ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
----- €€€€ ------
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ

Public by Nguyễn Phước Lộc – K09 – Tự động hóa
Email: loc.plsoft@gmail.com

BÁO CÁO THÍ NGHIỆM VI XỬ LÝ

Nguyễn Phước Lộc	*40901457*

PHỤ LỤC

Phần A: Thí nghiệm Vi điều khiển 89s52

Bài 1: Thí nghiệm với nút nhấn và Led đơn

Thí nghiệm 1: Viết chương trình thực hiện việc đọc liên tục trạng trạng thái của nút nhấn P1.0 và hiển thị ra LED được nối tại chân P1.1

Code:

```
        ORG 2000H
LOOP:   MOV C, P1.0
        MOV P1.1, C
        SJMP LOOP
        END
```

Thí nghiệm 2.1: Viết chương trình tạo xung vuông 1Hz ra chân P1.0,biết tần số dao động được dùng trong KIT là 11.059MHz.

Code:

```
        ORG 2000H
LAP:      CPL P1.0
          CALL DL500
          SJMP LAP
DL500:    PUSH 05
          PUSH 06
          PUSH 07
          MOV R5, #10
L2:       MOV R6, #100
L1:       MOV R7, #250
          DJNZ R7, $
          DJNZ R6, L1
          DJNZ R5, L2
          POP 07
          POP 06
          POP 05
          RET
          END
```

Thí nghiệm 2.2:

Thay đổi tần số xung vuông thành 0.5Hz, với chu kì nhiệm vụ là 30%

Code:

```
        ORG 2000H
LAP:    SETB P1.0
        CALL DL150
        CLR P1.0
        LCALL DL350
        SJMP LAP
DL150:  PUSH 07
        PUSH 06
```


```
        PUSH 05
        MOV R5, #2
L1_2:   MOV R6, #150
L1_1:   MOV R7, #250
        DJNZ R7, $
        DJNZ R6, L1_1
        DJNZ R5, L1_2
        POP 05
        POP 06
        POP 07
        RET
DL350:  PUSH 07
        PUSH 06
        PUSH 05
        MOV R5, #7
L2_2:   MOV R6, #100
L2_1:   MOV R7, #250
        DJNZ R7, $
        DJNZ R6, L2_1
        DJNZ R5, L2_2
        POP 05
        POP 06
        POP 07
        RET
        END
```

Thí nghiệm 3.1:

Viết chương trình tạo xung vuông 1 Hz dùng Timer0

Code:

```
        ORG 2000H
;Tao xung vuong 1Hz tren P1.0 su dung Timer0
LAP:    CPL P1.0
        CALL DL500
        SJPM LAP
DL500:  MOV TMOD, #01H ;Goi Timer0, Mode1
        PUSH 05
        MOV R5, #10
LOOP:   MOV TH0, #HIGH(-50000)
        MOV TL0, #LOW(-50000)
        SETB TR0
        JNB TF0, $
        CLR TF0
        CLR TR0
        DJNZ R5, LOOP
        POP R5
        RET
        END
```

Thí nghiệm 3.2:

Sử dụng ngắt Timer để tạo xung

Code:

```
        ORG 2000H
;Tao xung 1Hz tren P1.0 dung ngat Timer0
        LJMP MAIN
        ORG 200BH
        INC R5
        CJNE R5, #10, ISRT0
        MOV R5, #1
        CPL P1.0
ISRT0:  CLR TR0
        CLR TF0
        MOV TH0, #HIGH(-50000)
        MOV TL0, #LOW(-50000)
        SETB TR0
        RETI
        ORG 2030H
MAIN:   MOV TMOD ,#01H
        SETB TF0
        MOV IE, #82H
        SJMP $
        END
```

Thí nghiệm 4:

Viết chương trình làm cho LED nối đến chân P1.0 chớp tắt với tần số có thể thay đổi được bằng cách nhấn P1.1 (tăng) hoặc P1.2 (giảm)

Code:

```
        ORG 2000H
        MOV TMOD, #01H
        MOV A,#10
LOOP:
        JNB P1.1,TANG
        JNB P1.2,GIAM
        CALL XUNG
        SJMP LOOP
TANG:
        CJNE A, #255, TANG1
        SJMP LOOP
TANG1:
        INC A
        CALL XUNG
        SJMP LOOP

GIAM:
        CJNE A, #1, GIAM1
        SJMP LOOP
GIAM1:
        DEC A
        CALL XUNG
        SJMP LOOP


XUNG:
        MOV R5, A
        DL50:
        MOV TH0, #HIGH(-50000)
        MOV TL0, #LOW(-50000)
        SETB TR0
```


```
        JNB TF0, $
        CLR TR0
        CLR TF0

        DJNZ R5, DL50
        CPL P1.0
        RET

        END
```

Bài 2: Thí nghiệm hiển thị dùng LED 7 đoạn

Thí nghiệm 1:

Viết chương trình thực hiện bộ đếm từ 0 đến 9 và hiển thị ra LED 7 đoạn,mỗi lần cách nhau 2s.

Code:

```
        ORG 2000H
        MOV     TMOD, #01H
        MOV     DPTR, #0000H
AGAIN:
        MOV     A, #0E0H
NEXT:   MOVX    @DPTR, A
        LCALL   DELAY
        INC     A
        CJNE    A, #0EAH, NEXT
        SJMP    AGAIN
DELAY:
        MOV     R7, #40
DEL1:   MOV     TH0, #HIGH(-50000)
        MOV     TL0, #LOW(-50000)
        SETB    TR0
        JNB     TF0, $
        CLR     TR0
        CLR     TF0
        DJNZ    R7, DEL1
        RET
END
```

Thí nghiệm 2:

Viết chương trình hiển thị giá trị 1234 ra LED 7 đoạn.

Code:

```
        ORG 2000H
        MOV     TMOD,#01H
AGAIN:
```


```
        MOV     DPTR, #0000H
        MOV     A,#71H
        MOVX    @DPTR, A
        LCALL   DELAY

        MOV     A,#0B2H
        MOVX    @DPTR, A
        LCALL   DELAY

        MOV     A,#0D3H
        MOVX    @DPTR, A
        LCALL   DELAY

        MOV     A,#0E4H
        MOVX    @DPTR, A
        LCALL   DELAY

        SJMP    AGAIN
DELAY:
        MOV     TH0, #HIGH(-1000)
        MOV     TL0, #LOW(-1000)
        SETB    TR0
        JNB     TF0, $
        CLR     TR0
        CLR     TF0
        RET
END
```

Thí nghiệm 3: Viết chương trình thể hiện giá trị nhị phân trong thanh ghi R7 ra LED 7 đoạn.

Code:

```
        ORG 2000H
        MOV     R7, #128 ; Giá trị ví dụ
        CALL    BI2BCD
AGAIN:
; HIEN THI HANG DON VI
        MOV     A, 21H
        MOV     DPTR, #DONVI
        MOVC    A, @A+DPTR
        MOV     DPTR, #0000H
        MOVX    @DPTR, A
        LCALL   DELAY
; HIEN THI HANG CHUC
        MOV     A, 22H
        MOV     DPTR, #CHUC
        MOVC    A, @A+DPTR
        MOV     DPTR, #0000H
        MOVX    @DPTR, A
        LCALL   DELAY
; HIEN THI HANG TRAM
        MOV     A, 23H
        MOV     DPTR, #TRAM
        MOVC    A, @A+DPTR
        MOV     DPTR, #0000H
        MOVX    @DPTR, A
        LCALL   DELAY
```


```
        SJMP    EXIT
DELAY:
        MOV     TH0, #HIGH(-1000)
        MOV     TL0, #LOW(-1000)
        SETB    TR0
        JNB     TF0, $
        CLR     TR0
        CLR     TF0
        RET
        RET
BI2BCD:
        MOV     A, R7
        MOV     B, #10
        DIV     AB
        MOV     21H, B ; LUU HANG DON VI
        MOV     B, #10
        DIV     AB
        MOV     22H, B ; LUU HANG CHUC
        MOV     23H, A ; LUU HANG TRAM
        RET
DONVI:
        DB 0E0H, 0E1H, 0E2H, 0E3H, 0E4H, 0E5H, 0E6H, 0E7H, 0E8H, 0E9H
CHUC:
        DB 0D0H, 0D1H, 0D2H, 0D3H, 0D4H, 0D5H, 0D6H, 0D7H, 0D8H, 0D9H
TRAM:
        DB 0B0H, 0B1H, 0B2H, 0B3H, 0B4H, 0B5H, 0B6H, 0B7H, 0B8H, 0B9H
EXIT:
        NOP
END
```

Bài 3: Thí nghiệm hiển thị dùng LCD

Thí nghiệm : Hiển thị dịch trái chuỗi "DHBK Tp.HCM"

Code:

```
        ORG     2000H
        MOV     A,#01H
        CALL    WRITE_COMMAND
        EN      BIT     P3.4
        RS      BIT     P3.5
MAIN:

        MOV     DPTR, #8000H
        CALL    LCD_INIT

        MOV     A, #01H
        CALL    WRITE_COMMAND

        MOV     A, #90H
        CALL    WRITE_COMMAND

        MOV     R1, #11
        MOV     A, #0
        PUSH    ACC
LOOP:
        CALL    TRABANG
```


```
        CALL    WRITE_TEXT
        POP     ACC
        INC     A
        PUSH    ACC
        DJNZ    R1, LOOP
        MOV     R1,#27
LOOP1:
        MOV     A, #18H
        CALL    WRITE_COMMAND
        CALL    DELAY
        DJNZ    R1,LOOP1

        SJMP    MAIN

TRABANG:
        MOV     DPTR, #TABLE1
        MOVC    A, @A+DPTR
RET
;===============================================================
;       LCD_INIT: KHOI DONG LCD
;===============================================================
LCD_INIT:
        MOV     A, #38H
        CALL    WRITE_COMMAND

        MOV     A, #0EH
        CALL    WRITE_COMMAND

        MOV     A, #06H
        CALL    WRITE_COMMAND
RET
;===============================================================



;===============================================================
;       CHO 50MS DE LCD THUC HIEN XONG LENH
;===============================================================

WAIT_LCD:
        MOV     R7, #100
LL1:    MOV     R6, #250
        DJNZ    R6, $
        DJNZ    R7, LL1
RET

;===============================================================

DELAY:
        MOV     R5, #20
LLL1:   MOV     R7, #10
LLL2:   MOV     R6, #250
        DJNZ    R6, $
        DJNZ    R7, LLL2
        DJNZ    R5, LLL1
RET
;===============================================================



;===============================================================
```


```
;       TRUYEN DU LIEU CHO LCD - RS=1
;===============================================================

WRITE_TEXT:
        MOV     DPTR, #8000H
        LCALL   WAIT_LCD
        SETB    RS
        MOVX    @DPTR, A
        SETB    EN
        CLR     EN

RET

;===============================================================
;       TRUYEN LENH CHO LCD - RS=0
;===============================================================

WRITE_COMMAND:
        MOV     DPTR, #8000H
        LCALL   WAIT_LCD
        CLR     RS
        MOVX    @DPTR, A
        SETB    EN
        CLR     EN
RET
;===============================================================

TABLE1:
        DB      'DHBK Tp.HCM'
END
```

Bài 4: Thí nghiệm giao tiếp qua cổng nối tiếp

Thí nghiệm 1: Viết chương trình phát chuỗi ký tự "Hello, world" lên máy tính thông qua Hyper Terminal với tốc độ 19200pbs.

Code:

```
        ORG     2000H
        MOV     SCON,#52H
        MOV     TMOD,#20H
        MOV     TH1,#-3
        SETB    TR1

        MOV     R7,#11
        MOV     A,#0
        PUSH    ACC

LOOP:   MOV     DPTR, #TABLE
        MOVC    A, @A+DPTR
        CALL    OUT_CHAR
        POP     ACC
        INC     ACC
        PUSH    ACC
        DJNZ    R7, LOOP
        SJMP    $
```


```
OUT_CHAR:
        JNB TI, $
        CLR TI
        MOV SBUF, A
        RET

TABLE:
        DB 'Hello world'
        END
```

Thí nghiệm 2: Viết chương trình nhận các ký tự từ máy tính gửi đến EME-MC8 thông qua Hyper Terminal (bằng cách gõ các ký tự trên bàn phím khi đang ở trong chương trình Hyper Terminal) và hiển thị lên LCD.

Code:

```
        ORG 2000H
        EN      BIT     P3.4
        RS      BIT     P3.5


        MOV     A, PCON
        SETB    ACC.7
        MOV     PCON,A

        MOV     SCON,#52H
        MOV     TMOD,#20H
        MOV     TH1,#-3
        SETB    TR1

        MOV     A,#01H
        CALL    WRITE_COMMAND
        MOV     DPTR, #8000H
        CALL    LCD_INIT

MAIN:
        CALL    IN_CHAR
        CALL    WRITE_TEXT
        CALL    OUT_CHAR
        SJMP    MAIN

OUT_CHAR:
        JNB TI, $
        CLR TI
        MOV SBUF, A
        RET

IN_CHAR:
        JNB RI, IN_CHAR
        CLR RI
        MOV A, SBUF
        RET
;===============================================================
;       LCD_INIT: KHOI DONG LCD
;===============================================================
LCD_INIT:
        MOV     A, #38H
```


```
        CALL    WRITE_COMMAND

        MOV     A, #0EH
        CALL    WRITE_COMMAND

        MOV     A, #06H
        CALL    WRITE_COMMAND
RET
;===============================================================



;===============================================================
;       CHO 50MS DE LCD THUC HIEN XONG LENH
;===============================================================

WAIT_LCD:
        MOV     R7, #100
LL1:    MOV     R6, #250
        DJNZ    R6, $
        DJNZ    R7, LL1
RET

;===============================================================

DELAY:
        MOV     R5, #20
LLL1:   MOV     R7, #100
LLL2:   MOV     R6, #250
        DJNZ    R6, $
        DJNZ    R7, LLL2
        DJNZ    R5, LLL1
RET
;===============================================================



;===============================================================
;       TRUYEN DU LIEU CHO LCD - RS=1
;===============================================================

WRITE_TEXT:
        MOV     DPTR, #8000H
        LCALL   WAIT_LCD
        SETB    RS
        MOVX    @DPTR, A
        SETB    EN
        CLR     EN

RET

;===============================================================
;       TRUYEN LENH CHO LCD - RS=0
;===============================================================

WRITE_COMMAND:
        MOV     DPTR, #8000H
        LCALL   WAIT_LCD
        CLR     RS
        MOVX    @DPTR, A
        SETB    EN
```


```
        CLR     EN
RET
        END
```

Bài 5: Thí nghiệm điều khiển ADC

Thí nghiệm 1: Viết chương trình thực hiện đọc điện áp của biến trở trên kênh 0 và hiển thị ra led 7 đoạn.

Code:

```
        ORG 2000H
MAIN:
        MOV TMOD,#20H
LOOP:   MOV DPTR,#4000H
        MOV A,#0
        MOVX @DPTR,A
        CALL DELAY100US
        MOVX A,@DPTR
        CALL BINTOBCD
        MOV DPTR,#0000H
        MOV A,R2
        ORL A,#0E0H
        MOVX @DPTR,A
        CALL DELAY3MS
        MOV A,R3
        ORL A,#0D0H
        MOVX @DPTR,A
        CALL DELAY3MS
        MOV A,R4
        ORL A,#0B0H
        MOVX @DPTR,A
        CALL DELAY3MS
        SJMP LOOP

DELAY100US:
        MOV TH1,#-100
        MOV TL1,#-100
        SETB TR1
        JNB TF1,$
        CLR TF1
        CLR TR1
        RET

BINTOBCD:
        MOV B,#10
        DIV AB
        MOV R2,B
        MOV B,#10
        DIV AB
        MOV R3,B
        MOV R4,A
        RET

DELAY3MS:
        MOV R5,#10
```


```
LOOP1:  MOV R6,#150
        DJNZ R6,$
        DJNZ R5,LOOP1
        RET


END
```

Thí nghiệm 2: Viết chương trình thực hiện đọc điện áp của biến trở trên kênh 0 và hiển thị ra LCD.

Code:

```
        ORG 2000H
MAIN:
        CALL LCD_INIT
        MOV TMOD,#21H
LOOP:
        MOV DPTR,#4000H
        MOV A,#0
        MOVX @DPTR,A
        CALL DELAY100US
        MOVX A,@DPTR
        CALL BINTOASCII
        CALL CLR_LCD
        MOV A,R4
        CALL WRITE_TEXT
        MOV A,R3
        CALL WRITE_TEXT
        MOV A,R2
        CALL WRITE_TEXT
        CALL DELAY3MS
        CALL DELAY3MS
        SJMP LOOP

DELAY100US:
        MOV TH1,#-100
        MOV TL1,#-100
        SETB TR1
        JNB TF1,$
        CLR TF1
        CLR TR1
        RET

BINTOASCII:
        MOV B,#10
        DIV AB
        MOV R2,B
        MOV B,#10
        DIV AB
        MOV R3,B
        MOV R4,A
        MOV A,#30H
        ADD A,R2
        MOV R2,A
        MOV A,#30H
        ADD A,R3
        MOV R3,A
```


```
        MOV A,#30H
        ADD A,R4
        MOV R4,A
        RET

DELAY3MS:
        CLR TF0
        MOV TH0,#HIGH(-10000)
        MOV TL0,#LOW(-10000)
        SETB TR0
        JNB TF0,$
        CLR TR0
        RET

LCD_INIT:
        MOV DPTR,#8000H
        SETB P3.4
        CLR P3.5
        MOV A,#38H
        MOVX @DPTR,A
        CLR P3.4
        CALL DELAY3MS
        SETB P3.4
        CLR P3.5
        MOV A,#0EH
        MOVX @DPTR,A
        CLR P3.4
        CALL DELAY3MS
        SETB P3.4
        CLR P3.5
        MOV A,#06H
        MOVX @DPTR,A
        CLR P3.4
        CALL DELAY3MS
        RET

WRITE_TEXT:
        MOV DPTR,#8000H
        SETB P3.4
        SETB P3.5
        MOVX @DPTR,A
        CLR P3.4
        CALL DELAY3MS
        RET

CLR_LCD:
        MOV DPTR,#8000H
        SETB P3.4
        CLR P3.5
        MOV A,#01H
        MOVX @DPTR,A
        CLR P3.4
        CALL DELAY3MS
        RET
END
```


Bài 6: Thí nghiệm với LED ma trận

Thí nghiệm 1: Viết chương trình hiển thị chữ A ra LED ma trận

Code:

```
        ORG     2000H
;=======================================
;Xuat ky tu A tren LED ma tran
;=======================================
MAIN:
        MOV     A, #01H
        MOV     R0, #0
LAP:    ACALL   QUET_COT
        ACALL   QUET HANG
        INC     R0
        ACALL   DELAY
        CJNE    R0, #5, LAP
        SJMP    MAIN
;---------------------------------------
QUET_COT:
        PUSH    DPH
        PUSH    DPL
        RR      A
        MOV     DPTR, #
        MOVX    @DPTR, A
        POP     DPH
        POP     DPL
        RET

QUET_HANG:
        PUSH    ACC
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR, #
        ACALL   DU LIEU HANG
        MOVX    @DPTR, A
        POP     ACC
        POP     DPH
        POP     DPL
        RET

DU_LIEU_HANG:

        PUSH    DPH
        PUSH    DPL
        MOV     A, R0
        MOV     DPTR, #CHAR_A
        MOVC    A, @A+DPTR
        POP     DPH
        POP     DPL
        RET

DELAY:
        MOV     R7, #4
LAPC1:  MOV     R6, #250
        DJNZ    R6, $
        DJNZ    R7, LAPC1
        RET
```


```
CHAR_A: DB 03H, 0EDH, 0EEH, 0EEH, 0EDH, 03H
        END
```

Thí nghiệm 2: Viết lại chương trình hiển thị chữ A ra LED ma trận có sử dụng chương trình con

Code:

```
ORG     2000H
;=========================================
;Xuat ky tu A tren LED ma tran (dung CTCon)
;=========================================
MAIN:
        MOV     R0, #8
        MOV     R1, #30H
        MOV     DPTR, #CHAR_A
        MOV     A, #0
LAP:    MOVC    A, @A+DPTR
        MOV     @R1, A
        INC     R1
        INC     A
        DJNZ    R0, LAP
        ACALL   XUAT

;-----------------------------------------
XUAT:
        PUSH    ACC
        PUSH    R0
        PUSH    R1
        MOV     A, #01H
        MOV     R0, #0
        MOV     R1, #30H
LAPC:   ACALL   QUET_COT
        ACALL   QUET_HANG
        INC     R0
        INC     R1
        ACALL   DELAY
        CJNE    R0, #7, LAPC
        PUSH    ACC
        PUSH    R0
        PUSH    R1
        SJMP    XUAT
        RET

QUET_COT:
        PUSH    DPH
        PUSH    DPL
        RR      A
        MOV     DPTR, #
        MOVX    @DPTR, A
        POP     DPH
        POP     DPL
        RET

QUET_HANG:
        PUSH    ACC
```


```
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR, #
        MOV     A, @R1
        MOVX    @DPTR, A
        POP     ACC
        POP     DPH
        POP     DPL
        RET

DELAY:
        MOV     R7, #4
LAPC1:  MOV     R6, #250
        DJNZ    R6, $
        DJNZ    R7, LAPC1
        RET

CHAR_A: DB 03H, 0EDH, 0EEH, 0EEH, 0EDH, 03H
        END
```

Bài 7: Thí nghiệm với động cơ DC

Thí nghiệm1: Viết chương trình cho phép động cơ chạy theo chiều thuận trong vòng 2s, nghỉ 2s, chạy theo chiều ngược 2s, nghỉ 2s và lặp lại.

Code:

```
        ORG 2000H
MAIN:   MOV TMOD, #01H
        MOV R1, #40
        MOV DPTR, #0E000H
NEXT:   CLR A
        CALL THUAN
        MOVX @DPTR, A
        CALL DELAY
        CALL NGHI
        MOVX @DPTR, A
        CALL DELAY
        CALL NGHICH
        MOVX @DPTR, A
        CALL DELAY
        CALL NGHI
        MOVX @DPTR, A
        CALL DELAY
        SJMP NEXT

NGHICH:
        SETB ACC.0
        CLR ACC.1
        RET
NGHI:
        CLR ACC.0
        CLR ACC.1
        RET
THUAN:
        CLR ACC.0
```


```
        SETB ACC.1
        RET
DELAY:
        PUSH 01
LAP:    MOV TH0, #HIGH(-50000)
        MOV TL0, #LOW(-50000)
        SETB TR0
        JNB TF0, $
        CLR TR0
        CLR TF0
        DJNZ R1,LAP
        POP 01
        RET
```

Thí nghiệm2: Viết chương trình điều khiển động cơ với phương pháp PWM, trong đó thời gian bật của xung là 50% chu kỳ.

Code:

```
        ORG 2000H
        LJMP MAIN
        ORG 200BH
        LJMP ISR_T0
        ORG 201BH
        LJMP ISR_T1

MAIN:
        MOV TMOD,#11H
        MOV IE,#10001010B
        MOV DPTR,#0E000H
        SJMP $


ISR_T0:
        CLR TR0
        MOV TH0,#HIGH(-50000)
        MOV TL0,#LOW(-50000)
        SETB TR0

        RETI
ISR_T1:
        CLR TR1

        MOV TH0,#HIGH(-50000)
        MOV TL0,#LOW(-50000)
        SETB TR0
        MOV TH1,#HIGH(-25000)
        MOV TL1,#LOW(-25000)
        SETB TR1
        CLR A
        SETB ACC.0
        CLR ACC.1
        MOVX @DPTR,A
        RETI
        END
```


Phần B: Làm việc với PIC 16F690

Bài 8: Thí nghiệm với LED đơn

Thí nghiệm1: Viết chương trình chớp tắt LED được nối đến chân RC2 của PIC 16F690 với tần số chớp tắt là 1Hz.

Code:

```
#include <16F690.h>
#use delay(clock=4000000)
#use fast_io(C)
void main()
{
set_tris_C(0x00);
output_C(0x00);
while(1)
   {
   output_high(PIN_C0);
   delay_ms(500);
   output_low(PIN_C0);
   delay_ms(500);
   }
}
```

Thí nghiệm2: Viết chương trình thực hiện mạch LED chạy được nối đến 4 LED port C của PIC 16F690 như sau, biết thời gian giữa các trạng thái S (sáng) và T (tắt) là 0.5s: STTT->TSTT->TTST->TTTS->STTT->...

Code:

```
#include <16f690.h>
#use delay(clock=4000000)
#use fast_io(C)
#byte port_C=0x07
int   const a[4] = {0x08, 0x04, 0x02, 0x01};
void main()
{
   set_tris_C(0x00);
   output_C(0x00);
   while(1)
      {
       int i=0;
       while(i<4)
          {
            port_C = a[i];
            delay_ms(500);
            i++;
          }
            i = 0;
      }
```


```
}
```

Thí nghiệm3: Viết chương trình đếm số lần nhấn của SW2, kết quả được xuất ra 4 LED đơn được nối đến port C.

Code:

```
#include <16f690.h>
#use delay(clock=4000000)
#use fast_io(C)
#use fast_io(B)
#byte port_C=0x07
void main()
{
   int i=0;
   set_tris_C(0x00);
   set_tris_B(0xFF);
   while(1)
      {
            port_C = i;
            while(input(PIN_B4));
            while(!input(PIN_B4));
            i=i+1;
      }
}
```

Thí nghiệm4: Viết chương trình thực hiện mạch đếm lên hoặc xuống cách nhau 1s. (SW2=1 đếm lên, SW2=0 đếm xuống)

Code:

```
#include <16f690.h>
#use delay(clock=4000000)
#use fast_io(C)
#use fast_io(B)
#byte port_C=0x07
#byte port_B=0x06
#bit sw2=port_B.4
int   const a[4] = {0x08, 0x04, 0x02, 0x01};
void main()
{
   int i=0;
   set_tris_C(0x00);
   set_tris_B(0xFF);
   while(1)
      {
           if(sw2==1)
           {
            i=i+1;
            port_c=i;
            delay_ms(1000);
            if(i==15)
               i=0;
           }
           else
```


```
            {
             i=i+1;
             port_c= 15-i;
             delay_ms(1000);
             if(i==16)
                i=0;
            }
      }
}
```

Bài 9: Thí nghiệm với LED 7 đoạn

Thí nghiệm1: Viết chương trình thực hiện mạch đếm lên từ 0 đến F (số Hex), kết quả xuất ra LED 7 đoạn, giá trị bộ đếm tăng lên 1 đơn vị cách nhau mỗi 0.5s.

Code:

```
#include<16f690.h>
#use delay (clock= 4000000)
#use fast_io(a)
#use fast_io(c)
void main ()
    { int
state1[16]={0x06,0x04,0x07,0x07,0x05,0x03,0x03,0x06,0x07,0x07,0x07,0x01,0x0
2,0x05,0x03,0x03};
      int
state2[16]={0x0F,0x01,0x06,0x03,0x09,0x0B,0x0F,0x01,0x0F,0x0B,0x0D,0x0F,0x0
E,0x07,0x0E,0x0c};
      int i;
      set_tris_a(0x00);
      set_tris_c(0x00);
      while(1)
      {
      for(i=0;i<16;i++)
      {
         output_a(state1[i]);
         output_c(state2[i]);
         delay_ms(500);
      }
      }
     }
```

Thí nghiệm3: Viết chương trình thực hiện mạch đếm thoả các yêu cầu sau:

- Đếm lên từ 0 đến F (số Hex) khi mạch vừa reset,
- Đếm xuống từ F đến 0 (số Hex) ki SW1 được nhấn.

Kết quả xuất ra led đơn , giá trị bộ đếm tăng lên (giảm xuống) 1 đơn vị cách nhau mỗi 0.5s.

Code:

```
#include<16f690.h>
#use delay (clock= 4000000)
#use fast_io(a)
#use fast_io(c)
void main ()
    {
      int value=0;
      int value2=16;
      set_tris_a(0xFF);
      set_tris_c(0x00);

      while(1)
      {
         Laplai:
         value++;
         if(value==16) value=0;
         output_C(value);
         delay_ms(1000);
         if(!input(pin_A1))
         {
            while(1)
            {
               value2--;
               if(value2==0) value2=16;
               output_C(value2);
               delay_ms(1000);
               if(!input(pin_A1))
               {
                  value2=16;
                  value=0;
                  goto Laplai;
               }
            }

         }
      }
    }
```

Bài 10: Thí nghiệm với ADC

<u>Thí nghiệm1:</u> Viết chương trình đọc giá trị điện áp của biến trở RP1 và xuất mức điện áp tương ứng với giá trị điện áp đọc được ra LED 7 đoạn (dung ADC với độ phân giải 10 bits) được cho bởi bảng sau:

Mức (ngõ ra bộ ADC)	Giá trị hiển thị trên LED 7 đoạn
0-63	0
64-127	1
128-191	3

…	…
832-895	D
896-959	E
960-1023	F

Code:

```
#include <16f690.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#use delay (clock= 4000000)
#use fast_io(a)
#use fast_io(c)
void main ()
{  int i;
   int a;

   set_tris_c(0x00);
   set_tris_a(0xff);
   setup_adc_ports( ALL_ANALOG );

   setup_adc(ADC_CLOCK_DIV_32);

   set_adc_channel( 0);

   delay_ms(10);

while(1)
{
   a= read_adc();
   if (0<=a&&a<=63)
      {i=0;
      }
   else if(64<=a&&a<=127)
      {
      i=1;
      }
   else if(128<=a&&a<=191)
      {i=2;
      }
   else if(192<=a&&a<=255)
      {i=3;

      }
   else if(256<=a&&a<=319)
      {i=4;

      }
    else if(320<=a&&a<=383)
      {i=5;

      }
    else if(384<=a&&a<=447)
```


```
      {i=6;

      }
    else if(448<=a&&a<=511)
      {i=7;

      }
    else if(512<=a&&a<=575)
      {i=8;

      }
    else if(576<=a&&a<=639)
      {i=9;

      }
    else if(640<=a&&a<=703)
      {i=10;

      }
    else if(704<=a&&a<=767)
      {i=11;

      }
     else if(768<=a&&a<=831)
      {i=12;

      }
    else if(832<=a&&a<=895)
      {i=13;

      }
    else if(896<=a&&a<=959)
      {i=14;

      }
    else if(960<=a&&a<=1023)
      {i=15;

      }
   output_c(i);
}

}
```

Thí nghiệm2: Viết chương trình đọc giá trị ngõ ra ADC với độ phân giải 10 bits dung để điều khiển thời gian chớp tắt LED đơn được nối đến chân RC0 của PIC 16F690.

Code:

```
#include <16f690.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
```


```
#use delay (clock= 4000000)
#use fast_io(a)
#use fast_io(c)
void main ()
{  int i;
   int a;

   set_tris_c(0x00);
   set_tris_a(0xff);
   setup_adc_ports( ALL_ANALOG );

   setup_adc(ADC_CLOCK_DIV_32);

   set_adc_channel( 0);

   delay_ms(10);

while(1)
{
   a= read_adc();
   i=a/16;
   output_c(i);
}




}
```

Phần C: Bài tập lớn

*Viết chương trình mô phỏng máy tính mini với các phép toán +, - *, / số 3 chữ số.*

```
      ORG   2000H

      MOV         50H,#0
      EN          BIT   P3.4
      RS          BIT   P3.5


      MOV         A, PCON
      SETB        ACC.7
      MOV         PCON,A

      MOV         SCON,#52H
      MOV         TMOD,#20H
      MOV         TH1,#-3
      SETB  TR1

      MOV         A,#01H
      CALL  WRITE_COMMAND
      MOV         DPTR, #8000H
      CALL  LCD_INIT
```


```
MAIN:
NNEXT:

      CALL        IN_OUT
      CALL        SO_SANH
      MOV         22H,A

      CALL        IN_OUT
      CALL        SO_SANH
      MOV         21H, A

      CALL        IN_OUT
      CALL        SO_SANH
      MOV         20H, A

      CALL        IN_OUT
      CALL        SO_SANH
      SJMP        MAIN

IN_OUT:
      CALL        IN_CHAR
      CALL  WRITE_TEXT
      CALL        OUT_CHAR
      RET

OUT_CHAR:
      JNB   TI, $
      CLR   TI
      MOV   SBUF, A
      INC   50H
      RET

IN_CHAR:
      JNB   RI, IN_CHAR
      CLR   RI
      MOV   A, SBUF
      RET
;===============================================================
;     LCD_INIT: KHOI DONG LCD
;===============================================================
LCD_INIT:
      MOV         A, #38H
      CALL  WRITE_COMMAND

      MOV         A, #0CH
      CALL  WRITE_COMMAND

      MOV         A, #0EH
      CALL  WRITE_COMMAND

      MOV         A, #06H
      CALL  WRITE_COMMAND
RET
;===============================================================



;===============================================================
;     CHO 50MS DE LCD THUC HIEN XONG LENH
;===============================================================
```


```
WAIT_LCD:
            MOV   R7, #100
LL1:        MOV   R6, #250
            DJNZ  R6, $
            DJNZ  R7, LL1
RET

;=================================================================

DELAY:
            MOV   R5, #20
LLL1:       MOV   R7, #100
LLL2:       MOV   R6, #250
            DJNZ  R6, $
            DJNZ  R7, LLL2
            DJNZ  R5, LLL1
RET
;=================================================================



;=================================================================
;     TRUYEN DU LIEU CHO LCD - RS=1
;=================================================================

WRITE_TEXT:
      MOV   DPTR, #8000H
      LCALL       WAIT_LCD
      SETB  RS
      MOVX  @DPTR, A
      SETB  EN
      CLR   EN

RET

;=================================================================
;     TRUYEN LENH CHO LCD - RS=0
;=================================================================

WRITE COMMAND:
      MOV   DPTR, #8000H
      LCALL       WAIT_LCD
      CLR   RS
      MOVX  @DPTR, A
      SETB  EN
      CLR   EN
RET
; =================================================================
; =================================================================
;     CT CON 16BIT / 16BIT
;     [R3 R2 R1 R0] = [ R1 R0 ] / [ R3 R2]
; =================================================================
; =================================================================
CHIA_16:
            MOV   A, R2
            ADD   A, R3
            CJNE  A,#0,NNNN
            LJMP  ERROR

NNNN:
```


```
            MOV         R7, #0              ; CLEAR PARTIAL REMAINDER
            MOV         R6, #0
            MOV         B,   #16                 ; SET LOOP COUNT

DIV_LOOP:   CLR         C                   ; CLEAR CARRY FLAG
            MOV         A,  R0              ; SHIFT THE HIGHEST BIT OF
            RLC         A                   ; THE DIVIDEND INTO...
            MOV         R0,  A
            MOV         A,  R1
            RLC         A
            MOV         R1,  A
            MOV         A,  R6              ; ... THE LOWEST BIT OF THE
            RLC         A                   ; PARTIAL REMAINDER
            MOV         R6,  A
            MOV         A,  R7
            RLC         A
            MOV         R7,  A
            MOV         A,  R6              ; TRIAL SUBTRACT DIVISOR
            CLR         C                   ; FROM PARTIAL REMAINDER
            SUBB        A,  R2
            MOV         DPL,  A
            MOV         A,  R7
            SUBB        A,  R3
            MOV         DPH,  A
            CPL         C                   ; COMPLEMENT EXTERNAL BORROW
            JNC         DIV_1               ; UPDATE PARTIAL REMAINDER IF
          ; BORROW
            MOV         R7,  DPH            ; UPDATE PARTIAL REMAINDER
            MOV         R6,  DPL
DIV_1:      MOV         A,  R4              ; SHIFT RESULT BIT INTO PARTIAL
            RLC         A                   ; QUOTIENT
            MOV         R4,  A
            MOV         A,  R5
            RLC         A
            MOV         R5,  A
            DJNZ        B,  DIV_LOOP
            MOV         A,  R5              ; PUT QUOTIENT IN R0, AND R1
            MOV         R1,  A
            MOV         A,  R4
            MOV         R0,  A
            MOV         A,  R7              ; GET REMAINDER, SAVED BEFORE THE
            MOV         R3,  A              ; LAST SUBTRACTION
            MOV         A,  R6
            MOV         R2,  A
;''''''''''''''''''''''''''
            MOV         A,  R0
            MOV         R2, A

            MOV         A, R1
            MOV         R3, A

            LCALL       BINTOASC
            LCALL       HIENTHIKQ
;''''''''''''''''''''''''''''
RET
; ===============================================================
; ===============================================================
;     CT CON 16BIT * 16BIT
;     [R3 R2 R1 R0] = [ R1 R0 ] * [ R3 R2]
; ===============================================================
; ===============================================================
```


```
NHAN_16:
              MOV       A, R0
              MOV       B, R2
              MUL       AB                ; MULTIPLY XL X YL
              PUSH      ACC               ; STACK RESULT LOW BYTE
              PUSH      B                 ; STACK RESULT HIGH BYTE
              MOV       A, R0
              MOV       B, R3
              MUL       AB                ; MULTIPLY XL X YH
              POP       00H          ; RECALL XL*YL HIGH BYTE
              ADD       A, R0
              MOV       R0, A
              CLR       A
              ADDC      A, B
              MOV       DPL, A
              MOV       A, R2
              MOV       B, R1
              MUL       AB                ; MULTIPLY XH X YL
              ADD       A, R0
              MOV       R0, A
              MOV       A, DPL
              ADDC      A, B
              MOV       DPL, A
              CLR       A
              ADDC      A, #0
              PUSH      ACC               ; SAVE INTERMEDIATE CARRY
              MOV       A, R3
              MOV       B, R1
              MUL       AB                ; MULTIPLY XH X YH
              ADD       A, DPL
              MOV       R2, A
              POP       ACC               ; RETRIEVE CARRY
              ADDC      A, B
              MOV       R3, A
              MOV       R1, 00H
              POP       00H               ; RETRIEVE RESULT LOW BYTE
;''''''''''''''''''''''''''
            MOV   A,  R0
            MOV   R2, A

            MOV   A, R1
            MOV   R3, A

            LCALL       BINTOASC
            LCALL HIENTHIKQ
;''''''''''''''''''''''''''''
RET
; ===============================================================
; ===============================================================
;     CHI SU DUNG BCD 3 DIGIT
;     CT CON 16BIT + 16BIT
;     [R3 R2 R1 R0] = [ R1 R0 ] + [ R3 R2]
; ===============================================================
; ===============================================================
CONG_16:
          MOV           A,  R0          ; LOAD X LOW BYTE INTO ACC
          ADD           A,  R2          ; ADD Y LOW BYTE
          MOV           R0,  A          ; PUT RESULT IN Z LOW BYTE
          MOV           A,  R1          ; LOAD X HIGH BYTE INTO ACC
          ADDC          A,  R3          ; ADD Y HIGH BYTE WITH CARRY
          MOV           R1, A          ; SAVE RESULT IN Z HIGH BYTE
```


```
          MOV           C,  OV
          MOV           R3, #0
          MOV           R2, #0
;''''''''''''''''''''''''''''
            MOV   A,  R0
            MOV   R2, A

            MOV   A, R1
            MOV   R3, A

            LCALL       BINTOASC
            LCALL HIENTHIKQ
;''''''''''''''''''''''''''''''
RET
; ===============================================================
; ===============================================================
;     CHI SU DUNG BCD 3 DIGIT KQ = R1 R0
;     CT CON 16BIT - 16BIT
;     [R3 R2 R1 R0] = [ R1 R0 ] - [ R3 R2]
; ===============================================================
; ===============================================================
TRU_16:
            MOV         A,  R0            ; LOAD X LOW BYTE INTO ACC
            CLR         C                 ; CLEAR CARRY FLAG
            SUBB        A,  R2            ; SUBRACT Y LOW BYTE
            MOV         R0,  A            ; PUT RESULT IN Z LOW BYTE
            MOV         A,  R1            ; LOAD X HIGH INTO ACCUMULATOR
            SUBB        A,  R3            ; SUBTRACT Y HIGH WITH BORROW
            MOV         R1,  A            ; SAVE RESULT IN Z HIGH BYTE
            MOV         C,  OV
            MOV         R2,#0
            MOV         R3,#0
;''''''''''''''''''''''''''''
            MOV         A,  R0
            MOV         R2, A

            MOV         A, R1
            MOV         R3, A

            LCALL       BINTOASC
            LCALL       HIENTHIKQ
;''''''''''''''''''''''''''''''
RET
; ===============================================================
; ===============================================================
;     CHUYEN BCD 3 KI SO SANG BINARY
; ===============================================================
; ===============================================================
BCD_2_BI:

            CLR         C
            MOV         A,22H
            SUBB        A,#30H
                        MOV   22H, A

            CLR         C
            MOV         A,21H
            SUBB        A,#30H
                        MOV   21H, A

            CLR         C
```


```
            MOV         A,20H
            SUBB        A,#30H
                        MOV   20H, A

            CLR         C
            MOV         A, 50H
            CJNE        A, #02H, GG2
            MOV         20H, 22H
            MOV         22H, #0
            MOV         21H, #0
GG2:
            CLR         C
            CJNE        A, #03H, GG4
            MOV         20H, 21H
            MOV         21H, 22H
            MOV         22H, #0
GG4:
            CLR         C
            MOV         A, 22H
            MOV         B, #100
            MUL         AB
            MOV         R7,A
            MOV         R6, B
            MOV         A,21H
            MOV         B, #10
            MUL         AB
            ADD         A, 20H
            PUSH        ACC
            MOV         A,B
            ADDC        A, #0
            ADD         A,R6
            MOV         B,A
            POP         ACC
            ADD         A,R7
            MOV         23H, A            ; BYTE THAP
            MOV         A, B
            ADDC        A, #0
            MOV         24H, A            ; BYTE CAO
            MOV         50H, #0
RET
; =============================================================
; =============================================================
BI2BCD:

            MOV         B, #10
            DIV         AB
            MOV         71H, B ; LUU HANG DON VI
            MOV         B, #10
            DIV         AB
            MOV         72H, B ; LUU HANG CHUC
            MOV         73H, A ; LUU HANG TRAM
      RET
; =================================================================
; =================================================================
HIENTHIKQ:
            MOV         55H, #9
            MOV         50H, #0
            MOV         A, #0C9H
            CALL        WRITE_COMMAND

            CLR         C
```


```
      MOV   A, 30H
      SUBB  A,#30H
            MOV   30H, A

      CLR   C
      MOV   A, 31H
      SUBB  A,#30H
            MOV   31H, A

      CLR   C
      MOV   A, 32H
      SUBB  A,#30H
            MOV   32H, A

      CLR   C
      MOV   A, 33H
      SUBB  A,#30H
            MOV   33H, A

      CLR   C
      MOV   A, 34H
      SUBB  A,#30H
            MOV   34H, A

      MOV   A, 30H
      MOV   DPTR, #TABLELCD
      MOVC  A, @A+DPTR
      CALL  WRITE_TEXT

      MOV   A, 31H
      MOV   DPTR, #TABLELCD
      MOVC  A, @A+DPTR
      CALL  WRITE_TEXT

      MOV   A, 32H
      MOV   DPTR, #TABLELCD
      MOVC  A, @A+DPTR
      CALL  WRITE_TEXT

      MOV   A, 33H
      MOV   DPTR, #TABLELCD
      MOVC  A, @A+DPTR
      CALL  WRITE_TEXT

      MOV   A, 34H
      MOV   DPTR, #TABLELCD
      MOVC  A, @A+DPTR
      CALL  WRITE_TEXT

      RET
; ===============================================================
; ===============================================================
BINTOASC:

            MOV   R0,#30h                 ; R0 = POUT
            MOV   DPTR,#TAB               ; R=TAB(P)

COM1:

            CLR   A                       ; P <- 0
            MOVC  A, @A+DPTR              ; R <-  TAB(P)
```


```
            MOV   R7, A
            INC   DPTR
            CLR   A
            MOVC  A, @A+DPTR
            MOV   R6, A

            MOV   R4, #'0'                      ; C  <- '0'


SOMA:                                           ; N <-  N-R
            CLR   C
            MOV   A, R2
            SUBB  A, R6
            MOV   R2, A

            MOV   A, R3
            SUBB  A, R7
            MOV   R3, A
       JC   SAIDA              ;    If < 0 goto  SAIDA
            INC   R4                           ;    If >0 then C <- C +1
            SJMP  SOMA                         ;    goto SOMA
SAIDA:
            MOV   A, R4
            MOV   @R0, A                        ;TABOUT (POUT) <- C

            MOV   A, R2
            ADD   A, R6                         ;  N=N+R
            MOV   R2, A

            MOV   A, R3
            ADDC  A, R7
            MOV   R3, A

            INC   R0                            ; PSAIDA=PSAIDA +1

            CLR   A
            MOVC  A, @A+DPTR
            CJNE  A, #1, INCREMENTA       ; TAB(P) = 1 ?

      RET                                   ; If yes, END

INCREMENTA:                                       ; If No, P <- P+1
            INC   DPTR
            LJMP  COM1                          ; goto COM1
; ============================================================
; ============================================================
SO_SANH:

            CLR   C
                  MOV   60H,A
            SUBB  A, #2BH
            JZ    CONG

            CLR   C
            MOV   A,60H
            SUBB  A, #2DH
            JZ    TRU

                  CLR C
            MOV   A,60H
            SUBB  A, #2FH
```


```
            JZ    CHIA

                  CLR   C
            MOV   A,60H
            SUBB  A, #2AH
            JZ    NHAN

                  CLR   C
            MOV   A,60H
            SUBB  A, #0DH
            JZ    THUCHIEN

                  CLR   C
            MOV   A,60H
            CJNE  A,#30H,X1
X1:
            JC    ERROR
            CJNE  A,#3AH,X2
X2:
            JNC   ERROR
            MOV   A,60H
            RET
CONG:
            MOV   55H, #5
            MOV   40H,#10
            CALL  BCD_2_BI
            MOV   R0, 23H
            MOV   R1, 24H
            LJMP  NNEXT
TRU:
            MOV   55H, #5
            MOV   40H,#11
            CALL  BCD_2_BI
            MOV   R0, 23H
            MOV   R1, 24H
            LJMP  NNEXT

NHAN:
            MOV   55H, #5
            MOV   40H,#12
            CALL  BCD 2 BI
            MOV   R0, 23H
            MOV   R1, 24H
            LJMP  NNEXT
CHIA:
            MOV   55H, #5
            MOV   40H,#13
            CALL  BCD_2_BI
            MOV   R0, 23H
            MOV   R1, 24H
            LJMP  NNEXT

ERROR:

            MOV   R0,#0
            MOV   R1,#0
            MOV   R2,#0
            MOV   R3,#0
            MOV   A, #0C9H
            CALL  WRITE_COMMAND
            MOV   R4, #5
```


```
            MOV   A, #0

LOOPXC:           PUSH  ACC
            MOV   DPTR, #TABLE
            MOVC  A, @A+DPTR
            CALL  WRITE_TEXT
;           MOV   A, #07H
;           CALL  WRITE_COMMAND
            POP   ACC
            INC   A
            DJNZ  R4, LOOPXC
            MOV   55H, #9
            RET

THUCHIEN:
            CLR   C
            MOV   A, 55H
            SUBB  A, # 5
            JZ    TIEPTUC

            MOV   A, # 01H
            LCALL WRITE_COMMAND
            MOV   R0,#0
            MOV   R1,#0
            MOV   R2,#0
            MOV   R3,#0
            MOV   20H, #0
            MOV   21H, #0
            MOV   22H, #0
            MOV   23H, #0
            MOV   24H, #0
            MOV   30H,#0
            MOV   31H, #0
            MOV   32H, #0
            MOV   33H, #0
            MOV   34H, #0
            MOV   50H,#0
            LJMP  NNEXT

TIEPTUC:
            CALL  BCD 2 BI
            MOV   R2, 23H
            MOV   R3, 24H

            MOV   A,40H
            CJNE  A, #0AH, ZZ1
            CALL  CONG_16

            RET
ZZ1:
            CJNE  A, #0BH, ZZ2
            CALL  TRU_16

            RET
ZZ2:
            CJNE  A, #0CH, ZZ3
            CALL  NHAN_16

            RET
ZZ3:
            CJNE  A, #0DH,ERROR
```


```
            CALL  CHIA_16

            RET

TABLELCD:
            DB    '0123456789'
TABLE:
            DB    'ERROR'
TAB:
            DW    10000
            DW    1000
            DW    100
            DW    10
            DW    1
      END
```