



BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH  
**60** NĂM XÂY DỰNG VÀ PHÁT TRIỂN

PHẠM VĂN KHOA

**THỰC HÀNH  
KIẾN TRÚC VÀ TỔ CHỨC MÁY TÍNH**



NHÀ XUẤT BẢN  
ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT  
THÀNH PHỐ HỒ CHÍ MINH

\*\*\*\*\*



**HCMUTE**

**TS. PHẠM VĂN KHOA**

**THỰC HÀNH  
KIẾN TRÚC VÀ TỔ CHỨC MÁY TÍNH**

*(Sách tham khảo dành cho sinh viên  
Ngành Công nghệ Kỹ thuật Máy Tính)*

**NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA  
THÀNH PHỐ HỒ CHÍ MINH – NĂM 2020**

# THỰC HÀNH KIẾN TRÚC VÀ TỔ CHỨC MÁY TÍNH

PHẠM VĂN KHOA

**Chịu trách nhiệm xuất bản và nội dung**

TS. ĐỖ VĂN BIÊN

**Biên tập**

NGUYỄN THỊ NGỌC ANH

**Sửa bản in**

PHƯỚC HUỆ

**Trình bày bìa**

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

Website: <http://hcmute.edu.vn>

**Đối tác liên kết – Tổ chức bản thảo và chịu trách nhiệm tác quyền**

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

Website: <http://hcmute.edu.vn>

**NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

Phòng 501, Nhà Điều hành ĐHQG-HCM, phường Linh Trung, quận Thủ Đức, TP Hồ Chí Minh

**ĐT: 028 6272 6361 - 028 6272 6390      E-mail: [vnuhp@vnuhcm.edu.vn](mailto:vnuhp@vnuhcm.edu.vn)**

Website: [www.vnuhcmpress.edu.vn](http://www.vnuhcmpress.edu.vn)

**VĂN PHÒNG NHÀ XUẤT BẢN**

**PHÒNG QUẢN LÝ DỰ ÁN VÀ PHÁT HÀNH**

Tòa nhà K-Trường Đại học Khoa học Xã hội & Nhân văn, số 10-12 Đinh Tiên Hoàng, phường Bến Nghé, Quận 1, TP Hồ Chí Minh

**ĐT: 028 66817058 - 028 62726390 - 028 62726351**

Website: [www.vnuhcmpress.edu.vn](http://www.vnuhcmpress.edu.vn)

Nhà xuất bản ĐHQG-HCM và tác giả/ đối tác liên kết giữ bản quyền©

Copyright © by VNU-HCM Press and author/  
co-partnership. All rights reserved.

**ISBN: 978-604-73-7791-6**



**ISBN: 978-604-73-7791-6**

Xuất bản lần thứ 1. In 250 cuốn, khổ 16 x 24 cm, XNĐKXB số: 2442-2020/CXBIPH/3-53/ĐHQGTPHCM. QĐXB số 121/QĐ-NXB ĐHQGTPHCM, cấp ngày 29/6/2020.

In tại: Công ty TNHH In & Bao bì Hưng Phú. Địa chỉ: 162A/1, KP1A, P. An Phú, TX. Thuận An, Bình Dương. Nộp lưu chiểu: Quý III/2020.

**THỰC HÀNH  
KIẾN TRÚC VÀ TỔ CHỨC MÁY TÍNH | PHẠM VĂN KHOA**

---

Bản tiếng Việt ©, TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM,  
NXB ĐHQG-HCM và TÁC GIẢ.

Bản quyền tác phẩm đã được bảo hộ bởi Luật Xuất bản và Luật Sở hữu trí tuệ  
Việt Nam. Nghiêm cấm mọi hình thức xuất bản, sao chụp, phát tán nội dung khi  
chưa có sự đồng ý của Trường đại học Sư phạm Kỹ thuật TP. HCM và Tác giả.

**ĐỀ CÓ SÁCH HAY, CẦN CHUNG TAY BẢO VỆ TÁC QUYỀN!**



# LỜI NÓI ĐẦU

Với sự phát triển và thành công của ngành vi mạch bán dẫn và kỹ thuật truyền thông đã cho ra đời khái niệm Kết nối vạn vật (Internet of Things). Trong công nghệ này, các máy tính đa dụng (general purpose computer) được thu nhỏ thành các máy tính có chức năng xác định (specific purpose computer) được nhúng vào trong các thiết bị thu nhỏ nhằm để tính toán và xử lý dữ liệu. Nhờ đó, các thiết bị trở nên thông minh hơn bởi có sự hỗ trợ xử lý của máy tính cũng như dễ dàng kết nối với nhau hơn bởi các máy tính loại này cũng được hỗ trợ các chuẩn truyền thông nhằm trao đổi dữ liệu trước và sau quá trình xử lý.

Việc khảo sát một máy tính có chức năng xác định như vi điều khiển là thực sự cần thiết đối với sinh viên ngành Kỹ thuật Máy tính nói riêng cũng như sinh viên ngành Kỹ thuật Điện tử nói chung. Tài liệu này nhằm hướng dẫn người đọc khảo sát một dạng máy tính trên chip là vi điều khiển họ 8051. Qua đó, tài liệu này trình bày những nét sơ lược chung về mặt kiến trúc phần cứng của vi điều khiển như thanh ghi, kích thước dữ liệu, phương pháp đánh địa chỉ và mở rộng bộ nhớ được thể hiện cụ thể qua các ứng dụng minh họa cơ bản cũng như các câu hỏi ôn tập. Ngoài ra, chuẩn truyền thông giữa máy tính đa dụng và máy tính trên chip như UART cũng được trình bày, từ đó người đọc có thể hiểu hơn về phương pháp giao tiếp và điều khiển từ máy tính.

Phương pháp tiếp cận của tài liệu này là hướng dẫn người đọc xem xét các khái niệm lý thuyết thông qua các câu hỏi cũng như thực hành các đoạn mã nguồn sử dụng ngôn ngữ hợp ngữ (assembly) và mã nguồn cấp cao (C) trên một kiến trúc máy tính xác định. Bên cạnh đó, những phương pháp giao tiếp giữa vi điều khiển và các thiết bị ngoại vi cũng như bộ nhớ giúp người đọc dễ tiếp cận hơn về khía cạnh sử dụng máy tính trên chip để điều khiển thiết bị và xử lý các thông tin đơn giản.

Tác giả

TS. Phạm Văn Khoa



## **LỜI CẢM ƠN**

Để hoàn thành được cuốn sách này, tác giả đã nhận được sự hỗ trợ rất nhiều từ bạn bè, đồng nghiệp, Khoa Điện - Điện tử và Trường Đại học Sư phạm Kỹ thuật trong quá trình giảng dạy và nghiên cứu.

Tác giả

TS. Phạm Văn Khoa



# MỤC LỤC

**PHẦN I: Khảo sát kiến trúc phần cứng 8051 và ngôn ngữ hợp ngữ .....** 9

**PHẦN II: Khảo sát tập lệnh được hỗ trợ đối với vi điều khiển họ 8051 .....** 19

**Bài thực hành số 1:**

Chương trình di chuyển và trao đổi dữ liệu ..... 20

**Bài thực hành số 2:**

Chương trình tính toán số học ..... 26

**Bài thực hành số 3:**

Chương trình lệnh xử lý luận lý và bộ đếm..... 32

**Bài thực hành số 4:**

Chương trình chuyển đổi giá trị hệ thống số..... 38

**PHẦN III: Lập trình giao tiếp 8051 và ngoại vi sử dụng ASM..... 41**

**Bài thực hành số 1:**

Lệnh tính số học để thiết kế bộ cộng/trừ 2 số nhị phân..... 42

**Bài thực hành số 2:**

Bộ định thời tạo xung dao động tần số xác định ..... 46

**Bài thực hành số 3:**

Ngắt ngoài để giao tiếp với bàn phím ma trận ..... 51

**Bài thực hành số 4:**

Thiết kế máy tính để cộng/trừ số có 2 chữ số ..... 60

**PHẦN IV: Lập trình giao tiếp 8051 và ngoại vi sử dụng C..... 75**

**Bài thực hành số 1:**

Giao tiếp LED đơn ..... 84

**Bài thực hành số 2:**

Giao tiếp LED 7 đoạn..... 89

**Bài thực hành số 3:**

Giao tiếp bàn phím ma trận ..... 100

***Bài thực hành số 4:***

Giao tiếp màn hình tinh thể lỏng LCD..... 114

***Bài thực hành số 5:***

Giao tiếp bộ chuyển đổi tương tự sang số ADC..... 125

***Bài thực hành số 6:***

Giao tiếp truyền thông nối tiếp UART ..... 133

***Bài thực hành số 7:***

Giao tiếp bộ nhớ RAM/ROM..... 138

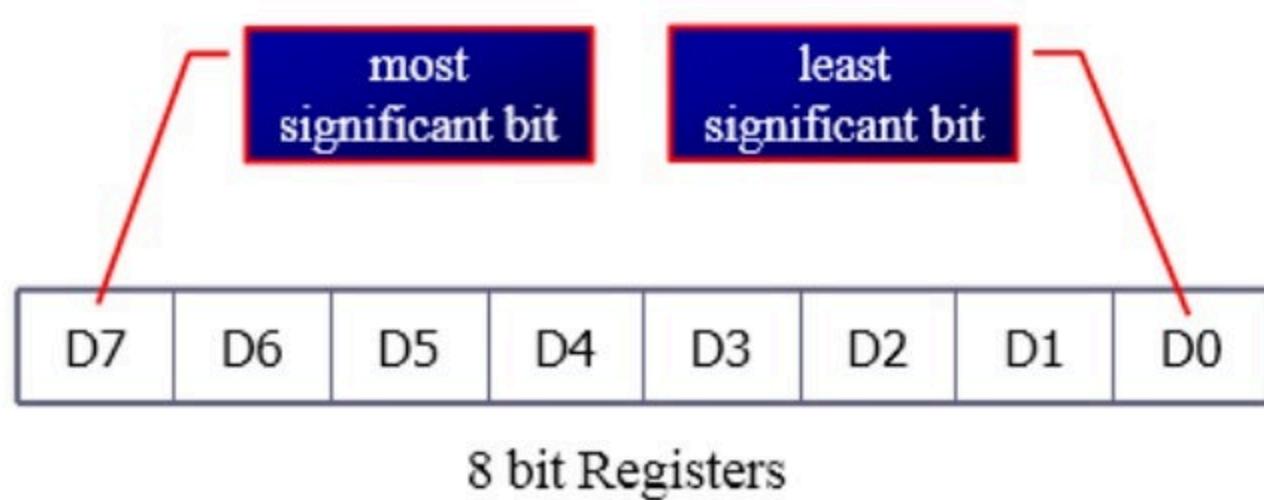
**Phụ lục..... 153**

**Tài liệu tham khảo ..... 169**

# **PHẦN I: KHẢO SÁT PHẦN CỨNG 8051 VÀ NGÔN NGỮ HỢP NGỮ**

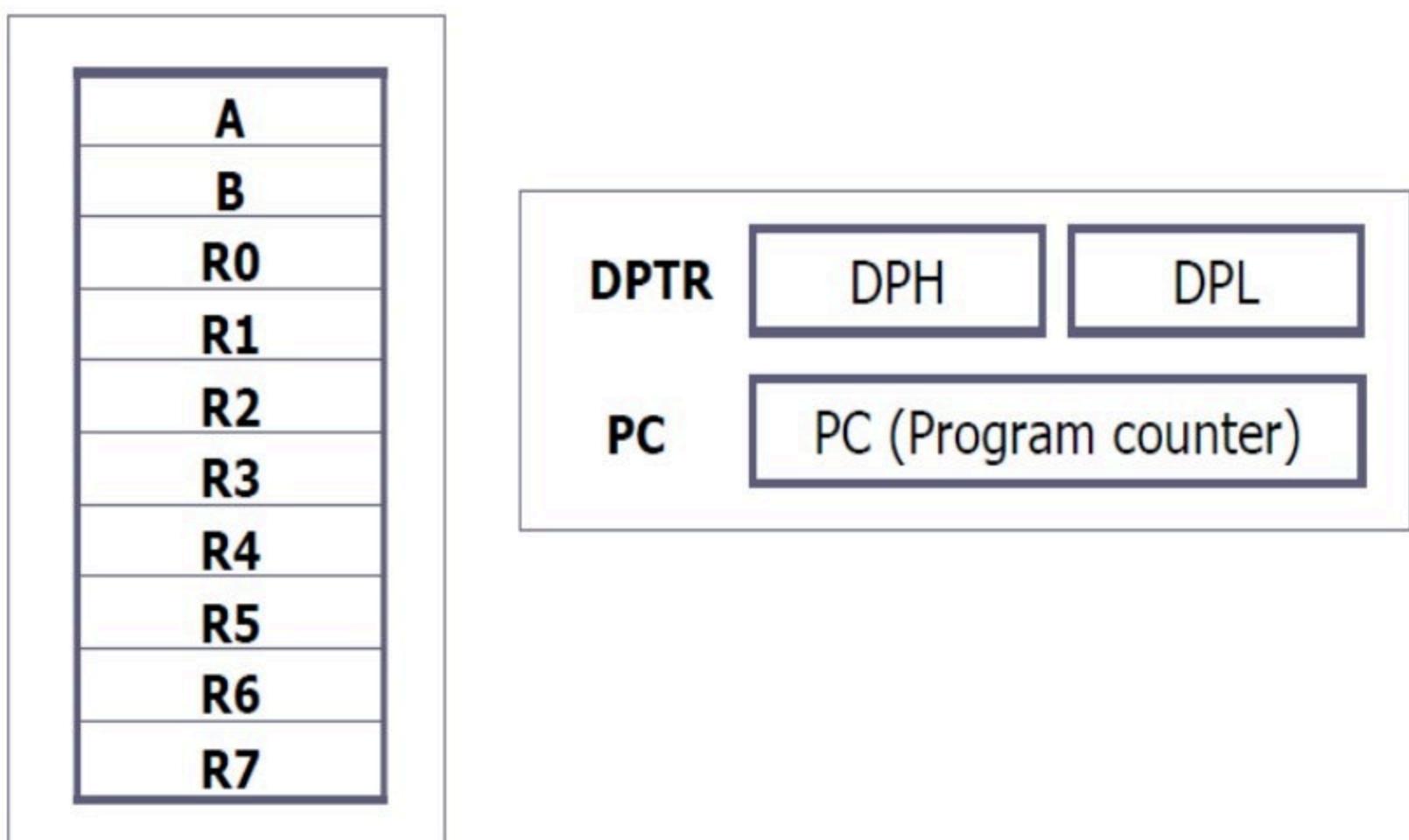
Mục đích chung của phần này nhằm giúp người học ôn tập một số vấn đề về kiến trúc máy tính từ việc sử dụng thiết kế của một máy tính trên chip là vi điều khiển họ 8051. Bên cạnh đó, kiến trúc tập lệnh được hỗ trợ cho họ vi điều khiển này cũng được khái quát sơ lược thông qua một số câu hỏi ôn tập. Việc khảo sát ngôn ngữ hợp ngữ và ý nghĩa của các quá trình biên dịch mã nguồn thành mã máy cũng được trình bày trong phần này.

Trong kiến trúc máy tính, các thanh ghi được sử dụng để lưu trữ thông tin một cách tạm thời, trong đó thông tin có thể là dữ liệu theo byte đã được xử lý, hoặc là con trỏ địa chỉ trỏ đến dữ liệu được nạp. Đối với máy tính trên chip họ 8051, các thanh ghi hầu hết là 8-bit. Nó được thể hiện như hình dưới đây, trong đó D0 là LSB và D7 là MSB.



**Hình 1:** Thanh ghi 8-bit

Cũng giống như trên một kiến trúc máy tính thông thường, các vi điều khiển (máy tính trên chip) cũng có các thanh ghi thông dụng như thể hiện bởi hình sau đây:



**Hình 2:** Các thanh ghi đa dụng được hỗ trợ trên họ 8051

Trong đó, chức năng của từng thanh ghi có thể liệt kê như sau:

Thanh ghi A	Được sử dụng cho hầu hết các lệnh về toán học và logic
Thanh ghi DPTR	Là con trỏ dữ liệu (data pointer)
Thanh ghi PC	Là bộ đếm chương trình (program counter)
Thanh ghi B, R0, R1, R2, R3, R4, R5, R6, R7	Là các thanh ghi đa dụng

Khảo sát cấu trúc phần cứng của vi điều khiển họ 8051 và sử dụng phần mềm mô phỏng 8051 (EdSim51DITM hoặc Keil) để trả lời các câu hỏi sau:

1. Chỉ ra số lượng bit (8 hoặc 16) cho mỗi thanh ghi sau:

$$PC = \quad A = \quad B =$$

$$R0 = \quad R1 = \quad R2 = \quad R7 =$$

2. Chỉ ra giá trị lớn nhất (hệ 10) mà mỗi thanh ghi sau có thể chứa được là bao nhiêu?

$$PC = \quad A = \quad B =$$

$$R0 = \quad R1 = \quad R2 = \quad R7 =$$

3. Chỉ ra giá trị lớn nhất (hệ 16) mà mỗi thanh ghi sau có thể chứa được là bao nhiêu?

$$PC = \quad A = \quad B =$$

$$R0 = \quad R1 = \quad R2 = \quad R7 =$$

4. Hãy chỉ ra các tập tin sau được tạo ra từ đâu? Ý nghĩa của mỗi tập tin sau là gì?

.asm	
.lst	
.obj	
.abs	
.hex	

5. Chỉ ra giá trị của cờ CY sau khi các lệnh dưới đây được thực hiện là bao nhiêu?

MOV A,#85H ADD A,#92H	
MOV A,#15H ADD A,#72H	
MOV A,#0F5H ADD A,#52H	
MOV A,#0FF INC A	

6. Hãy chỉ ra vào lúc reset giá trị của thanh ghi SP (stack pointer) là bao nhiêu?
7. Hãy chỉ ra vào thời điểm đưa (push) dữ liệu vào stack, giá trị của SP được tăng hay giảm?
8. Hãy chỉ ra vào thời điểm lấy (pop) dữ liệu ra khỏi stack, giá trị của SP được tăng hay giảm?
9. Người lập trình có thể thay đổi thanh ghi SP không? Nếu được thì cần thay đổi trong những trường hợp nào?
10. Vào thời điểm reset, tất cả các port trên 8051 được cấu hình là ngõ vào hay ngõ ra?
11. Để cho tất cả các bit trên một port nào đó là ngõ vào thì cần ghi giá trị mã hex là bao nhiêu đến cổng đó?
12. Những cổng nào trên 8051 là có thể truy cập theo bit?
13. Giải thích đặc tính “read-modify-write” trên port?
14. Viết chương trình để theo dõi bit P2.4 liên tục. Nếu chân nào xuống mức thấp hãy gửi giá trị 55H ra port 1.
15. Giải thích sự khác nhau giữa hai lệnh sau:

MOVC A,@A+DPTR	
MOV A,@R0	

16. Hãy chỉ ra lệnh nào sau đây là lệnh sai

MOV A,@R1	MOV A,@R2
MOVC A,@R0+DPTR	MOV @R3,A

17. Giải thích sự khác biệt giữa hai lệnh sau:

MOV A,40H	
MOV A,#40H	

18. Giải thích sự khác biệt giữa hai lệnh sau:

MOV 40H,A	
MOV 40H,#0A	

19. Hãy cho biết địa chỉ trên RAM của mỗi thanh ghi sau:

A =		B =		R0 =		R2 =	
PSW =		SP =		DPL =		DPH =	

20. Giải thích sự khác biệt giữa hai lệnh ADD và ADDC.

21. Có thể sử dụng lệnh “DA A” để chuyển dữ liệu 9CH thành định dạng BCD mà không cần thực hiện lệnh ADD không? Giải thích.

22. Đối với 8051, khi thực hiện phép chia 1 byte cho 1 byte khác, giá trị lớn nhất đối với tử số và mẫu số là bao nhiêu? Thể hiện câu trả lời ở dạng số HEX và DEC.

23. Đối với 8051, khi thực hiện phép chia 1 byte cho 1 byte khác, giá trị lớn nhất cho số chia và phần dư là bao nhiêu? Thể hiện câu trả lời ở dạng số HEX và DEC.

24. Đối với 8051, khi thực hiện phép nhân 2 số, giá trị lớn nhất cho mỗi số là bao nhiêu?

25. Giải thích vai trò của cờ OV trong các lệnh nhân và chia.
26. Xét về hiệu năng, khi thực hiện bình phương (square) của các số từ 0 đến 9 phương pháp nào sau đây sẽ nhanh hơn? Giải thích
- (a) Sử dụng lệnh MUL    (b) Sử dụng bảng tra (look-up table)
27. Giá trị trong thanh ghi A sẽ là bao nhiêu (số HEX) sau khi thực hiện lệnh sau:

MOV A,#45H RR    A RR    A RR A	A =
MOV A,#45H RL    A RL    A RL A	A =
CLR A XRL A,#0FFH	A =
CLR A CPL A XRL A,#0FFH	A =

28. Giả sử 8051 không được hỗ trợ lệnh “SWAP A”, làm thế nào để thực hiện được hoạt động này?
29. Lệnh SWAP có thể thực hiện với mọi thanh ghi không?
30. Trong kiến trúc tập lệnh (Instruction set architecture) được hỗ trợ cho vi điều khiển họ 8051, những nhóm lệnh nào có CPI thấp và những nhóm lệnh nào có CPI cao?
31. Xét về hiệu năng, hãy nêu các giải pháp để tăng hiệu năng hệ thống?

Cấu trúc của một chương trình hợp ngữ bao gồm bốn trường như sau:

[label:] Mnemonic [operands] [;comment]

```
ORG 0H ;start(origin) at  
Location 0  
MOV R5, #25H ;Load 25H into R5  
MOV R7, #34H ;Load 34H into R7  
MOV A, #0 ;Load 0 into A  
ADD A, R5 ;add contents of R5 to  
A  
;now A = A + R5  
ADD A, R7  
;now A = A + R7  
ADD A, #12H ;add to A value 12H  
;now A = A + 12H  
HERE: SJMP HERE ;stay in this  
Loop  
END ;end of asm source file
```

Trong đó:

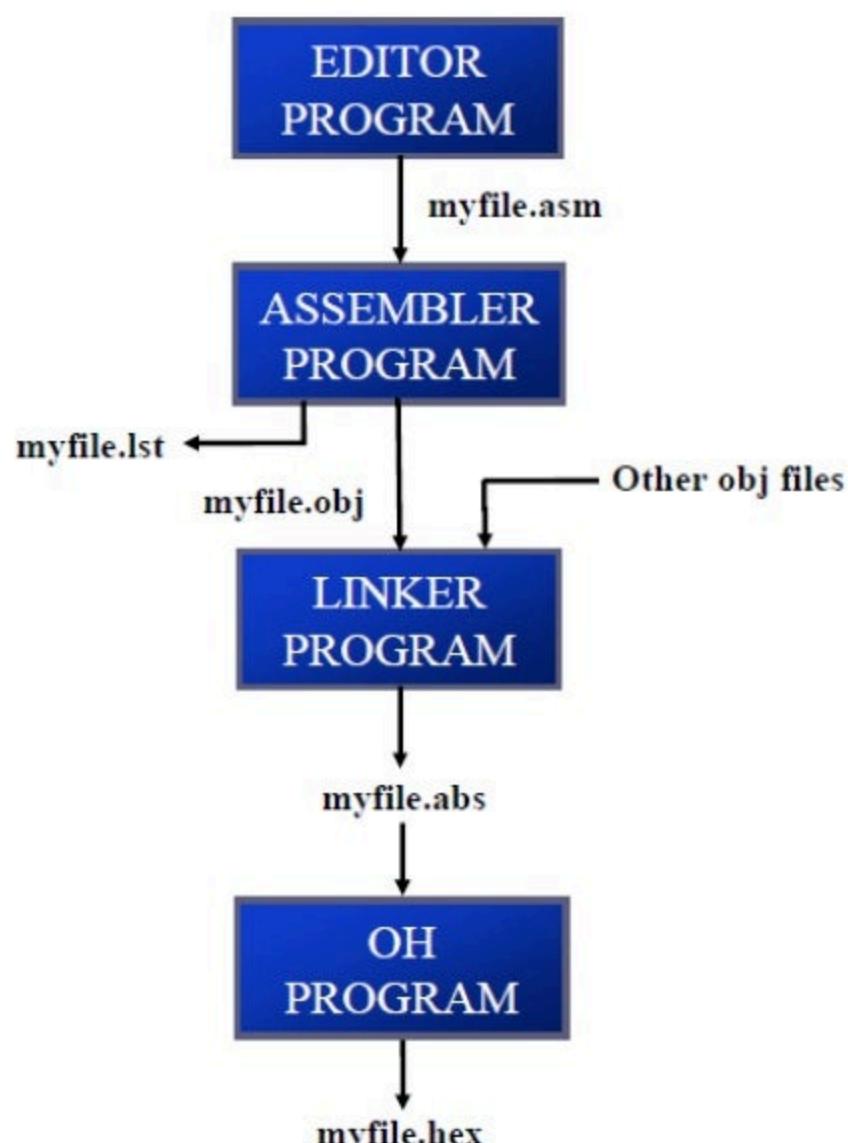
ORG là các directive được dùng bởi assembler, và không tạo ra bất cứ mã máy (machine code)

Nhãn (Label) là HERE

Mnemonic là MOV, ADD dùng để tạo opcode

Sau dấu ; là chú thích

Các bước để tạo ra một chương trình được khái quát theo hình sau:



**Hình 3:** Quá trình chuyển từ mã lập trình thành mã máy

Hãy sử dụng tập lệnh được hỗ trợ cho vi điều khiển họ 8051 để khảo sát chức năng và cách sử dụng của từng lệnh theo ví dụ như sau:

### Lệnh di chuyển (MOV)

Cú pháp	Ý nghĩa
MOV destination, source	Lệnh này nói cho bộ xử lý biết để di chuyển toán hạng nguồn (source) đến đích (destination)

Cách sử dụng:

Mã nguồn	Lưu ý
$MOV A, \#55H ; Load value 55H into reg.$ $A$ $MOV R0, A ; copy contents of A into R0$ $; (now A=R0=55H)$ $MOV R1, A ; copy contents of A into R1$ $; (now A=R0=R1=55H)$ $MOV R2, A ; copy contents of A into R2$ $; (now A=R0=R1=R2=55H)$ $MOV R3, \#95H ; Load value 95H into R3$ $; (now R3=95H)$ $MOV A, R3 ; copy contents of R3 into A$ $; now A=R3=95H$	Dấu # chỉ ra rằng đây là giá trị.
$MOV A, \#23H$	Nếu giá trị không được bắt đầu bằng #, điều đó có nghĩa toán hạng là một vùng nhớ.
$MOV R5, \#0F9H$	Thêm số 0 trước để lưu ý rằng F là một số HEX, không phải là ký tự.

<i>MOV A, #5</i>	Nếu giá trị từ 0 đến F được đưa vào một thanh ghi 8-bit điều đó có nghĩa phần còn lại của thanh ghi sẽ được định nghĩa giá trị 0. Giá trị thanh ghi A theo biểu diễn nhị phân là 00000101.
<i>MOV A, #7F2H</i>	Nạp một giá trị quá lớn vào thanh ghi 8-bit sẽ dẫn đến lỗi. như hình minh họa thì giá trị đã vượt tầm 0xFF.

### Lệnh cộng (ADD)

Cú pháp	Ý nghĩa
<i>ADD A, source</i>	Cộng toán hạng nguồn thêm vào giá trị thanh ghi A. Kết quả của phép cộng sẽ được nạp vào thanh ghi A.  Toán hạng nguồn có thể là thanh ghi hoặc dữ liệu tạm thời, tuy nhiên đích luôn là thanh ghi A.

Cách sử dụng:

Mã nguồn	Lưu ý
<i>ADD R4, A</i>	Điều này không cho phép vì A cần là toán hạng đích đối với bất cứ lệnh toán số học nào.
<i>ADD R2, #12H</i>	

<pre><code>MOV A, #25H ;Load 25H into A MOV R2, #34H ;Load 34H into R2 ADD A, R2 ;add R2 to Accumulator ;(A = A + R2)</code></pre>	<p>Có nhiều cách để viết một chương trình, sự khác nhau là việc sử dụng các thanh ghi khác nhau.</p>
<pre><code>MOV A, #25H ;Load one operand ;into A (A=25H) ADD A, #34H ;add the second ;operand 34H to A</code></pre>	
<pre><code>MOV A, #7F2H</code></pre>	<p>Nạp một giá trị quá lớn vào thanh ghi 8-bit sẽ dẫn đến lỗi. Như hình minh họa thì giá trị đã vượt tầm 0xFF.</p>

## **PHẦN II:**

# **KHẢO SÁT TẬP LỆNH ĐƯỢC HỖ TRỢ ĐỐI VỚI VI ĐIỀU KHIỂN HỌ 8051**

Mục đích chung của phần này nhằm giúp người học làm quen và vận dụng các lệnh (instruction) được hỗ trợ trong tập lệnh (instruction set) của họ vi điều khiển 8051. Từ các ví dụ cơ bản áp dụng tập lệnh được hỗ trợ đối với máy tính trên chip họ 8051, người học có thể hiểu sâu hơn ý nghĩa và cách sử dụng của từng lệnh thông qua các nhóm chương trình cơ bản sau:

- Di chuyển và trao đổi dữ liệu, sắp xếp, trao đổi và tìm phần tử trong mảng
- Xử lý số học như cộng, trừ, nhân, chia
- Xử lý luận lý
- Bộ đếm HEX, BCD
- Chuyển đổi hệ thống số BCD, DEC, HEX

# BÀI THỰC HÀNH SỐ 1

## Mục đích:

- Giới thiệu các lệnh di chuyển và trao đổi dữ liệu trên vi điều khiển họ 8051.

## Sau khi kết thúc học phần này, sinh viên có thể:

- Hiểu về các lệnh di chuyển và trao đổi dữ liệu trong vùng nhớ.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình hợp ngữ dành cho các vi điều khiển họ 8051 để thực hiện việc di chuyển và trao đổi dữ liệu.

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc di chuyển một khối dữ liệu từ một vùng nhớ nội đến vùng nhớ khác.

```
ORG 0000H
LJMP 8000H
ORG 8000H
MOV R0,#30H
MOV R1,#40H
MOV R2,#05H
NEXT: MOV A,@R0
      MOV @R1,A
      INC R0
      INC R1
      DJNZ R2, NEXT
      LCALL 0003H
```

### Yêu cầu thực hành:

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Hãy cho biết ý nghĩa của chương trình trên.
- Hãy tính số lượng machine cycle cần thiết để thực hiện chương trình trên.

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc trao đổi khối dữ liệu từ một vùng nhớ nội đến nơi khác.

```
ORG 0000H
LJMP 8000H
ORG 8000H
MOV R0,#30H
MOV R1,#40H
MOV R4,#04H
UP:   MOV A,@R0
      MOV R6,A
      MOV A,@R1
      MOV @R0,A
      MOV A,R6
      MOV @R1,A
      INC R0
      INC R1
      DJNZ R4, UP
      LCALL 0003H ;end of asm file
```

### **Yêu cầu thực hành:**

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Hãy cho biết ý nghĩa của chương trình trên.

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc di chuyển một khối dữ liệu từ bộ nhớ ngoại đến vùng nhớ khác.

```
ORG 0000H
LJMP 8000H
ORG 8000H
MOV R0, #04H
MOV R1, #81H
MOV R2, #85H
MOV R3, #00H
UP: MOV DPH, R1
    MOV DPL, R3
    MOVX A, @DPTR
    MOV DPH, R2
    MOVX @DPTR, A
    INC R3
    DJNZ R0, UP
    LCALL 0003H
```

### **Yêu cầu thực hành:**

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Lệnh MOV và MOVX khác nhau như thế nào?
- Hãy trình bày chức năng của thanh ghi DPTR.
- Hãy cho biết ý nghĩa của chương trình trên?

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc trao đổi dữ liệu từ một vùng nhớ trên bộ nhớ ngoại đến nơi khác.

```
ORG 0000H
LJMP 8000H
ORG 8000H
MOV R0, #04H
MOV R1, #81H
MOV R2, #85H
MOV R3, #00H
UP:    MOV DPH, R1
        MOV DPL, R3
        MOVX A, @DPTR
        MOV B, A
        MOV DPH, R2
        MOVX A, @DPTR
        MOV DPH, R1
        MOVX @DPTR, A
        MOV DPH, R2
        MOV A, B
        MOVX @DPTR, A
        INC R3
        DJNZ R0, UP
LCALL 0003H
```

### **Yêu cầu thực hành:**

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Hãy cho biết ý nghĩa của chương trình trên.

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc tìm phần tử có giá trị nhỏ nhất trong mảng.

```
ORG 0000H
LJMP 8000H
ORG 8000H
MOV DPTR, #8500H
MOV R0, #04H
MOVX A, @DPTR
MOV B, A
AGAIN: INC DPTR
        MOVX A, @DPTR
        CJNE A, B, NEXT
        AJMP SKIP
NEXT: JNC SKIP
        MOV B, A
SKIP: DJNZ R0, AGAIN
        MOV A, B
        MOV DPTR, #9000H
        MOVX @DPTR, A
LCALL 0003H
```

### **Yêu cầu thực hành:**

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Vận dụng mã nguồn ở trên, hãy viết lại chương trình sao cho tìm phần tử có giá trị lớn nhất trong mảng. Nếu biết mảng gồm 6 bytes được lưu tại vị trí ô nhớ 4000h. Phần tử có giá trị lớn nhất sẽ được lưu vào vùng nhớ 4062h.

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc sắp xếp một mảng gồm 6 bytes theo qui luật giá trị tăng dần hoặc giảm dần theo giá trị . Mảng được lưu tại vùng nhớ 9000h. Áp dụng thuật toán sắp xếp bubble sort.

```
ORG 0000H
MOV R0, #05H
LOOP1: MOV DPTR, #9000H
        MOV R1, #05H
LOOP2: MOVX A, @DPTR
        MOV B, A
        INC DPTR
        MOVX A, @DPTR
        CLR C
        MOV R2, A
        SUBB A, B
        JNC NOEXCHG
        MOV A, B
        MOVX @DPTR, A
        DEC DPL
        MOV A, R2
        MOVX @DPTR, A
        INC DPTR
NOEXCHG: DJNZ R1, LOOP2
        DJNZ R0, LOOP1
END
```

### **Yêu cầu thực hành:**

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Giả sử tần số hoạt động của hệ thống là 12Mhz, hãy tính thời gian CPU thực thi chương trình trên.
- Vận dụng mã nguồn ở trên với 60bytes, hãy triển khai một giải thuật sắp xếp khác sao cho thời gian thực thi chương trình giảm.

## BÀI THỰC HÀNH SỐ 2

### Mục đích:

- Giới thiệu các lệnh tính toán và xử lý số học.

### Sau khi kết thúc học phần này, sinh viên có thể:

- Hiểu về các lệnh tính toán/xử lý số học được hỗ trợ trên bộ xử lý.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình hợp ngữ cho các vi điều khiển họ 8051 để thực hiện các phép tính toán cơ bản như cộng/trừ/nhân/chia.

## Các lệnh về xử lý số học

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc cộng 2 số 8-bit.

```
ORG 0000H
LJMP 8000H
ORG 8000H
MOV DPTR, #8500H
MOVX A, @DPTR
MOV R0, A
INC DPTR
MOVX A, @DPTR
ADDC A, R0
JNC NEXT
INC R5
NEXT:
    MOV DPTR, #9001H
    MOVX @DPTR, A
    MOV A, R5
    DEC DPL
    MOVX @DPTR, A
LCALL 0003H
```

### Yêu cầu thực hành:

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Hãy viết lại chương trình trên sao cho không sử dụng thanh ghi DPTR.
- Vận dụng mã nguồn ở trên, hãy viết chương trình thực hiện phép trừ 2 số 8-bit.

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc nhân 2 số 8-bit.

```
ORG 0000H
LJMP 8000H
ORG 8000H
MOV DPTR, #8500H
MOVX A, @DPTR
MOV B, A
INC DPTR
MOVX A, @DPTR
MUL AB
INC DPTR
MOVX @DPTR, A
MOV A, B
INC DPTR
MOVX @DPTR, A
LCALL 0003H
```

**Yêu cầu thực hành:**

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc chia 2 số 8-bit.

```
ORG 0000H
LJMP 8000H
ORG 8000H
MOV DPTR, #8500H
MOVX A, @DPTR
MOV B, A
DEC DPL
MOVX A, @DPTR
DIV AB
MOV DPTR, #9500H
MOVX A, @DPTR
MOV A, B
INC DPTR
MOVX @DPTR, A
LCALL 0003H
```

### **Yêu cầu thực hành:**

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Hãy cho biết ý nghĩa của chương trình sau đây:

```
ORG 0000H  
LJMP 8000H  
ORG 8000H  
MOV DPTR, #8500H  
MOVX A, @DPTR  
MOV B, A  
MUL AB  
INC DPTR  
MOVX @DPTR, A  
MOV A, B  
INC DPTR  
MOVX @DPTR, A  
LCALL 0003H
```

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc cộng 2 số 16-bit.

```
ORG 0000H  
LJMP 8000H  
ORG 8000H  
MOV DPTR, #8503H  
MOVX A, @DPTR  
MOV R1, A  
MOV DPTR, #8501H  
MOVX A, @DPTR  
ADD A, R1  
MOV DPTR, #9002H  
MOVX @DPTR, A  
MOV DPTR, #8502H  
MOVX A, @DPTR  
MOV R2, A  
MOV DPTR, #8500H  
MOVX A, @DPTR  
ADDC A, R2  
JNC NEXT  
INC R7
```

```

NEXT: MOV DPTR, #9001H
      MOVX @DPTR, A
      DEC DPL
      MOV A, R7
      MOVX @DPTR, A
      LCALL 0003H

```

### Yêu cầu thực hành:

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Có thể thực hiện việc cộng số 16-bit không sử dụng thanh ghi DPTR được không?
- Vận dụng mã nguồn ở trên, hãy viết chương trình thực hiện phép trừ 2 số 16-bit.
- Hãy cho biết ý nghĩa của 2 chương trình được cho sau đây:

#### Chương trình 1

<i>ORG 0000H</i>	<i>MOV B, R1</i>
<i>LJMP 8000H</i>	<i>MUL AB</i>
<i>ORG 8000H</i>	<i>MOV R7, A</i>
<i>MOV DPTR, #8501H</i>	<i>MOV R2, B</i>
<i>MOVX A, @DPTR</i>	<i>MOV A, R1</i>
<i>MOV R0, A</i>	<i>MOV B, R1</i>
<i>DEC DPL</i>	<i>MUL AB</i>
<i>MOVX A, @DPTR</i>	<i>ADD A, R2</i>
<i>MOV R1, A</i>	<i>MOV R6, A</i>
<i>MOV A, R0</i>	<i>MOV A, B</i>
<i>MOV B, A</i>	<i>MOV R5, A</i>
<i>MUL AB</i>	<i>MOV A, R3</i>
<i>MOV DPTR, #9005H</i>	<i>ADD A, R7</i>
<i>MOVX @DPTR, A</i>	<i>DEC DPL</i>
<i>MOV R2, B</i>	<i>MOVX @DPTR, A</i>
<i>MOV A, R0</i>	<i>MOV A, R4</i>
<i>MOV B, R1</i>	<i>ADDC A, R6</i>
<i>MUL AB</i>	<i>DEC DPL</i>
<i>ADD A, R2</i>	<i>MOVX @DPTR, A</i>
<i>MOV R3, A</i>	<i>MOV A, R5</i>
<i>MOV R2, B</i>	<i>DEC DPL</i>
<i>CLR A</i>	<i>MOVX @DPTR, A</i>
<i>ADDC A, R2</i>	<i>LCALL 0003H end</i>
<i>MOV A, R0</i>	

Chương trình 2

<i>ORG 0000H</i>	<i>MOV A,R1</i>
<i>MOV R0,#34H</i>	<i>MOV B,R2</i>
<i>MOV R1,#12H</i>	<i>MUL AB</i>
<i>MOV R2,#78H</i>	<i>ADD A,R5</i>
<i>MOV R3,#56H</i>	<i>MOV 32H,A</i>
<i>MOV A,R0</i>	<i>MOV A,B</i>
<i>MOV B,R2</i>	<i>ADDC A,R6</i>
<i>MUL AB</i>	<i>MOV 00H,C</i>
<i>MOV 33H,A</i>	<i>MOV R7,A</i>
<i>MOV R4,B</i>	<i>MOV A,R3</i>
<i>MOV A,R0</i>	<i>MOV B,R1</i>
<i>MOV B,R3</i>	<i>MUL AB</i>
<i>MUL AB</i>	<i>ADD A,R7</i>
<i>ADD A,R4</i>	<i>MOV 31H,A</i>
<i>MOV R5,A</i>	<i>MOV A,B</i>
<i>MOV A,B</i>	<i>ADDC A,20H</i>
<i>ADDC A,#00H</i>	<i>MOV 30H,A</i>
<i>MOV R6,A</i>	<i>END</i>

## BÀI THỰC HÀNH SỐ 3

### Mục đích:

- Giới thiệu các lệnh xử lý luận lý và các lệnh tăng giảm đơn vị.

### Sau khi kết thúc học phần này, sinh viên có thể:

- Hiểu về xử lý luận lý và các lệnh tăng giảm đơn vị.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ hợp ngữ dành cho các vi điều khiển họ 8051 để thực hiện các bộ đếm và xử lý luận lý cơ bản.

## Các lệnh về logic và rẽ nhánh

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc so sánh 2 số 8-bit là NUM1 và NUM2 được lưu tại bộ nhớ ngoài ở vùng nhớ 8000h và 8001h.

```
ORG 0000H
mov dptr, #8000h
movx a, @dptr
mov r0, a
inc dptr
movx a, @dptr
clr c
sub a, r0
jz equal
jnc small
setb 7fh
sjmp end1
small: setb 78h
      sjmp end1
equal: clr 78h
      clr 7fh
end1:
end
```

### Yêu cầu thực hành:

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Hãy chạy chương trình và cho biết kết quả sau khi thực hiện chương trình.
- Hãy cho biết ý nghĩa của chương trình được cho sau đây:

```
ORG 0000H
LJMP 8000H
ORG 8000H
MOV A, 20H
MOV R1, A
```

```

MOV A, 21H
CJNE R1, #0, CKOR
ANL A, 22H
MOV 23H, A
SJMP END1
CKOR: CJNE R1, #1, CKXOR
      ORL A, 22H
      MOV 23H, A
      SJMP END1
CKXOR: CJNE R1, #2H, OTHER
      XRL A, 22H
      MOV 23H, A
      SJMP END1
OTHER: CLR A
END1: MOV 23H, A
HERE: SJMP HERE
END

```

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc đếm số lượng bit có giá trị 0 và 1 có trong số N có 8-bit.

<pre> ORG 0000H MOV R1, #00H MOV R2, #00H MOV R7, #08H MOV A, #97H AGAIN: RLC A JC NEXT INC R1 SJMP HERE NEXT: INC R2 HERE: DJNZ R7, AGAIN END </pre>	
---	--

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc tạo bộ đếm lên số HEX.

```
ORG 0000H
LJMP 8000H
ORG 8000H
MOV A, #00H
UP: LCALL DELAY
    INC A
    CJNE A, #40H, UP
    LCALL 0003H
DELAY: MOV R0, #0FFH
BACK1: MOV R1, #0FFH
BACK: MOV R2, #0FFH
HERE: DJNZ R2, HERE
DJNZ R1, BACK
DJNZ R0, BACK1
RET
```

### **Yêu cầu thực hành:**

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Hãy chạy chương trình và cho biết kết quả sau khi thực hiện chương trình.
- Giả sử tần số hoạt động của hệ thống là 12Mhz, hãy tính thời gian CPU thực thi chương trình trên.
- Giải thích ý nghĩa lệnh, chương trình và sự thay đổi về giá trị tại các thanh ghi của chương trình được cho sau đây:

```

ORG 0000H
LJMP 8000H
ORG 8000H
MOV A, #020H
UP:    LCALL DELAY
       DEC A
       CJNE A, #00H, UP
       LCALL 0003H
DELAY: MOV R0, #0FFH
BACK1: MOV R1, #0FFH
BACK:  MOV R2, #0FFH
HERE:  DJNZ R2, HERE
       DJNZ R1, BACK
       DJNZ R0, BACK1
RET

```

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện việc tạo bộ đếm lên số BCD.

```

ORG 0000H
LJMP 8000H
ORG 8000H
MOV A, #00H
UP:    LCALL DELAY
       ADD A, #01H
       DA A
       CJNE A, #30H, UP
       LCALL 0003H
DELAY: MOV R0, #0FFH
BACK1: MOV R1, #0FFH
BACK:  MOV R2, #0FFH
HERE:  DJNZ R2, HERE
       DJNZ R1, BACK
       DJNZ R0, BACK1
RET

```

### **Yêu cầu thực hành:**

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Hãy chạy chương trình và cho biết kết quả sau khi thực hiện chương trình.
- Vận dụng mã nguồn ở trên, hãy viết chương trình thực hiện chức năng tương tự với số N có 16-bit.
- Giải thích ý nghĩa lệnh, chương trình và sự thay đổi về giá trị tại các thanh ghi của hai chương trình được cho sau đây:

Chương trình 1	Chương trình 2
<pre>MOV 30H, #81H MOV R0, 30H MOV R1, #08H MOV 31H, #00H CLR C BACK: MOV A, 30H        RLC A        MOV 30H, A        MOV A, 31H        RRC A        MOV 31H, A        DJNZ R1, BACK        CJNE A, 00H, NPAL        MOV A, #0FFH        SJMP NEXT NPAL: MOV A, #00H NEXT: END</pre>	<pre>ORG 0000H MOV A, 21H ANL A, 22H MOV 30H, A MOV A, 21H ORL A, 22H MOV 31H, A MOV A, 21H XRL A, 22H MOV 32H, A END</pre>

## BÀI THỰC HÀNH SỐ 4

### Mục đích:

- Giới thiệu phương pháp chuyển đổi giữa các hệ thống số.

### Sau khi kết thúc học phần này, sinh viên có thể:

- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ hợp ngữ dành cho các vi điều khiển họ 8051 để xây dựng các Chương trình chuyển đổi giá trị giữa các hệ thống số.

## Lập trình chuyển đổi giá trị trong hệ thống số

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện chuyển một số BCD thành ASCII.

```
ORG 0000H
MOV A, #09H
MOV R0,A
SWAP A
MOV DPTR, #9000H
ACALL ASCII
MOV A,R0
ACALL ASCII
SJMP $
ASCII: ANL A,#0FH
       ADD A,#30H
       MOVX @DPTR,A
       INC DPTR
       RET
END
```

### Yêu cầu thực hành:

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Hãy chạy chương trình và cho biết kết quả sau khi thực hiện chương trình.

Viết chương trình sử dụng ngôn ngữ hợp ngữ để thực hiện chuyển một số nhị phân (bin) sang số thập phân (DEC).

```
ORG 0000H
MOV A, #0FEH
MOV B, #0AH
DIV AB
MOV R0,B
MOV B, #0AH
DIV AB
MOV 30H,A
MOV A,B
SWAP A
ORL A,R0
MOV 31H,A
END
```

### **Yêu cầu thực hành:**

- Giải thích ý nghĩa lệnh và sự thay đổi về giá trị tại các thanh ghi trong chương trình trên.
- Hãy chạy chương trình và cho biết kết quả sau khi thực hiện chương trình.
- Vận dụng mã nguồn ở trên, hãy viết chương trình thực hiện: (a) chuyển số ASCII thành số thập phân (b) Chuyển số một số thập phân sang nhị phân.
- Thực hiện chuyển đổi giá trị được thể hiện trong các bảng sau:

Hệ 10	Hệ 2	Hệ 16
63		
256		
10		
512		
255		

Hệ 2	Hệ 16	Hệ 10
11010		
10000		
11110		
101010		
1111		

Hệ 16	Hệ 2	Hệ 10
3BC		
10A		
20		
FFF		
FA		
BAD		

## **PHẦN III: LẬP TRÌNH GIAO TIẾP 8051 VÀ NGOẠI VI SỬ DỤNG ASM**

Mục đích chung của phần này nhằm giúp người học vận dụng các lệnh (instruction) được hỗ trợ trong tập lệnh (instruction set) của vi điều khiển họ 8051 nhằm xây dựng các chương trình giao tiếp và điều khiển thiết bị ngoại vi cơ bản như LED đơn, LED 7 đoạn, phím ma trận,... Từ các ví dụ cơ bản được cung cấp trong phần này, người học có thể hiểu sâu cách sử dụng của từng mã lệnh, cấu trúc và sự ảnh hưởng của các thanh ghi, cũng như phương pháp lập trình điều khiển khi thực hiện các tác vụ từ máy tính trên chip 8051.

# BÀI THỰC HÀNH SỐ 1

## Mục đích:

- Giới thiệu về giao tiếp vi điều khiển và ngoại vi vào/ra cơ bản như công tắc/LED đơn.
- Giới thiệu các lệnh số học cơ bản cộng và trừ, lệnh so sánh và rẽ nhánh được hỗ trợ trong bộ ALU của vi xử lý.

## Sau khi kết thúc học phần này, sinh viên có thể:

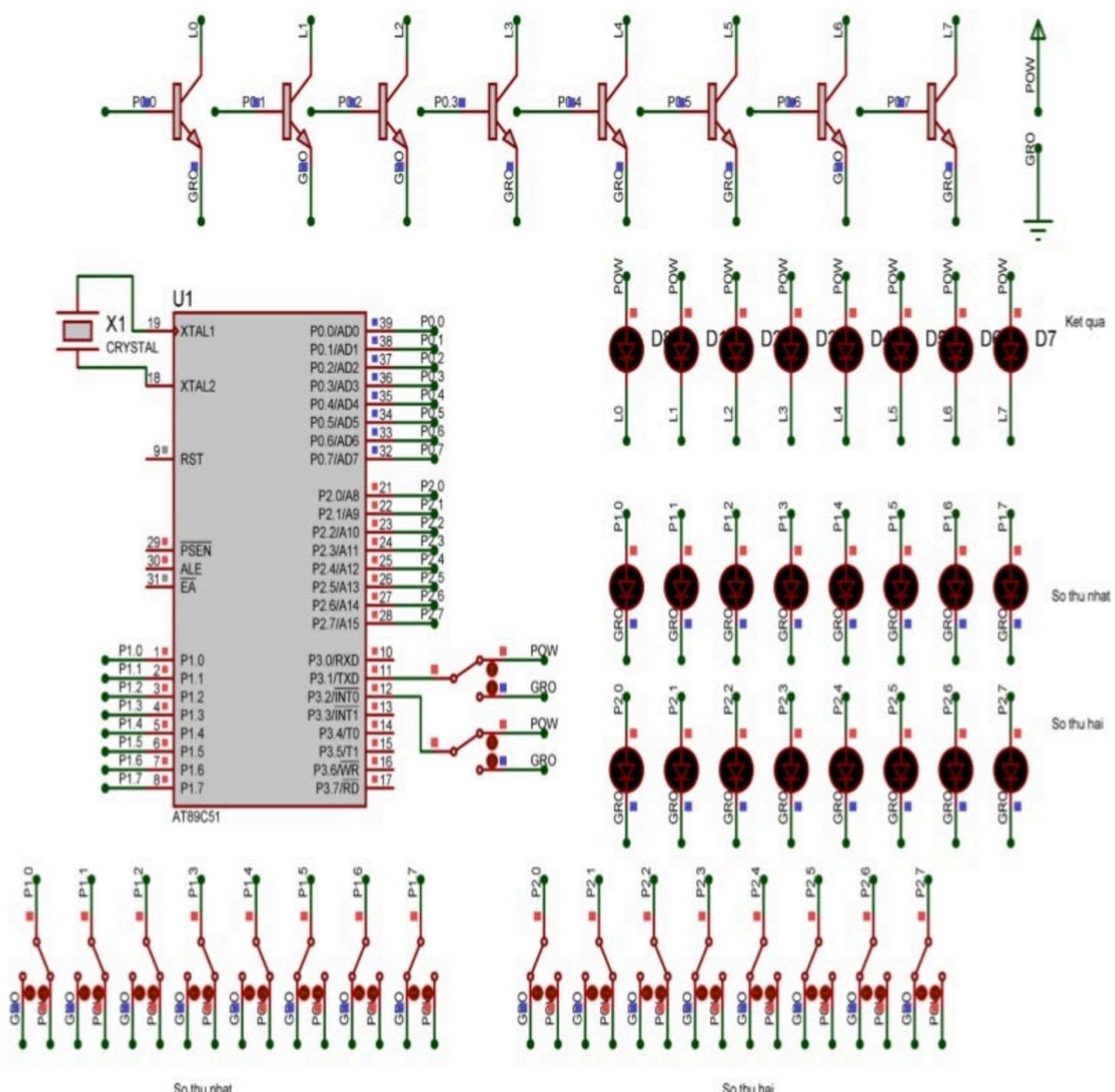
- Hiểu về giao tiếp vi điều khiển và ngoại vi vào/ra cơ bản như công tắc/LED đơn.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình hợp ngữ dành cho các vi điều khiển (máy tính trên chip) họ 8051 để thực hiện việc cộng/trừ hai số nhị phân khi giao tiếp với ngoại vi cơ bản như công tắc/LED đơn.

Thiết kế sơ đồ nguyên lý giao tiếp với ngoại vi vào/ra cơ bản công tắc/LED đơn theo phương pháp tích cực LED mức thấp như hình vẽ sau để thực hiện việc cộng/trừ 2 số nhị phân.

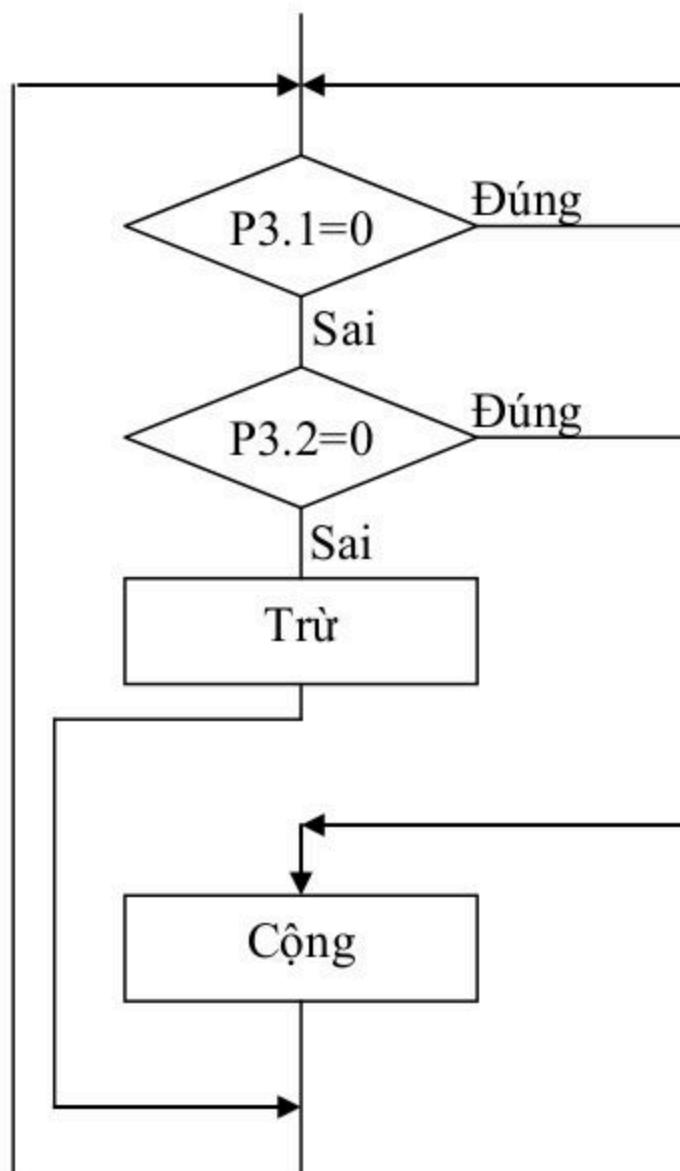
Trong đó:

- Vi điều khiển họ 8051 để thực hiện việc cộng/trừ hai số nhị phân.
- Dữ liệu nhập vào cho 2 số thông qua các công tắc kết nối vào vi điều khiển.
- Phép toán cộng/trừ được lựa chọn bởi 2 công tắc
- Dữ liệu tính toán được hiển thị trên LED đơn

Sử dụng phần mềm Proteus ISIS vẽ mạch điện sau:



**Hình 1:** Sơ đồ nguyên lý kết nối mạch điện cộng/trừ hai số nhị phân



**Hình 2:** Lưu đồ giải thuật

Sử dụng phần mềm Keil viết chương trình điều khiển như sau:

```

ORG 0H
    LJMP MAIN
MAIN:
    MOV P0, #0H
    ;MOV R1, #0H
WAIT: JNB P3.1, WAIT;
    MOV A, #0H
    JNB P3.2, ADD_OP
    MOV A, P1
    ADD A, P2
    MOV P0, A
    SJMP WAIT
ADD_OP:
    MOV A, #0H
    MOV A, P1
    CLR C
    SUBB A, P2
    MOV P0, A
    SJMP WAIT
END

```

### **Yêu cầu thực hành:**

- Từ lưu đồ như hình trên, hãy trình bày cách thức hoạt động của chương trình.
- Khảo sát chức năng của hệ thống sau khi biên dịch và nạp chương trình vào bộ nhớ của vi điều khiển.

## BÀI THỰC HÀNH SỐ 2

### Mục đích:

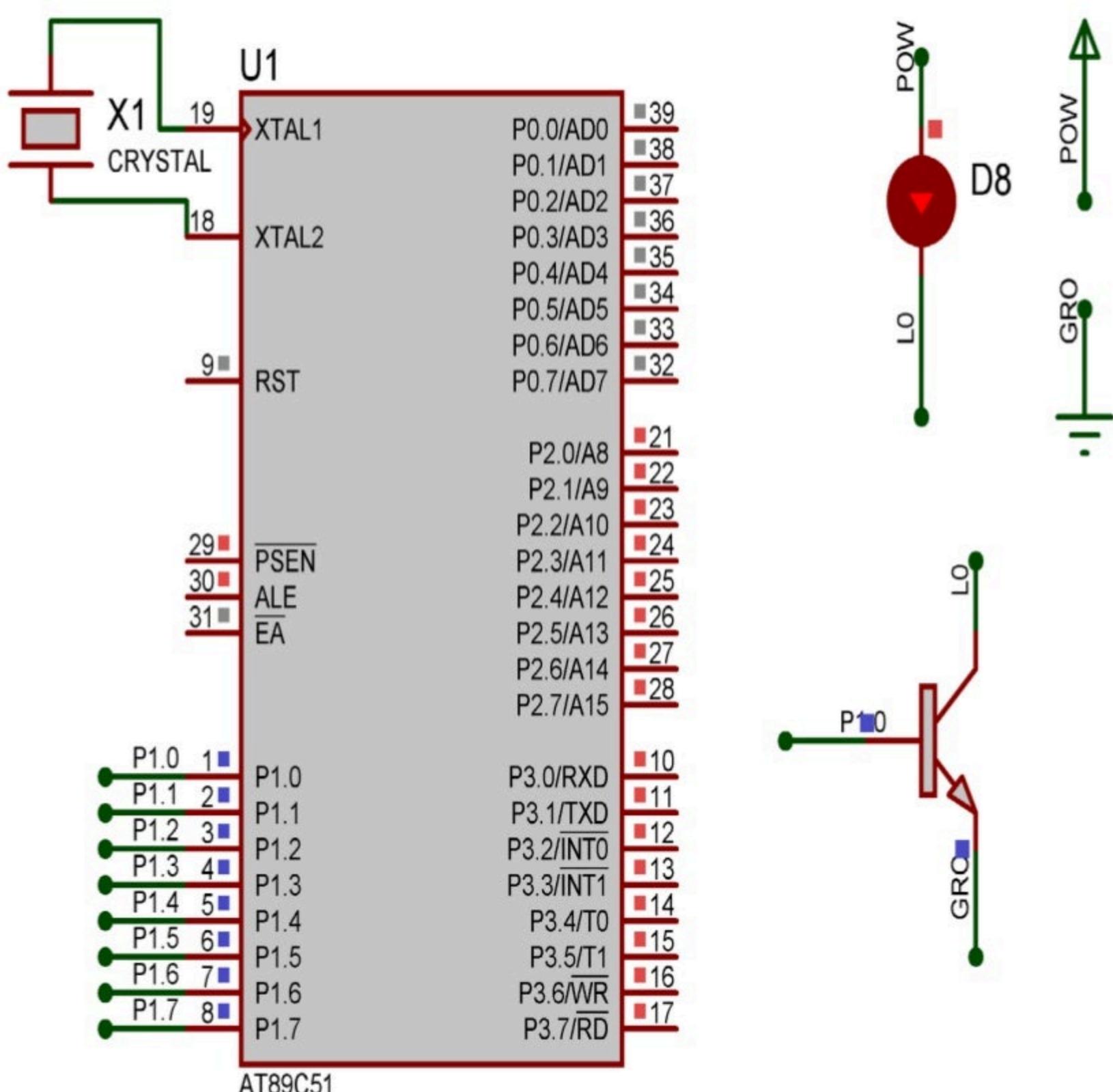
- Giới thiệu về bộ định thời được hỗ trợ trên khối ngoại vi của vi điều khiển họ 8051.
- Giới thiệu hoạt động ngắt khi điều khiển thiết bị ngoại vi đa dụng.

### Sau khi kết thúc học phần này, sinh viên có thể:

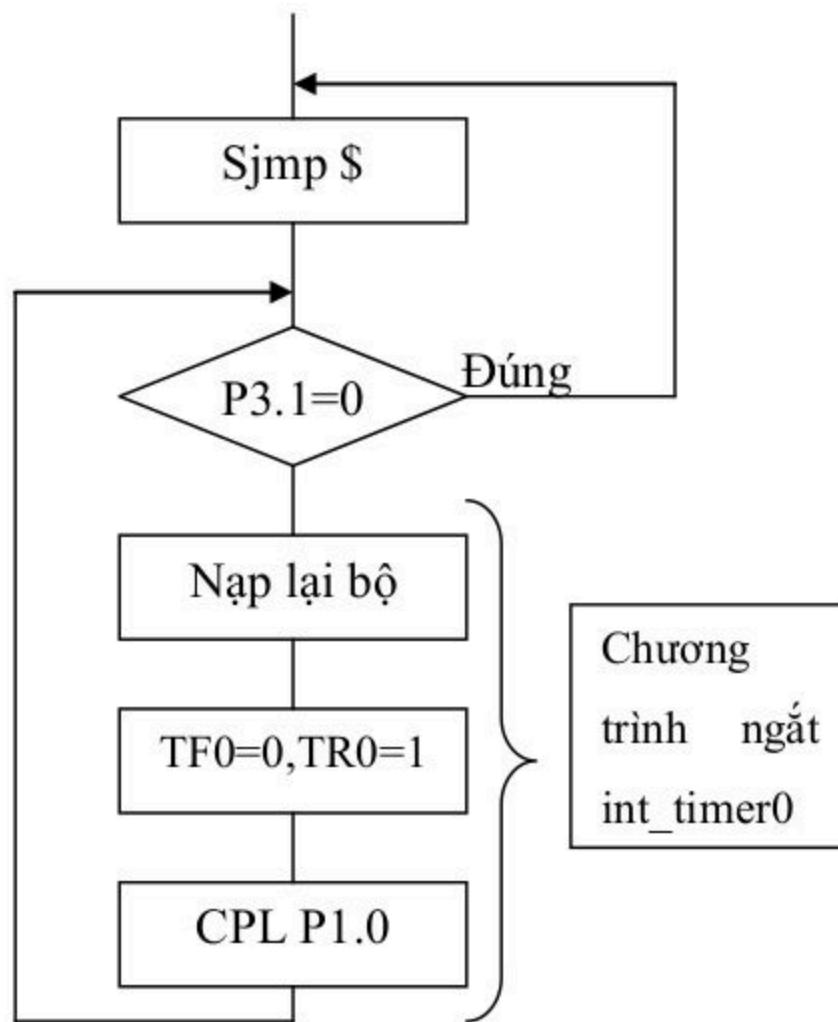
- Hiểu về phương pháp giao tiếp thiết bị ngoại vi thông qua ngắt.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ hợp ngữ dành cho các vi điều khiển họ 8051 để thực hiện việc xuất xung với tần số 100Hz.

Thiết kế sơ đồ nguyên lý của bộ phát xung tần số 100Hz, trong đó:

- Vi điều khiển họ 8051 để thực hiện việc tạo xung tần số 100Hz dựa trên ngắt của bộ timer.
  - Trạng thái xung ngõ ra được hiển thị trên LED đơn.
- Sử dụng phần mềm Proteus ISIS vẽ mạch điện sau:



**Hình 3:** Sơ đồ nguyên lý mạch xuất xung với tần số 100Hz



**Hình 4:** Lưu đồ giải thuật

Sử dụng phần mềm Keil viết chương trình điều khiển như sau:

Sử dụng hàm ngắt int\_timer0 (ngắt timer0), ngắt định kỳ cứ 5 miligiây thì ngắt một lần, mỗi lần ngắt thì đảo P1.0 để tạo xung 100Hz.

```

ORG 0H
    LJMP MAIN
ORG 03H
    LJMP EX_INT0;NGAT NGOAI 0
ORG 0BH
    LJMP TIMER_INT0;NGAT TIMER0
ORG 13H
    LJMP EX_INT1;NGAT NGOAI 1
ORG 1BH
    LJMP TIMER_INT1;NGAT TIMER1
ORG 23H
    LJMP LE_INT;NGAT NOI TIEP
ORG 30H
MAIN:
    MOV P1,#0H
    CLR TR0
    MOV TMOD,#11H
    MOV TL0,#LOW(-10000)

```

```

MOV TH0, #HIGH(-10000)
MOV IE, #8FH
SETB TR0
SJMP $
EX_INT0:
RETI
; *****
; (100HZ)
; *****
TIMER_INT0:
CLR TR0
CLR TF0
MOV TL0, #LOW(-5000)
MOV TH0, #HIGH(-5000)
SETB TR0
CPL P1.0
END1:
RETI
EX_INT1:
RETI
TIMER_INT1:
RETI
LE_INT:
RETI
END

```

### **Yêu cầu thực hành:**

- Từ cấu trúc phần cứng của vi điều khiển họ 8051 hãy giải thích ý nghĩa của các thanh ghi TMOD, IE.
- Giải thích tại sao cần phải xóa bit TR0 trong hàm main.
- Khảo sát chức năng của hệ thống sau khi biên dịch và nạp chương trình vào bộ nhớ của vi điều khiển.
- Nếu chỉnh sửa chương trình để tạo xung 10Hz nhằm điều khiển LED đơn, thì giá trị cần nạp thanh ghi TL0 và TH0 là bao nhiêu?
- Chạy đoạn chương trình sau, cho biết ý nghĩa của đoạn chương trình.

```
ORG 0000H
MOV TMOD, #02H
MOV TH0, #00H
CLR P1.0
CLR A
SETB TR0
AGAIN: MOV R7, #0FFH
LOOP: MOV R6, #14D
WAIT: JNB TF0, WAIT
CLR TF0
DJNZ R6, WAIT
DJNZ R7, LOOP
CPL P1.0
SJMP AGAIN
END
```

# BÀI THỰC HÀNH SỐ 3

## Mục đích:

- Giới thiệu về bộ định thời được hỗ trợ trên khối ngoại vi của vi điều khiển họ 8051.
- Giới thiệu phương pháp quét phím ma trận.
- Giới thiệu về kỹ thuật giải mã hiển thị ra thiết bị ngoại vi LED 7 đoạn.

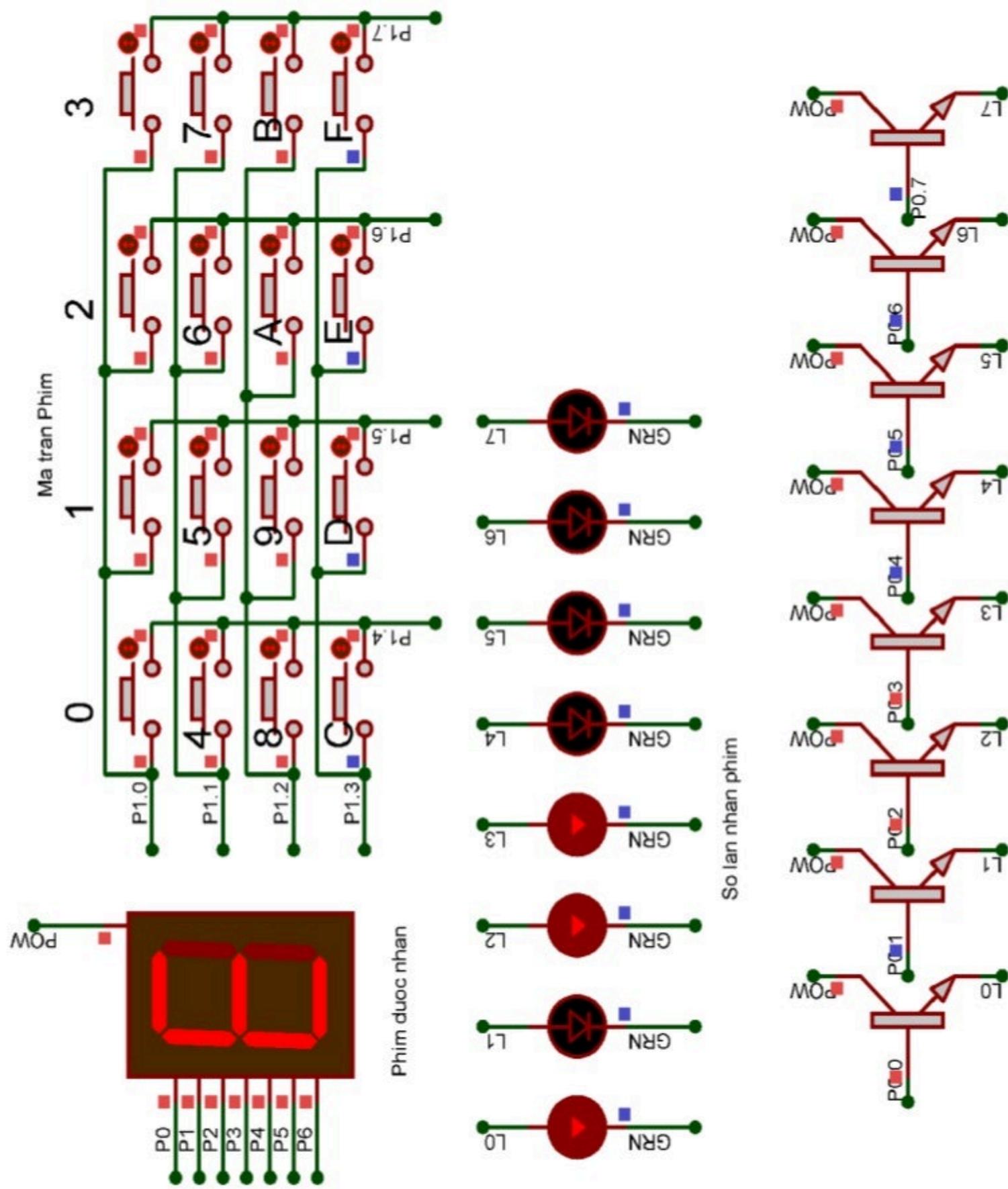
## Sau khi kết thúc học phần này, sinh viên có thể:

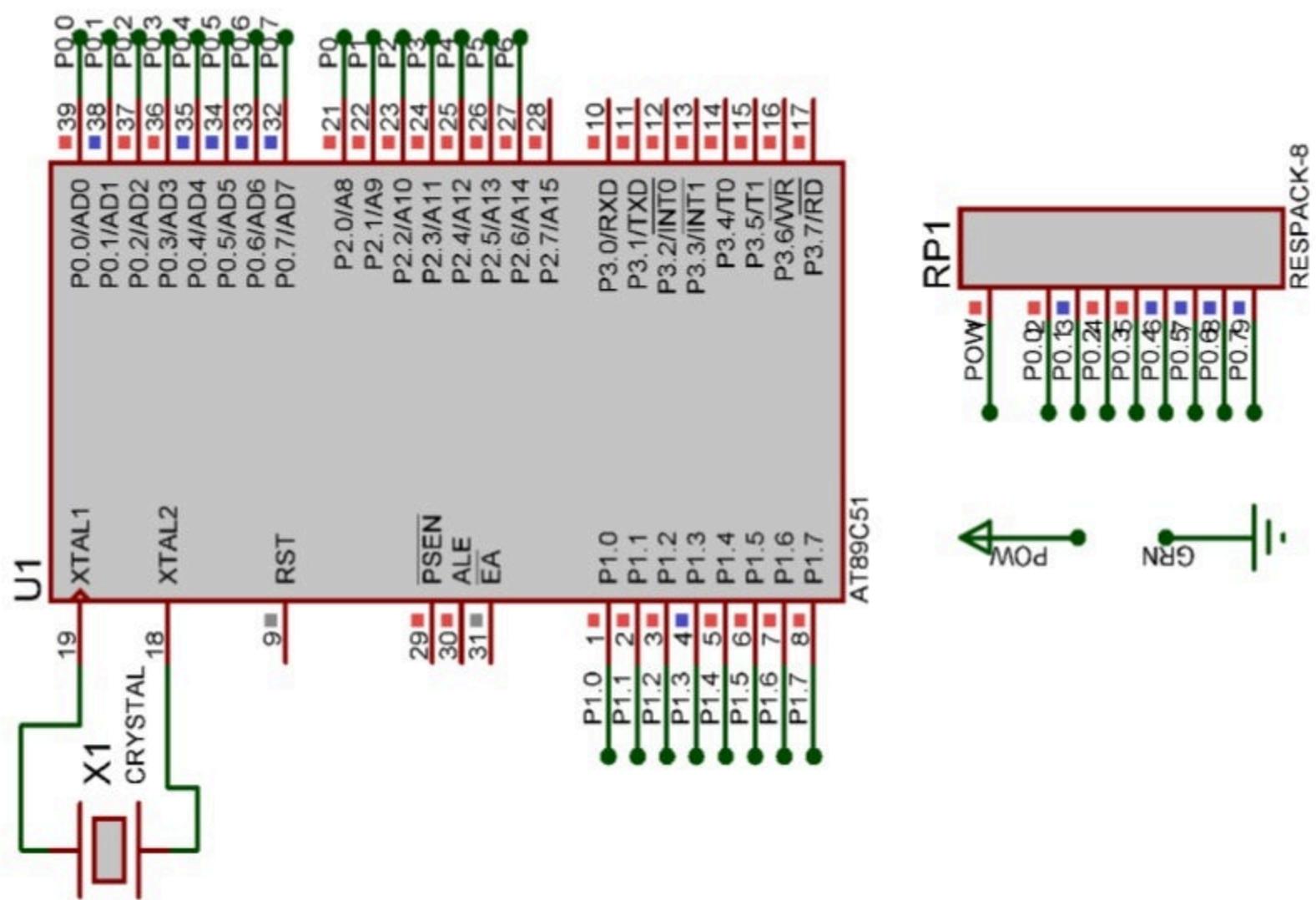
- Hiểu về phương pháp giao tiếp thiết bị ngoại vi bàn phím thông qua ngắt.
- Hiểu về phương pháp hiển thị qua LED 7 đoạn.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ hợp ngữ dành cho các vi điều khiển họ 8051 để thực hiện việc quét phím ma trận và hiển thị vị trí nhấn phím.

Thiết kế sơ đồ nguyên lý của bộ quét và hiển thị vị trí và số lần nhấn phím, trong đó:

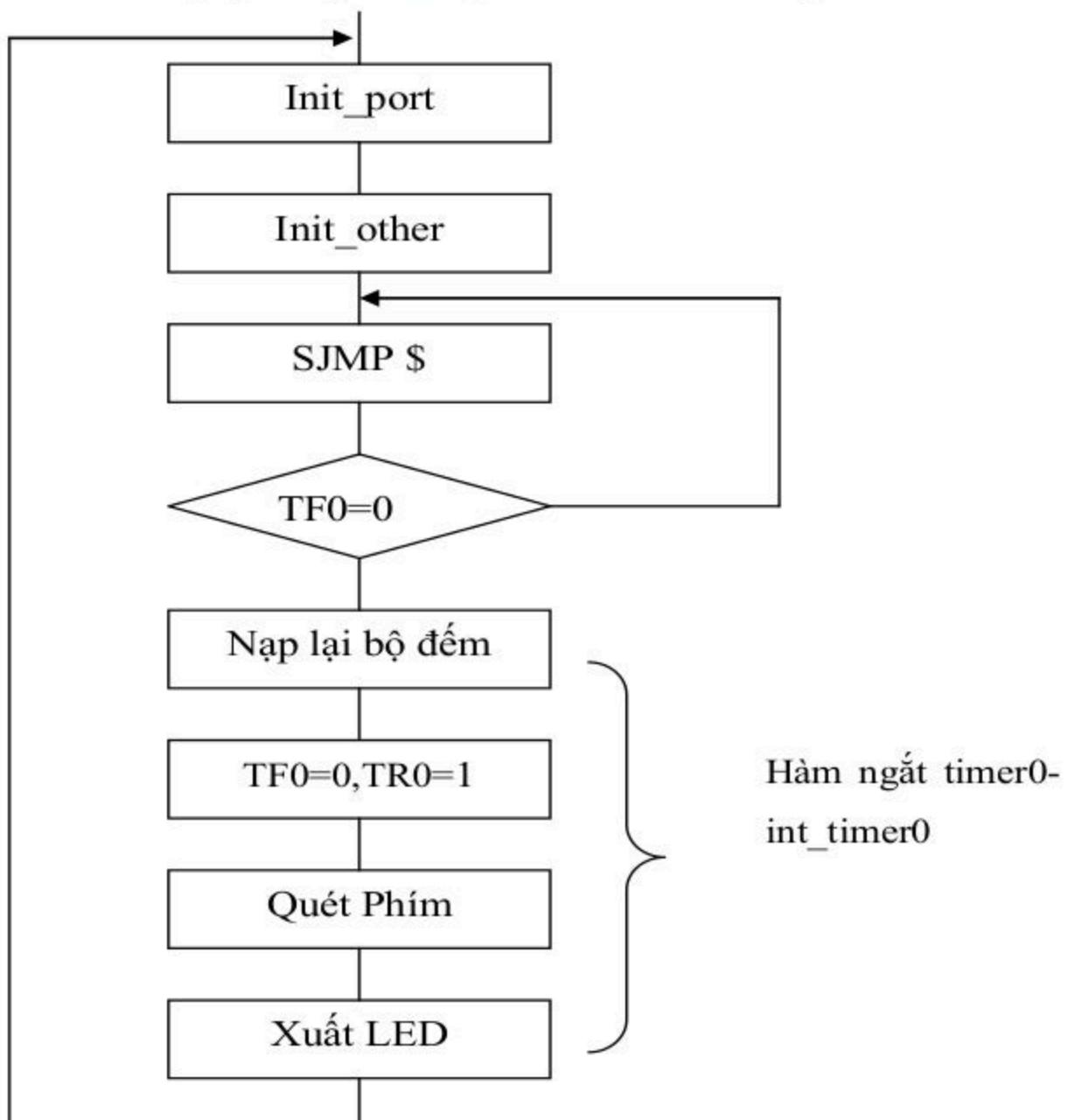
- Vi điều khiển họ 8051 để thực hiện việc giao tiếp với bàn phím, LED đơn, LED 7 đoạn.
- Bàn phím là thiết bị ngõ vào được kết nối qua cổng P1.
- Số lần nhấn phím được hiển thị qua LED đơn ở dạng nhị phân qua cổng P0.
- Vị trí phím nhấn từ bàn phím được hiển thị qua LED 7 đoạn qua cổng P2.

Sử dụng phần mềm Proteus ISIS vẽ mạch điện sau:





**Hình 5:** Sơ đồ nguyên lý nhận phím và hiển thị phím lên LED 7 đoạn



**Hình 6:** Lưu đồ giải thuật

Sử dụng phần mềm Keil viết chương trình điều khiển như sau:

- Sử dụng hàm int\_timer0 để quét phím (hàm này cứ 5ms thì kiểm tra phím một lần).
- Sử dụng hàm int\_timer1 để xuất kết quả phím nhận được ra LED 7 đoạn.

<pre> ORG 0H VARIABLE DATA 30H TIME_PRESS DATA 31H SE0 BIT 00H SE1 BIT 01H SE2 BIT 02H SE3 BIT 03H SE4 BIT 04H SE5 BIT 05H SE6 BIT 06H SE7 BIT 07H SE8 BIT 08H SE9 BIT 09H SE10 BIT 0AH SE11 BIT 0BH SE12 BIT 0CH SE13 BIT 0DH SE14 BIT 0EH SE15 BIT 0FH ORG 0H         LJMP MAIN ORG 0BH         LJMP INT_TIMER0 ORG 1BH         LJMP INT_TIMER1 MAIN:         LCALL INIT_PORT         LCALL INIT_OTHER         SJMP \$ ;***** *****; ;KHOI DONG CAC CONG </pre>	<pre> KEY7: SETB SE6         JB SE7,K7         LJMP EXIT K7:    CLR SE7         MOVA,#7         MOVC A,@A+DPT         MOV VARIABLE,A         INC TIME_PRESS         LJMP EXIT CONTINUE2:         MOVP1,#0FBH         MOVA,P1         CJNE A,#0FBH,PRESS2         SJMP CONTINUE3 PRESS2:;QUET HANG 2         SETB SE0         SETB SE1         SETB SE2         SETB SE3         SETB SE4         SETB SE5         SETB SE6         SETB SE7         SETB SE12         SETB SE13         SETB SE14         SETB SE15         CJNE A,#0EBH,KEY9;A=EBH - &gt;KEY8         JB SE8,K8         LJMP EXIT K8:    CLR SE8 </pre>
---	--

<pre> CUA VI XULY ;***** *****  INIT_PORT:     MOV P1,#0H     MOV P2,#0H     MOV     DPTR,#TABLE RET ;***** *****  ;KHOI DONG THONG SO KHAC ;***** *****  INIT_OTHER:     MOV VARIABLE,#0H     MOV TIME_PRESS,#0H     MOVI E,#1011111B     MOV TMOD,#1IH     MOV TH0,#HIGH(-100)     MOV TL0,#LOW(-100)     CLR TF0     SETB TR0     ; ;KHOI DONG CAC PHIM     SETB SE0     SETB SE1     SETB SE2     SETB SE3     SETB SE4     SETB SE5     SETB SE6     SETB SE7     SETB SE8     SETB SE9     SETB SE10     SETB SE11     SETB SE12 </pre>	<pre> MOV A,#8;KEY8 MOVC A,@A+DPTR MOV VARIABLE,A INC TIME_PRESS LJMP EXIT KEY9: SETB SE8 CJNE A,#0DBH,KEY10;A==DBH -&gt;KEY9 JB SE9,K9 LJMP EXIT K9: CLR SE9 MOV A,#9 MOVC A,@A+DPTR MOV VARIABLE,A INC TIME_PRESS LJMP EXIT KEY10: SETB SE9 CJNE A,#0BBH,KEY11; JB SE10,K10 LJMP EXIT K10: CLR SE10 MOV A,#10 MOVC A,@A+DPTR MOV VARIABLE,A INC TIME_PRESS LJMP EXIT KEY11: SETB SE10 JB SE10,K11 LJMP EXIT K11: MOV A,#11 MOVC A,@A+DPTR MOV VARIABLE,A INC TIME_PRESS LJMP EXIT CONTINUE3:     MOV P1,#0F7H     MOV A,P1     CJNE A,#0F7H,PRESS3     ; </pre>
---	---

<pre> SETB SE13 SETB SE14 SETB SE15 ; RET ;***** ***** ;HAM NGAT TIMER 0 THUC HIEN VIEC ;QUET PHIM, 5MS THI QUET MOT LAN ;***** ***** INT_TIMER0:     CLR TF0     CLR TR0 MOV TL0,#LOW(-50000) MOV TH0,#HIGH(-50000)     SETB TR0     MOV P1,#0FEH     MOVA,P1 CJNE A,#0FEH,PRESS0;NEU A KHONG BANG 1111 1110 THI NHAY SJMP CONTINUE1 PRESS0:;QUET HANG 0     SETB SE4     SETB SE5     SETB SE6     SETB SE7     SETB SE8     SETB SE9     SETB SE10     SETB SE11     SETB SE12     SETB SE13     SETB SE14     SETB SE15 ; SJMP EXIT </pre>	<pre> SETB SE0 SETB SE1 SETB SE2 SETB SE3 SETB SE4 SETB SE5 SETB SE6 SETB SE7 SETB SE8 SETB SE9 SETB SE10 SETB SE11 SETB SE12 SETB SE13 SETB SE14 SETB SE15 ; SJMP EXIT PRESS3: SETB SE0;QUET HANG 3 SETB SE1 SETB SE2 SETB SE3 SETB SE4 SETB SE5 SETB SE6 SETB SE7 SETB SE8 SETB SE9 SETB SE10 SETB SE11 SETB SE12 CJNE A,#0E7H,KEY13;A=E7H - &gt;KEY12 JB SE12,K12 LJMP EXIT K12: CLR SE12 MOVA,#12;KEY12 MOVC A,@A+DPTR MOV VARIABLE,A </pre>
--	--

<p><i>CJNE</i>  <i>A,#0EEH,KEY1;A==EEH -&gt;KEY0</i>  <i>KEY0: JB SE0,K0;SE0</i>  <i>DANG 1 THI NHAY</i>  <i>LJMP EXIT</i>  <i>K0: CLR SE0;</i>  <i>MOV A,#0;KEY0</i>  <i>MOVC A,@A+DPTR</i>  <i>MOV VARIABLE,A</i>  <i>INC TIME_PRESS</i>  <i>LJMP EXIT</i>  <i>KEY1:</i>  <i>SETB SE0;XOA KEY0</i>  <i>CJNE</i>  <i>A,#0DEH,KEY2;A==DEH -&gt;KEY1</i>  <i>JB SE1,K1;SE1</i>  <i>DANG O 0 THI NHAY</i>  <i>LJMP EXIT</i>  <i>K1: CLR SE1</i>  <i>MOV A,#1</i>  <i>MOVC A,@A+DPTR</i>  <i>MOV VARIABLE,A</i>  <i>INC TIME_PRESS</i>  <i>LJMP EXIT</i>  <i>KEY2: SETB SE1;XOAT</i>  <i>KEY1</i>  <i>CJNE</i>  <i>A,#0BEH,KEY3;</i>  <i>JB SE2,K2</i>  <i>LJMP EXIT</i>  <i>K2: CLR SE2</i>  <i>MOV A,#2</i>  <i>MOVC</i>  <i>A,@A+DPTR</i>  <i>MOV VARIABLE,A</i>  <i>INC TIME_PRESS</i>  <i>LJMP EXIT</i></p>	<p><i>INC TIME_PRESS</i>  <i>SJMP EXIT</i>  <i>KEY13: SETB SE12</i>  <i>CJNE A,#0D7H,KEY14;A==D7H</i>  <i>-&gt;KEY13</i>  <i>JB SE13,K13</i>  <i>LJMP EXIT</i>  <i>K13: CLR SE13</i>  <i>MOV A,#13</i>  <i>MOVC A,@A+DPTR</i>  <i>MOV VARIABLE,A</i>  <i>INC TIME_PRESS</i>  <i>SJMP EXIT</i>  <i>KEY14: SETB SE13</i>  <i>CJNE A,#0B7H,KEY15;</i>  <i>JB SE14,K14</i>  <i>LJMP EXIT</i>  <i>K14: CLR SE14</i>  <i>MOV A,#14</i>  <i>MOVC A,@A+DPTR</i>  <i>MOV VARIABLE,A</i>  <i>INC TIME_PRESS</i>  <i>SJMP EXIT</i>  <i>KEY15: SETB SE14</i>  <i>JB SE15,K15</i>  <i>LJMP EXIT</i>  <i>K15: CLR SE15</i>  <i>MOV A,#15</i>  <i>MOVC A,@A+DPTR</i>  <i>MOV VARIABLE,A</i>  <i>INC TIME_PRESS</i>  <i>EXIT:</i>  <i>MOV P2,VARIABLE</i>  <i>MOV P0,TIME_PRESS</i>  <i>RETI</i>  <del>*****</del>  <del>*****</del>  <del>;NGAT TIMER 1: THUC HIEN VIEC</del>  <del>DUA PHIM QUET DUOC RA CONG P2</del></p>
---	--

<p><i>KEY3: SETB SE2 JB SE3,K3 LJMP EXIT</i></p> <p><i>K3: CLR SE3 MOVA,#3 MOVC A,@A+DPTR MOV VARIABLE,A INC TIME_PRESS LJMP EXIT</i></p> <p><i>CONTINUE1: MOV P1,#0FDH MOVA,P1 CJNE A,#0FDH,PRESS1 LJMP CONTINUE2</i></p> <p><i>PRESS1:;QUET HANG 1 SETB SE0 SETB SE1 SETB SE2 SETB SE3 SETB SE8 SETB SE9 SETB SE10 SETB SE11 SETB SE12 SETB SE13 SETB SE14 SETB SE15 CJNE A,#0EDH,KEY5;A=EDH -&gt;KEY4</i></p> <p><i>KEY4: JB SE4,K4 LJMP EXIT</i></p> <p><i>K4: CLR SE4 MOVA,#4;KEY4 MOVC A,@A+DPTR MOV VARIABLE,A INC TIME_PRESS</i></p>	<pre>;***** ***** <i>INT_TIMER1: CLR TR1 CLR TF1 MOV TH1,#HIGH(-1000) MOV TL1,#LOW(-1000) SETB TR1 MOV P2,VARIABLE</i> <i>RETI</i> ;***** ***** <i>;BANG DU LIEU</i> ;***** ***** <i>TABLE: DB 0C0H;0 DB 0F9H;1 DB 0A4H;2 DB 0B0H;3 DB 099H;4 DB 092H;5 DB 082H;6 DB 0F8H;7 DB 080H;8 DB 090H;9 DB 088H;A DB 083H;B DB 0C6H;C DB 0A1H;D DB 086H;E DB 08EH;F</i> <i>END</i></pre>
--	--

<p><i>LJMP EXIT</i></p> <p><i>KEY5: SETB SE4</i></p> <p><i>CJNE</i></p> <p><i>A, #0DDH, KEY6; A == DEH</i></p> <p><i>-&gt; KEY1</i></p> <p><i>JB SE5, K5</i></p> <p><i>LJMP EXIT</i></p> <p><i>K5: CLR SE5</i></p> <p><i>MOV A, #5</i></p> <p><i>MOVC</i></p> <p><i>A, @A + DPTR</i></p> <p><i>MOV VARIABLE, A</i></p> <p><i>INC TIME_PRESS</i></p> <p><i>LJMP EXIT</i></p> <p><i>KEY6: SETB SE5</i></p> <p><i>CJNE</i></p> <p><i>A, #0BDH, KEY7;</i></p> <p><i>JB SE6, K6</i></p> <p><i>LJMP EXIT</i></p> <p><i>K6: CLR SE6</i></p> <p><i>MOV A, #6</i></p> <p><i>MOVC</i></p> <p><i>A, @A + DPTR</i></p> <p><i>MOV VARIABLE, A</i></p> <p><i>INC TIME_PRESS</i></p> <p><i>LJMP EXIT</i></p>	
---	--

### **Yêu cầu thực hành:**

- Khảo sát chức năng của hệ thống sau khi biên dịch và nạp chương trình vào bộ nhớ của vi điều khiển.
- Tìm hiểu phương pháp giao tiếp ngoại vi ngõ vào thông qua ngắt ngoài qua cổng P3.

## BÀI THỰC HÀNH SỐ 4

### Mục đích:

- Giới thiệu về kỹ thuật giải mã và quét hiển thị ra thiết bị ngoại vi LED 7 đoạn

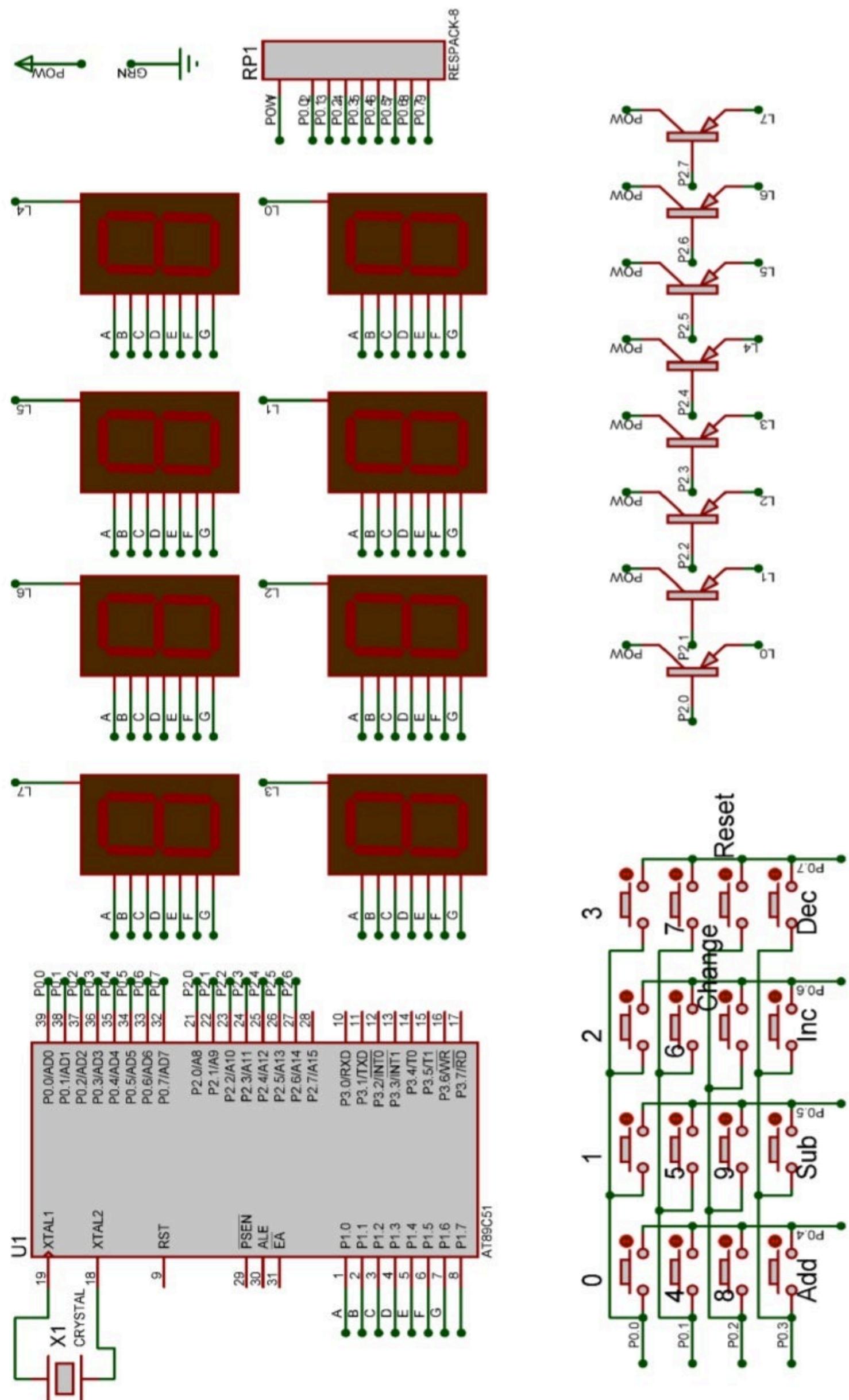
### Sau khi kết thúc học phần này, sinh viên có thể:

- Hiểu về phương pháp quét hiển thị thông tin qua LED 7 đoạn.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ hợp ngữ dành cho các vi điều khiển họ 8051 để thực hiện việc thực hiện tính toán cơ bản cộng/trừ 2 số có 2 chữ số được nhập từ bàn phím ma trận và hiển thị kết quả lên LED 7 đoạn.

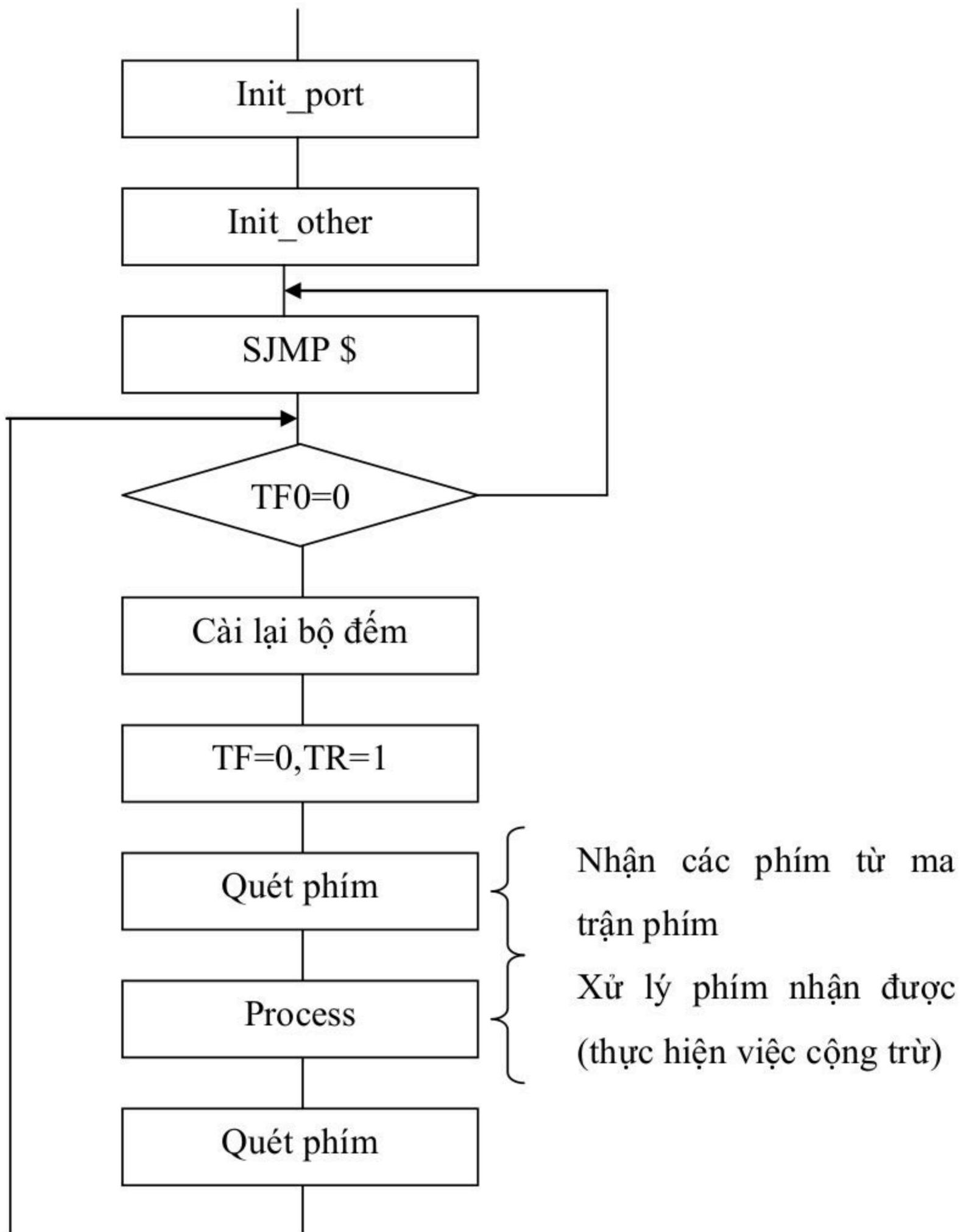
Thiết kế sơ đồ nguyên lý của bộ cộng/trừ 2 số có 2 chữ số, trong đó:

- Vi điều khiển họ 8051 để thực hiện việc giao tiếp với ngoại vi vào/ra và thực hiện các phép tính toán số học cơ bản.
- Bàn phím là thiết bị ngõ vào được kết nối qua cổng P0.
- Dữ liệu 2 số có 2 chữ số được nhập vào hệ thống qua phím nhấn và được hiển thị với 4 LED 7 đoạn, kết quả tính toán được hiển thị với 4 LED kết nối chung qua cổng P1.
- Trạng thái hoạt động của LED 7 đoạn được cho phép thông qua cổng P2.

Sử dụng phần mềm Proteus ISIS vẽ mạch điện sau:



Hình 7: Mạch nguyên lý cộng trừ hai số có hai chữ số



**Hình 8:** Lưu đồ giải thuật

Sử dụng phần mềm Keil viết chương trình điều khiển như sau:

Xây dựng các hàm con:

- Int\_timer0: hàm ngắt timer0 cứ 1 miligiây thì ngắt một lần. Hàm này thực hiện việc gọi hàm quét phím, hàm xử lý và hàm quét phím.

- Int\_timer1: hàm ngắt timer1 cứ 1 miligiây thì quét 1 lần
- Quet\_led: để xuất dữ liệu ra LED 7 đoạn.
- Process: Hàm xử lý phím bắt được.
- Quet\_phim: hàm quét phím.
- Ma trận phím được đánh số 0 đến 15 từ trái qua phải trên xuống dưới.
- Nhấn các phím từ 0 đến 9 để nhập hai thông số cần tính toán (hai số này sẽ được hiển thị trên 4 LED 7 đoạn, từ LED 0 đến 3).
- Nút thứ 10: (*change*) để thay đổi số nhập (ví dụ đang nhập số thứ 2 nhấn nút này sẽ chuyển sang nhập số thứ nhất).
- Nút 11: (*reset*) xóa hai số vừa nhập (đưa hai số đó về 0).
- Nút 12: (*add*) thực hiện phép cộng hai số vừa nhập (kết quả hiển thị trên 3 LED 4, 5, 6).
- Nút 13: (*sub*) thực hiện phép trừ hai số vừa nhập (kết quả hiển thị trên 3 LED 4, 5, 6).
- Nút 14: (*inc*) tăng số đang nhập lên 1 đơn vị (ví dụ đang nhập số thứ nhất thì nếu nhấn nút này số thứ nhất sẽ được cộng thêm 1 đơn vị).
- Nút 15: (*dec*) giảm số đang nhập xuống 1 đơn vị (ví dụ đang nhập số thứ nhất thì nếu nhấn nút này số thứ nhất sẽ được cộng trừ đi 1 đơn vị).

<pre> variable data 30h time_press data 31h se0 bit 00h se1 bit 01h se2 bit 02h se3 bit 03h se4 bit 04h se5 bit 05h se6 bit 06h se7 bit 07h se8 bit 08h se9 bit 09h se10 bit 0ah </pre>	<pre> K_13_15: CJNE R2,#0DFH,K_14_15 ;TRU MOV TEMP1,NUMBER1_1 MOV A,NUMBER1_2 MOV B,#10 MUL AB ADD A,TEMP1 MOV TEMP1,A ; MOV TEMP2,NUMBER2_1 MOV A,NUMBER2_2 MOV B,#10 </pre>
---	---

<pre> se11 bit 0bh se12 bit 0ch se13 bit 0dh se14 bit 0eh se15 bit 0fh CONTROL BIT 10H; CONTROL=0 -&gt; DUNG CHO SO THU NHAT, =1 DUNG CHO SO THU 2 KEY_STATUS0 DATA 32H KEY_STATUS1 DATA 33H NUMBER1_1 DATA 33H NUMBER1_2 DATA 34H NUMBER2_1 DATA 35H NUMBER2_2 DATA 36H RESULT1 DATA 37H RESULT2 DATA 38H RESULT3 DATA 39H TEMP1 DATA 3AH TEMP2 DATA 3BH ORG 0h         Ljmp main;ham chinh org 03h         Ljmp ex_int0;ngat ngoai 0 org 0bh         Ljmp int_timer0;ngat timer 0 org 13h         Ljmp ex_int1;ngat ngoai 1 org 1bh         Ljmp int_timer1;ngat timer 1 org 30h main: ;***** **  ;Chuong trinh chinh ;***** **          Lcall init_port </pre>	<pre> MUL AB ADD A, TEMP2 MOV TEMP2,A ; MOV A, TEMP1 CLR C SUBB A, TEMP2 MOV B, #10 DIV AB MOV RESULT1, 0F0H MOV RESULT2,A MOV RESULT3, #0H         LJMP END_PROCESS K_14_15: CJNE R2, #0BFH, K_15_15 ;TANG JB CONTROL, NUM2_14_15 MOV TEMP1, NUMBER1_1 MOV A, NUMBER1_2 MOV B, #10 MUL AB ADD A, TEMP1 INC A MOV TEMP1,A ; MOV B, #10 DIV AB MOV NUMBER1_1, 0F0H MOV NUMBER1_2,A SJMP END_14_15 ; NUM2_14_15: MOV TEMP2, NUMBER2_1 MOV A, NUMBER2_2 MOV B, #10 MUL AB ADD A, TEMP2 </pre>
---	--

<pre> lcall init_other sjmp \$ ;***** **  ;Hien thuc cac ham ;***** **  init_port:;khoi tao cac cong cho vi dieu khien     mov p1,#0h     mov p2,#0ffh     mov DPTR,#table ret init_other:;khoi tao cac thong so thanh ghi khac     mov variable,#0ffh     mov time_press,#0h     mov ie,#10111111b     mov tmod,#11h     mov th0,#high(-50000)     mov tl0,#Low(-50000)     clr tf0     setb tr0     MOV TH1,#HIGH(-1000)     MOV TL1,#LOW(-1000)     CLR TF0     SETB TR0     ;     setb se0     setb se1     setb se2     setb se3     setb se4     setb se5     setb se6     setb se7     setb se8     setb se9     setb se10 </pre>	<pre> INC A MOV TEMP2,A ; MOV B,#10 DIV AB MOV NUMBER2_1,0F0H MOV NUMBER2_2,A END_14_15:LJMP END_PROCESS K_15_15: ;GIAM JB CONTROL,NUM2_15_15 MOV TEMP1,NUMBER1_1 MOV A,NUMBER1_2 MOV B,#10 MUL AB ADD A,TEMP1 DEC A MOV TEMP1,A ; MOV B,#10 DIV AB MOV NUMBER1_1,0F0H MOV NUMBER1_2,A SJMP END_15_15 ; NUM2_15_15: MOV TEMP2,NUMBER2_1 MOV A,NUMBER2_2 MOV B,#10 MUL AB ADD A,TEMP2 DEC A MOV TEMP2,A ; MOV B,#10 DIV AB MOV NUMBER2_1,0F0H </pre>
--	---

<pre> setb se11 setb se12 setb se13 setb se14 setb se15 ; MOV R3,#0FEH; SU DUNG CHO QUET LED ; SETB CONTROL MOV KEY_STATUS0,#0FFH MOV KEY_STATUS1,#0FFH MOV NUMBER1_1,#0H MOV NUMBER1_2,#0H MOV NUMBER2_1,#0H MOV NUMBER2_2,#0H MOV RESULT1,#0H MOV RESULT2,#0H MOV RESULT3,#0H MOV TEMP1,#0H MOV TEMP2,#0H ret ex_int0: reti ;50MS THI QUET MOT LAN int_timer0: clr tf0 clr tr0 mov tl0,#Low(-50000) mov th0,#high(-50000) setb tr0 lcall quet_phim lcall process LCALL QUET_LED reti ex_int1: reti ;1MS THI QUET MOT LAN int_timer1: </pre>	<pre> MOV NUMBER2_2,A END_15_15: END_PROCESS: MOV KEY_STATUS0,#0FFH MOV KEY_STATUS1,#0FFH ;POP R2 ;POP R1 RET quet_phim:;quet phim tren cong p0 mov p0,#0feh mov a,p0 cjne a,#0feh,press0;neu a khong bang 1111 1110 thi nhay sjmp continue1 press0: setb se4 setb se5 setb se6 setb se7 setb se8 setb se9 setb se10 setb se11 setb se12 setb se13 setb se14 setb se15 cjne a,#0eeh,key1;a=eeh - &gt;key0 key0: jb se0,k0;se0 dang 1 thi nhay Ljmp exit k0: clr se0; </pre>
---	---

<pre> clr tr1 clr tf1 mov th1,#high(-1000) mov tl1,#low(-1000) setb tr1  reti QUET_LED:     MOV DPTR,#TABLE     MOV P2,#0FFH     CJNE R3,#0FEH,LED_1     MOV P2,R3     MOV A,NUMBER1_1     MOVC A,@A+DPTR     MOV P1,A     SJMP END_LED LED_1:     CJNE R3,#0FDH,LED_2     MOV P2,R3     MOV A,NUMBER1_2     MOVC A,@A+DPTR     MOV P1,A     SJMP END_LED LED_2:    CJNE R3,#0FBH,LED_3     MOV P2,R3     MOV A,NUMBER2_1     MOVC A,@A+DPTR     MOV P1,A     SJMP END_LED LED_3:    CJNE R3,#0F7H,LED_4     MOV P2,R3     MOV A,NUMBER2_2     MOVC A,@A+DPTR     MOV P1,A     SJMP END_LED LED_4:    CJNE R3,#0EFH,LED_5     MOV P2,R3     MOV A,RESULT1     MOVC A,@A+DPTR     MOV P1,A </pre>	<pre> MOV KEY_STATUS0,#0FEH;XO A BIT THU 0 XUONG 0 MOV KEY_STATUS1,#0FFH Ljmp exit key1: setb se0;xoa key0 cjne a,#0deh,key2;a==deh -&gt;key1 jb se1,k1;se1 dang o 0 thi THOAT (EXIT) Ljmp exit k1: clr se1 MOV KEY_STATUS0,#0FDH MOV KEY_STATUS1,#0FFH Ljmp exit key2: setb se1;xoat key1 cjne a,#0beh,key3; jb se2,k2 Ljmp exit k2: clr se2 MOV KEY_STATUS0,#0FBH MOV KEY_STATUS1,#0FFH Ljmp exit key3: setb se2 jb se3,k3 Ljmp exit k3: clr se3 MOV KEY_STATUS0,#0F7H </pre>
--	---

<pre> SJMP END_LED LED_5:    CJNE R3,#0DFH,LED_6           MOV P2,R3           MOV A,RESULT2           MOVC A,@A+DPTR           MOV P1,A           SJMP END_LED LED_6:    CJNE R3,#0BFH,LED_7           MOV P2,R3           MOV A,RESULT3           MOVC A,@A+DPTR           MOV P1,A           SJMP END_LED LED_7:           MOV P2,R3           MOV A,#0H           MOVC A,@A+DPTR           MOV P1,A END_LED: MOV A,R3           RL A           MOV R3,A RET PROCESS:           ;PUSH R1           ;PUSH R2           MOV R1,KEY_STATUS0           MOV R2,KEY_STATUS1           CJNE R1,#0FFH,K_0_7           CJNE R2,#0FFH,K_8_15_           LJMP END_PROCESS K_8_15_: Ljmp K_8_15 K_0_7:   CJNE R1,#0FEH,K_1_7           JB CONTROL,NUM2_0_7 ;CONTROL =1 -&gt; NHAY THUC HIEN CHO SO THU 2           MOV NUMBER1_2,NUMBER1_1           MOV NUMBER1_1,#0H           SJMP END_0_7 </pre>	<pre> MOV KEY_STATUS1,#0FFH Ljmp exit continue1:           mov p0,#0fdh           mov a,p0           cjne a,#0fdh,press1 Ljmp continue2 press1:           setb se0           setb se1           setb se2           setb se3           setb se8           setb se9           setb se10           setb se11           setb se12           setb se13           setb se14           setb se15           cjne a,#0edh,key5;a=edh -&gt;key4 key4: jb se4,k4 Ljmp exit k4:   clr se4           MOV KEY_STATUS0,#0EFH           MOV KEY_STATUS1,#0FFH Ljmp exit key5: setb se4           cjne a,#0ddh,key6;a==deh-&gt;key1           jb se5,k5 Ljmp exit </pre>
---	---

```

NUM2_0_7: MOV
NUMBER2_2,NUMBER2_1
    MOV NUMBER2_1,#0H
END_0_7: LJMP END_PROCESS
K_1_7: CJNE R1,#0FDH,K_2_7
        JB CONTROL,NUM2_1_7
;CONTROL =1 -> NHAY THUC HIEN
CHO SO THU 2
    MOV
NUMBER1_2,NUMBER1_1
    MOV NUMBER1_1,#1H
    SJMP END_1_7
NUM2_1_7: MOV
NUMBER2_2,NUMBER2_1
    MOV NUMBER2_1,#1H
END_1_7: LJMP END_PROCESS
K_2_7: CJNE R1,#0FBH,K_3_7
        JB CONTROL,NUM2_2_7
;CONTROL =1 -> NHAY THUC HIEN
CHO SO THU 2
    MOV
NUMBER1_2,NUMBER1_1
    MOV NUMBER1_1,#2H
    SJMP END_2_7
NUM2_2_7: MOV
NUMBER2_2,NUMBER2_1
    MOV NUMBER2_1,#2H
END_2_7: LJMP END_PROCESS
K_3_7: CJNE R1,#0F7H,K_4_7
        JB CONTROL,NUM2_3_7
;CONTROL =1 -> NHAY THUC HIEN
CHO SO THU 2
    MOV
NUMBER1_2,NUMBER1_1
    MOV NUMBER1_1,#3H
    SJMP END_3_7
NUM2_3_7: MOV
NUMBER2_2,NUMBER2_1
    MOV NUMBER2_1,#3H

```

```

k5: clr se5
    MOV
KEY_STATUS0,#0DFH
    MOV
KEY_STATUS1,#0FFH
    ljmp exit
key6: setb se5
    cjne
a,#0bdh,key7;
    jb se6,k6
    ljmp exit
k6: clr se6
    MOV
KEY_STATUS0,#0BFH
    MOV
KEY_STATUS1,#0FFH
    ljmp exit
key7: setb se6
    jb se7,k7
    ljmp exit
k7: clr se7
    MOV
KEY_STATUS0,#07FH
    MOV
KEY_STATUS1,#0FFH
    ljmp exit
continue2:
    mov p0,#0fbh
    mov a,p0
    cjne
a,#0fbh,press2
    sjmp continue3
press2:
    setb se0
    setb se1
    setb se2
    setb se3
    setb se4
    setb se5

```

<pre> END_3_7: LJMP END_PROCESS K_4_7: CJNE R1,#0EFH,K_5_7         JB CONTROL,NUM2_4_7 ;CONTROL =1 -&gt; NHAY THUC HIEN CHO SO THU 2         MOV NUMBER1_2,NUMBER1_1         MOV NUMBER1_1,#4H         SJMP END_4_7 NUM2_4_7:         MOV NUMBER2_2,NUMBER2_1         MOV NUMBER2_1,#4H END_4_7: LJMP END_PROCESS K_5_7: CJNE R1,#0DFH,K_6_7         JB CONTROL,NUM2_5_7 ;CONTROL =1 -&gt; NHAY THUC HIEN CHO SO THU 2         MOV NUMBER1_2,NUMBER1_1         MOV NUMBER1_1,#5H         SJMP END_5_7 NUM2_5_7:         MOV NUMBER2_2,NUMBER2_1         MOV NUMBER2_1,#5H END_5_7: LJMP END_PROCESS K_6_7: CJNE R1,#0BFH,K_7_7         JB CONTROL,NUM2_6_7 ;CONTROL =1 -&gt; NHAY THUC HIEN CHO SO THU 2         MOV NUMBER1_2,NUMBER1_1         MOV NUMBER1_1,#6H         SJMP END_6_7 NUM2_6_7: MOV NUMBER2_2,NUMBER2_1         MOV NUMBER2_1,#6H END_6_7: LJMP END_PROCESS </pre>	<pre> setb se6 setb se7 setb se12 setb se13 setb se14 setb se15 cjne a,#0ebh,key9;a=ebh -&gt;key8 jb se8,k8 Ljmp exit k8: clr se8 MOV KEY_STATUS0,#0FFH MOV KEY_STATUS1,#0FEH Ljmp exit key9: setb se8 cjne a,#0dbh,key10;a==dbh -&gt;key9 jb se9,k9 Ljmp exit k9: clr se9 MOV KEY_STATUS0,#0FFH MOV KEY_STATUS1,#0FDH Ljmp exit key10: setb se9 cjne a,#0bbh,key11; jb se10,k10 Ljmp exit k10: clr se10 MOV KEY_STATUS0,#0FFH MOV KEY_STATUS1,#0FBH </pre>
--	---

<p>K_7_7:</p> <p>    JB CONTROL,NUM2_7_7</p> <p>;CONTROL =1 -&gt; NHAY THUC HIEN</p> <p>CHO SO THU 2</p> <p>        MOV</p> <p>NUMBER1_2,NUMBER1_1</p> <p>        MOV NUMBER1_1,#7H</p> <p>        SJMP END_7_7</p> <p>NUM2_7_7: MOV</p> <p>NUMBER2_2,NUMBER2_1</p> <p>        MOV NUMBER2_1,#7H</p> <p>END_7_7: LJMP END_PROCESS</p> <p>K_8_15:</p> <p>    CJNE R2,#0FEH,K_9_15</p> <p>    JB CONTROL,NUM2_8_7</p> <p>;CONTROL =1 -&gt; NHAY THUC HIEN</p> <p>CHO SO THU 2</p> <p>        MOV</p> <p>NUMBER1_2,NUMBER1_1</p> <p>        MOV NUMBER1_1,#8H</p> <p>        SJMP END_8_7</p> <p>NUM2_8_7: MOV</p> <p>NUMBER2_2,NUMBER2_1</p> <p>        MOV NUMBER2_1,#8H</p> <p>END_8_7: LJMP END_PROCESS</p> <p>K_9_15:</p> <p>    CJNE</p> <p>R2,#0FDH,K_10_15</p> <p>    JB CONTROL,NUM2_9_7</p> <p>;CONTROL =1 -&gt; NHAY THUC HIEN</p> <p>CHO SO THU 2</p> <p>        MOV</p> <p>NUMBER1_2,NUMBER1_1</p> <p>        MOV NUMBER1_1,#9H</p> <p>        SJMP END_9_7</p> <p>NUM2_9_7: MOV</p> <p>NUMBER2_2,NUMBER2_1</p> <p>        MOV NUMBER2_1,#9H</p> <p>END_9_7: LJMP END_PROCESS</p>	<p>Ljmp exit</p> <p>key11: setb se10</p> <p>        jb se10,k11</p> <p>        Ljmp exit</p> <p>k11: CLR SE11</p> <p>        MOV</p> <p>KEY_STATUS0,#0FFH</p> <p>        MOV</p> <p>KEY_STATUS1,#0F7H</p> <p>        Ljmp exit</p> <p>continue3:</p> <p>        mov p0,#0f7h</p> <p>        mov a,p0</p> <p>        cjne</p> <p>a,#0f7h,press3</p> <p>        ;</p> <p>        setb se0</p> <p>        setb se1</p> <p>        setb se2</p> <p>        setb se3</p> <p>        setb se4</p> <p>        setb se5</p> <p>        setb se6</p> <p>        setb se7</p> <p>        setb se8</p> <p>        setb se9</p> <p>        setb se10</p> <p>        setb se11</p> <p>        setb se12</p> <p>        setb se13</p> <p>        setb se14</p> <p>        setb se15</p> <p>        ;</p> <p>        sjmp exit</p> <p>press3: setb se0</p> <p>        setb se1</p> <p>        setb se2</p> <p>        setb se3</p> <p>        setb se4</p>
---	---

<pre> K_10_15: CJNE R2,#0FBH,K_11_15         CPL CONTROL;CHANGE         LJMP END_PROCESS K_11_15: CJNE R2,#0F7H,K_12_15         MOV NUMBER1_1,#0H;RESET         MOV NUMBER1_2,#0H         MOV NUMBER2_1,#0H         MOV NUMBER2_2,#0H         LJMP END_PROCESS K_12_15: CJNE R2,#0EFH,K_13_15         MOV A,NUMBER1_1;CONG         ADD A,NUMBER2_1         MOV B,#10         DIV AB         MOV RESULT1,0F0H; MOV RESULT1,B         ADD A,NUMBER1_2         ADD A,NUMBER2_2         MOV B,#10         DIV AB         MOV RESULT2,0F0H         MOV RESULT3,A         LJMP END_PROCESS </pre>	<pre> setb se5 setb se6 setb se7 setb se8 setb se9 setb se10 setb se11 setb se12 cjne a,#0e7h,key13;a=e7h -&gt;key12 jb se12,k12 ljmp exit k12: clr se12         MOV KEY_STATUS0,#0FFH         MOV KEY_STATUS1,#0EFH         sjmp exit key13: setb se12         cjne a,#0d7h,key14;a==d7h -&gt;key13 jb se13,k13 ljmp exit k13: clr se13         MOV KEY_STATUS0,#0FFH         MOV KEY_STATUS1,#0DFH         sjmp exit key14: setb se13         cjne a,#0b7h,key15;         jb se14,k14         ljmp exit k14: clr se14         MOV KEY_STATUS0,#0FFH </pre>
---	--

```

    MOV
    KEY_STATUS1, #0BFH
    sjmp exit
key15:   setb se14
          jb se15,k15
          ljmp exit
k15:   clr se15
        MOV
        KEY_STATUS0, #0FFH
        MOV
        KEY_STATUS1, #07FH
exit:
ret
table: db 0C0h;0
        db 0F9h;1
        db 0A4h;2
        db 0B0h;3
        db 099h;4
        db 092h;5
        db 082h;6
        db 0F8h;7
        db 080h;8
        db 090h;9
        db 088h;a
        db 083h;b
        db 0C6h;c
        db 0A1h;d
        db 086h;e
        db 08Eh;f
end

```

### **Yêu cầu thực hành:**

- Khảo sát chức năng của hệ thống sau khi biên dịch và nạp chương trình vào bộ nhớ của vi điều khiển.
- Sự khác nhau nào khi sử dụng các lệnh rẽ nhánh như LJMP và SJMP.
- Hãy tối ưu chương trình trên sao cho thời gian thực thi chương trình giảm.
- Đánh giá phương pháp trên khi mở rộng yêu cầu phải thực hiện các phép toán cơ bản như: cộng, trừ, nhân, chia số có 3 chữ số.

## **PHẦN IV: LẬP TRÌNH GIAO TIẾP 8051 VÀ NGOẠI VI SỬ DỤNG C**

Mục đích chung của phần này nhằm giúp người học vận dụng ngôn ngữ lập trình cấp cao C để lập trình cho vi điều khiển. Các ví dụ nhằm mục đích để người học có thể tiếp cận được cấu trúc phần cứng bên trong của dạng máy tính này. Phần này sẽ đề cập nhiều hơn về việc đa dạng hóa các thiết bị ngoại vi có thể kết nối với máy tính trên chip họ 8051 cũng như các vấn đề minh họa việc giao tiếp mở rộng bộ nhớ. Ngoài ra, việc truyền thông trao đổi dữ liệu giữa máy tính đa dụng và máy tính trên chip họ 8051 cũng được trình bày trong phần này.

Từ các ví dụ cơ bản được cung cấp trong phần này, người học có thể hiểu sâu cách sử dụng của từng mã lệnh, cấu trúc và sự ảnh hưởng của các thanh ghi, cũng như phương pháp lập trình điều khiển khi thực hiện các tác vụ từ máy tính trên chip 8051.

## Sơ lược ngôn ngữ lập trình C dành cho vi điều khiển

Chức năng của trình biên dịch là nhằm để chuyển mã nguồn cấp cao C thành mã máy HEX cái mà phần cứng máy tính có thể hiểu và thực hiện lệnh như mong muốn thể hiện của người lập trình. Mã máy có định dạng HEX sẽ được tải xuống ROM của vi điều khiển. Kích thước của mã máy là rất quan trọng bởi nó bị giới hạn về tài nguyên phần cứng của vi điều khiển. Thông thường thì không gian bộ nhớ để lưu trữ mã khoảng 64K bytes.

Một số lý do để sử dụng C trong việc lập trình vi điều khiển đó là:

- Việc lập trình sử dụng mã cấp cao C sẽ tiết kiệm thời gian hơn so với lập trình bằng ngôn ngữ cấp thấp như ASM. Tuy nhiên, sử dụng ngôn ngữ C sẽ dẫn đến mã HEX lớn hơn.
- Có thể sử dụng khái niệm hàm và các thư viện hàm.
- Mã nguồn C có tính di động cao đối với nhiều kiến trúc vi điều khiển khác nhau. Như vậy, cùng một mã nguồn C có thể chạy trên một số vi điều khiển khác nhau mà không cần phải chỉnh sửa mã nguồn ban đầu. Việc hiểu rõ kiểu dữ liệu trong ngôn ngữ C sẽ giúp người lập trình có thể tối ưu mã nguồn và kích thước file HEX. Một số kiểu dữ liệu như là:

Ký tự không dấu	Unsigned char
Ký tự có dấu	Signed char
Số nguyên không dấu	Unsigned int
số nguyên có dấu	Signed int
kiểu bit đơn	Sbit (single bit)
Kiểu bit và thanh ghi có chức năng đặc biệt	Bit and SFR

## Kiểu ký tự không dấu (UNSIGNED CHAR)

Vì vi điều khiển họ 8051 được cấu tạo từ một bộ xử lý 8-bit, việc sử dụng một kiểu char không dấu là cần thiết và phù hợp. Với kiểu dữ liệu này, tầm giá trị trong khoảng 0-255 (tức 00-FF). Với kiểu dữ liệu này, có thể được sử dụng nhằm lưu các giá trị bộ đếm và các ký tự trong bảng mã ASCII. Trong trình biên dịch C, nó sẽ mặc định sử dụng kiểu dữ liệu có dấu signed nếu chúng ta không khai báo từ khóa unsigned.

### Ví dụ minh họa

Sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C để viết một chương trình xuất 00-FF tại cổng P1 của vi điều khiển.

```
#include <reg51.h>
void main(void)
{
    unsigned char z;
    for (z=0;z<=255;z++)
        P1=z;
}
```

Trong ví dụ này, chúng ta nên lưu ý vì port P1 của vi điều khiển là 8-bit vì thế hãy sử dụng kiểu dữ liệu unsigned char thay vì kiểu int.

Sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để gửi các giá trị HEX của các ký tự ASCII như 0, 1, 2, 3, 4, 5, A, B, C,... ra cổng P1.

```
#include <reg51.h>
void main(void)
{unsigned char
mynum[ ]="012345ABCD";
unsigned char z;
for (z=0;z<=10;z++)
P1=mynum[z];
}
```

Lưu ý các ký tự ASCII đều có thể hiện chỉ với 8-bit.

Sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle các bit trên cổng P1 liên tục.

```
//Toggle P1 forever
#include <reg51.h>
void main(void)
{
for (;;)
{
p1=0x55;
p1=0xAA;
}
}
```

Giải thích chương trình này, tại sao lại đặt giá trị 0x55 và 0xAA vào P1 để thực hiện toggle.

### Kiểu ký tự có dấu (SIGNED CHAR)

Cũng như kiểu ký tự không dấu, kiểu ký tự có dấu sử dụng 8-bit để lưu giá trị. Tuy nhiên, bit có trọng số lớn MSB được sử dụng để lưu miền giá trị + hoặc -. Như vậy, với kiểu dữ liệu có dấu này, tầm giá trị mà nó có thể lưu được là -128 đến 127. Chỉ sử dụng kiểu dữ liệu có dấu nếu muốn thể hiện giá trị số <0. Ví dụ như giá trị NHIỆT ĐỘ.

### Ví dụ minh họa

Sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để gửi giá trị số -4 đến +4 ra cổng P1.

```
//Signed numbers
#include <reg51.h>
void main(void)
{
char mynum[]={+1, -1, +2, -
2, +3, -3, +4, -4};
unsigned char z;
for (z=0;z<=8;z++)
P1=mynum[z];
}
```

Quan sát ngõ ra của cổng P1 trên vi điều khiển.

## Kiểu số nguyên không dấu và có dấu (UNSIGNED/SIGNED INT)

Đối với kiểu dữ liệu này, 16-bit được sử dụng để thể hiện dữ liệu. Đối với số nguyên không dấu (unsigned int) tầm giá trị vào khoảng 0 đến 65535 (tức 0000 đến FFFF). Việc sử dụng kiểu dữ liệu này nhằm để:

- Định nghĩa các biến 16-bit như là địa chỉ của bộ nhớ.
- Lưu trữ giá trị của bộ đếm nếu khi đếm tầm giá trị vượt quá 256.
- Đa phần vì thanh ghi và việc truy cập bộ nhớ trong khoảng 8-bit thế nên việc sử dụng các biến không đúng kích thước sẽ dẫn đến việc tạo HEX có kích thước lớn.

Cũng giống như kiểu ký tự không dấu, kiểu số nguyên có dấu sẽ sử dụng MSB để lưu trữ - hoặc +. Như vậy chỉ còn 15 bit để lưu trữ độ lớn của giá trị. Tầm giá trị sẽ từ -32768 to +32767.

### Kiểu bit đơn (SBIT)

Sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle bit D0 của P1 (P1.0) 50.000 lần.

```
#include <reg51.h>
sbit MYBIT=P1^0;
void main(void)
{
    unsigned int z;
    for (z=0;z<=50000;z++)
    {
        MYBIT=0;
        MYBIT=1;
    }
}
```

Từ khóa sbit cho phép truy cập đến các bit đơn của các thanh ghi SFR.

Sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle chỉ 1 bit D4 của P2 các bit khác trên P2 không bị ảnh hưởng.

```
//Toggling an individual
bit
#include <reg51.h>
sbit mybit=P2^4;
void main(void)
{
while (1)
{
mybit=1; //turn on P2.4
mybit=0; //turn off P2.4
}
}
```

Lưu ý:

Từ port P0 đến P3 của vi điều khiển họ 8051 đều là các cổng 8-bit có thể truy xuất theo bit.

Sbit là kiểu dữ liệu để truy cập bit đơn.

Sử dụng định dạng Px<sup>y</sup>, trong đó:

- X là port 0-3.
- Y là 0-7 tương ứng 8-bit.

Sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle chỉ 1 bit D5 của P1 50000 lần.

```
sbit MYBIT=0x95;
void main(void)
{
unsigned int z;
for (z=0;z<50000;z++)
{
MYBIT=1;
MYBIT=0;
}
}
```

Chúng ta có thể truy cập bit đơn của bất kỳ SFR nào nếu chỉ rõ địa chỉ bit.

Không cần thiết sử dụng.

#include <reg51.h>.

Điều này cho phép chúng ta truy cập đến bất cứ byte nào của bộ nhớ truy xuất ngẫu nhiên SFR tại 80-FF.

## TAO TRỄ

Có hai cách để tạo trễ thời gian:

- Sử dụng một vòng lặp đơn giản.
- Sử dụng bộ định thời timer.

Có ba yếu tố ảnh hưởng đến độ chính xác của trễ là:

1. Thiết kế của máy tính 8051
  - Số lượng chu kỳ máy.
  - Số lượng chu kỳ xung nhịp cho một chu kỳ máy.
2. Tần số của thạch anh được kết nối đến chân ngõ vào x1 và x2.
3. Lựa chọn trình biên dịch.

Trình biên dịch sẽ chuyển các câu lệnh C và hàm thành các lệnh hợp ngữ. Như vậy, trình biên dịch khác nhau sẽ tạo nên những mã nguồn khác nhau.

Ví dụ minh họa:

Sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle các bit của P1 liên tục sử dụng khái niệm thời gian trì hoãn.

```
#include <reg51.h>
void main(void)
{
    unsigned int x;
    for (;;) //repeat forever
    {
        p1=0x55;
        for (x=0;x<40000;x++);
        //delay size
        //unknown
        p1=0xAA;
        for (x=0;x<40000;x++);
    }
}
```

Sử dụng máy dao động ký để quan sát thời gian thay đổi trạng thái tại các chân của cổng P1.

Sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle các bit của P1 liên tục sử dụng thời gian trì hoãn là 250m giây.

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
{
    while (1) //repeat forever
    {
        p1=0x55;
        MSDelay(250);
        p1=0xAA;
        MSDelay(250);
    }
}
void MSDelay(unsigned int
itime)
{
    unsigned int i,j;
    for (i=0;i<itime;i++)
        for (j=0;j<1275;j++);
}
```

Sử dụng máy dao động ký để quan sát thời gian thay đổi trạng thái tại các chân của cổng P1 nhằm so sánh với thời gian dự kiến là 250ms.

Sử dụng vi điều khiển họ 8051 và ngôn ngữ lập trình C viết một chương trình để toggle các bit của P0, 1 và 2 liên tục sử dụng thời gian trì hoãn là 250m giây. Sử dụng từ khóa sfr để khai báo địa chỉ cổng.

```
//Accessing Ports as SFRs
using sfr data type
sfr P0=0x80;
sfr P1=0x90;
sfr P2=0xA0;
void MSDelay(unsigned int);
void main(void)
{
```

Một cách khác để truy cập đến bộ nhớ SFR tại 80-FF là sử dụng kiểu dữ liệu sfr.

```
while (1)
{
P0=0x55;
P1=0x55;
P2=0x55;
MSDelay(250);
P0=0xAA;
P1=0xAA;
P2=0xAA;
MSDelay(250);
}
}
```

# BÀI THỰC HÀNH SỐ 1

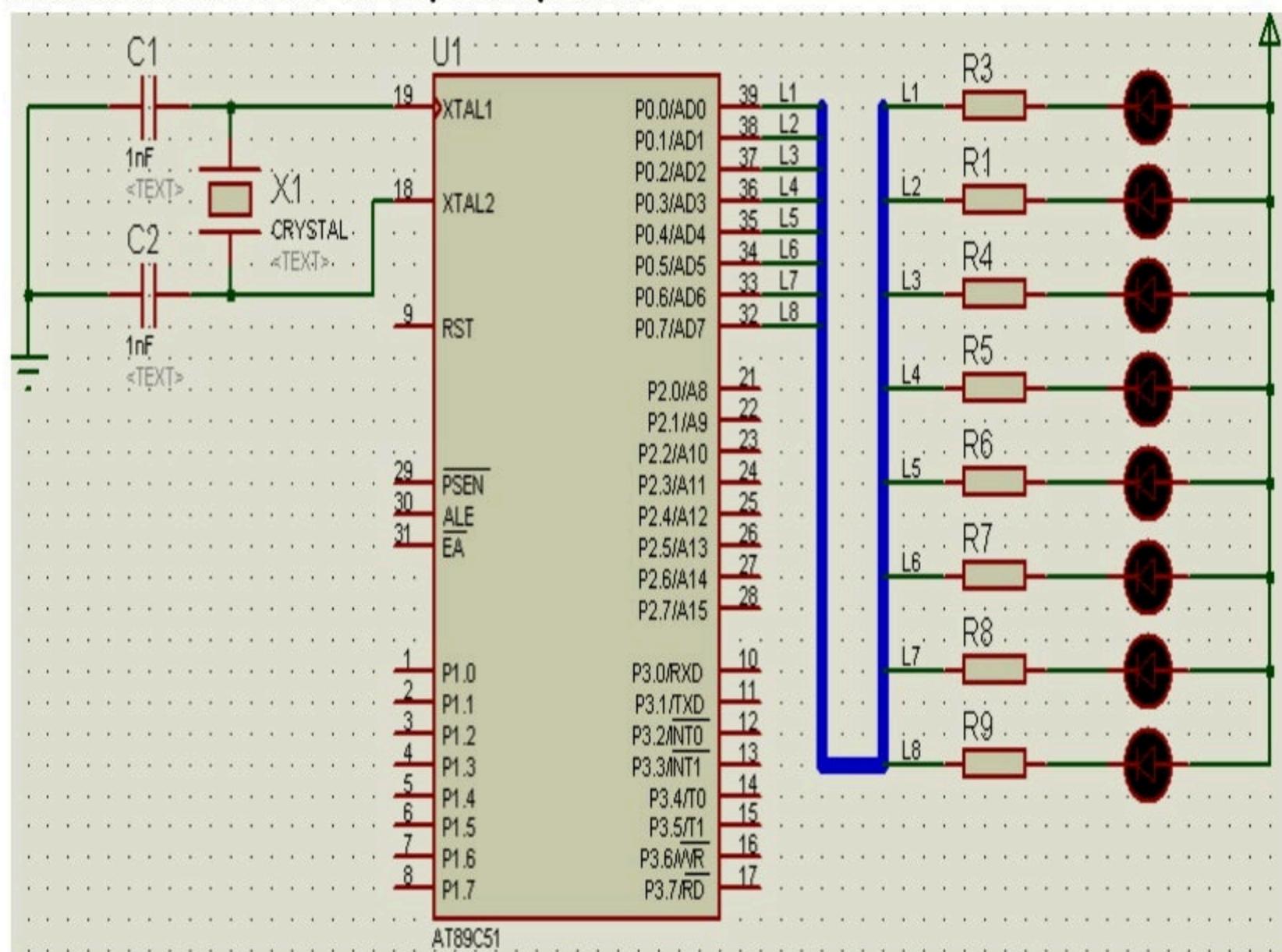
## Mục đích:

- Giới thiệu về giao tiếp vi điều khiển và ngoại vi LED đơn ở các dạng khác nhau.
- Giới thiệu một số hiệu ứng đơn giản có thể thực hiện được với LED đơn.

## Sau khi kết thúc học phần này, sinh viên có thể:

- Hiểu về giao tiếp vi điều khiển và ngoại vi LED đơn.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình cấp cao C dành cho các vi điều khiển (máy tính trên chip) họ 8051 để giao tiếp với LED đơn.

**Vấn đề 1:** Thiết kế sơ đồ nguyên lý giao tiếp LED đơn theo phương pháp tích cực LED mức thấp như hình vẽ sau. Sử dụng phần mềm Proteus ISIS vẽ mạch điện sau:



**Hình 1:** Sơ đồ nguyên lý kết nối VĐK  
và LED đơn tích cực mức thấp

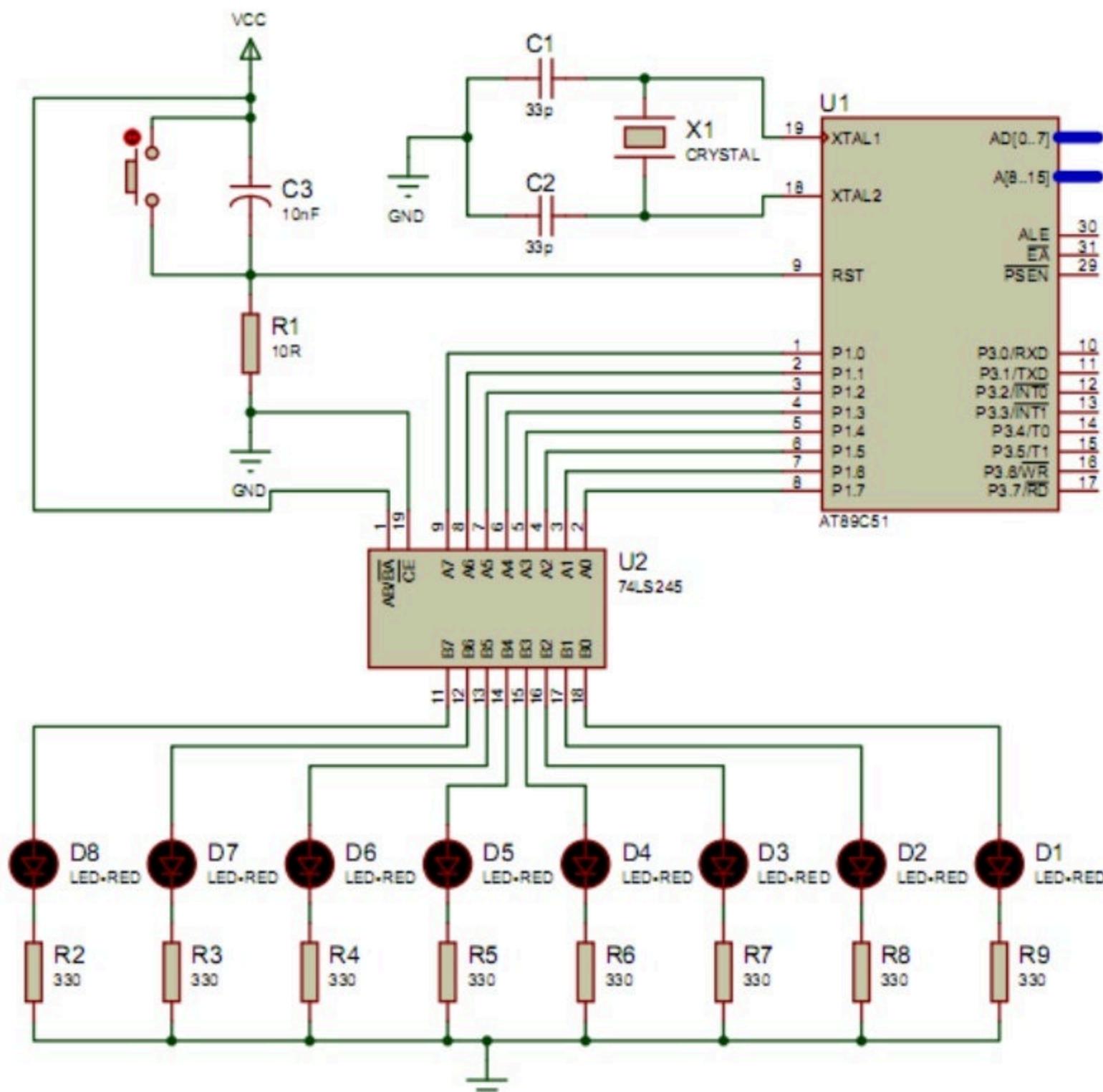
Sử dụng phần mềm Keil C viết chương trình điều khiển 8 LED lân lượt sáng nhấp nháy, so le nhau:

```
#include <at89x51.h>
void delay(int interval){
    int i, j;
    for(i=0;i<255;i++){
        for(j=0;j<interval;j++);
    }
}
void main(){
    while(1){
        P0=0x55;
        delay(100);
        P0=0xAA;
        delay(100);
    }
}
```

### Yêu cầu thực hành:

- Từ lý thuyết mạch điện và điện tử cơ bản, hãy phân tích và lựa chọn giá trị cho điện trở kết nối LED ở hình trên nếu điện áp cung cấp  $V_{DD}$  là 5V.
- Hãy giải thích chức năng của hàm delay trong mã nguồn trễ. Hãy giải thích phương pháp tẠP ĐỘ TRỄ trong hàm delay(). Dùng bộ dao động trong công cụ mô phỏng Proteus để khảo sát thời gian trễ.

**Vấn đề 2:** Thiết kế giao tiếp LED đơn theo phương pháp tích cực LED mức cao như hình vẽ sau. Sử dụng phần mềm Proteus ISIS vẽ mạch điện sau:



**Hình 2:** Sơ đồ nguyên lý kết nối VĐK  
và LED đơn tích cực mức cao

Sử dụng phần mềm Keil C viết chương trình điều khiển 8 LED lần lượt sáng từ trái sang phải.

```
#include <at89x51.h>
#include <stdio.h>

#define LED0 P1_0
#define LED1 P1_1
#define LED2 P1_2
#define LED3 P1_3
#define LED4 P1_4
#define LED5 P1_5
#define LED6 P1_6
#define LED7 P1_7

#define sang 1
#define tat 0
//-----
void deLay(unsigned int ms)
{
    unsigned int i,j;
    for (i=0;i<ms;i++)
        for (j=0;j<120;j++)
    {}
//-----
void display_LED(unsigned char number)
{
    switch (number)
    {
        case 5:
            LED4=sang;
            LED1=LED2=LED3=LED0=LED5=
            LED6=LED7=tat;
            break;
        case 6:
            LED5=sang;
            LED1=LED2=LED3=LED4=LED0=
            LED6=LED7=tat;
            break;
        case 7:
            LED6=sang;
            LED1=LED2=LED3=LED4=LED5=
            LED0=LED7=tat;
            break;
        case 8:
            LED7=sang;
            LED1=LED2=LED3=LED4=LED5=
            LED6=LED0=tat;
            break;
    }
}

main ()
{
    unsigned char m;
    while(1)
    {
        for (m=0;m<9;m++)
        {
            display_LED(m);deLay(500)
        }
    }
}
```

```

case 1:
LED0=sang;
LED1=LED2=LED3=LED4=LED5=
LED6=LED7=tat;
break;
case 2:
LED1=sang;
LED0=LED2=LED3=LED4=LED5=
LED6=LED7=tat;
break;
case 3:
LED2=sang;
LED1=LED0=LED3=LED4=LED5=
LED6=LED7=tat;
break;
case 4:
LED3=sang;
LED1=LED2=LED0=LED4=LED5=
LED6=LED7=tat;
break;

```

### **Yêu cầu thực hành:**

1. Hãy so sánh phương pháp kết nối và giao tiếp điều khiển LED ở vấn đề 1 và 2. Hãy nêu ưu và nhược điểm của hai phương pháp này.
2. Viết chương trình tạo hiệu ứng “sáng đuôi” (các LED sáng lần lượt từ LED 1 tới LED 8).
3. Viết chương trình tạo hiệu ứng cho các LED sáng từ hai đầu (từ LED 1 tới LED 4, từ LED 8 về LED 5) sau đó quay đầu (từ LED 4 về LED 1 và LED 5 về LED 8), quá trình lặp đi lặp lại liên tục.

## BÀI THỰC HÀNH SỐ 2

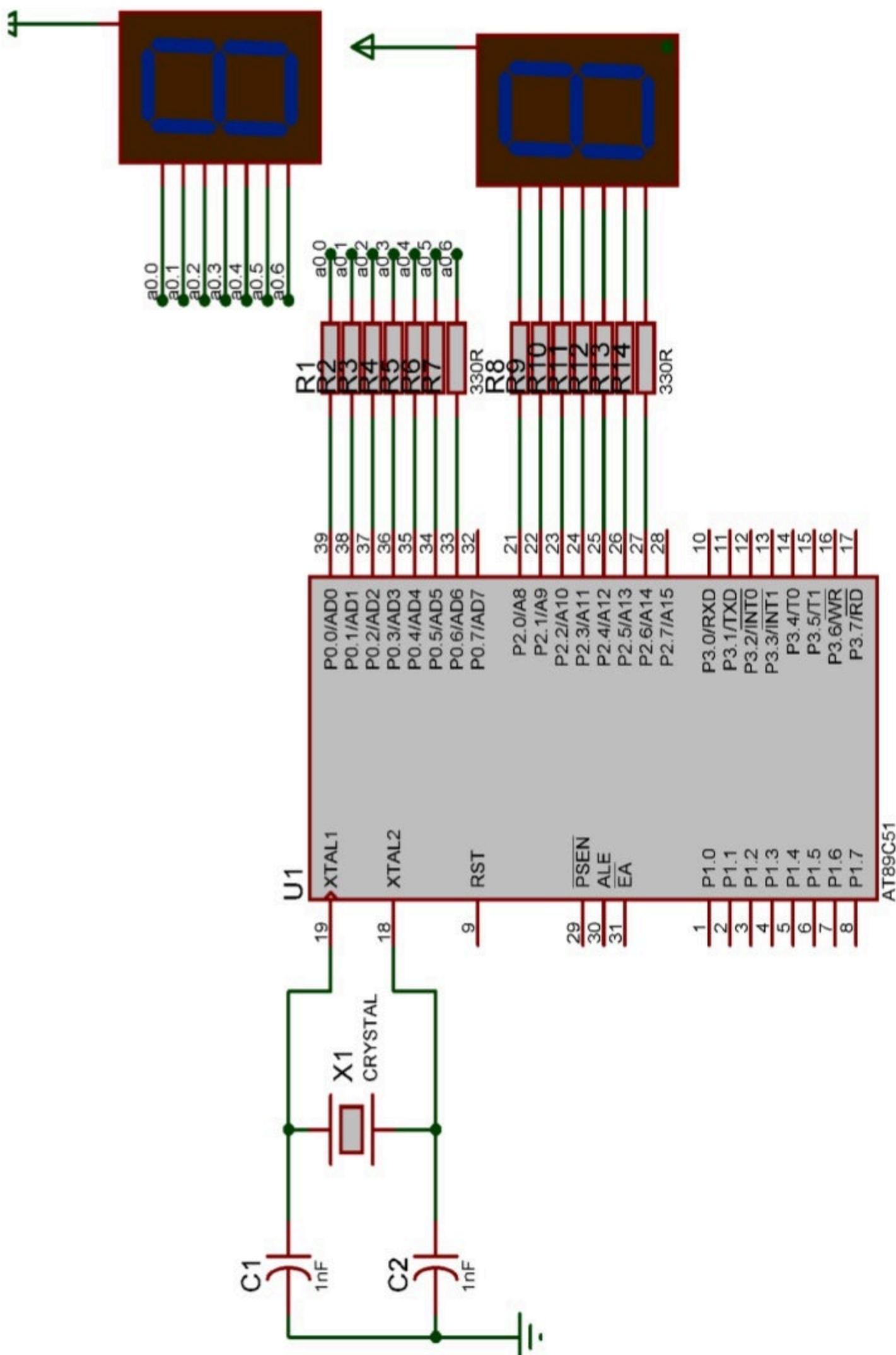
### Mục đích:

- Giới thiệu các phương pháp giao tiếp vi điều khiển và ngoại vi LED 7 đoạn như kết nối trực tiếp, giải mã, quét và một số phương pháp khác.
- Giới thiệu một số hiệu ứng đơn giản có thể thực hiện được với LED đơn.

### Sau khi kết thúc học phần này, sinh viên có thể:

- Hiểu và lựa chọn các phương pháp giao tiếp giữa vi điều khiển và ngoại vi LED 7 đoạn.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình cấp cao C dành cho các vi điều khiển (máy tính trên chip) họ 8051 để giao tiếp với LED 7 đoạn.

**Vấn đề 1:** Phương pháp kết nối trực tiếp, kết nối với LED 7 thanh kiểu Anode chung. Sử dụng phần mềm Proteus ISIS vẽ mạch điện như hình dưới đây.



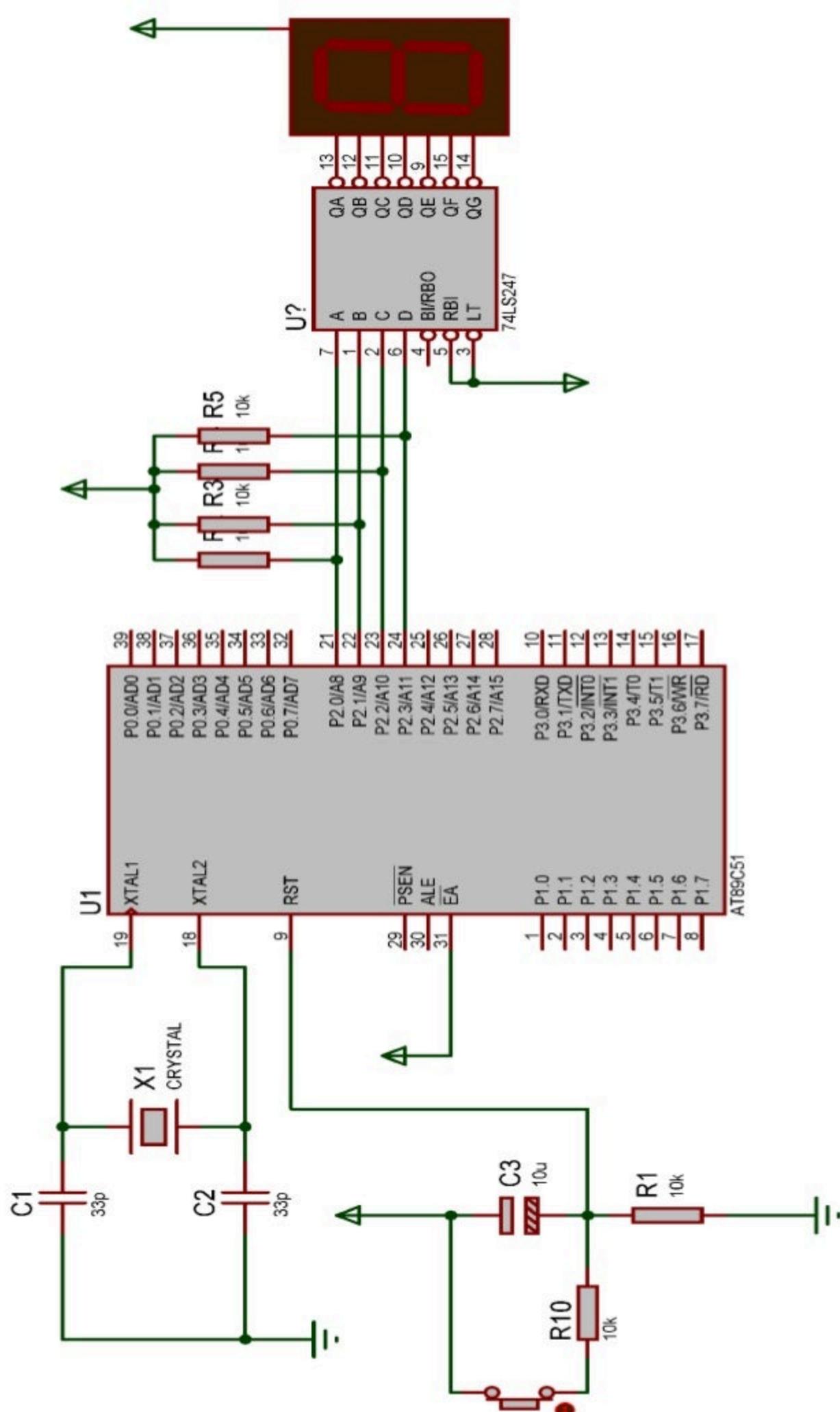
**Hình 3:** Sơ đồ nguyên lý kết nối VĐK và LED 7 đoạn trực tiếp

Sử dụng phần mềm Keil C viết chương trình và đặt tên là **7doan\_tructiep.c** để điều khiển cho LED 7 thanh lần lượt hiển thị các số từ 00 đến 99.

```
#include<at89x51.h>
#define uchar unsigned char
#define uint unsigned int
void deDelay_ms(uint x);
void giaima(void);
void hienthi(void);
uchar donvi, chuc;
uint i, j, a;
int
dig[]={0xC0, 0xF9, 0xA4, 0xB0, 0
x99, 0x92, 0x82, 0xF8, 0x80, 0x90
};
void main (void)
{
while(1)
{
    for(i=0;i<100;i++)
    {
        a=i;
        for(j=0;j<5000;j++)
        {
            giaima();
            hienthi();
        }
    }
}

}
//chuong trinh delay
void deDelay_ms(uint x)
{
    uchar k;
    while(x-->0)
    {
        for(j=0;k<125;k++)
    ){}}
}
//chuong trinh giai ma
void giaima(void)
{
    chuc=a/10;
    donvi=a%10;
}
//chuong trinh hien thi
void hienthi(void)
{
    P0=dig[donvi];
    P2=dig[chuc];
}
```

**Vấn đề 2:** Phương pháp kết nối sử dụng bộ giải mã, kết nối với LED 7 thanh kiểu Anode chung. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây. IC 74LS47 và IC 7446 là các IC giải mã từ mã BCD ra mã 7 thanh.



**Hình 4:** Sơ đồ nguyên lý kết nối VĐK  
và LED 7 đoạn thông qua IC giải mã

Sử dụng phần mềm Keil C viết chương trình và đặt tên là **7doan\_giaima.c** để điều khiển cho LED 7 thanh lần lượt hiển thị các số từ 0 đến 9 trên 1 LED 7 đoạn.

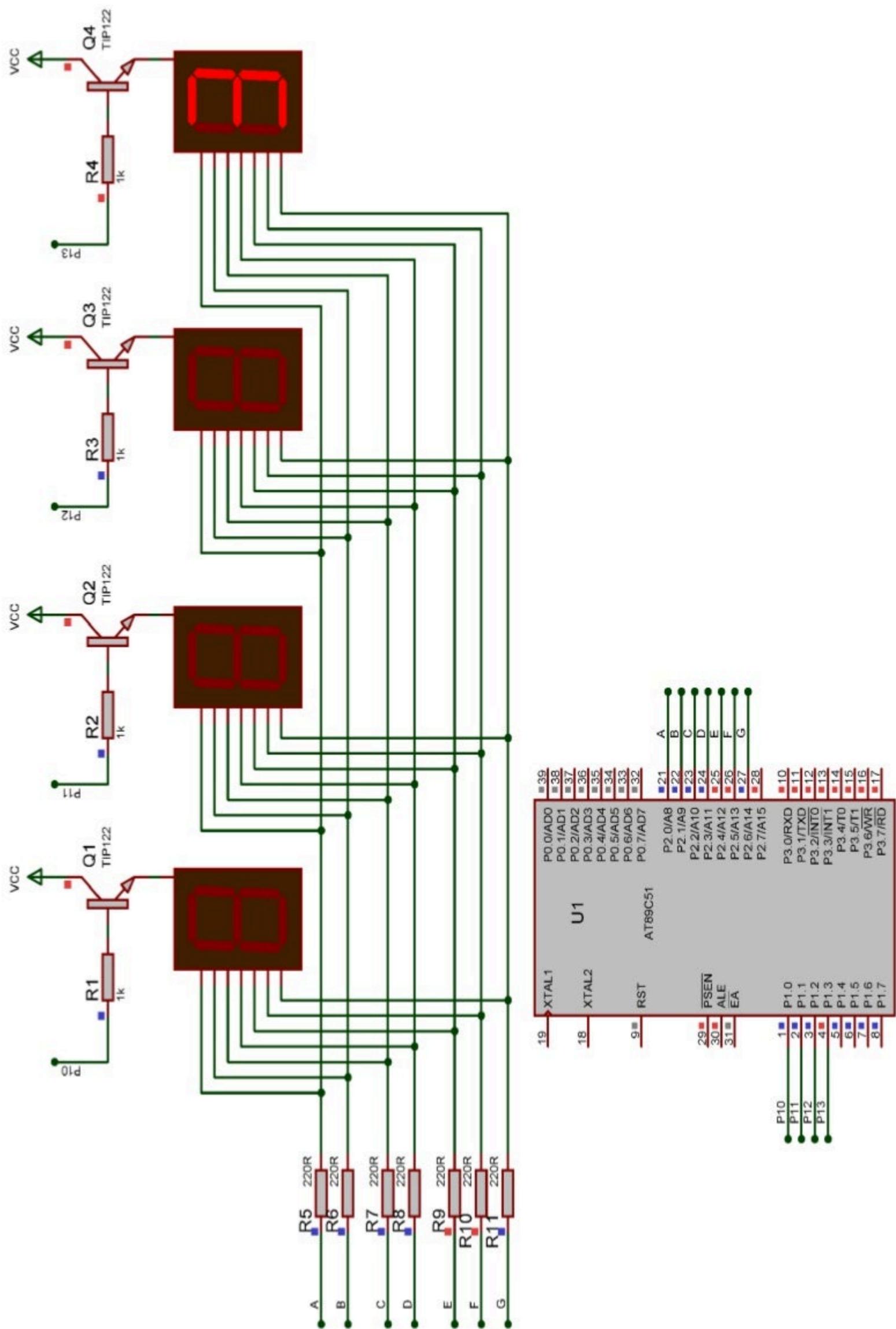
<pre>#include&lt;at89x51.h&gt; int dem=0; void deDelay(unsigned Long time) {     unsigned Long i;     for(i=0;i&lt;time;i++)     {     } } void main(void)</pre>	<pre>{ P2=0x00; while(dem&lt;10) { P2=dem; dem++; deDelay(10000); } }</pre>
--	---

### Yêu cầu thực hành:

- Khảo sát chức năng và đặc tính về điện của 2 IC giải mã 7447 và 7446.
- Hãy cho biết ưu và nhược điểm của phương pháp sử dụng IC giải mã so với phương pháp kết nối trực tiếp.

**Vấn đề 3:** Phương pháp kết nối trực tiếp kết hợp quét cho nhiều LED 7 đoạn. Sử dụng phương pháp quét LED để hiển thị. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây. Trong đó:

- Các chân A, B, C, D, E, F, G, H kết nối với các chân từ P2.0 đến P2.6.
- Các chân quét LED P10, P11, P12, P13 kết nối với các chân từ P1.0 đến P1.3.



**Hình 5:** Sơ đồ nguyên lý kết nối VĐK và LED 7 đoạn  
through qua phương pháp quét sử dụng transistor

Sử dụng phần mềm Keil C viết chương trình và đặt tên là **7doan\_quetLED1.c** để điều khiển cho LED 7 đoạn lần lượt hiển thị các số từ 0 đến F lần lượt trên 4 LED 7 đoạn.

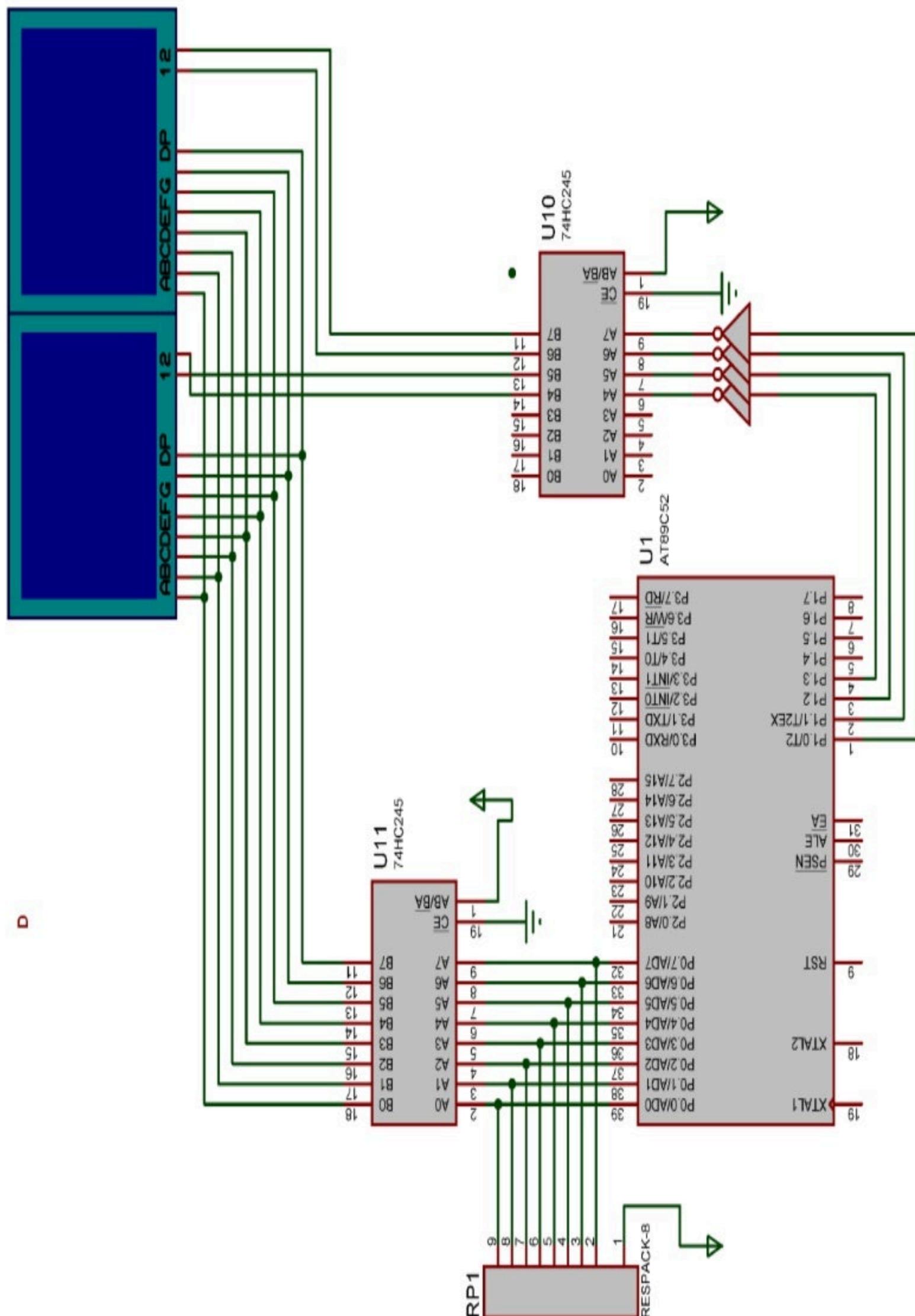
<code>#include &lt;at89x51.h&gt;</code>	<code>P1=8;</code>
<code>#include &lt;stdio.h&gt;</code>	<code>P2=0Xf8;</code>
<code>#define DELAYTIME 65000</code>	<code>temp1=DELAYTIME;</code>
<code>unsigned int temp1;</code>	
<code>void deDelay(unsigned int</code>	<code>deDelay(temp1);</code>
<code>temp)</code>	<code>P1=1;</code>
<code>{</code>	<code>P2=0X80;</code>
<code>    while(--temp);</code>	
<code>}</code>	
<code>void main()</code>	<code>temp1=DELAYTIME;</code>
<code>{</code>	
<code>    P1=0;//LED is off</code>	<code>deDelay(temp1);</code>
<code>    while(1)</code>	<code>P1=2;</code>
<code>        {</code>	<code>P2=0X90;</code>
<code>            P1=1;</code>	
<code>            P2=0Xc0;</code>	<code>temp1=DELAYTIME;</code>
<code>            temp1=DELAYTIME;</code>	<code>deDelay(temp1);</code>
<code>            deDelay(temp1);</code>	<code>P1=4;</code>
<code>            P1=2;</code>	<code>P2=0X08;</code>
<code>            P2=0XF9;</code>	<code>temp1=DELAYTIME;</code>
<code>            temp1=DELAYTIME;</code>	<code>deDelay(temp1);</code>
<code>            deDelay(temp1);</code>	<code>P1=8;</code>
<code>            P1=4;</code>	<code>P2=0X03;</code>
<code>            P2=0XA4;</code>	<code>temp1=DELAYTIME;</code>
<code>            temp1=DELAYTIME;</code>	<code>deDelay(temp1);</code>
<code>            deDelay(temp1);</code>	<code>P1=1;</code>
<code>            P2=0X46;</code>	

$P1=8;$	$temp1=DELAYTIME;$
$P2=0XB0;$	$delay(temp1);$
$temp1=DELAYTIME;$	$P1=2;$
$delay(temp1);$	$P2=0X21;$
$P1=1;$	$temp1=DELAYTIME;$
$P2=0X99;$	$delay(temp1);$
$temp1=DELAYTIME;$	$P1=4;$
$delay(temp1);$	$P2=0X06;$
$P1=2;$	$temp1=DELAYTIME;$
$P2=0X92;$	$delay(temp1);$
$temp1=DELAYTIME;$	$P1=8;$
$delay(temp1);$	$P2=0X0E;$
$P1=4;$	$temp1=DELAYTIME;$
$P2=0X82;$	$delay(temp1);$
$temp1=DELAYTIME;$	$temp1=DELAYTIME;$
$delay(temp1);$	$delay(temp1);$
$temp1=DELAYTIME;$	$delay(temp1);$
	$}$
	$}$

Yêu cầu thực hành:

Thay đổi thông số DELAYTIME ở các giá trị (20000, 40000, 80000) biên dịch lại chương trình và nạp lại cho VĐK. Sau đó quan sát kết quả mô phỏng.

**Vấn đề 4:** Phương pháp kết hợp quét LED và sử dụng các IC đếm 74245. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.



**Hình 6:** Sơ đồ nguyên lý kết nối VĐK và LED 7 đoạn thông qua phương pháp quét sử dụng IC đếm

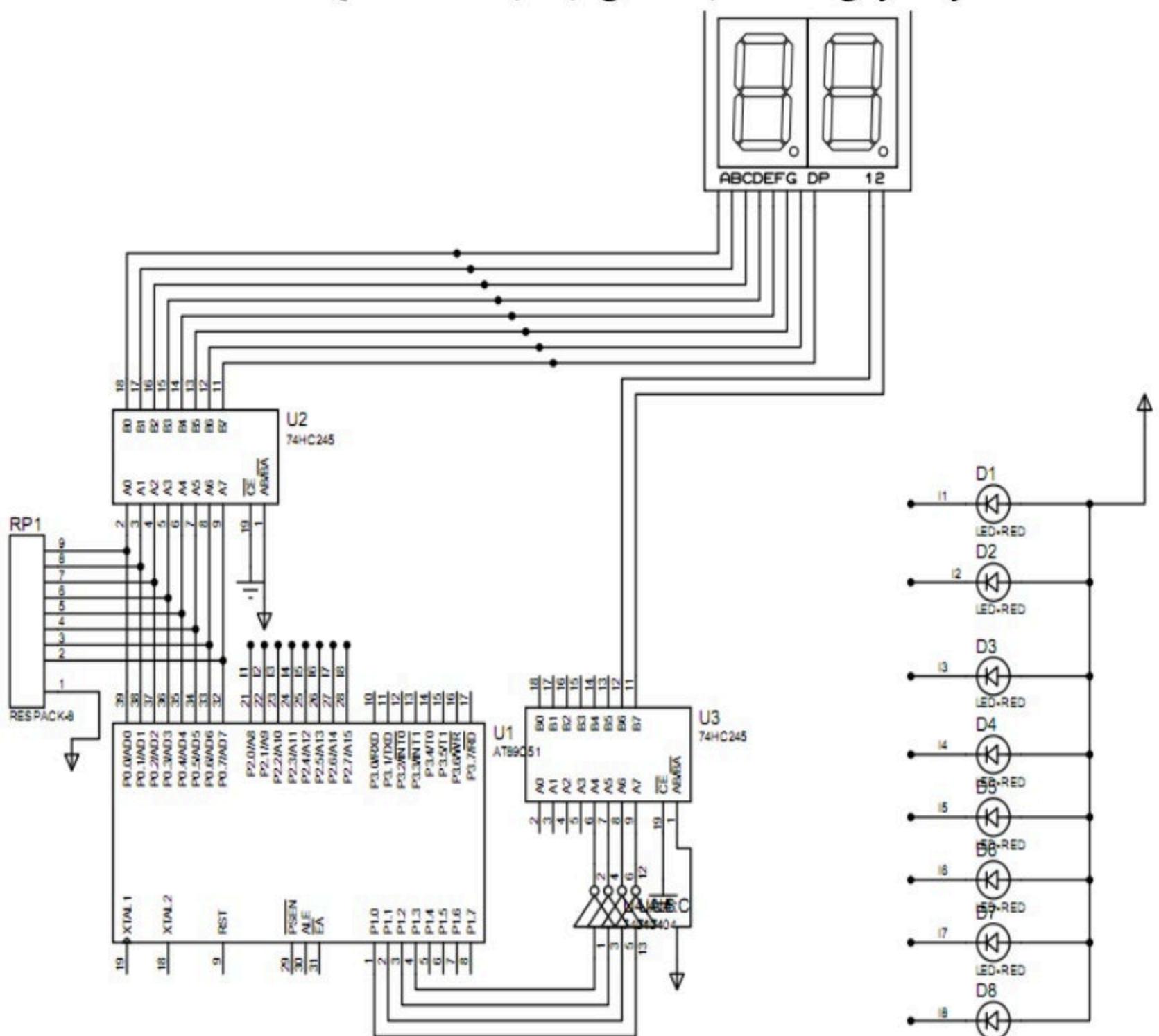
Sử dụng phần mềm Keil C viết chương trình và đặt tên là **7doan\_quet\_dem245.c** để điều khiển cho LED 7 thanh lần lượt hiển thị các số từ 0000 đến 9999 lần lượt trên 4 LED 7 đoạn.

<pre>#include &lt; at89x51.h &gt;  void delay() {     int i;  for(i=1;i&lt;200;i++);  }  unsigned char num[10] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8,0x80,0x90 };  void main( ) {     int x,m,n,j,i=0 ;     unsigned char p0;      while( 1 ) {     i=0;n=0,m=0;     j=0; } </pre>	<pre>for(m=0;m&lt;10;m++)     for (n=0;n&lt;10;n++)         for (i=0;i&lt;10;i++)             for (j = 0 ; j &lt;10 ; j ++)                 for(x=100;x&gt;0;x--) {     P0 = num[j] ;     P1 = 0xfe;     delay();     P1=0xff;     P1 = 0xfd;     P0 = num[i];     delay();     P1=0xff;     P1 = 0xfb;     P0 = num[n];     delay();     P1=0xff;     P1 = 0xf7;     P0 = num[m];     delay();     P1=0xff; } } </pre>
--	---

Yêu cầu thực hành:

- Áp dụng phương pháp kết nối ở trên hãy mở rộng thiết kế nhằm giao tiếp với 8 LED 7 đoạn.
- Hãy so sánh ưu và nhược điểm trong phương pháp vấn đề 3 và 4.

- Chính sửa lại chương trình **vấn đề 2** nhằm xây dựng chương trình con hiển thị LED 7 thanh có khai báo như sau **void Display\_7seg (unsigned char value)**. Hàm này có chức năng nhận đầu vào là các số 0, 1, ..., 9 và điều khiển LED 7 thanh để hiển thị các số đó.
- Với 8 đường ngõ ra từ vi điều khiển, có thể hiển thị được tối đa được bao nhiêu led7 đoạn. Trình bày phương án thiết kế với sơ đồ nguyên lý.
- Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây. Mạch hoạt động với nguyên lý sau: 2 LED 7 đoạn hoạt động với phương pháp quét và hiển thị từ 00-90. Số LED đơn sáng tương ứng với số hàng chục của 7 đoạn hiển thị. Ví dụ 23 thì 2 LED đơn sáng. Khi đếm đến 90 thì quay lại đếm lùi về 00. Mỗi lần giảm số hàng chục đi 1 thì 1 LED đơn tắt. Quá trình hoạt động liên tục theo nguyên lý ở trên.



**Hình 7:** Sơ đồ nguyên lý kết nối VĐK và LED 7 đoạn

# BÀI THỰC HÀNH SỐ 3

## Mục đích:

- Giới thiệu phương pháp giao tiếp ngoại vi ngõ vào nút nhấn, switch với vi điều khiển.
- Giới thiệu phương pháp giao tiếp bàn phím ma trận với vi điều khiển.
- Giới thiệu phương pháp giao tiếp ngoại vi sử dụng ngắt ngoài.
- Giới thiệu phương pháp sử dụng bộ định thời timer để tạo trễ chính xác.

## Sau khi kết thúc học phần này, sinh viên có thể:

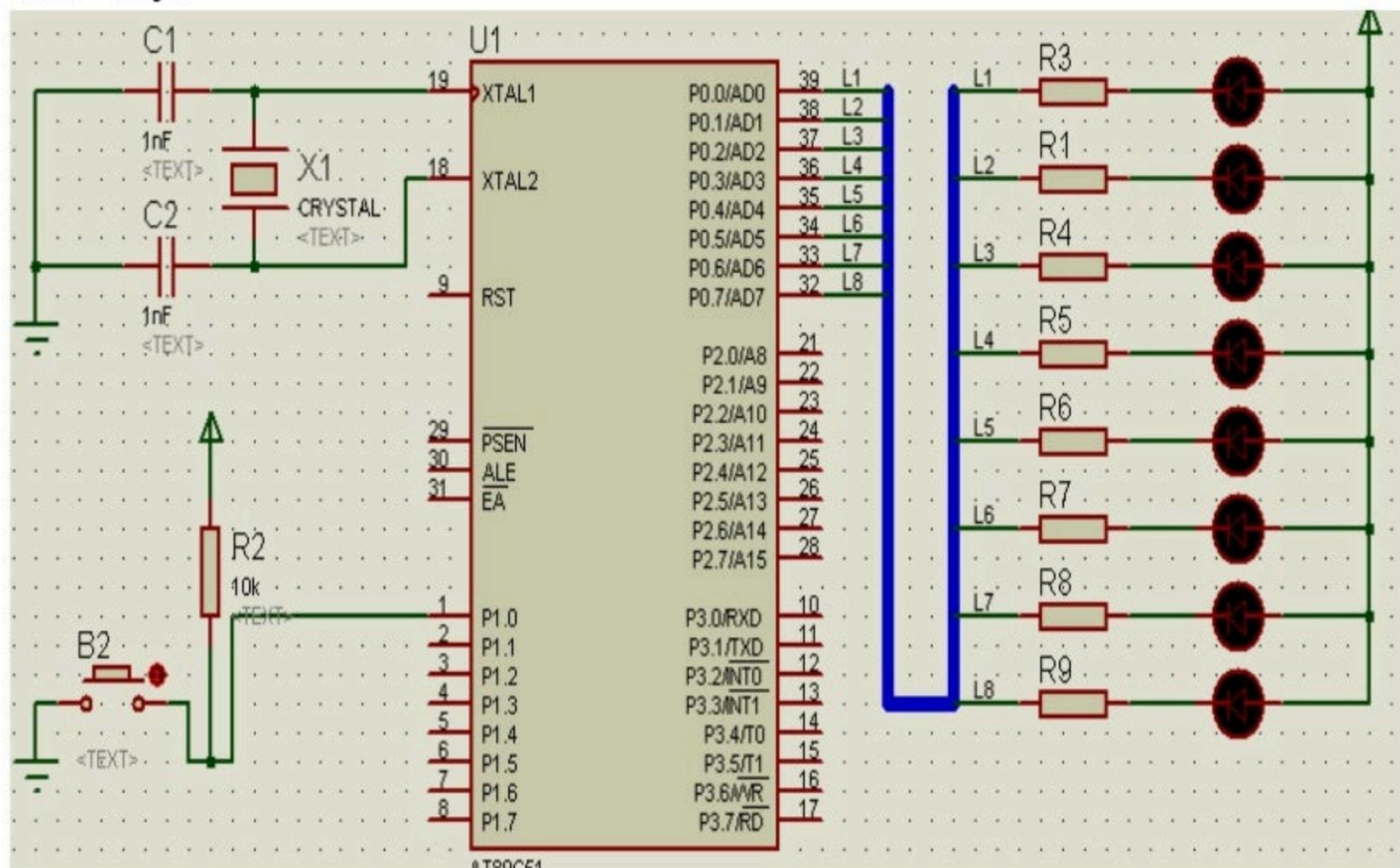
- Hiểu sự điểm khác biệt của phương pháp giao tiếp ngoại vi sử dụng khái niệm NGẮT.
- Hiểu về chu kỳ máy (machine cycle-MC) và sự kết hợp của MC với TIMER để tạo trễ.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình cấp cao C dành cho các vi điều khiển (máy tính trên chip) họ 8051 để vận dụng lập trình NGẮT và TIMER.

## **Yêu cầu thực hành:**

Tham khảo datasheet của vi điều khiển họ 8051 từ đó hãy trả lời các câu hỏi sau đây:

- Giải thích vai trò của bit C/T trong thanh ghi TMOD.
- Làm thế nào vi điều khiển có thể đếm các sự kiện tác động từ bên ngoài.
- Nếu một Timer/Counter 0 được sử dụng như một bộ đếm sự kiện, giá trị đếm tối đa là bao nhiêu cho mỗi chế độ Mode 1, Mode 2.
- Hãy chỉ ra các chân nào được sử dụng cho: (a) timer/counter 0; (b) timer/counter 1.
- Nếu Timer/Counter 0 được sử dụng trong mode 1 nhằm đếm các sự kiện từ bên ngoài, hãy giải thích khi nào bit TF0 được đặt lên mức cao?
- Nếu Timer/Counter 1 được sử dụng trong mode 2 nhằm đếm các sự kiện từ bên ngoài, hãy giải thích khi nào bit TF0 được đặt lên mức cao?

**Vấn đề 1:** Phương pháp kết nối nút nhấn. Khi nhấn nút B2 (ngõ vào P1.0) nhận mức 0. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.



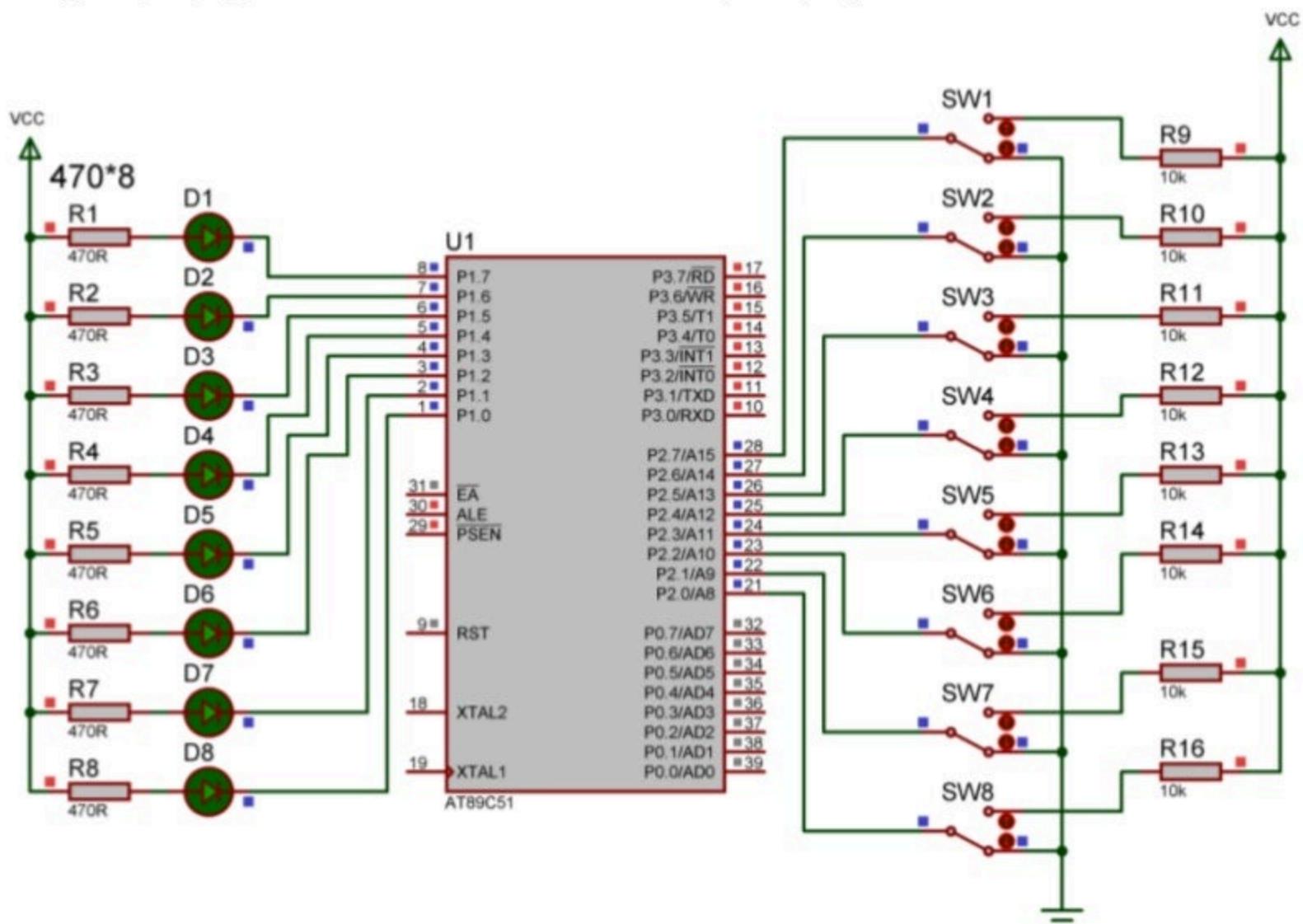
**Hình 8:** Sơ đồ nguyên lý kết nối VĐK và LED đơn và nút nhấn

Sử dụng phần mềm Keil C viết chương trình và đặt tên **nutnhan\_8led.c** để điều khiển 8 led lần lượt sáng nhấp nháy, so le nhau khi người dùng không bấm nút B2. Khi người dùng bấm và giữ nút B2 thì các led giữ nguyên trạng thái.

```
#include <at89x51.h>
void delay(int interval){
    int i,j;
    for(i=0;i<255;i++){
        for(j=0;j<interval;j++);
    }
}

void main(){
    while(1){
        //Kiem tra trang
        thai chan P1_0 (dau
        voi cong tac)
        if(P1_0 == 1){
            P0=0x55;
            delay(100);
            P0=0xAA;
            delay(100);
        }
    }
}
```

**Vấn đề 2:** Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây để nhận trạng thái đầu vào từ 8 công tắc (SW) và ngõ ra là 8 đèn LED đơn. Trong đó, trạng thái của 8 đèn LED thể hiện trạng thái của 8 SW.



**Hình 9:** Sơ đồ nguyên lý kết nối VĐK và LED đơn và công tắc

Sử dụng phần mềm Keil C viết chương trình và đặt tên là **8sw\_8led** để điều khiển cho led lân lượt sáng nếu có SW tương ứng tác động.

```
#include<at89x51.h>
void main()
{
    unsigned char temp;
    P1=0XFF;
    P2=0XFF;
    while(1)
    {
        temp=P2;
        P1=temp;
        while(P2==temp);
```

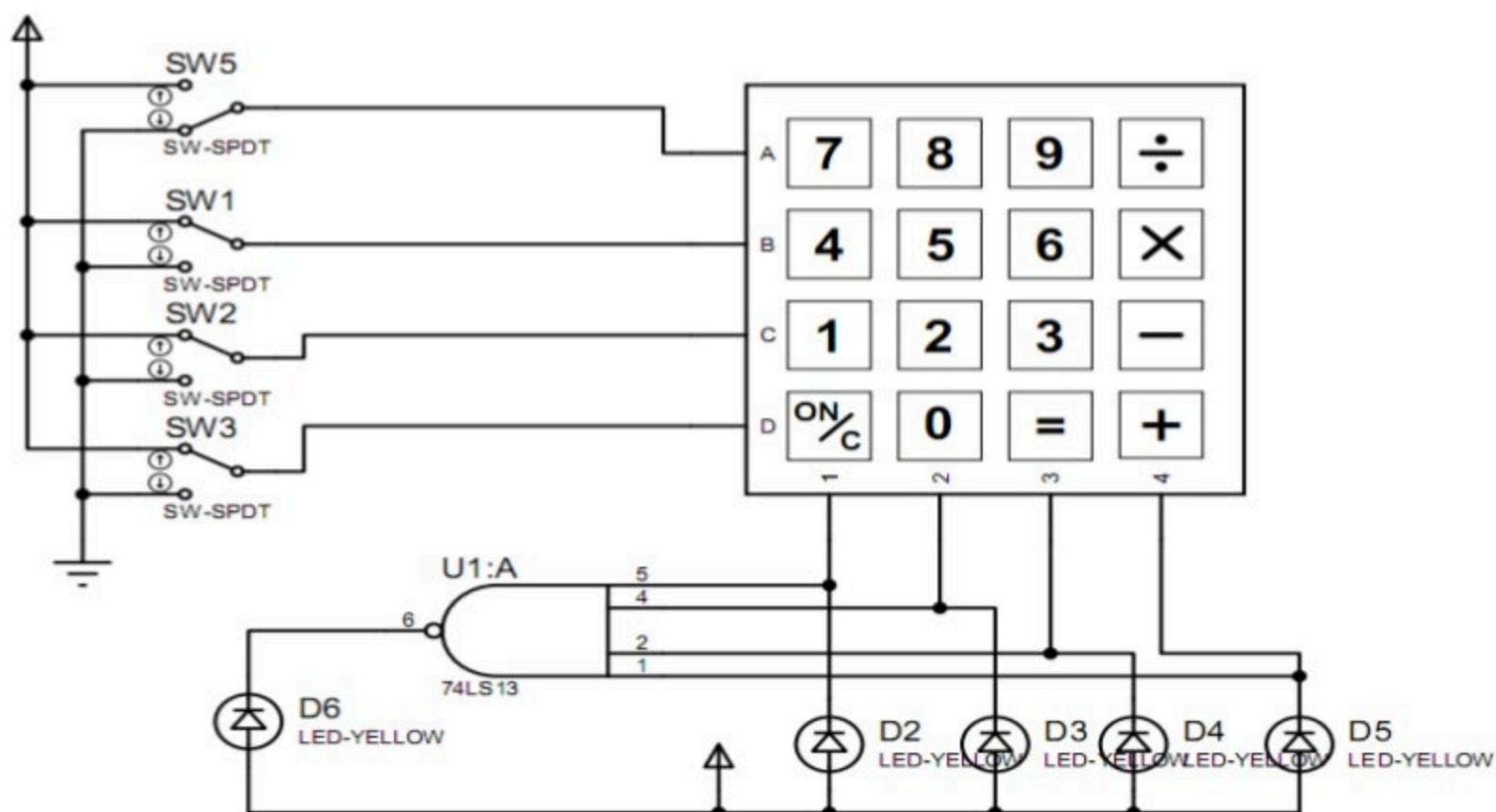
**Vấn đề 3:** Kiểm tra cơ chế hoạt động của bàn phím.

Bước 1. Thiết kế mạch theo sơ đồ dưới đây.

Bước 2. Thủ chuyển tất cả các công tắc gạt SW1 → SW4 sang nối đất sau đó nhấn các phím, quan sát kết quả và ghi nhận lại.

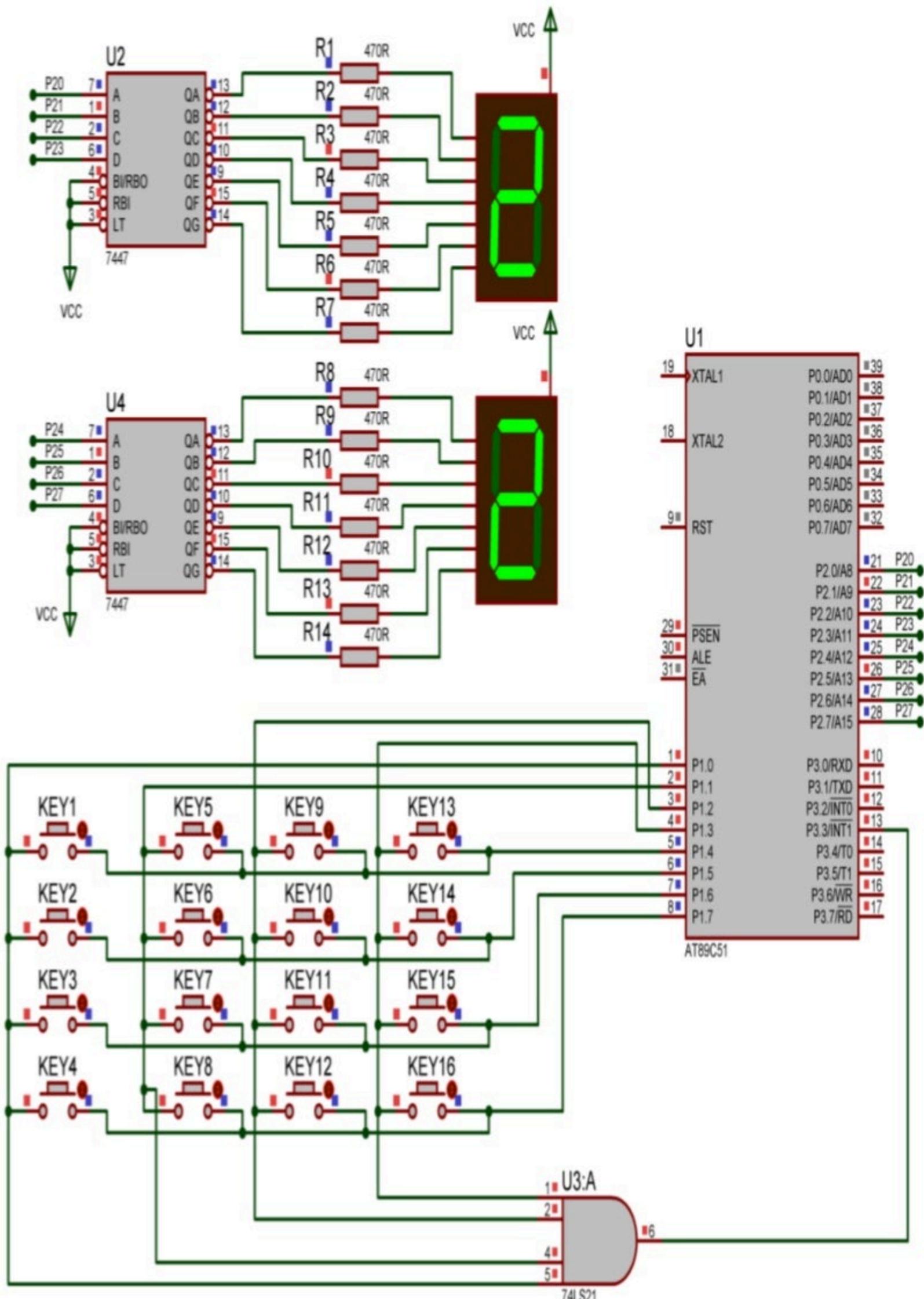
Bước 3. Thủ chuyển tất cả các công tắc gạt sang nối nguồn VCC sau đó nhấn các phím, quan sát kết quả và ghi nhận lại.

Bước 4. Mỗi thời điểm chỉ cho một công tắc gạt nối đất rồi nhấn các phím và quan sát kết quả và ghi nhận lại.



**Hình 10:** Khảo sát bàn phím ma trận

Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây nhằm khảo sát giao tiếp bàn phím 4x4 với vi điều khiển.



**Hình 11:** Sơ đồ nguyên lý kết nối bàn phím với vi điều khiển

Sử dụng phần mềm Keil C viết chương trình và đặt tên là **matranphim\_7doan.c** để nhận vị trí của phím ngõ vào và xuất báo vị trí ra 2 LED 7 đoạn.

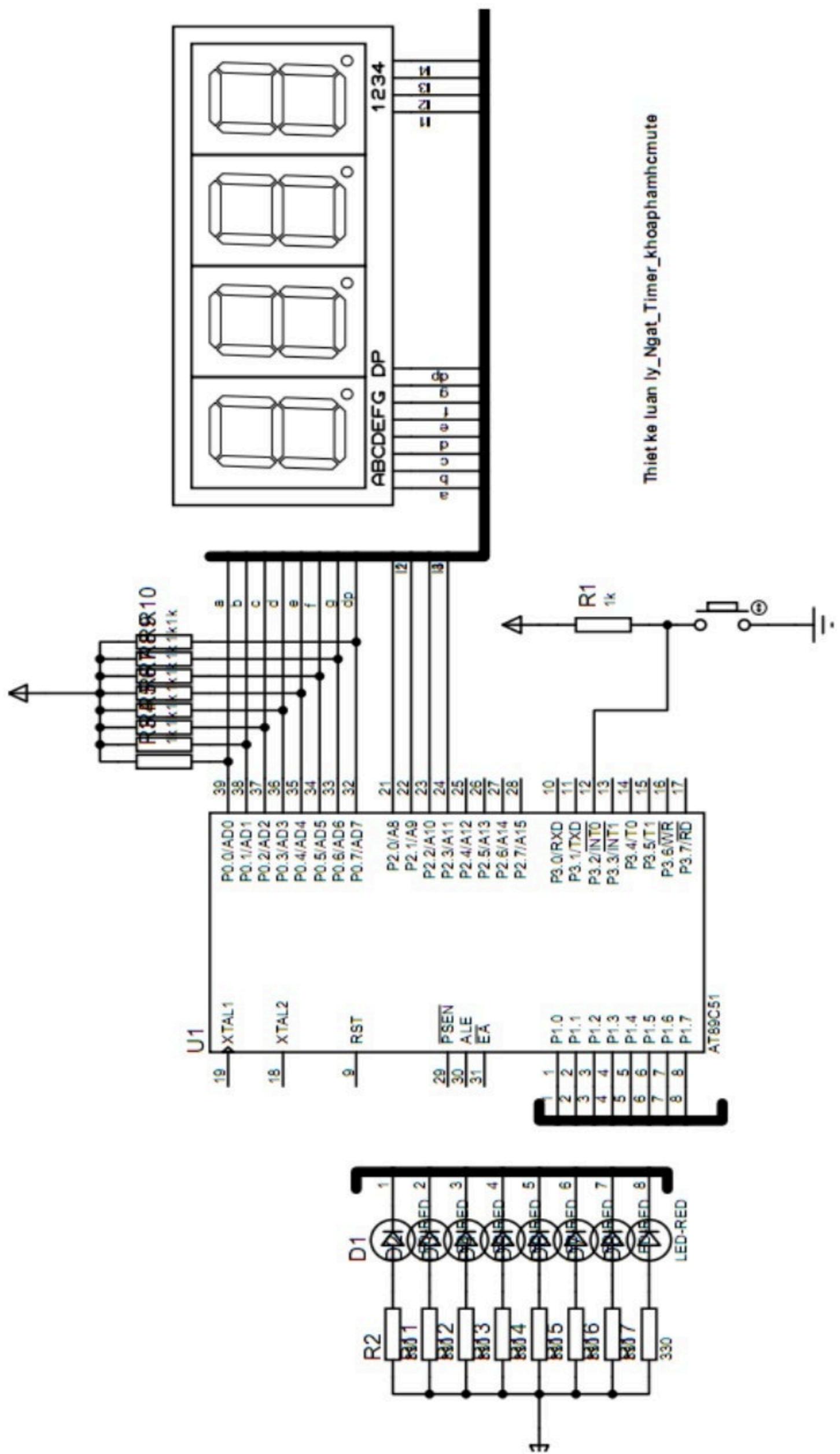
```
#include <at89x51.h>
#include <stdio.h>
#define byte unsigned char
void keyinterrupt() interrupt 2
{
    int t;
    byte keycode, scancode, flag=0xff;
    t=5000;
    while(t--);
    if(INT1==1) return;
    EX1=0;
    scancode=0xef;
    while(scancode!=0xff)
    {
        P1=scancode;
        keycode=P1;
        if((keycode&0x0f)!=0x0f)
            break;
        scancode=(keycode<<1)|0x0f;
    }
    keycode=~keycode;
    P2=keycode;
    P1=0X0F;
    while(1)
    {
        if(INT1==1)
        {
            flag=~flag;
            if(flag==0)
                break;
        }
    }
}
```

```
t=10000;  
while(t--);  
}  
EX1=1;  
return;  
}  
void main(void)  
{  
IE=0;  
EX1=1;  
EA=1;  
P2=0xFF;  
P1=0XF0;  
while(1)  
{  
}  
}
```

### **Yêu cầu thực hành:**

- Khảo sát ý nghĩa của các thanh ghi chức năng đặc biệt (SFR) IE. Ý nghĩa của việc gán EX=1, EA=1.
- Ôn tập lại lý thuyết về giao tiếp ngoại vi. Hãy trình bày ưu điểm của phương pháp sử dụng ngắt trong việc giao tiếp thiết bị ngoại vi.

**Vấn đề 4:** Áp dụng ngắt và giao tiếp hiển thị LED 7 đoạn. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.



Thiết kế luân lý Ngat\_Timer\_khoaphamhcmute

**Hình 12:** Sơ đồ nguyên lý LED 7 đoạn sử dụng ngắt

Sử dụng keilC để viết chương trình với tên **ex\_int1.c** để tạo ứng dụng tác động LED mỗi khi có sự kiện nhấn phím nhấn trên vi điều khiển.

<pre>#include &lt;AT89X51.H&gt; void SetupEx0(void); void main(void) {     SetupEx0();     while(1) ; }</pre>	<pre>void SetupEx0(void) {     EA=0;     IT0 = 1;     EX0=1;     EA = 1; }  void Ex0Isr(void) interrupt 0 {     P1_0 = !P1_0; }</pre>
---	---

**Vấn đề 5:** Sử dụng cùng sơ đồ nguyên lý như trên. Sử dụng keilC để viết chương trình với tên **ex\_int2.c** sử dụng ngắt ngoài 0 để tạo ứng dụng hiển thị LED 7 thanh tăng giá trị lên 1 mỗi khi có sự kiện nhấn phím nhấn trên vi điều khiển.

<pre>#include &lt;at89x51.h&gt; //Khai bao bien toan cuc int count=0; //Chuong trinh tao do tre void delay(int interval) {     int i,j;     for(i=0;i&lt;100;i++)     {         for(j=0;j&lt;interval;j++)     });     //Chuong trinh hien thi den     Led 7 thanh (khong dieu</pre>	<pre>//Chuong trinh con hien thi so (&lt;9999) void display_number(int iNum) {     int i;     unsigned char pos=0x08;     unsigned char temp;     for(i=0;i&lt;4;i++)         temp=iNum%10;     iNum=iNum/10;     P2=pos;      output_7seg(temp);     deLay(5);</pre>
--	---

<pre> khien dot) void output_7seg(unsigned char value) { unsigned char const mask[10]={0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90}; if(value &lt; 10){     P0=mask[value];} }  //Chuong trinh con xu Ly ngat ngoai 0 void EXT0_Process() interrupt 0{     EA=0; //Cam ngat     count++;     EA=1; //Cho phep ngat } </pre>	<pre> pos=pos&gt;&gt;1;  }  //Chuong trinh khoi tao he thong void init(){ P3_2=1; //Thiet Lap chan P3_2 Lam chan vao IE=0x81; //Cho phep ngat ngoai 0 IT0=1; //Ngat theo suon }  void main() {   init(); while(1){display_number(c ount);} } </pre>
--	---

**Vấn đề 6:** Sử dụng cùng sơ đồ nguyên lý như trên. Sử dụng keilC để viết chương trình với tên **delay\_timer.c** để tạo sự trì hoãn chính xác.

```

#include <at89x51.h>
//tao do tre chinh xac dung Timer
void delay_hardware_50ms()
{
    TMOD=TMOD & 0xF0;//Xoa thiet Lap Timer0
    TMOD=TMOD | 0x01;
    ET0=0;//Khong phat sinh ngat
    TH0=0x3C;//Thiet Lap gia tri khoi dau La 3CB0
    TL0=0xB0;//Tuong duong 15536 he 10
    TF0=0; //Xoa co tran timer 0
    TR0=1; //Khoi dong timer 0
    while(TF0==0); //Cho den khi tran
    TR0=0; //Dung timer 0
}

```

```
}

//Chuong trinh tao tre chinh xac 1 s
void delay_hardware_1s()
{
int i;
for(i=0;i<20;i++){delay_hardware_50ms();}

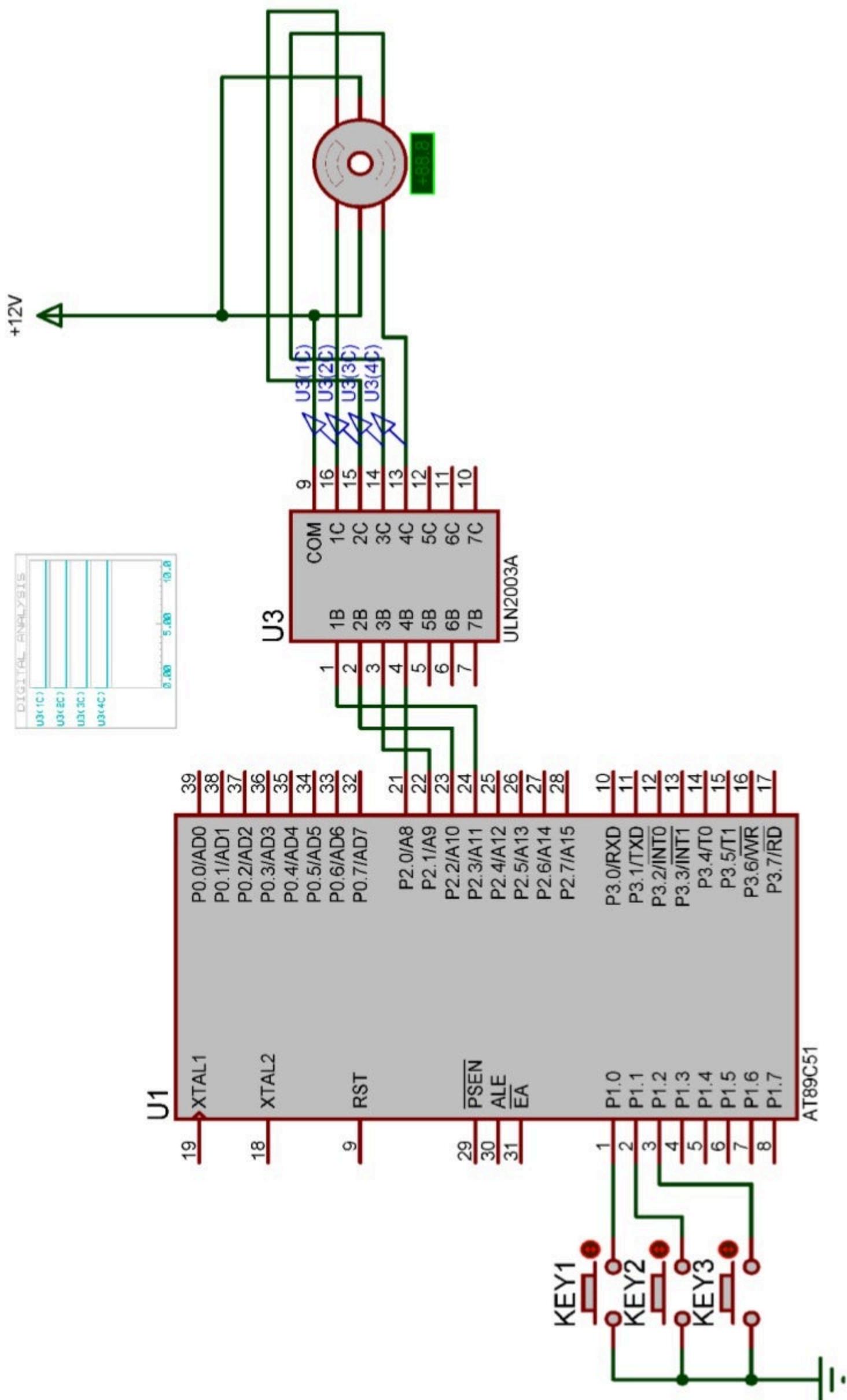
}

void main()
{
while(1)
{
P1=0xAA;
delay_hardware_1s();
P1=0x55;
delay_hardware_1s();
}
}
```

### **Yêu cầu thực hành:**

- Cho biết hạn chế của phương pháp tạo trễ ở các bài đã trình bày trước.
- Cho biết ý nghĩa của thanh ghi TMOD.
- Để sử dụng bộ định thời 0 (Timer 0) thì cần thiết phải thiết lập các bit nào?
- Sử dụng bộ đo dao động được hỗ trợ trong phần mềm Proteus, sẽ xem xét thời gian trễ mà bộ định thời timer 0 tạo ra.

**Vấn đề 7:** Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.



**Hình 13:** Sơ đồ nguyên lý giao tiếp động cơ

Sử dụng keilC để viết chương trình với tên **motor\_int1.c** để tạo ứng dụng điều khiển động cơ có nội dung như sau:

<code>#include &lt;reg51.h&gt;</code>	<code>switch(temp)</code>
<code>sbit p10=P1^0;</code>	<code>{</code>
<code>sbit p11=P1^1;</code>	<code>case DOWN : P2=0X01;</code>
<code>sbit p12=P1^2;</code>	<code>delay();</code>
<code>#define UP 20</code>	<code>delay();</code>
<code>#define DOWN 30</code>	<code>P2=0X02;//0110</code>
<code>#define STOP 40</code>	<code>delay();</code>
<code>void deDelay()</code>	<code>delay();</code>
<code>{</code>	<code>P2=0X04;//1100</code>
<code>    unsigned i,j,k;</code>	<code>delay();</code>
<code>    for(i=0;i&lt;0x02;i++</code>	<code>delay();</code>
<code>    )</code>	<code>P2=0X08;//1001</code>
<code>        for(j=0;j&lt;0x02;j++</code>	<code>delay();</code>
<code>    )</code>	<code>delay();</code>
<code>        for(k=0;k&lt;0xff;k++</code>	<code>break;</code>
<code>    );</code>	<code>case UP : P2=0X08;</code>
<code>}</code>	<code>delay();</code>
<code>main()</code>	<code>delay();</code>
<code>{</code>	<code>P2=0X04;</code>
<code>    unsigned char</code>	<code>delay();</code>
<code>temp;</code>	<code>delay();</code>
<code>    while(1)</code>	<code>P2=0X02;</code>
<code>    {</code>	<code>delay();</code>
<code>        if(p10==0)</code>	<code>delay();</code>
<code>        {</code>	<code>P2=0X01;</code>
<code>            temp=UP;</code>	<code>delay();</code>
	<code>delay();</code>
	<code>break;</code>
	<code>case STOP :</code>

<pre> P2=0X00; delay(); }  if(p11==0) {     temp=DOWN;  P2=0X00;  delay(); }  if(p12==0) {     temp=STOP; } </pre>	<pre> P2=0X00; delay(); delay(); break; }  } </pre>
--	---

### Yêu cầu thực hành:

- Có thể so sánh giá trị k trong hàm delay() với giá trị 1000 không? Tại sao?
- Sửa lại hàm delay(n) để áp dụng tạo độ trễ từ bộ định thời được hỗ trợ từ ngoại vi của bộ vi điều khiển họ 8051. Trong đó, n là tham số quyết định độ trễ của delay theo mili giây (ms).
- Vi điều khiển họ 8051 hỗ trợ các nguồn ngắn nào? Trong những trường hợp nào thì phương pháp thiết kế cần sử dụng đèn ngắn? Nêu 3 ví dụ minh họa.

## BÀI THỰC HÀNH SỐ 4

### Mục đích:

- Giới thiệu các phương pháp giao tiếp màn hình hiển thị tinh thể lỏng LCD 16\*2 với vi điều khiển.

### Sau khi kết thúc học phần này, sinh viên có thể:

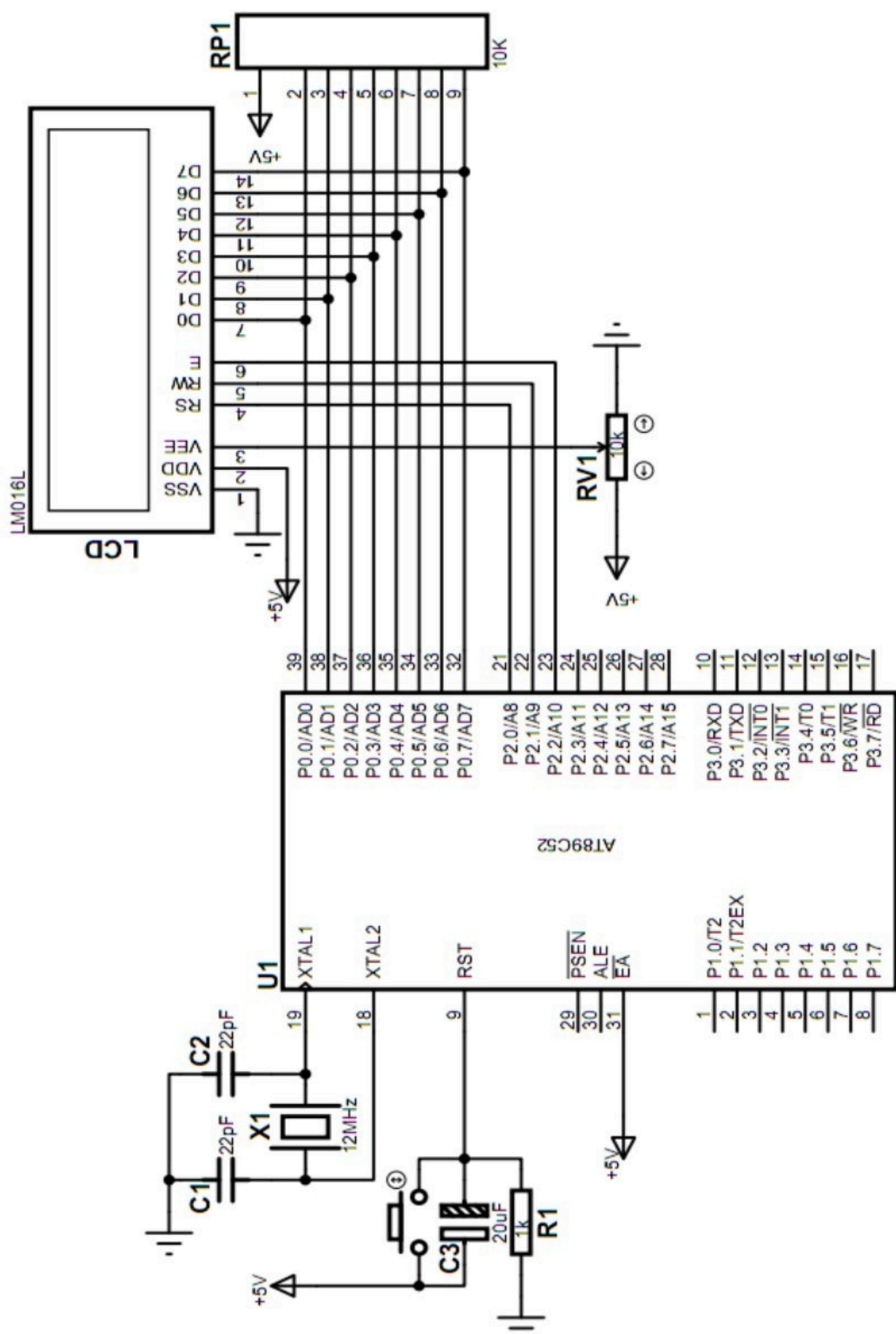
- Hiểu phương pháp giao tiếp giữa vi điều khiển và màn hình hiển thị tinh thể lỏng LCD 16\*2.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình cấp cao C dành cho các vi điều khiển (máy tính trên chip) họ 8051 để giao tiếp với màn hình LCD 16\*2.

### **Yêu cầu thực hành:**

Tham khảo datasheet của thiết bị hiển thị tinh thể lỏng LCD16\*2 từ đó hãy trả lời các câu hỏi sau đây:

- Làm thế nào LCD có thể phân biệt được mã lệnh (instruction code) và dữ liệu (data) khi nhận tín hiệu từ các chân data?
- Để gửi mã lệnh (instruction code) 01 nhằm xóa hiển thị trên LCD thì cần điều khiển chân RS như thế nào?
- Để gửi một ký tự ‘A’ hiển thị lên LCD thì chân RS cần được điều khiển như thế nào?
- Chức năng của tín hiệu E là gì?
- Khi nào thì thông tin gồm mã lệnh và dữ liệu trên các chân LCD được chốt (latched) vào trong LCD?

**Vấn đề 1:** Phương pháp kết nối LCD trực tiếp. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây:



**Hình 14:** Sơ đồ nguyên lý giao tiếp LCD

Sử dụng phần mềm Keil C viết chương trình và đặt tên là **lcd\_tructiep.c**. (đính kèm file **LCD1602.h**). Hãy cho biết ý nghĩa của chương trình sau đây:

```
#include <REGX52.H>
#include "LCD1602.h"
unsigned char TempBuffer[10];
void IntToStr(unsigned int t, unsigned char *str,
unsigned char n)
{
    unsigned char a[5]; char i, j;

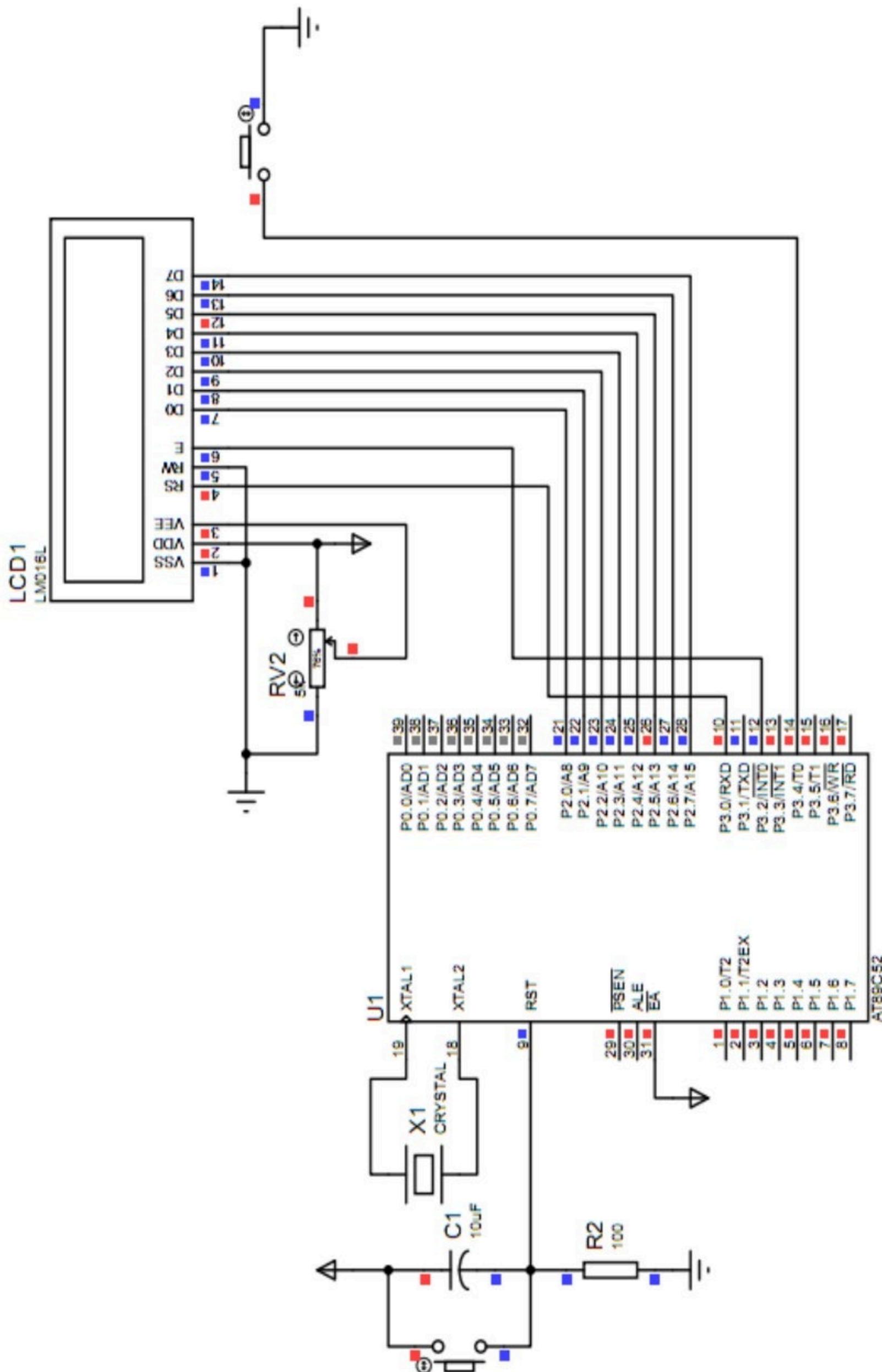
    a[0]=(t/10000)%10;
    a[1]=(t/1000)%10;
    a[2]=(t/100)%10;
    a[3]=(t/10)%10;
    a[4]=(t/1)%10;

    for(i=0; i<5; i++)
        a[i]=a[i]+'0';

    for(i=0; a[i]=='0' && i<=3; i++)
        for(j=5-n; j<i; j++)
            { *str=' '; str++; }
        for(; i<5; i++)
            { *str=a[i]; str++; }
        *str='\0';
} void DeDelay1ms(unsigned int count)
{
    unsigned int i,j;
    for(i=0;i<count;i++)
        for(j=0;j<120;j++);
}
main()
```

```
{  
    unsigned int Count = 0;  
    LCD_Initial();  
    GotoXY(0, 0);  
    Print("The 1602LCD Test");  
    GotoXY(0, 1);  
    Print("HCMUTE");  
    while(1)  
    {  
        IntToStr(Count++, &TempBuffer[0], 5);  
        GotoXY(8, 1);  
        Print(&TempBuffer[0]);  
        Delay1ms(100);  
    }  
}
```

**Vấn đề 2:** Hiệu ứng trên màn hình LCD1602 kết hợp phím nhấn.  
Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.



**Hình 15:** Sơ đồ nguyên lý giao tiếp LCD

Sử dụng phần mềm Keil C viết chương trình và đặt tên là **lcd\_hieuung\_nutnhan.c**.

```
#include "at89x52.h"
#define RS P3_0
#define RW P3_1
#define E P3_2
#define LCD P2
int nghin, tram, chuc, dv, dem=0, tg=0 ;
int const Digit_Map[10] =
{'0','1','2','3','4','5','6','7','8','9'};
unsigned char code string[] = "KHOA DIEN TU - TRUONG
DAI HOC SPKT ";
int m=0 ;
unsigned char tnghin, ttram, tchuc, tdv ;

/* ham delay */
void DeDelay(unsigned int n)
{
    unsigned int i,j;
    for(i=0;i<n;i++)
        for(j=0;j<100;j++);
}

/* ham ghi ma dieu khien Len LCD */
void ghimadk()
{
    RS=0;      /* chon thanh ghi dieu khien */
    RW=0;
    E=1;
    E=0;
    DeDelay(5);
}

/* ham goi du Lieu den LCD*/
```

```

void hienthi()
{
    RS=1; /* chon thanh ghi du Lieu */
    RW=0;
    E=1;
    E=0;
    DeLay(5);
}

/* chuong trinh con khai tao LCD */
void khoitaoLCD()
{
    LCD=0x38; /* chon kieu hien thi La ma tran
5x7 */
    ghimadk();
    LCD=0x0c; /* bat man hinh LCD */
    ghimadk();
    LCD=0x01; /* xoa man hinh LCD */
    ghimadk();
}

/* chuong trinh con giao ma - hien thi */
void giaima_ht()
{
    tnghin=Digit_Map[nghin];
    ttram=Digit_Map[tram];
    tchuc=Digit_Map[chuc];
    tdv=Digit_Map[dv];

    if (nghin==0)/* xoa so 0 vo nghia */
    {
        tnghin = ' ';
        if (tram==0) {ttram = ' ';}
        if(chuc==0){tchuc = ' ';}
        LCD = tnghin ;
        hienthi();
        LCD = ttram ;
        hienthi();
        LCD = tchuc ;
    }
}

```

```

        hienthi();
        LCD = tdv ;
        hienthi();
    }

/* chuong trinh ngat T0 */
void NGAT_T0() interrupt TF0_VECTOR
{
    dv=dv+1;
    if (dv==10)
    {
        chuc=chuc+1;
        dv=0;
    }
    if (chuc==10)
    {
        tram=tram+1;
        chuc=0;
        dv=0;
    }
    if (tram==10)
    {
        nghin=nghin+1;
        tram=0;
        chuc=0;
        dv=0;
    }
    if (nghin==10)
    {
        nghin=0;
        tram=0;
        chuc=0;
        dv=0;
    }
}

LCD=0xcc; /* hien thi ghi de len so cu */
Delay(1);
ghimadk();
giaima_ht();

```

```

TH0=0xff;
TL0=0xff;

LCD=0x80 + dem /* quay lai hien thi ky tu
tiep theo o vi tri tiep theo truoc khi ngat */
Delay(1);
ghimadk();
}

/* chuong trinh chinh */
void main(void)
{
    TMOD=0x05;          /* khai tao couter 0 */
    TH0=0xff; /* gan th0 va tl0 la 0ffh */
    TL0=0xff;
    TF0=0;             /* xoa co tran */
    TR0=1;             /* cho phep dem */
    IE=0x82;           /* cho phep ngat T0 */
    nghin=0;
        tram=0;
        chuc=0;
        dv=0;
        khoitaoLCD(); /* khai tao LCD */
        LCD=0xcc; /* dua con tro vao vi tri
thu 13 cua dong thu 2 */
    Delay(1);
    ghimadk();
        gaiima_ht();/* hien thi duoc so dem
o dong thu 2 */

while (1)          /* vong Lap vo tan */

{
    LCD=0x80;      /* dua con tro vao vi tri
thu nhat cua dong thu 1 */
    Delay(1);
    ghimadk();
}

```

```

        while(dem < 16)
        {
            dem++;
            LCD = string[m];/* Lay Lan Luot 16 ki tu
            hienthi();
            DeDelay(1);
            m++;
        }
        DeDelay(200);
        dem = 0;
        tg++; /* vi tri cua ky tu Lay ma dau
tien duoc tang Len 1 */
        if (tg==70)
        {
            tg = 0;
        }
        m = tg ;
    }
}

```

### **Yêu cầu thực hành:**

- Hãy cho biết trong những trường hợp thiết kế nào thì cần sử dụng thiết bị hiển thị là màn hình LCD16\*2 hoặc LED 7 đoạn. Từ đó, đưa ra ưu điểm của màn hình hiển thị LCD16\*2.
- Thiết kế một sơ đồ nguyên lý với 1 vi điều khiển họ 8051, 1 màn hình LCD16\*2, 8 LED 7 đoạn, 1 nút nhấn và một số thành phần phần cứng khác (tùy phương pháp thiết kế) hãy viết chương trình để thực hiện yêu cầu:
  - + LCD hiển thị tên sinh viên ở hàng 1.
  - + LCD hiển thị MSSV ở hàng 2.
  - + LED 7 đoạn hiển thị MSSV ví dụ 11119001.

Mặc định thì chỉ hiển thị thông tin lên LCD, nếu nhấn nút nhấn thì **lập tức** LCD tắt và hiển thị qua LED 7 đoạn, tiếp tục nhấn nút thì lại hiển thị qua LCD.

## BÀI THỰC HÀNH SỐ 5

### Mục đích:

- Giới thiệu phương pháp giao tiếp vi điều khiển và bộ chuyển đổi tương tự sang số ADC.
- Giới thiệu một số ứng dụng đơn giản với ADC0809.

### Sau khi kết thúc học phần này, sinh viên có thể:

- Hiểu phương pháp giao tiếp giữa vi điều khiển và bộ chuyển đổi tương tự sang số ADC.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình cấp cao C dành cho các vi điều khiển (máy tính trên chip) họ 8051 để giao tiếp với ADC.

### **Yêu cầu thực hành:**

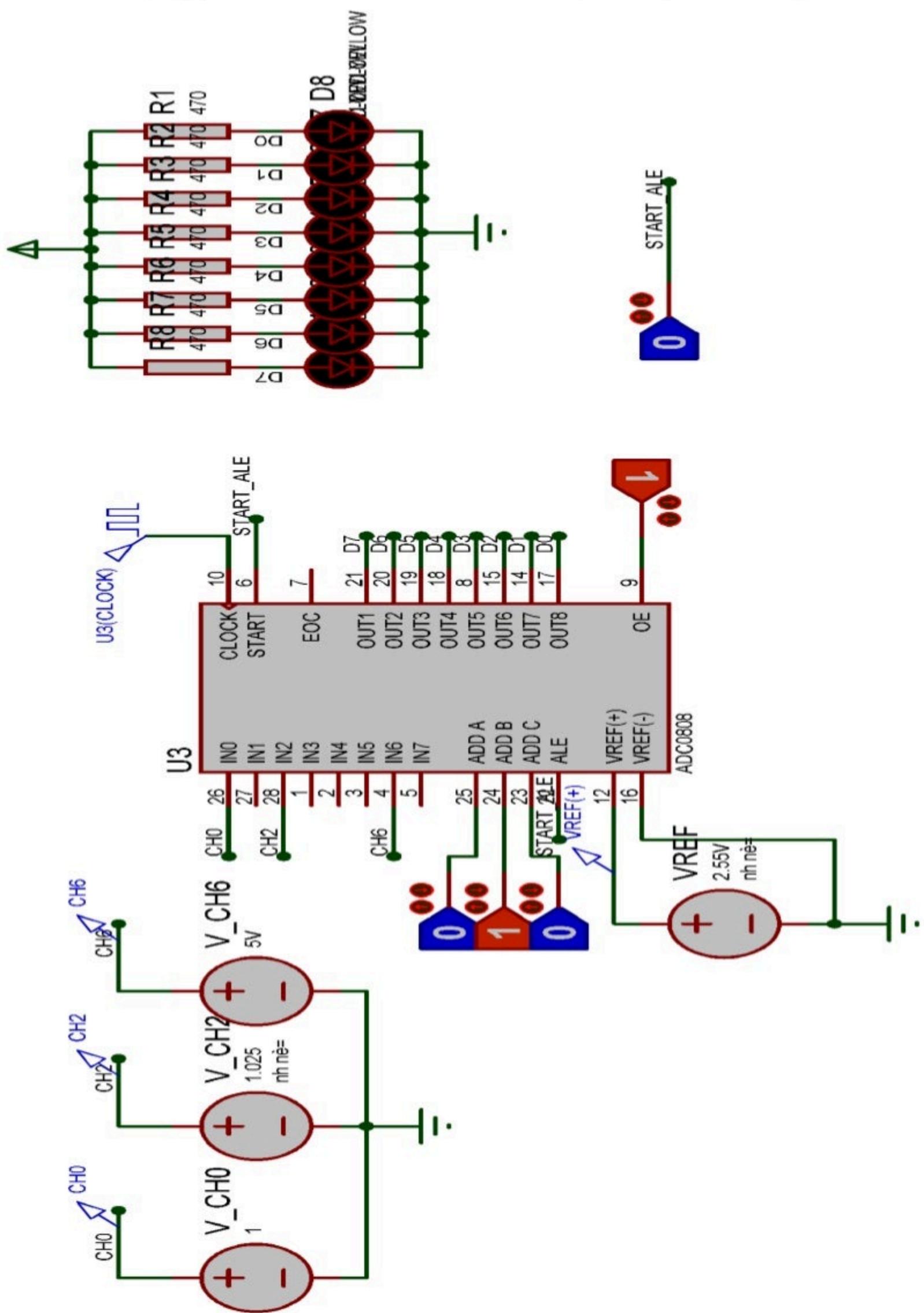
Tham khảo datasheet của linh kiện ADC như trên hình từ đó hãy trả lời các câu hỏi sau đây:

- Tần số hoạt động tối đa của ADC0809 là bao nhiêu?
- Độ rộng xung tối thiểu đối với tín hiệu ALE và OE là bao nhiêu?
- Ý nghĩa của tín hiệu ALE là gì?
- Hãy nêu các bước tuần tự để thực hiện thao tác chuyển dữ liệu và lấy dữ liệu nhị phân tại ngõ ra của ADC0809.
- Nếu ADC0809 được kết nối với nguồn tham chiếu Vref = 2.56V như hình 16. Hãy tính giá trị:
  - o Bước nhảy (step size) cho một bậc chuyển đổi là bao nhiêu?
  - o Nếu bước nhảy là 2mV thì giá trị điện áp tham chiếu đầu vào là bao nhiêu?
  - o Giá trị tối đa của mỗi kênh đầu vào là bao nhiêu?
  - o Nếu đầu vào là 1.2V thì giá trị ngõ ra từ D0 đến D7 sẽ là bao nhiêu?
  - o Nếu ngõ ra từ D0 đến D7 là 11111111 thì ngõ vào tương ứng chuyển đổi là bao nhiêu?
  - o Nếu ngõ ra từ D0 đến D7 là 10011100 thì ngõ vào tương ứng chuyển đổi là bao nhiêu?
  - o Nếu ngõ vào tham chiếu được thay đổi thành Vref = 5V, hãy trả lời lại các câu hỏi trên.

Tham khảo datasheet của cảm biến nhiệt độ bán dẫn LM35 và một số các cảm biến nhiệt độ khác từ đó hãy trả lời các câu hỏi sau đây:

- Trong một hệ thống đo lường và điều khiển bằng máy tính vai trò của Transducer là gì?
- Các dạng ngõ ra của transducer là gì?
- LM35/LM34 là một thiết bị tuyến tính hay phi tuyến (linear, nonlinear). Thảo luận lợi ích của các thiết bị tuyến tính và phi tuyến.
- Giải thích vai trò của tần xử lý tín hiệu đầu vào trong một hệ thống đo lường và thu thập dữ liệu (data acquisition).

**Vấn đề 1:** Khảo sát hoạt động trên bộ chuyển đổi tương tự sang số ADC. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.



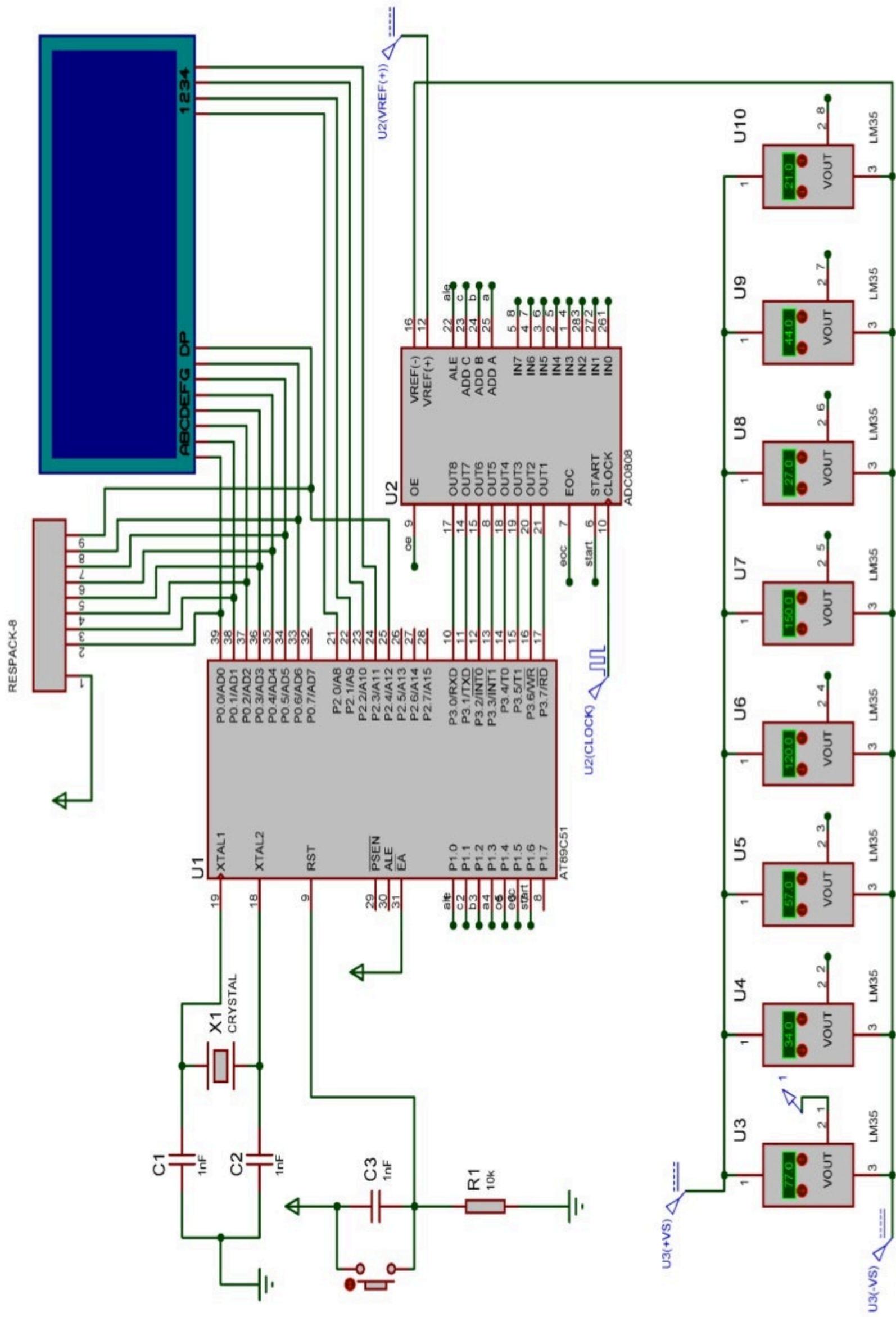
**Hình 16:** Sơ đồ kết nối ADC0808

Khảo sát hoạt động của bộ chuyển đổi tương tự sang số ADC được kết nối như trên hình, trong đó:

- Ngõ vào tương tự 8 đường được kết nối qua kênh IN0, IN2, IN6 với các nguồn điện áp.
- Ngõ vào địa chỉ 3 đường chọn kênh A, B, C được kết nối với các công tắc nhị phân.
- Ngõ vào đường tín hiệu điều khiển khác như ALE, OE được điều khiển độc lập bằng công tắc.
- Ngõ vào điện áp tham chiếu Vref được kết nối với nguồn điện áp 2.55V.
- Ngõ vào nguồn xung nhịp được cung cấp thông qua chân CLOCK.
- Ngõ ra kết quả chuyển đổi được kết nối với 8 LED đơn.

Hãy thay đổi nguồn điện áp tại 3 kênh ngõ vào tương ứng và điều khiển các đường tín hiệu ALE, OE, A, B, C như trong giản đồ thời gian (waveform) để khảo sát phương thức thực hiện chuyển đổi của ADC.

**Vấn đề 2:** Ứng dụng bộ chuyển đổi tương tự sang số ADC và cảm biến nhiệt độ LM35 để thiết kế bộ đo nhiệt độ đa kênh. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.



**Hình 17:** Sơ đồ nguyên lý mạch đo nhiệt độ hiển thị LED 7 đoạn

Sử dụng phần mềm Keil C viết chương trình và đặt tên là **7led\_lm35\_adc.c**.

```
#include <regx51.h>
#define uchar unsigned char
#define uint unsigned int
#define port0 P0
#define ale      P1_0
#define c P1_1
#define b P1_2
#define a P1_3
#define oe P1_4
#define eoc P1_5
#define start P1_6
Int dig[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92
,0x82,0xF8,0x80,0x90};
uchar tram,chuc,donvi;
uint x;
void delay(int x)
{
    int i;
    for(i=0;i<x;i++);
}
void tachso(void)
{
    tram=x/100;
    chuc=(x/10)%10;
    donvi=x%10;
}
void giaima(int x)
{
    if (x==1)
    {a=0;b=0;c=0;}
    if (x==2)
    {a=0;b=0;c=1;}
    if (x==3)
```

```
{a=0;b=1;c=0;}
if (x==4)
{a=0;b=1;c=1;}
if (x==5)
{a=1;b=0;c=0;}
if (x==6)
{a=1;b=0;c=1;}
if (x==7)
{a=1;b=1;c=0;}
if (x==8)
{a=1;b=1;c=1;}
}void main()
{
    int i;
    oe = 1;
    for(i=0;i<8;i++)
    {
        int tam;
        for(tam=0;tam<30;tam++)
        {
            P2=0x01;
            P0=dig[i+1];
            P2_4=0;
            delay(1000);
            start=0;ale=0;eoc=1;
            gaiima(i+1);
            delay(100);
            start=1;ale=1;
            delay(500);
            start=0;ale=0;
            while(eoc==0);
            delay(100);
            eoc=1;
            x=P3;
            tachso();
        }
    }
}
```

```
P2=0x02;  
P0=dig[tram];  
P2_4=1;  
delay(500);  
P2=0x04;  
P0=dig[chuc];  
P2_4=1;  
delay(500);  
P2=0x08;  
P0=dig[donvi];  
P2_4=1;  
delay(500);  
}  
}  
}
```

### **Yêu cầu thực hành:**

Hãy biên dịch và nạp Chương trình vào bộ nhớ vi điều khiển, từ đó khảo sát phương thức hoạt động của hệ thống.

# BÀI THỰC HÀNH SỐ 6

## Mục đích:

- Giới thiệu các phương pháp giao tiếp vi điều khiển sử dụng truyền dữ liệu nối tiếp bất đồng bộ UART.
- Giới thiệu ứng dụng giao tiếp máy tính cơ bản thông qua chuẩn truyền UART.
- Giới thiệu chức năng của bộ chuyển đổi MAX232 trong việc giao tiếp máy tính.
- Giới thiệu phương pháp xây dựng ứng dụng đồ họa (GUI) điều khiển từ máy tính.

## Sau khi kết thúc học phần này, sinh viên có thể:

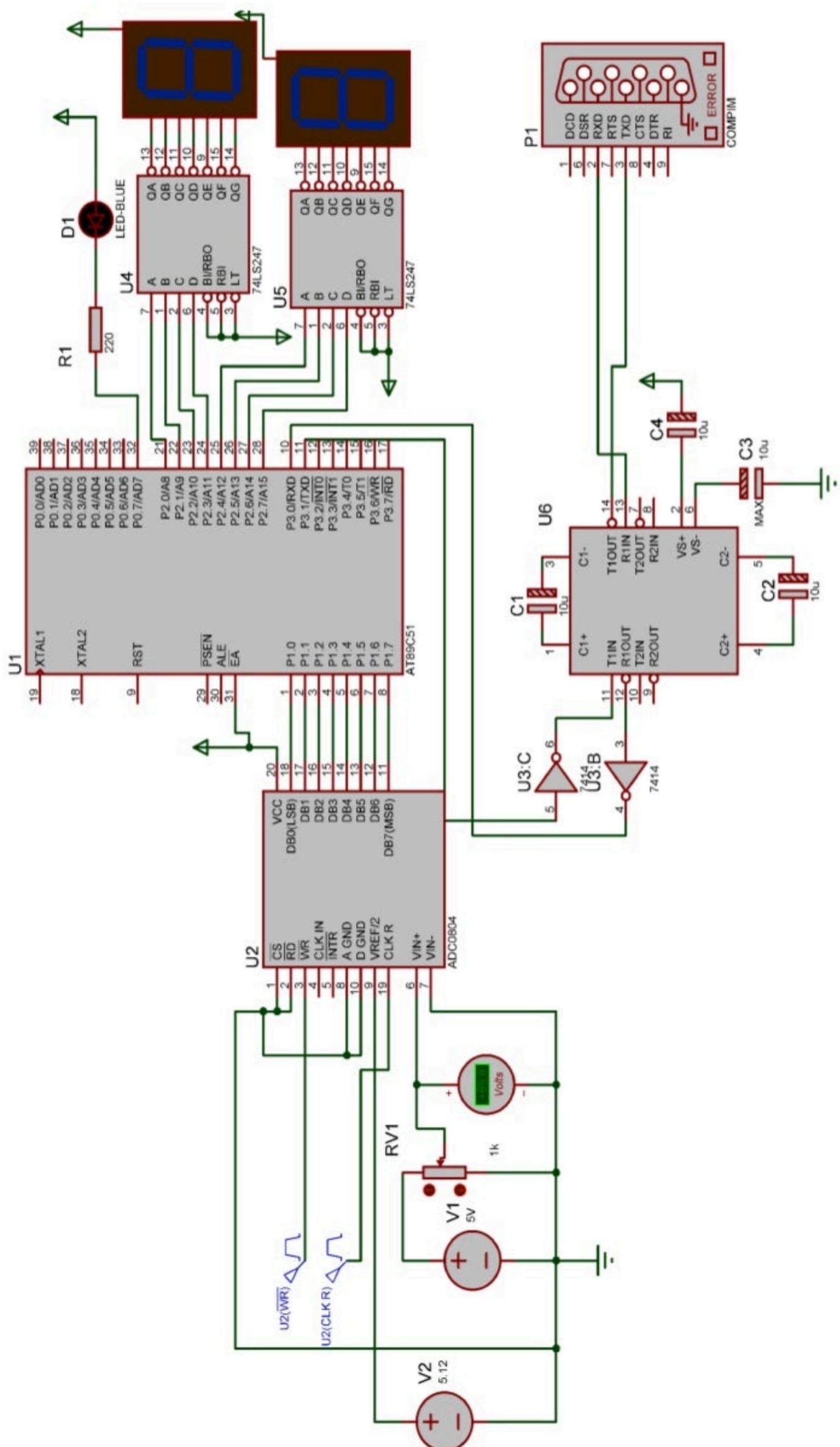
- Hiểu phương pháp giao tiếp giữa vi điều khiển và máy tính đa dụng thông qua UART.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình cấp cao C dành cho các vi điều khiển (máy tính trên chip) họ 8051 kết hợp với giao diện đồ họa GUI sử dụng C# để giao tiếp với máy tính cũng như vi điều khiển khác.

## **Yêu cầu thực hành:**

Tham khảo datasheet của linh kiện vi điều khiển 8051, MAX232 và chuẩn UART từ đó hãy trả lời các câu hỏi sau đây:

- Các đường tín hiệu TxD và RxD có tương thích với chuẩn TTL không?
- Vai trò của chip MAX232 là gì khi thực hiện kết nối với máy tính trong mô hình đo lường và điều khiển từ máy tính.
- Nếu tần số cung cấp cho vi điều khiển họ 8051 là 11.0592Mhz thì tốc độ baud tối đa cho họ 8051 là bao nhiêu?
- Làm thế nào để đạt được tốc độ baud tối đa?
- Vai trò của các thanh ghi TI và RI là gì?
- Vi điều khiển họ 8051 có thể truyền dữ liệu song công (full duplex) đúng hay sai?
- Để thực hiện truyền được song công, các đường tín hiệu nào cần thiết cần kết nối giữa 8051 và máy tính đa dụng (PC).

Giao tiếp giữa vi điều khiển họ 8051 với máy tính trong ứng dụng đo lường và điều khiển từ máy tính. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.



Hình 18: Sơ đồ nguyên lý giao tiếp máy tính

Sử dụng phần mềm Keil C viết chương trình và đặt tên là **serial\_MCU\_PC\_temp.c**.

```
#include <at89x52.h>
unsigned char chuc, dvi, rdata, nhietdo;

void setup()
{
    TMOD=0x20;                      //;enable timer1,
mode 2 (auto reload)
    TH1=-3;                          //9600 Baud rate
    SCON=0x50;                      //;8 bit uart, 1 stop
bit, REN enabled, timer1
    TR1=1;
    IE=0x90;
}
void serint(void) interrupt 4 using 1
{
    if(RI)
    {
        rdata=SBUF;
        RI=0;
        switch(rdata)
        {
            case('0'): {P0_7=0; break; }
            case('1'): {P0_7=1;break; }
            case('a'): {TH1=-24;break; }
            case('b'): {TH1=-12;break; }
            case('c'): {TH1=-6;break; }
            case('d'): {TH1=-3;break; }
        }
    }
}
void main()
{
```

```
unsigned char trans_data;
setup();
while(1)
{
    // Lay du dieu
    nhietdo=P1;
    // chuyen doi hien thi
    chuc= (nhietdo/10)<<4;
    dvi=nhietdo%10;
    P2=chuc|dvi;
    // truyen du Lieu
    trans_data=P2;
    SBUF=trans_data;
    while(TI==0);
    TI=0;
}
}
```

## BÀI THỰC HÀNH SỐ 7

### Mục đích:

- Giới thiệu các phương pháp giao tiếp vi điều khiển và bộ nhớ dữ liệu SRAM 62256.
- Giới thiệu các phương pháp giao tiếp vi điều khiển và bộ nhớ chương trình ROM 27256.

### Sau khi kết thúc học phần này, sinh viên có thể:

- Hiểu các phương pháp mở rộng bộ nhớ.
- Thiết kế một chương trình firmware đơn giản sử dụng ngôn ngữ lập trình cấp cao C dành cho các vi điều khiển (máy tính trên chip) họ 8051 để giao tiếp với các loại bộ nhớ.

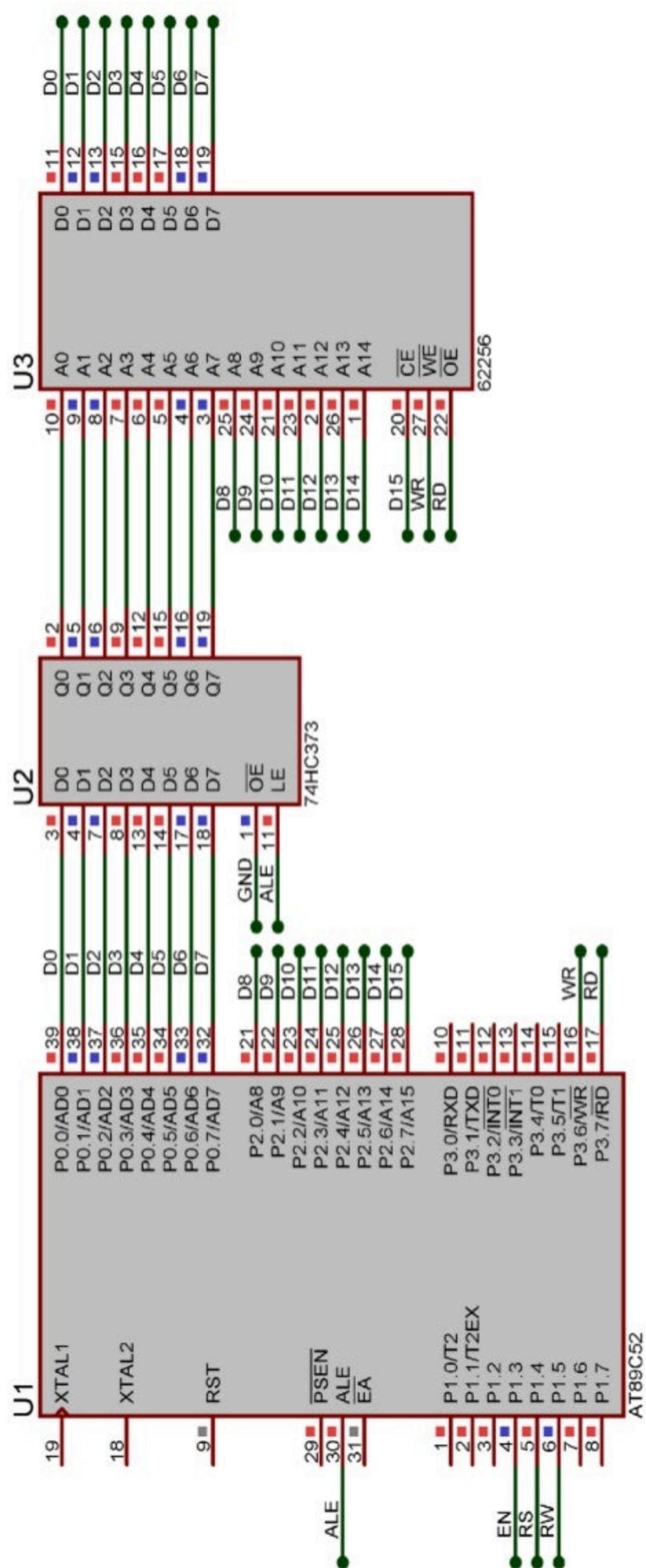
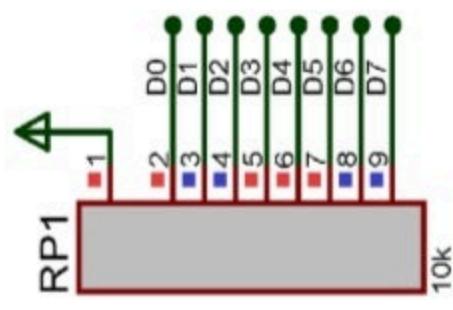
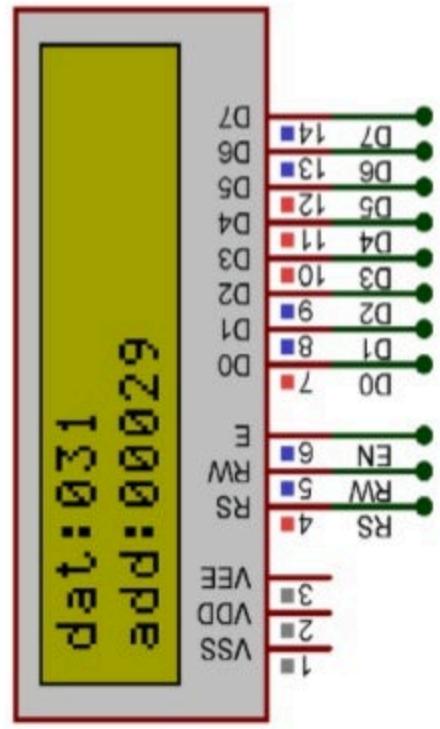
## **Yêu cầu thực hành:**

Tham khảo datasheet của bộ nhớ RAM 62256/ROM 27256, 74373 từ đó hãy trả lời các câu hỏi sau đây:

- Dung lượng bộ nhớ (Capacity) là bao nhiêu cho từng loại?
- Tổ chức bộ nhớ (sơ đồ khối) như thế nào?
- Số lượng chân địa chỉ là bao nhiêu?
- Số lượng chân dữ liệu là bao nhiêu?
- Trên vi điều khiển họ 8051 làm thế nào để xác định rằng Chương trình (program code) được nạp (fetched) từ bộ nhớ ROM bên ngoài (external ROM)?
- Vai trò của chân PSEN trên vi điều khiển họ 8051 là gì khi kết nối với ROM ngoại?
- Trong kết nối với ROM ngoại, giải thích sự khác nhau giữa các tín hiệu RD và PSEN, chúng được sử dụng ở đâu?
- Giải thích vai trò của các cổng P0, P2 trên vi điều khiển khi kết nối với ROM ngoại.
- Vai trò của IC 74373 là gì?
- Bộ nhớ tối đa bên ngoài có thể mở rộng của vi điều khiển họ 8051 là bao nhiêu cho (a) không gian bộ nhớ dữ liệu (b) không gian bộ nhớ Chương trình?
- Khi kết nối với một bộ nhớ bên ngoài, nếu chỉ các chân như P2.0 đến P2.5 được sử dụng thì kích thước bộ nhớ mở rộng được bao nhiêu?
- Khi kết nối đến bộ nhớ ngoại thì các chân P2.6 và P2.7 có được sử dụng cho mục đích điều khiển ngoại vi không?

**Vấn đề 1:** Giao tiếp giữa vi điều khiển họ 8051 với bộ nhớ RAM ngoại 62256. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.

LCD1  
LM016L



Hình 19: Sơ đồ nguyên lý giao tiếp bộ nhớ RAM ngoại

Sử dụng keilC để viết chương trình với tên **RAM\_62256\_LCD.c** để tạo chương trình minh họa cho giao tiếp giữa vi điều khiển và RAM ngoại (đi kèm là thư viện LCD1602.h).

```
#include<reg52.h>
#include<LCD1602.h>
unsigned char xdata *pp;

void write_ram(unsigned int add,unsigned char dat)
{
    pp=0x0000+add;
    *pp=dat;
}

unsigned char read_ram(unsigned int add)
{
    unsigned char dat;
    pp=0x0000+add;
    dat=*pp;
    return dat;
}

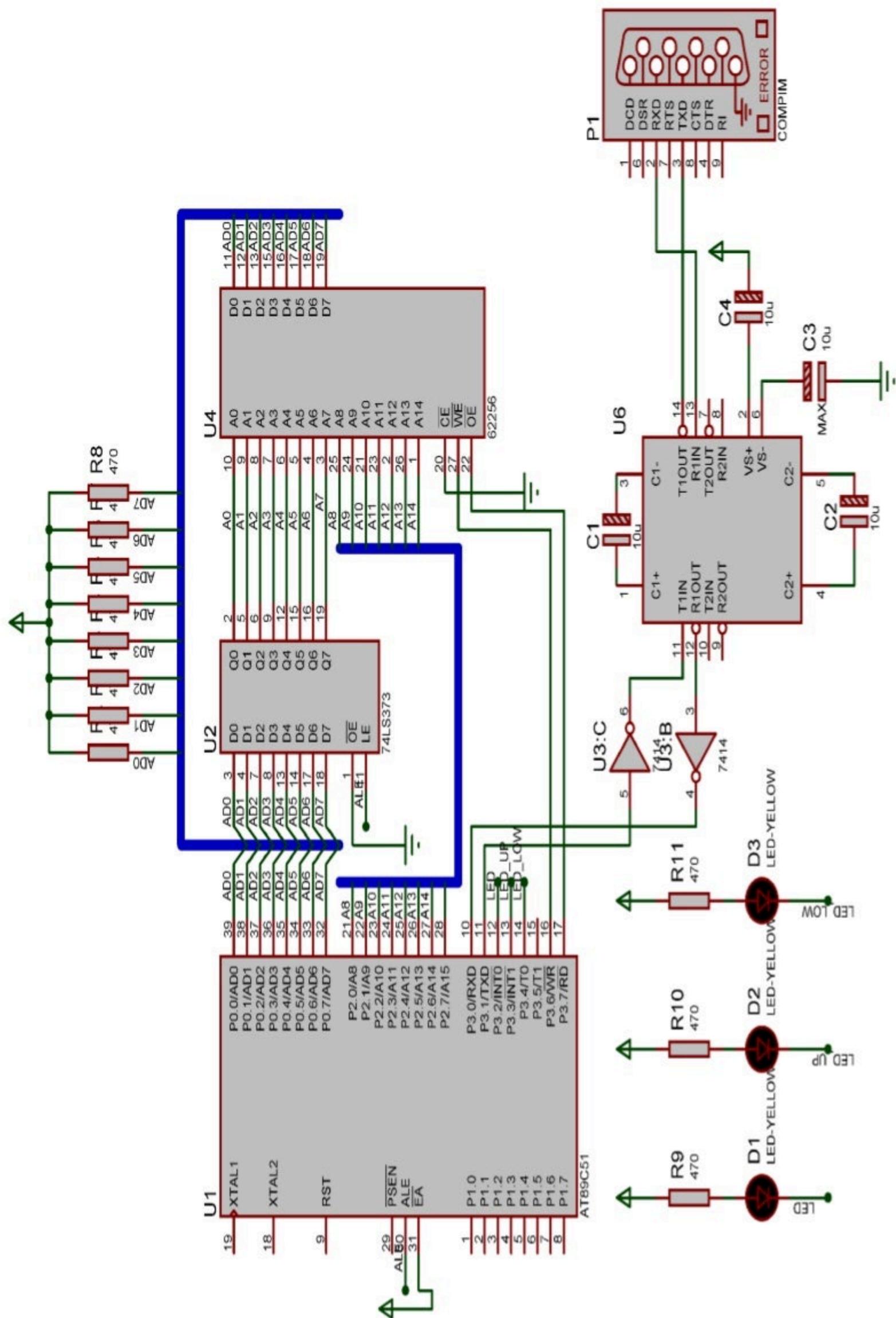
void main(void)
{
    unsigned int i,j;
    unsigned char dat;
    unsigned char
Line_one[]={ 'd', 'a', 't', ':', 0, 0, 0 };
    unsigned char
Line_two[]={ 'a', 'd', 'd', ':', 0, 0, 0, 0, 0 };
    init_LCD();
    for(i=0;i<32768;i++)
    {
        write_ram(i,i%256);
    }
}
```

```

for(i=0;i<4;i++)
{
    display_str(0,i,Line_one[i]);
    display_str(1,i,Line_two[i]);
}
while(1)
{
    for(i=0;i<32768;i++)
    {
        dat=read_ram(i);
        Line_one[4]=dat/100+0x30;
        Line_one[5]=dat/10%10+0x30;
        Line_one[6]=dat%10+0x30;
        Line_two[4]=i/10000+0x30;
        Line_two[5]=i/1000%10+0x30;
        Line_two[6]=i/100%10+0x30;
        Line_two[7]=i/10%10+0x30;
        Line_two[8]=i%10+0x30;
        for(j=4;j<7;j++)
        {
            display_str(0,j,Line_one[j]);
        }
        for(j=4;j<9;j++)
        {
            display_str(1,j,Line_two[j]);
        }
    }
    delay1ms(1000);
}
}

```

**Vấn đề 2:** Giao tiếp giữa vi điều khiển họ 8051 với bộ nhớ RAM ngoại 62256 và truyền nhận giữa liệu giữa máy tính và vi điều khiển. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.



**Hình 20:** Sơ đồ nguyên lý giao tiếp bộ nhớ RAM ngoại và máy tính

Sử dụng keilC để viết chương trình với tên **RAM\_62256\_uart.c** để tạo chương trình minh họa cho giao tiếp giữa vi điều khiển và RAM ngoại và truyền nhận dữ liệu với máy tính.

```
// RAM 62256 A0-A14
#include <AT89X51.H>

#define B9600    -3          //baud 9600  11.059MHz
#define B4800    -6          //baud 4800  11.059MHz
#define B2400    -12         //baud 2400  11.059MHz
#define B1200    -24         //baud 1200  11.059MHz
//*****Dinh nghia chan*****
#define DBLED    P3_2
#define LED_UP   P3_3
#define LED_DOWN P3_4

unsigned char xdata *pp;

unsigned int i;
void SetupSerial(char baud);

void TxChar(unsigned char ch);
void TxStr(char *str);
void TxCrLf(void);
unsigned char RxChar(void);

void DeDelay(unsigned int Loop);

void ResetXRam(void);
void write_ram(unsigned int add,unsigned char dat);
unsigned char read_ram(unsigned int add);
```

```

void main (void)
{
    unsigned char r_add,d_add;

    SetupSerial(B9600);
    DBLED = 1;
    LED_UP=1;
    LED_DOWN=1;

    TxStr("Test External RAM 62C256");
    TxCrLf();
    ResetXRam();
    /**
    while(1)
    {

        if(RI)
        {
            RI=0;
            r_add=Rxchar();
            DBLED=0;

            TxChar(r_add);
            TxCrLf();

            switch(r_add)
            {
                case 'u':
                    LED_UP=0;
                    d_add=read_ram(r_add);
                    TxChar(r_add-32);
                    break;
                case 'L':
                    LED_DOWN=0;
                    d_add=read_ram(r_add);
    
```

```

        TxChar(r_add+32);
        break;
    default:
        break;
    }
}
}

void SetupSerial(char baud)
{
    SCON = 0x50;
    TMOD = 0x20;
    TH1 = baud;
    TF1 = 0;
    TR1 = 1;
}
//*****Transmit*****
void TxChar(unsigned char ch)
{
    SBUF = ch;
    while(TI==0);
    TI=0;
}

void TxStr(char *str)
{
    while(*str!=0)
    {
        TxChar(*str);
        str++;
    }
}
void TxCrLf(void)
{
    TxChar(13);
    TxChar(10);
}

```

```

}

//*****Receive*****
unsigned char RxChar(void)
{
    RI = 0;
    return(SBUF);
}

//*****Delay*****
void Delay(unsigned int Loop)
{
    while(Loop--);
}

void ResetXRam(void)
{
    unsigned int j;
    TxCrLf();
    TxStr("Reset Xram");
    TxCrLf();

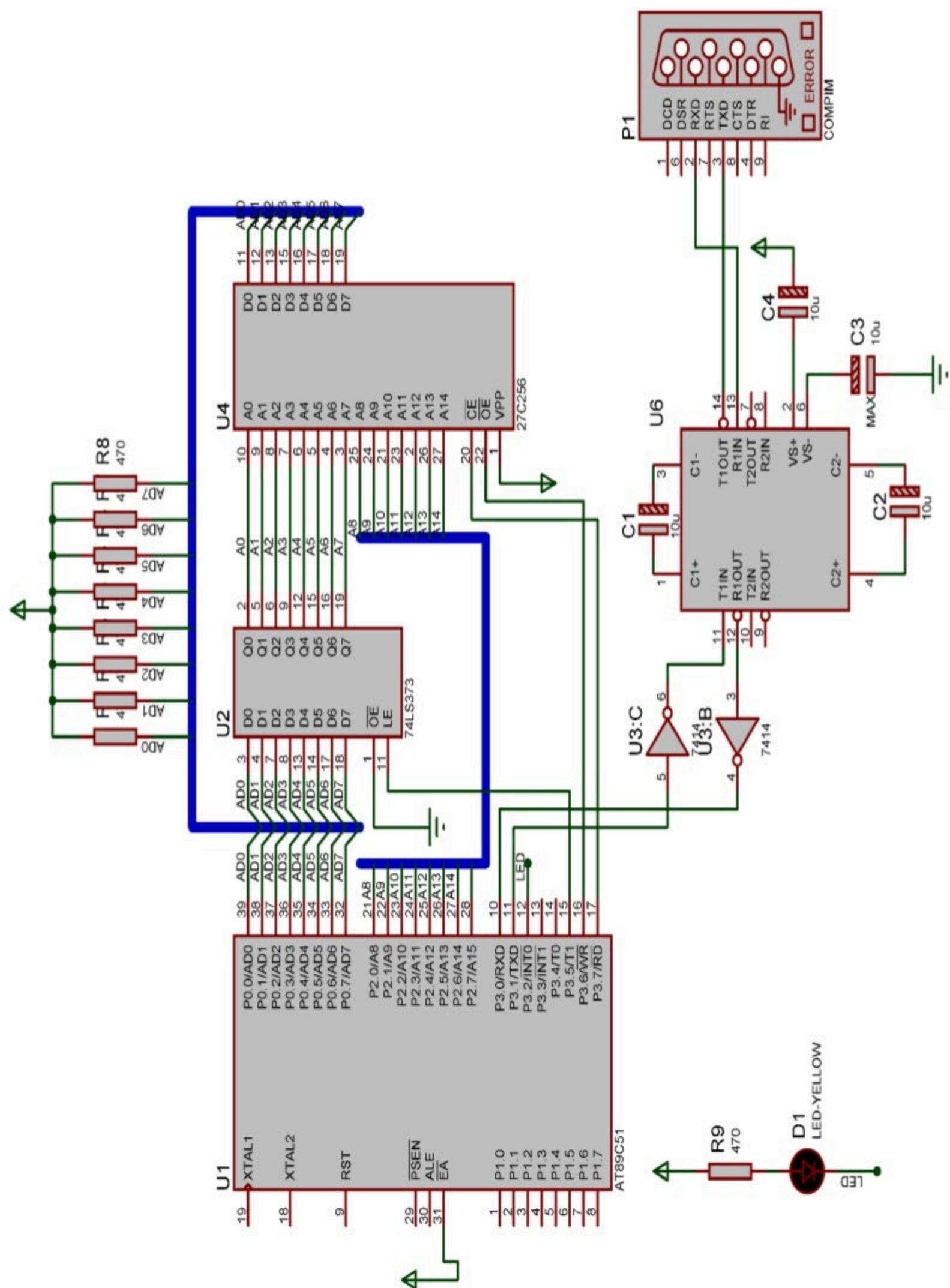
    //for(j=0;j<32767;j++)write_ram(j,j%256);
    for(j=0;j<257;j++)write_ram(j,j%256);
    TxStr("Reset Ram OK");
}

void write_ram(unsigned int add,unsigned char dat)
{
    pp=0x0000+add;
    *pp=dat;
}

unsigned char read_ram(unsigned int add)
{
    unsigned char dat;
    pp=0x0000+add;
    dat=*pp;
    return dat;
}

```

**Vấn đề 3:** Giao tiếp giữa vi điều khiển họ 8051 với bộ nhớ ROM ngoại 27256 và truyền nhận giữa liệu giữa máy tính và vi điều khiển. Sử dụng phần mềm Proteus ISIS vẽ mạch điện dưới đây.



**Hình 21:** Sơ đồ nguyên lý giao tiếp bộ nhớ ROM ngoại

Sử dụng keilC để viết chương trình.

```
#include <AT89X51.H>
#define B9600 -3 //baud 9600 11.059MHz
#define B4800 -6 //baud 4800 11.059MHz
#define B2400 -12 //baud 2400 11.059MHz
#define B1200 -24 //baud 1200 11.059MHz
#define DBLED P3_2
#define LAT P3_5
#define OE P3_6 //G
#define CE P3_7 //E
extern void SetupSerial(char baud);
extern void TxChar(unsigned char ch);
extern unsigned char RxChar(void);
extern void TxStr(char *str);
extern void TxHex(unsigned char ch);
extern void TxCrLf(void);
void SetReadXRom(void);
void ReadXRom(void);
void Delay(unsigned int Loop);
unsigned char ReadByte(unsigned int Addr);
unsigned int i;
void main (void)
{
    unsigned char ch;
    SP = 0x62;
    SetupSerial(B9600);
    TxStr("External rom");
    TxCrLf();
    while(1)
    {
        DBLED = 1;
        if(RI)
        {
            DBLED = 0;
```

```
ch = RxChar();
TxChar(ch);
switch(ch)
case 0x12: //CtrlL_R

SetReadXRom();
ReadXRom();
break;
default:
break;
}
}
}
}

void SetReadXRom(void)
{
CE = 1;
OE = 1;
Delay(15000);
}

void ReadXRom(void)
{
unsigned char ch;
for(i=256;i<276;i++)
{
ch = ReadByte(i);
TxHex(ch);
}
}

unsigned char ReadByte(unsigned int Addr)
{
unsigned char RetVal,HiAddr,LoAddr;
LoAddr = (unsigned char) Addr;
HiAddr = (unsigned char) (Addr >> 8);
```

```

P0 = LoAddr;
LAT = 1;
LAT = 0;
HiAddr = (HiAddr & 0x7f) | (P2 & 0x80);
P2 = HiAddr;
P0 = 0xff;
CE = 0;
OE = 0;
Delay(0);
RetVal = P0;
CE = 1;
OE = 1;
return(RetVal);
}
void Delay(unsigned int Loop)
{
while(Loop--);
}
//siop.c
#include <AT89X52.H>
void SetupSerial(char baud);
void TxChar(unsigned char ch);
unsigned char RxChar(void);
void TxStr(char *str);
void TxHex(unsigned char ch);
void TxCrLf(void);
void SetupSerial(char baud)
{
SCON = 0x52;
TMOD = 0x20;
TH1 = baud;
TF1 = 0;
TR1 = 1;
}
void TxChar(unsigned char ch)
{
while(!TI);
TI = 0;
SBUF = ch;
}

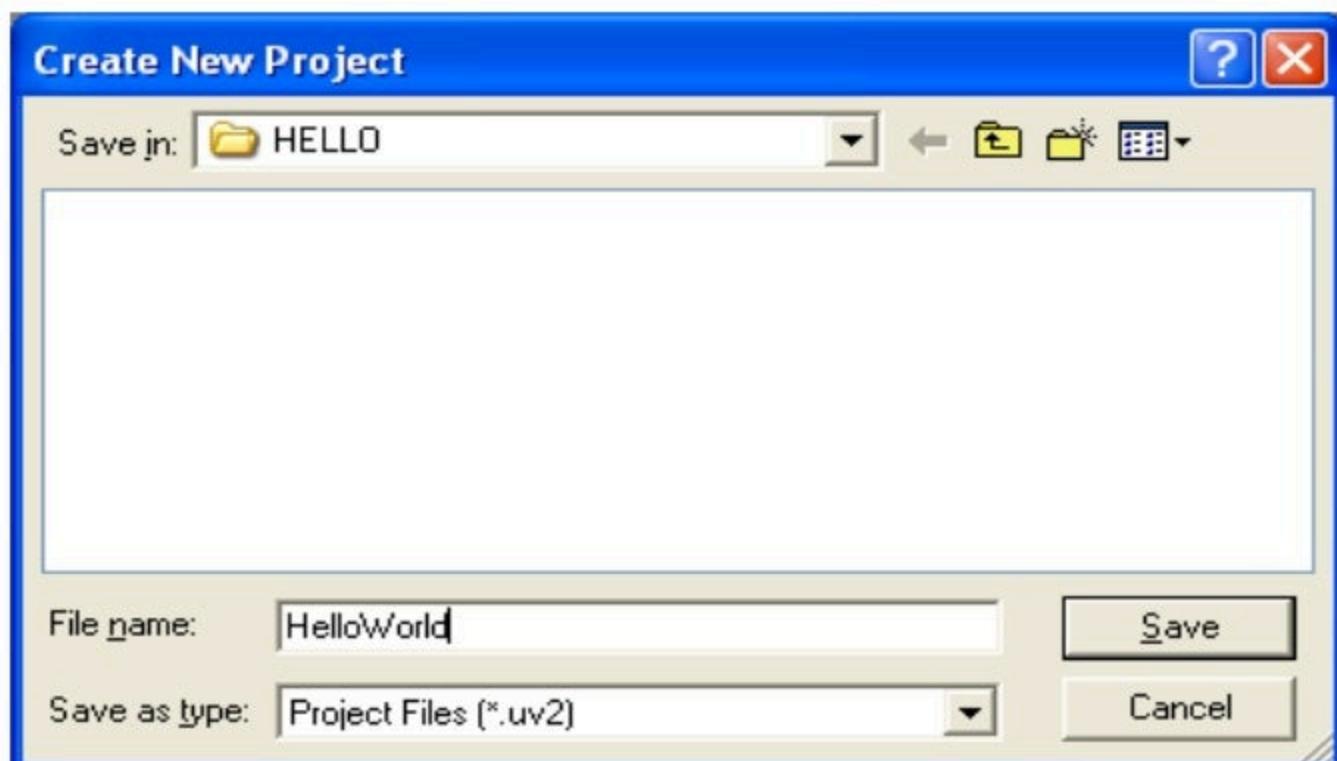
```

```
unsigned char RxChar(void)
{
RI = 0;
return(SBUF);
}
void TxStr(char *str)
{
while(*str)
{
TxChar(*str);
str++;
}
}
void TxNiblE(unsigned char ch)
{
if(ch < 10)
{
ch = ch + 48;
TxChar(ch);
}
else
{
ch= ch + 55;
TxChar(ch);
}
}
void TxHex(unsigned char ch)
{
unsigned char tmp;
tmp = (ch & 0xf0) >> 4;
TxNiblE(tmp);
tmp = ch & 0x0f;
TxNiblE(tmp);
}
void TxCrLf(void)
{
TxChar(13);
TxChar(10);
}
```

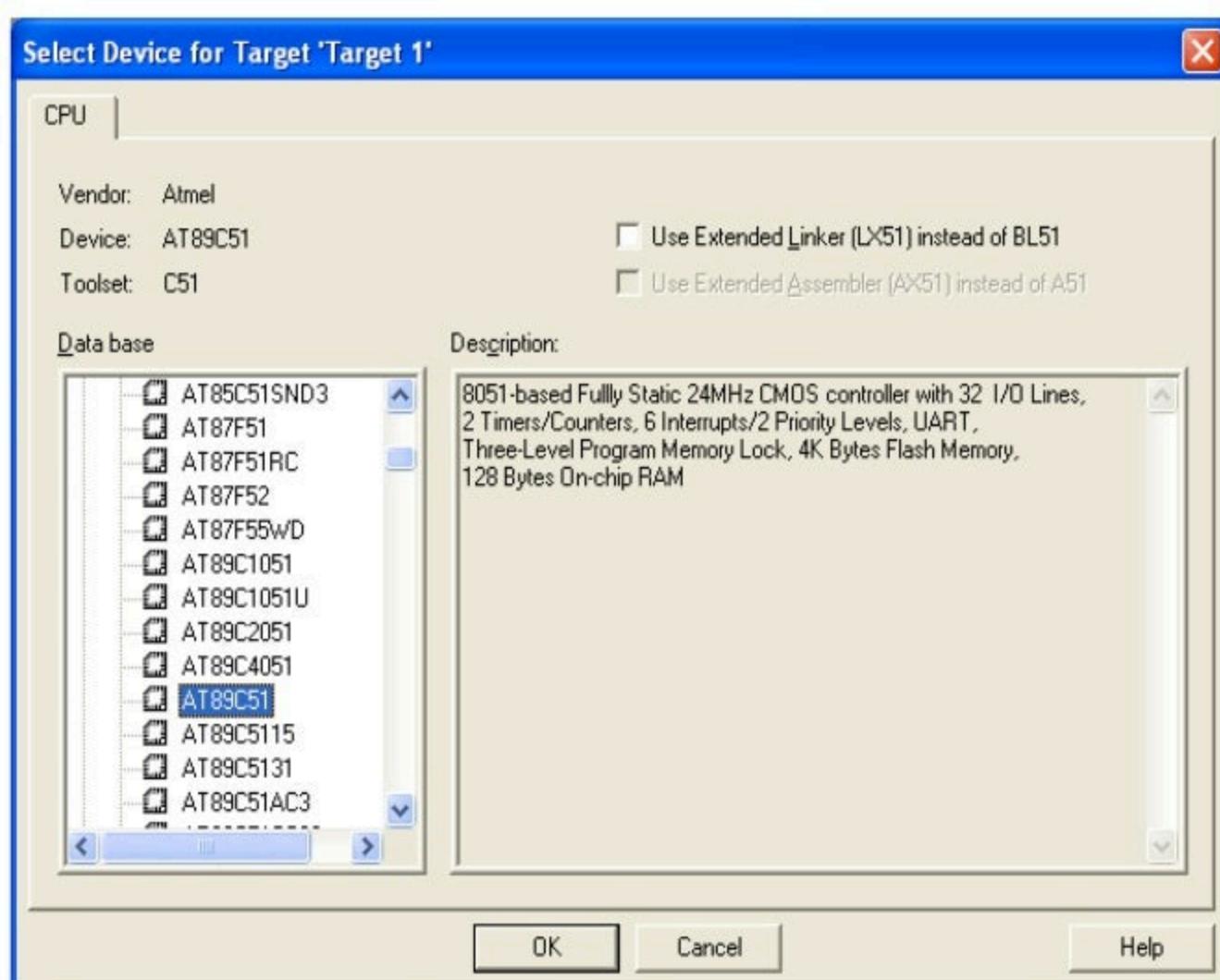
## PHỤ LỤC

### I. Hướng dẫn sử dụng phần mềm lập trình C cho họ vi điều khiển 8051 sử dụng phần mềm Keil c và phần mềm mô phỏng Proteus

**Bước 1:** Chọn menu Project → New Project. Nhập tên của project (HelloWorld)



**Bước 2:** Chọn vi điều khiển muốn lập trình (Chọn ATMEL → AT89C51)



**Bước 3:** Một cửa sổ hiện ra hỏi xem bạn có sử dụng một số đoạn code khởi động chuẩn cho 8051, hãy chọn No.



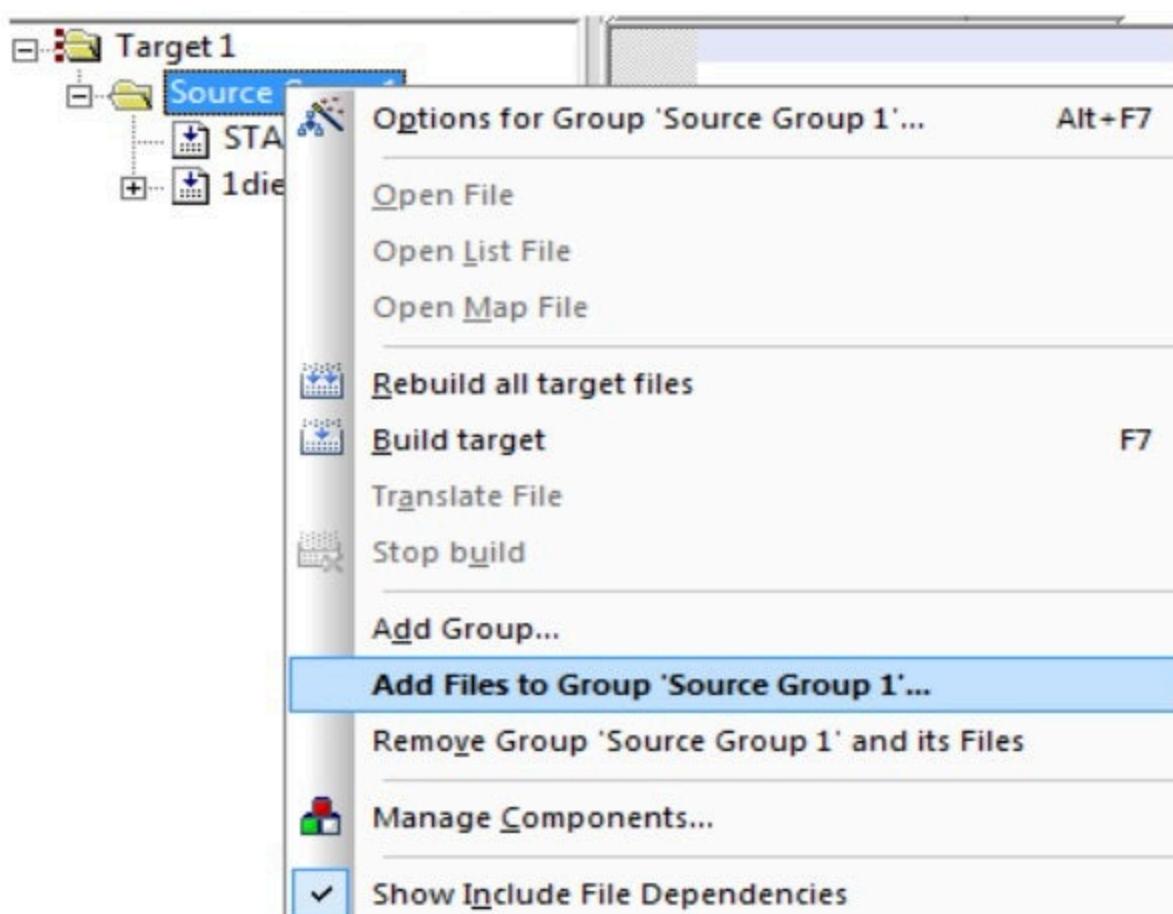
**Bước 4:** Tạo ra một file mã nguồn bằng cách chọn menu File → New, lưu file và đặt tên là Hello.C.

**Bước 5:** Viết mã cho file mã nguồn vừa tạo

```
#include <at89x51.h>
#define LED0 P1_0
#define sang 0
#define tat 1
-----
void delay(unsigned int ms)
{
    unsigned int i,j;
    for (i=0;i<ms;i++)
        for (j=0;j<120;j++);
}

}
main ()
{
    while(1)
    {
        LED0=sang;
        delay(500);
        LED0=tat;
        delay(500);
    }
}
```

**Bước 6:** Thêm file mã nguồn vừa tạo vào trong project

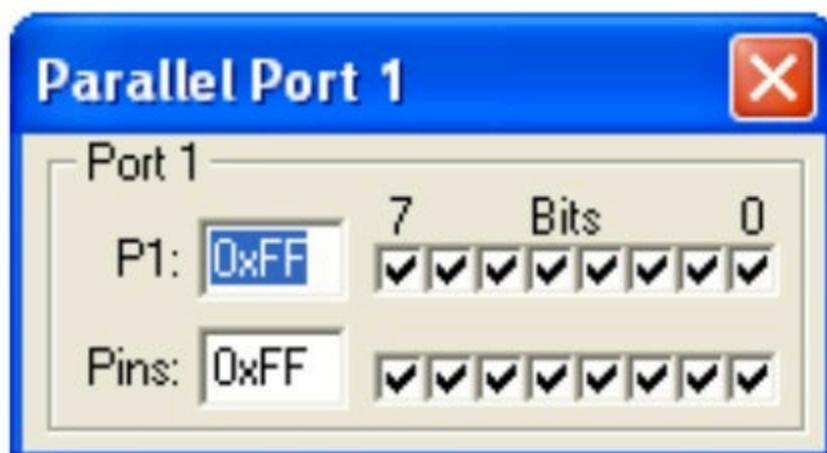


Chọn file Hello.c.

**Bước 7:** Biên dịch chương trình bằng cách bấm phím F7 hoặc Project → Build Target.

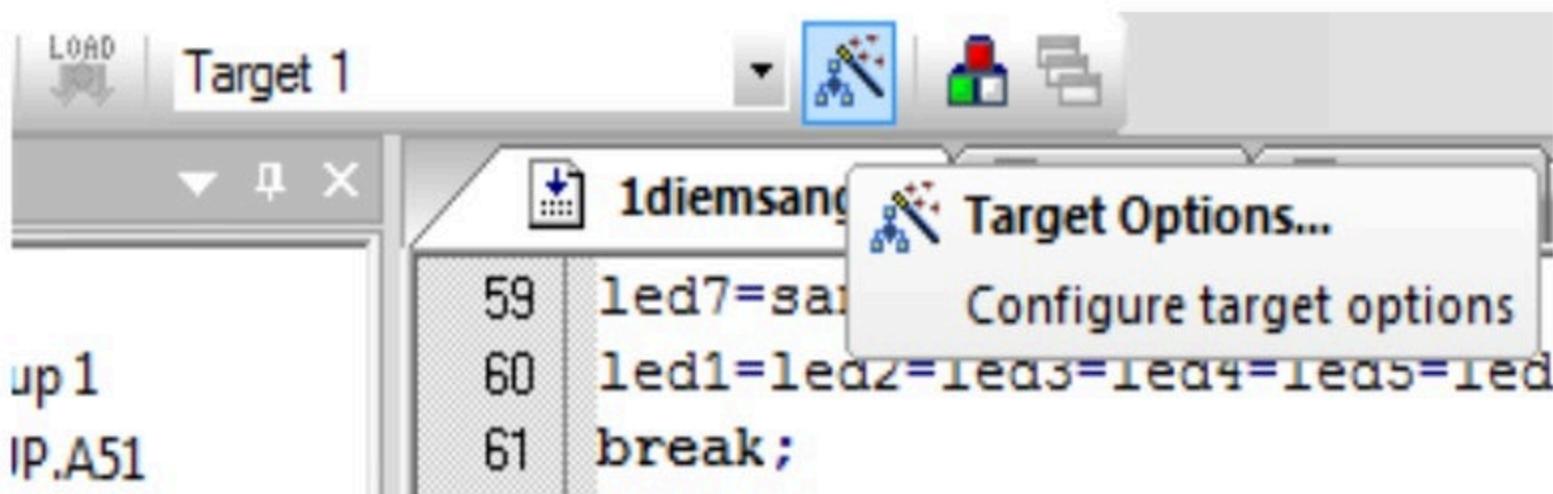
**Bước 8:** Debug → Start/Stop Debug Session hoặc bấm Ctrl + F5 để chuẩn bị gỡ lỗi cho chương trình,

**Bước 9:** Mở cửa sổ mô phỏng cổng 1 của vi điều khiển AT89C51 bằng cách chọn menu Peripherals → IO Port → Port 1.

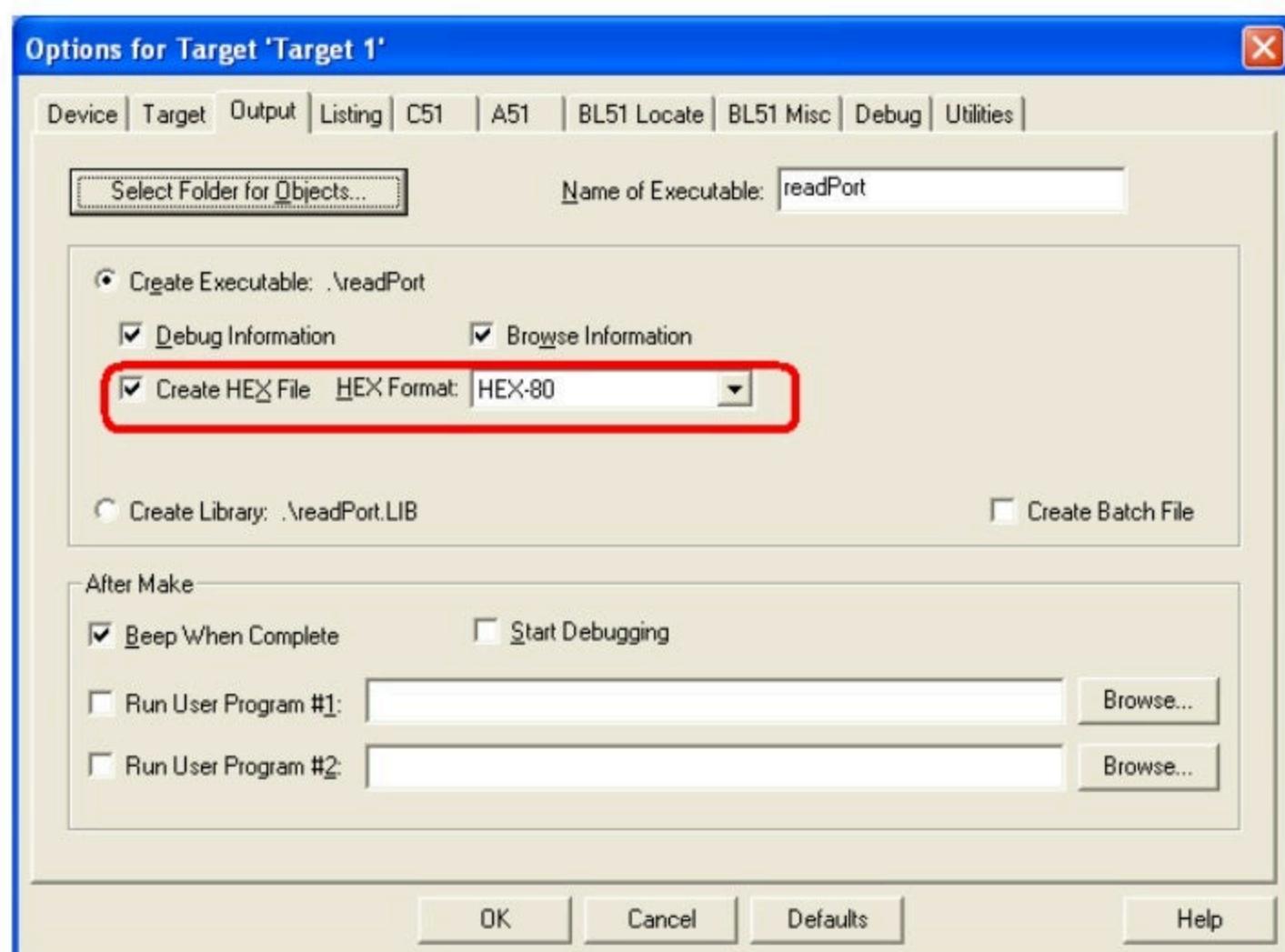


**Bước 10:** Tiến hành mô phỏng bằng cách bấm vào menu Debug → Run hoặc bấm F5.

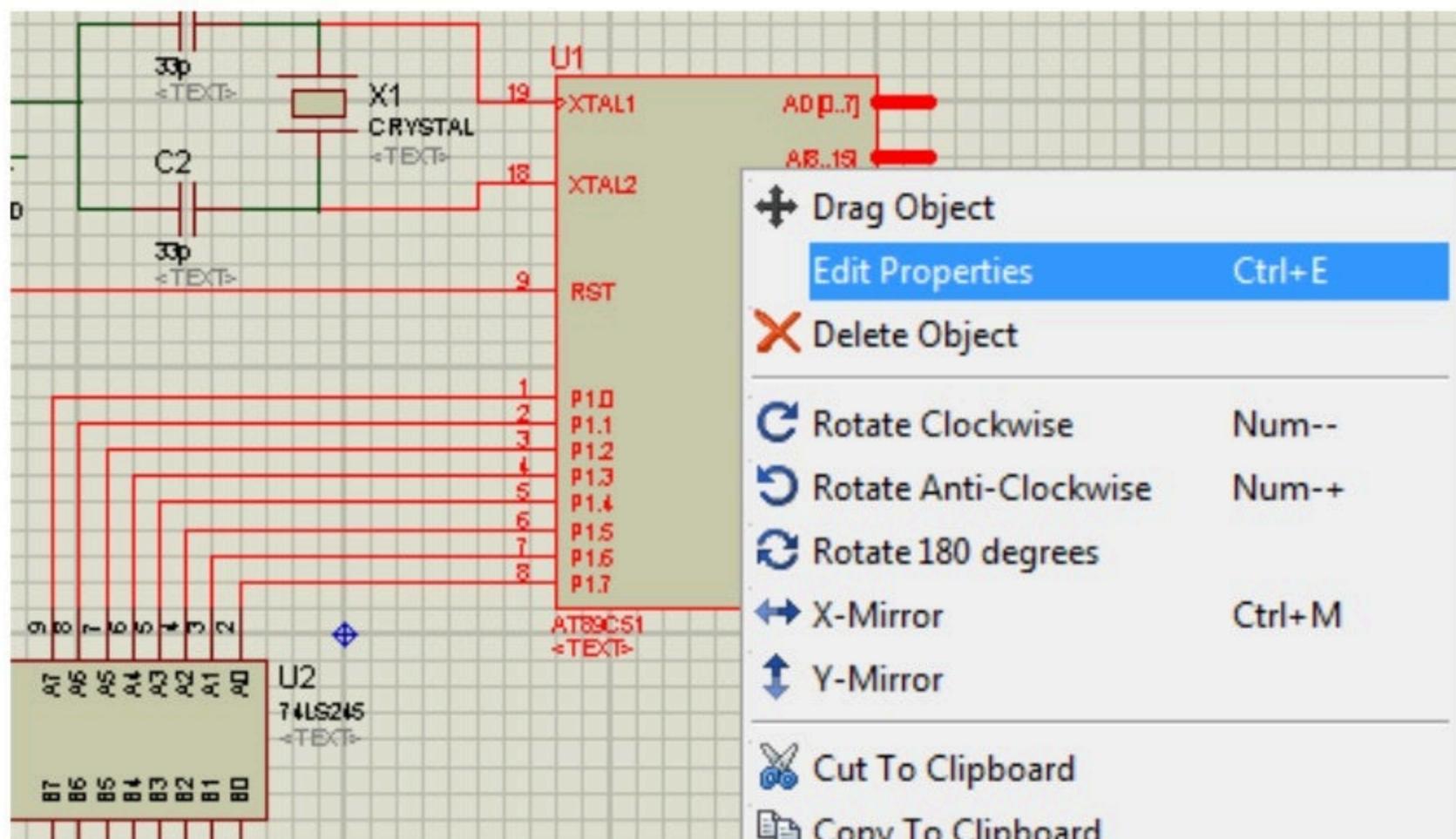
**Bước 11:** Tạo ra file .HEX và nạp vào vi điều khiển, chọn Icon Options for Target.



Chọn tab Output và tích vào tùy chọn để tạo ra .HEX file. Sau đó, nếu biên dịch chương trình và không có lỗi, một file .HEX sẽ được tạo ra (có tên mặc định trùng với tên file mã nguồn). File này sẽ được nạp vào bộ nhớ của vi điều khiển để thực thi chương trình.



**Bước 12:** Để nạp file này vào vi điều khiển, trong phần mềm Proteus, click chuột phải lên vi điều khiển, chọn Edit Properties.



Tiếp đến, một cửa sổ cho phép chỉnh sửa thuộc tính của vi điều khiển hiện lên, bạn tiến hành tìm tới đường dẫn file .HEX chứa mã nguồn muốn mô phỏng.

### Edit Component

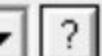
Component Reference: U1

 Hidden:

Component Value: AT89C51

 Hidden:

PCB Package: DIL40



Hide All

Program File: hello.hex



Hide All

Clock Frequency: 12MHz

Hide All

Advanced Properties:

Simulate Program Fetches ▾ No



Hide All

Other Properties:

- Exclude from Simulation
- Exclude from PCB Layout
- Edit all properties as text

- Attach hierarchy module
- Hide common pins

**OK****Help****Data****Hidden Pins****Cancel**

## II. Hướng lập trình tạo giao diện điều khiển từ máy tính sử dụng ngôn ngữ lập trình C#

Để thực hiện giao tiếp truyền nhận giữa một máy tính đa dụng (PC) với máy tính trên chip (MCU) thì công cụ C# được cài đặt trên máy tính phải được hỗ trợ NetFrameWork 2.0 để cung cấp đặc tính truyền nhận nối tiếp (Serial Communication). Đặc tính này trong NetFrameWork 2 có tên là: namespace “system.IO.Ports” bao gồm nhiều lớp trong đó quan trọng nhất là lớp “SerialPort”. Lớp này cung cấp việc đồng bộ và kiểm soát các sự kiện của I/O qua các chân của thiết bị cổng COM.

Các khả năng được hỗ trợ bởi “system.IO.Ports”:

- Khả năng truy xuất cổng nối tiếp trên PC.
- Giao tiếp với các thiết bị ngoại vi qua cổng nối tiếp. (Serial Port).

Đặc tính truyền dẫn: dùng giao diện RS 232 chuẩn DB9 để thực hiện giao tiếp ở chế độ Full-Duplex.

9 Pin Connector on a DTE device (PC connection)	
Male RS232 DB9	Pin Number
	Direction of signal:
	1
	Carrier Detect (CD)
	2
	Received Data (RD)
	3
	Transmitted Data (TD)
	4
	Data Terminal Ready (DTR)
	5
	Signal Ground Common reference voltage
	6
	Data Set Ready (DSR)
	7
	Request To Send (RTS)
	8
	Clear To Send (CTS)
	9
	Ring Indicator (RI) (from DCE) Incoming signal from a modem

- RTS & DTR là ngõ ra tín hiệu số, có thể được đặt bằng tay.
- DCD, DSR, CTS, and RI là những bit vào và chỉ có thể đọc.
- RX & TX hai chân này nhằm kiểm soát UART và không được đặt.
- Nguồn điện áp cổng là  $\pm 15V$  và ngõ ra ở 2 khoảng  $+5$  đến  $+15$  và  $-5$  đến  $-15$  và điện áp ngõ vào là 2 khoảng  $+3$  đến  $+15$  và  $-3$  đến  $-15$ . Điện áp ngõ vào ở khoảng  $-3$  đến  $+3V$  và ngõ ra ở khoảng  $-5$  đến  $+5V$  là vùng không xác định.

### Các bước lập trình phần mềm:

1. Xác định cơ chế giao tiếp qua RS 232, thông số của cổng COM:

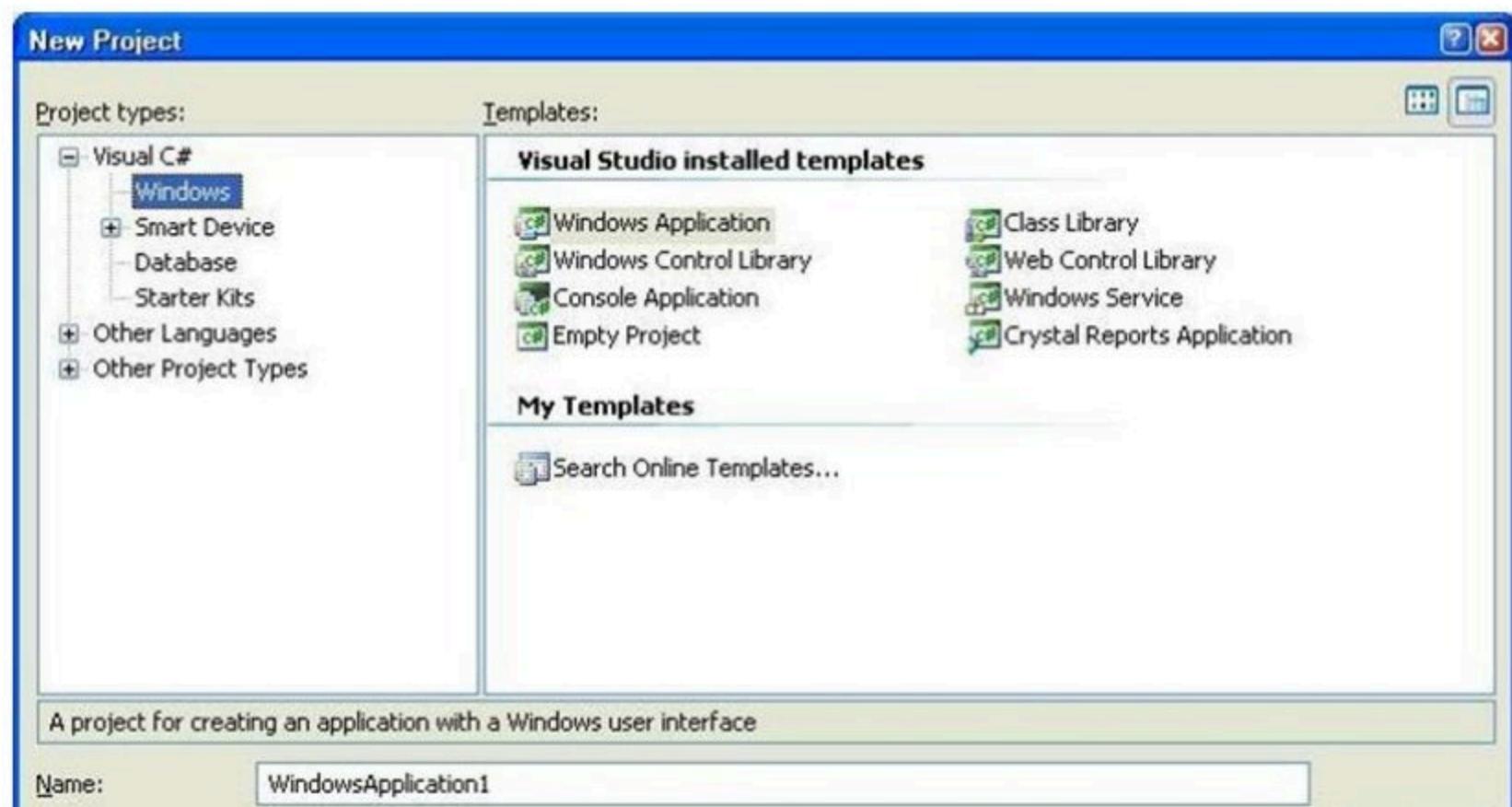
- Địa chỉ cổng (Port Name).
- Tốc độ baud (baudrate).
- Số bít trong một khung dữ liệu (data bits).
- Số bit dừng giữa các khung dữ liệu (stop bit).
- Bít kiểm tra chẵn lẻ (parity bit).

Đối với việc giao tiếp thì quan tâm đến giao thức truyền dữ liệu trong hướng dẫn này sử dụng chuẩn: 9600,8N1 (tức tốc độ baud 9600 bps, 8 bit data, no parity và 1 bit stop). Trong phần mềm có thể sử dụng được với cổng ảo (được hỗ trợ qua phần mềm **Virtual Com**) và cả cổng Com thực tế, nên có chế độ lấy thông tin của cổng Com và giao tiếp qua cổng nào thì do người dùng lựa chọn qua chế độ “Select port”.

2. Xây dựng giao diện bằng C# Windows Form trong bộ Visual Studio



## Chọn ứng dụng Windows Application



Xây dựng giao diện điều khiển có dạng như sau:

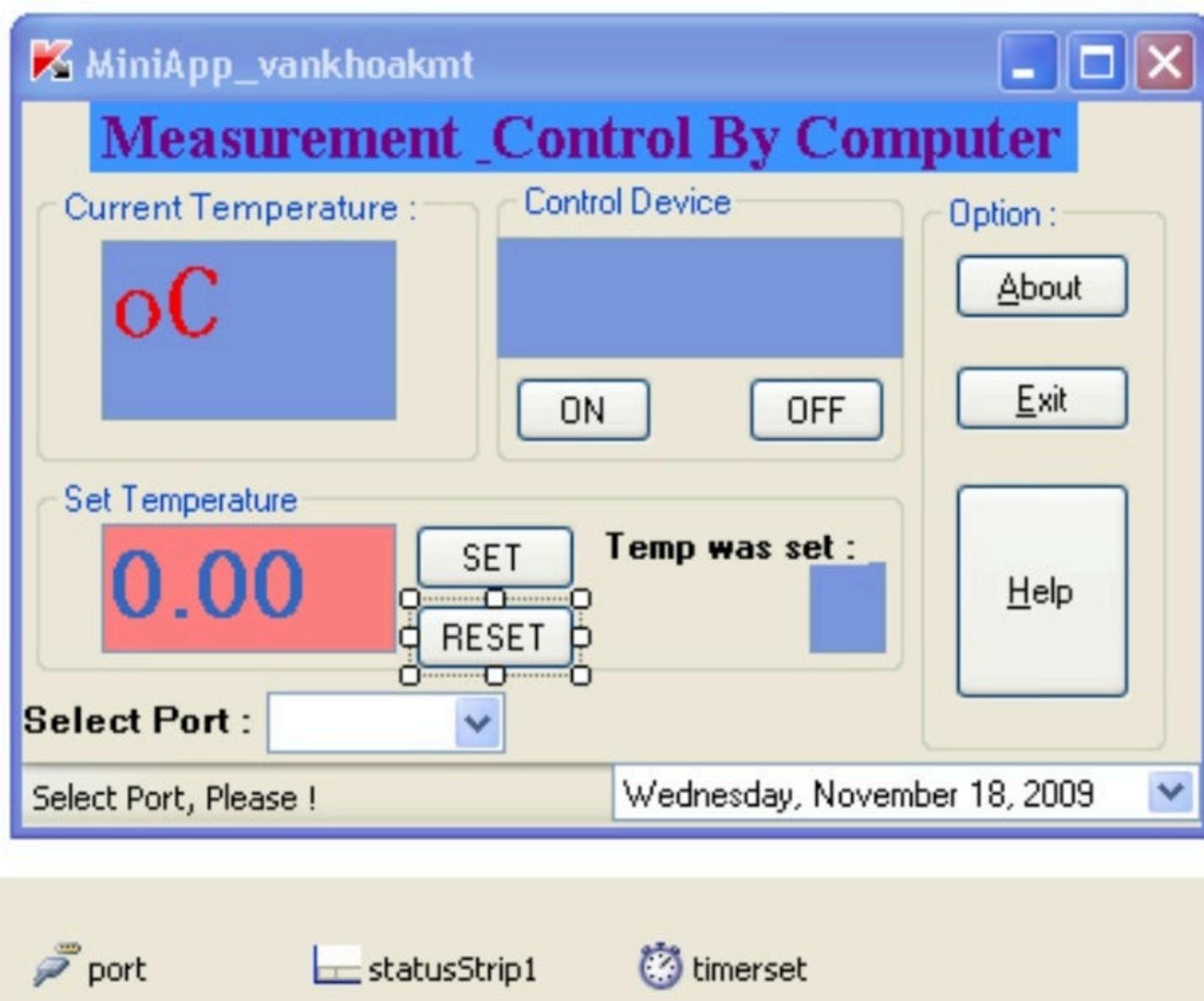


Trong đó có sử dụng chức năng Serial Port trong Toolbox để tạo thuộc tính giao tiếp cổng COM trong chương trình. Chọn các chế độ mặc định cho cổng COM như: tốc độ baud 9600 bps, 8-bit data, không bit parity và 1 bit stop ở phần properties, sau khi thiết kế giao diện như sau:

port System.IO.Ports.SerialPort	
	A Z ↓
(Name)	port
GenerateMember	True
Modifiers	Private
Misc	
BaudRate	9600
DataBits	8
DiscardNull	False
DtrEnable	False
Handshake	None
Parity	None

Hoặc cấu hình trực tiếp sử dụng lệnh như sau;

```
serialPort1.PortName="COM1";
serialPort1.BaudRate=9600;
serialPort1.DataBits=8;
serialPort1.Parity=Parity.None;
serialPort1.StopBits= StopBits.One;
```



Tạo giao diện cho phần About và Help:

### 3. Viết chương trình giao tiếp

- Đưa namespace vào chương trình.

```
using System.IO.Ports;
```

- Khởi tạo các cổng com có trong hệ thống. trong hệ thống có thể sử dụng cổng com ảo hay com thật của thiết bị nên phải quét tất cả các cổng com có trong hệ thống hiện tại.

#### Một số hàm cần chú ý như

- SerialPort.IsOpen(): Trả lại trạng thái của cổng là đang đóng hay mở.
- SerialPort.Open(): Mở cổng với thông số đã cài đặt.
- SerialPort.Close(): Đóng cổng.

```
public Form1()
{
    InitializeComponent();
    string[] ports =
SerialPort.GetPortNames();
// quét tất cả các port được kết nối vào PC.

    foreach (string port in ports)
// đưa tất cả các port vào mục chọn
combobox.
    {
        cmbBoxPort.Items.Add(port);
    }

}
```

Do các thông số của cổng COM đã được thiết lập mặc định nên chỉ cần chọn cổng (ở các địa chỉ như COM1: 3F8) để kết nối.

```
private void
cmbBoxPort_SelectedIndexChanged(object sender,
EventArgs e)
{
    if (port.IsOpen) port.Close();
    port.PortName =
cmbBoxPort.SelectedItem.ToString();
// lưu nhận tên port được chọn
```

```

        stsStatusPort.Text = "You choose :" +
port.PortName + ": 9600,8N1";
        MessageBox.Show ("You choose : " +
port.PortName + " ? ", " Note :");
        try
        {
            port.Open(); // thực
hiện mở port khi được chọn
        }

        catch
// nếu không thực hiện mở port được thì đưa ra
thông báo lỗi
        {
            MessageBox.Show("Serial port "
+ port.PortName + " cannot be opened!",
"Warning", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            cmbBoxPort.SelectedText = "";
            stsStatusPort.Text = "Select
serial port!";
        }
    }
}

```

Sau khi đã chọn cổng com kết nối. Thực hiện giao tiếp lấy thông tin từ thiết bị truyền lên và đưa thông tin xuống thiết bị (Hướng dẫn ở đây nhằm lấy thông tin 1 byte từ vi điều khiển đưa lên và truyền 1 byte xuống vi điều khiển).

Để thực hiện được việc đọc thông tin từ vi điều khiển đưa lên có 2 cách sau:

- Thường xuyên kiểm tra cổng xem dữ liệu có được truyền đến không (để kiểm tra thì dùng 1 Timer định thời gian).
- Phải bắt được sự kiện dữ liệu đến, khi dữ liệu đến ta gọi hàm xử lý.

Trong ứng dụng dùng phương pháp thứ hai là lấy sự kiện dữ liệu được truyền đến bằng “*SerialDataReceivedEvent*. ” Khi có sự kiện này ta sẽ đọc và xử lý dữ liệu từ bộ đệm cổng.

```
this.port.DataReceived += new  
System.IO.Ports.SerialDataReceivedEventHandler(th  
is.port_DataReceived_1);
```

Trong đó, *port\_DataReceived\_1* là hàm sẽ được gọi đến khi có sự kiện dữ liệu được nhận.

Một số hàm để đọc dữ liệu từ bộ đệm:

- *SerialPort.ReadExisting()*: Đọc một string từ bộ đệm cổng.
- *SerialPort.ReadChar()*: Đọc một giá trị kiểu char từ bộ đệm cổng.
- *SerialPort.ReadByte()*: Đọc một giá trị kiểu Byte từ bộ đệm cổng.

```
// sự kiện nhận dữ liệu trong buffer và lấy  
thông tin dữ liệu .  
public void port_DataReceived_1(object  
sender, SerialDataReceivedEventArgs e)  
{  
    InputData = port.ReadExisting();  
    // thực hiện đọc một string từ bộ đệm cổng  
    if (InputData != String.Empty)  
    {  
        SetText(InputData); //  
        // gọi hàm xuất dữ liệu .  
    }  
  
    }  
    public void SetText(string text)  
    {  
  
        if (this.temp.InvokeRequired)  
        {  
            SetTextCallback d = new  
SetTextCallback(SetText);  
            this.Invoke(d, new object[] {  
text});  
        }  
        else  
        {
```

```

        this.temp.Text =
ConvertToHex(text);
// mã được xuất ra là mã ASCII
}

}

public string ConvertToHex(string
asciiString)
// thực hiện chuyển ASCII sang Hex để hiển
thị
{
    string hex = "";
    foreach (char c in asciiString)
    {
        int tmp = c;
        hex +=

String.Format("{0:x2}",
(uint)System.Convert.ToInt32(tmp.ToString()))
);
    }
    return hex;
}

```

Kết quả sau khi đưa về chuỗi hex sẽ hiển thị thông tin nhiệt độ đo được.

Sử dụng hàm SerialPort.WriteLine, sự kiện truyền này sẽ được kích hoạt khi có nút bấm của người dùng.

```

private void btnon_Click(object sender,
EventArgs e)
{
    port.WriteLine("a");
    stsdevice.Show();
    stsdevice.Text = "Device is opened";
}

//các hàm để xử lý các sự kiện nút bấm
private void exit_Click(object sender,
EventArgs e)
{
    MessageBox.Show("Do you want to

```

```

Close the MiniApp ? "," Exit ");
    port.Close();
    Close();

}

About a = new About();
Help b = new Help();

private void about_Click(object sender,
EventArgs e)
{
    a.ShowDialog();
}

private void Info_Click(object sender,
EventArgs e)
{
    b.ShowDialog();
}

```

Xử lý việc truyền dữ liệu khi có sự kiện bấm:

```

private void btnon_Click(object
sender, EventArgs e)
{

    port.WriteLine("a");
    stsdevice.Show();
    stsdevice.Text = "Device is opened
";
}

private void btnoff_Click(object
sender, EventArgs e)
{
    port.WriteLine("b");
    stsdevice.Show();
    stsdevice.Text = "Device is
interrupted ";
}

```

Thiết lập chế độ đặt nhiệt độ tự động (lưu ý cần khai báo toàn cục: public int set).

```
public void btnset_Click(object sender, EventArgs e)
{
    timerset.Enabled = true;
    timerset.Start();
}

private void timerset_Tick(object sender, EventArgs e)
{
    set = int.Parse(settemp.Text);
    if (set > 64)
    {
        timerset.Enabled = false;
        MessageBox.Show("Temperature < 64", "Report !!!");
        return;
    }
    setsts.Enabled = true;
    setsts.Text = settemp.Text;
    int b = int.Parse(temp.Text);
    if (set == b)
    {
        timerset.Enabled = false;
        MessageBox.Show("Temperature is balance", "Report !!!");
    }

    if (settemp.Text.Trim().Length == 0)
        return;
}

private void btnreset_Click(object sender, EventArgs e)
{
    set = 0;
    setsts.Clear();
}
```

# TÀI LIỆU THAM KHẢO

## Tiếng Việt

1. Ngô Diên Tập, “Vi điều khiển với Lập trình C”, NXB Khoa học Kỹ thuật, 2006.
2. Tống Văn On, “Hệ vi điều khiển 8051”, NXB Lao động - Xã hội, 2007.
3. Hồ Trung Mỹ, “Vi xử lý”, NXB ĐHQG-HCM, 2013.

## Tiếng Anh

1. Thomas W. Schultz, “C and the 8051 Hardware, Modular Programming & Multitasking”, Prentice Hall, 1997.
2. Mazidi and McKinlay, “The 8051 Microcontroller and Embedded Systems”, Newnes, 2003.
3. Dogan Ibrahim “Microcontroller Projects in C for the 805”, Newnes, 2000.
4. David Calcutt, Frederick Cowan, Hassan Parchizadeh, “8051 Microcontroller - An Applications Based Introduction”, Newnes, 2004.
5. Salvador Pinillos Gimenez “8051 Microcontrollers\_ Fundamental Concepts, Hardware, Software and Applications in Electronics”, Springer, 2018.



## CHÍNH SÁCH CHẤT LƯỢNG

*Không ngừng nâng cao chất lượng dạy, học, nghiên cứu khoa học  
và phục vụ cộng đồng nhằm mang đến cho người học những điều kiện tốt nhất  
để phát triển toàn diện các năng lực đáp ứng nhu cầu phát triển và hội nhập quốc tế.*

ISBN: 978-604-73-7791-6  
9 NXB ĐHQG-HCM  
9 786047 377916