý từ các thiết bị bên ngoài. Các đường tín hiệu này sẽ được trình bày cụ thể trong sơ đồ của từng vi xử lý cụ thể.

Ngõ tín hiệu vào quan trọng nhất của khối điều khiển logic là tín hiệu clock cần thiết cho khối điều khiển logic hoạt động. Nếu không có tín hiệu clock thì vi xử lý không làm việc. Mạch tạo xung clock là các mạch dao động, tín hiệu được đưa đến ngõ vào clock của vi xử lý. Có nhiều vi xử lý có tích hợp mạch tạo dao động ở bên trong, khi đó chỉ cần thêm tụ thạch anh ở bên ngoài.

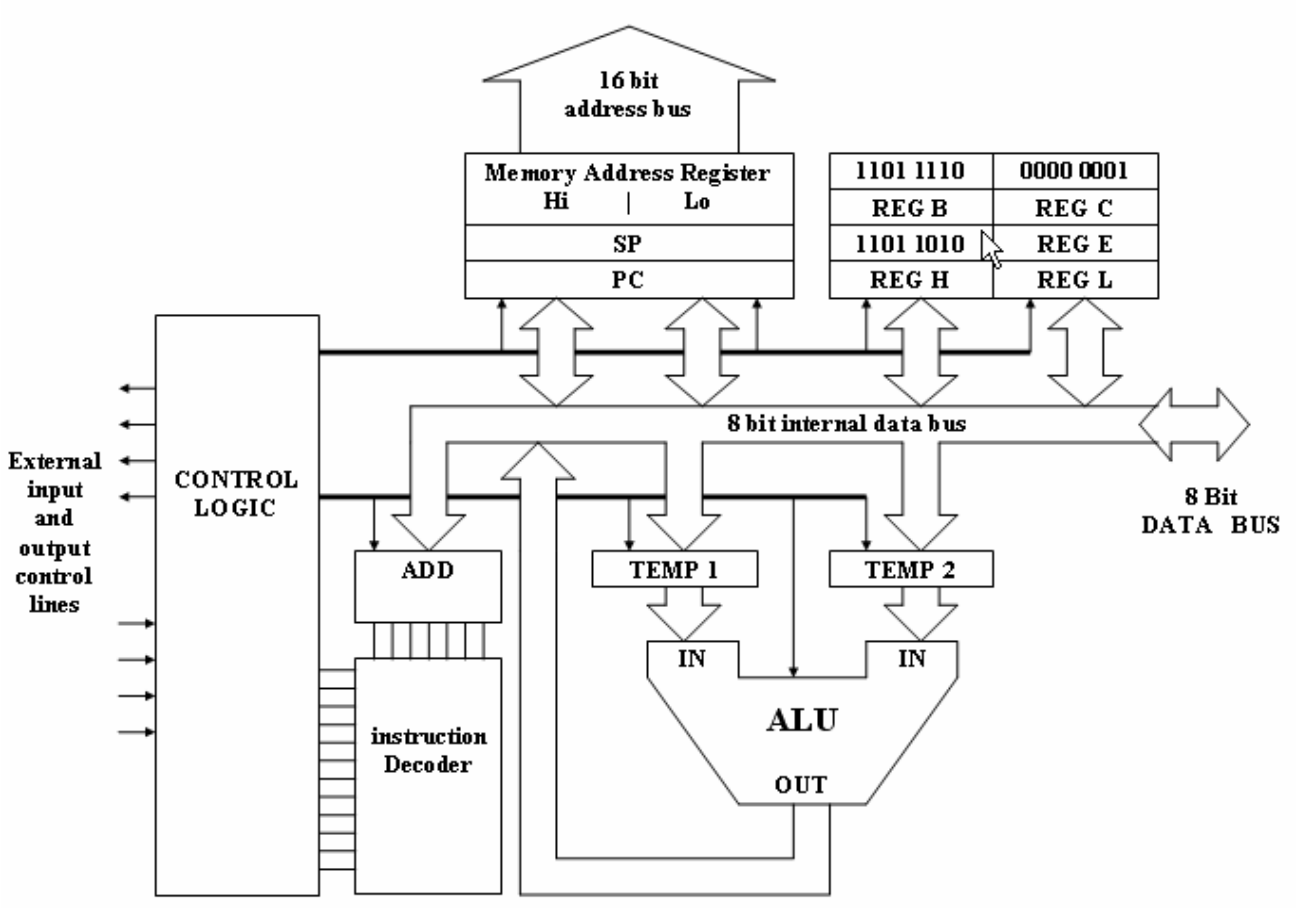
12. Bus dữ liệu bên trong vi xử lý (Internal data bus):

Bus dữ liệu dùng để kết nối các thanh ghi bên trong và ALU với nhau, tất cả các dữ liệu di chuyển trong vi xử lý đều thông qua bus dữ liệu này. Các thanh ghi bên trong có thể nhận dữ liệu từ bus hay có thể đặt dữ liệu lên bus nên bus dữ liệu này là bus dữ liệu 2 chiều. Bus dữ liệu bên trong có thể kết nối ra bus bên ngoài khi vi xử lý cần truy xuất dữ liệu từ bộ nhớ bên ngoài hay các thiết bị IO. Bus dữ liệu bên ngoài cũng là bus dữ liệu 2 chiều vì vi xử lý có thể nhận dữ liệu từ bên ngoài hay gởi dữ liệu ra.

Để biết trình tự làm việc của bus dữ liệu bên trong vi xử lý hoạt động, hãy cho vi xử lý thực hiện một lệnh cộng 2 số nhị phân chứa trong thanh ghi 2 thanh ghi: thanh ghi Accumulator (gọi tắt là A) =1101 1110_2 và thanh ghi D=1101 1010_2.

Trình tự cộng như sau:

- Trước khi thực hiện lệnh cộng, nội dung của 2 thanh ghi phải chứa 2 dữ liệu và 2 thanh ghi này có thể đang kết nối với các thiết bị khác. Để thực hiện lệnh cộng nội dung 2 thanh ghi A và D thì thanh ghi lệnh phải mang đúng mã lệnh của phép cộng này và giả sử mã lệnh đó là ADD. Được trình bày ở hình 2-4.

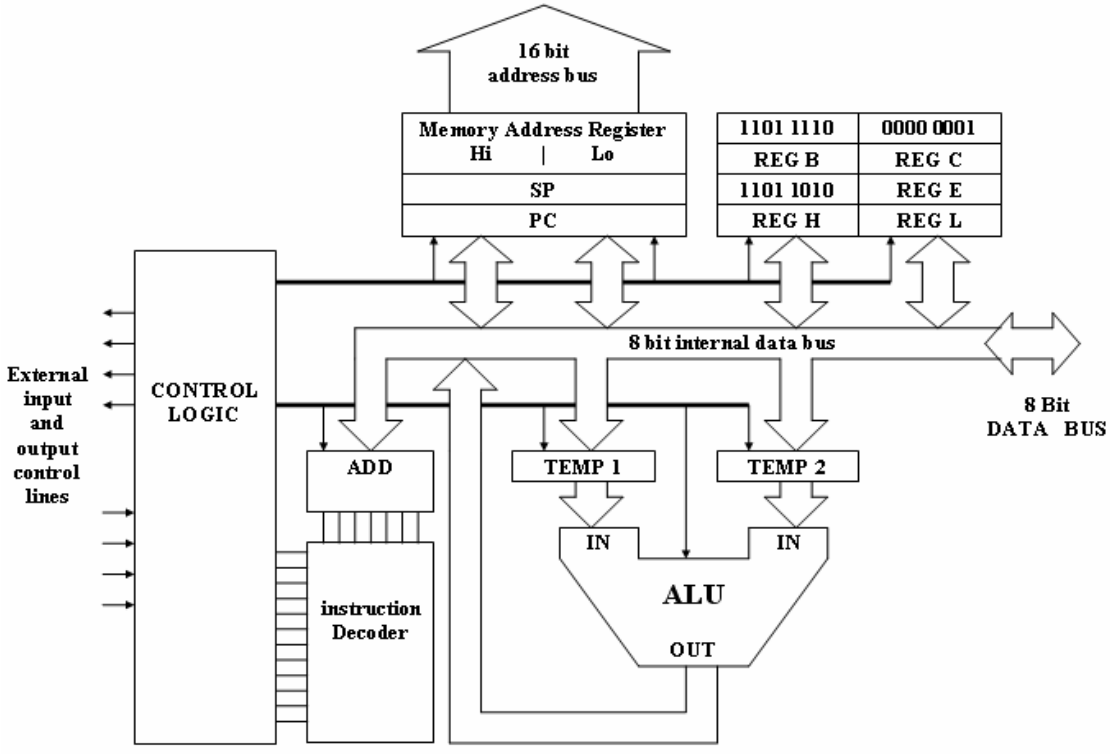


Hình 2-4.

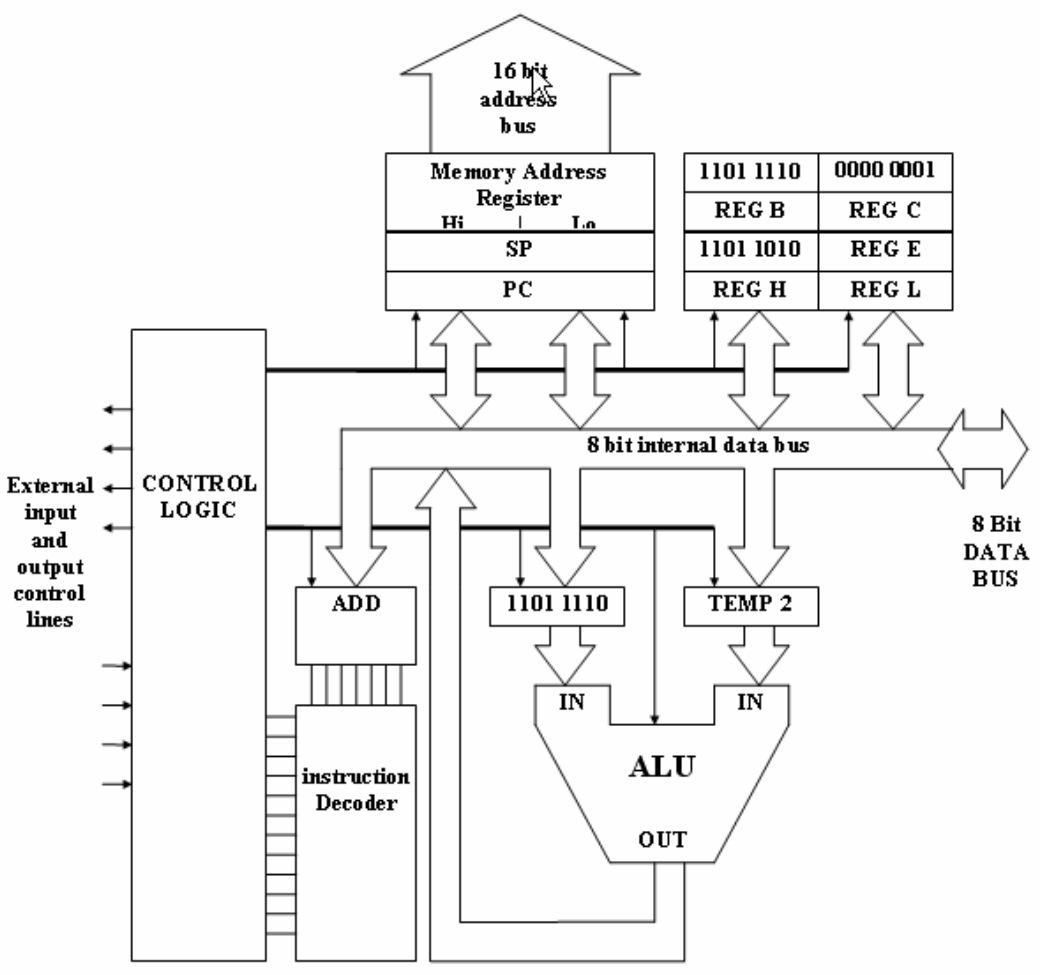
- Dữ liệu của thanh ghi A được đặt lên bus dữ liệu bên trong vi xử lý, một trong 2 thanh ghi lưu trữ dữ liệu tạm thời được kết nối với Bus dữ liệu. Thanh ghi tạm thời sẽ copy dữ liệu chứa trong thanh ghi A. Chỉ có thanh ghi A và thanh ghi tạm thời được kết nối với bus tại thời điểm này. Xem hình 2-5
- Dữ liệu của thanh ghi D được kết nối với bus dữ liệu và thanh ghi tạm thời còn lại cũng được phép kết nối với bus dữ liệu. Thanh ghi tạm thời sẽ copy nội dung của thanh ghi D. Chỉ có thanh ghi D và thanh ghi tạm thời được kết nối với bus tại thời điểm này. Xem hình 2-6.
- ALU sẽ cộng trực tiếp 2 dữ liệu tại 2 ngõ vào. Ngõ ra của ALU được kết nối với thanh ghi A, kết quả của phép cộng được nạp vào thanh ghi A. Xem hình 2-7.
- Sau khi đặt kết quả vào trong thanh ghi A và cặp nhật sự thay đổi các bit trong thanh ghi trạng thái thì sự kết nối giữa thanh A và khối ALU chấm dứt, các thanh ghi tạm thời trở lại trạng thái sẵn sàng cho lệnh tiếp theo. Xem hình 2-8

Phép cộng 2 số nhị phân:

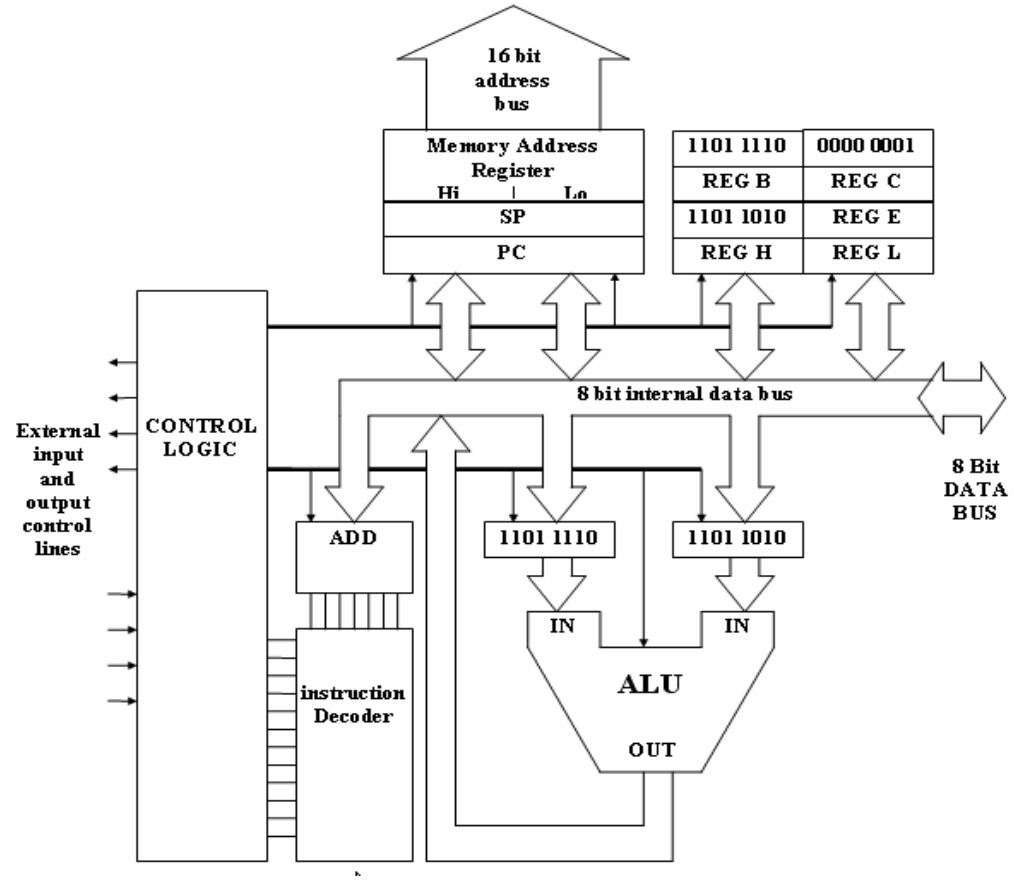
```
    1101 1110
+   1101 1010
1  1011 1000
Carry  negative
```



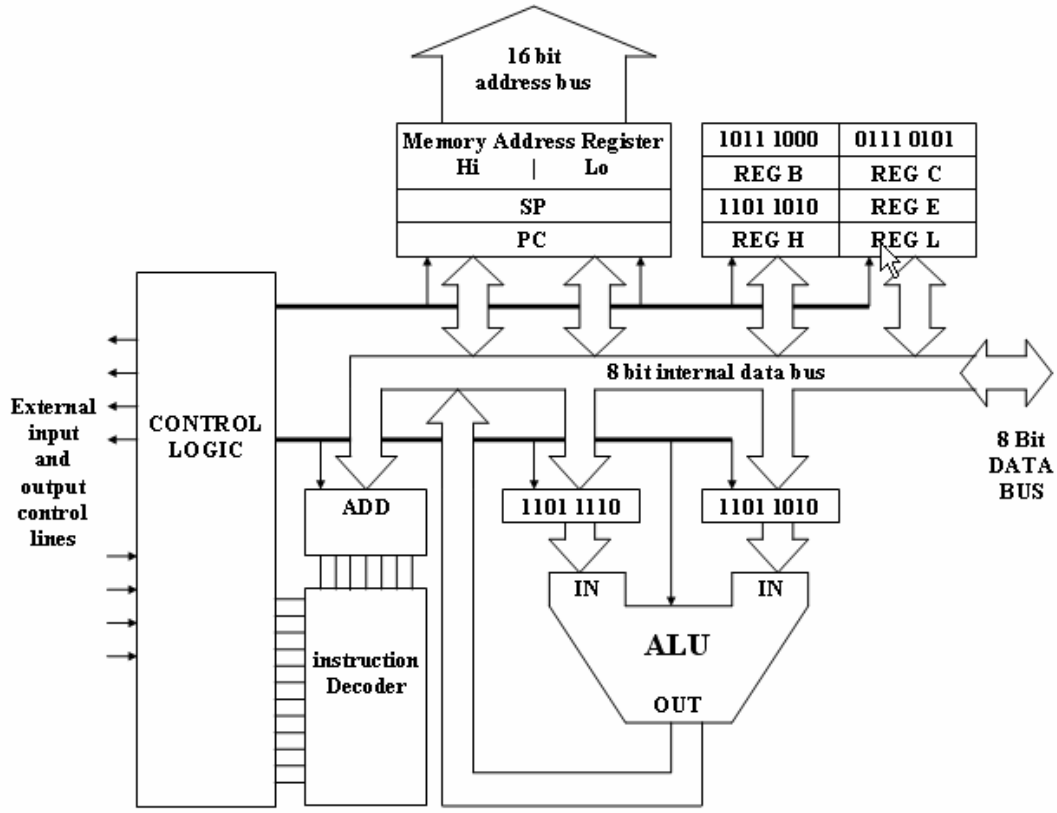
Hình 2-5.



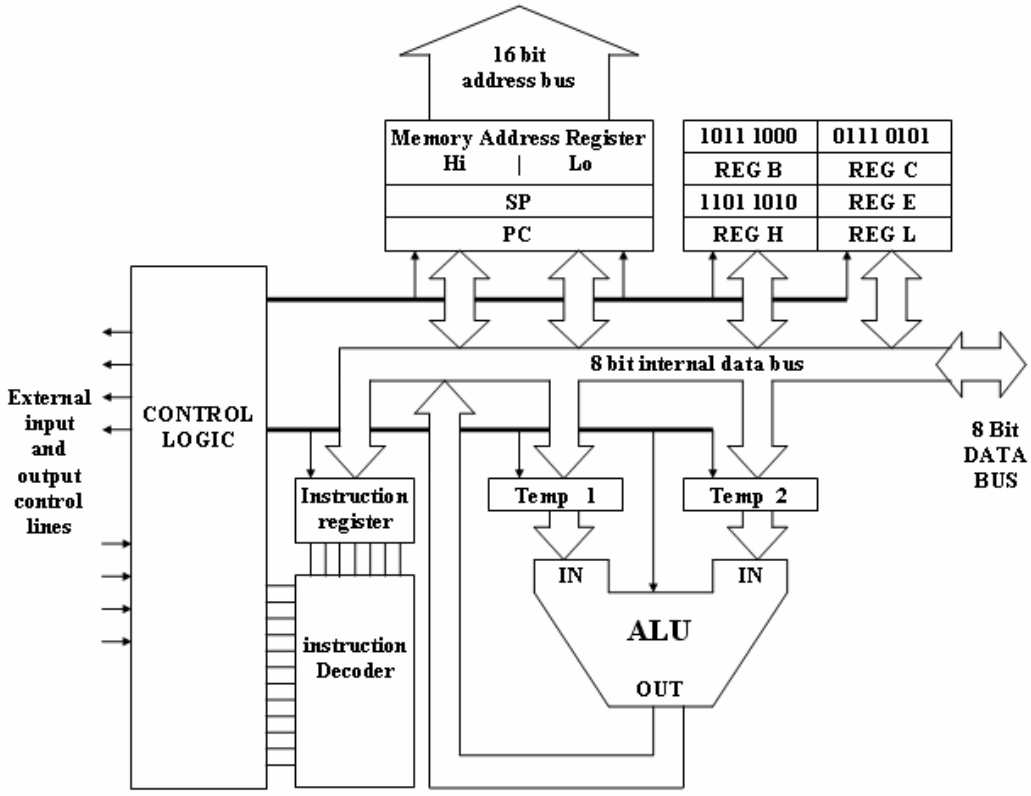
Hình 2-6.



Hình 2-7.



Hình 2-8.



Hình 2-9.

II. CÁC LỆNH CỦA VI XỬ LÝ:

1. Tập lệnh của vi xử lý:

Lệnh của vi xử lý là một dữ liệu số nhị phân, khi vi xử lý đọc một lệnh thì từ dữ liệu nhị phân này sẽ yêu cầu vi xử lý làm một công việc đơn giản. Mỗi một từ dữ liệu tương đương với một công việc mà vi xử lý phải làm. Hầu hết các lệnh của vi xử lý là các lệnh chuyển dữ liệu và xử lý dữ liệu.

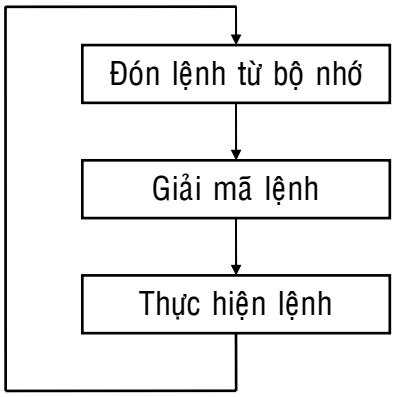
Khi nói đến tập lệnh của vi xử lý tức nói đến tất cả các lệnh mà vi xử lý có thể hiểu và thực hiện được.

Nếu tập lệnh của một vi xử lý giống với tập lệnh của một vi xử lý khác thì cấu trúc của 2 vi xử lý giống nhau.

Độ dài của một lệnh bằng với độ dài từ dữ liệu của vi xử lý, đối với vi xử lý 8 bit thì độ dài của một lệnh là 8 bit, đối với vi xử lý 16 bit thì độ dài của một lệnh là 16 bit...

Trong chu kỳ đón lệnh, mã lệnh sẽ được gởi đến thanh ghi lệnh, bộ giải mã lệnh, và bộ điều khiển logic của vi xử lý. Chức năng của các khối sẽ xác định lệnh này làm gì và sẽ gởi các tín hiệu điều khiển đến các mạch điện logic khác trong vi xử lý, các tín hiệu logic này sẽ thực hiện đúng chức năng mà lệnh yêu cầu.

Hình 2-10 minh họa chu kỳ thực hiện lệnh:



Hình 2-10. Chu kỳ thực hiện lệnh của vi xử lý.

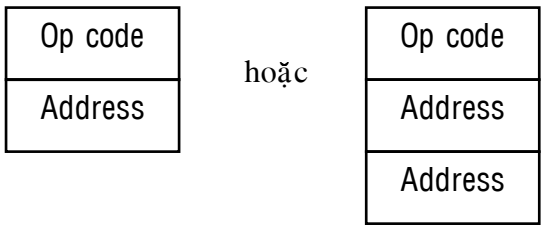
Một lệnh được thực hiện cần phải hội đủ 2 yếu tố:

Yếu tố thứ nhất là lệnh sẽ yêu cầu vi xử lý thực hiện công việc gì. Ví dụ yêu cầu vi xử lý thực hiện một lệnh cộng: ADD, một lệnh dịch chuyển dữ liệu MOV... là những lệnh mà vi xử lý có thực hiện được.

Yếu tố thứ hai là lệnh phải cho vi xử lý biết các thông tin địa chỉ tức là vị trí của các dữ liệu mà vi xử lý phải thực hiện. Ví dụ khi thực hiện một lệnh cộng nội dung 2 thanh ghi A và B, hoặc nội dung thanh ghi A và dữ liệu chứa trong một ô nhớ. Yếu tố thứ 2 trong trường hợp này là các thanh ghi A và B, hoặc thanh ghi A và địa chỉ của ô nhớ.

Yếu tố thứ nhất gọi là mã lệnh : op code (operation code) và yếu tố thứ 2 gọi là địa chỉ. Mã lệnh sẽ báo cho vi xử lý làm gì và địa chỉ sẽ cho vi xử lý biết vị trí của dữ liệu.

Sơ đồ hình 2-11 minh họa cho cấu trúc 1 lệnh.



Hình 2-11. Cấu trúc của một lệnh bao gồm mã lệnh và địa chỉ.

Từ dữ liệu đầu tiên luôn là mã lệnh, các từ dữ liệu tiếp theo là địa chỉ. Đối với các lệnh chỉ có một từ dữ liệu thì địa chỉ đã được hiểu ngầm.

Do có nhiều cách chỉ cho vi xử lý biết địa chỉ của dữ liệu được gọi là các kiểu truy xuất bộ nhớ. Khi sử dụng một vi xử lý cần phải biết các kiểu truy xuất này.

2. Từ gợi nhớ (Mnemonics):

Một lệnh của vi xử lý là các con số nhị phân. Đối với lệnh chỉ có một byte thì rất khó nhớ, nếu lệnh dài 2, 3, 4 hoặc nhiều hơn nữa thì không thể nào nhớ hết. Để giảm bớt sự phức tạp của số nhị phân có thể dùng số Hex để thay thế, khi đó các lệnh dễ viết và dễ đọc hơn

nhiều nhưng cũng không thể nào giúp người sử dụng nhớ hết được và quan trọng nhất là khi viết chương trình cũng như lúc gỡ rối chương trình.

Để giải quyết vấn đề này lệnh được viết thành các từ gợi nhớ rất gần với chức năng và ý nghĩa của các lệnh. Trong hầu hết các từ gợi nhớ của lệnh, mã lệnh được rút gọn chỉ còn khoảng 3 ký tự. Ví dụ lệnh di chuyển dữ liệu có từ gợi nhớ là MOV, lệnh trừ là SUB... Khi lệnh có các địa chỉ đi sau thì các địa chỉ này vẫn là các con số. Ví dụ lệnh nhảy đến một ô nhớ viết như sau:

JMP $FA90_H$.

Khi sử các từ gợi nhớ này giúp người lập trình rất dễ nhớ tất cả các lệnh của vi xử lý, khi viết chương trình người lập trình dùng các từ gợi nhớ để viết chương trình, các từ gợi nhớ này tạo thành một ngôn ngữ gọi là Assembly – khi viết chương trình bằng ngôn ngữ Assembly thì vi xử lý sẽ không hiểu – muốn vi xử lý hiểu và thực hiện chương trình thì phải sử dụng chương trình biên dịch Assembler để chuyển các lệnh viết dưới dạng ngôn ngữ Assembly thành các lệnh dạng số nhị phân và các địa chỉ dạng nhị phân tương ứng rồi nạp vào bộ nhớ thì vi xử lý mới có thể thực hiện được.

3. Các nhóm lệnh cơ bản của vi xử lý:

Đối với hầu hết các vi xử lý tập lệnh được chia ra làm 9 nhóm lệnh cơ bản:

- Nhóm lệnh truyền dữ liệu: Data transfers.
- Nhóm lệnh trao đổi, truyền khối dữ liệu, lệnh tìm kiếm: Exchanges, Block transfers, Searches.
- Nhóm lệnh số học và logic: arithmetic and logic.
- Nhóm lệnh xoay và dịch: Rotates and shifts.
- Nhóm lệnh điều khiển CPU.
- Nhóm lệnh về bit: Bit set, bit reset, and bit test.
- Nhóm lệnh nhảy: Jumps.
- Nhóm lệnh gọi, trở về và nhóm lệnh bắt đầu: Calls, Return, and Restarts.
- Nhóm lệnh xuất nhập: Input and Output.

Các mã gợi nhớ và các mã nhị phân của tất cả các lệnh sẽ được cho trong các sổ tay của nhà chế tạo đối với từng vi xử lý cụ thể.

4. Các kiểu truy xuất địa chỉ của một vi xử lý:

Như đã trình bày ở các phần trên, vi xử lý có thể truy xuất bộ nhớ bằng nhiều cách để lấy dữ liệu. Vi xử lý có nhiều cách truy xuất thì chương trình khi viết sẽ càng ngắn gọi rất có lợi cho người lập trình và làm giảm thời gian thực hiện chương trình.

Chú Ý: Danh từ truy xuất bộ nhớ có nghĩa là tạo ra địa chỉ để truy xuất dữ liệu, vi xử lý truy xuất dữ liệu có thể là lấy dữ liệu từ ô nhớ này hoặc lưu trữ dữ liệu vào ô nhớ này. Có thể gọi là các kiểu địa chỉ hóa bộ nhớ hay các kiểu tạo địa chỉ để truy xuất bộ nhớ.

Để biết vi xử lý có bao nhiêu các truy xuất bộ nhớ cần phải khảo sát từng vi xử lý cụ thể. Các kiểu truy xuất này được cho trong các sổ tay chế tạo.

Các kiểu truy xuất địa chỉ cơ bản của một vi xử lý (được gọi tắt là kiểu địa chỉ):

- Kiểu địa chỉ ngầm định (Implied Addressing Mode).
- Kiểu địa chỉ tức thời (Immediate Addressing Mode).
- Kiểu địa chỉ trực tiếp (Direct Addressing Mode).
- Kiểu địa chỉ gián tiếp dùng thanh ghi (Register Indirect Addressing Mode).
- Kiểu địa chỉ chỉ số (Indexed Addressing Mode).
- Kiểu địa chỉ tương đối (Relative Addressing Mode).

[a]. Kiểu địa chỉ ngầm định:

Để hiểu các kiểu truy xuất phải dùng tập lệnh của một vi xử lý 8 bit.

Ví dụ lệnh cộng: ADD reg

Lệnh này được hiểu là nội dung của thanh ghi A được cộng với nội dung của thanh ghi Reg kết quả lưu trữ vào thanh ghi A.

[b]. Kiểu địa chỉ tức thời:

Một lệnh được chia ra làm 2 phần thứ nhất là mã lệnh hay còn gọi là mã công tác, phần thứ 2 là địa chỉ. Đối với kiểu địa chỉ tức thời thì phần thứ 2 là dữ liệu không phải là địa chỉ.

Ví dụ lệnh nạp một dữ liệu tức thời vào thanh ghi A được viết như sau: MVI A, FE_H. Trong đó MVI là mã gợi nhớ, FE là dữ liệu dạng số Hex. Vì thanh ghi A chỉ có 8 bit nên dữ liệu tức thời có độ dài là 8 bit.

[c]. Kiểu địa chỉ trực tiếp:

Ví dụ lệnh di chuyển nội dung của một ô nhớ có địa chỉ 8000_H vào thanh ghi A: LDA 8000_H. LDA là mã gợi nhớ, địa chỉ 8000_H được viết trực tiếp trong câu lệnh, với vi xử lý 8 bit có 16 đường địa chỉ nên phải dùng 4 số Hex để chỉ định một ô nhớ.

Đối với những lệnh dùng kiểu địa chỉ trực tiếp thì lệnh có độ dài là 3 byte: một byte là mã lệnh, 2 byte còn lại là địa chỉ của ô nhớ (đối với vi xử lý 8 bit).

[d]. Kiểu địa chỉ gián tiếp dùng thanh ghi:

Để minh họa kiểu địa chỉ gián tiếp dùng thanh ghi ta dùng lệnh sau:

Ví dụ: MOV A,M. Lệnh này sẽ di chuyển nội dung của ô nhớ M có địa chỉ chứa trong một cặp thanh ghi. Đối với vi xử lý 8085 thì địa chỉ này thường chứa trong cặp thanh ghi HL, vì địa chỉ 16 bit nên phải dùng cặp thanh ghi mới chứa hết 16 bit địa chỉ.

Chú ý khi dùng lệnh kiểu này người lập trình phải quản lý giá trị trong cặp thanh ghi.

[e]. Kiểu địa chỉ chỉ số:

Đối với một vài vi xử lý có các thanh ghi chỉ số (Index register) được dùng cho kiểu địa chỉ chỉ số.

Kiểu địa chỉ này được thực hiện bằng cách cộng byte thứ 2 của lệnh với nội dung của thanh ghi chỉ số ID. Ví dụ: lệnh cộng nội dung thanh ghi A với nội dung của ô nhớ có địa chỉ

chứa trong thanh ghi chỉ số ID với byte dữ liệu thứ 2: ADD A, (ID +n). n là một số có chiều dài 8 bit.

[f]. Kiểu địa chỉ tương đối:

Kiểu địa chỉ này gần giống như kiểu địa chỉ chỉ số nhưng thanh ghi ID được thay thế bằng thanh ghi PC. Địa chỉ của ô nhớ cần truy xuất được tính bằng cách cộng nội dung hiện tại chứa trong thanh ghi PC cộng với byte dữ liệu thứ 2. Ví dụ lệnh JP 05_H : nhảy đến tới thực hiện lệnh có địa chỉ cách bộ đếm chương trình PC là 5 byte.

III. TÓM TẮT - CÂU HỎI ÔN TẬP- BÀI TẬP:

1. Tóm tắt:

1. Một vi xử lý có 3 phần chính: khối ALU, các thanh ghi, và khối control logic.
2. Khối ALU có 2 ngõ vào và một ngõ ra. Một trong 2 port nhận dữ liệu từ data bus, ngõ vào còn lại sẽ nhận dữ liệu từ thanh ghi A. khối ALU có thể xử lý 1 hoặc 2 dữ liệu, chức năng chính của khối ALU là thực hiện các phép toán số học, các phép toán logic và kiểm tra dữ liệu.
3. Thanh ghi trong vi xử lý có thể lưu trữ tạm thời các dữ liệu, các thanh ghi thông dụng, các thanh ghi có chức năng đặc biệt.
4. Tất cả các vi xử lý đều có các thanh ghi cơ bản
   - Accumulator.
   - Program counter.
   - Stack pointer.
   - General purpose registers.
   - Memory address register and logic.
   - Instruction register.
   - Temporary register.
5. Các thanh ghi rất cần thiết cho vi xử lý làm việc. Tuy nhiên người lập trình không thể sử dụng hết tất cả các thanh ghi này.
6. Thanh ghi Accumulator luôn làm việc với khối ALU, Accumulator là một thanh ghi quan trọng của vi xử lý trong xử lý dữ liệu. Chiều dài từ dữ liệu của thanh ghi Accumulator bằng với chiều dài từ dữ liệu của vi xử lý.
7. Thanh ghi Program counter có chức năng tạo địa chỉ để đón lệnh khi vi xử lý thực hiện chương trình, khi vi xử lý đón lệnh xong và thực hiện lệnh thì nội dung của PC tăng lên để chuẩn bị đón lệnh tiếp theo. Thanh ghi PC phải có chiều dài từ dữ liệu hay số bit có khả năng truy xuất hết bộ nhớ.
8. Một chương trình có thể bắt đầu tại bất kỳ vị trí nào trong bộ nhớ và có thể kết thúc tại bất kỳ vị trí nào trong bộ nhớ. Tuy nhiên các lệnh trong chương trình phải theo đúng một trình tự hợp lý.
9. Khi vi xử lý bắt đầu thực hiện chương trình:

- Từng lệnh trong chương trình sẽ được thực hiện theo một trình tự nối tiếp trừ khi có lệnh đặc biệt làm thay đổi trình tự này.
- Khi PC chỉ đến một ô nhớ thì khối control logic sẽ đón lệnh từ ô nhớ này.
- Mỗi khi lệnh được đón, vi xử lý sẽ tăng nội dung của PC để chuẩn bị cho lệnh kế và bắt đầu thực hiện lệnh vừa đón.
- Thanh ghi địa chỉ bộ nhớ chỉ đến mỗi ô nhớ để truy xuất dữ liệu khi vi xử lý yêu cầu.
- Dữ liệu trong thanh ghi địa chỉ bộ nhớ sẽ được gởi ra bus địa chỉ bên ngoài để kết nối với bộ nhớ. Thanh ghi địa chỉ bộ nhớ phải có đủ số bit để có thể truy xuất hết tất cả các ô nhớ mà vi xử lý có thể.

10. Thanh ghi trạng thái sẽ lưu trữ lại các kết quả của một số lệnh, một thanh ghi trạng thái thường có các bit sau: bit zero, bit negative, bit carry, bit haft carry, bit parity, bit overflow, bit interrupt...các bit trạng thái dùng để xác định trạng thái của một số lựa chọn trong chương trình.
11. Thanh ghi SP chỉ đến một ô nhớ dùng để lưu trữ dữ liệu tạm thời. Mỗi khi ngăn xếp dùng để lưu trữ dữ liệu thì giá trị trong SP sẽ giảm để chuẩn bị cho việc lưu trữ dữ liệu tiếp theo.
12. Thanh ghi lệnh lưu trữ lệnh dạng số nhị phân để ra lệnh cho khối control logic thực hiện cái mà lệnh yêu cầu.
13. Khi lệnh được đón từ bộ nhớ có nghĩa là thực hiện một quá trình copy dữ liệu trong ô nhớ của chương trình vào thanh ghi lệnh.
14. Trong quá trình thực hiện lệnh khối control logic và khối giải mã lệnh sẽ đọc lệnh trong thanh ghi lệnh.
15. Thanh ghi tạm thời dùng để lưu trữ dữ liệu cho ALU xử lý.
16. Khối giải mã lệnh thực hiện công việc giải mã lệnh để xem lệnh yêu cầu thực hiện công việc gì, sau đó khối control logic sẽ thực hiện đúng công việc đó.
17. Các khối trong vi xử lý được kết nối với nhau thông qua bus dữ liệu bên trong. Quá trình kết nối để trao đổi dữ liệu được khối control logic quyết định, sự quyết định này tùy thuộc vào lệnh. Bus dữ liệu luôn là bus 2 chiều.
18. Một lệnh của vi xử lý là một từ dạng số nhị phân, dùng để khối giải mã và khối control logic thực hiện một công việc nhất định.
19. Tập lệnh của vi xử lý là tất cả các lệnh mà vi xử lý có thể hiểu và thực hiện được.
20. Chiều dài của 1 lệnh (số lượng các bit của 1 lệnh) bằng với chiều dài từ dữ liệu
21. Lệnh của vi xử lý được giải mã và thực hiện khi lệnh được nạp thanh ghi lệnh bên trong vi xử lý ở chu kỳ đón lệnh. Ở chu kỳ thực hiện lệnh khối giải mã và khối control logic sẽ thực hiện các yêu cầu của lệnh.
22. Một lệnh của vi xử lý bao gồm 2 phần hay 2 thông tin. Thông tin thứ nhất để báo cho vi xử lý biết làm công việc gì. Thông tin thứ 2 báo cho vi xử lý địa chỉ của dữ liệu. Thông tin thứ nhất thường gọi là mã lệnh hay mã công tác. Thông tin thứ 2 gọi

là địa chỉ hay địa chỉ công tác (có nghĩa là lệnh xảy ra đối với dữ liệu tại địa chỉ đó).

23. Có nhóm lệnh cơ bản và có rất nhiều mã lệnh cho 1 nhóm lệnh.

24. Lệnh của vi xử lý là một số nhị phân gồm cả thông tin và địa chỉ được gọi là mã máy. Để dễ nhớ lệnh mã máy đã được chuyển sang mã gợi nhớ, từ gợi nhớ có ý nghĩa gần với chức năng của lệnh. Tập hợp các từ gợi nhớ gọi là ngôn ngữ Assembly. Khi viết chương trình bằng Assembly, để máy thực hiện chương trình này thì phải có chương trình dịch các lệnh viết bằng Assembly sang mã máy để vi xử lý xử lý, chương trình dịch được gọi là Assembler.

2. Câu hỏi ôn tập và bài tập trắc nghiệm:

1. Sơ đồ khối của vi xử lý dùng để
   A. Diễn tả chi tiết các cổng logic và các Flip Flop được dùng để thiết kế nên vi xử lý.
   B. Diễn tả các mạch logic của vi xử lý kết nối với các thiết bị IO và bộ nhớ bên ngoài.
   C. Dùng để trình bày các khối logic có chức năng xử lý dữ liệu để giải quyết một vấn đề.
   D. Cả 3 câu trên đều đúng.
2. Trong các câu sau câu nào không phải là chức năng của ALU:
   A. Add
   B. Shift
   C. Complement
   D. Lưu trữ dữ liệu.
3. ALU có 2 ngõ vào, 2 ngõ vào này được kết nối với:
   A. Program Counter.
   B. Bus dữ liệu bên trong.
   C. Control logic.
   D. Thanh ghi địa chỉ bộ nhớ.
4. Chức năng chính của khối ALU:
   A. Thực hiện phép cộng.
   B. Đóng vài trò xuất dữ liệu giống như thanh ghi Accumulator.
   C. Thực hiện các phép toán logic và số học để xử lý dữ liệu.
   D. Tất cả 3 câu trên đều đúng.
5. Hầu hết các phép toán logic và số học trong vi xử lý thực hiện giữa nội dung của một ô nhớ hoặc nội dung của một thanh ghi với:
   A. Nội dung của thanh ghi Accumulator.
   B. Nội dung thanh ghi Program Counter.
   C. Nội dung thanh ghi địa chỉ.
   D. Thanh ghi lệnh.
6. Một vi xử lý 16 bit có thể truy xuất 2^{20} = 1.048.567 ô nhớ có thể cho biết thanh ghi PC của Microprocessor này có chiều dài từ dữ liệu bao nhiêu bit:
   A. 4 B. 8 C. 16
   D. 20 E. 22 E. 32
7. Thanh ghi Program counter của vi xử lý là một trong những thanh ghi:
   A. Đặc biệt.
   B. Thông dụng

C. Memory.
D. Tất cả 3 câu trên.

8. Khi vi xử lý đang thực hiện lệnh, thanh ghi PC đang chỉ đến:
A. Lệnh vừa thực hiện.
B. Lệnh đang thực hiện.
C. Lệnh tiếp theo.
D. Cả 3 câu trên đều sai.

9. Hãy thực hiện các phép cộng các số nhị phân 8 bit. Sau khi cộng xong các con số này, hãy xác định ảnh hưởng của phép cộng đến các bit Zero (Z), bit Negative (N), bit Carry (C).

	0000 1111		0011 1011
+	1111 0000	+	1100 0101
	1110 1111		0000 0001
+	1111 1111	+	1111 0001
	0111 0100		0000 0001
+	1100 1100	+	0111 1111

10. Khi tăng nội dung của thanh ghi lên 3 lần (mỗi lần tăng 1) làm cho bit Zero của thanh trạng thái được Set ở mức logic 1 ở lần tăng thứ 3. Vậy giá trị ban đầu chứa trong thanh ghi là bao nhiêu ?

11. Thanh ghi địa chỉ bộ nhớ dùng để chỉ đến:
A. Nội dung của bộ nhớ.
B. Vị trí của ô nhớ.
C. Vị trí của CPU.
D. Vị trí của các thanh ghi.

12. Thanh ghi địa chỉ bộ nhớ kết nối với bus dữ liệu bên trong vi xử lý để nó có thể nạp giá trị từ:
A. Thanh ghi Program counter.
B. Các thanh ghi thông dụng.
C. Memory.
D. Cả 3 câu trên.

13. Các ngõ ra thanh ghi địa chỉ dùng để kết nối với
A. Thanh ghi Accumulator của vi xử lý.
B. Bus dữ liệu bên trong của Microprocessor.
C. Bus địa chỉ bộ nhớ bên ngoài của vi xử lý.
D. Với ngõ vào của bộ giải mã lệnh.

Chương 3

Giới thiệu vi điều khiển

I. Giới thiệu vi điều khiển.

II. Giới thiệu vi điều khiển MCS51.

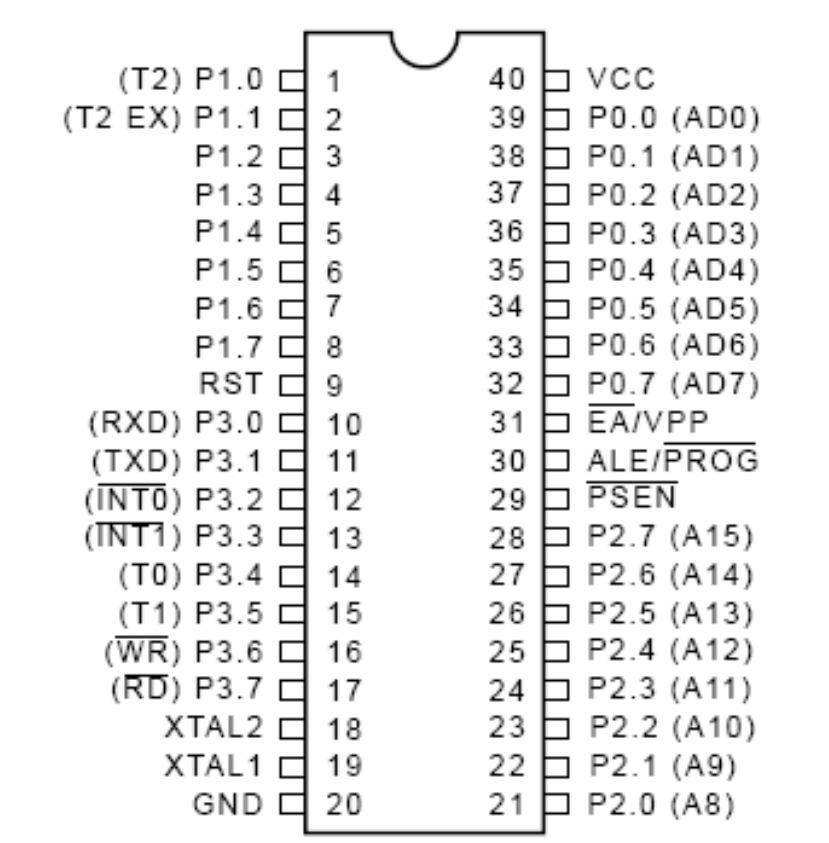
III. Tóm tắt phần cứng vi điều khiển MCS51.

IV. Khảo sát cấu trúc bên trong của vi điều khiển MCS51.

1. *Sơ đồ cấu trúc bên trong của vi điều khiển:*
2. *Khảo sát sơ đồ chân của 89C51.*

V. Tổ chức bộ nhớ của vi điều khiển MCS51.

1. *Tổ chức bộ nhớ:*
2. *Các thanh ghi có chức năng đặc biệt:*



I. GIỚI THIỆU VI ĐIỀU KHIỂN:

Ở chương 1 và 2 đã giới thiệu về cấu trúc bên trong và chức năng của các khối bên trong cũng như trình tự hoạt động xử lý dữ liệu của vi xử lý.

Vi xử lý có rất nhiều loại bắt đầu từ 4 bit cho đến 32 bit, vi xử lý 4 bit hiện nay không còn như vi xử lý 8 vit vẫn còn mặc dù đã có vi xử lý 32 bit.

Lý do sự tồn tại của vi xử lý 8 bit là phù hợp với 1 số yêu cầu điều khiển của các thiết bị điều khiển trong công nghiệp. Các vi xử lý 32 bit thường sử dụng cho các máy tính vì khối lượng dữ liệu của máy tính rất lớn nên cần các vi xử lý càng mạnh càng tốt.

Các hệ thống điều khiển trong công nghiệp sử dụng các vi xử lý 8 bit để điều khiển như hệ thống điện của xe hơi, hệ thống điều hòa, hệ thống điều khiển các dây chuyền sản xuất,...

Khi sử dụng vi xử lý cần phải thiết kế một hệ thống gồm có:

- Vi xử lý.
- Có bộ nhớ.
- Các IC ngoại vi.

Bộ nhớ dùng để chứa chương trình cho vi xử lý thực hiện và chứa dữ liệu xử lý, các IC ngoại vi dùng để xuất nhập dữ liệu từ bên ngoài vào xử lý và điều khiển trở lại. Các khối này liên kết với nhau tạo thành một hệ thống vi xử lý.

Yêu cầu điều khiển càng cao thì hệ thống càng phức tạp và nếu yêu cầu điều khiển có đơn giản ví dụ chỉ cần đóng mở 1 đèn led theo một thời gian yêu cầu nào đó thì hệ thống vi xử lý cũng phải có đầy đủ các khối trên.

Để kết nối các khối trên tạo thành một hệ thống vi xử lý đòi hỏi người thiết kế phải rất hiểu biết về tất cả các thành phần vi xử lý, bộ nhớ, các thiết bị ngoại vi. Hệ thống tạo ra khá phức tạp , chiếm nhiều không gian, mạch in, và vấn đề chính là đòi hỏi người thiết kế, người sử dụng hiểu thật rõ về hệ thống. Một lý do chính nữa là vi xử lý thường xử lý dữ liệu theo byte hoặc word trong khi đó các đối tượng điều khiển trong công nghiệp thường điều khiển theo bit.

Chính vì sự phức tạp nên các nhà chế tạo đã tích hợp một ít bộ nhớ và một số các thiết bị ngoại vi cùng với vi xử lý tạo thành một IC gọi là vi điều khiển – Microcontroller.

Khi vi điều khiển ra đời đã mang lại sự tiện lợi là dễ dàng sử dụng trong điều khiển công nghiệp, việc sử dụng vi điều khiển không đòi hỏi người sử dụng phải hiểu biết một lượng kiến thức quá nhiều như người sử dụng vi xử lý – dĩ nhiên người sử dụng hiểu biết càng nhiều thì càng tốt nhưng đối với người bắt đầu thì việc sử dụng vi xử lý là điều rất phức tạp trong khi đó mong muốn là sử dụng được ngay.

Các phần tiếp theo chúng ta sẽ khảo sát vi điều khiển để thấy rõ sự tiện lợi trong vấn đề điều khiển trong công nghiệp.

II. GIỚI THIỆU HỌ VI ĐIỀU KHIỂN MCS-51:

Có rất nhiều hãng chế tạo được vi điều khiển, hãng sản xuất nổi tiếng là ATMEL. Hãng Intel là nhà thiết kế. Có thể truy xuất để lấy tài liệu của hãng bằng địa chỉ "http://www.atmel.com/"

Có nhiều họ vi điều khiển mang các mã số khác nhau, một trong họ nổi tiếng là họ ***MCS-51.***

- Trong họ MCS-51 thì vi điều khiển đầu tiên là 80C31 không có bộ nhớ bên trong là do không tích hợp được.
- Vi điều khiển 80C51 tích hợp được 4 kbyte bộ nhớ Prom. Chỉ lập trìnnh 1 lần không thể xóa để lập trình lại được.
- Vi điều khiển 87C51 tích hợp được 4 kbyte bộ nhớ eprom. Cho phép lập trình nhiều lần và xóa bằng tia cực tím.
- Vi điều khiển 89C51 tích hợp được 4 kbyte bộ nhớ flash rom nạp và xóa bằng điện một cách tiện lợi và nhanh chóng. Có thể cho phép nạp xóa hàng ngàn lần.

Song song với họ MCS-51 là họ MCS-52 có 3 timer nhiều hơn họ MCS-51 một timer và dung lượng bộ nhớ nội lớn gấp đôi tức là 8kbyte.

Hiện nay có rất nhiều vi điều khiển thế hệ sau có nhiều đặc tính hay hơn, nhiều thanh ghi hơn, dung lượng bộ nhớ lớn hơn.

Ứng dụng của vi điều khiển rất nhiều trong các hệ thống điều khiển công nghiệp, các dây chuyền sản xuất, các bộ điều khiển lập trình, máy giặt, máy điều hòa nhiệt độ, máy bơm xăng tự động... có thể nói vi xử lý và vi điều khiển được ứng dụng trong hầu hết mọi lĩnh vực.

III. TÓM TẮT PHẦN CỨNG VI ĐIỀU KHIỂN HỌ MCS-51:

Đến thời điểm hiện nay có rất nhiều loại Vi điều khiển thuộc họ MCS-51, trong tài liệu sẽ giới thiệu về vi điều khiển 89C51 hoặc 89C52. Các vi điều khiển thế hệ sau sẽ được đề cập ở phần sau.

Các vi điều khiển họ MCS-51 có các đặc điểm chung như sau:

- Có 4 Kbyte bộ nhớ FLASH ROM bên trong dùng để lưu chương trình điều khiển.
- Có 128 Byte RAM nội.
- 4 Port xuất / nhập (Input/Output) 8 bit.
- Có khả năng giao tiếp truyền dữ liệu nối tiếp.
- Có thể giao tiếp với 64 Kbyte bộ nhớ bên ngoài dùng để lưu ***chương trình*** điều khiển.
- Có thể giao tiếp với 64 Kbyte bộ nhớ bên ngoài dùng để lưu ***dữ liệu***.
- Có 210 bit có thể truy xuất từng bit. Có các lệnh xử lý bit.

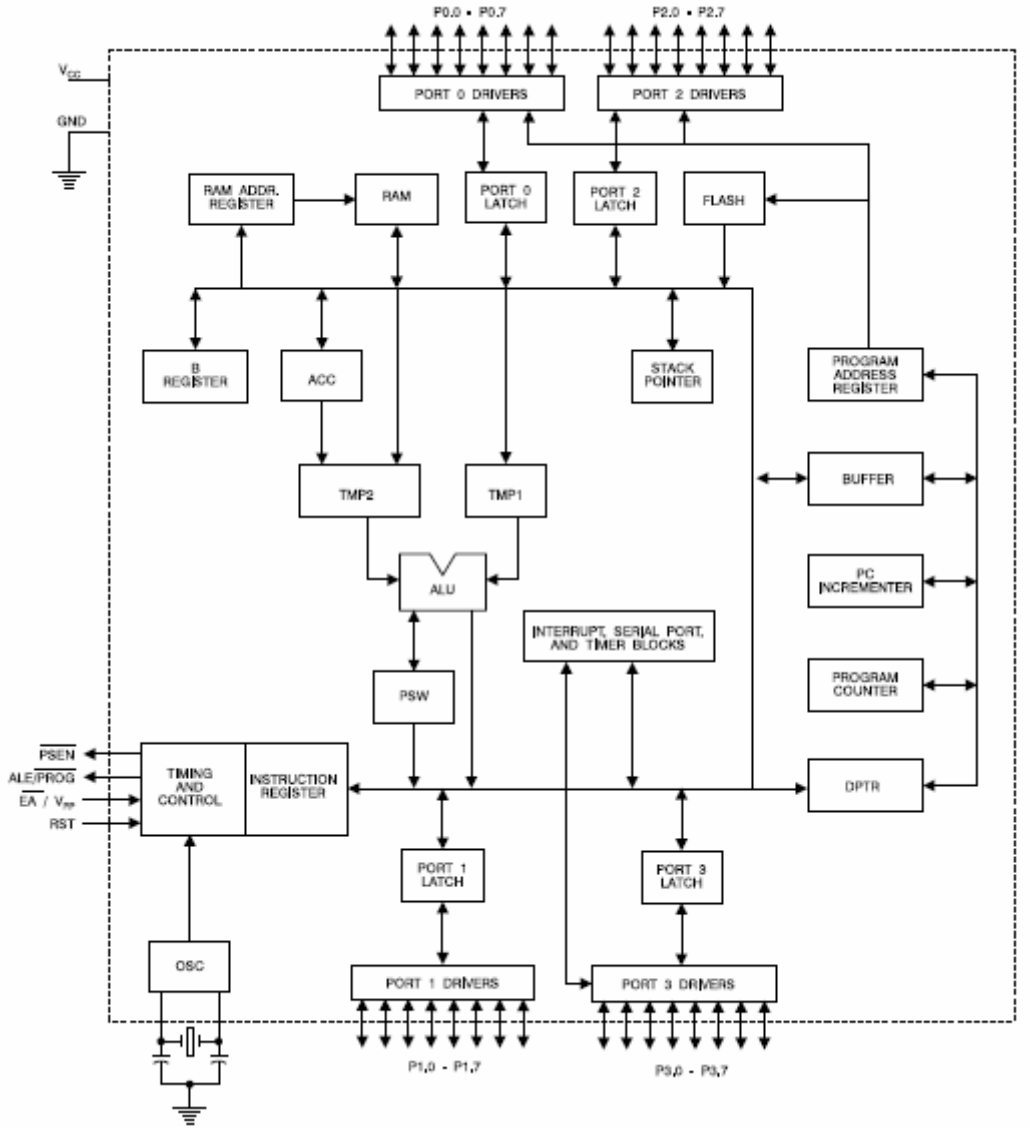
Tất cả các vi điều khiển cùng họ MCS-51 hoặc MCS-52 đều có các đặc tính cơ bản giống nhau như phần mềm, còn phần cứng thì khác nhau, các vi điều khiển sau này sẽ có nhiều tính năng hay hơn các vi điều khiển thế hệ trước. Ví dụ vi điều khiển 89C51 sẽ tiện cho việc sử dụng hơn vi điều khiển 80C51 hay 87C51. Vi điều khiển 89S51 sẽ hay hơn 89C51 vì có nhiều thanh ghi hơn, có thêm chế độ nạp nối tiếp rất tiện lợi. Những thế hệ đi sau sẽ kế thừa tất cả những gì của thế hệ đi trước. Trong phần này chỉ đề cập đến vi điều khiển 89C51/89C52.

IV. KHẢO SÁT CẤU TRÚC BÊN TRONG CỦA VI ĐIỀU KHIỂN 89C51:

1. Sơ đồ cấu trúc bên trong của vi điều khiển:

Sơ đồ cấu trúc của vi điều khiển được trình bày ở hình 3-1. Các thanh ghi có trong vi điều khiển bao gồm:

- Khối ALU đi kèm với các thanh ghi temp1, temp2 và thanh ghi trạng thái PSW.
- Bộ điều khiển logic (timing and control).
- Vùng nhớ RAM nội và vùng nhớ Flash Rom lưu trữ chương trình.
- Mạch tạo dao động nội kết hợp với tụ thạch anh bên ngoài để tạo dao động.
- Khối xử lý ngắt, truyền dữ liệu, khối timer/counmter.
- Thanh ghi A, B, dptr và 4 port0, port1, port2, port3 có chốt và đệm.
- Thanh ghi bộ đếm chương trình PC (program counter).
- Con trỏ dữ liệu dptr (data pointer).
- Thanh ghi con trỏ ngăn xếp SP (stack pointer).
- Thanh ghi lệnh IR (instruction register).
- Ngoài ra còn có 1 số các thanh ghi hổ trợ để quản lý địa chỉ bộ nhớ ram nội bên trong cũng như các thanh ghi quản lý địa chỉ truy xuất bộ nhớ bên ngoài.



Hình 3-1. Cấu trúc bên trong của vi điều khiển.

Các khối bên trong của vi điều khiển có các thành phần giống như đã trình bày ở phần chương 1 như khối ALU, thanh ghi temp1, thanh ghi temp2, thanh ghi bộ đếm chương trình PC, thanh con trỏ ngăn xếp, thanh ghi trạng thái PSW, thanh ghi lệnh IR, khối giải mã lệnh, khối điều khiển logic.

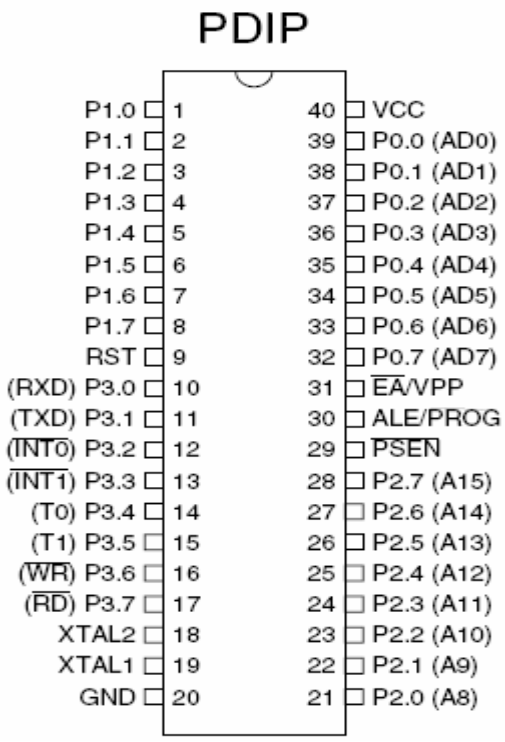
Khi khảo sát các khối này không cần thiết phải hiểu hết chức năng của từng khối hoạt động ra sao vào thời điểm này, các thông tin ở phần sau sẽ giúp hiểu rõ thêm về tổ chức phần cứng này.

Tập lệnh cho người lập trình là kết quả của sự liên kết các khối bên trong của vi điều khiển – những gì tập lệnh cung cấp là đều do phần cứng xây dựng nên.

2. Khảo sát sơ đồ chân 89C51:

Sơ đồ chân của vi điều khiển 89C51 được trình bày ở hình 3-2.

Vi điều khiển 89C51 có tất cả 40 chân. Trong đó có 24 chân có tác dụng kép (có nghĩa là 1 chân có 2 chức năng), mỗi đường có thể hoạt động như đường xuất nhập điều khiển IO [input output] hoặc là thành phần của các bus dữ liệu và bus địa chỉ để tải địa chỉ và dữ liệu khi giao tiếp với bộ nhớ ngoài.



Hình 3-2. Sơ đồ chân của 89C51

Chức năng các chân của 89C51:

a. Các Port:

- **Port 0:**

Port 0 là port có 2 chức năng với số thứ tự chân 32 – 39.

Trong các hệ thống điều khiển đơn giản sử dụng bộ nhớ bên trong không dùng bộ nhớ mở rộng bên ngoài thì port 0 được dùng làm các đường điều khiển IO (Input- Output).

Trong các hệ thống điều khiển lớn sử dụng bộ nhớ mở rộng bên ngoài thì port 0 có chức năng là bus địa chỉ và bus dữ liệu AD7 - AD0. (Address: địa chỉ, data: dữ liệu)

- **Port 1:**

Port 1 với số thứ tự chân 1- 8. Port1 chỉ có 1 chức năng dùng làm các đường điều khiển xuất nhập IO, port 1 không có chức năng khác.

❒ **Port 2:**

Port 2 là port có 2 chức năng với số thứ tự chân 21 – 28.

Trong các hệ thống điều khiển đơn giản sử dụng bộ nhớ bên trong không dùng bộ nhớ mở rộng bên ngoài thì port 2 được dùng làm các đường điều khiển IO (Input- Output).

Trong các hệ thống điều khiển lớn sử dụng bộ nhớ mở rộng bên ngoài thì port 2 có chức năng là bus địa chỉ cao A8 - A15.

❒ **Port 3:**

Port 3 là port có 2 chức năng với số thứ tự chân 10 -17. Các chân của port này có nhiều chức năng, các công dụng chuyển đổi có liên hệ với các đặc tính đặc biệt của 89C51 như ở bảng sau:

Bit	Tên	Chức năng chuyển đổi
P3.0	RxD	Ngõ vào nhận dữ liệu nối tiếp.
P3.1	TxD	Ngõ xuất dữ liệu nối tiếp.
P3.2	$\overline{INT0}$	Ngõ vào ngắt cứng thứ 0.
P3.3	$\overline{INT1}$	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	$\overline{WR}$	Tín hiệu điều khiển ghi dữ liệu lên bộ nhớ ngoài.
P3.7	$\overline{RD}$	Tín hiệu điều khiển đọc dữ liệu từ bộ nhớ ngoài.

b. Các ngõ tín hiệu điều khiển:

❒ **Ngõ tín hiệu $\overline{PSEN}$ (Program store enable):**

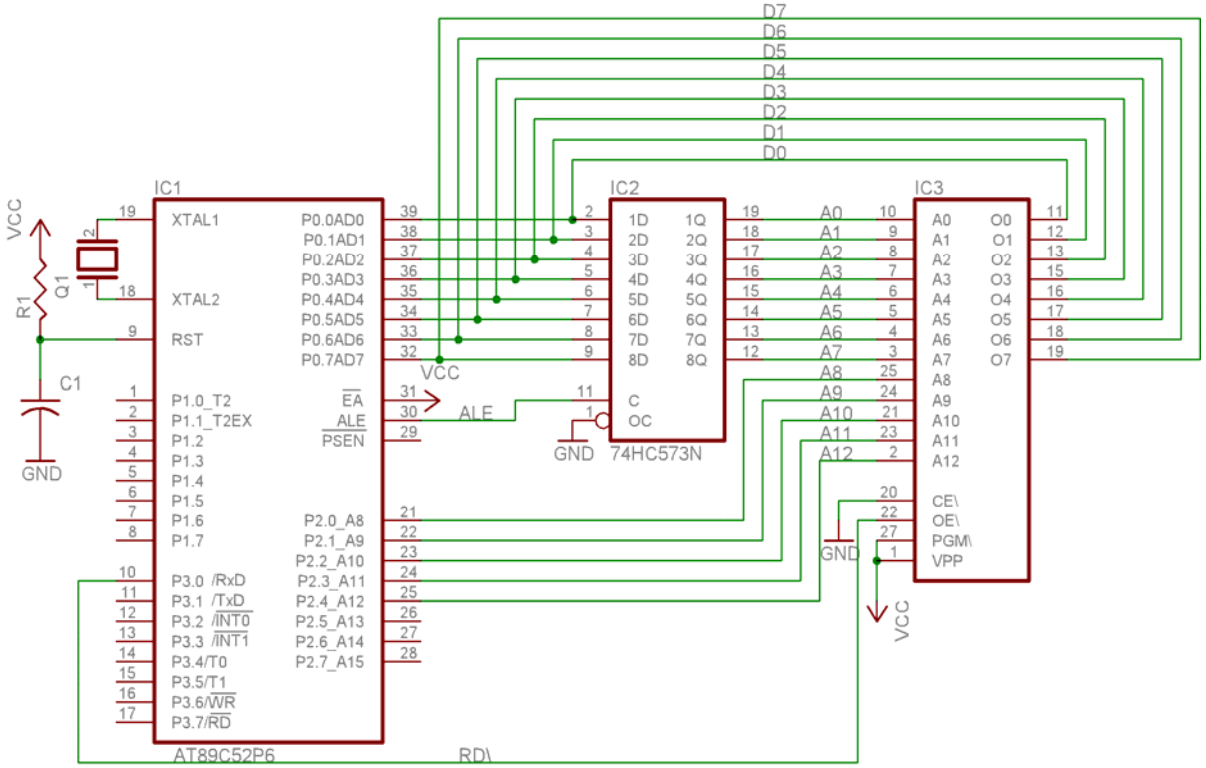
PSEN là tín hiệu ngõ ra ở chân 29 có tác dụng cho phép đọc bộ nhớ chương trình mở rộng thường nối đến chân $\overline{OE}$ (output enable hoặc $\overline{RD}$) của Eprom cho phép đọc các byte mã lệnh.

Khi có giao tiếp với bộ nhớ chương trình bên ngoài thì mới dùng đến $\overline{PSEN}$, nếu không có giao tiếp thì chân $\overline{PSEN}$ bỏ trống. $\overline{PSEN}$

($\overline{PSEN}$ ở mức thấp trong thời gian vi điều khiển 89C51 lấy lệnh. Các mã lệnh của chương trình đọc từ Eprom qua bus dữ liệu và được chốt vào thanh ghi lệnh bên trong 89C51 để giải mã lệnh. Khi 89C51 thi hành chương trình trong EPROM nội thì $\overline{PSEN}$ ở mức logic 1).

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

Khi vi điều khiển 89C51 truy xuất bộ nhớ bên ngoài, port 0 có chức năng là bus tải địa chỉ và bus dữ liệu [AD7 – AD0] do đó phải tách các đường dữ liệu và địa chỉ. Tín hiệu ra ALE ở chân thứ 30 dùng làm tín hiệu điều khiển để giải đa hợp các đường địa chỉ và dữ liệu khi kết nối chúng với IC chốt. Xem hình 3-3.



Hình 3-3. Kết nối vi điều khiển với IC chốt, bộ nhớ EPROM ngoại, mạch reset, tụ thạch anh.

Tín hiệu ra ở chân ALE là một xung trong khoảng thời gian port 0 đóng vai trò là địa chỉ thấp nên việc chốt địa chỉ được thực hiện 1 cách hoàn toàn tự động.

Các xung tín hiệu ALE có tốc độ bằng 1/6 lần tần số dao động của tụ thạch anh gắn vào vi điều khiển và có thể dùng tín hiệu xung ngõ ra ALE làm xung clock cung cấp cho các phần khác của hệ thống.

Trong chế độ lập trình cho bộ nhớ nội của vi điều khiển thì chân ALE được dùng làm ngõ vào nhận xung lập trình từ bên ngoài để lập trình cho bộ nhớ flash rom trong 89C51.

Ở hình 3-3 chỉ là minh hoạ kết nối vi điều khiển (89C52) với bộ nhớ EPROM ngoại để thấy vai trò của tín hiệu ALE, các đường còn lại của vi điều khiển có thể dùng để kết nối điều khiển các đối tượng khác.

❒ Ngõ tín hiệu $\overline{EA}$ (External Access):

Tín hiệu vào $\overline{EA}$ ở chân 31 thường nối lên mức 1 hoặc mức 0.

Nếu nối $\overline{EA}$ lên mức logic 1 (+5v) thì vi điều khiển sẽ thi hành chương trình từ bộ nhớ nội.

Nếu nối $\overline{EA}$ với mức logic 0 (0V) thì vi điều khiển sẽ thi hành chương trình từ bộ nhớ ngoại.

❒ Ngõ tín hiệu RST (Reset):

Ngõ vào RST ở chân 9 là ngõ vào Reset của 89C51. Sơ đồ kết nối mạch reset như hình vẽ 3-3. Khi cấp điện cho hệ thống hoặc khi nhấn nút reset thì mạch sẽ reset vi điều khiển. Khi reset thì tín hiệu reset phải ở mức cao ít nhất là 2 chu kỳ máy, khi đó các thanh ghi bên trong được nạp những giá trị thích hợp để khởi động hệ thống.

Trạng thái của tất cả các thanh ghi trong 89C51 sau khi reset hệ thống được tóm tắt như sau:

Thanh ghi	Nội dung
Bộ đếm chương trình PC	0000H
Thanh ghi tích lũyA	00H
Thanh ghi B	00H
Thanh ghi trạng thái PSW	00H
Thanh ghi con trỏ SP	07H
DPTR	0000H
Port 0 đến port 3	FFH (1111 1111)
IP	XXX0 0000 B
IE	0X0X 0000 B
Các thanh ghi định thời	00H
SCON SBUF	00H
PCON (HMOS)	00H
PCON (CMOS)	0XXX XXXXH 0XXX 0000 B

Thanh ghi quan trọng nhất là thanh ghi bộ đếm chương trình PC = 0000H. Sau khi reset vi điều khiển luôn bắt đầu thực hiện chương trình tại địa chỉ 0000H của bộ nhớ chương trình nên các chương trình viết cho vi điều khiển luôn bắt đầu viết tại địa chỉ 0000H.

Nội dung của RAM trên chip không bị thay đổi bởi tác động của ngõ vào reset [có nghĩa là vi điều khiển đang sử dụng các thanh ghi để lưu trữ dữ liệu nhưng nếu vi điều khiển bị reset thì dữ liệu trong các thanh ghi vẫn không đổi].

❒ **Các ngõ vào bộ dao động Xtal1, Xtal2:**

Bộ dao động được được tích hợp bên trong 89C51, khi sử dụng 89C51 người thiết kế chỉ cần kết nối thêm tụ thạch anh và các tụ như hình vẽ trong sơ đồ hình 3-3. Tần số tụ thạch anh thường sử dụng cho 89C51 là 12Mhz ÷ 24Mhz.

❒ **Chân 40 (Vcc) được nối lên nguồn 5V, chân 20 GND nối mass.**

3. Sơ đồ mạch kết nối một số ứng dụng đơn giản dùng bộ nhớ nội:

a. Mạch đồng hồ số hiển thị giờ phút giây trên led 7 đoạn:

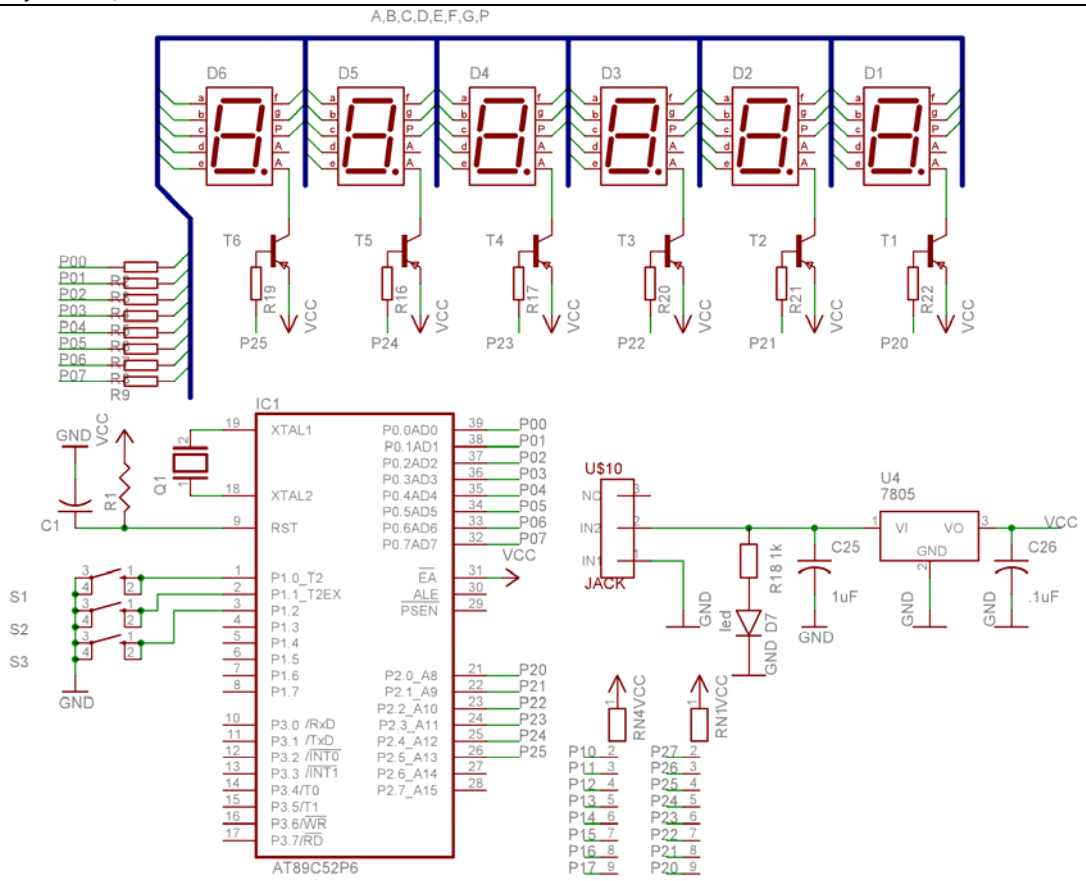
Trong sơ đồ này sử dụng 6 led 7 đoạn loại anode chung để hiển thị giờ, phút và giây sử dụng phương pháp quét, 6 transistor sử dụng là loại pnp thường là A564 và điện trở cực B có giá trị khoảng 10kΩ, điện trở hạn dòng cho các đoạn có giá trị 220Ω. Để hiểu hoạt động điều khiển quét led 7 đoạn hãy đọc chương 8.

Hai điện trở mạng 9 chân có giá trị là 4,7kΩ hoặc 10kΩ, tụ reset có giá trị 10µF, điện trở reset có giá trị 10kΩ, thạch anh có giá trị thường là 12MHz.

3 nút nhấn S1, S2 và S3 dùng để chỉnh các thông số giờ phút giây.

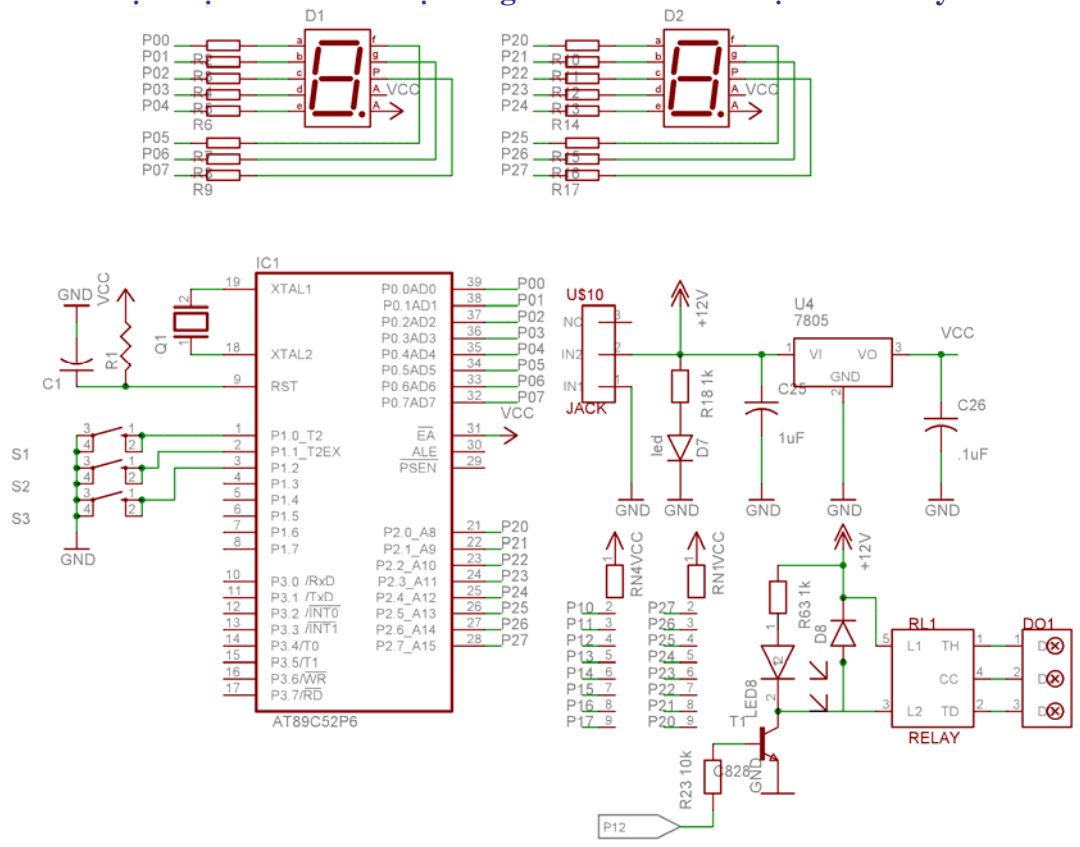
Mạch nguồn sử dụng IC ổn áp 5V.

Để hệ thống hoạt động thì phải có chương trình.



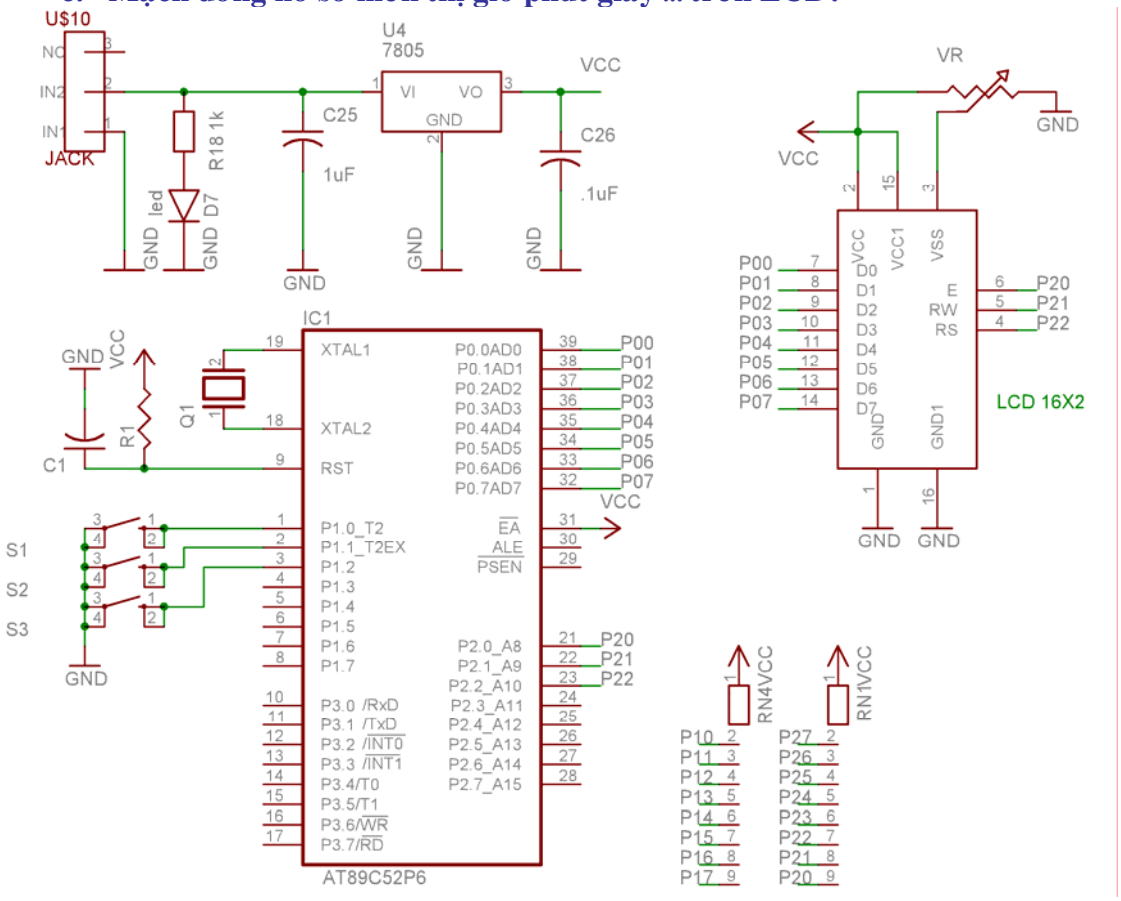
Hình 3-4. Mạch đồng hồ số dùng led 7 đoạn.

b. Mạch định thời hiển thị thời gian trên 2 led 7 đoạn có 1 relay điều khiển:



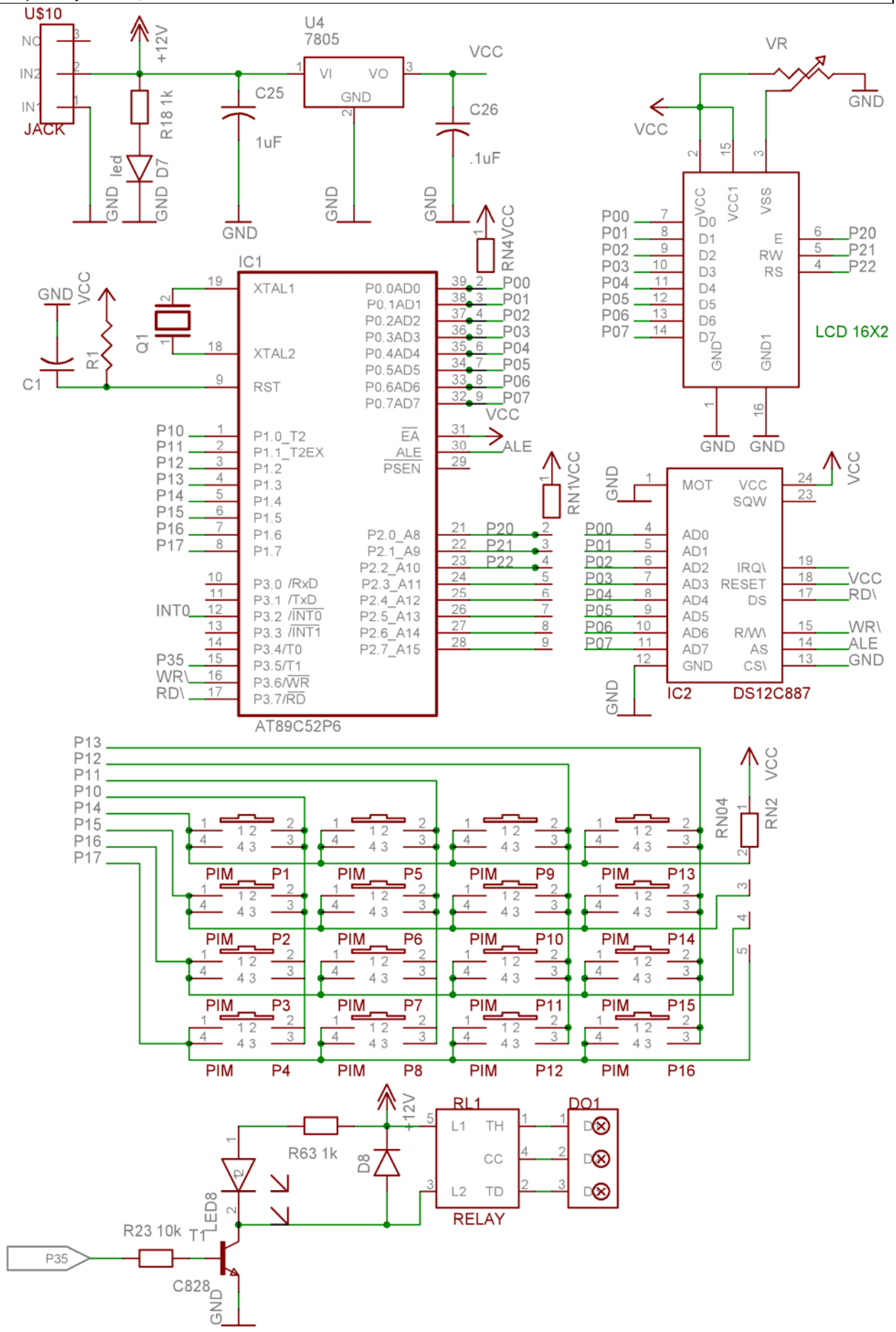
Hình 3-5. Mạch định thời điều khiển 1 relay và hiển thị thời gian trên 2 led..

c. Mạch đồng hồ số hiển thị giờ phút giây ... trên LCD:



Hình 3-6. Mạch đồng hồ số hiển thị dùng LCD.

d. Mạch đồng hồ có thể báo chuông tiết học sử dụng realtime:



Hình 3-7. Mạch đồng hồ số hiển thị dùng LCD có thêm báo chuông giờ học.

V. CẤU TRÚC BỘ NHỚ CỦA VI ĐIỀU KHIỂN:

1. Tổ chức bộ nhớ:

Vi điều khiển 89C51 có ***bộ nhớ nội bên trong*** và có thêm khả năng giao tiếp với ***bộ nhớ bên ngoài*** nếu bộ nhớ bên trong không đủ khả năng lưu trữ chương trình.

Bộ nhớ nội bên trong gồm có 2 loại bộ nhớ: bộ nhớ dữ liệu và bộ chương trình. Bộ nhớ dữ liệu có 256 byte, bộ nhớ chương trình có dung lượng 4kbyte. [89C52 có 8 kbyte, 89W55 có 16kbyte].

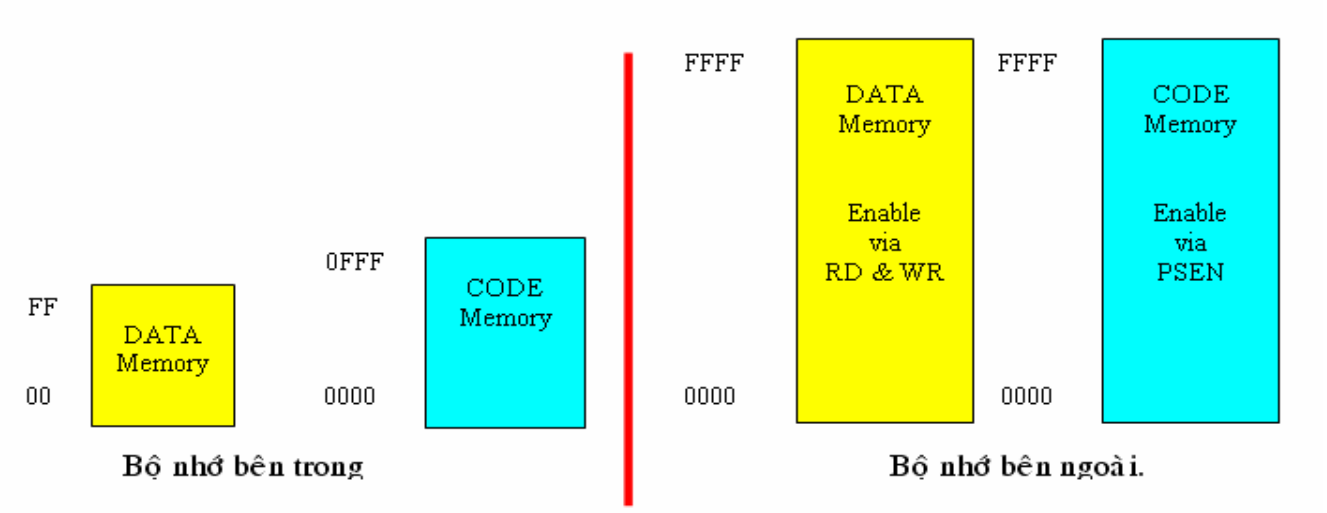
Bộ nhớ mở rộng bên ngoài cũng được chia ra làm 2 loại bộ nhớ: bộ nhớ dữ liệu và bộ nhớ chương trình. Khả năng giao tiếp là 64kbyte cho mỗi loại. Hình 3-8 minh họa khả năng giao tiếp bộ nhớ của vi điều khiển 89C51.

Bộ nhớ mở rộng bên ngoài và bộ nhớ chương trình bên trong không có gì đặc biệt – chỉ có chức năng lưu trữ dữ liệu và mã chương trình nên không cần phải khảo sát.

Bộ nhớ chương trình bên trong của vi điều khiển thuộc loại bộ nhớ Flash rom cho phép xóa bằng xung điện và lập trình lại.

Bộ nhớ ram nội bên trong là một bộ nhớ đặc biệt người sử dụng vi điều khiển cần phải nắm rõ các tổ chức và các chức năng đặc biệt của bộ nhớ này.

Sơ đồ cấu trúc bên trong của bộ nhớ này được trình bày như hình 3-9.



Hình 3-8. Bảng tóm tắt các vùng nhớ 89C51.

RAM bên trong 89C51 được phân chia như sau:

- ☆ Các bank thanh ghi có địa chỉ từ 00H đến 1FH.
- ☆ RAM địa chỉ hóa từng bit có địa chỉ từ 20H đến 2FH.
- ☆ RAM đa dụng từ 30H đến 7FH.
- ☆ Các thanh ghi chức năng đặc biệt từ 80H đến FFH.

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

RAM

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

Các Thanh Ghi có Chức Năng Đặc Biệt

Hình 3-9: Cấu trúc bộ nhớ RAM bên trong vi điều khiển:

❒ Các bank thanh ghi :

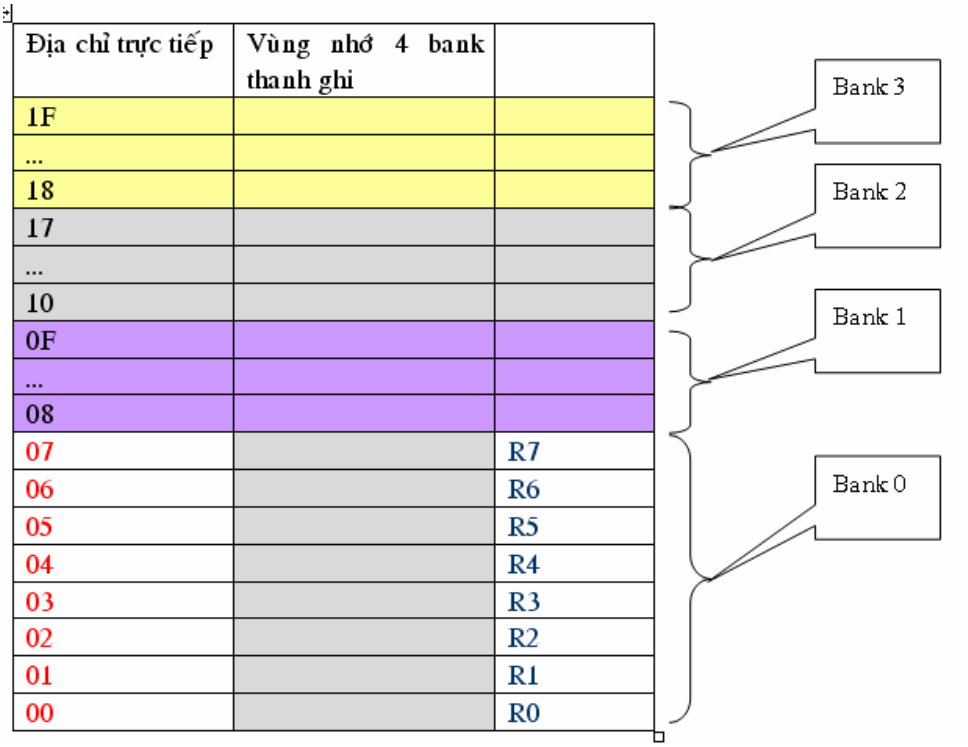
32 byte thấp của bộ nhớ nội được dành cho 4 bank thanh ghi.

Bộ lệnh 89C51 hổ trợ thêm 8 thanh ghi có tên là R0 đến R7 và theo mặc định sau khi reset hệ thống thì các thanh ghi R0 đến R7 được gán cho 8 ô nhớ có địa chỉ từ 00H đến 07H được minh họa bởi hình 3-10, khi đó bank 0 có 2 cách truy xuất bằng địa chỉ trực tiếp và bằng thanh ghi R.

Các lệnh dùng các thanh ghi R0 đến R7 sẽ có số lượng byte mã lệnh ít hơn và thời gian thực hiện lệnh nhanh hơn so với các lệnh có chức năng tương ứng nếu dùng kiểu địa chỉ trực tiếp.

Các dữ liệu được dùng thường xuyên nên lưu trữ ở một trong các thanh ghi này.

Do có 4 bank thanh ghi nên tại một thời điểm chỉ có một bank thanh ghi được truy xuất bởi các thanh ghi R0 đến R7, để chuyển đổi việc truy xuất các bank thanh ghi ta phải thay đổi các bit chọn bank trong thanh ghi trạng thái.



Hình 3-10. Minh họa cách gán bank thanh ghi cho nhóm thanh ghi R.

Người lập trình dùng vùng nhớ 4 bank thanh ghi để lưu trữ dữ liệu phục vụ cho việc xử lý dữ liệu khi viết chương trình.

Chức năng chính của 4 bank thanh ghi này là nếu trong hệ thống có sử dụng nhiều chương trình thì chương trình thứ nhất bạn có thể sử dụng hết các thanh ghi R0 đến R7 của bank0, khi bạn chuyển sang chương trình thứ 2 để xử lý một công việc gì đó và vẫn sử dụng các thanh ghi R0 đến R7 để lưu trữ cho việc sử lý dữ liệu mà không làm ảnh hưởng đến các dữ liệu R0 đến R7 trước đây và không cần phải thực hiện công việc cất dữ liệu thì cách nhanh nhất là bạn gán nhóm thanh ghi R0 đến R7 cho bank1 là xong. Tương tự bạn có thể mở thêm hai chương trình nữa và gán cho các bank 3 và 4. Nếu bạn chưa hiểu thì cứ tiếp tục sau này sẽ hiểu.

❒ RAM có thể truy xuất từng bit:

Vi điều khiển 89C51 có 210 ô nhớ bit có thể truy xuất từng bit, trong đó có 128 bit nằm ở các các ô nhớ byte có địa chỉ từ 20H đến 2FH và các bit còn lại chứa trong nhóm thanh ghi có chức năng đặc biệt.

Các ô nhớ cho phép truy xuất từng bit và các lệnh xử lý bit là một thế mạnh của vi điều khiển. Các bit có thể được đặt, xóa, AND, OR bằng 1 lệnh duy nhất trong khi đó để xử lý các bit thì ***vi xử lý*** vẫn có thể xử lý được nhưng phải sử dụng rất nhiều lệnh để đạt được cùng một kết quả vì ***vi xử lý*** thường xử lý byte.

Các port cũng có thể truy xuất được từng bit.

128 ô nhớ bit cho phép truy xuất từng bit và cũng có thể truy xuất byte phụ thuộc vào lệnh được dùng là lệnh xử bit hay lệnh xử lý byte. Chú ý địa chỉ của ô nhớ byte và bit trùng nhau.

Người lập trình dùng vùng nhớ này để lưu trữ dữ liệu phục vụ cho việc xử lý dữ liệu byte hoặc bit. Các dữ liệu xử lý bit nên lưu vào vùng nhớ này.

Chú ý: các ô nhớ nào mà chia ra làm 8 và có các con số bên trong là các ô nhớ vừa cho truy xuất byte và cả truy xuất bit. Những ô nhớ còn lại thì không thể truy xuất bit.

❒ RAM đa dụng :

Vùng nhớ ram đa dụng gồm có 80 byte có địa chỉ từ 30H đến 7FH – vùng nhớ này không có gì đặc biệt so với 2 vùng nhớ trên. Vùng nhớ bank thanh ghi 32 byte từ 00H đến 1FH cũng có thể dùng làm vùng nhớ ram đa dụng mặc dù các các ô nhớ này đã có chức năng như đã trình bày.

Mọi địa chỉ trong vùng RAM đa dụng đều có thể truy xuất tự do dùng kiểu địa chỉ trực tiếp hoặc gián tiếp.

Bộ nhớ ngăn xếp của vi điều khiển dùng bộ nhớ Ram nội nên dung lượng của bộ nhớ ngăn xếp nhỏ trong khi đó các bộ vi xử lý dùng bộ nhớ bên ngoài làm bộ nhớ ngăn xếp nên dung lượng tùy ý mở rộng.

2. Các thanh ghi có chức năng đặc biệt :

Các thanh ghi nội của 89C51 được truy xuất ngầm định bởi bộ lệnh.

Các thanh ghi trong 89C51 được định dạng như một phần của RAM trên chip vì vậy mỗi thanh ghi sẽ có một địa chỉ (ngoại trừ thanh ghi bộ đếm chương trình và thanh ghi lưu trữ mã lệnh vì các thanh ghi này đã có chức năng cố định). Cũng như các thanh ghi R0 đến R7, vi điều khiển 89C51 có 21 thanh ghi có chức năng đặc biệt nằm ở vùng trên của RAM nội có địa chỉ từ 80H đến FFH.

Chú ý: 128 ô nhớ có địa chỉ từ 80H đến FFH thì chỉ có 21 thanh ghi có chức năng đặc biệt được xác định các địa chỉ – còn các ô nhớ còn lại thì chưa thiết lập và trong tương lai sẽ được các nhà thiết kế vi điều khiển thiết lập thêm khi đó sẽ có các vi điều khiển thế hệ mới hơn.

Trong phần này chỉ mang tính giới thiệu các phần tiếp theo sẽ trình bày chi tiết hơn về các thanh ghi này.

- ***Các ô nhớ có địa chỉ 80H, 90H, A0h, B0h:***

Là các Port của 89C51 bao gồm Port0 có địa chỉ 80H, Port1 có địa chỉ 90H, Port2 có địa chỉ A0H và Port3 có địa chỉ B0H. Tất cả các Port này đều có thể truy xuất từng bit nên rất thuận tiện trong điều khiển IO. Địa chỉ của các bit được đặt tên với ô bắt đầu chính là địa chỉ của port tương ứng ví dụ như bit đầu tiên của port 0 là 80h cũng chính là địa chỉ bắt đầu của port 0. Người lập trình không cần nhớ địa chỉ các bit trong các port vì phần mềm lập trình cho phép truy xuất bằng tên từng bit dễ nhớ như sau: P0.0 chính là bit có địa chỉ 80h của port0.

Ngoại trừ thanh ghi A có thể được truy xuất ngầm, đa số các thanh ghi có chức năng đặc biệt SFR có thể địa chỉ hóa từng bit hoặc byte.

- ***Ô nhớ có địa chỉ 81h:***

Là thanh ghi con trỏ ngăn xếp SP (stack pointer) - có chức năng quản lý địa chỉ của bộ nhớ ngăn xếp. Bộ nhớ ngăn xếp dùng để lưu trữ tạm thời các dữ liệu trong quá trình thực hiện chương trình của vi điều khiển.

Các lệnh liên quan đến ngăn xếp bao gồm các lệnh cất dữ liệu vào ngăn xếp (lệnh push) và lấy dữ liệu ra khỏi ngăn xếp (lệnh pop).

Lệnh cất dữ liệu vào ngăn xếp sẽ làm tăng SP trước khi ghi dữ liệu vào.

Sau lệnh lấy ra khỏi ngăn xếp sẽ làm giảm SP.

Bộ nhớ ngăn xếp của 89C51 nằm trong RAM nội và bị giới hạn về cách truy xuất địa chỉ - chỉ cho phép truy xuất địa chỉ gián tiếp. Dung lượng bộ nhớ ngăn xếp lớn nhất là 128 byte ram nội của 89C51.

Khi Reset 89C51 thì thanh ghi SP sẽ mang giá trị mặc định là 07H và dữ liệu đầu tiên sẽ được cất vào ô nhớ ngăn xếp có địa chỉ 08H.

Nếu phần mềm ứng dụng không khởi tạo SP một giá trị mới thì bank 1 có thể cả 2 và 3 sẽ không dùng được vì vùng nhớ này đã được dùng làm ngăn xếp.

Ngăn xếp được truy xuất trực tiếp bằng các lệnh PUSH và POP để lưu trữ tạm thời và lấy lại dữ liệu, hoặc truy xuất ngầm bằng lệnh gọi chương trình con (ACALL, LCALL) và các lệnh trở về (RET, RETI) để lưu trữ địa chỉ của bộ đếm chương trình khi bắt đầu thực hiện chương trình con và lấy lại địa chỉ khi kết thúc chương trình con.

- ***Ô nhớ có địa chỉ 82h và 83h :***

Là 2 thanh ghi dpl (byte thấp) có địa chỉ là 82H và dph (byte cao) có địa chỉ 83H. Hai thanh ghi này có thể sử dụng độc lập để lưu trữ dữ liệu và có thể kết hợp lại tạo thành 1 thanh ghi 16 bit có tên là dptr và gọi là con trỏ dữ liệu - được dùng để lưu địa chỉ 16 bit khi truy xuất dữ liệu của bộ nhớ dữ liệu bên ngoài. Các ví điều khiển sau này có thêm một thanh ghi dptr.

- ***Ô nhớ có địa chỉ 87h: :***

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

- ***Các ô nhớ có địa chỉ từ 88h đến 8dh :***

Là các thanh ghi phục vụ cho 2 timer/ counter T1, T0.

Thanh ghi tcon (timer control): thanh ghi điều khiển timer / counter.

Thanh ghi tmod (timer mode): thanh ghi lựa chọn mode hoạt động cho timer / counter.

Thanh ghi TH0 và TL0 kết hợp lại tạo thành 1 thanh ghi 16 bit có chức năng lưu trữ xung đếm cho timer/counter T0. Tương tự cho 2 thanh ghi TH1 và TL1 kết hợp lại để lưu trữ xung đếm cho timer/counter T1. Khả năng lưu trữ số lượng xung đếm được là 65536 xung.

Chức năng của các thanh ghi này sẽ được trình bày rõ ở chương timer – counter.

- ***Các ô nhớ có địa chỉ từ 98h đến 99h :***

Là 2 thanh ghi scon và sbuf: scon (series control): thanh ghi điều khiển truyền dữ liệu nối tiếp. Sbuf (series buffer): thanh ghi đệm dữ liệu truyền nối tiếp. Dữ liệu muốn truyền đi thì phải lưu vào thanh ghi SBUF và dữ liệu nhận về nối tiếp cũng lưu ở thanh ghi này. Khi có sử dụng truyền dữ liệu thì phải sử dụng 2 thanh ghi này.

Chức năng của các thanh ghi này sẽ được trình bày rõ ở chương truyền dữ liệu.

- ***Các ô nhớ có địa chỉ từ a8h đến b9h :***

Là 2 thanh ghi IE và IP – thanh ghi IE (interrupt enable): thanh ghi điều khiển cho phép / không cho phép ngắt. IP (interrupt priority): thanh ghi điều khiển ưu tiên ngắt. Khi có sử dụng đến ngắt thì phải dùng đến 2 thanh ghi này. Mặc nhiên các thanh ghi này được khởi tạo ở chế độ cấm ngắt.

Chức năng của các thanh ghi này sẽ được trình bày rõ ở chương ngắt.

- ***Thanh ghi trạng thái chương trình (PSW: Program Status Word):***

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

BIT	SYMBOL	ADDRESS	DESCRIPTION
PSW.7	C hoặc Cy	D7H	Cary Flag
PSW.6	AC	D6H	Auxiliary Cary Flag
PSW.5	F0	D5H	Flag 0 còn gọi là cờ Zero kí hiệu là Z
PSW4	RS1	D4H	Register Bank Select 1: bit lựa chọn bank thanh ghi.
PSW.3	RS0	D3H	Register Bank Select 0: bit lựa chọn bank thanh ghi.
			00 = Bank 0; ô nhớ có address 00H÷07H gán cho R0-R7
			01 = Bank 1; ô nhớ có address 08H÷0FH gán cho R0-R7
			10 = Bank 2; ô nhớ có address 10H÷17H gán cho R0-R7
			11 = Bank 3; ô nhớ có address 18H÷1FH gán cho R0-R7
PSW.2	OV	D2H	Overflow Flag: cờ tràn số nhị phân có dấu.
PSW.1	-	D1H	Reserved: chưa thiết kế nên chưa sử dụng được.
PSW.0	P	D0H	Even Parity Flag: cờ chẵn lẻ.

Chức năng từng bit trạng thái:

- *Cờ Carry CY (Carry Flag):*

Cờ nhớ có tác dụng kép. Cờ C được sử dụng cho các lệnh toán học:

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

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

- *Cờ Carry phụ AC (Auxiliary Carry Flag):*

Khi cộng những giá trị BCD (Binary Code Decimal), cờ nhớ phụ AC được set [AC=1] nếu kết quả 4 bit lớn hơn 09H, ngược lại AC= 0. Cờ AC được dùng để chỉnh số BCD khi thực hiện lệnh cộng 2 số BCD.

- *Cờ 0 (Flag 0):*

Cờ 0 (F0) còn gọi là cờ zero, cờ zero =1 khi kết qủa xử lý bằng 0 và cờ zero = 0 khi kết quả xử lý khác 0.

- *Các bit chọn bank thanh ghi truy xuất:*

Hai bit RS1 và RS0 dùng để thay đổi cách gán 8 thanh ghi R7 – R0 cho 1 trong 4 bank thanh ghi. Hai bit này sẽ bị xóa sau khi reset vi điều khiển và được thay đổi bởi chương trình của người lập trình.

Hai bit RS1, RS0 = 00, 01, 10, 11 sẽ được chọn Bank thanh ghi tích cực tương ứng là Bank0, Bank1, Bank2, Bank3.

RS1	RS0	Bank thanh ghi được lựa chọn
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

- *Cờ tràn OV (Over Flag) :*

Khi các số có dấu được cộng hoặc trừ với nhau, phần mềm có thể kiểm tra bit này để xác định xem kết quả có nằm trong vùng giá trị xác định hay không. Với số nhị phân 8 bit có dấu thì số dương từ 0 đến +127, số âm từ -128 đến – 1. Nếu kết quả cộng 2 số dương lớn hơn +127 hoặc cộng 2 số âm kết quả nhỏ hơn –128 thì kết quả đã vượt ra ngoài vùng giá trị cho phép thì khối ALU trong vi điều khiển sẽ làm bit OV = 1.

Khi cộng các số nhị phân không dấu thì không cần quan tâm đến bit OV.

- *Bit Parity (P) :*

Bit P tự động được Set hay Clear ở mỗi chu kỳ máy để lập Parity chẵn với thanh ghi A. Đếm các bit 1 trong thanh ghi A cộng với bit Parity luôn luôn là số chẵn. Ví dụ thanh ghi A chứa nhị phân 10101101B thì bit P set lên một để cho biết tổng số bit 1 trong thanh ghi A và cả bit P tạo thành số chẵn.

Bit Parity thường được dùng kết hợp với những thủ tục truyền dữ liệu nối tiếp để tạo ra bit Parity cho dữ liệu trước khi truyền đi hoặc kiểm tra bit Parity sau khi nhận dữ liệu.

- ***Thanh ghi B :***

Thanh ghi B ở địa chỉ F0H được dùng cùng với thanh ghi A để thực hiện các phép toán nhân chia. Lệnh MUL AB: sẽ nhân những giá trị không dấu 8 bit với 8 bit trong hai thanh ghi A và B, rồi trả về kết quả 16 bit trong A (byte cao) và B(byte thấp). Lệnh DIV AB: lấy giá trị trong thanh ghi A chia cho giá trị trong thanh ghi B, kết quả nguyên lưu trong A, số dư lưu trong B.

Thanh ghi B có thể được dùng như một thanh ghi đệm trung gian nhiều chức năng.

Tóm tắt chương 3:

- Vi điều khiển có tất cả các thành phần cấu trúc bên trong giống như vi xử lý như khối ALU, các thanh ghi: thanh ghi A, thanh ghi bộ nhớ chương trình PC, thanh ghi con trỏ ngăn xếp SP, ...
- Vi điều khiển có tích hợp bộ nhớ nội bên trong bao gồm bộ nhớ chương trình và bộ nhớ ram.
- Bộ nhớ chương trình dùng để chứa chương trình điều khiển.
- Bộ nhớ Ram dùng để lưu trữ dữ liệu phục vụ cho việc xử lý chương trình.
- Kích thước của bộ nhớ chương trình bên trong tùy thuộc vào từng loại vi điều khiển cụ thể.
- Ngoài các bộ nhớ bên trong vi điều khiển còn có thể giao tiếp với bộ nhớ mở rộng bên ngoài. Khi giao tiếp với bộ nhớ bên ngoài thì port 0 và port 2 có chức năng giao tiếp địa chỉ và dữ liệu và các đường điều khiển của port 3, số lượng đường điều khiển IO còn lại rất ít. Muốn có nhiều đường điều khiển IO thì phải giao tiếp thêm IC ngoại vi.
- Vi điều khiển có 32 đường IO, có timer/counter, có truyền dữ liệu nối tiếp, có ngắt cho phép giao tiếp dễ dàng với đối tượng điều khiển.
- Một trong những thế mạnh của vi điều khiển là khả năng xử lý dữ liệu bit – rất phù hợp trong lãnh vực điều khiển.

Chương 4

KHẢO SÁT TẬP LỆNH CỦA VI ĐIỀU KHIỂN

I. CÁC KHÁI NIỆM

Vi điều khiển hay vi xử lý là các IC lập trình, khi bạn đã thiết kế hệ thống điều khiển có sử dụng vi xử lý hay vi điều khiển ví dụ như hệ thống điều khiển đèn giao thông cho một ngã tư gồm có các đèn Xanh, Vàng, Đỏ và các led 7 đoạn để hiển thị thời gian thì đó mới chỉ là phần cứng, muốn hệ thống vận hành thì bạn phải viết một chương trình điều khiển nạp vào bộ nhớ nội bên trong vi điều khiển hoặc bộ nhớ bên ngoài và gắn vào trong hệ thống để hệ thống vận hành và dĩ nhiên bạn phải viết đúng thì hệ thống mới vận hành đúng. Chương trình gọi là phần mềm.

Phần mềm và phần cứng có quan hệ với nhau, người lập trình phải hiểu rõ hoạt động của phần cứng để viết chương trình. Ở chương này sẽ trình bày chi tiết về tập lệnh của vi điều khiển giúp bạn hiểu rõ từng lệnh để bạn có thể lập trình được.

Các khái niệm về chương trình, lệnh, tập lệnh và ngôn ngữ gợi nhớ đã trình bày ở chương 1 và 2, ở đây chỉ tóm tắt lại.

Chương trình là một tập hợp các lệnh được tổ chức theo một trình tự hợp lí để giải quyết đúng các yêu cầu của người lập trình.

Người lập trình là người biết giải thuật để viết chương trình và sắp xếp đúng các lệnh theo giải thuật. Người lập trình phải biết chức năng của tất cả các lệnh của vi điều khiển để viết chương trình.

Tất cả các lệnh có thể có của một ngôn ngữ lập trình còn gọi là ***tập lệnh***.

Họ vi điều khiển MCS-51 đều có chung 1 tập lệnh, các vi điều khiển thế hệ sau chỉ phát triển nhiều về phần cứng còn lệnh thì ít mở rộng.

Tập lệnh họ MCS-51 có mã lệnh 8 bit nên có khả năng cung cấp 2^8= 256 lệnh.

Có lệnh có 1 hoặc 2 byte bởi dữ liệu hoặc địa chỉ thêm vào Opcode.

Trong toàn bộ tập lệnh của vi điều khiển có139 lệnh 1 byte, 92 lệnh 2 byte và 24 lệnh 3 byte.

Lệnh của vi điều khiển là một số nhị phân 8 bit [còn gọi là mã máy]. 256 byte từ 0000 0000b đến 1111 1111b tương ứng với 256 lệnh khác nhau. Do mã lệnh dạng số nhị phân quá dài và khó nhớ nên các nhà lập trình đã xây dựng một ngôn ngữ lập trình Assembly cho dễ nhớ, điều này giúp cho việc lập trình được thực hiện một cách dễ dàng và nhanh chóng cũng như đọc hiểu và gỡ rối chương trình.

Khi viết chương trình bằng ngôn ngữ lập trình Assembly thì vi điều khiển sẽ không thực hiện được mà phải dùng chương trình biên dịch Assembler để chuyển đổi các lệnh viết bằng Assembly ra mã lệnh nhị phân tương ứng rồi nạp vào bộ nhớ – khi đó vi điều khiển mới thực hiện được chương trình.

Ngôn ngữ lập trình Assembly do con người tạo ra, khi sử dụng ngôn ngữ Assembly để viết thì người lập trình vi điều khiển phải học hết tất cả các lệnh và viết đúng theo qui ước về cú pháp, trình tự sắp xếp dữ liệu để chương trình biên dịch có thể biên dịch đúng.

II. CÁC KIỂU ĐỊNH ĐỊA CHỈ BỘ NHỚ CỦA VI ĐIỀU KHIỂN:

Phần này đã trình bày một cách tổng quát ở chương 2, ở đây sẽ trình bày một cách chi tiết hơn. Các kiểu định địa chỉ là một qui ước thống nhất của tập lệnh.

Các kiểu định địa chỉ cho phép định rõ nơi lấy dữ liệu hoặc nơi nhận dữ liệu tùy thuộc vào cách thức sử dụng lệnh của người lập trình.

Vi điều khiển họ MCS-51 có 8 kiểu định địa chỉ như sau:

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

a. Kiểu định địa chỉ dùng thanh ghi (Register Addressing) :

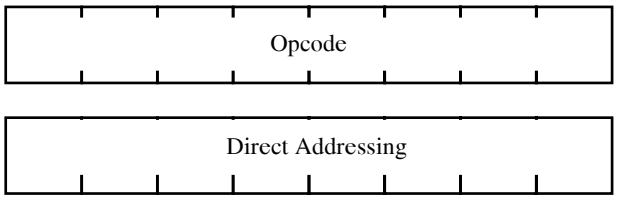
Kiểu này thường được dùng cho các lệnh xử lý dữ liệu mà dữ liệu luôn lưu trong ***các thanh ghi***. Đối với vi điều khiển thì mã lệnh thuộc kiểu này chỉ có 1 byte.

```
Ví dụ:    Mov A,R1          ; copy nội dung thanh ghi R1 vào thanh ghi A
```

b. Kiểu định địa chỉ trực tiếp (Direct Addressing) :

Kiểu này thường được dùng để truy xuất dữ liệu của bất kỳ ô nhớ nào trong 256 byte bộ nhớ RAM nội của vi điều khiển 89C51.

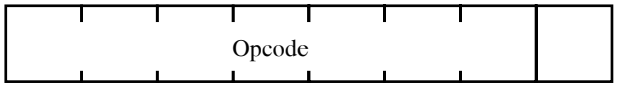
Các lệnh thuộc kiểu này thường có mã lệnh 2 byte: byte thứ nhất là mã lệnh, byte thứ 2 là ***địa chỉ của ô nhớ***:




```
Ví dụ:    Mov A,05H         ; copy nội dung ô nhớ có địa chỉ 05H vào thanh ghi A
```

c. Định địa chỉ gián tiếp (Indirect Addressing) :

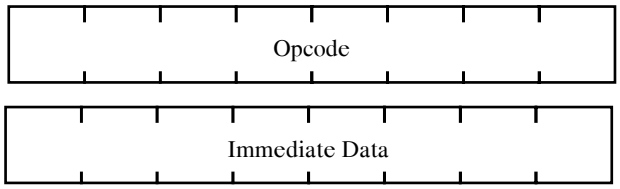
Kiểu định địa chỉ gián tiếp được tượng trưng bởi ký hiệu @ và được đặt trước các thanh ghi R0, R1 hay DPTR. R0 và R1 có thể hoạt động như một thanh ghi con trỏ, nội dung của nó cho biết địa chỉ của một ô nhớ trong RAM nội mà dữ liệu sẽ ghi hoặc sẽ đọc. Còn dptr dùng để truy xuất ô nhớ ngoại. Các lệnh thuộc dạng này chỉ có 1 byte.



```
Ví dụ:    Mov A,@R1          ; copy nội dung ô nhớ có địa chỉ trong thanh ghi R1 vào
                             ;thanh ghi A
```

d. Định địa chỉ tức thời (Immediate Addressing) :

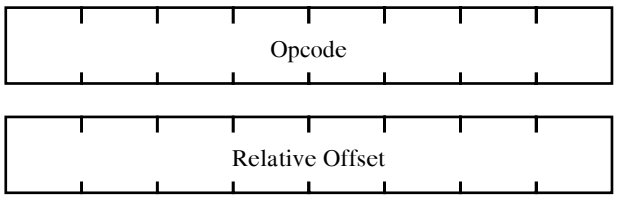
Kiểu định địa chỉ tức thời được tượng trưng bởi ký hiệu # và được đặt trước một hằng số. Lệnh này thường dùng để nạp 1 giá trị là 1 hằng số ở byte thứ 2 (hoặc byte thứ 3) vào thanh ghi hoặc ô nhớ.




```
Ví dụ:    Mov a,#30H         ; nạp dữ liệu là con số 30H vào thanh ghi A
```

e. Định địa chỉ tương đối :

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



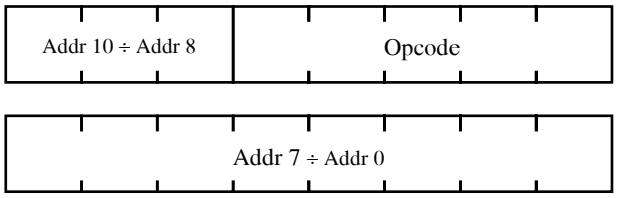
Nơi nhảy đến thường được xác định bởi nhãn (label) và trình biên dịch sẽ tính toán giá trị lệch.

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

```
Ví dụ:    Sjmp X1            ;nhảy đến nhản có tên là X1 nằm trong tầm vực 256 byte
```

f. Định địa chỉ tuyệt đối (Absolute Addressing) :

Kiểu định địa chỉ tuyệt đối được dùng với các lệnh ACALL và AJMP. Các lệnh này có mã lệnh 2 byte cho phép phân chia bộ nhớ theo trang - mỗi trang có kích thước đúng bằng 2Kbyte so với giá trị chứa trong thanh ghi PC hiện hành. 11 bit địa chỉ A10÷A0 được thay thế cho 11 địa chỉ thấp trong thanh ghi PC nằm trong cấu trúc mã lệnh như sau:

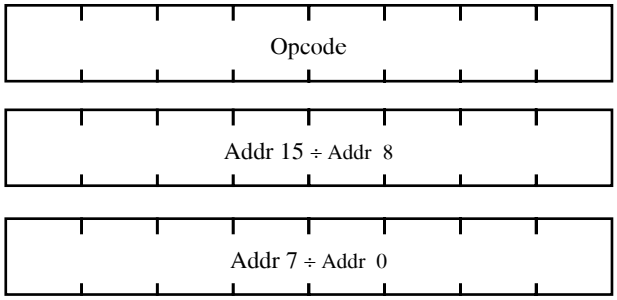


Định địa chỉ tuyệt đối có ưu điểm là mã lệnh ngắn (2 byte), nhưng khuyết điểm là mã lệnh thay đổi và giới hạn phạm vi nơi nhảy đến, gọi đến không quá 2 kbyte.

Ví dụ: Ajmp X1 ;nhảy đến nhản có tên là X1 nằm trong tầm vực 2 kbyte

g. Định địa chỉ dài (Long Addressing) :

Kiểu định địa chỉ dài được dùng với lệnh LCALL và LJMP. Các lệnh này có mã lệnh 3 byte – trong đó có 2 byte (16bit) là địa chỉ của nơi đến. Cấu trúc mã lệnh là 3 byte như sau:

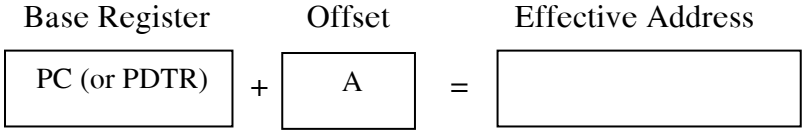


Ưu điểm của định địa chỉ dài là có thể gọi 1 chương trình con hoặc có thể nhảy đến bất kỳ vùng nhớ nào vùng nhớ 64K, nhược điểm là các lệnh kiểu này dài 3 byte và phụ thuộc vào vị trí đến – điều này sẽ bất tiện bởi không thể dời toàn bộ mã lệnh của chương trình từ vùng nhớ này sang các vùng nhớ khác – có nghĩa là khi chương trình đã viết nơi đến tại địa chỉ 1000h thì sau khi dịch ra mã lệnh dạng số nhị phân thì sau đó nạp vào bộ nhớ thì địa chỉ bắt đầu phải đúng với địa chỉ đã viết là 1000h; nếu nạp ở vùng địa chỉ khác địa chỉ 1000h thì chương trình sẽ thực hiện sai.

Ví dụ: Ljmp X1 ;nhảy đến nhản có tên là X1 nằm trong tầm vực 64kbyte

h. Định địa chỉ chỉ số (Index Addressing) :

Kiểu định địa chỉ chỉ số "dùng một thanh ghi cơ bản: là bộ đếm chương trình PC hoặc bộ đếm dữ liệu DPTR" kết hợp với "một giá trị lệch (offset) còn gọi là giá trị tương đối [thường lưu trong thanh ghi]" để tạo ra 1 địa chỉ của ô nhớ cần truy xuất hoặc là địa chỉ của nơi nhảy đến. Việc kết hợp được minh họa như sau:



Ví dụ: MOVX A, @A + DPTR ;lấy dữ liệu trong ô nhớ có địa chỉ bằng DPTR + A

Khi khảo sát tập lệnh một cách chi tiết thì chức năng của các thanh ghi và các kiểu truy xuất này sẽ được trình bày rõ ràng hơn.

III. KHẢO SÁT TẬP LỆNH VI ĐIỀU KHIỂN MCS51:

Để khảo sát tập lệnh thì phải thống nhất một số qui định về các từ ngữ kí hiệu trong tập lệnh thường được sử dụng:

- Direct tượng trưng cho ô nhớ nội có địa chỉ Direct.
- Rn tượng trưng cho các thanh ghi từ thanh ghi R0 đến thanh ghi R7.
- @Ri tượng trưng cho ô nhớ có địa chỉ lưu trong thanh ghi Ri và Ri chỉ có 2 thanh ghi là R0 và R1.
- Các lệnh thường xảy ra giữa các đối tượng sau:

  + Thanh ghi A.

  + Thanh ghi Rn.

  + Ô nhớ có địa chỉ direct.

  + Ô nhớ có địa chỉ lưu trong thanh ghi @Ri.

  + Dữ liệu 8 bit #data.

  + addr11 là địa chỉ 11 bit từ A11 – A0: địa chỉ này phục vụ cho lệnh nhảy hoặc lệnh gọi chương trình con trong phạm vi 2 kbyte.

  + Addr16 là địa chỉ 16 bit từ A15 – A0: địa chỉ này phục vụ cho lệnh nhảy và lệnh gọi chương trình con ở xa trong phạm vi 64 kbyte – đó chính là địa chỉ nhảy đến, hoặc địa chỉ của chương trình con.

Khi viết chương trình người lập trình có thể thay thế địa chỉ bằng nhản (label) để khỏi phải tính toán các địa chỉ cụ thể. Nhản (label) sẽ được đặt tại vị trí addr thay cho addr và phải có một nhản đặt tại nơi muốn nhảy đến - gọi là 1 cặp nhản cùng tên.

Có thể nhiều nơi nhảy đến cùng một nhản. Không được đặt các nhản cùng tên.

Các lệnh có ảnh hưởng đến thanh ghi trạng thái thì có trình bày trong lệnh, còn các lệnh không đề cập đến thanh ghi trạng thái thì có nghĩa là nó không ảnh hưởng.

A. Nhóm lệnh di chuyển dữ liệu (8 bit) :

1. Lệnh chuyển dữ liệu từ một thanh ghi vào thanh ghi A:

- Cú pháp : **Mov A,Rn**
- Mã lệnh :

1	1	1	0	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Chuyển nội dung của thanh ghi Rn vào thanh ghi A, nội dung thanh ghi Rn vẫn giữ nguyên.

Ví dụ: Giả sử thanh ghi R0 có nội dung là 32h , lệnh:

Mov A,R0 ;kết quả như sau: (A) = 32h, (R0) = 32h.

Giá trị ban đầu chứa trong A thì không cần quan tâm.

2. Lệnh chuyển dữ liệu từ ô nhớ trực tiếp vào thanh ghi A :

- Cú pháp : **Mov A, direct**
- Mã lệnh :

1	1	1	0	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Chuyển nội dung của ô nhớ trong Ram nội có địa chỉ direct ở byte thứ hai vào thanh ghi A. Trực tiếp có nghĩa là địa chỉ của ô nhớ được ghi ở trong lệnh.

Ví dụ : Giả sử ô nhớ có địa chỉ 30h lưu nội dung 32h. Lệnh:

```
Mov A,30h      ;chuyển nội dung của ô nhớ có địa chỉ là 30h sang thanh ghi A.
               ;Kết quả như sau: (A)= 32h. chú ý địa chỉ 30h ghi trong lệnh.
               ;a7..a0 = 00110000b là địa chỉ của ô nhớ 30h
```

3. Lệnh chuyển dữ liệu từ ô nhớ gián tiếp vào thanh ghi A :

- Cú pháp : **MOV A,@Ri**
- Mã lệnh :

1	1	1	0	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Chuyển nội dung ô nhớ trong Ram nội, có địa chỉ chứa trong thanh ghi Ri, vào thanh ghi A.
  - Ví dụ1: thay vì thực hiện "mov A,30h" của ví dụ trên thì ta có thể thay bằng lệnh "mov A,@R0" sẽ có cùng 1 kết quả nếu địa chỉ 30h lưu vào R0. Địa chỉ 30h không còn ghi trong lệnh mà được thay bằng @R0 – nên kiểu lệnh này gọi là lệnh gián tiếp vì địa chỉ 30h không còn xuất hiện trong lệnh. Chú ý trước khi sử dụng lệnh này ta phải làm cho R0 mang giá trị là 30h.

Ví dụ : Giả sử R0 có nội dung là 70h, ô nhớ có địa chỉ 70h chứa nội dung là 0B8h. Lệnh:

```
Mov A,@R0      ;kết quả như sau: (A) = 0B8h.
```

4. Lệnh nạp dữ liệu 8 bit vào thanh ghi A :

- Cú pháp : **MOV A, #data**
- Mã lệnh :

0	1	1	1	0	1	0	0
d7	d6	d5	d4	d3	d2	d1	d0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nạp dữ liệu 8 bit data (d0 đến d7) vào thanh ghi A.

Ví dụ : Giả sử A có nội dung 47h, dữ liệu trực tiếp là 32h, lệnh:

```
Mov A,#32h     ;kết quả như sau: (A) = 32h.
               ;d7..d0 = 00110010b
```

5. Lệnh chuyển dữ liệu từ thanh ghi A vào thanh ghi :

- Cú pháp : **Mov Rn, A**
- Mã lệnh :

1	1	1	1	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Chuyển nội dung của thanh ghi A vào thanh ghi Rn.

Ví dụ : Giả sử A có nội dung 47h , lệnh:

Mov R0, A ;kết quả như sau: (R0) = 47h, (A) = 47h.

6. Lệnh chuyển dữ liệu từ ô nhớ trực tiếp vào thanh ghi Rn :

- Cú pháp : **MOV Rn, direct**
- Mã lệnh :

1	0	1	0	1	n2	n1	n0
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Chuyển nội dung của ô nhớ trong Ram nội có địa chỉ direct vào thanh ghi Rn.

Ví dụ : Giả sử R1 có nội dung 47h, ô nhớ có địa chỉ 30h chứa nội dung 0afh. Lệnh:

Mov R1,30h

Lệnh chuyển nội dung ô nhớ có địa chỉ 30h sang thanh ghi R1.

Kết quả như sau: (R1) = 0afh, dữ liệu trong ô nhớ có địa chỉ 30h không đổi.

7. Lệnh chuyển tức thời dữ liệu 8 bit vào thanh ghi Rn :

- Cú pháp : **MOV Rn, #data**
- Mã lệnh :

0	1	1	1	1	n2	n1	n0
d7	D6	d5	d4	D3	d2	d1	d0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nạp dữ liệu 8 bit data (d0 đến d7) vào thanh ghi Rn.

Ví dụ: Giả sử muốn chuyển dữ liệu 47h vào thanh ghi R1:

Mov R1,#47h ;kết quả như sau: (R1)= 47h.

8. Lệnh chuyển dữ liệu từ thanh ghi A vào ô nhớ trực tiếp :

- Cú pháp : **MOV direct, A**
- Mã lệnh :

1	1	1	1	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Chuyển nội dung của thanh ghi A vào ô nhớ trong Ram nội có địa chỉ direct.

Ví dụ : Cho nội dung thanh ghi (A) = 35H, nội dung ô nhớ có địa chỉ 10H bằng 50H.

Mov 10h,A

Sau khi thực hiện xong thì nội dung ô nhớ có địa chỉ 10h bằng 35H.

9. Lệnh chuyển dữ liệu từ thanh ghi Rn vào ô nhớ trực tiếp :

- Cú pháp : **MOV direct, Rn**
- Mã lệnh :

1	0	0	0	1	n2	n1	n0
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Chuyển nội dung của thanh ghi Rn vào ô nhớ trong Ram nội có địa chỉ direct.

Ví dụ : Cho nội dung thanh ghi (R0) = 35H, nội dung ô nhớ 10H bằng 50H.

Mov 10h,R0

Sau khi thực hiện xong thì nội dung ô nhớ có địa chỉ 10h bằng 35H.

10. Lệnh chuyển dữ liệu từ ô nhớ trực tiếp vào ô nhớ trực tiếp :

- Cú pháp : **MOV direct, direct**
- Mã lệnh :

1	0	0	0	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Chuyển nội dung của ô nhớ trong Ram nội có địa chỉ direct vào ô nhớ có địa chỉ trực direct.

Ví dụ : Cho nội dung ô nhớ có địa chỉ 20H bằng 35H và nội dung ô nhớ có địa chỉ 10H bằng 50H.

Mov 10h,20h

Sau khi thực hiện xong thì nội dung ô nhớ có địa chỉ 10h bằng 35H.

11. Lệnh chuyển dữ liệu từ ô nhớ gián tiếp vào ô nhớ trực tiếp :

- Cú pháp : **MOV direct, @Ri**
- Mã lệnh :

1	0	0	0	0	1	1	i
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Chuyển nội dung ô nhớ có địa chỉ chứa trong thanh ghi Ri vào ô nhớ có địa chỉ direct.

Ví dụ : Cho nội dung thanh ghi (R0) = 05H, nội dung ô nhớ có địa chỉ 05h bằng FFH và nội dung ô nhớ có địa chỉ 10H bằng 50H.

Mov 10h,@r0

Sau khi thực hiện xong thì nội dung ô nhớ có địa chỉ 10h bằng FFH.

12. Lệnh chuyển dữ liệu vào ô nhớ trực tiếp :

- Cú pháp : **MOV direct, #data**
- Mã lệnh :

0	1	1	1	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0
d7	d6	d5	d4	d3	d2	d1	d0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Nạp dữ liệu data 8 bit (d0 đến d7) vào ô nhớ có địa chỉ direct.

Ví dụ : Cho nội dung ô nhớ có địa chỉ 05h bằng FFH.

Mov 05h,#25H

Sau khi thực hiện xong thì nội dung ô nhớ có địa chỉ 05h bằng 25H.

13. Lệnh chuyển dữ liệu từ thanh ghi A vào ô nhớ gián tiếp :

- Cú pháp : **MOV @Ri, A**
- Mã lệnh :

1	1	1	1	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng : Chuyển nội dung của thanh ghi A vào ô nhớ trong Ram nội có địa chỉ chứa trong thanh ghi Ri.

14. Lệnh chuyển dữ liệu từ ô nhớ trực tiếp vào ô nhớ gián tiếp :

- Cú pháp : **MOV @Ri, direct**
- Mã lệnh :

1	0	1	0	0	1	1	i
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Chuyển nội dung ô nhớ có địa chỉ direct vào ô nhớ có địa chỉ chứa trong thanh ghi Ri.

15. Lệnh chuyển dữ liệu tức thời vào ô nhớ gián tiếp :

- Cú pháp : **MOV @Ri, #data**
- Mã lệnh :

0	1	1	1	0	1	1	I
d7	d6	d5	d4	d3	d2	d1	D0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nạp dữ liệu data 8 bit (d0 đến d7) vào ô nhớ có địa chỉ chứa trong thanh ghi Ri.

16. Lệnh chuyển dữ liệu tức thời 16 bit vào thanh ghi con trỏ dữ liệu :

- Cú pháp : **MOV dptr, #data16**
- Mã lệnh :

1	0	0	1	0	0	0	0
d15	d14	d13	d12	d11	d10	d9	d8
d7	d6	d5	d4	d3	d2	d1	d0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Nạp dữ liệu data 16 bit vào thanh ghi con trỏ dữ liệu dptr.

17. Lệnh chuyển dữ liệu từ ô nhớ có địa chỉ là Dptr + A vào thanh ghi A :

- Cú pháp : **MOVC A,@A+DPTR**
- Mã lệnh :

1	0	0	1	0	0	1	1

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng : chuyển nội dung của ô nhớ ngoài, có địa chỉ chứa bằng dptr cộng với giá trị chứa trong A, chuyển vào thanh ghi A.

18. Lệnh chuyển dữ liệu từ ô nhớ có địa chỉ là PC + A vào thanh ghi A :

- Cú pháp : **MOVC A,@A+PC**
- Mã lệnh :

1	0	0	0	0	0	1	1

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng : chuyển nội dung của ô nhớ ngoài có địa chỉ chứa bằng PC cộng với giá trị chứa trong A được chuyển vào thanh ghi A.

19. Lệnh chuyển dữ liệu từ ô nhớ ngoài gián tiếp (8 bit địa chỉ) vào thanh ghi A :

- Cú pháp : **MOVX A, @Ri**
- Mã lệnh :

1	1	1	0	0	0	1	I

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng : chuyển nội dung ô nhớ ngoài có địa chỉ chứa trong thanh ghi Ri vào thanh ghi A.

20. Lệnh chuyển dữ liệu từ ô nhớ ngoài gián tiếp (16 bit địa chỉ) vào thanh ghi A :

- Cú pháp : **MOVX A,@DPTR**
- Mã lệnh :

1	1	1	0	0	0	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng : chuyển nội dung của ô nhớ ngoài có địa chỉ chứa trong thanh ghi dptr vào thanh ghi A.

21. Lệnh chuyển dữ liệu từ thanh ghi A vào ô nhớ ngoài gián tiếp (8 bit địa chỉ) :

- Cú pháp : **MOVX @ Ri, A**
- Mã lệnh :

1	1	1	1	0	0	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng : chuyển nội dung của thanh ghi A ra ô nhớ ngoài có địa chỉ chứa trong thanh ghi Ri.

22. Lệnh chuyển dữ liệu từ thanh ghi A vào ô nhớ ngoài gián tiếp (16 bit địa chỉ) :

- Cú pháp : **MOVX @DPTR, A**
- Mã lệnh :

1	1	1	1	0	0	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng : Chuyển nội dung của thanh ghi A ra ô nhớ ngoài có địa chỉ chứa trong thanh ghi dptr.

23. Lệnh cất nội dung ô nhớ trực tiếp vào ngăn xếp :

- Cú pháp : **PUSH direct**
- Mã lệnh :

1	1	0	0	0	0	0	0
a7	a6	A5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: cất nội dung của ô nhớ có địa chỉ direct vào ô nhớ ngăn xếp. Con trỏ ngăn xếp SP tăng lên 1 trước khi lưu nội dung.

24. Lệnh lấy dữ liệu từ ngăn xếp trả về ô nhớ trực tiếp :

- Cú pháp : **POP direct**
- Mã lệnh :

1	1	0	1	0	0	0	0
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: lấy nội dung của ô nhớ ngăn xếp trả cho ô nhớ có địa chỉ direct. Con trỏ ngăn xếp SP giảm 1 sau khi lấy dữ liệu ra.

25. Lệnh trao đổi dữ liệu giữa thanh ghi với thanh ghi A :

- Cú pháp : **XCH A,Rn**
- Mã lệnh :

1	1	0	0	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng : Trao đổi nội dung của thanh ghi Rn với thanh ghi A.
- Ví dụ : cho nội dung của thanh ghi (A) = 35H và (R0) = 70H

  XCH A,R0

  Kết quả sau khi thực hiện (A) = 70H và (R0) = 35H

26. Lệnh trao đổi dữ liệu giữa ô nhớ trực tiếp với thanh ghi A :

- Cú pháp : **XCH A,Direct**
- Mã lệnh :

1	1	0	0	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng : Trao đổi nội dung của thanh ghi A với nộ dung ô nhớ có địa chỉ direct.

27. Lệnh trao đổi dữ liệu giữa ô nhớ gián tiếp với thanh ghi A :

- Cú pháp : **XCH A,@Ri**
- Mã lệnh :

1	1	0	0	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng : Trao đổi nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri với thanh ghi A.

28. Lệnh trao đổi 4 bit dữ liệu giữa ô nhớ gián tiếp với thanh ghi A :

- Cú pháp : **XCHD A,@Ri**
- Mã lệnh :

1	1	0	1	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng : Trao đổi dữ liệu 4 bit thấp của ô nhớ có địa chỉ chứa trong thanh ghi Ri với dữ liệu 4 bit thấp trong thang ghi A.

B. Nhóm lệnh số học :

1. Lệnh cộng thanh ghi A với thanh ghi :

- Cú pháp : **ADD A,Rn**
- Mã lệnh :

0	0	1	0	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng : cộng nội dung thanh ghi A với nội dung thanh ghi Rn, kết quả lưu trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ 1: Giả sử A có nội dung 47h và R0 có nội dung là 32h, lệnh:
ADD A,R0 ;kết quả như sau: (A) = 79h, (C) = 0.
Ví dụ 2: Giả sử A có nội dung 0D9h và R0 có nội dung là 0B8h, lệnh:
ADD A,R0 ;kết quả như sau: (A) = 91h, (C) =1.

2. Lệnh cộng nội dung ô nhớ trực tiếp vào thanh ghi A :

- Cú pháp : **ADD A, direct**
- Mã lệnh :

0	0	1	0	0	1	0	1

a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Cộng nội dung của ô nhớ có địa chỉ direct với nội dung thanh ghi A, kết quả chứa ở thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ 1: Giả sử A có nội dung 0D9h và ô nhớ có địa chỉ 30h lưu nội dung 0B8h, lệnh:

ADD A,30h ;kết quả như sau: (A) = 81h, (C) =1.

Ví dụ 2: Giả sử A có nội dung 47h và ô nhớ có địa chỉ 30h lưu nội dung 32h, lệnh:

ADD A,30h ;kết quả như sau: (A) = 79h, (C) =0.

3. Lệnh cộng nội dung ô nhớ gián tiếp vào thanh ghi A :

- Cú pháp: **ADD A,@Ri**
- Mã lệnh:

0	0	1	0	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: cộng nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri với thanh ghi A, kết qủa lưu trữ trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ: Giả sử A có nội dung 0D9h, ô nhớ có địa chỉ 30h có nội dung là 0B8h, R0 có nội dung là 30h, lệnh:

ADD A,@R0 ;kết qủa như sau: (A) = 91h, (C) =1.

4. Lệnh cộng dữ liệu tức thời 8 bit vào thanh ghi A :

- Cú pháp : **ADD A, #data**
- Mã lệnh :

0	0	1	0	0	1	0	0
d7	d6	d5	d4	d3	d2	d1	d0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Cộng dữ liệu data 8 bit (d0 đến d7) với nội dung thanh ghi A, kết quả lưu trữ trong A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ1: Giả sử A có nội dung 47h, dữ liệu trực tiếp là 32h, lệnh:
ADD A,#32h ;kết quả như sau: (A) = 79h, (C) = 0.
Ví dụ2: Giả sử A có nội dung D9h, dữ liệu trực tiếp là B8h, lệnh:
ADD A,#0B8h ;kết quả như sau: (A) = 91h, (C) = 1.

5. Lệnh cộng thanh ghi A với thanh ghi có bit carry :

- Cú pháp : **ADDC A,Rn**
- Mã lệnh :

0	0	1	1	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng : cộng nội dung thanh ghi A với nội dung thanh ghi Rn với bit C, kết quả lưu trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ1: Giả sử A có nội dung 47h, R1 có nội dung 32h và cờ (C) = 1, lệnh:

ADDC A,R1 ;kết quả như sau: (A) = 7ah, (C) = 0.

Ví dụ2: Giả sử A có nội dung 0D9h, R0 có nội dung là 0B8h, (C) =1, lệnh:

ADDC A,R0 ;kết quả như sau: (A) = 92h, (C)=1.

6. Lệnh cộng nội dung ô nhớ trực tiếp vào thanh ghi A có bit carry :

- Cú pháp : **ADDC A, direct**
- Mã lệnh :

0	0	1	1	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Cộng nội dung của ô nhớ có địa direct nội dung thanh ghi A và bit C, kết quả chứa ở thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ 1: Giả sử A có nội dung 47h, ô nhớ 30h có nội dung 32h và cờ (C) = 0, lệnh:

ADDC A,30h ;kết quả như sau: (A) = 79h, (C) = 0.

Ví dụ 2: Giả sử A có nội dung 0D9h, ô nhớ 30h có nội dung là 0B8h, C:=1, lệnh:

ADDC A,30h ;kết quả như sau: (A) = 92h, (C) = 1.

7. Lệnh cộng nội dung ô nhớ gián tiếp vào thanh ghi A có bit carry :

- Cú pháp : **ADDC A,@Ri**
- Mã lệnh :

0	0	1	1	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng : cộng nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri với thanh ghi A với bit C, kết quả lưu trữ trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ 1: Giả sử A có nội dung 47h, ô nhớ 30h có nội dung 32h, R0 có nội dung là 30h và cờ (C) = 0, lệnh:

ADDC A,@R0 ;kết quả như sau: (A) = 79h, (C) = 0.

Ví dụ 2: Giả sử A có nội dung 0D9h, ô nhớ 30h có nội dung là 0B8h, R0 có nội dung 30h và (C) =1, lệnh:

ADDC A,@R0 ;kết quả như sau: (A) = 92h, (C)=1.

8. Lệnh cộng dữ liệu 8 bit vào thanh ghi A có bit carry :

- Cú pháp : **ADDC A, #data**
- Mã lệnh :

0	0	1	1	0	1	0	0
d7	d6	d5	d4	d3	d2	d1	d0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy.

- Chức năng: Cộng dữ liệu data 8 bit (d0 đến d7) với nội dung thanh ghi A và bit C, kết quả lưu trữ trong A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

9. Lệnh trừ thanh ghi A với thanh ghi :

- Cú pháp : **SUBB A,Rn**
- Mã lệnh :

1	0	0	1	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng : Trừ nội dung thanh ghi A với nội dung thanh ghi Rn và trừ cho cờ Carry, kết quả lưu trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

Ví dụ 1: Giả sử A có nội dung 47h, nội dung thanh ghi R0 là 32h và cờ (C)=0, lệnh:

SUBB A,R0 ;kết quả như sau: (A) = 15h (C)=0.

Ví dụ 2: Giả sử A có nội dung 0B9h, thanh ghi R0 có nội dung là 5Ah và (C)=1, lệnh:

SUBB A,R0 ;kết quả như sau: (A) = 5Eh, (C) =0.

10. Lệnh trừ nội dung thanh ghi A cho nội dung ô nhớ trực tiếp :

- Cú pháp : **SUBB A, direct**
- Mã lệnh :

1	0	0	1	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Trừ nội dung thanh ghi A cho nội dung của ô nhớ có địa chỉ direct và trừ cho cờ Carry, kết quả chứa ở thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

11. Lệnh trừ nội dung thanh ghi A cho nội dung ô nhớ gián tiếp :

- Cú pháp : **SUBB A,@Ri**
- Mã lệnh :

1	0	0	1	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Trừ nội dung của thanh ghi A cho dữ liệu của ô nhớ có địa chỉ chứa trong thanh ghi Ri và trừ cho cờ carry, kết qủa lưu trữ trong thanh ghi A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

12. Lệnh trừ nội dung thanh ghi A cho dữ liệu tức thời 8 bit :

- Cú pháp : **SUBB A, #data (subbtract: trừ)**
- Mã lệnh :

1	0	0	1	0	1	0	0
d7	d6	d5	d4	d3	d2	D1	d0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

- Chức năng: Trừ nội dung thanh ghi A cho dữ liệu 8 bit d0 đến d7 và trừ cho cờ carry, kết qủa lưu trữ trong A. Lệnh có ảnh hưởng đến thanh ghi trạng thái.

13. Lệnh tăng nội dung thanh ghi A :

- Cú pháp : **INC A** **(increment: tăng lên 1 đơn vị)**
- Mã lệnh :

0	0	0	0	0	1	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Tăng nội dung thanh ghi A lên 1.

Ví dụ 1: Giả sử A có nội dung 35h, lệnh:

INC A ;kết qủa như sau: (A) = 36h.

Ví dụ 2: Giả sử A có nội dung FFh, lệnh:

INC A ;kết qủa như sau: (A) = 00h.

14. Lệnh tăng nội dung của thanh ghi :

- Cú pháp : **INC Rn**
- Mã lệnh :

0	0	0	0	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Tăng nội dung thanh ghi Rn lên 1.

15. Lệnh tăng nội dung ô nhớ trực tiếp :

- Cú pháp : **INC direct**
- Mã lệnh :

0	0	0	0	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Tăng nội dung của ô nhớ có địa chỉ trực tiếp ở byte thứ 2 lên 1.

16. Lệnh tăng nội dung ô nhớ gián tiếp :

- Cú pháp : **INC @Ri**
- Mã lệnh :

0	0	0	0	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Tăng nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri lên 1.

Ví dụ : Giả sử nội dung ô nhớ 30h là 35h, thanh ghi R0 có nội dung là 30h, lệnh:

INC R0 ;kết quả như sau: ô nhớ (30h) = 36h

17. Lệnh tăng nội dung con trỏ dữ liệu Dptr :

- Cú pháp : **INC dptr**

- Mã lệnh :

1	0	1	0	0	0	1	1

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Tăng nội dung của thanh ghi con trỏ dữ liệu dptr lên 1.

18. Lệnh giảm nội dung thanh ghi A :

- Cú pháp : **DEC A**
- Mã lệnh :

0	0	0	1	0	1	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Giảm nội dung thanh ghi A xuống 1.

Ví dụ 1: Giả sử A có nội dung 35h, lệnh:

DEC A ;kết qủa như sau: (A) = 34h.

Ví dụ 2: Giả sử A có nội dung 00h, lệnh:

DEC A ;kết qủa như sau: (A) = FFh.

19. Lệnh giảm nội dung của thanh ghi :

- Cú pháp : **DEC Rn**
- Mã lệnh :

0	0	0	1	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy.
- Chức năng: Giảm nội dung thanh ghi Rn xuống 1.

Ví dụ : Giả sử R0 là 35h , lệnh:

DEC R0 ;kết quả như sau: (R0) =34h.

20. Lệnh giảm nội dung ô nhớ trực tiếp :

- Cú pháp : **DEC direct**
- Mã lệnh :

0	0	0	1	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Giảm nội dung của ô nhớ có địa chỉ direct ở byte thứ 2 xuống 1.

Ví dụ: Giả sử ô nhớ 30h có nội dung là 35h , lệnh:

DEC 30h ;kết quả như sau: ô nhớ có địa chỉ là 30h lưu 34h.

21. Lệnh giảm nội dung ô nhớ gián tiếp :

- Cú pháp : **DEC @Ri**
- Mã lệnh :

0	0	0	1	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Giảm nội dung của ô nhớ có địa chỉ chứa trong thanh ghi Ri xuống 1.

22. Lệnh nhân thanh ghi A với thanh ghi B :

- Cú pháp : **MUL AB**
- Mã lệnh :

1	0	1	0	0	1	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 4 chu kỳ máy
- Chức năng: Nội dung của thanh ghi A nhân với nội dung của thanh ghi B, kết qủa là một dữ liệu 16 bit, 8 bit thấp lưu trữ trong thanh ghi A, 8 bit cao lưu trữ trong thanh ghi B.

Ví dụ : Giả sử thanh ghi A có nội dung là 50h,thanh ghi B có nội dung 0A0h , lệnh:

MUL AB ;Kết qủa như sau: 50h*A0h:= 3200h thì (A) = 00 và (B) = 32h.

23. Lệnh chia thanh ghi A cho thanh ghi B :

- Cú pháp : **DIV AB**
- Mã lệnh :

1	0	0	0	0	1	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 4 chu kỳ máy
- Chức năng: Nội dung của thanh ghi A chia cho nội dung của thanh ghi B, kết qủa của phép chia lưu trữ trong thanh ghi A, số dư lưu trữ trong thanh ghi B. Lệnh ảnh hưởng đến thanh ghi trạng thái: Bit C và bit OV bị xóa về 0, nếu phép chia này mà dữ liệu trong thanh ghi B = 00h thì nội dung thanh ghi A không thay đổi, nội dung chứa trong thanh ghi B không xác định và bit OV = 1, bit Cy = 0.

24. Lệnh điều chỉnh thập phân nội dung thanh ghi A :

- Cú pháp : **DA A**
- Mã lệnh :

1	1	0	1	0	1	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 4 chu kỳ máy
- Chức năng: Nếu 4 bit thấp A3A2A1A0>9 hoặc bit AC = 1thì A3A2A1A0 + 6, kết qủa lưu trở lại trong A. Nếu 4 bit cao A7A6A5A4>9 hoặc bit Cy = 1 thì A7A6A5A4 + 6, kết quả lưu trở lại thanh ghi A. Kết quả sau cùng trong thanh ghi A là số BCD.

C. Nhóm lệnh logic :

1. Lệnh and thanh ghi A với thanh ghi :

- Cú pháp : **ANL A,Rn (and logic)**
- Mã lệnh :

0	1	0	1	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

- Chức năng: Nội dung thanh ghi A and với nội dung thanh ghi Rn, kết quả lưu trữ trong thanh ghi A.

Ví dụ :

```
Mov A  ,#10110011b
Mov R0,#11001011b
Anl a,r0            ;kết quả (A) = 10000011b
```

2. Lệnh and thanh ghi A với nội dung ô nhớ trực tiếp :

- Cú pháp : **ANL A, direct**
- Mã lệnh :

0	1	0	1	0	1	0	1
a7	A6	a5	a4	a3	A2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A and với nội dung của ô nhớ có địa chỉ direct, kết quả chứa ở thanh ghi A.

3. Lệnh and thanh ghi A với nội dung ô nhớ gián tiếp :

- Cú pháp : **ANL A, @Ri**
- Mã lệnh :

0	1	0	1	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A and với ô nhớ có địa chỉ chứa trong thanh ghi Ri, kết quả lưu trữ trong thanh ghi A.

4. Lệnh and thanh ghi A với dữ liệu tức thời 8 bit :

- Cú pháp : **ANL A, #data**
- Mã lệnh :

0	1	0	1	0	1	0	0
d7	d6	d5	d4	d3	D2	d1	d0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung của thanh ghi A and với dữ liệu d0 đến d7 , kết quả lưu trữ trong thanh ghi A.

Ví dụ :

```
Mov A  ,#10110011b
Anl a,#00001111b          ;kết quả (A) = 00000011b
```

5. Lệnh and nội dung ô nhớ trực tiếp với nội dung thanh ghi A :

- Cú pháp : **ANL direct, A**
- Mã lệnh :

0	1	0	1	0	0	1	0

a7	a6	A5	a4	a3	A2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung ô nhớ có địa chỉ direct and với nội dung của thanh ghi A, kết qủa lưu trữ vào ô nhớ.

Ví dụ :

```
Mov A  ,#10110011b
Mov 10h,#11110000b
Anl 10h,a          ;kết quả ô nhớ có địa chỉ 10h lưu 10110000b.
```

6. Lệnh and nội dung ô nhớ trực tiếp với dữ liệu tức thời 8 bit :

- Cú pháp : **ANL direct, #data**
- Mã lệnh :

0	1	0	1	0	0	1	1
a7	a6	a5	a4	a3	A2	a1	a0
d7	d6	d5	d4	d3	d2	d1	d0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Nội dung của ô nhớ có địa chỉ direct and với 8 bit dữ liệu 8 bit, kết quả lưu trữ vào ô nhớ.

7. Lệnh or thanh ghi A với thanh ghi :

- Cú pháp : **ORL A, Rn**
- Mã lệnh :

0	1	0	0	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A or với nội dung thanh ghi Rn, kết quả lưu trữ trong thanh ghi A.

Ví dụ :

```
Mov A  ,#10110011b
Mov R0,#11001011b
Orl a,r0           ;kết quả (A) = 11111011b.
```

8. Lệnh or thanh ghi A với nội dung ô nhớ trực tiếp :

- Cú pháp : **ORL A, direct**
- Mã lệnh :

0	1	0	0	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A or với nội dung của ô nhớ có địa chỉ direct, kết qủa chứa ở thanh ghi A.

9. Lệnh or thanh ghi A với nội dung ô nhớ gián tiếp :

- Cú pháp : **ORL A, @Ri**
- Mã lệnh :

0	1	0	0	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A or với ô nhớ có địa chỉ chứa trong thanh ghi Ri, kết quả lưu trữ trong thanh ghi A.

10. Lệnh or thanh ghi A với dữ liệu tức thời 8 bit :

- Cú pháp : **ORL A, #data**
- Mã lệnh :

0	1	0	0	0	1	0	0
d7	d6	d5	d4	d3	d2	d1	d0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung của thanh ghi A or với dữ liệu 8 bit data (từ d0 đến d7), kết quả lưu trữ trong thanh ghi A.

11. Lệnh or nội dung ô nhớ trực tiếp với nội dung thanh ghi A :

- Cú pháp : **ORL direct, A**
- Mã lệnh :

0	1	0	0	0	0	1	0
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung ô nhớ có địa chỉ direct or với nội dung của thanh ghi A, kết qủa lưu trữ trong ô nhớ có địa chỉ direct.

12. Lệnh or nội dung ô nhớ trực tiếp với dữ liệu tức thời 8 bit :

- Cú pháp : **ORL direct, #data**
- Mã lệnh :

0	1	0	0	0	0	1	1
a7	a6	a5	a4	a3	a2	a1	a0
d7	d6	d5	d4	d3	d2	d1	d0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Nội dung của ô nhớ có địa chỉ direct or với dữ liệu 8 bit (từ d0 đến d7) ở byte thứ 3, kết quả lưu trữ trong ô nhớ.

13. Lệnh ex-or thanh ghi A với thanh ghi :

- Cú pháp : **XRL A, Rn**
- Mã lệnh :

0	1	1	0	1	n2	n1	n0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy

- Chức năng: Nội dung thanh ghi A ex-or với nội dung thanh ghi Rn, kết quả lưu trữ trong thanh ghi A.

14. Lệnh ex-or thanh ghi A với nội dung ô nhớ trực tiếp :

- Cú pháp : **XRL A, direct**
- Mã lệnh :

0	1	1	0	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A ex-or với nội dung của ô nhớ có địa chỉ direct, kết qủa chứa ở thanh ghi A.

15. Lệnh ex-or thanh ghi A với nội dung ô nhớ gián tiếp :

- Cú pháp : **XRL A, @Ri**
- Mã lệnh :

0	1	1	0	0	1	1	i

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A ex-or với ô nhớ có địa chỉ chứa trong thanh ghi Ri, kết quả lưu trữ trong thanh ghi A.

16. Lệnh ex-or thanh ghi A với dữ liệu tức thời 8 bit :

- Cú pháp : **XRL A, #data**
- Mã lệnh :

0	1	1	0	0	1	0	0
d7	d6	d5	d4	d3	d2	d1	d0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung của thanh ghi A ex-or với dữ liệu datat, kết qủa lưu trữ trong thanh ghi A.

17. Lệnh ex-or nội dung ô nhớ trực tiếp với nội dung thanh ghi A :

- Cú pháp : **XRL direct, A**
- Mã lệnh :

0	1	1	0	0	0	1	0
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung ô nhớ có địa chỉ direct ex-or với nội dung của thanh ghi A, kết qủa lưu trữ vào ô nhớ.

18. Lệnh ex-or nội dung ô nhớ trực tiếp với dữ liệu tức thời 8 bit :

- Cú pháp : **XRL direct, #data**
- Mã lệnh :

0	1	1	0	0	0	1	1
a7	a6	a5	a4	a3	a2	a1	a0

d7	d6	d5	d4	d3	d2	d1	d0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Nội dung của ô nhớ có địa chỉ direct ex-or với 8 bit dữ liệu data 8 bit, kết quả lưu trữ vào ô nhớ.

19. Lệnh xóa nội dung thanh ghi A :

- Cú pháp : **CLR A** **(clear a)**
- Mã lệnh :

1	1	1	0	0	1	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A bằng zero.

20. Lệnh bù nội dung thanh ghi A :

- Cú pháp : **CPL A** **(complement A)**
- Mã lệnh :

1	1	1	1	0	1	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A được lấy bù 1, kết quả chứa trong A.

Ví dụ :

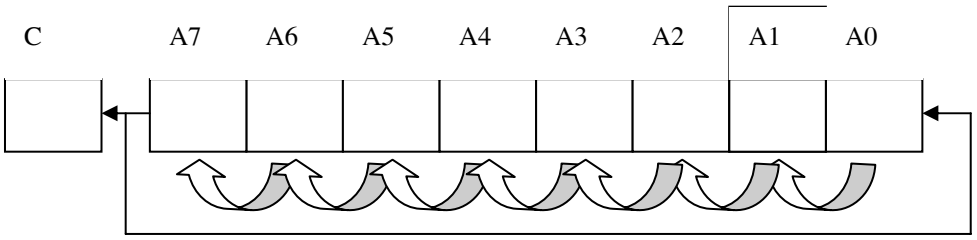
```
Mov A ,#10110011b
Cpl  a              ;kết quả (A) = 01001100b.
```

21. Lệnh xoay trái nội dung thanh ghi A :

- Cú pháp : **RL A** **(rotate left)**
- Mã lệnh :

0	0	1	0	0	0	1	1

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A được xoay trái 1 bit minh họa như hình vẽ.



Ví dụ :

```
Mov A,#1011 0011b      ;
RL  A                  ;lệnh thứ nhất
```

Gía trị ban đầu của C ta không cần quan tâm đến kết quả sau khi xoay thì (A) = 0110 0111b và cờ (C) = 1 là do bit A7 bằng 1 chuyển sang.

```
RL A            ;lệnh thứ 2
```

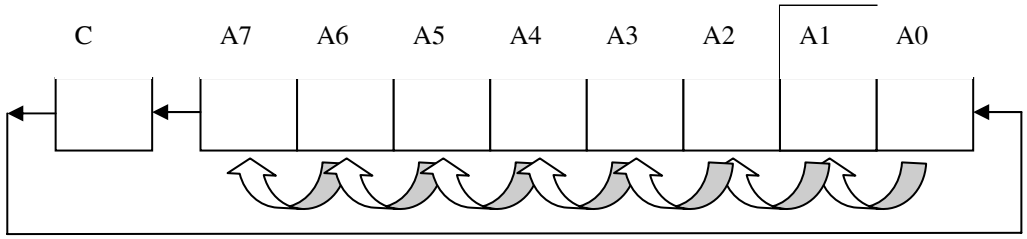
Kết quả sau khi xoay thì (A) = 11001110b và cờ (C) = 0 là do bit A7 bằng 0 chuyển sang.

22. Lệnh xoay trái nội dung thanh ghi A và bit carry :

- Cú pháp : **RLC A**
- Mã lệnh :

0	0	1	1	0	0	1	1

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A và bit C được xoay trái 1 bit.



Ví dụ 1: Giả sử cho cờ C = 0 trước khi thực hiện lệnh

```
Mov A ,#10110011b
RLC A           ;kết quả (A) = 01100110b và cờ (C) = 1
```

Ví dụ 2:

```
Setb c          ;làm cờ C bằng 1
Mov A,#00000000b
RLC A           ;kết quả (A) = 0000 0001b và cờ (C) = 0
Setb c          ;làm cờ C bằng 1
RLC A           ;kết quả (A) = 0000 0011b và cờ (C) = 0
                ....
Setb c          ;làm cờ C bằng 1
RLC A           ;kết quả (A) = 0111 1111b và cờ (C) = 0
Setb c              ;làm cờ C bằng 1 (lần thứ 8)
RLC A           ;kết quả (A) = 1111 1111b và cờ (C) = 0

Setb c              ;làm cờ C bằng 1 (lần thứ 9)
RLC A           ;kết quả (A) = 1111 1111b và cờ (C) = 1
;xxxx
clr c               ;làm cờ C bằng 0
```

```
RLC  A          ;kết quả (A) = 1111 1110b và cờ (C) = 1
```

23. Lệnh xoay phải nội dung thanh ghi A :

- Cú pháp : **RR A** **(rotate right)**
- Mã lệnh :

0	0	0	0	0	0	1	1

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A được xoay phải 1 bit ngược với lệnh RL A.

24. Lệnh xoay phải nội dung thanh ghi A và bit carry :

- Cú pháp : **RRC A**
- Mã lệnh :

1	1	1	0	0	1	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung thanh ghi A và bit C được xoay phải 1 bit ngược với lệnh RLC A.

25. Lệnh xoay thanh ghi A 4 bit :

- Cú pháp : **SWAP A**
- Mã lệnh :

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: hoán chuyển 4 bit thấp và 4 bit cao trong thanh ghi A.

Ví dụ :

```
Mov A,#3EH
Swap  A          ;kết quả (A) = E3H
```

D. Nhóm lệnh chuyển quyền điều khiển :

Nhóm lệnh này là nhóm lệnh chuyển quyền điều khiển có nghĩa là vi điều khiển đang thực hiện lệnh tại địa chỉ này thì có thể nhảy đến hoặc chuyển đến thực hiện lệnh tại một địa chỉ khác.

Trong nhóm này gồm có lệnh gọi chương trình con, lệnh kết thúc chương trình con trở về chương trình chính, lệnh nhảy không điều kiện và lệnh nhảy có điều kiện.

Các lệnh nhảy bao gồm lệnh nhảy tương đối, lệnh nhảy tuyệt đối, lệnh nhảy dài.

Các lệnh nhảy có điều kiện thì khi thỏa điều kiện thì lệnh sẽ nhảy còn nếu không thỏa điều kiện thì sẽ thực hiện lệnh kế ngay sau lệnh nhảy. Ở đây chỉ trình bày điều kiện thỏa còn điều kiện không thỏa thì ta hiểu ngầm.

1. Lệnh gọi chương trình con dùng địa chỉ tuyệt đối :

- Cú pháp : **ACALL addr11**
- Mã lệnh :

a10	a9	a8	1	0	0	0	1
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy.
- Chức năng: Khi lệnh này được thực hiện thì vi điều khiển sẽ thực hiện chương trình con tại địa chỉ addr11. Chương trình con không được cách lệnh gọi quá 2 kbyte. Addr11 của chương trình con có thể thay bằng nhản (tên của chương trình con).
- Chú ý: Trước khi nạp địa chỉ mới vào thanh ghi PC thì địa chỉ của lệnh kế trong chương trình chính được cất vào bộ nhớ ngăn xếp.

2. Lệnh gọi chương trình con dùng địa chỉ dài 16 bit :

- Cú pháp : **LCALL addr16**
- Mã lệnh :

0	0	0	1	0	0	1	0
A15	a14	a13	a12	a11	a10	a9	a8
A7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Khi lệnh này được thực hiện thì vi điều khiển sẽ thực hiện chương trình con tại địa chỉ addr16. Lệnh này có thể gọi chương trình con ở đâu cũng được trong vùng 64kbyte. Addr16 của chương trình con có thể thay bằng nhản (tên của chương trình con).
- 16 bit địa chỉ A15 – A0 được nạp vào PC, vi điều khiển sẽ thực hiện chương trình con tại địa chỉ vừa nạp vào PC. Chú ý: Trước khi nạp địa chỉ vào thanh ghi PC thì địa chỉ của lệnh kế trong chương trình chính được cất vào bộ nhớ ngăn xếp.

3. Lệnh trở về từ chương trình con :

- Cú pháp : **RET**
- Mã lệnh :

0	0	1	0	0	0	1	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Lệnh này sẽ kết thúc chương trình con, vi điều khiển sẽ trở lại chương trình chính để tiếp tục thực hiện chương trình.
- Chú ý: lệnh này sẽ lấy địa chỉ của lệnh kế đã lưu trong bộ nhớ ngăn xếp (khi thực hiện lệnh gọi) trả lại cho thanh ghi PC để tiếp tục thực hiện chương trình chính. ***Khi viết chương trình con thì phải luôn luôn kết thúc bằng lệnh ret.***

4. Lệnh trở về từ chương trình con phục vụ ngắt :

- Cú pháp : **RETI**
- Mã lệnh :

0	0	1	1	0	0	1	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

- Chức năng: Lệnh này sẽ kết thúc chương trình phục vụ ngắt, vi điều khiển sẽ trở lại chương trình chính để tiếp tục thực hiện chương trình.

5. Lệnh nhảy dùng địa chỉ tuyệt đối :

- Cú pháp : **AJMP addr11**
- Mã lệnh :

a10	a9	a8	0	0	0	0	1
a7	a6	a5	a4	A3	a2	a1	a0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Y nghĩa của lệnh: vi điều khiển sẽ nhảy đến đến địa chỉ addr11 để thực hiện chương trình tại đó. Addr11 có thể thay thế bằng nhản. Nhản hay địa chỉ nhảy đến không quá 2 kbyte.
- 11 bit địa chỉ A10 – A0 được nạp vào PC, các bit cao của PC không thay đổi, vi điều khiển sẽ nhảy đến thực hiện lệnh tại địa chỉ PC mới vừa nạp.
- Lệnh này khác với lệnh gọi chương trình con là không cất địa chỉ trở về. Nơi nhảy đến không quá 2 kbyte so với lệnh nhảy.

6. Lệnh nhảy dùng địa chỉ 16 bit :

- Cú pháp : **LJMP addr16**
- Mã lệnh :

0	0	0	0	0	0	1	0
a15	a14	a13	a12	a11	a10	a9	a8
a7	a6	a5	a4	a3	a2	a1	a0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: vi điều khiển sẽ nhảy đến địa chỉ addr16 để thực hiện chương trình tại đó. Nơi nhảy đến tùy ý nằm trong vùng 64 kbyte.

7. Lệnh nhảy tương đối :

- Cú pháp : **SJMP rel**
- Mã lệnh :

1	0	0	0	0	0	0	0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: vi điều khiển sẽ nhảy đến lệnh có địa chỉ tương đối (rel) để thực hiện tiếp. Có thể thay thế rel bằng nhản.
- Lệnh này chỉ nhảy trong tầm vực 256 byte: có thể nhảy tới 128 byte và có thể nhảy lùi 128 byte. Khi tầm vực nhảy xa hơn ta nên dùng lệnh AJMP hay LJMP.
- Chú ý: rel [relative: tương đối]: các lệnh có xuất hiện "rel" đều liên quan đến lệnh nhảy: nơi nhảy đến được tính bằng cách lấy nội dung của PC cộng với số lượng byte của các lệnh nằm giữa lệnh nhảy và nơi nhảy đến. Chúng ta không cần quan tâm đến điều này vì chương trình biên dịch của máy tính sẽ tính giúp chúng ta.

8. Lệnh nhảy gián tiếp :

- Cú pháp : **JMP @A + DPTR**
- Mã lệnh :

0	1	1	1	0	0	1	1

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: lệnh sẽ nhảy đến nơi có địa chỉ bằng nội dung của A cộng với dptr để tiếp tục thực hiện chương trình tại đó.

9. Lệnh nhảy nếu cờ Z =1 (nội dung thanh ghi A bằng 0) :

- Cú pháp : **JZ rel** **(jump zero)**
- Mã lệnh :

0	1	1	0	0	0	0	0
r7	r6	r5	r4	r3	r2	R1	r0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: nếu bit Z = 1 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel (thỏa điều kiện), nếu Z = 0 thì vi điều khiển sẽ tiếp tục thực hiện lệnh kế (không thỏa điều kiện).

10. Lệnh nhảy nếu cờ Z = 0 (nội dung thanh ghi A khác 0):

- Cú pháp : **JNZ rel**
- Mã lệnh :

0	1	1	1	0	0	0	0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: nếu Z = 0 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel.

11. Lệnh nhảy nếu bit carry = 1 :

- Cú pháp : **JC rel**
- Mã lệnh :

0	1	0	0	0	0	0	0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: nếu bit carry C = 1 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel.

12. Lệnh nhảy nếu bit carry = 0 :

- Cú pháp : **JNC rel**
- Mã lệnh :

0	1	0	1	0	0	0	0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy

- Chức năng: nếu bit carry C = 0 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel.

13. Lệnh nhảy nếu bit = 1 :

- Cú pháp : **JB bit, rel**
- Mã lệnh :

0	0	1	0	0	0	0	0
b7	b6	b5	b4	b3	b2	b1	b0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: nếu nội dung của bit có địa chỉ bit [được xác định bởi byte thứ 2] bằng 1 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel.

14. Lệnh nhảy nếu bit = 0 :

- Cú pháp : **JNB bit, rel**
- Mã lệnh :

0	0	1	1	0	0	0	0
b7	b6	b5	b4	b3	b2	b1	b0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: nếu nội dung của bit có địa chỉ bit [được xác định bởi byte thứ 2] bằng 0 thì vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ bằng rel.

15. Lệnh nhảy nếu bit = 1 và xóa bit :

- Cú pháp : **JBC bit, rel**
- Mã lệnh :

0	0	0	1	0	0	0	0
b7	b6	b5	b4	b3	b2	b1	b0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: nếu bit được xác định bởi byte thứ 2 bằng 1 thì bit này được xóa về 0 và vi điều khiển sẽ nhảy đến thực hiện chương trình tại địa chỉ rel.

16. Lệnh so sánh ô nhớ trực tiếp với nội dung thanh ghi A :

- Cú pháp : **CJNE A, direct, rel (compare jump if not equal)**
- Mã lệnh :

1	0	1	1	0	1	0	0
a7	a6	a5	a4	a3	a2	a1	a0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: lệnh này ảnh hưởng đến cờ C và thực hiện việc nhảy như sau:
  - ✓ Nếu nội dung của A ≥ nội dung của ô nhớ có địa chỉ direct thì bit C = 0.
  - ✓ Nếu nội dung của A < nội dung của ô nhớ có địa chỉ direct thì bit C = 1.

- ✓ Nếu nội dung của A khác nội dung của ô nhớ có địa chỉ direct thì lệnh sẽ nhảy đến và thực hiện lệnh tại địa chỉ rel.
- ✓ Nếu nội dung của A bằng nội dung của ô nhớ có địa chỉ direct thì không nhảy và làm lệnh kế.

17. Lệnh so sánh dữ liệu tức thời với nội dung thanh ghi A :

- Cú pháp : **CJNE A, #data, rel**
- Mã lệnh :

1	0	1	1	0	1	0	0
d7	d6	d5	d4	d3	d2	d1	d0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: lệnh này ảnh hưởng đến cờ C và thực hiện việc nhảy như sau:
  - ✓ Nếu nội dung của A ≥ data 8 bit thì bit C = 0.
  - ✓ Nếu nội dung của A < data 8 bit thì bit C = 1.
  - ✓ Nếu nội dung của A khác data 8 bit thì lệnh sẽ nhảy đến thực hiện lệnh tại địa chỉ rel.
  - ✓ Nếu nội dung của A bằng data 8 bit thì không nhảy và làm lệnh kế.

18. Lệnh so sánh dữ liệu tức thời với nội dung thanh ghi :

- Cú pháp : **CJNE Rn, #data, rel**
- Mã lệnh :

1	0	1	1	1	n2	n1	n0
d7	d6	d5	d4	d3	d2	d1	d0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: lệnh này ảnh hưởng đến cờ C và thực hiện việc nhảy như sau:
  - ✓ Nếu nội dung của thanh ghi Rn ≥ data 8 bit thì bit C = 0.
  - ✓ Nếu nội dung của thanh ghi Rn < data 8 bit thì bit C = 1.
  - ✓ Nếu nội dung của thanh ghi Rn khác data 8 bit thì lệnh sẽ nhảy đến thực hiện lệnh tại địa chỉ rel.
  - ✓ Nếu nội dung của thanh ghi Rn bằng data 8 bit thì không nhảy và làm lệnh kế.

19. Lệnh so sánh dữ liệu tức thời với dữ liệu gián tiếp :

- Cú pháp : **CJNE @Ri, #data, rel**
- Mã lệnh :

1	0	1	1	0	1	1	i
d7	d6	d5	d4	d3	d2	d1	d0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: lệnh này ảnh hưởng đến cờ C và thực hiện việc nhảy như sau:

- ✓ Nếu nội dung của ô nhớ có địa chỉ lưu trong thanh ghi Ri ≥ data 8 bit thì bit C = 0.
- ✓ Nếu nội dung của ô nhớ có địa chỉ lưu trong thanh ghi Ri < data 8 bit thì bit C = 1.
- ✓ Nếu nội dung của ô nhớ có địa chỉ lưu trong thanh ghi Ri khác data 8 bit thì lệnh sẽ nhảy đến thực hiện lệnh tại địa chỉ rel.
- ✓ Nếu nội dung của ô nhớ có địa chỉ lưu trong thanh ghi Ri bằng data 8 bit thì không nhảy và làm lệnh kế.

20. Lệnh giảm thanh ghi và nhảy :

- Cú pháp : **DJNZ Rn, rel (decrement and jump if not zero)**
- Mã lệnh :

1	1	0	1	1	n2	n1	n0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Nội dung của thanh ghi Rn giảm đi 1 và nếu kết quả trong thanh ghi Rn sau khi giảm khác 0 thì vi điều khiển sẽ thực hiện chương trình tại địa chỉ rel, nếu kết quả bằng 0 thì vi điều khiển sẽ tiếp tục thực hiện lệnh kế.

21. Lệnh giảm ô nhớ trực tiếp và nhảy :

- Cú pháp : **DJNZ direct, rel**
- Mã lệnh :

1	1	0	1	0	1	0	1
a7	a6	a5	a4	a3	a2	a1	a0
r7	r6	r5	r4	r3	r2	r1	r0

- Lệnh này chiếm 3 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Nếu nội dung của ô nhớ có địa chỉ direct giảm đi 1 và nếu kết quả sau khi giảm khác 0 thì vi điều khiển sẽ thực hiện chương trình tại địa chỉ rel, ngược lại nếu kết quả bằng 0 thì vi điều khiển sẽ tiếp tục thực hiện lệnh kế.

22. Lệnh Nop :

- Cú pháp : **NOP**
- Mã lệnh :

0	0	0	0	0	0	0	0

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Nội dung của PC tăng lên 1 và tiếp tục thực hiện lệnh tiếp theo.

E. Nhóm lệnh xử lý bit :

1. Lệnh xóa bit carry :

- Cú pháp : **CLR C**
- Mã lệnh :

1	1	0	0	0	0	1	1

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Xóa bit C về 0.

2. Lệnh xóa bit :

- Cú pháp : **CLR bit**
- Mã lệnh :

1	1	0	0	0	0	1	0
b7	b6	b5	b4	b3	b2	b1	b0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Xóa bit có địa chỉ được xác định bởi byte thứ 2 về 0.

3. Lệnh đặt bit carry :

- Cú pháp : **SETB C**
- Mã lệnh :

1	1	0	1	0	0	1	1

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Đặt bit C = 1.

4. Lệnh đặt bit :

- Cú pháp : **SETB bit**
- Mã lệnh :

1	1	0	1	0	0	1	0
b7	b6	b5	b4	B3	b2	b1	b0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Đặt bit có địa chỉ được xác định bởi byte thứ 2 lên 1.

5. Lệnh bù bit carry :

- Cú pháp : **CPL C**
- Mã lệnh :

1	0	1	1	0	0	1	1

- Lệnh này chiếm 1 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Bù bit carry, nếu trước đó C = 1 thì C = 0, ngược lại C = 0 thì C = 1.

6. Lệnh bù bit :

- Cú pháp : **CPL bit**
- Mã lệnh :

1	0	1	1	0	0	1	0
b7	b6	b5	b4	b3	b2	b1	b0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Bù bit có địa chỉ xác định bởi byte thứ 2, nếu trước đó bit này = 0 thì kết quả bit này bằng 1 và ngược lại nếu trước đó bằng 1 thì nó sẽ bằng 0.

7. Lệnh and bit carry với bit :

- Cú pháp : **ANL C, bit**
- Mã lệnh :

1	0	0	0	0	0	1	0
b7	b6	b5	b4	b3	b2	b1	b0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Bit C and với bit có địa chỉ được xác định bởi byte thứ 2, kết quả chứa ở bit C.

8. Lệnh and bit carry với bù bit :

- Cú pháp : **ANL C, /bit**
- Mã lệnh :

1	0	1	1	0	0	0	0
b7	b6	b5	b4	b3	b2	b1	b0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Bit C and với bù bit có địa chỉ được xác định bởi byte thứ 2, kết quả chứa ở bit C.

9. Lệnh or bit carry với bit :

- Cú pháp : **ORL C, bit**
- Mã lệnh :

0	1	1	1	0	0	1	0
b7	b6	b5	b4	b3	b2	b1	b0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Bit C or với bit có địa chỉ được xác định bởi byte thứ 2, kết quả chứa ở bit C.

10. Lệnh or bit carry với bù bit :

- Cú pháp : **ORL C, /bit**
- Mã lệnh :

1	0	1	0	0	0	0	0
b7	b6	b5	b4	b3	b2	b1	b0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Bit C or với bù bit có địa chỉ được xác định bởi byte thứ 2, kết quả chứa ở bit C.

11. Lệnh di chuyển bit vào bit carry :

- Cú pháp : **MOV C, bit**
- Mã lệnh :

1	0	1	0	0	0	1	0
b7	b6	b5	b4	b3	b2	b1	b0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 1 chu kỳ máy
- Chức năng: Bit có địa chỉ được xác định bởi byte thứ 2 được chuyển vào bit C.

12. Lệnh di chuyển bit carry vào bit :

- Cú pháp : **MOV bit, C**
- Mã lệnh :

1	0	0	1	0	0	1	0
b7	b6	b5	b4	b3	b2	b1	b0

- Lệnh này chiếm 2 byte và thời gian thực hiện lệnh là 2 chu kỳ máy
- Chức năng: Bit C được chuyển vào bit có địa chỉ được xác định bởi byte thứ 2.

IV. TÓM TẮT TẬP LỆNH VI ĐIỀU KHIỂN MCS51:

Data Transfer Instructions.

Mnemonic	Instruction code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Hexa decimal	Explanation
MOV A,Rn	1 1 1 0 1 n_2 n_1 n_0	E8 ÷ EF	(A) ← (Rn)
MOV A,direct	1 1 1 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	E5 Byte 2	(A) ← (direct)
MOV A,@Ri	1 1 1 0 0 1 1 I	E6 ÷ E7	(A) ← ((Ri))
MOV A, #data	0 1 1 1 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	74 Byte 2	(A) ← #data
MOV Rn, A	1 1 1 1 1 n_2 n_1 n_0	F8 ÷ FF	(Rn) ← (A)
MOV Rn,direct	1 0 1 0 1 n_2 n_1 n_0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	A8 ÷ AF Byte 2	(Rn) ← (direct)
MOV Rn #data	0 1 1 1 1 n_2 n_1 n_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	78 ÷ 7F Byte 2	(Rn) ← #data
MOV direct,A	1 1 1 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	F5 Byte 2	(direct) ← (A)
MOV direct,Rn	1 0 0 0 1 n_2 n_1 n_0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	88 ÷ 8F Byte 2	(direct) ← (Rn)
MOV direct,direct	1 0 0 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	85 Byte 2 Byte 3	(direct) ← (direct) (source) (destination)
MOV direct,@Ri	1 0 0 0 0 1 1 I a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	86 ÷ 87 Byte 2	(direct) ← ((Ri))
MOV direct,#data	0 1 1 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	75 Byte 2 Byte 3	(direct) ← #data
MOV @Ri, A	1 1 1 1 0 1 1 I	F6 ÷ F7	((Ri)) ← (A)
MOV @Ri, direct	1 0 1 0 0 1 1 I a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	A6 ÷ A7 Byte 2	((Ri)) ← (direct)
MOV @Ri, #data	0 1 1 1 0 1 1 i d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	76 ÷ 77 Byte 2	((Ri)) ← (data)
MOV dptr,#data16	1 0 0 1 0 0 0 0 d_{15} d_{14} d_{13} d_{12} d_{11} d_{10} d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	90 Byte 2 Byte 3	(dptr) ← #data $_{15\text{-}0}$ (dpH) ← #data $_{15\text{-}8}$ (dpL) ← #data $_{7\text{-}0}$
MOVC A, @A + dptr	1 0 0 1 0 0 1 1	93	(A) ← ((A) + (dptr)) External Ram

MOVC A, @A + PC	1 0 0 0 0 0 1 1	83	(A) ← ((A) + (PC)) External Ram
MOVX A, @Ri	1 1 1 0 0 0 1 i	E2 ÷ E3	(A)←((Ri)) External Ram
MOVX A,@dptr	1 1 1 0 0 0 0 0	E0	(A)←((dptr)) External Ram
MOVX @Ri, A	1 1 1 1 0 0 1 i	F2 ÷ F3	((Ri)) ← (A)
MOVX @ dptr, A	1 1 1 1 0 0 0 0	F0	((dptr)) ← (A)
PUSH direct	1 1 0 0 0 0 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	C0 Byte 2	(SP) ← (SP) + 1 ((SP)) ← (direct)
POP direct	1 1 0 1 0 0 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	D0 Byte 2	(direct) ← ((SP)) (SP) ← (SP) - 1
XCH A, Rn	1 1 0 0 1 n_2 n_1 n_0	C8 ÷ CF	(A) ↔ (Rn)
XCH A, direct	1 1 0 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	C5 Byte 2	(A) ↔ (direct)
XCH A, @Ri	1 1 0 0 0 1 1 i	C6 ÷ C7	(A) ↔ ((Ri))
XCHD A, @Ri	1 1 0 1 0 1 1 i	D6 ÷ D7	(A_{3-0}) ↔ (Ri_{3-0}))

Mathematical (Arithmetic) Instructions.

Mnemonic	Instruction code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Hexa Decimal	Explanation
ADD A,Rn	0 0 1 0 1 n_2 n_1 n_0	28 ÷ 2F	(A) ← (A) + (Rn)
ADD A,direct	0 0 1 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	25 Byte 2	(A) ← (A) + (direct)
ADD A,@Ri	0 0 1 0 0 1 1 I	26 ÷ 27	(A) ← (A) + ((Ri))
ADD A,#data	0 0 1 0 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	24 Byte 2	(A) ← (A) + #data
ADDC A, Rn	0 0 1 1 1 n_2 n_1 n_0	38 ÷ 3F	(A) ← (A) + (Rn) + (C)
ADDC A, direct	0 0 1 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	35 Byte 2	(A) ← (A) + (direct) + (C)
ADDC A, @Ri	0 0 1 1 0 1 1 I	36 ÷ 37	(A) ← (A) + ((Ri)) + (C)
ADDC A, #data	0 0 1 1 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	34 Byte 2	(A) ← (A) + #data +(C)
SUBB A,Rn	1 0 0 1 1 n_2 n_1 n_0	98 ÷ 9F	(A) ← (A) - (Rn) - (C)
SUBB A,direct	1 0 0 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	95 Byte 2	(A) ← (A) - (direct) - (C)
SUBB A,@Ri	1 0 0 1 0 1 1 I	96 ÷ 97	(A) ← (A) - ((Ri)) - (C)
SUBB A,#data	1 0 0 1 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	94 Byte 2	(A) ← (A) - #data -(C)
INC A	0 0 0 0 0 1 0 0	04	(A) ← (A) + 1
INC Rn	0 0 0 0 1 n_2 n_1 n_0	08 ÷ 0F	(Rn) ← (Rn) + 1
INC direct	0 0 0 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	05 Byte 2	(direct) ← (direct) + 1
INC @Ri	0 0 0 0 0 1 1 I	06 ÷ 07	((Ri)) ← ((Ri)) + 1
INC dptr	1 0 1 0 0 0 1 1	A3	(dptr) ← (dptr) + 1
DEC A	0 0 0 1 0 1 0 0	14	(A) ← (A) - 1
DEC Rn	0 0 0 1 1 n_2 n_1 n_0	18 ÷ 1F	(Rn) ← (Rn) - 1

DEC direct	0 0 0 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	15 Byte 2	(direct) ← (direct) - 1
DEC @Ri	0 0 0 1 0 1 1 I	16 ÷ 17	((Ri)) ← ((Ri)) - 1
MUL AB	1 0 1 0 0 1 0 0	A4	$(B_{15-8}),(A_{7-0})$ ← (A)×(B)
DIV AB	1 0 0 0 0 1 0 0	84	$(A_{15-8}),(B_{7-0})$ ← (A)/(B)
DA A	1 1 0 1 0 1 0 0	D4	Content of A là BCD

Logic Instructions.

Mnemonic	Instruction code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Hexa Decimal	Explanation
ANL A,Rn	0 1 0 1 1 n_2 n_1 n_0	58 ÷ 5F	(A) ← (A) AND (Rn)
ANL A,direct	0 1 0 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	55 Byte 2	(A) ← (A) AND (direct)
ANL A,@Ri	0 1 0 1 0 1 1 I	56 ÷ 57	(A) ← (A) AND ((Ri))
ANL A,#data	0 1 0 1 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	54 Byte 2	(A) ← (A) AND #data
ANL direct, A	0 1 0 1 0 0 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	52 Byte 2	(direct)←(direct) and (A)
ANL direct, #data	0 1 0 1 0 0 1 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	53 Byte 2 Byte 3	(direct)←(direct) and #data
ORL A, Rn	0 1 0 0 1 n_2 n_1 n_0	48 ÷ 4F	(A) ← (A) OR (Rn)
ORL A, direct	0 1 0 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	45 Byte 2	(A) ← (A) OR (direct)
ORL A, @Ri	0 1 0 0 0 1 1 I	46 ÷ 47	(A) ← (A) OR ((Ri))
ORL A, #data	0 1 0 0 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	44 Byte 2	(A) ← (A) OR #data
ORL direct, A	0 1 0 0 0 0 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	42 Byte 2	(direct)←(direct) OR (A)
ORL direct, #data	0 1 0 0 0 0 1 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	43 Byte 2 Byte 3	(direct)←(direct) OR #data
XRL A, Rn	0 1 1 0 1 n_2 n_1 n_0	68 ÷ 6F	(A) ← (A) XOR (Rn)
XRL A, direct	0 1 1 0 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	65 Byte 2	(A) ← (A) XOR (direct)
XRL A, @Ri	0 1 1 0 0 1 1 I	66 ÷ 67	(A) ← (A) XOR ((Ri))
XRL A, #data	0 1 1 0 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	64 Byte 2	(A) ← (A) XOR #data
XRL direct, A	0 1 1 0 0 0 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	62 Byte 2	(direct)←(direct) XOR (A)
XRL direct, #data	0 1 1 0 0 0 1 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	63 Byte 2 Byte 3	(direct)←(direct) XOR #data
CLR A	1 1 1 0 0 1 0 0	E4	(A) ← 0
CPL A	1 1 1 1 0 1 0 0	F4	(A) ← ($\bar{A}$) lệnh not hay bù 1
RL A	0 0 1 0 0 0 1 1	23	The contents of the

			accumulator are rotated left by one bit.
RLC A	0 0 1 1 0 0 1 1	33	The contents of the accumulator and carry are rotated left by one bit.
RR A	0 0 0 0 0 0 1 1	03	The contents of the accumulator are rotated right by one bit.
RRC A	0 0 0 1 0 0 1 1	13	The contents of the accumulator and carry are rotated right by one bit.
SWAP A	1 1 0 0 0 1 0 0	C4	$(A_{3-0}) \leftrightarrow (A_{7-4})$

Control Transfer Instructions.

Mnemonic	Instruction code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Hexa decimal	Explanation
ACALL addr 11	A_{10} a_9 a_8 1 0 0 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	Byte 1 Byte 2	(PC) ← (PC) + 2 (SP) ← (SP) + 1 ((SP)) ← (PC_{7-0}) (SP) ← (SP) + 1 ((SP)) ← (PC_{15-8}) (PC) ← page address
LCALL addr 16	0 0 0 1 0 0 1 0 a_{15} a_{14} a_{13} a_{12} a_{11} a_{10} a_9 a_8 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	12 Byte 2 Byte 3	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← (PC_{7-0}) (SP) ← (SP) + 1 ((SP)) ← (PC_{15-8}) (PC) ← $addr_{15-0}$
RET	0 0 1 0 0 0 1 0	22	(PC_{15-8}) ← ((SP)) (SP) ← (SP) - 1 (PC_{7-0}) ←((SP)) (SP) ← (SP) - 1
RETI	0 0 1 1 0 0 1 0	32	(PC_{15-8}) ← ((SP)) (SP) ← (SP) - 1 (PC_{7-0}) ←((SP)) (SP) ← (SP) - 1
AJMP addr 11	a_{10} a_9 a_8 0 0 0 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	Byte 1 Byte 2	(PC) ← (PC) + 2 (PC) ← page address
LJMP addr 16	0 0 0 0 0 0 1 0 a_{15} a_{14} a_{13} a_{12} a_{11} a_{10} a_9 a_8 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	02 Byte 2 Byte 3	(PC) ← $addr_{15-0}$
SJMP rel	1 0 0 0 0 0 0 0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	80 Byte 2	(PC) ← (PC) + 2 (PC) ← (PC) + rel
JMP @A + dptr	0 1 1 1 0 0 1 1	73	(PC) ← (A) + (dptr)
JZ rel	0 1 1 0 0 0 0 0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	60 Byte 2	(PC) ← (PC) + 2 IF (A) = 0 then (PC)←(PC)

			+ rel
JNZ rel	0 1 1 1 0 0 0 0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	70 Byte 2	(PC) ← (PC) + 2 IF (A) ≠ 0 then (PC)←(PC) + rel
JC rel	0 1 0 0 0 0 0 0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	40 Byte 2	(PC) ← (PC) + 2 IF (C) = 1 then (PC)←(PC) + rel
JNC rel	0 1 0 1 0 0 0 0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	50 Byte 2	(PC) ← (PC) + 2 IF (C) = 0 then (PC)←(PC) + rel
JB bit, rel	0 0 1 0 0 0 0 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	20 Byte 2 Byte 3	(PC) ← (PC) + 3 IF (bit) =1 then (PC)←(PC) + rel
JNB bit, rel	0 0 1 1 0 0 0 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	30 Byte 2 Byte 3	(PC) ← (PC) + 3 IF (bit) = 0 then (PC)←(PC) + rel
JBC bit, rel	0 0 0 1 0 0 0 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	10 Byte 2 Byte 3	(PC) ← (PC) + 3 IF (bit)= 1 then (bit) ← 0 (PC)←(PC) + rel

CJNE A, direct, rel	1 0 1 1 0 1 0 1
	a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0
	r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0
		B5
		Byte 2
		Byte 3
			(PC) ← (PC) + 3
			IF (direct) < (A) then (C)←0 and
			(PC)←(PC) + rel
			IF (direct) > (A) then (C)←1 and
			(PC)←(PC) + rel

CJNE A, #data, rel	1 0 1 1 0 1 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	B4 Byte 2 Byte 3	(PC) ← (PC) + 3 IF #data < (A) then (C)←0 and (PC)←(PC) + rel IF #data > (A) then (C)←1 and (PC)←(PC) + rel
CJNE Rn,#data, rel	1 0 1 1 0 n_2 n_1 n_0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	B8 ÷ BF Byte 2 Byte 3	(PC) ← (PC) + 3 IF #data < (Rn) then (C)←0 and (PC)←(PC) + rel IF #data > (Rn) then (C)←1 and (PC)←(PC) + rel
CJNE @Ri,#data, rel	1 0 1 1 0 1 1 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	B6 ÷ B7 Byte 2 Byte 3	(PC) ← (PC) + 3 IF #data <((Ri)) then (C)←0 and (PC)←(PC) + rel IF #data >((Ri)) then (C)←1 and (PC)←(PC) + rel
DJNZ Rn, rel	1 1 0 1 1 n_2 n_1 n_0	D8 ÷ DF	(PC) ← (PC) + 2

	r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	Byte 2	(Rn) ← (Rn) – 1 IF ((Rn)) ≠ 0 then (PC)←(PC) + rel
DJNZ direct, rel	1 1 0 1 0 1 0 1 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0	D5 Byte 2 Byte 3	(PC) ← (PC) + 3 (direct) ← (direct) – 1 IF (direct) ≠ 0 then (PC)←(PC) + rel
NOP	0 0 0 0 0 0 0 0	00	(PC) ← (PC) + 1

Bit Oriented Instructions.

Mnemonic	Instruction code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Hexa decimal	Explanation
CLR C	1 1 0 0 0 0 1 1	C3	(C) ← 0
CLR bit	1 1 0 0 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	C2 Byte 2	(bit) ← 0
SETB C	1 1 0 1 0 0 1 1	D3	(C) ← 1
SETB bit	1 1 0 1 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	D2 Byte 2	(bit) ← 1
CPL C	1 0 1 1 0 0 1 1	B3	(C) ← ($\overline{C}$)
CPL bit	1 0 1 1 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	B2 Byte 2	(bit) ← ($\overline{bit}$)
ANL C,bit	1 0 0 0 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	82 Byte 2	(C) ← (C) AND (bit)
ANL C,/bit	1 0 1 1 0 0 0 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	B0 Byte 2	(C) ←(C) AND ($\overline{bit}$)
ORL C,bit	0 1 1 1 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	72 Byte 2	(C) ← (C) OR (bit)
ORL C,/bit	1 0 1 0 0 0 0 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	A0 Byte 2	(C) ←(C) OR ($\overline{bit}$)
MOV C,bit	1 0 1 0 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	A2 Byte 2	(C) ← (bit)
MOV bit,C	1 0 0 1 0 0 1 0 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	92 Byte 2	(bit) ← (C)

Chương 5

KHẢO SÁT TIMER – COUNTER CỦA VI ĐIỀU KHIỂN

I. Giới thiệu.
II. Thanh ghi chọn kiểu làm việc cho timer.
III. Thanh ghi điều khiển timer.
IV. Các kiểu hoạt động của timer và cờ tràn.
V. Các nguồn xung đếm.
VI. Điều khiển các timer: đếm, ngừng đếm.
VII. Khởi tạo và truy xuất các thanh ghi của timer/counter.
VIII. Timer /counter T2 của họ MCS52.

I. GIỚI THIỆU:

Trong vi điều khiển MCS51 có 2 timer/counter T0 và T1, còn MCS52 thì có 3 timer / counter. Các timer hay counter chỉ là một và chính là bộ đếm có chức năng đếm xung.

Nếu ta sử dụng ở chế độ timer thì thời gian định thời nhân với chu kỳ của mỗi xung sẽ tạo ra lượng thời gian cần thiết – ở chế độ timer vi điều khiển thường đếm xung nội lấy từ mạch dao động bên trong vi điều khiển có chu kỳ ổn định. Chế độ timer dùng để định thời gian chính xác để điều khiển các thiết bị theo thời gian.

Nếu chúng ta sử dụng ở chế độ counter thì ta chỉ cần quan tâm đến số lượng xung đếm được – không cần quan tâm đến chu kỳ của xung đếm. Chế độ counter thường thì đếm xung nhận từ bên ngoài đưa đến ngõ vào T0 đối với timer/counter thứ 0 và ngõ vào T1 đối với timer/counter thứ 1. Đếm xung từ bên ngoài còn gọi là đếm sự kiện. Một ứng dụng cho chế độ counter là có thể sử dụng vi điều khiển làm các mạch đếm sản phẩm.

Đến đây ta có thể xem timer hay counter là 1 và chú ý rằng tại mỗi một thời điểm ta chỉ sử dụng một trong 2 hoặc là timer hoặc là counter.

Các timer / counter của vi điều khiển sử dụng 16 flip flop nên ta gọi là timer/ counter 16 bit và số lượng xung mà timer/ counter có thể đếm được tính theo số nhị phân bắt đầu từ 0000 0000 0000 0000_2 đến 1111 1111 1111 1111_2 , nếu viết theo số thập lục phân thì bắt đầu từ 0000H đến FFFFH và nếu tính theo giá trị thập phân thì bắt đầu từ 0 đến 65535.

Khi đạt đến giá trị cực đại và nếu có thêm 1 xung nữa thì bộ đếm sẽ bị tràn, khi bị tràn thì giá trị đếm sẽ tự động về 0 (giống như mạch đếm nhị phân 4 bit khi đếm lên 1111 và nếu có 1 xung nữa thì giá trị đếm về 0000) và cờ tràn của timer/counter lên 1 để báo hiệu timer/counter đã bị tràn (trước khi đếm thì phải xoá cờ tràn).

Người lập trình sử dụng trạng thái cờ tràn lên 1 để rẽ nhánh hoạt chấm dứt thời gian cần thiết đã định để chuyển sang làm một công việc khác. Và khi cờ tràn lên 1 sẽ tạo ra ngắt cũng để rẽ nhánh chương trình để thực hiện một chương trình khác – bạn sẽ nắm rõ ở phần ứng dụng.

Các giá trị đếm được của timer/counter T0 thì lưu trong 2 thanh ghi TH0 và TL0 – mỗi thanh ghi 8 bit kết hợp lại thành 16 bit.

Tương tự, các giá trị đếm được của timer/counter T1 thì lưu trong 2 thanh ghi TH1 và TL1 – mỗi thanh ghi 8 bit kết hợp lại thành 16 bit.

Ngoài các thanh ghi lưu trữ số xung đếm vừa giới thiệu thì còn có 2 thanh ghi hổ trợ kèm theo: thanh ghi TMOD và thanh ghi TCON dùng để thiết lập nhiều chế độ hoạt động khác nhau cho timer để đáp ứng được sự đa dạng các yêu cầu ứng dụng trong thực tế.

Bảng sau đây sẽ liệt kê tên, chức năng, địa chỉ của các thanh ghi liên quan đến các timer/counter của vi điều khiển 89C51.

Tên	Chức năng	Địa chỉ	Cho phép truy xuất bit
TCON	Control	88H	YES
TMOD	Mode	89H	NO
TL0	Timer 0 low-byte	8AH	NO

TL1	Timer 1 low-byte	8BH	NO
TH0	Timer 0 high-byte	8CH	NO
TH1	Timer 1 high-byte	8DH	NO

II. THANH GHI CHỌN KIỂU LÀM VIỆC CHO TIMER/COUNTER:

Thanh ghi tmod gồm hai nhóm 4 bit: 4 bit thấp dùng để thiết lập các chế độ hoạt động cho Timer 0 và 4 bit cao thiết lập các chế độ hoạt động cho Timer 1.

Các bit của thanh ghi TMOD được tóm tắt như sau :

Bit	Tên	Timer	Chức năng
7	GATE	1	Nếu GATE = 1 thì Timer 1 chỉ làm việc khi INT1= 1.
6	C/$\overline{T}$	1	Bit lựa chọn counter hay timer:
			C/$\overline{T}$ = 1 : đếm xung từ bên ngoài đưa đến ngõ vào T1.
			C/$\overline{T}$ = 0 : định thời đếm xung nội bên trong.
5	M1	1	Bit chọn mode của Timer 1.
4	M0	1	Bit chọn mode của Timer 1.
3	GATE	0	Nếu GATE = 1 thì Timer 0 chỉ làm việc khi INT1= 1.
2	C/$\overline{T}$	0	Bit lựa chọn counter hay timer: giống như trên.
1	M1	0	Bit chọn mode của Timer 0.
0	M0	0	Bit chọn mode của Timer 0.

Hai bit M0 và M1 tạo ra 4 trạng thái tương ứng với 4 kiểu làm việc khác nhau của Timer 0 hoặc của Timer 1.

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 mode Timer0. Timer 8 bit TH0 được điều khiển bởi các bit của mode Timer1. ***Timer1*** : không được hoạt động ở mode 3.

III. THANH GHI ĐIỀU KHIỂN TIMER/COUNTER:

Thanh ghi điều khiển tcon chứa các bit trạng thái và các bit điều khiển cho Timer0 và Timer1. Hoạt động của từng bit của thanh ghi tcon được tóm tắt như sau :

Bit	Kí hiệu	Địa chỉ	Chức năng
7	TF1	8FH	Cờ tràn Timer 1: TF1 = 1 khi timer 1 bị tràn và có thể xóa bằng phần mềm hoặc khi vi điều khiển thực hiện chương trình con phục vụ ngắt timer1 thì tự động xóa luôn cờ tràn TF1.
6	TR1	8EH	Bit điều khiển Timer 1 đếm / ngừng đếm: TR1 = 1 thì timer 1 được phép đếm xung.

			TR1 = 0 thì timer 1 không được phép đếm xung (ngừng).
5	TF0	8DH	Cờ tràn Timer 0 (hoạt động tương tự TF1)
4	TR0	8CH	Bit điều khiển Timer 0 (giống TR1)
3	IE1	8BH	Cờ báo ngắt của ngắt INT1. Khi có ngắt xảy ra ở ngõ vào INT1 (cạnh xuống) thì cờ IE1 tác động lên mức 1. Khi vi điều khiển thực hiện chương trình con phục vụ ngắt INT1 thì tự động xóa luôn cờ báo ngắt IE1.
2	IT1	8AH	Bit điều khiển cho phép ngắt INT1 tác động bằng mức hay bằng cạnh. IT1 = 0 thì ngắt INT1 tác động bằng mức. IT1 = 1 thì ngắt INT1 tác động bằng cạnh xuống.
1	IE0	89H	Giống như IE1 nhưng phục vụ cho ngắt INT0
0	IT0	88H	Giống như IT1 nhưng phục vụ cho ngắt INT0

IV. CÁC KIỂU HOẠT ĐỘNG CỦA TIMER VÀ CỜ TRÀN :

MCS51 có 2 timer là timer0 và timer1. Ta dùng ký hiệu TLx và Thx để chỉ 2 thanh ghi byte thấp và byte cao của Timer0 hoặc Timer1. Như đã trình bày ở trên các timer có 4 kiểu hoạt động, phần này ta sẽ khảo sát chi tiết các kiểu hoạt động của timer.

1. MODE 0 (Mode Timer 13 bit) :

Mode 0 là mode đếm 13 bit: trong đó 8 bit cao sử dụng hết 8 bit của thanh ghi Thx, 5 bit còn lại chỉ sử dụng 5 bit trọng số thấp của thanh ghi TLx, còn 3 bit cao của TLx không dùng như hình 5-1a.

Ở mode này giá trị đếm lớn nhất là 2^{13} = 8192 tức đếm từ 0 0000 0000 0000_2 đến 1 1111 1111 1111_2 và nếu có thêm một xung nữa thì bộ đếm sẽ tràn và làm cho cờ tràn lên 1.

2. MODE 1 (Mode Timer 16 bit) :

Mode 1 là mode đếm 16 bit. Ở mode này giá trị đếm là lớn nhất là 2^{16} như hình 5-1b.

3. MODE 2 (Mode Timer tự động nạp 8 bit) :

Mode 2 là mode tự động nạp 8 bit, byte thấp TLx của Timer hoạt động như một Timer 8 bit trong khi byte cao THx của Timer dùng để lưu trữ giá trị để nạp lại cho thanh ghi TLx.

Khi bộ đếm TLx chuyển trạng thái từ FFH sang 00H: thì cờ tràn được set và giá trị lưu trong THx được nạp vào TLx. Bộ đếm TLx tiếp tục đếm từ giá trị vừa nạp từ THx lên và cho đến khi có chuyển trạng thái từ FFH sang 00H kế tiếp và cứ thế tiếp tục. Sơ đồ minh họa cho timer hoạt động ở mode 2 như hình 5-1c.

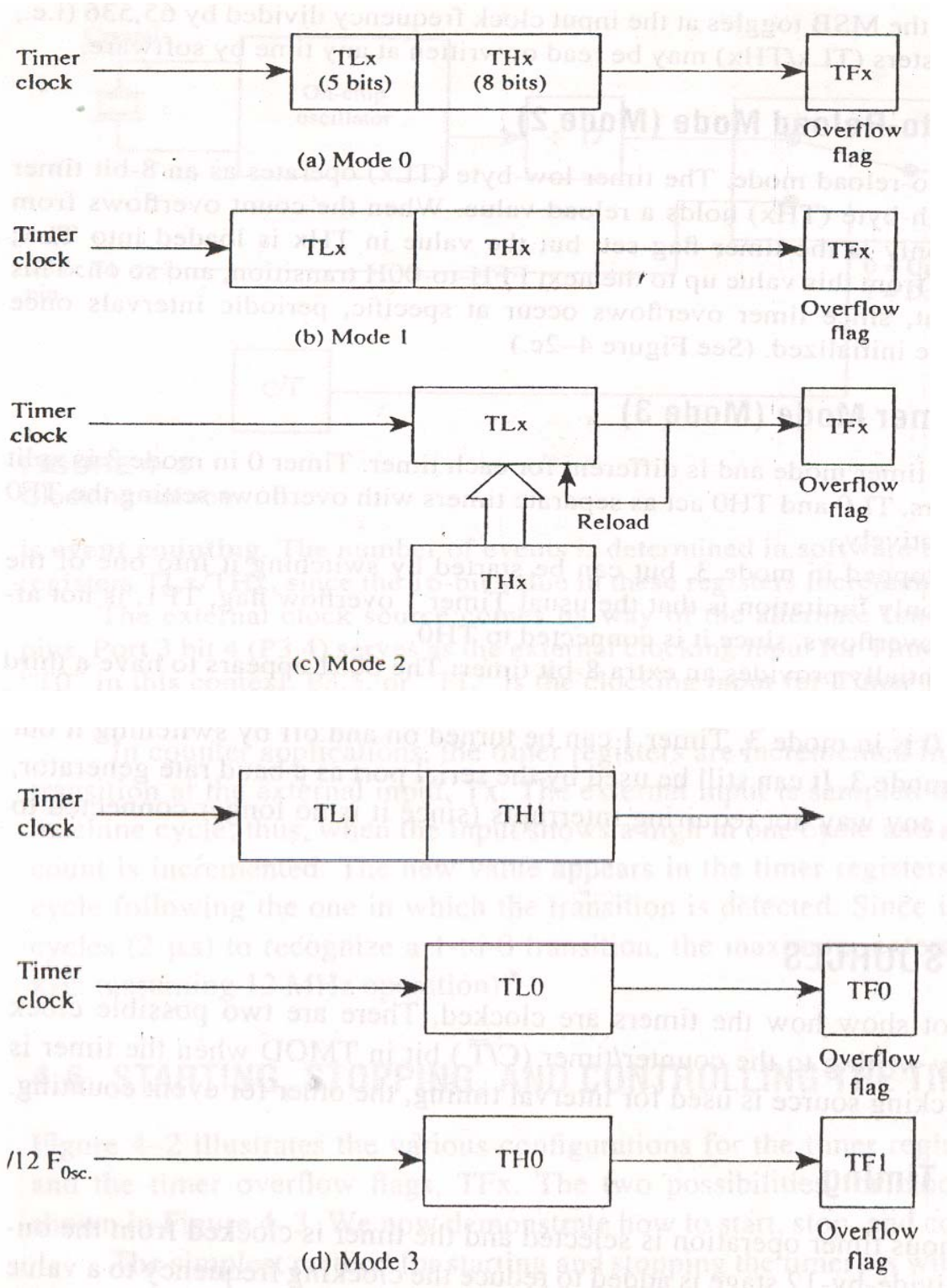
4. MODE 3 (Mode Timer tách ra) :

Mode 3 là mode Timer0 tách ra làm 2 timer cùng với timer 1 tạo thành 3 timer.

Khi Timer0 định ở cấu hình mode 3 thì timer0 được chia là 2 timer 8 bit TL0 và TH0 hoạt động như những Timer riêng lẻ và sử dụng các bit TF0 và TF1 làm các bit cờ tràn tương ứng như hình 5-1d.

Timer 1 không thể sử dụng ở mode 3, nhưng có thể được khởi động trong các mode khác và không thể báo tràn vì cờ tràn TF1 đã dùng để báo tràn cho timer TH0.

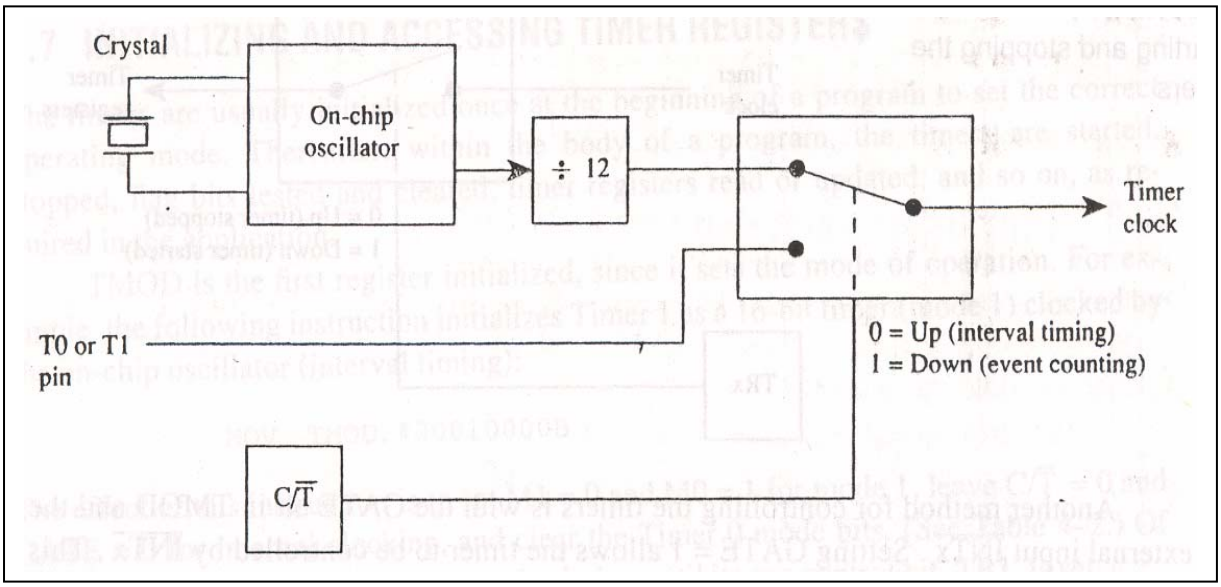
Khi timer 0 hoạt động ở Mode 3 sẽ cung cấp thêm 1 Timer 8 bit thứ ba. Khi Timer0 ở mode 3, Timer 1 có thể hoạt động như là một bộ dao động thiết lập tốc độ Baud phục vụ cho Port nối tiếp để truyền và nhận dữ liệu, hoặc nó có thể dùng trong các ứng dụng mà không sử dụng chế độ báo tràn và báo ngắt.



Hình 5-1. Các kiểu hoạt động của timer.

V. CÁC NGUỒN XUNG ĐẾM :

Timer/counter có thể đếm xung từ hai nguồn: nếu timer/counter sử dụng ở chế độ định thời timer thì sẽ đếm xung bên trong (xung nội) đã biết tần số, nếu timer/counter sử dụng ở chế độ counter thì sẽ đếm xung từ bên ngoài như hình 5-2. Bit C/$\overline{T}$ trong TMOD cho phép chọn chế độ timer hay counter khi khởi tạo ở thanh ghi tmod.



Hình 5-2. Các nguồn xung đưa đến timer / counter.

1. Đếm thời gian:

Nếu bit C/$\overline{T}$ = 0 thì Timer hoạt động đếm nội xung liên tục lấy từ dao động trên Chip. Tần số ngõ vào tụ thạch anh được đưa qua một mạch chia 12 để giảm tần số phù hợp với các ứng dụng. Nếu dùng thạch anh 12MHz thì sau khi qua bộ chia 12 tần số đưa đến bộ đếm timer là 1MHz.

Timer sẽ sinh ra tràn khi nó đã đếm đủ số xung tương ứng thời gian qui định, phụ thuộc vào giá trị khởi tạo được nạp vào các thanh ghi THx và TLx.

2. Đếm các sự kiện bên ngoài (Event Counting):

Nếu bit C/$\overline{T}$ = 1 thì Timer hoạt động đếm xung đến từ bên ngoài và chu kỳ của mỗi xung do nguồn tạo tín hiệu bên ngoài quyết định. Hoạt động này thường dùng để đếm các sự kiện. Số lượng các sự kiện được lưu trữ trong thanh ghi của các Timer.

Nguồn xung clock bên ngoài đưa vào chân P3.4 là ngõ nhập xung clock của Timer0 (T0) và P3.5 là ngõ nhập xung clock của bởi Timer1 (T1).

Trong các ứng dụng đếm xung từ bên ngoài: các thanh ghi Timer sẽ tăng giá trị đếm khi xung ngõ vào Tx chuyển trạng thái từ 1 sang 0 (tác động xung clock cạnh xuống). Ngõ vào nhận xung bên ngoài được lấy mẫu trong suốt khoảng thời gian S5P2 của mỗi chu kỳ máy, do đó khi xung ở mức H (1) trong một chu kỳ này và chuyển sang mức L (0) trong một chu kỳ kế thì bộ đếm tăng lên một. Để nhận ra sự chuyển đổi từ 1 sang 0 phải mất 2 chu kỳ máy, nên tần số xung bên ngoài lớn nhất là 500KHz nếu hệ thống vi điều khiển sử dụng dao động thạch anh 12 MHz.

VI. ĐIỀU KHIỂN CÁC TIMER: ĐẾM, NGỪNG ĐẾM :

Bit TRx trong thanh ghi TCON được điều khiển bởi phần mềm để cho phép các Timer bắt đầu quá trình đếm hoặc ngừng.

Để bắt đầu cho các Timer đếm thì phải set bit TRx bằng lệnh:

```
SETB TR0        ; cho phép timer T0 bắt đầu đếm
SETB TR1        ; cho phép timer T1 bắt đầu đếm
```

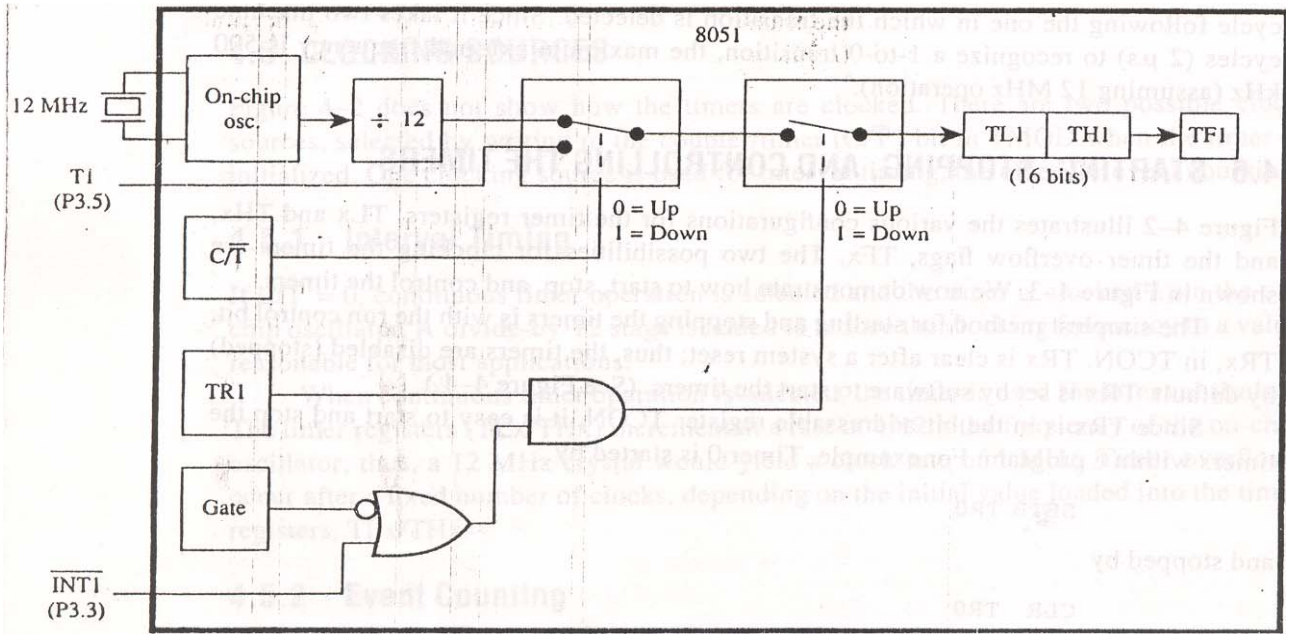
Để các Timer ngừng đếm ta dùng lệnh Clear bit TRx.

Ví dụ Timer 0 bắt đầu bởi lệnh SETB TR0 và ngừng đếm bởi lệnh CLR TR0.

Bit TRx bị xóa khi reset hệ thống, do đó ở chế độ mặc định khi mở máy các Timer bị cấm.

Một phương pháp khác để điều khiển các Timer là dùng bit GATE trong thanh ghi TMOD và ngõ nhập bên ngoài INTx như hình 5-3. Phương pháp này được dùng để đo các độ rộng xung.

Giả sử xung cần đo độ rộng đưa vào chân INT0, ta phải khởi tạo Timer 0 hoạt động ở mode 1 là mode Timer 16 bit với giá trị khởi tạo ban đầu là TL0/TH0 = 0000H, bit GATE = 1, bit TR0 = 1. Khi xung đưa đến ngõ vào INT0 = 1 thì "cổng được mở" để cho xung nội có tần số 1MHz vào tiemr 0. Quá trình timer 0 đếm xung nội sẽ dừng lại cho đến khi xung đưa đến ngõ vào INT0 xuống mức 0. Thời gian đếm được của timer 0 chính là độ rộng xung cần đo.



Hình 5-3. Đo độ rộng xung từ bên ngoài.

VII. KHỞI TẠO VÀ TRUY XUẤT CÁC THANH GHI CỦA TIMER/COUNTER :

Các Timer thường được khởi tạo 1 lần ở đầu chương trình để thiết lập mode hoạt động phục vụ cho các ứng dụng điều khiển liên quan đến định thời hay đếm xung ngoại. Tùy thuộc vào yêu cầu điều khiển cụ thể mà ta điều khiển các timer bắt đầu đếm, ngừng hay khởi động đếm lại từ đầu ...

Thanh ghi TMOD là thanh ghi đầu tiên cần phải khởi tạo để thiết lập mode hoạt động cho các Timer. Ví dụ khởi động cho Timer0 hoạt động ở mode 1 (mode Timer 16 bit) và hoạt động định thời đếm xung nội bên trong thì ta khởi tạo bằng lệnh: MOV TMOD, # 00000001B. Trong lệnh này M1 = 0, M0 = 1 để vào mode 1 và C/T = 0, GATE = 0 để cho phép đếm xung nội bên trong đồng thời xóa các bit mode của Timer 1. Sau lệnh trên Timer 0 vẫn chưa đếm và timer 0 chỉ đếm khi set bit điều khiển chạy TR0.

Nếu ta không thiết lập các giá trị bắt đầu đếm cho các thanh ghi TLx/THx thì Timer sẽ bắt đầu đếm từ 0000H lên và khi chuyển trạng thái từ FFFFH sang 0000H sẽ sinh ra tràn làm cho bit TFx = 1 rồi tiếp tục đếm từ 0000H lên tiếp . . .

Nếu ta thiết lập giá trị bắt đầu đếm cho TLx/THx khác 0000H, thì Timer sẽ bắt đầu đếm từ giá trị thiết lập đó lên nhưng khi chuyển trạng thái từ FFFFH sang 0000H thì timer lại đếm từ 0000H lên.

Để timer luôn bắt đầu đếm từ giá trị ta gán thì ta có thể lập trình chờ sau mỗi lần tràn ta sẽ xóa cờ TFx và gán lại giá trị cho TLx/THx để Timer luôn luôn bắt đầu đếm từ giá trị khởi gán lên.

Đặc biệt nếu bộ định thời hoạt động trong phạm vi nhỏ hơn 256 µs thì ta nên dùng Timer ở mode 2 (tự động nạp 8 bit). Sau khi khởi tạo giá trị đầu cho thanh ghi THx, và TLx, khi set bit TRx thì Timer sẽ bắt đầu đếm từ giá trị đã gán trong TLX và khi tràn từ FFH sang 00H trong TLx, thì cờ tràn TFx tự động được set, đồng thời giá trị trong Thx tự động nạp sang cho TLx và Timer bắt đầu đếm từ giá trị khởi gán này lên. Nói cách khác, sau mỗi lần tràn ta không cần khởi gán lại cho các thanh ghi Timer mà chúng vẫn đếm được lại từ giá trị đã gán.

Ví dụ 1: Chương trình tạo xung vuông tần số 1kHz sử dụng timer mode1:

```
                mov tmod,#01h   ;chọn mode 1 timer 0 đếm 16 bit
loop1:          mov th0,#0feh   ;độ rộng xung 500µs
                mov tl0,#0ch    ;
                setb tr0        ;cho timer bắt đầu đếm
loop:           jnb tf0,loop    ;chờ báo ngắt
                clr tf0         ;xóa cờ ngắt
                cpl p1.0        ;nghịch đảo bit p1.0
                sjmp loop1      ;quay trở lại làm tiếp
```

Ví dụ 2: Chương trình tạo xung vuông tần số 10 kHz sử dụng timer mode2:

```
                mov tmod,#02h   ;chọn mode 2 chế dộ tự động nạp lại 8 bit
loop1:          mov th0,#-50    ;tạo độ rộng xung 50µs
                setb tr0        ;cho timer bắt đầu đếm
loop:           jnb tf0,loop    ;chờ báo ngắt
                clr tf0         ;xóa cờ ngắt
                cpl p1.0        ;nghịch đảo bit p1.0
                sjmp loop1      ;tro lai loop1
```

VIII. TIMER/COUNTER T2 CỦA HỌ MCS52 :

Họ vi điều khiển MCS52 có 3 timer T0, T1, T2. Các timer T0 và T1 có các thanh ghi và hoạt động giống như họ 51. Ở đây chỉ trình bày thêm phần hoạt động của timer T2.

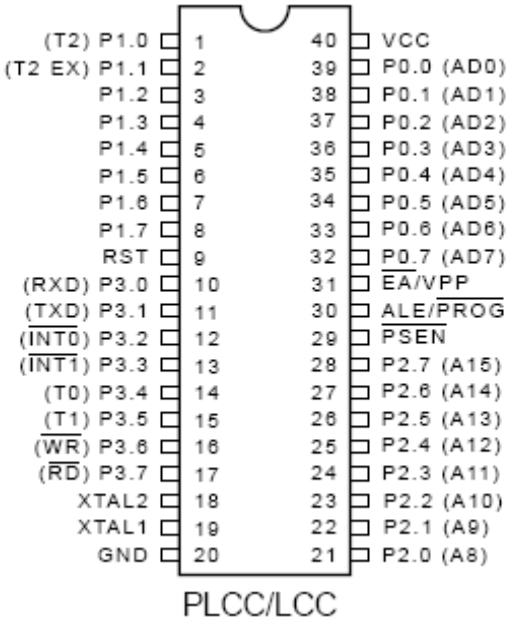
Các thanh ghi của timer/counter T2 bao gồm: thanh ghi TL2, TH2, thanh ghi điều khiển T2CON, thanh ghi RCAP2L và RCAP2H.

Timer/counter T2 có thể dùng để định thời timer hoặc dùng như bộ đếm counter để đếm xung ngoài đưa đến ngõ vào T2 chính là chân P1.0 của port 1 như hình 5-4.

Timer/counter T2 có 3 kiểu hoạt động: tự động nạp lại, thu nhận và thiết lập tốc độ baud để phục vụ cho truyền dữ liệu.

Chức năng của thanh ghi điều khiển T2CON:

Bit	Kí hiệu	Địa chỉ	Chức năng
7	TF2	CFH	Cờ tràn Timer 2: hoạt động giống như các timer trên (TF2 sẽ không được thiết lập lên mức 1 nếu bit TCLK hoặc RCLK ở mức 1).
6	EXF2	CEH	Cờ ngoài của timer T2: chỉ được set khi xảy ra sự thu nhận hoặc nạp lại dữ liệu bởi sự chuyển trạng thài từ 1 sang 0 ở ngõ vào T2EX và EXEN2 = 1; khi cho phép timer T2 ngắt, EXF2=1 thì CPU sẽ thực hiện hiện chương trình con phục vụ ngắt Timer T2, bit EXF2 có thể bằng phần mềm.
5	RCLK	CDH	Xung clock thu của timer 2. Khi RCLK=1 thì timer T2 cung cấp tốc độ baud cho port nối tiếp để nhận dữ liệu về và timer T1 sẽ cung cấp tốc độ baud cho port nối tiếp để phát dữ liệu đi.
4	TCLK	CCH	Xung clock phát của timer 2. Khi TCLK=1 thì timer T2 cung cấp tốc độ baud cho port nối tiếp để phát dữ liệu đi và timer T1 sẽ cung cấp tốc độ baud cho port nối tiếp để nhận dữ liệu về.
3	EXEN2	CBH	Bit điều khiển cho phép tác động từ bên ngoài. Khi EXEN2 = 1 thì hoạt động thu nhận và nạp lại của timer T2 chỉ xảy ra khi ngõ vào T2EX có sự chuyển trạng thái từ 1 sang 0.
2	TR2	CAH	Bit điều khiển Timer 1 đếm / ngừng đếm: TR2 = 1 thì timer 1 được phép đếm xung. TR2 = 0 thì timer 1 không được phép đếm xung (ngừng). Dùng lệnh điều khiển bit TR2 để cho phép timer1 đếm hay ngừng đếm.
1	C/$\overline{T2}$	C9H	Bit lựa chọn counter hay timer: C/$\overline{T2}$ = 1 : đếm xung từ bên ngoài đưa đến ngõ vào T2. C/$\overline{T2}$ = 0 : định thời đếm xung nội bên trong.
0	CP/$\overline{RL2}$	C8H	Cờ thu nhận/nạp lại dữ liệu của timer T2. Khi bit này = 1 thì thu nhận chỉ xảy ra khi có sự chuyển trạng thái từ 1 sang 0 ở ngõ vào T2EX và EXEN2=1; khi bit này = 0 thì quá trình tự động nạp lại khi timer T2 tràn hoặc khi có sự chuyển trạng thái ở ngõ vào T2EX và bit EXEN2 = 1; nếu bit RCLK hoặc TCLK = 1 thì bit này xem như bỏ.



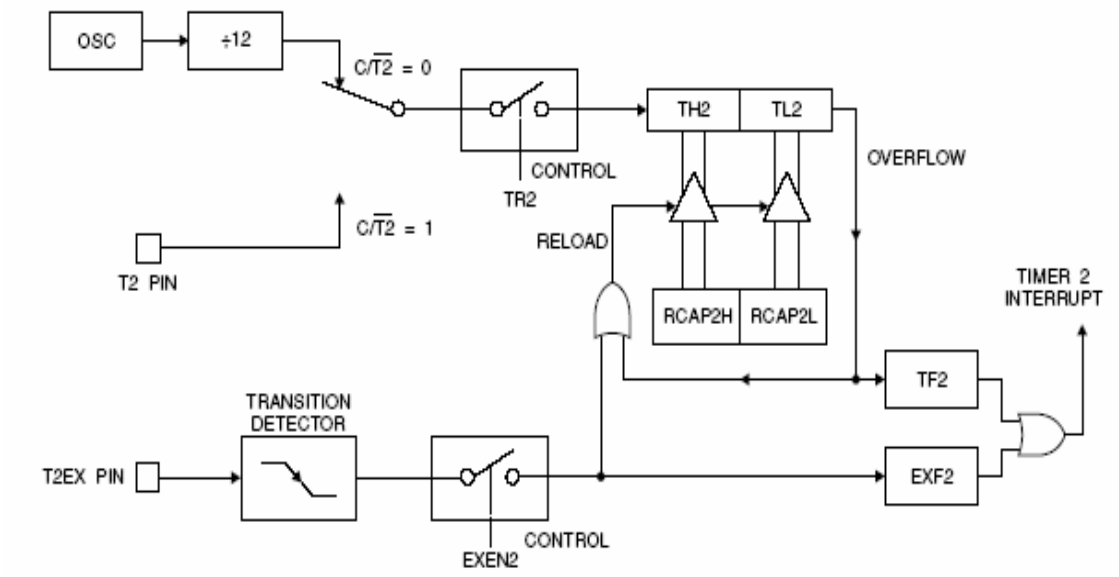
Hình 5-4. Sơ đồ chân của 89C52 với ngõ vào T2 là P1.0 và T2EX là P1.1 .

1. Chế độ tự động nạp lại:

Bit thu nhận/nạp lại $CP/\overline{RL2}$ lựa chọn một trong hai chế độ: tự động nạp lại và thu nhận. Khi $CP/\overline{RL2}=0$ thì timer hoạt động ở chế độ tự động nạp lại: các thanh ghi TL2, TH2 sẽ lưu trữ số xung đếm còn 2 thanh ghi RCAP2L và RCAP2H lưu trữ giá trị để nạp lại cho TL2, TH2. Giá trị lưu và nạp lại là 16 bit.

Khi timer đếm tràn thì làm cho cờ báo tràn TF2 bằng 1 đồng thời tự động thực hiện nạp lại dữ liệu.

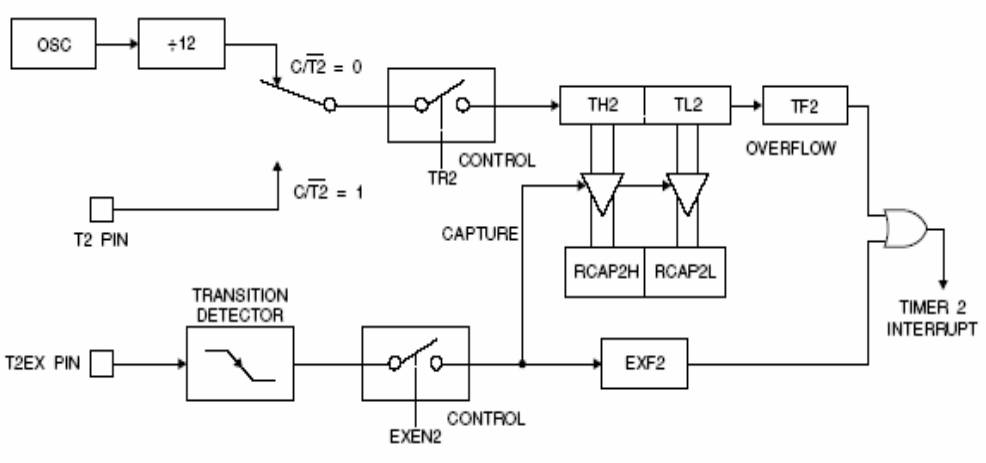
Tương tự nếu bit EXEN2 = 1 thì chế độ tự động nạp lại xảy ra khi có sự chuyển trạng thái từ 1 sang 0 ở ngõ vào T2EX đồng thời làm cho bit EXF2 = 1. Tương tự như cờ TF2 thì cờ EXF2 cũng có thể được kiểm tra bằng phần mềm hoặc tạo ngắt. Bit EXF2 phải xóa bằng phần mềm. Hoạt động tự nạp của timer T2 được trình bày như hình 5-5.



Hình 5-5.Hoạt động của timer T2 ở chế độ tự động nạp lại.

2. Chế độ thu nhận:

Khi $CP/\overline{RL2}=1$ thì timer hoạt động ở chế độ thu nhận. Khi đó timer T2 hoạt động bình thường như một timer/counter 16 bit, thanh ghi TL2, TH2 sẽ lưu trữ xung đếm và nếu có sự cuyển trạng thái từ FFFFH sang 0000H thì sẽ sinh ra tràn và làm cho cờ tràn TF2=1. Bit cờ tràn có thể kiểm tra bằng phần mềm hay có thể tạo ra ngắt.



Hình 5-6.Hoạt động của timer T2 ở chế độ thu nhận dữ liệu.

Để cho phép chế độ thu nhận hoạt động thì làm cho bit EXEN2 = 1. Nếu bit EXEN2 = 1 và khi có sự chuyển trạng trái từ 1 sang 0ở ngõ vào T2EX thì chế độ thu nhận sẽ xảy ra: giá trị đếm được trong thanh ghi TL2, TH2 sẽ được chuyển sang 2 thanh ghi RCAP2L và RCAP2H. Cờ EXF2 cũng được chuyển lên mức 1 để báo hiệu quá trình thu nhận đã xảy ra, cờ EXF2 có thể kiểm tra bằng phần mềm hoặc tạo ngắt.

Hoạt động thu nhận dữ liệu của timer T2 được trình bày ở hình 5-6.

Chương 6

KHẢO SÁT TRUYỀN DỮ LIỆU CỦA VI ĐIỀU KHIỂN

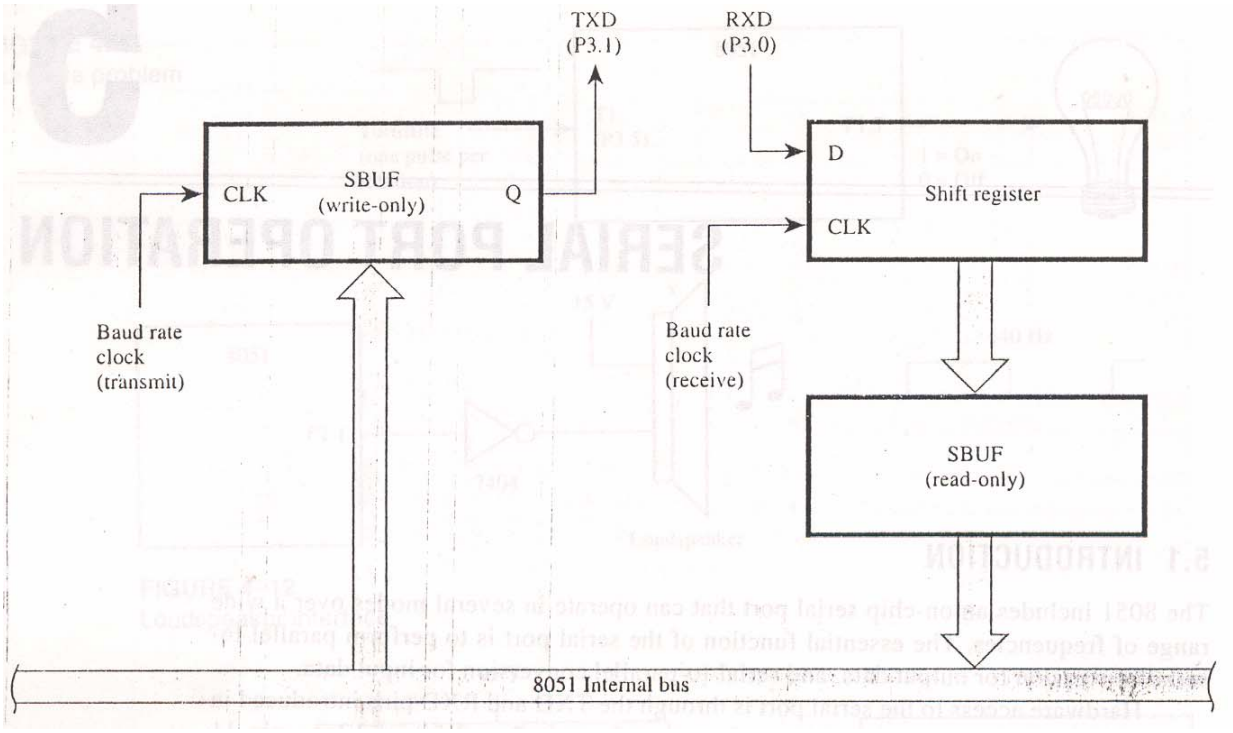
I. GIỚI THIỆU:

Truyền dữ liệu nối tiếp của MCS51 có thể hoạt động ở nhiều kiểu riêng biệt trong phạm vi cho phép của tần số. Dữ liệu dạng song song được chuyển thành nối tiếp để truyền đi và dữ liệu nhận về dạng nối nối tiếp được chuyển thành song song.

Chân TxD (P3.1) là ngõ xuất dữ liệu đi và chân RxD (P3.0) là ngõ nhận dữ liệu về.

Đặc trưng của truyền dữ liệu nối tiếp là hoạt động song công có nghĩa là có thể thực hiện truyền và nhận dữ liệu cùng một lúc.

Hai thanh ghi chức năng đặc biệt phục vụ cho truyền dữ liệu là thanh ghi đệm SBUF và thanh ghi điều khiển SCON. Thanh ghi đệm sbuf nằm ở địa chỉ 99H có 2 chức năng: nếu vi điều khiển ghi dữ liệu lên thanh ghi sbuf thì dữ liệu đó sẽ được truyền đi, nếu hệ thống khác gởi dữ liệu đến thì sẽ được lưu vào thanh ghi đệm sbuf. Sơ đồ khối của hệ thống truyền dữ liệu như hình 6-1



Hình 6-1.Sơ đồ khối của truyền dữ liệu nối tiếp.

Thanh ghi điều khiển truyền dữ liệu SCON nằm ở địa chỉ 98H là thanh ghi cho phép truy suất bit bao gồm các bit trạng thái và các bit điều khiển. Các bit điều khiển dùng để thiết lập nhiều kiểu hoạt động truyền dữ liệu khác nhau, còn các bit trạng thái cho biết thời điểm kết thúc khi truyền xong một kí tự hoặc nhận xong một kí tự. Các bit trạng thái có thể được kiểm tra trong chương trình hoặc có thể lập trình để sinh ra ngắt.

Tần số hoạt động của truyền dữ liệu nối tiếp còn gọi tốc độ BAUD (số lượng bit dữ liệu được truyền đi trong một giây) có thể hoạt động cố định (sử dụng dao động trên chip) hoặc có thể thay đổi. Khi cần tốc độ Baud thay đổi thì phải sử dụng Timer 1 để tạo tốc độ baud.

II. THANH GHI ĐIỀU KHIỂN TRUYỀN NỐI TIẾP:

Thanh ghi scon sẽ thiết lập các kiểu hoạt động truyền dữ liệu khác nhau cho MCS51. Bảng 6-1 tóm tắt thanh ghi điều khiển Port nối tiếp scon như sau :

Bit	Ký hiệu	Địa chỉ	Mô tả hoạt động
7	SM0	9FH	Bit chọn kiểu truyền nối tiếp: bit thứ 0.
6	SM1	9EH	Bit chọn kiểu truyền nối tiếp: bit thứ 1.
5	SM2	9DH	Bit cho phép truyền kết nối nhiều vi xử lý ở mode 2 và 3; RI sẽ không tích cực nếu bit thứ 9 đã thu vào là 0.
4	REN	9CH	Bit cho phép nhận kí tự. REN = 1 sẽ cho phép nhận kí tự.
3	TB8	9BH	Dùng để lưu bit 9 để tryền đi khi hoạt động ở mode 2 và 3. TB8 bằng 0 hay 1 là do người lập trình thiết lập.
2	RB8	9AH	Dùng để lưu bit 9 nhận về khi hoạt động ở mode 2 và 3.
1	TI	99H	Cờ báo hiệu này lên mức 1 khi truyền xong 1 kí tự và xóa bởi người lập trình để sẵn sàng truyền kí tự tiếp theo.
0	RI	98H	Cờ báo hiệu này lên mức 1 khi nhận xong 1 kí tự và xóa bởi người lập trình để sẵn sàng nhận kí tự dữ liệu tiếp theo.

Bảng 6-1. Các bit trong thanh ghi điều khiển truyền dữ liệu.

III. CÁC KIỂU TRUYỀN DỮ LIỆU NỐI TIẾP (MODE OF OPERATION):

Trước khi truyền dữ liệu thì thanh ghi SCON phải được khởi tạo đúng kiểu. Ví dụ để khởi tạo truyền dữ liệu kiểu 1 thì 2 bit: SM0 SM1 = 01, bit cho phép thu: REN =1, và cờ ngắt truyền TI = 1 để sẵn sàng truyền, ta dùng lệnh sau : MOV SCON, # 01010010b.

Truyền dữ liệu nối tiếp của MCS51 có 4 kiểu hoạt động tùy thuộc theo 4 trạng thái của 2 bit SM0 SM1 được liệt kê ở bảng 6-2.

Ba trong bốn kiểu cho phép truyền đồng bộ với mỗi kí tự thu hoặc phát sẽ được kết hợp với bit Start hoặc bit Stop.

SM0	SM1	Kiểu	Mô tả	Tốc độ baud
0	0	0	Thanh ghi dịch	Cố định (tần số dao động f/12).
0	1	1	UART 8 bit	Thay đổi (được đặt bởiTimer).
1	0	2	UART 9 bit	Cố định (tần số dao động f/12 or f/64)
1	1	3	UART 9 bit	Thay đổi (được đặt bởi Timer).

Bảng 6-2. Các kiểu truyền dữ liệu.

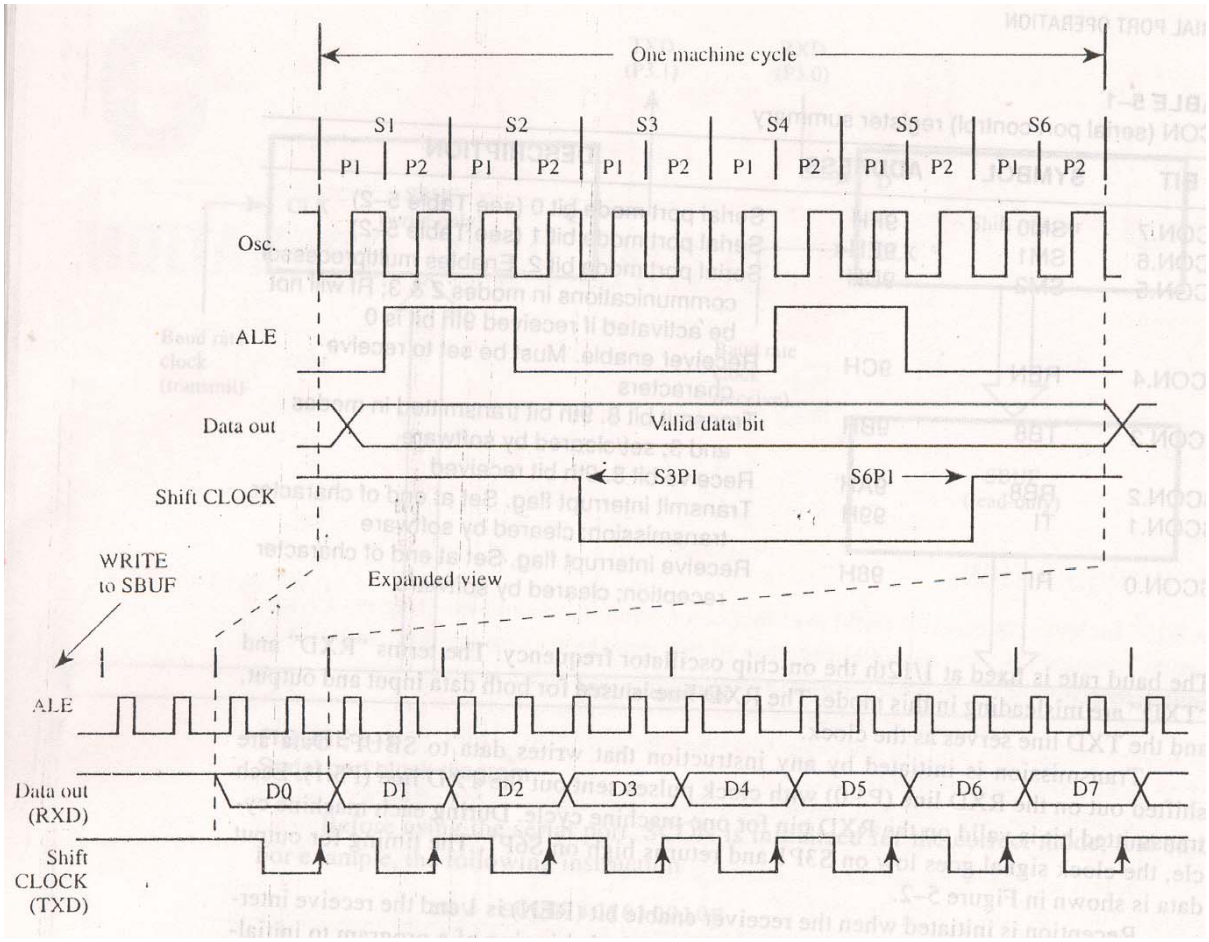
1. Truyền dữ liệu kiểu 0 – kiểu thanh ghi dịch 8 bit :

Để định cấu hình cho truyền dữ liệu nối tiếp ở kiểu 0 thì 2 bit SM1 SM0 = 00. Dữ liệu nối tiếp nhận vào và dữ liệu truyền đi đều thông qua chân RxD, còn chân TxD thì dùng để dịch chuyển xung clock. 8 bit dữ liệu để truyền đi hoặc nhận về thì luôn bắt đầu với bit có trọng số nhỏ nhất LSB. Tốc độ Baud được thiết lập cố định ở tần số bằng $1/12$ tần số dao động thạch anh trên Chip.

Khi thực hiện lệnh ghi dữ liệu lên thanh ghi sbuf thì quá trình truyền dữ liệu bắt đầu. Dữ liệu được dịch ra ngoài thông qua chân RxD cùng với các xung nhịp cũng được gởi ra ngoài thông qua chân TxD. Mỗi bit truyền đi chỉ có xuất hiện trên chân RxD trong khoảng thời gian một chu kỳ

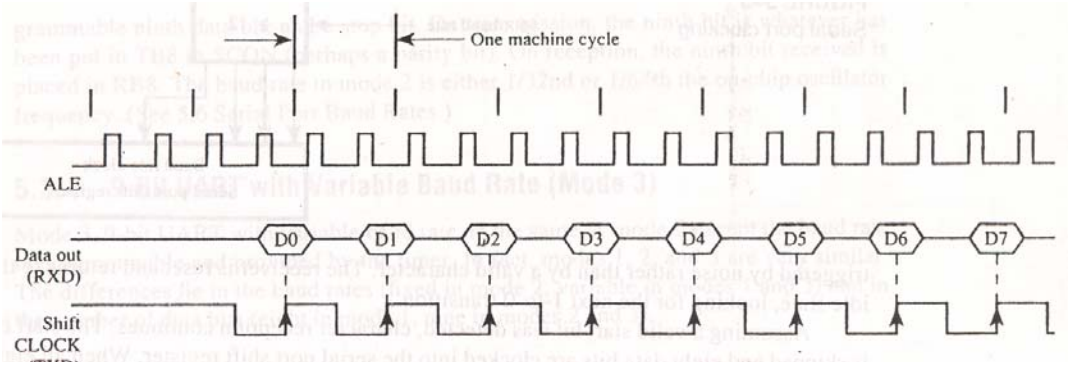
máy. Trong khoảng thời gian của mỗi chu kỳ máy, tín hiệu xung clock xuống mức thấp tại thời điểm S3P1 và lên mức cao tại thời điểm S6P1 trong giản đồ thời gian hình 6-2.

Quá trình nhận được khởi động khi bit cho phép nhận REN = 1 và cờ nhận RI = 0. Nguyên tắc chung là khởi tạo bit REN = 1 ở đầu chương trình để khởi động truyền dữ liệu, và xóa bit RI để sẵn sàng nhận dữ liệu vào. Khi bit RI bị xóa, các xung clock sẽ xuất ra bên ngoài thông qua chân TxD, bắt đầu chu kỳ máy kế tiếp thì dữ liệu từ bên ngoài sẽ được dịch vào bên trong thông qua chân RxD.



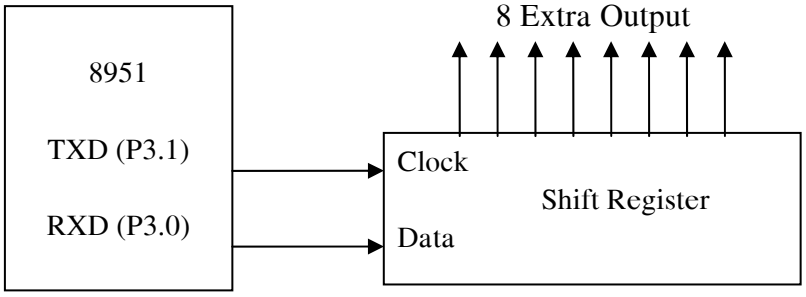
Hình 6-2.Giản đồ thời gian.

Biểu đồ thời gian của dữ liệu nối tiếp truyền vào vi điều khiển ở kiểu 0 như sau :



Hình 6-3. Giản đồ thời gian ở kiểu 0 .

Một ứng dụng cụ thể sử dụng mode 0 là dùng để mở rộng thêm số lượng ngõ ra cho MCS51 với cách thức thực hiện như sau: một thanh ghi dịch từ nối tiếp thành song song được nối đến các đường TxD và RxD của MCS51 để mở rộng thêm 8 đường ra như hình 6-4. Nếu dùng thêm nhiều thanh ghi dịch mắc nối tiếp vào thanh ghi dịch đầu tiên sẽ mở rộng được nhiều ngõ ra.



Hình 6-4. Một ứng dụng kiểu 0 để tăng thêm ngõ ra bằng thanh ghi dịch.

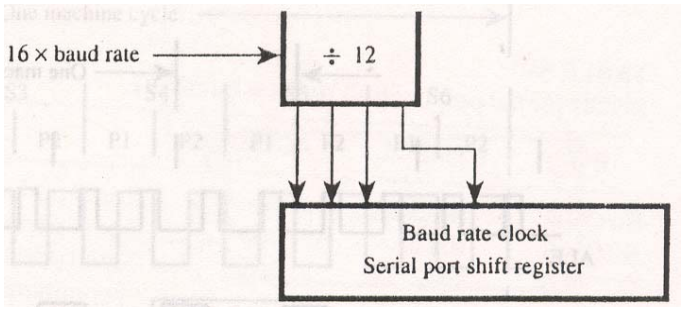
2. Truyền dữ liệu kiểu 1 – Thu phát bất đồng bộ 8 bit với tốc độ Baud thay đổi :

Trong mode này, truyền dữ liệu nối tiếp hoạt động bất đồng bộ UART 8 bit có tốc độ Baud thay đổi được. UART là bộ thu và phát dữ liệu nối tiếp với mỗi ký tự dữ liệu luôn bắt đầu bằng 1 bit Start (ở mức 0) và kết thúc bằng 1 bit Stop (ở mức 1), bit parity đôi khi được ghép vào giữa bit dữ liệu sau cùng và bit Stop.

Trong kiểu này, 10 bit dữ liệu sẽ phát đi ở chân TxD và nếu nhận thì sẽ nhận ở chân RxD. 10 bit đó bao gồm: 1 bit start, 8 bit data (LSB là bit đầu tiên), và 1 bit stop. Đối với hoạt động nhận dữ liệu thì bit Stop được đưa vào bit RB8 trong thanh ghi SCON.

Trong MCS51, tốc độ Baud được thiết lập bởi tốc độ tràn của Timer T1. Đối với họ 52 có 3 timer thì tốc độ baud có thể thiết lập bởi tốc độ tràn của timer T1 hoặc timer T2 hoặc cả 2 timer T1 và T2: một timer cho máy phát và 1 timer cho máy thu.

Nguồn cung cấp xung clock để đồng bộ các thanh ghi truyền dữ liệu nối tiếp hoạt động ở kiểu 1, 2, 3 được thiết lập bởi bộ đếm 16 như hình 6-5, ngõ ra của bộ đếm là xung clock tạo tốc độ baud. Xung ngõ vào của bộ đếm có thể lập trình bằng phần mềm.



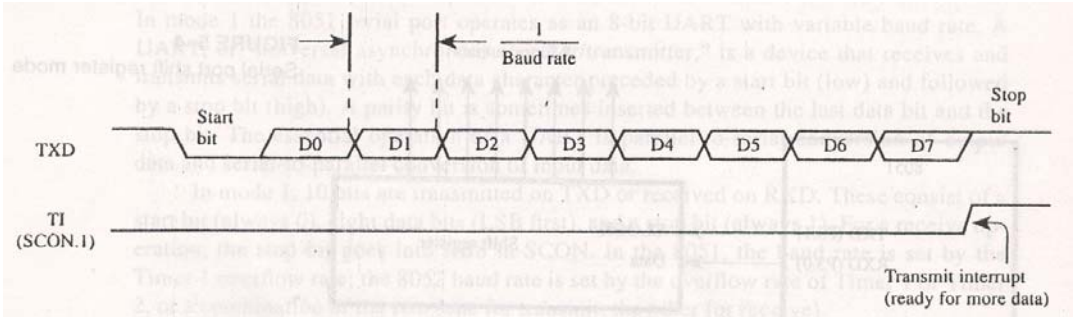
Hình 6-5.Cung cấp xung cho truyền dữ liệu nối tiếp.

Khi có một lệnh ghi dữ liệu lên thanh ghi sbuf thì quá trình truyền dữ liệu bắt đầu nhưng nó chưa truyền mà chờ cho đến khi bộ chia 16 (cung cấp tốc độ Baud cho truyền dữ liệu nối tiếp) bị tràn. Dữ liệu được xuất ra trên chân TxD bắt đầu với bit start theo sau là 8 bit data và sau cùng là

bit stop. Các cờ phát TI được nâng lên mức 1 cùng lúc với thời điểm xuất hiện bit Stop trên chân TxD như hình 6-6.

Qúa trình nhận dữ liệu được khởi động khi có sự chuyển đổi từ mức 1 sang mức 0 ở ngõ vào RxD. Bộ đếm 4 bit được reset ngay lập tức để sắp xếp bit dữ liệu đang đến từ ngõ vào RxD. Mỗi bit dữ liệu đến được lấy mẫu ở trạng thái đếm thứ 8 trong một chu kỳ 16 trạng thái của bộ đếm 4 bit.

Khi có sự chuyển trạng thái từ 1 xuống 0 ở ngõ vào RxD của bộ thu thì trạng thái 0 này phải tồn tại trong 8 trạng thái liên tục của bộ đếm 4 bit. Nếu trường hợp này không đúng thì bộ thu xem như bị tác động bởi tín hiệu nhiễu. Bộ thu sẽ reset và trở về trạng thái nghỉ và chờ sự chuyển trạng thái tiếp theo.



Hình 6-6. Cờ báo phát xong dữ liệu TI.

Giả sử việc kiểm tra bit Start là hợp lệ thì bit Start sẽ được bỏ qua và 8 bit data được nhận vào thanh ghi dịch nối tiếp.

Khi tất cả 8 bit được ghi vào thanh ghi dịch thì 3 công việc sau sẽ được thực hiện tiếp theo:

- Bit thứ 9 (bit Stop) được dịch vào bit RB8 trong SCON.
- 8 bit data được nạp vào thanh ghi SBUF.
- Cờ ngắt nhận RI =1.

Tuy nhiên, 3 công việc trên chỉ xảy ra nếu hai điều kiện sau tồn tại :

- RI = 0 và
- SM2 = 1 và bit Stop nhận được = 1 hoặc SM2 = 0.

3. Truyền dữ liệu kiểu 2 – Thu phát bất đồng bộ 9 bit có tốc độ Baud cố định :

Khi SM1 SM0 = 10 thì truyền dữ liệu hoạt động ở kiểu 2 có tốc độ Baud cố định. Có 11 bit được phát hoặc thu : 1 bit Start, 8 bit data, 1 bit data thứ 9 được lập trình và 1 bit Stop. Khi phát thì bit thứ 9 được đặt vào TB8 của SCON (có thể bit parity). Khi thu thì bit thứ 9 được đặt vào bit RB8 của thanh ghi SCON. Tốc độ Baud trong mode 2 bằng 1/12 hoặc 1/64 tần số dao động trên Chip.

4. Truyền dữ liệu kiểu 3 – Thu phát bất đồng bộ 9 bit có tốc độ Baud thay đổi:

Khi SM1 SM0 = 11 thì truyền dữ liệu hoạt động ở kiểu 3 là kiểu UART 9 bit có tốc độ Baud thay đổi. Kiểu 3 tương tự kiểu 2 ngoại trừ tốc độ Baud được lập trình và được cung cấp bởi Timer. Các kiểu 1, kiểu 2 và kiểu 3 rất giống nhau, những điểm khác nhau là ở tốc độ Baud (kiểu 2 cố định, kiểu 1 và kiểu 3 thay đổi) và số bit dữ liệu (kiểu 1 có 8 bit, kiểu 2 và kiểu 3 có 9 bit data).

IV. KHỞI TẠO VÀ TRUY XUẤT CÁC THANH GHI TRUYỀN DỮ LIỆU:

1. Bit cho phép thu (Receive Enable) :

Để cho phép thu dữ liệu thì chương trình phải làm cho bit REN = 1 và điều này được thực hiện ở đầu chương trình.

Ta có thể khởi tạo cho phép truyền dữ liệu bằng lệnh :

Setb ren hoặc *mov scon, # xxx1xxxxb*

2. Bit dữ liệu thứ 9 (the9th data bit) :

Bit dữ liệu thứ 9 được phát trong kiểu 2 và kiểu 3 phải được nạp vào bit TB8 bằng phần mềm có nghĩa là người lập trình phải thực hiện công việc này trước khi truyền dữ liệu đi, còn bit dữ liệu thứ 9 của dữ liệu thu được thì tự động đặt vào trong bit RB8.

Phần mềm có thể hoặc không đòi hỏi bit dữ liệu thứ 9 tham gia vào quá trình truyền dữ liệu tùy thuộc vào đặc tính của các thiết bị nối tiếp kết nối với nhau thiết lập ra qui định. Bit dữ liệu thứ 9 đóng 1 vai trò quan trọng trong truyền thông nhiều vi xử lý.

3. Bit kiểm tra chẵn lẻ Parity :

Bit thứ 9 thường được dùng là bit kiểm tra chẵn lẻ. Ở mỗi chu kỳ máy, bit P trong thanh ghi trạng thái PSW bằng 1 hay bằng 0 tùy thuộc vào quá trình kiểm tra chẵn 8 bit dữ liệu chứa trong thang ghi A.

Ví dụ nếu hệ thống truyền dữ liệu yêu cầu 8 bit data cộng thêm 1 bit kiểm tra chẵn, thì các lệnh sau đây sẽ phát 8 bit trong thanh ghi A cộng với bit kiểm tra chẵn được cộng vào bit thứ 9.

```
Mov     C,P             ;chuyển cờ chẵn lẻ P sang cờ C
Mov     TB8,C           ;chuyển cờ C sang bit TB8 để chuẩn bị truyền đi
Mov     sbuf,A          ;truyền dữ liệu 8 bit trong A và bit thứ 9 trong TB8 đi.
```

Nếu kiểm tra lẻ được yêu cầu thì các lệnh trên được sửa lại là :

```
Mov     C,P             ;chuyển cờ chẵn lẻ P sang cờ C
Cpl     C               ;nghịch đảo chẵn thành lẻ
Mov     TB8,C           ;chuyển cờ C sang bit TB8 để chuẩn bị truyền đi
Mov     sbuf,A          ;truyền dữ liệu 8 bit trong A và bit thứ 9 trong TB8 đi.
```

Trong kiểu 1 ta vẫn có thể sử dụng bit kiểm tra chẵn lẻ như sau: 8 bit data được phát trong kiểu 1 có thể bao gồm 7 bit dữ liệu, và 1 bit kiểm tra chẵn lẻ. Để phát một mã ASCII 7 bit với 1 bit kiểm tra chẵn vào 8 bit, các lệnh sau đây được dùng:

```
MOV   C, P          ; Đưa Parity chẵn vào C
MOV   ACC.7, C      ; Đưa Parity chẵn vào bit MSB của A
MOV   SBUF, A       ; Gởi bit data cùng bit Parity chẵn
```

4. Cờ ngắt :

Cờ ngắt nhận RI và phát TI trong thanh ghi SCON đóng một vai trò quan trọng trong truyền dữ liệu của MCS51. Cả hai bit đều được set bởi phần cứng nhưng phải xóa bởi phần mềm.

Điển hình là cờ RI được set ở mức 1 khi kết thúc quá trình nhận đầy đủ 1 kí tự và cho biết thanh ghi đệm thu đã đầy. Trạng thái của cờ RI có thể kiểm tra bằng phần mềm hoặc có thể lập trình để sinh ra ngắt. Nếu muốn nhận một ký tự từ một thiết bị đã được kết nối đến Port nối tiếp, thì chương trình phải chờ cho đến khi cờ RI = 1, sau đó xóa cờ RI và đọc ký tự từ thanh ghi SBUF. Quá trình này được lập trình như sau :

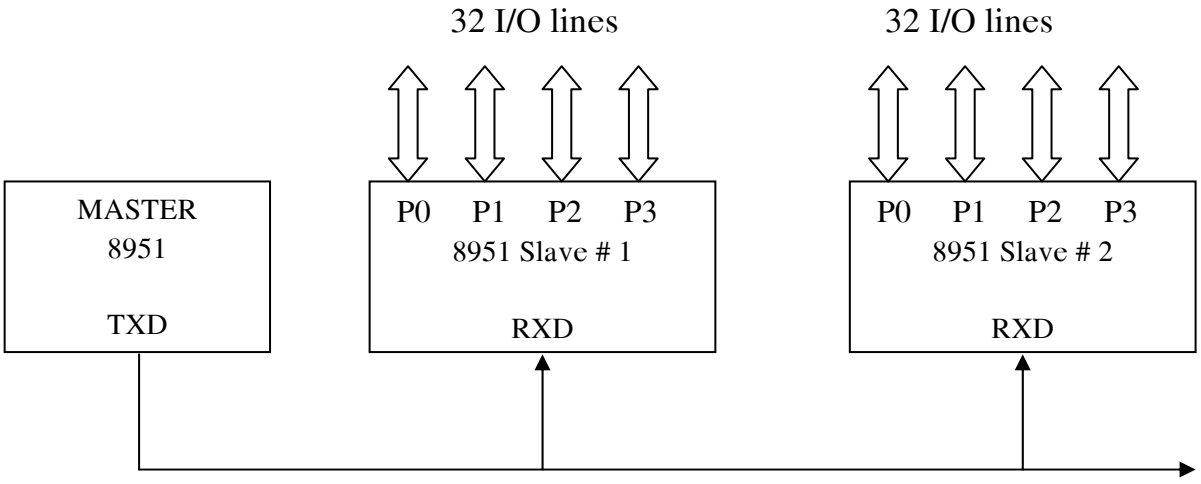
```
WAIT :    JNB  RI, WAIT    : Kiểm tra RI xem có bằng 1 hay không. Chờ nếu = 0
          CLR  RI          : khi cờ RI = 1 thì đã nhận xong dữ liệu và xóa cờ RI
          MOV A, SBUF      : đọc ký tự nhận được từ thanh ghi Sbuf
```

Cờ TI lên mức 1 cho biết đã phát xong ký tự và cho biết thanh ghi đệm sbuf đã rỗng. Nếu muốn gởi 1 ký tự đến một thiết bị đã được kết nối đến Port nối tiếp thì trước tiên phải kiểm tra xem Port nối tiếp đã sẵn sàng chưa. Nếu ký tự trước đang được gởi đi, thì phải chờ cho đến khi kết thúc quá trình gởi. Các lệnh sau đây dùng để phát một ký tự trong thanh ghi A ra :

```
WAIT :    JNB  TI, WAIT    : Kiểm tra TI có bằng 1 hay không và chờ nếu bằng 0.
          CLR  TI          : Xóa cờ ngắt thu TI
          MOV  SBUF,A      : gởi nội dung trong thanh ghi A đi
```

V. TRUYỀN DỮ LIỆU TRONG HỆ THỐNG NHIỀU BỘ XỬ LÝ:

Kiểu 2 và kiểu 3 có một chức năng đặc biệt cho việc truyền thông đa xử lý. Ở các mode 2 và 3, 9 bit dữ liệu được thu và bit thứ 9 được lưu vào bit RB8. Truyền dữ liệu có thể lập trình sao cho khi thu được bit Stop thì ngắt của truyền dữ liệu nối tiếp tác động chỉ khi bit RB8 =1. Cấu trúc này được phép bởi bằng cách set bit SM2 = 1 trong thanh ghi SCON. Kiểu này được ứng dụng trong mạng sử dụng nhiều MCS51 được tổ chức theo cấu hình máy chủ và máy tớ như hình 6-7.



Hình 6-7. Kết nối nhiều vi xử lý.

Trong cấu hình kết nối ở trên thì mỗi một vi xử lý tớ sẽ có một địa chỉ duy nhất do chúng ta qui định.

Khi bộ xử lý chủ muốn phát một khối dữ liệu đến một trong các bộ xử lý tớ thì trước tiên vi xử lý chủ phải gởi ra 1 byte địa chỉ để nhận diện bộ xử lý tớ muốn kết nối.

Byte địa chỉ được phân biệt với byte dữ liệu bởi bit thứ 9: trong byte địa chỉ thì bit thứ 9 bằng 1 và trong byte dữ liệu thì bit thứ 9 bằng 0.

Các vi xử lý tớ sau khi nhận được byte địa chỉ sẽ biết được vi xử lý chủ muốn giao tiếp tớ nào. Khi có vi xử lý tớ được phép thì nó sẽ xóa bit SM2 để bắt đầu nhận các byte dữ liệu tiếp theo. Còn các vi xử lý không được phép thì vẫn giữ nguyên bit SM2=1 để không nhận các byte dữ liệu truyền giữa vi xử lý chủ và vi xử lý tớ đang được phép. Vi xử lý tớ sau khi kết nối với vi xử lý chủ xong thì phải làm cho bit SM2=1 để sẵn sàng kết nối cho những lần tiếp theo.

Sau khi thực hiện xong việc trao đổi dữ liệu thì vi xử lý muốn truy xuất một vi xử lý khác thì phải tạo ra một địa chỉ mới và vi xử lý tớ tương ứng với địa chỉ đó được phép và hoạt động giống như vừa trình bày.

VI. TỐC ĐỘ TRUYỀN DỮ LIỆU NỐI TIẾP:

Truyền dữ liệu nối tiếp nếu hoạt động ở kiểu 0 và kiểu 2 thì có tốc độ truyền cố định. Trong kiểu 0 thì tốc độ truyền bằng $1/12$ tần số dao động trên Chip. Nếu sử dụng thạch anh 12 MHz thì tốc độ truyền của kiểu 0 là 1MHz như hình 6-8a.

Trong thanh ghi PCON có một bit SMOD có chức năng làm tăng gấp đôi tốc độ baud, mặc nhiên sau khi reset hệ thống thì bit SMOD = 0 thì các kiểu truyền dữ liệu hoạt động với tốc độ qui định, khi bit SMOD = 1 thì tốc độ tăng gấp đôi.

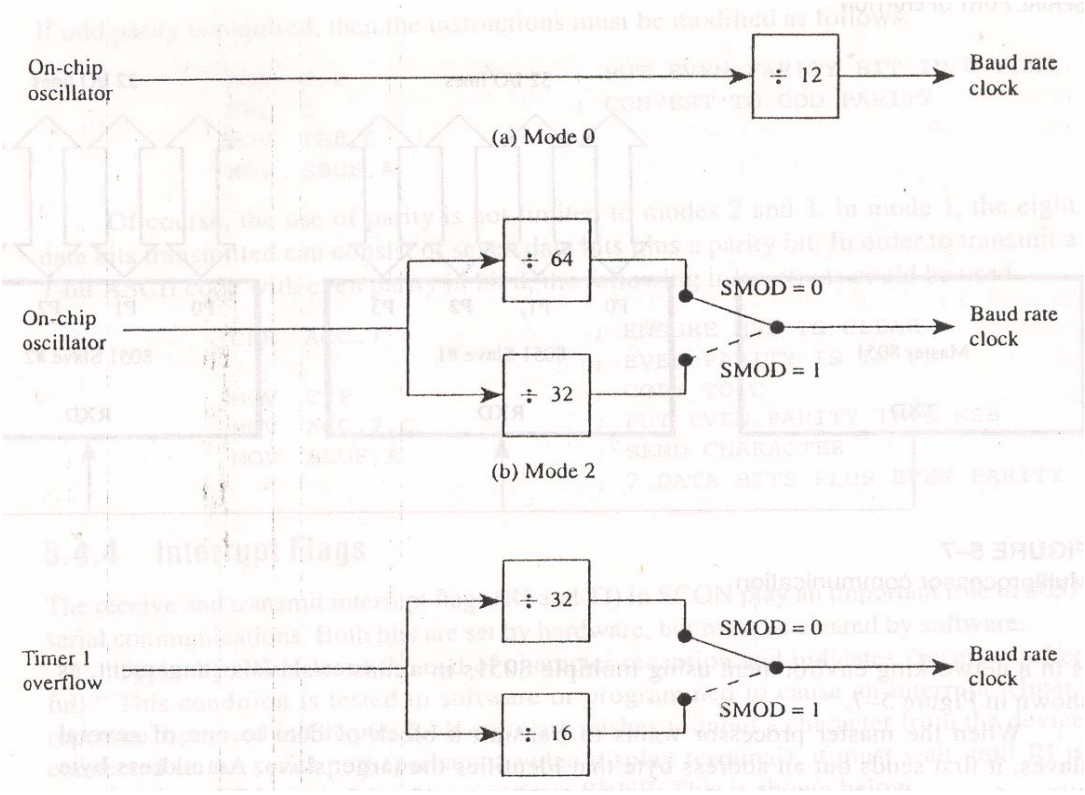
Ví dụ trong kiểu 2, tốc độ truyền có thể tăng gấp đôi từ giá trị mặc định 1/64 tần số dao động trên Chip (SMOD = 0) lên đến 1/32 tần số dao động trên Chip (ứng với SMOD =1) như hình 6-8b.

Do thanh ghi PCON không cho phép truy suất bit nên để set bit SMOD mà không thay đổi các bit khác của thanh ghi PCON thì phải thực hiện lệnh sau.

Lệnh sau đây set bit SMOD để tăng gấp đôi tốc độ truyền:

```
OR      PCON, #1000 0000b  ;bit Smod ở vị trí thứ 7
```

Các tốc độ Baud trong kiểu 1 và kiểu 3 của MCS51 được xác định bởi tốc độ tràn của Timer 1. Bởi vì Timer hoạt động ở tần số tương đối cao nên phải chia cho 32 khi bit smod = 0 và chia cho 16 nếu SMOD = 1 trước khi cung cấp xung clock để thiết lập tốc độ Baud cho Port nối tiếp. Tốc độ Baud ở kiểu 1 và 3 của MCS51 được xác định bởi tốc độ tràn của Timer 1 hoặc Timer 2, hoặc cả 2 như hình 6-8c.



Hình 6-8. Thiết lập tốc độ Baud.

Thiết lập tốc độ Baud dùng timer 1:

Muốn có tốc độ Baud thì ta khởi tạo thanh ghi TMOD ở kiểu tự động nạp 8 bit (kiểu 2) và đặt giá trị nạp lại vào thanh ghi TH1 của Timer 1 để tạo ra tốc độ tràn chính xác để thiết lập tốc độ Baud. Thanh ghi tmod được khởi tạo để thiết lập tốc độ baud như sau:

Mov tmod,#0010xxxxB ;chỉ quan tâm đến timer 1

Một cách khác để tạo tốc độ baud là nhận tín hiệu xung clock từ bên ngoài đưa đến ngõ vào T1. Công thức chung để xác định tốc độ Baud trong mode 1 và mode 3 là :

BAUD RATE = TIMER 1 OVERFLOW RATE ÷ 32

Ví dụ1: truyền dữ liệu cần tốc độ baud là 1200 thì ta tính toán như sau:
Tốc độ tràn của timer 1 bằng 1200 x 32 = 38,4KHz. Nếu hệ thống sử dụng thạch anh 12 MHz thì xung cung cấp cho Timer 1 đếm có tần số là 1 MHz hay 1000KHz. Vậy để đạt tốc độ tràn 38,4 KHz thì ta tính được số lượng xung đếm cho mỗi chu kỳ tràn là 1000 KHZ/38,4 KHz = 26,4 xung (làm tròn bằng 26).

Do các Timer đếm lên và thời điểm tràn xảy ra khi chuyển trạng thái đếm từ từ FFH → 00H nên ta phải nạp giá trị bắt đầu từ (256 – 26 = 230) để từ giá trị này timer 1 đếm lên 26 xung nữa thì sinh ra tràn. Giá trị 230 được nạp vào thanh ghi TH1 để tự động nạp lại cho thanh ghi TL1 khi tràn bằng lệnh: "*mov th1,#230*". Bạn có thể không cần phải tính toán ra giá trị 230 mà có thể thay bằng lệnh : "mov th1,#-26" thì trình biên dịch sẽ tính cho bạn.

Bảng tóm tắt tốc độ Baud ứng với 2 loại thạch anh 12 MHz và 11, 059 MHz :

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

Bảng 6-3. Tóm tắt tốc độ baud.

Ví dụ 2: hãy khởi tạo truyền dữ liệu nối tiếp hoạt động như UART 8 bit ở tốc độ Baud 2400, dùng Timer 1 để tạo tốc độ Baud.
Chương trình sau sẽ thiết lập đúng theo yêu cầu đề ra:

```
MOV     SCON, # 01010010B   : Port nối tiếp mode 1.
MOV     TMOD, # 20H         : Timer 1 mode 2
MOV     TH1, # -13          : Nạp vào bộ đếm  tốc độ 2400 Baud.
SETB    TR1                 : Start Timer 1.
```

Trong thanh ghi SCON có: hai bit SM0 SM1 = 01 thiết lập mode UART 8 bit, bit REN = 1 cho phép sẳn sàng nhận dữ liệu, bit TI = 1 báo cho biết thanh ghi đếm rỗng sẳn sàng cho phép phát dữ liệu.

Thanh ghi TMOD có: hai bit M1M0 = 10 để thiết lập Timer 1 ở mode 2 tự động nạp 8 bit. Lệnh setb TR1 cho phép Timer làm việc tạo tốc độ baud.

Từ tốc độ Baud 2400 ta tính được tốc độ tràn cho Timer 1 là 2400 x 32 = 76,8 KHz và giả sử Timer 1 đếm xung nội ở tần số 1000 KHz (ứng với thạch anh 12 MHz).

Vậy để đạt tốc độ tràn 76,8 KHz thì ta tính được số lượng xung đếm cho mỗi chu kỳ tràn là 1000KHZ/76,8KHz = 13,02 xung (làm tròn bằng 13). Nên lệnh thứ 3 sẽ nạp giá trị -13 vào thanh ghi TH1 để tạo tốc độ baud là 2400.

Thủ tục chờ nhận một kí tự:

```
Mainr:  jnb ri,$        ;chờ cho đến khi cờ báo nhận lên 1
        Clr ri          ;xóa cờ để nhận kí tự tiếp theo
        Mov @r0,sbuf    ;cất dữ liệu vào ô nhớ.
```

Nếu chỉ nhận một kí tự thì sau khi nhận xong CPU sẽ thực hiện công việc xử lý khác, còn nếu muốn nhận nữa thì quay lại.

Thủ tục chờ gởi một kí tự:

```
Mains:  jnb ti,$        ;kiểm tra máy phát sẳn sàng hay chưa
        Clr ti          ;xóa cờ để chuẩn bị phát dữ liệu
```

```
Mov sbuf,@r0     ;lấy dữ liệu từ ô nhớ truyền đi.
```

Việc sử dụng truyền dữ liệu ở tốc độ baud nào tùy thuộc vào yêu cầu thực tế. Tốc độ càng cao thì dữ liệu truyền càng nhanh. Khi truyền nhiều dữ liệu thì ngoài tốc độ qui định thống nhất giữa 2 hệ thống kết nối với nhau còn phải quan tâm đến tốc độ xử lý dữ liệu nhận về và lấy dữ liệu để gởi đi để không bị mất dữ liệu trong quá trình truyền và nhận. Một trong những giải pháp để kiểm tra xem dữ liệu có bị mất hay không thì phải sử dụng thủ tục bắt tay.

Chương 7

KHẢO SÁT NGẮT CỦA VI ĐIỀU KHIỂN

I. Giới thiệu.

II. Tổ chức ngắt.

III. Xử lý ngắt.

I. GIỚI THIỆU :

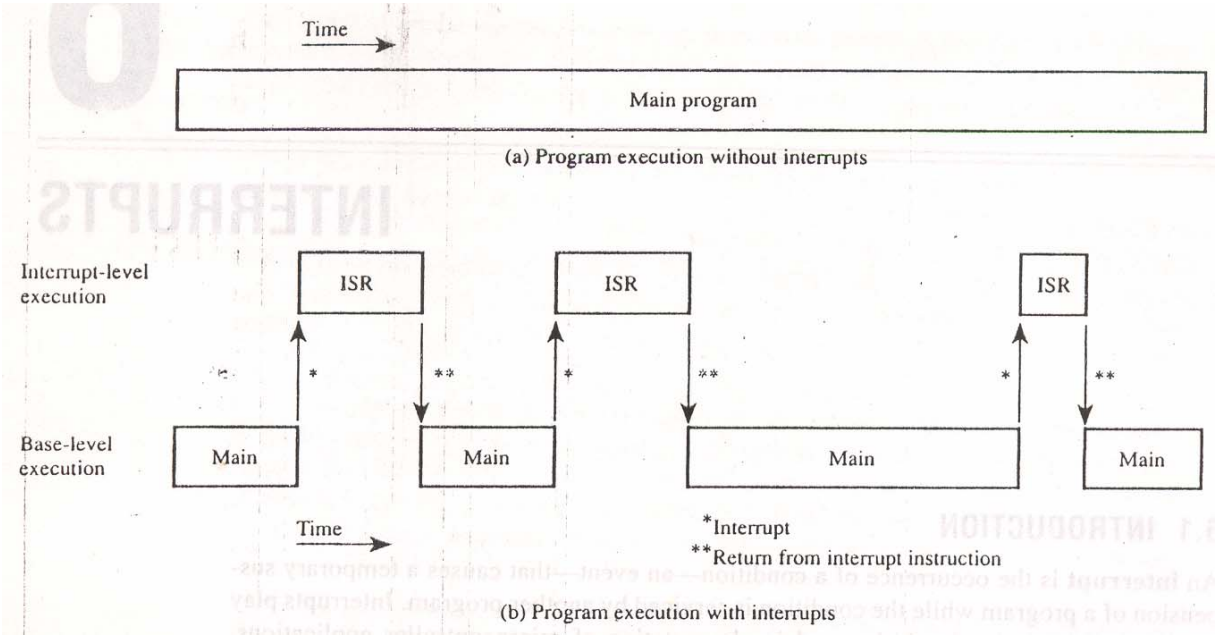
Ngắt sử dụng trong vi xử lý hay vi điều khiển hoạt động như sau: vi xử lý hay vi điều khiển luôn thực hiện một chương trình mà ta thường gọi là chương trình chính, khi có một sự tác động từ bên ngoài bằng phần cứng hay sự tác động bên trong làm cho vi xử lý ngừng thực hiện chương trình chính để thực hiện một chương trình khác (còn gọi là chương trình phục vụ ngắt ISR) và sau khi thực hiện xong vi xử lý trở lại thực hiện tiếp chương trình chính. Quá trình làm gián đoạn vi xử lý thực hiện chương trình chính xem như là ngắt.

Có nhiều sự tác động làm ngừng chương trình chính gọi là các nguồn ngắt, trong vi điều khiển khi timer/counter đếm tràn sẽ tạo ra ngắt. Ngắt đóng một vai trò quan trọng trong lập trình điều khiển.

Khi sử dụng ngắt sẽ cho phép vi xử lý hay vi điều khiển đáp ứng nhiều sự kiện quan trọng và giải quyết sự kiện đó trong khi chương trình khác đang thực thi. Ví dụ: vi điều khiển đang thực hiện chương trình chính thì có dữ liệu từ hệ thống khác gởi đến thì vi điều khiển ngừng chương trình chính để thực hiện chương trình phục vụ ngắt nhận dữ liệu xong rồi trở lại tiếp tục thực hiện chương trình chính, hoặc có một tín hiệu báo ngắt từ bên ngoài thì vi điều khiển sẽ ngừng thực hiện chương trình chính để thực hiện chương trình ngắt rồi tiếp tục thực hiện chương trình chính.

Ta có thể sử dụng ngắt để yêu cầu vi điều khiển thực hiện nhiều chương trình cùng một lúc có nghĩa là các chương trình được thực hiện xoay vòng.

Ta có thể minh họa quá trình thực hiện 1 chương trình trong trường hợp có ngắt và không có ngắt như hình 7-1.



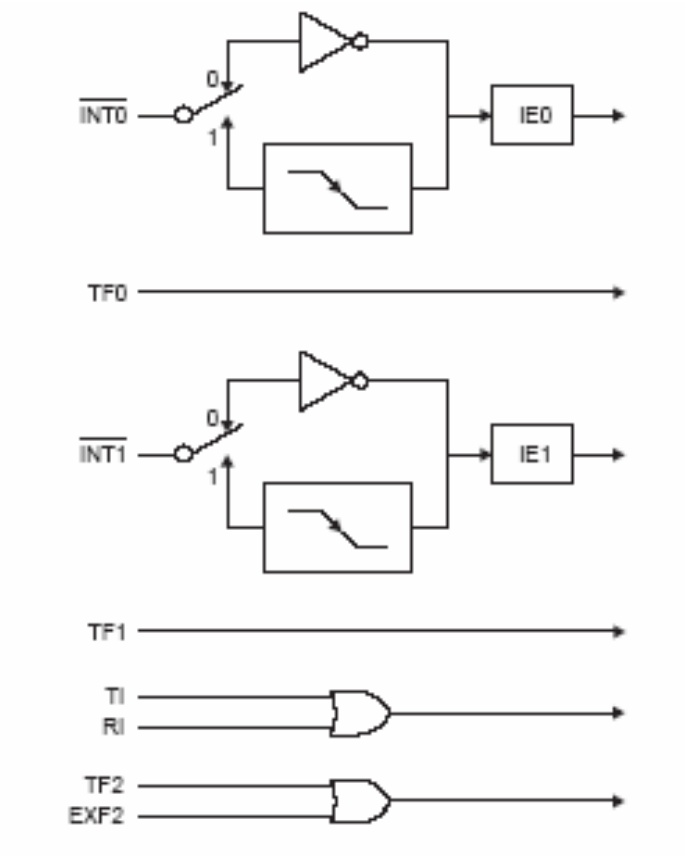
Hình 7-1. Vi điều khiển thực hiện chương trình chính trong 2 trường hợp không và có ngắt.

Trong đó : Ký hiệu * cho biết vi điều khiển ngừng chương trình chính để thực thi chương trình con phục vụ ngắt ISR. Còn ký hiệu ** cho biết vi điều khiển quay trở lại thực hiện tiếp chương trình chính sau khi thực hiện xong chương trình con phục vụ ngắt ISR.

II. TỔ CHỨC NGẮT (INTERRUPT ORGANIZATION) :

Vi điều khiển 89C51 có 5 nguồn ngắt: 2 ngắt ngoài, 2 ngắt Timer và một ngắt Port nối tiếp. Vi điều khiển 89C52 có thêm một nguồn ngắt là của timer T2 như hình 7-2. Mặc nhiên khi vi điều khiển bị reset thì tất cả các ngắt sẽ mất tác dụng và được cho phép bởi phần mềm.

Trong trường hợp có hai hoặc nhiều nguồn ngắt tác động đồng thời hoặc vi điều khiển đang phục vụ ngắt thì xuất hiện một ngắt khác, thì sẽ có hai cách giải quyết là kiểm tra liên tiếp và sử dụng chế độ ưu tiên.



Hình 7-2. Vi điều khiển 89C52 có 6 nguồn ngắt.

1. *Cho phép / cấm ngắt (Enable and disabling Interrupt)*

Trước tiên chúng ta phải hiểu cho phép và không cho phép ngắt là như thế nào ? Khi ta cho phép ngắt và khi ngắt tác động thì vi điều khiển sẽ ngừng chương trình chính để thực hiện chương trình con phục vụ ngắt, còn khi không cho phép thì dù có sự tác động đến ngắt vi điều khiển vẫn tiếp tục thực hiện chương trình chính – không thực hiện chương trình phục vụ ngắt.

Trong vi điều khiển có 1 thanh ghi IE (Interrupt Enable) ở tại địa chỉ 0A8H có chức năng cho phép / cấm ngắt. Ta sử dụng thanh ghi này để cho phép hay không cho phép đối với từng nguồn ngắt và cho toàn bộ các nguồn ngắt.

Hoạt động của từng bit trong thanh ghi cho phép ngắt IE được tóm tắt trong bảng 7-1:

Bit	Kí hiệu	Địa chỉ bit	Chức năng (Enable = 1; Dissble = 0)

IE.7	EA	AFH	Cho phép toàn bộ hoặc cấm toàn bộ các nguồn ngắt.
IE.6	-	AEH	Chưa dùng đến
IE.5	ET2	ADH	Cho phép ngắt Timer 2 (8052).
IE.4	ES	ACH	Cho phép ngắt Port nối tiếp.
IE.3	ET1	ABH	Cho phép ngắt Timer 1.
IE.2	EX1	AAH	Cho phép ngắt ngoài External 1 (INT1).
IE.1	ET0	A9H	Cho phép ngắt Timer 0.
IE.0	EX0	A8H	Cho phép ngắt ngoài External 0 (INT0).

Bảng 7-1. Tóm tắt chức năng các bit của thanh ghi IE.

Trong thanh ghi IE có bit IE.6 chưa dùng đến, bit IE.7 là bit cho phép/cấm ngắt toàn bộ các nguồn ngắt. Khi bit IE.7= 0 thì cấm hết tất cả các nguồn ngắt, khi bit IE.7=1 thì cho phép tất cả các nguồn ngắt nhưng còn phụ thuộc vào từng bit điều khiển ngắt của từng nguồn ngắt.

Ví dụ để cho phép Timer 1 ngắt ta có thể thực hiện trên bit:

```
SETB EA          ;cho phép ngắt toàn bộ
SETB ET1         ;cho phép timer 1 ngắt
```

Hoặc có thể dùng lệnh sau:

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

Đối với yêu cầu của ví dụ trên thì 2 cách thực hiện trên là xong nhưng ta hãy so sánh 2 cách thực hiện và chú ý một vài điều trong lập trình:

Các lệnh của cách 1 không ảnh hưởng các bit còn lại trong thanh ghi IE.

Cách thứ hai sẽ xóa các bit còn lại trong thanh ghi IE.

Ở đầu chương trình ta nên khởi gán IE với lệnh MOV BYTE, nhưng khi điều khiển cho phép hay cấm trong chương trình thì ta sẽ dùng các lệnh SET BIT và CLR BIT để tránh làm ảnh hưởng đến các bit khác trong thanh ghi IE.

2. *Ưu tiên ngắt (Interrupt Priority) :*

Khi có nhiều nguồn ngắt tác động cùng lúc thì ngắt nào quan trọng cần thực hiện trước và ngắt nào không quan trọng thì thực hiện sau giống như các công việc mà ta giải quyết hằng ngày. Ngắt cũng được thiết kế có sự sắp xếp thứ tự ưu tiên từ thấp đến cao để người lập trình sắp xếp các nguồn ngắt theo yêu cầu công việc mà mình xử lý.

Thanh ghi có chức năng thiết lập chế độ ưu tiên trong vi điều khiển là thanh ghi IP (Interrupt Priority) tại địa chỉ 0B8H. Hoạt động của từng bit trong thanh ghi IP được tóm tắt trong bảng 7-2.

Bit	**Kí hiệu**	**Địa chỉ bit**	**Chức năng**
IP.7	-	-	Chưa sử dụng
IP.6	-	-	Chưa sử dụng
IP.5	PT2	BDH	Ưu tiên cho sự ngắt Timer 2 (8052).
IP.4	PS	BCH	Ưu tiên cho sự ngắt Port nối tiếp.
IP.3	PT1	BBH	Ưu tiên cho sự ngắt Timer 1.

IP.2	PX1	BAH	Ưu tiên cho sự ngắt ngoài External 1.
IP.1	PT0	B9H	Ưu tiên cho sự ngắt Timer 0.
IP.0	PX0	B8H	Ưu tiên cho sự ngắt ngoài External 0.

Bảng 7-2. Tóm tắt chức năng các bit của thanh ghi IP.

Khi reset hệ thống thì thanh ghi ưu tiên ngắt IP bị xóa và tất cả các ngắt ở mức ưu tiên thấp nhất.

Trong 89C51 có 2 mức ưu tiên thấp và 2 mức ưu tiên cao. Nếu vi điều khiển đang thực hiện chương trình con phục vụ ngắt có mức ưu tiên thấp và có một yêu cầu ngắt với mức ưu tiên cao hơn xuất hiện thì vi điều khiển phải ngừng thực hiện chương trình con phục vụ ngắt có mức ưu tiên thấp để thực hiện chương trình con phục vụ ngắt mới có ưu tiên cao hơn.

Ngược lại nếu vi điều khiển đang thực hiện chương trình con phục vụ ngắt có mức ưu tiên cao hơn và có yêu cầu ngắt với mức ưu tiên thấp hơn xuất hiện thì vi điều khiển vẫn tiếp tục thực hiện cho đến khi thực hiện xong chương trình phục vụ ngắt có ưu tiên cao hơn rồi mới thực hiện chương trình phục vụ ngắt có ưu tiên thấp đang yêu cầu.

Chương trình chính mà vi điều khiển luôn thực hiện trong một hệ thống thì ở mức thấp nhất, không có liên kết với yêu cầu ngắt nào, luôn luôn bị ngắt bất chấp ngắt ở mức ưu tiên cao hay thấp. Nếu có 2 yêu cầu ngắt với các ưu tiên khác nhau xuất hiện đồng thời thì yêu cầu ngắt có mức ưu tiên cao hơn sẽ được phục vụ trước.

3. *Kiểm tra vòng quét liên tiếp.*

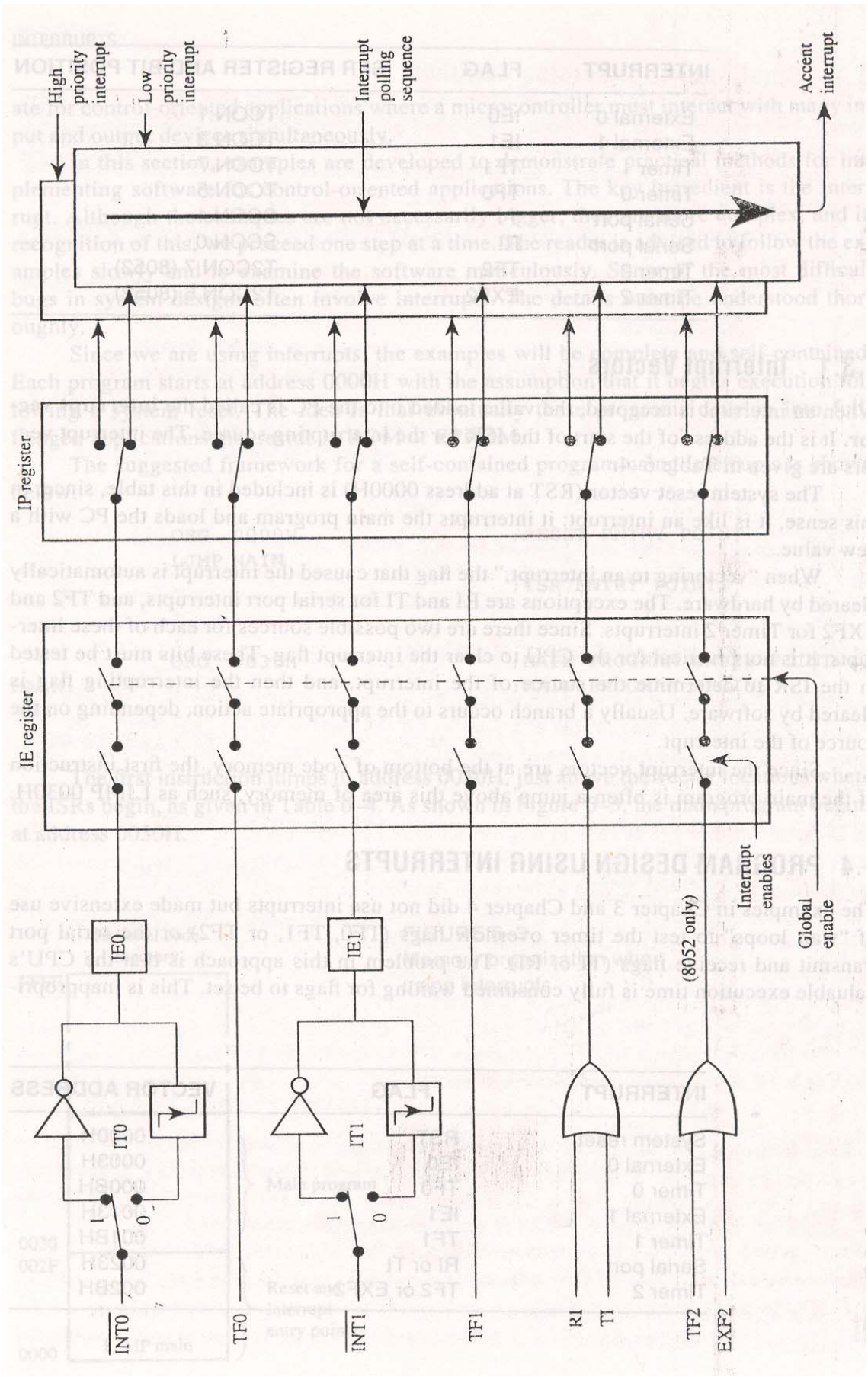
Nếu 2 yêu cầu ngắt có cùng mức ưu tiên xuất hiện đồng thời thì vòng quét kiểm tra liên tiếp sẽ xác định yêu cầu ngắt nào sẽ được phục vụ trước tiên. Vòng quét kiểm tra liên tiếp theo thứ tự ưu tiên từ trên xuống là: ngắt ngoài thứ 0 (INT0), ngắt timer T0, ngắt ngoài thứ 1 (INT1), ngắt Timer 1, ngắt truyền dữ liệu nối tiếp (serial Port), ngắt timer 2. Hình 7-3 sẽ minh họa cho trình tự trên.

Quan sát trong hình 7-3 chúng ta thấy có 6 nguồn ngắt của 89C52 và tác dụng của các thanh ghi IE hoạt động như một contact On/Off còn thanh ghi IP hoạt động như một contact chuyển mạch giữa 2 vị trí để lựa chọn 1 trong 2.

Ta hãy bắt đầu từ thanh ghi IE trước: bit cho phép ngắt toàn cục (global enable) nếu được phép sẽ đóng toàn bộ các contact và tùy thuộc vào bit cho phép của từng nguồn ngắt có được phép hay không và chúng hoạt động cũng giống như một contact: nếu được phép thì đóng mạch và tín hiệu yêu cầu ngắt sẽ đưa vào bên trong để xử lý, nếu không được phép thì contact hở mạch nên tín hiệu yêu cầu ngắt sẽ không đưa vào bên trong và không được xử lý.

Tiếp theo là thanh ghi IP: tín hiệu sau khi ra khỏi thanh ghi IE thì đưa đến thanh ghi IP để sắp xếp ưu tiên cho các nguồn ngắt. Có 2 mức độ ưu tiên: mức ưu tiên cao và mức ưu tiên thấp. Nếu các nguồn nào có ưu tiên cao thì contact chuyển mạch sẽ đưa tín hiệu yêu cầu ngắt đó đến vòng kiểm tra có ưu tiên cao, nếu các nguồn nào có ưu tiên thấp thì contact chuyển mạch sẽ đưa tín hiệu yêu cầu ngắt đó đến vòng kiểm tra có ưu tiên thấp.

Vòng kiểm tra ngắt ưu tiên cao sẽ được thực hiện trước và sẽ kiểm tra theo thứ tự từ trên xuống và khi gặp yêu cầu ngắt nào thì yêu cầu ngắt đó sẽ được thực hiện. Sau đó tiếp tục thực hiện cho vòng kiểm tra ưu tiên ngắt có mức ưu tiên thấp hơn.



Hình 7-3. Cấu trúc ngắt của vi điều khiển.

Trong hình còn cho chúng ta thấy yêu cầu ngắt truyền dữ liệu nối tiếp tạo ra từ tổ hợp OR của 2 cờ báo nhận RI và cờ báo phát TI. Khi ngắt truyền dữ liệu xảy ra và ta muốn biết là do cờ nhận hay cờ phát tạo ra ngắt để thực hiện 2 công việc khác nhau thì ta phải kiểm tra cờ RI và TI để biết thực hiện công việc nào tương ứng.

Ví dụ trong truyền dữ liệu: khi có báo ngắt truyền dữ liệu thì ta phải kiểm tra xem cờ RI = 1 hay không? Nếu đúng thì hệ thống khác đang gởi dữ liệu đến và ta phải chuyển hướng chương trình phục vụ ngắt sang hướng nhận dữ liệu, nếu không phải thì chắc chắn là cờ TI=1 báo cho chúng ta biết rằng dữ liệu đã truyền đi xong và sẵn sàng truyền kí tự tiếp theo và khi đó ta phải chuyển hướng chương trình phục vụ ngắt sang phát dữ liệu tiếp theo.

Tương tự, các yêu cầu ngắt của Timer 2 tạo ra từ tổ hợp OR của cờ tràn TF2 và cờ nhập ngoài EXF2.

Các bit cờ của các nguồn ngắt được tóm tắt ở bảng 7-3:

Interrupt	Flag	SFR Register and Bit Position
External 0	IE 0	TCON 1
External 1	IE 1	TCON 3
Timer 1	TF 1	TCON 7
Timer 0	TF 0	TCON 5
Serial Port	TI	SCON 1
Serial Port	RI	SCON 0
Timer 2	TF 2	T2CON 7 (8052)
Timer 2	EXF 2	T2CON 6 (8052)

Bảng 7-3. Tóm tắt các bit cờ của các nguồn ngắt.

III. XỬ LÝ NGẮT:

Khi tín hiệu yêu cầu ngắt xuất hiện và được chấp nhận bởi CPU thì CPU thực hiện các công việc sau:

- Nếu CPU đang thực hiện lệch thì phải chờ thực hiện xong lệnh đang thực hiện.
- Giá trị của bộ đếm chương trình PC được cất giữ vào Stack (chính là địa chỉ của lệnh tiếp theo trong chương trình chính).
- Trạng thái ngắt hiện hành được lưu vào bên trong.
- Các yêu cầu ngắt khác sẽ bị ngăn lại.
- Địa chỉ của chương trình phục vục ngắt tương ứng sẽ được nạp vào bộ đếm chương trình PC.
- Bắt đầu thực hiện chương trình phục vụ ngắt ISR.

Trong chương trình phục vụ ngắt luôn kết thúc bằng lệnh ***RETI.*** Khi gặp lệnh RETI thì CPU sẽ lấy lại địa chỉ của lệnh tiếp theo trong ngăn xếp trả lại cho thanh ghi PC để tiếp tục thực hiện các công việc tiếp theo của chương trình chính.

Chú ý: chương trình con phục vụ ngắt ***không được làm mất hoặc làm sai địa chỉ của PC đã lưu trong ngăn xếp*** nếu điều này xảy ra thì khi trở lại chương trình chính CPU sẽ không thực hiện tiếp công việc của chương trình chính và chúng ta cũng không xác định CPU đang làm gì và ở đâu. Khi đó chúng ta mất quyền kiểm soát vi xử lý.

Giống như ta đang đọc một cuốn sách vì một công việc khác ta phải ngừng lại và ta có làm dấu tại trang ta tạm ngừng, sau khi làm xong công việc thì ta tiếp tục quay lại để đọc tiếp cuốn sách tại nơi ta đã dừng. Tất cả đều xảy ra như vậy thì rất là bình thường nhưng trong khi ta thực hiện công việc thì có một người khác xem cuốn sách của ta và vô tình làm mất dấu thì khi ta quay lại ta sẽ đọc không đúng trang chúng ta đang dừng lại. Nguyên tắc làm việc của vi xử lý hoàn toàn giống như vậy.

Trong "vi điều khiển" thì bộ nhớ ngăn xếp là bộ nhớ RAM nội nên chúng sẵn sàng hoạt động cho việc lưu trữ tạm, còn đối với "vi xử lý" thì bộ nhớ ngăn xếp sử dụng bộ nhớ ngoài nên bạn phải khởi tạo bộ nhớ ngăn xếp phải là vùng nhớ RAM để có thể ghi và đọc lại được, nếu bạn khởi tạo tại vùng nhớ EPROM hoặc khởi tạo tại nơi mà bộ không ghi vào được thì sẽ làm mất địa chỉ – dữ liệu lưu vào bộ nhớ ngăn xếp dẫn đến chương trình sẽ thực hiện sai.

Một điều cần phải chú ý nữa là trong lập trình chúng ta không được nhảy từ chương trình con sang chương trình chính để thực hiện tiếp chương trình vì làm như vậy sau nhiều lần thực hiện thì bộ nhớ ngăn xếp sẽ bị tràn và ghi đè lên các dữ liệu khác làm sai chương trình. Trong trường hợp này chúng ta sẽ thấy rằng chương trình chúng ta thực hiện đúng một vài lần và sau đó thì sai.

Các vectơ ngắt (Interrupt Vectors) :

Như đã trình bày ở trên, khi có một yêu cầu ngắt xảy ra thì sau khi cất giá trị địa chỉ trong PC vào ngăn xếp thì địa chỉ của chương trình con phục vụ ngắt tương ứng còn gọi bởi vectơ địa chỉ ngắt sẽ được nạp vào thanh ghi PC, địa chỉ này là cố định và do nhà chế tạo vi điều khiển qui định. Các chương trình ngắt phải bắt đầu viết đúng tại địa chỉ quy định đó. Các vectơ địa chỉ ngắt được cho trong bảng 7-4:

Interrupt	Flag	Vectors Address
System Reset	RST	0000H
External 0	IE 0	0003H
Timer 0	TF 0	000BH
External 1	IE 1	0013H
Timer 1	TF1	001BH
Serial Port	RI or TI	0023H
Timer 2	TF 2 or EXF2	002BH

Bảng 7-4. Tóm tắt vector địa chỉ ngắt.

Vectơ reset hệ thống bắt đầu tại địa chỉ 0000H: khi reset vi điều khiển thì thanh ghi PC = 0000H và chương trình chính luôn bắt đầu tại địa chỉ này.

Khi bạn sử dụng yêu cầu ngắt nào thì chương trình con phục vụ ngắt phải viết đúng tại địa chỉ tương ứng. Ví dụ bạn sử dụng ngắt timer T0 thì chương trình ngắt bạn phải viết tại địa chỉ 000BH.

Do khoảng vùng nhớ giữa các vector địa chỉ của các nguồn ngắt chỉ có vài ô nhớ ví dụ như vector địa chỉ ngắt của ngắt INT0 tại 0003H và vector địa chỉ ngắt của ngắt T0 tại 000BH chỉ cách nhau có 9 ô nhớ. Nếu chương trình phục vụ ngắt của ngắt INT0 có kích thước lớn hơn 9 byte thì nó sẽ đụng đến vùng nhớ của ngắt T0. Cách giải quyết tốt nhất là ngay tại địa chỉ 0003H ta viết lệnh nhảy đến một vùng nhớ khác rộng hơn. Còn nếu các ngắt T0 và các ngắt khác không sử dụng thì ta có thể viết chương trình tại đó cũng được.

Chương trình chính luôn bắt đầu tại địa chỉ 0000H sau khi reset hệ thống, nếu trong chương trình có sử dụng ngắt thì ta phải dùng lệnh nhảy tại địa chỉ 0000H để nhảy đến một vùng nhớ khác rộng hơn không bị giới hạn để viết tiếp.

Chương 8

CẤU HÌNH BỘ THÍ NGHIỆM VI ĐIỀU KHIỂN CHƯƠNG TRÌNH SPKT-C

I. Giới thiệu:

II. Cấu hình bộ thí nghiệm:

III. Hướng dẫn sử dụng phần mềm SPKT_C:

I. GIỚI THIỆU:

Sau khi đã nghiên cứu vi điều khiển ở các chương từ 1 đến 7 thì bạn có thể bắt đầu thực hiện các bài thực hành đối với vi điều khiển để giúp bạn hiểu rõ hơn những gì bạn đã đọc.

Các ứng dụng của vi điều khiển rất đa dạng nên trong chương này tôi muốn giới thiệu đến các bạn các bộ thí nghiệm vi điều khiển tương đối đầy đủ các yêu cầu phần cứng và rất nhiều chương trình điều khiển có thể giúp bạn thực hành, thí nghiệm và có thể tự nghiên cứu, tự học.

Bộ thí nghiệm vi điều khiển do chúng tôi sản xuất có nhiều loại:

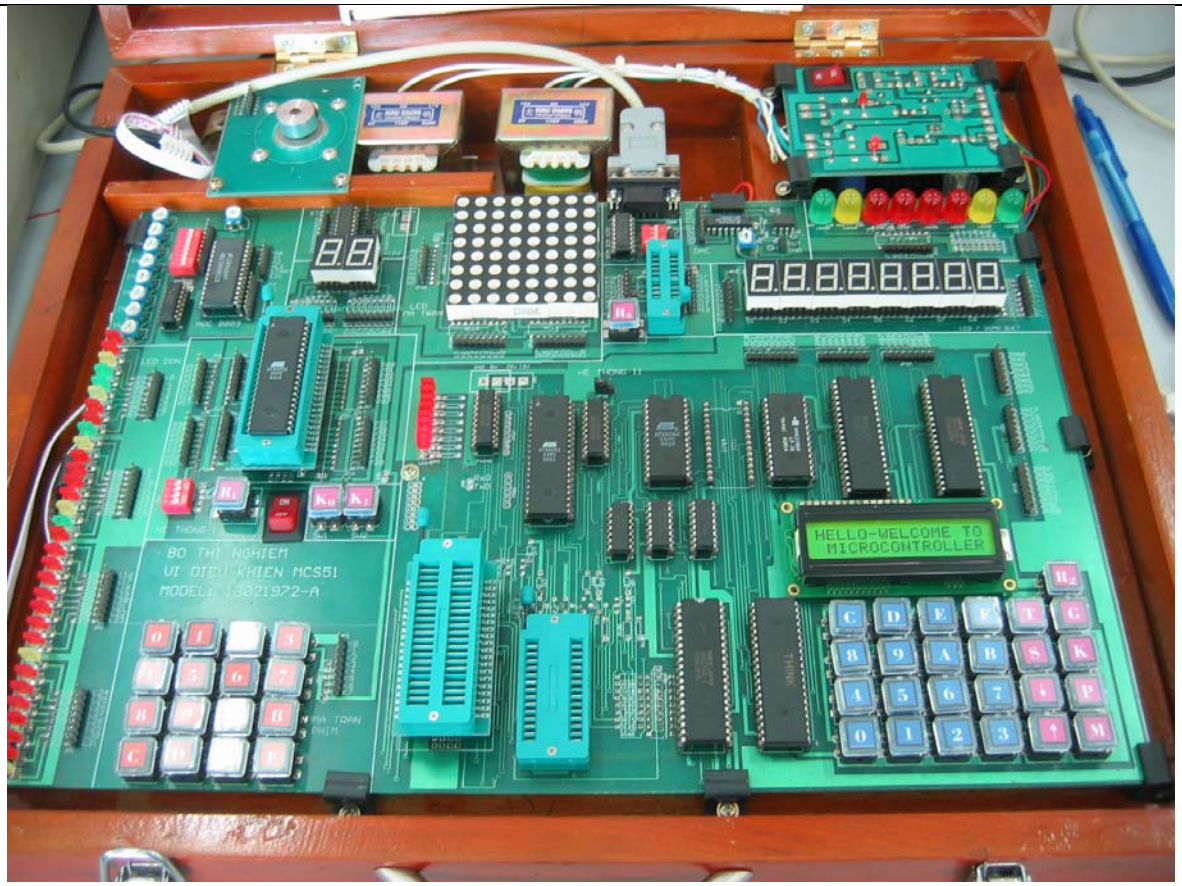
Bộ thí nghiệm loại lớn và bộ thí nghiệm loại nhỏ.

Bộ thí nghiệm loại lớn hình 8-1 có cấu hình như sau:

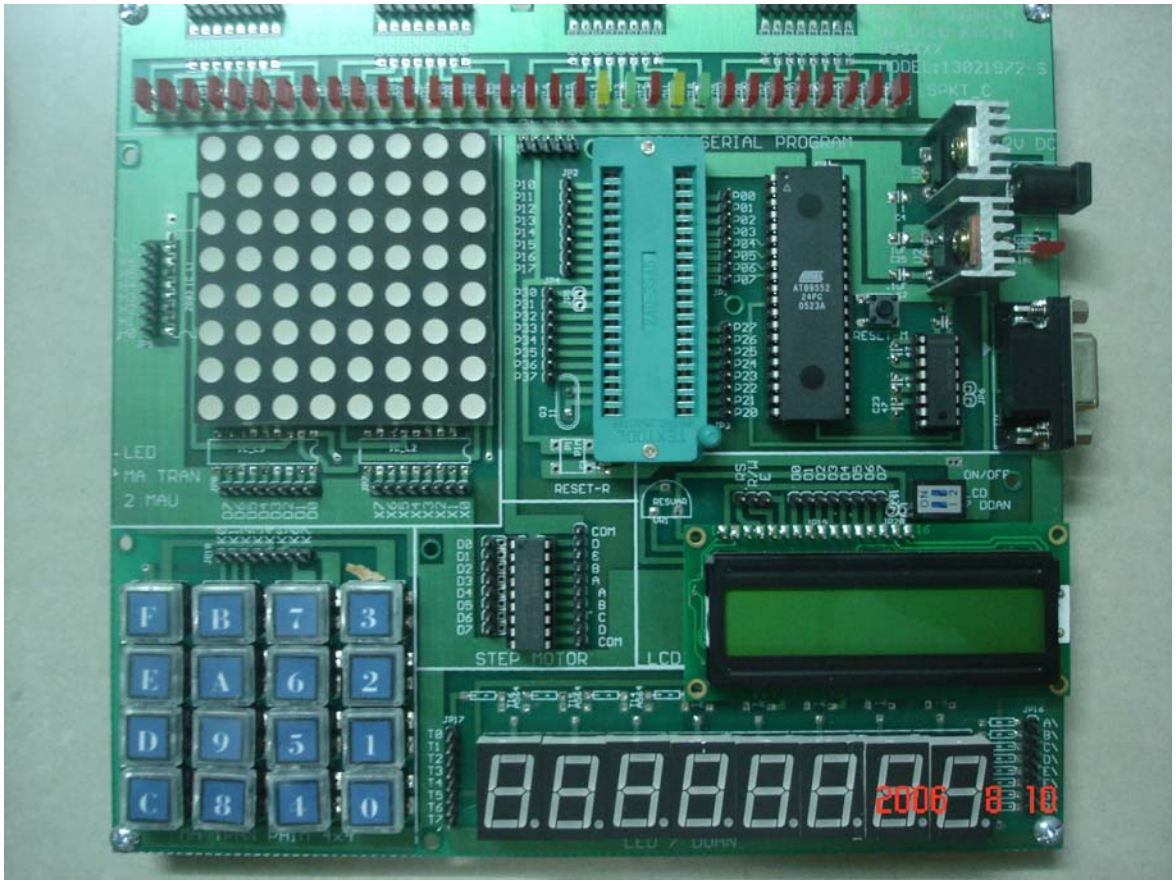
- ✓ Bộ thí nghiệm sử dụng các loại vi điều khiển 89C51, 89C52, 89S52, 89S51, 89S8252, 89C1051, 89C2051, 89C4051. Đặc biệt là họ 89S có nhiều tính năng hay hơn họ 89C.
- ✓ Các modul thí nghiệm gồm có:
  - Hệ thống thí nghiệm vi điều khiển dùng bộ nhớ nội có đệm các port ngõ ra.
  - Led ma trận 8x8 hai màu xanh và đỏ.
  - Có 32 led đơn 3mm và 8 led đơn 10mm tích cực mức 1.
  - Led 7 đoạn gồm 8 Led kết nối theo phương pháp quét.
  - Bàn phím 16 phím tổ chức theo ma trận.
  - Chuyển đổi tín hiệu tương tự sang số (ADC) với số lượng kênh ngõ vào là 8.
  - Chuyển đổi tín hiệu số sang tín hiệu tương tự (DAC).
  - Mạch giao tiếp điều khiển động cơ bước.
  - Một hệ thống vi điều khiển sử dụng bộ nhớ ngoài có giao tiếp với máy tính qua cổng COM, giao tiếp với 2 IC ngoại vi 8255 để mở rộng 48 ngõ vào ra, giao tiếp với bộ nhớ RAM 62256 có dung lượng 32kbyte, và 8279 chuyên dùng,
  - Giao tiếp LCD 16 x2.
  - Trong bộ thí nghiệm có cả bộ nạp cho các vi điều khiển trên và nạp dữ liệu cho các loại bộ nhớ EPROM họ 2716, 2732, 2764, 27128, 27256, 27512 và bộ nhớ EEPROM 2816, 2864.
  - Nguồn cung cấp cho hệ thống.
  - Cáp kết nối với cổng COM.
  - Các bus dây để kết nối điều khiển.

Bộ thí nghiệm loại nhỏ hình 8-2 có cấu hình như sau:

- ✓ Bộ thí nghiệm sử dụng các loại vi điều khiển 89S52, 89S51, 89S8252
- ✓ Các modul thí nghiệm gồm có:
  - Led ma trận 8x8 hai màu xanh và đỏ.
  - Có 32 led đơn 3mm tích cực mức 0.
  - Led 7 đoạn gồm 8 Led kết nối theo phương pháp quét.
  - Bàn phím 16 phím tổ chức theo ma trận.
  - Mạch giao tiếp điều khiển động cơ bước .
  - Giao tiếp LCD 16 x2.
  - Nguồn cung cấp cho hệ thống.
  - Cáp kết nối với cổng COM.
  - Các bus dây để kết nối điều khiển.



Hình 8-1. Bộ thí nghiệm vi điều khiển loại lớn.



Hình 8-2. Bộ thí nghiệm vi điều khiển loại nhỏ.

II. CẤU HÌNH BỘ THÍ NGHIỆM LOẠI LỚN:

Bộ thí nghiệm lớn có thể sử dụng được nhiều loại vi điều khiển 89C51, 89C52, 89S51, 89S52, 89S8252, 89C1051, 89C2051 và 89C4051. Bộ thí nghiệm có thể nạp chương trình cho nhiều loại vi điều khiển như vừa nêu ra ở trên.

Đặc biệt là vi điều khiển 89S51, 89S52 và 89S8252 có thể nạp chương trình ngay trong hệ thống đang chạy – điều này tiết kiệm cho bạn không phải mất nhiều thời gian trong quá trình gắn IC vào bo nạp rồi sau khi nạp xong lại IC gắn vào bo chạy nếu không đúng phải làm đi làm lại nhiều lần đối với họ 89C – dĩ nhiên 89C vẫn có thể làm được nhưng mạch điện khá phức tạp. Điều tiện lợi thứ 2 là đối với 89C bạn tháo gắn IC trên các socket nạp và nếu bạn gắn ngược thì có thể làm hỏng IC, còn 89S thì do không cần tháo gắn nên điều này sẽ không xảy ra.

Chương trình sử dụng cho bộ thí nghiệm này là SPKT_C rất dễ sử dụng, cho phép bạn soạn thảo và biên dịch chương trình một cách nhanh chóng, dễ dàng tìm ra lỗi trong chương trình.

Các phần tiếp theo sẽ trình bày chi tiết cấu hình bộ thí nghiệm, cách sử dụng chương trình và chương 9 sẽ cung cấp các bài thí nghiệm thực hành có thể phục vụ cho các bạn tự thực hành.

Trước khi thực hiện các bài thực hành thì bạn phải tìm hiểu hệ thống bộ thí nghiệm. Bộ thí nghiệm gồm 2 hệ thống:

- **Hệ thống I**: được thiết kế để sử dụng bộ nhớ nội của vi điều khiển - các chương trình điều khiển lưu trong bộ nhớ nội.

- **Hệ thống II**: được thiết kế để sử dụng khả năng mở rộng bộ nhớ ngoại và giao tiếp với các ngoại vi.

- Các ứng dụng giao tiếp với các hệ thống I và II.

Sau đây sẽ trình bày các mạch điện của tất cả các hệ thống và trình tự điều khiển.

Phần 1: Hệ thống vi điều khiển dùng bộ nhớ trong.

Vi điều khiển 89C51 có 4 port (từ port 0 đến port 3) được giao tiếp với 4 IC đệm 74245 hai chiều có điều khiển chiều bằng 4 switch. Ngõ vào / ra của các IC đệm được nối với các pinheader. Sơ đồ nguyên lý như hình 8-3 và hình bố trí linh kiện trong bộ thí nghiệp như hình 8-4.

Trong hình 8-3 có 1 socket để gắn vi điều khiển vào thực hiện các thí nghiệm, chiều gắn IC vào giống như hình trên. Có 6 jumper [JP] để giao tiếp tín hiệu, 4 JP cho 4 port 0, 1, 2,3 đã qua 4 IC đệm 74245 và hai JP kết nối với port1 và port3 – chưa qua đệm dùng để kết nối với các thiết bị vào ra [IO] tùy ý.

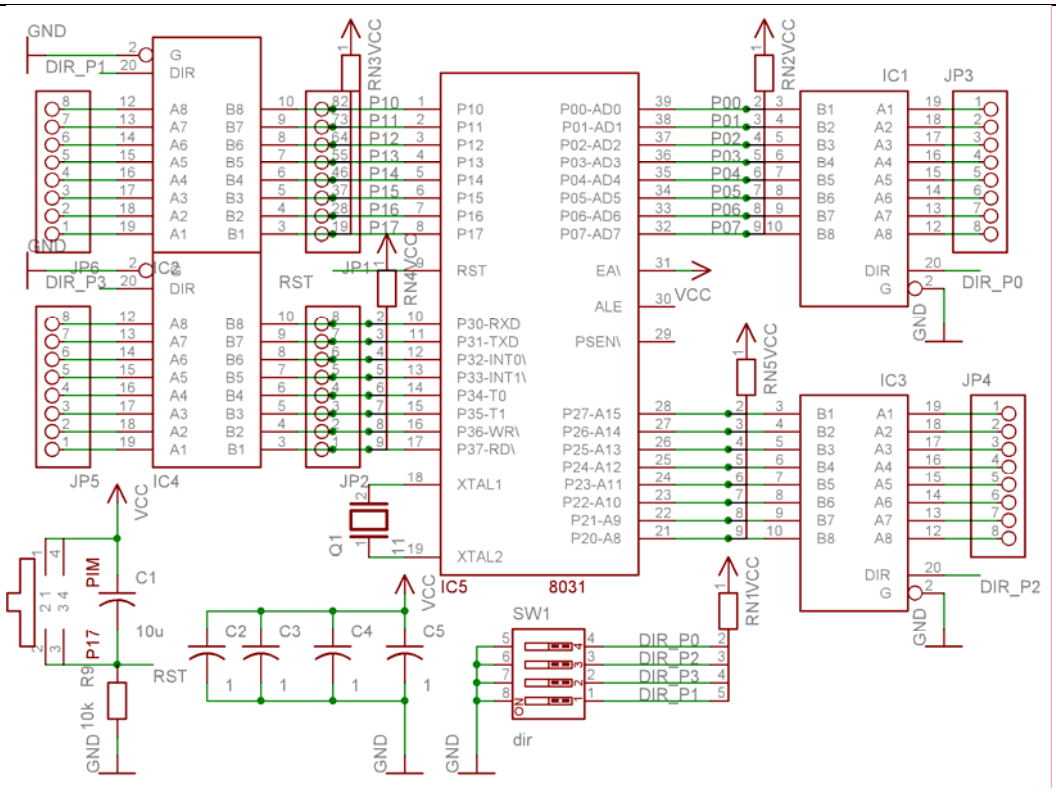
4 IC đệm 2 chiều chỉ có thể hoạt động 1 trong 2 chiều bằng cách chuyển 4 SW1, 2, 3, 4 trong SW11 tương ứng với 4 port 0, 1, 2, 3.

SW ở vị trí ON thì tín hiệu từ vi điều khiển đưa đến IC – đệm – đưa ra ngoài.

SW ở vị trí OFF thì tín hiệu từ bên ngoài đưa đến IC – đệm – đưa vào vi điều khiển.

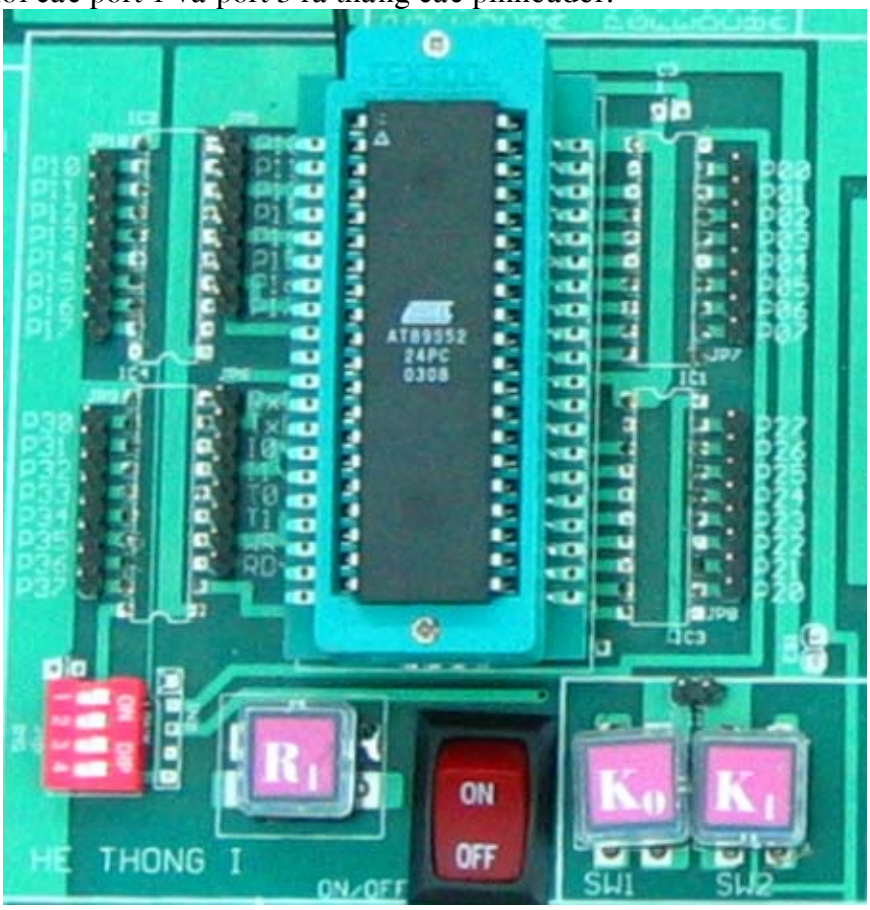
Nút nhấn reset R1 để reset vi điều khiển bắt đầu thực hiện lại từ đầu.

Contact ON/OFF dùng để tắt mở nguồn cung cấp cho hệ thống. Trước khi gắn IC vào hoặc lấy IC ra khỏi đế cài thì nên tắt nguồn.



Hình 8-3. Sơ đồ mạch của hệ thống.

Switch ở mức OFF thì 74245 đóng vai trò là xuất dữ liệu: có nghĩa là dữ liệu từ vi điều khiển gởi ra các port sẽ qua các IC đệm ra ngoài. Để tăng tính tích cực vào ra độc lập của các port một cách tự động ta nối các port 1 và port 3 ra thẳng các pinheader.



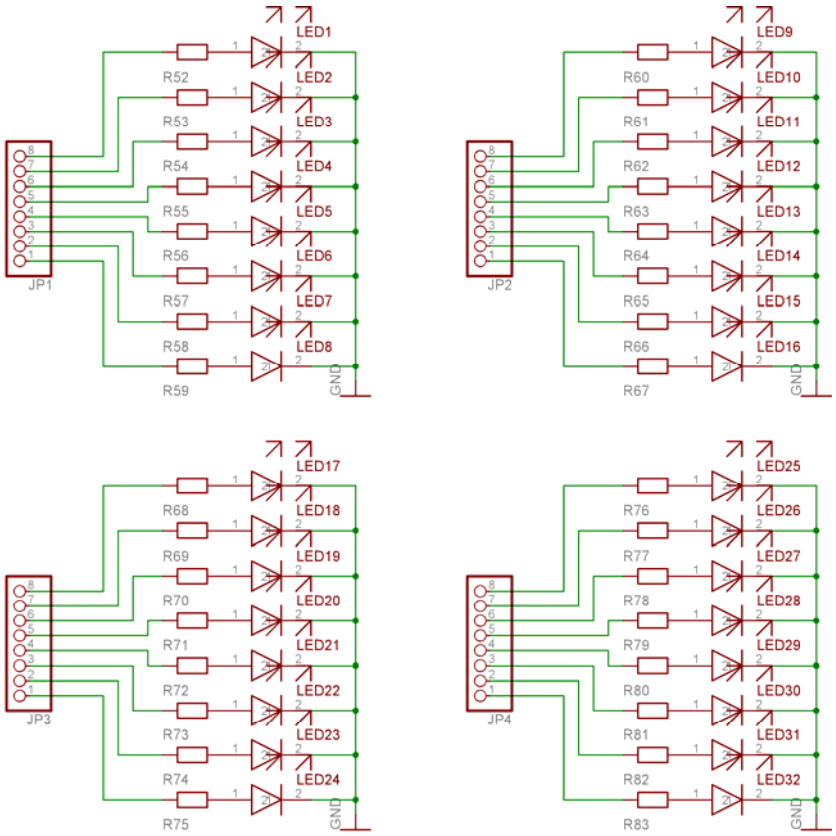
Hình 8-4. Hình chụp của hệ thống 1 – sử dụng bộ nhớ nội.

Ứng dụng 1: Ứng dụng giao tiếp với Led đơn

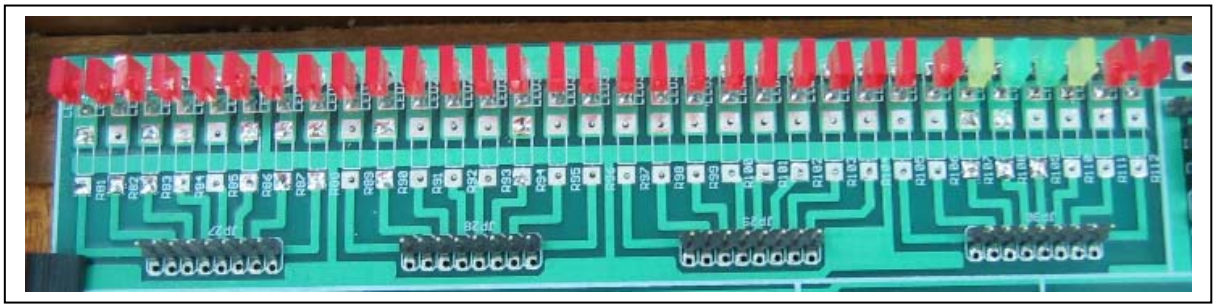
Một trong những ứng dụng đơn giản để sử dụng 4 port của vi điều khiển xuất dữ liệu làm quen với lập trình ta có một hệ thống 32 led đơn có kết nối với 4 pinheader 8 chân.

Khi muốn dùng port 1 để điều khiển 8 led thì chỉ cần dùng một bus dây 8 sợi kết nối 2 pinheader 8 chân từ hệ thống vi điều khiển đến hệ thống led đơn.

Sơ đồ mạch của 32 led đơn như hình 8-5 và sơ đồ bố trí linh kiện như hình 8-6:



Hình 8-5. Sơ đồ nguyên lý của 32 led.

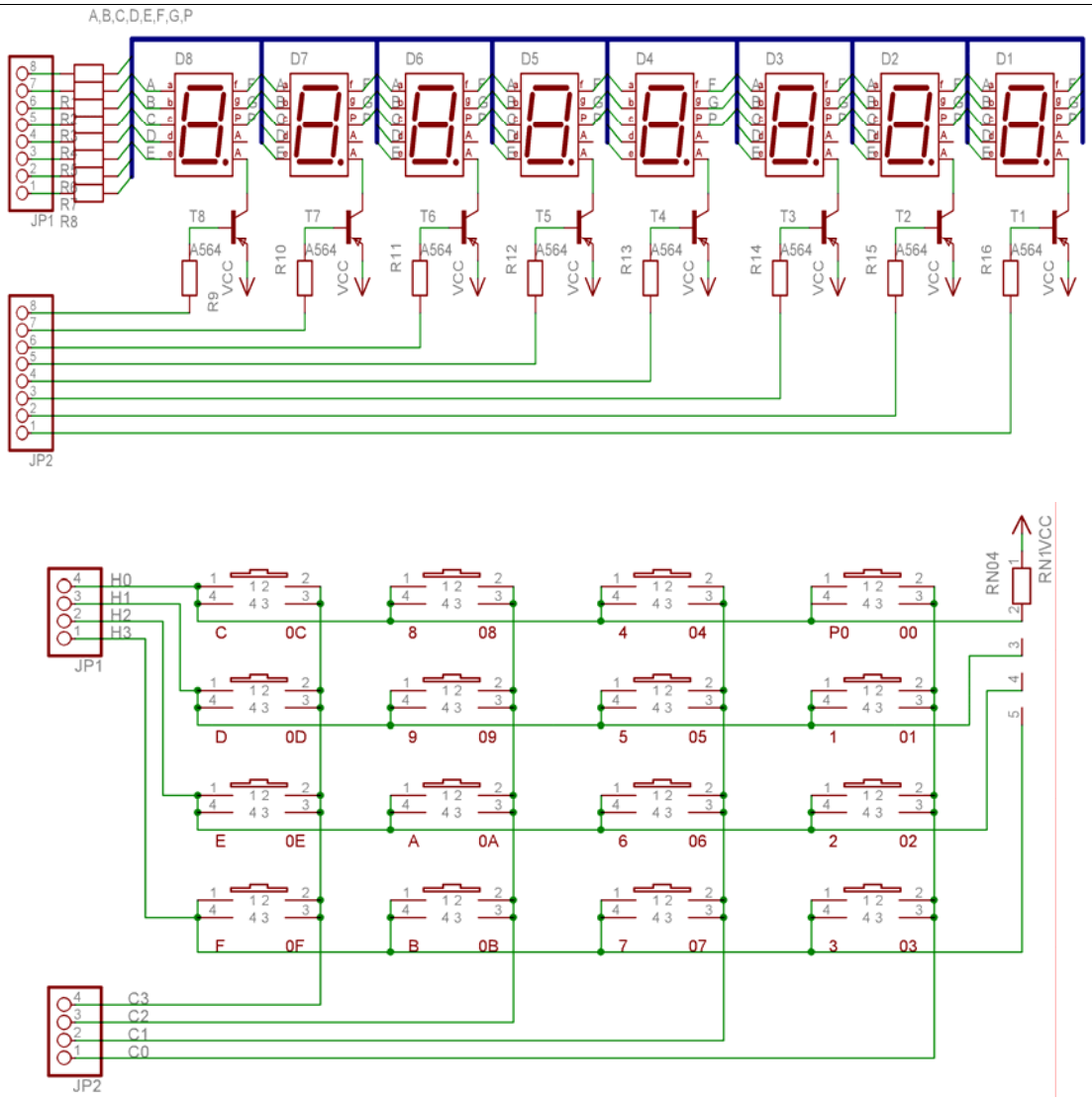


Hình 8-6. Sơ đồ linh kiện trên bộ thí nghiệm của 32 led đơn.

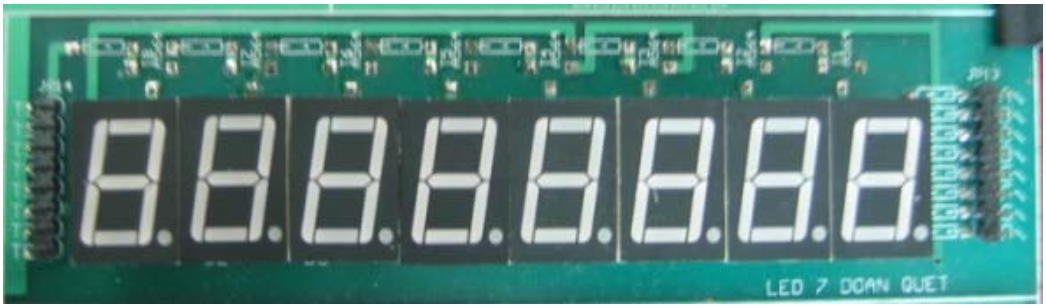
4 pinheader dùng để kết nối với 32 led, ngõ vào mức 1 thì led sáng , mức 0 led tắt.

Chức năng của khối hiển thị led đơn dùng để kết nối với 4 port của vi điều khiển thực hiện các chương trình điều khiển led làm quen với lập trình vi điều khiển.

Ứng dụng 2 : Ứng dụng giao tiếp với 8 Led 7 đoạn và bàn phím.



Hình 8-7. Sơ đồ giao tiếp với led 7 đoạn và bàn phím.



Hình 8-8. Sơ đồ linh kiện trên bộ thí nghiệm của 8 led 7 đoạn.

Trong các ứng dụng điều khiển, nhiều thông tin được nhập từ bàn phím cũng như các thông tin hiển thị trên các led 7 đoạn ví dụ như thiết lập nhiệt độ khống chế và đo nhiệt độ của một hệ thống.

Để làm quen với cách thức giao tiếp điều khiển led và bàn phím thì trong hệ thống này có xây dựng mạch điện giao tiếp với 8 led 7 đoạn loại anode chung theo phương pháp quét và bàn phím có 16 phím theo dạng ma trận. Sơ đồ nguyên lý của led 7 đoạn và ma trận phím như hình 8-5 và hình 8-6.

Để điều khiển 8 led 7 đoạn phải dùng 16 đường điều khiển: 8 đường điều khiển 7 đoạn và dấu chấm thập phân, 8 đường điều khiển đóng ngắt 8 transistor.

Tại mỗi một thời điểm ta chỉ cho 1 tranistor dẫn và 7 transistor còn lại tắt, dữ liệu gởi ra sẽ sáng trên led tương ứng với transistor dẫn. Sau đó cho 1 transistor khác dẫn và gởi dữ liệu hiển thị cho led đó, quá trình điều khiển này diễn ra lần lượt cho đến khi hết 8 led.

Với tốc độ gởi dữ liệu nhanh và do mắt ta có lưu ảnh nên ta nhìn thấy 8 led sáng cùng 1 lúc.

Mã quét: mức logic 0 thì transistor dẫn, mức logic 1 thì transistor ngắt.

MÃ HEX	**Mã quét điều khiển các transistor**								
FE	1	1	1	1	1	1	1	0	Transistor 1 ON
FD	1	1	1	1	1	1	0	1	Transistor 2 ON
FB	1	1	1	1	1	0	1	1	Transistor 3 ON
F7	1	1	1	1	0	1	1	1	Transistor 4 ON
EF	1	1	1	0	1	1	1	1	Transistor 5 ON
DF	1	1	0	1	1	1	1	1	Transistor 6 ON
BF	1	0	1	1	1	1	1	1	Transistor 7 ON
7F	0	1	1	1	1	1	1	1	Transistor 8 ON

Mã 7 đoạn: trong hệ thống sử dụng led 7 đoạn loại Anode chung nên mức logic 0 thì led sáng và mức logic 1 thì led tắt.

Số hex	**dp**	**g**	**f**	**e**	**d**	**c**	**b**	**a**	**Mã số hex**
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F1
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	0	0	0	0	90
A	1	0	0	0	1	0	0	0	88
B	1	0	0	0	0	0	1	1	83

C	1	1	0	0	0	0	1	0	C2
D	1	0	1	0	0	0	0	1	A1
E	1	0	0	0	0	1	1	0	86
F	1	0	0	0	1	1	1	0	8E

Các mã khác bạn có thể tự thiết lập.

Các transistor và các điện trở gắn bên dưới bo mạch nên bạn sẽ không nhìn thấy trong hình.

Với bàn phím ta sử dụng 1 port nào đó tùy ý chẳng hạn như port 1.

Để điều khiển quét phím thì ta xuất 1 dữ liệu 4 bit: trong đó có 1 bit ở mức thấp và 3 bit ở mức cao ra 4 đường điều khiển quét của bàn phím.

Sau đó ta kiểm tra mức logic của 4 ngõ nhập để xem có phím nào nhấn hay không:

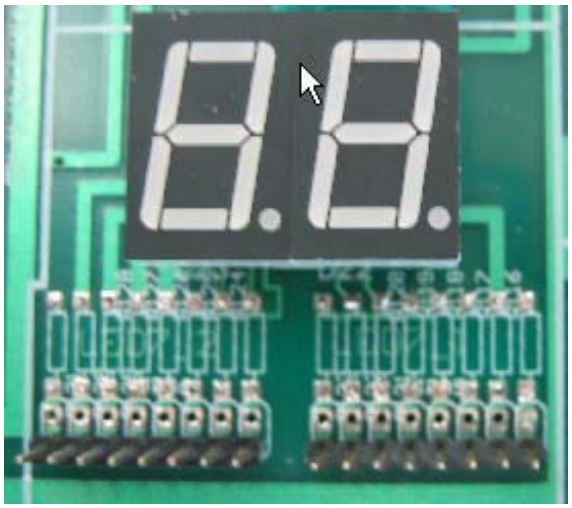
- Nếu có phím nhấn thì 4 bit nhập sẽ có 1 bit ở mức loigc 0 và tiến hành thiết lập mã phím.
- Nếu không có phím nhấn thì 4 bit nhập sẽ ở mức logic 1 – khi đó ta chuyển mức logic 0 sang bit quét kế để dò tìm phím khác.

Với led 7 đoạn thì có thể cho phép hiển thị chữ và số - khi đó có rất nhiều chương trình ứng dụng có thể thực hiện được trên hệ thống này như chương trình đếm sản phẩm, chương trình đếm tần số, chương trình đồng hồ số, chương trình đồng hồ thể thao ...

Ứng dụng 3 : Ứng dụng giao tiếp trực tiếp với 2 Led 7 đoạn.

Ở trên đã trình bày giao tiếp với 8 led 7 đoạn theo phương pháp quét tuy nhiên có nhiều ứng dụng trong thực tế ta chỉ sử dụng 1 một 2 led thì nếu sử dụng hệ thống trên khá phức tạp và đối với người bắt đầu nghiên cứu vi điều khiển thì việc viết chương trình sẽ khá phức tạp nên trong hệ thống thí nghiệm này có thiết kế thêm phần giao tiếp với 2 led 7 đoạn loại anode chung để tiện cho việc sử dụng.

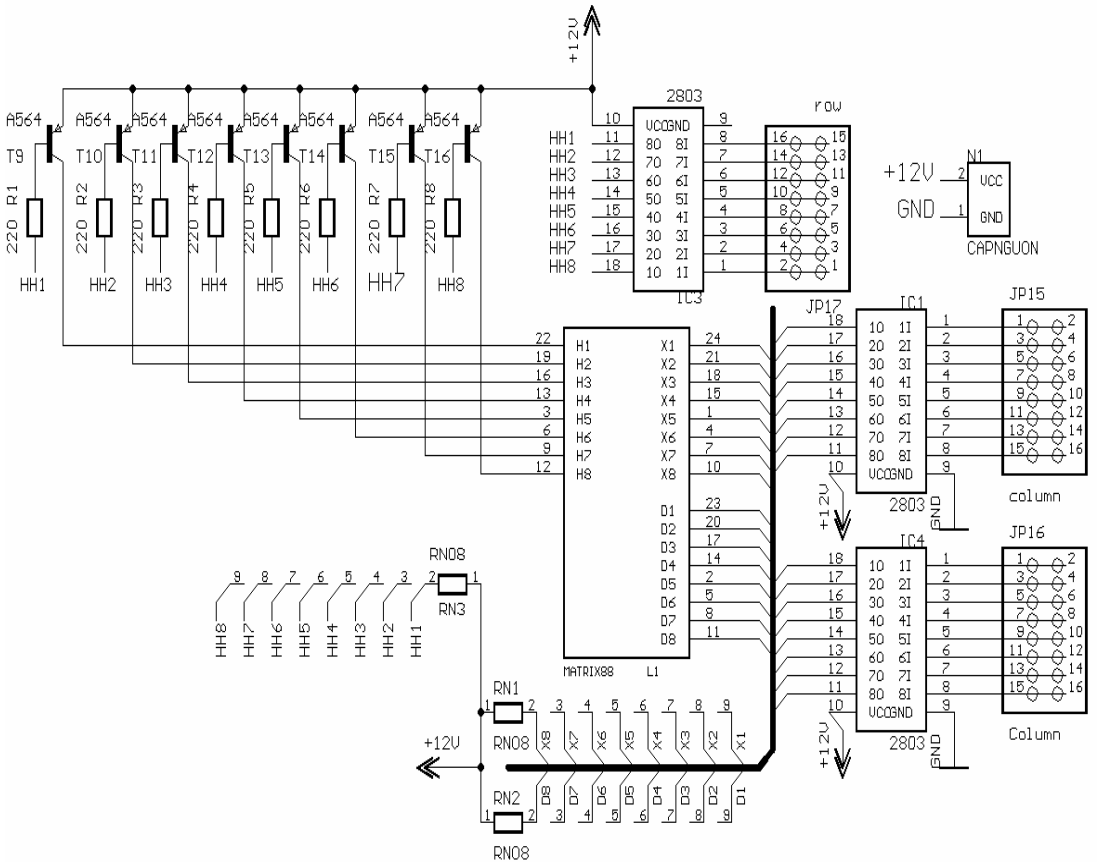
Nên nhớ rằng nếu sử dụng nhiều led thì phương phá quét ở trên là tối ưu. Sơ đồ mạch giao tiếp của 2 led đã qua điện trở hạn dòng và có 2 pinheader để nhận tín hiệu như hình 8-9.



Hình 8-9. Sơ đồ linh kiện trên bộ thí nghiệm của 2 led 7 đoạn.

Ứng dụng 4 : Ứng dụng giao tiếp với led ma trận Led 8x8

Một trong những ứng dụng phổ biến trong quảng cáo là thông tin được hiển thị trên led ma trận, để giúp người học hiểu được nguyên lý điều khiển led ma trận như thế nào thì trong hệ thống có thiết kế giao tiếp với 1 led ma trận 8x8 hai màu xanh và đỏ. Sơ đồ nguyên lý trình bày ở hình 8-10.



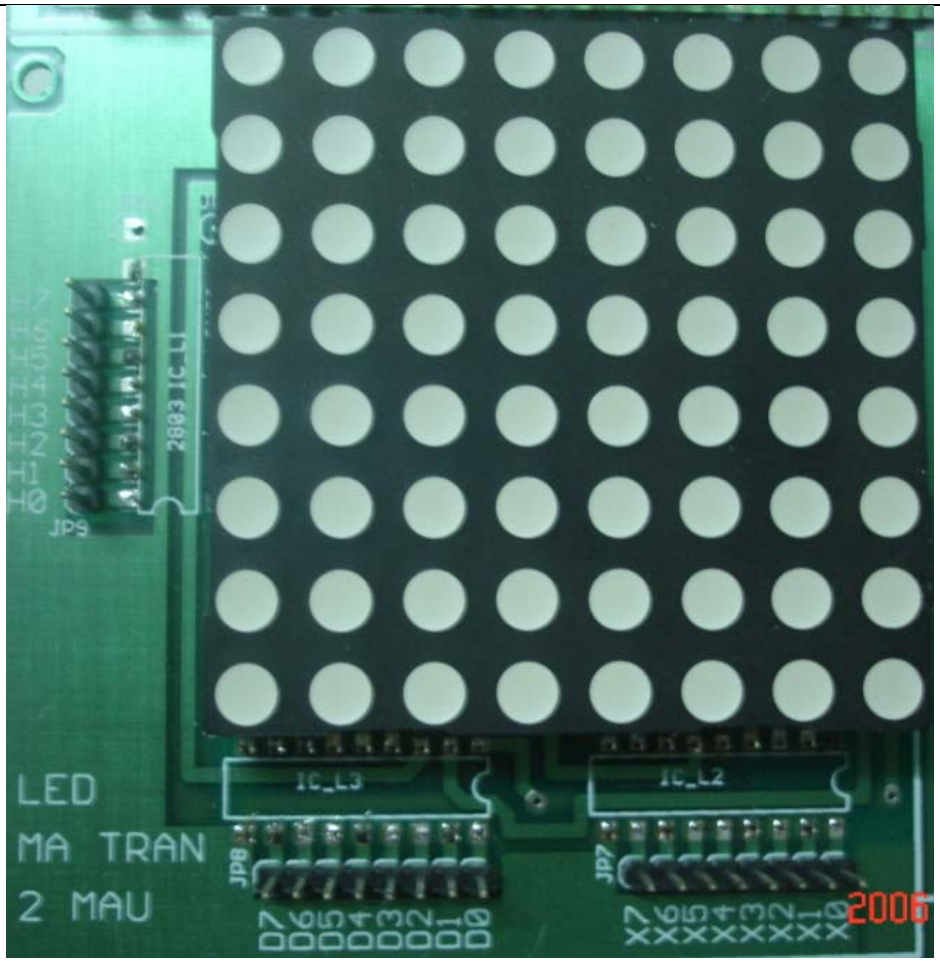
Hình 8-10. Sơ đồ giao tiếp với ma trận led 8x8.

Các hàng và cột được đưa qua IC đệm 2803. Led ma trận có 2 màu xanh và đỏ có 8 đường điều khiển hàng, 8 đường điều khiển cột màu xanh và 8 đường điều khiển cột màu đỏ – tổng cộng là 24 đường được kết nối với 3 pinheader.

Khi muốn điều khiển led ma trận ví dụ ta dùng hệ thống vi điều khiển thì trình tự kết nối mạch như sau: dùng một port nào đó tùy ý như port 3 điều khiển hàng, dùng port 2 hoặc port 0 để điều khiển cột xanh hoặc cột đỏ nếu muốn hiển thị 1 màu, còn muốn hiển thị 2 màu thì ta dùng cả 2 port.

Với phần cứng đã thiết kế ở trên sử dụng led ma trận 8x8 có 2 màu xanh và đỏ, để điều khiển led ma trận sáng ta tiến hành gởi dữ liệu ra hàng và mã quét ra cột. Trong 4 port của hệ thống 1 ta sử dụng port 3 làm port điều khiển hàng và port 2 điều khiển cột xanh hoặc đỏ.

Sơ đồ của ma trận led 8x8 trong bộ thí nghiệm như hình 8-11.

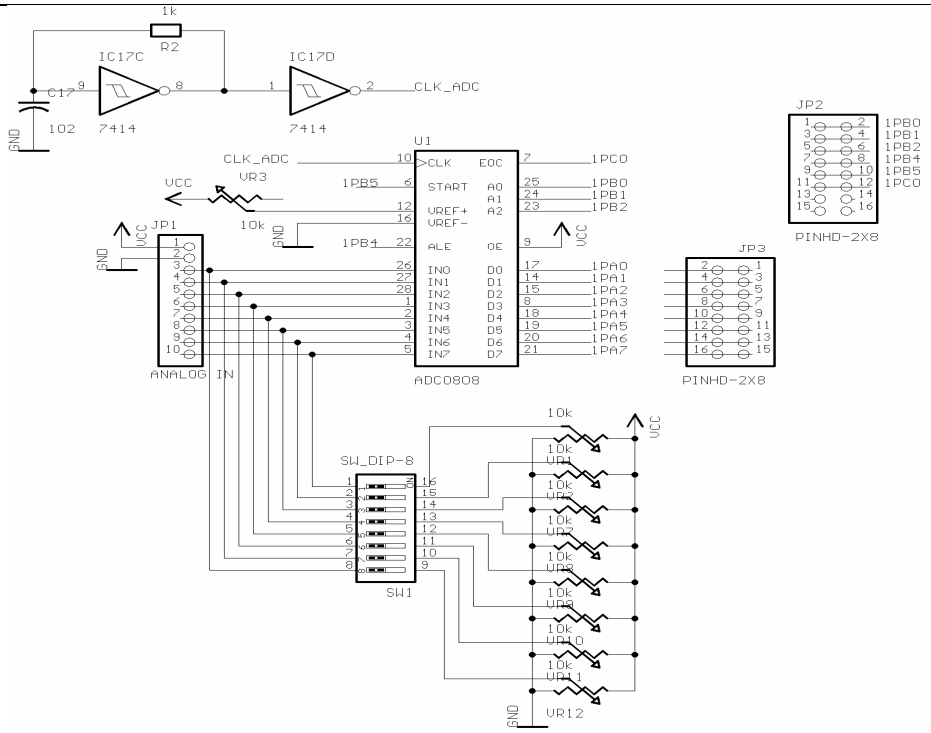


Hình 8-11. Sơ đồ bố trí linh kiện của mạch điều khiển led ma trận.

Pinheader điều khiển hàng bên trái và 2 pinheader điều khiển cột xanh và đỏ bên dưới.

Ứng dụng 5 : Ứng dụng giao tiếp với ADC 0809 và DAC0808.

Hệ thống ADC: trên bộ thí nghiệm có thiết kế mạch giao tiếp với một IC ADC 0809. Sơ đồ nguyên lý như hình 8-12 và sơ đồ bố trí linh kiện trên hệ thống như hình 8-13.



Hình 8-12. Sơ đồ giao tiếp ADC 0809.

Trong sơ đồ trên sử dụng ADC0809 có thể chuyển đổi 8 kênh dữ liệu ngõ vào tương tự. Các đường tín hiệu điều khiển bao gồm:

- 8 đường dữ liệu số truyền tải kết quả chuyển đổi từ tương tự sang số: D7- D0 – tín hiệu ra.
- 3 đường tín hiệu điều khiển chọn kênh: CBA – tín hiệu vào.
- 2 đường tín hiệu điều khiển bắt đầu quá trình chuyển đổi Start, ALE – tín hiệu vào.
- 1 đường tín hiệu điều khiển EOC báo cho biết quá trình chuyển đổi kết thúc hay chưa – tín hiệu ra.

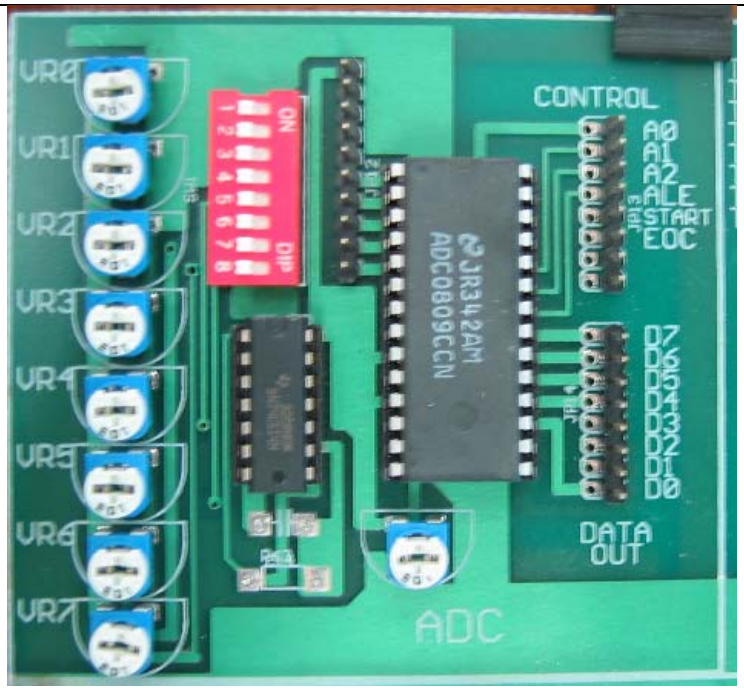
Tần số xung clock để ADC chuyển đổi được cung cấp từ mạch dao động RC có tần số được thiết kế khoảng 500KHz theo sổ tay tra cứu IC.

Khi sử dụng giao tiếp với ADC bằng vi điều khiển thì phải sử dụng port 1 và port 3 ***chưa đệm*** để giao tiếp: port 1 chưa đệm để nhận dữ liệu vào và port 3 chưa đệm để vừa xuất tín hiệu điều khiển vừa nhận tín hiệu báo kết thúc.

Trong sơ đồ nguyên lý để thực hiện việc chuyển đổi thì ngõ vào có giả lập tín hiệu analog bằng 8 biến trở cho 8 kênh, đồng thời 8 switch [SW ADC] phải ở vị trí ON để đưa tín hiệu vào mạch ADC.

Khi muốn giao tiếp với nguồn tín hiệu tương tự từ bên ngoài thì chuyển SW ADC về vị trí OFF để hở mạch, các nguồn tín hiệu từ bên ngoài có thể đưa đến JP đơn nằm cạnh biến trở. Trong JP này có cả nguồn và mass 0V.

Biến trở VR3 dùng để chỉnh độ phân giải Step Size cho khối ADC.



Hình 8-13. Sơ đồ linh kiện trên bộ thí nghiệm của ADC 0809 – 8 kênh.

Các thí nghiệm sử dụng hệ thống I điều khiển quá trình chuyển đổi ADC 0809, phải sử dụng 2 port để giao tiếp: dùng port 1 nhập dữ liệu từ ADC vào; dùng port 3 điều khiển chọn kênh, Start, ALE của ADC.

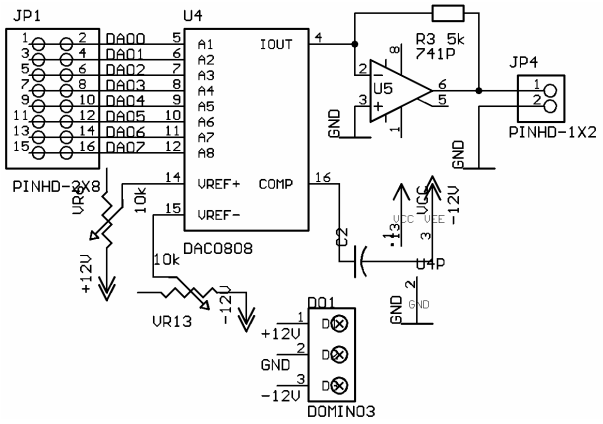
Kết quả chuyển đổi đọc về lưu vào bộ nhớ trong vi điều khiển cần phải hiển thị ra led để thấy nên phải giao tiếp với led 7 đoạn ta dùng port 0 điều khiển các đoạn a, b, c, d, e, f, g và port 2 điều khiển quét 8 led.

Ứng dụng 6 : Ứng dụng giao tiếp với DAC0808.

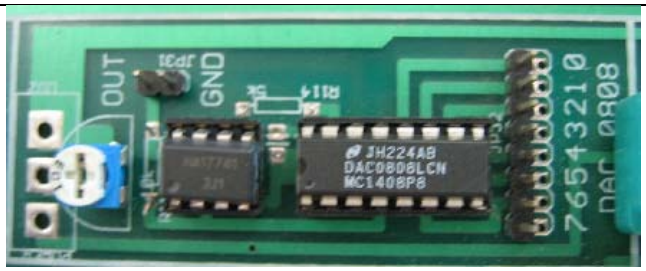
Sơ đồ nguyên lý của hệ thống DAC như hình 8-14 và sơ đồ bố trí linh kiện như hình 8-15.

Trong sơ đồ sử dụng DAC0808 có chức năng chuyển đổi tín hiệu số 8 bit sang tín hiệu tương tự, 8 đường D7 – D0 dùng để nhận dữ liệu số từ hệ thống điều khiển. Ngõ ra của DAC 0808 là tín hiệu tương tự được đưa qua IC khuếch đại đệm 741, sau đó đưa đến JP out [chốt trên là tín hiệu ra, chốt dưới là 0V].

Độ phân giải của DAC 0808 được điều chỉnh bằng biến trở.



Hình 8-14. Sơ đồ giao tiếp DAC 0808.



Hình 8-15. Sơ đồ linh kiện DAC 0808.

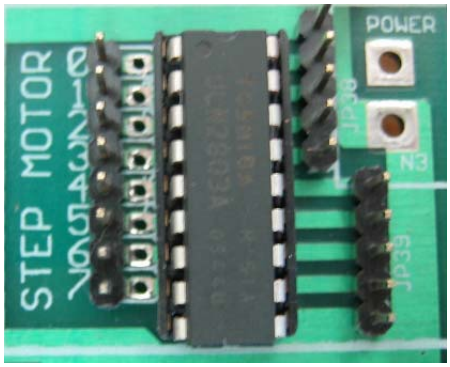
Pinheader 8 chân để nhận tín hiệu số vào và pinheader 2 chân để xuất tín hiệu tương tự ra.

Ứng dụng 7 : **Ứng dụng giao tiếp trực tiếp động cơ bước công suất nhỏ.**

Có nhiều loại động cơ bước nhưng trong hệ thống này chỉ sử dụng động cơ bước loại nhỏ, mạch điện giao tiếp để điều khiển động cơ bước sử dụng IC giao tiếp 2803 và có thể giao tiếp với 2 động cơ bước.

Động cơ 7 bước sử dụng là loại có 4 cuộn dây – có 5 đầu dây ra hoặc 6 đầu dây ra. Nếu là loại 5 đầu thì có 1 đầu dây chung và 4 đầu dây còn lại sẽ nhận tín hiệu điều khiển, còn loại 6 đầu dây ra thì trong đó sẽ có 2 đầu dây chung nên nối lại thành 1 đầu dây chung.

Sơ đồ bố trí linh kiện của IC 2803 để giao tiếp với động cơ bước như hình 8-16.



Hình 8-16. Sơ đồ linh kiện của IC 2803 giao tiếp với động cơ bước.

Mã điều khiển động cơ bước hãy xem trong chương trình điều khiển.

Ứng dụng 8: **giao tiếp với hệ thống 8 led đơn.**

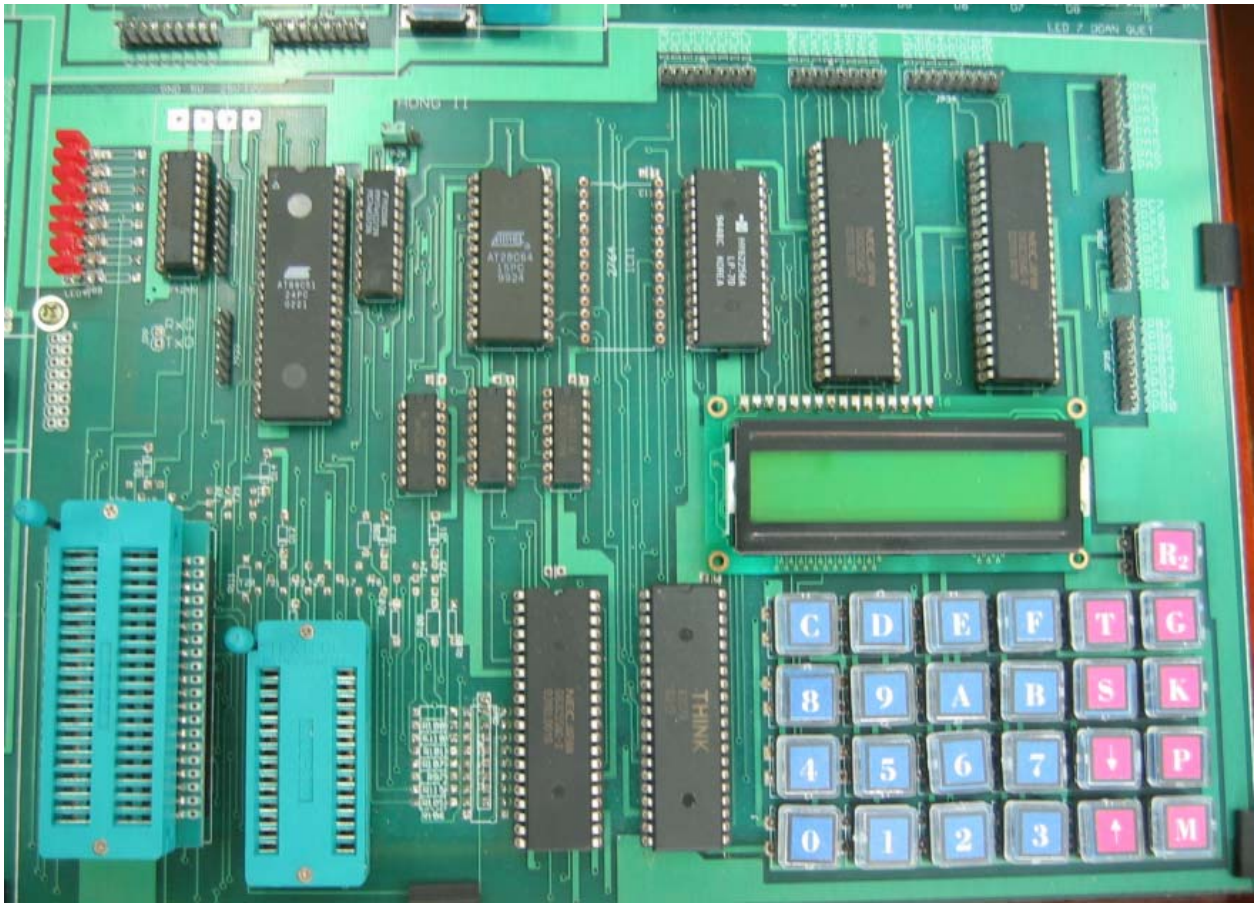
Ngoài 32 led đơn đã trình bày thì bộ thí nghiệm còn giao tiếp với 8 led đơn loại 10 ly như hình 8-17 để làm sinh động thêm các ứng dụng như hệ thống điều khiển đèn giao thông.

Hình 8-17. Giao tiếp với 8 led 10 ly.

Phần 2 :Hệ thống vi điều khiển dùng bộ nhớ ngoài – hệ thống II.

Đối với các ứng dụng nhỏ – đơn giản thì chỉ sử dụng 1 Vi điều khiển 89C51 có thể đáp ứng được các yêu cầu điều khiển, nhưng nếu số lượng đường tín hiệu điều khiển vào ra nhiều hơn 4 port và dung lượng chương trình lớn hơn thì 1 vi điều khiển không đáp ứng được. Khi đó phải sử dụng bộ nhớ bên ngoài để tăng dung lượng bộ nhớ lớn hơn và mở rộng giao tiếp nhiều đường vào ra hơn.

Cấu trúc hệ thống II như hình 8-18.



Hình 8-16. Hệ thống II giao tiếp với bộ nhớ ngoài.

Một vấn đề cần phải quan tâm hơn nữa là khi sử dụng bộ nhớ bên trong mỗi lần thay đổi chương trình thì phải xóa và nạp lại, đối với hệ thống sử dụng bộ nhớ bên ngoài thì ta có thể nạp chương trình vào bộ nhớ RAM chạy thử và hiệu chỉnh cho đến khi hoàn chỉnh rất tiện dụng cho việc nghiên cứu và thực hành, thí nghiệm.

Trong hệ thống có sử dụng 2 IC nhớ Eprom 2746 có dung lượng tổng cộng là 16 kByte chiếm vùng nhớ có địa chỉ:

- Từ 0000H - 1FFFH cho eprom 1: lưu trữ chương trình hệ thống và nếu sử dụng các vi điều khiển thế hệ mới sau này thì có thể sử dụng bộ nhớ nội bên trong để lưu chương trình và Eprom này sẽ không có trong hệ thống.

- Từ 2000H - 3FFFH cho eprom 2: ở dạng đế cấm để lưu trữ các chương trình ứng dụng hoặc sao chép eprom.

Bộ nhớ RAM sử dụng RAM 62256 có dung lượng 32Kbyte chiếm địa chỉ từ 8000H đến FFFFH. Vùng nhớ này dùng để lưu trữ dữ liệu cho vi điều khiển xử lý hoặc lưu trữ chương trình từ máy tính gởi xuống để chạy thử.

Hệ thống giao tiếp bàn phím 24 phím dạng ma trận do IC 8279 phụ trách. Địa chỉ của IC 8279 được thiết kế trong bản đồ nhớ là 4020H và 4021H.

- Địa chỉ 4020H là địa chỉ để đọc mã phím.
- Địa chỉ 4021H là địa chỉ dùng để gởi từ điều khiển.

Hệ thống sử dụng LCD để hiển thị thông tin và phục vụ cho lập trình bằng tay nhập mã lệnh dưới dạng số hex hoặc hiển thị các thông tin liên quan đến lập trình bằng máy tính.

Để mở rộng thêm số lượng tín hiệu điều khiển trong hệ thống có giao tiếp với 2 IC 8255 kết quả được 48 đường điều khiển IO.

IC 8255 – 1 giao tiếp với hệ thống tại địa chỉ 4000H – 4003H. Trong đó:

- Địa chỉ 4000H dùng để truy xuất portA.
- Địa chỉ 4001H dùng để truy xuất portB.
- Địa chỉ 4002H dùng để truy xuất portC.
- Địa chỉ 4003H dùng để truy xuất thanh ghi điều khiển.
- IC này đã sử dụng port a và 3 bit thấp của port C để điều khiển LCD.

Chú ý không nên sử dụng các đường điều khiển của port C vì đã phục vụ cho điều khiển LCD.

IC 8255 – 2 giao tiếp với hệ thống tại địa chỉ 4010H – 4013H. Trong đó:

- Địa chỉ 4010H dùng để truy xuất portA.
- Địa chỉ 4011H dùng để truy xuất portB.
- Địa chỉ 4012H dùng để truy xuất portC.
- Địa chỉ 4013H dùng để truy xuất thanh ghi điều khiển.

Tổng cộng có 6 port – 48 đường điều khiển được kết nối với 6 pinheader.

Trong hệ thống có thiết kế mạch nạp bộ nhớ EPROM, bộ nhớ bên trong vi điều khiển họ MCS-51, 52 thông qua IC điều khiển 8255-3, IC này được thiết kế tại địa chỉ 6000H đến 6003H, chỉ phục vụ cho việc nạp bộ nhớ không được sử dụng cho các chức năng khác.

Trên bộ thí nghiệm có 2 đế nạp: đế 40 chân dùng để nạp bộ nhớ của vi điều khiển. Đế 28 chân dùng để nạp bộ nhớ EPROM từ 2732 đến 27512. Để nạp bộ nhớ của vi điều khiển loại 20 chân 89Cx051 ta sử dụng đế 40 chân gắn IC vào bắt đầu từ chân số 1.

Sơ đồ các đế nạp chương trình cho vi điều khiển và bộ nhớ như hình 8-17.

Hình 8-17. Sơ đồ 2 socket nạp chương trình và dữ liệu cho vi điều khiển và bộ nhớ.

Khi nạp chương trình cho vi điều khiển 40 chân thì hãy gắn vào đế 40 chân với chiều như trong hình 8-18.

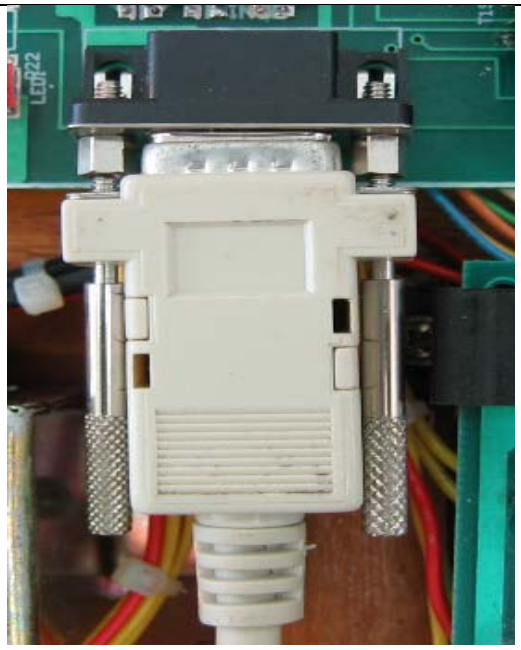
Khi muốn nạp vi điều khiển 20 chân thì hãy gắn vào đế 40 chân bắt đầu từ trên xuống như hình 8-19.

Khi muốn nạp cho bộ nhớ Eprom thì gắn IC nhớ như trong hình 8-20.

Hình 8-18. Gắn vi điều khiển 89C51 hoặc 89C52 hoặc 89S51hoặc 89S52 vào để nạp chương trình.

Hình 8-19. Gắn vi điều khiển 89C1051 hoặc89C2051 hoặc 89C4051 vào để nạp chương trình.

Hình 8-20. Gắn bộ nhớ họ 27 hoặc 28 vào để nạp.



Hình 8-21. Bộ thí nghiệm giao tiếp với máy tính bằng cổng COM.

Các đế 40 chân tần số sử dụng cao nên nhanh hư và hệ thống đã thiết kế có thể thay thế dễ dàng.

Trong hệ thống có giao tiếp nối tiếp với máy tính thông qua cổng COM, vi mạch chuyển đổi mức logic dùng MAX 232 như hình 8-21.

Chương trình hệ thống điều khiển được lưu trong bộ nhớ EPROM.

III. CẤU HÌNH BỘ THÍ NGHIỆM LOẠI NHỎ:

Bộ thí nghiệm nhỏ chỉ sử dụng được nhiều loại vi điều khiển 89S51, 89S52, 89S8252. Bộ thí nghiệm có thể nạp chương trình cho các loại vi điều khiển như vừa nêu ra ở trên và có thể nạp chương trình ngay trong hệ thống đang chạy ISP, tiện lợi của kiểu ISP chỉ có đối với vi điều khiển họ 89S.

Cấu hình của bộ thí nghiệm loại nhỏ là rút gọn của bộ thí nghiệm lớn và các ứng dụng có đặc tính giống như hệ thống bộ thí nghiệm lớn gồm: 32 led đơn, 8 led 7 đoạn, 16 phím ma trận, led ma trận 8x8 hai màu xanh đỏ, LCD 16x2, và giao tiếp với IC 2803 để điều khiển động cơ bước loại nhỏ. 32 led đơn tích cực mức 0, hệ thống không dùng IC đệm.

IV. HƯỚNG DẪN SỬ DỤNG PHẦM MỀM SPKT_C:

1. Cài đặt chương trình:

Các thiết bị của chúng tôi là các bộ nạp hay các bộ thí nghiệm đều sử dụng chung 1 chương trình điều khiển SPKT_C với các version đã được cập nhật ngày càng nhiều tính năng mới.

Chương trình điều khiển bộ nạp không cần cài đặt, bạn chỉ cần copy tất cả các chương trình có trong đĩa CD vào thư mục đĩa C hoặc đĩa nào tùy ý trong máy tính của bạn.

Khi ghi dữ liệu lên CD thì thuộc tính các file dữ liệu là “read only”, nếu bạn chép bằng phần mềm Windows commander thì các thuộc tính này tự động chuyển sang thuộc tính “archive” và bạn không phải thực hiện thêm thao tác nào nữa.

Nhưng nếu bạn copy bằng các chương trình khác thì thuộc tính này vẫn còn và bạn phải tự bỏ bằng cách thực hiện theo trình tự như sau:

Bước 1: vào "My computer" chọn ổ đĩa và thư mục lưu các files đã copy từ đĩa CD sang. Thư mục này được chọn bằng cách trỏ chuột đến và bấm nút trái chuột, thư mục được chọn sẽ ở trạng thái tích cực.

Bước 2: sau khi chọn xong hãy bấm nút phải chuột thì 1 menu con xuất hiện. Bạn hãy chọn lệnh "properties" ở bên dưới và 1 menu mới sẽ xuất hiện. Phía bên dưới có 3 ô để chọn thuộc tính là

- ô "read only".

- ô "hidden"

- ô "archive"

Bạn hãy chọn ô "archive" và chọn "apply" và nhấn ok là xong.

Sau khi bạn bỏ xong bạn có thể kiểm tra lại bằng cách vào thư mục và chọn một tập tin nào đó và xem thuộc tính của nó khi đó nó sẽ ở trạng thái "archive".

Chú ý bạn chỉ thao tác được chương trình khi bạn đã bỏ hết các thuộc tính này. Nếu chưa bỏ thì khi bạn truy xuất các chương trình thì sẽ có thông báo không cho phép truy xuất "files not access" và bộ nạp sẽ bị treo vì nó không còn ở trạng thái sẵn sàng nữa và bạn phải reset lại bộ nạp bằng cách nhấn nút reset hoặc tắt và mở lại nguồn cung cấp cho bộ nạp.

Bước 3: Hãy tạo một thư mục "C:\tam" trên đĩa C của máy tính.

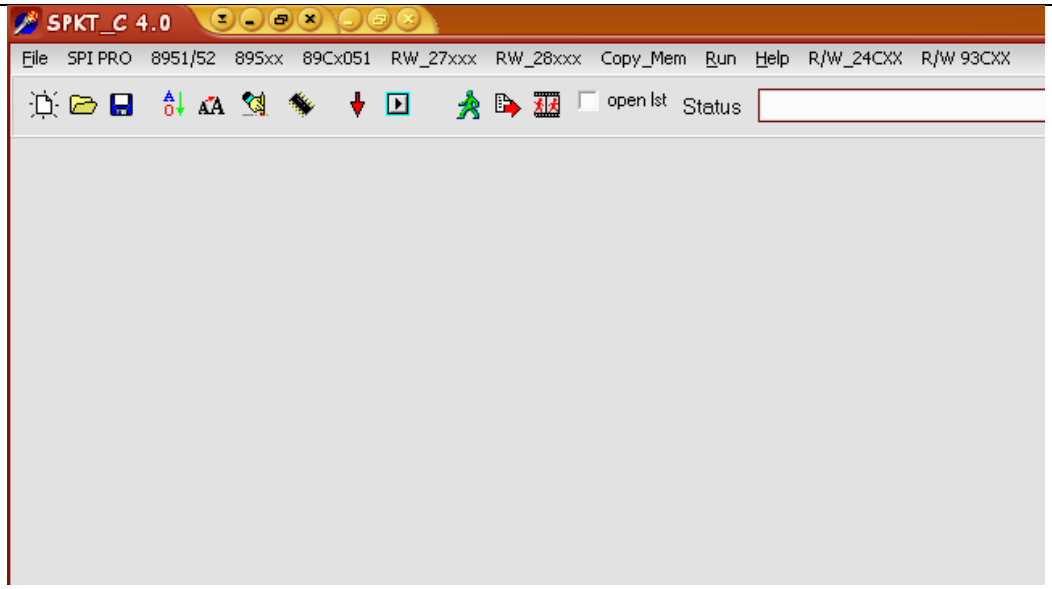
2. Cách sử dụng chương trình:

Sau khi copy xong bạn nên tạo một biểu tượng cho chương trình để truy xuất nhanh. Kích vào biểu tượng để chạy chương trình. (nếu bạn chưa biết tạo biểu tượng thì hãy xem sách hướng dẫn sử dụng windows hoặc hỏi người nào biết chỉ cho bạn sử dụng hoặc xem phần hướng dẫn phía cuối tài liệu này).

Trong chương trình điều khiển này cho phép các bạn soạn thảo chương trình cho vi điều khiển khiển và cho phép biên dịch và kiểm tra lỗi rất dễ sử dụng.

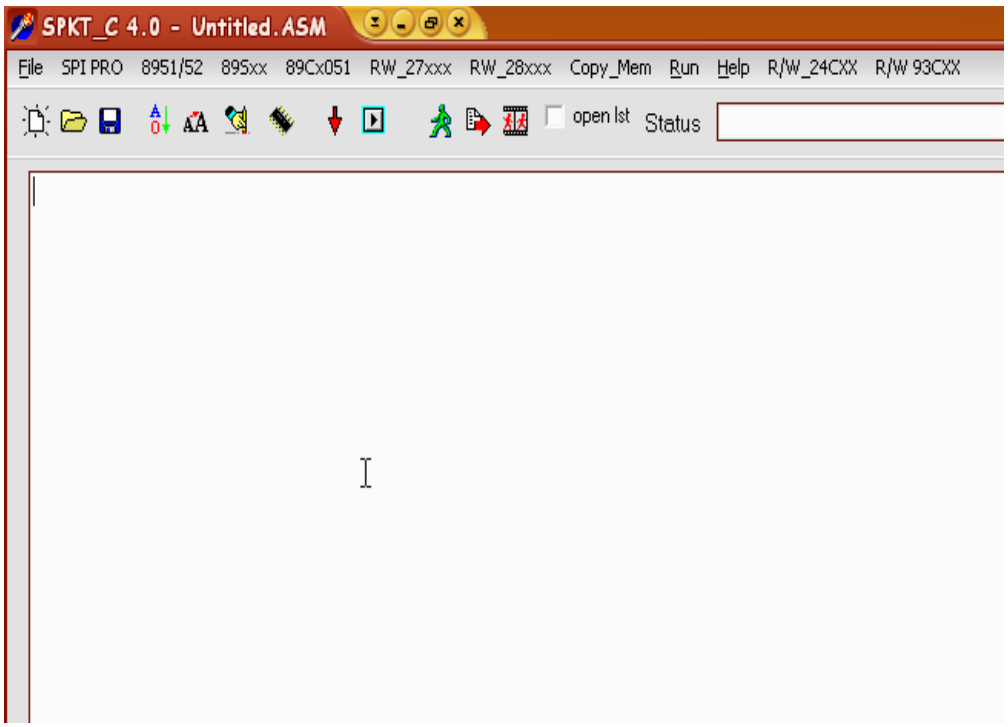
Sau đây là trình tự hướng dẫn sử dụng phần mềm điều khiển và soạn thảo chương trình.

Sau khi cho chương trình chạy thì màn hình của chương trình như hình 8-22:



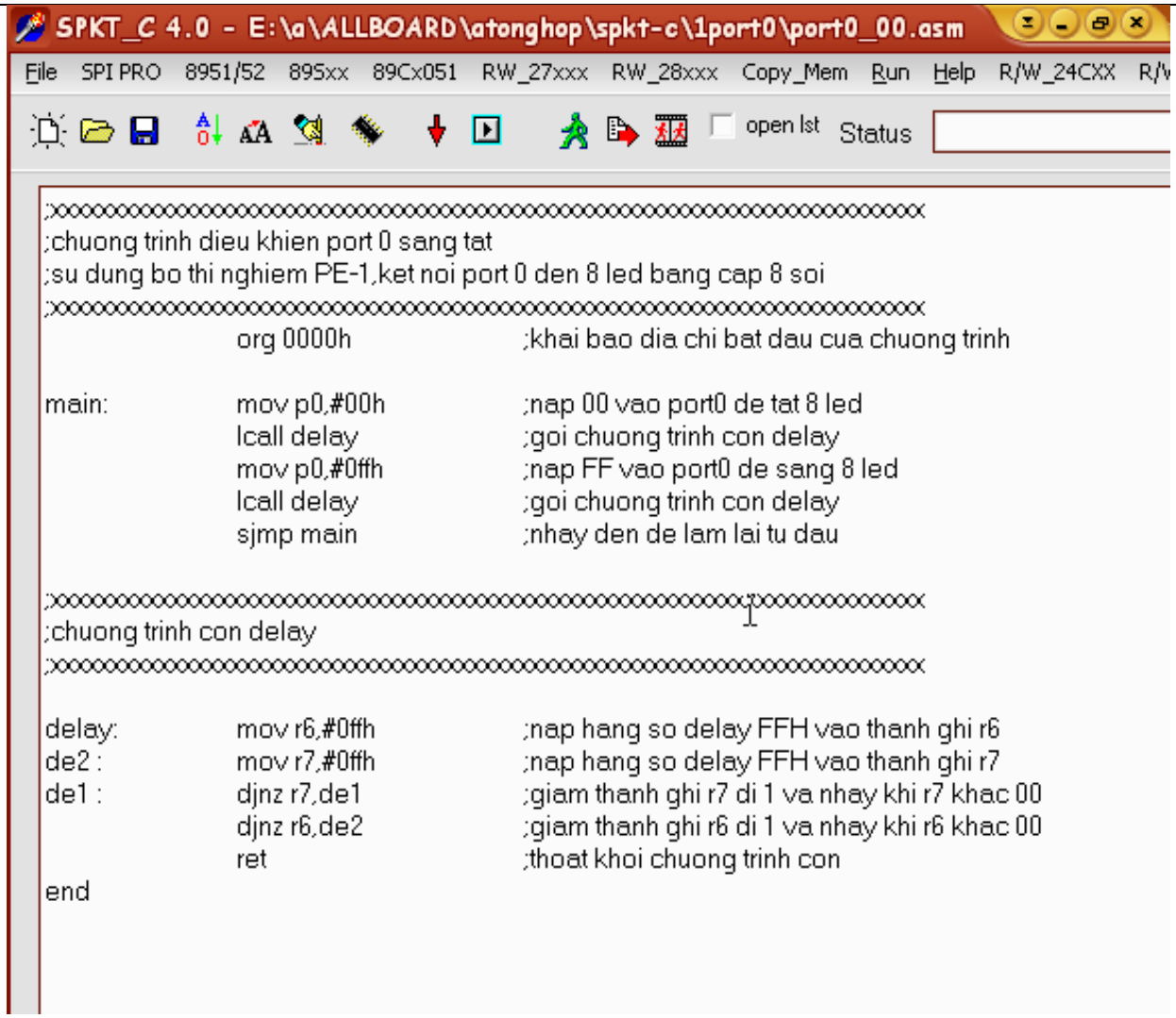
Hình 8-22. Màn hình của chương trình.

Nếu bạn muốn dùng chương trình này để soạn thảo và biên dịch thì hãy thực hiện theo đúng trình tự này. Hãy chọn menu File, rồi chọn New và bắt đầu soạn thảo, màn hình soạn thảo như hình 8-23.



Hình 8-23. Màn hình của chương trình soạn thảo.

Cách soạn thảo chương trình giống như bạn đã soạn thảo ở các phần mềm khác và đúng theo yêu cầu nếu chưa biết thì hãy xem các chương trình mẫu. Hãy xem chương trình ví dụ mẫu như hình 8-24:



Hình 8-24. Màn hình của chương trình soạn thảo của chtr port0_00.asm.

Trong bài này tên của file đã soạn thảo là port0_00 và phần mở rộng là asm.

Trong bài này các bạn không nhất thiết phải hiểu chương trình tôi trình bày trong này mà chỉ quan tâm đến cách thức sử dụng chương trình.

Một số điều các bạn cần phải quan tâm là cách viết chương trình sau cho dễ xem: như chương trình trên.

Các hàng đầu tiên là chú thích chương trình làm gì hay mục đích của chương trình – điều này giúp các bạn nhớ lại chương trình một cách nhanh chóng mà không cần phải đọc lại toàn bộ chương trình.

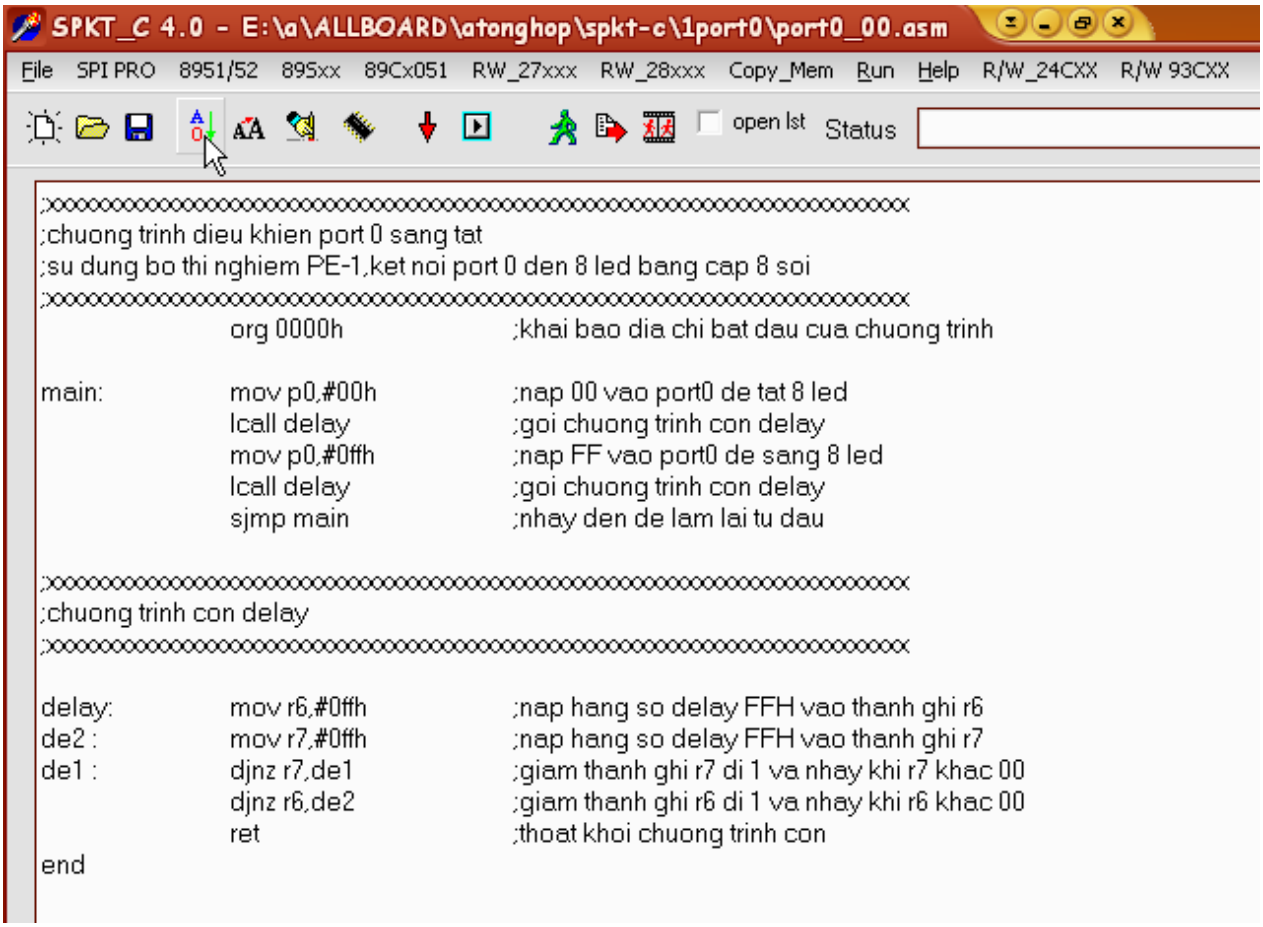
Chú ý luôn có dấu ; đứng trước mỗi hàng.

Org 0000H là khai báo địa chỉ bắt đầu của chương trình, chúng ta còn nhớ khi vi điều khiển bị reset thì giá trị 0000H nạp vào thanh ghi PC và chương trình bắt đầu thực hiện tại địa chỉ này.

Nếu không khai báo hàng này thì mặc nhiên chương trình biên dịch sẽ tự biên dịch tại địa chỉ 0000H.

Sau khi soạn thảo xong bạn hãy tiến hành lưu chương trình vào bộ nhớ và đánh tên file cần lưu vào, không cần đánh thêm phần mở rộng, chương trình soạn thảo tự động thêm vào phần mở rộng asm cho bạn. Sau đó bạn hãy tiến hành biên dịch bằng cách bấm vào biểu tượng có kí hiệu chữ AO để biên dịch file chương trình đã soạn sang file chương trình với phần mở rộng là OBJ, LST và HEX (Compile Asm file to Obj file and hex file).

Bạn hãy nhìn vào hình 8-25 với mũi tên trỏ đến biểu tượng truy xuất nhanh và biểu tượng nổi lên:



Hình 8-25. Tiến hành biên dịch..

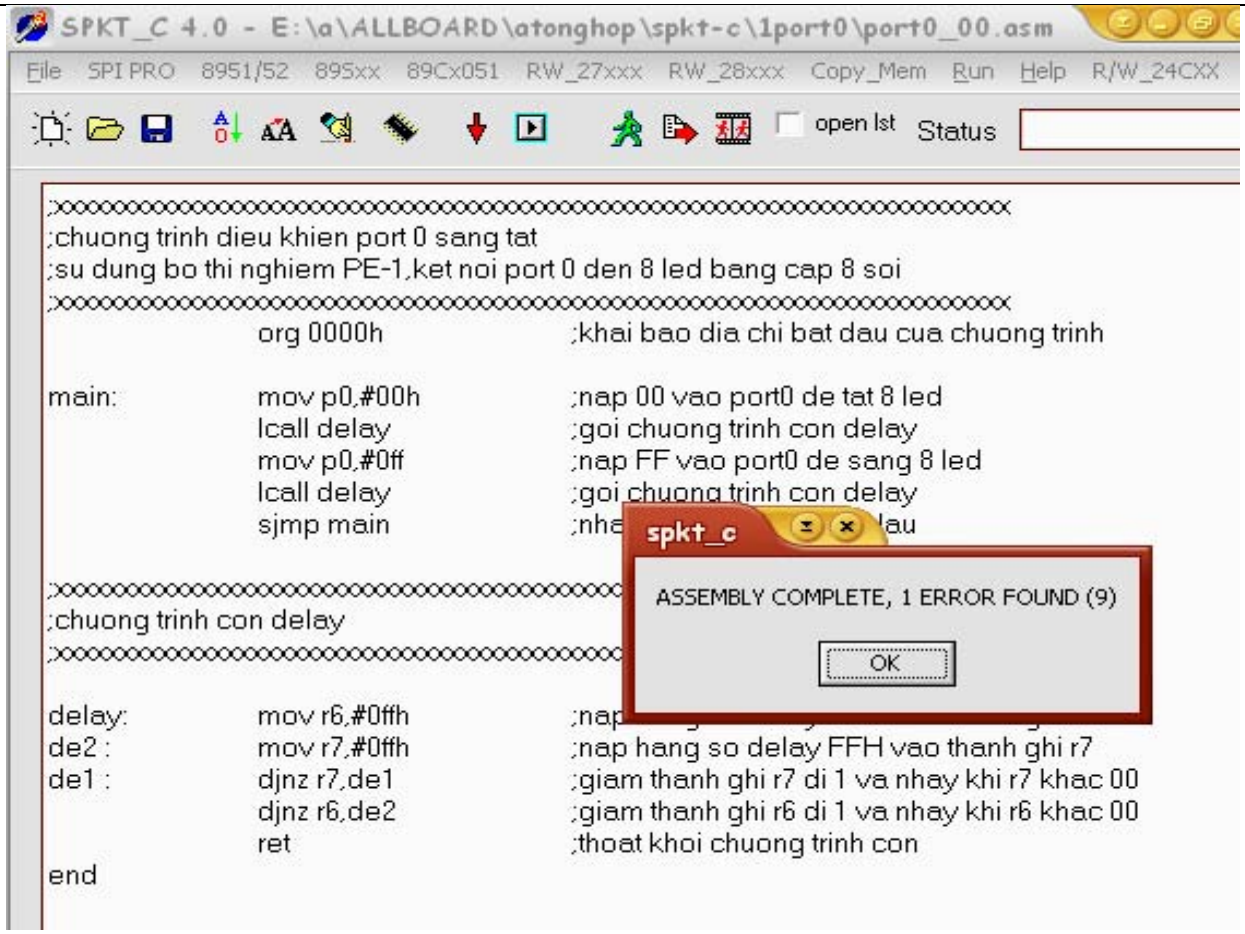
Hàng thông báo cuối chương trình là ASSEMBLY COMLETE, NO ERRORS FOUND chương trình soạn thảo tốt nên không có lỗi.

Đến đây bạn đã có luôn file "hex" để bạn có thể nạp vào bộ nhớ của vi điều khiển.

Để biết chương trình có lỗi thì bạn hãy xem một chương trình soạn thảo có lỗi hình 8-26 và chương trình biên dịch sẽ xác định được bao nhiêu lỗi và vị trí lỗi trong chương trình để người soạn thảo chương trình biết và hiệu chỉnh rồi biên dịch lại cho đến khi hết lỗi.

Lỗi trong chương trình như đã trình bày ở phần tập lệnh, một chương trình không có lỗi chưa chắc chương trình đó thực hiện đúng công việc của bạn, việc kiểm tra ở đây chỉ kiểm tra về cú pháp. Còn các chương trình trong bài thực hành vì đã kiểm nghiệm nên đảm bảo hoạt động đúng.

Nếu bạn thực hành một chương trình nào đó trong này mà chương trình chạy không đúng thì hãy xem lại thật kỹ các lệnh xem bạn có đánh thiết lệnh nào hay không.



Hình 8-26. Chương trình có lỗi.

Trong quá trình biên dịch chương trình biên dịch cũng có thông báo số lượng lỗi và vị trí của lỗi nhưng không được đầy đủ tốt nhất là bạn hãy xem lỗi trong file với phần mở rộng "lst" do chương trình biên dịch tạo ra.

Trong file với phần mở rộng lst sẽ chỉ rõ cho bạn từng hàng lệnh và lỗi nằm ở hàng nào và bạn biết được nguyên nhân gây ra lỗi và tiến hành hiệu chỉnh lại chương trình.

Do sử dụng chương trình biên dịch của nước ngoài nên các thông báo lỗi hiển thị bằng tiếng anh.

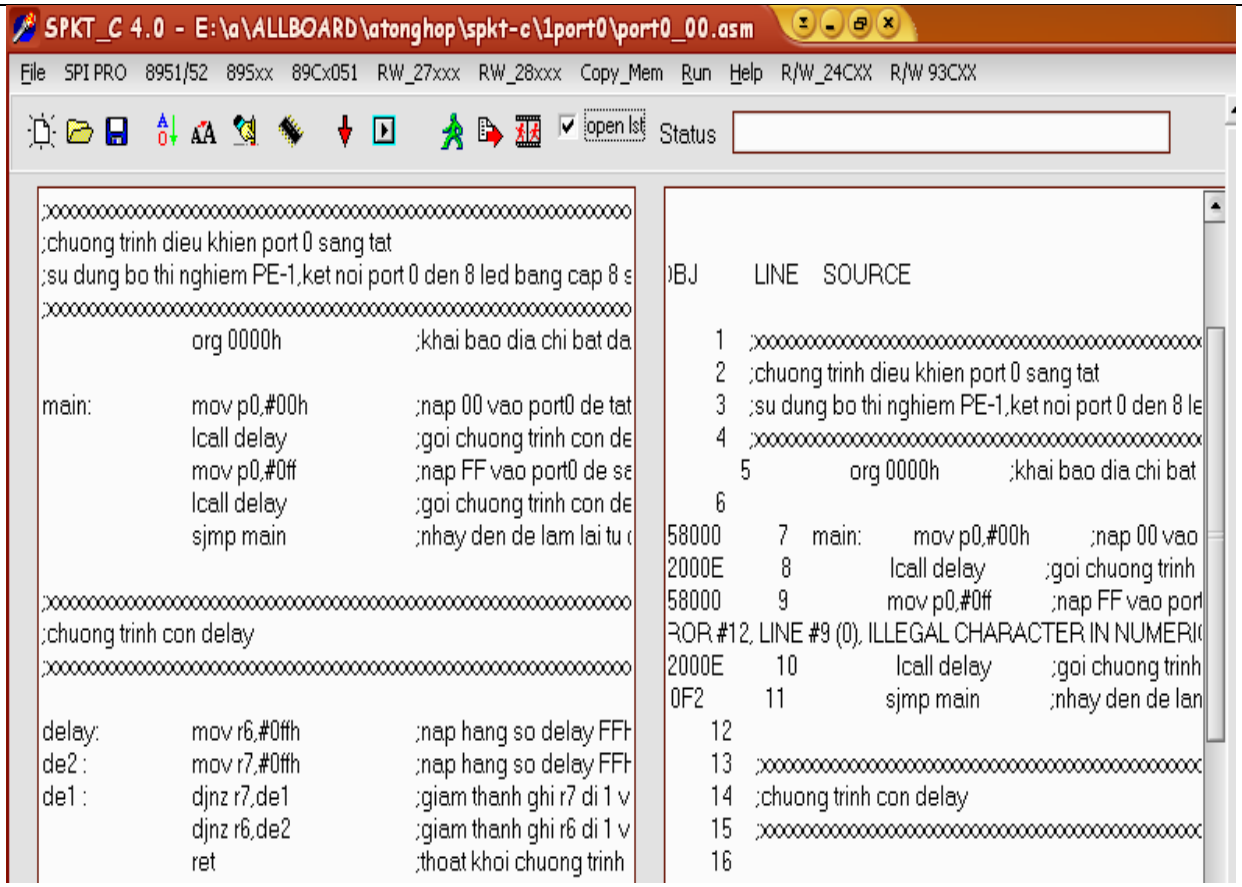
Để tiện lợi cho việc mở file với phần mở rộng lst thì bạn hãy bấm nút chọn có tên là open list trên thanh menu khi đó màn hình sẽ chia làm 2, bên trái là chương trình gốc với phần mở rộng là asm và bên phải là chương trình với phần mở rộng lst để xác định lỗi như hình 8-27.

Trong chương trình trên có 2 lỗi đã được tìm thấy. Lỗi ở hàng nào thì sẽ xuất hiện tại hàng đó trong file lst. Bạn hãy hiệu chỉnh rồi biên dịch lại thì sẽ hết lỗi.

Sau khi đã hết lỗi bạn muốn trở lại màn hình như ban đầu thì hãy bấm bỏ open lst.

Các lỗi thường xảy ra rất nhiều đối với người bắt đầu học lập trình vì thường đánh sai cú pháp lệnh, sai nhãn, địa chỉ,...

Trong đĩa CD Rom có sẵn một số file đã biên dịch bạn có thể nạp thử và đọc lại. Các file biên soạn sẵn lưu trong các thư mục.



Hình 8-27. Mở file lst.

3. Cổng giao tiếp:

Chương trình có thể tự động chọn từ cổng COM1 đến COM20. Tốc độ là cố định 19200 không nên thay đổi.

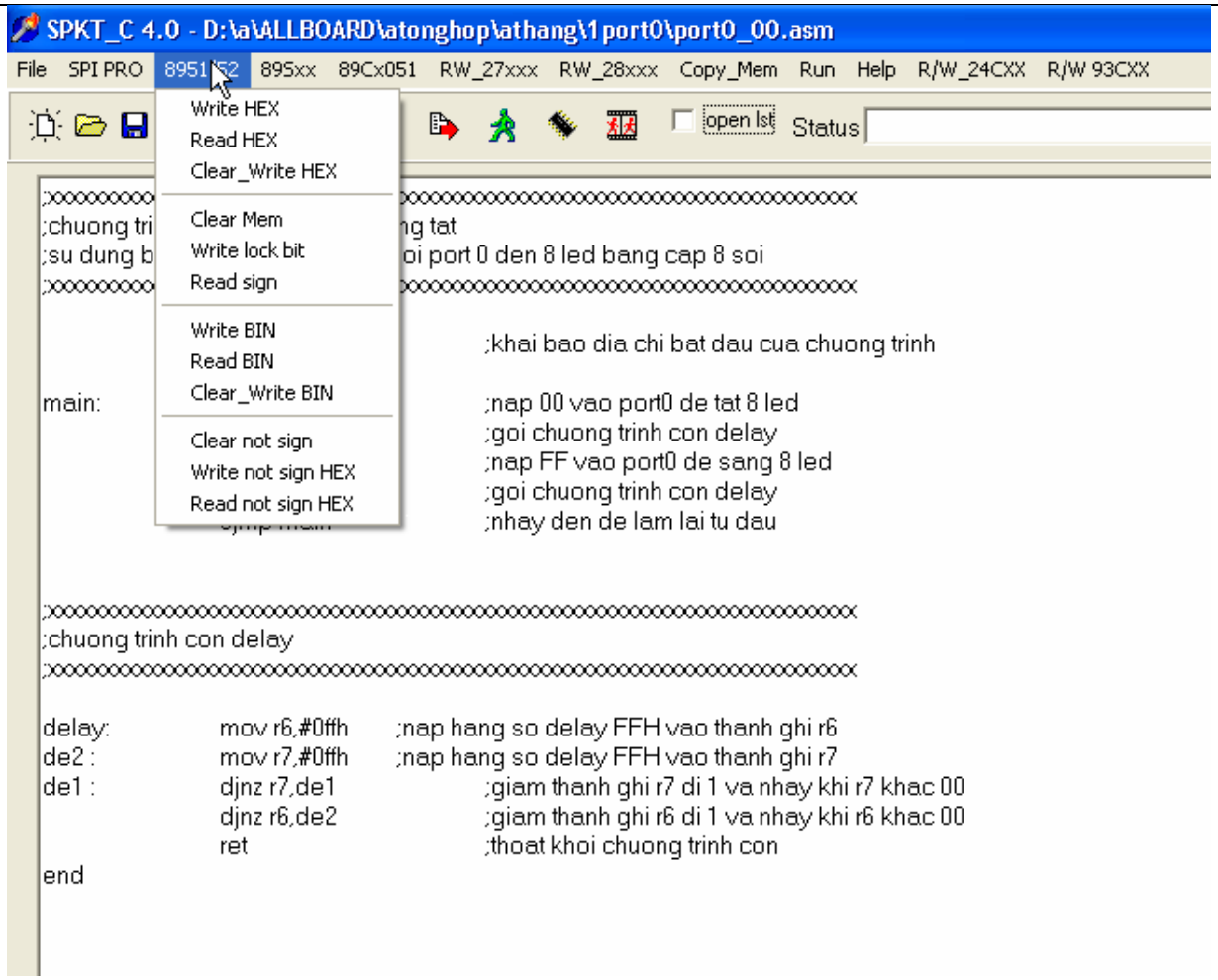
4. Nạp bộ nhớ vi điều khiển 89C51/52:

Trong bộ thí nghiệm có thể nạp được chương trình cho nhiều loại vi điều khiển. Ở phần này sẽ trình bày trình tự nạp chương trình cho vi điều khiển loại 89C51 và 89C52, 2 vi điều khiển này được thiết kế để cho phép nạp song song.

Chạy chương trình SPKT_C và máy tính có kết nối bộ nạp hoặc bộ thí nghiệm thì bạn có thể nạp file hex cần nạp vào vi điều khiển tiến trình thực hiện như sau:

- Gắn vi điều khiển vào socket nạp như đã chỉ rõ ở trên.

- Chọn menu lệnh 8951/52 khi đó có một menu xuất hiện như hình 8-28.



Hình 8-28. Mở menu để nạp vi điều khiển họ 89C..

Lệnh "Write Hex" : có chức năng ghi dữ liệu số hex từ máy tính gởi xuống và nạp vào vùng nhớ đã xóa. Quá trình ghi dữ liệu vào và có kiểm tra lại, nếu ghi không được sẽ có thông báo về cho máy tính biết và quá trình ghi sẽ ngừng ngay lập tức. ***Không ghi được dữ liệu vào vi điều khiển là do có nhiều lí do***: vi điều khiển hỏng hay bộ nhớ chưa xóa hoặc viết chương trình nằm ngoài vùng nhớ của vi điều khiển hoặc kích thước của chương trình lớn hơn dung lượng bộ nhớ nội cho phép.

Lệnh "Read Hex": có chức năng đọc dữ liệu từ bộ nhớ của vi điều khiển về hiển thị trên máy tính dạng số hex.

Lệnh :Clear Write – Hex": có chức năng vừa xóa bộ nhớ và nạp dữ liệu cho vi điều khiển.

Lệnh "Clear mem": có chức năng xóa bộ nhớ của vi điều khiển.

Lệnh "Write – lock bit": có chức năng khoá bộ nhớ không cho đọc dữ liệu ra.

Lệnh "Read sign": có chức năng đọc mã của IC do nhà sản xuất qui định. Bạn hãy xem data sheet của từng IC để biết mã của từng loại vi điều khiển.

Lệnh "Write bin" : có chức năng ghi dữ liệu dạng file BIN từ máy tính gởi xuống và nạp vào vùng nhớ đã xóa. Quá trình ghi dữ liệu vào và có kiểm tra lại, nếu ghi không được sẽ có thông báo về cho máy tính biết và quá trình ghi sẽ ngừng ngay lập tức.

Lệnh "Read Bin": có chức năng đọc dữ liệu từ bộ nhớ của vi điều khiển về máy tính và lưu dạng file bin – không thể hiển thị trên màn hình.

Lệnh :Clear Write – Bin": có chức năng vừa xóa bộ nhớ và nạp dữ liệu cho vi điều khiển dạng file BIN.

Lệnh "Clear -Not sign": có chức năng xóa bộ nhớ của vi điều khiển mà không cần kiểm tra IC có hay không.

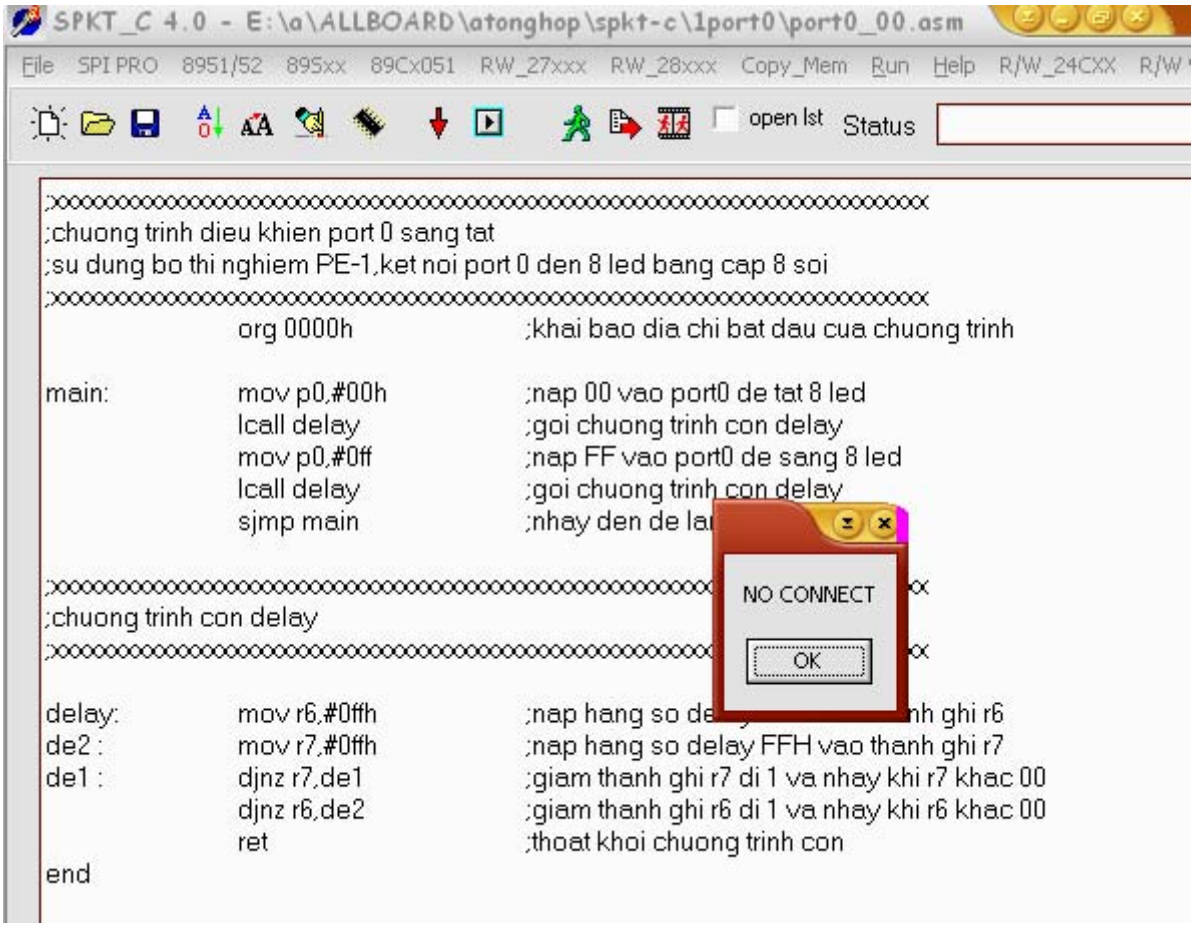
Lệnh "Read not sign hex": có chức năng đọc mã dạng file hex mà không kiểm tra IC có hay không.

Lệnh "Write - Not sign hex": có chức ghi dữ liệu vào bộ nhớ mà không cần kiểm tra có IC hay không. Nếu không ghi được thì sẽ ngừng ngay lập tức.

Các lệnh sau cùng được thực hiện vì một số IC do nhà sản xuất không có mã của nhà sản xuất.

Khi bạn thực hiện 1 trong các lệnh trên nếu hệ thống kết nối không tốt hoặc sai cổng thì sẽ có 1 câu thông báo không kết nối được như hình 8-29:

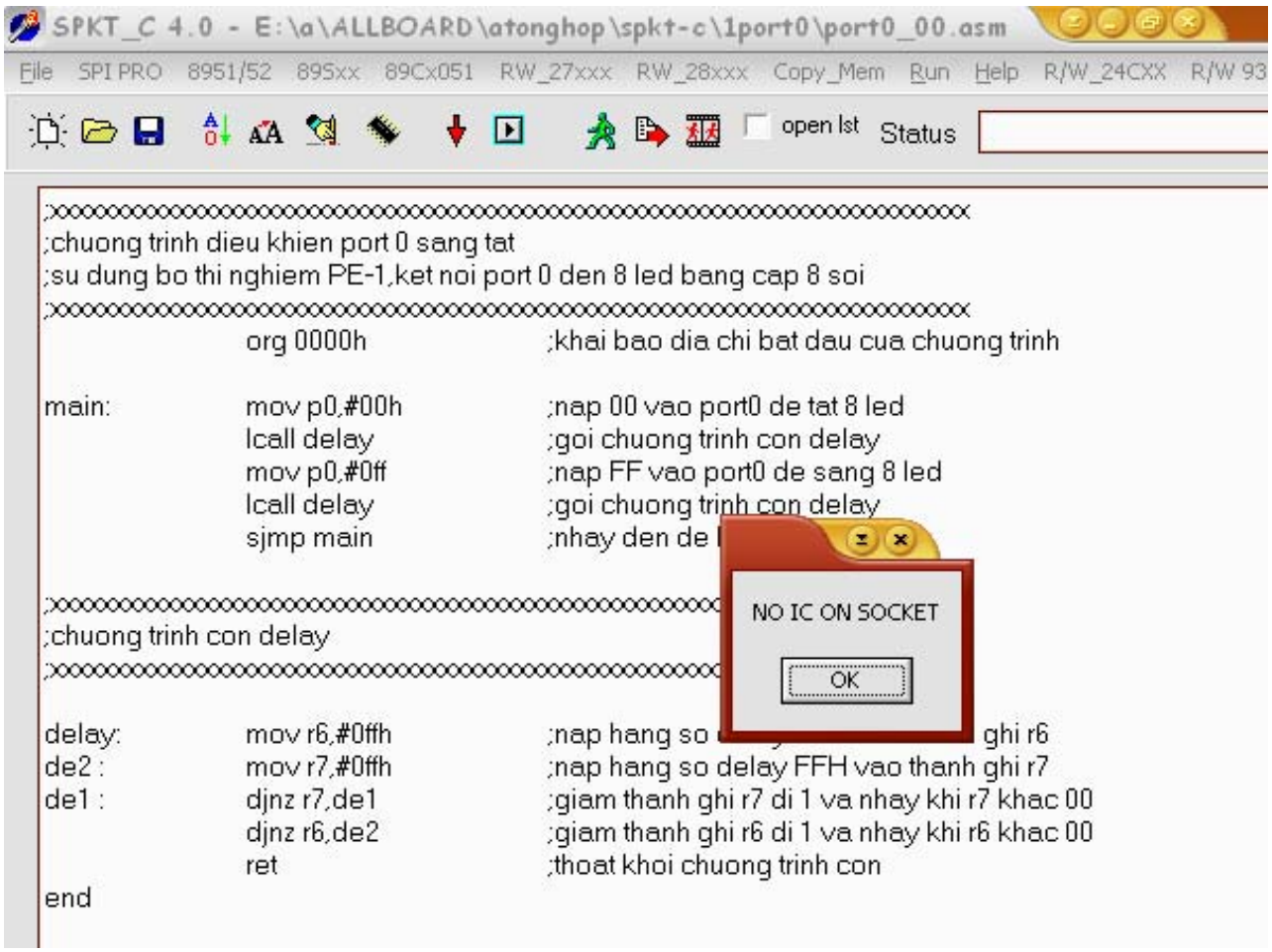
Chú ý: trên các biểu tượng truy xuất nhanh có 1 biểu tượng hình IC cho phép bạn nạp và xóa vi điều khiển 89C51.



Hình 8-29. Thông báo máy tính chưa kết nối với bộ thí nghiệm.

Khi không kết nối được thì nên xem lại nguồn cung cấp cho thiết bị nạp hoặc bộ thí nghiệm có hay chưa và hãy reset lại thiết bị hoặc tắt mở lại nguồn cung cấp. Nếu đang gắn vi điều khiển vào socket để nạp thì hãy lấy ra và thử lại cho đến khi kết nối được. Nếu vẫn không được thì hãy gọi cho chúng tôi xử lý.

Khi kết nối tốt mà bạn chưa gắn IC vi điều khiển trên socket nạp thì sẽ có 1 câu thông báo xuất hiện như hình 8-30:



Hình 8-30. Chương trình thông báo chưa tìm thấy IC vi điều khiển trên socket..

Thông báo cho biết là bạn chưa gắn IC trên socket nạp hoặc nếu có gắn mà IC đã hỏng hoặc gắn chưa đúng hoặc chưa tiếp xúc tốt - bạn phải xem lại.

Chú ý là gắn IC phải đúng cực tính đã qui định nếu không sẽ làm hỏng IC.

Khi mọi thứ đã sẵn sàng thì bạn hãy chọn lệnh xóa và ghi để thực hiện 2 chức năng là xóa và ghi dữ liệu vào bộ nhớ.

Khi thực hiện tốt thì quá trình ghi sẽ được thực hiện và cuối cùng sẽ có 1 câu thông báo là quá trình ghi hoàn tất như hình 8-31.

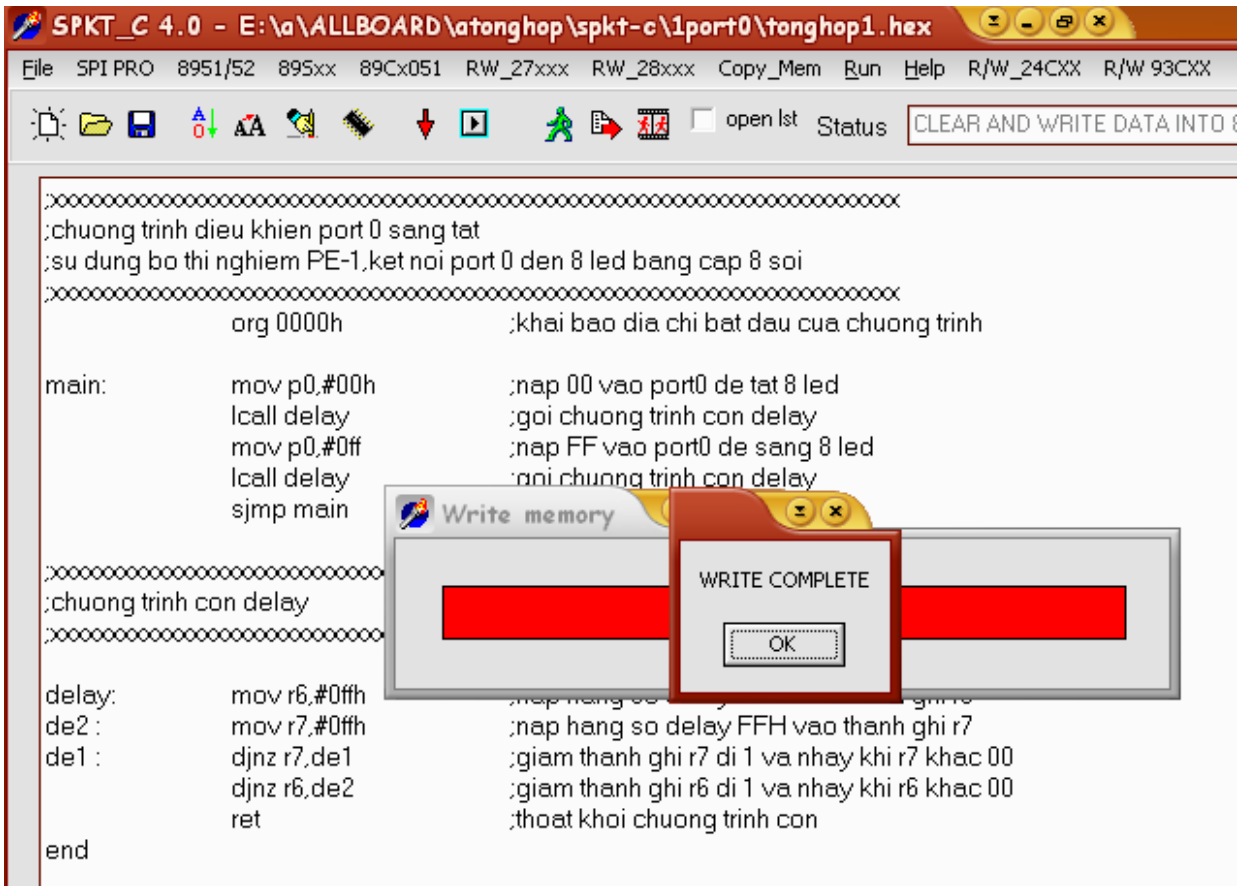
Khi bạn muốn đọc dữ liệu từ bộ nhớ của vi điều khiển về thì bạn hãy chọn lệnh "Read hex" và sau đó quá trình đọc sẽ diễn ra và kết thúc quá trình đọc khi hết dữ liệu. Hệ thống nạp tự động nhận biết vi điều khiển 89C51 hoặc 89C52.

Dữ liệu đọc về bạn cần lưu thì lưu dưới dạng file hex với tên và phần mở rộng bạn phải đánh đầy đủ nếu không lưu thì bạn chọn "No" khi thực hiện 1 thao tác khác.

Sau khi nạp xong bạn muốn chạy thử thì gắn vi điều khiển đã nạp chương trình vào socket thử và kết nối các port với led đơn hoặc led 7 đoạn hoặc các thiết bị bên ngoài và sau đó mở nguồn và xem kết quả chương trình chạy có đúng hay không?

Nếu đúng thì bạn đã viết đúng còn nếu sai thì bạn phải xem lại phần cứng kết nối đúng hay chưa – nếu đúng rồi thi thôi và nếu chưa đúng thì kiểm tra lại chương trình. Trong từng bài thí

nghiệm chúng tôi trình bày rất rõ cách kết nối giữa các port của vi điều khiển với các đối tượng điều khiển.



Hình 8-31. Thông báo sau khi nạp xong chương trình.

Khi muốn copy dữ liệu từ một vi điều khiển A sang vi điều khiển B thì ta phải gắn IC A vào đế, thực hiện chức năng đọc dữ liệu, lưu trữ dữ liệu đọc về trên màn hình bằng lệnh Save trong menu File và đặt tên file tuỳ ý khoảng 8 kí tự với phần mở rộng là .hex. Sau đó lấy IC A ra khỏi đế, gắn IC B vào đế và tiến hành nạp bằng lệnh Write với file mới vừa lưu trữ. Có thể đọc về và lưu file ở dạng BIN rồi ghi cho các vi điều khiển khác.

Trong chương trình này còn nhiều menu lệnh khác phục vụ chung cho tất cả các bộ thí nghiệm khác mà bộ thí nghiệm này không đáp ứng được nên không thể tác động được.

Chú ý khi thực hiện thao tác đọc – ghi đối với họ vi điều khiển nếu chọn nhầm lệnh khác thành lệnh **clear** thì dữ liệu trong bộ nhớ sẽ bị xóa hết.

Trong quá trình nạp đã có quá trình kiểm tra và nếu không nạp được thì hệ thống sẽ ngừng và thông báo không nạp được.

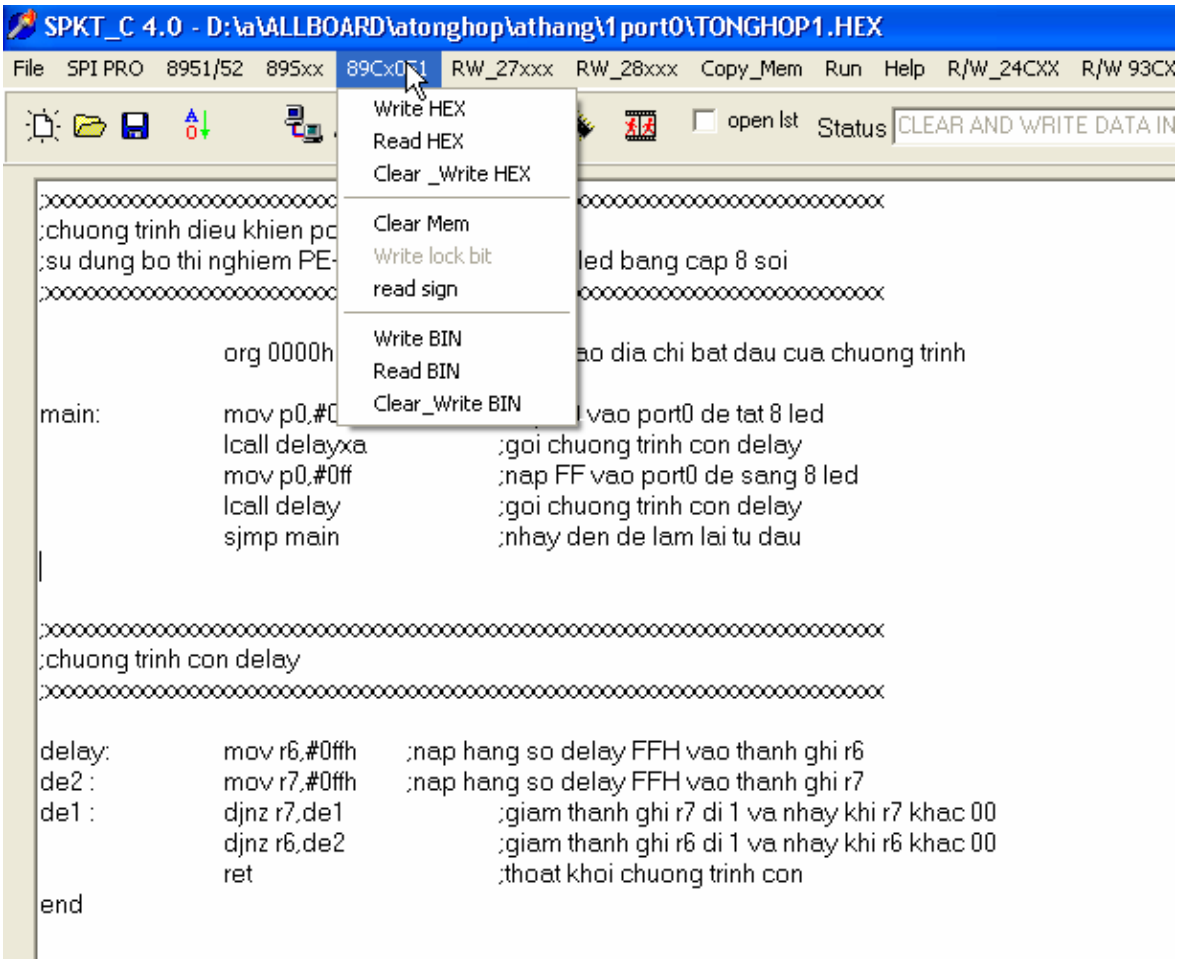
Trong một chương trình nếu bạn phân chia nhiều vùng nhớ thí hãy chú ý đừng để các vùng nhớ xếp chồng lên nhau và khi đó chương trình sẽ không nạp được.

5. Nạp bộ nhớ vi điều khiển 89C2051:

Nếu bạn muốn nạp chương trình vào vi điều khiển 89C1051 hoặc 89C2051 hoặc 89C4051 thì hãy tiến hành như sau:

- Gắn vi điều khiển vào socket nạp 40 chân từ trên xuống [chỉ sử dụng 20 chân trên] như đã trình bày ở trên.

- Chọn menu lệnh 89Cx051 khi đó có một menu xuất hiện như hình 8-32.



Hình 8-32. Các lệnh để lập trình cho vi điều khiển 89C2051, 89C4051 .

Các lệnh trong menu này có chức năng giống như các lệnh trong menu lệnh nạp cho vi điều khiển 89C51/52.

6. Nạp bộ nhớ 27xx:

Nếu bạn muốn nạp chương trình vào bộ nhớ họ 27xx [từ 2716 đến 27512] thì hãy tiến hành như sau:

- Gắn vi điều khiển vào socket nạp 28 chân – chú ý cách gắn như đã trình bày ở phần 1. chú ý các IC nhớ có dung lượng nhỏ hơn thì gắn giống chiều như vậy nhưng bắt đầu tính từ dưới lên.

- Chọn menu lệnh RW_27xx khi đó có một menu xuất hiện như hình 8-33.

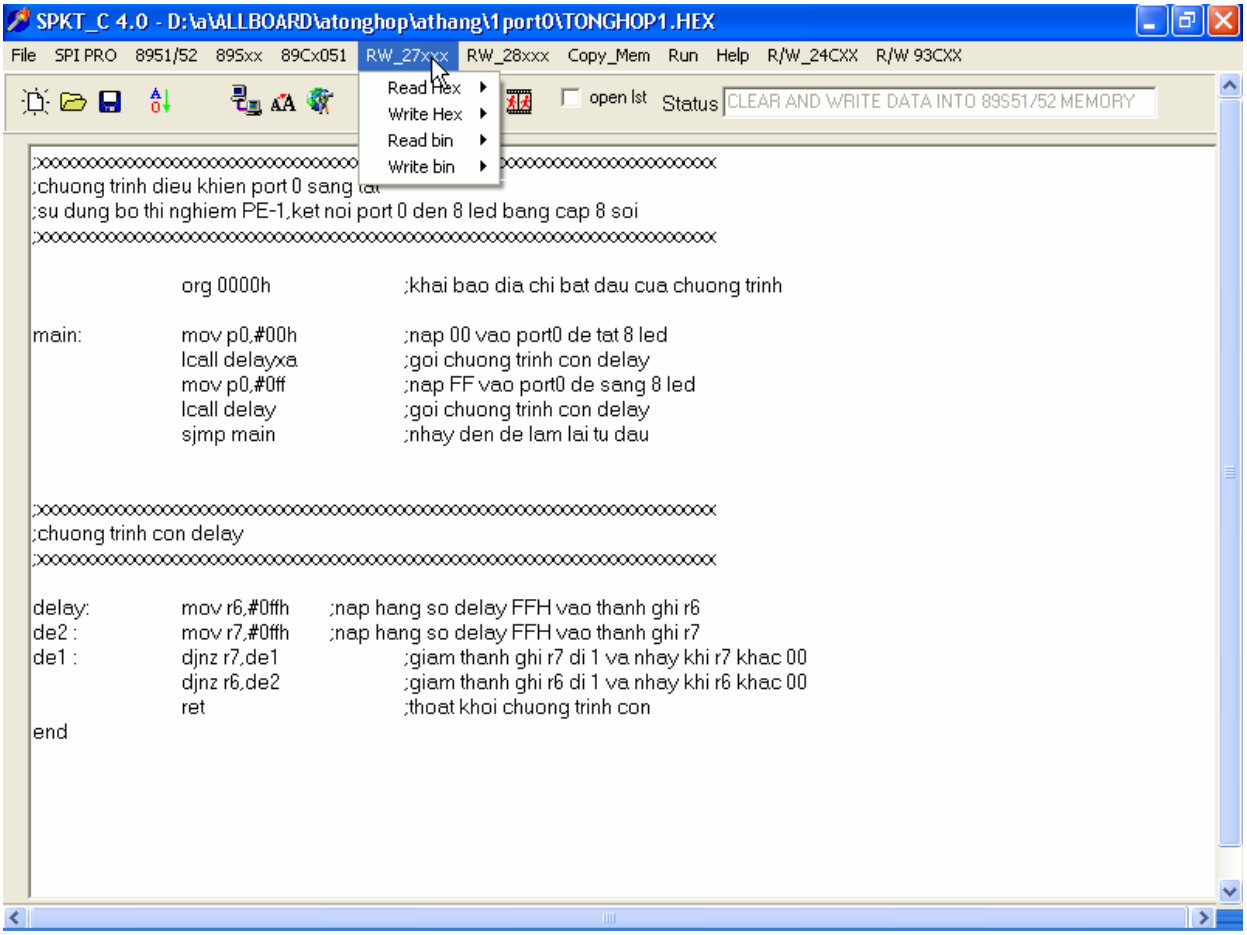
Trong menu lệnh này có các lệnh đọc dữ liệu từ bộ nhớ và các lệnh ghi dữ liệu vào bộ nhớ với phần mở rộng là hex và bin.

Trong từng menu có thêm menu con để cho phép lựa chọn IC nạp theo từng loại và từng cấp điện áp khi thực hiện quá trình ghi dữ liệu vào bộ nhớ. Khi đọc thì không cần thiết lựa chọn điện áp.

Loại bộ nhớ này xóa bằng ánh sáng tia cực tím. Khi xóa xong thì dữ liệu đọc về là FF.

Nếu bộ nạp không nạp được thì hãy xem lại bộ nhớ đã xóa hết dữ liệu hay chưa bằng cách đọc và hiển thị trên màn hình máy tính.

Đối với bộ nhớ Eprom thì có 2 loại: loại nạp nguồn 12.5V và loại nạp nguồn 25V. Bạn hãy quan sát điện áp nạp có ghi trên IC nhớ, nếu nạp 12.5V thì có ghi, còn không ghi thì mặc nhiên nạp 25V.



Hình 8-33. Các lệnh để lập trình cho bộ nhớ eprom họ 27xx .

7. Nạp bộ nhớ 28xx:

Nếu bạn muốn nạp chương trình vào bộ nhớ họ 28xx [từ 2816 và 2864] thì hãy tiến hành như sau:

- Gắn vi điều khiển vào socket nạp 28 chân giống như gắn Eprom – chú ý cách gắn cho đúng chiều.

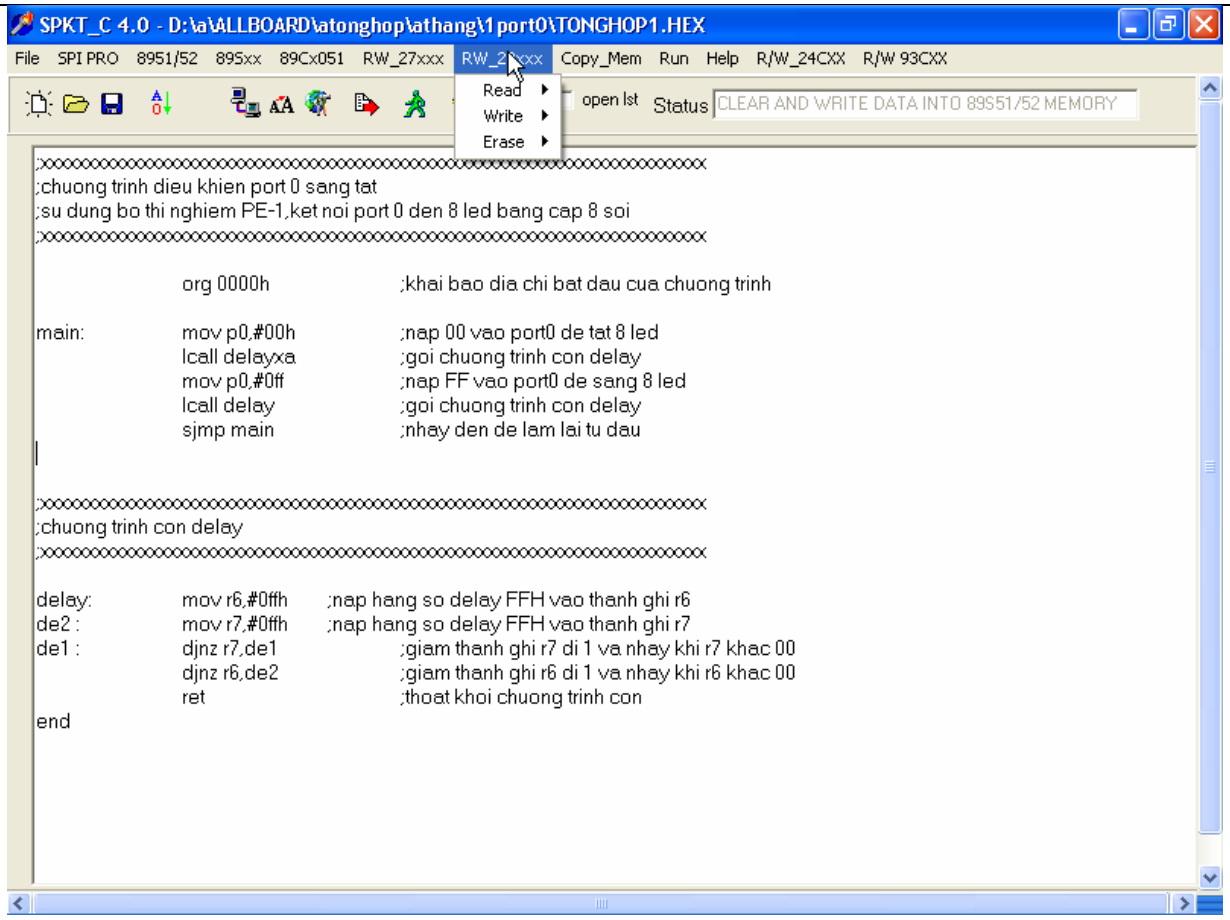
- Chọn menu lệnh RW_28xx khi đó có một menu xuất hiện như hình 8-34:

Trong menu lệnh này có các lệnh đọc dữ liệu từ bộ nhớ và các lệnh ghi dữ liệu vào bộ nhớ với phần mở rộng là hex và bin.

Trong từng menu có thêm menu con để cho phép lựa chọn IC nạp theo từng loại.

Loại bộ nhớ này xóa bằng xung điện và trong menu lệnh cho phép xoá nếu không thì bạn có thể ghi chồng lên không cần quan tâm đến dữ liệu trước đó.

Trong quá trình nạp đã có quá trình kiểm tra và nếu không nạp được thì hệ thống sẽ ngừng và thông báo không nạp được.



Hình 8-34. Các lệnh để lập trình cho bộ nhớ eeprom họ 28xx .

8. Nạp chương trình xuống bộ nhớ ngoại và chạy chương trình :

Vi điều khiển có thể lưu chương trình ở bộ nhớ nội và nếu chương trình điều khiển lớn hơn bộ nhớ nội thì phải sử dụng bộ nhớ ngoại. Khả năng giao tiếp với bộ nhớ ngoại là 64kbyte bộ nhớ chương trình và 64kbyte bộ nhớ dữ liệu. Phần thiết kế và cấu hình của hệ thống II đã trình bày ở trên, ở đây chỉ trình bày phần mềm có liên quan đến hệ thống thống II.

Hệ thống thứ 2 của bộ thí nghiệm này sử dụng vi điều khiển kết nối với bộ nhớ bên ngoài với dung lượng 32kbyte ram có địa chỉ từ 8000H đến FFFFH.

Các chương trình thực hành sử dụng bộ nhớ ngoài phải viết nằm trong vùng địa chỉ này.

Sự tiện lợi của việc sử dụng bộ ngoài trong hệ thống này là khi bạn viết một chương trình nào đó và muốn chạy thử thì chỉ cần nạp chương trình đó xuống bộ nhớ ram rồi chạy thử, nếu chưa đúng thì hiệu chỉnh và thực hiện lại cho đến khi chạy đúng, điều này giúp bạn không cần phải nạp chương trình vào bộ nhớ rất mất nhiều thời gian.

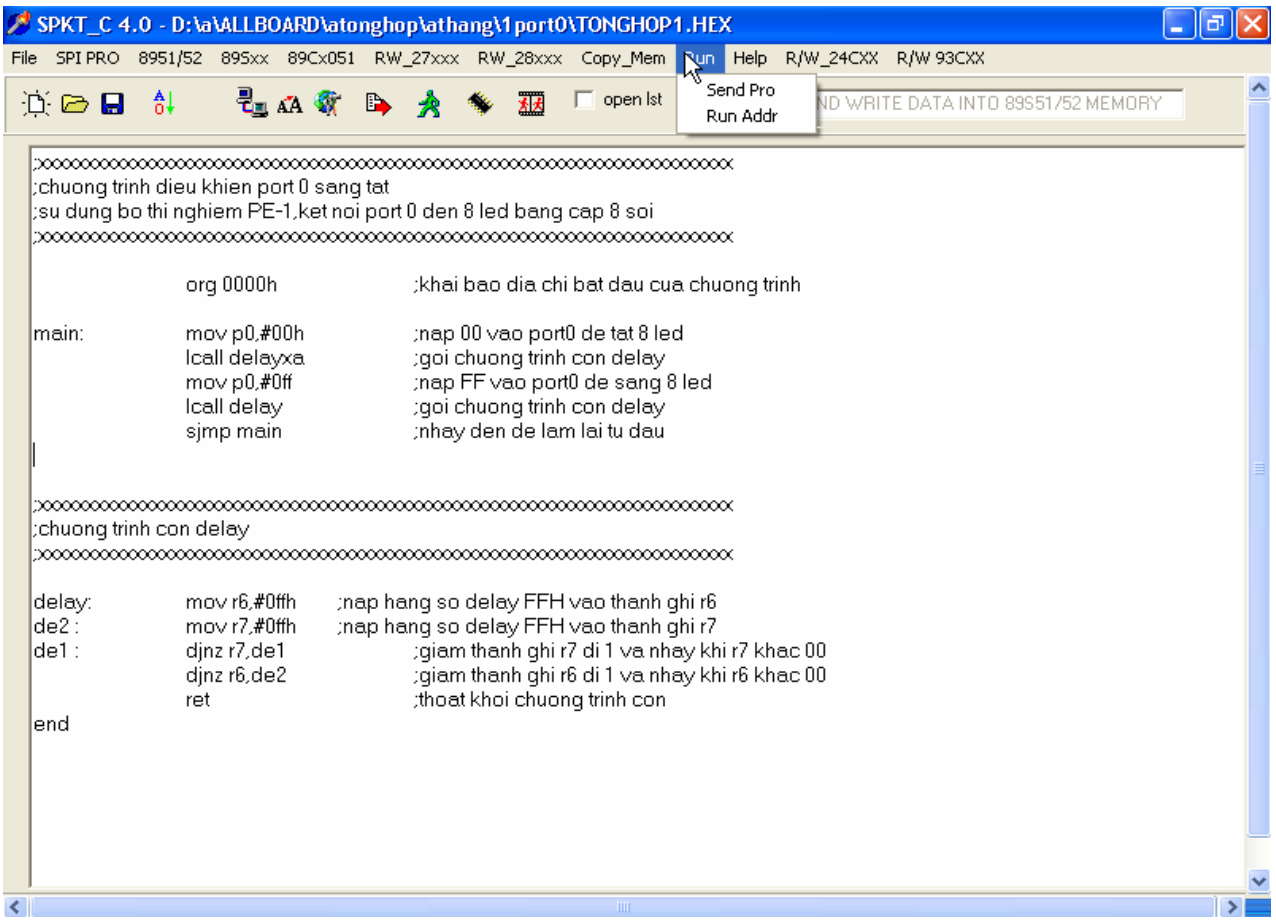
Sau khi viết xong chương trình và biên dịch thành file hex thì bạn có thể gởi chương trình này xuống bộ nhớ của hệ thống bằng lệnh “send pro” hãy xem hình 8-35.

Khi bạn chọn lệnh “send pro” thì một menu xuất hiện cho phép bạn chọn file và gởi xuống bộ thí nghiệm và sẽ có thông báo kết thúc quá trình gởi dữ liệu.

Sau khi gởi xong bạn có thể chạy chương trình bằng lệnh “Run addr” và sau đó đánh địa chỉ bắt đầu của chương trình và ấn enter. [chú ý: đánh đúng số không có kí hiệu H cho số hex].

Nếu đánh sai địa chỉ thì chương trình sẽ không chạy hoặc chạy không đúng chương trình đã viết.

Sau khi chạy chương trình xong bạn muốn chạy chương trình khác thì phải reset lại để thoát khỏi chương trình đang chạy để bắt đầu lại chương trình mới.



Hình 8-35. Các lệnh để nạp và chạy chương trình trên hệ thống 2 .

Ở hệ thống II do sử dụng bộ nhớ ngoài nên chỉ còn port1 và một số bit của port 3 là chưa sử dụng còn các port 0, 2 và 2 bit WR và RD của port 3 đã sử dụng nên trong các chương trình điều khiển chỉ sử dụng port1 và các port của IC ngoại vi 8255 để điều khiển.

Các thông số và địa chỉ của 8255 thì bạn hãy xem ở phần cấu hình hệ thống bộ thí nghiệm.

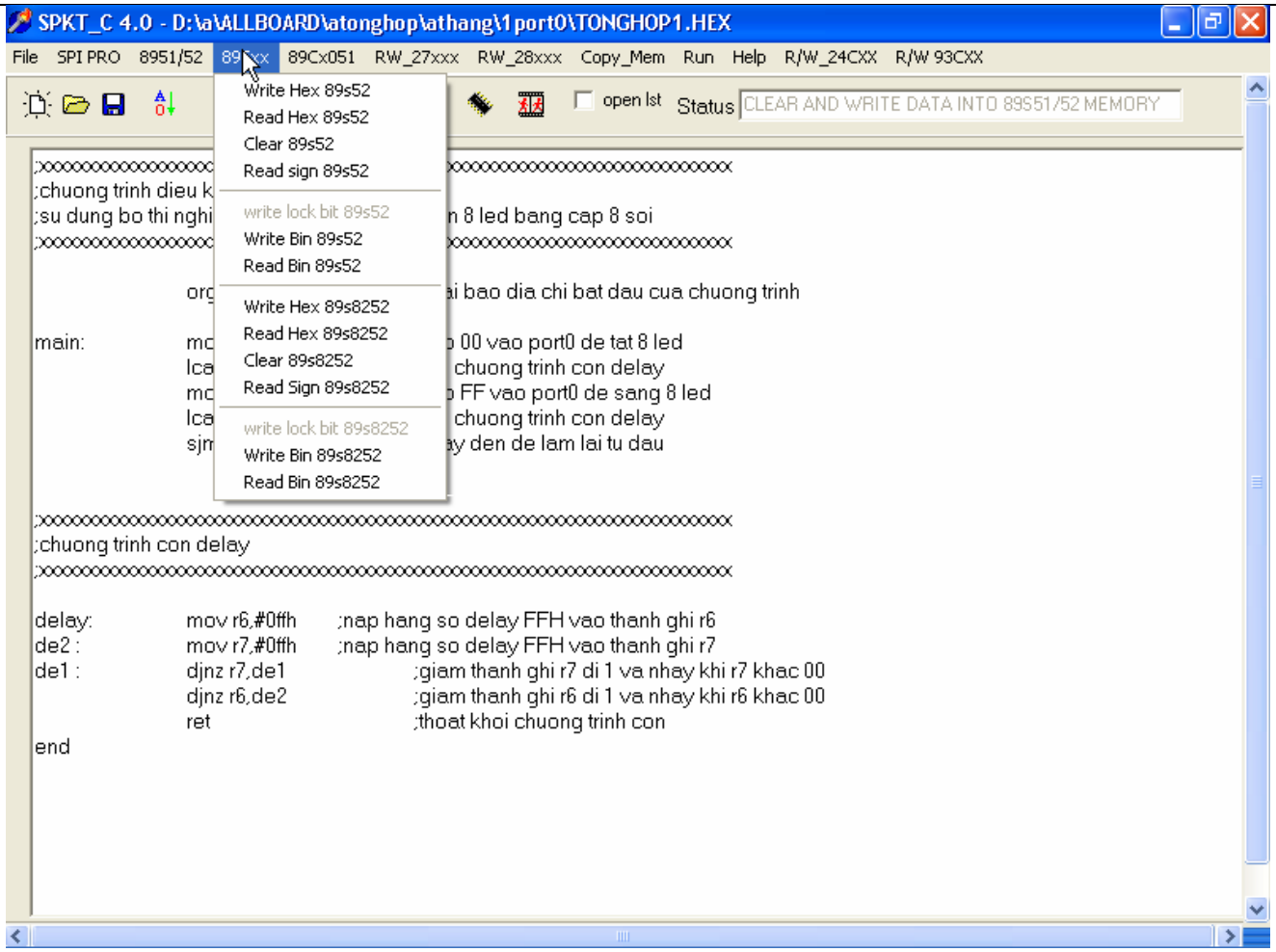
9. Nạp bộ nhớ vi điều khiển 89S51/52 và 89S8252:

Các vi điều khiển thế hệ mới hơn như 89S51, 89S52 hoặc 89S8252 co nhiều tính năng hay hơn cho phép nạp song song giống như vi điều khiển 89C51 và có thêm chức năng nạp nối tiếp.

Trong bộ thí nghiệm mới này cho phép nạp chương trình cho các vi điều khiển 89S song song giống như 89C bằng các lệnh trong menu 89Sxx như hình 8-36.

Các lệnh có chức năng giống như 89C.

Trong quá trình nạp đã có quá trình kiểm tra và nếu không nạp được thì hệ thống sẽ ngừng và thông báo không nạp được.



Hình 8-36. Menu lệnh 89Sxx nạp chương trình cho 89S.

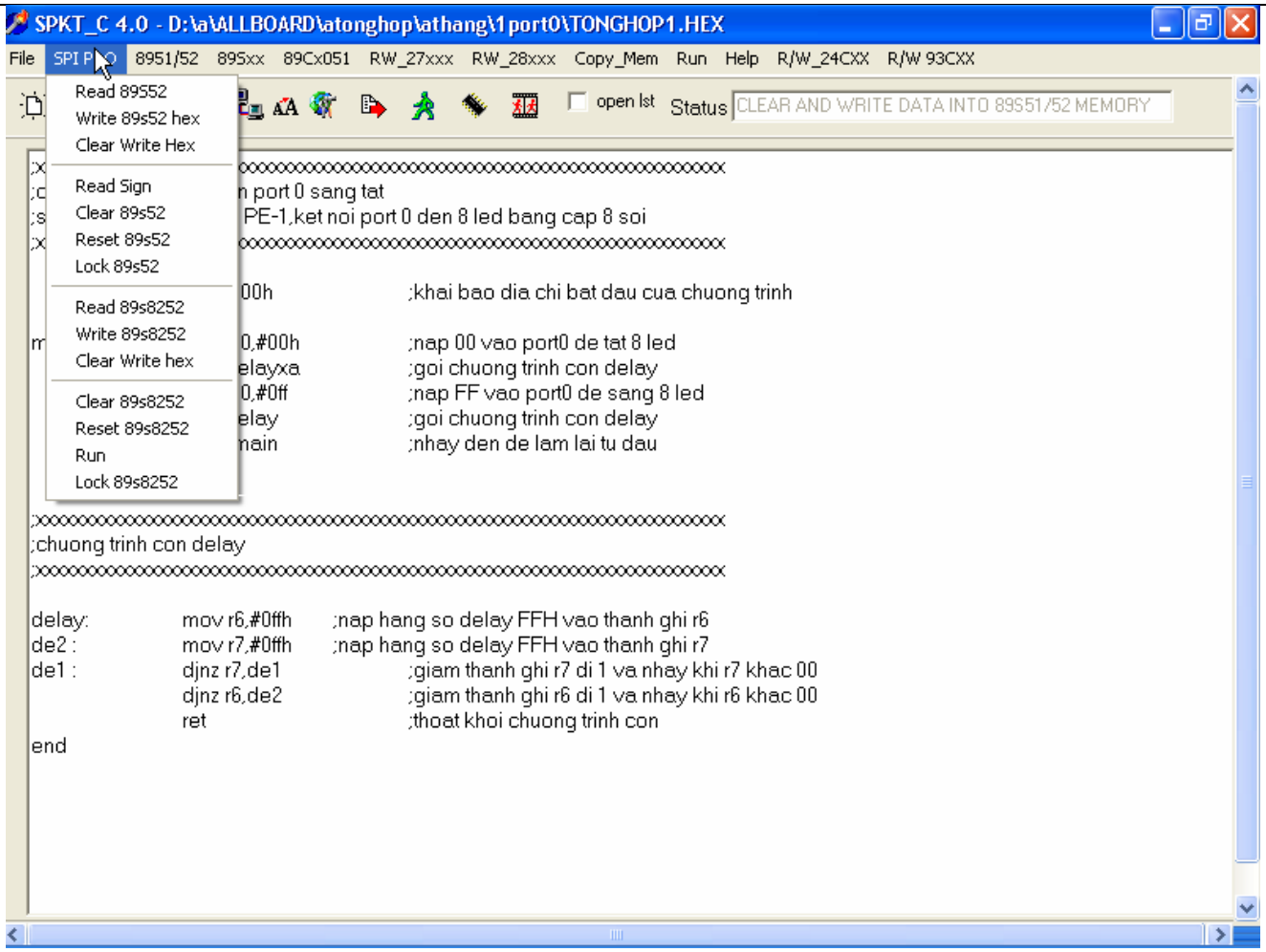
Điều đặc biệt là 89S còn có thêm chức năng nạp nối tiếp có nghĩa là bạn có thể gắn IC vào socket chạy chương trình của hệ thống I, rồi có thể xóa chương trình củ và nạp chương trình mới không cần phải lấy IC ra và gắn vào socket nạp như 89C.

Sau khi nạp xong hệ thống tự động chạy chương trình mới nạp, nếu bạn viết không đúng thì hiệu chỉnh lại chương trình và nạp lại cho đến khi chạy đúng thì thôi.

Sự tiện lợi này giúp cho người học đỡ mất nhiều thời gian gắn thao IC giữa socket nạp và socket chạy thử và quan trọng nhất là dễ làm chết IC nếu gắn ngược chân.

Khi sử dụng chế đ.ộ nạp nối tiếp thì hãy sử dụng các lệnh có trong menu SPI PRO như hình 8-37.

Trong menu này có thể nạp cho 2 loại IC 89S51, 89S52 và 89S8252. Hãy lựa chọn lệnh cho đúng với menu của IC đang nạp.



Hình 8-37. Menu lệnh SPI PRO cho phép nạp chương trình trực tiếp trên socket chạy chạy- chỉ dùng cho họ 89S.

Nếu có gì xin các bạn liên hệ với Phú với số điện thoại 0903 982 443 hoặc email: "phu_nd@yahoo.com".

Chúc thành công.

Chương 9

CÁC BÀI THỰC HÀNH

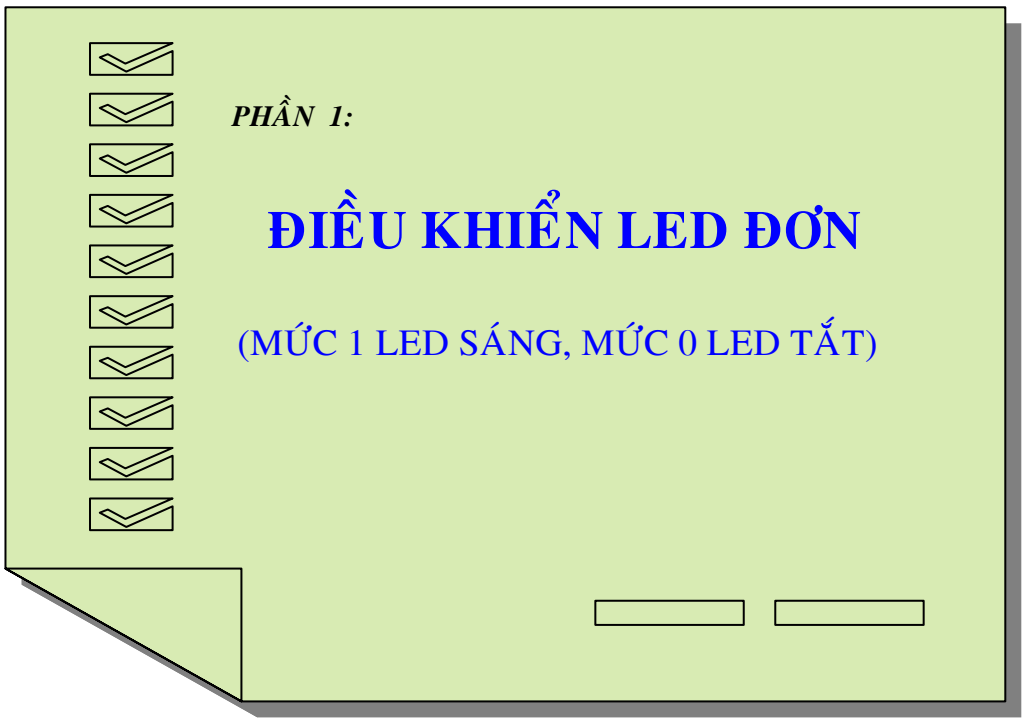
MỘT SỐ QUI ĐỊNH VÀ CÁC LỖI THƯỜNG GẶP

Một số qui định về kết nối:

- Trên hệ thống I có 4 port 0, 1, 2, 3 và các port có ghi trên hệ thống.
- Port 0, 1,3 với các bit thứ 0 [LSB] đến bit thứ 7 [MSB] theo thứ tự từ trên xuống.
- Riêng port 2 thì ngược lại từ dưới lên. Trên bộ thí nghiệm đã có tên cho từng ngõ ra.
- Khi kết nối chú ý phải theo thứ tự bit 0 của port với bit 0 của đối tượng điều khiển.
- Khi bit thứ 0 đúng thì các bit còn lại sẽ đúng.
- Tất cả các chương trình trong hệ thống này đều được kiểm tra rất kỹ và viết đúng theo thứ tự kết nối trên.
- Nếu 1 yêu cầu nào đó không đúng thì hãy xem lại phần kết nối và chương trình.

Chú ý: khi viết chương trình thường xảy ra các lỗi như sau:

- Số không "0" thường được đánh nhầm bằng chữ o.
- Sau lệnh end thì không có hàng hay một ký tự nào nếu không thì khi biên dịch chương trình sẽ thông báo có lỗi. Lỗi này có thể bỏ qua.
- Các nhản trong chương trình phải đánh đúng như trong sách hướng dẫn.
- Các chú thích cho các lệnh thì phải nằm sau dấu chấm phẩy ";". Có phần chú thích hay không có cũng được.
- Giữa lệnh và thanh ghi phải có khoảng trắng, giữ "org " và địa chỉ phải có khoảng trắng.
- Hãy dùng nút tab để viết chương trình cho thẳng hàng để dễ xem và tìm lỗi nhanh chóng.
- Nếu đánh 1 chương trình nào đó trong tài liệu hướng dẫn mà chương trình chạy không đúng thì hãy xem kỹ lại có đánh đầy đủ tất các lệnh của chương trình hay chưa? Tất cả các chương trình trong tài liệu hướng dẫn đã được chạy thử và luôn luôn đúng.
- Khi bạn tự viết một chương trình thì sẽ có 1 số trường hợp chương trình không nạp được có thể do IC vi điều khiển hỏng thì bạn có thể nạp một chương trình nào đó đã chạy tốt, nếu vẫn không nạp được thì IC chắc chắn đã hỏng. Còn nếu nạp được và chạy tốt thì lỗi nằm ở chương trình mới viết và nguyên nhân có thể là chương trình bị xếp chồng bộ nhớ.
- Một trong những lý do làm cho chương trình nhưng không chạy là do bạn viết chương trình không bắt đầu tại địa chỉ 0000H,
- Hãy đọc kỹ các yêu cầu trước khi thực hành.



PHẦN 1:

ĐIỀU KHIỂN LED ĐƠN

(MỨC 1 LED SÁNG, MỨC 0 LED TẮT)

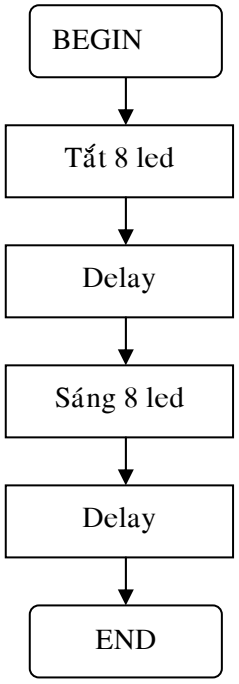
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 1-1** CHƯƠNG TRÌNH ĐIỀU KHIỂN LED CHÓP TẮT.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Nắm vững lệnh điều khiển xuất dữ liệu ra các port, biết cách viết chương trình con delay. Làm quen với phần mềm soạn thảo chương trình, cách hiệu chỉnh lỗi.

II. Trình tự thực hiện:

1. Giải thuật:



2. Kết nối mạch theo trình tự :
   - Dùng bus dây (8 sợi) kết nối port 0 với một trong bốn PINHD của dãy 32 led.
3. Khởi động phần mềm, tạo File mới, biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien 8 led chop tat ket noi voi port 0
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h           ;khai bao dia chi bat dau cua chuong trinh

lb:             mov     p0,#00h         ;nap 00 vao port0 de tat 8 led
                lcall   delay           ;goi chuong trinh con delay
                mov     p0,#0ffh        ;nap FF vao port0 de sang 8 led
                lcall   delay           ;goi chuong trinh con delay
                sjmp    lb              ;nhay den de lam lai tu dau

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

delay:          mov     r6,#0ffh        ;nap hang so FFH vao thanh ghi r6
de:             mov     r7,#0ffh        ;nap hang so FFH vao thanh ghi r7
                djnz    r7,$            ;giam r7 di 1 va nhay khi r7 khac 00
                djnz    r6,de           ;giam r6 di 1 va nhay khi r6 khac 00
                ret                     ;thoat khoi chuong trinh con
                end
```

4. Lưu chương trình và biên dịch chương trình. Kiểm tra lỗi và hiệu chỉnh rồi biên dịch lại.
5. Nạp chương trình vào vi điều khiển.
6. Quan sát kết quả hiển thị của chương trình, nếu kết quả hiển thị không đúng yêu cầu đề bài thì phải quay lại chương trình chỉnh sửa và làm lại.

III. Chương trình mẫu:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 16 led chop tat dung port0, 1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h

lb:             mov     p0,#00h         ;tat port0
                mov     p1,#00h         ;tat port1
                lcall   delay           ;delay

                mov     p0,#0ffh        ;sang 8 led
                mov     p1,#0ffh        ;sang 8 led
                lcall   delay           ;delay
                sjmp    lb              ;lam lai tu dau

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de:             mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de
                ret
                end
```

IV. Bài tập:

1. Hãy xem chương trình mẫu điều khiển 16 led chóp tắt dùng 2 port 0 và 1 và hãy viết chương trình sáng tắt 3 port 0, 1 và 3.
2. Hãy viết chương trình sáng tắt 4 port: port0, port1, port2, port3.

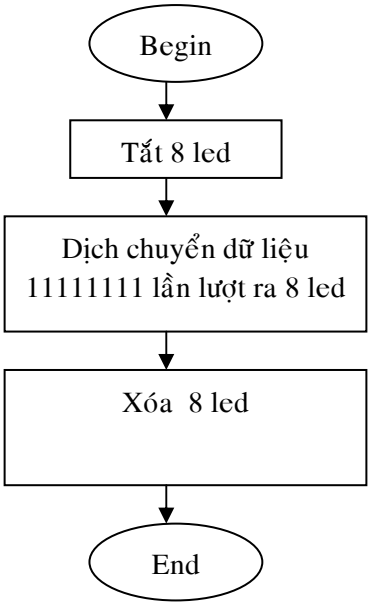
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 1-2** CHƯƠNG TRÌNH ĐIỀU KHIỂN LED SÁNG VÀ TẮT DẦN.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

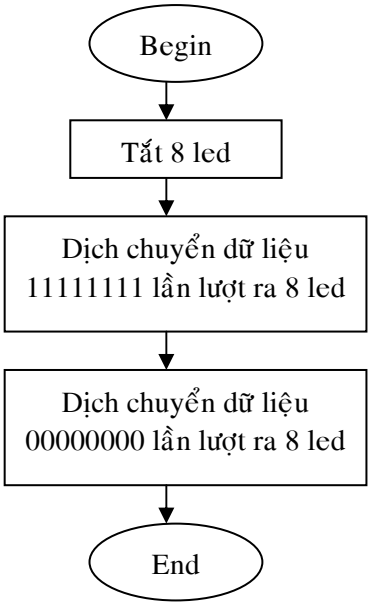
Hiểu cách sử dụng lệnh xoay 8 bit, lệnh nhảy có điều kiện để thực hiện chương trình điều khiển led sáng dần, tắt dần.

II. Trình tự thực hiện :

1. Giải thuật: sáng dần và tắt hết



sáng dần và tắt dần



2. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 0 với một trong bốn PINHD của dãy 32 led.
3. Khởi động phần mềm, tạo File mới và biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang dan len va tat het - cach I
;ket noi port 0 den 8 led bang cap 8 soi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org 0000h

lb:             mov     p0,#00000000b       ;tat port 0
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#00000001b       ;sang 1 led
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#00000011b       ;sang 2 led
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#00000111b       ;sang 3 led
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#00001111b       ;sang 4 led
                lcall   delay               ;goi chuong trinh con delay
```

```
                mov     p0,#00011111b       ;sang 5 led
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#00111111b       ;sang 6 led
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#01111111b       ;sang 7 led
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#11111111b       ;sang 8 led
                lcall   delay               ;goi chuong trinh con delay

                sjmp    lb

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de:             mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de
                ret
end
```

Trong lập trình có nhiều cách viết chương trình từ *đơn giản dễ hiểu nhưng dài dòng* đến chương trình *phức tạp khó hiểu nhưng ngắn gọn* tùy thuộc vào đối tượng nghiên cứu và đối tượng học. Ở đây trình bày luôn cả 2 cách viết.

Trong cách viết trên ta thấy chương trình dễ hiểu nhưng khá dài. Hãy cho chạy chương trình trên và xem cách viết thứ 2.

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang dan len va tat het – cach II
;ket noi port 0 den 8 led bang cap 8 soi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h

lb:             mov     p0,#00h             ;tat port 0
lb1:            lcall   delay               ;goi chuong trinh con delay
                setb    c                   ;lam cho bit C = 1
                mov     a,p0                ;chuyen noi dung port0 vao thanh ghi A
                rlc     a                   ;xoay noi dung thanh ghi A sang trai
                mov     p0,a                ;tra lai cho port0
                jnc     lb1                 ;nhay ve de thuc hien tiep
                sjmp    lb                  ;quay lai tu dau
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de :            mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de
                ret
                end
```

Giải thích : để led sáng dần lên ta phải dịch mức 1 vào thanh ghi A, mức 1 được chứa trong bit Cy, lệnh xoay thanh ghi A sang trái sẽ dịch mức 1 từ C vào bit A0 của thanh ghi A. Bit A7 sẽ dịch sang bit Cy.

Trong 8 lần dịch đầu tiên thì sau khi dịch, bit Cy luôn bằng 0. Nên ta dùng lệnh nhảy có điều kiện khi C = 0 thì nhảy để quay lại tiếp tục thực hiện.

Cho đến lần xoay thứ 9 thì C = 1 thì điều kiện không còn thõa mãn nên lệnh nhảy có điều kiện thì lệnh nhảy "sjmp " mới được thực hiện để làm lại từ đầu.

4. Thực hiện các bước giống như các bài trước.

III. Các chương trình mẫu:

1. Chương trình điều khiển port 0 sáng dần và tắt dần:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang dan len va tat dan
;ket noi port 0 den 8 led bang cap 8 soi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h
lb:             mov     p0,#00h             ;tat port 0

lb1:            lcall   delay               ;goi chuong trinh con delay
                setb    c                   ;lam cho bit C = 1
                mov     a,p0                ;chuyen noi dung port0 vao thanh ghi A
                rlc     a                   ;xoay noi dung thanh ghi A sang trai
                mov     p0,a                ;tra lai cho port0
                jnc     lb1                 ;nhay ve de thuc hien tiep khi c=0

lb2:            lcall   delay               ;goi chuong trinh con delay
                clr     c                   ;lam cho bit C = 0
                mov     a,p0                ;chuyen noi dung port0 vao thanh ghi A
                rlc     a                   ;xoay noi dung thanh ghi A sang trai
                mov     p0,a                ;tra lai cho port0
                jc      b2                  ;nhay ve de thuc hien tiep khi c=1

                sjmp    lb                  ;quay ve lam lai tu dau

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de:             mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de
                ret
                end
```

2. Chương trình điều khiển port 0 và port 1 sáng dần và tắt dần:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0, 1 sang dan len va tat dan
;ket noi port 0 den 8 led bang cap 8 soi tuong tu cho port1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h
lb:             mov     p0,#00h             ;tat port 0
                mov     p1,#00h             ;tat port 1

lb1:            lcall   delay               ;goi chuong trinh con delay
                setb    c                   ;lam cho bit C = 1
                mov     a,p0                ;chuyen noi dung port0 vao thanh ghi A
                rlc     a                   ;xoay noi dung thanh ghi A sang trai
                mov     p0,a                ;tra lai cho port0

                mov     a,p1                ;chuyen noi dung port1 vao thanh ghi A
                rlc     a                   ;xoay noi dung thanh ghi A sang trai
                mov     p1,a                ;tra lai cho port1
```

```
                jnc     lb1                 ;nhay ve de thuc hien tiep khi c=0

lb2:            lcall   delay               ;goi chuong trinh con delay
                clr     c                   ;lam cho bit C = 0
                mov     a,p0                ;chuyen noi dung port0 vao thanh ghi A
                rlc     a                   ;xoay noi dung thanh ghi A sang trai
                mov     p0,a                ;tra lai cho port0

                mov     a,p1                ;chuyen noi dung port1 vao thanh ghi A
                rlc     a                   ;xoay noi dung thanh ghi A sang trai
                mov     p1,a                ;tra lai cho port1

                jc      lb2                 ;nhay ve de thuc hien tiep khi c=1

                sjmp    lb                  ;quay ve lam lai tu dau

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de:             mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de
                ret
                end
```

IV. Bài tập ứng dụng:

1. Dùng port 0 kết nối với 8 led, hãy viết chương trình điều khiển 1 led sáng và di chuyển từ trái sang phải.
2. Hãy viết chương trình điều khiển 4 port 0, 1, 2, 3 sáng dần và tắt dần từ trên xuống và từ dưới lên.
3. Hãy viết chương trình điều khiển 4 port 0, 1, 2, 3 sáng dần và tắt dần từ ngoài vào trong và từ trong ra ngoài.

THỰC HÀNH VI ĐIỀU KHIỂN

BÀI SỐ : 1-3

CHƯƠNG TRÌNH ĐIỀU KHIỂN LED SÁNG DỒN

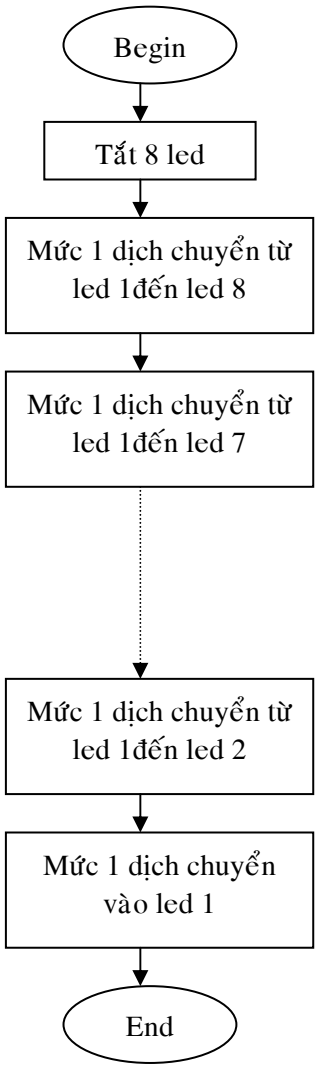
NGÀY :
SỐ TIẾT :
LỚP :
MSSV :

I. Mục đích yêu cầu:

Hiểu cách sử dụng lệnh xoay kết hợp với lệnh logic để thực hiện chương trình điều khiển led sáng dồn.

II. Trình tự thực hiện:

1. Giải thuật:



2. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 0 với một trong bốn PINHD của dãy 32 led.
3. Khởi động phần mềm, tạo File mới, và biên soạn chương trình sau :

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang don va tat het - cach I
;ket noi port 0 den 8 led bang cap 8 soi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0800h           ;khai bao dia chi luu tru vung du lieu
        ma:     db      00000000b
                db      00000001b
```

```
                db      00000010b
                db      00000100b
                db      00001000b
                db      00010000b
                db      00100000b
                db      01000000b
                db      10000000b
;lan thu hai la 7 byte
                db      10000001b
                db      10000010b
                db      10000100b
                db      10001000b
                db      10010000b
                db      10100000b
                db      11000000b
;lan thu 3 la 6 byte
                db      11000001b
                db      11000010b
                db      11000100b
                db      11001000b
                db      11010000b
                db      11100000b
;lan thu 4 la 5 byte
                db      11100001b
                db      11100010b
                db      11100100b
                db      11101000b
                db      11110000b
;lan thu 5 la 4 byte
                db      11110001b
                db      11110010b
                db      11110100b
                db      11111000b
;lan thu 6 la 3 byte
                db      11111001b
                db      11111010b
                db      11111100b
;lan thu 7 la 2 byte
                db      11111101b
                db      11111110b
;lan thu 8 la 1 byte
                db      11111111b       ;byte du lieu cuoi cung = FFH
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h           ;khai bao dia chi chtr chinh

lb:             mov     dptr,#0800h     ;nap dia chi luu du lieu vao thghi dptr
lb1:            clr     a
                movc    a,@A+dptr       ;lay du lieu tu bo nho dua vao A
                mov     p0,a            ;goi ra port 0
                lcall   delay           ;goi chtr con delay
                inc     dptr            ;tang dptr len o nho ke
                cjne    a,#0ffh,lb1     ;ktra co phai la byte ket thuc hay chua
                sjmp    lb              ;quay tro lam lai khi da het du lieu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de2 :           mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de
                ret
                end
```

Theo cách viết 1 ta hãy quan sát dữ liệu trong chương trình đã được sắp xếp theo đúng trình tự và chương trình chỉ thực hiện nhiệm vụ là di chuyển lần lược các byte dữ liệu có trong bộ nhớ đem gởi vào A và sau đó gởi ra port 0.

Lệnh "ma: db dữ liệu " có chức năng nạp các byte dữ liệu vào vùng nhớ có địa chỉ 0800H.

Byte cuối cùng là FFH là byte báo cho biết hết d74 liệu.

Dữ liệu viết dưới dạng số nhị phân cho dễ nhìn thấy và có thể viết dưới dạng số hex – khi đó chương trình sẽ ngắn hơn rất nhiều. Phần khai báo dữ liệu dưới dạng số hex như sau:

```
        org     0800h           ;khai bao dia chi luu tru vung du lieu
ma:     db      00H
        DB      01H,02H,04H,08H,10H,20H,40H,80H     ;
        DB      81H,82H,84H,88H,90H,0A0H,0C0H
        DB      0C1H,0C2H,0C4H,0C8H,0D0H,0E0H
        DB      0E1H,0E2H,0E4H,0E8H,0F0H
        DB      0F1H,0F2H,0F4H,0F8H
        DB      0F9H,0FAH,0FCH
        DB      0FDH,0FEH
        DB      0FFH
```

Chương trình giống như trên nhưng viết theo cách II:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien 8 led sang don dung port 0
;ket noi port 0 den 8 led bang 1 soi cap 8 soi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;dinh nghia cac bien
        x0      equ     10h
        y0      equ     20h

        dem     equ     30h
        tam     equ     31h

        led0    equ     p0
        led1    equ     p1
        led2    equ     p2
        led3    equ     p3

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;bat dau chuong trinh chinh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        org     0000h
        mov     led2,#0             ;tat cac led chua su dung
        mov     led3,#0
        mov     led1,#0

lb:     mov     led0,#0
        lcall   delay               ;tat 16 led va delay

        mov     dem,#8              ;led 1 se di chuyen 8 vi tri
        mov     x0,#0

lb2:    mov     tam,dem
        mov     y0,#00000001b       ;luu trng thai ban dau

lb1:    mov     a,y0
        orl     a,x0
        mov     led0,a

        lcall   delay
```

```
                clr     c                       ;xoa Cy de chi dich 1 led di
                mov     a,y0
                rlc     a
                mov     y0,a

                djnz    tam,lb1                 ;giam ndung o nho (11h)<> 0 thi quay lai
                mov     x0,led0

                djnz    dem,lb2                 ;giam bien dem de xu li lan ke
                ljmp    lb

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

Chương trình không khó !, bạn hãy tự nghiên cứu giải thuật?

III. Các chương trình mẫu:

Chương trình điều khiển 16 led sáng dồn:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien 16 led sang don dung port 0, 1
;ket noi port 0 va port 1 den 16 led bang 2 soi cap 8 soi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;dinh nghia cac bien
                x0      equ     10h
                x1      equ     11h

                y0      equ     20h
                y1      equ     21h

                dem     equ     30h
                tam     equ     31h

                led0    equ     p0
                led1    equ     p1
                led2    equ     p2
                led3    equ     p3

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;bat dau chuong trinh chinh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h
                mov     led2,#0         ;tat cac led chua su dung
                mov     led3,#0

lb:             mov     led1,#0
                mov     led0,#0
                lcall   delay           ;tat 16 led va delay

                mov     dem,#16                 ;led 1 se di chuyen 16 vi tri

                mov     x0,#0
                mov     x1,#0
```

```
lb2:        mov     tam,dem
            mov     y0,#00000001b ;luu trng thai ban dau
            mov     y1,#00000000b

lb1:        mov     a,y0
            orl     a,x0
            mov     led0,a

            mov     a,y1
            orl     a,x1
            mov     led1,a

            lcall   delay

            clr     c                   ;xoa Cy de chi dich 1 led di
            mov     a,y0
            rlc     a
            mov     y0,a

            mov     a,y1
            rlc     a
            mov     y1,a
            djnz    tam,lb1             ;giam ndung o nho (11h)<> 0 thi quay lai

            mov     x0,led0
            mov     x1,led1
            djnz    dem,lb2             ;giam bien dem de xu li lan ke
            ljmp    lb

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

IV. Bài tập ứng dụng:

1. Hãy viết chương trình điều khiển 3 port: port0, port1, port2 sáng dồn.
2. Hãy viết chương trình điều khiển 4 port: port0, port1, port2 và port3 sáng dồn.
3. Hãy viết chương trình sáng dồn 2 port 0 và 1 từ ngoài vào trong và từ trong ra ngoài.
4. Hãy viết chương trình sáng dồn 4 port 0, 1, 2 và 3 từ ngoài vào trong và từ trong ra ngoài.

THỰC HÀNH VI ĐIỀU KHIỂN

BÀI SỐ : 1-4

CHƯƠNG TRÌNH ĐIỀU KHIỂN
ĐIỂM SÁNG DỊCH CHUYỂN MẤT DẦN

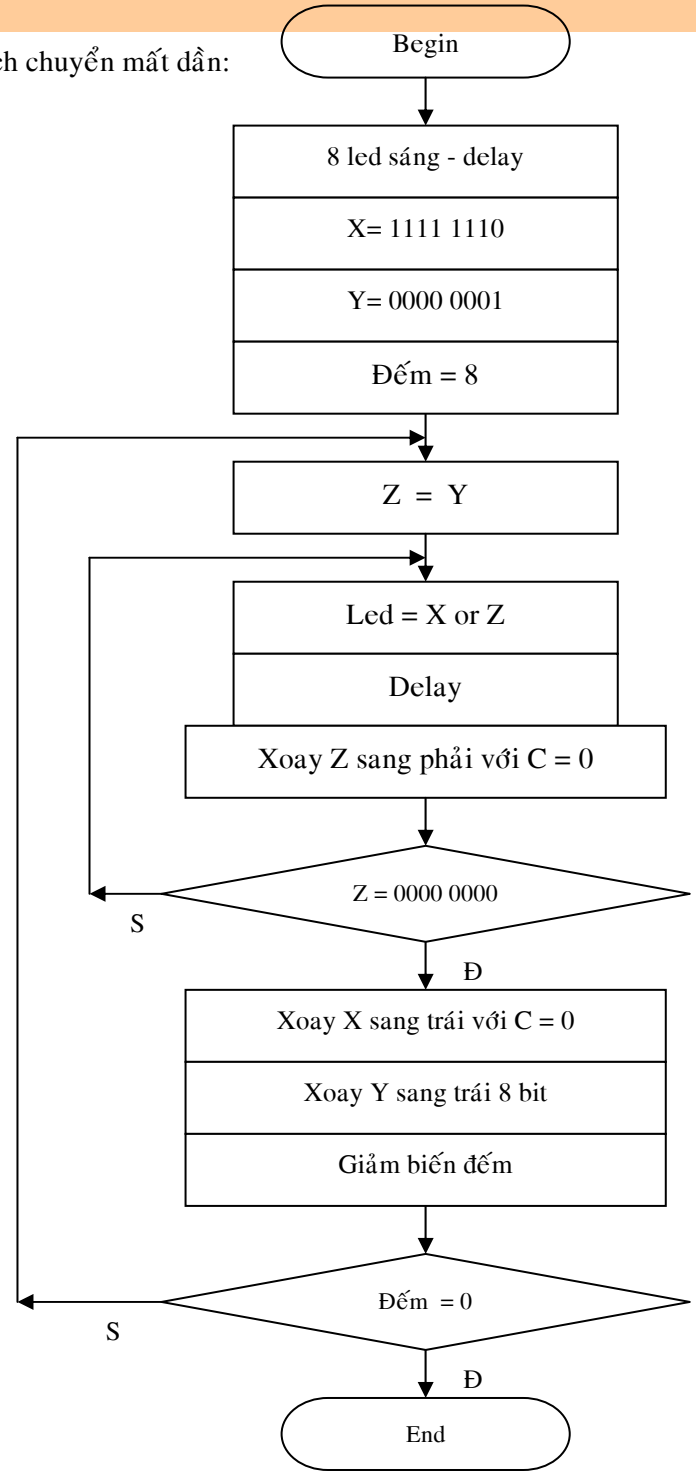
NGÀY :
SỐ TIẾT :
LỚP :
MSSV :

I. Mục đích yêu cầu:

Hiểu cách sử dụng lệnh xoay kết hợp với lệnh logic để thực hiện chương trình điều khiển led làm quen với lập trình.

II. Trình tự thực hiện:

1. Giải thuật điều khiển 8 led dịch chuyển mất dần:



2. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 0 với một trong bốn PINHD của dãy 32 led.
3. Khởi động phần mềm, tạo File mới và biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien 8 led sang het va diem sang dich chuyen tat dan
;su dung 1 port 0
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;dinh nghia cac nhan

                x0      equ     11h
                y0      equ     21h
                z0      equ     r0

                led0    equ     p0
                led1    equ     p1
                led2    equ     p2
                led3    equ     p3


                dem     equ     40h

                org     0000h
                mov     led2,#0
                mov     led3,#0
                mov     led1,#0h

main:           mov     led0,#0ffh
                lcall   delay
                mov     dem,#8

                mov     x0,#11111110b
                mov     y0,#00000001b

m2:             mov     z0,y0

m1:             lcall   xoay_z
                lcall   x_or_z
                lcall   delay

                cjne    z0,#00,m1
                lcall   xoay_x
                lcall   xoay_y
                djnz    dem,m2
                ljmp    main
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;bat dau cac chuong trinh con
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xoay_z:         clr     c
                mov     a,z0
                rrc     a
                mov     z0,a
                ret

x_or_z:         mov     a,x0
                orl     a,z0
                mov     led0,a
                ret
```

```
xoay_x:     clr     c
            mov     a,x0
            rlc     a
            mov     x0,a
            ret

xoay_y:     clr     c
            mov     a,y0
            rlc     a
            mov     y0,a
            ret

delay:      mov     r6,#0ffh
de:         mov     r7,#0ffh
            djnz    r7,$
            djnz    r6,de
            ret
           end
```

III. Các chương trình mẫu:

Chương trình mẫu điều khiển 16 led dịch chuyển tắt dần

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien 16 led sang het va diem sang dich chuyen tat dan
;su dung 2 port 0 va port 1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;dinh nghia cac nhan

            x1      equ     10h
            x0      equ     11h

            y1      equ     20h
            y0      equ     21h

            z1      equ     r1
            z0      equ     r0

            led0    equ     p0
            led1    equ     p1
            led2    equ     p2
            led3    equ     p3


            dem     equ     40h

            org     0000h
            mov     led2,#0
            mov     led3,#0


main:       mov     led1,#0ffh
            mov     led0,#0ffh
            lcall   delay
            mov     dem,#16

            mov     x1,#11111111b       ;byte cao
            mov     x0,#11111110b       ;byte thap

            mov     y1,#00000000b       ;byte cao
```

```
                mov     y0,#00000001b           ;byte thap

m2:             mov     z1,y1
                mov     z0,y0

m1:             lcall   xoay_z
                lcall   x_or_z

                lcall   delay

                cjne    z0,#00,m1
                cjne    z1,#00,m1
                lcall   xoay_x
                lcall   xoay_y

                djnz    dem,m2
                ljmp    main

xoay_z:         clr     c
                mov     a,z1
                rrc     a
                mov     z1,a

                mov     a,z0
                rrc     a
                mov     z0,a
                ret

x_or_z:         mov     a,x0
                orl     a,z0
                mov     led0,a

                mov     a,x1
                orl     a,z1
                mov     led1,a
                ret

xoay_x:         clr     c
                mov     a,x0
                rlc     a
                mov     x0,a

                mov     a,x1
                rlc     a
                mov     x1,a
                ret

xoay_y:         clr     c
                mov     a,y0
                rlc     a
                mov     y0,a

                mov     a,y1
                rlc     a
                mov     y1,a
                ret

delay:          mov     r6,#0ffh
de:             mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de
                ret
```

```
end
```

IV. Bài tập:

1. Hãy viết chương trình điều khiển 3 port: 0, 1, 2 giống như trên.
2. Hãy viết chương trình điều khiển 4 port: 0, 1, 2, 3 giống như trên.
3. Hãy viết chương trình điều khiển 4 port 0, 1, 2, 3 với điểm sáng dịch chuyển mất dần từ theo chiều từ trong ra và từ ngoài vào.

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 1-5** CHƯƠNG TRÌNH DELAY SỬ DỤNG TIMER	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách tính toán các thông số delay của timer để viết các chương trình delay chính xác.

II. 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.
2. Khởi động phần mềm, tạo File mới và biên soạn chương trình sau

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

        org     0000h
        mov     tmod,#01        ; khoi tao timer T0 mode 1 dem 16 bit
        setb    tr0             ; cho phep timer 0 bat dau dem xung

b61:    mov     p1,#00h
        lcall   delay           ; delay 65536 micro giay
        mov     p1,#0ffh
        lcall   delay
        sjmp    b61
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con delay 65535 micro giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:  clr     tf0             ; xoa co ngat cua timer 0
        mov     tl0,#0          ; nap 0 vao TL0
        mov     th0,#0          ; nap 0 vao TH0
del1 :  jnb     tf0,del1        ; kiem tra co tran
        ret
        end
```

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

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.

Hàng lệnh đầu tiên trong chương trình chính là chọn mode làm việc cho timer T0 – hãy xem chương timer (timer T1 chưa sử dụng nên không cần quan tâm).

Lệnh thứ 2 là cho phép timer bắt đầu đếm xung nội từ mạch dao động bên trong.

Với chương trình trên thì timer T0 sẽ đếm từ gía trị nạp ban đầu 0000H đến FFFFH và khi có thêm 1 xung nữa thì giá trị đếm sẽ là 10000H. Một cách đơn giản ta có thể xem con số 10000H được chia ra làm 2: số "1" được lưu trong cờ tràn TF0 và số "0000H" được lưu trong 2 thanh ghi TH0TL0. 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.

Trong chương trình con có 2 lệnh nạp lại giá trị cho TH0 và TL0 của bài này là dư vì khi timer bị tràn thì nó tự động là cho 2 thanh ghi trên mang giá trị 0.

III. Các chương trình mẫu:

Để đếm lượng thời gian nhỏ hơn ta hãy tham khảo bài mẫu dưới đây:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh sang tat port1 su dung timer lam bo dinh thoi delay 250 micro giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

        org     0000h
        mov     tmod,#01            ; khoi tao timer T0 mode 1 dem 16 bit
        setb    tr0                 ; cho phep timer 0 bat dau dem xung

b62:    mov     p1,#00h
        lcall   delay               ; delay 250 micro giay
        mov     p1,#0ffh
        lcall   delay
        sjmp    b62
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con delay 250 micro giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

delay:  clr     tf0
        mov     tl0,#low(-250)      ; nap 05 vao TL0
        mov     th0,#high(-250)     ; nap FF vao TH0
del1 :  jnb     tf0,del1
        ret
        end
```

Giải thích :

Lượng thời gian của bài này chỉ có 250 micro giây, vì giá trị ban đầu nạp cho timer là FF05H nên khi đếm đến 10000H kết quả xung đếm được 10000H – FF06H = 00FA (250) và mỗi xung có chu kỳ 1 micro giây nên lượng thời gian mà timer T0 đếm đượclà 250 micro giây.

Để khỏi phải tính toán phức tạp ta có thể viết bằng hai lệnh như sau:

```
        mov     tl0,#low(-250)      ; nap 05 vao TL0
        mov     th0,#high(-250)     ; nap FF vao TH0
```

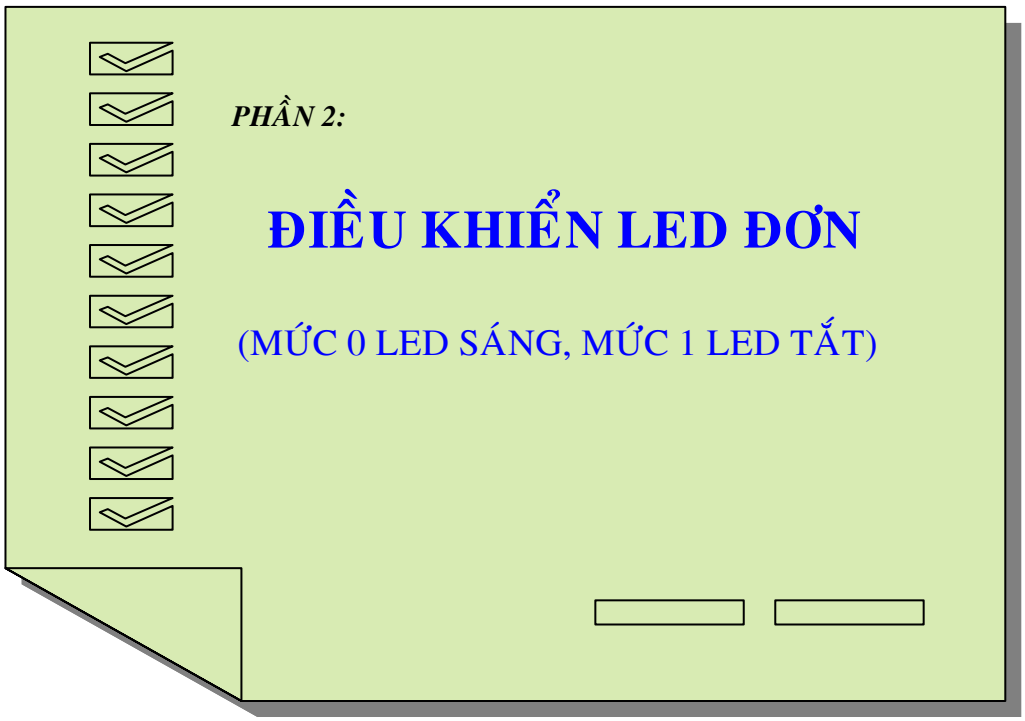
Khi biên dịch thì trình biên dịch tự động tính toán cho chúng ta.

Chú ý với các chương trình điều khiển led sáng với thời gian trể nhỏ thì led sáng mờ nhưng không chóp tắt như các bài ta đã viết ở trên. Trong phần tính toán chúng ta chưa tính toán các lệnh trong chương trình con delay.

Muốn viết chương trình với các khoảng thời gian lớn hơn thì phải thêm thanh ghi ví dụ muốn viết delay 5 giây thì ta viết chương trình trình con delay 50 000 µs = 50ms và cho chúng thực hiện 100 lần – sau này chúng ta sẽ dùng timer để tạo ra các xung chính xác về thời gian cho các bài sau.

IV. Bài tập:

1. Hãy viết chương trình sáng tắt port 2 sử dụng timer làm bộ định thời delay 5 giây.
2. Hãy viết chương trình giống trên nhưng delay 10 giây.
3. Tương tự hãy viết chương trình delay 1 giờ.



PHẦN 2:

ĐIỀU KHIỂN LED ĐƠN

(MỨC 0 LED SÁNG, MỨC 1 LED TẮT)

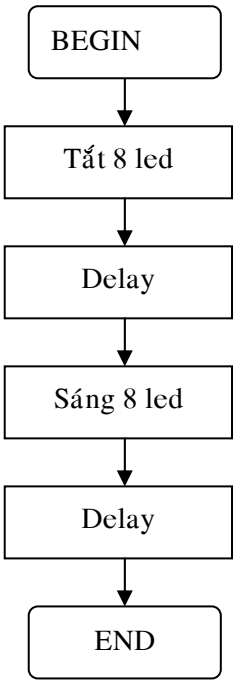
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 2-1** CHƯƠNG TRÌNH ĐIỀU KHIỂN LED CHÓP TẮT.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Nắm vững lệnh điều khiển xuất dữ liệu ra các port, biết cách viết chương trình con delay. Làm quen với phần mềm soạn thảo chương trình, cách hiệu chỉnh lỗi.

II. Trình tự thực hiện:

1. Giải thuật:



2. Kết nối mạch theo trình tự :
   - Dùng bus dây (8 sợi) kết nối port 0 với một trong bốn PINHD của dãy 32 led.
3. Khởi động phần mềm, tạo File mới, và biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang tat
;muc 0 led sang - muc 1 led tat
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h                   ;khai bao dia chi bat dau cua chuong trinh

main:           mov     p0,#00000000b           ;sang 8 led
                lcall   delay                   ;goi chuong trinh con delay
                mov     p0,#11111111b           ;tat 8 led
                lcall   delay                   ;goi chuong trinh con delay
                sjmp    main                    ;lam lai

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de2 :           mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de2
                ret
end
```

4. Lưu chương trình và biên dịch chương trình. Kiểm tra lỗi và hiệu chỉnh rồi biên dịch lại.
5. Nạp chương trình vào vi điều khiển.
6. Quan sát kết quả hiển thị của chương trình, nếu kết quả hiển thị không đúng yêu cầu đề bài thì phải quay lại chương trình chỉnh sửa .

III. Các chương trình mẫu:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 va 1 sang tat
;muc 0 led sang - muc 1 led tat
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h                   ;khai bao dia chi bat dau cua chuong trinh

main:           mov     p0,#00000000b           ;sang 8 led
                mov     p1,#0
                lcall   delay                   ;goi chuong trinh con delay
                mov     p0,#11111111b           ;tat 8 led
                mov     p1,#0ffh
                lcall   delay                   ;goi chuong trinh con delay
                sjmp    main                    ;lam lai

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de2 :           mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de2
                ret
                end
```

IV. Bài tập:

1. Hãy xem chương trình mẫu điều khiển 16 led chóp tắt dùng 2 port 0 và 1 và hãy viết chương trình sáng tắt 3 port 0, 1 và 2.
2. Hãy viết chương trình sáng tắt 4 port 0, 1, 2, 3.

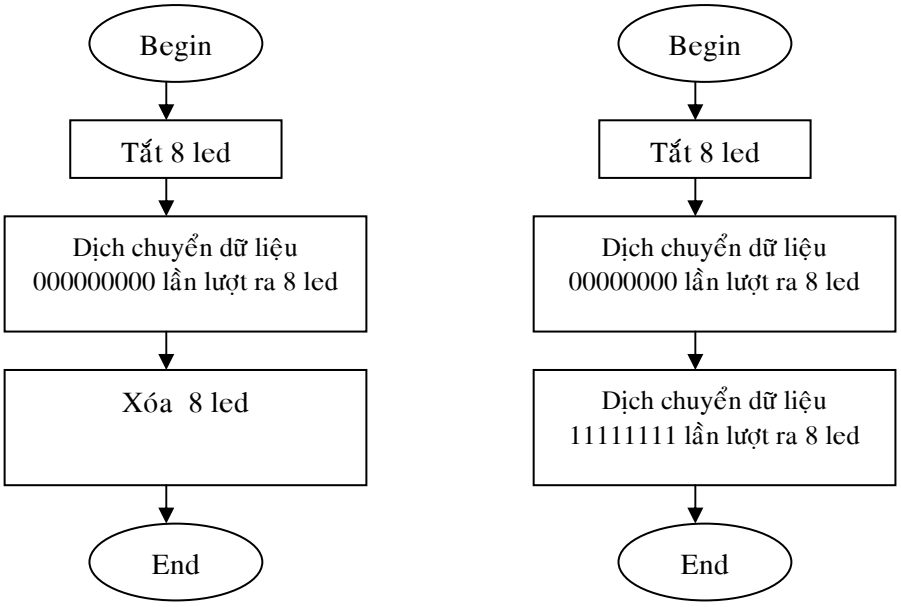
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 2-2** CHƯƠNG TRÌNH ĐIỀU KHIỂN LED SÁNG VÀ TẮT DẦN.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Hiểu cách sử dụng lệnh xoay 8 bit, lệnh nhảy có điều kiện để thực hiện chương trình điều khển led sáng dần, tắt dần.

II. Trình tự thực hiện :

1. Giải thuật: sáng dần và tắt hết sáng dần và tắt dần



2. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 0 với một trong bốn PINHD của dãy 32 led.
3. Khởi động phần mềm, tạo File mới và biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang dan len va tat het - cach 1
;truong dai hoc su pham ky thuat-nguyen dinh phu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org 0000h

lb:             mov     p0,#11111111b       ;tat port 0
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#11111110b       ;sang 1 led
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#11111100b       ;sang 2 led
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#11111000b       ;sang 3 led
                lcall   delay               ;goi chuong trinh con delay

                mov     p0,#11110000b       ;sang 4 led
                lcall   delay               ;goi chuong trinh con delay
```

```
                mov     p0,#11100000b           ;sang 5 led
                lcall   delay                   ;goi chuong trinh con delay

                mov     p0,#11000000b           ;sang 6 led
                lcall   delay                   ;goi chuong trinh con delay

                mov     p0,#10000000b           ;sang 7 led
                lcall   delay                   ;goi chuong trinh con delay

                mov     p0,#00000000b           ;sang 8 led
                lcall   delay                   ;goi chuong trinh con delay

                sjmp    lb

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de:             mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de
                ret
        end
```

Trong lập trình có nhiều cách viết chương trình từ *đơn giản dễ hiểu nhưng dài dòng* đến chương trình *phức tạp khó hiểu nhưng ngắn gọn* tùy thuộc vào đối tượng nghiên cứu và đối tượng học. Ở đây trình bày luôn cả 2 cách viết.

Trong cách viết trên ta thấy chương trình dễ hiểu nhưng khá dài. Hãy cho chạy chương trình trên và xem cách viết thứ 2.

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang dan len va tat het
;truong dai hoc su pham ky thuat-nguyen dinh phu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h

lb:             mov     p0,#0ffh                ;tat port 0
lb1:            lcall   delay           ;goi chuong trinh con delay
                clr     c               ;lam cho bit C = 1
                mov     a,p0            ;chuyen noi dung port0 vao thanh ghi A
                rlc     a               ;xoay noi dung thanh ghi A sang trai
                mov     p0,a
                jc      lb1             ;nhay ve de thuc hien tiep
                sjmp    lb              ;sau khi 8 led sang het thi quay lai tu dau

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de2 :           mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de2
                ret
end
```

Giải thích : để led sáng dần lên ta phải đưa dữ liệu P0 sang thanh ghi A rồi dịch mức 0 chứa trong cờ C vào thanh ghi A bằng lệnh xoay. Bit A_7 sẽ dịch sang bit C.

Trong 8 lần dịch đầu tiên thì sau khi dịch, bit C luôn bằng 1. Nên ta dùng lệnh nhảy có điều kiện khi C = 1 thì nhảy để quay lại tiếp tục thực hiện.

Cho đến lần xoay thứ 9 thì C = 0 thì điều kiện không còn thõa mãn nên lệnh nhảy có điều kiện thì lệnh nhảy "sjmp " mới được thực hiện để làm lại từ đầu.

4. Thực hiện các bước giống như các bài trước.

III. Các chương trình mẫu:

1. Chương trình điều khiển port 0 sáng dần và tắt dần:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang dan len va tat dan
;truong dai hoc su pham ky thuat-nguyen dinh phu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org 0000h

lb:             mov     p0,#0ffh        ;tat port 0
lb1:            lcall   delay           ;goi chuong trinh con delay
                clr     c               ;lam cho bit C = 1
                mov     a,p0            ;chuyen noi dung port0 vao thanh ghi A
                rlc     a               ;xoay noi dung thanh ghi A sang trai
                mov     p0,a            ;tra lai cho port0
                jc      lb1             ;nhay ve de thuc hien tiep khi c=0

lb2:            lcall   delay           ;goi chuong trinh con delay
                mov     a,p0            ;chuyen noi dung port0 vao thanh ghi A
                setb    c               ;lam cho bit C = 0
                rlc     a               ;xoay noi dung thanh ghi A sang trai
                mov     p0,a            ;tra lai cho port0
                jnc     lb2             ;nhay ve de thuc hien tiep khi c=1

                sjmp    lb              ;quay ve lam lai tu dau
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
del2 :          mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,del2
                ret
end
```

2. Chương trình điều khiển port 0 và port 1 sáng dần và tắt dần:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang dan len va tat dan
;truong dai hoc su pham ky thuat-nguyen dinh phu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org 0000h

lb:             mov     p0,#0ffh                ;tat port 0
                mov     p1,#0ffh

lb1:            lcall   delay           ;goi chuong trinh con delay
                clr     c               ;lam cho bit C = 1
                mov     a,p0            ;chuyen noi dung port0 vao thanh ghi A
                rlc     a               ;xoay noi dung thanh ghi A sang trai
                mov     p0,a            ;tra lai cho port0

                mov     a,p1            ;chuyen noi dung port0 vao thanh ghi A
                rlc     a               ;xoay noi dung thanh ghi A sang trai
                mov     p1,a            ;tra lai cho port0
                jc      lb1             ;nhay ve de thuc hien tiep khi c=0
```

```
lb2:        lcall   delay       ;goi chuong trinh con delay
            setb    c           ;lam cho bit C = 0
            mov     a,p0        ;chuyen noi dung port0 vao thanh ghi A
            rlc     a           ;xoay noi dung thanh ghi A sang trai
            mov     p0,a        ;tra lai cho port0

            mov     a,p1        ;chuyen noi dung port0 vao thanh ghi A
            rlc     a           ;xoay noi dung thanh ghi A sang trai
            mov     p1,a        ;tra lai cho port0
            jnc     lb2         ;nhay ve de thuc hien tiep khi c=1

            sjmp    lb          ;quay ve lam lai tu dau
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:      mov     r6,#080h
del2 :      mov     r7,#0ffh
            djnz    r7,$
            djnz    r6,del2
            ret
end
```

IV. Bài tập:

1. Dùng port 0 kết nối với 8 led, hãy viết chương trình điều khiển 1 led sáng và di chuyển từ trái sang phải.
2. Hãy viết chương trình điều khiển 4 port 0, 1, 2, 3 sáng dần và tắt dần từ trên xuống và từ dưới lên.
3. Hãy viết chương trình điều khiển 4 port 0, 1, 2, 3 sáng dần và tắt dần từ ngoài vào trong và từ trong ra ngoài.

THỰC HÀNH VI ĐIỀU KHIỂN

BÀI SỐ : 2-3

CHƯƠNG TRÌNH ĐIỀU KHIỂN LED SÁNG DỒN

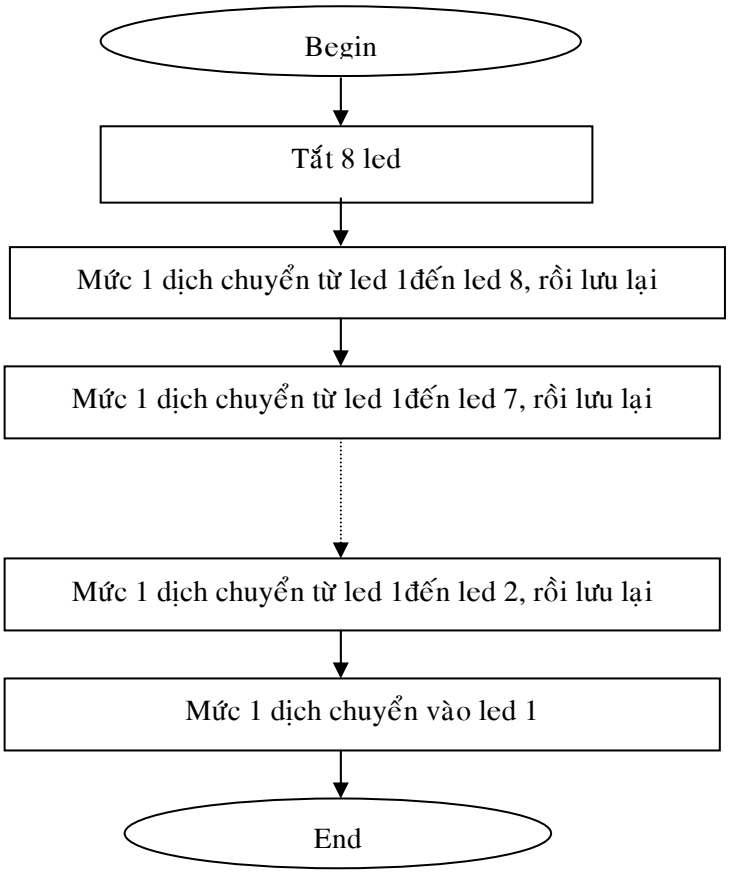
NGÀY :
SỐ TIẾT :
LỚP :
MSSV :

I. Mục đích yêu cầu:

Hiểu cách sử dụng lệnh xoay kết hợp với lệnh logic để thực hiện chương trình điều khiển led sáng dồn.

II. Trình tự thực hiện:

1. Giải thuật:



2. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 0 với một trong bốn PINHD của dãy 32 led.
3. Khởi động phần mềm, mở File mới và đặt tên file.
4. Viết chương trình với tên file vừa đặt :

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang don va tat het - cach I
;truong dai hoc su pham ky thuat-nguyen dinh phu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao dia chi luu tru vung du lieu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

        org     0800h
ma:     db      11111111b
        db      11111110b
```

```
                db      11111101b
                db      11111011b
                db      11110111b
                db      11101111b
                db      11011111b
                db      10111111b
                db      01111111b
;lan thu hai la 7 byte
                db      01111110b
                db      01111101b
                db      01111011b
                db      01110111b
                db      01101111b
                db      01011111b
                db      00111111b
;lan thu 3 la 6 byte
                db      00111110b
                db      00111101b
                db      00111011b
                db      00110111b
                db      00101111b
                db      00011111b
;lan thu 4 la 5 byte
                db      00011110b
                db      00011101b
                db      00011011b
                db      00010111b
                db      00001111b
;lan thu 5 la 4 byte
                db      00001110b
                db      00001101b
                db      00001011b
                db      00000111b
;lan thu 6 la 3 byte
                db      00000110b
                db      00000101b
                db      00000011b
;lan thu 7 la 2 byte
                db      00000010b
                db      00000001b

;lan thu 8 la 1 byte
                db      00000000b       ;byte du lieu cuoi cung = 00H de ket thuc
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh chinh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h           ;khai bao dia chi chtr chinh

lb:             mov     dptr,#0800h     ;nap dia chi luu du lieu vao thghi dptr
lb1:            clr     a
                movc    a,@A+dptr       ;lay du lieu tu bo nho dua vao A
                mov     p0,a            ;goi ra port 0

                lcall   delay           ;goi chtr con delay
                inc     dptr            ;tang dptr len o nho ke
                cjne    a,#000h,lb1     ;ktra co phai la byte ket thuc hay chua

                sjmp    lb              ;quay tro lam lai tu dau khi da het du lieu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
```

```
de2 :       mov     r7,#0ffh
            djnz    r7,$
            djnz    r6,de2
            ret
end
```

5. Thực hiện các bước từ 5 đến 9 giống như các bài trước.

Theo cách viết 1 ta hãy quan sát dữ liệu trong chương trình đã được sắp xếp theo đúng trình tự và chương trình chỉ thực hiện nhiệm vụ là di chuyển lần lược các byte dữ liệu có trong bộ nhớ đem gởi vào A và sau đó gởi ra port 0.

Lệnh "ma: db dữ liệu " có chức năng nạp các byte dữ liệu vào vùng nhớ có địa chỉ 0800H.

Dữ liệu viết dưới dạng số nhị phân cho dễ nhìn thấy và có thể viết dưới dạng số hex – khi đó chương trình sẽ ngắn hơn rất nhiều. Phần khai báo dữ liệu dưới dạng số hex như sau:

Chương trình giống như trên nhưng viết theo cách II:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang don va tat het
;truong dai hoc su pham ky thuat-nguyen dinh phu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            biendem     equ     30H     ;dinh nghia cac bien
            bientamequ      31h

            x0      equ     r0
            y0      equ     20h
            led0    equ     p0

            org     0000h

            mov     led0,#0ffh
            lcall   delay

lb:         mov     x0,#11111111b       ;X0 luu trng thai ban dau
            mov     biendem,#08         ;bien dem so lan dich chuyen LAN DAU = 8


lb2:        mov     bientam,biendem     ;chuyen biendem sang bientam
            mov     y0,#11111110b

lb1:        mov     a,y0
            anl     a,x0                ;lay x0 or voi y0 roi goi ra led0
            mov     led0,a              ;xuat ra led
            lcall   delay
            setb    c
            mov     a,y0
            rlc     a
            mov     y0,a

            djnz    bientam,lb1         ;giam bientam neu chua bang 0 thi quay ve lai

            mov     x0,led0             ;cat noi dung sau cung khi da dich chuyen 1 led
            djnz    biendem,lb2         ;giam bien dem de xu li lan ke
            sjmp    lb                  ;nhay ve lam lai tu dau

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
delay:      mov     r6,#0ffh
del2 :      mov     r7,#0ffh
            djnz    r7,$
            djnz    r6,del2
            ret
      end
```

Chương trình không khó!, bạn hãy tự nghiên cứu thử xem sao?

III. Các chương trình mẫu:

Chương trình điều khiển 2 port sang don va tat het:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien 2 port  sang don va tat het
;truong dai hoc su pham ky thuat-nguyen dinh phu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            biendem     equ     30h     ;dinh nghia cac bien
            bientamequ  31h

            x0      equ     r0
            x1      equ     r1
            y0      equ     20h
            y1      equ     21h

            led0    equ     p0
            led1    equ     p1

            org     0000h

            mov     led0,#0ffh
            mov     led1,#0ffh
            lcall   delay

lb:         mov     x0,#11111111b       ;X0 luu trng thai ban dau
            mov     x1,#11111111b       ;X1 luu trng thai ban dau
            mov     biendem,#16         ;bien dem so lan dich chuyen lan dau = 16

lb2:        mov     bientam,biendem     ;chuyen biendem sang bientam
            mov     y0,#11111110b
            mov     y1,#11111111b

lb1:        mov     a,y0
            anl     a,x0                ;lay x0 or voi y0 roi goi ra led0
            mov     led0,a              ;xuat ra led0

            mov     a,y1
            anl     a,x1                ;lay x1 or voi y1 roi goi ra led1
            mov     led1,a              ;xuat ra led1

            lcall   delay
            setb    c
            mov     a,y0
            rlc     a
            mov     y0,a

            mov     a,y1
            rlc     a
            mov     y1,a
```

```
                djnz    bientam,lb1             ;giam bientam neu chua bang 0 thi quay ve lai

                mov     x0,led0                 ;cat noi dung sau cung khi da dich chuyen 1 led
                mov     x1,led1                 ;cat noi dung sau cung khi da dich chuyen 1 led
                djnz    biendem,lb2             ;giam bien dem de xu li lan ke
                ljmp    lb                      ;nhay ve lam lai tu dau

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

delay:          mov     r6,#0ffh
del2 :          mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,del2
                ret
end
```

IV. Bài tập:

1. Hãy viết chương trình điều khiển 3 port: port0, port1, port2 sáng dồn.
2. Hãy viết chương trình điều khiển 4 port: port0, port1, port2 và port3 sáng dồn.
3. Hãy viết chương trình sáng dồn 2 port 0 và 1 từ ngoài vào trong và từ trong ra ngoài.
4. Hãy viết chương trình sáng dồn 4 port 0, 1, 2 và 3 từ ngoài vào trong và từ trong ra ngoài.

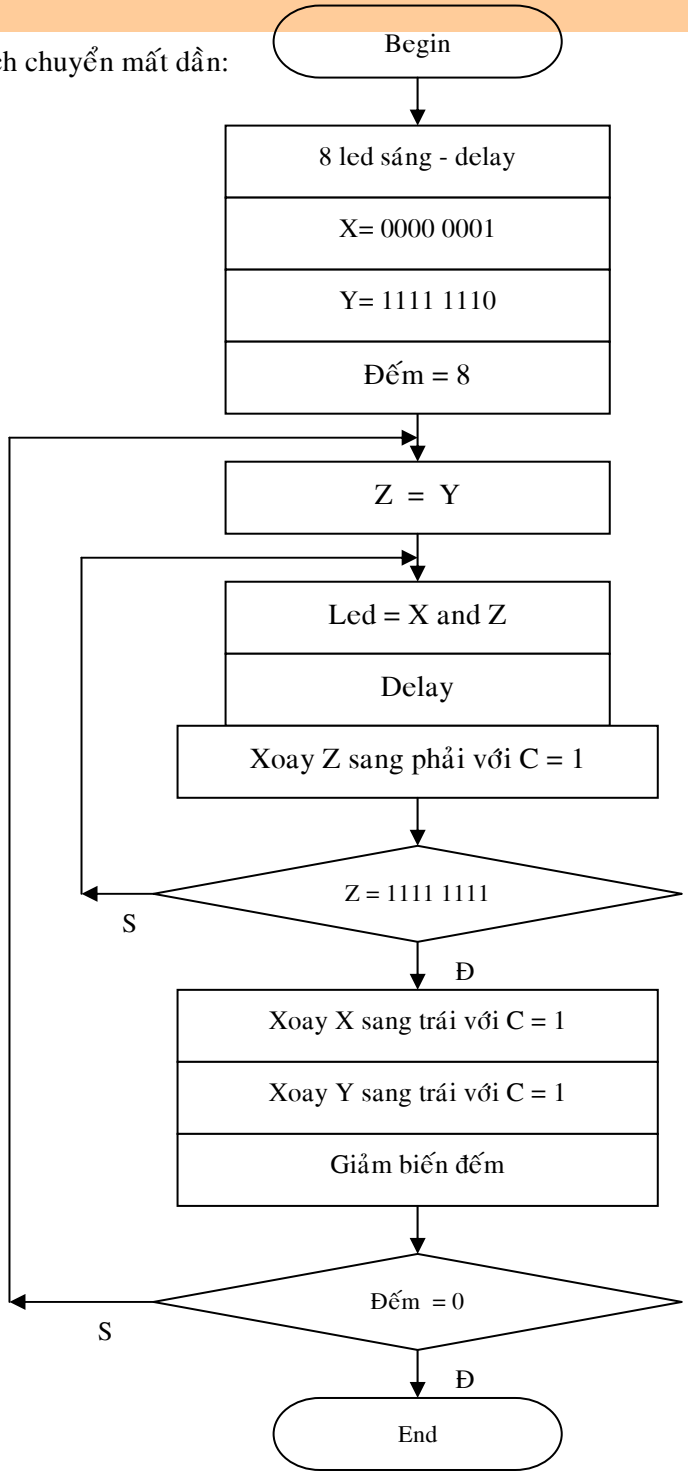
BÀI SỐ : 2-4	**THỰC HÀNH VI ĐIỀU KHIỂN** CHƯƠNG TRÌNH ĐIỀU KHIỂN ĐIỂM SÁNG DỊCH CHUYỂN MẤT DẦN	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Hiểu cách sử dụng lệnh xoay kết hợp với lệnh logic để thực hiện chương trình điều khiển led làm quen với lập trình.

II. Trình tự thực hiện:

1. Giải thuật điều khiển 8 led dịch chuyển mất dần:



2. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 0 với một trong bốn PINHD của dãy 32 led.
3. Khởi động phần mềm, tạo File mới và biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang het va tat dan tu trai sang phai
;truong dai hoc su pham ky thuat-nguyen dinh phu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;dinh nghia cac nhan

                x0      equ     11h
                y0      equ     21h
                z0      equ     r0

                led0    equ     p0
                dem     equ     40h

                org     0000h

lb:             mov     led0,#0ffh
                lcall   delay
                mov     dem,#8

                mov     x0,#00000001b
                mov     y0,#11111110b
lb2:            mov     z0,y0
lb1:            lcall   x_and_z_out
                lcall   delay
                lcall   xoay_z

                cjne    z0,#0ffh,lb1

                lcall   xoay_x
                lcall   xoay_y
                djnz    dem,lb2
                ljmp    lb
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;bat dau cac chuong trinh con
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xoay_z:         setb    c
                mov     a,z0
                rrc     a
                mov     z0,a
                ret

x_and_z_out:    mov     a,x0
                anl     a,z0
                mov     led0,a
                ret

xoay_x:         setb    c
                mov     a,x0
                rlc     a
                mov     x0,a
                ret

xoay_y:         setb    c
                mov     a,y0
```

```
                rlc     a
                mov     y0,a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#0ffh
de:             mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de
                ret

end
```

III. Các chương trình mẫu:

Chương trình mẫu điều khiển 16 led dịch chuyển tắt dần

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien port 0 sang het va tat dan tu trai sang phai
;truong dai hoc su pham ky thuat-nguyen dinh phu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;dinh nghia cac nhan

                x0      equ     11h
                x1      equ     12h
                y0      equ     21h
                y1      equ     22h
                z0      equ     r0
                z1      equ     r1

                led0    equ     p0
                led1    equ     p1

                dem     equ     40h

                org     0000h

lb:             mov     led0,#0ffh
                mov     led1,#0ffh

                lcall   delay
                mov     dem,#16

                mov     x0,#00000001b
                mov     x1,#00000000b

                mov     y0,#11111110b
                mov     y1,#11111111b

lb2:            mov     z0,y0
                mov     z1,y1

lb1:            lcall   x_and_z_out
                lcall   delay
                lcall   xoay_z

                cjne    z0,#0ffh,lb1
                cjne    z1,#0ffh,lb1

                lcall   xoay_x
                lcall   xoay_y
                djnz    dem,lb2
```

```
                ljmp    lb
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;bat dau cac chuong trinh con
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xoay_z:         setb    c
                mov     a,z1
                rrc     a
                mov     z1,a
                mov     a,z0
                rrc     a
                mov     z0,a
                ret

x_and_z_out:    mov     a,x0
                anl     a,z0
                mov     led0,a
                mov     a,x1
                anl     a,z1
                mov     led1,a
                ret

xoay_x:         setb    c
                mov     a,x0
                rlc     a
                mov     x0,a
                mov     a,x1
                rlc     a
                mov     x1,a
                ret

xoay_y:         setb    c
                mov     a,y0
                rlc     a
                mov     y0,a
                mov     a,y1
                rlc     a
                mov     y1,a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#090h
de:             mov     r7,#0ffh
                djnz    r7,$
                djnz    r6,de
                ret

end
```

IV. Bài tập:

1. Hãy viết chương trình điều khiển 3 port: 0, 1, t2 giống như trên.
2. Hãy viết chương trình điều khiển 4 port: 0, 1, 2, 3 giống như trên.
3. Hãy viết chương trình điều khiển 4 port 0, 1, 2, 3 với điểm sáng dịch chuyển mất dần từ theo chiều từ trong ra và từ ngoài vào.

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 2-5** CHƯƠNG TRÌNH DELAY SỬ DỤNG TIMER	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách tính toán các thông số delay của timer để viết các chương trình delay chính xác.

II. 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.
2. Khởi động phần mềm, tạo File mới và biên soạn chương trình sau:

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

        org     0000h
        mov     tmod,#01        ; khoi tao timer T0 mode 1 dem 16 bit
        setb     tr0            ; cho phep timer 0 bat dau dem xung
b61:    mov     p1,#00h
        lcall   delay           delay 65536 micro giay
        mov     p1,#0ffh
        lcall   delay
        sjmp    b61
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con delay 65535 micro giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:  clr     cf0             ; xoa co ngat cua timer 0
        mov     tl0,#0          ; nap 0 vao TL0
        mov     th0,#0          ; nap 0 vao TH0
        jnb     tf0,$           ; kiem tra co tran
        ret
        end
```

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

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.

Hàng lệnh đầu tiên trong chương trình chính là chọn mode làm việc cho timer T0 – hãy xem chương timer (timer T1 chưa sử dụng nên không cần quan tâm).

Lệnh thứ 2 là cho phép timer bắt đầu đếm xung nội từ mạch dao động bên trong.

Với chương trình trên thì timer T0 sẽ đếm từ gía trị nạp ban đầu 0000H đến FFFFH và khi có thêm 1 xung nữa thì giá trị đếm sẽ là 10000H. Một cách đơn giản ta có thể xem con số 10000H được chia ra làm 2: số "1" được lưu trong cờ tràn TF0 và số "0000H" được lưu trong 2 thanh ghi TH0TL0. 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.

Trong chương trình con có 2 lệnh nạp lại giá trị cho TH0 và TL0 của bài này là dư vì khi timer bị tràn thì nó tự động là cho 2 thanh ghi trên mang giá trị 0.

III. Các chương trình mẫu:

Để đếm lượng thời gian nhỏ hơn ta hãy tham khảo bài mẫu dưới đây:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh sang tat port1 su dung timer lam bo dinh thoi delay 250 micro giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            org     0000h
            mov     tmod,#01        ; khoi tao timer T0 mode 1 dem 16 bit
            setb    tr0             ; cho phep timer 0 bat dau dem xung

      b62:  mov     p1,#00h
            lcall   delay           ; delay 250 micro giay
            mov     p1,#0ffh
            lcall   delay
            sjmp    b62
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con delay 250 micro giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      delay:clr     tf0             ; xoa co ngat cua timer 0
            mov     tl0,#06h        ; nap 05 vao TL0
            mov     th0,#0FFh       ; nap FF vao TH0
            djnb    tf0,$
            ret
      end
```

Lượng thời gian của bài này chỉ có 250 micro giây, vì giá trị ban đầu nạp cho timer là FF05H nên khi đếm đến 10000H kết quả xung đếm được 10000H – FF06H = 00FA (250) và mỗi xung có chu kỳ 1 micro giây nên lượng thời gian mà timer T0 đếm đượclà 250 micro giây.

Để khỏi phải tính toán phức tạp ta có thể viết bằng hai lệnh như sau:

```
            mov     tl0,#low(-250)          ; nap 05 vao TL0
            mov     th0,#high(-250)         ; nap FF vao TH0
```

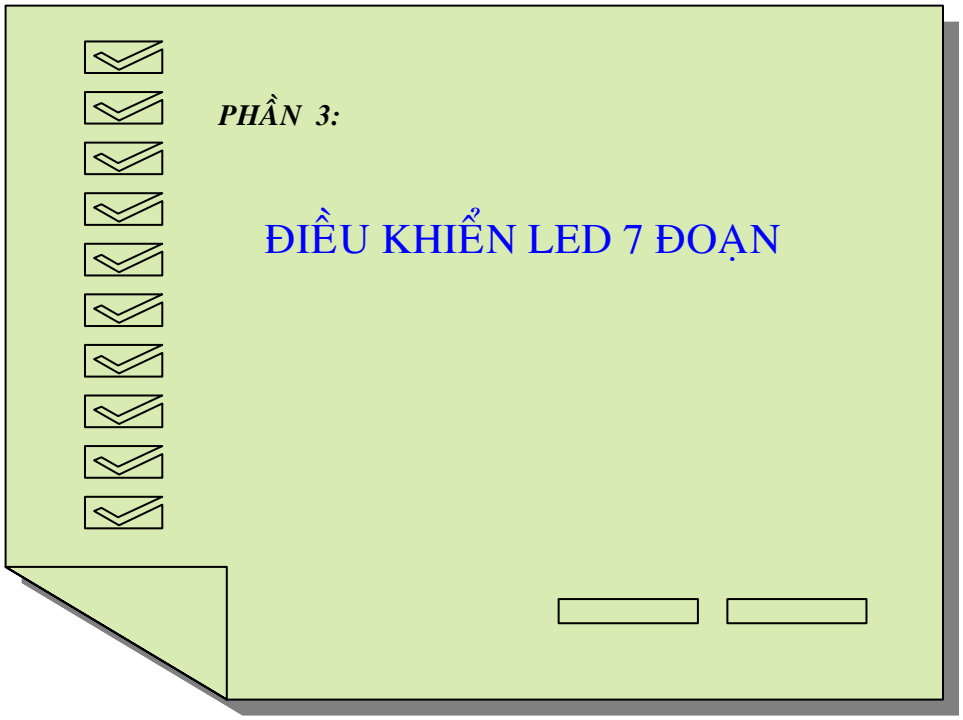
Khi biên dịch thì trình biên dịch tự động tính toán cho chúng ta.

Chú ý với các chương trình điều khiển led sáng với thời gian trể nhỏ thì led sáng mờ nhưng không chóp tắt như các bài ta đã viết ở trên. Trong phần tính toán chúng ta chưa tính toán các lệnh trong chương trình con delay.

Muốn viết chương trình với các khoảng thời gian lớn hơn thì phải thêm thanh ghi ví dụ muốn viết delay 5 giây thì ta viết chương trình trình con delay 50 000 µs = 50ms và cho chúng thực hiện 100 lần – sau này chúng ta sẽ dùng timer để tạo ra các xung chính xác về thời gian cho các bài sau.

IV. Bài tập:

1. Hãy viết chương trình sáng tắt port 2 sử dụng timer làm bộ định thời delay 5 giây.
2. Hãy viết chương trình giống trên nhưng delay 10 giây.
3. Tương tự hãy viết chương trình delay 1 giờ.



PHẦN 3:

ĐIỀU KHIỂN LED 7 ĐOẠN

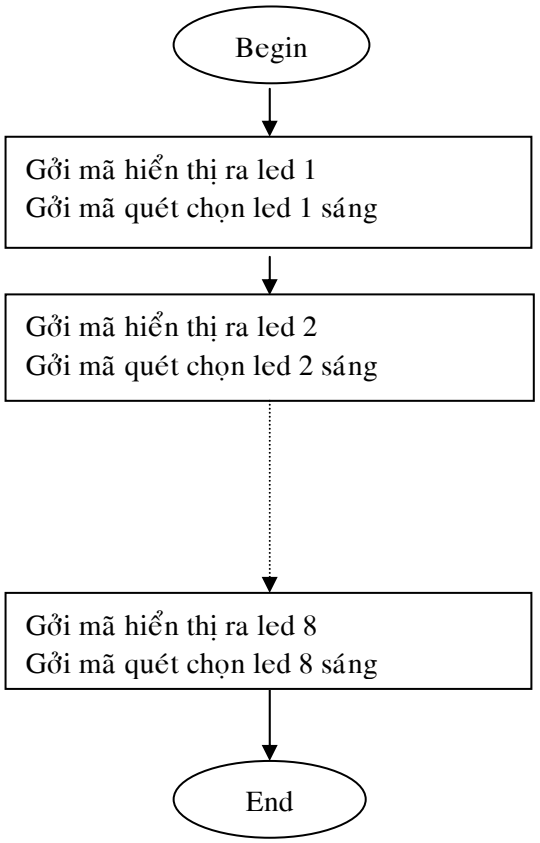
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 3-1** CHƯƠNG TRÌNH ĐIỀU KHIỂN LED 7 ĐOẠN SÁNG	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết được cách tìm mã điều khiển led 7 đoạn , biết nguyên lý điều khiển led theo phương pháp quét và cách viết chương trình điều khiển led 7 đoạn. Sinh viên phải hiểu rằng 1 yêu cầu điều khiển có thể thực hiện bằng nhiều chương trình khác nhau.

II. Trình tự thực hiện:

1. Giải thuật:



2. 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.
   - Dùng bus dây kết nối port 2 đến pinhd điều khiển quét hàng từ T0 đến T7.
3. Khởi động phần mềm, tạo file mới để biên soạn chương trình sau:

III. Các chương trình mẫu:

Chương trình điều khiển led 7 đoạn sáng số 9

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien led 7 doan sang so 9 o 1 led ben phai
;ket noi port 0 den pinhd dieu khien cac doan a,b,c,d,e,f,g,dp
;ket noi port 2 dem pinhd dieu khien quet hang
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        org     0000h

        mov     p2,#01111111b       ;cho phep D1 sang
        mov     p0,#10010000b       ;ma cua so 9
        sjmp    $                   ;ngung lai
```

```
        end
```

Chương trình trên chỉ có tác dụng thử cho một led sáng. Để có thể sáng 8 led từ số 0 đến số 7 ta hãy viết chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh thu 8 led 7 doan sang cac so 0 den so 7 tren 8 led
;ket noi port 0 den pinhd dieu khien cac doan a,b,c,d,e,f,g,dp
;ket noi port 2 dem pinhd dieu khien quet hang
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h

b11:            mov     p2,#01111111b       ; cho phep D1 sang
                mov     p0,#11000000b       ; ma cua so 0
                lcall   delay

                mov     p2,#10111111b       ; cho phep D2 sang
                mov     p0,#11111001b       ; ma cua so 1
                lcall   delay

                mov     p2,#11011111b       ; cho phep D3 sang
                mov     p0,#10100100b       ; ma cua so 2
                lcall   delay

                mov     p2,#11101111b       ; cho phep D4 sang
                mov     p0,#10110000b       ; ma cua so 3
                lcall   delay

                mov     p2,#11110111b       ; cho phep D5 sang
                mov     p0,#10011001b       ; ma cua so 4
                lcall   delay

                mov     p2,#11111011b       ; cho phep D6 sang
                mov     p0,#10010010b       ; ma cua so 5
                lcall   delay

                mov     p2,#11111101b       ; cho phep D7 sang
                mov     p0,#10000010b       ; ma cua so 6
                lcall   delay

                mov     p2,#11111110b       ; cho phep D8 sang
                mov     p0,#11111000b       ; ma cua so 7
                lcall   delay
                sjmp    b11

delay:          mov     r7,#01h
del2 :          mov     r6,#0ffh
                djnz    r6,$
                djnz    r7,del2
                ret
        end
```

4. Thực hiện các bước giống như bài chuẩn.

Chú ý: Nếu khi chạy mà kết quả hiển thị không đúng thì hãy xem lại chương trình, nếu chương trình hoàn toàn đúng thì hãy xem kết nối 2 port điều khiển với led có đúng [trên bo mạch có ghi tên và thứ tự các pinhhd.

Ở ví dụ 2, để sáng cùng 1 lúc 8 led ta sử dụng phương pháp quét led tức là tại một thời điểm chỉ có 1 led sáng và khi giảm thời gian delay vừa với thời gian lưu ảnh của mắt. Lúc này, chúng ta sẽ thấy 8 led sáng cùng một lúc.

Chương trình điều khiển 8 led sáng từ số 0 đến số 7 cách 2:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh hien thi 8 so tu 0 den 7 tren 8 led theo cach viet so 2
;ket noi port 0 den pinhd dieu khien cac doan a,b,c,d,e,f,g,dp
;ket noi port 2 dem pinhd dieu khien quet hang
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                quet    equ     p2
                led7    equ     p0

                org     0000h                   ;bat dau chuong trinh
b213:           mov     dptr,#datahthi          ;nap dia chi vung ma vao dptr
                mov     r4,#08h                 ;bien dem 8 lan goi
                mov     r3,#07fh                ;ma quet 01111111h              ;

b212:           clr     a
                movc    a,@a+dptr               ;lay ma 7 doan
                mov     led7,a

                mov     quet,r3
                lcall   delay
                mov     quet,#0ffh              ;tat het de chong lem
                mov     a,r3
                rr a
                mov     r3,a
                inc     dptr
                djnz    r4,b212
                sjmp    b213                    ;quay lai lam lai tu dau
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r7,#01h
del2 :          mov     r6,#0ffh
                djnz    r6,$
                djnz    r7,del2
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;vung du lieu ma cac so tu 0 den 7
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
datahthi:       db      0C0h,0F9h,0A4h,0B0h,99h,92h,82h,0F8h
end
```

IV. Bài tập:

1. Ở ví dụ 1, muốn sáng số 2 ở led D3 thì phải thay đổi gì trong chương trình?
2. Trong trường hợp nào thì nhiều led sáng cùng một lúc và cùng một dữ liệu?
3. Ở chương trình mẫu thứ 2, hãy cho biết led sáng như thế nào:
    a. Từng led sáng các led còn lại tắt.
    b. Tất cả các led sáng .
4. Hãy thử trên máy, khi thay đổi thời gian delay:
    a. Muốn thời gian delay lớn nhất thì thông số đó là bao nhiêu?
    b. Muốn thời gian delay nhỏ nhất thì thông số đó là bao nhiêu?
    c. Trong từng trường hợp hãy cho biết cường độ sáng của led có thay đổi hay không và cho biết trong trường hợp nào thì led sáng rõ?
    d. Hãy tính thông số cho chương trình delay là bao nhiêu để 8 led sáng đều và rõ nhất.
5. Hãy giải thích cho biết chức năng của hàng lệnh chống lem ?

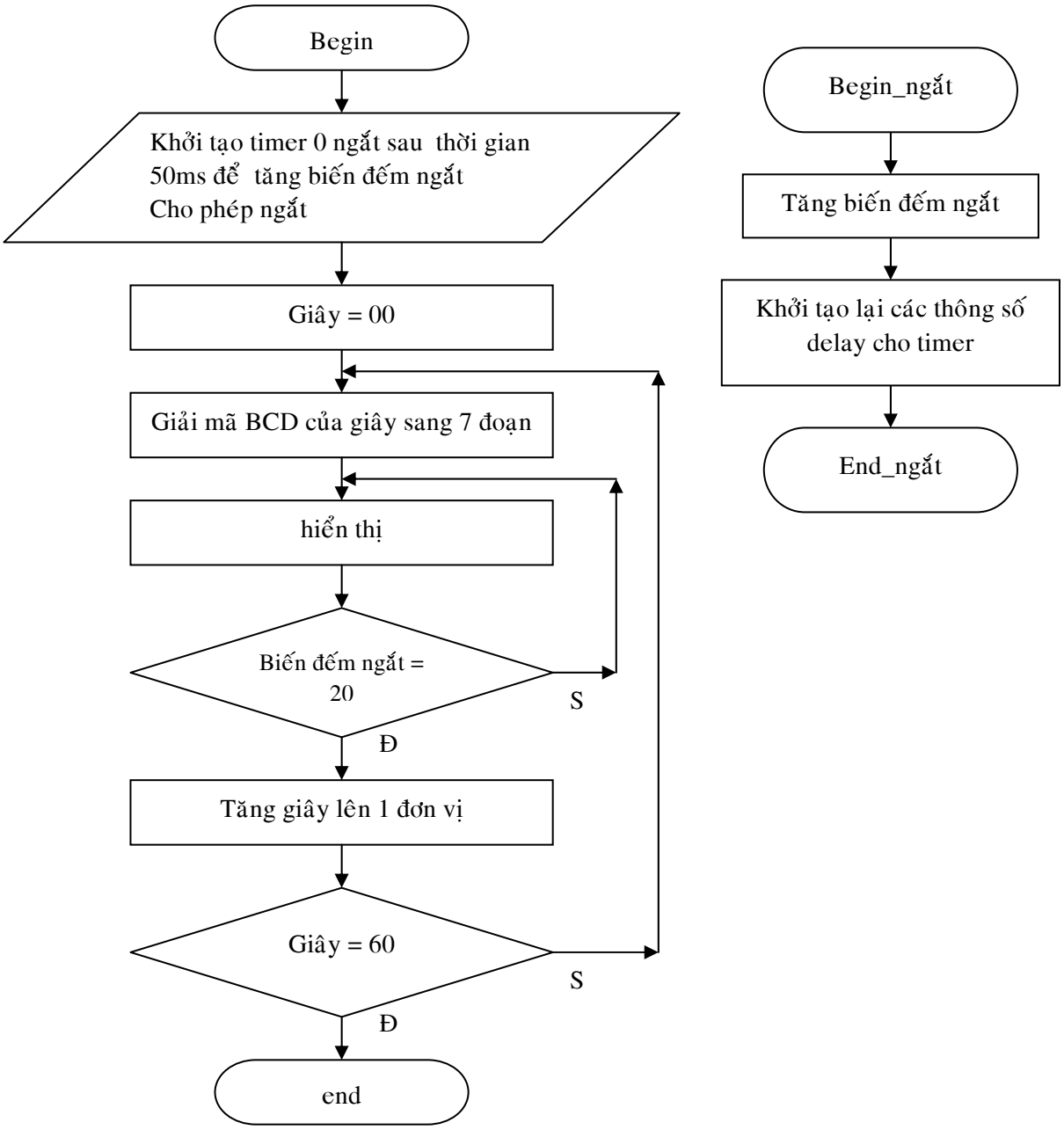
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 3-2** CHƯƠNG TRÌNH ĐẾM GIÂY HIỂN THỊ Ở 2 LED – SỬ DỤNG NGẮT CỦA TIMER ĐỂ ĐẾM CHÍNH XÁC VỀ THỜI GIAN.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Làm quen với cách viết chương trình đếm và sau khi thực hành xong sinh viên có thể viết các chương trình đếm với số đếm tùy ý.

II. Trình tự thực hiện:

1. Giải thuật:



2. 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.

- Dùng bus dây kết nối port 2 đến pinhd điều khiển quét hàng từ T0 đến T7.

3. Khởi động phần mềm và biên soạn chương trình sau.

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dem len tu 00 den 60 hien thi tren 2 led cua 8 led quet
;su dung ngat timer t0 de dem chinh xac ve thoi gian
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                giay     equ     r2              ;gan bien dem giay la R2
                bdn      equ     r1              ;gan bien dem ngat

                quet     equ     p2
                led7     equ     p0
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh chinh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org      0000h          ;bat dau chuong trinh
                ljmp     main           ;nhay den chtr chinh

                org      000bh
                ljmp     int_t0         ;nhay den chtr con ngat timer0

main:           mov      tmod,#01h      ;timer0: mod 1 - dem 16 bit
                mov      dptr,#ma7doan  ;dptr quan ly vung ma 7 doan

                clr      tf0            ;xoa co tran
                mov      IE,#10000010B  ;cho phep timer0 ngat
                mov      TH0,#high(-50000)  ;khoi tao timer delay 50ms
                mov      TL0,#low(-50000)
                setb     tr0            ;cho phep timer bat dau dem

main0:          mov      giay,#00h      ;giay=00
main1:          mov      bdn,#00        ;nap bien den so lan ngat
                lcall    gma
main2:          lcall    hthi           ;goi chtr con hien thi

                cjne     bdn,#20,main2  ;chua dung 20 lan [tuc 1 giay]
                mov      a,giay         ;chuyen  giay sang A
                add      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

                ljmp     main0          ;lam lai tu dau
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con giai ma
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
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
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;CHUONG TRINH CON NGAT TIMER0 SAU KHOANG THOI GIAN 50MS
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
int_t0:         inc      bdn            ;tang bien dem giay len 1
```

```
                mov     TH0,#high(-50000)       ;khoi tao timer delay 50ms
                mov     TL0,#low(-50000)
                clr     TF0
                reti

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           mov     a,#01111111b            ;ma quet
                mov     r0,#27h

ht1:            mov     led7,@r0
                mov     quet,a
                lcall   delay1
                mov     quet,#0ffh
                dec     r0
                rr      a                       ;chuyen sang led ke
                cjne    r0,#25h,ht1
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1:         mov     r7,#0fh
                djnz    r7,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao ma 7 doan tu so '0' den so '9'
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:        db      0C0h,0F9h,0A4h,0B0h,99h,92h,82h,0F8h,80h,90h
                end
```

4. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch đếm đúng từ 00 đến 59.

III. Câu hỏi và bài tập ứng dụng:

1. Hãy viết chương trình đếm lên từ 00 đến 99 thì làm như thế nào?
2. Hãy viết chương trình đếm xuống từ 60 về 00 thì làm như thế nào?

THỰC HÀNH VI ĐIỀU KHIỂN

BÀI SỐ : 3-3

CHƯƠNG TRÌNH ĐẾM PHÚT - GIÂY HIỂN THỊ Ở 4 LED.

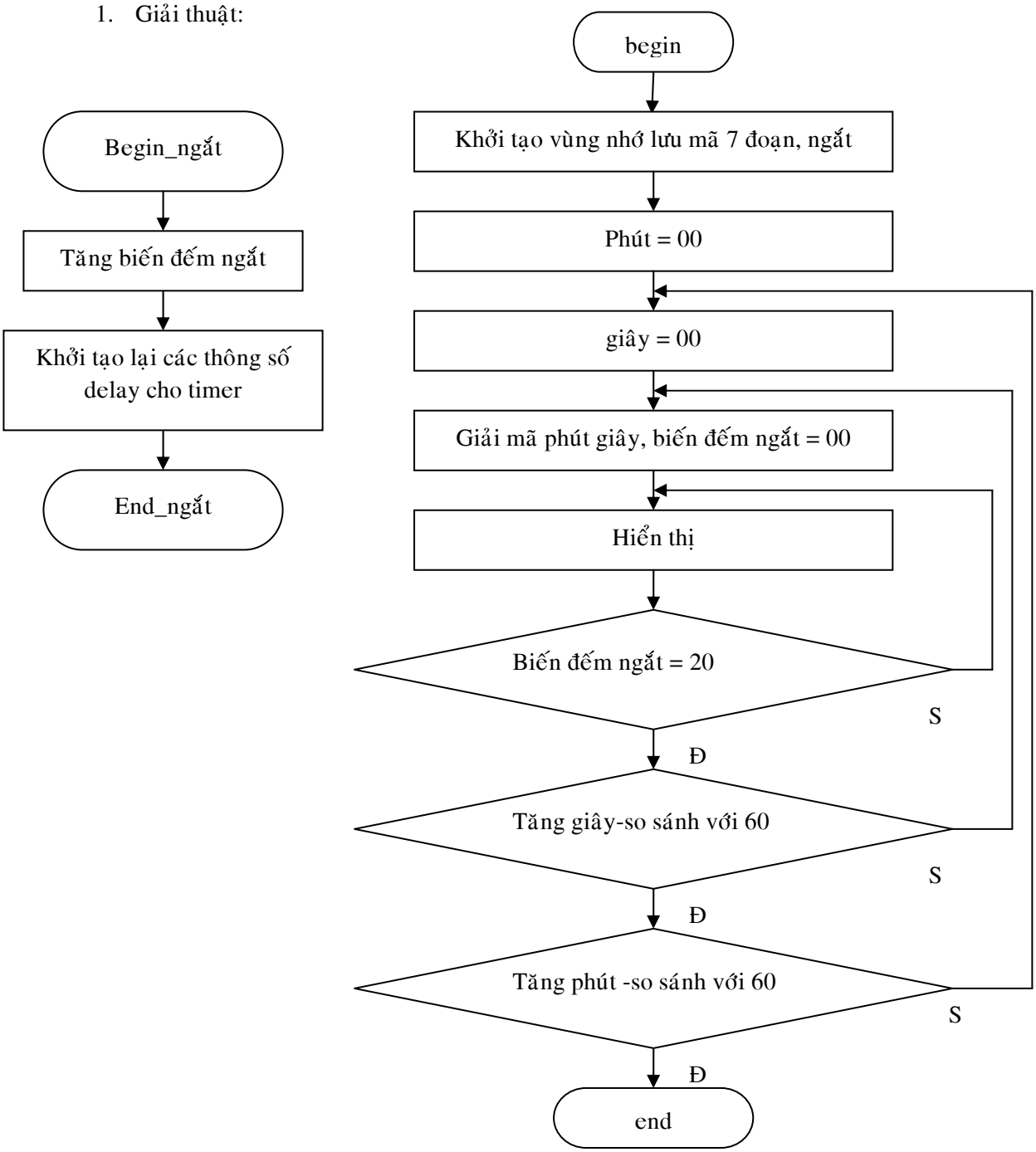
NGÀY :
SỐ TIẾT :
LỚP :
MSSV :

I. Mục đích yêu cầu:

Biết cách viết chương trình đếm phút giây, cách kiểm tra chương trình đếm.

II. Trình tự thực hiện:

1. Giải thuật:



2. 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.
   - Dùng bus dây kết nối port 2 đến pinhd điều khiển quét hàng từ T0 đến T7.
3. Khởi động phần mềm, biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dem phut giay
;su dung ngat timer t0 de dem chinh xac ve thoi gian
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                giay     equ     r2              ;gan bien dem giay la R2
                phut     equ     r3              ;gan bien dem phut cho R3
                bdn      equ     r1              ;gan bien dem ngat

                quet     equ     p2
                led7     equ     p0

                org      0000h                   ;bat dau chuong trinh
                ljmp     main                    ;nhay den chtr chinh

                org      000bh
                ljmp     int_t0                  ;nhay den chtr con ngat timer0

main:           mov      tmod,#01h               ;timer0: mod 1 - dem 16 bit
                mov      dptr,#ma7doan           ;dptr quan ly vung ma 7 doan
                mov      22h,#0ffh
                mov      25h,#0ffh

                clr      tf0                     ;xoa co bao ngat
                MOV      IE,#10000010B           ;cho phep timer ngat
                MOV      TH0,#high(-50000)       ;khoi tao timer delay 50ms
                MOV      TL0,#low(-50000)
                setb     tr0                     ;cho timer bat dau dem

main3:          mov      phut,#00h               ;phut=00
main0:          mov      giay,#00h               ;giay=00
main1:          mov      bdn,#00                         ;nap bien den so lan ngat
                lcall    gma
main2:          lcall    hthi                    ;goi chtr con hien thi

                cjne     bdn,#20,main2           ;chua dung 20 lan [tuc 1 giay]
                mov      a,giay                  ;chuyen  giay sang A
                add      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

                mov      a,phut                  ;chuyen  phut sang A
                add      a,#1                    ;tang phut len 1
                da       a                       ;hieu chinh so BCD trong A
                mov      phut,a                  ;tra lai cho phut
                cjne     phut,#60h,main0                 ;ss giay voi 60

                ljmp     main3                   ;lam lai tu dau
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con giai ma
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
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

                mov     a,phut
                anl     a,#0fh              ;xoa 4 bit cao hang chuc phut
                movc    a,@a+dptr           ;lay ma 7 doan
                mov     24h,a               ;cat ma vao o nho 20h
                mov     a,phut
                anl     a,#0f0h             ;xoa 4 bit thap hang dvi phut
                swap    a                   ;chuyen 4 bit cao xuong vi tri thap
                movc    a,@a+dptr           ;lay ma 7 doan  hang chuc
                mov     23h,a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;CHUONG TRINH CON NGAT TIMER0 SAU KHOANG THOI GIAN 50MS
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
int_t0:         INC     bdn                 ;TANG BIEN DEM GIAY
                MOV     TH0,#high(-50000)   ;khoi tao lai cho timer delay 50ms
                MOV     TL0,#low(-50000)
                CLR     TF0
                RETI
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           mov     a,#01111111b        ;ma quet
                mov     r0,#27h
ht1:            mov     led7,@r0
                mov     quet,a
                lcall   delay1
                mov     quet,#0ffh
                dec     r0
                rr      a                   ;chuyen sang led ke
                cjne    r0,#22h,ht1
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1:         mov     r7,#0fh
                djnz    r7,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; khai bao ma 7 doan tu so '0' den so '9'
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:        db      0C0h,0F9h,0A4h,0B0h,99h,92h,82h,0F8h,80h,90h
                end
```

4. Thực hiện các bước giống như bài chuẩn và xem kết quả mạch có đếm đúng phút và giây hay không, nếu không đúng thì hiệu chỉnh lại.

Chú ý: có thể giảm giá trị so sánh của biến bdn bằng 1 để thời gian trong chương trình này giảm nhỏ giúp kiểm tra nhanh hàng phút, sau khi đếm đúng thì hãy hiệu chỉnh lại lệnh so sánh bdn với số 20 thì mạch sẽ đếm đúng thời gian.

III. Câu hỏi và bài tập ứng dụng:

1. Hãy điều chỉnh chương trình đếm phút giây để đếm BCD từ 0000 đến 9999 hiển thị ở các led 5,6,7,8.
2. Hãy viết chương trình đếm số hex từ 0000H đến FFFFH.

"Có công mài sắt có ngày nên kim - vạn sự khởi đầu nan "

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 3-4** CHƯƠNG TRÌNH ĐẾM GIỜ - PHÚT - GIÂY HIỂN THỊ Ở 6 LED.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách viết chương trình đếm giờ phút giây, cách kiểm tra chương trình đếm.

II. Trình tự thực hiện:

1. 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.
   - Dùng bus dây kết nối port 2 đến pinhd điều khiển quét hàng từ T0 đến T7.
2. Khởi động phần mềm, soạn thảo chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dem dong ho so gio phut giay
;su dung ngat timer t0 de dem chinh xac ve thoi gian
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                giay    equ     r2              ;gan bien dem giay la R2
                phut    equ     r3              ;gan bien dem phut cho R3
                gio     equ     r4              ;gan bien dem gio cho r4
                bdn     equ     r1              ;gan bien dem ngat

                quet    equ     p2
                led7    equ     p0

                org     0000h                   ;bat dau chuong trinh
                ljmp     main                   ;nhay den chtr chinh

                org     000bh
                ljmp    int_t0                  ;nhay den chtr con ngat timer0

main:           mov     22h,#0ffh
                mov     25h,#0ffh
                mov     tmod,#01h               ;timer0: mod 1 - dem 16 bit
                mov     dptr,#ma7doan           ;dptr quan ly vung ma 7 doan

                clr     tf0                     ;xoa co tran timer 0
                mov     ie,#10000010B           ;cho phep timer 0 ngat
                mov     th0,#high(-50000)               ;khoi tao timer delay 50ms
                mov     tl0,#low(-50000)
                setb    tr0                     ;cho phep timer0 bat dau dem

main4:          mov     gio,#00                 ;gio=00
main3:          mov     phut,#00h               ;phut=00
main0:          mov     giay,#00h               ;giay=00
main1:          mov     bdn,#00                         ;nap bien den so lan ngat
                lcall   gma
main2:          lcall   hthi                    ;goi chtr con hien thi

                cjne    bdn,#20,main2           ;chua dung 20 lan [tuc 1 giay]
                mov     a,giay                  ;chuyen  giay sang A
                add     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
```

```
                mov     a,phut                  ;chuyen  phut sang A
                add     a,#1                    ;tang phut len 1
                da      a                       ;hieu chinh so BCD trong A
                mov     phut,a                  ;tra lai cho phut
                cjne    phut,#60h,main0                 ;ss phut voi 60

                mov     a,gio                   ;chuyen gio sang A
                add     a,#1                    ;tang gio len 1
                da      a                       ;hieu chinh so BCD trong A
                mov     gio,a                   ;tra lai cho gio
                cjne    gio,#24h,main3          ;ss gio voi 24

                ljmp    main4                   ;lam lai tu dau
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con giai ma
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
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 27h

                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                   ;cat vao o nho 26h

                mov     a,phut
                anl     a,#0fh                  ;xoa 4 bit cao hang chuc phut
                movc    a,@a+dptr               ;lay ma 7 doan
                mov     24h,a                   ;cat ma vao o nho 25h

                mov     a,phut
                anl     a,#0f0h                 ;xoa 4 bit thap hang dvi phut
                swap    a                       ;chuyen 4 bit cao xuong vi tri thap
                movc    a,@a+dptr               ;lay ma 7 doan  hang chuc
                mov     23h,a                   ;cat vao o nho 24h

                mov     a,gio
                anl     a,#0fh                  ;xoa 4 bit cao hang chuc gio
                movc    a,@a+dptr               ;lay ma 7 doan
                mov     21h,a                   ;cat ma vao o nho 23h

                mov     a,gio
                anl     a,#0f0h                 ;xoa 4 bit thap hang dvi gio
                swap    a                       ;chuyen 4 bit cao xuong vi tri thap
                movc    a,@a+dptr               ;lay ma 7 doan  hang chuc
                mov     20h,a                   ;cat vao o nho 22h
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;CHUONG TRINH CON NGAT TIMER0 SAU KHOANG THOI GIAN 50MS
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
int_t0:         inc      bdn                    ;tang bien dem ngat
                mov     th0,#high(-50000)       ;khoi tao lai cho timer delay 50ms
                mov     tl0,#low(-50000)
                clr     tf0
                reti
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           mov     a,#01111111b            ;ma quet
                mov     r0,#27h
```

```
ht1:            mov     led7,@r0
                mov     quet,a
                lcall   delay1
                mov     quet,#0ffh
                dec     r0
                rr      a                       ;chuyen sang led ke
                cjne    r0,#1Fh,ht1
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1:         mov     r7,#0fh
                djnz    r7,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao ma 7 doan tu so '0' den so '9'
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:        db      0C0h,0F9h,0A4h,0B0h,99h,92h,82h,0F8h,80h,90h
end
```

3. Thực hiện các bước giống như bài chuẩn và xem kết quả mạch có đếm đúng phút và giây hay không, nếu không đúng thì hiệu chỉnh lại.

Chú ý: có thể giảmgiá trị so sánh của biến bdn bằng 1 để thời gian trong chương trình này giảm nhỏ giúp kiểm tra nhanh hàng phút, sau khi đếm đúng thì hãy hiệu chỉnh lại lệnh so sánh bdn với số 20 thì mạch sẽ đếm đúng thời gian.

III. **Bài tập ứng dụng**:

1. Hãy điều chỉnh chương trình đếm giờ phút giây để đếm BCD từ 000000 đến 999999 hiển thị ở các led 3,4,5,6,7,8.
2. Hãy cho biết sai số và cách làm giảm sai số.
3. Chương trình trên luôn bắt đầu chạy tại 00 giờ 00 phút 00 giây. Hãy viết chương trình quét phím để điều chỉnh giờ phút giây theo ý muốn.

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 3-5** CHƯƠNG TRÌNH ĐIỀU KHIỂN BÀN PHÍM MA TRẬN VÀ HIỂN THỊ MÃ CỦA PHÍM NHẤN TRÊN 1 LED 7 ĐOẠN.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách viết chương trình điều khiển quét phím, tạo mã phím và hiển thị mã phím. ***Khi chạy chương trình này thì trên led sẽ có dấu chấm sáng và khi nhấn phím nào thì mã của phím đó sáng trên led.***

II. Trình tự thực hiện:

1. Chức năng của phím hay nút nhấn hay contact:

Dùng để giao tiếp điều khiển giữa con người và thiết bị ví dụ như contact tắt mở bóng đèn và người sử dụng tác động đến contact để tắt mở thiết bị. Máy tính cộng trừ nhân chia thì chức năng của bàn phím là nhập các thông số vào máy và yêu cầu máy thực hiện các phím tính, tương tự bàn phím máy tính cũng vậy dùng để giao tiếp con người và máy.

Trong thực tế các thiết bị điều khiển lập trình đều có sử dụng nút nhấn, có thiết bị sử dụng ít nút nhấn như tivi, máy giặt, ..., có thiết bị sử dụng nhiều nút nhấn như bàn phím vi máy tính, điện thoại, bàn phím máy tính cộng trừ nhân chia,...

2. Nguyên lý:

Với nút nhấn thường hở thì khi ta nhấn thì sẽ ngắn mạch cho tín hiệu hay dòng điện chạy qua và khi không nhấn thì hở mạch sẽ ngắt tín hiệu hay ngắt dòng điện, còn nút nhấn thường hở thì ngược lại. Đối với tín hiệu số thì ta có thể xem "khi contact hở làm cho 1 ngõ vào ở mức logic 1 và khi contact nhấn làm thay đổi sang trạng thái mức logic 0 hoặc ngược lại".

Với cách kết nối kiểu như đã trình bày thì một contact phải sử dụng một đường tín hiệu giao tiếp. Nếu ứng dụng dùng vài chục phím thì theo cách này là không khả thi vì số lượng tín hiệu không đủ, nếu muốn thì phải dùng thêm IC giao tiếp.

Khi ứng dụng sử dụng nhiều phím thì ta nên kết nối theo dạng ma trận phím, với ma trận [m hàng, n cột] thì số phím bằng [n x m]. Ma trận 4 x 4 sẽ có 16 phím thì chỉ cần dùng 8 đường tín hiệu giao tiếp, ma trận 4 x 5 sẽ có 20 phím thì chỉ cần dùng 9 đường tín hiệu giao tiếp, ma trận 8 x 8 thì sẽ có 64 phím thì chỉ cần dùng 16 đường tín hiệu giao tiếp.

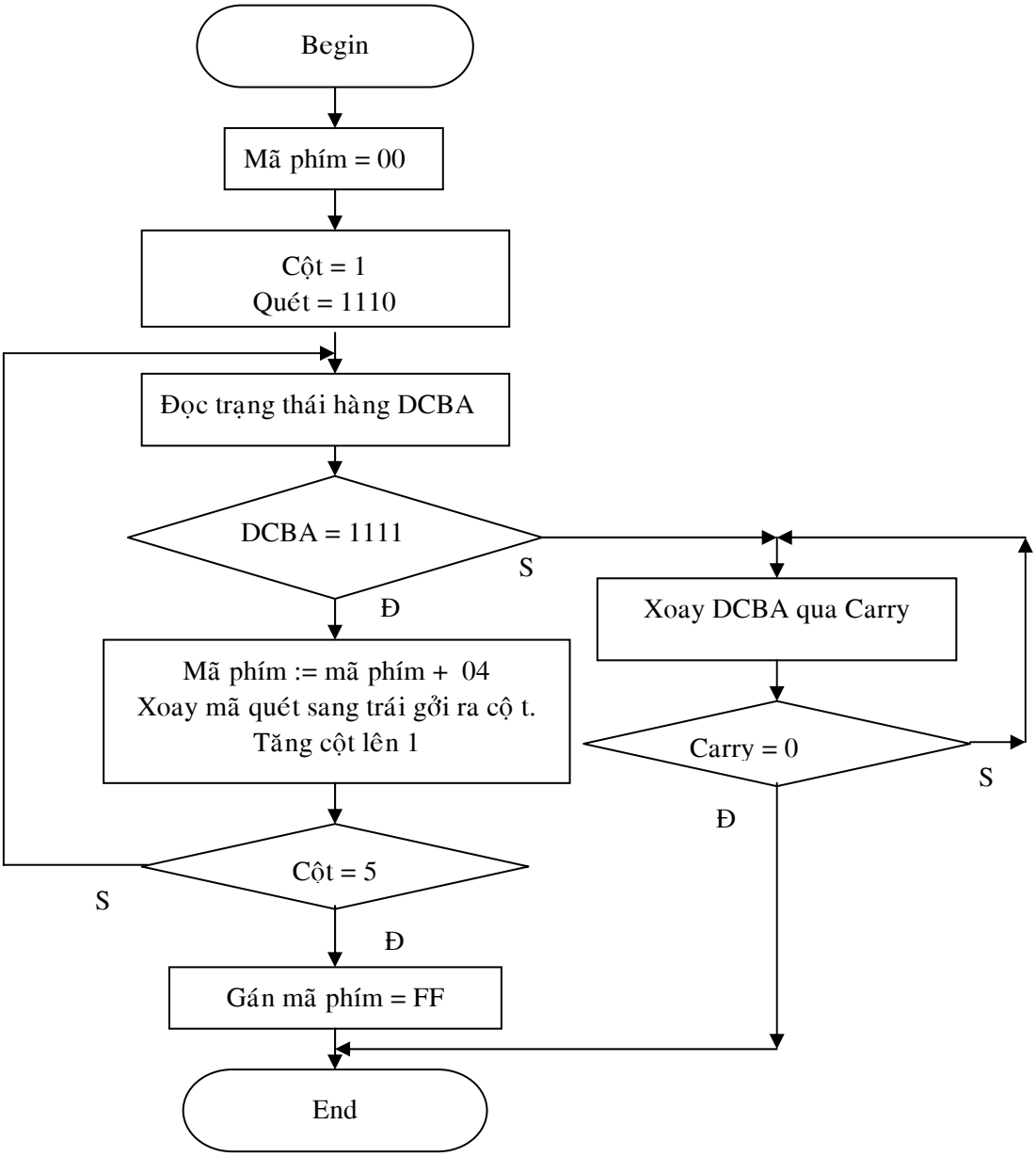
3. Giải thuật cho chương trình con quét phím dùng ma trận:

Với ma trận 4x4 thì phải có 4 ngõ vào và 4 ngõ ra (xem chương 8), 4 ngõ vào thường treo lên nguồn 5V qua điện trở nên mức logic của các ngõ này thường ở mức 1.

4 ngõ còn lại là 4 ngõ ra thường để xuất tín hiệu quét hay mã quét có 1 bit ở mức 0 các bit còn lại ở mức 1.

Chương trình quét phím thường là chương trình con, chương trình chính sẽ gọi chương trình con quét phím để kiểm tra xem có phím nào bị nhấn hay không: nếu không có phím nào bị nhấn thì chương trình con quét phím kết thúc với 1 mã do người lập trình qui định, nếu có phím nhấn thì phải tiến hành thiết lập mã phím của phím đó (mỗi phím có 1 mã duy nhất) rồi tiến hành chống dội phím, sau đó có thể kết thúc quá trình chống dội bằng cách chờ buông phím hay hết thời gian qui định.

Giải thuật của bàn phím ma trận như sau:



Giải thích lưu đồ: Với bàn phím ta sử dụng 1 port nào đó tùy ý chẳng hạn như port 1.

Để điều khiển quét phím thì ta xuất 1 dữ liệu 4 bit: trong đó có 1 bit ở mức thấp và 3 bit ở mức cao ra 4 đường điều khiển quét của bàn phím.

Sau đó ta kiểm tra mức logic của 4 ngõ nhập để xem có phím nào nhấn hay không:

- Nếu có phím nhấn thì 4 bit nhập sẽ có 1 bit ở mức loigc 0 và tiến hành thiết lập mã phím.
- Nếu không có phím nhấn thì 4 bit nhập sẽ ở mức logic 1 – khi đó ta chuyển mức logic 0 sang bit quét kế để dò tìm phím khác.

4. 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.
   - Dùng bus dây kết nối port 2 đến pinhd điều khiển quét hàng từ T0 đến T7.
   - Dùng bus dây kết nối port 3 (chưa qua IC đệm) kết nối với pindhn của bàn phím.
5. Khởi động phần mềm, soạn thảo chương trình sau:

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
;chuong trinh quet phim dung he thong 1 ma phim hien thi tren 1led
;dung port3  chua qua IC dem ket noi voi pinhd cua ban fim
;dung port0 va port 2 ket noi dieu khien led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                quet    equ     p2
                led7    equ     p0
                mtphim  equ     p3              ;ket noi voi ma tran ban phim

                org     0000h
                mov     dptr,#ma7doan
                mov     quet,#07fh              ;xuat ma quet cho 1 led sang
                mov     led7,#7fh

main:           lcall   keypres                 ;goi chtr con quet phim
                cjne    a,#0ffh,main1
                sjmp    main

main1:          lcall   gma_hthi                ;goi chtr con giai ma hien thi
                sjmp    main
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con giai ma fim nhan va hien thi ra 1 led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
gma_hthi:       movc    a,@a+dptr
                mov     led7,a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con quet phim va chong doi phim
;su dung cac thanh ghi:  R4, R5, R6, R7, A
;neu khong nhan thi (A) = FF, neu nhan thi (A) chua ma phim nhan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
keypres:        mov     r4,#10                  ;nhap so dem 10 lan
keypres1 :      lcall   KEY                     ;Neu co phim an thi co c=1
                jc      pn1                     ;kiem tra tiep neu c = 1
                ret                             ;Neu khong co phim nhan thi co c=0

pn1:            djnz    r4,keypres1             ;Quay ve lap lai chong nay
                push    acc                     ;Cat noi dung ma phim trong A

keypres2:       mov     r4,#10                  ;Nhap so dem 10 lan cho nha phim
keypres3:       lcall   key                     ;Co phim nhan hay khong
                jc      keypres2                ;Co thi kiem tra lai
                djnz    r4,keypres3             ;Khong thi lap lai 50 lan va dam bao
                pop     acc                     ;Khoi phuc lai gia tri cho A
                ret                             ;ket thuc mot chuong trinh con
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con quet phim
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
key:            mov     r7,#0feh                        ;bat dau voi cot so 0(feh)
                mov     r6,#4                   ;Su dung r6 lam bo dem
                mov     r5,#00

key1:           mov     mtphim,r7               ;xuat ma quet ra cot
                mov     a,mtphim                ;Doc lai port1 de xu ly tiep theo
                anl     a,#0f0h                 ;xoa 4 bit thap la hang
                cjne    a,#0f0h,key2            ;co nhan fim thi nhay

                mov     a,r7
                rl      a                       ;xoay de chuyen den cot ke tiep
                mov     r7,a

                mov     a,r5                    ;chuyen ma fim sang cot ke
                add     a,#4
```

```
                mov     r5,a

                djnz    r6,key1                 ;Neu nhu sau moi lan 1 cot ma khong

                clr     c                       ;clr c neu nhu khong co phim duoc an
                mov     a,#0ffh                 ;thoat voi ma trong a = FFh
                ret

key2:           swap    a
key4:           rrc     a                       ;xoay sang phai tim bit 0
                jnc     key3                    ;nhay neu (c)=0
                inc     r5                      ;tang ma fim len cot ke
                sjmp    key4                    ;tiep tuc cho den khi duoc (C)=0

key3:           mov     a,r5
                setb    c
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao du lieu ma phim
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:        db      0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h
                db      080h,090h,088h,083h,0c6h,0a1h,086h,08eh

        end
```

6. Thực hiện các bước giống như bài chuẩn và nhấn bất kì phím nào thì trên led 7 đoạn sẽ hiển đúng mã 7 đoạn của phím đó nếu không đúng thì hãy hiệu chỉnh lại cho đúng.

III. Các chương trình mẫu:

CHƯƠNG TRÌNH ĐIỀU KHIỂN BÀN PHÍM MA TRẬN VÀ HIỂN THỊ MÃ CỦA PHÍM NHẤN TRÊN 8 LED 7 ĐOẠN DỊCH CHUYỂN DẦN TỪ PHẢI SANG TRÁI.

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh quet phim dung he thong 1 ma phim hien thi tren 8 led
;dung port3  chua qua IC dem ket noi voi pinhd cua ban fim
;dung port0 va port 2 ket noi dieu khien led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                quet    equ     p2
                led7    equ     p0
                mtphim  equ     p3              ;ket noi voi ma tran ban phim

                org     0000h
                mov     20h,#0ffh               ;tat cac led
                mov     21h,#0ffh
                mov     22h,#0ffh
                mov     22h,#0ffh
                mov     23h,#0ffh
                mov     24h,#0ffh
                mov     25h,#0ffh
                mov     26h,#0ffh
                mov     27h,#0c0h

                mov     dptr,#ma7doan
                mov     quet,#07fh              ;xuat ma quet cho 1 led sang
                mov     led7,#7fh

main:           lcall   keypres                 ;goi chtr con quet phim
                cjne    a,#0ffh,main1
                lcall   hthi
```

```
                sjmp    main

main1:          lcall   dichchuyen
                lcall   giaima          ;goi chtr con giai ma hien thi
                sjmp    main

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con dich chuyen du lieu trong 8 o nho chua du lieu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
dichchuyen:     mov     20h,21h
                mov     21h,22h
                mov     22h,23h
                mov     23h,24h
                mov     24h,25h
                mov     25h,26h
                mov     26h,27h
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con giai ma fim nhan va hien thi ra 1 led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
giaima:         movc    a,@a+dptr
                mov     27h,a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           mov     a,#01111111b            ;ma quet
                mov     r0,#27h

ht1:            mov     led7,@r0
                mov     quet,a
                lcall   delay1
                mov     quet,#0ffh
                dec     r0
                rr      a                       ;chuyen sang led ke
                cjne    r0,#1fh,ht1
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1:         mov     r7,#0fh
                djnz    r7,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con quet phim va chong doi phim
;su dung cac thanh ghi:  R4, R5, R6, R7, A
;neu khong nhan thi (A) = FF, neu nhan thi (A) chua ma phim nhan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
keypres:        mov     r4,#10                  ;nhap so dem 10 lan
keypres1 :      lcall   KEY                     ;Neu co phim an thi co c=1
                jc      pn1                     ;kiem tra tiep neu c = 1
                ret                             ;Neu khong co phim nhan thi co c=0

pn1:            djnz    r4,keypres1             ;Quay ve lap lai chong nay
                push    acc                     ;Cat noi dung ma phim trong A

keypres2:       mov     r4,#10                  ;Nhap so dem 10 lan cho nha phim
keypres3:       lcall   key                     ;Co phim nhan hay khong
                jc      keypres2                ;Co thi kiem tra lai
                djnz    r4,keypres3             ;Khong thi lap lai 50 lan va dam bao
                pop     acc                     ;Khoi phuc lai gia tri cho A
                ret                             ;ket thuc mot chuong trinh con
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con quet phim
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
key:        mov     r7,#0feh                    ;bat dau voi cot so 0(feh)
            mov     r6,#4               ;Su dung r6 lam bo dem
            mov     r5,#00

key1:       mov     mtphim,r7           ;xuat ma quet ra cot
            mov     a,mtphim            ;Doc lai port1 de xu ly tiep theo
            anl     a,#0f0h             ;xoa 4 bit thap la hang
            cjne    a,#0f0h,key2        ;co nhan fim thi nhay

            mov     a,r7
            rl      a                   ;xoay de chuyen den cot ke tiep
            mov     r7,a

            mov     a,r5                ;chuyen ma fim sang cot ke
            add     a,#4
            mov     r5,a

            djnz    r6,key1             ;Neu nhu sau moi lan 1 cot ma khong

            clr     c                   ;clr c neu nhu khong co phim duoc an
            mov     a,#0ffh             ;thoat voi ma trong a = FFh
            ret

key2:       swap    a
key4:       rrc     a                   ;xoay sang phai tim bit 0
            jnc     key3                ;nhay neu (c)=0
            inc     r5                  ;tang ma fim len cot ke
            sjmp    key4                ;tiep tuc cho den khi duoc (C)=0

key3:       mov     a,r5
            setb    c
            ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao du lieu ma phim
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:    db      0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h
            db      080h,090h,088h,083h,0c6h,0a1h,086h,08eh

        end
```

Nhận xét: khi viết đúng thì trên màn hình 8 led sẽ hiển 1 số 0 bên phải và 7 led còn lan lại tắt. Khi nhấn 1 phím bất kỳ thì các số hiện tại sẽ dịch sang trái, mã 7 đoạn của phím mới sẽ hiển thị ở led tận cùng bên phải. Chương trình này còn 2 khuyết điểm (1) không xoá được con số "0" vô nghĩa, (2) khi ta nhấn phím mà chưa buông phím thì 8 led đều tắt.

Chương trình sau đây sẽ khắc phục được 2 khuyết điểm trên:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh quet phim dung he thong 1 ma phim hien thi tren 8 led
;dung port3  chua qua IC dem ket noi voi pinhd cua ban fim
;dung port0 va port 2 ket noi dieu khien led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            quet    equ     p2
            led7    equ     p0
            mtphim  equ     p3              ;ket noi voi ma tran ban phim
            maso0   equ     0c0h            ;ma 7 doan cua so 0
```

```
                org     0000h                   ;dia chi bat dau cua chtr chinh
                ljmp    chtrchinh

                org     000bh                   ;khai bao dia chi bat dau cua chtr gat
                ljmp    ngat_timer0


chtrchinh:      mov     20h,#0ffh               ;tat cac led
                mov     21h,#0ffh
                mov     22h,#0ffh
                mov     22h,#0ffh
                mov     23h,#0ffh
                mov     24h,#0ffh
                mov     25h,#0ffh
                mov     26h,#0ffh
                mov     27h,#0c0h

                mov     dptr,#ma7doan
                mov     quet,#07fh              ;xuat ma quet cho 1 led sang
                mov     led7,#7fh

                mov     tmod,#00000001b         ;T0 mod 1 dem 16 bit
                mov     th0,#high(-500)
                mov     tl0,#low(-500)
                clr     tf0
                setb    tr0
                mov     ie,#10000010b           ;cho phep timer0 ngat

main:           lcall   xoaso0                  ;goi chtr xoa so 0 vo nghia
main1:          lcall   keypres         ;goi chtr con quet phim
                cjne    a,#0ffh,main2
                sjmp    main1

main2:          lcall   dichchuyen
                lcall   giaima                  ;goi chtr con giai ma hien thi
                sjmp    main
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con dich chuyen du lieu trong 8 o nho chua du lieu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
dichchuyen:     mov     20h,21h
                mov     21h,22h
                mov     22h,23h
                mov     23h,24h
                mov     24h,25h
                mov     25h,26h
                mov     26h,27h
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con giai ma fim nhan va hien thi ra 1 led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
giaima:         movc    a,@a+dptr
                mov     27h,a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con xoa so 0 vo nghia
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xoaso0:         mov     r0,#20h
xoaso_c:        mov     a,@r0
                cjne    a,#maso0,xoaso_e
                mov     @r0,#0ffh               ;nap FF de tat led
xoaso_a:        inc     r0
                cjne    r0,#27h,xoaso_c
```

```
xoaso_d:        ret

xoaso_e:        cjne    a,#0ffh,xoaso_d
                sjmp    xoaso_a
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con ngat cua timer0 de hien thi lien tuc
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ngat_timer0:    lcall   hthi
                mov     th0,#high(-500)
                mov     tl0,#low(-500)
                reti
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           push    acc                     ;cat tam thanh ghi a
                mov     a,#01111111b            ;ma quet
                mov     r0,#27h

ht1:            mov     led7,@r0
                mov     quet,a
                lcall   delay1
                mov     quet,#0ffh
                dec     r0
                rr      a                       ;chuyen sang led ke
                cjne    r0,#1fh,ht1
                pop     acc                     ;lay lai noi dung thanh ghi a

                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1:         mov     r7,#0h
                djnz    r7,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con quet phim va chong doi phim
;su dung cac thanh ghi:  R4, R5, R6, R7, A
;neu khong nhan thi (A) = FF, neu nhan thi (A) chua ma phim nhan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
keypres:        mov      r4,#10                 ;nhap so dem 10 lan
keypres1 :      lcall    KEY                    ;Neu co phim an thi co c=1
                jc       pn1                    ;kiem tra tiep neu c = 1
                ret                             ;Neu khong co phim nhan thi co c=0

pn1:            djnz     r4,keypres1            ;Quay ve lap lai chong nay
                push     acc                    ;Cat noi dung ma phim trong A

keypres2:       mov      r4,#10                 ;Nhap so dem 10 lan cho nha phim
keypres3:       lcall    key                    ;Co phim nhan hay khong
                jc       keypres2               ;Co thi kiem tra lai
                djnz     r4,keypres3            ;Khong thi lap lai 50 lan va dam bao
                pop      acc                    ;Khoi phuc lai gia tri cho A
                ret                             ;ket thuc mot chuong trinh con
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con quet phim
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
key:            mov     r7,#0feh                        ;bat dau voi cot so 0(feh)
                mov     r6,#4                   ;Su dung r6 lam bo dem
                mov     r5,#00

key1:           mov     mtphim,r7               ;xuat ma quet ra cot
                mov     a,mtphim                ;Doc lai port1 de xu ly tiep theo
```

```
                anl     a,#0f0h             ;xoa 4 bit thap la hang
                cjne    a,#0f0h,key2        ;co nhan fim thi nhay

                mov     a,r7
                rl      a                   ;xoay de chuyen den cot ke tiep
                mov     r7,a

                mov     a,r5                ;chuyen ma fim sang cot ke
                add     a,#4
                mov     r5,a

                djnz    r6,key1             ;Neu nhu sau moi lan 1 cot ma khong

                clr     c                   ;clr c neu nhu khong co phim duoc an
                mov     a,#0ffh             ;thoat voi ma trong a = FFh
                ret

key2:           swap    a
key4:           rrc     a                   ;xoay sang phai tim bit 0
                jnc     key3                ;nhay neu (c)=0
                inc     r5                  ;tang ma fim len cot ke
                sjmp    key4                ;tiep tuc cho den khi duoc (C)=0

key3:           mov     a,r5
                setb    c
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao du lieu ma phim
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:        db      0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h
                db      080h,090h,088h,083h,0c6h,0a1h,086h,08eh

    end
```

Trong chương trình này chúng ta đã sử dụng timer ngắt thường xuyên để quét led 7 đoạn hiển thị liên tục nên khi ta còn nhấn phím chưa buông tay thì chương trình ngắt hiển thị vẫn xảy ra nên led luôn sáng và không bị tắt như chương trình trên.

IV. Bài tập:

1. Hãy hiệu chỉnh chương trình trên để chỉ cho phép nhập các số thập phân từ 0 đến 9 (các nút nhấn từ "A" đến "F" không có tác dụng.
2. Sau khi thực hiện xong câu 1 thì viết thêm các yêu cầu như sau: phím "C" có tác dụng xoá toàn bộ và màn hình chỉ hiển thị đúng 1 số "0".
3. Tiếp tục thêm phím "D" có chức năng xoá số mới vừa nhập sau cùng.

Trong các chương trình đếm giây, đếm phút giây, đếm giờ phút giây chúng ta chưa sử dụng các phím để hiệu chỉnh các thông số thời gian cần thiết nên sau khi làm quen với chương trình con quét phím thì ta bắt đầu ứng dụng vào điều chỉnh các thông số cần thiết.

Chương trình đồng hồ số có chỉnh giờ phút giây bằng bàn phím ma trận:

Khi chạy chương trình thì màn hình 8 led sẽ hiển thị giờ phút giây mặc nhiên bắt đầu từ 00 00 00 - dấu chấm thập phân sẽ xuất hiện ở led hàng chục giờ và cho phép chỉnh giá trị hàng chục giờ. Nhấn phím chỉnh hàng chục giờ thì kết quả led sẽ hiển thị đúng đồng thời dịch chuyển dấu chấm đến led hàng đơn vị giờ. Tương tự như vậy cho led chục phút và đơn vị phút.

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dem dong ho so gio phut giay
;su dung ngat timer t0 de dem chinh xac ve thoi gian
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                giay    equ     r2              ;gan bien dem giay la R2
                phut    equ     r3              ;gan bien dem phut cho R3
                gio     equ     r4              ;gan bien dem gio cho r4
                bdn     equ     70h             ;gan bien dem ngat
                mtphim  equ     p3
                bvt     equ     71h             ;bien vi tri chinh cac thong so

                quet    equ     p2
                led7    equ     p0

                org     0000h                   ;bat dau chuong trinh
                ljmp     main                   ;nhay den chtr chinh

                org     000bh
                ljmp    int_t0                  ;nhay den chtr con ngat timer0

main:           mov     sp,#50h                 ;khoi tao ngan xep

                mov     22h,#0ffh
                mov     25h,#0ffh
                mov     tmod,#01h               ;timer0: mod 1 - dem 16 bit
                mov     dptr,#ma7doan           ;dptr quan ly vung ma 7 doan

                clr     tf0                     ;xoa co tran timer 0
                mov     ie,#10000010B           ;cho phep timer 0 ngat
                mov     th0,#high(-50000)       ;khoi tao timer delay 50ms
                mov     tl0,#low(-50000)
                setb    tr0                     ;cho phep timer0 bat dau dem

                mov     bvt,#20h                ;20h=chucgio; 21h=dvigio;  23h=chucphut; 24h=dviphut
main4:          mov     gio,#00                 ;gio=00
main3:          mov     phut,#00h               ;phut=00
main0:          mov     giay,#00h               ;giay=00
main1:
                lcall   gma
main2:          lcall   hthi                    ;goi chtr con hien thi
                mov     a,bdn
                cjne    a,#20,main5             ;chua dung 20 lan [tuc 1 giay]
main5:          jc      main2_key               ;nhay neu nho hon 20
                clr     c
                subb    a,#20
                mov     bdn,a

                mov     a,giay                  ;chuyen  giay sang A
                add     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

                mov     a,phut                  ;chuyen  phut sang A
                add     a,#1                    ;tang phut len 1
                da      a                       ;hieu chinh so BCD trong A
                mov     phut,a                  ;tra lai cho phut
                cjne    phut,#60h,main0                 ;ss phut voi 60

                mov     a,gio                   ;chuyen gio sang A
                add     a,#1                    ;tang gio len 1
                da      a                       ;hieu chinh so BCD trong A
```

```
                mov     gio,a                   ;tra lai cho gio
                cjne    gio,#24h,main3          ;ss gio voi 24

                ljmp    main4                   ;lam lai tu dau

main2_key:      lcall   key_chinh
                sjmp    main2
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con chinh gio phut giay tu bang ban phim
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

key_chinh:      setb    rs0                     ;chon bank1 gan cho cac thanh ghi R
                lcall   keypres         ;goi quet phim
                clr     rs0                     ;chon lai bank0
                cjne    a,#0ffh,key_chinha
                ret

key_chinha:
                push    acc                     ;cat tam a
                mov     a,bvt                   ;chuyen bvt  sang a
                cjne    a,#20h,key_chinhb

                mov     a,gio                   ;chinh hang chuc gio
                anl     a,#0fh
                mov     gio,a
                pop     acc
                swap    a
                orl     a,gio
                mov     gio,a
                mov     bvt,#21h                ;chuyen sang hang don vi gio
                ret

key_chinhb:     cjne    a,#21h,key_chinhc
                mov     a,gio                   ;chinh hang don vi gio
                anl     a,#0f0h
                mov     gio,a
                pop     acc
                orl     a,gio
                mov     gio,a
                mov     bvt,#23h                ;chuyen sang hang chuc phut
                ret

key_chinhc:     cjne    a,#23h,key_chinhd
                mov     a,phut                  ;chinh hang chuc phut
                anl     a,#0fh
                mov     phut,a
                pop     acc
                swap    a
                orl     a,phut
                mov     phut,a
                mov     bvt,#24h                ;chuyen sang hang don vi phut
                ret

key_chinhd:
                mov     a,phut                  ;chinh hang don vi gio
                anl     a,#0f0h
                mov     phut,a
                pop     acc
                orl     a,phut
                mov     phut,a
                mov     bvt,#20h                ;chuyen sang hang chuc gio
                ret
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con giai ma
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
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 27h

            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             ;cat vao o nho 26h

            mov   a,phut
            anl   a,#0fh            ;xoa 4 bit cao hang chuc phut
            movc  a,@a+dptr         ;lay ma 7 doan
            mov   24h,a             ;cat ma vao o nho 25h

            mov   a,phut
            anl   a,#0f0h           ;xoa 4 bit thap hang dvi phut
            swap  a                 ;chuyen 4 bit cao xuong vi tri thap
            movc  a,@a+dptr         ;lay ma 7 doan  hang chuc
            mov   23h,a             ;cat vao o nho 24h

            mov   a,gio
            anl   a,#0fh            ;xoa 4 bit cao hang chuc gio
            movc  a,@a+dptr         ;lay ma 7 doan
            mov   21h,a             ;cat ma vao o nho 23h

            mov   a,gio
            anl   a,#0f0h           ;xoa 4 bit thap hang dvi gio
            swap  a                 ;chuyen 4 bit cao xuong vi tri thap
            movc  a,@a+dptr         ;lay ma 7 doan  hang chuc
            mov   20h,a             ;cat vao o nho 22h

;xu ly dau cham tuong ung voi bien vi tri
            mov   r0,bvt
            mov   a,@r0
            anl   a,#7fh            ;cho dau cham sang
            mov   @r0,a

            ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;CHUONG TRINH CON NGAT TIMER0 SAU KHOANG THOI GIAN 50MS
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
int_t0:     inc    bdn              ;tang bien dem ngat
            mov   th0,#high(-50000)         ;khoi tao lai cho timer delay 50ms
            mov   tl0,#low(-50000)
            clr   tf0
            reti
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:       mov   a,#01111111b      ;ma quet
            mov   r0,#27h

ht1:        mov   led7,@r0
            mov   quet,a
            lcall delay1
            mov   quet,#0ffh
```

```
                dec     r0
                rr      a                       ;chuyen sang led ke
                cjne    r0,#1Fh,ht1
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1:         mov     7fh,#0fh
                djnz    7fh,$
                ret

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con quet phim va chong doi phim
;su dung cac thanh ghi:  R4, R5, R6, R7, A
;neu khong nhan thi (A) = FF, neu nhan thi (A) chua ma phim nhan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
keypres:        mov     r4,#10                  ;nhap so dem 10 lan
keypres1 :      lcall   KEY                     ;Neu co phim an thi co c=1
                jc      pn1                     ;kiem tra tiep neu c = 1
                ret                             ;Neu khong co phim nhan thi co c=0

pn1:            djnz    r4,keypres1             ;Quay ve lap lai chong nay
                push    acc                     ;Cat noi dung ma phim trong A

keypres2:       mov     r4,#10                  ;Nhap so dem 10 lan cho nha phim
keypres3:       lcall   key                     ;Co phim nhan hay khong
                jc      keypres2                ;Co thi kiem tra lai
                djnz    r4,keypres3             ;Khong thi lap lai 50 lan va dam bao
                pop     acc                     ;Khoi phuc lai gia tri cho A
                ret                             ;ket thuc mot chuong trinh con
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con quet phim
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
key:            mov     r7,#0feh                        ;bat dau voi cot so 0(feh)
                mov     r6,#4                   ;Su dung r6 lam bo dem
                mov     r5,#00

key1:           mov     mtphim,r7               ;xuat ma quet ra cot
                mov     a,mtphim                ;Doc lai port1 de xu ly tiep theo
                anl     a,#0f0h                 ;xoa 4 bit thap la hang
                cjne    a,#0f0h,key2            ;co nhan fim thi nhay

                mov     a,r7
                rl      a                       ;xoay de chuyen den cot ke tiep
                mov     r7,a

                mov     a,r5                    ;chuyen ma fim sang cot ke
                add     a,#4
                mov     r5,a

                djnz    r6,key1                 ;Neu nhu sau moi lan 1 cot ma khong

                clr     c                       ;clr c neu nhu khong co phim duoc an
                mov     a,#0ffh                 ;thoat voi ma trong a = FFh
                ret

key2:           swap    a
key4:           rrc     a                       ;xoay sang phai tim bit 0
                jnc     key3                    ;nhay neu (c)=0
                inc     r5                      ;tang ma fim len cot ke
                sjmp    key4                    ;tiep tuc cho den khi duoc (C)=0
```

```
key3:           mov     a,r5
                setb    c
                ret

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao ma 7 doan tu so '0' den so '9'
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:        db      0C0h,0F9h,0A4h,0B0h,99h,92h,82h,0F8h,80h,90h
end
```

Chương trình còn nhiều vấn đề cần xử lý thêm như:

2. Chỉnh thêm phần giây.
3. Chưa xử lý dữ liệu khi nhập sai ví dụ như khi ta chỉnh phút vượt quá con số 59.
4. Khi nhấn nút mà chưa buông thì tất cả các led đều tắt.

Bạn hãy xử lý từng bước để được chương trình hoàn chỉnh.

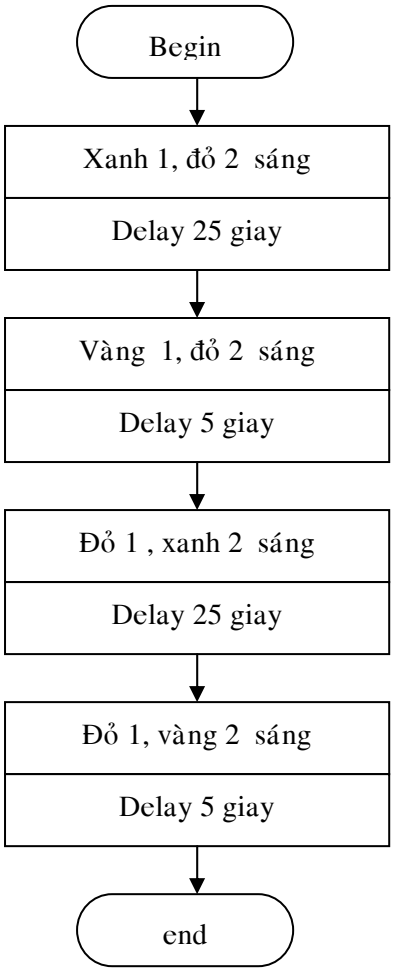
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 3-5** CHƯƠNG TRÌNH ĐIỀU KHIỂN ĐÈN GIAO THÔNG XANH – VÀNG – ĐỎ TRÊN 8 LED.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách viết chương trình điều khiển đèn giao thông.

II. Trình tự thực hiện:

1. Yêu cầu: Viết chương trình điều khiển đèn giao thông cho 1 ngã tư gồm Xanh1, Vàng1, Đỏ1, Xanh2, Vàng2, Đỏ2. Cho thời gian xanh sáng 25 giây, thời gian vàng sáng 5 giây, thời gian đỏ bằng thời gian xanh cộng thời gian vàng. Các đèn xanh vàng đỏ dùng led đơn để hiển thị, các thông số thời gian dùng led 7 đoạn để hiển thị.
2. Giải thuật:



3. 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.
   - Dùng bus dây kết nối port 2 đến pinhd điều khiển quét hàng từ T0 đến T7.
   - Dùng bus dây kết nối port 1 đến pinhd điều khiển 8 led đơn có sẵn led xanh vàng đỏ.
4. Khởi động phần mềm, soạ thảo chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh  dieu khien den giao thong co hien thi thong so thoi gian tren 2 led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;du lieu dieu khien den cho bo thi nghiem dieu khien led don tich cuc muc 0
                X1_d2           EQU     11110011B       ;XANH 1, DO 2 SANG
                V1_d2           EQU     11101011B       ;VANG 1, DO 2 SANG

                d1_X2           EQU     11011110B       ;DO 1, XANH 2 SANG
                D1_V2           EQU     11011101B       ;DO 1, VANG 2 SANG

;du lieu dieu khien den cho bo thi nghiem dieu khien led don tich cuc muc 1
;               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     24              ;24 dem xuong 0 tuc da dem 25
        tg_vang         equ     4               ;4 dem xuong 0 tuc da dem 5
        tg_do           equ     29              ;29 dem xuong 0 tuc da dem 30

        led7            equ     p0              ;dieu khien cac doan a,b,c,d,e,f,g,dp
        quet            equ     p2              ;dieu khien quet cac transistor T0 den T7
        leddonto        equ     p1              ;dieu khien led don

;bat dau chuong trinh chinh

                org     0000h
                mov     tmod,#00000001b
                mov     dptr,#ma7doan

                mov     22h,#0ffh               ;xoa cac vung nho hien thi khong
                mov     23h,#0ffh               ;dung de tat led
                mov     24h,#0ffh
                mov     25h,#0ffh

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xanh1 va do2 sang 25 giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
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
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;vang1 va do2 sang 5 giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

                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 25 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
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                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

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con chuyen so hex sang so bcd
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
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

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con giai ma
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
gma:            mov     a,37h
                anl     a,#0fh
                movc    a,@a+dptr
                mov     27h,a           ;hang don vi

                mov     a,37h
                anl     a,#0f0h
                swap    a
                movc    a,@a+dptr
                mov     26h,a           ;hang chuc

                mov     a,36h
                anl     a,#0fh
                movc    a,@a+dptr
                mov     21h,a           ;hang don vi

                mov     a,36h
                anl     a,#0f0h
                swap    a
                movc    a,@a+dptr
                mov     20h,a           ;hang chuc
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh delay co goi chuong trinh hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
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

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           mov     r1,#01111111b   ;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
                rr      a
                mov     r1,a

                djnz    r5,ht1          ;chi co 2 so nen so sanh voi 62H de ket thuc
                ret

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
;chuong trinh con delay1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1:         mov     r7,#0fh
                djnz    r7,$
                ret

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;vung ma 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:        db      0c0h,0f9h,0a4h,0b0h,099h,092h,082h,0f8h,080h,090h

end
```

III. BÀI TẬP:

1. Hãy kết hợp với chương trình con quét phím để có thể thay đổi các thông số thời gian tuỳ ý nằm trong giới hạn tối đa là 99.
2. Hãy mở rộng hệ thống trên để điều khiển 2 hệ thống đèn giao thông cho 2 ngã tư : ngã tư thứ I và ngã tư thứ II. Ngã tư thứ II trể hơn ngã tư thứ I đúng bằng 10 giây. Cả 2 đều có thời gian điều khiển giống như trên.
3. Hãy mở rộng thêm bài trên dùng 4 led chính giữa hiển thị giờ – phút và thời gian từ 22 giờ đến 06 giờ thì đèn vàng nhấp nháy 1 giây.

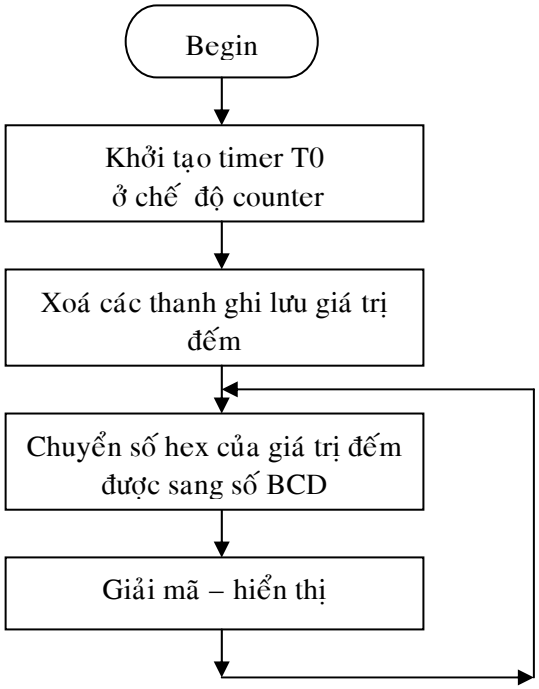
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 3-6** CHƯƠNG TRÌNH ĐẾM SẢN PHẨM DÙNG COUNTER T0.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách sử dụng counter để đếm xung ngoại. Ứng dụng dùng để đếm sản phẩm.

II. Trình tự thực hiện:

1. Yêu cầu: sử dụng một mạch thu phát hồng ngoại gồm có 1 led phát và 1 led thu để tạo ra xung khi có sản phẩm đi qua hoặc mạch tạo xung vuông có tần số thấp để nhìn thấy.
2. Giải thuật:

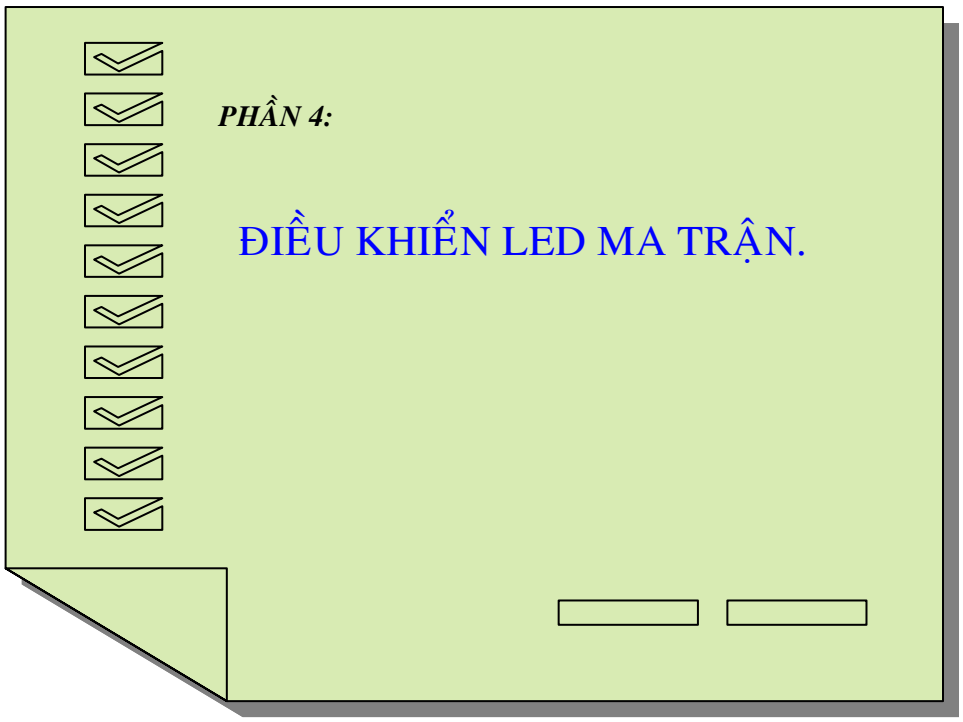


3. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 0 đến pinhd điều khiển các đoạn a, b, c, d, e, f, g, dp.
   - Dùng bus dây kết nối port 2 đến pinhd điều khiển quét hàng từ T0 đến T7.
   - Nguồn phát tín hiệu xung đến ngõ vào counter T0 của port3.
4. Khởi động phần mềm, soạ thảo chương trình sau:

III. BÀI TẬP:

4. Hãy kết hợp với chương trình con quét phím để có thể thay đổi các thông số thời gian tuỳ ý nằm trong giớ hạn tối đa là 99.
5. Hãy mở rộng hệ thống trên để điều khiển 2 hệ thống đèn giao thông cho 2 ngã tư : ngã tư thứ I và ngã tư thứ II. Ngã tư thứ II trể hơn ngã tư thứ I đúng bằng 10 giây. Cả 2 đều có thời gian điều khiển giống như trên.

6. Hãy mở rộng thêm bài trên dùng 4 led chính giữa hiển thị giờ – phút và thời gian từ 22 giờ đến 06 giờ thì đèn vàng nhấp nháy 1 giây.



PHẦN 4:

ĐIỀU KHIỂN LED MA TRẬN.

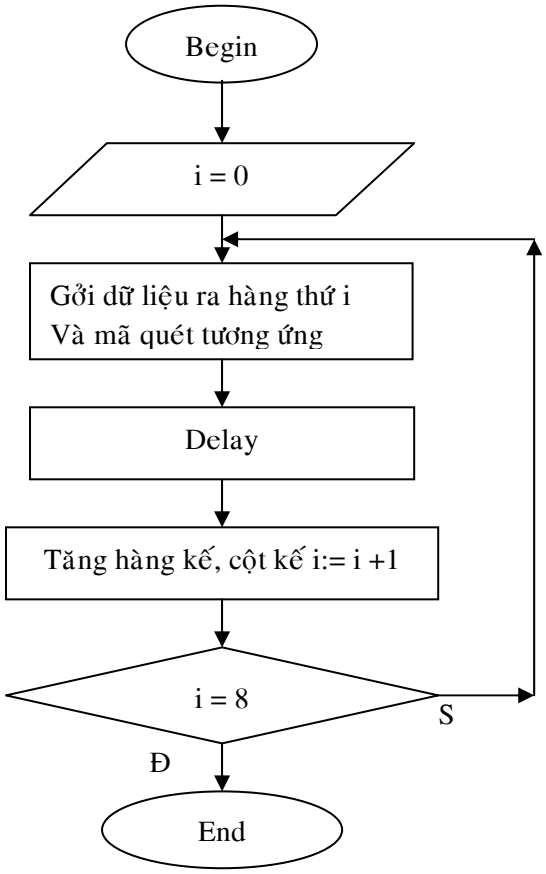
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 4-1** CHƯƠNG TRÌNH HIỂN THỊ KÍ TỰ TRÊN LED MA TRẬN	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Hiểu được cách viết chương trình điều khiển led ma trận sáng 1 kí tự đứng yên.

II. Trình tự thực hiện:

1. Giải thuật:



2. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 3 đến pinhd điều khiển hàng H7 – H0 của led ma trận.
   - Dùng bus dây kết nối port 0 đến pinhd điều khiển cột màu xanh X7 – X0.
   - Dùng bus dây kết nối port 2 đến pinhd điều khiển cột màu đỏ Đ7 – Đ0.
3. Khởi động phần mềm, soạn thảo chương trình như sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh hien thi ki tu chu A tren ma tran led dung yen
;dung port 3 ket noi dieu khien hang
;ma chu A = 07H,0DBH,0DDH,0DBH,07H
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                hang    equ     p3
                cotx    equ     p1
                cotd    equ     p2
                org     0000h

                mov     cotx,#0                 ;tat quet neu co ket noi
                mov     cotd,#0                 ;tat quet neu co ket noi

main:           mov     hang,#07h               ;goi du lieu ra hang 1
                mov     cotx,#00000001b         ;goi ma quet cho 1 transistor dan
                lcall   delay
                mov     cotx,#00h               ;chong lem

                mov     hang,#0DBh              ;goi du lieu ra hang 2
                mov     cotx,#00000010b
                lcall   delay
                mov     cotx,#00h

                mov     hang,#0DDh              ;goi du lieu re hang 3
                mov     cotx,#00000100b
                lcall   delay
                mov     cotx,#00h

                mov     hang,#0DBh              ;goi du lieu ra hang 4
                mov     cotx,#00001000b
                lcall   delay
                mov     cotx,#00h

                mov     hang,#07h               ;goi du lieu ra hang 5
                mov     cotx,#00010000b
                lcall   delay
                mov     cotx,#00h

                sjmp    main
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r5,#1
de:             mov     r6,#50
                djnz    r6,$
                djnz    r5,de
                ret
end
```

4. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch chạy đúng yêu cầu.

III. Câu hỏi:

1. Tại sao phải dùng chương trình con delay, nếu không trả lời được hãy thử bỏ lệnh gọi chương trình con delay và xem kết quả hiển thị?
2. Tại sao ta phải gởi 00 ra port 2 sau lệnh lcall delay, nếu không trả lời được thì hãy bỏ lệnh Mov cotx,#00 có trong chương trình và xem kết quả hiển thị?

IV. Bài tập ứng dụng:

1. Muốn hiển thị chữ A bắt đầu từ cột thứ 2 trở đi thì làm thế nào?
2. Hãy viết chương trình hiển thị kí tự M hoặc các kí tự khác.
3. Hãy viết chương trình hiển thị số 2 hoặc các con số khác.

Chú ý: hãy chạy chương trình tạo mã ma trận để lấy các mã cho các kí tự tuỳ ý và các số nằm bên dưới

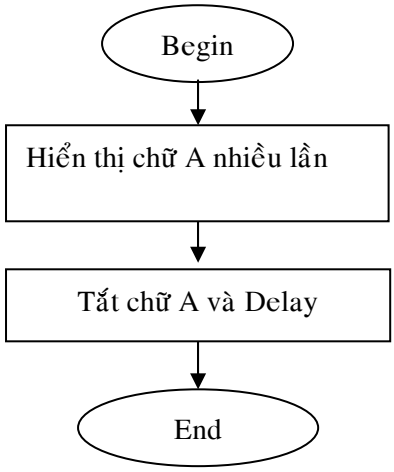
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 4-2** CHƯƠNG TRÌNH CHỚP TẮT KÍ TỰ A TRÊN LED MA TRẬN	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách viết chương trình điều khiển led ma trận sáng 1 kí tự nhấp nháy.

II. Trình tự thực hiện:

1. Giải thuật:



2. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 3 đến pinhd điều khiển hàng H7 – H0 của led ma trận.
   - Dùng bus dây kết nối port 0 đến pinhd điều khiển cột màu xanh X7 – X0.
   - Dùng bus dây kết nối port 2 đến pinhd điều khiển cột màu đỏ Đ7 – Đ0.
3. Khởi động phần mềm, soạn thảo chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh hien thi ki tu chu A nhap nhay tren ma tran led
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                hang    equ     p3              ;hang ket noi voi port 3
                cotx    equ     p0              ;cot xanh ket noi voi port 0
                cotd    equ     p2              ;cot do ket noi voi port 2

                org     0000h

                mov     cotx,#0                 ;tat quet neu co ket noi
                mov     cotd,#0                 ;tat quet neu co ket noi

                mov     dptr,#machu_a
main:           mov     r2,#250
mainb:          mov     r1,#00
                mov     r3,#00000001b           ;ma quet

mainc:          mov     a,r1
                movc    a,@a+dptr
```

```
                mov     hang,a

                mov     cotx,r3
                lcall   delay
                mov     cotx,#00h

                inc     r1                  ;tang len de lay byte ke
                mov     a,r3                ;dich chuyen ma quet
                rl      a
                mov     r3,a

                cjne    r1,#5,mainc         ;chi co 5 byte
                djnz    r2,mainb

                lcall   delay1s             ;goi chtr con delay 1s de tat led
                sjmp    main

delay:          mov     r6,#1
de:             mov     r7,#50
                djnz    r7,$
                djnz    r6,de
                ret

delay1s:        mov      r6,#0
pnde:           mov     r7,#0
                djnz    r7,$
                djnz    r6,pnde
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao ma chu A
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
machu_a:        db      007H,0DBH,0DDH,0DBH,007H
                end
```

4. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch chạy đúng yêu cầu.

III. Câu hỏi:

1. Hãy giải thích tại sao khi muốn điều khiển tắt led ta chỉ thực hiện lệnh gọi chương trình con delay.
2. Hãy cho biết hai chương trình con delay và delay1s cùng sử dụng các thanh ghi R6, R7 có ảnh hưởng gì hay không – giải thích?

IV. Các chương trình mẫu:

Chương trình hiển thị kí tự chữ A hai màu xanh đỏ chớp tắt trên led ma trận

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh hien thi ki tu chu A hai mau xanh va do va chop tat
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                hang    equ     p3      ;hang ket noi voi port 3
                cotx    equ     p0      ;cot xanh ket noi voi port 0
                cotd    equ     p2      ;cot do ket noi voi port 2

                  org   0000h

                mov     cotx,#0         ;tat quet neu co ket noi
                mov     cotd,#0         ;tat quet neu co ket noi

main:           mov     dptr,#machu_a
                lcall   htxanh
                lcall   htdo
                sjmp    main
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;hien thi mau xanh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
htxanh:     mov     r2,#250
mainb:      mov     r1,#00
            mov     r3,#00000001b       ;ma quet

mainc:      mov     a,r1
            movc    a,@a+dptr
            mov     hang,a

            mov     cotx,r3
            lcall   delay
            mov     cotx,#00h

            inc     r1                  ;tang len de lay byte ke
            mov     a,r3                ;dich chuyen ma quet
            rl      a
            mov     r3,a

            cjne    r1,#5,mainc
            djnz    r2,mainb

            lcall   delay1s     ;goi chtr con delay 1s de tat led
            ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;hien thi mau do
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

htdo:       mov     r2,#250
maind:      mov     r1,#00
            mov     r3,#00000001b       ;ma quet

maine:      mov     a,r1
            movc    a,@a+dptr
            mov     hang,a

            mov     cotd,r3
            lcall   delay
            mov     cotd,#00h

            inc     r1                  ;tang len de lay byte ke
            mov     a,r3                ;dich chuyen ma quet
            rl      a
            mov     r3,a

            cjne    r1,#5,maine
            djnz    r2,maind

            lcall   delay1s     ;goi chtr con delay 1s de tat led
            ret


delay:      mov     r5,#1
de:         mov     r6,#50
            djnz    r6,$
            djnz    r5,de
            ret

delay1s:    mov     r5,#0
pnde:       mov     r6,#0
```

```
                djnz    r6,$
                djnz    r5,pnde
                ret

machu_a:        db      007H,0DBH,0DDH,0DBH,007H
                end
```

Chương trình đếm từ 0 đến 9 hiển thị trên led ma trận

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh hien thi so dem tu 0 den 9 tren led ma tran
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                hang            equ     p3              ;hang ket noi voi port 3
                cotx            equ     p0              ;cot xanh ket noi voi port 0
                cotd            equ     p2              ;cot do ket noi voi port 2

                solanhthi       equ     r2              ;luu so lan hien hti
                sokitu          equ     r0
                maquet          equ     r3
                bdem8byte       equ     r1              ;dem dem so byte cua 1 ki tu

                org     0000h

                mov     cotx,#0                         ;tat quet neu co ket noi
                mov     cotd,#0                         ;tat quet neu co ket noi

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

main:           mov     dptr,#machu_so                  ;nap dia chi machu_so vao dptr
                mov     sokitu,#10                      ;co 10 con so

maina:          mov     solanhthi,#250
mainb:          mov     bdem8byte,#00
                mov     maquet,#00000001b               ;ma quet

mainc:          mov     a,bdem8byte
                movc    a,@a+dptr
                mov     hang,a

                mov     cotx,r3
                lcall   delay
                mov     cotx,#00h

                inc     bdem8byte                       ;tang len de lay byte ke
                mov     a,maquet                        ;dich chuyen ma quet
                rl      a
                mov     maquet,a

                cjne    r1,#5,mainc             ;chi co 5 byte
                djnz    solanhthi,mainb

                mov     a,dpl
                add     a,#8
                mov     dpl,a

                djnz    sokitu,maina
                sjmp    main

delay:          mov     r5,#1
de:             mov     r6,#50
                djnz    r6,$
```

```
                djnz    r5,de
                ret

delay1s:        mov     r5,#0
pnde:           mov     r6,#0
                djnz    r6,$
                djnz    r5,pnde
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao vung ma cac so tu 0 den 9
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
machu_so:       db      083h,07dh,07dh,07dh,083h,0ffh,0ffh,0ffh         ;ma so 0
                db      077h,07bh,001h,07fh,07fh,0ffh,0ffh,0ffh         ;ma so 1
                db      01bh,05dh,05dh,05dh,063h,0ffh,0ffh,0ffh         ;ma so 2
                db      0bbh,06dh,06dh,06dh,093h,0ffh,0ffh,0ffh         ;ma so 3
                db      0cfh,0d7h,0dbh,001h,0dfh,0ffh,0ffh,0ffh         ;ma so 4
                db      0b1h,075h,075h,075h,08dh,0ffh,0ffh,0ffh         ;ma so 5
                db      083h,06dh,06dh,06dh,09bh,0ffh,0ffh,0ffh         ;ma so 6
                db      07dh,0bdh,0ddh,0edh,0f1h,0ffh,0ffh,0ffh         ;ma so 7
                db      093h,06dh,06dh,06dh,093h,0ffh,0ffh,0ffh         ;ma so 8
                db      0b3h,06dh,06dh,06dh,083h,0ffh,0ffh,0ffh         ;ma so 9
                end
```

Trong chương trình này với số lần lặp tối đa là 255 (trong bài viết 250) nên thời gian led hiển thị rất ngắn ta khó nhìn thấy. Để tăng thời gian hiển thị dài hơn ta có thể thêm 1 biến nữa để số lần lặp lớn hơn 255, tuy nhiên muốn hiển thị đếm chính xác thì cách viết sau là tốt nhất:

Chương trình đếm từ 0 đến 9 hiển thị trên led ma trận chính xác 1 giây

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh hien thi so dem tu 0 den 9 tren led ma tran chinh xac thoi gian 1 giay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                hang            equ     p3              ;hang ket noi voi port 3
                cotx            equ     p0              ;cot xanh ket noi voi port 0
                cotd            equ     p2              ;cot do ket noi voi port 2

                solanhthi       equ     r2              ;luu so lan hien hti
                sokitu          equ     r0
                maquet          equ     r3
                bdem8byte       equ     r1              ;dem dem so byte cua 1 ki tu

                org     0000h

                mov     cotx,#0                         ;tat quet neu co ket noi
                mov     cotd,#0                         ;tat quet neu co ket noi

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

main:           mov     dptr,#machu_so                  ;nap dia chi machu_so vao dptr
                mov     sokitu,#10                      ;co 10 con so

maina:          mov     solanhthi,#250
mainb:          mov     bdem8byte,#00
                mov     maquet,#00000001b               ;ma quet

mainc:          mov     a,bdem8byte
                movc    a,@a+dptr
                mov     hang,a

                mov     cotx,r3
                lcall   delay
                mov     cotx,#00h
```

```
                inc     bdem8byte                   ;tang len de lay byte ke
                mov     a,maquet                    ;dich chuyen ma quet
                rl      a
                mov     maquet,a

                cjne    r1,#5,mainc                 ;chi co 5 byte
                djnz    solanhthi,mainb

                mov     a,dpl
                add     a,#8
                mov     dpl,a

                djnz    sokitu,maina
                sjmp    main

delay:          mov     r5,#1
de:             mov     r6,#50
                djnz    r6,$
                djnz    r5,de
                ret

delay1s:        mov      r5,#0
pnde:           mov     r6,#0
                djnz    r6,$
                djnz    r5,pnde
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao vung ma cac so tu 0 den 9
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
machu_so:       db      083h,07dh,07dh,07dh,083h,0ffh,0ffh,0ffh         ;ma so 0
                db      077h,07bh,001h,07fh,07fh,0ffh,0ffh,0ffh         ;ma so 1
                db      01bh,05dh,05dh,05dh,063h,0ffh,0ffh,0ffh         ;ma so 2
                db      0bbh,06dh,06dh,06dh,093h,0ffh,0ffh,0ffh         ;ma so 3
                db      0cfh,0d7h,0dbh,001h,0dfh,0ffh,0ffh,0ffh         ;ma so 4
                db      0b1h,075h,075h,075h,08dh,0ffh,0ffh,0ffh         ;ma so 5
                db      083h,06dh,06dh,06dh,09bh,0ffh,0ffh,0ffh         ;ma so 6
                db      07dh,0bdh,0ddh,0edh,0f1h,0ffh,0ffh,0ffh         ;ma so 7
                db      093h,06dh,06dh,06dh,093h,0ffh,0ffh,0ffh         ;ma so 8
                db      0b3h,06dh,06dh,06dh,083h,0ffh,0ffh,0ffh         ;ma so 9
                end
```

V. Bài tập ứng dụng:

1. Hãy viết chương trình hiển thị kí tự A sáng một giây, kí tự B sáng một giây và lặp lại.
2. Hãy viết chương trình đếm lên từ 0 đến F và từ F về 0 hiển thị trên led ma trận.
3. Giống như bài 2 nhưng hiển thị màu xanh và màu đỏ.

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 4-3** CHƯƠNG TRÌNH HIỂN THỊ CHUỖI "SPKT" TRÊN LED MA TRẬN	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách viết chương trình điều khiển led ma trận hiển thị một chuỗi kí tự dịch chuyển.

II. Trình tự thực hiện:

1. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 3 đến pinhd điều khiển hàng H7 – H0 của led ma trận.
   - Dùng bus dây kết nối port 0 đến pinhd điều khiển cột màu xanh X7 – X0.
   - Dùng bus dây kết nối port 2 đến pinhd điều khiển cột màu đỏ Đ7 – Đ0.
2. Khởi động phần mềm, biên soạn chương trình như sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh chay chuoi SPKT tu trai sang phai
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                hang    equ     p3      ;hang ket noi voi port 3
                cotx    equ     p0      ;cot xanh ket noi voi port 0
                cotd    equ     p2      ;cot do ket noi voi port 2

                org     0000h
                mov     cotx,#0                 ;tat quet neu co ket noi
                mov     cotd,#0                 ;tat quet neu co ket noi

main:           mov     dptr,#mach
maina:          mov     r2,#150
mainb:          mov     r1,#00
                mov     r3,#00000001b           ;ma quet

mainc:          mov     a,r1
                movc    a,@a+dptr
                mov     hang,a

                mov     cotx,r3
                lcall   delay
                mov     cotx,#00h

                inc     r1                      ;tang len de lay byte ke
                mov     a,r3                    ;dich chuyen ma quet
                rl a
                mov     r3,a

                cjne    r1,#8,mainc
                djnz    r2,mainb

                inc     dptr
                mov     a,dpl
                cjne    a,#32,maina
                ljmp    main
```

```
delay:      mov     r5,#1
de:         mov     r6,#50
            djnz    r6,$
            djnz    r5,de
            ret

            org     0500h
mach:  db   0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh          ;8 byte ff de tat 8 led
       db   0B3H,06DH,06DH,06DH,09BH,0ffh                    ;ma chu S
       db   001H,0EDH,0EDH,0EDH,0F3H,0ffh                    ;ma chu P
       db   001H,0EFH,0D7H,0BBH,07DH,0ffh                    ;ma chu K
       db   0FDH,0FDH,001H,0FDH,0FDH,0ffh                    ;ma chu T
       db   0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh          ;8 byte ff de tat 8 led
            end
```

3. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch chạy đúng yêu cầu.
4. Giải thích nguyên lý hoạt động của chương trình:
   - Chương trình thực hiện công việc lấy 8 byte dữ liệu bắt đầu từ 0500H đến 0507H gởi ra điều khiển 8 cột của led ma trận nhiều lần (trong chương trình viết là 150 lần).
   - Sau đó lặp lại công việc giống như trên nhưng lấy 8 byte dữ liệu bắt đầu từ 0501H đến 0508H, ở lần này ta sẽ thấy có 1 cột bên phải sáng – đó chính là mã đầu tiên của chữ S.
   - Tiếp tục thực hiện cho đến khi hết dữ liệu.
   - Do chương trình không phức tạp nên tác giả không viết lưu đồ, bạn hãy tự viết thử.

III. Câu hỏi:

1. Muốn cho chuỗi chạy nhanh hơn hoặc chậm hơn thì phải làm gì ?

IV. Các chương trình mẫu:

Chương trình điều khiển led ma trận hiển thị 2 màu xanh đỏ chuỗi SPKT:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh chay chuoi SPKT tu trai sang phai hai mau xanh do
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            hang        equ     p3          ;hang ket noi voi port 3
            cotx        equ     p0          ;cot xanh ket noi voi port 0
            cotd        equ     p2          ;cot do ket noi voi port 2

            solanhthi   equ     r2          ;luu so lan hien hti
            maquet      equ     r3
            bdem8byte   equ     r1          ;dem dem so byte cua 1 ki tu
            bdn         equ     20h

            org     0000h

            mov     cotx,#0                 ;tat quet neu co ket noi
            mov     cotd,#0                 ;tat quet neu co ket noi

main:       lcall   htdo                    ;goi chuong trinh con chay chu mau do
            lcall   htxanh                  ;goi chuong trinh con chay chu mau xanh
            sjmp    main
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;hien thi mau do
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
htdo:       mov     dptr,#machuoi
maina:      mov     solanhthi,#150
mainb:      mov     bdem8byte,#00
            mov     maquet,#00000001b       ;ma quet
```

```
mainc:      mov     a,bdem8byte
            movc    a,@a+dptr
            mov     hang,a

            mov     cotx,r3
            lcall   delay
            mov     cotx,#00h

            inc     bdem8byte           ;tang len de lay byte ke
            mov     a,maquet            ;dich chuyen ma quet
            rl      a
            mov     maquet,a

            cjne    bdem8byte,#8,mainc
            djnz    solanhthi,mainb

            inc     dptr
            mov     a,dpl
            cjne    a,#32,maina
            ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;hien thi mau xanh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
htxanh:     mov     dptr,#machuoi

maini:      mov     solanhthi,#150
mainh:      mov     bdem8byte,#00
            mov     maquet,#00000001b   ;ma quet

maing:      mov     a,maquet
            movc    a,@a+dptr
            mov     hang,a

            mov     cotd,r3
            lcall   delay
            mov     cotd,#00h

            inc     bdem8byte           ;tang len de lay byte ke
            mov     a,maquet            ;dich chuyen ma quet
            rl a
            mov     maquet,a

            cjne    bdem8byte,#8,maing
            djnz    solanhthi,mainh

            inc     dptr
            mov     a,dpl
            cjne    a,#32,maini
            ret

delay:      mov     r5,#1
de:         mov     r6,#50
            djnz    r6,$
            djnz     r5,de
            ret


            org     0500h
machuoi:    db      0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh     ;8 byte ff de tat 8 led
            db      0B3H,06DH,06DH,06DH,09BH,0ffh               ;ma chu S
            db      001H,0EDH,0EDH,0EDH,0F3H,0ffh               ;ma chu P
            db      001H,0EFH,0D7H,0BBH,07DH,0ffh               ;ma chu K
```

```
                db      0FDH,0FDH,001H,0FDH,0FDH,0ffh           ;ma chu T
                db      0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh ;8 byte ff de tat 8 led
                end
```

Chương trình điều khiển led ma trận hiển thị 3 màu xanh đỏ cam chuỗi SPKT:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh chay chuoi SPKT tu trai sang phai 3 mau xanh do cam
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                hang            equ     p3      ;hang ket noi voi port 3
                cotx            equ     p0      ;cot xanh ket noi voi port 0
                cotd            equ     p2      ;cot do ket noi voi port 2

                solanhthi       equ     r2      ;luu so lan hien hti
                maquet          equ     r3
                bdem8byte       equ     r1      ;dem dem so byte cua 1 ki tu
                bdn             equ     20h

                org     0000h

                mov     cotx,#0                 ;tat quet neu co ket noi
                mov     cotd,#0                 ;tat quet neu co ket noi

main:           lcall   htdo                    ;goi chuong trinh con chay chu mau do
                lcall   htxanh                  ;goi chuong trinh con chay chu mau xanh
                lcall   htcam                   ;goi chuong trinh con chay chu mau cam
                sjmp    main
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;hien thi mau do
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
htdo:           mov     dptr,#machuoi
maina:          mov     solanhthi,#100          ;so lan lap lai
mainb:          mov     bdem8byte,#00
                mov     maquet,#00000001b       ;ma quet

mainc:          mov     a,bdem8byte
                movc    a,@a+dptr
                mov     hang,a

                mov     cotx,maquet
                lcall   delay
                mov     cotx,#00h

                inc     bdem8byte               ;tang len de lay byte ke
                mov     a,maquet                ;dich chuyen ma quet
                rl      a
                mov     maquet,a

                cjne    bdem8byte,#8,mainc
                djnz    solanhthi,mainb

                inc     dptr
                mov     a,dpl
                cjne    a,#32,maina
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;hien thi mau xanh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
htxanh:         mov     dptr,#machuoi
```

```
maini:      mov   solanhthi,#100          ;so lan lap lai
mainh:      mov   bdem8byte,#00
            mov   maquet,#00000001b       ;ma quet

maing:      mov   a,bdem8byte
            movc  a,@a+dptr
            mov   hang,a

            mov   cotd,r3
            lcall delay
            mov   cotd,#00h

            inc   bdem8byte               ;tang len de lay byte ke
            mov   a,maquet                ;dich chuyen ma quet
            rl    a
            mov   maquet,a

            cjne  bdem8byte,#8,maing
            djnz  solanhthi,mainh

            inc   dptr
            mov   a,dpl
            cjne  a,#32,maini
            ret

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;hien thi mau cam
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
htcam:      mov   dptr,#machuoi

mainx:      mov   solanhthi,#250
mainy:      mov   bdem8byte,#00
            mov   maquet,#00000001b       ;ma quet

mainw:      mov   a,bdem8byte
            movc  a,@a+dptr
            mov   hang,a

            mov   cotx,r3
            mov   cotd,r3

            lcall delay
            mov   cotx,#00h
            mov   cotd,#00h

            inc   bdem8byte               ;tang len de lay byte ke
            mov   a,maquet                ;dich chuyen ma quet
            rl    a
            mov   maquet,a

            cjne  bdem8byte,#8,mainw
            djnz  solanhthi,mainy

            inc   dptr
            mov   a,dpl
            cjne  a,#32,mainx
            ret

delay:      mov   r5,#1
de:         mov   r6,#50
            djnz  r6,$
            djnz  r5,de
```

```
                ret

                org     0500h
machuoi:        db      0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh          ;8 byte ff de tat 8 led
                db      0B3H,06DH,06DH,06DH,09BH,0ffh       ;ma chu S
                db      001H,0EDH,0EDH,0EDH,0F3H,0ffh       ;ma chu P
                db      001H,0EFH,0D7H,0BBH,07DH,0ffh       ;ma chu K
                db      0FDH,0FDH,001H,0FDH,0FDH,0ffh       ;ma chu T
                db      0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh      ;8 byte ff de tat 8 led
                end
```

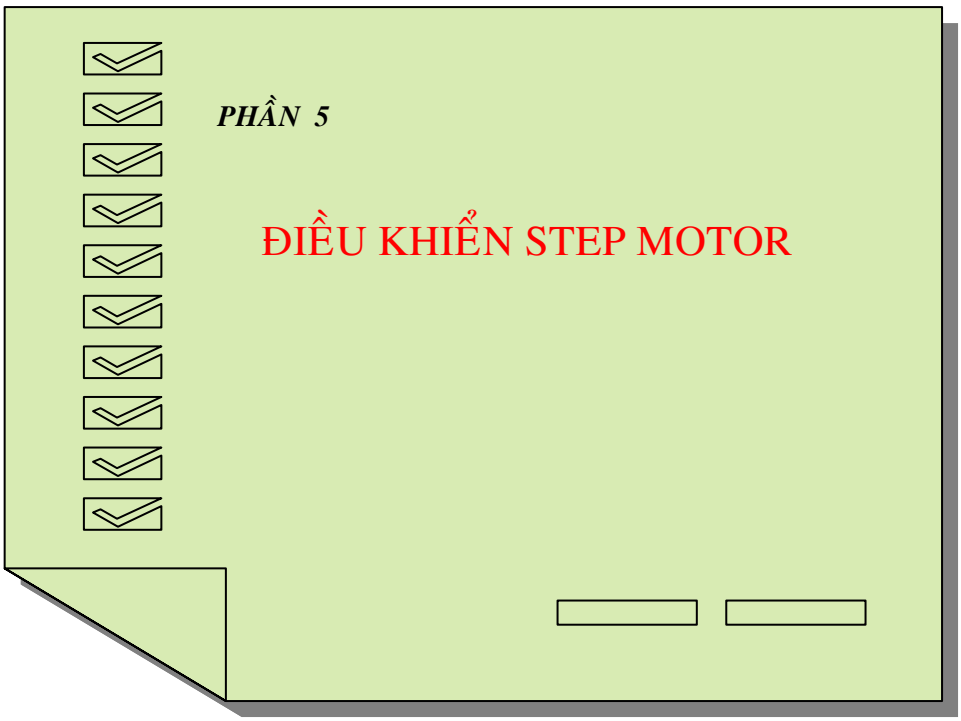
Chương trình điều khiển led ma trận hiển thị chuỗi SPKT với phần trên led xanh, phần dưới led đỏ:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh chay chuoi SPKT tu trai sang phai 4 hang tren mau xanh 4 hang duoc
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

                lap     equ     100

                solanhthi       equ     r2      ;luu so lan hien hti
                maquet          equ     r3
                bdem8byte       equ     r1      ;dem dem so byte cua 1 ki tu

                hang    equ     p3              ;hang ket noi voi port 3
                cotx    equ     p0              ;cot xanh ket noi voi port 0
                cotd    equ     p2              ;cot do ket noi voi port 2

                org     0000h
                mov     cotx,#0                 ;tat quet neu co ket noi
                mov     cotd,#0                 ;tat quet neu co ket noi

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chay mau xanh do : 4 hang tren xanh - 4 hang duoi do
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

chpn4:          mov     dptr,#machuoi           ;
chpn3:          mov      solanhthi,#lap         ;thuc hien lan quet

chpn2:          mov     bdem8byte,#00           ;copy dia chi dau de quan li data
                mov     maquet,#1               ;ma quet cot

chpn1:          mov     a,bdem8byte             ;chuyen dia chi tuong toi cho A
                movc    a,@a+dptr               ;lay data dua vao A

                push    acc                     ;cat tam du lieu vao ngan xep
                orl     a,#0fh                  ;set 4 bit thap len thanh muc 1 tat led
                mov     hang,a

                mov     cotd,maquet
                lcall   delay                   ;goi chuong trinh con delay
                mov     cotd,#00


                pop     acc                     ;lay lai data da cat
                orl     a,#0f0h                 ;set 4 bit cao len thanh muc 1 de tat led
                mov     hang,a

                mov     cotx,maquet
                lcall   delay                   ;goi chuong trinh con delay
```

```
                mov     cotx,#00

                mov     a,maquet                ;lay data tu port 2
                rl      a                       ;xoay a sang phai de quet cot ke
                mov     maquet,a

                inc     bdem8byte               ;chuan bi lay byte data ke
                cjne    bdem8byte,#8,chpn1      ;chuan bi sang cot ke

                djnz    solanhthi,chpn2         ;thuc hien chu ky tiep theo

                inc     dptr
                mov     a,dpl
                cjne    a,#32,chpn3
                ljmp    chpn4                   ;chua het thi quay lai


;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay co kiem tra nhan du lieu
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     75h,#1
del2:           mov     76h,#50
del1:           djnz    76h,del1
                djnz    75h,del2
                ret

                org     0500h
machuoi:        db      0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh     ;8 byte ff de tat 8 led
                db      0B3H,06DH,06DH,06DH,09BH,0ffh               ;ma chu S
                db      001H,0EDH,0EDH,0EDH,0F3H,0ffh               ;ma chu P
                db      001H,0EFH,0D7H,0BBH,07DH,0ffh               ;ma chu K
                db      0FDH,0FDH,001H,0FDH,0FDH,0ffh               ;ma chu T
                db      0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh     ;8 byte ff de tat 8 led

end
```

V. Bài tập:

1. Hãy điều chỉnh chương trình đã viết để chuỗi chạy theo chiều từ trái sang phải.
2. Hãy viết chương trình hiển thị chuỗi “SPKT-DT”.
3. Hãy viết chương trình hiển thị chuỗi dài ví du: “MICROCONTROLLER 89CXX”.
4. Hãy tổ hợp các bài đã viết và bài tập để viết chương trình chuỗi chạy từ phải qua trái và từ trái qua phải.
5. Hãy viết chương trình điều khiển led ma trận có giao tiếp với bàn phím và khi chạy chương trình thì led hiển thị dấu gạch ngang, khi nhấn kí tự nào thì mã của kí tự đó xuất hiện trên led ma trận.
6. Hãy dùng bàn phímma trận để nhập chuổi cần hiển thị.



PHẦN 5

ĐIỀU KHIỂN STEP MOTOR

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 5-1** CHƯƠNG TRÌNH ĐIỀU KHIỂN ĐỘNG CƠ BƯỚC QUAY LIÊN TỤC	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách điều khiển động cơ bước. Động cơ bước có nhiều loại với các thông số về điện áp, dòng điện và số bước của 1 vòng. Trong tài liệu này trình bày các chương trình điều khiển động cơ bước với áp 12V và số bước của 1 vòng là 200 và loại thứ 2 là 96 bước.

II. Trình tự thực hiện:

1. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 1 đến pinhd ngõ vào của IC 2803 và dùng bus dây 5 sợi kết nối giữa động cơ và một trong 2 pinheader 5 chân ở ngõ ra của 2803.
2. Khởi động phần mềm, viết chương trình như sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien step motor quay lien tuc theo 1 chieu, thoi gian delay dai thi dong co quay cham
;thoi gain nho thi dong co quay nhanh, thoi gian qua nho thi dong co khong dap ung duoc se dung yen
;dung 1 port de dieu khien motor qua ic giao tiep 2803 - dung 4 bit thap hoac 4 bit cao
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

                outstep         equ     p1

                org     0000h

main:           mov     outstep,#10001000b
                lcall   delay

                mov     outstep,#01000100b
                lcall   delay

                mov     outstep,#00100010b
                lcall   delay

                mov     outstep,#00010001b
                lcall   delay

                sjmp    main

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

delay:          mov     r6,#80h
del:            mov     r7,#0
                djnz    r7,$
                djnz    r6,del
                ret
            end
```

3. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch chạy đúng yêu cầu.
4. Nếu động cơ không chạy đúng thì hãy quay đầu bus dây 5 sợi.

III. Các chương trình mẫu:

Chương trình điều khiển động cơ bước chạy ngược lại:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien step motor quay lien tuc theo  chieu nguoc lai
;dung 1 port de dieu khien motor qua ic giao tiep 2803 - dung 4 bit thap hoac 4 bit cao
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                outstep         equ     p1
                org     0000h

main:           mov     outstep,#00010001b
                lcall   delay

                mov     outstep,#00100010b
                lcall   delay

                mov     outstep,#01000100b
                lcall   delay

                mov     outstep,#10001000b
                lcall   delay
                sjmp    main
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay voi thoi gian ngan hon
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#10h
del:            mov     r7,#0
                djnz    r7,$
                djnz    r6,del
                ret
        end
```

Trong chương trình chạy ngược này bạn chỉ cần đảo chiều dịch chuyển dữ liệu điều khiển.

Nếu chỉ điều khiển một động cơ thì bạn chỉ cần xuất 4 bit cho port1 nhưng do mạch điện giao tiếp với động cơ sử dụng IC 2803 8 bit nên có thể xuất luôn 8 bit và bạn có thể kết nối động cơ bước vào 4 bit thấp hoặc cao đều được. Còn trong thực tế thì bạn có thể dùng 4 bit thì hiệu chỉnh lại chương trình là xong.

IV. Các chú ý:

Nếu động cơ không chạy đúng theo yêu cầu thì thứ tự các đầu dây không đúng bạn có thể đổi đầu dây hoặc thay đổi thứ tự mã điều khiển để động cơ chạy đúng.

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 5-2** CHƯƠNG TRÌNH ĐIỀU KHIỂN ĐỘNG CƠ BƯỚC QUAY 1 VÒNG RỒI NGỪNG.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách điều khiển động cơ bước quay đúng 1 vòng.

II. Trình tự thực hiện:

1. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 1 đến pinhd ngõ vào của IC 2803 và dùng bus dây 5 sợi kết nối giữa động cơ và một trong 2 pinheader 5 chân ở ngõ ra của 2803.
2. Khởi động phần mềm, viết chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien step motor quay 1 vong roi ngung luon
;dung 1 port de dieu khien motor qua ic giao tiep 2803
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                outstep     equ     p1
                sobuoc      equ     50      ;loai dco: 50x4=200 buoc

                org     0000h

                mov     dptr,#datastep      ;nap dia chi quan ly ma
                mov     r0,#0
                mov     r2,#sobuoc          ;50 chu ky la vong
main2:          mov     r1,#4               ;1 chu ky 8 buoc
main1:          mov     a,r0
                movc    a,@a+dptr
                mov     outstep,a
                lcall   delay

                inc     r0
                anl     00h,#03h            ;anl r0 voi 00000011b
                djnz    r1,main1

                djnz    r2,main2
                sjmp    $                   ;dung lai
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#10h
del:            mov     r7,#0
                djnz    r7,$
                djnz    r6,del
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai du lieu dieu khien dong co buoc
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
datastep:       db      10001000b
                db      01000100b
```

```
                db      00100010b
                db      00010001b
end
```

3. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch chạy đúng yêu cầu.

III. Bài tập:

Hãy viết chương trình điều khiển động cơ quay:

a. Hai vòng rồi ngừng.
b. Năm vòng rồi ngừng.
c. 10 vòng rồi ngừng.
d. Quay thuận 1 vòng rồi quay nghịch 1 vòng.

Chương trình quan thuận 1 vòng rồi quay nghịch 1 vòng rồi dừng lại.

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien step motor quay thuan 1 vong va quay nguoc 1 vong roi ngung
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                outstep         equ     p1
                madkdc          equ     r0

                org     0000h
                mov     madkdc,#00010001b       ;nap ma dieu khien dc buoc
                lcall   quaythuan
                lcall   quaynghich
                sjmp    $
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con dieu khien dong co quay thuan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
quaythuan:      mov     r2,#50                  ;50 chu ky la vong
main2:          mov     r1,#4                   ;1 chu ky 8 buoc
main1:          mov     outstep,madkdc          ;xuat ma dk dong co
                lcall   delay

                mov     a,madkdc
                rl      a
                mov     madkdc,a
                djnz    r1,main1
                djnz    r2,main2
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con dieu khien dong co quay  nguoc
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
quaynghich:     mov     r2,#50                  ;50 chu ky la vong
nmain2:         mov     r1,#4                   ;1 chu ky 8 buoc
nmain1:         mov     outstep,madkdc
                lcall   delay

                mov     a,madkdc
                rr      a
                mov     madkdc,a
                djnz    r1,nmain1
                djnz    r2,nmain2
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#10h
del:            mov     r7,#0
                djnz    r7,$
                djnz    r6,del
                ret
```

end

Chú ý nếu muốn chạy lại để xem thì chỉ cần nhấn nút reset trên bộ thí nghiệm.

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 5-3** CHƯƠNG TRÌNH ĐIỀU KHIỂN ĐỘNG CƠ BƯỚC QUAY BẰNG 2 NÚT NHẤN START VÀ STOP	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách điều khiển động cơ bước và giao tiếp với bàn phím và led 7 đoạn.

II. Trình tự thực hiện:

1. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 1 đến pinhd ngõ vào của IC 2803 và dùng bus dây 5 sợi kết nối giữa động cơ và một trong 2 pinheader 5 chân ở ngõ ra của 2803.
   - Dùng bus dây kết nối port 3 đến pinhd bàn phím ma trận từ K0 đến K7.
2. Khởi động phần mềm, viết chương trình như sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien step motor: khi nhan nut start thi motor quay
;khi nhan nut stop thi motor ngung
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

                cot_c0          bit     p3.0
                start           bit     p3.4            ;phim so 0
                stop            bit     p3.5            ;phim so 1
                madkdc          equ     r0
                outstep         equ     p1

                org     0000h
                clr     cot_c0                          ;cho cot thu 0 cua ma tran ban phim = 0
                mov     madkdc,#00010001b               ;nap ma dieu khien dc buoc

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien dong co quay thuan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
main:           jb      start,$                         ;cho nhan satrt
                jnb     start,$                         ;cho cho buong nut nhan

main1:          mov     outstep,madkdc          ;xuat ma dk dong co
                lcall   delay
                mov     a,madkdc
                rl      a
                mov     madkdc,a
                jc      main
                sjmp    main1

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con delay co kiem tra stop
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#10h
de:             mov     r7,#0
                jnb     stop,dend
                djnz    r7,$
                djnz    r6,de
```

```
                clr     c                       ;c=0 la khong nhan Stop
                ret

dend:           jb      stop,$
                setb    c                       ;c=1 la co nhan Stop
                ret
end
```

Chú ý nếu muốn động cơ chạy thì nhấn phím số "0" và muốn ngừng thì nhấn phím số "1". Chương trình có thể viết gọn hơn nữa nếu dùng thanh ghi A lưu trữ madkdc.

3. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch chạy đúng yêu cầu.

Tại sao làm cột thứ 0 của ma trận bàn phím xuống mức 0.

III. Các chương trình mẫu:

Chương trình điều khiển động cơ bước quay có các phím điều khiển start, stop, đảo chiều.

- Dùng bus dây kết nối port 1 đến pinhd ngõ vào của IC 2803 và dùng bus dây 5 sợi kết nối giữa động cơ và một trong 2 pinheader 5 chân ở ngõ ra của 2803.
- Dùng bus dây kết nối port 3 đến pinhd bàn phím ma trận từ K0 đến K7.
- Dùng bus dây kết nối port 0 đến pinhd của 8 led đơn xanh vàng đỏ để biết trạng thái đèn sáng tương ứng với động cơ.

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien step motor: khi nhan nut start thi motor quay
;khi nhan nut stop thi motor ngung, nut dao chieu dong co
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

                cot_c0          bit     p3.0
                start           bit     p3.4            ;phim so 0
                stop            bit     p3.5            ;phim so 1
                inv             bit     p3.6            ;phim so 2

                bit_inv         bit     p0.0            ;bit trang thai bao dao chieu dong co
                bit_startstop   bit     p0.1            ;bit trang thai bao strat stop

                madkdc  equ     r0
                outstep         equ     p1

                org     0000h
                clr     cot_c0                          ;cho cot thu 0 cua ma tran ban phim = 0
                clr     bit_inv                         ;bit_inv = 0 thi chay thuan, bit_inv = 1 thi chay
nguoc
                clr     bit_startstop
                mov     madkdc,#00010001b               ;nap ma dieu khien dc buoc

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien dong co quay thuan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
run_for:        jb      start,$                         ;cho nhan satrt
                jnb     start,$                         ;cho cho buong nut nhan
                jb      bit_inv,run_inv                 ;nhay den chtr chay nguoc

run_for1:       mov     outstep,madkdc          ;xuat ma dk dong co
                lcall   delay
                mov     a,madkdc
                rl      a                               ;;xoay thuan
                mov     madkdc,a
                jb      bit_startstop,run_for
```

```
                jb      bit_inv,run_inv                 ;nhay den chtr chay nguoc
                sjmp    run_for1

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien dong co quay thuan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
run_inv:        mov     outstep,madkdc          ;xuat ma dk dong co
                lcall   delay
                mov     a,madkdc
                rr      a                       ;xoay nguoc
                mov     madkdc,a
                jb      bit_startstop,run_for
                jnb     bit_inv,run_for1        ;nhay den chtr chay nguoc
                sjmp    run_inv
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con delay co kiem tra stop
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#10h
de:             mov     r7,#0
                jnb     stop,dend
                jnb     inv,dend1
                djnz    r7,$
                djnz    r6,de
                clr     bit_startstop           ;c=0 la khong nhan Stop
                ret

dend:           jb      stop,$
                setb    bit_startstop           ;c=1 la co nhan Stop
                ret

dend1:          jb      inv,$
                cpl     bit_inv
                lcall   delay1s
                jnb     inv,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con delay de chong doi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1s:        mov     7fh,#10
xde:            mov     7eh,#0
                djnz    7eh,$
                djnz    7fh,xde
                ret
end
```

Chương trình điều khiển động cơ bước quay có các phím điều khiển start, stop, đảo chiều.

- Dùng bus dây kết nối port 1 đến pinhd ngõ vào của IC 2803 và dùng bus dây 5 sợi kết nối giữa động cơ và một trong 2 pinheader 5 chân ở ngõ ra của 2803.
- Dùng bus dây kết nối port 3 đến pinhd bàn phím ma trận từ K0 đến K7.
- Dùng bus dây kết nối port 0 đến pinhd điều khiển các đoạn a, b, c, d, e, f, g, dp.
- Dùng bus dây kết nối port 2 đến pinhd điều khiển quét hàng từ T0 đến T7.

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien step motor: khi nhan nut start thi motor quay
;khi nhan nut stop thi motor ngung, nut dao chieu dong co
;co them phan hien thi so buoc - so vong tren led 7 doan
```

```
;co hien thi cac trang thai OFF, UP, Down tren led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

cot_c0          bit     p3.0
start           bit     p3.4            ;phim so 0
stop            bit     p3.5            ;phim so 1
inv             bit     p3.6            ;phim so 2

bit_inv         bit     40h             ;bit trang thai bao dao chieu dong co
bit_startstop   bit     41h             ;bit trang thai bao strat stop

madkdc equ      r0
outstep         equ     p1

quetled7d       equ     p2
cacdoan         equ     p0

demsobuoc       equ     r3              ;luu so buoc
demsovong       equ     r4              ;luu so vong
sobuoc          equ     200             ;1 vong bang 200 buoc
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                org     0000h
                ljmp    main

                org     000bh
                ljmp    ngat_timer0             ;ngat timer0
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

main:           mov     tmod,#00000001b         ;T0 mod 1 dem 16 bit
                mov     th0,#high(-500)
                mov     tl0,#low(-500)
                clr     tf0
                setb    tr0
                mov     ie,#10000010b           ;cho phep timer0 ngat

                mov     demsovong,#0
                mov     demsobuoc,#0

                clr     cot_c0                  ;cho cot thu 0 cua ma tran ban phim = 0
                clr     bit_inv                 ;bit_inv = 0 thi chay thuan, bit_inv = 1 thi chay
nguoc
                clr     bit_startstop
                mov     madkdc,#00010001b       ;nap ma dieu khien dc buoc
                mov     dptr,#ma7doan

                lcall   giaima_buoc
                lcall   giaima_vong
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien dong co quay thuan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
run_for:        mov     20h,#0c0h               ;ma chu O
                mov     21h,#8eh                ;ma chu F

                jb      start,$                 ;cho nhan satrt
                jnb     start,$                 ;cho cho buong nut nhan
                jb      bit_inv,run_inv         ;nhay den chtr chay nguoc

                mov     20h,#0c1h               ;ma chu U
                mov     21h,#8ch                ;ma chu P

run_for1:       mov     outstep,madkdc      ;xuat ma dk dong co
                lcall   delay
```

```
                mov     a,madkdc
                rl      a                       ;xoay thuan
                mov     madkdc,a

                inc     demsobuoc
                cjne    demsobuoc,#sobuoc,run_for2
                mov     demsobuoc,#0            ;het 200 buoc
                inc     demsovong
                lcall   giaima_vong

run_for2:       lcall   giaima_buoc
                jb      bit_startstop,run_for
                jb      bit_inv,run_inv         ;nhay den chtr chay nguoc
                sjmp    run_for1

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr dieu khien dong co quay thuan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
run_inv:        mov     20h,#0a1h               ;ma chu d
                mov     21h,#0a3h               ;ma chu o


run_inv1:       mov     outstep,madkdc          ;xuat ma dk dong co
                lcall   delay
                mov     a,madkdc
                rr      a                       ;xoay nguoc
                mov     madkdc,a

                inc     demsobuoc
                cjne    demsobuoc,#sobuoc,run_inv2
                mov     demsobuoc,#0            ;het 200 buoc
                inc     demsovong
                lcall   giaima_vong

run_inv2:       lcall   giaima_buoc
                jb      bit_startstop,run_for
                jnb     bit_inv,run_for1        ;nhay den chtr chay nguoc
                sjmp    run_inv1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con delay co kiem tra stop
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#10h
de:             mov     r7,#0
                jnb     stop,dend
                jnb     inv,dend1
                djnz    r7,$
                djnz    r6,de
                clr     bit_startstop           ;c=0 la khong nhan Stop
                ret

dend:           jb      stop,$
                setb    bit_startstop           ;c=1 la co nhan Stop
                ret

dend1:          jb      inv,$
                cpl     bit_inv
                lcall   delay1s
                jnb     inv,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con delay de chong doi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
delay1s:        mov     7fh,#10
xde:            mov     7eh,#0
                djnz    7eh,$
                djnz    7fh,xde
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; chuong trinh con giai ma
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
giaima_buoc:    mov     a,demsobuoc
                mov     b,#10
                div     ab
                mov     37h,b
                mov     b,#10
                div     ab
                mov     36h,b
                mov     35h,a

                mov     a,37h
                movc    a,@a+dptr
                mov     27h,a

                mov     a,36h
                movc    a,@a+dptr
                mov     26h,a

                mov     a,35h
                movc    a,@a+dptr
                mov     25h,a
                ret

giaima_vong:    mov     a,demsovong
                mov     b,#10
                div     ab
                mov     34h,b
                mov     b,#10
                div     ab
                mov     33h,b
                mov     32h,a

                mov     a,32h
                cjne    a,#0,gmv_1
                mov     22h,#0ffh
                sjmp    gmv_2

gmv_1:          movc    a,@a+dptr
                mov     22h,a

gmv_2:          mov     a,33h
                cjne    a,#0,gmv_3
                mov     23h,#0ffh
                sjmp    gmv_4

gmv_3:          movc    a,@a+dptr
                mov     23h,a

gmv_4:          mov     a,34h
                movc    a,@a+dptr
                anl     a,#7fh
                mov     24h,a
                ret

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
;Chuong trinh con ngat cua timer0 de hien thi lien tuc
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ngat_timer0:    lcall   hthi
                mov     th0,#high(-500)
                mov     tl0,#low(-500)
                reti
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           push    acc             ;cat tam thanh ghi a
                mov     a,#01111111b    ;ma quet
                mov     r1,#27h

ht1:            mov     cacdoan,@r1
                mov     quetled7d,a
                lcall   delay1
                mov     quetled7d,#0ffh
                dec     r1
                rr      a               ;chuyen sang led ke
                cjne    r1,#1fh,ht1
                pop     acc             ;lay lai noi dung thanh ghi a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay1
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay1:         mov     7ch,#50h
                djnz    7ch,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao ma 7 doan tu so '0' den so '9'
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:        db      0C0h,0F9h,0A4h,0B0h,99h,92h,82h,0F8h,80h,90h

end
```

7. Bài tập ứng dụng:

1. Hãy viết chương trình điều khiển động cơ quay có điều khiển start và stop nhưng chỉ dùng 1 nút nhấn.
2. Hãy viết lưu đồ điều khiển của các chương trình trên.

Hãy mở rộng bài điều khiển trên với các yêu cầu điều khiển như sau như sau:

Phím “F” làm phím stop có hiển thị chữ OF

Phím “E” làm phím start có hiển thị chữ UP hoặc chữ Do

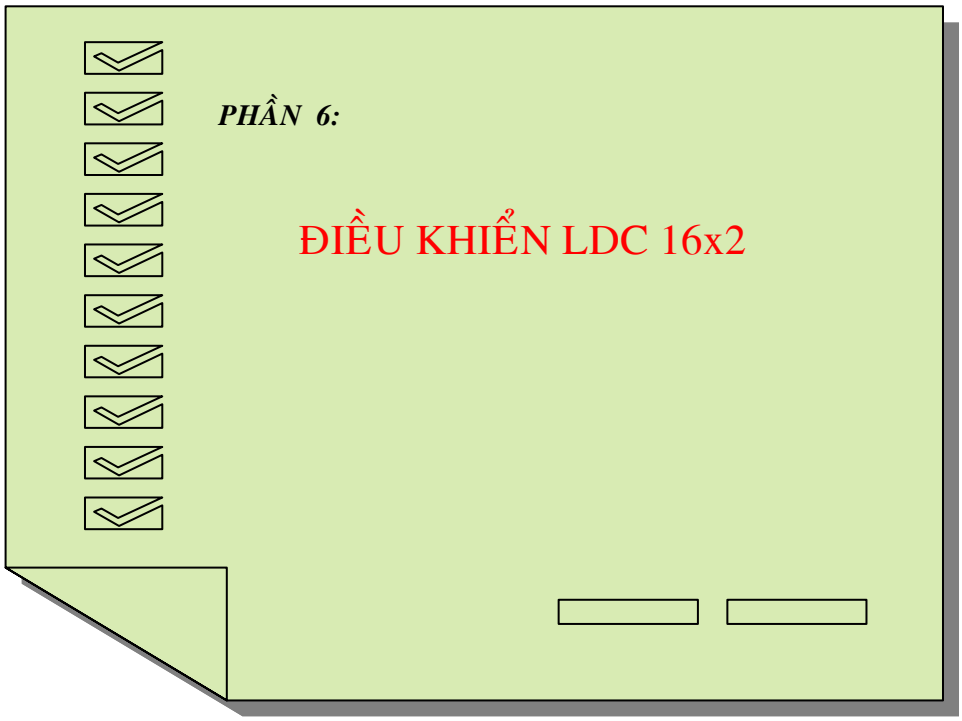
Phím “D” làm phím đảo chiều.

Phím “A” làm phím tăng tốc, phím “B” làm phím giảm tốc.

Chú ý: phải sử dụng bàn phím ma trận và chương trình con quét 16 phím.

Còn nhiều yêu cầu nhưng theo tác gải nếu bạn hiểu và thực hiện được các yêu cầu của sách hướng dẫn này thì việc mở rộng các yêu cầu bạn có thể giải quyết được.

“ Không có việc gì khó chỉ sợ lòng không bền”



PHẦN 6:

ĐIỀU KHIỂN LDC 16x2

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 6-1** CHƯƠNG TRÌNH ĐIỀU KHIỂN LCD SÁNG CHUỖI KÝ TỰ LƯU TRONG VÙNG NHỚ CHƯƠNG TRÌNH.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách giao tiếp điều khiển LCD.

II. Trình tự thực hiện:

1. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 3 đến pinhd bàn phím ma trận từ K0 đến K7.
   - Dùng bus dây kết nối port 0 đến pinhd điều khiển LCD từ D7 đến D0.
   - Dùng bus dây (3 hoặc 8) kết nối 3 chân p2.0 p2.1 p2.2 đến 3 chân tương ứng E, R/W và RS.
2. Khởi động phần mềm, viết chương trình như sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;HIEN THI NOI DUNG CHO O BEN DUOI
;DUNG PORT 0 KET DOI VOI CAC DUONG DU LIEU CUA LCD P0-7 ->D0-7
;DUNG 3 BIT CUA PORT2: P20,P21,P22 DIEU KHIEN E,R/W,RS
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        e               BIT     P2.0
                        rw              BIT     P2.1
                        rs              BIT     P2.2

                        byteout         equ     p0

                        ORG     0000H

                        lcall   khtaolcd                ;khoi tao lcd
                        lcall   first_line              ;goi chtr con hien thi hang thu nhat
                        lcall   scond_line              ;goi chtr con hien thi hang thu hai
                        sjmp    $
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con hien thi noi dung hang thu 1 tren LCD
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
first_line:             mov     A,#080h                 ;set DDRAM
                        lcall   ktao
                        mov     dptr,#fline_data
                        lcall   Write
                        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con hien thi noi dung hang thu 2 tren LCD
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
scond_line:             mov     a,#0c0h                 ;set DDRAM
                        lcall   ktao
                        mov     dptr,#sline_data
                        lcall   write
                        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con goi data hien thi ra LCD
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
write:                  mov     A,#0
```

```
                    movc    A,@a+dptr
                    cjne    A,#99h,Writea
                    ret

Writea:             mov     byteout,a
                    acall   data_byte
                    inc     dptr
                    sjmp    Write
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con khoi tao LCD
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ktao:               mov     byteout,a
                    lcall   command_byte
                    ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Feed command/data to the LCD module
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
command_byte:       clr     rs              ;RS low for a command byte
                    sjmp    data_bytea

data_byte:          setb    rs              ;RS high for a data byte
data_bytea:         clr     rw              ;R/W low for a write mode
                    clr     e
                    nop

                    setb    e               ;Enable pulse
                    nop
                    nop
                    mov     byteout,#0ffh   ;configure port1 to input mode

                    setb    rw              ;set RW to read
                    clr     rs              ;set RS to command
                    clr     e               ;generate enable pulse

                    nop
                    nop
                    setb    e
                    lcall   delay100
                    ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con khoi tao LCD
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
khtaolcd:           setb    e               ;Enable
                    clr     rs              ;RS low
                    clr     rw              ;RW low

                    mov     a,#38h          ;tu dieu khien LCD
                    lcall   ktao
                    lcall   delay41         ;delay 4.1 mSec

                    mov     A,#38h          ;function set
                    lcall   ktao
                    lcall   delay100        ;delay

                    mov     A,#38h          ;function
                    lcall   ktao

                    mov     A,#0ch          ;tu dieu khien display on
                    lcall   ktao
                    mov     A,#01h          ;tu dieu khien Clear display
                    lcall   ktao
```

```
                        mov     A,#06h              ;tu dieu khien entry mode set
                        lcall   ktao

                        mov     A,#80h              ;thiet lap dia chi LCD (set DD RAM)
                        lcall   ktao
                        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay 4.1 ms
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay41:                mov     7eh,#90h
de_a:                   mov     7fh,#200
                        djnz    7fh,$
                        djnz    7eh,de_a
                        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay 255 microgiay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay100:               mov     7fh,#00
                        djnz    7fh,$
                        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;       Data bytes
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
          fline_data:   db      'NGUYEN DINH PHU ',099h
          sline_data:   db      'DAI HOC SPKT HCM',099h
end
```

3. Biên dịch và nạp chương trình và xem kết kết quả.

Chú ý: Nếu LCD chưa có nguồn thì hãy chuyển SW nguồn của LCD nằm phía trên LCD sang vị trí ON. Các LCD đã chỉnh biến trở ở vị trí thích hợp cho việc hiển thị, khi cấp nguồn cho LCD và nếu chưa có chương trình điều khiển thì hàng trên của LCD sẽ sáng màu đen, còn hàng dưới thì sáng giống màu nền.

III. Bài tập ứng dụng:

1. Hãy thay đổi dữ liệu ở hàng fline và sline để hiển thị các kí tự theo yêu cầu.
2. Sau khi thực hiện bài thực hành trên bạn hãy cho biết ý kiến của bạn.

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 6-2** CHƯƠNG TRÌNH ĐIỀU KHIỂN LCD SÁNG CHUỖI KÝ TỰ LƯU TRONG VÙNG NHỚ RAM NỘI	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách giao tiếp điều khiển LCD và dùng vùng nhớ RAM nội để lưu dữ liệu hiển thị.

II. Trình tự thực hiện:

1. Kết nối mạch theo trình tự :
   - Dùng bus dây kết nối port 3 đến pinhd bàn phím ma trận từ K0 đến K7.
   - Dùng bus dây kết nối port 0 đến pinhd điều khiển LCD từ D7 đến D0.
   - Dùng bus dây (3 hoặc 8) kết nối 3 chân p2.0 p2.1 p2.2 đến 3 chân tương ứng E, R/W và RS.
2. Khởi động phần mềm, viết chương trình như sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien lcd su dung
;vung nho 30h den 3fh den luu tru thong tin de hien thi len hang thu nhat cua lcd
;vung nho 40h den 4fh den luu tru thong tin de hien thi len hang thu hai cua lcd
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;dinh nghia cac bien
                E               bit     p2.0
                rw              bit     p2.1
                rs              bit     p2.2
                byteout         equ     p0

;cac vung nho se su dung de luu thong tin
;               30h ->  3fh de luu thong tin cho LCD hang thu 1
;               40h ->  4fh de luu thong tin cho LCD hang thu 2
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh chinh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        org     0000h
                        mov     sp,#50h
                        mov     070h,#0
                        lcall   khtaolcd                ;goi chtr khoi tao LCD

                        lcall   napdatahienthi
                        lcall   hienthichung            ;goi chtr con hien thi thong tin ra LCD
                        sjmp    $

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;nap du lieu can hien thi vao bo nho ram luu tu 30h den 4FH - 32 byte
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
napdatahienthi:         mov     30h,#'N'
                        mov     31h,#'G'
                        mov     32h,#'U'
                        mov     33h,#'Y'
                        mov     34h,#'E'
                        mov     35h,#'D'
                        mov     36h,#' '
                        mov     37h,#'D'
                        mov     38h,#'I'
```

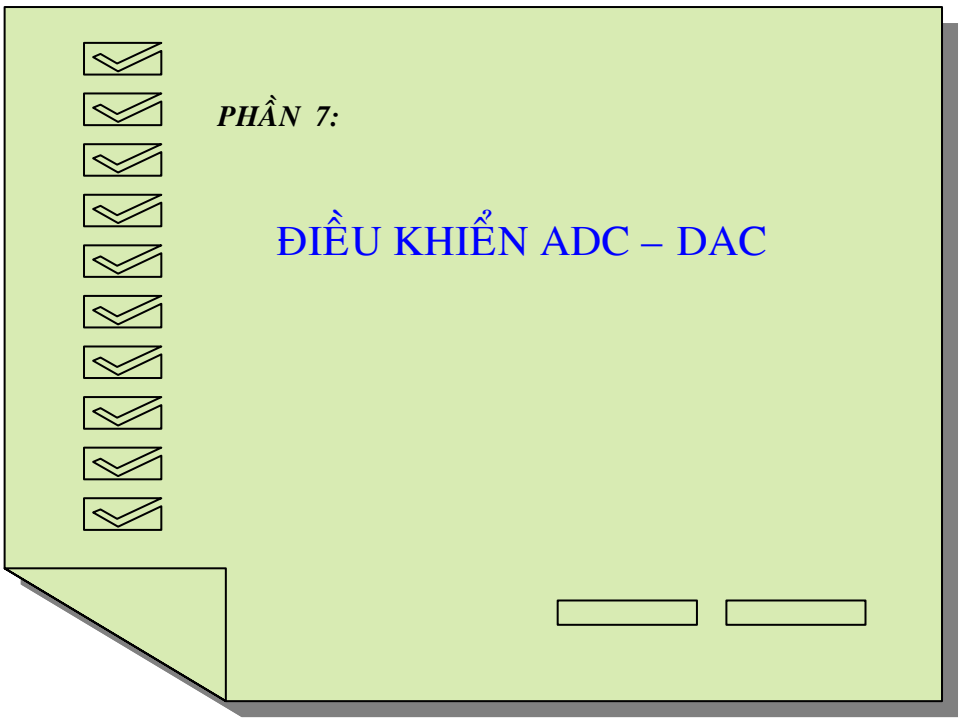
```
                        mov     39h,#'N'
                        mov     3ah,#'H'
                        mov     3bh,#' '
                        mov     3ch,#'P'
                        mov     3dh,#'H'
                        mov     3eH,#'U'
                        mov     3fh,#' '

                        mov     40h,#'D'
                        mov     41h,#'T'
                        mov     42h,#'D'
                        mov     43h,#'D'
                        mov     44h,#' '
                        mov     45h,#'0'
                        mov     46h,#'9'
                        mov     47h,#'0'
                        mov     48h,#'3'
                        mov     49h,#'9'
                        mov     4ah,#'8'
                        mov     4bh,#'2'
                        mov     4ch,#'4'
                        mov     4dh,#'4'
                        mov     4eH,#'3'
                        mov     4fh,#' '
                        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh con hien thi noi dung  tren LCD cua2 vung nho
;30H->3Fh hang 1; 40H-> 4Fh hang 2;
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hienthichung:           mov     A,#080h                 ;set DDRAM
                        lcall   ktao
                        mov     r1,#16
                        mov     R0,#30H
fline:                  lcall   Write
                        inc      r0
                        djnz    r1,fline

                        mov     a,#0c0h                 ;set DDRAM
                        lcall   ktao
                        mov     r1,#16
                        mov     r0,#40H
sline:                  lcall   Write
                        inc     r0
                        djnz    r1,sline
                        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con goi data hien thi ra LCD
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
write:                  mov     byteout,@R0
                        lcall   data_byte
                        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con khoi tao LCD
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ktao:                   mov     byteout,a
                        lcall   command_byte
                        ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Feed command/data to the LCD module
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
command_byte: clr       rs                      ;RS low for a command byte
                        sjmp    data_bytea
```

```
data_byte:      setb    rs              ;RS high for a data byte
data_bytea:     clr     rw              ;R/W low for a write mode
                clr     e
                nop
                setb    e               ;Enable pulse
                nop
                nop
                mov     byteout,#0ffh   ;configure port1 to input mode

                setb    rw              ;set RW to read
                clr     rs              ;set RS to command
                clr     e               ;generate enable pulse

                nop
                nop
                setb    e
                lcall   delay100
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con khoi tao LCD
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
khtaolcd:       setb    e               ;Enable
                clr     rs              ;RS low
                clr     rw              ;RW low

                mov     a,#38h          ;tu dieu khien LCD
                lcall   ktao
                lcall   delay41         ;delay 4.1 mSec

                mov     A,#38h          ;function set
                lcall   ktao
                lcall   delay100        ;delay
                mov     A,#38h          ;function
                lcall   ktao

                mov     A,#0ch          ;tu dieu khien display on
                lcall   ktao
                mov     A,#01h          ;tu dieu khien Clear display
                lcall   ktao

                mov     A,#06h          ;tu dieu khien entry mode set
                lcall   ktao
                mov     A,#80h          ;thiet lap dia chi LCD (set DD RAM)
                lcall   ktao
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay 4.1 ms
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay41:        mov     7eh,#90h
de_a:           mov     7fh,#200
                djnz    7fh,$
                djnz    7eh,de_a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay 255 microgiay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay100:       mov     7fh,#00
                djnz    7fh,$
                ret
end
```

3. Biên dịch và nạp chương trình và xem kết kết quả.

III. Bài tập ứng dụng:

1. Hãy thay đổi dữ liệu ở hàng fline và sline để hiển thị các kí tự theo yêu cầu.
2. Hãy viết lại các chương trình điều khiển động cơ bước có các nút điều khiển “start”, “stop”, “đảo chiều” có hiển thị số bước và số vòng trên LCD.
3. Điều khiển động cơ bước chạy với số vòng đặt trước hiển thị trên LCD.



Chú ý: Từ phần 7 trở về sau thì chỉ sử dụng cho bộ thí nghiệm lớn.

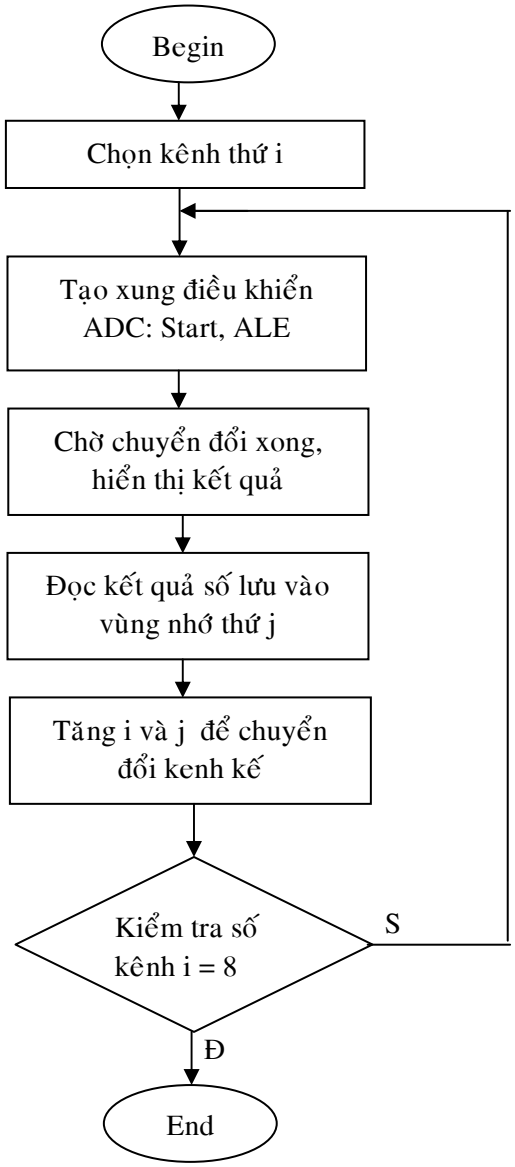
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 7-1** CHƯƠNG TRÌNH ĐIỀU KHIỂN ADC0809 CHUYỂN ĐỔI DỮ LIỆU 8 KÊNH – HIỂN THỊ LED 7 ĐOẠN	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách điều khiển ADC 0809 chuyển đổi tín hiệu tương tự ở ngõ vào sang dữ liệu số hex và hiển thị trên 2 led.

II. Trình tự thực hiện:

1. Giải thuật:



2. Kết nối mạch theo trình tự :
   - Kết nối port1 điều khiển các đoạn a,b,c,d,e,f,g,dp.
   - Kết nối port2 điều khiển các transistor của 8 led 7 đoạn.
   - Kết nối port3 nhận dữ liệu số từ ngõ ra của ADC 0808.
   - Kết nối port0: P0.3 điều khiển ALE và P0.4 điều khiển Start
3. Khởi động phần mềm, biên soạn chương trình theo sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh dieu khien ADC 0809 chuyen doi tin hieu kenh 1 - hien thi so hex
;dung he thong 1: vi dieu khien 8951 ket noi voi khoi ADC0809 va 8 led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

                led7        equ     p1      ;dieu khien cac doan a,b,c,...
                quet        equ     p2      ;dieu khien quet transistor
                inadc       equ     p3      ;nhap du lieu tu adc
                control     equ     p0
                ALE         bit     p0.3
                start       bit     p0.4

                org     0000h
                mov     dptr,#ma7doan
main:           lcall   ctcd_adc            ;goi chtr con chuyen doi du lieu
                lcall   giaimahex           ;goi chtrinh con giai ma so hex sang led 7 doan
                ljmp    main                ;nhay ve chuyen doi tro lai

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con chuyen doi du lieu analog sang so ket qua luu trong A
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ctcd_adc:       mov     control,#00000000B  ;goi ra port 3
                setb    ale                 ;cho ALE=1
                nop                         ;delay 1 it thoi gian
                nop
                setb    start               ;start = 1
                nop
                nop
                clr     ale
                clr     start
                lcall   delayhthi           ;goi chtr con delay co hien thi
                mov     a,inadc             ;doc du lieu sau khi chuyen doi
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtrinh con chuyen so hex sau khi chuyen doi thanh ma 7 doan tuong ung sang
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
giaimahex:      mov     r1,a
                anl     a,#0fh              ;xoa 4 bit thap
                movc    a,@a+dptr           ;lay ma 7 doan tuong ung voi so hex
                mov     27h,a               ;cat vao o nho 27h de hien thi

                mov     a,r1
                anl     a,#0f0h             ;xoa 4 bit cao
                swap    a
                movc    a,@a+dptr
                mov     26h,a               ;cat vao o nho 26h de hien thi
                ret

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con co ghep chtr con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delayhthi:      mov     7eh,#1
pqn:            mov     7fh,#30
```

```
delpqn:     lcall   hthi                    ;goi chuong trinh con hien thi
            djnz    7fh,delpqn
            djnz    7eh,pqn
            ret

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi ket qua dang so hex sau khi giai ma ra led
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:       mov     led7,27h                ;lay du lieu tung kenh
            mov     quet,#01111111b         ;goi ma quet cho 1 led sang
            lcall   delay10
            mov     quet,#0ffh              ; tat het de chong lem

            mov     led7,26h                ;lay du lieu tung kenh
            mov     quet,#10111111b         ;goi ma quet cho 1 led sang
            lcall   delay10
            mov     quet,#0ffh              ; tat het de chong lem
            ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay10:    mov     7ch,#50h
            djnz    7ch,$
            ret

;khai bao ma 7 doan tu so '0' den so '9'
ma7doan:    db      0C0h,0F9h,0A4h,0B0h,099h,092h,082h,0F8h
            db      080h,090h,088h,083h,0c6h,0a1h,086h,08eh
end
```

4. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch chạy đúng yêu cầu.

Khi chạy chương trình này thì 2 led 7 đoạn sẽ hiển thị số hex là giá trị chuyển đổi của kênh tương tự thứ nhất. Bạn dùng 1 cây vít để điều chỉnh giá trị điện áp thay đổi của biến trở trên cùng của khối ADC [có tên là VR0] thì số hiển thị trên led sẽ thay đổi theo dưới dạng số hex. ADC0809 chỉ có tám bit nên giá trị có thể chỉnh được nằm trong vùng giá trị từ 00 đến FF.

Chú ý: điều chỉnh nhẹ nhàng biến trở cho đến khi đạt giá trị ngưỡng 00 hay FF thì dừng lại, đừng dùng quá sức chẳng được gì mà có làm hỏng biến trở thôi.

Nếu chỉnh mà không thấy thay đổi gì thì hãy xem lại chương trình hoặc chuyển sang kênh thứ 2 để xem.

III. Bài tập:

1. Hãy viết thêm vào chương trình điều khiển ADC chuyển đổi dữ liệu 1 kênh có thêm phần hiển thị các kí tự ADC ở các led còn lại.
2. Hãy viết chương trình điều khiển ADC chuyển đổi dữ liệu 2 kênh – hiển thị trên 4led.
3. Hãy viết chương trình điều khiển ADC chuyển đổi dữ liệu 4 kênh – hiển thị trên 8led.

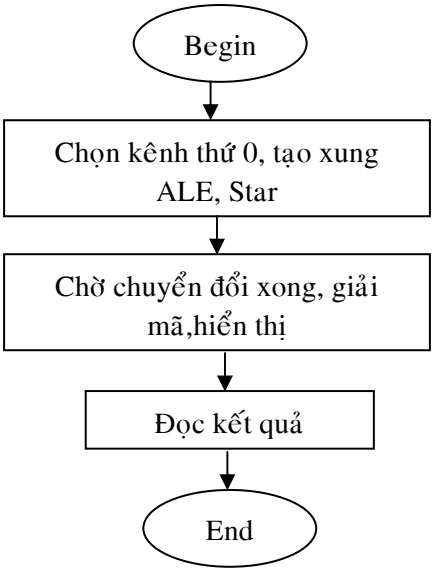
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 7-2** CHƯƠNG TRÌNH ĐIỀU KHIỂN ADC0809 CHUYỂN ĐỔI DỮ LIỆU KÊNH THỨ 0 – HIỂN THỊ KẾT QUẢ BẰNG SỐ THẬP PHÂN	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách viết chương trình điều khiển ADC 0809 chuyển đổi 1 kênh sang dữ liệu số và kết hợp với các chương trình giải mã, hiển thị kết quả bằng số thập phân.

II. Trình tự thực hiện:

1. Giải thuật:



2. Kết nối mạch theo trình tự : giống như bài 7-1.
3. Khởi động phần mềm, biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh dieu khien ADC 0809 chuyen doi tin hieu kenh 1 - hien thi so thap phan
;dung he thong 1: vi dieu khien 8951 ket noi voi khoi ADC0809 va 8 led 7 doan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
led7        equ     p1      ;dieu khien cac doan a,b,c,...
quet        equ     p2      ;dieu khien quet transistor
inadc       equ     p3      ;nhap du lieu tu adc
control     equ     p0
ALE         bit     p0.3
start       bit     p0.4

            org     0000h
            mov     dptr,#ma7doan
            mov     20h,#88h            ;nap ma chu A
            mov     21h,#0a1h           ;nap ma chu d
            mov     22h,#0c6h           ;nap ma chu C
```

```
                mov     23h,#0c0h               ;luu thu tu kenh so 0
                mov     24h,#0ffh
                mov     25h,#0ffh

main:           lcall   ctcd_adc                ;goi chtr con chuyen doi du lieu
                lcall   gma_hex_bcd             ;goi chtrinh con chuyen so hex sang ma 7 doan
                ljmp    main                    ;nhay ve chuyen doi tro lai

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con chuyen doi du lieu analog sang so ket qua luu trong A
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

ctcd_adc:       mov     control,#00000000B      ;goi ra port 3
                setb    ale                     cho ALE=1
                nop                             ;delay 1 it thoi gian
                nop
                setb    start                   ;start = 1
                nop
                nop
                clr     ale
                clr     start
                lcall   delayhthi               ;goi chtr con delay co hien thi
                mov     a,inadc                 ;doc du lieu sau khi chuyen doi
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtrinh con chuyen so hex thanh so BCD va sau do thi chuyen
;ma BCD thanh ma 7 doan de hien thi so thap phan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
gma_hex_bcd:
                mov     b,#10                   ;chuyen so hex sang ma BCD
                div     ab
                mov     10h,b                   ;luu hang don vi BCD

                mov     b,#10
                div     ab                      ;(a) chua so hang tram, (b) chua hang chuc

                movc    a,@a+dptr
                mov     25h,a                   ;cat so hang tram

                mov     a,b
                movc    a,@a+dptr
                mov     26h,a

                mov     a,10h
                movc    a,@a+dptr
                mov     27h,a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delayhthi:      mov     7eh,#1
pqn:            mov     7fh,#30
delpqn:         lcall   hthi                            ;goi chuong trinh con hien thi
                djnz    7fh,delpqn
                djnz    7eh,pqn
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           mov     r0,#27h
                mov     a,#01111111b

hth:            mov     led7,@r0                        ;lay du lieu
                mov     quet,a
                lcall   delay10
                mov     quet,#0ffh              ; tat het de chong lem
```

```
                dec     r0
                setb    c
                rrc     a
                jc      hth
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay10:        mov     7ch,#50h
                djnz    7ch,$
                ret

;khai bao ma 7 doan tu so '0' den so '9'
ma7doan:        db      0C0h,0F9h,0A4h,0B0h,099h,092h,082h,0F8h
                db      080h,090h,088h,083h,0c6h,0a1h,086h,08eh
end
```

4. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch chạy đúng yêu cầu.

Chú ý: nếu muốn làm mạch đo nhiệt độ chỉ cần xử lý phần cảm biến và khuếch đại tín hiệu cho tương thích với độ phân giải của ADC và dùng hệ thống vi điều khiển và dùng chương trình này là có thể thực hiện được quá trình đo. Nếu muốn khống chế điều khiển thì thêm chương trình và ngõ ra điều khiển.

III. Bài tập:

1. Hãy viết chương trình điều khiển ADC chuyển đổi dữ liệu 2 kênh thứ 0 và thứ 1 – hiển thị bằng số thập phân.

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 7-3** CHƯƠNG TRÌNH ĐIỀU KHIỂN ADC0809 CHUYỂN ĐỔI DỮ LIỆU 8 KÊNH– HIỂN THỊ KẾT QUẢ LẦN LƯỢT TỪNG KÊNH BẰNG SỐ THẬP PHÂN	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết được cách viết chương trình điều khiển ADC 0809 chuyển đổi lần lượt từng kênh sang dữ liệu số và kết hợp với các chương trình giải mã, hiển thị kết quả bằng số thập phân.

II. Trình tự thực hiện:

1. Kết nối mạch theo trình tự :
   - Kết nối mạch giống bài 7-1 .
2. Khởi động phần mềm, viết chương trình như sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh dieu khien ADC 0809 chuyen doi tin hieu kenh lan luot 8 kenh
;dung he thong 1: vi dieu khien 8951 ket noi voi khoi ADC0809 va 8 led 7 doan
;nhap du lieu tu 8 kenh ngo vao lien tuc
;hien thi tren 8 led 7 doan so nhi phan la ket qua sau khi chuyen doi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

          led7      equ   p1      ;dieu khien cac doan a,b,c,...
          quet      equ   p2      ;dieu khien quet transistor
          inadc     equ   p3      ;nhap du lieu tu adc
          control   equ   p0
          ALE       bit   p0.3
          start     bit   p0.4

          org   0000h
          mov   dptr,#ma7doan

          mov   20h,#88h          ;nap ma chu A
          mov   21h,#0a1h         ;nap ma chu d
          mov   22h,#0c6h         ;nap ma chu C

          mov   24h,#0ffh
          mov   25h,#0ffh

          mov   r6,#000           ;bien dem thu tu kenh can chuyen
maina:    mov   r5,#200           ;so lan lap lai cho moi kenh

main:     lcall ctcd_adc          ;goi chtr con chuyen doi du lieu
          lcall gma_hex_bcd       ;goi chtrinh con chuyen so hex sang ma 7 doan
          djnz  r5,main

          inc   r6                ;khoa gia tri R7 chi trong vung[0->7]
          mov   a,r6
          anl   a,#7
          mov   r6,a
          ljmp  maina             ;nhay ve chuyen doi tro lai
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con chuyen doi du lieu analog sang so ket qua luu trong A
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ctcd_adc:       mov     control,r6              ;goi ma chon kenh ra port 3
                setb    ale                     ;cho ALE=1
                nop                             ;delay 1 it thoi gian
                nop
                setb    start                   ;start = 1
                nop
                nop
                clr     ale
                clr     start
                lcall   delayhthi               ;goi chtr con delay co hien thi
                mov     a,inadc                 ;doc du lieu sau khi chuyen doi
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtrinh con chuyen so hex thanh so BCD va sau do thi chuyen
;ma BCD thanh ma 7 doan de hien thi so thap phan
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
gma_hex_bcd:
                mov     b,#10                   ;chuyen so hex sang ma BCD
                div     ab
                mov     10h,b                   ;luu hang don vi BCD

                mov     b,#10
                div     ab                      ;(a) chua so hang tram, (b) chua hang chuc

                movc    a,@a+dptr
                mov     25h,a                   ;cat so hang tram

                mov     a,b
                movc    a,@a+dptr
                mov     26h,a

                mov     a,10h
                movc    a,@a+dptr
                mov     27h,a

                mov     a,r6                    ;giai ma thu tu kenh tuong ung
                movc    a,@a+dptr
                mov     23h,a
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh co delay co hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delayhthi:      mov     7eh,#1
pqn:            mov     7fh,#30
delpqn:         lcall   hthi                    ;goi chuong trinh con hien thi
                djnz    7fh,delpqn
                djnz    7eh,pqn
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           mov     r0,#27h
                mov     a,#01111111b

hth:            mov     led7,@r0                ;lay du lieu
                mov     quet,a
                lcall   delay10
                mov     quet,#0ffh              ; tat het de chong lem
```

```
                dec     r0
                setb    c
                rrc     a
                jc      hth
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay10:        mov     7ch,#50h
                djnz    7ch,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao ma 7 doan tu so '0' den so '9'
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:        db      0C0h,0F9h,0A4h,0B0h,099h,092h,082h,0F8h
                db      080h,090h,088h,083h,0c6h,0a1h,086h,08eh
end
```

3. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch chạy đúng yêu cầu.

III. Bài tập:

1. Thay led 7 đoạn bằng LCD, hãy viết lại chương trình trên cho phù hợp.
2. Viết chương trình điều khiển ADC chuyển đổi 8 kênh hiển thị trên LCD, mỗi kênh sử dụng 4 kí tự: 1 kí tự cho số thứ tự kênh và 3 số còn lại là dữ liệu số chuyển đổi.
3. Hãy kết nối vi điều khiển với ADC (8 đường data số, 2 đường điều khiển), kết nối với LCD (8 đường data, 3 đường điều khiển), kết nối với 1 led đơn tượng trưng cho 1 relay. Đèn led đơn sáng khi giá trị chuyển đổi nhỏ hơn 80, nếu giá trị lớn hơn thì đèn tắt, LCD hiển thị giá trị chuyển đổi. Để thay đổi tín hiệu tương tự ở ngõ vào ta dùng biến trở hay dùng cảm biến nhiệt độ LM35.
4. Hãy kết nối vi điều khiển với ADC (8 đường data số, 2 đường điều khiển), kết nối với LCD (8 đường data, 3 đường điều khiển), kết nối với 2 led đơn tượng trưng cho 2 relay. Khi nhiệt độ nằm trong khoảng từ (80 đến 90) thì tắt bớt 1 relay và khi lớn hơn 95 thì tắt luôn relay còn lại. Khi nhiệt độ hạ thì đóng lại
5. Hãy kết nối thêm bàn phím ma trận để có thêm chức năng thay đổi giá trị điều khiển tuỳ ý.

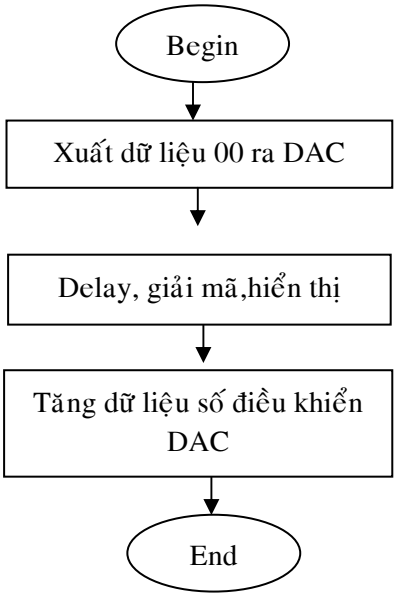
THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 7 -4** CHƯƠNG TRÌNH ĐIỀU KHIỂN DAC 0808 CHUYỂN ĐỔI TÍN HIỆU SỐ THÀNH TÍN HIỆU TƯƠNG TỰ.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách viết chương trình điều khiển DAC 0808 chuyển đổi dữ liệu số sang tín hiệu và kết hợp với các chương trình giải mã, hiển thị kết quả bằng số thập phân.

II. Trình tự thực hiện:

1. Giải thuật:



2. Kết nối mạch theo trình tự :
   - Kết nối port1 điều khiển các đoạn a,b,c,d,e,f,g,dp.
   - Kết nối port2 điều khiển các transistor của 8 led 7 đoạn.
   - Kết nối port3 điều khiển các ngõ vào số của IC DAC 0808.
3. Khởi động phần mềm, biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Chuong trinh dieu khien DAC 0808 chuyen doi tin hieu so thanh tin hieu tuong tu
;so nhi phan can chuyen doi luu trong thanh ghi r2
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

led7        equ     p1      ;dieu khien cac doan a,b,c,...
quet        equ     p2      ;dieu khien quet transistor
outdac      equ     p3      ;nhap du lieu tu adc

org     0000h               ;khai bao dia chi bat dau cua chuong trinh

mov     dptr,#ma7doan       ;nap ma 7 doan vao thanh ghi dptr
mov     20h,#0A1h           ;ma chu d
mov     21h,#088h           ;ma chu a
mov     22h,#0C6h           ;ma chu C

mov     23h,#0FFh
```

```
                mov     24h,#0FFh
                mov     r2,#00
dac2:           lcall   hex_bcd_gma         ;chuyen so hex thanh so BCD, ma 7 doan
                mov     outdac,r2           ;chuyen ma nhi phan ra port 1 dieu khien dac
                inc     r2                  ;tang du lieu so len 1
                lcall   delay_hthi
                sjmp    dac2                ;quay lai chuyen doi tiep

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;ch trinh chuyen so hex trong thanh ghi r1
;xu li phan chuyen so hex sang so bcd
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hex_bcd_gma:    mov     a,r2
                mov     b,#10
                div     ab
                mov     r7,b                ;cat tam hang don vi
                mov     b,#10
                div     ab
                mov     r6,b                ;cat tam hang chuc
                mov     r5,a                ;cat tam hang tram

;xu li phan giai ma sang led 7 doan
                mov     a,r7
                movc    a,@a+dptr
                mov     27h,a               ;cat so hang don vi vao o nho 27h

                mov     a,r6
                movc    a,@a+dptr
                mov     26h,a               ;cat so hang chuc vao o nho 26h

                mov     a,r5
                movc    a,@a+dptr
                mov     25h,a               ;cat so hang tram vao o nho 25h
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay co ghep chuong trinh hien thi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

delay_hthi:     mov     77h,#1
xde3:           mov     78h,#3
xde2:           mov     79h,#0
xde1:           lcall   hthi
                djnz    79h,xde1
                djnz    78h,xde2
                djnz    77h,xde3
                ret

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con hien thi tren led so thap phan cua du lieu can chuyen doi
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
hthi:           mov     r4,#08h             ;bien dem 8 lan goi
                mov     r1,#20h             ;dia chi vung nho hien thi
                mov     a,#11111110b        ;ma quet

hthilb:         mov     led7,@r1            ;lay du lieu tung kenh
                mov     quet,a
                lcall   delay
                mov     quet,#0ffh          ;tat het de chong lem

                inc     r1                  ;tang len de lay byte ke
                rl      a                   ;chuyen sang led ke
                djnz    r4,hthilb
```

```
                ret

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh con delay
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:          mov     r6,#50h
                djnz    r6,$
                ret
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao ma 7 doan tu so '0' den so '9'
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ma7doan:        db      0C0h,0F9h,0A4h,0B0h,099h,092h,082h,0F8h
                db      080h,090h,088h,083h,0c6h,0a1h,086h,08eh


end
```

4. Thực hiện các bước giống như các bài chuẩn cho đến khi mạch chạy đúng yêu cầu.

III. Bài tập:

1. Hãy viết chương trình điều khiển DAC 0808 tạo xung tam giác.
2. Hãy viết chương trình điều khiển DAC 0808 tạo xung vuông.

PHẦN 8

ỨNG DỤNG HỆ THỐNG II

BÀI SỐ : 8-1	**THỰC HÀNH VI ĐIỀU KHIỂN** CHƯƠNG TRÌNH ĐIỀU KHIỂN LED SÁNG TẮT THÔNG QUA 8255 CỦA HỆ THỐNG II.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách sử dụng hệ thống II, khởi tạo 8255 và viết chương trình dùng các port của 8255 điều khiển led sáng tắt.

II. Trình tự thực hiện:

1. Kết nối mạch theo trình tự :

   Dùng bus dây (8 sợi) kết nối 1PORTA, 1PORTB với 16 LED đơn.

2. Khởi động phần mềm, biên soạn chương trình sau:.

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien 16 led chop tat bang vi dieu khien thong qua 8255_1
;8255_1 co dia chi portA = 4000h; portB = 4001h; portC = 4002h; cw1   = 4003h
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

            porta1      equ   4000h
            portb1      equ   4001h
            portc1      equ   4002h
            cw1         equ   4003h

            org   8000h             ;khai bao dia chi bat dau
            mov   dptr,#cw1         ;nap dia chi khoi tao vao dptr
            mov   a,#80h            ;tu dieu khien la 80 de 3 port deu xuat
            movx  @dptr,a           ;goi ra thanh ghi dieu khien

b51:        mov   dptr,#porta1      ;nap dia chi port A vao dptr
            mov   a,#0ffh           ;goi 11111111 ra de 8 led sang
            movx  @dptr,a           ;goi data ra portA

            mov   dptr,#portb1      ;nap dia chi port B vao dptr
            mov   a,#0ffh           ;goi 11111111 ra de 8 led sang
            movx  @dptr,a           ;goi data ra port B

            lcall delay             ;goi delay de co the nhin thay

            mov   dptr,#porta1      ;nap dia chi port A vao dptr
            mov   a,#0h             ;goi 00000000 ra de 8 led tat
            movx  @dptr,a           ;goi data ra portA

            mov   dptr,#portb1      ;nap dia chi port b vao dptr
            mov   a,#0h             ;goi 00000000 ra de 8 led tat
            movx  @dptr,a           ;goi data ra portA
            lcall delay
            sjmp  b51               ;nhay tro lai de lam tiep

delay:      mov   r6,#0
de2:        mov   r7,#0
            djnz  r7,$
            djnz  r6,de2
            ret
end
```

3. Sau khi viết xong tiến hành biên dịch rồi dùng menu lệnh “RUN” và dùng lệnh “send program” để gởi chương trình xuống bộ nhớ của hệ thống II. Sau khi gởi xong tiến hành chạy chương trình bằng cách vào menu lệnh “RUN” chọn lệnh “Run Addr” và gõ địa chỉ bắt đầu của chương trình là 8000 vào rồi ấn enter. (trong trường hợp này chương trình viết tại địa chỉ 8000H nếu bạn viết tại địa chỉ khác ví dụ như 9000H thì bạn gõ vào 9000 rồi ấn enter).

   - Trên menu có nút lệnh 3 truy xuất nhanh như sau:  là gởi chương trình xuống bộ nhớ RAM của hệ thống II, nút  là nút chạy chương trình mặc nhiên tại 8000 bạn không cần gõ địa chỉ. Nút  là nút có chức năng như “run addr”.

III. Bài tập:

6. Hãy viết chương trình điều khiển 24 led kết nối với 3 port A, B, C sáng dần và tắt dần.
7. Hãy viết một số chương trình đã học bằng hệ thống II.

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 7-2** CHƯƠNG TRÌNH ĐIỀU KHIỂN LED MA TRẬN SÁNG HÌNH TRÁI TIM RƠI THÔNG QUA 8255 CỦA HỆ THỐNG II.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách sử dụng hệ thống II, khởi tạo 8255 và viết chương trình dùng các port của 8255 điều khiển led sáng tắt.

II. Trình tự thực hiện:

1. Kết nối mạch theo trình tự :
   - Dùng bus dây (8 sợi) kết nối 1PORTA với hàng của led ma trận.
   - Dùng bus dây (8 sợi) kết nối 1PORTB với cột xanh hoặc đỏ của led ma trận.
2. Khởi động phần mềm, biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh dieu khien led ma tran hinh "trai tim"  roi tu tren xuong
;dung he thong 2 qua giao tiep voi 8255.
;ket noi 1PA (port A) voi hang, 1PB voi cot xanh hoac do
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao dia chi cac port cua 8255

       porta  equ    4000h          ;dia chi cua porta
       portb  equ    4001h          ;dia chi cua portb
       portc  equ    4002h          ;dia chi cua portc
       cw1    equ    4003h          ;dia chi cua thanh ghi dieu khien
              org    8000h

              mov    dptr,#cw1
              mov    a,#80h
              movx   @dptr,a

  main:       mov    r0,#14
              mov    r4,#0          ;luu dia chi vung du lieu byte
  loop1:      mov    r2,#0          ;bien dem so lan lap lai cua 1 ki tu

  looplan:    mov    r3,04h         ;copy r4 sang r3
              mov    r5,#1          ;ma quet cot
              mov    r1,#08         ;bien dem xu ly 8 byte


  loop8b:     mov    a,r3
              mov    dptr,#madata
              movc   a,@a+dptr

              mov    dptr,#porta
              movx   @dptr,a

              mov    a,r5
              mov    dptr,#portb
              movx   @dptr,a

              lcall  delay
              mov    a,#00          ;chong lem
              movx   @dptr,a
```

```
            inc     r3
            mov     a,r5
            rl      a
            mov     r5,a

            djnz    r1,loop8b
            djnz    r2,looplan

            mov     a,r4
            add     a,#08h
            mov     r4,a
            djnz    r0,loop1
            ljmp    main

delay:      mov     r7,#1
del2:       mov     r6,#50
            djnz    r6,$
            djnz    r7,del2
            ret

delay1s:    mov     r7,#0
pndel2:     mov     r6,#0
            djnz    r6,$
            djnz    r7,pndel2
            ret

            org     9000h
madata:     db       0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh
            db       0ffh,0ffh,0ffh,07fh,07fh,0ffh,0ffh,0ffh

            db       0ffh,0ffh,07fh,03fh,03fh,07fh,0ffh,0ffh
            db       0ffh,07fh,03fh,01fh,01fh,03fh,07fh,0ffh

            db       07fh,03fh,01fh,00fh,00fh,01fh,03fh,07fh
            db       0bfh,01fh,00fh,087h,087h,00fh,01fh,0bfh

            db       0dfh,08fh,087h,0c3h,0c3h,087h,08fh,0dfh
            DB      0efH,0C7H,0C3H,0E1H,0E1H,0C3H,0C7H,0EFH

            DB      0F7H,0E3H,0E1H,0F0H,0F0H,0E1H,0E3H,0F7H
            DB      0FBH,0F1H,0F0H,0F8H,0F8H,0F0H,0F1H,0FBH

            DB      0FDH,0F8H,0F8H,0FCH,0FCH,0F8H,0F8H,0FDH
            DB      0FEH,0FCH,0FCH,0FEH,0FEH,0FCH,0FCH,0FEH

            DB      0FFH,0FEH,0FEH,0FFH,0FFH,0FEH,0FEH,0FFH
            db       0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh
        end
```

3. Thực hiện các bước giống như các bài trên cho đến khi mạch chạy đúng yêu cầu.

III. **Bài tập ứng dụng**:

THỰC HÀNH VI ĐIỀU KHIỂN **BÀI SỐ : 7-3** CHƯƠNG TRÌNH ĐIỀU KHIỂN LED MA TRẬN SÁNG CHUỖI CÁC KÍ TỰ THÔNG QUA 8255 CỦA HỆ THỐNG II.	NGÀY : SỐ TIẾT : LỚP : MSSV :

I. Mục đích yêu cầu:

Biết cách sử dụng hệ thống II, khởi tạo 8255 và viết chương trình dùng các port của 8255 điều khiển led sáng tắt.

II. Trình tự thực hiện:

1. Kết nối mạch theo trình tự:
   - Dùng bus dây (8 sợi) kết nối 1PORTA với hàng của led ma trận.
   - Dùng bus dây (8 sợi) kết nối 1PORTB với cột xanh của led ma trận.
   - Dùng bus dây (8 sợi) kết nối 1PORTB với cột đỏ của led ma trận.
2. Khởi động phần mềm, biên soạn chương trình sau:

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chuong trinh chay chuoi SPKT VA CAC KI TU TU A DEN y
;tu trai sang phai hien thi 2 mau xanh do
;dung he thong 2 qua giao tiep voi 8255.
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;khai bao dia chi cac port cua 8255

       porta    equ        4000h    ;dia chi cua porta
       portb    equ        4001h    ;dia chi cua portb
       portc    equ        4002h    ;dia chi cua portc
       cw1      equ        4003h    ;dia chi cua thanh ghi dieu khien

         org    8000h

         mov    dptr,#cw1
         mov    a,#80h
         movx   @dptr,a
main:    lcall  xanh
         lcall  do
         sjmp   main

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con hien thi mau xanh
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

xanh:    mov    dptr,#portc
         mov    a,#00
         movx   @dptr,a

         mov    r4,#0       ;luu dia chi vung du lieu byte
loop1:   mov    r2,#0       ;bien dem so lan lap lai cua 1 ki tu
looplan: mov    r3,04h      ;copy r4 sang r3
         mov    r5,#1       ;ma quet cot
         mov    r1,#08      ;bien dem xu ly 8 byte

loop8b:  mov    a,r3
         mov    dptr,#madata
         movc   a,@a+dptr
         cjne   a,#00,loope
         ret
```

```
loope:    mov    dptr,#porta
          movx   @dptr,a

          mov    a,r5
          mov    dptr,#portb
          movx   @dptr,a

          lcall  delay
          mov    a,#00                ;chong lem
          movx   @dptr,a

          inc    r3

          mov    a,r5
          rl     a
          mov    r5,a

          djnz   r1,loop8b
          djnz   r2,looplan

          inc    r4
          sjmp   loop1

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;chtr con hien thi mau do
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
do:       mov    dptr,#portb
          mov    a,#00
          movx   @dptr,a

          mov    r4,#0                ;luu dia chi vung du lieu byte
dloop1:   mov    r2,#0                ;bien dem so lan lap lai cua 1 ki tu

dlooplan: mov    r3,04h               ;copy r4 sang r3
          mov    r5,#1                ;ma quet cot
          mov    r1,#08               ;bien dem xu ly 8 byte

dloop8b:  mov    a,r3
          mov    dptr,#madata
          movc   a,@a+dptr
          cjne   a,#00,dloope
          ret

dloope:   mov    dptr,#porta
          movx   @dptr,a

          mov    a,r5
          mov    dptr,#portc
          movx   @dptr,a

          lcall  delay
          mov    a,#00                ;chong lem
          movx   @dptr,a
          inc    r3
          mov    a,r5
          rl     a
          mov    r5,a

          djnz   r1,dloop8b
          djnz   r2,dlooplan
          inc    r4
          sjmp   dloop1
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
delay:    mov     r7,#1
del2:     mov     r6,#50
          djnz    r6,$
          djnz    r7,del2
          ret

delay1s:   mov    r7,#0
pndel2:   mov     r6,#0
          djnz    r6,$
          djnz    r7,pndel2
          ret

madata:db         0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh

        db        0CDH,0B6H,0B6H,0B6H,0D9H,0ffh
        db        080H,0B7H,0B7H,0B7H,0CFH,0ffh
        db        080H,0E7H,0DBH,0BDH,0FEH,0ffh
        db        0BFH,0BFH,080H,0BFH,0BFH,0ffh

        db        0E0h,0DBh,0BBh,0DBh,0E0h,0ffh
        db        080h,0B6h,0B6h,0B6h,0C9h,0ffh
        db        0C1h,0BEh,0BEh,0BEh,0DDh,0ffh

        db        080h,0BEh,0BEh,0BEh,0C1h,0ffh
        db        080h,0B6h,0B6h,0B6h,0BEh,0ffh
        db        080h,0B7h,0B7h,0B7h,0BFh,0ffh

        db        0C1h,0BEh,0B6h,0B6h,0D1h,0ffh
        db        080h,0F7h,0F7h,0F7h,080h,0ffh
        db        0BEh,0BEh,080h,0BEh,0BEh,0ffh

        db        080h,0E7h,0DBh,0BDh,0FEh,0ffh
        db        080h,0FEh,0FEh,0FEh,0FDh,0ffh
        db        080h,0DFh,0EFh,0DFh,080h,0ffh

        db        080h,0DFh,0EFh,0F7h,080h,0ffh
        db        0C1h,0BEh,0BEh,0BEh,0C1h,0ffh
        db        080H,0B7H,0B7H,0B7H,0CFH,0ffh
        db        0C1h,0BEh,0BAh,0BCh,0C0h,0ffh

        db        080h,0B7h,0B3h,0B5h,0CEh,0ffh
        db        0CDH,0B6H,0B6H,0B6H,0D9H,0ffh
        db        0BFH,0BFH,080H,0BFH,0BFH,0ffh
        db        081h,0FEh,0FEh,0FEh,081h,0ffh
        db        083h,0FDh,0FEh,0FDh,083h,0ffh
        db        0BEh,0DDh,0E3h,0DDh,0BEh,0ffh
        db        08Fh,0F7h,0F0h,0F7h,08Fh,0ffh
        db        0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh     ;
        db        00h                                         ;dataend
        end
```

3. Thực hiện các bước giống như các bài 7-1 cho đến khi mạch chạy đúng yêu cầu.

III. **Bài tập ứng dụng**:

PHẦN 8:

CÁC CHƯƠNG TRÌNH ỨNG DỤNG

Mạch đồng hồ số dùng vi điều khiển 89S52, realtime và led 7 đoạn
Mạch đồng hồ số dùng vi điều khiển 89S52, realtime và led 7 đoạn
Mạch đồng hồ số có thêm báo chuông tiết học dùng vi điều khiển 89S52, realtime và led 7 đoạn
Mạch đồng hồ số có thêm báo chuông tiết học dùng vi điều khiển 89S52, realtime và LCD

Mạch nạp vi điều khiển 89C51 và 89C52
Mạch nạp vi điều khiển 89S51, 89S52 và 89S8252
Mạch giao tiếp vi điều khển với máy tính qua cổng giao tiếp RS232
Mạch giao tiếp vi điều khển với máy tính qua cổng giao tiếp RS232
Mạch giao tiếp vi điều khển với máy tính qua cổng giao tiếp RS232 chuyển thành RS485

Mạch giao tiếp 2 hệ thống vi điều khiển với nhau.
Mạch giao tiếp nhiều hệ thống vi điều khiển
Mạch giao tiếp vi điều khiển với bàn phím máy tính và LCD
Mạch giao tiếp vi điều khiển với đầu đọc mã vạch và LCD

Mạch điều khiển quang báo dùng nhiều led ma trận 8x8
Mạch điều khiển quang báo dùng nhiều led ma trận 8x8_ có giao tiếp với máy tính
Mạch điều khiển quang báo dùng nhiều led ma trận 8x8_ có giao tiếp với bàn phím máy tính