BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG……..

LUẬN VĂN

Thiết kế và thi công hệ thống Kit Vi Điều Khiển 8951

PHẦN MỞ ĐẦU

I. KHÁI QUÁT VẤN ĐỀ

Ngày nay kỹ thuật vi điều khiển đã trở nên quen thuộc trong các ngành kỹ thuật và trong dân dụng. Từ các dây chuyền sản xuất lớn đến các thiết bị gia dụng, chúng ta đều thấy sự hiện diện của vi điều khiển. Các bộ vi điều khiển có khả năng xử lý nhiều hoạt động phức tạp mà chỉ cần một chip vi mạch nhỏ, nó đã thay thế các tủ điều khiển lớn và phức tạp bằng những mạch điện gọn nhẹ, dễ dàng thao tác sử dụng.

Vi điều khiển không những góp phần vào kỹ thuật điều khiển mà còn góp phần to lớn vào việc phát triển thông tin. Đó chính là sự ra đời của hàng loạt thiết bị tối tân trong ngành viễn thông, truyền hình, đặc biệt là sự ra đời của mạng Internet –siêu xa lộ thông tin, góp phần đưa con người đến đỉnh cao của nền văn minh nhân loại.

Chính vì các lý do trên, việc tìm hiểu, khảo sát vi điều khiển là điều mà các sinh viên ngành điện mà đặc biệt là chuyên ngành kỹ thuật điện-điện tử phải hết sức quan tâm. Đó chính là một nhu cầu cần thiết và cấp bách đối với mỗi sinh viên, đề tài này được thực hiện chính là đáp ứng nhu cầu đó.

Các bộ điều khiển sử dụng vi điều khiển tuy đơn giản nhưng để vận hành và sử dụng đươc lại là một điều rất phức tạp. Phần công việc xử lý chính vẫn phụ thuộc vào con người, đó chnh là chương trình hay phần mềm. Tuy chúng ta thấy các máy tính ngày nay cực kỳ thông minh, giải quyết các bài toán phức tạp trong vài phần triệu giây, nhưng đó cũng là dựa trên sự hiểu biết của con người. Nếu không có sự tham gia của con người thì hệ thống vi điều khiển cũng chỉ là một vật vô tri. Do vậy khi nói đến vi điều khiển cũng giống như máy tính bao gồm 2 phần là phần cứng và phần mềm.

Các bộ vi điều khiển theo thời gian cùng với sự phát triển của công nghệ bán dẫn đã tiến triển rất nhanh, từ các bộ vi điều khiển 4 Bit đơn giản đến các bộ vi điều khiển 32 Bit. Với công nghệ tiên tiến ngày nay các máy tính có thể đi đến việc suy nghĩ, tri thức các thông tin đưa vào, đó là các máy tính thuộc thế hệ trí tuệ nhân tạo.

Mặc dù vi điều khiển đã đi được những bước dài như vậy nhưng để tiếp cận được với kỹ thuật này không thể là một việc có được trong một sớm một chiều. Việc hiểu được cơ chế hoạt động của bộ vi điều khiển 8 Bit là cơ sở để chúng ta tìm hiểu và sử dụng các bộ vi điều khiển tối tân hơn, đây chính là bước đi đầu tiên khi chúng ta muốn xâm nhập sâu hơn vào lĩnh vực này.

Để tìm hiểu bộ vi điều khiển một cách khoa học và mang lại hiệu quả cao làm nền tản cho việc xâm nhập vào những hệ thống tối tân hơn. Việc trang bị những kiến thức về vi điều khiển cho sinh viên là hết sức cần thiết. Xuất phát từ thực tiễn này em đã đi đến quyết định **Thiết kế và thi công hệ thống Kit Vi Điều Khiển 8951**. Nhằm đáp ứng nhu cầu ham muốn học hỏi của bản thân.

II.GIỚI HẠN VẤN ĐỀ

Do thời gian nghiên cứu và thực hiện đề tài chỉ giới hạn trong vòng 7 tuần lễ, vốn kiến thức và việc tìm hiểu sâu về một hệ vi điều khiển còn hạn chế, luận án này chỉ thực hiện trong phạm vi sau:

*** Phần I : Giới thiệu các linh kiện sử dụng trong mạch**
*** Phần II : Thiết kế và thi công phần cứng .**
*** Phần III : Thiết kế phần mềm**
*** Phần IV : Phụ lục**

III. MỤC TIÊU NGHIÊN CỨU

Dựa trên cơ sở của các đề tài vi xử lý và vi điều khiển, đặc biệt là các tính năng của chúng cũng như các họ IC giao tiếp, hiển thị và giải mã …, nhằm thiết kế một hệ thống vi điều khiển góp phần làm phong phú thêm cho việc hiểu biết về lĩnh vực này đồng thời có thể mở rộng và định hướng cho những đề tài sau.

PHẦN I : GIỚI THIỆU CÁC LINH KIỆN SỬ DỤNG TRONG MẠCH

CHƯƠNG I KHẢO SÁT VI ĐIỀU KHIểN 8951

I. GIỚI THIỆU CẤU TRÚC PHẦN CỨNG HỌ MCS-51 (8951):

1.*Giới thiệu họ MCS-51:*

MCS-51 là họ IC vi điều khiển do hãng Intel sản xuất. Các IC tiêu biểu cho họ là 8051 và 8031. Các sản phẩm MCS-51 thích hợp cho những ứng dụng điều khiển. Việc xử lý trên Byte và các toán số học ở cấu trúc dữ liệu nhỏ được thực hiện bằng nhiều chế độ truy xuất dữ liệu nhanh trên RAM nội. Tập lệnh cung cấp một bảng tiện dụng của những lệnh số học 8 Bit gồm cả lệnh nhân và lệnh chia. Nó cung cấp những hỗ trợ mở rộng trên Chip dùng cho những biến một Bit như là kiểu dữ liệu riêng biệt cho phép quản lý và kiểm tra Bit trực tiếp trong điều khiển và những hệ thống logic đòi hỏi xử lý luận lý.

8951 là một vi điều khiển 8 Bit, chế tạo theo công nghệ CMOS chất lượng cao, công suất thấp với 4 KB PEROM (Flash Programeable and erasable read only memory). Thiết bị này được chế tạo bằng cách sử dụng bộ nhớ không bốc hơi mật độ cao của ATMEL và tương thích với chuẩn công nghiệp MCS-51 về tập lệnh và các chân ra. PEROM ON-CHIP cho phép bộ nhớ lập trình được lập trình trong hệ thống hoặc bởi một lập trình viên bình thường. Bằng cách kết hợp một CPU 8 Bit với một PEROM trên một Chip đơn, ATMEL AT89C51 là một vi điều khiển mạnh (có công suất lớn) mà nó cung ấp một sự linh động cao và giải pháp về giá cả đối với nhiều ứng dụng vi điều khiển.

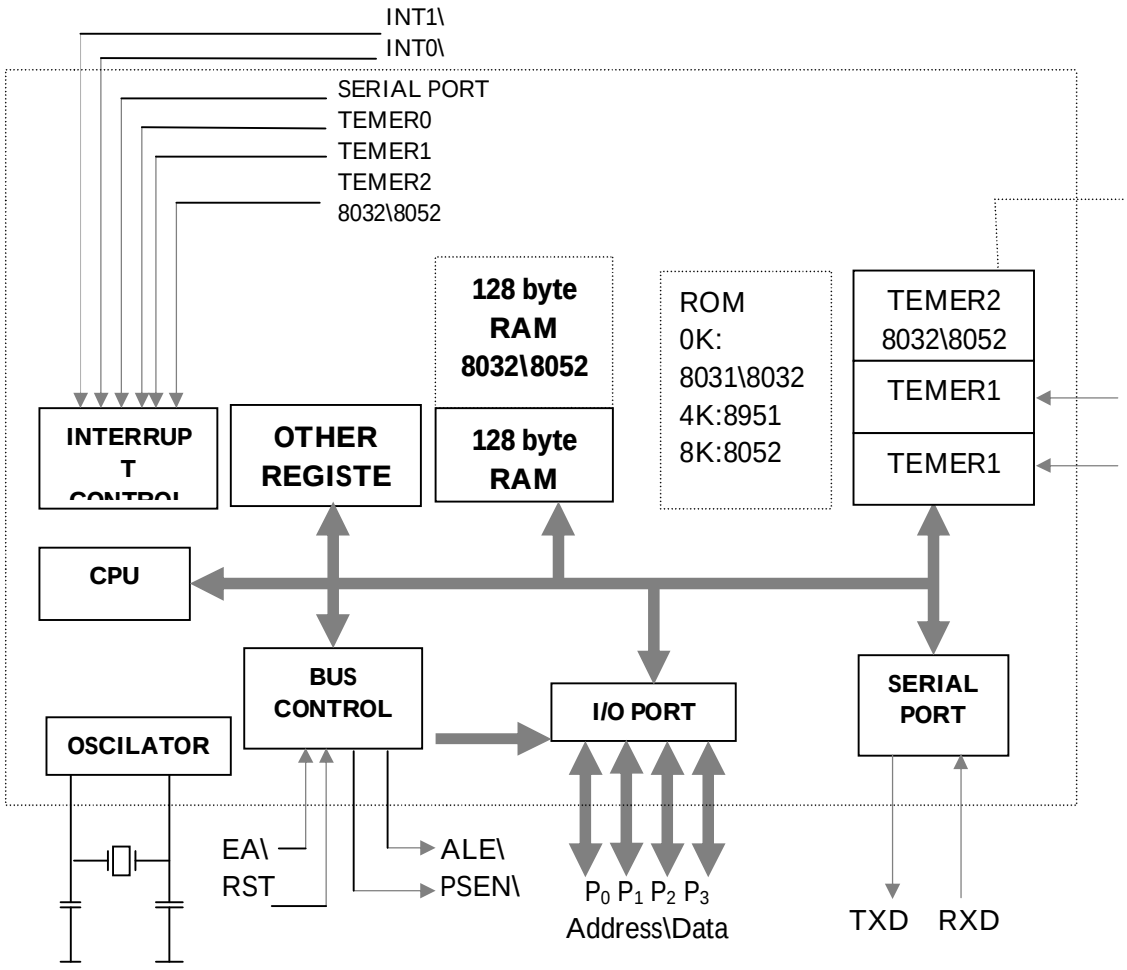
AT89C51 cung cấp những đặc tính chuẩn như sau: 4 KB bộ nhớ chỉ đọc có thể xóa và lập trình nhanh (EPROM), 128 Byte RAM, 32 đường I/O, 2 TIMER/COUNTER 16 Bit, 5 vectơ ngắt có cấu trúc 2 mức ngắt, một Port nối tiếp bán song công, 1 mạch dao động tạo xung Clock và bộ dao động ON-CHIP. Thêm vào đó, AT89C51 được thiết kế với logic tĩnh cho hoạt động đến mức không tần số và hỗ trợ hai phần mềm có thể lựa chọn những chế độ tiết kiệm công suất, chế độ chờ (IDLE MODE) sẽ dừng CPU trong khi vẫn cho phép RAM, timer/counter, port nối tiếp và hệ thống ngắt tiếp tục hoạt động. Chế độ giảm công suất sẽ lưu nội dung RAM nhưng sẽ treo bộ dao động làm mất khả năng hoạt động của tất cả những chức năng khác cho đến khi Reset hệ thống.

Các đặc điểm của 8951 được tóm tắt như sau:

- 4 KB bộ nhớ có thể lập trình lại nhanh, có khả năng tới 1000 chu kỳ ghi xoá
- Tần số hoạt động từ: 0Hz đến 24 MHz
- 3 mức khóa bộ nhớ lập trình
- 2 bộ Timer/counter 16 Bit
- 128 Byte RAM nội.
- 4 Port xuất /nhập I/O 8 bit.
- Giao tiếp nối tiếp.
- 64 KB vùng nhớ mã ngoài

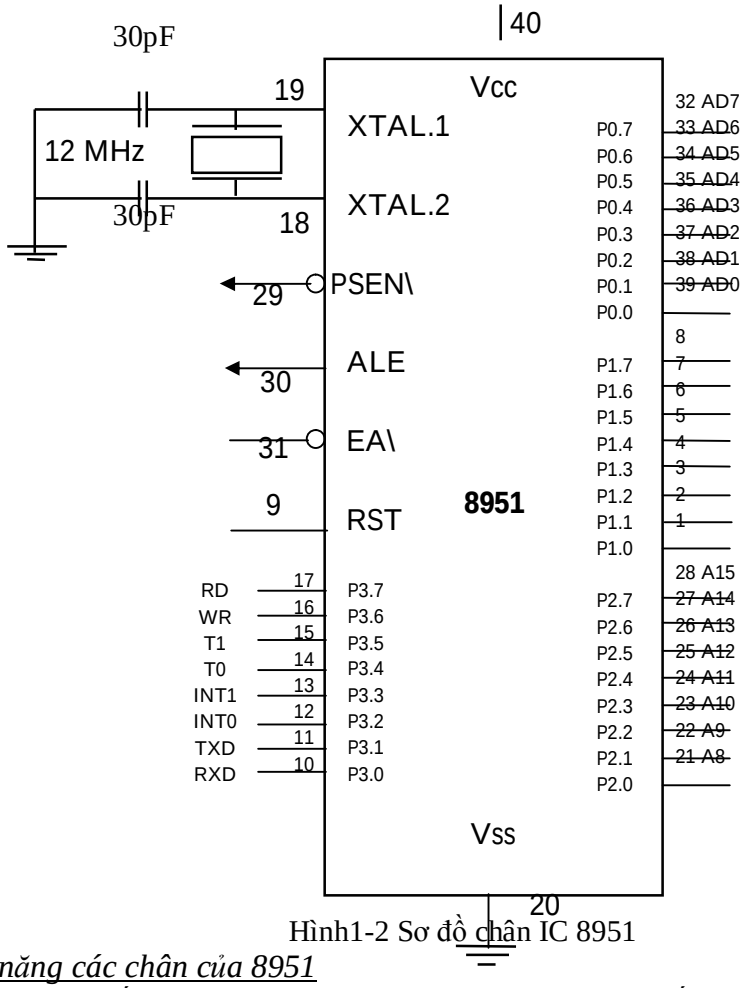
- 64 KB vùng nhớ dữ liệu ngoại.
- Xử lý Boolean (hoạt động trên bit đơn).
- 210 vị trí nhớ có thể định vị bit.
- 4 µs cho hoạt động nhân hoặc chia.

2.*Sơ đồ khối của AT89C51 được trình bày ở hình 1-1*



II. KHẢO SÁT SƠ ĐỒ CHÂN 8951, CHỨC NĂNG TỪNG CHÂN:

1.Sơ đồ chân 8951:



Hình1-2 Sơ đồ chân IC 8951

2.Chức năng các chân của 8951

- 8951 có tất cả 40 chân có chức năng như các đường xuất nhập. Trong đó có 24 chân có tác dụng kép (có nghĩa 1 chân có 2 chức năng), mỗi đường có thể hoạt động như đường xuất nhập hoặc như đường điều khiển hoặc là thành phần của các bus dữ liệu và bus địa chỉ.

a.Các Port:

- **Port 0:**

- Port 0 là port có 2 chức năng ở các chân 32 – 39 của 8951. Trong các thiết kế cỡ nhỏ không dùng bộ nhớ mở rộng nó có chức năng như các đường IO. Đối với các thiết kế cỡ lớn có bộ nhớ mở rộng, nó được kết hợp giữa bus địa chỉ và bus dữ liệu.

- **Port 1:**

- Port 1 là port IO trên các chân 1-8. Các chân được ký hiệu P1.0, P1.1, P1.2, … có thể dùng cho giao tiếp với các thiết bị ngoài nếu cần. Port 1 không có chức năng khác, vì vậy chúng chỉ được dùng cho giao tiếp với các thiết bị bên ngoài.

- **Port 2:**

- Port 2 là 1 port có tác dụng kép trên các chân 21- 28 được dùng như các đường xuất nhập hoặc là byte cao của bus địa chỉ đối với các thiết bị dùng bộ nhớ mở rộng.

- **Port 3:**

- Port 3 là port có tác dụng kép trên các chân 10-17. Các chân của port này có nhiều chức năng, các công dụng chuyển đổi có liên hệ với các đặc tính đặc biệt của 8951 như ở bảng sau:

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

b.Các ngõ tín hiệu điều khiển:

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

- PSEN là tín hiệu ngõ ra ở chân 29 có tác dụng cho phép đọc bộ nhớ chương trình mở rộng thường được nói đến chân 0E\ (output enable) của Eprom cho phép đọc các byte mã lệnh.

- PSEN ở mức thấp trong thời gian Microcontroller 8951 lấy lệnh. Các mã lệnh của chương trình được đọc từ Eprom qua bus dữ liệu và được chốt vào thanh ghi lệnh bên trong 8951 để giải mã lệnh. Khi 8951 thi hành chương trình trong ROM nội PSEN sẽ ở mức logic 1.

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

- Khi 8951 truy xuất bộ nhơd bên ngoài, port 0 có chức năng là bus địa chỉ và bus dữ liệu do đó phải tách các đường dữ liệu và địa chỉ. Tín hiệu ra ALE ở chân thứ 30 dùng làm tín hiệu điều khiển để giải đa hợp các đường địa chỉ và dữ liệu khi kết nối chúng với IC chốt.

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

Các xung tín hiệu ALE có tốc độ bằng 1/6 lần tần số dao động trên chip và có thể được dùng làm tín hiệu clock cho các phần khác của hệ thống. Chân ALE được dùng làm ngõ vào xung lập trình cho Eprom trong 8951.

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

- Tín hiệu vào EA\ ở chân 31 thường được mắt lên mức 1 hoặc mức 0. Nếu ở mức 1, 8951 thi hành chương trình từ ROM nội trong khoảng địa chỉ thấp 8 Kbyte. Nếu ở mức 0, 8951 sẽ thi hành chương trình từ bộ nhớ mở rộng. Chân EA\ được lấy làm chân cấp nguồn 21V khi lập trnh cho Eprom trong 8951.

- **Ngõ tín hiệu RST (Reset)**:

-Ngõ vào RST ở chân 9 là ngõ vào Reset của 8951. Khi ngõ vào tín hiệu này đưa lên cao ít nhất là 2 chu kỳ máy, các thanh ghi bên trong được nạp những giá trị thích hợp để khởi động hệ thống. Khi cấp điện mạch tự động Reset.

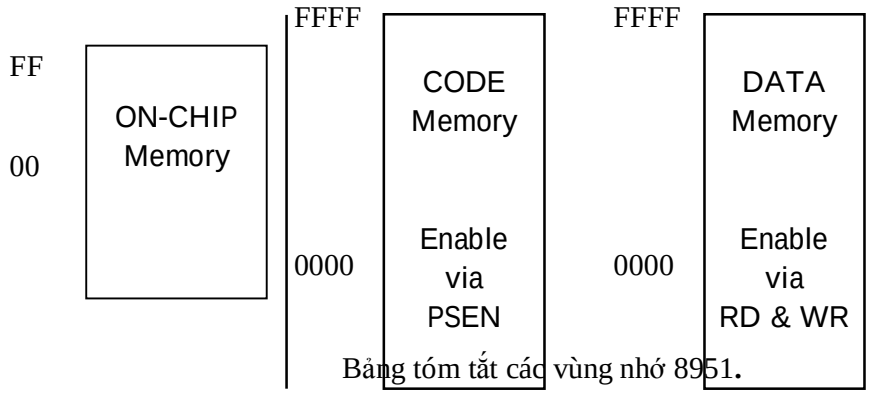
- **Các ngõ vào bộ giao động X1,X2:**

-Bộ dao động được được tích hợp bên trong 8951, khi sử dụng 8951 người thiết kế chỉ cần kết nối thêm thạch anh và các tụ như hình vẽ trong sơ đồ. Tần số thạch anh thường sử dụng cho 8951 là 12Mhz.

- Chân 40 (Vcc) được nối lên nguồn 5V.

III.CẤU TRÚC BÊN TRONG VI ĐIỀU KHIỂN

1. Tổ chức bộ nhớ:



Bảng tóm tắt các vùng nhớ 8951.

Hình 1.3 : External Momery

Bản đồ bộ nhớ Data trên Chip như sau:

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 (mặc định cho R0 -R7)							
00								

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

CÁC THANH GHI CHỨC NĂNG ĐẶC BIỆT

- Bộ nhớ trong 8951 bao gồm ROM và RAM. RAM trong 8951 bao gồm nhiều thành phần: phần lưu trữ đa dụng, phần lưu trữ địa chỉ hóa từng bit, các bank thanh ghi và các thanh ghi chức năng đặc biệt.

- 8951 có bg nhớ theo cấu trúc Harvard: có những vùng bộ nhớ riêng biệt cho chương trình và dữ liệu. Chương trình và dữ liệu có thể chứa bên trong 8951 nhưng 8951 vẫn có thể kết nối với 64K byte bộ nhớ chương trình và 64K byte dữ liệu.

Hai đặc tính cần chú ý là:

◆ Các thanh ghi và các port xuất nhập đã được định vị (xác định) trong bộ nhớ và có thể truy xuất trực tiếp giống như các địa chỉ bộ nhớ khác.

◆ Ngăn xếp bên trong Ram nội nhỏ hơn so với Ram ngoại như trong các bộ Microprocontroller khác.

RAM bên trong 8951 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.

❒ **RAM đa dụng:**

- Mặc dù trên hình vẽ cho thấy 80 byte đa dụng chiếm các địa chỉ từ 30H đến 7FH, 32 byte dưới từ 00H đến 1FH cũng có thể dùng với mục đích tương tự (mặc dù các địa chỉ này đã có mục đích khác).

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

❒ **RAM có thể truy xuất từng bit:**

- 8951 chứa 210 bit được địa chỉ hóa, trong đó có 128 bit có chứa các byte có chứa các địa chỉ từ 20F đến 2FH và các bit còn lại chứa trong nhóm thanh ghi có chức năng đặc biệt.

- Ý tưởng truy xuất từng bit bằng phần mềm là các đặc tính mạnh của microcontroller xử lý chung. Các bít có thể được đặt, xóa, AND, OR, …, với 1 lệnh đơn. Đa số các microcontroller xử lý đòi hỏi một chuỗi lệnh đọc– sửa- ghi để đạt được mục đích tương tự. Ngoài ra các port cũng có thể truy xuất được từng bít.

- 128 bit truy xuất từng bit này cũng có thể truy xuất như các byte hoặc như các bit phụ thuộc vào lệnh được dùng.

❒ **Các bank thanh ghi:**

- 32 byte thấp của bộ nhớ nội được dành cho các bank thanh ghi. Bộ lệnh 8951 hỗ trợ 8 thanh ghi có tên là R0 đến R7 và theo mặc định sau khi reset hệ thống, các thanh ghi này có các địa chỉ từ 00H đến 07H.

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

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

2*. Các thanh ghi có chức năng đặc biệt*:

- Các thanh ghi nội của 8951 được truy xuất ngầm định bởi bộ lệnh.

- Các thanh ghi trong 8951 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 điếm chương trình và thanh ghi lệnh vì các thanh ghi này hiếm khi bị tác động trực tiếp). Cũng như R0 đến R7, 8951 có 21 thanh ghi có chức năng đặc biệt (SFR: Special Function Register) ở vùng trên của RAM nội từ địa chỉ 80H đến FFH.

Chú ý: tất cả 128 địa chỉ từ 80H đến FFH không được định nghĩa, chỉ có 21 thanh ghi có chức năng đặc biệt được định nghĩa sẵn các địa chỉ.

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

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

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

BIT	SYMBOL	ADDRESS	DESCRIPTION
PSW.7	CY	D7H	Cary Flag

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

Chức năng từng bit trạng thái chương trình

- *Cờ Carry CY (Carry Flag):*

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

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

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

- *Cờ 0 (Flag 0):*

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

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

- RS1 và RS0 quyết định dãy thanh ghi tích cực. Chúng được xóa sau khi reset hệ thống và được thay đổi bởi phần mềm khi cần thiết.

- Tùy theo RS1, RS0 = 00, 01, 10, 11 sẽ được chọn Bank tích cực tương ứng là Bank 0, Bank1, Bank2, Bank3.

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

- *Cờ tràn OV (Over Flag):*

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

- *Bit Parity (P):*

- Bit tự động được set hay Clear ở mỗi chu kỳ máy để lập Parity chẵn với thanh ghi A. Sự đếm các bit 1 trong thanh ghi A cộng với bit Parity luôn luôn chẵn. Ví dụ A chứa 10101101B thì bit P set lên một để tổng số bit 1 trong A và P tạo thành số chẵn.

- Bit Parity thường được dùng trong sự kết hợp với những thủ tục của Port nối tiếp để tạo ra bit Parity trước khi phađt đi hoặc kiểm tra bit Parity sau khi thu.

- *Thanh ghi B :*

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

- Thanh ghi B có thể được dùng như một thanh ghi đệm trung gian đa mục đích. Nó là những bit định vị thông qua những địa chỉ từ F0H÷F7H.

- *Con trỏ Ngăn xếp SP (Stack Pointer):*

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

- Để khởi động SP với ngăn xếp bắt đầu tại địa chỉ 60H, các lệnh sau đây được dùng:

```
MOV SP, #5F
```

- Với lệnh trên thì ngăn xếp của 8951 chỉ có 32 byte vì địa chỉ cao nhất của RAM trên chip là 7FH. Sỡ dĩ giá trị 5FH được nạp vào SP vì SP tăng lên 60H trước khi cất byte dữ liệu.

- Khi Reset 8951, 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 động SP một giá trị mới thì bank thanh ghi 1 có thể cả 2 và 3 sẽ không dùng được vì vùng RAM 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 giá trị của bộ đếm chương trình khi bắt đầu thực hiện chương trình con và lấy lại khi kết thúc chương trình con …

- *Con trỏ dữ liệu DPTR (Data Pointer) :*

-Con trỏ dữ liệu (DPTR) được dùng để truy xuất bộ nhớ ngoài là một thanh ghi 16 bit ở địa chỉ 82H (DPL: byte thấp) và 83H (DPH: byte cao). Ba lệnh sau sẽ ghi 55H vào RAM ngoài ở địa chỉ 1000H:

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

- Lệnh đầu tiên dùng để nạp 55H vào thanh ghi A. Lệnh thứ hai dùng để nạp địa chỉ của ô nhớ cần lưu giá trị 55H vào con trỏ dữ liệu DPTR. Lệnh thứ ba sẽ di chuyển nội dung thanh ghi A (là 55H) vào ô nhớ RAM bên ngoài có địa chỉ chứa trong DPTR (là 1000H)

- *Các thanh ghi Port (Port Register):*

- Các Port của 8951 bao gồm Port0 ở địa chỉ 80H, Port1 ở địa chỉ 90H, Port2 ở địa chỉ A0H, và Port3 ở địa chỉ B0H. Tất cả các Port này đều có thể truy xuất từng bit nên rất thuận tiện trong khả năng giao tiếp.

- *Các thanh ghi Timer (Timer Register):*

- 8951 có chứa hai bộ định thời/bộ đếm16 bit được dùng cho việc định thời được đếm sự kiện. Timer0 ở địa chỉ 8AH (TL0: byte thấp) và 8CH (TH0: byte cao). Timer1 ở địa chỉ 8BH (TL1: byte thấp) và 8DH (TH1: byte cao). Việc khởi động timer được SET bởi Timer Mode (TMOD) ở địa chỉ 89H và thanh ghi điều khiển Timer (TCON) ở địa chỉ 88H. Chỉ có TCON được địa chỉ hóa từng bit.

- *Các thanh ghi Port nối tiếp (Serial Port Register):*

- 8951 chứa một Port nối tiếp cho việc trao đổi thông tin với các thiết bị nối tiếp như máy tính, modem hoặc giao tiếp nối tiếp với các IC khác. Một thanh ghi đệm dử liệu nối tiếp (SBUF) ở địa chỉ 99H sẽdữ cảhai dữ liệu truyền và dữ liệu nhập. Khi truyền dữ liệu ghi lên SBUF, khi nhận dữ liệu thì đọc SBUF. Các mode vận khác nhau được lập trình qua thanh ghi điều khiển Port nối tiếp (SCON) được địa chỉ hóa từng bit ở địa chỉ 98H.

- *Các thanh ghi ngắt (Interrupt Register):*

- 8951 có cấu trúc 5 nguồn ngắt, 2 mức ưu tiên. Các ngắt bị cấm sau khi bị reset hệ thống và sẽ được cho phép bằng việt ghi thanh ghi cho phép ngắt (IE) ở địa chỉ A8H. Cả hai được địa chỉ hóa từng bit.

- *Thanh ghi điều khiển nguồn PCON (Power Control Register):*

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

  - Bit 7 (SMOD): Bit có tốc độ Baud ở mode 1, 2, 3 ở Port nối tiếp khi set.
  - Bit 6, 5, 4: Không có địa chỉ.
  - Bit 3 (GF1) : Bit cờ đa năng 1.
  - Bit 2 (GF0) : Bit cờ đa năng 2 .
  - Bit 1 * (PD) : Set để khởi động mode Power Down và thoát để reset.
  - Bit 0 (IDL): Set để khởi động mode Idle và thoát khi ngắt mạch hoặc reset.

Các bit điều khiển Power Down và Idle có tác dụng chính trong tất cả các IC họ MSC-51 nhưng chỉ được thi hành trong sự biên dịch của CMOS.

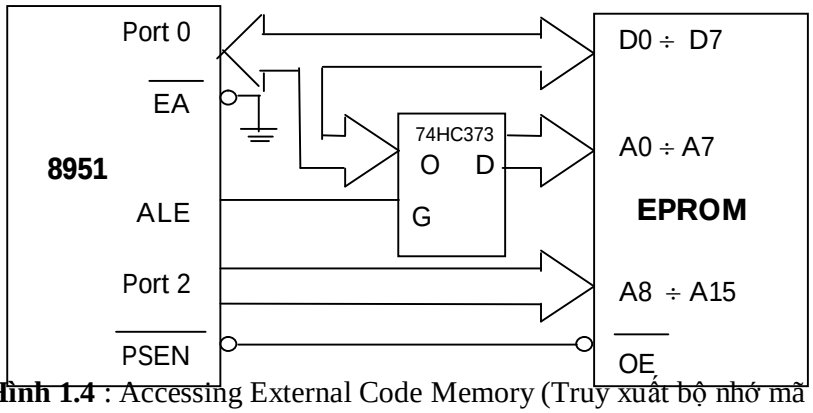
3. Bộ nhớ ngoài (external memory):

- 8951 có khả năng mở rông bộ nhớ lên đến 64K byte bộ nhớ chương trình và 64k byte bộ nhớ dữ liệu ngoài. Do đó có thể dùng thêm RAM và ROM nếu cần.

- Khi dùng bộ nhớ ngoài, Port0 không còn chức năng I/O nữa. Nó được kết hợp giữa bus địa chỉ (A0-A7) và bus dữ liệu (D0-D7) với tín hiệu ALE để chốt byte của bus địa chỉ chỉ khi bắt đầu mỗi chu kỳ bộ nhớ. Port được cho là byte cao của bus địa chỉ.

Truy xuất bộ nhớ mã ngoài (Acessing External Code Memory):

- Bộ nhớ chương trình bên ngoài là bộ nhớ ROM được cho phép của tín hiệu PSEN\. Sự kết nối phần cứng của bộ nhớ EPROM như sau:



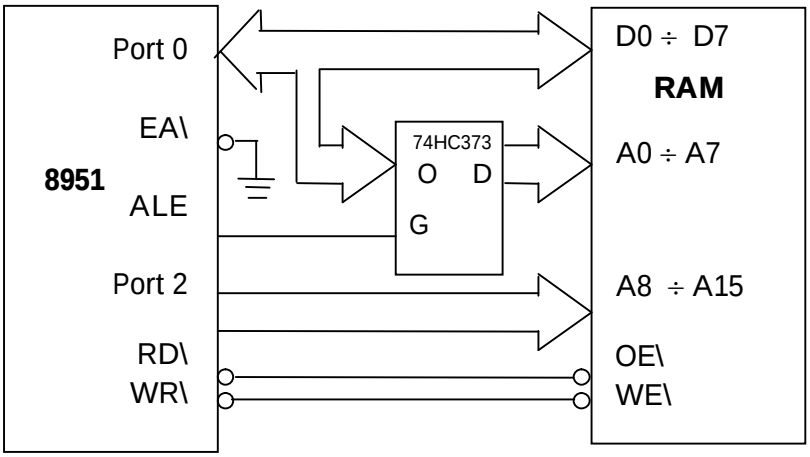
Hình 1.4 : Accessing External Code Memory (Truy xuất bộ nhớ mã ngoài)

- Trong một chu kỳ máy tiêu biểu, tín hiệu ALE tích cực 2 lần. Lần thứ nhất cho phép 74HC373 mở cổng chốt địa chỉ byte thấp, khi ALE xuống 0 thì byte thấp và byte cao của bộ đếm chương trình đều có nhưng EPROM chưa xuất vì PSEN\ chưa tích cực, khi tín hiệu lên một trở lại thì Port 0 đã có dữ liệu là Opcode. ALE tích cực lần thứ hai được giải thích tương tự và byte 2 được đọc từ bộ nhớ chương trình. Nếu lệnh đang hiện hành là lệnh 1 byte thì CPU chỉ đọc Opcode, còn byte thứ hai bỏ đi.

• *Truy xuất bộ nhớ dữ liệu ngoài (Accessing External Data Memory):*

- Bộ nhớ dữ liệu ngoài là một bộ nhớ RAM được đọc hoặc ghi khi được cho phép của tín hiệu RD\ và WR. Hai tín hiệu này nằm ở chân P3.7 (RD) và P3.6 (WR). Lệnh MOVX được dùng để truy xuất bộ nhớ dữ liệu ngoài và dùng một bộ đệm dữ liệu 16 bit (DPTR), R0 hoặc R1 như là một thanh ghi địa chỉ.

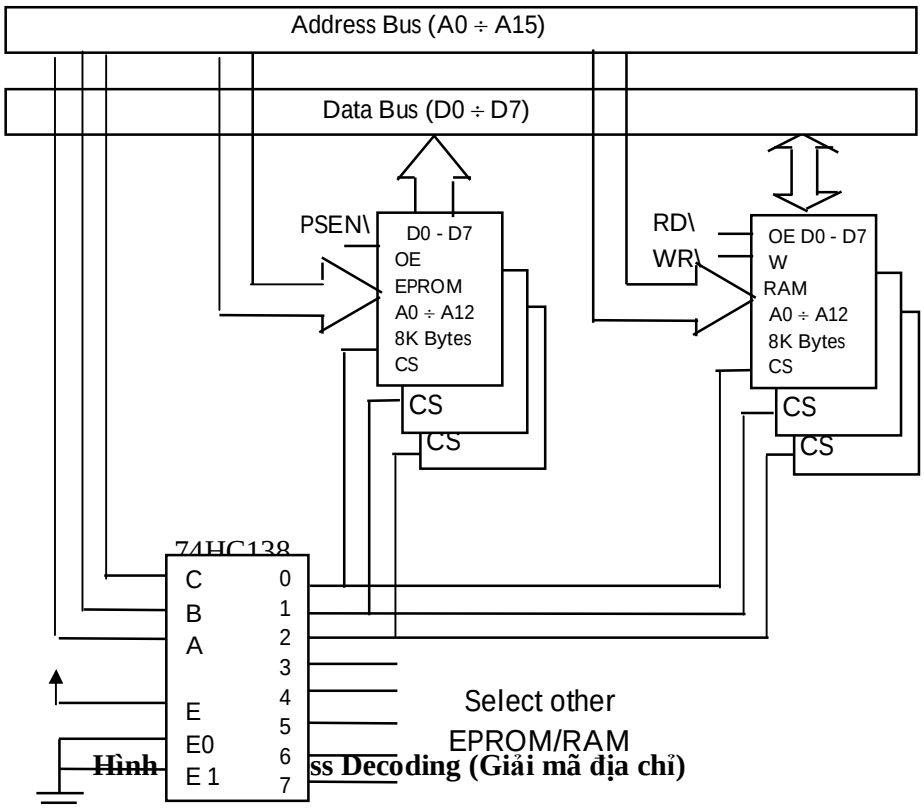
- Các RAM có thể giao tiếp với 8951 tương tự cách thức như EPROM ngoại trừ chân RD\ của 8951 nối với chân OE\ (Output Enable) của RAM và chân WR\ của 8951 nối với chânWE \của RAM. Sự nối các bus địa chỉ và dữ liệu tương tự như cách nối của EPROM.



- *Sự giải mã địa chỉ (Address Decoding):*

- Sự giải mã địa chỉ là một yêu cầu tất yếu để chọn EPROM, RAM, 8279, … Sự giải mã địa chỉ đối với 8951 để chọn các vùng nhớ ngoài như các vi điều khiển. Nếu các con EPROM hoặc RAM 8K được dùng thì các bus địa chỉ phải được giải mã để chọn các IC nhớ nằm trong phạm vi giới hạn 8K: 0000H÷1FFFH, 2000H÷3FFFH, …

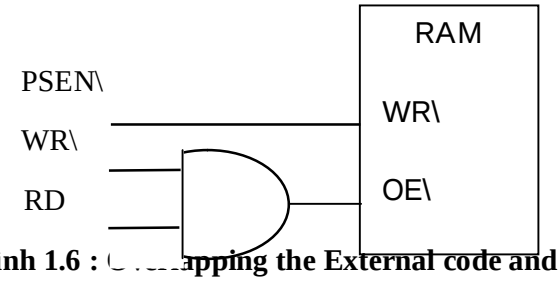
- Một cách cụ thể, IC giải mã 74HC138 được dùng với những ngõ ra của nó được nối với những ngõ vào chọn Chip CS (Chip Select) trên những IC nhớ EPROM, RAM, … Hình sau đây cho phép kết nối nhiều EPRGM và RAM.



Hình ss Decoding (Giải mã địa chỉ)

- *Sự đè lên nhau của các vùng nhớ dữ liệu ngoài:*

- Vì bộ nhớ chương trình là ROM, nên nẩy sinh một vấn đề bất tiện khi phát triển phần mềm cho vi điều khiển. Một nhược điểm chung của 8951 là các vùng nhớ dữ liệu ngoài nằm đè lên nhau, vì tín hiệu PSEN\ được dùng để đọc bộ nhớ mã ngoài và tín hiệu RD\ được dùng để đọc bộ nhớ dữ liệu, nên một bộ nhớ RAM có thể chứa cả chương trình và dữ liệu bằng cách nối đường OE\ của RAM đến ngõ ra một cổng AND có hai ngõ vào PSEN\ và RD\. Sơ đồ mạch như hình sau cho phép cho phép bộ nhớ RAM có hai chức năng vừa là bộ nhớ chương trình vừa là bộ nhớ dữ liệu:

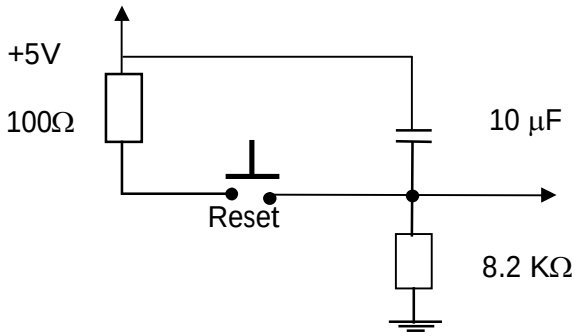


Hình 1.6 : Mapping the External code and data space

-Vậy một chương trình có thể được tải vào RAM bằng cách xem nó như bộ nhớ dữ liệu và thi hành chương trình băng cách xem nó như bộ nhớ chương trình.

Hoạt động Reset:

- 8951 có ngõ vào reset RST tác động ở mức cao trong khoảng thời gian 2 chu kỳ xung máy, sau đó xuống mức thấp để 8951 bắt đầu làm việc. RST có thể kích bằng tay bằng một phím nhấn thường hở, sơ đồ mạch reset như sau:



Hình 1.7 : Manual Reset

Trạng thái của tất cả các thanh ghi trong 8951 sau khi reset hệ thống được tóm tắt như sau:

Thanh ghi	Nội dung
Đếm chương trình PC	0000H
Thanh ghi tích lũyA	00H
Thanh ghi B	00H
Thanh ghi thái PSW	00H
SP	07H
DPRT	0000H
Port 0 đến port 3	FFH
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 được reset tại địa chỉ 0000H. Khi ngõ vào RST xuống mức thấp, chương trình luôn bắt đầu tại địa chỉ 0000H của bộ nhớ chương trình. Nội dung của RAM trên chip không bị thay đổi bởi tác động của ngõ vào reset.

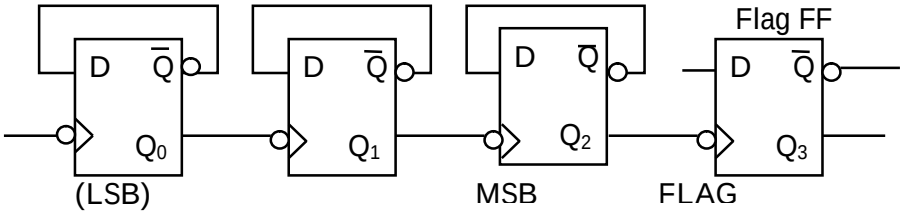
II. HOẠT ĐỘNG TIMER CỦA 8951:

1. Giới Thiệu:

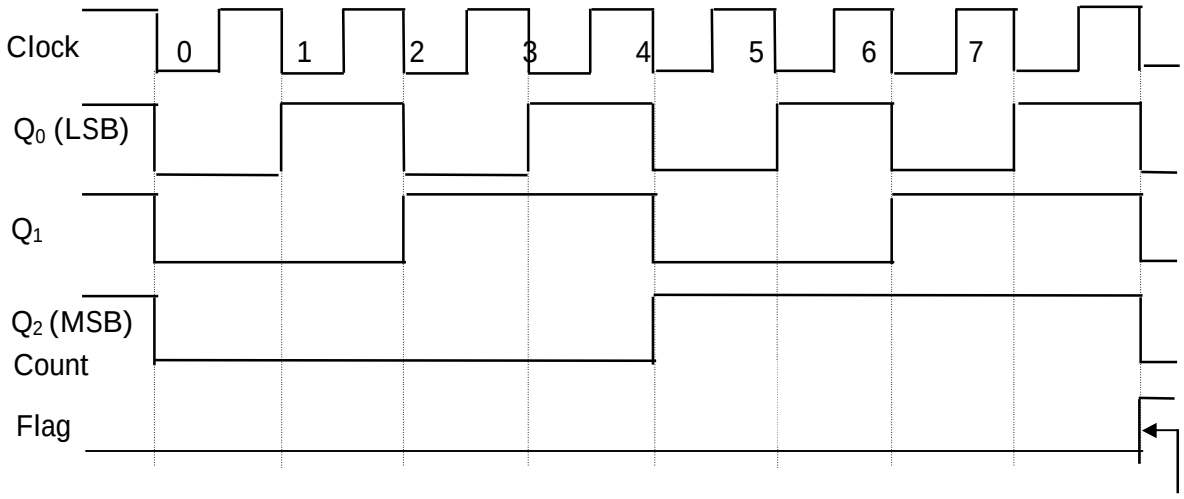
- Bộ định thời của Timer là một chuỗi các Flip Flop được chia làm 2, nó nhận tín hiệu vào là một nguồn xung clock, xung clock được đưa vào Flip Flop thứ nhất là xung clock của Flip Flop thứ hai mà nó cũng chia tần số clock này cho 2 và cứ tiếp tục.

- Vì mỗi tầng kế tiếp chia cho 2, nên Timer n tầng phải chia tần số clock ngõ vào cho 2^n. Ngõ ra của tầng cuối cùng là clock của Flip Flop tràn Timer hoặc cờ mà nó kiểm tra bởi phần mềm hoặc sinh ra ngắt. Giá trị nhị phân trong các FF của bộ Timer có thể được nghỉ như đếm xung clock hoặc các sự kiện quan trọng bởi vì Timer được khởi động. Ví dụ Timer 16 bit có thể đếm đến từ FFFFH sang 0000H.

- Hoạt động của Timer đơn giản 3 bit được minh họa như sau:



Hình 1.8 : **Timer Flip-Flops**



Hình 1.9 : Biểu Đồ Thời Gian

- Trong hình trên mỗi tầng là một FF loại D phủ định tác động cạnh xuống được hoạt động ở mode chia cho 2 (ngõ ra Q\ được nối vào D). FF cờ là một bộ chốt đơn giản loại D được set bởi tầng cuối cùng trong Timer. Trong biểu đồ thời gian, tầng đầu đổi trạng thái ở ½ tần số clock, tầng thứ hai đổi trạng thái ở tần số ¼ tần số clock … Số đếm được biết ở dạng thập phân và được kiểm tra lại dễ dàng bởi việc kiểm tra các tầng của 3 FF. Ví dụ số đếm "4" xuất hiện khi Q2=1, Q1=0, Q0=0 ($4_{10}=100_2$).

- Các Timer được ứng dụng thực tế cho các hoạt động định hướng. 8951 có 2 bộ Timer 16 bit, mỗi Timer có 4 mode hoạt động. Các Timer dùng để đếm giờ, đếm các sự kiện cần thiết và sự sinh ra tốc độ của tốc độ Baud bởi sự gắn liền Port nối tiếp.

- Mỗi sự định thời là một Timer 16 bit, do đó tầng cuối cùng là tầng thứ 16 sẽ chia tần số clock vào cho 2^{16} = 65.536.

- Trong các ứng dụng định thời, 1 Timer được lập trình để tràn ở một khoảng thời gian đều đặn và được set cờ tràn Timer. Cờ được dùng để đồng bộ chương trình để thực hiện một hoạt động như việc đưa tới 1 tầng các ngõ vào hoặc gởi dữ liệu đếm ngõ ra. Các ứng dụng khác có sử dụng việc ghi giờ đều đều của Timer để đo thời gian đã trôi qua hai trạng thái (ví dụ đo độ rộng xung).Việc đếm một sự kiện được dùng để xác định số lần xuất hiện của sự kiện đó, tức thời gian trôi qua giữa các sự kiện.

- Các Timer của 8951 được truy xuất bởi việc dùng 6 thanh ghi chức năng đặc biệt như sau:

Timer SFR	**Purpose**	**Address**	**Bit-Addressable**
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

2. Thanh ghi mode timer tmod **(TIMER MODE REGITER):**

- Than` ghi mode gồm hai nhóm 4 bit là: 4 bit thấp đặt mode hoạt động cho Timer 0 và 4 bit cao đặt mode hoạt động cho Timer 1. 8 bit của thanh ghi TMOD được tóm tắt như sau:

Bit	Name	Timer	Description
7	GATE	1	Khi GATE = 1, Timer chỉ làm việc khi INT1=1
6	C/T	1	Bit cho đếm sự kiện hay ghi giờ
			C/T = 1 : Đếm sự kiện
			C/T = 0 : Ghi giờ đều đặn
5	M1	1	Bit chọn mode của Timer 1
4	M0	1	Bit chọn mode của Timer 1
3	GATE	0	Bit cổng của Timer 0
2	C/T	0	Bit chọn Counter/Timer của Timer 0
1	M1	0	Bit chọn mode của Timer 0
0	M0	0	Bit chọn mode của Timer 0

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

M1	M0	MODE	DESCRIPTION
0	0	0	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 Timer tách ra : Timer 0 : TL0 là Timer 8 bit được điều khiển bởi các bit của Timer 0. TH0 tương tự nhưng được điều khiển bởi các bit của mode Timer 1. Timer 1 : Được ngừng lại.

- TMOD không có bit định vị, nó thường được LOAD một lần bởi phần mềm ở đầu chương trình để khởi động mode Timer. Sau đó sự định giờ có thể dừng lại, được khởi động lại như thế bởi sự truy xuất các thanh ghi chức năng đặc biệt của Timer khác.

3. *Thanh ghi điều khiển timer tcon* **(TIMER CONTROL REGISTER) :**

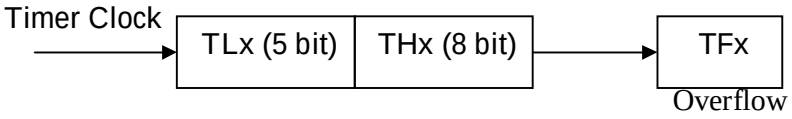
- Thanh ghi điều khiển bao gồm các bit trạng thái và các bit điều khiển bởi Timer 0 và Timer 1. Thanh ghi TCON có bit định vị. Hoạt động của từng bit được tóm tắt như sau:

Bit	Symbol	Bit Address	Description
TCON.7	TF1	8FH	Cờ tràn Timer 1 được set bởi phần cứng ở sự tràn, được xóabởi phần mềm hoặc bởi phần cứng khi các vectơxử lý đến thủ tục phục vụ ngắt ISR
TCON.6	TR1	8EH	Bit điều khiển chạy Timer 1 được set hoặc xóa bởi phần mềm để chạy hoặc ngưng chạy Timer.
TCON.5	TF0	8DH	Cờ tràn Timer 0(hoạt động tương tự TF1)
TCON.4	TR0	8CH	Bit điều khiển chạy Timer 0 (giống TR1)
TCON.3	IE1	8BH	Cờ kiểu ngắt 1 ngoài. Khi cạnh xuống xuất hiện trên INT1 thì IE1 được xóa bởi phần mềm hoặc phần cứng khi CPU định hướng đến thủ tục phục vụ ngắt ngoài.
TCON.2	IT1	8AH	Cờ kiểu ngắt 1 ngoài được set hoặc xóa bằng phần mềm bởi cạnh kích hoạt bởi sự ngắt nggài.
TCON.1	IE0	89H	Cờ cạnh ngắt 0 ngoài
TCON	IT0	88H	Cờ kiểu ngắt 0 ngoài.

4. Các mode và cờ tràn **(TIMER MODES AND OVERFLOW) :**

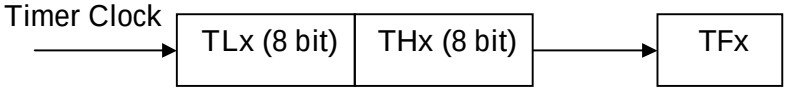
- 8951 có 2 Timer là Timer 0 và timer 1. Ta dùng ký hiệu TLx và Thx để chỉ 2 thanh ghi byte thấp và byte cao của Timer 0 hoặc Tmer 1.

4.1. Mode Timer 13 bit (MODE 0) :



- Mode 0 là mode Timer 13 bit, trong đó byte cao của Timer (Thx) được đặt thấp và 5 bit trọng số thấp nhất của byte thấp Timer (TLx) đặt cao để hợp thành Timer 13 bit. 3 bit cao của TLx không dùng.

4.2. Mode Timer 16 bit (MODE 1):



Overflow

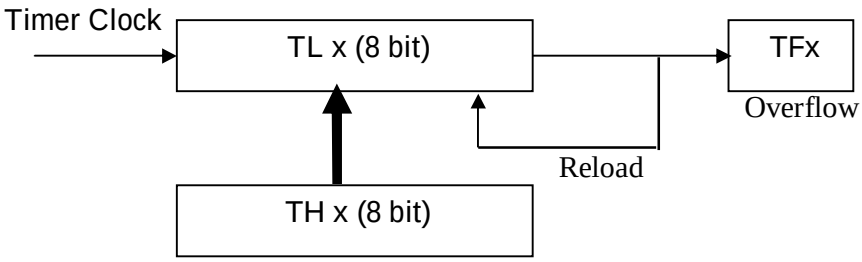
- Mode 1 là mode Timer 16 bit, tương tự như mode 0 ngoại trừ Timer này hoạt động như một Timer đầy đủ 16 bit, xung clock được dùng với sự kết hợp các thanh ghi cao và thấp (TLx, THx). Khi xung clock được nhận vào, bộ đếm Timer tăng lên 0000H, 0001H, 0002H, . . ., và một sự tràn sẽ xuất hiện khi có sự chuyển trên bộ đếm Timer từ FFFH sang 0000H và sẽ set cờ tràn Time, sau đó Timer đếm tiếp.

- Cờ tràn là bit TFx trong thanh ghi TCON mà nó sẽ được đọc hoặc ghi bởi phần mềm.

- Bit có trọng số lớn nhất (MSB) của giá trị trong thanh ghi Timer là bit 7 của THx và bit có trọng số thấp nhất (LSB) dà bit 0 của TLx. Bit LSB đổi trạng thái ở tần số clock vào được chia $2^{16} = 65.536$.

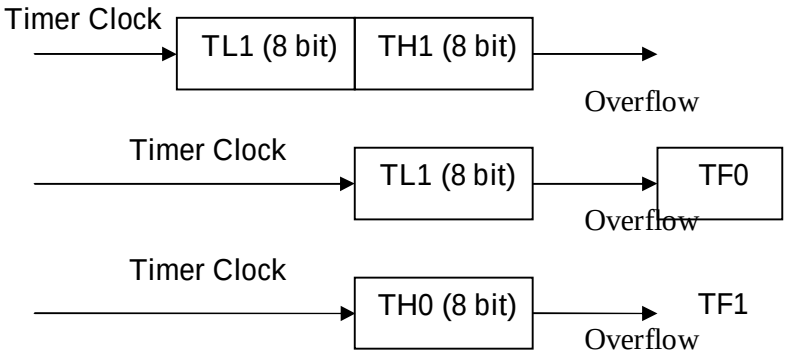
- Các thanh ghi Timer TLx và Thx có thể được đọc hoặc ghi tại bất kỳ thời điểm nào bởi phần mềm.

4..3. Mode tự động nạp 8 bit (MODE 2) :



- 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 giữ giá trị Reload. Khi bộ đếm tràn từ FFH sang 00H, không chỉ cờ tràn được set mà giá trị trong THx cũng được nạp vào TLx: Bộ đếm được tiếp tục từ giá trị này lên đến sự chuyển trạng thái từ FFH sang 00H kế tiếp và cứ thế tiếp tục. Mode này thì phù hợp bởi vì các sự tràn xuất hiện cụ thể mà mỗi lúc nghỉ thanh ghi TMOD và THx được khởi động.

4.4 Mode Timer tách ra (MODE 3):



- Mode 3 là mode Timer tách ra và là sự khác biệt cho mỗi Timer.

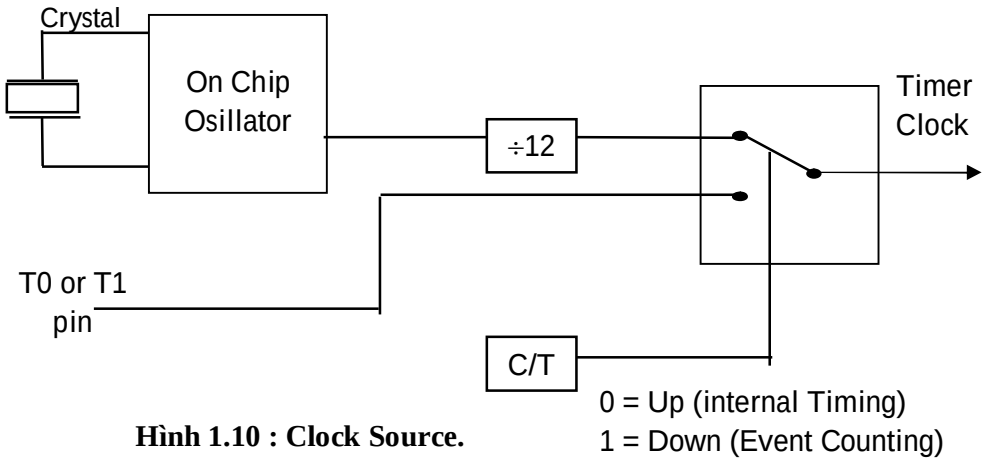
- Timer 0 ở mode 3 được chia là 2 timer 8 bit. TL0 và TH0 hoạt động như những Timer riêng lẻ với sự tràn sẽ set các bit TL0 và TF1 tương ứng.

- Timer 1 bị dừng lại ở mode 3, nhưng có thể được khởi động bởi việc ngắt nó vào một trong các mode khác. Chỉ có nhược điểm là cờ tràn TF1 của Timer 1 không bị ảnh hưởng bởi các sự tràn của Timer 1 bởi vì TF1 được nối với TH0.

- Mode 3 cung cấp 1 Timer ngoại 8 bit là Timer thứ ba của 8951. Khi vào Timer 0 ở mode 3, Timer có thể hoạt động hoặc tắt bởi sự ngắt nó ra ngoài và vào trong mode của chính nó hoặc có thể được dùng bởi Port nối tiếp như là một máy phát tốc độ Baud, hoặc nó có thể dùng trong hướng nào đó mà không sử dụng Interrupt.

5. Các nguồn xung clock **(CLOCK SOURCES):**

- Có hai nguồn xung clock có thể đếm giờ là sự định giờ bên trong và sự đếm sự kiện bên ngoài. Bit C/T trong TMOD cho phép chọn 1 trong 2 khi Timer được khởi động.



Hình 1.10 : Clock Source.

5.1 Sự bấm giờ bên trong (Interval Timing):

- Nếu bit C/T = 0 thì hoạt động của Timer liên tục được chọn vào bộ Timer được ghi giờ từ dao động trên Chip. Một bộ chia 12 được thêm vào để giảm tần số clock đến 1 giá trị

phù hợp với các ứng dụng. Các thanh ghi TLx và THx tăng ở tốc độ 1/12 lần tần số dao động trên Chip. Nếu dùng thạch anh 12MHz thì sẽ đưa đến tốc độ clock 1MHz.

- Các sự tràn Timer sinh ra sau một con số cố định của những xung clock, nó phụ thuộc vào giá trị khởi tạo được LOAD vào các thanh ghi THx và TLx.

5.2 Sự đếm các sự kiện (Event Counting):

- Nếu bit C/T = 1 thì bộ Timer được ghi giờ từ nguồn bên ngoài trong nhiều ứng dụng, nguồn bên ngoài này cung cấp 1 sự định giờ với 1 xung trên sự xảy ra của sự kiện. Sự định giờ là sự đếm sự kiện. Con số sự kiện được xác định trong phần mềm bởi việc đọc các thanh ghi Timer. Tlx/THx, bởi vì giá trị 16 bit trong các thanh này tăng lên cho mỗi sự kiện.

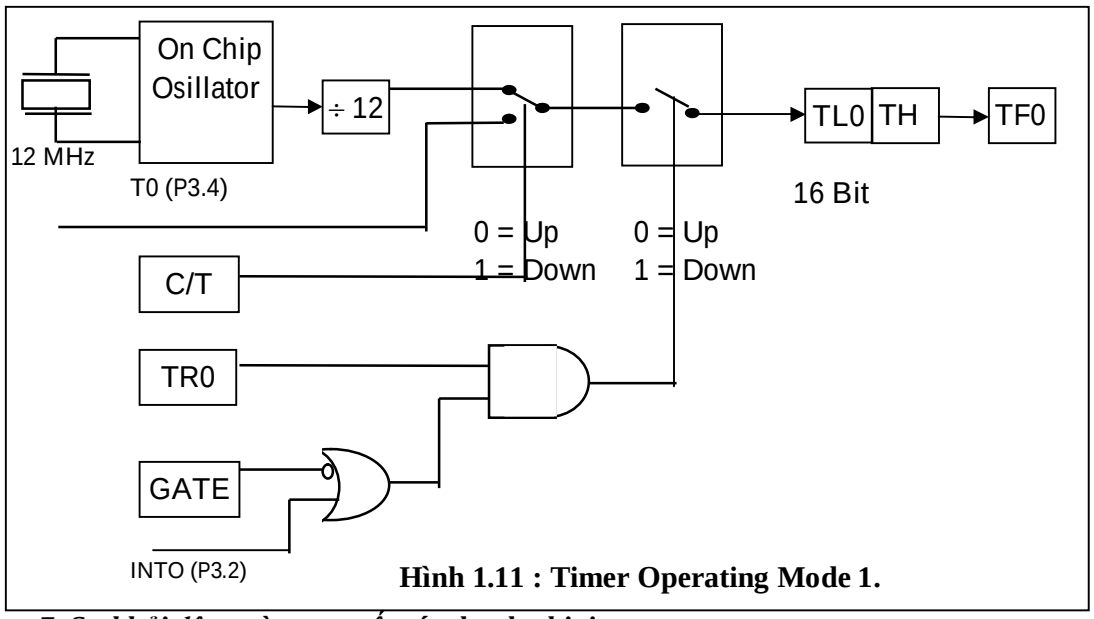
- Nguồn xung clock bên ngoài đưa vào chân P3.4 là ngõ nhập của xung clock bởi Timer 0 (T0) và P3.5 là ngõ nhập của xung clock bởi Timer 1 (T1).

- Trong các ứng dụng đếm các thanh ghi Timer được tăng trong đáp ứng của sự chuyển trạng thái từ 1 sang 0 ở ngõ nhập Tx. Ngõ nhập bên ngoài được thử trong suốt S5P2 của mọi chu kỳ máy: Do đó khi ngõ nhập đưa tới mức cao trong một chu kỳ và mức thấp trong một chu kỳ kế tiếp thì bộ đếm tăng lên một. Giá trị mới xuất hiện trong các thanh ghi Timer trong suốt S5P1 của chu kỳ theo sau một sự chuyển đổi. Bởi vì nó chiếm 2 chu kỳ máy (2μs) để nhận ra sự chuyển đổi từ 1 sang 0, nên tần số bên ngoài lớn nhất là 500KHz nếu dao động thạch anh 12 MHz.

6. *Sự bắt đầu, kết thúc và sự điều khiển các timer* **(STARTING, STOPPING AND CONTROLLING THE TIMER):**

- Bit TRx trong thanh ghi có bit định vị TCON được điều khiển bởi phần mềm để bắt đầu hoặc kết thúc các Timer. Để bắt đầu các Timer ta set bit TRx và để kết thúc Timer ta Clear TRx. Ví dụ Timer 0 được bắt đầu bởi lệnh SETB TR0 và được kết thúc bởi lệnh CLR TR0 (bit Gate= 0). Bit TRx bị xóa sau sự reset hệ thống, do đó các Timer bị cấm bằng sự mặc định.

- Thêm phương pháp nữa để điều khiển các Timer là dùng bit GATE trong thanh ghi TMOD và ngõ nhập bên ngoài INTx. Điều này được dùng để đo các độ rộng xung. Giả sử xung đưa vào chân INT0 ta k`ởi động Timer 0 cho mode 1 là mode Timer 16 bit với TL0/TH0 = 0000H, GATE = 1, TR0 = 1. Như vậy khi INT0 = 1 thì Timer "được mở cổng" và ghi giờ với tốc độ của tần số 1MHz. Khi INT0 xuống thấp thì Timer "đóng cổng" và khoảng thời gian của xung tính bằng μs là sự đếm được trong thanh ghi TL0/TH0.



Hình 1.11 : Timer Operating Mode 1.

***7. Sự khởi động và truy xuất các thanh ghi timer*:**

- Các Timer được khởi động 1 lần ở đầu chương trình để đặt mode hoạt động cho chúng. Sau đó trong chương trình các Timer được bắt đầu, được xóa, các thanh ghi Timer được đọc và cập nhật . . . theo yêu cầu của từng ứng dụng cụ thể.

- Mode Timer TMOD là thanh ghi đầu tiên được khởi gán, bởi vì đặt mode hoạt động cho các Timer. Ví dụ khởi động cho Timer 1 hoạt động ở mode 1 (mode Timer 16bit) và được ghi giờ bằng dao động trên Chip ta dùng lệnh: MOV TMOD,# 00001000B.
- Trong lệnh này M1 = 0, M0 = 1 để vào mode 1 và C/T 5 0, GATE=0 để cho phép ghi giờ bên trong đồng thời xóa các bit mode của Timer 0. Sau lệnh trên Timer vẫn chưa đếm giờ, nó chỉ bắt đầu đếm giờ khi set bit điều khiển chạy TR1 của nó.

- Nếu ta không khởi gán giá trị đầu cho các thanh ghi TLx/THx thì Timer sẽ bắt đầu đếm từ 0000H lên và khi tràn từ FFFFH sang 0000H nó sẽ bắt đầu tràn TFx rồi tiếp tục đếm từ 0000H lên tiếp . . .

- Nếu ta khởi gán giá trị đầu cho TLx/THx, thì Timer sẽ bắt đầu đếm từ giá trị khởi gán đó lên nhưng khi tràn từ FFFFH sang 0000H lại đếm từ 0000H lên.

- Chú ý rằng cờ tràn TFx tự động được set bởi phần cứng sau mỗi sự tràn và sẽ được xóa bởi phần mềm. Chính vì vậy ta có thể lập trình chờ sau mỗi lần tràn ta sẽ xóa cờ TFx và quay vòng lặp khởi gán cho TLx/THx để Timer luôn luôn bắt đầu đếm từ giá trị khởi gán lên theo ý ta mong muốn.

- Đặc biệt những sự khởi gán nhỏ hơn 256 µs, ta sẽ gọi mode Timer tự động nạp 8 bit của mode 2. Sau khi khởi gán giá trị đầu vào THx, khi set bit TRx thì Timer sẽ bắt đầu đếm giá trị khởi gán và khi tràn từ FFH sang 00H trong TLx, cờ TFx tự động được set đồng thời giá trị khởi gán mà ta khởi gađn cho Thx được nạp tự động vào TLx và Timer lại được đếm từ giá trị khởi gán này lên. Nói cách khác, sau mỗi tràn ta không cần khởi gán lại cho các thanh ghi Timer mà chúng vẫn đếm được lại từ giá trị ban đầu.

8. Sự đọc thanh ghi timer trên tuyến:

- Trong một số ứng dụng cần thiết đọc giá trị trong các thanh ghi Timer trên tuyến, có một vấn đề tiềm năng đơn giản để bảo vệ lại phần mềm. Bởi vì 2 thanh ghi Timer phải được đọc, nên "lỗi giai đoạn" có thể xuất hiện nếu byte tràn và byte cao giữa 2 hoạt động đọc. Một giải pháp để khắc phục là đọc byte cao trước, sau đó đọc byte thấp, và đọc lại byte cao: Nếu byte cao thay đổi thì lặp lại các hoạt động đọc.

CHƯƠNG II

GIAO TIẾP MÁY TÍNH

I.CÁC PHƯƠNG PHÁP ĐIỀU KHIỂN VÀO RA

1.Vào ra điều khiển bằng chương trình

Thiết bị ngoại vi được ghép tới Bus của hệ thống vi điều khiển thông qua các phần thích ứng về công nghệ chế tạo và logic. Thích ứng về công nghệ chế tạo là điều chỉnh công nghệ sản xuất thiết bị ngoại vi và công nghệ sản xuất của mạch trong hệ vi điều khiển. Thích ứng về logic là nhiệm vụ tạo tín hiệu điều khiển ngoại vi tử tín hiệu trêf Bus hệ thống.

Trong hệ vi điều khiển một vùng nhớ dùng làm nơi chứa địa chỉ cổng vào/ra và CPU xuất hoặc nhập dữ liệu từ các cổng vào/ra này bằng các lệnh xuất/nhập IN/OUT. Lúc này cổng vào ra được xem như một thanh ghi ngoài, chúng được viết vào hoặc đọc ra như ô nhớ RAM qua hai lệnh trên. Để phân biệt hướng xuất hoặc nhập dữ liệu từ cổng vào ra CPU phát ra tín hiệu điều khiển đọc hoặc viết. Dể phân biệt vùng nhớ với thiết bị vào ra CPU phát ra tín hiệu điều khiển IO/M. Khi có các lệnh này thì lệnh IN/OUT mới có tác dụng.

Ngoài ra các lệnh qui chế độ nhớ, cũng như khả năng trao đổi dữ liệu giữa thiết bị ngoài và hệ vi điều khiển. Lúc đó cổng vào ra được gán như địa chỉ ô nhớ của bộ nhớ. Các thanh ghi liên quan đến cổng vào ra được xem như ngăn nhớ. Khi bộ vi điều khiển gọi địa chỉ và xung điều khiển đọc hay viết bộ nhớ, nó không cần xác định nơi gởi là bộ nhớ hay thaết bị vào ra. Nó chỉ đòi hỏi nơi gởi dữ liệu vào trong khoảng thời gian cho phép. Bộ logic bên ngoài sẽ giải mã địa chỉ kết hợp với xung MR, MW để chọn thiết bị mà không phân biệt ngăn nhớ hay thiết bị vào ra.

2.Vào ra điều khiển bằng ngắt:

Với phương pháp điều khiển vào ra bằng chương trình CPU phải liên tục kiểm tra trạng thái của thiết bị ngoại vi đến khi sẵn sàng. Đó là sự lãng phí thời gian của CPU và làm cho chương trình dài và phức tạp. Khi bộ vi điều khiển có nhiều thiết bị ngoại vi CPU không đáp ứng được nhu cầu của chúng. Có thể đáp ứng được yêu cầu ngoại vi nhanh chóng và không teo trình tự như định trước nhờ cơ cấu ngắt của CPU.

Nhờ tính chất đáp ứng ngắt tức thời của các vi điều khiển, khi có yêu cầu ngắt từ thiết bị ngoại vi. Do đó các ngắt thường được dùng ở những trường hợp yêu cầu đáp ứng nhanh, thời gian trả lời ngắn thực hiện ở bất cứ thời điểm nào. Khi đó CPU phải chuyển đến chương trình con phục vụ yêu cầu ngắt ở cuối bất kỳ lệnh nào trong chương trình chính. Các chương trình con phục vụ ngắt có thể lưu giữ nội dung các thanh ghi và khôi phục lại khi thực hiện xong chương trình phục vụ ngắt và trước khi trở lại chương trình chính.

Phần cứng này trở nên phức tạp do phải giải quyết ưu tiên về vấn đề mã hóa. Khi sử dụng hệ ngắt ta xét đến các vấn đề sau:

Yêu cầu ngắt: Kiểm tra ngắt ở cuối chu kỳ lệnh, một chu kỳ lệnh có thể kéo dài nên tin hiệu yêu cầu ngắt thường được chốt lại cho đến khi bộ vi điều khiển ghi nhận. Ta có thể dùng đầu vào ngắt của vi điều khiển cho nhiều thiết bị ngoại vi nhưng CPU không xác định được nguồn yêu cầu ngắt.

Chuyển điều kiện ngắt đến phục vụ ngắt : tùy vào chế độ ngắt, loại đầu vào ngắt, dữ liệu vào và họ vi điều khiển được chuyển sang chương trình phục vụ ngắt theo mỗi cách khác nhau.

Cất giữ khôi phục trạng thái: Tất cả các vi điều khiển khi thực hiện ngắt phải tự động ngắt cất giữ một trạng thái điều khiển, nó là nội dung thanh ghi bộ đếm chương trình PC. Phần còn lại của trạng thái được cất giữ tùy theo yêu cầu cụ thể của phần mềm.

*Có hai phương pháp cất giữ trạng thái:

+ Dùng ngăn xếp: Địa chỉ ngăn xếp được điều chỉnh và nhớ ở thanh ghi con trỏ ngăn xếp.

+ Trao đổi giữa hai thanh ghi: chương trình con phục vụ ngắt được gọi đến bằng một lệnh. Khi đó CPU chuyển sang làm việc ở thanh ghi thứ hai.

Xác định nguồn ngắt:

+ Phương pháp hỏi vòng: Thứ tự chính là thứ tự ưu tiên. Bộ vi điều khiển hỏi vòng khi chắc chắn có một thiết bị yêu cầu ngắt.

+ Phương pháp tạo vectơ hỏi vòng: tất cả các nguồn ngắt được mã hóa để tạo vectơ ngắt. Vectơ ngắt được hiểu như một phần của địa chỉ, phân nhánh đến chương trình con phục vụ ngắt hoặc một lệnh mà vi điều khiển phải thực hiện trong chu kỳ lệnh kế tiếp. Vectơ ngắt được đưa vào Bus dữ liệu bằng lệnh điều khiển của CPU.

Ưu tiên: vấn đề ưu tiên trong thời điểm có nhiều yêu cầu ngắt, để dữ liệu vào ra của hệ thống không bị xáo trộn, thường mỗi thiết bi ngoại vi được gán một mức ưu tiên cố định. CPU phục vụ theo mức ưu tiên giảm dần. Trong mạch điều khiển ngắt, mức ưu tiên ngắt có tám mức được mã hóa từ 3 bit. Mã đó được so sánh với nội dung ưu tiên mà người xử dụng nạp vào chương trình. Nếu mức ưu tiên cao hơn mức qui định thì yêu cầu ngắt được cho phép CPU ngắt.

Cho phép ngắt và cấm ngắt: có thể điều khiển các ngắt vi điều khiển bằng phần mềm. Nghĩa là ta có thể thực hiện cho phép ngắt hặc cấm ngắt vi điều khiển bằng cách thiết lập cơ điều khiển bằng phần mềm.

Bộ vi điều khiển tự động cấm ngắt trong các trường hợp sau:

+Khởi động hệ thống.

+Sau khi ngắt.

3.Vào ra điều khiển bằng thâm nhập trực tiếp (DMA).

Thâm nhập trực tiếp là phương pháp vào ra dữ liệu nhanh nhất bằng phần cứng. Khi chuển dữ liệu CPU không cần phải đọc, giải mã và thực hiện các lệnh chuyển dữ liệu mà nó chuyển quyền điều khiển các Bus cho DMAC. DMAC tạo địa chỉ các tín hiệu cần đọc.

Để thực hiện một phép chuyển đổi DMAC mỗi DMAC phải giải quyết các vấn đề sau.

+ Thông báo CPU yêu cầu thực hiện DMAC.

+ Điều khiển Bus và không gây ảnh hưởng đến hoạt động bình thường của CPU và không gây xung đột ở Bus.

+ Xác định địa chỉ số từ đã chuyển đổi

Để thực hiện việc chuyển đổi bằng cách thâm nhập trực tiếp, phương pháp thông dụng nhất là bắt CPU tự treo thay vì yêu cầu ngắt CPU. Thiết bị yêu cầu đến DMAC khi nhận được tín hiệu yêu cầu ngắt, DMAC tạo tín hiệu HOLD đến CPU. CPU dựa vào tín hiệu HOLD khi thực hiện chu kỳ cuối cùng của lệnh hiện tại sau đó tự treo và thông báo đến DMAC bằng tín hiệu HALT. Khi nhận được tín hiệu trả lời HALT của CPU, DMAC lấy quyền điều khiển Bus, tạo địa chỉ, ghi nhận số liệu của thiết bị ngoại vi bằng DACK. Số liệu được chuyển trực tiếp giữa thiết bị ngoại vi và bộ nhớ.

II.SƠ LƯỢC VỀ CÁCH GIAO TIẾP GIỮA MÁY TÍNH VÀ THẾT BỊ NGOẠI VI

Có ba cách giao tiếp giữa máy tính và thiết bị ngoại vi. Tuỳ theo trường hợp ứng dụng cụ thể mà chọn cách giao tiếp thích hợp.

1.Giao tiếp bằng SLOT-CARD.

Trong máy tính trên board mạch hệ thống thường chế tạo sẵn các Slot chằm mục đích mở rộng bộ nhớ, cũng như mở rộng phạm vi ứng dụng của máy vi tính bằng cách gắn thêm trên các board mở rộng vào các Slot này.

Mỗi Sdot đều có các Bus dữ liệu, Bus địa chỉ và các đường tín hiệu điều khiển như: CLK, IOW, IOR,… Do đó việc thiết kế các SLOT-CARD từ các đầu cấm Slot sẽ đơn giản số linh kiện ít và tận dụng được các nguồn điện của máy vi tính (±5V, ±12V) nên giá thành rẻ, dễ dàng đưa tín hiệu điều khiển ra ngoài và tốc độ truyền nhanh.

Bên cạnh những ưu điểm nó có một số nhược điểm sau:

+ SLOT-CARD phải cắm vào các Slot trên Board mạch hệ thống nên phải gỡ nắp máy ra.

+ Phạm vi truyền tín hiệu gần và các dạng phức tạp. Trong một số trường hợp không thực hiện được.

Vì vậy khi sử dụng SLOT-CARD để giao tiếp với thiết bị ngoại vi phải cân nhắc kỹ gưa ưu và khuyết điểm. Tuỳ theo mục đích sử dụng mà chọn cách thích hợp.

2.Giao tiếp bằng cổng máy in (Giao tiếp song song)

Port giao tiếp máy in dùng để giao tiếp với máy in. Trong cách giao tiếp này dữ liệu đợc truyền song song gồm 8 Bit và một số tín hiệu bắt tay. Đầu nối (conecter) gồm 25 chân trong đó có 8 chân được nối với 8 đường dữ liệu, một số chân còn lại được nối với tín hiệu bắt tay (Hand-Shaking). Tất cả các đường dữ liệu và tín hiệu điều khiển đều ở mức logic tương thích với mức TTL. Hơn nữa người lập trình có thể cho phép hay không cho phép sử dụng các ngắt ở ngõ vào, nên việc giao tiếp được dễ dàng. Tuy nhiên với mức logic TTL thì không thể truyền đi xa dược mà chỉ truyền dược khoảng ngắn, cáp truyền cũng phức tạp hơn cổng COM. Đây là nhược điểm của cổng máy in.

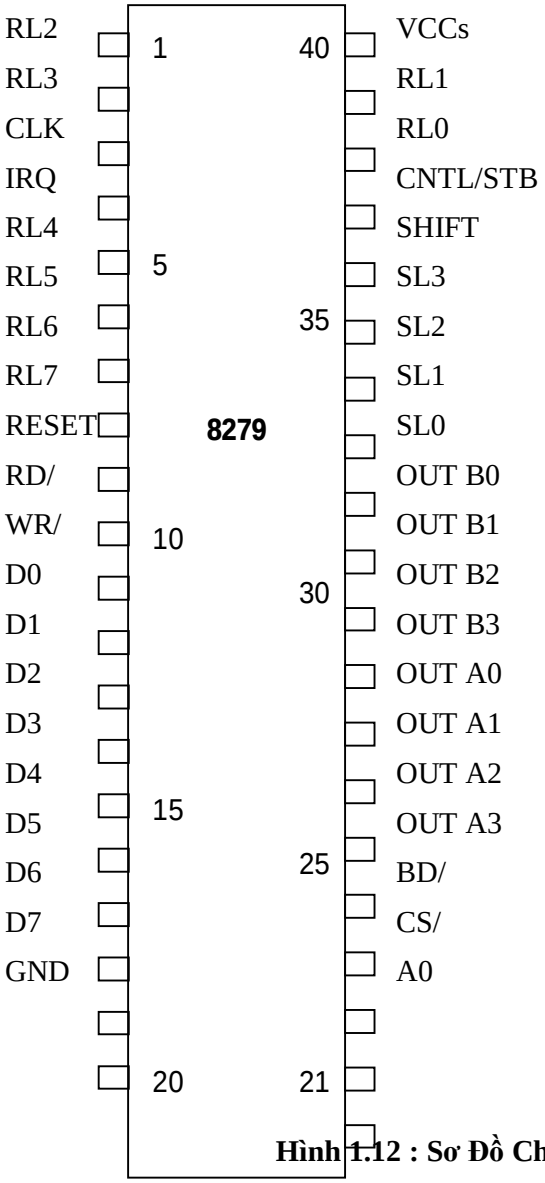
3.Giao tiếp bằng cổng COM (Giao tiếp nối tiếp)

Khác với cổng máy in, cổng COM là cổng truyền dữ liệu nối tiếp. Nó thường được dùng để giao tiếp với thiết bị ngoại vi có tốc độ xử lý dữ liệu chậm như: chuột hoặc moderm … cổng này giao tiếp theo tiêu chuẩn RS232.

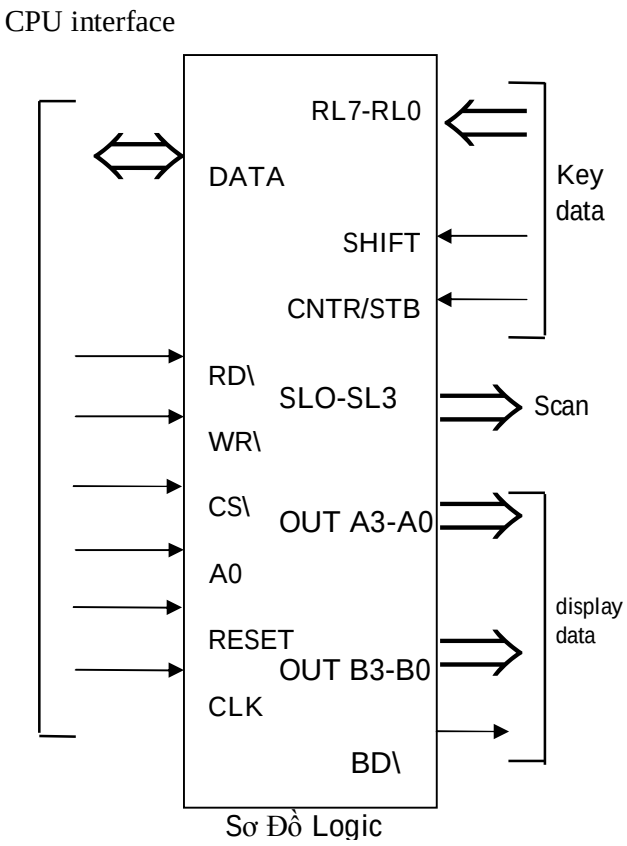
Dữ liệu được truyền dưới dạng nối tiếp theo từng Bit một. Tốc độ truyền Bit do người lập trình quyết định (thường là 1200 bps, 2400bps, 4800bps, 9600bps, …) chiều dài dữ liệu có thể là 5, 6, 7, hoặc 8 Bit kèm theo các Bit Start, Stop, parity tạo thành một khung gọi la Frame. Cổng này gồ các đường phát, đường thu và đường mass chung. Vì giao tiếp với tiêu chuẩn RS232 nên khoảng truyền xa hơn so với các truyền song song như cổng máy in nhưng nó có tốc độ truyền rất chậm.

CHƯƠNG III
KHẢO SÁT VI MẠCH 8279 QUÉT BÀN PHÍM VÀ HIỂN THỊ

I. CẤU TRÚC IC 8279



Sơ Đồ Chn



Sơ Đồ Logic

Hình 1.12 : Sơ Đồ Chân Và Sơ Đồ Logic

Tên các chân 8279:

Tên	I/O	Chức năng
DB_7 - DB_0	I/O	Data bus (Bi-Direction)
CLK	I	Clock input
RESET	I	Reset input
CS\	I	Chip select
RD\	I	Read input
WR\	I	Write input
A_0	I	Address
IRQ	0	Interrupt Request input
SL_0 - SL_3	0	Scan Lines
RL_0 - RL_7	I	Return Lines
SHIFT	I	Shift input
CNTL/STB	I	Control/Strobe input
OUT A_{3-0}	0	Display (A) output
OUT B_{3-0}	0	Display (B) output
BD\	0	Blank Display Output

- 8279 kết nối với vi điều khiển thông qua 3 bus gồm bus dữ liệu D7-D0, bus địa chỉ có một đường A0, bus điều khiển RD\, WR\, CS\, Resert, CLK.

- Tín hiệu chọn CS\ được kết nối đến một ngõ ra của IC giải mã địa chỉ. Nếu xem bộ nhớ thì bộ nhớ này có 2 ô nhớ.

II. CẤU TRÚC PHẦN MỀM CỦA 8279

- IC 8279 có 1 đường địa chỉ Ao có chức năng lựa chọn như sau:
  - Ao = O_2 : 8279 xem dữ liệu từ vi điều khiển gởi đến là dữ liệu để hiển thị.
  - Ao =1_2 : 8279 xem dữ liệu từ vi điều khiển gởi đến là dữ liệu của lệnh điều khiển 8279.
- Các lệnh điều khiển của 8279:

1. Keyboard/ Display Mode Set:

+ Mã:

O	O	O	D	D	K	K	K

+ Trong đó 2 bit DD dùng để thiết lập mode hiển thị, 3 bit KKK dùng để thiết lập mode quét bàn phím.

+ Hai bit DD:

DD = 00 : hiển thị 8 ký tự - lối vào trái.

DD = 01 : hiển thị 16 ký tự - lối vào trái.

DD = 10 : hiển thị 8 ký tự - lối vào phải.

DD = 11 : hiển thị 16 ký tự - lối vào phải.

+ Ba bit KKK:

000 encode scan keyboard - 2 key lockout.

001 decode scan keyboard - 2 key lockout.

010 encode scan keyboard - N key rollover.

011 decode scan keyboard - N key rollover.

100 encode scan sensor matrix.

101 decode scan sensor matrix.

110 strobe input, encode display scan.

111 strobe input, decode display scan &

2. Program Clock:

+ Mã:

O	O	1	P	P	P	P	P

+ Lệnh này có chức năng chia tần số xung clock ở ngõ vào clk ở chân số 3, các bit PPPPP dùng để xác định số chia nằm trong khoảng từ 2 đến 30, tần số hoạt động của mạch quét hiển thị và chóng dội của 8279 thường là 100 Khz, nếu tần số ở ngõ vào là 2Mhz thì phải chia cho 20 để được 100 Khz, khi đó các bit PPPPP có giá trị là 10100.

3. Read FIFO/ sensor RAM:

+Mã

O	1	1	AI	X	A	A	A

8279 có 8 byte RAM bên trong để chứa mã của phím ấn hay mã của các sensor, để truy xuất đến từng byte dữ liệu mã của phím ấn hay của sensor ta có thể điều chỉnh các bit AAA tương ứng. Bộ nhớ này thuộc kiểu FIFO.

+ AI (automatically increment): ở mức một có chức năng làm con trỏ tự động tăng lên byte kế để sẵn sàng cho việc đọc dữ liệu. Nếu AI= 0 con trỏ sẽ không thay đổi.

4. Read Display RAM:

+ Mã:

O	1	1	Al	A	A	A	A

+ 8279 có 16 byte RAM bên trong do con trỏ 4 bit AAAA quản lý, 16 byte RAM này dùng để chứa dữ liệu cần hiển thị do vi điều khiển gởi đến, để đọc dữ liệu ô nhớ nào trong vùng nhớ RAM này ta có thể điều chỉnh các bit AAA tương ứng. Bộ nhớ hiển thị này thuộc kiểu FIFO.

+ AI (automatically increment): ở mức một chức năng làm con trỏ tự động tăng lên byte kế để sẵn sàng cho việc đọc byte dữ liệu. Nếu AI=0 con trỏ sẽ không thay đổi.

5. End Interrupt:

+ Mã:

1	1	1	E	0	0	0	0

+ Bit E bằng 1 sẽ xóa ngắt IRQ về mức logic 0.

6. Lệnh Write Display Ram:

+ Mã:

1	0	0	Al	A	A	A	A

+ 8279 có 16 byte RAM bên trong con trỏ 4 bit AAAA quản lý, 16 ô nhớ RAM này dùng để chứa dữ liệu cần hiển thị do vi điều khiển gởi đến, để gởi dữ liệu đến 8279 tại byte Ram thứ mấy trong 16 byte RAM ta có thể điều chỉnh các bit AAAA tương ứng.

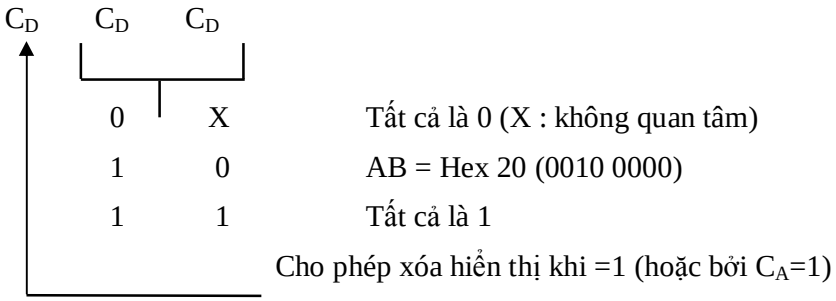
+ AI (automatically increment): ở mức một chức năng làm con trỏ tự động tăng lên byte kế để sẵn sàng nhận dữ liệu. Nếu AI=0 con trỏ sẽ không thay đổi do đó byte dữ liệu sau sẽ ghi đè lên byte dữ liệu trước đó.

7. Lệnh Clear:

+ Mã

1	1	0	C_D	C_D	C_D	C_F	C_A

+ Những bit C_D trong lệnh này dùng để xóa tất cả các hàng của Ram hiển thị đến một mã xóa được chọn dọc như sau:



+ Trong suốt thời gian Ram hiển thị đang xóa (≈ 160 μs) nó klhông được viết vào, bit lớn nhất của từ trạng thái FIFO được đặt lên 1 trong suốt thời gian này. Khi Ram hiển thị được sử dụng lại thì bit này được reset về 0.

+ Nếu như bit C_F tích cực (C_F =1), từ trạng thái FIFO sẽ bị xóa và ngõ ra Interrup bị reset.

+ Bit C_A có chức năng xóa tất cả các bit, nó còn ảnh hưởng bởi bit C_D và C_F. Nó dùng bit C_D để xóa mã trên Ram hiển thị và nó cũng xóa luôn trạng thái FIFO.

8279 là IC chuyên về giải mã hiển thị LED đoạn và quét theo nhiều phương pháp khác nhau .

Dữ liệu hiển thị từ vi xử lý gởi đến sẽ chứa trong 16 Byte RAM bên trong được gọi là bộ nhớ hiển thị. Các dữ liệu này lần lược được gởi ra 8279 đường tín hiệu từ A_{3-0} ,B_{3-0}.

Các đường tín hiệu SL_{3-0} dùng để quét, để dữ liệu trên các đường này có thể được thiết lập theo hai kiểu Encode và Decode tùy thuộc vào kiểu thiết kế phần cứng. Các đường này có hai chức năng vừa quet hiển thị vừa quet giải mã bàn phím.

Các đường tín hiệu RL_{7-0} là các đường tín hiệu Input kết hợ p với các đường tín hiệu quét SL_{3-0} tạo thành ma trận phím , phím được ấn sẽ làm cho một hoặc nhiều ngõ vào RL xuống 0, kết hợp với các đường tín hiệu quét sẽ cho mã của phím được ấn. Chú ý các đường SL_{3-0} ở chế độ Decode.

Các ngõ vào Shift và CNTL được dùng để mở rộng các phím tổ hợp.

Số lượng phím có thể lên đến 64 phím rời.

8279 gởi dữ liệu trên vùng nhớ RAM hiển thị ra Led 7 đoạn và tự động quét bàn phím để tìm phím bị tác động và tự động chống dội sau khoảng 10.3 ms và kiểm tra lại một lần nữa để xem phím đó còn bị ấn nữa hay không, nếu còn thì 8259 sẽ thiết lập mã cho phím ấn này và lưu trữ mã của phím vào bộ nhớ RAM bên trong. Sau đó sẽ báo cho CPU biết đã có một phím tác động và yêu cầu CPU hãy nhận mã của phím này bằng cách tác động đến tín hiệu ngắt IRQ .Tn hiệu IRQ được kết nối đến một ngõ vào ngắt của vi điều khiển và chương trình phục vụ cho ngắt này là chương trình xử lý phím. Nhiệm vụ của vi điều khiển là đọc mã của phím biấn vào để xử lý và Reset ngắt của 8279 trơi về mức logic 0 chuẩn bị cho phím tiếp theo.

Khung dữ liệu của phím bị ấn như sau:

CNTL	CNTL		SCAN			RETURN	

CHƯƠNG IV

VI MẠCH GIAO TIẾP NGOẠI VI 8255A

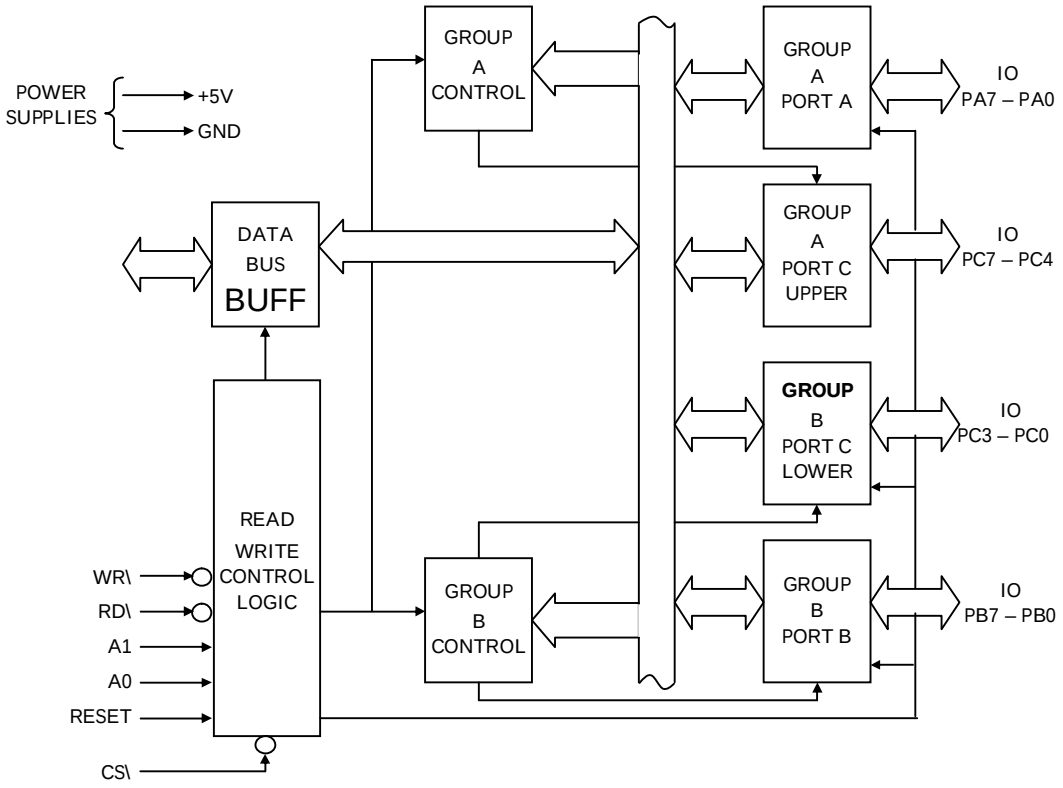
___ oOo ___

I. CẤU TRÚC PHẦN CỨNG 8255A:

8255A là IC ngoại vi được chế tạo theo công nghệ LSI dùng để giao tiếp song song giữa vi xử lý và thiết bị bên ngoài.

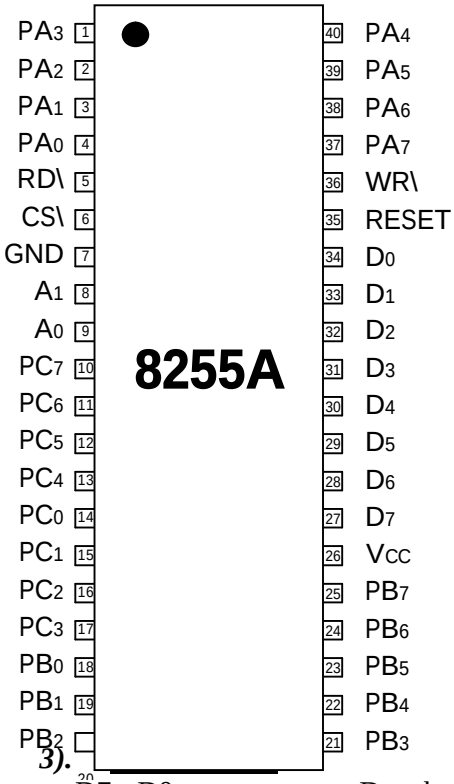
Mạch 8255A thường được gọi là mạch phối ghép vào/ra lập trình được (Programmable Peripheral Interface – PPI). Do khả năng mềm dẻo trong trong các ứng dụng thực tế nó là mạch phối ghép được dùng rất phổ biến cho các hệ vi xử lý 8 bit – 16 bit.

1). Sơ đồ khối của 8255A:

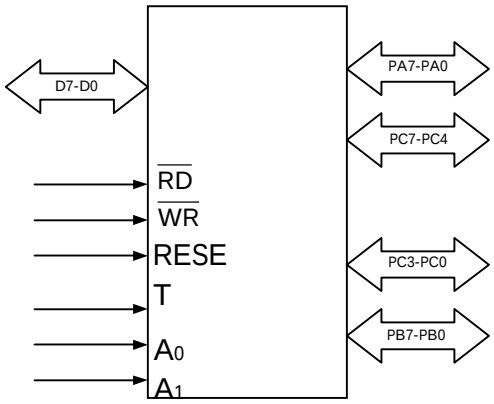


Hình 1.13: Sơ □ồ khối 8255A

2). ***Sơ đồ chân và sơ đồ Logic:***



Hình 1.14 : Sơ đồ chân 8255A



Hình 1.15 : Sơ đồ Logic

3).

D7 - D0	Data bus (Bi-Direction)
RESET	Reset input (nối với tín hiệu Reset toàn bộ hệ)
CS\	Chip Select
WR\	Write input
RD\	Read input
A0, A1	Port Address
PA7 – PA0	Port A
PB7 – PB0	Port B
PC7 – PC0	Port C

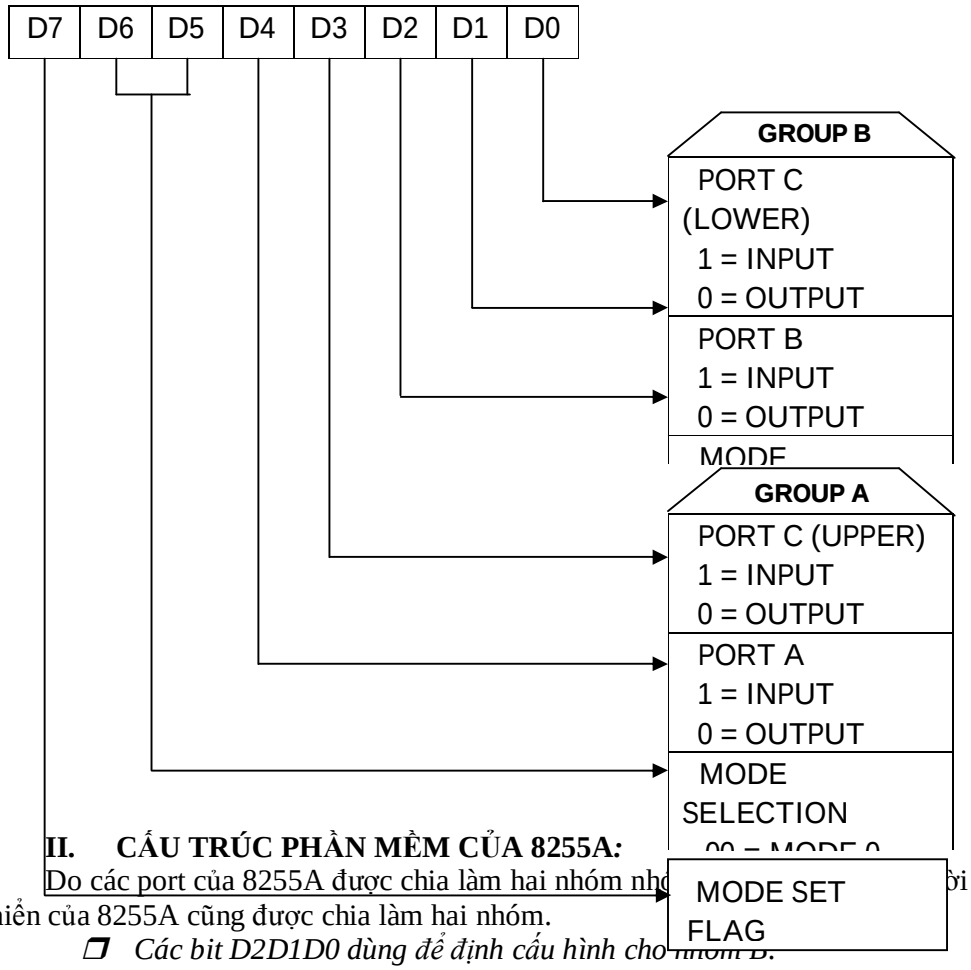
8255A giao tiếp với vi xử lý thông qua 3 bus: bus dữ liệu 8 bit D_7-D_0, bus địa chỉ, bus điều khiển RD\; WR\; CS\; Reset.

- Mã lệnh, thông qua trạng thái và và dữ liệu đều được truyền trên 8 đường dữ liệu D7-D0. Vi xử lý gởi dữ liệu đến 8255A hoặc vi xử lý đọc dữ liệu từ 8255A tùy thuộc vào lệnh điều khiển. Các đường tín hiệu RD\, WR\ của 8255A được kết nối với các đường RD\, WR\ của vi xử lý.
- Tín hiệu Reset dùng để khởi động 8255A khi cấp điện, khi bị Reset các thanh ghi các thanh ghi bên trong 8255A đều bị xóa và 8255A ở trạng thái sẵn sàng làm việc. Khi giao tiếp với vi xử lý ngỏ vào tín hiệu Reset này được kết nối với tín hiệu Reset Out của vi xử lý.
- Tín hiệu Chip Select (CS\) dùng để lựa chọn 8255A khi vi xử lý giao tiếp với nhiều 8255A.

8255A có 3 port xuất nhập (I/O) có tên port A, port B, port C, mỗi port 8 bit. Port A gồm các bit PA_0-PA_7, port B gồm các bit PB_0-PB_7 và port C gồm PC_0-PC_7. Các port này có thể là các port input hoặc output tùy thuộc vào lệnh điều khiển, lệnh điều khiển do vi xử lý gởi tới chứa trong thanh ghi (còn gọi là thanh ghi điều khiển) để điều khiển 8255A.

Các địa chỉ A_1-A_0 của 8255A dùng để lựa chọn các port và thanh ghi, A_1A_0=00_2 dùng để chọn Port A, A_1A_0 = 01_2 dùng để chọn Port B, A_1A_0 = 10_2 dùng để chọn Port C, A_1A_0 = 11_2 dùng để chọn thanh ghi điều khiển. Trong sơ đồ khối 8255A, các port I/O chia làm hai nhóm: nhóm A gồm port A và 4 bit cao của port C, nhóm B gồm port B và 4 bit thấp của port C. Để sử dụng các port của 8255A người lập trình phải gởi từ điều khiển ra thanh ghi điều khiển để 8255A định cấu hình cho các port đúng theo yêu cầu mà người lập trình mong muốn.

*4). **Cấu trúc từ điều khiển của 8255A**:*



II. CẤU TRÚC PHẦN MỀM CỦA 8255A:

Do các port của 8255A được chia làm hai nhóm nh[illegible]ời nên từ điều khiển của 8255A cũng được chia làm hai nhóm.

❒ *Các bit D2D1D0 dùng để định cấu hình cho nhóm B:*

♦ Bit D0 dùng để thiết lập 4 bit thấp của port C, D0 = 0 port C thấp là port xuất dữ liệu (output), D0 = 1 port C thấp là port nhập dữ liệu (input).

♦ Bit D1 dùng thiết lập port B, D1 = 0 port B là port xuất dữ liệu (output), D1 = 1 port B là port nhập dữ liệu (input).

♦ Bit D2 dùng thiết lập Mode điều khiển của nhóm B:

- D2 = 0: nhóm B hoạt động ở Mode 0.
- D2 =1: nhóm B hoạt động ở Mode 1.

❒ *Các bit D6, D5, D4, D3 dùng để định cấu hình cho nhóm A:*

♦ Bit D3 dùng để thiết lập 4 bit cao của port C, D3 = 0 port C là port xuất dữ liệu (output), D3 = 1 port C là port nhập dữ liệu (input).

♦ Bit D4 dùng để thiết lập port A, D4 = 0 port A là port xuất dữ liệu (output), D4 = 1 port A là port nhập dữ liệu (input).

♦ Bit D6D5 dùng thiết lập Mode điều khiển của nhóm A:

- D6D5 = 00: nhóm A hoạt động ở Mode 0.
- D6D5 = 01: nhóm A hoạt động ở Mode 1.

- D6D5 = 1X: nhóm A hoạt động ở Mode 2.

*1). **Các nhóm A, B được cấu hình ở Mode 0**:*

Từ điều khiển nhóm A & B hoạt động ở Mode 0:

1	0	0	D4	D3	0	D1	D0

Ở Mode 0 các port A, port B, port C thấp và port C cao là các port xuất hoặc nhập dữ liệu độc lập. Do có 4 bit để lựa chọn nên có 16 từ điều khiển khác nhau cho 16 trạng thái xuất nhập của 4 port.

2). Các nhóm A & B được cấu thành ở Mode 1:

Từ điều khiển nhóm A B hoạt động ở Mode 1:

1	0	1	D4	D3	1	D1	D0

Ở Mode 1 các port A & B làm việc xuất nhập có chốt (Strobe I/O). Ở Mode này hai port A & B hoạt động độc lập với nhau và mỗi port có 1 port 4 bit điều khiển/dữ liệu. Các port 4 bit điều khiển/dữ liệu được hình thành từ 4 bit thấp và 4 bit cao của port C.

Khi 8255A được cấu hình ở Mode 1, thiết bị giao tiếo muốn 8255A nhận dữ liệu, thiết bị đó phải tạo ra tín hiệu yêu cầu 8255A nhận dữ liệu, ngược lại 8255A muốn gởi tín hiệu đến thiết bị khác, 8255A phải tạo ra tín hiệu yêu cầu thiết bị đó nhận dữ liệu, tín hiệu yêu cầu đó gọi là tín hiệu Strobe.

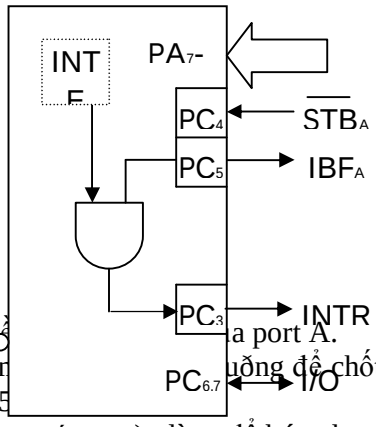
❒ *Nhóm A làm việc ở cấu hình Mode 1:*

♦ Port A được cấu hình là port nhập dữ liệu.

Chức năng của các đường tín hiệu được trình bày ở hình vẽ sau đây:

Từ điều khiển :

1	0	1	1	D3	X	X	X



Hình 1.16 : Mode 1 Port A

Các đường tín hiệu của port C trở thành các đường điề a port A.

Bit PC_4 trở thành bit STBA (Strobe Input, tác động r ưởng để chốt dữ liệu ở các ngỏ vào $PA_7 – PA_0$ vào mạch chốt bên trong 8255

Bit PC_5 trở thành bit IBTA (Input Bufer Full, tác động mức cao), dùng để báo cho thiết bị bên ngoài biết dữ liệu đã được chốt bên trong.

Bit PC_3 trở thành bit INTRA (Interrupt Request, tác động ở mức cao), bit này có mức Logic 1 khi hai bit STBa = 1, IBF = 1 và bit INTEa (Interrupt Enable) ở bên trong 8255A bằng 1. Bit $INTE_a$ được thiết lập mức Logic 1 hay 0 dưới sự điều khiển của phần mềm dùng cấu trúc Set/Reset của 8255A. Ở hình vẽ trên, bit INTEa = 1 dùng để cho phép tín hiệu IBF xuất hiện tại ngõ ra INTRA của cổng AND. Tín hiệu INTA tác động đến ngõ vào của ngắt vi xử lý để bao cho vi xử lý biết: dữ liệu mới đã xuất hiện ở port A chương trình phục vụ ngắt sẽ đọc dữ liệu vào xóa yêu cầu ngắt.

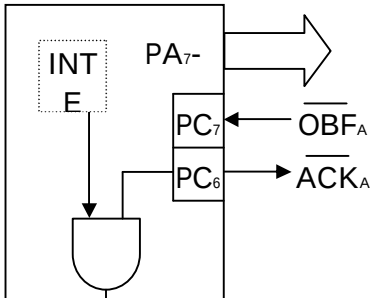
Các bit còn lại của port C: PC6, PC7 là các bit xuất/nhập bình thường tùy thuộc vào bit D3 trong từ điều khiển hình trên. Các bit XXX được dùng để thiết lập cho nhóm B.

♦ Port A được cấu hình là port xuất dữ liệu:

Chức năng của đường tín hiệu được trình bày ở hình sau:

Từ điều khiển:

1	0	1	0	D3	X	X	X



Hình 1.17 : Port A Xuất

Bit PC_7 trở thàfh bit OBFa (Output Buffer Full, tác động mức thấp), khi có dữ liệu từ vi xử lý gởi ra port A, tín hiệu OBFa sẽ yêu cầu thiết bị bên ngoài nhận dữ liệu.

Bit PC_6 trở thành bit ACKa (Acknowledge Input, tác động mức thấp) thiết bị nhận dữ liệu dùng tín hiệu này để báo cho 8255A biết tín hiệu đã được nhận và sẵn sàng nhận dữ liệu tiếp theo.

Bit PC_3 trở thành bit INTRa (Interrupt Request, tác động mức cao), bit này có mức Logic khi hai bit OBF a = 1, ACKa = 1 và bit INTEa (Interrupt Enable) ở bên trong 8255A ở mức 1. Tín hiệu INTRa tác động đến ngõ vào ngắt của vi xử lý để báo cho vi xử lý biết: thiết bị bên ngoài đã nhận dữ liệu ở port A.

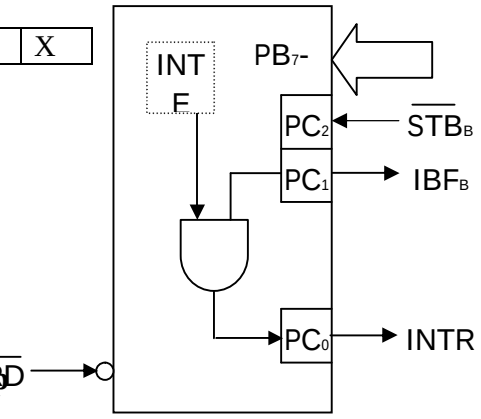
Các bit còn lại của port C: PC_4, PC_5 là các bit xuất/nhập bình thường tùy thuộc vào bit D_3 trong từ điều khiển hình trên. Các bit XXX dùng để thiết lập cho nhóm B.

- ***Nhóm B làm việc ở Mode 1:***
  - Port B được cấu hình là port nhập dữ liệu :

Chức năng của các đườfg tín hiệu được trình bày ở hình sau:

Từ điều khiển:

1	X	X	X	X	1	1	X



Hình 1.18 : Port B Nhập

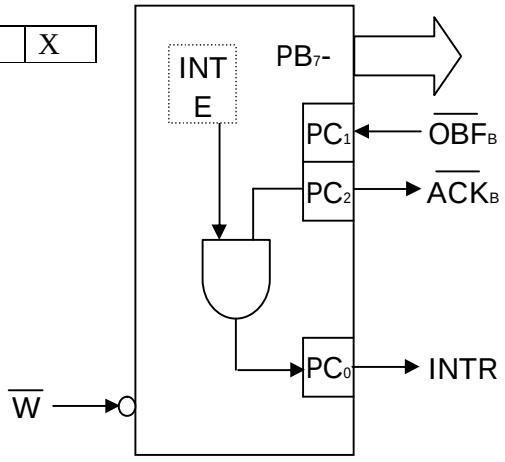
Chức năng của các bit điều khiển giống như nhóm A hoạt động ở Mode 1.

- Port B được cấu hình là port xuất dữ liệu:

Chức năng của các đường tín hiệu được trình bày ở hình sau:

Từ điều khiển :

1	X	X	X	X	1	0	X



Hình 1.19 : Port B Xuất

Chức năng của các bit điều khiển giống như nhóm A hoạt động ở Mode 1.

Các bit XXX được dùng thiết lập cho nhóm A, bit D_0 không có tác dụng trong trường hợp cả hai nhóm cùng làm việc ở Mode 1.

3). ***Nhóm A của 8255A làm việc ở Mode 2:***

Mode 2 là kiểu hoạt động *Strobe Bi-directional IO*, sự khác biệt với Mode 1 là port có hai chức năng xuất – nhập dữ liệu.

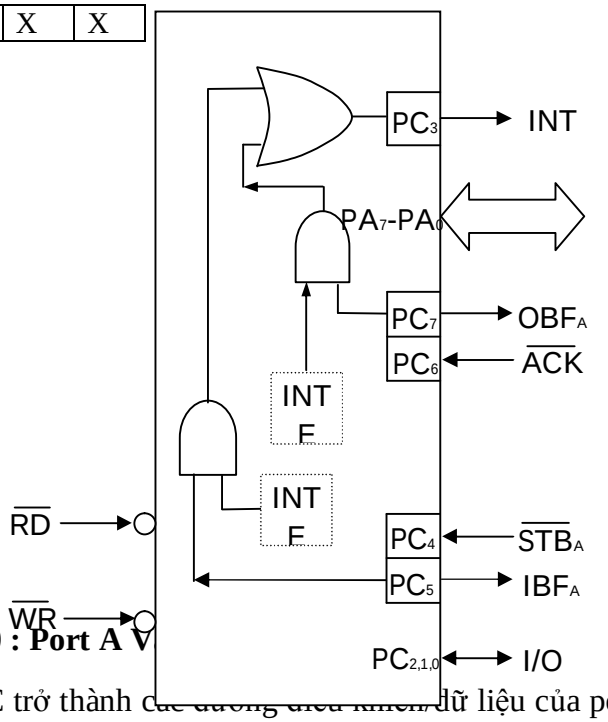
Từ điều khiển khi hai nhóm A và B hoạt động ở Mode 2:

1	1	X	X	X	X	X	X

Chức năng của các đường tín hiệu được trình bày ở hình sau:

Từ điều khiển:

1	0	1	1	D3	X	X	X



Hình 1.20 : Port A V

Các đường tín hiệu của port C trở thành các đường điều khiển dữ liệu của port A.

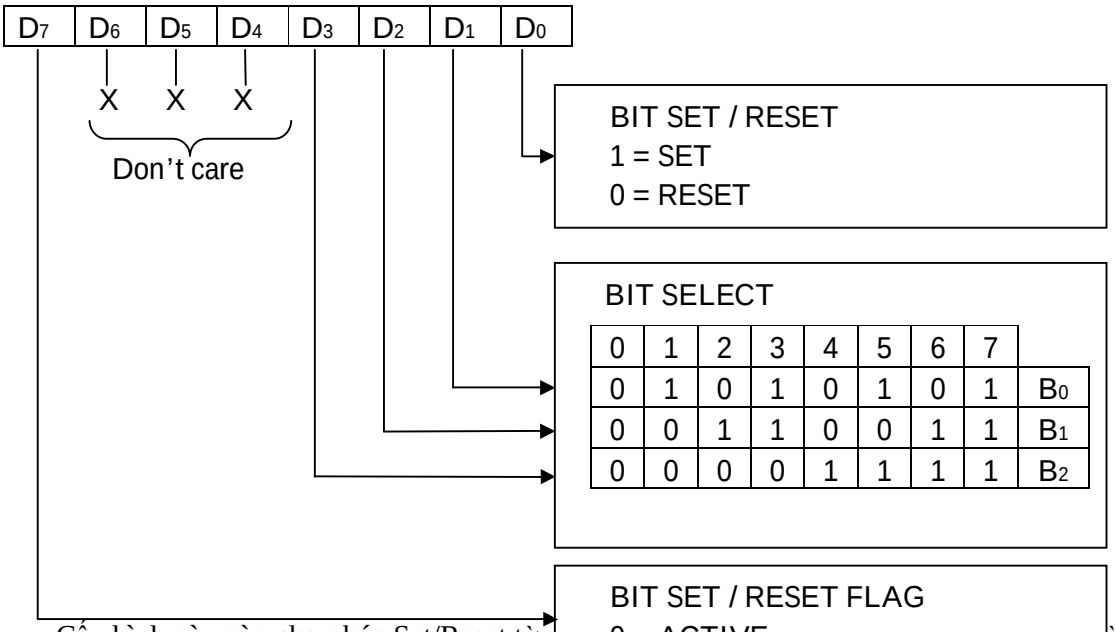
Bit PC_7 trở thành bit OBFa, PC_6 trở thành bit ACKa, PC_4 trở thành bit STBa, PC_5 trở thành IBFa và bit PC_3 trở thành bit INTRa. Chức năng của các đường tín hiệu giống như Mode 1, chỉ khác là tín hiệu ngỏ ra INTRa =1, INTE1 = 1 hoặc IBFa = 1, INTE2 = 1.

Các bit PC 2,1,0 còn lại có thể là các bit I/O tùy thuộc vào bit điều khiển c3 nhóm B.

Chú ý khi nhóm A làm việc ở Mode 2, nhóm B chỉ được phép hoạt động ở Mode 0.

Cấu hình của từ điều khiển Set/Reset bit INTE khi 8255A hoạt động ở Mode 1 hoặc

Mode 2 được trình bày ở hình sau:



Cấu hình này còn cho phép Set/Reset từng bit của port C khi làm việc ở Mode 1 hoặc Mode 2. Từ điều khiển cấu hình là bit $D_7 = 0$.

Bit D_0 dùng để Set/Reset bit INTE, khi D0 = 1 thì INTE = 1 (cho phép ngắt), khi $D_0 = 0$ thì INTE = 0 (không cho phép ngắt). Ba bit D_1, D_2, D_3 dùng để chọn một bit của port C, gán mức Logic của bit D_0 cho bit của port đã chọn.

Trong thực tế port A và port B thường được cấu hình với nhiều Mode khác nhau. Ví dụ nhóm A hoạt động ở Mode 2 nhóm B làm việc ở Mode 0.

Phần II

THIẾT KẾ VÀ THI CÔNG PHẦN CỨNG

Chương I:

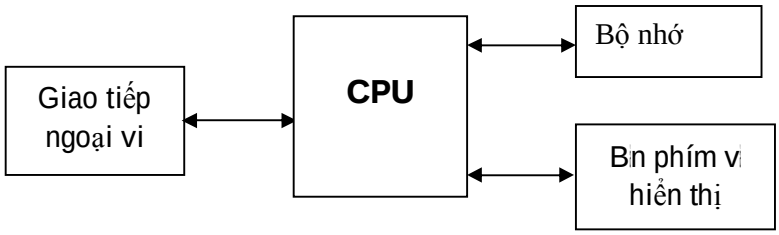
THIẾT KẾ PHẦN CỨNG VÀ TÍNH TOÁN

I.)Tóm tắt thiết kế:

Yêu cầu của đề tài là thiết kế hệ thống kít vi điều khiển 8951 để áp dụng vào phương pháp giảng dạy và thực tập cho bộ môn vi xữ lý. Các lệnh đaều khiển và chương trình được nhập từ bàn phím do đó việc thiết kế các phần cứng phải đảm bảo các yêu cầu sau:

- Hoạt động cuả máy phải chính xác, dễ sử dụng.
- Kết cấu phần cứng không quá phức tạp, linh kiện thi công phải có trên thị trường Việt Nam và giá thành hợp lý.

Sơ đồ khối cuả hệ thống:



Hình 2.1 : Sơ đồ khối hệ thống

Hệ thống vi điều khiển gồm bốn phần chủ yếu sau:

+ Đơn vị xữ lý trung tâm CPU.
+ Bộ nhớ.
+ Giao tiếp ngoại vi.
+ Khối quét phím – Hiển thị.

II.)Chức năng cuả từng khối:

1/. Đơn vị xữ lý trung tâm CPU(Center Processing Unit)

Đây là khối quan trọng nhất cuả hệ thống. CPU giữ nhiệm vụ tiếp nhận và xử lý trung tâm, khả năng tiếp nhận và phân tích các yêu cầu tác động, từ đó có đáp ứng thích hợp. Điều này được thể hiện qua khái niệm tập lệnh cuả vi điều khiển, chẵn hạn tập lệnh của 8951. Tập lệnh này cho ta khã năng hoạt động có mức độ cuả đơn vị xử lý trung tâm, khắc phục và hạn chế các điều đó các nhà sản xuất đã cố gắng thiết lập tập lệnh sao cho khi kết hợp chúng lại với nhau đơn vị xữ lý trung tâm xử lý thêm nhiều tình huống khác mà từng lệnh riêng biệt không thể giãi quyết được. Đây chính là cơ sở cuả chương trình hệ thống.

2) Khối bộ nhớ: **(Memory)**

Đây là nơi lưu trử chương trình cũng như các số liệu thu nhận và các kết quả sau quá trình làm việc nào đó khối này không thể thiếu được trong hệ thống vi điều khiển và đó là nơi cất giữ khả năng mà người lập trình tạo ra cho hệ thống

3)*Khối giao tiếp ngoại vi*:

Đây là phần kết nối giữa CPU và bên ngoài. Do yếu tố khách quan là CPU chỉ có một tuyến dữ liệu trong khi yêu cầu giao tiếp bên ngoài nhiều, vì vậy phần giao tiếp là đơn vị chịu trách nhiệm thiết lập các mối quan hệ từ bên ngoài với hệ thống tại thời điểm có yêu cầu.

Để đảm nhiệm việc này thiết bị ngoại vi cũng được gán cho 1 địa chỉ để tiện cho việc truy xuất và dĩ nhiên kèm theo những tính hiệu điều khiển thích hợp từ CPU và tuyến dữ liệu để trao đổi thông tin

4) *Khối hiển thị và bàn phím:*

Đây là khối phục vụ đắc lực cuả hệ thống vi điều khiển, bàn phím là nơi người lập trình nhập số liệu cũng như chương trình vào bộ nhớ, bộ hiển thị giúp người lập trình kiểm soát được việc nhập số liệu cũng như 1 kết quả trong quá trình làm việc. Trong một số trường hợp đôi khi chúng ta phải công nhận chúng là hai thiết bị ngoại vi luôn đi kèm với một hệ thống điều khiển. Mặt khác vì đây là những thiêt bị ngoại vi bộ hiển thị và bàn phím không làm việc trực tiếp vơí CPU mà phải thông qua giao tiếp ngoại vi. Việc định vị chúng dựa trên bộ phận cuả khối giao tiếp mà mỗi thiết bị trực tiếp làm việc.

III.)THIẾT KẾ VÀ PHÂN TÍCH NGUYÊN LÝ HOẠT ĐỘNG THEO TỪNG KHỐI:

***1.Khối xữ lý trung tâm CPU*:**

Đây là khối tiếp nhận và xử lý mọi thông tin liên quan đến hoạt động cuả máy. Khối xử lý trung tâm hoạt động không ngừng trong suốt thời gian làm việc do đó độ bền bỉ và khả năng xử lý nhanh chófg linh hoạt với mọi tình huống phụ thuộc vào phẩm chất linh kiện mà chúng ta sử dụng cũng như nguồn cung cấp cho hệ thống. Chính vì vậy chọn bộ xử lý trung tâm là đối tượng đầu tiên cho thiết kế hệ thống không ngoài lý do trên, nó sẽ quyết định khả năng hoạt động cho hệ thống cuả chúng ta.

a) Phân tích yêu cầu chọn linh kiện:

Chúng ta sẽ căn cứ vào các yêu cầu sau để chọn linh kiện:

- Có nền tảng cơ bản cuả một hệ thống vi điều khiển.
- Khả năng ưu việt so với hệ thống số.
- Dể sữ dụng cũng như thiết kế các ứng dụng.
- Có tài liệu liên quan.
- Không yêu cầu cao trong thiết kế phần cứng.
- Mua được trên thị trường Việt Nam.

Cho đến nay lĩnh vực vi xử lý đã phát triển rất xa so với thời kỳ đầu cuả nó từ hệ thống 8 bít đã được nâng lên 16 bit, 32 bit, 64 bit có khả năng quản lý tới 128 kbytes. Cùng với sự phát triển cuả công nghệ phần mềm đã cho người lập trình cách triệt để nhất các thông tin được cập nhật hoá ngày nay như vậy sẽ khó khăn cho việc lựa chọn bộ vi xử lý để đáp ứng yêu cầu trên.

Tuy nhiên chúng ta dựa vào những loại hiện đã có trên thị trường Việt Nam mà chúng ta thường gặp như: INTEL8085, INTEL8080, INTEL 8951, ZILOG Z80,

MOTOROLA 6802 đây là những hệ thống 8 bit cách đây khá lâu nhưng vẫn còn chỗ đứng trong một số thiết bị máy móc.

Các họ trên điều thỏa mãn hai điều kiện đầu tiên vì chúng là các họ xử lý đầu tiên đại diện cho hệ thống máy tính hiện nay và chương trình phần mềm ứng dụng linh hoạt . Đây cũng chính là điểm yếu nhất cuả hệ thống số.

Về tài liệu do việc cập nhật tài liệu hàng ngày nên tài liệu rất phong phú về họ vi xử lý này và giá thành cuả nó cũng không chênh lệch nhiều nên cũng rất khó khăn cho việc lựa chọn vi xử lý thích hợp, chỉ còn dựa trên độ phức tạp cuả phần cứng để chọn ra bộ vi xử lý thích hợp nhất.

Trước hết ta xem về cấu trúc chân ta thấy 8085 có 8 bit thấp cuả address bus được đa hợp để tạo ra 8 bit data bus (kí hiệu AD_0-AD_7). Đặc điểm phải cần 1 mạch chốt để gởi tín hiệu ra tuyến dữ liệu cuả vi xử lq. Mặc khác ở 8085 hai tín hiệu điều khiển bus và I/O được cấu tạo chung trên một chân phân biệt nhau bởi trạng thái logic mà không tách rời hai chân. Những yếu tố trên đủ làm cho mạch điện thêm phức tạp, không rõ ràng và khó kiểm sót.

Với 8080 mặc dù address bus và data bus được tách rời nhưng gặp phải một trở ngoại khác đó là không có chân điều khiển bus và I/O mà phải thông qua một linh kiện khác đó là 8255 vừa là bộ đệm hai chiều vừa là bộ tạo tín hiệu điều khiển hệ thống. Hơn nữa, CPU cần có một bộ nguồn 3 cấp điện áp +5v,-5V và 12v. Đây là một trở ngại về phần cứng.

Như vậy ta chỉ còn hai họ vi xử lý và một họ vi điều khiển đó là: MOTOROLA 6802, ZILOG Z80 và INTEL 8951. Tuy nhiên đối với một họ vi xử lý không chỉ đơn thuần dựa vào phần cứng, tính linh hoạt chủ yếu dựa vào phần mềm. Muốn thay đổi năng lực phần mềm của từng loại ta phải xem xét mỗi họ giải quyết bài toán như thế nào căn cứ vào kích thước chương trình, vào thuật giải từ đó chúng ta mới có câu trả lời thích hợp. Ta lập bảng so sánh các linh kiện với nhau:

Các thanh ghi	6802	Z80	8951
Bộ tích lũy 8 bit	A,B	A	A,B
Thanh ghi chỉ số 16 bit	IY,IY	IX,IY	
Bộ đếm chương trình	PC	PC	PC
Con trỏ ngăn xếp SP(16 bit)	SP	SP	SP
Thanh ghi cỡ			CY,AC
Thanh ghi đa năng		B,C,D,H,L	
Thanh ghi dữ trữ		B',C',D',H',L'	
Timer /counter(16bit)			2timer/counter
Thanh ghi ngắt		I,R	IP,IE
Thanh ghi đặc biệt			22
Ngân hàng thanh ghi			4

Một điểm khác biệt ở cấu tạo của vi mạch giữa 6802, Z80 và 8951 là dao động của 6802 và 8951 được cấu tạo ngay trong IC chỉ cần trang bị thêm bên ngoài một thạch anh là đủ. Đây chính là điểm mà 6802 và 8951 hơn hẳn Z80. Trong các chương trình viết bằng ngôn ngữ cấp thấp các lệnh chuyển dời dữ liệu chiếm một vị trí quan trong hơn nữa các phép toán số học cũng như logic chỉ thực hiện trên các thanh ghi nên số lượng thanh ghi cũng chiếm một vị trí quan trọng. Đây là một yếu tố giúp cho

người lập trình lựa chọn hê thống thích hợp. Qua bảng so sánh ta thấy họ vi mạch điều khiển 8951 đáp ứng được hầu hết các điều kiện đặt ra.

Hơn nữa một trong những đặc tính nổi bậc của vi điều khiển là giúp cho người lập trình có thể can thiệp vào từng bit của port xuất nhập, mà chỉ dùng một lệnh duy nhất (ví dụ: SETB P1.3:đặc bit 3 port 1 lên 1) điều này sẽ rất khó khăn thực hiện với các vi xử lý khác, ngoài ra còn hai bộ TIMER/COUNTERS được dùng như một đồng để đo các chu kỳ thời gian hoặc có thể hoạt động như một bộ đếm.

*Tóm lại:chúng ta sẽ chọn vi điều khiển 8951 cho thiết kế hệ thống như mục tiêu đề ra.

b) Thiết kế mạch xử lý trung tâm:

Cấu trúc mạch xử lý trung tâm quyết định toàn bộ hệ thống cho nên đây là khâu đầu tiên được thiết kế và cũng là khâu đơn giản nhất bởi vì nó không phụ thuộc vào các thành phần còn lại cuả hệ thống. Phần quan trọng nhất chúng ta đã thực hiện trong việc lựa chọn vi điều khiển 8951, còn xử lý các công việc do phần mềm thực hiện. Công việc của chúng ta là thiết kế các mạch xung quanh CPU như: mạch dao động, mạch chốt, mạch Reset, mạch nguồn…

c) Thiết kế mạch tạo tín hiệu điều khiển: (mạch dao động)

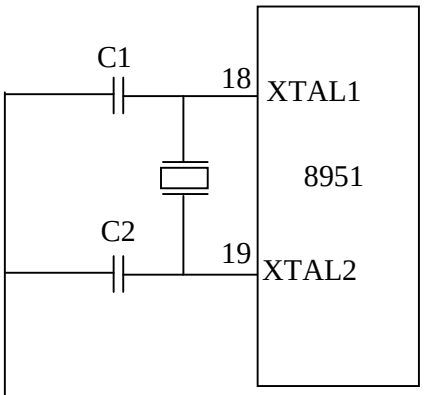
Trong hệ thống số nói chung và hệ thống vi điều khiển nói riêng xung clock đóng vai trò quan trọng trong toàn bộ hệ thống, một phần do tính chất làm việc của các mạch logic như: **counter, timers**… nhưng chức năng quan trọng nhất của xung clock là đồng bộ các hoạt động cuả các linh kiện khác nhau trong mạch do đó mạch tạo xung thiết kế phải thoả mản điều kiện sau:

- Đảm bảo độ ổn định cuả tần số làm việc, giảm tối thiểu sai số ngẫu nhiên.
- Thích ứng với các linh kiện làm việc liên quan đến kỹ thuật số như đã giới thiệu ở phần trước, mạch tạo xung được chế tạo trong IC 8951 do vậy điều kiện thứ hai coi như đã thỏa. Đối với mạch dao động dùng RL độ ổn định không cao do khó xác định được chính xác giá trị RL do đó không thỏa yêu cầu đặt ra.

Sử dụng thạch anh là có tính thuyết phục nhất bởi thạch anh có tính ổn định cao và có giá trị xác định sai số rất nhỏ trong hệ thống bit, có thể nói hầu hết các linh kiện đều trực tiếp hoặc gián tiếp liên quan đến tần số clock. Chính vì vậy việc lựa chọn tần số làm việc thích hợp là một trong những buớc quan trọng nhất.

Như chúng ta đã biết ở điều kiện lý tưởng tần số làm việc của CPU phải hoàn toàn tương thích với tốc độ truy xuất dữ liệu cuả bộ nhớ. Điều này khó có thể thực hiện được vì khó có thể kiếm trên thị trường hiện nay do vậy ta phải chọn giải pháp khác linh hoạt hơn mà vẫn đáp ứng được tần số làm việc cho hệ thống.

Qua các tài liệu cho thấy tốc độ truy xuất dữ liệu trung bình khoảng 120ms đến 450ms tương ứng với 2,2MHz đến 8,3 MHz với 8951 tần số làm việc thường từ 0Hz đến 24MHz do đó ta chọn tần số trong khoảng này là được, ở đây tần số làm việc được chọn là 12MHz do thạch anh 12MHz rất phổ biến hiện nay và giá thành hạ so với các loại khác, mạch được mắc như sau: C1, C2 ổn định cho thạch anh.



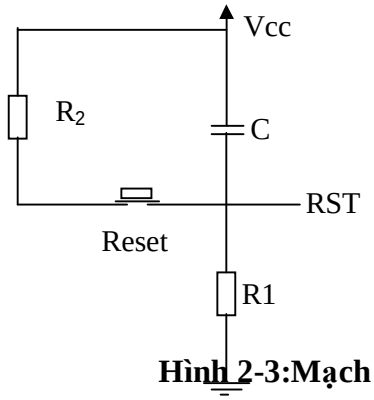
Hình 2-2:mạch dao động

d) Thiết kế mạch Reset:

Do chương trình quản lý và điều khiển hệ thống là chương trình đầu tiên khi tiến hành khi mới cấp điện. Cho nên tại thời điểm đó thanh ghi pc phải lưu tại địa chỉ đầu tiên của chương trình, muốn làm được việc này cần phải có một mạch tác động bên ngoài. Đó chính là mạch Reset, chính lúc nhận Reset, CPU sẽ xoá thanh ghi PC về địa chỉ ban đầu. Đây chính là tầm quan trọng của mạch reset.

Thực ra mạch reset chỉ là một mạch nhỏ với chức năng tạo ra một xung tác động vào chân reset của CPU tại thời điểm cấp nguồn cho hệ thống, cũng có thể sử dụng mạch reset này để reset một số linh kiện khác nếu có nhu cầu như 8255. Đối với 8951 chân reset được kí hiệu RST(9) chịu tác động tương ứng với trạng thái [H], có nghĩa là khi chúng ta đưa tín hiệu vào ở mức [1] thì sẽ làm CPU quay trở lại trạng thái ban đầu. Tác động này gọi là reset CPU. Lưu ý là chân mang kí hiệu reset luôn thường trực để ở trạng thái thấp [L] chỉ lúc nào cần reset CPU, ta mới tạm thời đưa lên trạng thái cao [H].

Mạch điện sâu đây đáp ứng được tất cả các yêu cầu đặt ra:



Hình 2-3:Mạch reset.

Giải thích: Đối với mạch này khi chúng ta cho điện áp vào mạch thì mạch reset sẽ tự động tác động nên được gọi là mạch tự động reset hay mạch reset khi đóng nguồn cung cấp (power on reset). Ngoài ra bất kì lúc nào cần thiết chúng ta vẫn có thể nhấn công tắt reset xuống để khởi động lại hệ thống.

Thời gian reset cuả 8951 tác động ở mức cao trong khoảng hai chu kì máy tức là khoảng 2μs (trường hợp mạch dao động sử dụng thạch anh 12MHz) sau đó xuống thấp để 8951 bắt đầu làm việc.

Dựa vào thời hằng Rl để tính toán ta chọn được các giá trị như sau:

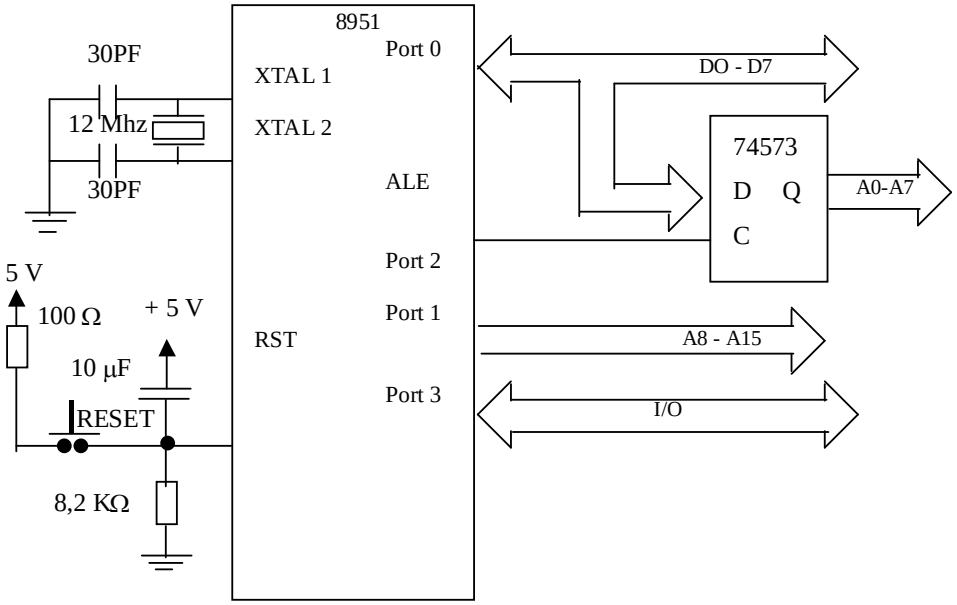
R_1=8,2 kΩ

R_2=100 Ω

C=10 μF

e) thiết kế mạch chốt địa chỉ:

Với 8951 8 bit thấp cuả địa chỉ được đa hợp để tạo ra 8 bit cuả data bus (kí hiệu từ Do đến D_7) CPU sẽ điều khiển mạch chốt cho xung ra ở chân ALE cho phép chốt địa chỉ vào thanh ghi bên ngoài trong suốt nửa chu kì đầu bộ nhớ. Sau khi thi hành xong lúc này các đường cuả port0 rảnh rỗi sẽ cho dữ liệu vào hoặc ra trong nữa chu kì còn lại. Trong phần thiết kế này ta chọn mạch chốt 74ALS573 là vi mạch có 8 ngõ vào và 8 ngõ ra phù hợp 8 bit thấp cuả address bus.



Hình 2.4 Sơ đồ kết nối 8951 với mạch reset, dao động và mạch chốt

Vi mạch 74ALS573 có chứa 8 Flip-Flop 0 và 8 cổng đệm điều khiển khi xung đồng hồ ở mức 1 đầu ra chép lại đầu vào. Khi xung vào chuyển trạng thái từ 1 xuống 0 số liệu ở đầu ra sẻ bị chốt lại. Như vậy các mạch chốt lại theo mức đầu ra chép lại giá trị đầu vào. Tín hiệu điều chiển chốt địa chỉ được CPU đưa ra ở chân ALE (Adress Latch Enable: cho phép chốt địa chỉ) phải được nối vào chân C (chip). Khi chân OE (Output Enable) ở mức thấp thì 74ALS573 sẽ chốt địa chỉ vào thanh ghi 74ALS573. IC này gồm 8 chốt theo mức dương loại D với xung đồng hồ chung. Như vậy đầu vào xung đồng hồ được kích bằng xung chọn cửa ra có mức tích hợp cao. Độ dốc cuả đường xung phải thích hợp khảo sát chu kì vào/ra trên giản đồ sóng ta thấy 8951 cấp dữ liệu ổn định kể từ cạnh xuống cuả xung WR 74ALS573 có đầu ra 3 trạng thái, các bộ đệm đầu ra điều có mức điều khiển với mức tích cực thấp.

f) Thiết kế bộ nguồn:

*Yêu cầu:

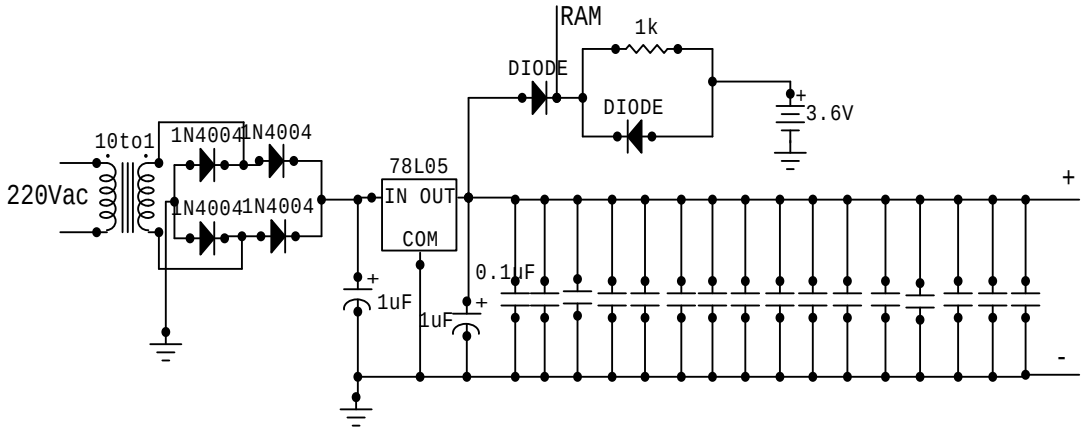
1. Cung cấp một nguồn V_{DC} cho tất cả các IC trong mạch.
2. Có nguồn dự phòng trong trường hợp gặp sự cố ở nguồn điện.
3. Có nguồn nuôi RAM(back-up) vì tất cã các dữ liệu trong quá trình tính toán, xử lý hệ thống được lưu trữ trong RAM các dữ liệu này sẽ mất đi nếu chông có nguồn

nuôi RAM. Xuất phát từ những yêu cầu trên em đưa ra mãch điện như sau, đảm bảo tốt cho hoạt động cuả hệ thống.

Mạch nguồn bao gồm:

- Ổn áp7805 cung cấp điện cho toàn mạch.
- Tụ chống nhiễu.
- Nguồn nuôi RAM 3,6 v

Ta có sơ đồ mạch nguồn như sau:



Hình 2.5 : Mạch Nguồn

Trong đó các DIODE tạo thành cầu nắn điện

Hai tụ C 10 uF : lọc nhiễu nguồn.

16 tụ 0,1uF: các tụ chống nhiễu.

IC ổn áp được chọn là LM 7805 là loại ổn áp đứng có đặc điểm như sau:

- Điện áp ngõ vào 8-35v
- Điện áp ngõ ra ổn áp 5v
- Dòng điện ra danh định 1A

Nguồn 3,6v dùng để nuôi RAM khi có sự cố mất điện. Ta biết rằng dưới điện áp cung cấp là 3,6 v thì dữ liệu trong RAM sẽ không bị mất với mạch điện như trên khi bị mất điện diode D_6 được phân cực thuận cấp nguồn 3,6v bảo vệ RAM khỏi bị mất dữ liệu.

2.Thiết Kế Bộ Nhớ:

Khối bộ nhơ là khối quan trọng thứ hai sau khối xử lý trung tâm các chương trình điều khiển các dữ liệu nhập từ bên ngoài cũng như phát sinh từ bên trong chương trình điều phải được lưu trữ trong bộ nhớ. Có thể nói bộ nhớ là nơi CPU thường xuyên trao đổi thông tin nhất. Vì vậy mà từ chi máy tính ra đời cho đến nay cộng với sự cải thiện không ngừng cuả kỹ thuật xử lý, bộ nhớ ngày càng được tối ưu hoá không chỉ về mặt dung lượng, kích thước mà còn cả về thời gian truy xuất dữ liệu. Chúng ta có hai loại bộ nhớ thông dụng thứ nhất là loại điện từ thường thấy nhất ở dạng băng từ điã từ loại này có ưu điểm có thể mang đi được dung lượng lớn, nhược điểm truy xuất chậm. mach điều khiển dữ liệu cồng kềnh. Loại thứ hai là các loại mạch nhớ bán dẫn phương thức nhớ dựa trên tính chất vật lý của chất bán dẫn hay các trạng thái logic cuả mạch số. Loại này có ưu điểm tốc độ truy xuất dữ liệu cao (hàng nano giây-ns), kích thước

nhỏ, điều khiển dễ. Nhược điểm cuả nó là không có khã năng tích trữ dữ liệu với dung lượng lớn.

Để có thể chọn ra loại bộ nhớ thích hợp nhất cho hệ thống chúng ta xét đến đặc tính cuả mỗi loại để chọn lựa cho phù hợp với yêu cầu cuả mạch.

a)Phân tích yêu cầu cuả hệ thống – chọn linh kiện:

*Các yêu cầu cuả hệ thống kit:

1. Mạch nhớ phải được gắn cùng băng mạch chính.
2. Dung lượng đáp ứng yêu cầu của hệ thống
3. Chương trình điều khiển kiểm soát không được mất sau khi cắt nguồn cung cấp
4. Tốc độ trao đổi dữ liệu phải cao hơn tốc độ truy xuất dữ liệu cuã CPU sữ được dụng.
5. Gọn nhẹ không chiếm nhiều diện tích.
6. Công suất tiêu thụ thấp.

Dựa vào yêu cầu thứ nhất và thứ saú ta thấy chỉ có bộ nhớ bán dẫn là đáp ứng được. Bộ nhớ từ điện không thích hợp vì phải có mạch điều khiển trao đổi dữ liệu, mạch điều khiển trao đổi các chuyển động cơ khí. Hơn nữa việc đọc ghi liên tục không cho phép sử dụng bộ nhớ điện từ và cuối cùng là bộ nhớ điện từ tiêu thụ công suất gấp nhiều lần bộ nhớ bán dẫn chính từ lý do trên bộ nhớ bán dẫn được chọn thiết kế trong đề tài này.

b) Kết nối chi tiết:

Để kết nối vi điều khiển với bộ nhớ một cách chi tiết phải đặt ra một số yêu cầu sau:

- Dùng Microcontroller để truy xuất được 64 kbyte dung lượng bộ nhớ.
- Thiết kế 16 kbytes bộ nhớ EPROM dùng hai IC 6264 có dung lượng 8kbytes/1 IC.
- Thiết kế bộ nhớ RAM có dung lượng 16 kbytes dùng 2 IC 2764 có dung lượng 8 kbytes cho mỗi IC
- Trong vùng 64 kbyte chỉ sữ dụng 32 kbytes đầu từ kbyte thứ 1 đến kbyte thứ 32. Kbyte thứ 33 đến kbyte thứ 64 không sữ dụng cho bộ nhớ mà sử dụng cho các mục đích khác.
- Địa chỉ cuả vùng nhớ 64 kbyte là0000H-FFFFH vì chỉ sữ dụng 32 kbyte đầu tiên nên vùng nhớ này có địa chỉ từ 0000H-7FFFH.
- Trong vùng 32 kbyte đầu tiên. từ kbyte thứ 1 đến kbyte thứ 16 được dùng cho bộ nhớ EPROM có địa chĩ từ 0000H -3FFFH. 16 kbyte kế tiếp theo đuợc sữ dụng cho bộ nhớ RAM có địa chỉ từ 4000H-7FFFH

+Thiết kế bộ nhớ EPROM:

- Do EPROM trong yêu cầu sử dụng có dung lượng 16 kbyte nên IC này phải có 13 đường địa chỉ $A_{12,}A_{11},A_{10},A_9 \ldots A_0$. Địa chỉ đầu tiên là$0.0000.0000.0000_2$ Và địa chỉ cuả ô nhớ cuối cùng là $1.1111.1111.1111_2$ nếu viết theo số HEXA 16 bit thì có địa chỉ đầu tiên là 0000H. và địa chỉ cuối cùng là FFFFH.

Như vậy nếu gắn EPROM đầu tiên vào 16 kbyte đầu tiên nó sẽ chiếm bắt đầu từ 0000H-1FFFH và EPROM thứ 2 sẽ chiếm vùng nhờ kế tiếp có địa chỉ từ 2000H-3FFFH ta có bản đồ như sau:

IC	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$	Hex	Kbyte
EFROM1	0 0 0 0 0 0 0 1	0 0 0 0 1 1 1 1	0 0 0 0 1 1 1 1	0 0 0 0 1 1 1 1	0000 1FFF	1-8

EFROM2	0 0 1 0 0 0 1 1	0 0 0 0 1 1 1 1	0 0 0 0 1 1 1 1	0 0 0 0 1 1 1 1	2000 3FFF	9-16

Mỗi EPROM có 8 kbyte để EPROM truy xuất hết 8 kbyte đòi hỏi người thiết kết nối 13 đường địa chỉ $A_{12,}A_{11},A_{10},A_9…A_0$ cuả vi điều khiển đến 13 đường địa chỉ $A_{12,}A_{11},A_{10},A_9…A_0$ cuả bộ nhớ, 8 đường dữ liệu D7,D6,…D0 cuả vi điều khiển được nối với 8 đường dữ liệu D7,D6…D0 cuả 2 EPROM, đường tín hiệu EA\ được kết nối với đường OE\ cuả 2 EPROM. Đến đây Microcontroller chỉ giao tiếp với bộ nhớ thông qua ba bus là bus địa chỉ, bus dữ liệu và bus điều khiển, nếu dừng lại ở đây thì khi Microcontroller tạo ra một địa chỉ để truy xuất một ô nhớ thì cả hai EPROM đều nhận được địa chỉ từ bus địa chỉ và cùng dữ liệu ra từ bus dữ liệu. Khi đó Microcontroller nhận vào địa chỉ không biết là cuả ô nhớ nào. Để Microcontroller nhận đúng dữ liệu cần truy xuất cuả ô nhớ nào thì phải thiết kế mạch giải mã địa chỉ (sẽ được trình bày sau).

- **Thiết kế bộ nhớ RAM:**

Bộ nhớ RAM dùng là 16 kbyte tiếp theo bộ nhớ EPROM, sử dụng 2 RAM 2764 có dung lượng 8 kbyte, do dung lượng của RAM 2764 EPROM 6264 là bằng nhau nên vấn đề thiết kế cho RAM tương tự như EPROM chỉ khác địa chỉ xuất phát và chân OE\ cuả RAM được đến chân OE\ cuả vi điều khiển. Bản đồ nhớ cuả RAM:

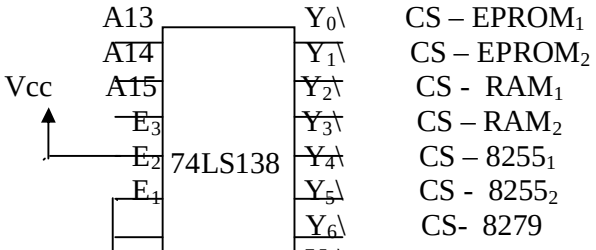
IC	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$	Hex	Kbyte
RAM1	0 1 0 0 0 1 0 1	0 0 0 0 1 1 1 1	0 0 0 0 1 1 1 1	0 0 0 0 1 1 1 1	4000 5FFF	17-24
RAM2	0 1 1 0 0 1 1 1	0 0 0 0 1 1 1 1	0 0 0 0 1 1 1 1	0 0 0 0 1 1 1 1	6000 7FFF	25-32

Nhìn vào bản đồ nhớ ta thấy rằng vùng nhớ Ram có địa chỉ tiếp theo của vùng nhớ Rom và địa chỉ đầu taên 4000H kết thúc ở ô nhớ cuối cùng tại 7FFFH. Vấn đề còn lại là giải mã để chọn ô nhớ tương ứng với từng Ram.

+ **Thiết kế mạch giải mã địa chỉ.**

Từ bảng đồ nhớ ta thấy có 3 bit A15A14A13 có tám trạng thái tương đươfg với 8 vùng nhớ, mỗi vùng nhớ có dung lượng 8kbyte. EPROM tương đương với trạng thái 000_2, $EPROM_2$ tương với trạng thái 001_2, Ram1 trạng thái 010_2, RAM_2 trạng thái 0112. Các còn lại sử dụng để giải mã cho các thiết bị khác cần truy xuất.

Từng trạng thái 3 bit cho phép truy xuất từng EPROM, từng RAM. Khi một EPROM hoặc một RAM được truy xuất thì các EPROM hoặc RAM khác không được truy xuất. Để đảm bảo cho các ứng dụng của kit, ngoài yêu cầu ROM, RAM chúng ta còn phải quan tâm đến việc giao tiếp với thiết bị bên ngoài thông qua IC 8255 và IC 8279 đảm nhận hai cực kì quan trọng, vừa quét hiển thị vừa giải mã bàn phím. Nhìn chung tất cả những linh kiện xung quanh 8951 đều kết nối song song vào Data Bus, Address Bus và control BUS như vậy cần phải có một sự giải đa hợp chọn linh kiện cho CPU vào một thời điểm nào đó nhằm tránh sự xung đột Bus làm cho các hoạt động của CPU bị rối loạn. Mạch giaĩi mã địa chỉ được thiết kế như sau:



Hình 2 .6 : Giải Mã Địa Chỉ

Chân A,B, C lần lượt nối vào các chân A_{13} A_{14} A_{15} của 8951. Các ngõ ra của 74LS138 lần lượt nối các chân CS (chip select) của ROM, RAM , 8255 và 8279.

3.Thiết kế bộ giao iếp ngoại vi:

Tương tự như bộ nhớ 8255 cũng có ba nhóm chân cơ bản

-Các chân dữ liệu đó từ Do -D_7;P_{A0}-P_{A7} ;P_{B0}-P_{B7} ;P_{C0}-P_{C7}

Các địa chỉA_0,A_1

-Các chân điều khiển: CS\(chip select), WR\(write), RD\(read), RST(reset) Vì đây là giao tiếp dữ liệu giữa hệ thống với dữ liệu bên ngoài nên khác với hệ thống nhớ chúng có hai nhóm dữ liệu. Nhóm dữ liệu D_0-D_7 được nối trực tiếp đến bus dữ liệu của hệ thống. Nhóm cổng xuất nhập P_A, P_B, P_C được đưa ra ngoài qua connector để giao tiếp với thiết bị bên ngoài. Trong 8255 có ba bộ đệm dữ liệu cho ba port và một thanh ghi dùng cho việc ấn định chế độ hoạt động (thanh ghi từ điều khiển). Việc định vị bốn vùng dữ liệu này thông qua các chân A_0, A_1 của từng IC 8255. Tương tự như bộ nhớ 8255 cũng chiếm một chỗ trong bản đồ nhớ, do có 2 cổng xuất nhập nên sẽ có hai vùng nhớ được chọn sao cho dễ dàng trong việc truy xuất. Trong hệ thốfg của chúng ta mỗi IC được chọn có địa chỉ như sau:

+ 8255 -1 có địa chỉ từ 8000H – 8003H
+ 8255 –2 Có địa chỉ từ A00H – A003H

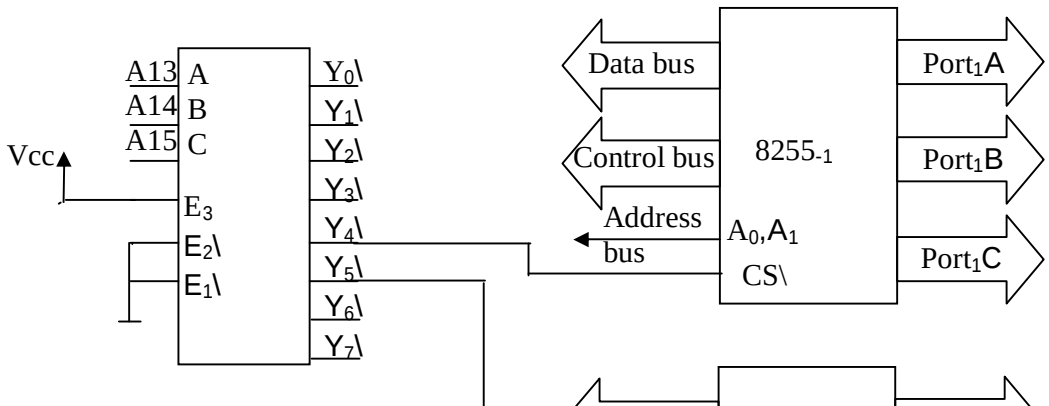
Sở dĩ ta chọn địa chỉ của 8255 như vậy là do ta qui đồng dung lượng của mỗi 82555 là 8 Kbyte để dễ dàng cho việc giải mã. Ta sử dụng phương pháp định vị theo cách định vị bộ nhớ IO(mapped memory) có nghĩa là định vị theo cách định vị bộ nhớ, do đó địa chỉ dùng cho việc định vị này phải đủ 16 bit. Để thuận tiện cho việc định vị này ta tận dụng tiếp ngõ ra kế tiếp của IC giải mã 74LS138 đã thiết kế ở phần trước. Với cách lý luận tương tự ta chọn chân Y_4 để nối đến chân CS của 8255 -1 và chân Y_5 nối đến chân CS của 8255 – 2.

Các đường điều khiển WR\, RD\, RST được nối đến các đường WR\, RD\, RST cuả vi điều khiển.

Các đường địa chỉ của A_1,Ao,cuả 8255 được nối đến các đường A_1,Ao cuả vi điều khiển.

Các đường dữ liệu Do-D_7 được nối đến các đường dữ liệu Do-D_7 cuả vi điều khiển.

Từ các lý luận trên ta có sơ đồ nối kết sau:



Hình 2 .7 : Sơ đồ kết nối hai 8255 với 74138

3)Thiết Kế Bàn Phím Và Bộ Hiển Thị:

Mục đích thiết kế một hệ thống ứng dụng vi điều khiển trong tự động điều khiển, hệ thống mà chúng ta đang khảo sát nhất thiết phải có hai thiết bị bàn phím và bộ hiển thị (keyboard and display). Trong đó bộ hiển thị có tác dụng giúp người sữ dụng kiểm tra được chương trình điều khiển và có thể dùng làm nơi thông báo các kết quả thu nhận được trong một tác vụ nào đó với bàn phím chúng ta dùng làm nơi nhập các chương trình thử nghiệm vào RAM trước khi chính thức đưa vào ROM. Như vậy chức năng cuả hai thiết bị này khá rõ ràng chúng ta sẽ phân tích các yêu cầu liên quan để có một thiết bị tối ưu nhất.

a)Thiét Kế Bàn Phím:

Phân tích yêu cầu hệ thống:

Bàn phím là một đơn vị lối vào đơn giản nhấttrong hệ thống máy tính nó chỉ vào một mạch mã hóa bàn phím để đổi thành mã nhị phân. Đa số bàn phím hiện nay đều là loại bàn phím dạng ma trận.

Việc mã hóa tín hiệu bàn phím kiểu ma trận đòi hỏi phải dùng nhiều maçch logic vì phải có một mâch đếm tiến hành quét trên các công tắc phím từng cột phải được quét qua để biết có phím nào được ấn hay không nếu có thì mạch đếm sẽ chựng lại và con số trong mạch đếm tương ứng lúc đó sẽ tương ứng với mã số nhị phân cuả phím được ấn. Đây là nguyên tắc hoạt động cuả bàn phím có mã hoá. Ưu điểm cuả bàn phím mã hoá là tốc độ đáp ứng cao nhưng mạch điện phức tạp và độ linh hoạt không phong phú.

Ngày nay người ta thường dùng loại bàn phím không mã hoá sử dụng một chíp vi tính chuyên dùng nguyên lý hoạt động cuả mạch này như sau:

Tất cả các đường cột được nối chung với một cổng ra cuả chip vi tính, các đường hàng đuợc nối với cổng ra thứ hai. Một phần mềm mô phỏng theo hoạt động cuả mạch phần cứng sẽ tiến hành quét lên các phím và mã hoá vị trí của phím ấn thành một số nhị phân, việc chuyển thành mã tương ứng với phím này được tiến hành bằng phần mềm chứ không cần thêm một mạch phần cứng nào khác. Ưu điểm cuả loại này là mạch điện đơn giản và độ linh hoạt cao nhưng có nhược điểm là đáp ứng không cao bằng loại mã hóa.

Như đã khảo sát ở phần một vi mạch 8279 là một chíp vi tính chuyên dùng có hai chức năng quét lên bàn phím đến 64 phím rời và hiển thị được đưa 16 LED 7 đoạn, do phạm vi của kit nên việc chọn vi mạch này là thích hợp nhất.

Mô tả bàn phím:

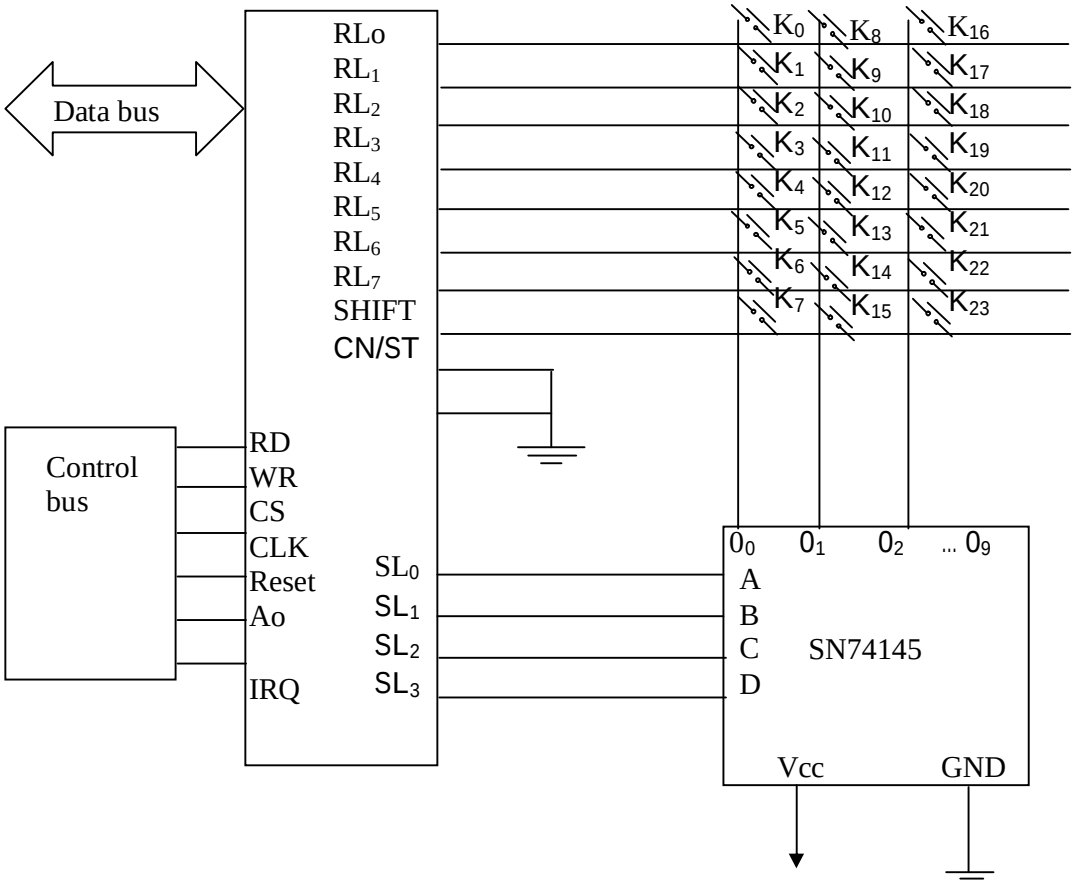
				Q
G	T	P	K	I
C	D	E	F	R
8	9	A	B	S
4	5	6	7	↓
0	1	2	3	↑

Bàn phím được thiết kế như nhưng công tắc thường hở việc tiếp xúc với bàn phím là nhiều nhất so với bất kì thiết bị nào trong hệ thống máy tính, bàn phím cơ học sẽ giúp ta biến đổi "tác vụ ấn một phím" thành tín hiệu gởi đến máy tính. Ở đây do yêu cầu thiết kế nên số phím có 16 phím nhập dữ liệu thể hiện dưới dạng số thập lục phân từ 0÷F và 8 phím chức năng cộng với một phím Reset.

Sơ đồ logic được trình bày ở phần một, để cho 8279 làm việc được chúng ta trước tiên phải thiết lập các từ điều khiển gửi ra cho 8279 các đường tín hiệu SCAN SL_0÷SL_3 dùng để quét dử liệu trên đường này có thể thiết lập theo haa kiểu Decode và Encode, nếu thiết lập theo kiểu Decode thì 4 đườngSL_3÷Slo chỉ có khã năng quét hiển thị 4 LED 7 đoạn.

Mà yêu cầu cuả 4 đường SL_0÷SL_3 phải ở chế độ Decode. Do vậy đầu tiên ta phải chọn 8279 ở chế độ Encode rồi sau đó đem giải mã 4 đường SL_0÷SL_3 để trở thành chế độ Decode và lúc bấy giờ SL_0÷SL_3 trở thàng 16 đường nên có khả năng hiện thị 16 LED. Để làm được điều này ta chọn IC giải mã 4 đường ra 10 đường (BCD to Decimal) 74145 vì kit chỉ cần 8 LED hiển thị là 4 LED dữ liệu và 4 LED hiển th ị địa chỉ cuả dữ liệu đó.

Từ các lý luận trên ta có sơ đồ nối kết bàn phím:



Hình 2 .8: Kết nối bàn phím

Các ngõ vào SHIFT và CNTL được dùng để mở rộng các phím tổ hợp đối với kit thiết kế này không cần mở rộng thêm nên ta nối mass.

- Nguyên Lý Làm Việc:

Để IC 8279 làm việc ở chế độ bàn phím và hiển thị ta phải chọn chế độ KKK= 000 (Encode Scan ceyboard keyclock out) và gửi các từ đều khiển này vào Ao để khởi tạo 8279 các đường SLo÷SL_3 liên tục quét qua 74145 để hiển thị và dò tìm phím ấn. khi có một phím ấn 8279 sẽ tự động chống dội sau khoảng 10,3 µs và kiểm tra dại một lần nữa để xem phím đó có còn được ấn hay không, nếu còn thì 8279 sẽ thiết lập mã phím ấn và lưu trữ mã cuả phím ấn vào bộ nhớ RAM bên trong sau đó sẽ báo cho vi điều khiển biết có một phím tác động và yêu cầu vi điều khiển nhận mã cuã phím này bằng cách làm thay đổi thanh ghi trạng thái FIFO làm cho 3 bit KKK sẽ khác 000 khi có một phím ấn.

Nhiệm vụ cuả 8279 là đọc mã của phím ấn vào để xử lý và Reset ngắt từ trạng thái FIFO trở về mức logic 0 chuẩn bị cho phím tiếp theo.

- Bảng mã Scan và mã giá trị cuả phím:

Phím	Mã Scan	Mã Hexa	Mã 7 đoạn
0	00h	00h	3Fh
1	01h	01h	06h
2	02h	02h	5Bh
3	03h	03h	4Fh
4	04h	04h	66h
5	05h	05h	7Dh
6	06h	06h	07h
7	07h	07h	7Fh
8	08h	08h	6Fh
9	09h	09h	77h
A	0Ah	0Ah	76h
B	0Bh	0Bh	39h
C	0Ch	0Ch	5Eh
D	0Dh	0Dh	79h
E	0Eh	0Eh	71h
F	0Fh	0Fh	
G	10h	10h	
T	11h	11h	
P	12h	12h	
K	13h	13h	
R	14h	14h	
S	15h	15h	
↑	16h	16h	
↓	17h	17h	

a.Thiết Kế Bộ Hiển Thị:

- Phân tích yêu cầu chọn linh kiện:

Đối vơđi bộ hiển thị chúng ta có một số yêu cầu sau:

+ Đảm bảo tính trực quan.
+ Có khả năng hiển thị 16 kí tự trong hệ thập lục phân.
+ Có thể trình bày cùng một lúc đia chỉ và nội dung trong địa chỉ tương ứng.
+ Mạch điều khiển đơn giản hiệu quả.
+ Tiêu thụ ít năng lượng.

Khác với những phần trước ở đây ta không đặc những chỉ tiêu kỹ thuật là yêu cầu trước tiên. Điều đó thật dễ hiểu vì bộ hiển thị là nơi người sữ dụng quan sát công việc cuả mình muốn tế nó phải thuận tiện cho việc quan sát tức là phải đảm bảo tính trực quan của thiết bị, chúng ta không tham vọng có một màn hình như máy tính cá nhân cho dù đó là bộ hiển thị lý tưởng. Việc đó đòi hỏi một mạch điện rất phức tạp và qúa khó đối với một sinh viên.

Với yêu cầu trình bày được các bộ hiển thị trên thị trường có thể đáp ứng được. Tuy nhiên ở đây chúng ta có được một sự lựa chọn. Bởi vì có hai linh kiện thuộc loại hiển thị vừa nêu: Led 7 đoạn (seven segment Led) và bộ hiển thị tinh thể lỏng liquid (crystal display).

Nhưng so sánh về chế độ phức tạp của mạch điều khiển giá thành, tính phổ biến trên thị trường và nhất là khả făng thiết kế. Nên người thiết kế quyết định chọn LED 7 đoạn làm bộ hiển thị cuả hệ thống.

Ngoài phần hiển thị dùng LED 7 đoạn dùng IC chuyên dùng 8279 hệ thống cuả chúng ta còn có thêm bộ hiển thị dùng 8 LED để hiển thị trạng thái hoạt động port1 cuả 8951 trong quá trình giao tiếp sẽ được giới thiệu ở phần tiếp theo.

• Thiết Kế Bộ Hiển Thị Led 7 Đoạn:

Bộ hiển thị giúp ta trực tiếp quan sát các tác vụ mà chúng ta ấn phím trong hệ thống hay nó giúp cho hệ thống cuã chúng ta hiển thị đia chỉ của ô nhớ và dữ liệu cần nạp vào cũng như dữ liệu đã được lưu trữ hay giúp ta xem được dữ liệu tức thời cuả các thanh ghi. Do đó ta chỉ cần bộ hiển thị gồm 8 LED đoạn. Để hiện thị một gía trị có nghiã ra LED 7 đoạn ta thực hiện các bước sau:

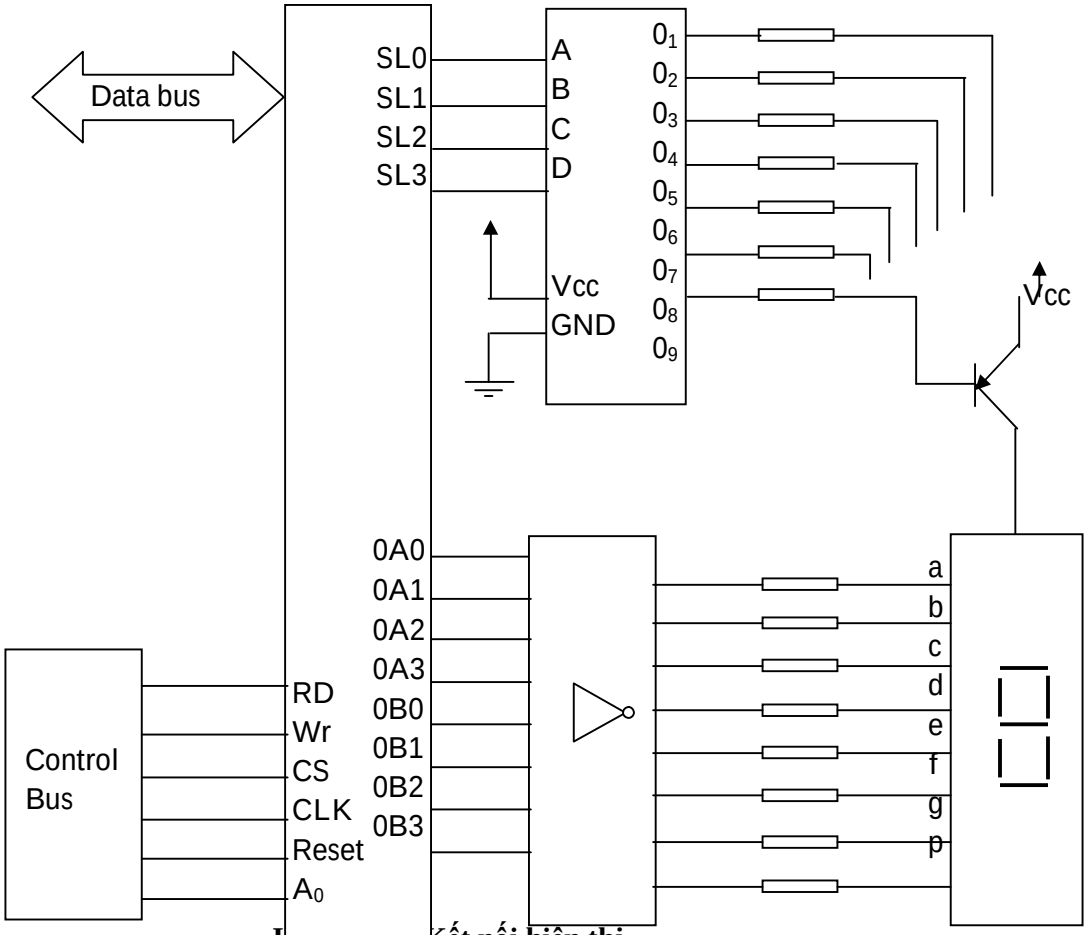
+ Đặt chế độ điều khiển cho 8279 hoạt động ở chế độ Decode tức là các đường SLo÷SL_7 phải luôn được quét để điều khiển 74145 cũng lần lượt quét thông qua các transitor switch để điều khiển LED hiển thị.

+ Nối các Anode chung cuả LED 7 đoạn lại và nối lên nguồn Vcc thông qua công tắc Transitor.

+ Nối các đoạn tương ứng a, b, c, d, e, f, g, dp cuả 8 LED lại với nhau và đưa chúng đến các đường dữ liệu hiển thị cuả 8279 ($0A_0$÷$0Ã_3$; $0Bo$÷$0B_3$) thông qua một bộ đếm đảo 7414 để lật ngược trạng thái (vì LED ở đây được chọn là loại Anode chung).

+ Dữ liệu nhị phân cũng như tín hiệu điều khiển các công tắc transitor sẽ được CPU quản lý và gửi ra theo một qui luật nhất định.

- Sơ Đồ Nguyên Lý Kết Nối Hiển Thị:



H Kết nối hiển thị

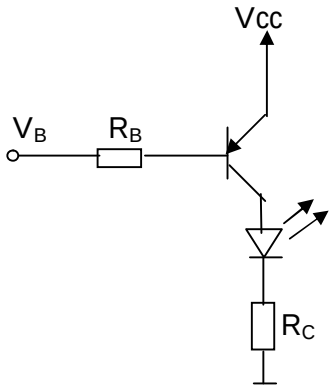
Nguyên lý hoạt

Để 8 LED hiển vào từ phải sang trái ta chọn Mode DD-=10, sau khi khởi tạo 8279 xong, để hiển thị ta gởi dữ liệu vào A_0 và 8279 đem cất các dữ liệu này lần lượt vào 16 ô nhớ RAM bên trong và lần lượt xuất các dữ liệu trong RAM này ra ngoài LED để hiển thị theo lối vào từ phải sang trái mà ta đã định trước, để xoá LED nào tắt ta gửi từ điều khiển C2 vào thanh ghi điều khiển, trong mạch các transitor đóng vai trò như một công tắc đóng mở LED và được điều khiển bởi các ngõ ra thấp của IC 74145. Các transistor này phải đóng mở sao cho nhỏ hơn thời gian lưu ảnh trên võng mạc của mắt khi đó chúng ta sẽ có cảm giác 8 LED sáng cùng lúc.

- Thiết kế hệ thống hệ thống hiển chế độ của port1 dùng 8 LED:

Như đã giới thiệu trong phần I: Port 1 của 8951 là port I/0 trên các chân từ 1÷8. Các chân có kí hiệu $P_{1.0}$ ÷$P_{1.7}$ là port chỉ có một chức năng là giao tiếp với thiết bị bên ngoài. Để quan sát được trạng thái làm việc xuất nhập của port1 trong quá trình giao tiếp ta thiết kế hệ thống 8 LED, các LED này được xấp xếp từ trên xuống ứng với thứ tự của các bit trên port1. Các cathode của 8LED được nối chung với nhau và nối xuống mass, anode của các LED đưa đến port1 của 8951 thông qua một bộ đệm sử dụng hai IC 74244.

- Tính toán các giá trị điện trở hiển thị:



Để một đọan LED đủ sáng thì dòng trung bình I_{tb} qua mỗi đoạn LED bằng 15 mA, điện áp rơi trên hai cực của LED là V_γ = 1.8v.

Transistor thiết kế ở trạng thái bão hoà $\Rightarrow V_{sat}$ = 0.2v, do đó diện trở hạn dòng là:

$$Rc = \frac{Vcc - V\gamma - Vsat}{Itb \times 8} = \frac{5v - 1.8v - 0.8v}{0.015 \times 8} = 24\Omega$$

Vậy ta chọn Rc= 22Ω

Vì mỗi LED có 7 đoạn nên giá trị cực đại qua transitor là: Itb=Ic=15mA*8=120mA nên ta có thể chọn transitor làA564 có các thông số sau βmin=85.

Ic=1A thoả điều kiện thiết kế:

+ Tính R_B (điện trở phân cực cho transitor):

Ic=120 mA => I_B=120/βmin=120/85=1,5 mA

Để transitor dẫn bảo hoà thì $I_B \geq$ 1,5 mA

$V_{BE} \leq$ -0,7 v

Mức thấp cuả ngõ ra TTL thường vào khoảng 0,5v và do đó ta chọn:

- Ta chọn R_B = 3.3 kΩ

Vậy các thông số chọn là:

- LED loại Anode chung
- Transitor loại A562(PNP)
- Điện trở hạn dòng Rc=22Ω
- Điện trở phân cực R_B=3k3

Chương II:

THI CÔNG HỆ THỐNG

I.)THIẾT KẾ MẠCH IN:

Đây là giai đoạn khá quan trọng trong toàn bộ tiến trình thi công bởi vì một sự cố hư hỏng nhỏ sau này hay xấu hơn là phải làm lại từ đầu rất có thể do những sai sót trong giai đoạn này, chính vì thế thời gian dành cho việc thiết kế và thi công mạch in chiếm gần ½ tổng số thời gian thi công hệ thống.

Vì cấu trúc mạch phức tạp nên người thực hiện dùng loại mạch in hai mặt. Để thực hiện sơ đồ nối dây người thực hiện đã sử dụng phần mềm vẽ mạch để thiết kế

Với chế độ tự động thời gian hoàn tất tất cã mọi đường nối có khả năng trên mạch không tới 20 phút. Nhưng trước nhất phải mất nhiều thời gian để sửa file trước khi chạy chế độ vẽ mạch in tự động, công việc tiếp sau khi đã hoàn tất việc nối dây là chuyển tất cã những mối hàn lên bề mặt gần linh kiện xuống dưới. Đây là công đoạn tốn nhiều thời gian. Sau khi hoàn tất, bản thiết kế được đem đi gia công để hình thành mạch in theo yêu cầu.

II.) KIỂM TRA:

Gồm 2 phần là mạch in và linh kiện.

- Sau khi thiết kế và gia công mạch in nhiệm vụ tiếp theo là hãy kiểm tra toàn bộ mạch trước khi lắp linh kiện vào. Bởi kit thiết kế tương đối phức tạp nên phần mạch in gia công chắc không tránh khỏi những lỗi tuy là nhỏ.
- Ngay từ đầu khi phần thiết kế cuả mạch được đặt ra, em đã liệt kê tất cả các linh kiện trong mạch và chuẩn bị đi mua.
- Trước khi lắp ráp một lần nửa em phải kiểm tra lại từng linh kiện một lần nữa để tránh sự thiếu xót và nhầm lẫn.

III).LẮP RÁP MẠCH IN:

Qua kinh nghiệm của người đi trước cũng như những vấp phải mà em đã từng gặp trong các mạch điện phức tạp, nếu lắp ráp luôn một loạt xong rồi mới kiểm tra lại là một điều tối kỵ. Những sai sót nhỏ cũng có thể làm ta rối tung lên cho nên em đã lắp ráp từng phần vào và kiểm tra ngay bằng các thiết bị đo ngay trước khi lắp các phần khác. Thứ tự lắp ráp như sau:

1. Lắp mạch dao động sau đó dùng máy hiện sóng để kiểm tra biên độ cũng như tần số dao động bởi đây là thông số quan trọng.
2. Lắp mạch Reset và kiểm tra bằng cách dò mức logic khi ấn.
3. Lắp vi điều khiển vào kiểm tra chân ALE cuả vi điều khiển có xung chưa.
4. Lắp ROM và RAM vào mạch.
5. Lắp bộ hiển thị.
6. Lắp bộ phận giao tiếp ngoại vi và socket ngõ ra.
7. Lắp mạch quét phím-hiển thị và kiểm tra lại từng p`ím nhấn.
8. Lắp nguồn nuôi RAM.

IV.) GIAI ĐOẠN HÀN CHÌ:

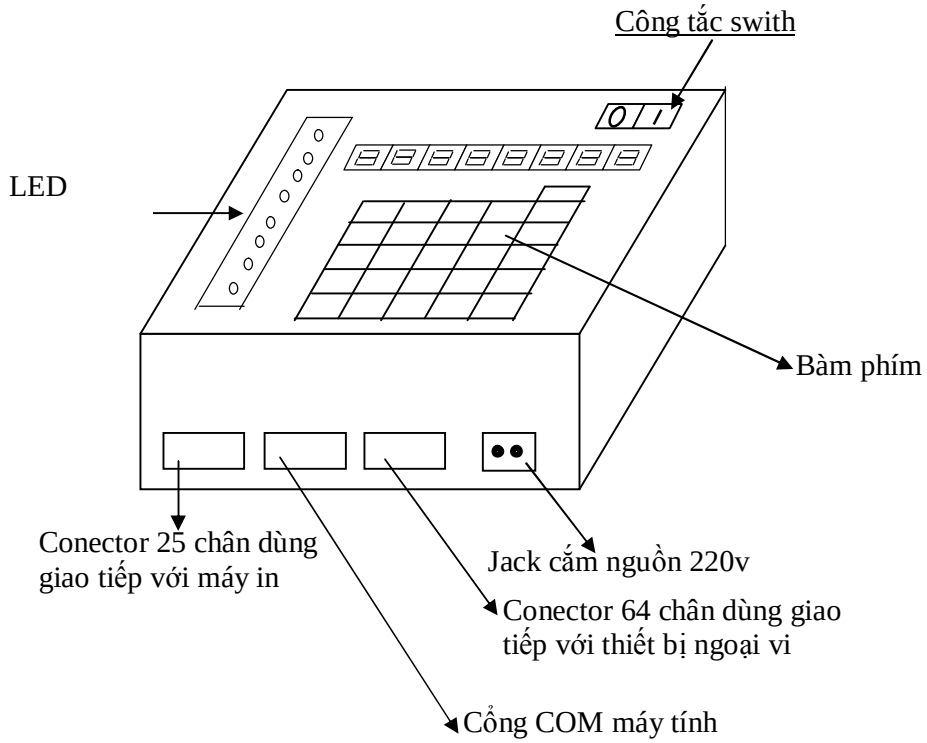
Chia làm 3 công đoạn nhỏ:

1. Trước tiên hoàn tất tất cã những lỗ xuyên mạch, đây là những chỗ mà các đường dây chạy trên một mặt tạm dừng để chuyển sang mặt kia.
2. Sau khi hoàn tất các lỗ xuyên mạch người thực hiện kiểm tra laị thông mạch các đường nối cũng như các lỗ xuyên mạch.

3. Công đoạn cuối cùng là hàn linh kiện, do hầu hết các linh kiện sử dụng là loại IC Cmos rất dễ hư hỏng bởi nhiệt nên người thực hiện chọn giải pháp thực hiện chân đế cắm IC nhằm dễ dàng thay đổi linh kiện khi gặp sự cố. Với Bàn Phím Sử Dụng Bàn Phím Cuả Máy Tính.

V.) HÌNH DẠNG BÊN NGOẠI VÀ CÁCH SỮ DỤNG:

1) Hình Dạng Bên Ngoài Cuả Hệ Thống Kit 8951:



2. Hướng Dẫn Cách Sữ Dụng Kit Vi Điều Khiển:

Các bước tiến hành:

- Cấp nguồn cho hệ thống bằng cách cấm jack cắm vào nguồn 220v. Sau đó bật công tắc swith và quan sát trên màn hình LED 7 đoạn nếu thấy có 4 LED bên phải màn hình sáng tức là hệ thống đã được cấp nguồn.
- Thao tác trên bàn phím.

Kit vi điều khiển có tất cã 26 phím chia làm các nhóm sau:

- 16 phím nhập dữ liệu cuả chương trìn` dạng số thập phân từ 0 đến F.
- 8 phím chức năng và một phím Reset.

2.1) chức năng cuả phím Q:

+ Khi mới cắm điện cho máy 4 LED bên trái sẽ hiển thị 4 số 0000 4 LED bên phải tắt.

+ Nếu không hiển thị đúng hảy nhấn phím “Q” để reset lại, khởi động laị hoạc muốn thoát khi muốn thóat khỏi chương trình vi điều khiển đang thực hiện.

2.2) chức năng cuả phím S:

- Muốn nhập dữ liệu mới vào ô nhớ có địa chỉ ví dụ 4000 hãy dùng các phím nhập dữ liệu đánh 4000, địa chỉ này sẽ xuất hiện ở 4 LED bên phải.
- Nhấn phím "S" thì địa chỉ 4000 sẽ thay thế địa chỉ trước đó cuả 4 LED bên phải.
- 4 LED còn lại chỉ có 2 LED sáng đó chính là nội dung cuả ô nhớ tương ứng với địa chỉ cuả 4 LED bên trái.

2.3) chức năng phím "↑":

Dùng để lưu trữ dữ liệu vào ô nhớ có địa chỉ ở 4 LED bên trái, ví dụ muốn lưu trữ dữ liệu có địa chỉ là "3F" vào ô nhớ có địa chỉ là 4000 ta đánh "3F" từ các phím dữ liệu, dữ liệu mơđi "3F" sẽ thay thế các dữ liệu trước đó.

Sau đó nhấn phím" ↑" để lưu dữ liệu này vào ô nhớ 4000 để sẵn sàng nhận dữ liệu tiếp theo và hai LED bên phải hiển thị nội dung cuả ô nhớ 4000.

Chức năng cuả phím này là lưu trữ dữ liệu đồng thời tăng địa chỉ cuả ô nhớ.

2.4) chức năng của phím "↓":

- Có chức năng giảm địa chỉ cuả ô nhớ xuống 1 đơn vị tương ứng với mỗi lần nhấn. Ví dụ muốn tra lại ô nhớ mới vừa nhập là 4000 xem có đúng dữ liệu vừa nhập là "3F" không ta ấn phím "↓", nếu sai thì nhập lại, nếu đúng thì nhấn phím tăng địa chỉ để nạp dữ liệu tiếp theo.

2.5) chức năng của phím"P":

Sau khi nhập dữ liệucuả một chương trìng tại địa chỉ 4000, để vi điều khiển thực hiện chương trình này hảy nhấn phím "P" khi đó màn hình 8 LED sẽ xuất hiện"PC4000" sau đó nhấn phím G chương trình sẽ thi hành.

Nếu chương trình lưu tại địa chỉ khác 4000 thì trước khi ấn phím tăng địa chỉ hãy đánh địa chỉ cuả chương trình đó vào bằng các phím nhập dữ liệu sau đó nhấn phím tăng địa chỉ, ví dụ muốn thực hiện chương trình tại địa chỉ 5000 thì trên màn hình LED sẽ hiển thị chữ PC=5000. Nhấn phím "G" chương trình sẽ thi hành tại địa c`ỉ 5000.

2.6) chức năng của phím"R":

Dùng để xem nội dung các thanh ghi trước tiên nhấn các phím thập phân tương đương từ 6 đến F.

2.7) chức năng của phím"I":

Phím này sẽ tác động đến ngắt đóng cuả hệ thống vi xữ lý chương trình sẽ bị ngưng sau khi nhấn phím "I", nếu nhấn phím "I" thêm một lần nữa hệ thống sẽ được trở lại trạng thái mặc định tương đương với Reset máy bằng phím Q.

2.8) chức năng của phím"T":

Chức năng cuả phím này là thực hiện chương trình từng bước. Trình tự nhấn phím giống như nhấn phím "G". Nếu nhấn phím "G" để thực hiện chương trình tại địa chỉ chứa trong các thanh ghi PC ta nhấn phím "T" chương trình sẽ thực hiện từng lệnh tại địa chỉ chứa PC.

Sau đây là các sơ đồ nguyên lý, sơ đồ bố trí linh kiện, mạch in mặt trước và sau.

Phần III:

THIẾT KẾ PHẦN MỀM

Trong hệ thống vi điều khiển, phần cứng được xem như thể xác còn phần mềm được xem như linh hồn chi phối toàn bộ hgạt động. Khả năng làm việc cuả hệ thống linh hoạt hay không chính là ở đây. Trong chương trình sẽ trình bày khái quát cách viết một chương trình cùng các vấn đề liên quan.

Chương I : MÔ TẢ CHƯƠNG TRÌNH PHẦN MỀM

Để viết một chương trình có nhiều cách ví dụ như viết một mạch từ trên xuống dưới theo cách này CPU sẽ đọc từng tự theo các chỉ thị trong chương trình từ địa chĩ thấp đến địa chỉ cao và thực hiện chúng cho đến địa chỉ cuối cùng. Trong trường hợp này người đọc rất dễ theo dõi chương trình và nắm được ý đồ cuả người viết, tuy nhiên nó có nhược điểm là kích thước chương trình lớn. Giới hạn cuả phương pháp lập trình tuần tự làm phát sinh một phương pháp lập trình khác là lập trình cấu trúc. Trong chương trình này với những đoạn thường xuyên lặp lại trong chương trình người ta đem chúng ra khỏi chương trình chính chúng có thể được đặt ở đầu hoặc cuối chương trình chính (tuỳ theo phần mềm). Tại một địa chỉ xác định tại nơi chúng ta đem đi được thay bằng lệnh LCALLxxxx. Trong đó xxxx là địa chỉ chúng ta đặt chương trình được đem đi. Khi gặp chỉ thị này CPU sẽ nhảy đến chỉ thị được đặt sau chỉ thị LCALL và thi hàfh đoạn chương trình đó. Để quá trình làm việc không bị gián đoạn ở cuối đoạn chương trình ta đặt chỉ thị RET(Return). Khi gặp chỉ thị này CPU sẽ quay về chương trình chính và tiếp tục công việc bị bỏ dở. Phương pháp này khá hiệu quả trong việc giảm kích thước chương trình. Tuy nhiên nó làm cho người sử dụng khó theo dõi chương trình do mất tính liên tục. Để khắc phục nhược điểm này người ta đặt cho mỗi đoạn chương trình như thế một cái tên hay nhãn (Label). Tên đặt phải gíup người đọc hình dung chức năng nhớ rằng đoạn chương trình này có tác dụng dừng chương trình chính trong một khoảng thời gian t nào đó và chúng ta cũng qui định với đoạn chương trình được gọi là nơi chương trình chính đặt dữ liệu xử lý cũng như nơi chương trình chính sẽ lấy kết quả về bằng cách này người đọc chỉ còn nhớ đoạn chương trình được gọi sẽ làm công việc gì và nơi đặt dữ liệu có liên quan.

Phần mềm phục vụ cho hệ thống cuả chúng ta củng được thiết kế dựa trên quan điểm này. Để viết chương trình điều khiển hệ thống có thể dùng một trong các ngôn ngữ như Assembler, passcal, C… ở đây người thiết kế viết chương trình bằng ngôn ngữ assembler của hệ thống 8 bit dùng 8951. Pascal hay C đều có thể sử dụng để viết chương trình. Tuy nhiên khi dịch ra mã máy sẽ chiếm nhiều bộ nhớ chúng không minh họa được khả năng sử dụng các chỉ thị của 8951 trong việc tạo cho hệ thống các chức năng thay thế mạch số.

I.THUẬT GIẢI:

Là cách giải quyết vấn đề bằng những thao tác cụ thể được sắp xếp theo một trình tự nhất định.

Trong kỹ thuật máy tính, thuật ngữ là cốt lõi mang tính sáng tạo việc lập trình. Thuật giải thường đi kèm với tổ chức dữ liệu, bản thân thuật giải là một chuyên ngành được nghiên cứu chuyên sâu và luôn phát triển.

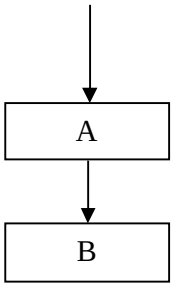
II.THAO TÁC:

Còn gọi là tác vụ. Lệnh hoặc chỉ thị là một hành động cần được thực hiện bởi cơ chế thực hiện thuật giải thao tác được diễn giải bởi một nhóm từ mà chủ yếu là một động từ, cần chọn động từ chỉ dẫn chính xác và xúc tích

Mỗi một thao tác cần một thời gian và tiêu hag vật chất để thực hiện, thời gian và tiêu hao tuỳ thuộc vào từng thao tác. Mỗi thao tác có thể phân thành các thao tác nhỏ. Vấn đề là chọn thao tác ở mức độ chi tiết nào để trình bày là hợp lý nhất. Nếu thao tác tổng quát thì sẽ khó hiểu ngược lại nếu quá chi tiết thì sẽ rắc rối dễ nhầm. Cần chọn thao tác ở mức độ tổng quát nhất mà đối tượng sử dụng có thể hiểu được.

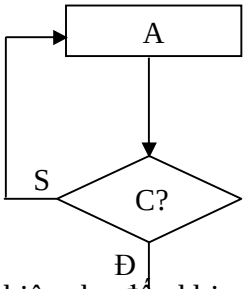
Cùng một thao tác nhưng sắp đặt theo trình tự khác nhau sẽ cho kết quả khác nhau. Cơ cấu trình tự cần thể hiện trong thao tác gọi là cấu trúc điều khiển.

Cấu trúc tuần tự:

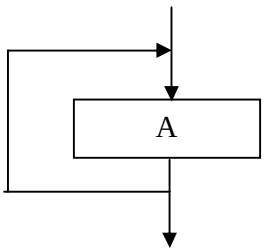


Cấu trúc lặp:

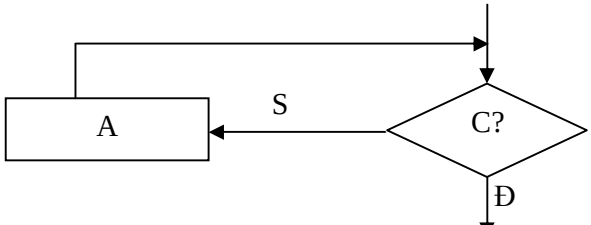
Lặp laị A cho đến khi điều kiện C đúng, thao tác phải thực hiện ít nhất một lần.



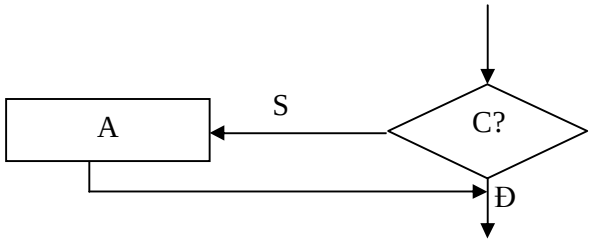
Lặp lại A vô điều kiện cho đến khi có lệnh thoát.



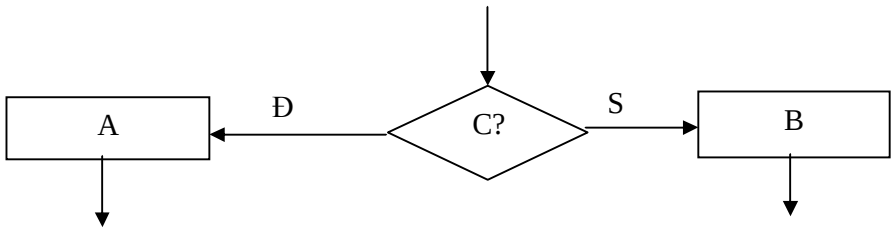
Nếu điều kiện C đúng thì thực hiện A (tuỳ theo giá trị cuả C thao tác A có thể thực hiện 0,1, hoặc nhiều lần)



Cấu trúc lựa:
Chọn thực hiện hoặc hiển thị một thao tác.



Chọn lựa một trong hai thao tác.

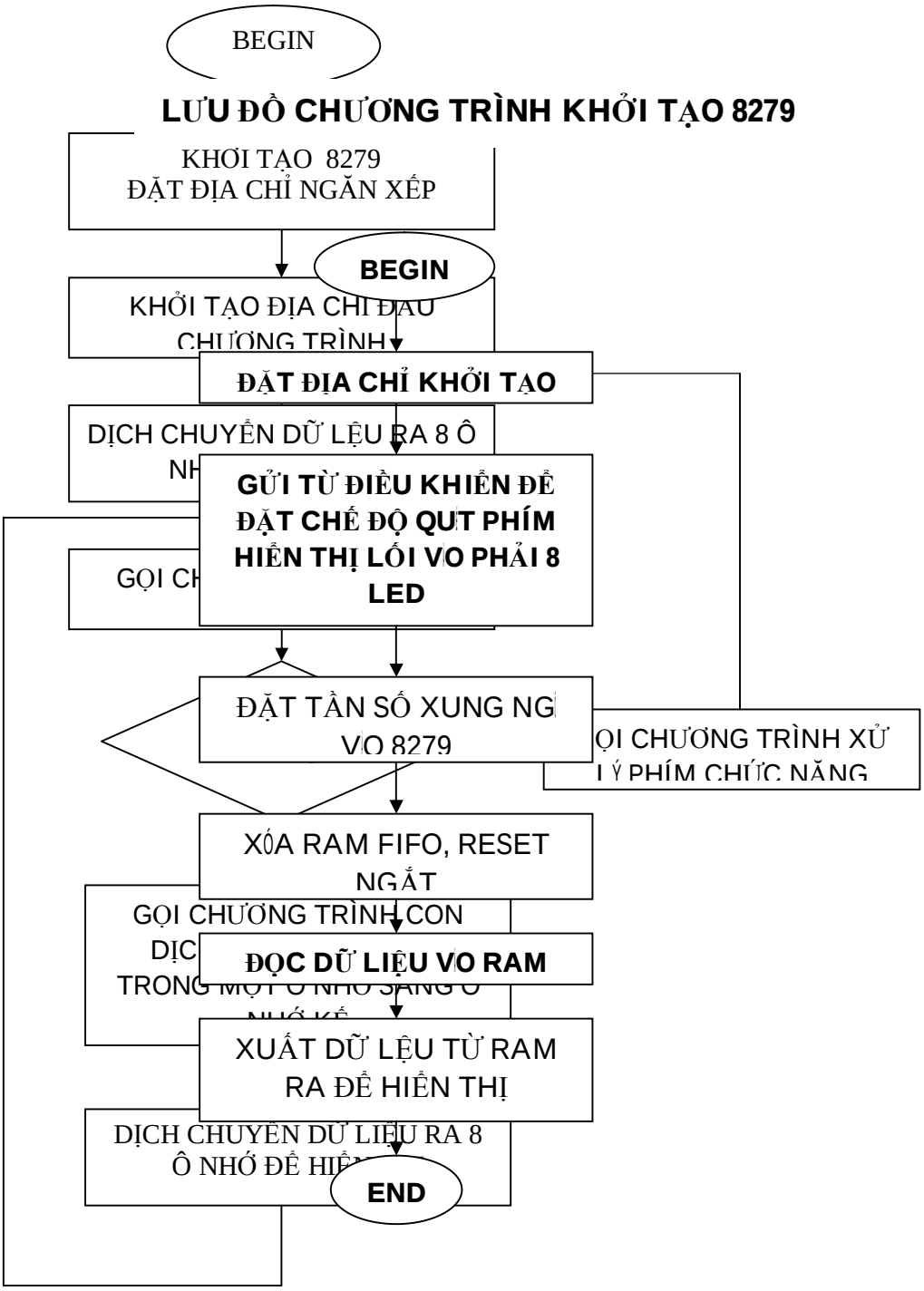


* Tóm lại một thuật giải tối thiểu cần những điều kiện sau:
1. Công việc phải cụ thể và thực hiện được trên máy tính.
2. Số bước thực hiện phải rõ ràng và hữu hạn.
3. Có số liệu vào.
4. Có số liệu ra

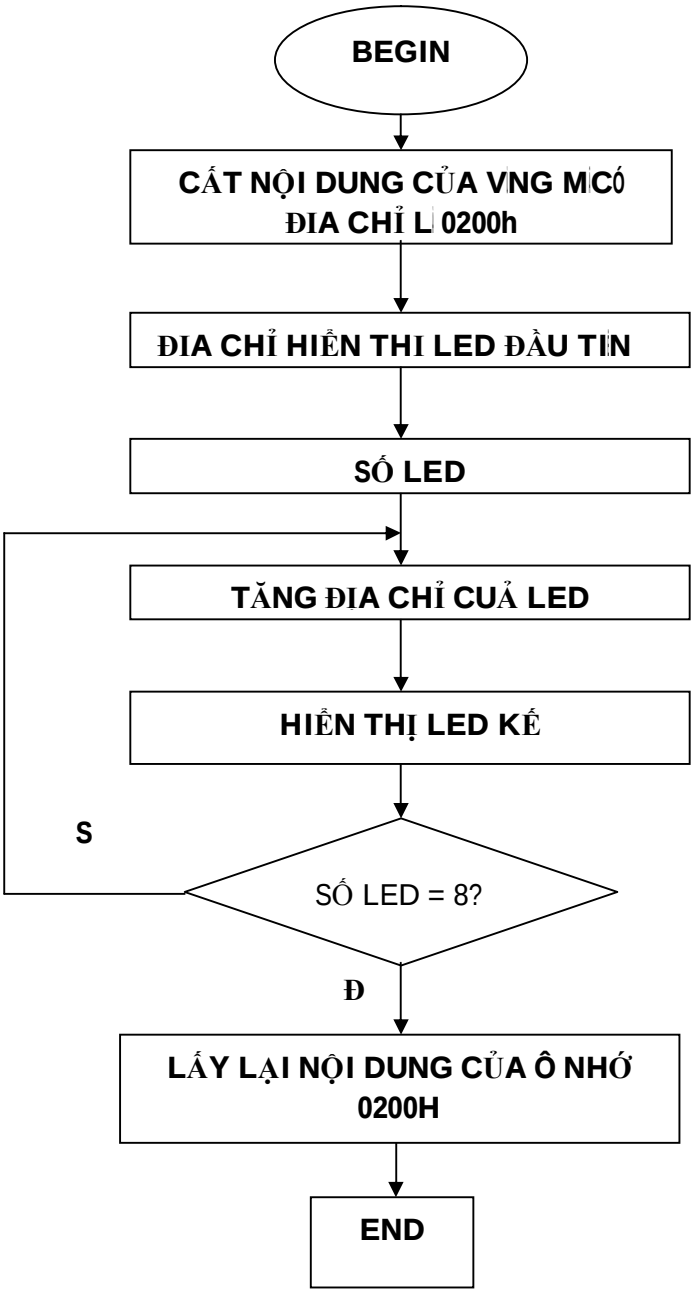
Chương II : XÂY DỰNG CHƯƠNG TRÌNH HỆ THỐNG

I.GIẢI THUẬT CỦA HỆ THỐNG KIT VI ĐIỀU KHIỂN:

LƯU ĐỒ CHƯƠNG TRÌNH CHÍNH ĐIỀU KHIỂN KIT VI ĐIỀU KHIỂN 8951

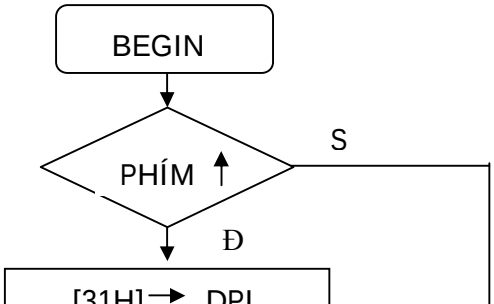


LƯU ĐỒ CHƯƠNG TRÌNH HIỂN THỊ

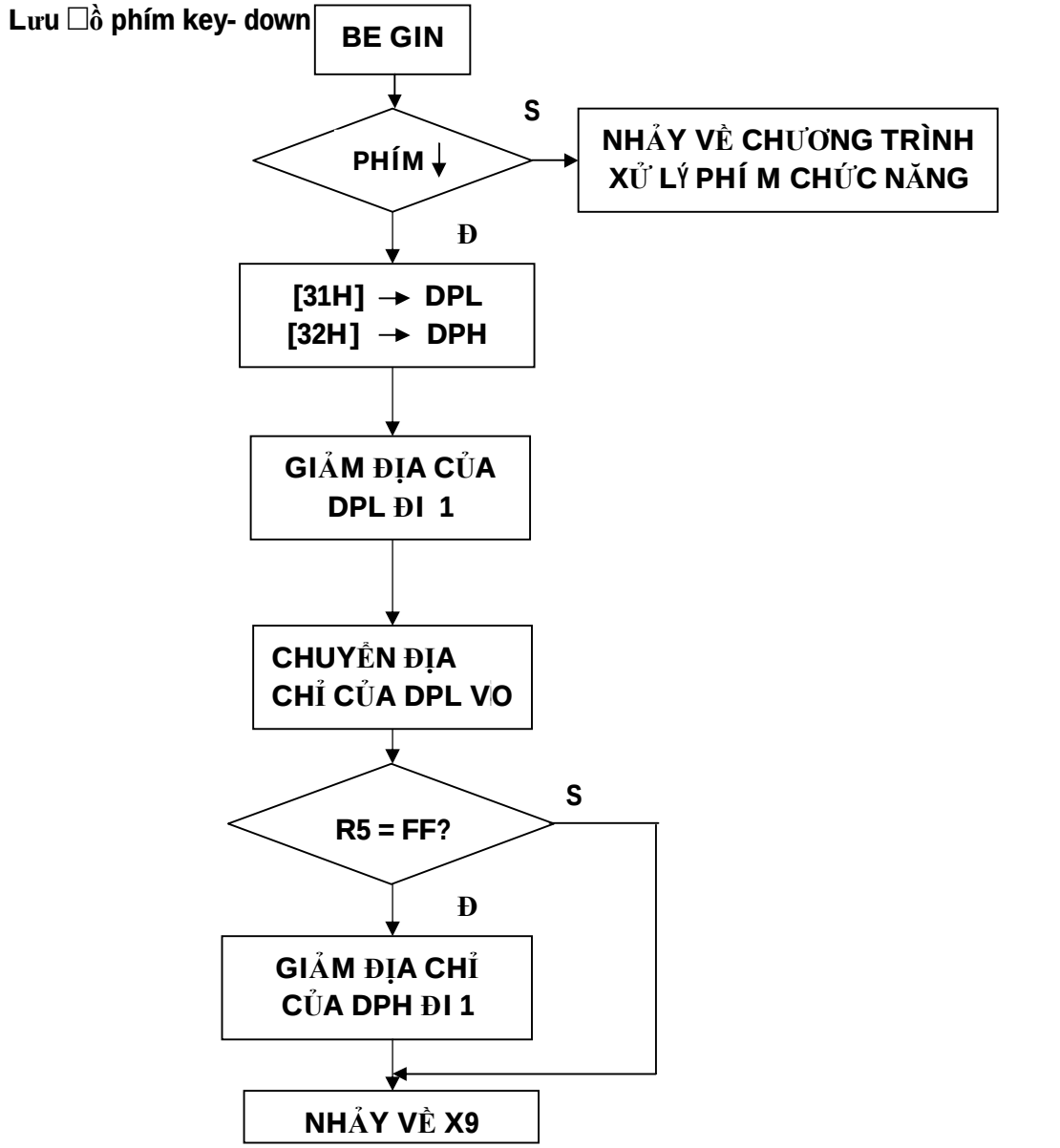


LƯU ĐỒ CHƯƠNG TRÌNH XỬ LÝ PHÍM CHỨC NĂNG

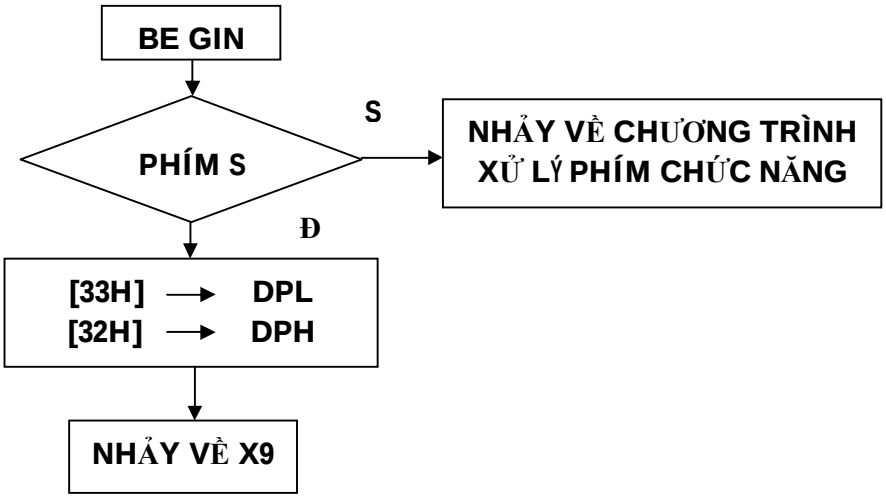
Lưu ồ phím key- up



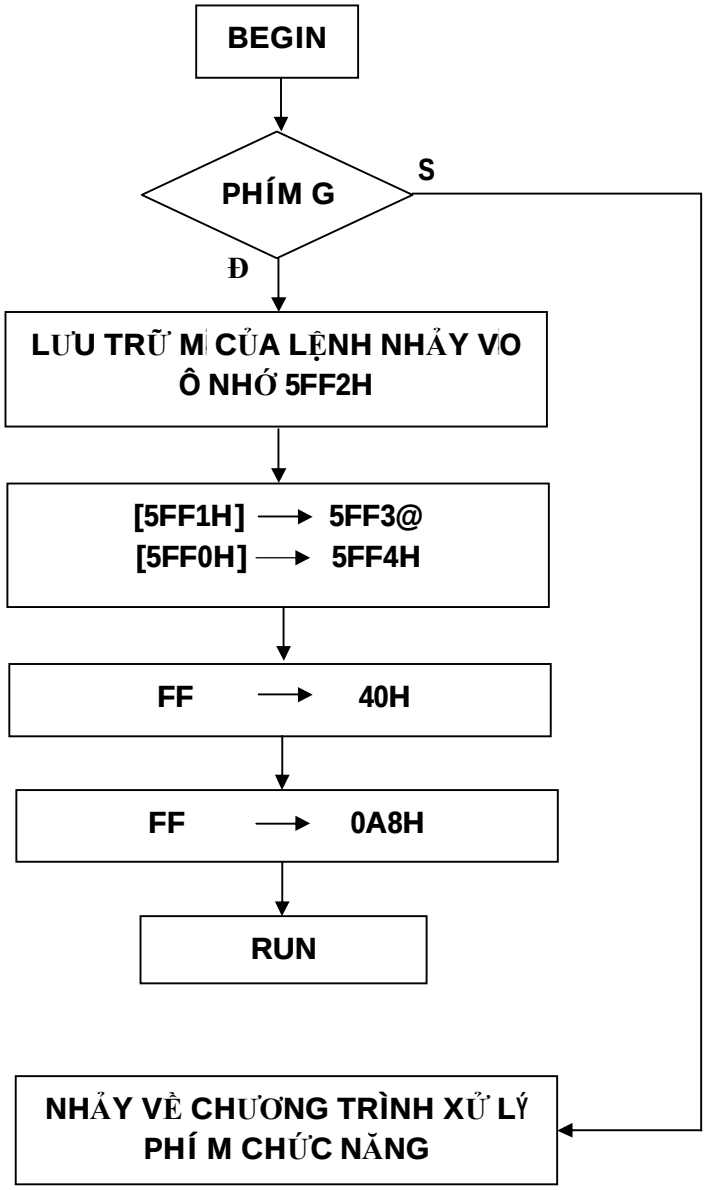
Lưu □ồ phím key- down



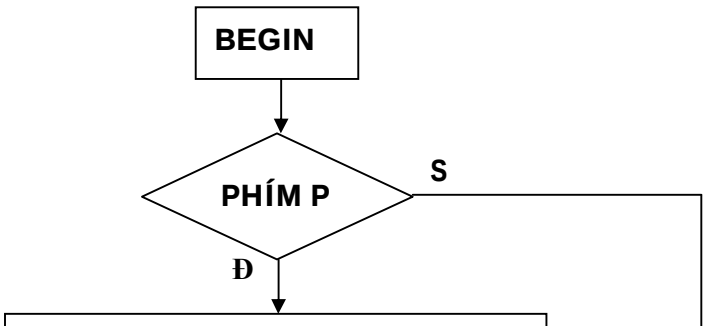
Phím chức năng s



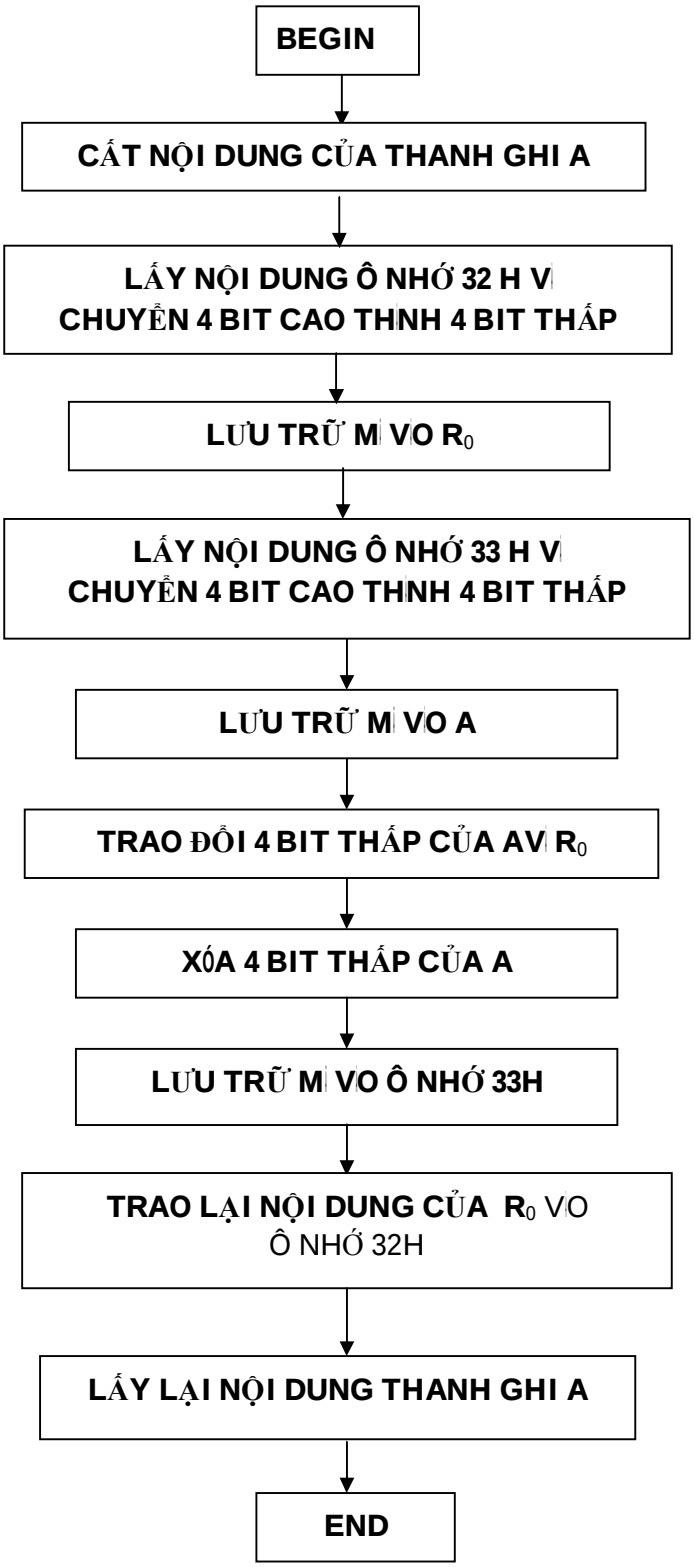
Lưu □ồ phím G



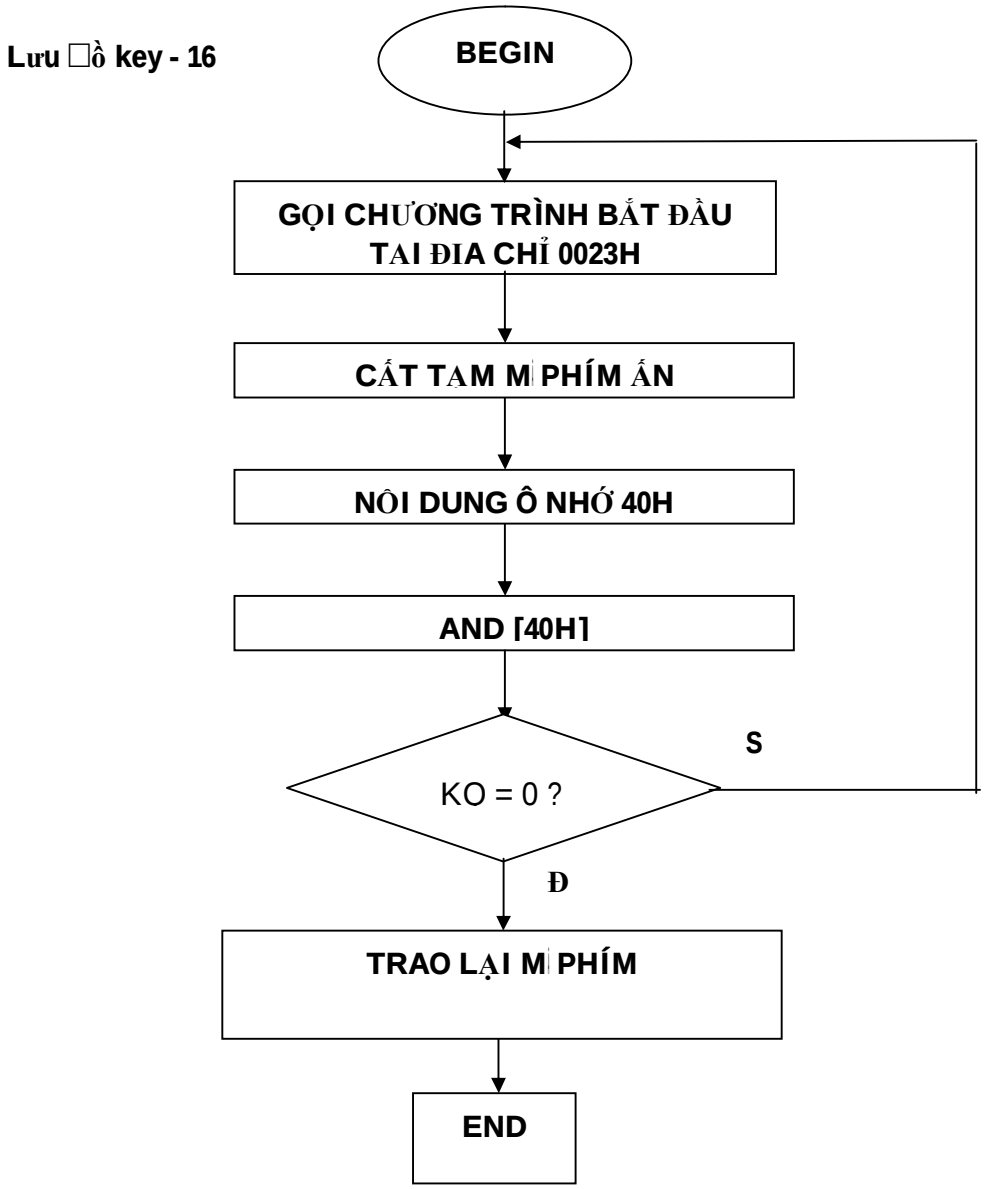
Lưu □ồ phím p



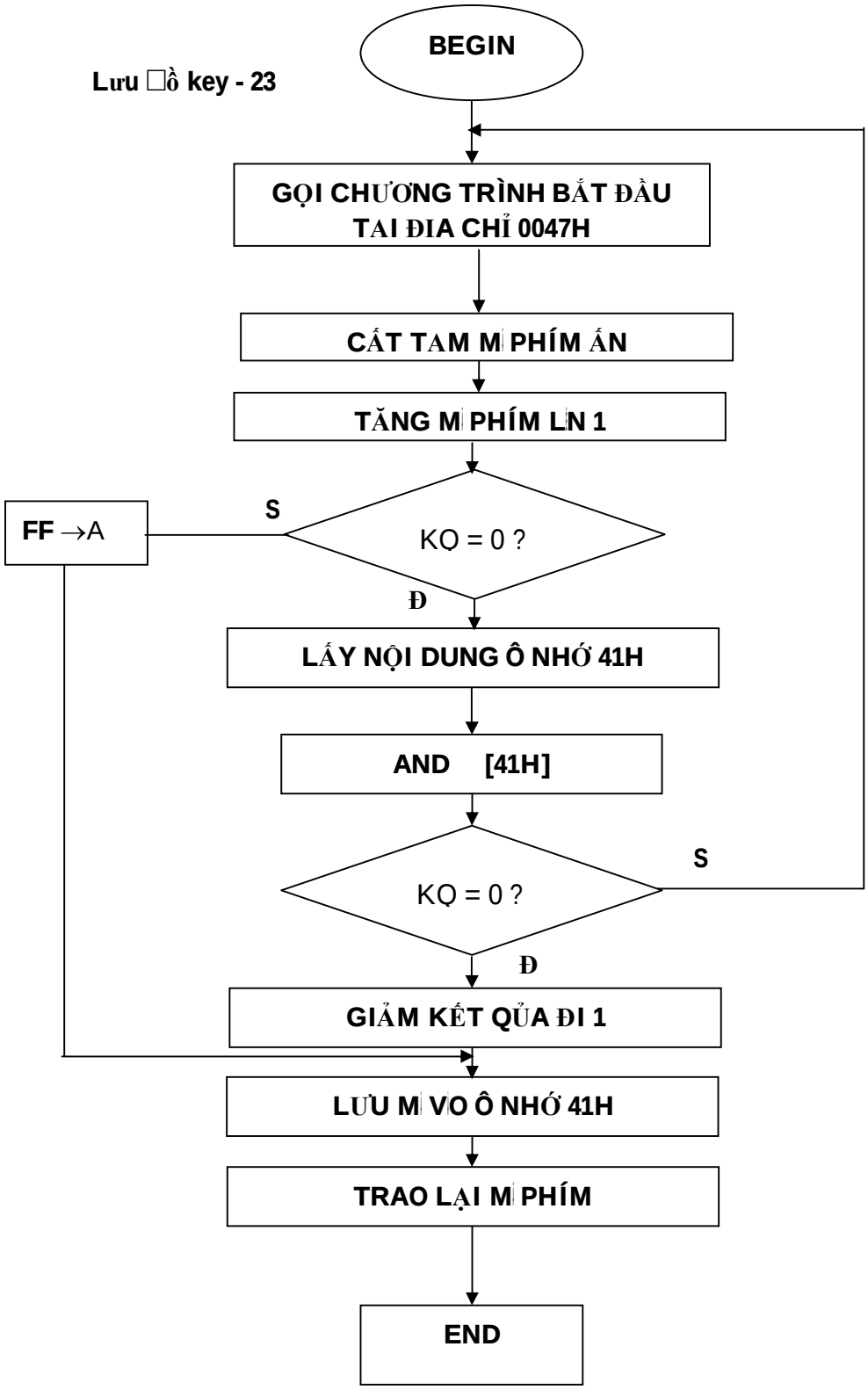
CHƯƠNG TRÌNH CONVERT



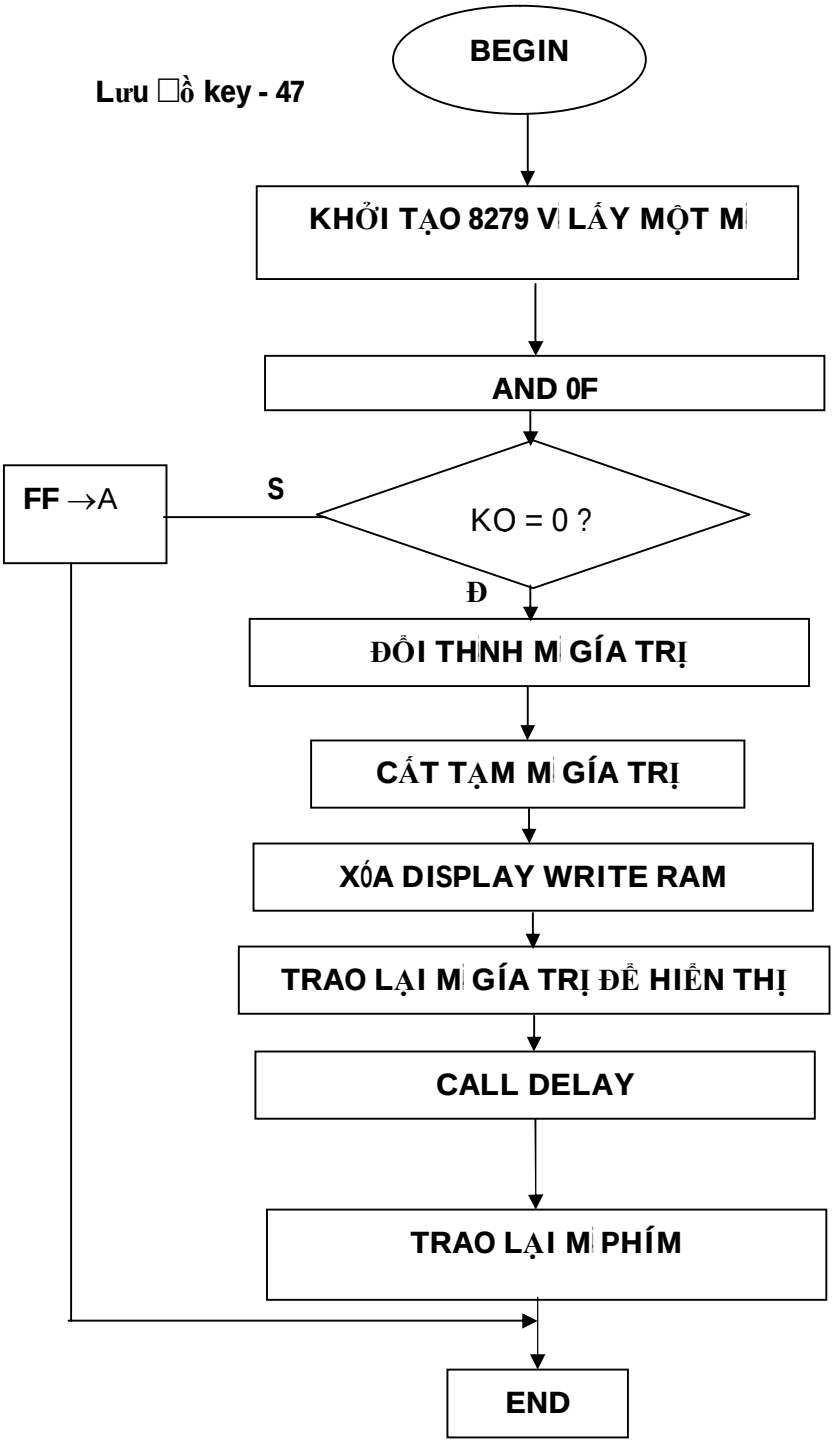
LƯU ĐỒ CHƯƠNG TRÌNH QUT PHÍM



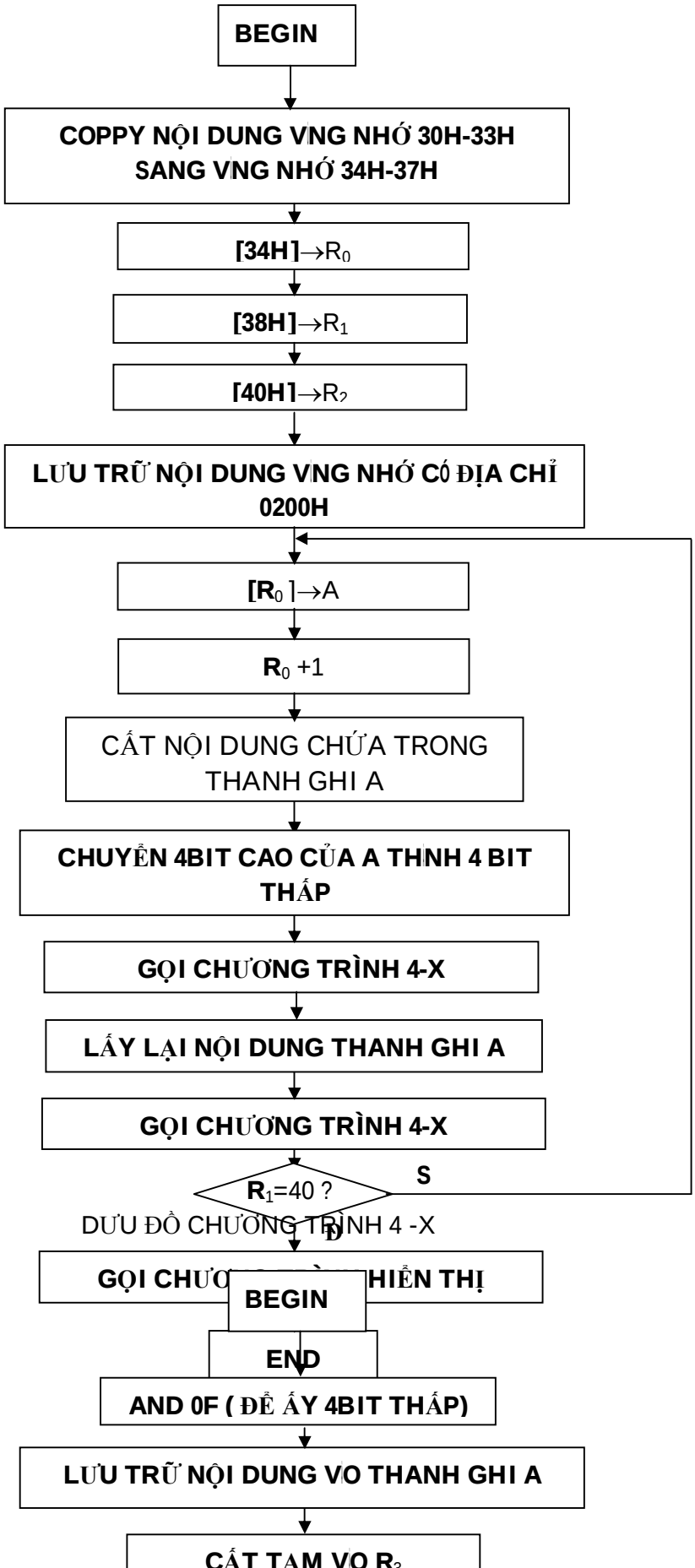
Lưu □ồ key - 23


BEGIN
GỌI CHƯƠNG TRÌNH BẮT ĐẦU TẠI ĐỊA CHỈ 0047H
CẤT TẠM MỜI PHÍM ẤN
TĂNG MỜI PHÍM LÊN 1
KQ = 0 ?
S
FF →A
Đ
LẤY NỘI DUNG Ô NHỚ 41H
AND [41H]
KQ = 0 ?
S
Đ
GIẢM KẾT QỦA ĐI 1
LƯU MỜI VO Ô NHỚ 41H
TRAO LẠI MỜI PHÍM
END

Lưu □ồ key - 47


BEGIN
KHỞI TẠO 8279 V LẤY MỘT M
AND 0F
KO = 0 ?
S
FF →A
Đ
ĐỔI THNH M GÍA TRỊ
CẤT TẠM M GÍA TRỊ
XÓA DISPLAY WRITE RAM
TRAO LẠI M GÍA TRỊ ĐỂ HIỂN THỊ
CALL DELAY
TRAO LẠI M PHÍM
END

CHƯƠNG TRÌNH CHUYỂN DATA TRONG 4 Ô NHỚ SANG 8 Ô NHỚ



DƯU ĐỒ CHƯƠNG TRÌNH 4 -X

II.CHƯƠNG TRÌNH MONITOR

MCS-51 MACRO ASSEMBLER BINH
02/15/:0 PAGE 1

DOS 7.10 (038-N) MCS-51 MACRO ASSEMBLER, V2.2
OBJECT MODULE PLACED IN BINH.OBJ
ASSEMBLER INVOKED BY: C:\TRUNG\ASM51.EXE BINH.ASM

```
LOC  OBJ          LINE     SOURCE

                     1     ; CHUONG TRINH CHINH
0000                 2          ORG 0000H
0000 020033          3          LJMP X1
0033                 4          ORG 0033H
0033 020FFD          5     X1:  LJMP X2
0FFD                 6          ORG 0FFDH
0FFD 758165          7     P2:  MOV SP,#65H
1000 120280          8          LCALL KD79
                                9          ;NAP DIA CHI CHUONG CHAY O 2
                                           O NHO 5FFF0 VA 5FF1H
1003 905FF0          10         MOV DPTR,#5FF0H
1006 7400            11         MOV A,#00H
1008 F0              12         MOVX @DPTR,A
1009 905FF1          13         MOV DPTR,#5FF1H
100C 7440            14         MOV A,#40H
100E F0              15         MOVX @DPTR,A
                     16
    ;NAP 00 VAO VUNG NHO 30H,31H,32H,33H
100F 753300          17         MOV 33H,#00H
1012 753200          18         MOV 32H,#00H
1015 753100          19         MOV 31H,#00H
1018 753000          20         MOV 30H,#00H
                     21
101B 7540F0          22         MOV 40H,#0F0H
                                           ;NAP F0H VAO O NHO 40
101E 1201A1          23         LCALL CON4_8
                     24
1021 120216          25    X4:  LCALL KEY_16
                                           ;THOAT KHI CO  PHIM NHAN
1024 FA              26         MOV R2,A
1025 5410            27         ANL A,#10H
1027 7013            28         JNZ X3
                     ;NHAY NEU LA PHIM CHUC NANG
```

```
1029 120180          29          LCALL CONVERT
102C E540            30          MOV A,40H
                  ;LAY NOI DUNG O NHO 40H
102E 23              31          RL A
102F 44F1            32          ORL A,#0F1H
    ;OR VOI F1,KET QUA LAN DAU LA F1,F3,F7,FF
1031 F540            33          MOV 40H,A
1033 E533            34          MOV A,33H
1035 4A              35          ORL A,R2
1036 1201A1          36          LCALL CON4_8
1039 021021          37          LJMP X4
                  38    ;XU LY PHIM CHUC NANG
103C EA              39    X3:   MOV A,R2
103D 020600          40          LJMP X5
                  41    ;PHIM CHUC NANG
0600                 42          ORG 0600H
0600 B4172D          43    X5:   CJNE A,#17H,X6
0603 853182          44          MOV DPL,31H
               ;NEU DUNG LA PHIM KEY_UP MA 17
0606 853083          45          MOV DPH,30H
0609 E537            46          MOV A,37H
060B F0              47          MOVX @DPTR,A
060C A3              48          INC DPTR
060D E0              49    X9:   MOVX A,@DPTR
060E 858231          50          MOV 31H,DPL
MCS-51 MACRO ASSEMBLER    BINH
02/15/:0   PAGE    2


LOC  OBJ          LINE     SOURCE

0611 858330          51          MOV 30H,DPH
0614 F533            52          MOV 33H,A
0616 7540F3          53          MOV 40H,#0F3H
0619 1201A1          54          LCALL CON4_8
061C 7540F0          55          MOV 40H,#0F0H
061F 753300          56          MOV 33H,#00H
0622 753200          57          MOV 32H,#00H
0625 021021          58          LJMP X4
                  59
0630                 60          ORG 0630H
0630 B41617          61    X6:   CJNE A,#16H,X7
                  ; NEU DUNG LA PHIM KEY_DOWN
0633 853182          62          MOV DPL,31H
0636 853083          63          MOV DPH,30H
0639 1582            64          DEC DPL
```

```
063B AD82          65          MOV R5,82H
063D BDFF02        66          CJNE R5,#0FFH,X8
0640 1583          67          DEC DPH
0642 02060D        68    X8:   LJMP X9
                  ;XU LY GIONG PHIM TANG
                   69
064A               70          ORG 064AH
064A B41513        71    X7:   CJNE A,#15H,X10 ;PHIM CHUC NANG S
064D 853382        72          MOV DPL,33H
0650 853283        73          MOV DPH,32H
0653 02060D        74          LJMP X9
                   75
0660               76          ORG 0660H
0660 B41230        77    X10:  CJNE A,#12H,X11
                  ;NEU DUNG LA PIM CHUC NANG P
0663 758339        78          MOV DPH,#39H   ;MA CHU C
0666 758273        79          MOV DPL,#73H   ;MA CHU P
0669 1206A0        80          LCALL VVV1
066C 753A48        81          MOV 3AH,#48H
066F 1205D0        82          LCALL DISPLAY
0672 905FF0        83          MOV DPTR,#5FF0H
0675 E533          84          MOV A,33H
0677 F0            85          MOVX @DPTR,A
0678 905FF1        86          MOV DPTR,#5FF1H
067B E532          87          MOV A,32H
067D F0            88          MOVX @DPTR,A
067E 7540F0        89          MOV 40H,#0F0H
0681 753300        90          MOV 33H,#00H
0684 753200        91          MOV 32H,#00H
0687 021021        92          LJMP X4
                   93
0693               94          ORG 0693H
0693 020720        95    X11:  LJMP X12
                   96
0720               97          ORG 0720H
0720 B4102D        98    X12:  CJNE A,#10H,X13
0723 905FF2        99          MOV DPTR, #5FF2H
0726 7402          100         MOV A,#02H
          ;NAP MA 02 CUA LENH NHAY VAO O NHO 5FF2H
0728 F0            101         MOVX @DPTR,A
0729 905FF1        102         MOV DPTR,#5FF1H
          ;LAY NOI DUNG CUA O NHO 5FF1H DUA VAO 5FF3H
072C E0            103         MOVX A,@DPTR
072D 905FF3        104         MOV DPTR,#5FF3H
0730 F0            105         MOVX @DPTR,A
```

MCS-51 MACRO ASSEMBLER BINH
02/15/:0 PAGE 3

```
LOC  OBJ          LINE     SOURCE

0731 905FF0         106          MOV DPTR,#5FF0H
    ;LAY NOI DUNG CUA O NHO 5FF0H DUA VAO O NHO 5FF4H
0734 E0             107          MOVX A,@DPTR
0735 905FF4         108          MOV DPTR,#5FF4H
0738 F0             109          MOVX @DPTR,A
0739 7540FF         110          MOV 40H,#0FFH
073C 75A8FF          111          MOV 0A8H,#0FFH
073F 025FF2         112          LJMP RUN
                  113
5FF2                114          ORG 5FF2H
                  115     RUN:
0750                116          ORG 0750H
0750 021021         117     X13:    LJMP X4
                   ;THOAT VE CHUNG TRINH CHINH
  ;CHUONG TRINH CON CONVERT CHUYEN NOI DUNG 2 O NHO
0180                119          ORG 0180H
0180 C0E0           120     CONVERT:PUSH 0E0H  ;CAT A
0182 C000           121          PUSH 00H
0184 E532           122          MOV A,32H
0186 C4             123          SWAP A
0187 F532           124          MOV 32H,A
0189 7832           125          MOV R0,#32H
018B E533           126          MOV A,33H
018D C4             127          SWAP A
018E D6             128          XCHD A,@R0
018F 54F0           129          ANL A,#0F0H
0191 F533           130          MOV 33H,A
0193 8632           131          MOV 32H,@R0
0195 D000           132          POP 00H
0197 D0E0           133          POP 0E0H
0199 22             134          RET
                  135     ;CHUONG TRINH CON AN PHIM
0216                136          ORG 0216H
0216 120223         137     KEY_16:LCALL KEY_23
0219 FA             138          MOV R2,A
021A E541           139          MOV A,41H
021C 5541           140          ANL A,41H
021E 60F6           141          JZ KEY_16
0220 EA             142          MOV A,R2
0221 22             143          RET
```

```
 ; SU DUNG CAC O NHO VA THNH GHI R2,A,41H,DPTR,R6,R7
0223              145           ORG 0223H
0223 120247       146     KEY_23: LCALL KEY_47
0226 FA           147           MOV R2,A
0227 04           148           INC A
0228 600B         149           JZ M1
022A E541         150           MOV A,41H
022C 5541         151           ANL A,41H
022E 70F3         152           JNZ KEY_23
0230 14           153           DEC A
0231 F541         154     KE_1:   MOV 41H,A
0233 EA           155           MOV A,R2
0234 22           156           RET
0235 7AFF         157     M1:     MOV R2,#0FFH
0237 020231       158           LJMP KE_1
0247              159           ORG 0247H
0247 90C001       160     KEY_47:  MOV DPTR,#0C001H
MCS-51 MACRO ASSEMBLER    BINH
02/15/:0   PAGE    4


LOC  OBJ          LINE     SOURCE

024A E0           161           MOVX A,@DPTR
024B 540F         162           ANL A,#0FH
024D 6010         163           JZ M2
024F 900C00       164           MOV DPTR,#0C00H
0252 E0           165           MOVX A,@DPTR
0253 FA           166           MOV R2,A
0254 74C2         167           MOV A,#0C2H
0256 90C001       168           MOV DPTR,#0C001H
0259 F0           169           MOVX @DPTR,A
025A 120270       170           LCALL DELAY
025D EA           171           MOV A,R2
025E 22           172           RET
025F 74FF         173     M2:     MOV A,#0FFH
0261 22           174           RET
0270              175           ORG 0270H
0270 7E30         176     DELAY:  MOV R6,#30H
0272 7FFF         177     DE2:    MOV R7,#0FFH
0274 DFFE         178     DE1:    DJNZ R7,DE1
0276 DEFA         179           DJNR R6,DE2
0278 22           180           RET
  ;CHUONG TRINH CON 01A1H VA 01DDH LA HAI CHUONG TRINH HIEN
THI 4 SO 0 BEN PHAI O
 ;LED DIA CHI VA TAT 4  LED DATA KHI KHOI DONG
```

```
01A1              184           ORG 01A1H
01A1 853034       185     CON4_8: MOV 34H,30H
01A4 853135       186             MOV 35H,31H
01A7 853236       187             MOV 36H,32H
01AA 853337       188             MOV 37H,33H
01AD 7834         189            MOV R0,#34H
01AF 7938         190            MOV R1,#38H
01B1 AA40         191             MOV R2,40H
01B3 900200       192             MOV DPTR,#0200H
01B6 E6           193     C_1:    MOV A,@R0
01B7 08           194           INC R0
01B8 C0E0         195            PUSH 0E0H
01BA C4           196           SWAP A
01BB 1201DD       197             LCALL CON4_X
01BE D0E0         198            POP 0E0H
01C0 1201DD       199             LCALL CON4_X
01C3 70F1         200            JNZ C_1
01C5 8A40         201            MOV 40H,R2
01C7 1205D0       202             LCALL DISPLAY
01CA 22           203           RET
                  204
01DD              205           ORG 01DDH
01DD 540F         206     CON4_X: ANL A,#0FH
01DF F582         207            MOV DPL,A
01E1 E0           208           MOVX A,@DPTR
01E2 FB           209           MOV R3,A
01E3 E540         210            MOV A,40H
01E5 D3           211           SETB C
01E6 33           212           RLC A
01E7 F540         213            MOV 40H,A
01E9 EB           214           MOV A,R3
01EA 4002         215            JC CO_1
MCS-51 MACRO ASSEMBLER    BINH
02/15/:0   PAGE    5


LOC  OBJ          LINE     SOURCE

01EC 7400         216            MOV A,#00H
01EE F7           217     CO_1:   MOV @R1,A
01EF 09           218           INC R1
01F0 7440         219            MOV A,#40H
01F2 C3           220           CLR C
01F3 99           221           SUBB A,R1
01F4 22           222           RET
                  223
```

```
05D0              224          ORG 05D0H
05D0 C083          225     DISPLAY:PUSH DPH
05D2 C082          226          PUSH DPL
05D4 C000          227          PUSH 00H
05D6 C002          228          PUSH 02H
05D8 7838          229          MOV R0,#38H
05DA 900C00         230          MOV DPTR,#0C00H
05DD 7A08          231          MOV R2,#08H
05DF E6            232     DIS_1:  MOV A,@R0
05E0 F0            233          MOVX @DPTR,A
05E1 08            234          INC R0
05E2 DAFB           235          DJNZ R2,DIS_1
05E4 D002          236          POP 02H
05E6 D000          237          POP 00H
05E8 D082          238          POP DPL
05EA D083          239          POP DPH
05EC 22            240          RET
06A0              241          ORG 06A0H
06A0 C083          242     VVV1:   PUSH DPH
06A2 C082          243          PUSH DPL
06A4 905FF0         244          MOV DPTR,#5FF0H
06A7 F0            245          MOVX @DPTR,A
06A8 F533          246          MOV 33H,A
06AA 905FF1         247          MOV DPTR,#5FF1H
06AD E0            248          MOVX A,@DPTR
06AE F532          249          MOV 32H,A
06B0 75400F         250          MOV 40H,#0FH
06B3 1201A1         251          LCALL CON4_8
06B6 D082          252          POP DPL
06B8 D083          253          POP DPH
06BA 858238         254          MOV 38H,DPL
06BD 858339         255          MOV 39H,DPH
06C0 1205D0         256          LCALL DISPLAY
06C3 754000         257          MOV 40H,#00H
06C6 C083          258          PUSH DPH
06C8 C082          259          PUSH DPL
06CA 753600         260          MOV 36H,#00H
06CD 753700         261          MOV 37H,#00H
06D0 0206EB         262          LJMP VVV_6
                  263
06DA              264          ORG 06DAH
06DA 1201A1         265     VVV_7:  LCALL CON4_8
06DD D082          266          POP DPL
06DF D083          267          POP DPH
06E1 858238         268          MOV 38H,DPL
06E4 858339         269          MOV 39H,DPH
```

```
06E7 C083          270          PUSH DPH
MCS-51 MACRO ASSEMBLER    BINH
02/15/:0   PAGE    6


LOC  OBJ          DINE     SOURCE

06E9 C082          271          PUSH DPL
06EB 120216         272     VVV_6:  LCALL KEY_16
06EE B4170F         273           CJNE A,#17H,VVV_5
06F1 D082          274          POP DPL
06F3 D083          275          POP DPH
06F5 853382         276          MOV DPL,33H
06F8 853283         277          MOV DPH,32H
06FB 22            278          RET
                279
0700               280          ORG 0700H
0700 FB            281     VVV_5:  MOV R3,A
0701 5410          282          ANL A,#10H
0703 70E6          283          JNZ VVV_6
0705 120180         284          LCALL CONVERT
0708 EB            285          MOV A,R3
0709 4233          286          ORL 33H,A
070B E540          287          MOV A,40H
070D D3            288          SETB C
070E 33            289          RLC A
070F 540F          290          ANL A,#0FH
0711 F540          291          MOV 40H,A
0713 0206DA         292           LJMP VVV_7
                293     ;CHUONG TRINH CON KHOI TAO 8279
0280               294          ORG 0280H
0280 900C01         295     KD79:   MOV DPTR,#0C01H
0283 7410          296          MOV A,#10H
0285 F0            297          MOVX @DPTR,A
0286 E534          298          MOV A,34H
0288 F0            299          MOVX @DPTR,A
0289 74C3          300          MOV A,#0C3H
028B F0            301          MOVX @DPTR,A
028C 7490          302          MOV A,#90H
028E F0            303          MOVX @DPTR,A
028F 7440          304          MOV A,#40H
0291 F0            305          MOVX @DPTR,A
0292 22            306          RET
                307          END
```

Phần IV: PHỤ LỤC

Chương I : TẬP LỆNH CỦA 8951

I . TÓM TẮT TẬP LỆNH CỦA 8951 :

- Các chương trình được cấu tạo từ nhiều lệnh, chúng được xây dựng logic, sự nối tiếp của các lệnh được nghĩ ra một cách hiệu quả và nhanh, kết quả của chương trình khả quan.

- Tập lệnh họ MSC-51 được sự kiểm tra của các mode định vị và các lệnh của chúng có các Opcode 8 bit. Điều này cung cấp khả năng 2^8= 256 lệnh được thi hành và một lệnh không được định nghĩa. Vài lệnh có 1 hoặc 2 byte bởi dữ liệu hoặc địa chỉ thêm vào Opcode. Trong toàn bộ các lệnh có 139 lệnh 1 byte, 92 lệnh 2 byte và 24 lệnh 3 byte.

1. *Các mode định vị* (addressing mode):

- Các mode định vị là một bộ phận thống nhất của tập lệnh. Chúng cho phép định rõ nguồn hoặc nơi gởi tới của dữ liệu ở các đường khác nhau tùy thuộc vào trạng thái của người lập trình. 8951 có 8 mode định vị được dùng như sau:

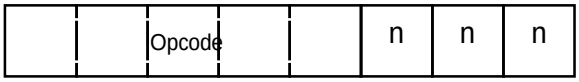
- Thanh ghi.
- Trực tiếp.
- Gián tiếp.
- Tức thời.
- Tương đối.
- Tuyệt đối.
- Dài.
- Định vị.

1.1 Sự định vị thanh ghi (Register Addressing) :

- Có 4 dãy thanh ghi 32 byte đầu tiên của RAM dữ liệu trên Chip địa chỉ 00H ÷ 1FH, nhưng tại một thời điểm chỉ có một dãy hoạt động các bit PSW3, PSW4 của từ trạng thái chương trình sẽ quyết định dãy nào hoạt động.

- Các lệnh để định vị thanh ghi được ghi mật mã bằng cách dùng bit trọng số thấp nhất của Opcode lệnh để chỉ một thanh ghi trong vùng địa chỉ theo logic này. Như vậy 1 mã chức năng và địa chỉ hoạt động có thể được kết hợp để tạo thành một lệnh ngắn 1 byte như sau :

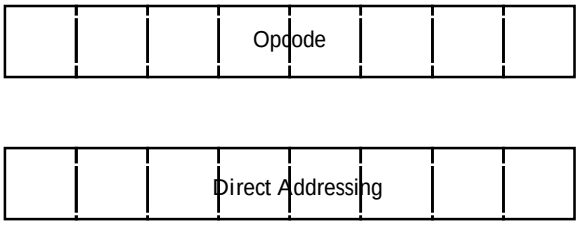
Register Addressing.



- Một vài lệnh dùng cụ thể cho 1 thanh ghi nào đó như thanh ghi A, DPTR . . . mã Opcode tự nó cho biết thanh ghi vì các bit địa chỉ không cần biết đến.

1.2 Sự định địa chỉ trực tiếp (Direct Addressing) :

- Sự định địa chỉ trực tiếp có thể truy xuất bất kỳ giá trị nào trên Chip hoặc thanh ghi phần cứng trên Chip. Một byte địa chỉ trực tiếp được đưa vào Opcode để định rõ vị trí được dùng như sau :

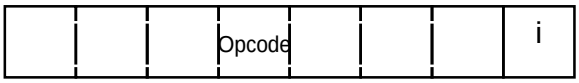


- Tùy thuộc các bit bậc cao của địa chỉ trực tiếp mà một trong 2 vùng nhớ được chọn. Khi bit 7 = 0, thì địa chỉ trực tiếp ở trong khoảng 0÷127 (00H÷7FH) và 128 vị trí nhớ thấp của RAM trên Chip được chọn.

- Tất cả các Port I/O, các thanh ghi chức năng đặc biệt, thanh ghi điều khiển hoặc thanh ghi trạng thái bao giờ cũng được quy định các địa chỉ trong khoảng 128÷255 (80÷FFH). Khi byte địa chỉ trực tiếp nằm trong giới hạn này (ứng với bit 7 = 1) thì thanh ghi chức năng đặc biệt được truy xuất. Ví dụ Port 0 và Port 1 được quy định địa chỉ trực tiếp là 80H và 90H, P0, P1 lađ dạng thức rút gọn thuật nhớ của Port, thì sự biến thiên cho phép thay thế và hiểu dạng thức rút gọn thuật nhớ của chúng. Chẳng hạn lệnh : MOV P1, A sự biên dịch sẽ xác định địa chỉ trực tiếp của Port 1 là 90H đặt vào hai byte của lệnh (byte 1 của port 0).

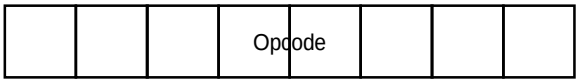
1.3 Sự định vị địa chỉ gián tiếp (Indirect Addressing):

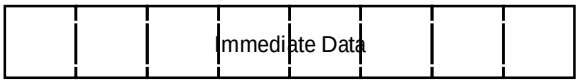
- Sự định địa chỉ gián tiếp được tượng trưng bởi ký hiệu @ được đặt trước R0, R1 hay DPTR. R0 và R1 có thể hoạt động như một thanh ghi con trỏ mà nội dung của nó cho biết một địa chỉ trong RAM nội ở nơi mà dữ liệu được ghi hoặc được đọc. Bit có trọng số nhỏ nhất của Opcode lệnh sẽ xác định R0 hay R1 được dùng con trỏ Pointer.



1.4. Sự định địa chỉ tức thời (Immediate Addressing):

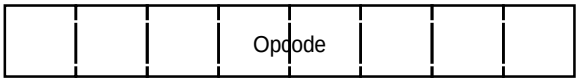
- Sự định địa chỉ tức thời được tượng trưng bởi ký hiệu # được đứng trước một hằng số, 1 biến ký hiệu hoặc một biểu thức số học được sử dụng bởi các hằng, các ký hiệu, các hoạt động do người điều khiển. Trình biên dịch tính toán giá trị và thay thế dữ liệu tức thời. Byte lệnh thêm vô chứa trị số dữ liệu tức thời như sau:

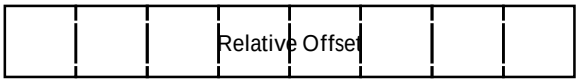




1.5 Sự định địa chỉ tương đối:

- Sự định địa chỉ tương đối chỉ sử dụng với những lệnh nhảq nào đó. Một địa chỉ tương đối (hoặc Offset) là một giá trị 8 bit mà nó được cộng vào bộ đếm chương trình PC để tạo thành địa chỉ một lệnh tiếp theo được thực thi. Phạm vi của sự nhảy nằm trong khoảng -128 ÷ 127. Offset tương đối được gắn vào lệnh như một byte thêm vào như sau:



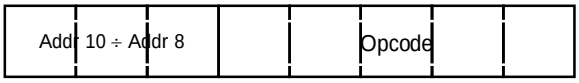


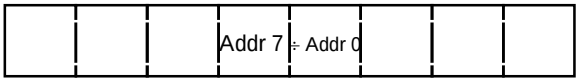
- Những nơi nhảy đến thường được chỉ rõ bởi các nhãn và trình biên dịch xác định Offset Relative cho phù hợp.

- Sự định vị tương đối đem lại thuận lợi cho việc cung cấp mã vị trí độc lập, nhưng bất lợi là chỉ nhảy ngắn trong phạm vi -128÷127 byte.

1.6 Sự định địa chỉ tuyệt đối (Absolute Addressing) :

- Sự định địa chỉ tuyệt đối được dùng với các lệnh ACALL và AJMP. Các lệnh 2 byte cho phép phân chia trong trang 2K đang lưu hành của bộ nhớ mã của việc cung cấp 11 bit thấp để xác định địa chỉ trong trang 2K (A0÷A10 gồm A10÷A8 trong Opcode và A7÷A0 trong byte)và 5 bit cao để chọn trang 2K (5 bit cao đang lưu hành trong bộ đếm c`ương trình là 5 bit Opcode).

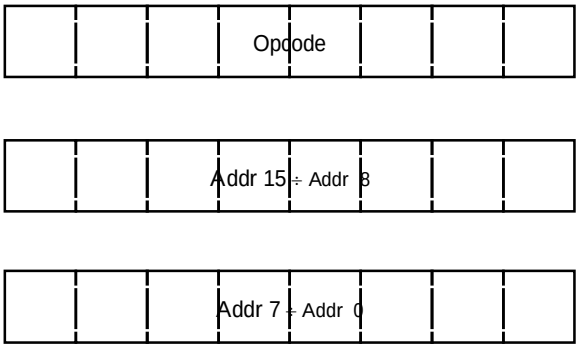




- Sự định vị tuyệt đối đem lại thuận lợi cho các lệnh ngắn (2 byte), nhưng bất lợi trong việc giới hạn phạm vi nơi gởi đến và cung cấp mã có vị trí độc lập.

1.7 Sự định vị dài (Long Addressifg):

- Sự định vị dài được dùng với lệnh LCALL và LJMP. Các lệnh 3 byte này bao gồm một địa chỉ nơi gởi tới 16 bit đầy đủ là 2 byte và 3 byte của lệnh.



- Ưu điểm của sự định dài là vùng nhớ mã 64K có thể được dùng hết, nhược điểm là các lệnh đó dài 3 byte và vị trí lệ thuộc. Sự phụ thuộc vào vị trí sẽ bất lợi bởi chương trình không thể thực thi tại địa chỉ khác.

1.8 Sự định địa chỉ phụ lục (Index Addressing) :

- Sự định địa chỉ phụ lục dùng một thanh ghi cơ bản (cũng như bộ đếm chương trình hoặc bộ đếm dữ liệu) và Offset (thanh ghiA) trong sự hình thành 1 địa chỉ liên quan bởi lệnh JMP hoặc MOVC.

Index Addressing.

- Các bảng của lệnh nhảy hoặc các bảng tra được tạo nên một cách dễ dàng bằng cách dùng địa chỉ phụ lục.

II . CÁC KIỂU LỆNH (INSTRUCTION TYPES) CỦA 8951:

8951 chia ra 5nhóm lệnh chính :

- Các lệnh số học.
- Lệnh logic.
- Dịch chuyển dữ diệu.
- Lý luận.
- Rẽ nhánh chương trình.

Từng kiểu lệnh được mô tả như sau :

1. Các lệnh số học (Arithmetic Instrustion) :

ADD A, <src, byte>

ADD	A, Rn	: (A) ← (A) + (Rn)
ADD	A, direct	: (A) ← (A) + (direct)
ADD	A, @ Ri	: (A) ← (A) + ((Ri))
ADD	A, # data	: (A) ← (A) + # data
ADDC	A, Rn	: (A) ← (A) + (C) + (Rn)
ADDC	A, direct	: (A) ← (A) + (C) + (direct)
ADDC	A, @ Ri	: (A) ← (A) + (C) + ((Ri))
ADDC	A, # data	: (A) ← (A) + (C) + # data

SUBB A, <src, byte>

SUBB	A, Rn	: (A) ← (A) - (C) - (Rn)
SUBB	A, direct	: (A) ← (A) - (C) - (direct)
SUBB	A, @ Ri	: (A) ← (A) - (C) - ((Ri))
SUBB	A, # data	: (A) ← (A) - (C) - # data

INC <byte>

INC	A	: (A) ← (A) + 1
INC	direct	: (direct) ← (direct) + 1
INC	Ri	: ((Ri)) ← ((Ri)) + 1
INC	Rn	: (Rn) ← (Rn) + 1
INC	DPTR	: (DPTR) ← (DPTR) + 1

DEC <byte>

DEC	A	: (A) ← (A) - 1
DEC	direct	: (direct) ← (direct) - 1

DEC	@Ri	: ((Ri)) ← ((Ri)) - 1
DEC	Rn	: (Rn) ← (Rn) - 1
MULL	AB	: (A) ← LOW [(A) x (B)] ; có ảnh hưởng cờ OV
		: (B) ← HIGH [(A) x (B)] ; cờ Cary được xóa.
DIV	AB	: (A) ← Integer Result of [(A)/(B)]; cờ OV
		: (B) ← Remainder of [(A)/(B)]; cờ Carry xóa
DA	A	: Điều chỉnh thanh ghi A thành số BCD đúng trong phép cộng BCD (thường DA A đi kèm với ADD, ADDC)

- Nếu [(A3-A0)>1] và [(AC)=1] ⇒ (A3÷A0) ← (A3÷A0) + 6.
- Nếu [(A7-A4)>9] và [(C)=1] ⇒ (A7÷A4) ← (A7÷A4) + 6.

2. Các hoạt động logic (Logic Operation) :

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

Các hoạt động logic được tóm tắt như sau :

ANL <dest - byte> <src - byte>

ANL	A, Rn	: (A) ← (A) AND (Rn).
ANL	A, direct	: (A) ← (A) AND (direct).
ANL	A,@ Ri	: (A) ← (A) AND ((Ri)).
ANL	A, # data	: (A) ← (A) AND (# data).
ANL	direct, A	: (direct) ← (direct) AND (A).
ANL	direct, # data	: (direct) ← (direct) AND # data.

ORL <dest - byte> <src - byte>

ORL	A, Rn	: (A) ← (A) OR (Rn).
ORL	A, direct	: (A) ← (A) OR (direct).
ORL	A,@ Ri	: (A) ← (A) OR ((Ri)).
ORL	A, # data	: (A) ← (A) OR # data.
ORL	direct, A	: (direct) ← (direct) OR (A).
ORL	direct, # data	: (direct) ← (direct) OR # data.

XRL <dest - byte> <src - byte>

XRL	A, Rn	: (A) ← (A)) (Rn).
XRL	A, direct	: (A) ← (A)) (direct).

XRL	A,@ Ri	: (A) ← (A)) ((Ri)).
XRL	A, # data	: (A) ← (A)) # data.
XRL	direct, A	: (direct) ← (direct)) (A).
XRL	direct, # data	: (direct) ← (direct)) # data.
CLR	A	: (A) ← 0
CLR	C	: (C) ← 0
CLR	Bit	: (Bit) ← 0
RL	A	: Quay vòng thanh ghi A qua trái 1 bit
		(An + 1) ← (An); n = 0÷6
		$(A_0) \leftarrow (A_7)$
RLC	A	: Quay vòng thanh ghi A qua trái 1 bit có cờ Carry
		(An + 1) ← (An); n = 0÷6
		(C) ← (A_7)
		(A_0) ← (C)
RR	A	: Quay vòng thanh ghi A qua phải 1 bit
		(An + 1) → (An); n = 0÷6
		$(A_0) \rightarrow (A_7)$
RRC	A	: Quay vòng thanh ghi A qua phải 1 bit có cờ Carry
		(An + 1) → (An); n = 0÷6
		(C) → (A_7)
		(A_0) → (C)
SWAP	A	: Đổi chổ 4 bit thấp và 4 bit cao của A cho nhau (A3÷A0)Ỉ(A7÷A4).

3. Các lệnh rẽ nhánh :

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

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

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

JC	rel	: Nhảy đến "rel" nếu cờ Carry C = 1.
JNC	rel	: Nhảy đến "rel" nếu cờ Carry C = 0.
JB	bit, rel	: Nhảy đến "rel" nếu (bit) = 1.
JNB	bit, rel	: Nhảy đến "rel" nếu (bit) = 0.
JBC	bit, rel	: Nhảy đến "rel" nếu bit = 1 và xóa bit.

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

(PC) ← (PC) + 2

(SP) ← (SP) + 1

((SP)) ← (PC7÷PC0)

(SP) ← (SP) + 1

((SP)) ← (PC15÷PC8)

(PC10÷PC0) ← page Address.

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

(PC) ← (PC) + 3

(SP) ← (SP) + 1

((SP)) ← (PC7÷PC0)

(SP) ← (SP) + 1

((SP)) ← (PC15÷PC8)

(PC) ← Addr15÷Addr0.

RET : Kết thúc chương trình con trở về chương trình chính.

(PC15÷PC8) ← (SP)

(SP) ← (SP) - 1

(PC7÷PC0) ← ((SP))

(SP) ← (SP) -1.

RETI : Kết thúc thủ tục phục vụ ngắt quay về chương trình chính hoạt động tương tự như RET.

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

(PC) ← (PC) + 2

(PC10÷PC0) ← page Address.

LJMP Addr16 : Nhảy dài không điều kiện trong 64K

Hoạt động tương tự lệnh LCALL.

SJMP rel : Nhảy ngắn không điều kiện trong (-128÷127) byte

(PC) ← (PC) + 2

(PC) ← (PC) + byte 2

JMP @ A + DPTR:Nhảy không điều kiện đến địa chỉ (A) + (DPTR)

(PC) ← (A) + (DPTR)

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

(PC) ← (PC) + 2

(A) = 0 ⇒ (PC) ← (PC) + byte 2

JNZ rel : Nhảy đến A ≠ 0. Thực hành lệnh kế nếu A = 0.

(PC) ← (PC) + 2

(A) < > 0 ⇒ (PC) ← (PC) + byte 2

CJNE A, direct, rel : So sánh và nhảy đến A ≠ direct

(PC) ← (PC) + 3

(A) < > (direct) ⇒ (PC) ← (PC) + Relative Address.

(A) < (direct) ⇒ C = 1

(A) > (direct) ⇒ C = 0

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

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

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

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

DJNE Rn, rel : Giảm Rn và nhảy nếu Rn ≠ 0.

(PC) ← (PC) + 2

(Rn) ← (Rn) -1

(Rn) < > 0 ⇒ (PC) ← (PC) + byte 2.

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

4. Các lệnh dịch chuyển dữ liệu:

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

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

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

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

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

MOV A,Rn : (A) ← (Rn)

MOV A, direct : (A) ← (direct)

MOV	A, @ Ri	: (A) ← ((Ri))
MOV	A, # data	: (A) ← # data
MOV	Rn, A	: (Rn) ← (A)
MOV	Rn, direct	: (Rn) ← (direct)
MOV	Rn, # data	: (Rn) ← # data
MOV	direct, A	: (direct) ← (A)
MOV	direct, Rn	2 (direct) ← (Rn)
MOV	direct, direct	: (direct) ← (direct)
MOV	direct, @ Ri	: (direct) ← ((Ri))
MOV	direct, # data	: (direct) ← data
MOV	@ Ri, A	: ((Ri)) ← (A)
MOV	@ Ri, direct	: ((Ri)) ← (direct)
MOV	@ Ri, # data	: ((Ri)) ← # data
MOV	DPTR, # data16	: (DPTR) ← # data16
MOV	A, @ A + DPTR	: (A) ← (A) + (DPTR)
MOV	@ A + PC	: (PC) ← (PC) + 1
		(A) ← (A) + (PC)
MOVX	A, @ Ri	: (A) ← ((Ri))
MOVX	A, @ DPTR	: (A) ← ((DPTR))
MOVX	@ Ri, A	: ((Ri)) ← (A)
MOVX	@ DPTR, A	: ((DPTR)) ← (A)
PUSH	direct	: Cất dữ liệu vào Ngăn xếp
		(SP) ← (SP) + 1
		(SP) ← (Drirect)
POP	direct	: Lấy từ Ngăn xếp ra direct
		(direct) ← ((SP))
		(SP) ← (SP) - 1
XCH	A, Rn	: Đổi chổ nội dung của A với Rn
		(A) $\mathscr{U}$(Rn)
XCH	A, direct	: (A) $\mathscr{U}$(direct)
XCH	A, @ Ri	: (A) $\mathscr{U}$((Ri))
XCHD	A, @ Ri	: Đổi chổ 4 bit thấp của (A) với ((Ri))
		(A3÷A0) $\mathscr{U}$((Ri3÷Ri0))

5. Các lệnh luận lý (Boolean Instruction) :

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

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

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

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

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

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

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

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

SET BIT_: Set bit lên 1. Không ảnh hưởng cờ Carry.

CPL C : Đão bit cờ Carry. Có ảnh hưởng cờ Carry.

CPL BIT : Đão bit. Không ảnh hưởng cờ Carry.

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

ANL C, /BIT : (C) ← (C) AND NOT (BIT):Không ảnh hưởng cờ Carry.

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

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

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

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

6. Các lệnh xen vào (Miscellamous Intstruction) :

NOP : Không hoạt động gì cả, chỉ tốn 1 byte và 1 chu kỳ máy. Ta dùng để delay những khoảng thời gian nhỏ.

Chương II .

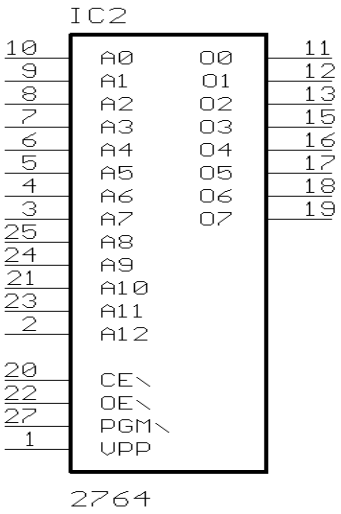
KHẢO SÁT IC SỬ DỤNG

I . Khảo sát bộ nhớ EPROM 2764

Các bộ nhớ EPROM thông dụng tồn tại dưới nhiều kiểu mạch khác nhau. Họ 27XXX có các loại vi mạch sau:2708 (1Kx8), 2716 (2Kx8), 2732 (4Kx8), 2764 (8Kx8) …… với T_{ac} = 250_ 450 ns tuỳ theo loại EPROM cụ thể. Số đường địa chỉ thay đổi tùy thuộc vào dung lượng của mỗi loại EPROM, số đường dữ liệu là 8. Hoạt động của họ 27XXX là tương tự nhau, ở đây do thời lượng có hạn nên chỉ khảo sát một EPROM được sử dụng trong mạch là EPROM 2764.

1 . Sơ đồ chân:

Trong đó A0 – A12 : 13 đường địa chỉ vào



O0 _ O7 : 8 đường dữ liệu
OE\ : Output Enable
CE\ : Chip Enable
PGM\ : Nạp chương trình

EPROM 2764 có 13 đường địa chỉ nên có dung lượng là 8 KB và 8 đường dữ liệu ngõ vào cung cấp Vpp và Vcc. Ngõ vào Vcc luôn được nối lên 5v khi EPROM đang đọc dữ liệu và nối với 30v khi nạp trình cho EPROM.

Thời gian truy xuất là T_{ac} = 150 ns .

Hai ngõ vào điều khiển CE\ và OE\:

+ OE\ : Được dùng để điều khiển bộ đệm (Output Buffer) để cho phép dữ liệu của EPROM có được xuất ra ngoài hay không.

+ CE\ : Là ngõ vào cho phép hai chức năng:

- Khi hoạt động bình thường CE\ là tín hiệu cho phép, để đọc dữ liệu từ EPROM CE\ phải ở mức logic 0 ,để mạch điện bên ngoài lựa chọn dữ liệu và chuyển nó đến Output Buffer kết hợp với tín hiệu cho phép OE\ ở mức logic 0 thì dữ liệu xuất hiện ở các ngõ ra $D_7 – D_0$.
- Khi CE\ =1 thì EPROM ở trạng thái chờ (Standby)

Công suất tiêu thụ ở trạng thái đọc dữ liệu là 525 mW, ở trạng thái chờ là 132 mW.

2 . Bảng trạng thái làm việc của EPROM 2764:

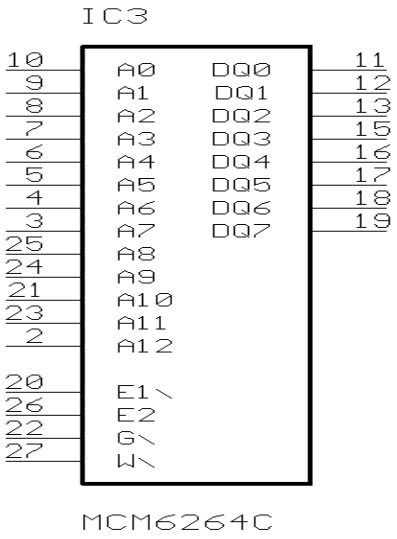
	OE	**CE**	**PGM**	**Vpp**	**Vcc**	**Output**
Read	V_{IL}	V_{IL}	V_{IH}	V_{cc}	V_{cc}	D out
Standby	V_{IH}	X	X	V_{cc}	V_{cc}	High Z

Program	V_{IL}	X	V_{IL}	V_{pp}	V_{cc}	D in
Program Verify	V_{IL}	V_{IL}	V_{IH}	V_{pp}	V_{cc}	D out
Program Inhibit	V_{IH}	X	X	V_{pp}	V_{cc}	High Z

II . Khảo sát SRAM 6264

SRAM 6264 có dung lượng 8 KB dùng để lưu trữ chương trình và dữ liệu.

1 . Sơ đồ chân:



Trong đó A_0 _ A_{12} :13 đường địa chỉ vào

WR\ : cho phép ghi ở mức logic 0

OE\ : cho phép xuất ở mức logic 0

CS \ : cho phép chọn vi mạch hoạt động (mức 0)

NC : No Conection

Bộ nhớ SRAM có khả năng lưu trữ thông tin trong nó chừng nào còn được cấp điện

Thời gian truy xuất T_{ac} = 250 ns .

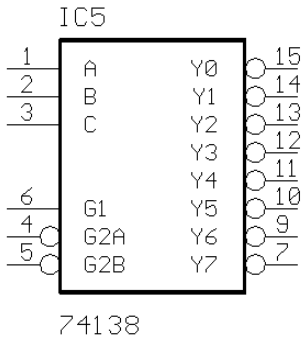
2 . Bảng trạng thái làm việc của SRAM 6264

	WR	**CS**	**CS**	**OE**	**Output**
Not Select	X	H	X	X	High Z
Not Select	X	X	L	X	High Z
Output Disable	H	L	H	H	High Z
Read	H	L	H	L	D Out
Write	L	L	H	H	D in

III . Khảo sát IC giải mã 74138 :

Khi ta muốn có nhiều đầu ra chọn vỏ từ bộ giải mã mà vẫn dùng các mạch logic đơn giản thì thiết kế sẽ trở nên rất cồng kềnh do số lượng các mạch cửa tăng lên. Trong trường hợp như vậy ta thường dùng các mạch giải mã có sẵn. Một trong các mạch giải mã hay được sử dụng là 74LS138 ,mạch giải mã 3 ra 8 đường.

1. Sơ đồ chân:



Trong đó:

Y_0-Y_7: 8 đầu ra chọn mạch giải mã.

A,B,C:Ba đầu vào chọn.

E_1\,E_2\,E_3:đầu vào cho phép.

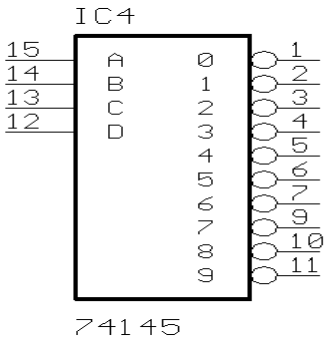
2. Bảng trạng thái làm việc của 74138:

Các đầu vào						Các đầu ra							
Cho phép			Điều khiển										
C	B	A	G_2B\	G_2A\	G_1	Y_0\	Y_1\	Y_2\	Y_3\	Y_4\	Y_5\	Y_6\	Y_7\
X	X	X	1	X	X	1	1	1	1	1	1	1	1
X	X	X	X	1	X	1	1	1	1	1	1	1	1
X	X	X	X	X	0	1	1	1	1	1	1	1	1
0	0	0	0	0	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1	1
1	0	0	0	0	1	1	1	1	1	0	1	1	1
1	0	1	0	0	1	1	1	1	1	1	0	1	1
1	1	0	0	0	1	1	1	1	1	1	1	0	1
1	1	1	0	0	1	1	1	1	1	1	1	1	0

Trong đó : X là giá trị không quan tâm.

IV: khảo sát IC giải mã BCD to Decimal SN74145.

1. ***Sơ đồ chân:***



Trong đó :0.9 là 10 đầu ra.

A,B,C,D là 4 đầu vào.

2. ***Bảng trạng thái.***

No	Input					output								
	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
Valid	1	0	1	0	1	1	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1	1	1	1	1
	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1

TÀI LIỆU THAM KHẢO

1. Kỹ Thuật Vi Xử Lý:
   Tác giả: Trần Văn Trọng (ĐHSP Kỹ Thuật-TP.HCM)
   Xuất bản năm 1995.
2. Giáo trình Vi Xử Lý – Vi điều khiển
   Người soạn: Nguyễn Đình Phú
3. Lập Trình Cho Các Hệ Vi Xử Lý
   Biên soạn: Hùynh Thúc Cước
   Đặng Văn Đức
   Nghiêm Mỹ
   Nguyễn Văn Tam
   Trần Bá Thái
   Nguyễn Chí Thức
   Nhà xuất bản Thống Kê
4. Kỹ Thuật Vi Điều Khiển
   Tác giả : Lê Văn Doanh
   Phạm Chắc Chương
   Nhà Xuất Bản Khoa Học Và Kỹ Thuật năm 1998.
5. THE 8051 MICROCONTROLLER
   Tác Giả: Scott MacKenzie
6. Trình Biên Dịch ASM51.