PRACTICE EXERCISES OF THE MICROPROCESSORS & MICROCONTROLLERS

Instructor: The Tung Than

Student's name: Nguyen Quoc Truong An

Student code: 21521810

PRACTICE REPORT NO 4

LAB4: USING UART

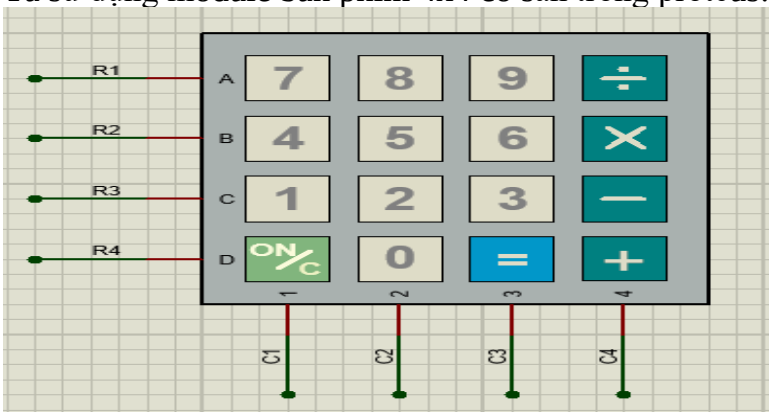
I. Student preparation

- Knowledge of how to install and use UART.
- Knowledge of how to use Serial UART LCD

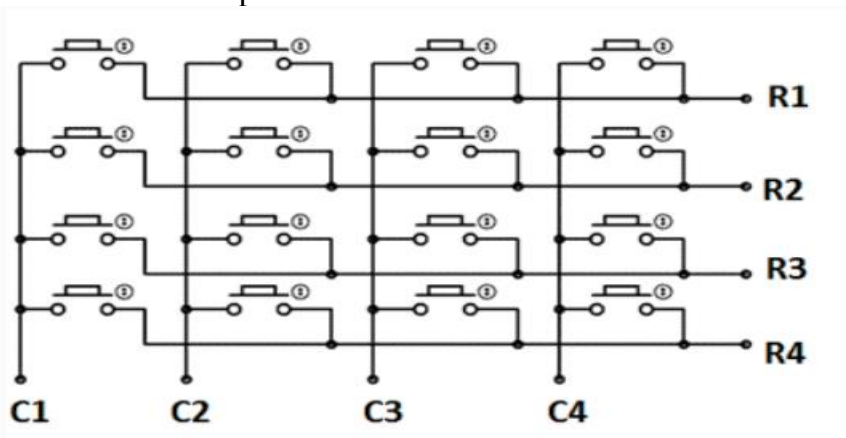
II. Practice content

1) Design a 4x4 keyboard set including the following buttons:
   - From 0 to 9
   - The sign + - * /
   - Sign =
   - Reset button

Ta sử dụng module bàn phím 4x4 có sẵn trong proteus:

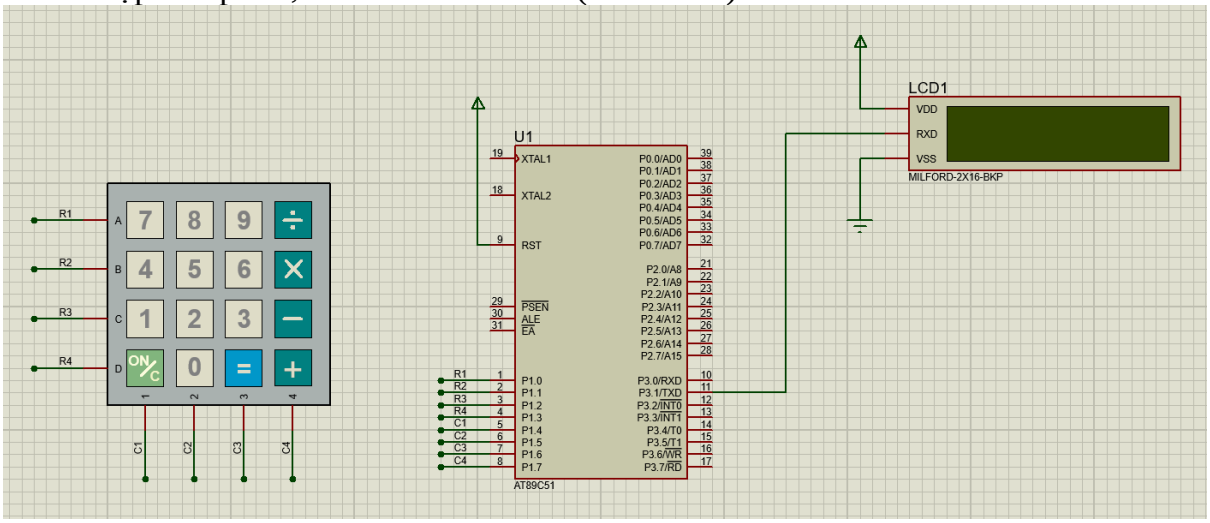


Schematic của bàn phím trên:



2) Using AT89C51/AT89C52 in combination with the module just designed above to design a handheld calculator, display calculations and results on an LCD that receives data via UART.

Ta kết hợp bàn phím, vi điều khiển 8051 (AT89C51) và LCD như sau:



Chương trình đầy đủ thực hiện chức năng tính toán: cộng (+), trừ (-), nhân (x), chia lấy nguyên (%), phím '=' để tính toán kết quả, phím 'ON/C' để thực hiện xóa toàn bộ màn hình LCD:

```
$NOMOD51
$INCLUDE (8051.MCU)

;=====================================================================
; DEFINITIONS
ROW1 BIT P1.0
ROW2 BIT P1.1
ROW3 BIT P1.2
ROW4 BIT P1.3
COL1 BIT P1.4
COL2 BIT P1.5
COL3 BIT P1.6
COL4 BIT P1.7

;=====================================================================
; VARIABLES
 KEY_CODE        EQU 34H
 COL             EQU 32H

;=====================================================================
; RESET and INTERRUPT VECTORS
; Reset Vector
ORG 0000H
JMP START

;=====================================================================
; CODE SEGMENT
ORG 0100H
```

```
; KHAI BAO MANG KY TU TRONG KEYPAD
CHAR_CODE: DB '7','8','9','%','4','5','6','x','1','2','3','-','C','0','=','+'

START:
   MOV TMOD, #20H        ; TIMER1 CHE DO 2
   MOV TH1, #0FDH        ; NAP TIMER1 TOC DO BAUD 9600
   MOV SCON, #50H        ; TRUYEN DU LIEU O CHE DO 1, BAT CHO PHEP TRUYEN
   MOV R0, #38H          ; R0 LUU DIA CHI BAT DAU CUA PHEP TOAN

LOOP:
   MOV R1, #0            ; TOAN HANG 1
   MOV R2, #0            ; TOAN TU
   MOV R3, #0            ; TOAN HANG 2
   MOV R4, #0            ; HANG CHUC KQ
   MOV R5, #0            ; HANG DON VI KQ
   MOV R6, #0            ; DANH DAU KET QUA AM

   CLR ROW1
   CLR ROW2
   CLR ROW3
   CLR ROW4

   JNB COL1, SCAN        ; KIEM TRA COT 1 = 0 ?, NEU CO NHAY VAO QUET PHIM DE TIM HANG TUONG UNG
  DUOC BAM
   JNB COL2, SCAN        ; KIEM TRA COT 2 = 0 ?, ...
   JNB COL3, SCAN        ; KIEM TRA COT 3 = 0 ?, ...
   JNB COL4, SCAN        ; KIEM TRA COT 4 = 0 ?, ...

JMP LOOP

SCAN:
   CALL SCAN_KEYPAD      ; QUET PHIM
   JNB COL1, $           ; CHONG DOI PHIM COT 1
   JNB COL2, $           ; CHONG DOI PHIM COT 2
   JNB COL3, $           ; CHONG DOI PHIM COT 3
   JNB COL4, $           ; CHONG DOI PHIM COT 4

   MOV @R0, KEY_CODE     ; LUU KY TU QUET DUOC VAO PHEP TOAN
   DEC R0

   CLR C                 ; XU LY BAM NUT 'C'
   MOV A, KEY_CODE
   CALL NUM_TO_CHAR
   SUBB A, #'C'
   JZ CLEAR_SCREEN       ; XOA TOAN BO MAN HINH LCD

   MOV A, KEY_CODE       ; HIEN THI KY TU BAM LEN MAN HINH LCD
   CALL DISPLAY_LCD

   CLR C                 ; XU LY BAM NUT '='
   MOV A, KEY_CODE
   CALL NUM_TO_CHAR
   SUBB A, #'='
   JZ CALCULATE          ; THUC HIEN TINH TOAN

JMP LOOP
```

```
CALCULATE:
  MOV A, 38H            ; TOAN HANG 1
  CALL NUM_TO_CHAR
  CLR C
  SUBB A, #'0'          ; CHUYEN CHAR THANH SO
  MOV R1, A             ; LUU TOAN HANG 1

  MOV A, 36H            ; TOAN HANG 2
  CALL NUM_TO_CHAR
  CLR C
  SUBB A, #'0'          ; CHUYEN CHAR THANH SO
  MOV R3, A             ; LUU TOAN HANG 2

  MOV A, 37H            ; TOAN TU '+'
  CALL NUM_TO_CHAR
  MOV R2, A
  CLR C
  SUBB A, #'+'          ; NEU LA TOAN TU '+', NHAY DEN NHAN THUC HIEN PHEP '+'
  JZ CAL_ADD

  MOV A, R2             ; TOAN TU '-'
  CLR C
  SUBB A, #'-'          ; NEU LA TOAN TU '-', NHAY DEN NHAN THUC HIEN PHEP '-'
  JZ CAL_SUB

  MOV A, R2             ; TOAN TU 'x'
  CLR C
  SUBB A, #'x'          ; NEU LA TOAN TU 'x', NHAY DEN NHAN THUC HIEN PHEP 'x'
  JZ CAL_MUL

  MOV A, R2             ; TOAN TU '%'
  CLR C
  SUBB A, #'%'          ; NEU LA TOAN TU '%', NHAY DEN NHAN THUC HIEN PHEP '%'
  JZ CAL_DIV

  CAL_ADD:              ; TINH TOAN PHEP '+'
    MOV A, R1
    ADD A, R3           ; KQ = a + b
    JMP DONE

  CAL_SUB:              ; TINH TOAN PHEP '-'
    MOV A, R1
    CLR C
    SUBB A, R3          ; KQ = a - b
    JNC SUB_DONE        ; KIEM TRA XEM KET QUA CO AM

    MOV R6, #1
    MOV A, R3
    CLR C
    SUBB A, R1          ; KQ = b - a
    JZ SUB_DONE
  SUB_DONE:
    JMP DONE

    CAL_MUL:            ; TINH TOAN PHEP 'x'
    MOV A, R1
    MOV B, R3
    MUL AB              ; KQ = a x b
    JMP DONE
```

```
  CAL_DIV:                ; TINH TOAN PHEP '%'
    MOV A, R3
    JZ ERROR_DISPLAY      ; HIEN THI MATH ERROR NEU A%B KHI B = 0
    MOV A, R1
    MOV B, R3
    DIV AB                ; KQ = a % b , CHIA LAY PHAN NGUYEN
  DONE:
    CALL SPLIT_BCD        ; TACH KET QUA RA 2 HANG CHUC, HANG DON VI
    JMP WRITE_RESULT      ; HIEN THI KET QUA RA MAN HINH LCD

    CLEAR_SCREEN:         ; GOI HAM XOA MAN HINH LCD
    CALL CLEAR_LCD

  JMP LOOP

  ERROR_DISPLAY:          ; GOI HAM HIEN THI "MATH ERROR"
    CALL ERROR_DISPLAY_FUNCT

    JMP EXIT_CAL          ; NHAY TOI KET THUC VIEC TINH TOAN

  WRITE_RESULT:           ; GOI HAM HIEN THI KET QUA LEN MAN HINH LCD
    CALL WRITE_RESULT_FUNCT

  EXIT_CAL:               ; KET THUC VIEC TINH TOAN

JMP LOOP

 ;============================SUB-PROGRAM==========================
SCAN_KEYPAD:              ; HAM QUET PHIM
  CLR ROW1                ; QUET HANG 1
  SETB ROW2
  SETB ROW3
  SETB ROW4
  CLR C
  CALL CHECK_COL
  MOV A, COL
  JZ CHECK_ROW2
  SUBB A, #1              ; 0, 1, 2, 3
  MOV KEY_CODE, A
  JMP EXIT

CHECK_ROW2:               ; QUET HANG 2
  SETB ROW1
  CLR ROW2
  SETB ROW3
  SETB ROW4

  CALL CHECK_COL          ; KIEM TRA COT TUONG UNG
  MOV A, COL

  JZ CHECK_ROW3

  ADD A, #3               ; 4, 5, 6, 7
  MOV KEY_CODE, A
  JMP EXIT

CHECK_ROW3:               ; QUET HANG 3
  SETB ROW1
  SETB ROW2
  CLR ROW3
  SETB ROW4
```

```
  CALL CHECK_COL          ; KIEM TRA COT TUONG UNG
  MOV A, COL
  JZ CHECK_ROW4
  ADD A, #7               ; 8, 9, 10, 11
  MOV KEY_CODE, A
  JMP EXIT

CHECK_ROW4:               ; QUET HANG 4
  SETB ROW1
  SETB ROW2
  SETB ROW3
  CLR ROW4

  CALL CHECK_COL          ; KIEM TRA COT TUONG UNG
  MOV A, COL
  JZ EXIT
  ADD A, #11              ; 12, 13, 14, 15
  MOV KEY_CODE, A

EXIT:
RET

CHECK_COL:                ; HAM KIEM TRA COT DUOC BAM
  JB COL1, CHECK_COL2     ; KIEM TRA COT 1
  MOV COL, #1
  JMP FINISH

CHECK_COL2:               ; KIEM TRA COT 2
  JB COL2, CHECK_COL3
  MOV COL, #2
  JMP FINISH

CHECK_COL3:               ; KIEM TRA COT 3
  JB COL3, CHECK_COL4
  MOV COL, #3
  JMP FINISH

CHECK_COL4:               ; KIEM TRA COT 4
  JB COL4, NO_COL
  MOV COL, #4
  JMP FINISH

NO_COL:
  MOV COL, #0             ; KHONG CO COT NAO DUOC BAM
FINISH:
RET

DISPLAY_LCD:              ; HAM HIEN THI PHIM BAM LEN MAN HINH LCD
  CALL NUM_TO_CHAR        ; CHUYEN KY TU BAM TUONG UNG TU BAN PHIM THANH CHAR
  CALL WRITE_LCD          ; HIEN THI KY TU LEN MAN HINH LCD
RET

WRITE_LCD:                ; HAM HIEN THI KY TU LEN MAN HINH LCD
  SETB TR1

  MOV SBUF, A             ; GHI DU LIEU CAN TRUYEN LEN THANH GHI SBUF
  JNB TI, $               ; DOI TRUYEN XONG
  CLR TI
  CLR TR1
RET
```

```
NUM_TO_CHAR:                    ; HAM CHUYEN KY TU BAM TUONG UNG TU BAN PHIM THANH CHAR
  MOV DPTR, #CHAR_CODE
  MOVC A, @A+DPTR
RET

RESET_DATA:                     ; HAM RESET LAI DU LIEU VUNG LUU PHEP TINH
  MOV 38H, #0
  MOV 37H, #0
  MOV 36H, #0
  MOV R0, #38H                  ; GAN LAI R0 O DAU PHEP TINH
RET

SPLIT_BCD:                      ; HAM TACH KET QUA THANH HANG CHUC, HANG DON VI
  MOV B, #10
  DIV AB
  MOV R4, A
  MOV R5, B
RET

CLEAR_LCD:                      ; HAM XOA TOAN BO MAN HINH LCD
  MOV A, #0FEH
  CALL WRITE_LCD
  MOV A, #01H
  CALL WRITE_LCD
  CALL RESET_DATA
RET

DELAY:                          ; HAM DELAY TRONG KHOANG 5MS
    MOV R0, #10
  LOOP_DELAY1:
    MOV R1, #250
  LOOP_DELAY2:
    DJNZ R1, LOOP_DELAY2
    DJNZ R0, LOOP_DELAY1
RET

ERROR_DISPLAY_FUNCT:            ; HAM HIEN THI 'MATH ERROR' LEN MAN HINH LCD
  CALL CLEAR_LCD                ; XOA MAN HINH LCD TRUOC KHI HIEN THI 'MATH ERROR'
  CALL DELAY                    ; DELAY 5MS
  MOV A, #'M'                   ; HIEN THI KY TU 'M'
  CALL WRITE_LCD
  MOV A, #'A'                   ; ... 'A'
  CALL WRITE_LCD
  MOV A, #'T'                   ; ... 'T'
  CALL WRITE_LCD
  MOV A, #'H'                   ; ... 'H'
  CALL WRITE_LCD
  MOV A, #' '                   ; ... ' '
  CALL WRITE_LCD
  MOV A, #'E'                   ; ... 'E'
  CALL WRITE_LCD
  MOV A, #'R'                   ; ... 'R'
  CALL WRITE_LCD
  MOV A, #'R'                   ; ... 'R'
  CALL WRITE_LCD
  MOV A, #'O'                   ; ... 'O'
  CALL WRITE_LCD
  MOV A, #'R'                   ; ... 'R'
```

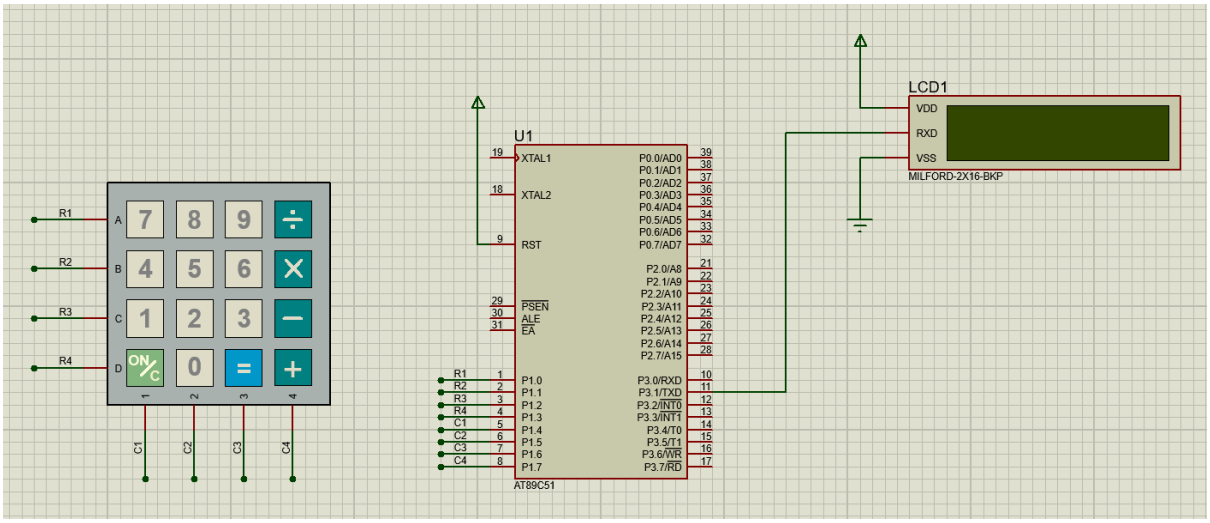
```
  CALL WRITE_LCD
RET

WRITE_RESULT_FUNCT:          ; HAM HIEN THI KET QUA LEN MAN HINH LCD
  MOV A, R6                  ; KIEM TRA DAU '-' TRONG KET QUA VA HIEN THI NEU CO
  JZ NOT_NEG
  MOV A, #'-'
  CALL WRITE_LCD
NOT_NEG:
  MOV A, R4                  ; NEU KET QUA LA '0N' THI CHI HIEN THI 'N'
  JZ HC_ZERO
  ADD A, #'0'
  CALL WRITE_LCD             ; HIEN THI HANG CHUC
HC_ZERO:
  MOV A, R5
  ADD A, #'0'
  CALL WRITE_LCD             ; HIEN THI HANG DON VI
RET

END
```

III. Report

1) Design result (screenshot and pasted in the report). **(1 point)**



2) Explain the operating principle of the effects, accompanied by a video (send a Google Drive link) to demonstrate the circuit operation in case the instructor cannot run the design file. **(3 points)**

*** Google Drive link:**

https://drive.google.com/drive/folders/1qUYuJ_4o-ohXSD7gFYcQVBCwcztED-Le?usp=sharing

****Nguyên tắc hoạt động của mạch trên proteus:**

-Module bàn phím 4x4 kết nối với vi điều khiển 8051 thông qua port 1. Các chân P1.0-P1.3 tương ứng với row1-row4 của bàn phím. Các chân P1.4-P1.7 tương ứng với col1-col4 của bàn phím.

-Chân P3.1/TXD được nối với chân RXD của module LCD để thực hiện truyền dữ liệu nối tiếp.

-Ta cấp nguồn 5V và nối đất cho vi điều khiển 8051 và module LCD.

*****Giải thích chi tiết code:**

***Các khai báo và chương trình vòng lặp chính:**

```
$NOMOD51
$INCLUDE (8051.MCU)

;===================================================================
; DEFINITIONS
ROW1 BIT P1.0
ROW2 BIT P1.1
ROW3 BIT P1.2
ROW4 BIT P1.3
COL1 BIT P1.4
COL2 BIT P1.5
COL3 BIT P1.6
COL4 BIT P1.7

;===================================================================
; VARIABLES
 KEY_CODE          EQU 34H
 COL               EQU 32H

;===================================================================
; RESET and INTERRUPT VECTORS
; Reset Vector
ORG 0000H
JMP START

;===================================================================
; CODE SEGMENT
ORG 0100H
```

-Ta định nghĩa các pin tương ứng với tên hàng, tên cột

-Khai báo biến KEY_CODE để lưu trữ phím sau khi quét phím tìm ra

-Khai báo biến COL để lưu giá trị cột

```
; KHAI BAO MANG KY TU TRONG KEYPAD
CHAR_CODE: DB '7','8','9','%','4','5','6','x','1','2','3','-','C','0','=','+'

START:
  MOV TMOD, #20H      ; TIMER1 CHE DO 2
  MOV TH1, #0FDH      ; NAP TIMER1 TOC DO BAUD 9600
  MOV SCON, #50H      ; TRUYEN DU LIEU O CHE DO 1, BAT CHO PHEP TRUYEN
  MOV R0, #38H        ; R0 LUU DIA CHI BAT DAU CUA PHEP TOAN
```

-Ta khai báo mảng ký tự quy định các phím bấm trên bàn phím 4x4

-Chọn timer1 chế độ 2

-Cài đặt tốc độ Baud là 9600

-Lựa chọn truyền thông nối tiếp ở chế độ 1, bật cho phép nhận dữ liệu

-R0 lưu địa chỉ đầu tiên của phép tính

```
LOOP:
  MOV R1, #0            ; TOAN HANG 1
  MOV R2, #0            ; TOAN TU
  MOV R3, #0            ; TOAN HANG 2
  MOV R4, #0            ; HANG CHUC KQ
  MOV R5, #0            ; HANG DON VI KQ
  MOV R6, #0            ; DANH DAU KET QUA AM

  CLR ROW1
  CLR ROW2
  CLR ROW3
  CLR ROW4

  JNB COL1, SCAN      ; KIEM TRA COT 1 = 0 ?, NEU CO NHAY VAO QUET PHIM DE TIM HANG TUONG UNG
 DUOC BAM
  JNB COL2, SCAN      ; KIEM TRA COT 2 = 0 ?, ...
  JNB COL3, SCAN      ; KIEM TRA COT 3 = 0 ?, ...
  JNB COL4, SCAN      ; KIEM TRA COT 4 = 0 ?, ...

JMP LOOP
```

-Khai báo các thiết lập ban đầu

-Gán tất cả các hàng ở mức 0

-Thực hiện kiểm tra xem có nút nào được bấm, nếu có, nhảy tới SCAN để gọi hàm quét phím, nếu không quay lại LOOP kiểm tra các nút bấm.

```
SCAN:
  CALL SCAN_KEYPAD   ; QUET PHIM
  JNB COL1, $        ; CHONG DOI PHIM COT 1
  JNB COL2, $        ; CHONG DOI PHIM COT 2
  JNB COL3, $        ; CHONG DOI PHIM COT 3
  JNB COL4, $        ; CHONG DOI PHIM COT 4

  MOV @R0, KEY_CODE  ; LUU KY TU QUET DUOC VAO PHEP TOAN
  DEC R0

  CLR C              ; XU LY BAM NUT 'C'
  MOV A, KEY_CODE
  CALL NUM_TO_CHAR
  SUBB A, #'C'
  JZ CLEAR_SCREEN    ; XOA TOAN BO MAN HINH LCD

  MOV A, KEY_CODE    ; HIEN THI KY TU BAM LEN MAN HINH LCD
  CALL DISPLAY_LCD

  CLR C              ; XU LY BAM NUT '='
  MOV A, KEY_CODE
  CALL NUM_TO_CHAR
  SUBB A, #'='
  JZ CALCULATE       ; THUC HIEN TINH TOAN

JMP LOOP
```

-Gọi hàm quét phím

-Thực hiện chống dội phím bằng cách nhảy tại chỗ nếu phím bấm còn đang được nhấn giữ

-Gán ký tự vừa quét được vào phép tính

-Thực hiện kiểm tra nếu phím bấm là ‘C’ tức ứng với ‘ON/C’ trên bàn phím thì thực hiện nhảy tới nhãn CLEAR_SCREEN để xóa màn hình LCD.

-Nếu không phải ký tự ‘C’ thì hiển thị ký tự vừa bấm lên màn hình

-Kiểm tra xem ký tự vừa bấm có phải là ‘=’, nếu phải thì nhảy tới nhãn CALCULATE để thực hiện tính toán.

-Nhảy lặp lại LOOP.

```
CALCULATE:
  MOV A, 38H            ; TOAN HANG 1
  CALL NUM_TO_CHAR
  CLR C
  SUBB A, #'0'          ; CHUYEN CHAR THANH SO
  MOV R1, A             ; LUU TOAN HANG 1

  MOV A, 36H            ; TOAN HANG 2
  CALL NUM_TO_CHAR
  CLR C
  SUBB A, #'0'          ; CHUYEN CHAR THANH SO
  MOV R3, A             ; LUU TOAN HANG 2

  MOV A, 37H            ; TOAN TU '+'
  CALL NUM_TO_CHAR
  MOV R2, A
  CLR C
  SUBB A, #'+'          ; NEU LA TOAN TU '+', NHAY DEN NHAN THUC HIEN PHEP '+'
  JZ CAL_ADD

  MOV A, R2             ; TOAN TU '-'
  CLR C
  SUBB A, #'-'          ; NEU LA TOAN TU '-', NHAY DEN NHAN THUC HIEN PHEP '-'
  JZ CAL_SUB

  MOV A, R2             ; TOAN TU 'x'
  CLR C
  SUBB A, #'x'          ; NEU LA TOAN TU 'x', NHAY DEN NHAN THUC HIEN PHEP 'x'
  JZ CAL_MUL

  MOV A, R2             ; TOAN TU '%'
  CLR C
  SUBB A, #'%'          ; NEU LA TOAN TU '%', NHAY DEN NHAN THUC HIEN PHEP '%'
  JZ CAL_DIV

  CAL_ADD:              ; TINH TOAN PHEP '+'
    MOV A, R1
    ADD A, R3           ; KQ = a + b
    JMP DONE
```

```
  CAL_SUB:              ; TINH TOAN PHEP '-'
    MOV A, R1
    CLR C
    SUBB A, R3          ; KQ = a - b
    JNC SUB_DONE        ; KIEM TRA XEM KET QUA CO AM

    MOV R6, #1
    MOV A, R3
    CLR C
    SUBB A, R1          ; KQ = b - a
    JZ SUB_DONE
  SUB_DONE:
    JMP DONE

    CAL_MUL:            ; TINH TOAN PHEP 'x'
    MOV A, R1
    MOV B, R3
    MUL AB              ; KQ = a x b
    JMP DONE
```

```
CAL_DIV:                ; TINH TOAN PHEP '%'
  MOV A, R3
  JZ ERROR_DISPLAY      ; HIEN THI MATH ERROR NEU A%B KHI B = 0
  MOV A, R1
  MOV B, R3
  DIV AB                ; KQ = a % b , CHIA LAY PHAN NGUYEN
DONE:
  CALL SPLIT_BCD        ; TACH KET QUA RA 2 HANG CHUC, HANG DON VI
  JMP WRITE_RESULT      ; HIEN THI KET QUA RA MAN HINH LCD

  CLEAR_SCREEN:         ; GOI HAM XOA MAN HINH LCD
  CALL CLEAR_LCD

JMP LOOP

ERROR_DISPLAY:          ; GOI HAM HIEN THI "MATH ERROR"
  CALL ERROR_DISPLAY_FUNCT

  JMP EXIT_CAL          ; NHAY TOI KET THUC VIEC TINH TOAN

WRITE_RESULT:           ; GOI HAM HIEN THI KET QUA LEN MAN HINH LCD
  CALL WRITE_RESULT_FUNCT

EXIT_CAL:               ; KET THUC VIEC TINH TOAN

JMP LOOP
```

-Nhãn CALCULATE thực hiện chuyển tính toán kết quả của phép tính.

-Ban đầu chuyển giá trị các toán hạng từ kiểu char sang kiểu int. Toán hạng 1 được lưu trong R1, toán hạng 2 được lưu trong R3

-Thực hiện kiểm tra toán tử, giá trị của toán tử được lưu trong R2:

+Nếu toán tử là ‘+’, nhảy tới nhãn CAL_ADD để thực hiện cộng phép tính.

+Nếu toán tử là ‘-’, nhảy tới nhãn CAL_ADD để thực hiện trừ phép tính.

+Nếu toán tử là ‘x’, nhảy tới nhãn CAL_ADD để thực hiện nhân phép tính.

+Nếu toán tử là ‘%’, nhảy tới nhãn CAL_ADD để thực hiện chia lấy nguyên phép tính.

-Ở nhãn CAL_ADD và nhãn CAL_MUL, ta thực hiện tính toán tương ứng rồi nhảy tới nhãn DONE để gọi hàm SPLIT_BCD để tách kết quả thành hàng chục, hàng đơn vị sau đó nhảy tới nhãy WRITE_RESULT để hiển thị kết quả.

-Ở nhãn CAL_SUB, nếu toán hạng 1 < toán hạng 2, kết quả = toán hạng 2 – toán hạng 1 và thanh ghi R6 lưu giá trị 1, ứng với ý nghĩa kết quả âm. Ngược lại kết quả = toán hạng 1 – toán hạng 2, R6 = 0 ứng với kết quả dương. Sau đó nhảy tới nhãn DONE để gọi hàm SPLIT_BCD để tách kết quả thành hàng chục, hàng đơn vị sau đó nhảy tới nhãy WRITE_RESULT để hiển thị kết quả.

-Ở nhãn CAL_DIV, nếu toán hạng 2 = 0, thực hiện nhảy tới nhãn ERROR_DISPLAY để hiển thị lỗi “MATH ERROR”, ngược lại, thực hiện tính phép chia rồi nhảy tới nhãn DONE để gọi hàm SPLIT_BCD để tách kết quả thành hàng chục, hàng đơn vị sau đó nhảy tới nhãy WRITE_RESULT để hiển thị kết quả.

****Các hàm con:**

```
;============================SUB-PROGRAM============================
SCAN_KEYPAD:            ; HAM QUET PHIM
  CLR ROW1              ; QUET HANG 1
  SETB ROW2
  SETB ROW3
  SETB ROW4
  CLR C
  CALL CHECK_COL
  MOV A, COL
  JZ CHECK_ROW2
  SUBB A, #1            ; 0, 1, 2, 3
  MOV KEY_CODE, A
  JMP EXIT

CHECK_ROW2:             ; QUET HANG 2
  SETB ROW1
  CLR ROW2
  SETB ROW3
  SETB ROW4

  CALL CHECK_COL        ; KIEM TRA COT TUONG UNG
  MOV A, COL

  JZ CHECK_ROW3

  ADD A, #3             ; 4, 5, 6, 7
  MOV KEY_CODE, A
  JMP EXIT

CHECK_ROW3:             ; QUET HANG 3
  SETB ROW1
  SETB ROW2
  CLR ROW3
  SETB ROW4
```

```
  CALL CHECK_COL            ; KIEM TRA COT TUONG UNG
  MOV A, COL
  JZ CHECK_ROW4
  ADD A, #7                 ; 8, 9, 10, 11
  MOV KEY_CODE, A
  JMP EXIT

CHECK_ROW4:                 ; QUET HANG 4
  SETB ROW1
  SETB ROW2
  SETB ROW3
  CLR ROW4

  CALL CHECK_COL            ; KIEM TRA COT TUONG UNG
  MOV A, COL
  JZ EXIT
  ADD A, #11                ; 12, 13, 14, 15
  MOV KEY_CODE, A

EXIT:
RET
```

-Hàm SCAN_KEYPAD thực hiện cho lần lượt 1 trong các hàng ở mức 0 (vị trí x), các hàng còn lại mức 1. Sau đó kiểm tra từng cột, nếu phát hiện cột y đang ở mức 1 (vị trí y), tức phím bấm có vị trí tương ứng là (x, y).

-Sau khi biết được vị trí của phím bấm, ta lưu giá trị tương ứng với từng vị trí phím bấm vào biến KEY_CODE (giá trị biến KEY_CODE từ 0->15, ứng với 16 phím của bàn phím 4x4).

```
CHECK_COL:              ; HAM KIEM TRA COT DUOC BAM
  JB COL1, CHECK_COL2   ; KIEM TRA COT 1
  MOV COL, #1
  JMP FINISH

CHECK_COL2:             ; KIEM TRA COT 2
  JB COL2, CHECK_COL3
  MOV COL, #2
  JMP FINISH

CHECK_COL3:             ; KIEM TRA COT 3
  JB COL3, CHECK_COL4
  MOV COL, #3
  JMP FINISH

CHECK_COL4:             ; KIEM TRA COT 4
  JB COL4, NO_COL
  MOV COL, #4
  JMP FINISH

NO_COL:
  MOV COL, #0           ; KHONG CO COT NAO DUOC BAM
FINISH:
RET
```

-Hàm CHECK_COL thực hiện kiểm tra từng cột xem cột nào đang ở mức 0, tức phím trên cột đó đang được bấm, trả về giá trị cột, nếu không có phím nào được bấm thì trả về 0.

```
DISPLAY_LCD:            ; HAM HIEN THI PHIM BAM LEN MAN HINH LCD
  CALL NUM_TO_CHAR      ; CHUYEN KY TU BAM TUONG UNG TU BAN PHIM THANH CHAR
  CALL WRITE_LCD        ; HIEN THI KY TU LEN MAN HINH LCD
RET

WRITE_LCD:              ; HAM HIEN THI KY TU LEN MAN HINH LCD
  SETB TR1

  MOV SBUF, A           ; GHI DU LIEU CAN TRUYEN LEN THANH GHI SBUF
  JNB TI, $             ; DOI TRUYEN XONG
  CLR TI
  CLR TR1
RET
```

-Hàm DISPLAY_LCD dùng để hiển thị các ký tự quét phím có được lên màn hình, tức là hiển thị phần nhập liệu cho phép toán.

-Trong hàm này, ta thực hiện gọi hàm NUM_TO_CHAR có chức năng chuyển giá trị KEY_CODE thành ký tự kiểu char tương ứng đã khai báo trong mảng ở trên. Sau đó gọi hàm WRITE_LCD để hiển thị lên màn hình.

-Trong hàm WRITE_LCD, ta thực hiện truyền dữ liệu nối tiếp bằng cách bật timer1 sau đó ghi dữ liệu lên thanh ghi SBUF, thực hiện nhảy tại chỗ để chờ quá trình truyền dữ liệu hoàn tất. Sau đó xóa cờ TI, xóa TR1 để tắt timer1.

```
NUM_TO_CHAR:                    ; HAM CHUYEN KY TU BAM TUONG UNG TU BAN PHIM THANH CHAR
  MOV DPTR, #CHAR_CODE
  MOVC A, @A+DPTR
RET
```

-Chức năng của hàm này như đã đề cập phía trên, là chuyển giá trị trả về từ hàm SCAN_KEYPAD thành ký tự tương ứng trong mảng ký tự đã khai báo quy định các phím bấm của bàn phím 4x4.

```
RESET_DATA:                     ; HAM RESET LAI DU LIEU VUNG LUU PHEP TINH
  MOV 38H, #0
  MOV 37H, #0
  MOV 36H, #0
  MOV R0, #38H                  ; GAN LAI R0 O DAU PHEP TINH
RET
```

-Hàm RESET_DATA này dùng để khởi tạo lại các địa chỉ lưu biểu thức sau khi bấm phím ‘ON/C’

```
CLEAR_LCD:                      ; HAM XOA TOAN BO MAN HINH LCD
  MOV A, #0FEH
  CALL WRITE_LCD
  MOV A, #01H
  CALL WRITE_LCD
  CALL RESET_DATA
RET
```

-Khi bấm phím ‘ON/C’, hàm CLEAR_LCD được gọi, ta gửi tới LCD mã lệnh 0xFE để chuyển sang chế độ nhận lệnh.

-Tiếp theo ta truyền vào LCD mã lệnh 0x01 để thực hiện xóa toàn bộ màn hình

-Tiếp theo gọi hàm RESET_DATA để reset lại các giá trị ban đầu.

```
DELAY:                          ; HAM DELAY TRONG KHOANG 5MS
    MOV R0, #10
  LOOP_DELAY1:
    MOV R1, #250
  LOOP_DELAY2:
    DJNZ R1, LOOP_DELAY2
    DJNZ R0, LOOP_DELAY1
RET
```

-Hàm DELAY thực hiện delay trong khoảng thời gian khoảng 10*250*2us = 5ms.

```
ERROR_DISPLAY_FUNCT:        ; HAM HIEN THI 'MATH ERROR' LEN MAN HINH LCD
  CALL CLEAR_LCD            ; XOA MAN HINH LCD TRUOC KHI HIEN THI 'MATH ERROR'
  CALL DELAY                ; DELAY 5MS
  MOV A, #'M'               ; HIEN THI KY TU 'M'
  CALL WRITE_LCD
  MOV A, #'A'               ; ... 'A'
  CALL WRITE_LCD
  MOV A, #'T'               ; ... 'T'
  CALL WRITE_LCD
  MOV A, #'H'               ; ... 'H'
  CALL WRITE_LCD
  MOV A, #' '               ; ... ' '
  CALL WRITE_LCD
  MOV A, #'E'               ; ... 'E'
  CALL WRITE_LCD
  MOV A, #'R'               ; ... 'R'
  CALL WRITE_LCD
  MOV A, #'R'               ; ... 'R'
  CALL WRITE_LCD
  MOV A, #'O'               ; ... 'O'
  CALL WRITE_LCD
  MOV A, #'R'               ; ... 'R'
  CALL WRITE_LCD
RET
```

-Hàm ERROR_DISPLAY_FUNCT thực hiện gọi hàm CLEAR_LCD để xóa màn hình, sau đó delay 5ms và hiển thị chuỗi “MATH ERROR” lên màn hình LCD.

```
WRITE_RESULT_FUNCT:         ; HAM HIEN THI KET QUA LEN MAN HINH LCD
  MOV A, R6                 ; KIEM TRA DAU '-' TRONG KET QUA VA HIEN THI NEU CO
  JZ NOT_NEG
  MOV A, #'-'
  CALL WRITE_LCD
NOT_NEG:
  MOV A, R4                 ; NEU KET QUA LA '0N' THI CHI HIEN THI 'N'
  JZ HC_ZERO
  ADD A, #'0'
  CALL WRITE_LCD            ; HIEN THI HANG CHUC
HC_ZERO:
  MOV A, R5
  ADD A, #'0'
  CALL WRITE_LCD            ; HIEN THI HANG DON VI
RET
```

-Hàm WRITE_RESULT_FUNCT dùng để hiển thị kết quả của phép tính lên màn hình LCD.

-Ban đầu thực hiện kiểm tra giá trị của thanh ghi R6, nếu R6 = 1 thì in ra dấu ‘-‘ ứng với kết quả âm. Ngược lại không in.

-Tiếp theo thực hiện kiểm tra hàng chục của kết quả được lưu trong thanh ghi R4, nếu R4 = 0 không thì bỏ qua, không hiển thị. Ngược lại thì hiển thị hàng chục của kết quả lên màn hình LCD.

-Sau cùng hiển thị hàng đơn vị của kết quả lên màn hình LCD.

IV. References

[1] Giao tiếp LCD

[2] Quét phím ma trận 4x4

[3] Tống Văn On – Hoàng Đức Hải, *HỌ VI ĐIỀU KHIỂN 8051,* Nhà xuất bản Lao Động – Xã Hội