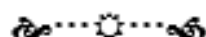


ĐẠI HỌC QUỐC GIA
ĐẠI HỌC BÁCH KHOA TP HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ



MÔN: VI XỬ LÝ (EE2039)
BÀI TẬP LỚN
MÁY TÍNH SỐ HỌC ĐƠN GIẢN

LỚP P01--- NHÓM 01 --- HK212

NGÀY NỘP: 11/5/2022

Giảng viên hướng dẫn: LƯU PHÚ

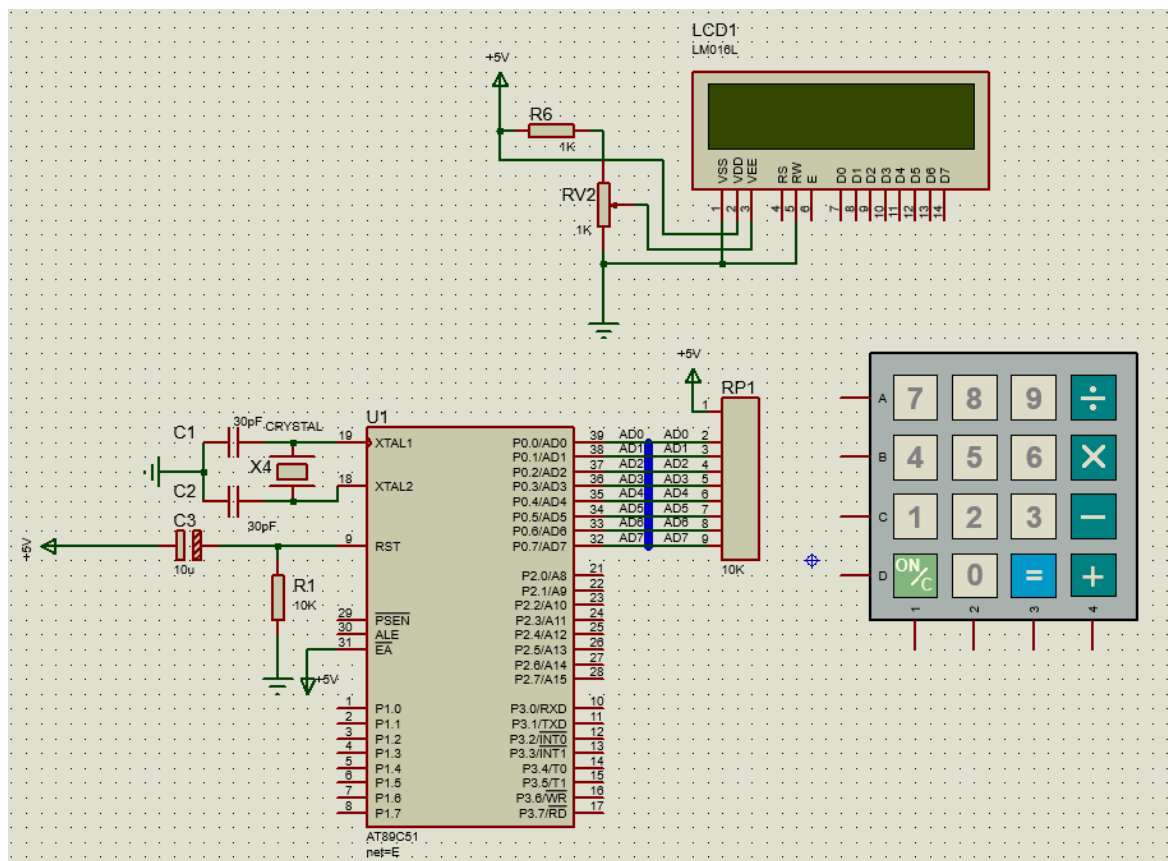
Sinh viên thực hiện	MSSV	Chấm điểm
Nguyễn Thành Nhân	1911757	100%
Trần Thị Linh	1913963	100%
Huỳnh Đăng Khoa	1911398	100%
Nguyễn Phương Nam	1910356	100%

Thành phố Hồ Chí Minh – 2021

ĐỀ BÀI: Máy tính số học đơn giản

- Thực hiện 4 phép tính cộng, trừ, nhân và chia.
- Giới hạn toán hạng là số không dấu từ 0 – 999.
- Nhập số liệu và phép tính từ bàn phím.
- Hiển thị phép tính và kết quả ra LCD ký tự 16x2.

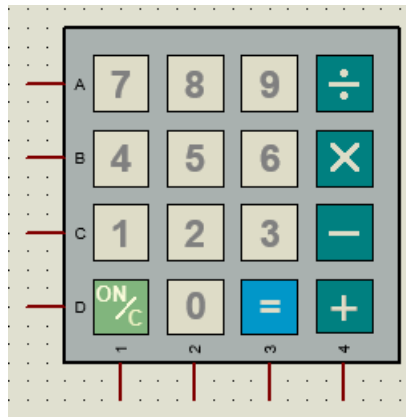
A. VẼ VÀ MÔ TẢ SƠ ĐỒ KHỐI PHẦN CỨNG:



Hình 1. Sơ đồ khối phần cứng máy tính số học đơn giản

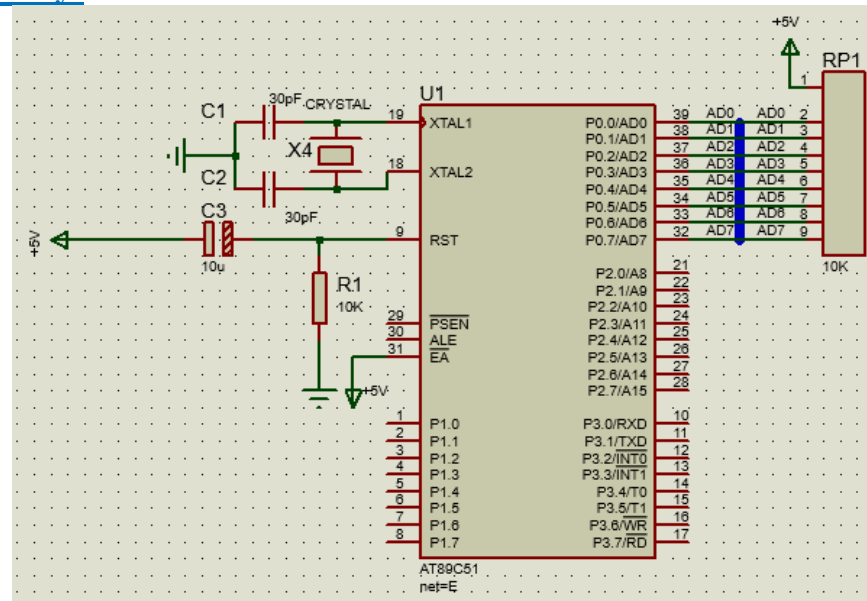
- Sơ đồ khối phần cứng máy tính số học đơn giản được sắp xếp như hình 1 gồm các khối:

+ Khối ma trận phím để nhập dữ liệu:

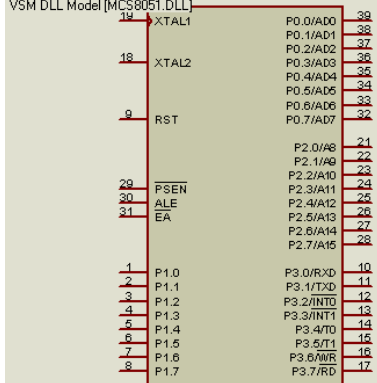
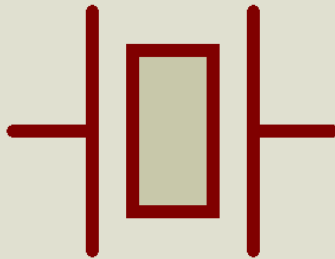








STT	Tên phần tử	Số lượng	Thông số	Hình ảnh
1	KEYPAD-SMALLCALC	1		

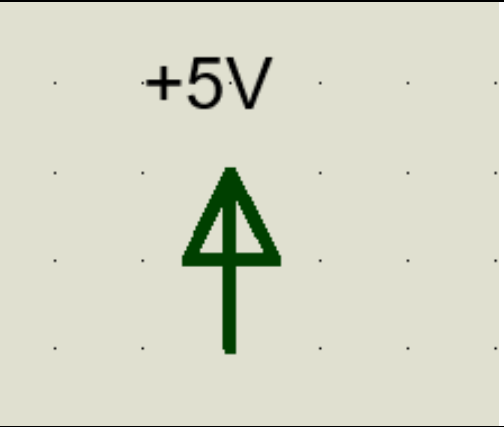
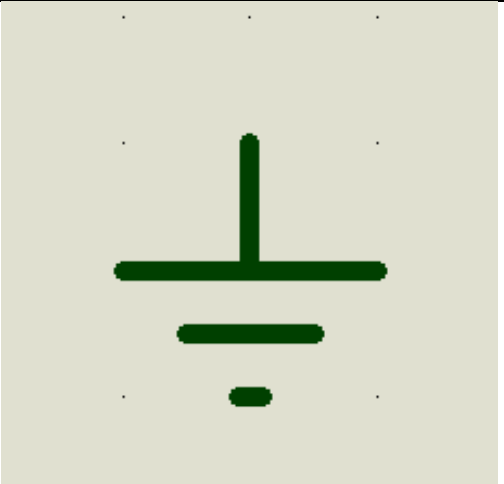
+ Khối xử lý:



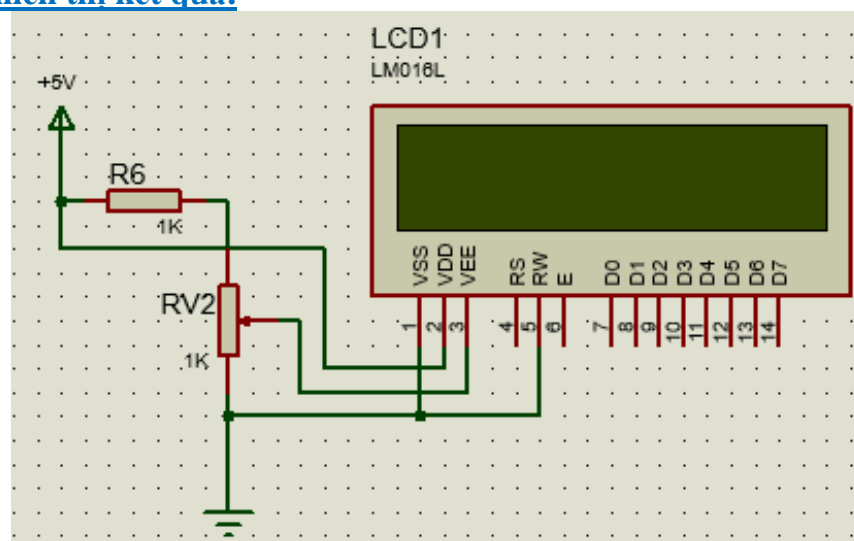
STT	Tên phần tử	Số lượng	Thông số	Hình ảnh
-----	-------------	----------	----------	----------

1	AT89C51	1	Clock frequency: 11.059Mhz	<p>VSM DLL Model [MCS8051.DLL]</p> 
2	CRYSTAL	1	Frequency: 11.059Mhz	<p>Schematic Model [CRYSTAL.MDF]</p> 
3	TỤ GỐM	2	30pF	<p>Analogue Primitive [CAPACITOR]</p> 
4	TỤ HÓA	1	10uF	<p>Analogue Primitive [CAPACITOR]</p> 

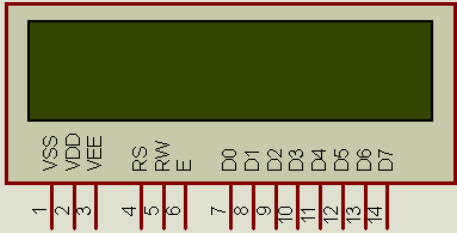

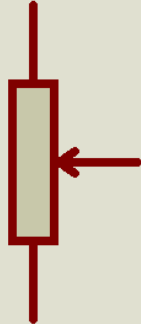

5	Điện trở	1	10K	<p>Analogue Primitive [RESISTOR]</p> 
6	Điện trở	1	6K	<p>Analogue Primitive [RESISTOR]</p> 
7	Respack - 8	1	10K	<p>Schematic Model [RESPACK8]</p> 
8	Biến trở	1	1K	<p>Schematic Model [POT_UNI]</p> 

9	Nguồn	3	+5V	
10	GND	2		

+ Khởi hiển thị kết quả:

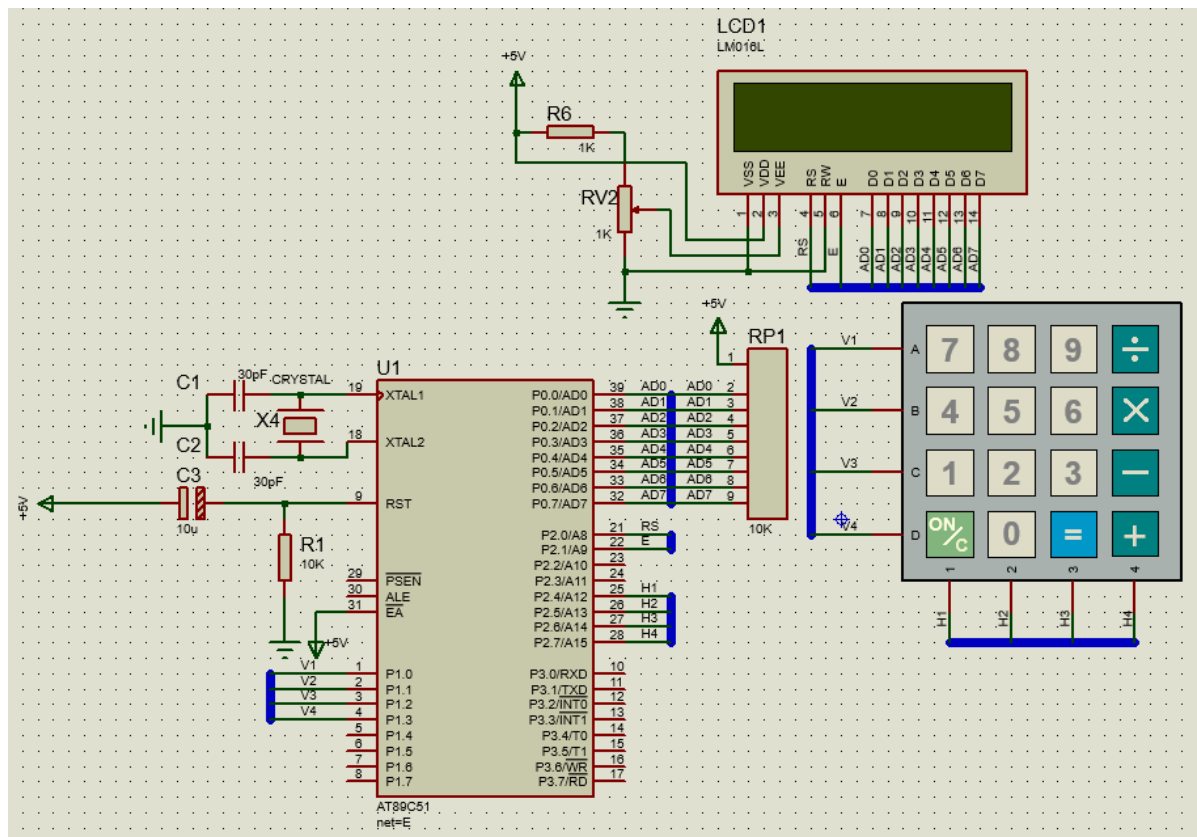


STT	Tên phần tử	Số lượng	Thông số	Hình ảnh
-----	-------------	----------	----------	----------

1	Màn hình LCD 16x2 LM016L	1		<p>VSM DLL Model [LCDALPHA]</p> 
2	Điện trở	1	1K	<p>Analogue Primitive [RESISTOR]</p> 
3	Biến trở	1	1K	<p>Schematic Model [POT_LIN]</p> 
4	Nguồn	1	+5V	

5	GND	1		
---	-----	---	--	--

B. THIẾT KẾ CHI TIẾT PHẦN CỨNG:



Hình 2. Sơ đồ kết nối hoàn chỉnh phần cứng máy tính số học đơn giản

I. Sơ đồ kết nối:

- Đối với ma trận phím thì ta có:

+ 4 chân hàng của ma trận phím là A, B, C, D lần lượt kết nối với P1.0, P1.1 và P1.2 của Port1 của MCU89C51.

+ 4 chân cột của ma trận phím là 1, 2, 3, 4 lần lượt kết nối với P2.4, P2.5 và P2.6 của Port2 của MCU89C51.

⇒ Port2 là output và Port1 là input.

⇒ Ma trận phím có 8 chân nhưng ta không kết nối 8 chân ấy với cùng 1 Port để tránh hiện tượng Wire-and.

- Đối với MCU8051 thì ta kết nối như những gì đã học ở những chương trước như kết nối mạch Auto-Reset, mạch tạo xung,...
- Đối với LCD16x2:
 - + Chân RS kết nối với chân P2.0 của MCU8051.
 - + Chân RW kết nối GND vì ta chỉ ghi data ra LCD.
 - + Chân E kết nối với chân P2.1 của MCU8051.
 - + Các chân dữ liệu D0 → D7 lần lượt kết nối với đường dữ liệu AD0 → AD7 của MCU8051.

II. Mô tả hoạt động:

- MCU8051 quét liên tục ma trận phím, check nhấn/nhả 50 lần để chống rung rồi sau đó hiển thị ký tự được nhấn lên LCD16x2. Chương trình ghi nhận phép toán (có nhận biết đúng/sai cú pháp), nếu người dùng nhấn dấu “=” thì MCU sẽ tính toán và đưa ra kết quả lên LCD nếu đúng cú pháp. Ngược lại, nếu người dùng nhập sai cú pháp hoặc nằm ngoài phạm vi giới hạn đề bài (>999) thì LCD sẽ in dòng chữ báo lỗi. Chương trình tiếp tục quét phím để thực hiện phép toán mới.

C. THIẾT KẾ CHI TIẾT PHẦN MỀM: (Viết bằng trình hợp ngữ)

I. Các lưu đồ giải thuật:

1.1 Khởi động, quét phím

Đầu tiên, ta sẽ định nghĩa cho trình biên dịch một số chỉ thị để phục vụ cho việc giao tiếp với LCD, với KEYPAD ở phần cứng. Hệ thống các cờ phục vụ quá trình xét điều kiện, nhận biết lỗi, hiển thị kết quả,... sẽ được trình bày xuyên suốt ở phần sau.

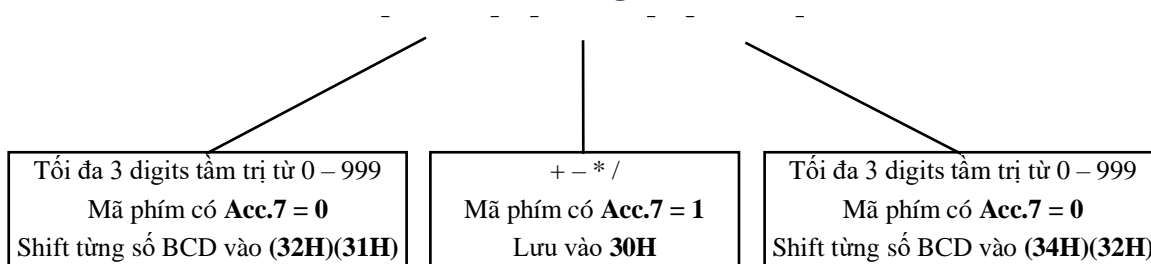
Tên	Chỉ thị	Địa chỉ	Ghi chú
DBUS	EQU	P0	Giao tiếp LCD 16x2
RS	BIT	P2.0	
RW	BIT	P2.2	
E	BIT	P2.1	
ROW	EQU	P1	Hàng ma trận phím
COLUMN	EQU	P2	Cột ma trận phím
FLAG_0	BIT	00H	Cờ toán tử, =1 khi nhập vào toán tử
FLAG_1	BIT	01H	Cờ khối 1
FLAG_2	BIT	02H	Cờ khối 2
FLAG_3	BIT	03H	Cờ tràn khối 1
FLAG_4	BIT	04H	Cờ tràn khối 1
FLAG_5	BIT	05H	Cờ ERROR
FLAG_6	BIT	06H	
FLAG_7	BIT	07H	Cờ ANSWER
FLAG_8	BIT	08H	Cờ in ra số 0

FLAG_9	BIT	09H	Cờ NEGATIVE ANSWER
--------	-----	-----	--------------------

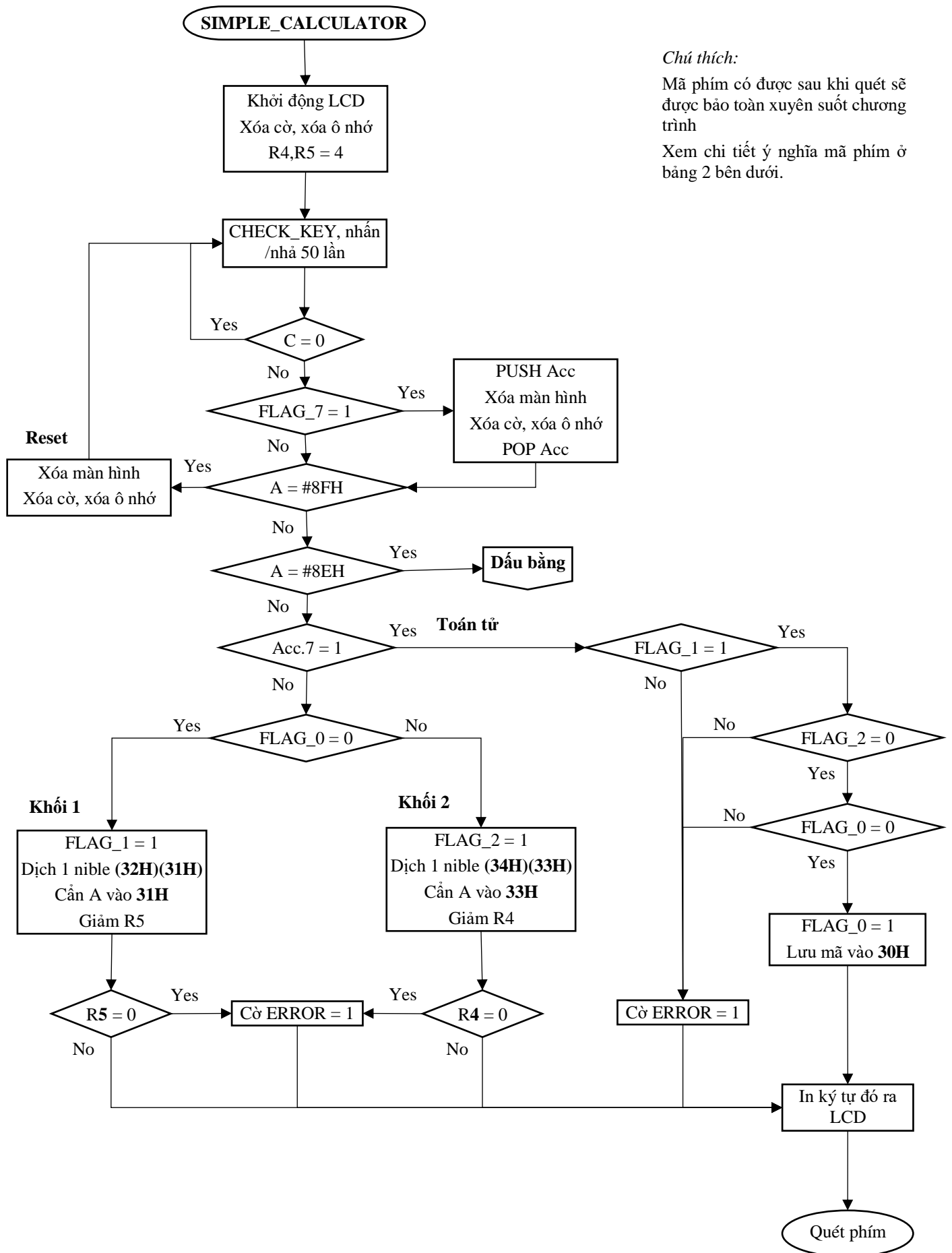
Dưới đây là định dạng cơ bản khi người dùng nhập một phép toán vào máy tính, lưu ý là chỉ nhập một và chỉ một phép tính mà thôi. Nếu nhập nhiều hơn một phép tính, máy sẽ báo lỗi.

Phép tính bao gồm toán hạng thứ nhất (ta tạm gọi là khối 1), với tầm trị từ 0 – 999. Một toán tử trong số 4 toán tử $+ - * /$. Toán hạng thứ hai (ta tạm gọi là khối 2), với tầm trị cũng từ 0 – 999.

Bảng 1



SƠ ĐỒ KHỐI TỔNG QUÁT CHO TOÀN BỘ CHƯƠNG TRÌNH

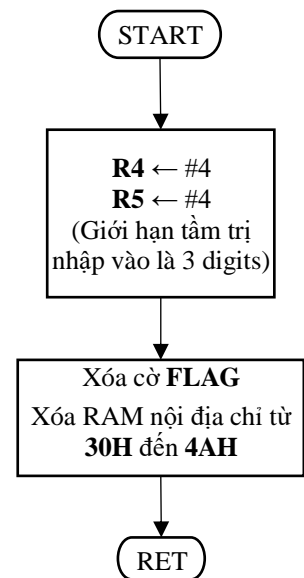


Chú thích:

Mã phím có được sau khi quét sẽ được bảo toàn xuyên suốt chương trình

Xem chi tiết ý nghĩa mã phím ở bảng 2 bên dưới.

Khi mới vừa khởi động, chương trình sẽ đẩy STACK lên vùng cao, gọi chương trình con INTIAL để khởi động LCD, cài đặt mode giao tiếp mà mình muốn. Về trình con INTIAL ta sẽ không đề cập đến vì đã có mẫu sẵn trong giáo trình. Điều đặc biệt đáng quan tâm chính là trình con START sẽ đặt R4, R5 = 4 mục đích để đếm số digits của các toán hạng để đảm bảo toán hạng được nhập vào có tầm trị từ 0 – 999. Các cờ như đã định nghĩa ở bảng 1 sẽ được xóa, các ô nhớ RAM nội cũng sẽ được xóa để phục vụ quá trình tính toán.



Khởi gọi chương trình con CHECK_KEY nhân /nhả kiểm tra 50 lần. Nếu thực sự có phím nhấn thì C = 1 và A chứa mã phím đã nhấn là một trong số những mã đã liệt kê ở bảng dưới đây.

Mã phím	Ý nghĩa
00H	0
01H	1
02H	2
03H	3
04H	4
05H	5
06H	6
07H	7
08H	8
09H	9
8AH	+
8BH	-
8CH	*
8DH	/
8EH	=
8FH	Reset

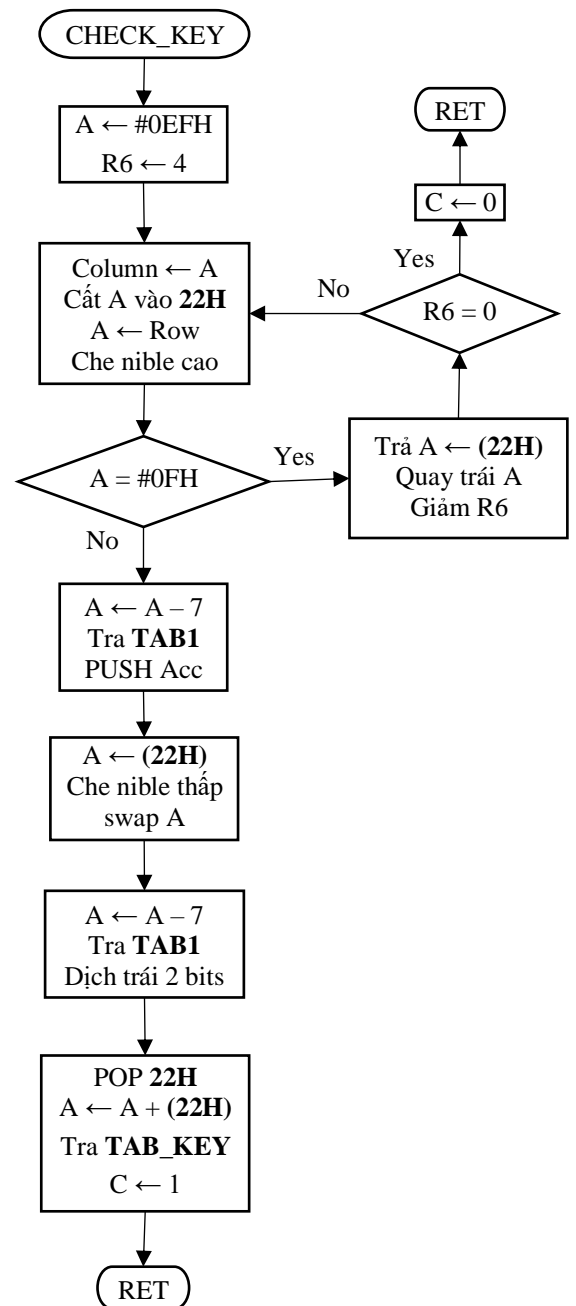
Bảng 2

CHECK_KEY: quét liên tục cột của ma trận phím bằng bit 0 (VD 1101 là quét cột 2), đọc hàng của ma trận phím.

– Nếu có phím nhấn thì sẽ có ít nhất một bit hàng sẽ ở mức 0. Chương trình sẽ nhận biết phím nào của KEYPAD đã nhấn nhờ việc tra các bảng tra. Cuối chương trình **C = 1** và **A = mã phím**.

– Nếu không có phím nào được nhấn thì **C = 0** rồi kết thúc chương trình con.

Đến đây ta cần lưu ý rằng nếu ta đồng thời nhấn nhiều phím (ở cùng cột) thì chương trình sẽ đọc sai.



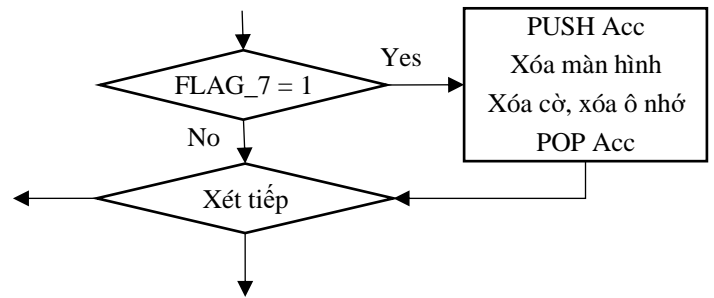
TAB1: DB 03H, 0FFH, 0FFH, 0FFH,
DB 02H, 0FFH, 01H, 00H

TAB_KEY:

DB 07H, 04H, 01H, 8FH, 08H, 05H,
DB 02H, 00H, 09H, 06H, 03H, 8EH,
DB 8DH, 8CH, 8BH, 8AH

1.2 Xét mã phím

Trước khi xét mã phím được nhấn, chương trình chính sẽ kiểm tra cờ Answer, nếu cờ ở mức 1 thì nghĩa là người dùng đã vừa thực hiện xong một phép tính rồi và chương trình đã in ra kết quả (trường hợp nhập sai cú pháp, LCD báo lỗi thì cuối cùng cờ Answer cũng sẽ được đưa lên 1 rồi tiếp tục quét phím).



Khi đó phím chương trình chính sẽ hiểu phím vừa được nhấn là phép toán mới. Khi đó nó sẽ cất mã phím, xóa màn hình, xóa cờ để thực hiện lại như vừa mới được reset, bắt đầu quá trình quét phím.

Nếu mã phím là Reset, chương trình xóa màn hình LCD rồi quay trở lại START để xóa cờ, xóa ô nhớ, nạp R4, R5.

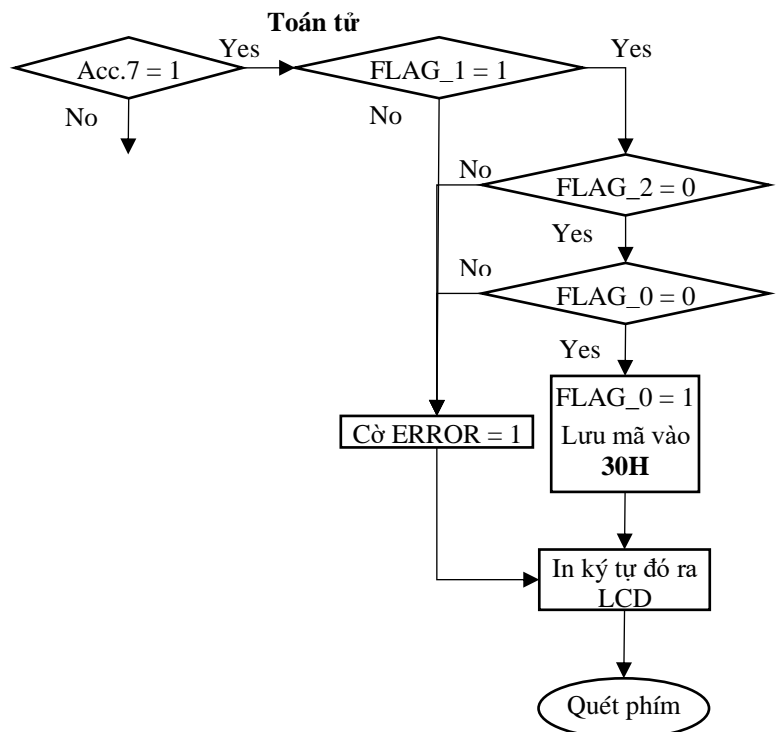
Nếu mã phím là dấu bằng, chương trình sẽ xét cờ Error và lỗi (20H) = 03H (chỉ nhập toán hạng thứ nhất và toán tử mà không nhập toán hạng thứ hai). Nếu một trong hai lỗi đó xảy ra, chương trình sẽ xóa màn hình LCD, in ra dòng thứ nhất "Syntax ERROR" và dòng thứ hai "OUT of RANGE", rồi set cờ Answer lên 1, quay trở lại tiếp tục quét phím. Nếu người dùng nhập đúng cú pháp thì chương trình chính sẽ tính toán rồi đưa ra kết quả, ta sẽ trình bày chi tiết quá trình này ở phần sau.

1.3 Mã phím là toán tử +-* /

Nếu mã phím đã được nhấn không phải là Reset và không phải dấu bằng, chương trình sẽ tiếp tục xét bit Acc.7. Nếu Acc.7 = 1 nghĩa là mã phím là toán tử.

Toán tử được nhập vào chỉ hợp lệ khi người dùng đã nhập vào khối 1 (FLAG_1 = 1), khối 2 chưa nhập (FLAG_2 = 0) và chưa nhập toán tử nào trước đó (FLAG_0 = 0). Nếu ít nhất một trong số ba điều kiện trên không thỏa mãn, cờ Error sẽ được set lên 1, toán tử đó vẫn được in ra màn hình, chương trình tiếp tục quét phím.

Nếu cả ba lỗi trên không xảy ra, chương trình chấp nhận toán tử đã nhập, lưu mã vào ô nhớ RAM nội 30H rồi in ký tự đó ra màn hình, quay về tiếp tục quét phím.



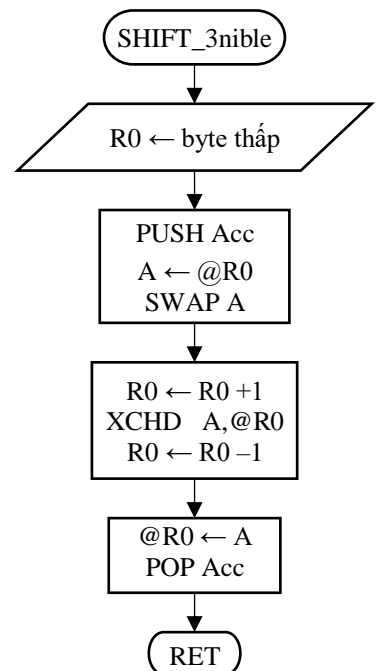
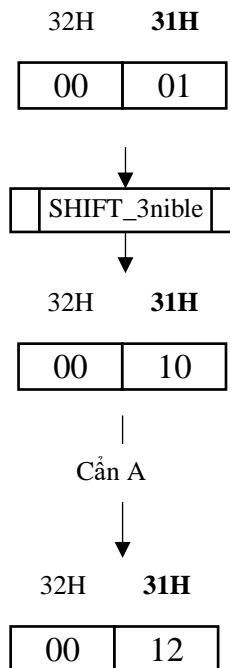
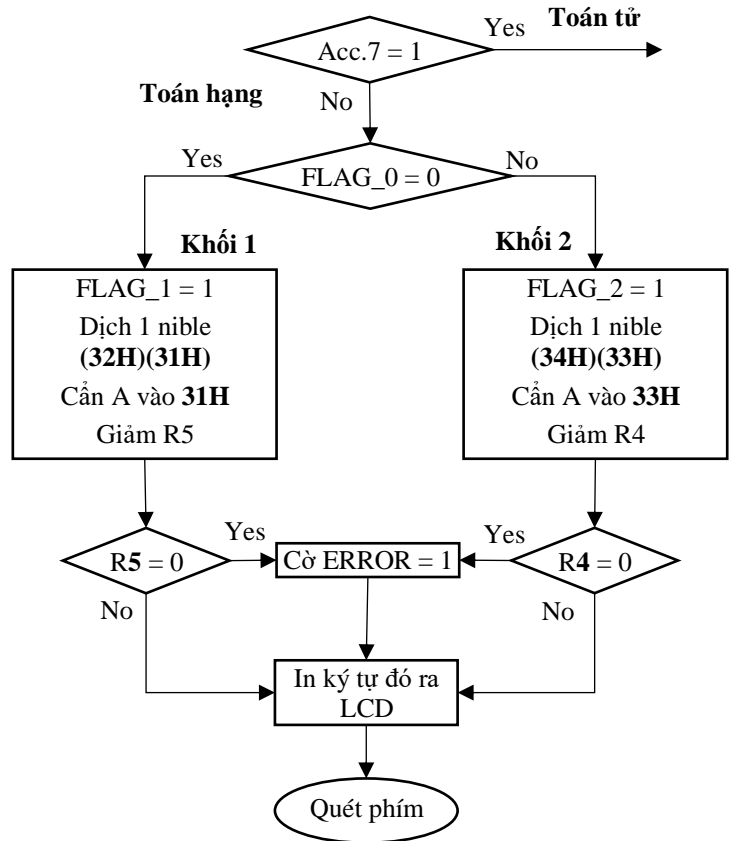
1.4 Mã phím là chữ số

Nếu bit $\text{Acc.7} = 0$ thì phím người dùng vừa nhấn là chữ số, nhưng làm sao để nhận biết được nó là chữ số của toán hạng thứ nhất hay toán hạng thứ hai? Để biết được điều đó ta sẽ xét cờ toán tử FLAG_0 , nếu cờ bằng 0 nghĩa là người dùng nhập toán hạng thứ nhất, ngược lại cờ bằng 1 thì nghĩa là người dùng đang nhập toán hạng thứ 2.

Đầu chương trình của cả hai khối, cờ tương ứng của chúng sẽ được set lên 1 để nhận biết người dùng đã nhập khối đó vào rồi. Chương trình dịch 1 nibble tương ứng rồi cần vào, giảm R4/R5 để xét tràn, báo lỗi.

Ví dụ ban đầu khi mới Reset chương trình, người dùng nhập chữ số 1 đầu tiên, sau đó nhấn số 2 (chưa có toán tử) chương trình sẽ đưa số 2 vào khối 1 bằng cách nạp $\text{R0} = \#31\text{H}$, gọi chương trình con **SHIFT_3nibble** để dịch **(32H)(31H)** lên 1 nibble như được minh họa ở hình bên. Nếu người dùng nhập vào quá 3 chữ số R4/R5 sẽ về 0, cờ Error được set lên 1, cứ tự do vẫn được hiển thị ra màn hình.

Trong trường hợp đã có toán tử ($\text{FLAG}_0 = 1$) thì khối được nhập sẽ là khối hai, với cơ chế hoàn toàn tương tự, chỉ khác là chúng ta cần nạp $\text{R0} = \#33\text{H}$ để chương trình con **SHIFT_3nibble** nhận biết, dịch **(34H)(33H)** lên 1 nibble.



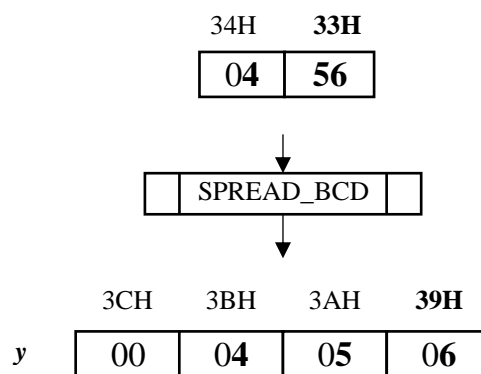
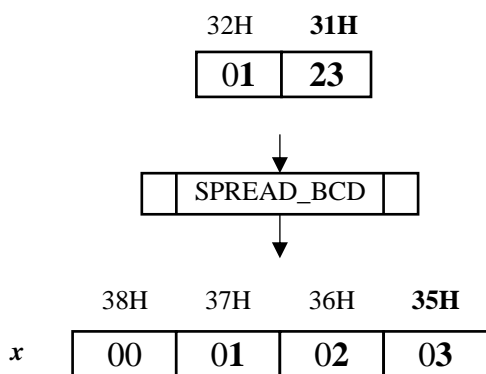
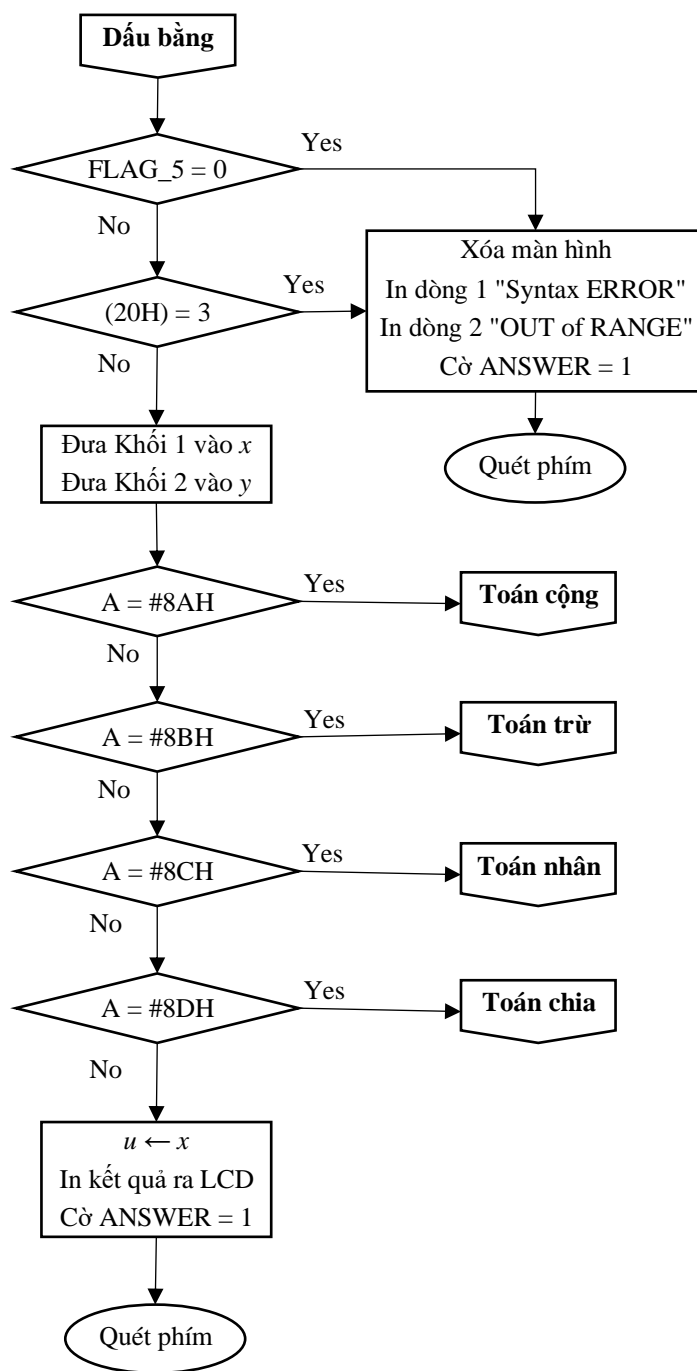
1.5 Mã phím là dấu bằng. Tính toán và in ra kết quả

Nếu mã phím là dấu bằng, chương trình sẽ xét cờ Error và lỗi (20H) = 03H (chỉ nhập toán hạng thứ nhất và toán tử mà không nhập toán hạng thứ hai). Nếu một trong hai lỗi đó xảy ra, chương trình sẽ xóa màn hình LCD, in ra dòng thứ nhất "Syntax ERROR" và dòng thứ hai "OUT of RANGE", rồi set cờ Answer lên 1, quay trở lại tiếp tục quét phím.

Nếu người dùng nhập đúng cú pháp thì chương trình chính sẽ đưa toán hạng thứ nhất vào x , đưa toán hạng thứ 2 vào y để phục vụ quá trình tính toán, nhờ đoạn chương trình **SPREAD_BCD** như được minh họa ở hình bên dưới.

Sau đó chương trình chính sẽ xét mã toán tử mà người dùng đã nhập trước đó (lưu ở ô nhớ 30H) để biết phép toán mà chương trình cần thực hiện là + - * hay /. Thực hiện phép toán đó rồi đưa ra kết quả (xem phần sau).

Trong trường hợp người dùng mới chỉ nhập toán hạng thứ nhất mà đã bấm dấu bằng, chương trình sẽ in ra kết quả đúng như con số mà người dùng đã nhập bằng cách chuyển x ra u , gọi chương trình con **PRINT_ANS** (xem ngay bên dưới), sau đó set cờ Answer lên 1, quay trở lại tiếp tục quét phím.

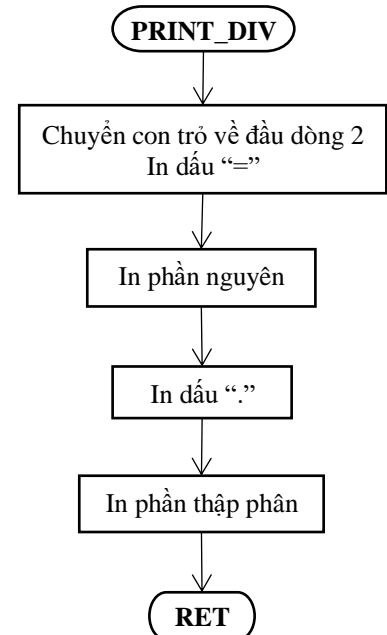
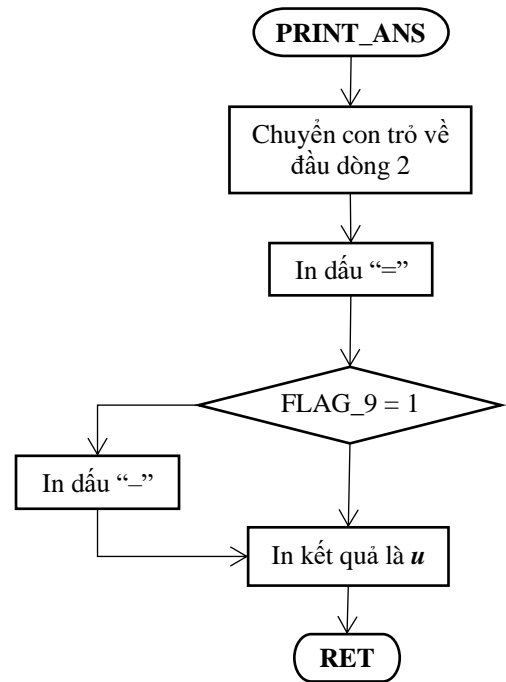


☞ chương trình con **PRINT_ANS** sẽ chuyển con trỏ xuống dòng 2, in dấu bằng rồi in nội dung của **u** ra màn hình, có xét cờ Negative Answer, không in các số 0 thừa phía trước (VD: **u** = 000109 sẽ in ra 109). Kết quả của phép $+-*$ sẽ được đưa ra **u** rồi in ra nhờ **PRINT_ANS**.

	46H	45H	44H	43H	42H	41H
u	00	00	00	01	00	09

Kết quả của phép $/$ sẽ được xét như sau: Nếu phép chia là hết thì đưa kết quả ra **u** hiển thị như phép $+-*$, ngược lại nếu là phép chia có dư, kết quả sẽ đưa ra **v** (dài hơn **u**) rồi in ra nhờ chương trình con **PRINT_DIV**. Hoạt động tương tự **PRINT_ANS**, **PRINT_DIV** sẽ chuyển con trỏ xuống dòng 2, in dấu bằng rồi in nội dung của **v** ra màn hình, không xét cờ Negative Answer, không in các số 0 thừa phía trước cũng như phía sau (VD: **v** = 009.3750000000 sẽ in ra 9.375).

	4FH	4EH	4DH	4CH	4BH	4AH	49H	48H	47H	46H	45H	44H	43H	42H	41H
v	00	00	00	09	.	03	07	05	00	00	00	00	00	00	00

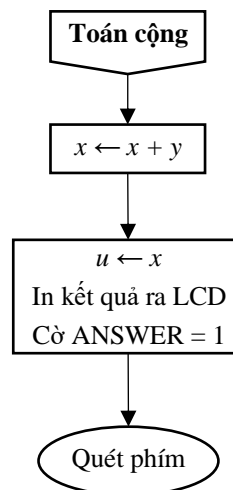
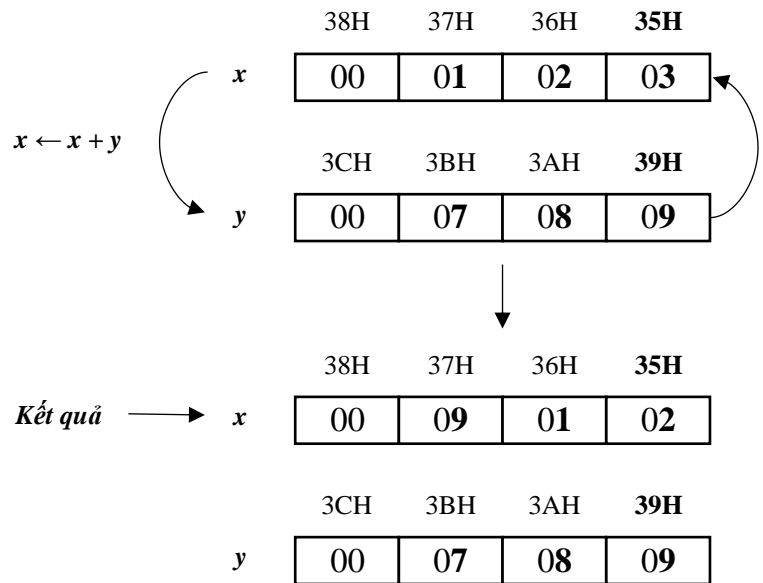


1.6 Toán cộng

Sau khi đã đưa toán hạng thứ nhất vào x, toán hạng thứ hai vào y, chương trình sẽ nạp $R1 = \#35H$, $R0 = \#39H$, gọi chương trình con ADD_4DIGITS để thực hiện phép cộng, chương trình con ADD_4DIGITS (trở x bởi R1, y bởi R0) sẽ tự động trả kết quả về x. Ta gán kết quả vừa tìm được vào u rồi in ra màn hình, set cờ Answer = 1, quay về quét phím.

Lưu ý, ta có thể trở R0, R1 tùy ý chương trình con ADD_4DIGITS đều hiểu được và tính toán. Ví dụ, $R1 = \#35H$, $R0 = \#3DH$ thì chương trình sẽ cộng

$$(38H)(37H)(36H)(\mathbf{35H}) \leftarrow (38H)(37H)(36H)(\mathbf{35H}) + (40H)(3FH)(3EH)(\mathbf{3DH})$$



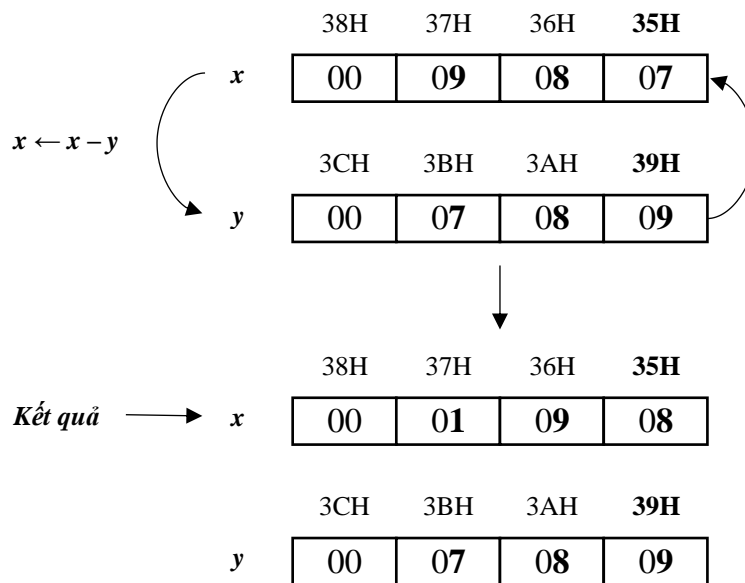
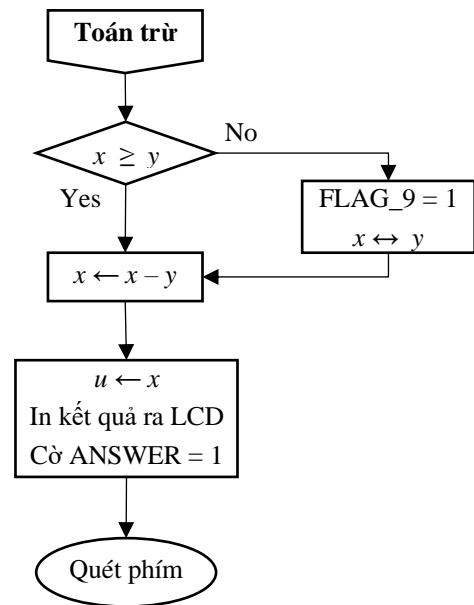
1.7 Toán trừ

Nếu phép toán là toán trừ, đầu tiên chương trình sẽ so sánh số bị trừ và số trừ. Nạp R1 = #38H, R0 = #3CH, gọi chương trình con COMPARE_4DIGITS (R1, R0 trở địa chỉ byte cao nhất vì so sánh từ số có trọng số cao).

Trường hợp $x \geq y$ thì cờ C = 1, ta thực hiện trừ như bình thường, lấy số lớn trừ số bé, trở R1 = #35H, R0 = #39H, gọi chương trình con SUBB_4DIGITS lấy $x \leftarrow x - y$, gán kết quả tính được cho u rồi in ra màn hình, set cờ Answer = 1, quay về quét phím.

Trường hợp $x < y$ thì cờ C = 0, ta set cờ Negativ Answer lên 1 rồi đổi chỗ x và y để lấy số lớn trừ số nhỏ, gán kết quả tính được cho u rồi in ra màn hình, set cờ Answer = 1, quay về quét phím.

Cũng tương tự như ADD_4DIGITS, SUBB_4DIGITS cũng được trở bởi R1, R0. Ta hoàn toàn có thể nạp R1, R0 tùy ý để phục vụ mục đích tính toán.



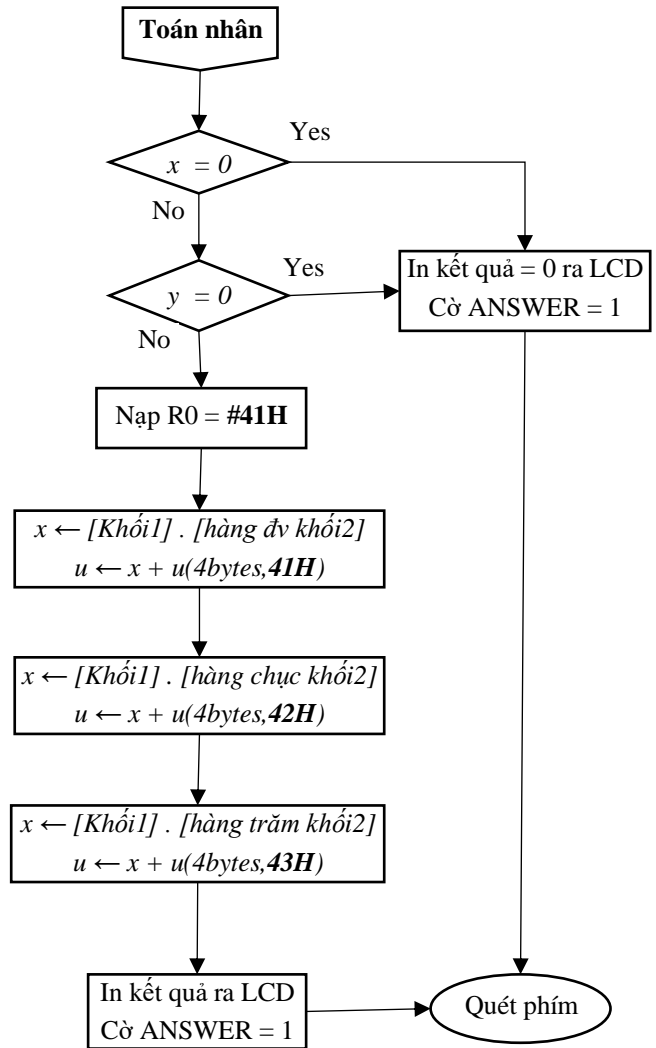
1.8 Toán nhân

Để tính phép toán nhân, đầu tiên ta xét x, y có bằng 0 hay không, nếu có thì kết quả đương nhiên bằng 0, ta in ra màn hình, set cờ Answer rồi quay về quét phím.

Nếu kết quả khác không, ta sẽ viết chương trình con SINGLE_MUL để nhân thừa số thứ nhất cho từng chữ số của thừa số thứ hai, cộng lại theo trọng số như khi chúng ta tính toán bằng số thập phân. Trả kết quả về u , in ra màn hình, set cờ Answer rồi quay về quét phím.

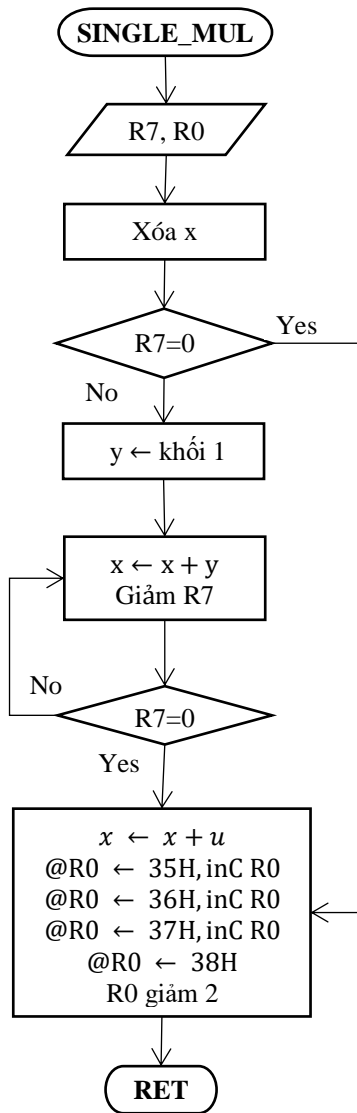
Nạp $R7 =$ chữ số của thừa số hai, nạp $R0 = \#41H$ (địa chỉ ô nhớ đầu tiên của u). Do có tối đa 3 chữ số nên ta sẽ gọi SINGLE_MUL 3 lần, mỗi lần sẽ nạp $R7$ lần lượt hàng đơn vị, hàng chục của thừa số thứ 2. Không cần nạp lại $R0$ vì sau mỗi lần gọi xong SINGLE_MUL thì $R0$ tự tăng thêm 1.

Ta hãy xét một ví dụ như sau:



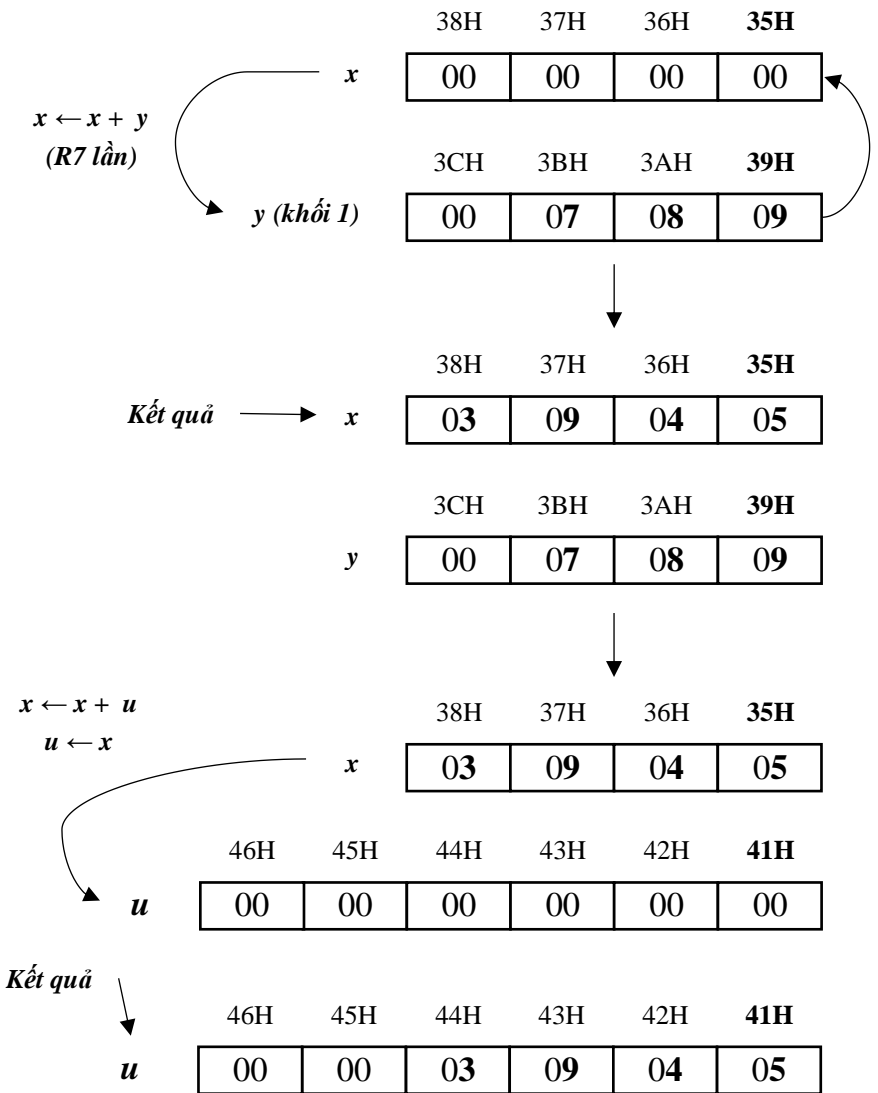
Chú thích:

$u(4bytes, 41H)$ tập thanh ghi u chỉ lấy 4 bytes, địa chỉ byte thấp nhất là 41H

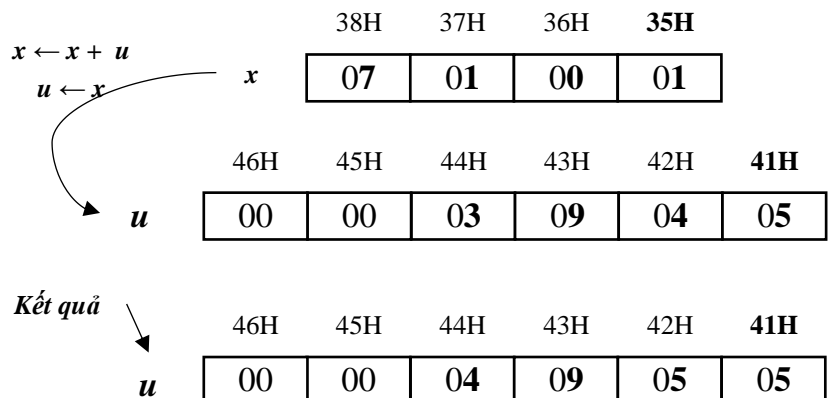


Ví dụ: lấy 789×95

Đầu tiên lấy $789 \times 5 = 3945$, nạp $R7 = 5$



Tương tự, lần thứ hai $789 \times 9 = 7101$, nạp $R7 = 9$, lúc này $R0$ đã tự tăng lên 1, khi đó cuối cùng:



1.9 Toán chia

Đầu tiên ta xét số chia, nếu số chia là 0 thì phép chia không thực hiện được, chương trình sẽ xóa màn hình, in ra dòng thứ nhất "Syntax ERROR" và dòng thứ hai "OUT of RANGE", rồi set cờ Answer lên 1, quay trở lại tiếp tục quét phím.

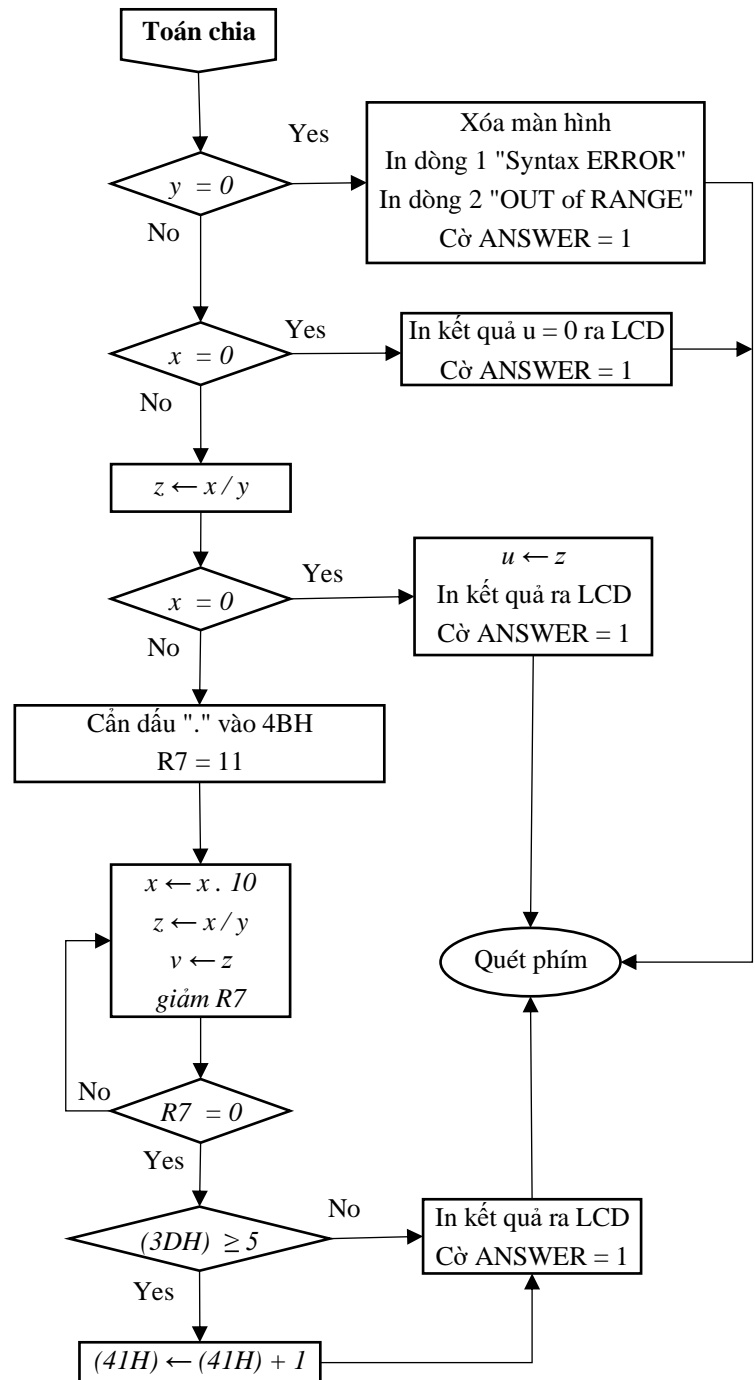
Trường hợp số bị chia bằng 0, số chia khác 0, thì kết quả sẽ bằng 0, in ra màn hình, rồi set cờ Answer lên 1, quay trở lại tiếp tục quét phím.

Khi đó, ta đảm bảo được rằng kết quả của phép chia này sẽ khác không. Chương trình chính sẽ gọi chương trình con DIV_4DIGITS.

Chương trình con DIV_4DIGITS lấy x/y kết quả trả về z , số dư trả về x . Nếu số dư bằng 0, phép chia sẽ hết, ta đưa kết quả (z) ra u rồi in ra LCD, set cờ Answer lên 1, quay trở lại tiếp tục quét phím.

Nếu phép chia có dư, chương trình cần dấu "." vào 4BH. Nạp $R7 = 11$ để tính 11 số sau dấu phẩy (ta hiển thị 10 số, số thứ 11 để làm tròn).

Bắt đầu lặp 11 lần, mỗi lần sẽ dịch x sang 1 đơn vị (tức là nhân 10), lấy x/y rồi đưa kết quả ra lần lượt 4AH ... 41H. Hết vòng lặp, xét chữ số thứ 11 sau dấu "." để làm tròn rồi in v ra màn hình, set cờ Answer lên 1, quay trở lại tiếp tục quét phím.

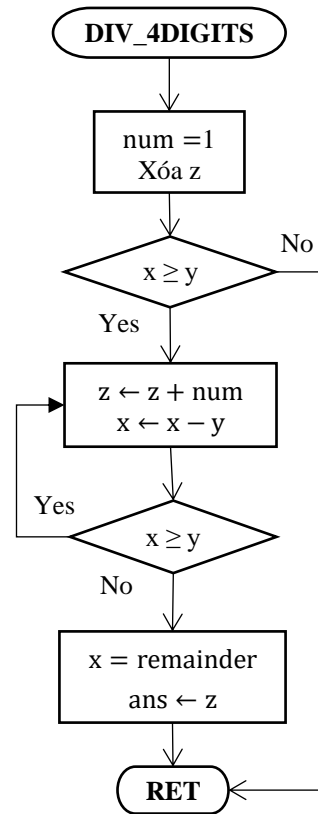


Chú thích:

x/y thì kết quả là z , số dư là x

$v \leftarrow z$ là đưa lần lượt z vào 4AH ... 41H (10 số sau dấu phẩy)

(3DH) là chữ số thứ 11 sau dấu phẩy



Ví dụ: lấy $789/6 = 131$ dư 3

$x = 3$

$y = 6$

$z = 131$

Đưa z ra v

$z \leftarrow x/y$
 $x \leftarrow \text{remaindder}$

	38H	37H	36H	35H
x	00	00	00	03

	3CH	3BH	3AH	39H
y	00	00	00	06

	40H	3FH	3EH	3DH
z	00	01	03	01

	4FH	4EH	4DH	4CH	4BH	4AH	49H	48H	47H	46H	45H	44H	43H	42H	41H
v	00	01	03	01	.	00	00	00	00	00	00	00	00	00	00

I. Trình hợp ngữ hoàn chỉnh:

;BTL - VI XU LI

;SIMPLE CACULATOR

;=====

DBUS EQU P0 ;LCD

RS BIT P2.0

RW BIT P2.2

E BIT P2.1

;-----

ROW EQU P1 ;KEY PAD

COLUMN EQU P2

;-----

FLAG_0 BIT 00H ;Co toan tu

FLAG_1 BIT 01H ;Co khoi 1

FLAG_2 BIT 02H ;Co khoi 2

FLAG_3 BIT 03H ;Co tran khoi 1

FLAG_4 BIT 04H ;Co tran khoi 2

FLAG_5 BIT 05H ;Co ERROR SYNTAX

FLAG_6 BIT 06H ;Co OUT OF RANGE

FLAG_7 BIT 07H ;Co ANSWER

FLAG_8 BIT 08H ;Co de in ra so 0

FLAG_9 BIT 09H ;Co negative answer

;=====

ORG 0000H

LJMP MAIN

LJMP EX0_ISR

ORG 00BH

LJMP T0_ISR

ORG 0013H

LJMP EX1_ISR

ORG 0023H

LJMP SPI_ISR

;=====

ORG 0030H

MAIN: MOV SP,#5FH

CALL INITIAL


```

;-----

;-----
LOOP_00:      CALL      START
;-----
LOOP_01:
                MOV      R7,#50                ;CHECK KEY
PAD
LOOP_02:      CALL      CHECK_KEY
                JNC      LOOP_01
                DJNZ     R7,LOOP_02
                PUSH     ACC
LOOP_03:      MOV      R7,#50
LOOP_04:      CALL      CHECK_KEY
                JC       LOOP_03
                DJNZ     R7,LOOP_04
                POP      ACC
;-----
                JNB      FLAG_7,CHECK_RESET
                PUSH     ACC
                CALL     START
                MOV      A,#01H                ;RESET, xoa
man hinh
                CALL     DELAY_2MS
                CLR      RS
                CALL     WRITE_OUT
                POP      ACC
;-----
CHECK_RESET:  CJNE      A,#8FH,CHECK_EQUAL
                MOV      A,#01H                ;RESET, xoa
man hinh
                CALL     DELAY_2MS
                CLR      RS
                CALL     WRITE_OUT
                SJMP     LOOP_00
;-----
CHECK_EQUAL:  CJNE      A,#8EH,SAVE_PRINT
;DAU "="

```

```

                                JNB          FLAG_5,CHECK_EQUAL_2

CHECK_EQUAL_1:  MOV          A,#01H                ;Co ERROR
                CALL        DELAY_2MS             ;xoa man hinh
                CLR         RS
                CALL        WRITE_OUT

                MOV         DPTR,#TAB_ERROR1
                CALL        PRINT_LINE

                MOV         A,#0C0H                ;Chuyen con tro
ve dau dong 2
                CALL        DELAY_2MS
                CLR         RS
                CALL        WRITE_OUT

                MOV         DPTR,#TAB_ERROR2
                CALL        PRINT_LINE
                SETB        FLAG_7                ;SET
FLAG ANSWER
                JMP         LOOP_01

;-----
CHECK_EQUAL_2:  MOV         A,20H
                ANL         A,#00000111B
                CJNE        A,#03H,SPREAD_BCD_00
                SJMP        CHECK_EQUAL_1

;-----
SPREAD_BCD_00:
                JMP         SPREAD_BCD

;-----
CALCULATE:
                MOV         A,30H
                CLR         ACC.7
                CJNE        A,#0AH,CALCULATE_1
                JMP         PLUS
CALCULATE_1:   CJNE        A,#0BH,CALCULATE_2
                JMP         MINUS
CALCULATE_2:   CJNE        A,#0CH,CALCULATE_3
                JMP         MULTIPLY
CALCULATE_3:   CJNE        A,#0DH,CALCULATE_4

```

```

                                JMP      DIVIDE
;-----
CALCULATE_4:
                                MOV      41H,35H
                                MOV      42H,36H
                                MOV      43H,37H
                                CALL     PRINT_ANS
                                JMP      LOOP_01
;-----
SAVE_PRINT:                    JNB      ACC.7,NUM

OPERAND:                        JNB      FLAG_1,ERROR      ;MA LA TOAN
HANG
                                JB       FLAG_2,ERROR

                                JB       FLAG_0,ERROR
                                SETB     FLAG_0
                                MOV      30H,A
                                SJMP     OPERAND1
ERROR:                          SETB     FLAG_5
OPERAND1:                      CLR      ACC.7
                                MOV      DPTR,#TAB_ASCII

                                CALL     PRINT_TAB
                                JMP      LOOP_01
;-----
NUM:                            JB       FLAG_0,KHOI2_00
    ;MA LA SO

KHOI1_00:                      JNB      FLAG_3,KHOI1_01
    ;KHOI 1
                                JMP      ERROR
    ;TRAN ROI

KHOI1_01:                      SETB     FLAG_1
                                MOV      R0,#31H
                                CALL     SHIFT_3nibles
                                PUSH     ACC
                                ADD      A,31H
                                MOV      31H,A
                                DJNZ     R5,KHOI1_02

```

```

                                SETB     FLAG_3
                                SETB     FLAG_5
KHOI1_02:                      POP      ACC
                                MOV      DPTR,#TAB_ASCII
                                CALL     PRINT_TAB
                                JMP      LOOP_01

;-----
KHOI2_00:                      JNB      FLAG_4,KHOI2_01
                                ;KHOI 2

                                JMP      ERROR

                                ;TRAN ROI

KHOI2_01:                      SETB     FLAG_2
                                MOV      R0,#33H
                                CALL     SHIFT_3nibles
                                PUSH     ACC
                                ADD      A,33H
                                MOV      33H,A
                                DJNZ     R4,KHOI2_02
                                SETB     FLAG_4
                                SETB     FLAG_5
KHOI2_02:                      POP      ACC
                                MOV      DPTR,#TAB_ASCII
                                CALL     PRINT_TAB
                                JMP      LOOP_01

;-----
PLUS:                          MOV      R1,#35H
                                MOV      R0,#39H
                                CALL     ADD_4DIGITS

;-----
PLUS_1:                        MOV      41H,35H
                                MOV      42H,36H
                                MOV      43H,37H
                                MOV      44H,38H
                                CALL     PRINT_ANS
                                JMP      LOOP_01

;-----
ADD_4DIGITS:                   CLR      C ;CONG (R1+3)(R1+2)(R1+1)R1 <=
                                (R1+3)(R1+2)(R1+1)R1 + (R1+3)(R0+2)(R0+1)R0
                                ;KHONG TRAN VI 999*9 = 8991

```

ADD_4DIGITS_1:	MOV	R6,#4
	CALL	FULL_ADDDER
	DJNZ	R6,ADD_4DIGITS_1
	RET	
;-----		
FULL_ADDDER:	MOV	A,@R1
	MOV	B,@R0
	ADDC	A,B
	DA	A
	CJNE	A,#10,\$+3
	CPL	C
	ANL	A,#0FH
	MOV	@R1,A
	INC	R1
	INC	R0
	RET	
;-----		
MINUS:	MOV	R1,#38H
	MOV	R0,#3CH
	CALL	COMPARE_4DIGITS
	JC	POSITIVE
;-----		
	MOV	R1,#35H
	MOV	R0,#39H
	CALL	XCH_4DIGITS
	SETB	FLAG_9
;-----		
POSITIVE:	MOV	R1,#35H
	MOV	R0,#39H
	CALL	SUBB_4DIGITS
	JMP	PLUS_1
;-----		
XCH_4DIGITS:		
XCH_4DIGITS_0:	MOV	R7,#4
	MOV	A,@R1
	XCH	A,@R0
	MOV	@R1,A
	INC	R0

```

                INC        R1
                DJNZ       R7,XCH_4DIGITS_0
                RET

;-----
COMPARE_4DIGITS:
                MOV        R2,#4
COM_0:          MOV        A,@R1 ;S0 SANH 4DIGITS TU CAO DEN
                THAP, IF R1>=R0 THEN C=1, ELSE C=0
                CLR        C
                SUBB       A,@R0
                CJNE       A,#0,COM_1
                DEC        R0
                DEC        R1
                DJNZ       R2,COM_0
                SETB       C
                RET
COM_1:          JNB        ACC.7,COM_2
                CLR        C
                RET
COM_2:          SETB       C
                RET

;-----
SUBB_4DIGITS: ;SO LON - SO BE > 0
                CLR        C
                CALL       FULL_SUBTRACTOR
                CALL       FULL_SUBTRACTOR

                CALL       FULL_SUBTRACTOR
                CALL       FULL_SUBTRACTOR

                RET
;-----
FULL_SUBTRACTOR:
                PUSH       PSW
                MOV        A,@R1
                MOV        B,@R0
                SUBB       A,B
                JB         ACC.7,F_S_0
                POP        PSW
                CLR        C
                SJMP       F_S_1

```

```

F_S_0:      MOV      A,@R1
            ADD      A,#10
            POP      PSW
            SUBB     A,B
            SETB     C
F_S_1:      MOV      @R1,A
            INC      R1
            INC      R0
            RET

;-----
MULTIPLY:   MOV      A,#0
            CJNE     A,31H,CHECK_ZERO
            CJNE     A,32H,CHECK_ZERO
ZERO:       CALL     PRINT_ANS
            JMP      LOOP_01

CHECK_ZERO: CJNE     A,33H,NON_ZERO_01
            CJNE     A,34H,NON_ZERO_01
            SJMP     ZERO

;-----
NON_ZERO_01:
            PUSH     3AH
            MOV      R7,39H
            MOV      R0,#41H
            CALL     SINGLE_MUL

;-----
            POP      ACC
            MOV      R7,A
            CALL     SINGLE_MUL

;-----
            MOV      R7,34H
            CALL     SINGLE_MUL

;-----
            CALL     PRINT_ANS
            JMP      LOOP_01

;-----
SINGLE_MUL:
            MOV      2FH,R0                ;SAVE R0

;-----
            MOV      35H,#0
            MOV      36H,#0

```

```

MOV      37H,#0
MOV      38H,#0
;-----
CJNE     R7,#0,SINGLE_MUL_0
SJMP     SINGLE_MUL_2
;-----
SINGLE_MUL_0:  MOV      R0,#31H
               MOV      R1,#39H
               CALL     SPREAD_BCD_01
;-----
SINGLE_MUL_1:  MOV      R1,#35H
               MOV      R0,#39H
               CALL     ADD_4DIGITS
               DJNZ     R7,SINGLE_MUL_1
;-----
SINGLE_MUL_2:  MOV      R1,#35H
               MOV      R0,2FH           ;LAY LAI R0
               CALL     ADD_4DIGITS

               MOV      R0,2FH
               MOV      @R0,35H
               INC      R0
               MOV      @R0,36H
               INC      R0
               MOV      @R0,37H
               INC      R0
               MOV      @R0,38H
               DEC      R0
               DEC      R0
               RET
;-----
DIVIDE:      MOV      A,#0
               CJNE     A,33H,NON_ZERO_02
               CJNE     A,34H,NON_ZERO_02
               JMP      CHECK_EQUAL_1
NON_ZERO_02:  CJNE     A,31H,NON_ZERO_03
               CJNE     A,32H,NON_ZERO_03
               SJMP     ZERO
;-----
NON_ZERO_03:  CALL     DIV_4DIGITS

```



```

;-----
MOV      R7,#4
MOV      R0,#35H
DIVIDE_0: CJNE    @R0,#0,DIVIDE_1      ;Remainder = 0?
INC      R0
DJNZ     R7,DIVIDE_0

;-----
MOV      R1,#3DH      ;Remainder = 0 then
PRINT_ANS
MOV      R0,#41H
CALL     XCH_4DIGITS
CALL     PRINT_ANS
JMP      LOOP_01

;-----
DIVIDE_1:
MOV      R1,#3DH
;Remainder != 0
MOV      R0,#4CH
CALL     XCH_4DIGITS
MOV      4BH,#0EH      ;DAU '.' THAP
PHAN

;-----
MOV      R7,#11 ;10 SO SAU DAU PHAY + 40H DE
XET LAM TRON
MOV      R0,#4AH
DIVIDE_2:
MOV      2FH,R0
CALL     SHIFT_4DIGITS      ;REMAINDER X
10
CALL     DIV_4DIGITS
MOV      R0,2FH
MOV      @R0,3DH
DEC      R0
DJNZ     R7,DIVIDE_2
MOV      P3,41H

;-----
MOV      A,3DH      ;LAM
TRON
CJNE     A,#5,$+3
CPL      C
MOV      A,41H
ADDC     A,#0

```

```

MOV      41H,A
;-----
CALL     PRINT_DIV
JMP      LOOP_01

;-----
DIV_4DIGITS:
MOV      50H,#1
MOV      3DH,#0
MOV      3EH,#0
MOV      3FH,#0
MOV      40H,#0

;-----
MOV      R1,#38H
MOV      R0,#3CH
CALL     COMPARE_4DIGITS
JC       DIV_4DIGITS_1
RET

;-----
DIV_4DIGITS_1:
MOV      R1,#3DH
MOV      R0,#50H
CALL     ADD_4DIGITS
MOV      R1,#35H
MOV      R0,#39H
CALL     SUBB_4DIGITS
;-----
MOV      R1,#38H
MOV      R0,#3CH
CALL     COMPARE_4DIGITS
JC       DIV_4DIGITS_1
RET
;-----

SHIFT_4DIGITS:
10 35H
MOV      R0,#35H
MOV      A,#0
XCH      A,@R0
INC      R0

XCH      A,@R0
;NHAN

```

```

                                INC      R0
                                XCH      A,@R0
                                INC      R0
                                XCH      A,@R0
                                RET

;-----
SPREAD_BCD:
                                MOV      R0,#31H
                                MOV      R1,#35H
                                CALL     SPREAD_BCD_01
                                MOV      R0,#33H
                                MOV      R1,#39H
                                CALL     SPREAD_BCD_01

                                JMP      CALCULATE

;=====
=====
SPREAD_BCD_01:  PUSH     ACC
                                MOV      A,@R0
                                ANL      A,#0FH
                                MOV      @R1,A
                                MOV      A,@R0
                                ANL      A,#0F0H
                                SWAP     A
                                INC      R1
                                MOV      @R1,A
                                INC      R0
                                MOV      A,@R0
                                INC      R1
                                MOV      @R1,A
                                INC      R1
                                MOV      @R1,#0
                                POP      ACC
                                RET

;=====
=====
PRINT_ANS:      MOV      A,#0C0H                ;Chuyen con tro
ve dau dong 2

                                CALL     DELAY_2MS
                                CLR      RS
                                CALL     WRITE_OUT

```

```

MOV      A,#0
MOV      DPTR,#TAB_ANSWER
PRINT_ANS_0:  PUSH  ACC
               CALL  PRINT_TAB
               POP   ACC
               INC   A
               JNC   PRINT_ANS_0

;-----
               MOV   DPTR,#TAB_ASCII
               JNB   FLAG_9,PRINT_ANS_1  ;Xet co negative
answer
               MOV   A,#0BH
               CALL  PRINT_TAB
;-----
PRINT_ANS_1:  MOV   R0,#46H
               MOV   R7,#5      ;41H HIEN THI BANG TAY
NEN 6 - 1 = 5
               CLR   FLAG_8
;-----
PRINT_ANS_2:
               MOV   A,@R0
               JB    FLAG_8,PRINT_ANS_3
               JZ    PRINT_ANS_4
PRINT_ANS_3:  CALL  PRINT_TAB
               SETB  FLAG_8
PRINT_ANS_4:  DEC   R0
               DJNZ  R7,PRINT_ANS_2

               MOV   A,41H
               CALL  PRINT_TAB
               SETB  FLAG_7          ;SET
FLAG ANSWER
               RET
;=====
PRINT_DIV:   MOV   A,#0C0H          ;Chuyen con tro
ve dau dong 2
               CALL  DELAY_2MS
               CLR   RS
               CALL  WRITE_OUT

```

```

MOV      A,#0
MOV      DPTR,#TAB_DIV
PRINT_DIV_0:  PUSH  ACC
               CALL  PRINT_TAB
               POP   ACC
               INC   A
               JNC   PRINT_DIV_0
;-----
MOV      DPTR,#TAB_ASCII
PRINT_DIV_1:  MOV   R0,#4FH
               MOV   R7,#3           ;4CH HIEN THI
BANG TAY
               CLR   FLAG_8
;-----
PRINT_DIV_2:
               MOV   A,@R0
               JB    FLAG_8,PRINT_DIV_3
               JZ    PRINT_DIV_4
PRINT_DIV_3:  CALL  PRINT_TAB
               SETB  FLAG_8
PRINT_DIV_4:  DEC   R0
               DJNZ  R7,PRINT_DIV_2
;-----
               MOV   A,4CH
               CALL  PRINT_TAB
               MOV   A,4BH
               CALL  PRINT_TAB
;-----
               MOV   R7,#10
               MOV   R0,#41H

PRINT_DIV_5:  MOV   A,@R0
               JNZ   PRINT_DIV_6
               INC   R0
               DEC   R7
               JMP   PRINT_DIV_5

PRINT_DIV_6:
               MOV   R0,#4AH
PRINT_DIV_7:  MOV   A,@R0
               CALL  PRINT_TAB
               DEC   R0

```

```

                                DJNZ      R7,PRINT_DIV_7

                                SETB      FLAG_7                ;SET FLAG
ANSWER
                                RET

;=====
START:      MOV      R5,#04H                ;CAM SU
DUNG R4, R5

                                MOV      R4,#04H
                                MOV      20H,#00H                ;Xoa co FLAG
                                MOV      21H,#00H
                                MOV      R1,#48                ;Xoa RAM noi tu
30H toi 5FH

                                MOV      R0,#30H
CLR_REG:    MOV      @R0,#00H
                                INC      R0
                                DJNZ     R1,CLR_REG
                                RET

;=====
PRINT_LINE: MOV      R7,#16
                                MOV      A,#0
PRINT_LINE_1: PUSH     ACC
                                CALL     PRINT_TAB
                                POP      ACC
                                INC      A
                                DJNZ     R7,PRINT_LINE_1
                                RET

;=====
PRINT_TAB:  MOVC     A,@A+DPTR
                                CJNE     A,#00H,PRINT_TAB1
                                SETB     C
                                SJMP     PRINT_TAB2
PRINT_TAB1: CALL     DELAY_2MS
                                SETB     RS
                                CALL     WRITE_OUT
                                CLR      C
PRINT_TAB2: RET

;=====

```

```

CHECK_KEY:      MOV      A,#0EFH
                MOV      R6,#4
CHECK_KEY1:     MOV      COLUMN,A
                MOV      22H,A
                MOV      A,ROW
                ANL      A,#0FH
                CJNE     A,#0FH,CHECK_KEY2
                MOV      A,22H
                RL       A
                DJNZ     R6,CHECK_KEY1
                CLR      C
                SJMP     CHECK_KEY3

CHECK_KEY2:     MOV      DPTR,#TAB1
                ADD      A,#-7
                MOVC     A,@A+DPTR
                PUSH ACC

                MOV      A,22H
                ANL      A,#0F0H
                SWAP     A
                ADD      A,#-7
                MOVC     A,@A+DPTR
                RL       A
                RL       A
                POP      22H
                ORL      A,22H

                MOV      DPTR,#TAB_KEY
                MOVC     A,@A+DPTR
                SETB     C

CHECK_KEY3:     RET
;-----
TAB1:           DB      03H,0FFH,0FFH,0FFH,02H,0FFH,01H,00H
TAB_KEY:        DB
07H,04H,01H,8FH,08H,05H,02H,00H,09H,06H,03H,8EH,8DH,8CH,8BH,8AH
TAB_ASCII:      DB      '0123456789+-*/.' ;7FH = ma DEL
TAB_ERROR1:     DB      ' Syntax ERROR '
TAB_ERROR2:     DB      ' OUT of RANGE '
TAB_ANSWER:     DB      '      = ',00H

```

TAB_DIV: DB '= '00H ;16 - 2 - 4 = 10
SO SAU DAU PHAY

;=====

```
SHIFT_3nibles:  PUSH    ACC
                  MOV     A,@R0
                  SWAP    A
                  INC     R0
                  XCHD    A,@R0
                  DEC     R0
                  MOV     @R0,A
                  POP     ACC
                  RET
```

;=====

```
INITIAL:        MOV     A,#38H      ;Giao tiep 8bit, 2dong, 5x8dots
                  CALL    DELAY_2MS
                  CLR     RS
                  CALL    WRITE_OUT
```

;-----

```
                  MOV     A,#01H      ;Xoa man hinh
                  CALL    DELAY_2MS
                  CLR     RS
                  CALL    WRITE_OUT
```

;-----

```
boi con tro      MOV     A,#0FH      ;Hien man hinh, chop ky tu chi
                  CALL    DELAY_2MS
                  CLR     RS
                  CALL    WRITE_OUT
```

;-----

```
ghi/doc data)    MOV     A,#06H      ;Dich con tro sang phai (khi
                  CALL    DELAY_2MS
                  CLR     RS
                  CALL    WRITE_OUT
```

;-----

```
                  RET
```

;-----

```
DELAY_2MS:       MOV     TMOD,#00000001B
                  MOV     TH0,#HIGH(-1989)      ;2ms = 20000us
                  MOV     TL0,#LOW(-1989)
```



```

                                SETB    TR0
                                JNB      TF0,$
                                CLR      TR0
                                CLR      TF0
                                RET

;-----
WRITE_OUT:    MOV      DBUS,A
              CLR      RW
              SETB E
              CLR      E
              RET

;=====
=====
EX0_ISR:
              RETI

;=====
T0_ISR:
              RETI

;=====
EX1_ISR:
              RETI

;=====

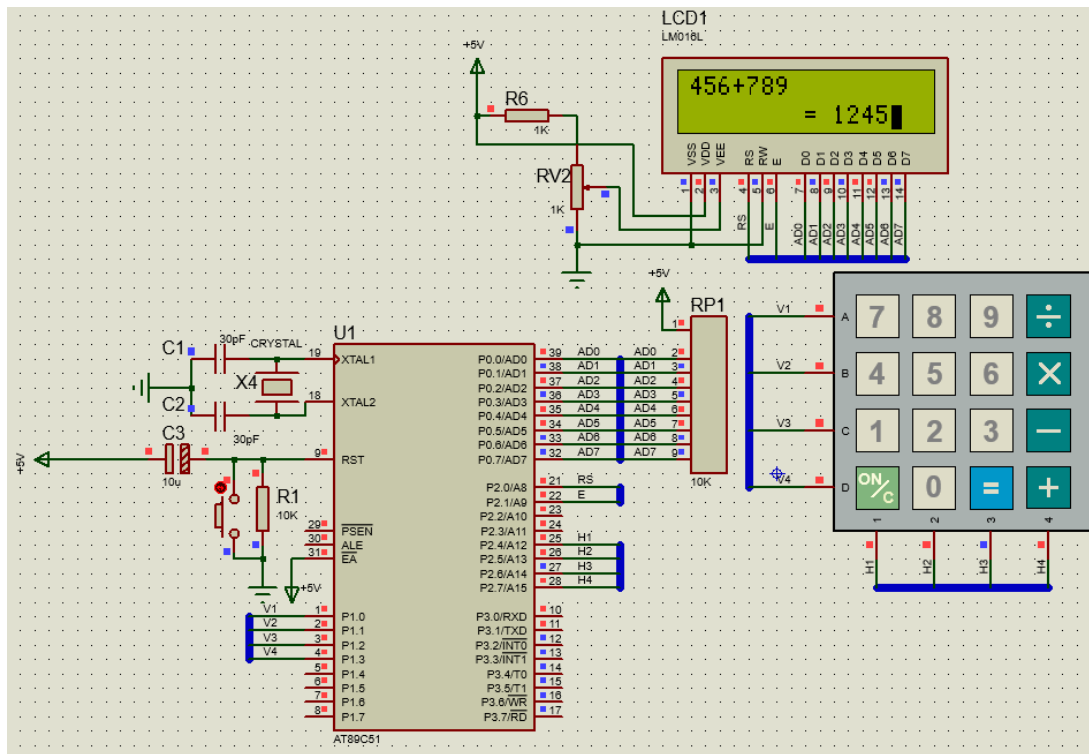
SPI_ISR:
              RETI
              END

```

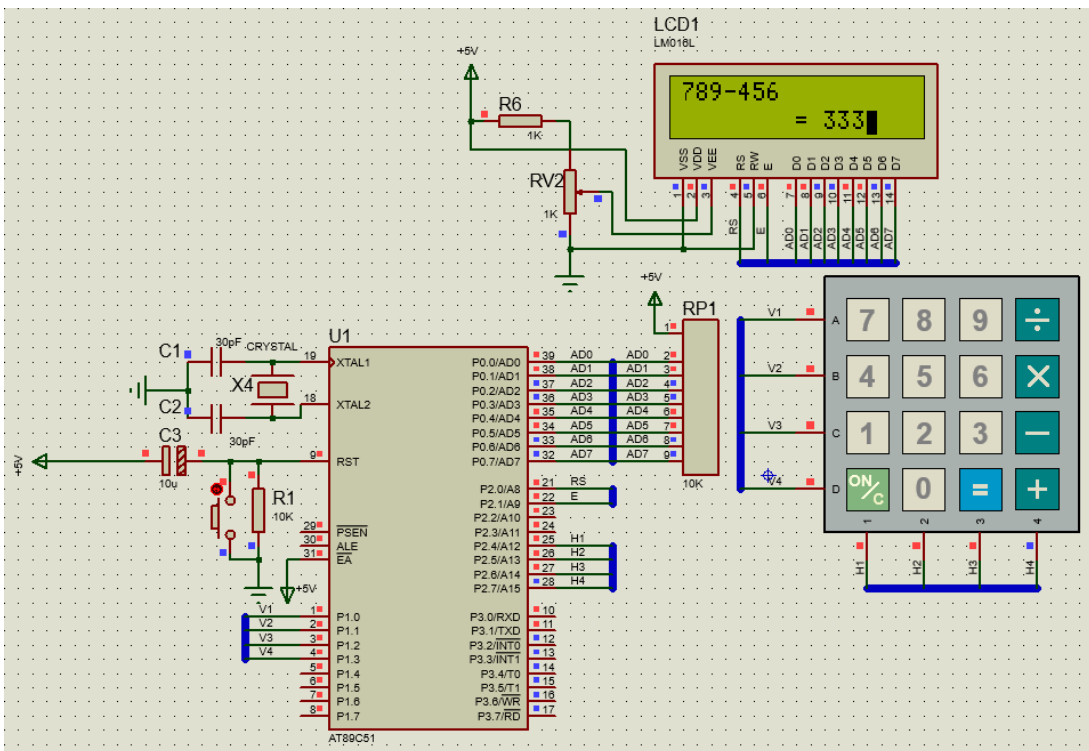
D. BÁO CÁO KẾT QUẢ MÔ PHÒNG TRÊN PROTEUS:

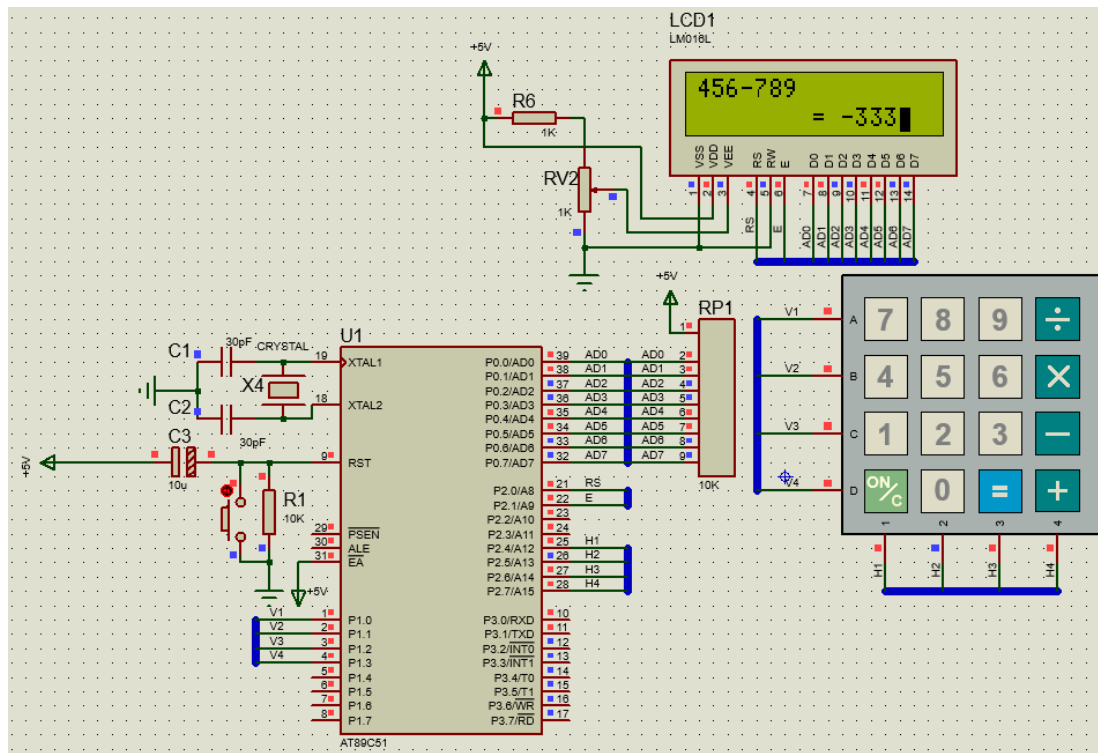
- Sau khi thiết kế phần cứng và phần mềm như trên đã trình bày thì nhóm em thu được những kết quả sau:

+ **TH1:** Thực hiện phép cộng 2 số ($0 < x < 999$):

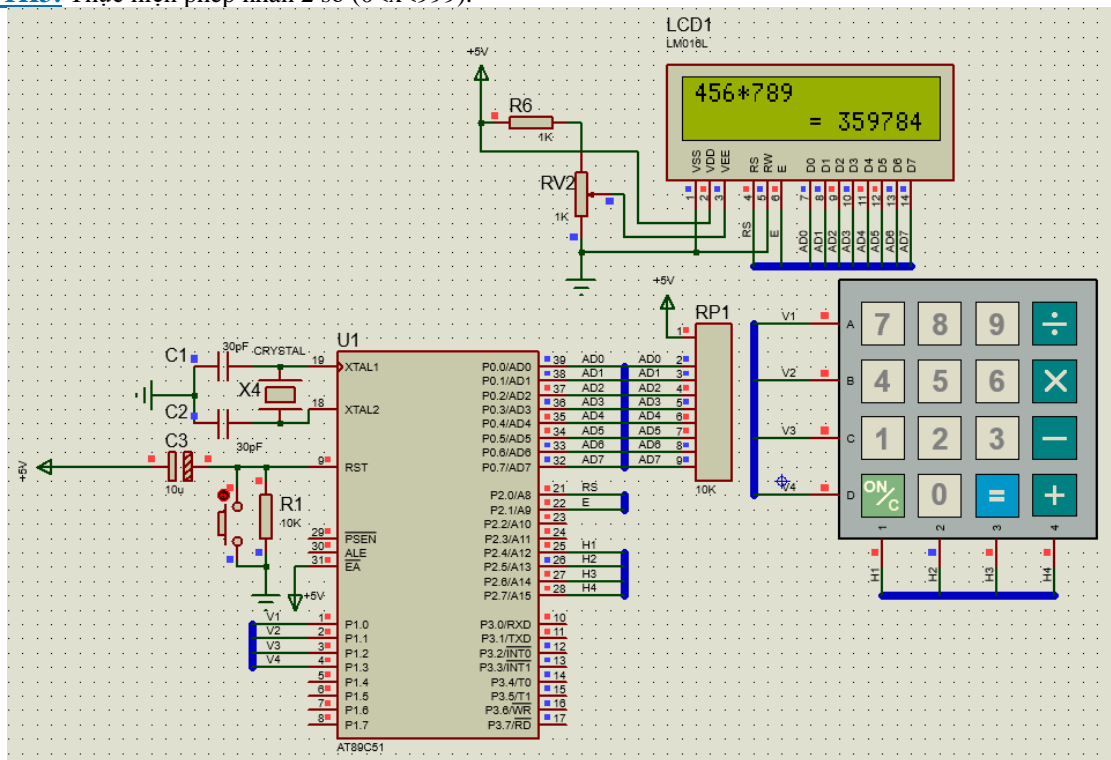


+ **TH2:** Thực hiện phép trừ 2 số ($0 < x < 999$):

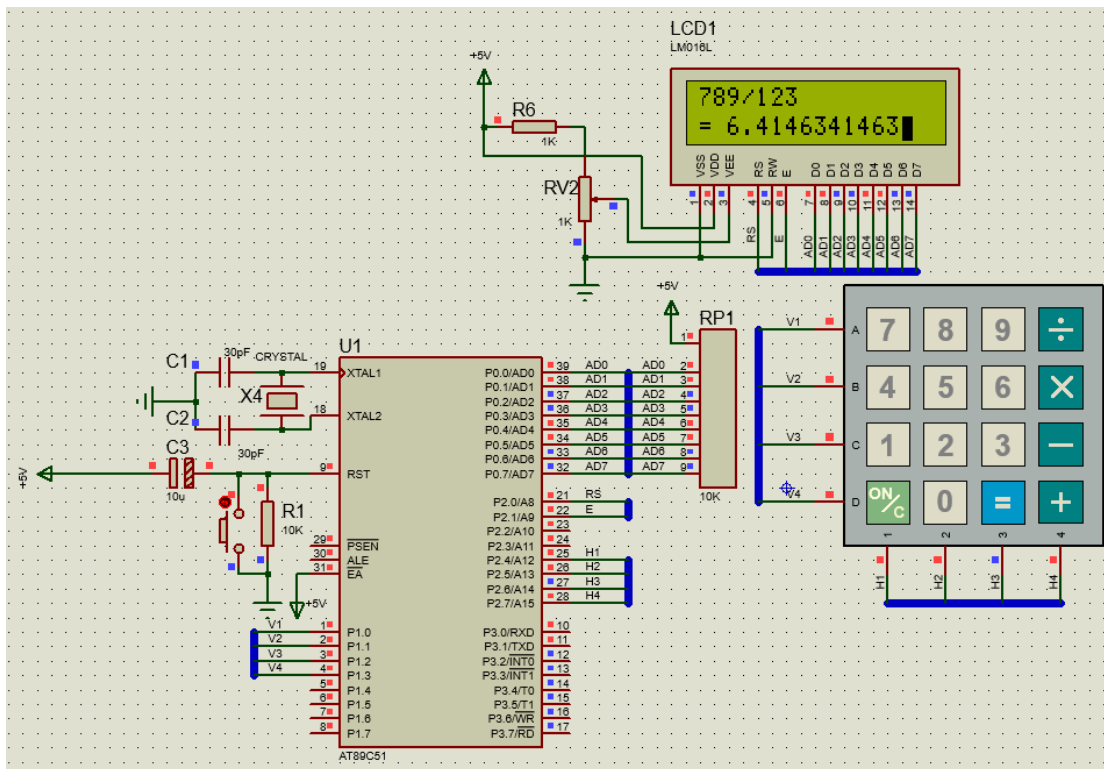
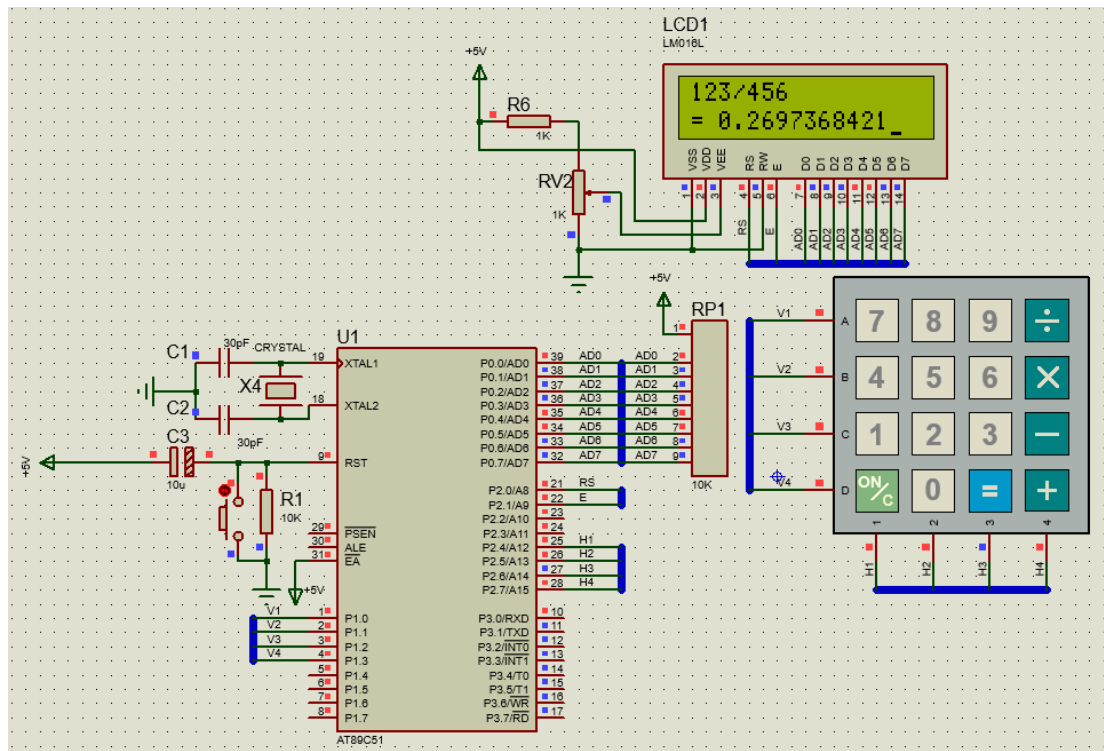




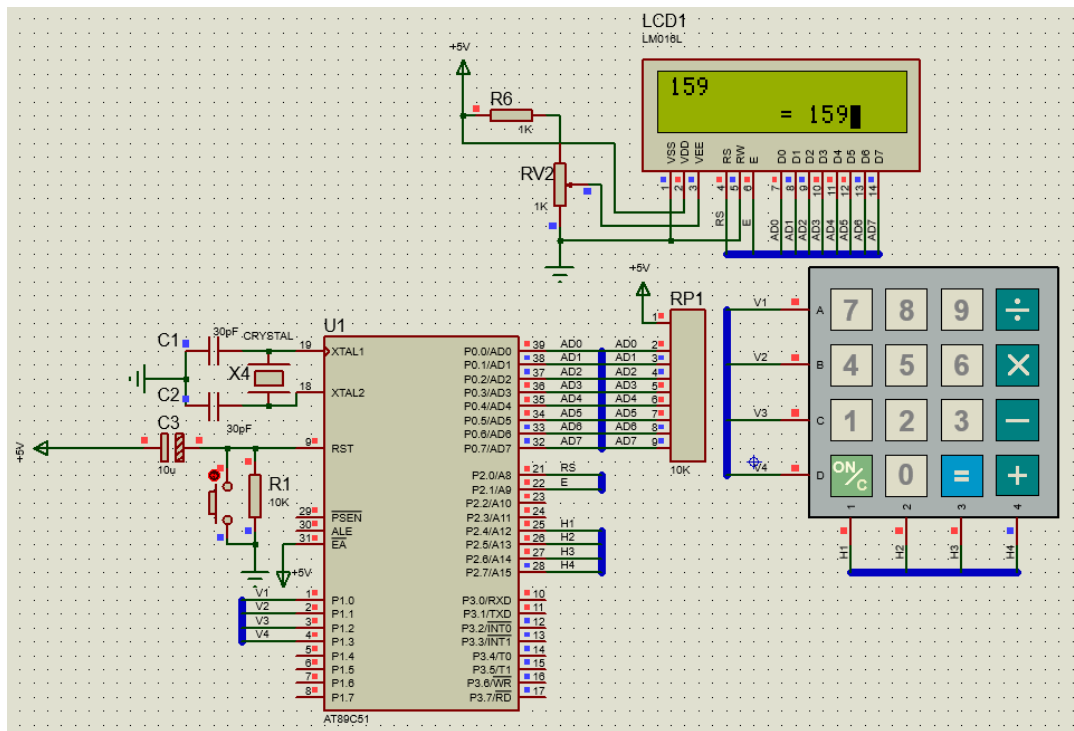
+ **TH3:** Thực hiện phép nhân 2 số ($0 < x < 999$):



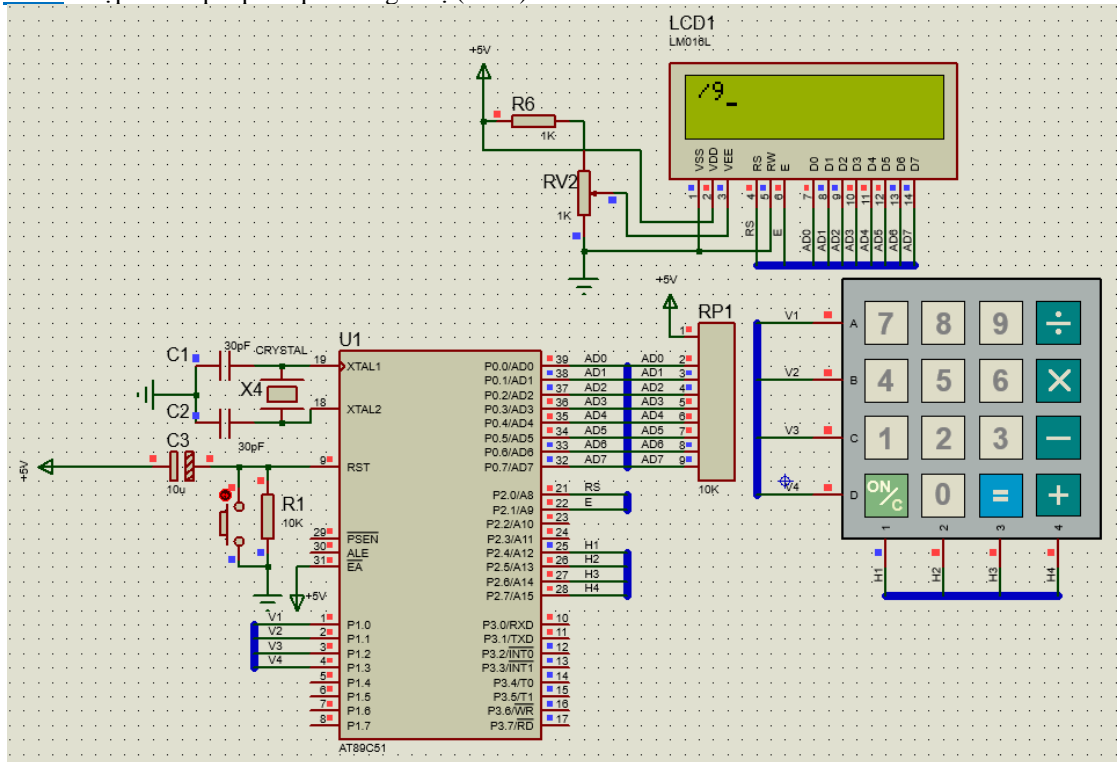
+ **TH4:** Thực hiện phép chia 2 số ($0 < x < 999$), có thể hiển thị đến 10 số sau dấu chấm thập phân:

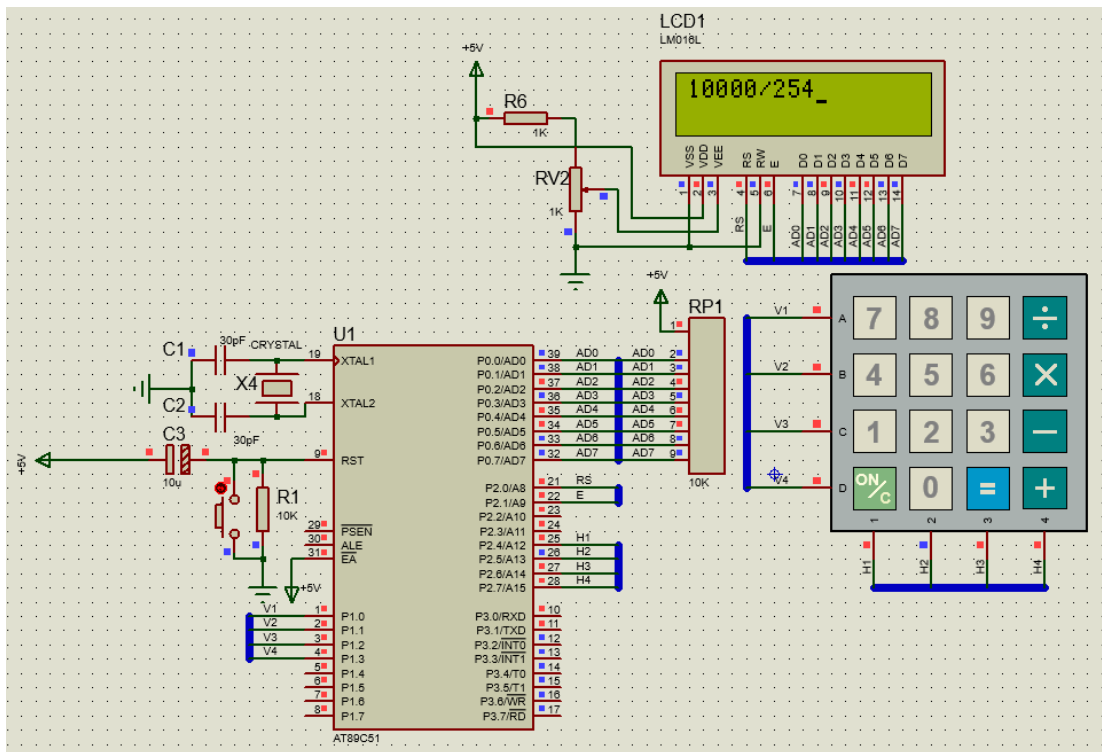
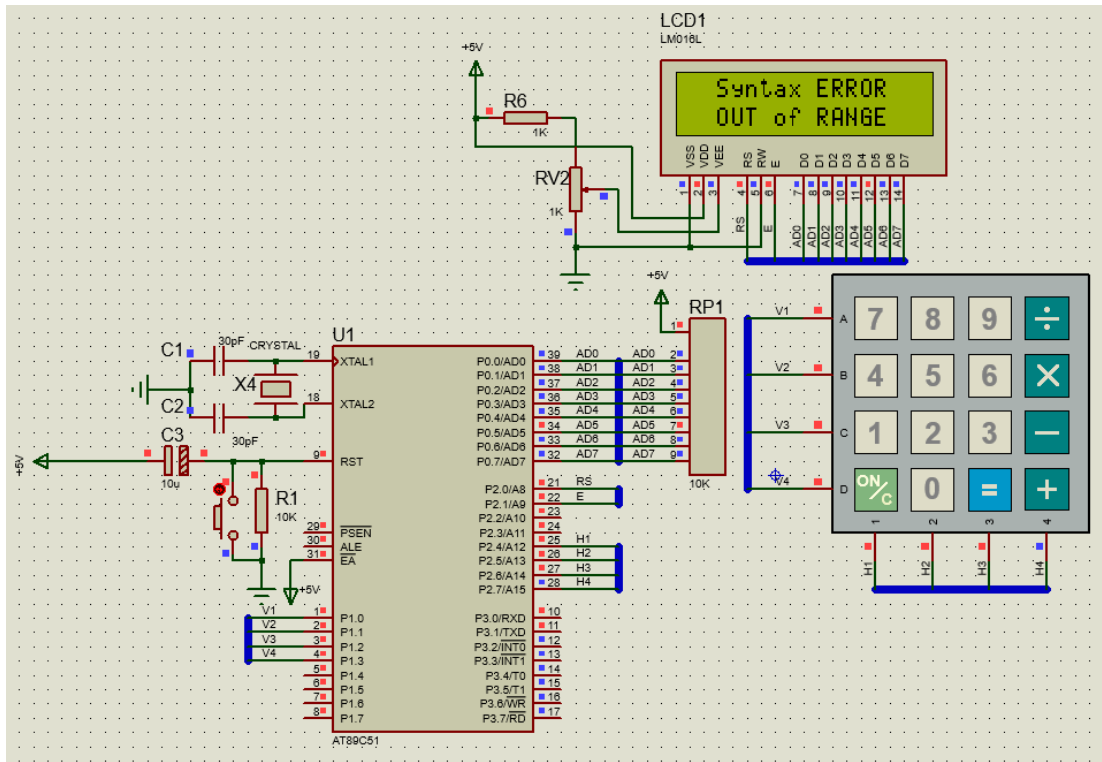


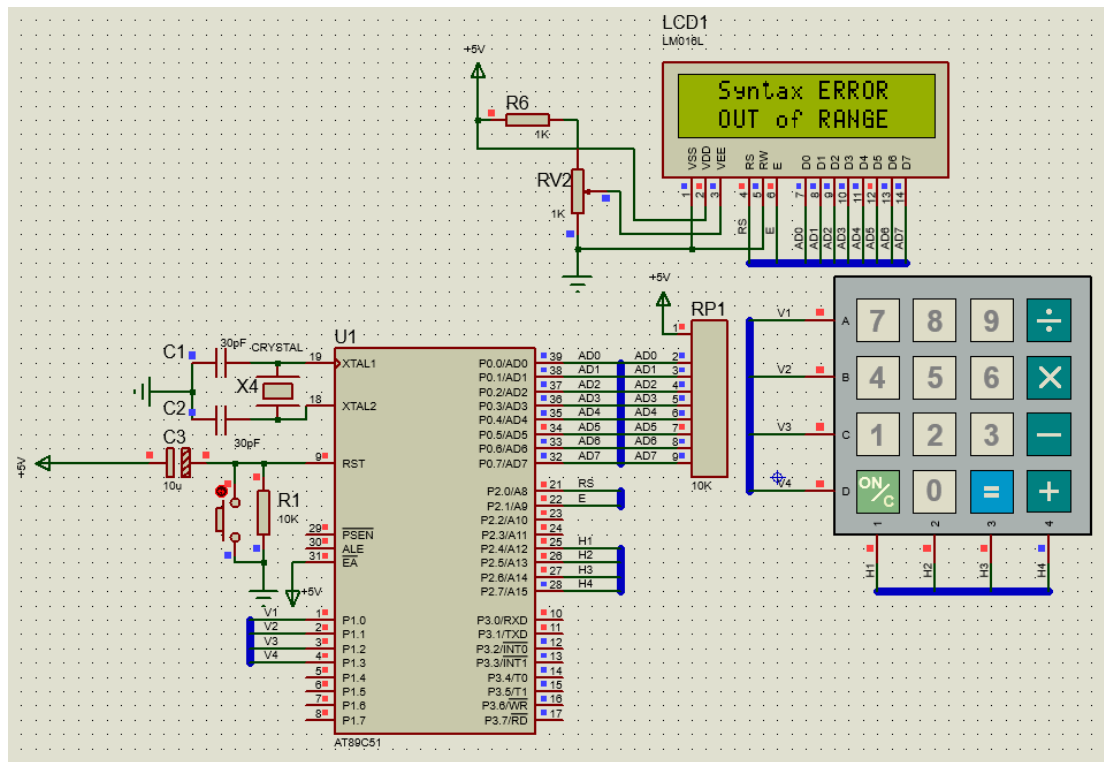
+ **TH5:** Chỉ nhập số hạng thứ nhất, không nhập toán tử rồi bấm dấu “=”:



+ **TH6:** Nhập sai cú pháp và quá tầm giá trị (>999):







- Như vậy với những kết quả mô phỏng thu được như trên thì Sơ đồ máy tính số học đơn giản của nhóm chúng em thiết kế đã đáp ứng được những yêu cầu đề bài đặt ra.
- Vì là lần đầu nhóm chúng em được tiếp xúc với lập trình MCU8051 nên không thể tránh được những sai sót, chúng em mong sẽ nhận được những phản hồi cũng như góp ý của thầy để chúng em có thể hoàn thiện hơn. Chúng em xin cảm ơn thầy!

— HẾT —