ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

ଌ★ଌ

BÁO CÁO THỰC HÀNH LAB 04

MÔN THỰC HÀNH VI XỬ LÍ –VI ĐIỀU KHIỂN

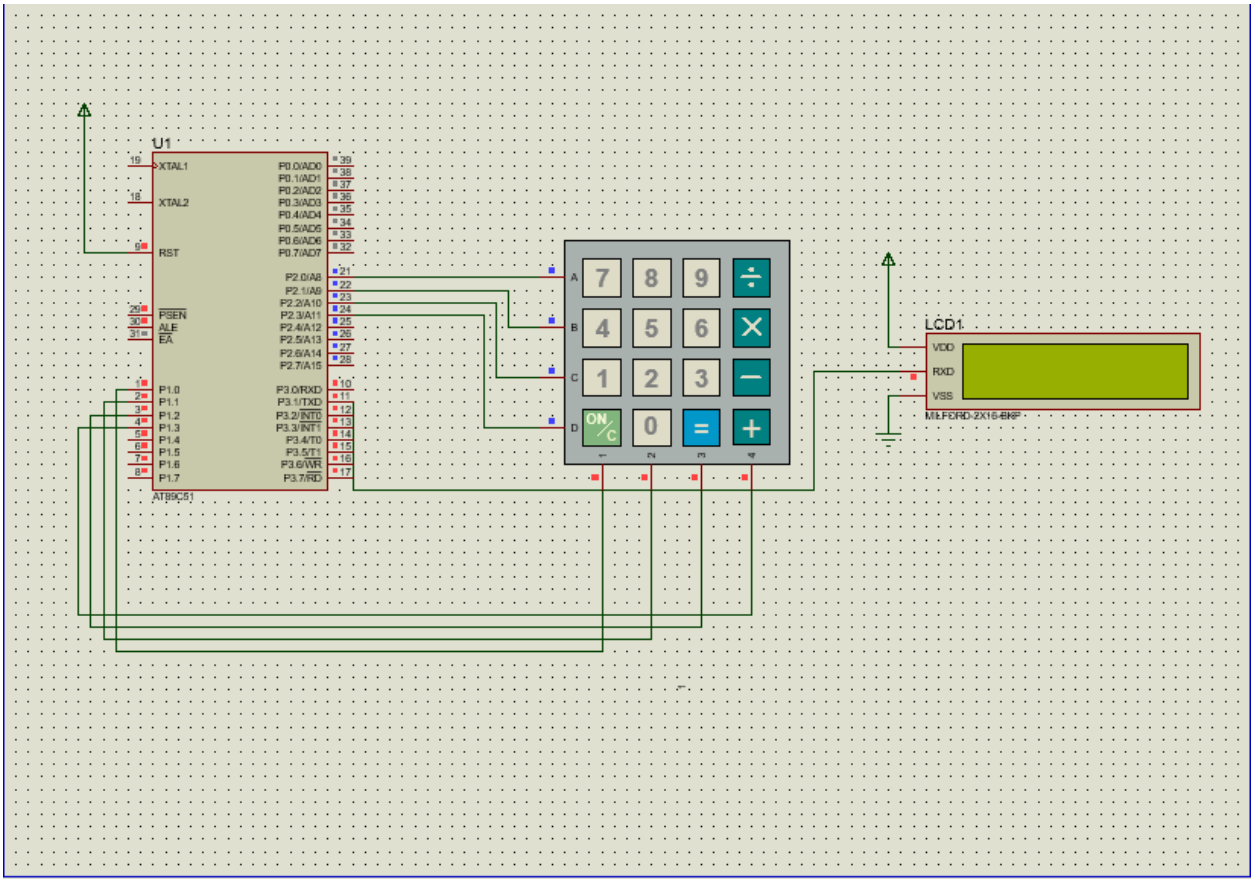
Họ và tên sinh viên : Nguyễn Hữu Tứ

Mã số sinh viên: 19522453

GIẢNG VIÊN HƯỚNG DẪN

Bùi Phùng Hữu Đức

TP. HỒ CHÍ MINH, 5/2021



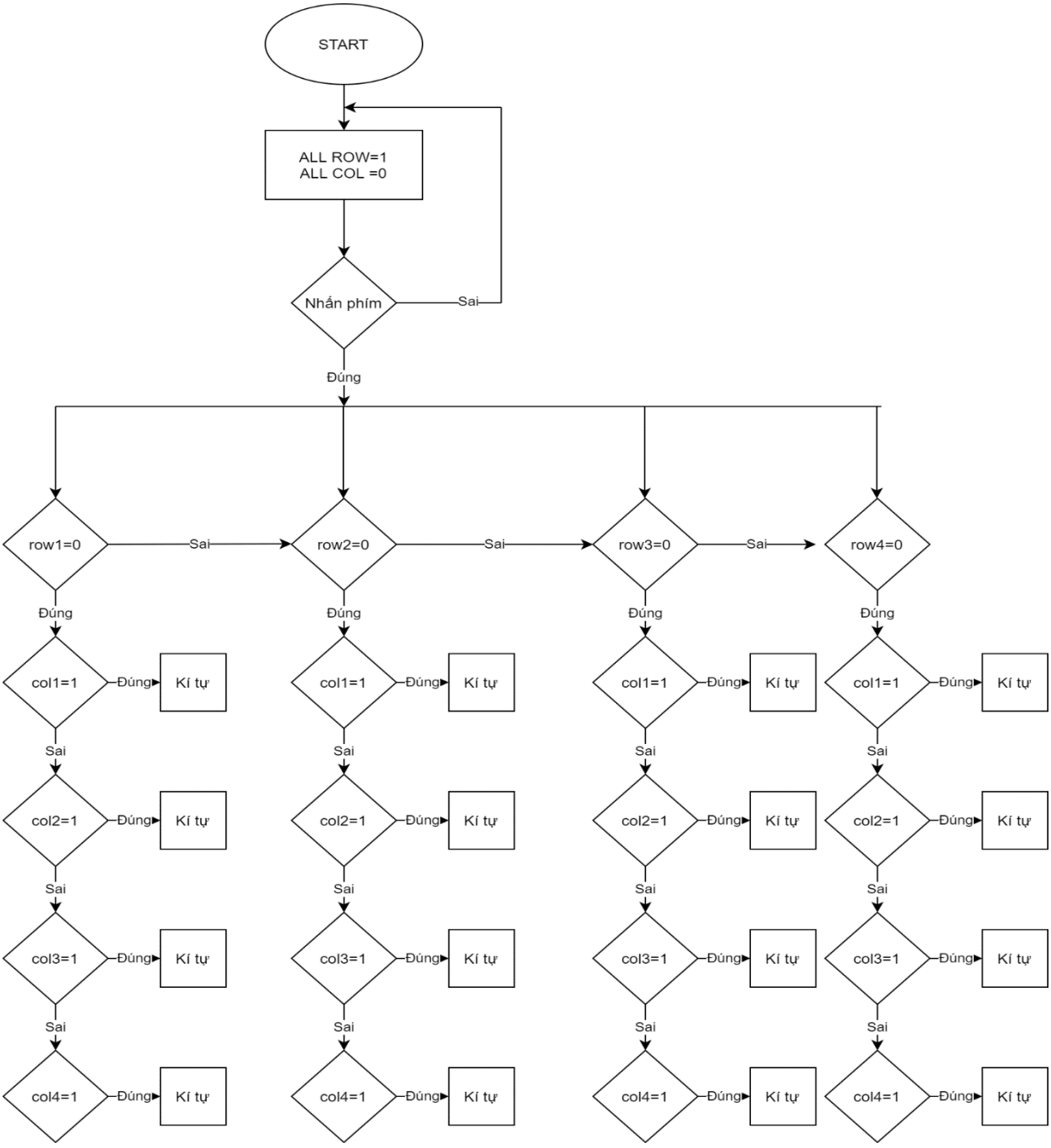
Hình1: Schematic

I. Cách sử dụng KeyPad:

-KeyPad dùng để đọc giá trị của người dùng nhập vào qua thuật toán quét phím .

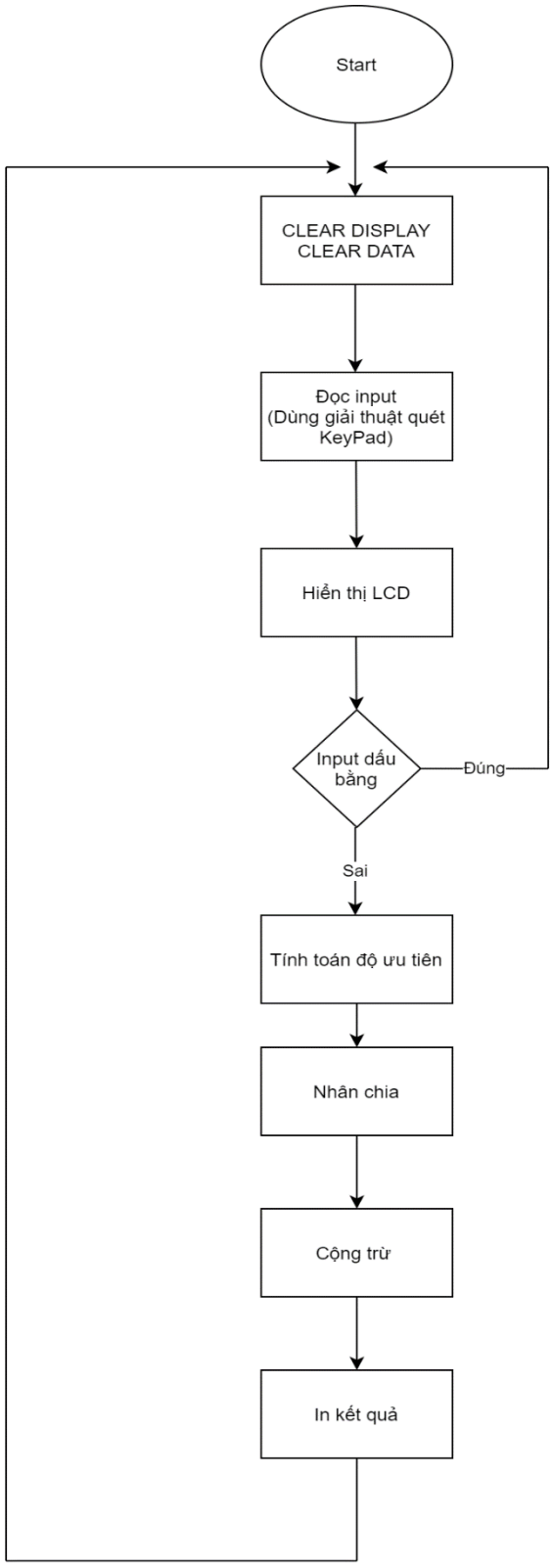
-Cách làm của thuật toán giống với cách tính tọa độ của trên mặt phẳng Oxy sẽ tính xem hàng nào cột nào, rồi sau đó xuất ra kí tự tương ứng với tọa độ đó.

- Flowchart để thực hiện thuật toán quét phím của Keypad :



Hình 1: Flowchart thể hiện thuật toán quét Keypad

II. Lưu đồ giải thuật của bài toán:



Hình 2: Lưu đồ giải thuật của bài toán

III. Phần giải thích code:

Phần code	Phần giải thích
X0 EQU 30H	Ta định nghĩa các toán tử,toán hạng vào các địa chỉ
X1 EQU 31H	trực tiếp trong ram
X2 EQU 32H	Các Xi là lưu chữ số
X3 EQU 33H	Các Opi là lưu dấu
X4 EQU 34H	
OP0 EQU 35H	
OP1 EQU 36H	
OP2 EQU 37H	
OP3 EQU 38H	
KQ EQU 39H	Lưu kết quả tính toán
SIGN EQU 40H	Lưu dấu
TEMP EQU 41H	Lưu giá trị tính toán tạm
CHU_SO BIT 0AH	Bit để phát hiện đó là chữ số
OPR_DAU BIT 0BH	Bit để phát hiện đó là dấu
DAU_BANG BIT 0CH	Bit để phát hiện đó là dấu bằng
;------------------------------	
ORG 0000H	
JMP MAIN	
MAIN:	
CALL XOA_MAN_HINH	Gọi hàm xóa màn hình
CALL INPUT	Gọi hàm nhập
CALL MUL_DIV; nhanchia	Gọi hàm nhân chia
CALL ADD_SUB ; cong tru	Gọi hàm cộng trừ
CALL OUTPUT	Gọi hàm xuất
JMP MAIN	Nhảy lại hàm Main để thực hiện vòng lặp
;------------------------------	Hàm xóa màn hình thì ta gán tất cả các số,dấu,..

;----------	Gán tất cả các biến bằng 0.
XOA_MAN_HINH:	
CLR A	Xóa cờ A

MOV X0,A
MOV X1,A
MOV X2,A
MOV X3,A
MOV X4,A
MOV OP0,A
MOV OP1,A
MOV OP2,A
MOV OP3,A
MOV KQ,A
MOV SIGN,A
MOV TEMP,A
CLR C
MOV CHU_SO,C
MOV OPR_DAU,C
MOV DAU_BANG,C
RET
	Các biến được gán cho A mà A=0

;----------	
CAL:	Hàm Cal dùng để phân loại xem đây toán tử gì
CJNE R7,#"+",N1	So sánh R7 với + nếu đúng thì thực hiện phép cộng, không thì nhảy tới nhãn N2
ADD A,B	A+B

MOV KQ,A	Lưu kết quả từ A vào KQ
MOV SIGN,#"+"	Lưu dấu + vào SIGN
RET	
N1:	So sánh R7 với - nếu đúng thì thực hiện phép cộng,
CJNE R7,#"-",N2	không thì nhảy tới nhãn N2
SUBB A,B	A-B
JC BU_2	Nhảy tới BU_2 nếu A=0, nghĩa là phép tính âm
MOV KQ,A	Lưu kết quả vào KQ
MOV SIGN,#"+"	Lưu dấu + vào SIGN
RET	
BU_2:	Hàm tính bù 2
CPL A	Đảo bit của A
INC A	Tiến hành +1
MOV KQ,A	Lưu kết quả vào KQ
MOV SIGN,#"-"	Lưu dấu – vào SIGN
RET	
N2:	So sánh R7 với * nếu đúng thì thực hiện phép cộng,
CJNE R7,#"*",N3	không thì nhảy tới nhãn N3
MUL AB	A nhân B
MOV KQ,A	Lưu kết quả vào KQ
MOV SIGN,#"+"	Lưu dấu + vào SIGN
RET	
N3:	So sánh R7 với / nếu đúng thì thực hiện phép cộng,
CJNE R7,#"/",N4	không thì nhảy tới nhãn N4
MOV TEMP,B	Lưu B vào biến tạm TEMP

DIV AB	A/B
MOV KQ,A	Phần nguyên lưu vào KQ
MOV A,#0AH	Cho A=10 mục đích tính tiếp phần sau phần dư
MUL AB	A*B
MOV B,TEMP	Biến temp gán giá trị cho B
DIV AB	A/B
MOV TEMP,A	Gán giá trị A vào temp
MOV SIGN,#"+"	Sign dc gán bằng dấu +
N4:	Thoát khi tới nhãn N4
RET	
;----------	Các hàm ưu tiên tính toán
MUL_DIV:	Ý tưởng: Thực hiện nhân chia trước
MOV R7, OP0	Gán giá trị OP0 là dấu đầu tiên cho R7
CJNE R7, #"/",TINH1	Sao sánh R7 với dấu / nếu không = nhảy tới TINH1,
TINH3:	bằng thì nhảy tới TINH3 để tính toán
MOV A,X0	Hàm TINH3 thực hiện tính toán và dời các toàn tử từ
MOV B,X1	sau lên trước, bản chất chỉ thực hiện 1 phép tính đầu
CALL CAL	với 2 số và 1 phép tính.
MOV X0, KQ	Kết quả tính toán được lưu vào X0
MOV OP0,OP1	Di chuyển giá trị của OP1->OP0
MOV X1,X2	Di chuyển giá trị của X2->X1
MOV OP1,OP2	Di chuyển giá trị của OP2->OP1
MOV X2,X3	Di chuyển giá trị của X3->X2
MOV OP2,OP3	Di chuyển giá trị của OP3->OP2
MOV X3,X4	Di chuyển giá trị của X4->X3
CLR A	Xóa A
MOV X4,A	Di chuyển giá trị A->X4,xóa toán tử thứ 5
MOV OP3,A	Di chuyển xóa dấu thứ 4

JMP MUL_DIV	Thực hiện lại vòng lặp để tiếp tục quét
TINH1:	Với ý tưởng trên ta thực hiện tương tự như dấu *
CJNE R7,#"*",TINH2	
JMP TINH3	
TINH2:	
MOV R7,OP1	
CJNE R7,#"/",TINH4	
TINH5:	
MOV A,X1	
MOV B,X2	Kết quả tính toán được lưu vào X1
CALL CAL	Di chuyển giá trị của OP1->OP0
MOV X1,KQ	Di chuyển giá trị của X2->X1
MOV OP1,OP2	Di chuyển giá trị của OP2->OP1
MOV X2,X3	Di chuyển giá trị của X3->X2
MOV OP2,OP3	Di chuyển giá trị của OP3->OP2
MOV X3,X4	Di chuyển giá trị của X4->X3
CLR A	Xóa A
MOV X4,A	Di chuyển giá trị A->X4,xóa toán tử thứ 5
MOV OP3,A	Di chuyển xóa dấu thứ 4
JMP TINH2	Thực hiện lại vòng lặp nhảy tới TINH2
TINH4:	Thực hiện tương tự cho đến khi nào hết toán tử thứ 5
CJNE R7,#"*",TINH6	và dấu thứ 4 thì dừng
JMP TINH5	
TINH6:	
MOV R7,OP2	
CJNE R7,#"/",TINH7	
TINH9:	
MOV A,X2	

```
MOV B,X3
CALL CAL
MOV X2,KQ
MOV OP2,OP3
MOV X3,X4
CLR A
MOV X4,A
MOV OP3,A
JMP TINH6
TINH7:
CJNE R7,#"*",TINH8
JMP TINH9
TINH8:
MOV R7,OP3
CJNE R7,#"/",TINH10
TINH12:
MOV A,X3
MOV B,X4
CALL CAL
CLR A
MOV X4,A
MOV OP3,A
RET
TINH10:
CJNE R7,#"*",TINH11
JMP TINH12
TINH11:
RET
```

;--------------------------------	Hàm ưu tiên tính dấu + và –
ADD_SUB:	Với ý tưởng tương tự hàm trên
MOV R7,OP0	OP0 được đưa vào R7
CJNE R7,#0,TINH15	So sánh R7 với 0 nếu khác thì nhảy tới TINH15
RET	Thoát
TINH15:	Hàm TINH15
MOV A,X0	Giá trị của X0->A
MOV B,X1	Giá trị của X0->A
CALL CAL	Hàm tính toán
MOV X0,KQ	Lưu KQ vào X0,thực hiện dịch các toán hạng tới trước
MOV OP0,OP1	Di chuyển giá trị OP1->OP0
MOV X1,X2	Di chuyển giá trị X2->X1
MOV OP1,OP2	Di chuyển giá trị OP2->OP1
MOV X2,X3	Di chuyển giá trị X3->X2
MOV OP2,OP3	Di chuyển giá trị OP3->OP2
MOV X3,X4	Di chuyển giá trị X4->X3
CLR A	Xóa A
MOV X4,A	Xóa phần tử đứng cuối
MOV OP3,A	Xóa dấu đứng cuối
JMP ADD_SUB	Nhảy lại hàm
;--------------------------------	
INPUT:	Hàm nhập
CALL LUU_SO	Hàm lưu số
MOV X0,A	Lưu số vào X0
CALL LUU_DAU	Hàm lưu dấu
MOV OP0,A	Lưu dấu vào OP0
CALL LUU_SO	Hàm lưu số
MOV X1,A	Lưu số vào X1

CALL LUU_DAU	Hàm lưu dấu
JB DAU_BANG,PASS	Lưu dấu vào OP1
MOV OP1,A	Hàm lưu số
CALL LUU_SO	Lưu số vào X2
MOV X2,A	Hàm lưu dấu
CALL LUU_DAU	Lưu dấu vào OP2
JB DAU_BANG,PASS	Nếu nhấn dấu = thì nhảy tới hàm PASS
MOV OP2,A	Lưu dấu vào X2
CALL LUU_SO	Hàm lưu dấu
MOV X3,A	Lưu số vào X3
CALL LUU_DAU	Tương tự
JB DAU_BANG,PASS	Nếu nhấn dấu = thì nhảy tới hàm PASS
MOV OP3,A	Lưu dấu vào OP3
CALL LUU_SO	Hàm lưu số
MOV X4,A	Lưu số vào X4
CALL LUU_DAU	Gọi hàm lưu dấu
JMP PASS	Nhảy tới hàm PASS
LOI:	Hàm lỗi error
CALL LOI_MESS	Gọi hàm hiển thị lỗi
LJMP MAIN	Nhảy lại tới hàm main
PASS:	Hàm Pass là thoát
RET	
;------------------------------	
LUU_SO:	Hàm lưu số
CALL QUET_KEYPAD	Đầu tiên quét KeyPad để nhận giá trị
CALL TOAN_TU	Gọi hàm toán tử để kiểm tra và lưu số
JB OPR_DAU,LOI	Kiểm tra bit dấu có phải hay không, nếu đúng thì lỗi
JB DAU_BANG,LOI	Gọi hàm kiển tra dấu bằng

CALL TRUYEN	Gọi hàm để truyền UART để hiện thị lên LCD
ANL A,#0FH	Xóa giá trị thừa ở A , ta xóa 4 bit có trọng số cao nhất
RET	Thoát
;----------	
LUU_DAU:	Hàm lưu dấu
CALL QUET_KEYPAD	Quét keypad nhận giá trị
CALL TOAN_TU	Gọi hàm toán tử để lưu và kiểm tra số
JB CHU_SO,LOI	Gọi hàm chứ số và kiểm tra lỗi
CALL TRUYEN	Gọi hàm truyền đi
RET	
;----------	
;----------	
OUTPUT:	Hàm xuất dữ liệu ra LCD
MOV R7,TEMP	Di chuyển giá trị temp -> R7
MOV R6,SIGN	Di chuyển giá trị SIGN -> R6
CJNE R7,#0,POINTED	So sánh R7 với 0
RETURN:	Hàm trả về
MOV A,KQ	A=KQ
MOV B,#100D	B=100
DIV AB	A/B
JZ LESSTEN	A=0 thì nhảy
ORL A,#30H	Or A với 30H
CALLTRUYEN	Gọi để truyền
MOV A,B	A=B
MOV B,#0AH	B=10

DIV AB	A/B
ORL A,#30H	Or A với 30H
CALL TRUYEN	Gọi truyền dữ liệu hiển thị
MOV A,B	A=B
ORL A,#30H	Or A với #30h
CALLTRUYEN	Gọi truyền dữ liệu hiển thị
JMP DONE	
LESSTEN:	
CJNE R6,#"-",DAU	
MOV A,#"-"	
CALL TRUYEN	
DAU:	
MOV A,B	
MOV B,#0AH	
DIV AB	
JZ LESSTEN1	
ORL A,#30H	
CALLTRUYEN	
LESSTEN1:	
MOV A,B	
ORL A,#30H	
CALL TRUYEN	
CJNE R7,#0,EXIT_OUT	
JMP DONE	

EXIT_OUT:
RET
POINTED:
CALL RETURN
MOV A,#"."
ACALL TRUYEN
MOV A,TEMP
ORL A,#30H
CALLTRUYEN
AJMP DONE
SIGNED:
MOV A,#"-"
CALLTRUYEN
JMP RETURN
DONE:
RET
	Hàm thực hiện in ra số float

;------------------------------- TOAN_TU: CJNE A,#"+", SUB AJMP FOUND_EXIT SUB: CJNE A,#"-", MULTI AJMP FOUND_EXIT MULTI: CJNE A,#"*", DIVI AJMP FOUND_EXIT	Hàm toán tử check các dấu,chữ số và dấu bằng để thực hiện in ra nếu có lỗi. So sanh với dấu + So sanh với dấu -

DIVI: CJNE A,#"/", EQUAL AJMP FOUND_EXIT	
EQUAL: CJNE A,#"=", EXIT CLR CHU_SO CLR OPR_DAU SETB DAU_BANG RET	Set bit dấu = lên 1 báo đây là dấu bằng
EXIT: SETB CHU_SO CLR OPR_DAU CLR DAU_BANG RET	Set bit của chữ số lên 1 báo đấy là chữ số
FOUND_EXIT: CLR CHU_SO SETB OPR_DAU CLR DAU_BANG RET	Set bit của dấu lên 1 báo đấy là dấu

;----------
QUET_KEYPAD:
MOV P1,#11111111B
S1: MOV P2,#0
MOV A,P1
ANL A,#00001111B
 	
CJNE A,#00001111B,S2
SJMP S1
	Thực hiện giải thuật quét KeyPad xác định tọa đó của phím nhấn.
	Cho các cột đều bằng 1
	Cho các hàng =0
	Di chuyển giá trị của P1->A
	Xóa 4 bit có trọng số cao nhất vì 4 bit này không dùng tới
	So sánh A với 00001111B, nếu khác thì có nghĩa có phím đã nhấn thì nhảy tới S2, không quay lại S1

S2: CALL DELAY_20MS	Gọi hàm delay 20ms
MOV A,P1	Di chuyển dữ liệu từ P1->A
ANL A,#00001111B	Xóa bỏ bớt 4 bit đầu do không cần sử dụng tới
CJNE A,#00001111B,OUT	So sánh A với 00001111H nếu khác nhảy tới OUT
SJMP S1	Nếu bằng thì nhảy lại S1
OUT:CALL DELAY_20MS	Gọi hàm Delay 20ms
MOV A,P1	Di chuyển dữ liệu từ P1->A
ANL A,#00001111B	Xóa bỏ bớt 4 bit đầu do không cần sử dụng tới
CJNE A,#00001111B,OUT1	So sánh A với 00001111H nếu khác nhảy tới OUT1
OUT1:MOV P2,#11111110B	Hàm OUT1 kiểm tra các hàng
MOV A,P1	Di chuyển dữ liệu từ P1->A
ANL A,#00001111B	Xóa 4 bit không cần thiết
CJNE A,#00001111B,ROW_0	So sánh nếu A khác với 00001111H thì nhảy tới ROW_0
MOV P2,#11111101B	Nếu bằng qua check hàng tiếp theo
MOV A,P1	Di chuyển dữ liệu từ P1->A
ANL A,#00001111B	Xóa 4 bit không cần thiết
CJNE A,#00001111B,ROW_1	So sánh nếu A khác với 00001111H thì nhảy tới ROW_1
MOV P2,#11111011B	Nếu bằng qua check hàng tiếp theo
MOV A,P1	Di chuyển dữ liệu từ P1->A
ANL A,#00001111B	Xóa 4 bit không cần thiết
CJNE A,#00001111B,ROW_2	So sánh nếu A khác với 00001111H thì nhảy tới ROW_2
MOV P2,#11110111B	Nếu bằng qua check hàng tiếp theo
MOV A,P1	Di chuyển dữ liệu từ P1->A

ANL A,#00001111B	Xóa 4 bit không cần thiết
CJNE A,#00001111B,ROW_3	So sánh nếu A khác với 00001111H thì nhảy tới ROW_3
LJMP S2	Nhảy tới lại hàm S2
ROW_0:	Hàm ROW_0
MOV DPTR,#ROW0	Gán địa chỉ bảng đầu của mảng ROW0 cho DPTR
JMP KT	Nhảy tới hàm KT
ROW_1:	Hàm ROW_1
MOV DPTR,#ROW1	Gán địa chỉ bảng đầu của mảng ROW1 cho DPTR
JMP KT	Nhảy tới hàm KT
ROW_2:	Hàm ROW_2
MOV DPTR,#ROW2	Gán địa chỉ bảng đầu của mảng ROW2 cho DPTR
JMP KT	Nhảy tới hàm KT
ROW_3:	Hàm ROW_3
MOV DPTR,#ROW3	Gán địa chỉ bảng đầu của mảng ROW3 cho DPTR
JMP KT	Nhảy tới hàm KT
KT:	Hàm KT
RRC A	Quay phải A với Cy là bit nhỏ nhất
JNC FOUND	Nếu C=0 thì nhảy tới hàm FOUND
INC DPTR	Không thì tăng DPTR+1
JMP KT	Nhảy lại tới hàm KT
FOUND:	Hàm FOUND
CLR A	Xóa A
MOVC A,@A+DPTR	Lấy giá trị từ mảng sang cho A
CJNE A,#"X",F1	Nếu là phím AC thì xóa màn hình
CALL XOA_MAN_HINH_CMD	Hàm xóa màn hình
F1:	Hàm F1
RET	Thoát

;----------	
LOI_MESS:	Hàm in ra lỗi
MOV DPTR,#LOIOR	
E1:	
CLR A	
MOVC A,@A+DPTR	
CALL TRUYEN	
CALL DELAY_20MS	
INC DPTR	
JZ E2	
SJMP E1	
E2:	
CALL DELAY_20MS	
CALL XOA_MAN_HINH_CMD	
RET	
;----------	
XOA_MAN_HINH_CMD:	Hàm xóa màn hình
MOV A,#254D	
CALL TRUYEN	
MOV A,#1D	
CALL TRUYEN	
LJMP MAIN	
RET	
;----------	
DELAY_20MS:	Hàm tính Delay 20ms
MOV R0,#4	
LOOP:	
MOV TMOD,#01H	

MOV TH0,#HIGH(-5000)
MOV TL0,#LOW(-5000)
SETB TR0
HERE1:
JNB TF0,HERE1
CLR TF0
CLR TR0
DJNZ R0,LOOP
RET
;----------
TRUYEN:
MOV TMOD,#20H
MOV TH1,#0FDH
MOV SCON,#50H
SETB TR1
MOV SBUF,A
HERE:
JNB TI,HERE
CLR TI
RET
;----------
ORG 300H
LOIOR: DB "LOIOR",0
XOA_MAN_HINH_LCD: DB 254D,01D
ROW0: DB "7","8","9","/"
ROW1: DB "4","5","6","*"
ROW2: DB "1","2","3","-"
ROW3: DB "X","0","=","+"
	Hàm truyền sử dụng UART
	Phần khai báo mảng và chuỗi

END ;;----------	