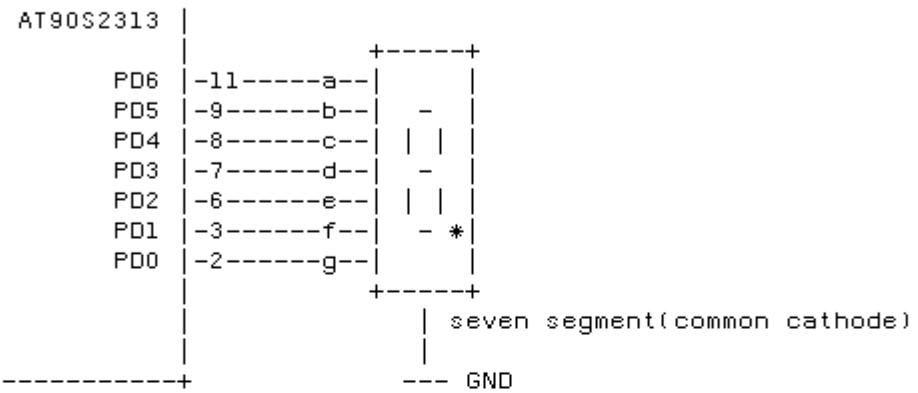
```
+-----+
| MCU | ปรับปรุงการแสดงผลสำหรับ seven segment(2)
| U04 | โดย อ.กำธร เรือนฝายกาศ
+-----+
```

(1) กล่าวโดยทั่วไป

เราจะปรับปรุงโปรแกรม OutSeg1.ASM ที่เราเขียนได้ไว้โดยโปรแกรม จะแสดงผลตัวเลข 0,2,3,4,5,6,7,8,9 ได้โดยการผ่านค่าตัวเลขเข้ามาทางรีจิสเตอร์อินพุต ซึ่งเราจะเลือกใช้ r16 เป็นรีจิสเตอร์อินพุตสำหรับฟังค์ชั่นนี้

(2) วงจร ใช้วงจรเดียวกับการทดลอง OutSeg1



(3) รายการอุปกรณ์

1. seven segment แบบ commnon cathode 1 ตัว

(4) โปรแกรม OutSeg2.ASM

```
;***************************************************************************
; AT90S2313
; 1 seven segment II
; XTAL  = 4.0MHz
; MCU   = AT90S2313,Atmel corperation,www.atmel.com
; copyright (c) 2001 by Gumtorn Ruanfaigad.Allright reserved.
;***************************************************************************

.include        "2313def.inc"           ;000
.DEVICE         AT90S2313               ;001
.CSEG                                   ;002
.ORG            0x0000                  ;003
                                        ;004 ======================
                                        ;005 interrupt vector table
                                        ;006 ======================
        rjmp    F_main                  ;004 #0 RESET
        reti                            ;005 #1 INT0
        reti                            ;006 #2 INT1
        reti                            ;007 #3 TIMER1 CAPT1I
        reti                            ;008 #4 TIMER1 COMP1
        reti                            ;009 #5 TIMER1 OVF1
        reti                            ;010 #6 TIMER0 OVF0
        reti                            ;011 #7 UART,RX
        reti                            ;012 #8 UART,UDRE
        reti                            ;013 #9 UART,TX
        reti                            ;014 #10 ANA_COMP
```

```
;***************************************************************************
;Function OutSegment
;(Yin)  Y16=number to display 0..9
;(Yout) none
;(port) PORTD[x6543210] (segment)
;         AT90S2313 |
;                   |           +-----+
;               PD6 |-11-----a--|     |
;               PD5 |-9------b--|  -  |
;               PD4 |-8------c--| | | |
;               PD3 |-7------d--|  -  |
;               PD2 |-6------e--| | | |
;               PD1 |-3------f--|  - *|
;               PD0 |-2------g--|     |
;                   |           +-----+
;                   |              | seven segment(common cathode)
;                   |              |
;        -----------+              --- GND
;*****************************************
F_OutSeg:                               ;
        push    Y0                      ;
        push    Y16                     ;
        push    ZL                      ;
        push    ZH                      ;
        ldi     ZL,     LOW(L_SegTab*2) ; get table base address low byte
        ldi     ZH,     HIGH(L_SegTab*2); get table base address high byte
        lsl     Y16                     ;
        add     ZL,     Y16             ;
        lpm                             ; load program memory  Y0<-(Z)
        ldi     Y16,    0b11111111      ; all PORTD pin = output
        out     DDRD,   Y16             ; -
        mov     Y16,            Y0      ; output number to 7segment
        out     PORTD,          Y16     ; -
        pop     ZH                      ;
        pop     ZL                      ;
        pop     Y16                     ;
        pop     Y0                      ;
        ret                             ;
                                        ; ====================
                                        ; 7segment lookup table
L_SegTab:                               ; ====================
                                        ; number
        .DW     0b01111110              ; 0           a
        .DW     0b00110000              ; 1          ---
        .DW     0b01101101              ; 2       f |   | b
        .DW     0b01111001              ; 3         | g |
        .DW     0b00110011              ; 4          ---
        .DW     0b01011011              ; 5       e |   | c
        .DW     0b01011111              ; 6         |   |
        .DW     0b01110000              ; 7          ---
        .DW     0b01111111              ; 8           d
        .DW     0b01111011              ; 9
;...........................................................................
```

```
;***************************************************************************
;void main(void)
;***************************************
F_Main:                                 ;202 main function start here
        ldi     r16,    low(RAMEND)     ;203 set stack pointer
        out     SPL,    r16             ;204 -
L_MainLoop:                             ;205 main loop
        ldi     r16,    3               ;206 display number "3"
        rcall           F_OutSeg        ;207 function call
        rjmp            L_MainLoop      ;208 infinite loop
;.....................................................................
```

(5) ทำการ assembling โปรแกรม

ตรงนี้เราจะเปลี่ยนไปใช้วิธีใหม่ที่สะดวกขึ้นคือแทนที่จะเรียกใช้โปรแกรม avrasm.exe เราจะใช้ editv เป็นตัวเรียก avrasm.exe ขึ้นมาทำงานแทน เพราะจากที่ผ่านมาเราจะเห็นว่าการเรียกใช้ avrasm.exe จะต้องพิมพ์ชื่อไฟล์ถึง 3 ไฟล์ ทำให้เสียเวลา ขั้นตอนการใช้ editv เรียกโปรแกรม avrasm.exe มีดังนี้

1. เข้าโปรแกรม editv แล้วกดคีย์ F8 จะปรากฏ Execurte Dialog Box ขึ้นมา
2. ในช่อง Command: ให้พิมพ์คำสั่ง

   ```
   avrasm outseg2.asm outseg2.lst outseg2.hex
   ```

3. พิมพ์เสร็จแล้ว อย่ากดปุ่ม enter แต่ให้กดปุ่ม Alt ค้างไว้แล้วกดปุ่มตัว A เพื่อ Add คำสั่ง จะพบว่าในช่อง Command: ได้เพิ่มคำสั่งของเราเข้าไปดังนี้

   ```
    Command:
   [avrasm outseg2.asm outseg2.lst outseg2.hex]
    avrasm outseg2.asm outseg2.lst outseg2.hex
   ```

4. กดคีย์ Tab ที่อยู่ด้านซ้ายของคีย์บอร์ด 2 ครั้ง แล้วเคอเซอร์จะย้ายไปที่ช่อง

   ```
   [ ] Remain in DOS (Type Exit to return...)
   ```

   กดคีย์ space bar หนึ่งครั้ง จะเกิดตัวอักษรตัว X ในช่องว่างดังนี้

   ```
   [X] Remain in DOS (Type Exit to return...)
   ```

5. กดคีย์ ESC เพื่อออกจาก dialog box นี้ แล้วเลือกเมนู File>>Chang dir... จะปรากฏ Change Directory Dialog Box ขึ้นมา ในช่อง Directory name ให้พิมพ์คำว่า c:\myavr ลงไป แล้วกดปุ่ม Enter แต่ถ้ามีคำว่า c:\myavr อยู่แล้วก็ให้กดปุ่ม Enter ผ่านไปได้เลย
6. ตรงนี้ทดลองให้คำสั่งที่เราได้ทำไปโดย
   - 6.1 กดคีย์ Alt ค้างไว้ แล้วกดคีย์ F1
   - 6.2 โปรแกรมจะออกไปที่ดอส แล้วทำคำสั่งของเราโดยอัตโนมัติ ได้ผลดังนี้

```
AVRASM: AVR macro assembler version 1.21  (May  5 1998 01:21:00)
Copyright (C) 1995-1997 ATMEL Corporation

Creating   'outseg2.eep'
Creating   'outseg2.hex'
Creating   'outseg2.obj'
Creating   'outseg2.lst'

Assembling 'outseg2.asm'
Including  '2313def.inc'

Program memory usage:
Code              :   34 words
Constants (dw/db):   10 words
Unused            :    0 words
Total             :   44 words

Assembly complete with no errors.
Deleting   'outseg2.eep'

C:\MYAVR>
```

6.3 จะเห็นว่าการทำงานเหมือนกับเราออกไปสั่ง ที่ดอสเอง แต่ใช้งานได้สะดวกกว่ามาก เพราะไม่ต้องไปพิมพ์คำสั่งที่ดอสทุกครั้ง โดยเฉพาะอย่างยิ่งถ้าโปรแกรมที่เขียนมีข้อผิดพลาดจะต้องคอยเข้าออก editv กับดอสบ่อยๆ

6.4 พิมพ์คำสั่ง

exit <-:

เพื่อกลับเข้ามาโปรแกรม editv

ผลจาก assembling จะพบว่าไม่มีข้อผิดพลาด ใช้หน่วยความจำสำหรับโปรแกรมไป เท่ากับ 31 เวิร์ด ใช้สำหรับค่าคงที่ คือตารางรหัสสำหรับ seven segment อีก 10 เวิร์ด รวมทั้งสิ้นเป็น 41 เวิร์ด

(6) เขียนโปรแกรมลง MCU

เพิ่มคำสั่ง ลงใน Execute Dialog Box เป็นคำสั่งที่ 2 และ 3 อีก 2 คำสั่งดังนี้

```
spl2 -wpfc outseg2.hex
spl2 -P2
```

ตอนนี้รายการคำสั่งใน Execute Dialog Box จะเพิ่มรวมเป็น 3 คำสั่งดังนี้

```
 Command:
[avrasm outseg2.asm outseg2.lst outseg2.hex]
 avrasm outseg2.asm outseg2.lst outseg2.hex
 spl2 -wpfc outseg2.hex
 spl2 -P2
```

กดคีย์ Alt ค้างไว้แล้วกดปุ่ม F2 เพื่อให้คำสั่งที่ 2 ของเราทำงาน หน้าจอภาพ จะปรากฏข้อความแสดงว่าได้ทำการเขียนโปรแกรม outseg2.hex ลง ไอซี MCU เรียบร้อยแล้ว

```
C:\MYAVR>spl2 -wpfc outseg2.hex
SP12 version 1.0.2 performing init...
rcPath: _spl2rc
Running in SP12 cable/dongle compatible mode.
The character `c' is not a valid option.
Enabling AVR serial reading/programming...
```

```
The device code bytes 0,1,2: 0x1e, 0x91, 0x1 were read
from parallel port 0x378 and indicate the following:
You have connected an AT90S2313
The device was made by Atmel

Performing chip erase...
Writing content of outseg2.hex into program area.
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
outseg2.hex written and verified.
write retries: 0, byteWrite: 16 percent of default

C:\MYAVR>
```

กลับเข้ามายังโปรแกรม editv โดยคีย์คำสั่ง

```
                        exit <-:
```

ไปดูที่ seven segment จะพบว่ายังไม่สว่าง เพราะขา reset ของ MCU
ยังเป็นลอจิก 0 อยู่ ต้องใช้คำสั่งที่ 3 คือ spl2 -P2 โดยการกดปุ่ม Alt
ค้างไว้ แล้วกดปุ่ม F3 จะได้

```
C:\MYAVR>spl2 -P2
SP12 version 1.0.2 performing init...
rcPath: _spl2rc
Running in SP12 cable/dongle compatible mode.
Writing the byte 0x02 (2, B00000010) to the parallel port data bits.
Enabling AVR serial reading/programming...
The device code bytes 0,1,2: 0x1e, 0x91, 0x1 were read
from parallel port 0x378 and indicate the following:
You have connected an AT90S2313
The device was made by Atmel

Reset was left high.
```

โปรแกรม spl2 รายงานเราว่าได้สั่งให้ขา reset ของ MCU เป็นลอจิก 1 แล้ว
seven segment จะสว่างเป็นเลข 3 ควรทดลองเปลี่ยนให้โปรแกรมของเรา
แสดงตัวเลขอื่นๆ ดูบ้าง

(6) สรุป

1. เราได้สร้างฟังค์ชันขึ้นมาหนึ่งฟังค์ชันคือ ฟังค์ชัน OutSeg
   มีคุณสมบัติดังนี้

   <> รับข้อมูลตัวเลขที่จะแสดงเข้ามาทางรีจิสเตอร์ r16
   <> แสดงผลได้ 1 หลัก ตัวเลข 0,1,2,3,4,5,6,7,8,9
   <> ข้อมูลตัวเลขที่เข้ามาทาง r16 หลังจากทำงานในฟังค์ชันนี้
      เสร็จแล้ว จะไม่ถูกทำลาย คือ main function ยังนำกลับไปใช้
      ได้อีก

2. เราได้ใช้ความสามารถของโปรแกรม edtv ที่สามารถ execute คำสั่งของดอส
   ได้ โดยกำหนดไว้ 3 คำสั่งคือ

   <> Alt+F1 avrasm outseg2.asm outseg2.lst outseg2.hex
              - ใช้ในการ assembling โปรแกรมที่เราเขียนขึ้น
   <> Alt+F2 spl2 -wpfc outseg2.hex
              - ใช้เขียนโปรแกรมลง MCU
   <> Alt+F3 spl2 -P2
              - ใช้เพื่อให้ MCU เริ่มต้นทำงาน

(7) อธิบายการทำงานของโปรแกรม

```
;***************************************************************************
;Function OutSegment
;(vin)  r16=number to display 0..9
;(vout) none
;(port) PORTD[x65432100] (segment)
;         AT90S2313 |
;                   |            +-----+
;               PD6 |-11-----a--|     |
;               PD5 |-9------b--|  -  |
;               PD4 |-8------c--| | | |
;               PD3 |-7------d--|  -  |
;               PD2 |-6------e--| | | |
;               PD1 |-3------f--|  - *|
;               PD0 |-2------g--|     |
;                   |            +-----+
;                   |               | seven segment(common cathode)
;                   |               |
;        -----------+              --- GND
;*****************************************
        <> ฟังค์ชันชื่อ OutSeg การส่งผ่านค่ารีจิสเตอร์เข้ามา คือ r16 เป็นตัวเลขที่
        ต้องการให้แสดงออกทาง seven segment ค่าที่ส่งกลับไม่มี ใช้ PORTD เพื่อขับ
        seven segment
.....................................................................

F_OutSeg:                               ;
        push    r0                      ;
        push    r16                     ;
        push    ZL                      ;
        push    ZH                      ;

        <> เก็บค่ารีจิสเตอร์ลง stack ในที่นี้มีรีจิสเตอร์สองตัวคือ ZL,ZH ซึ่งที่จริง
        รีจิสเตอร์ ZL ก็คือ r30 และ ZH ก็คือ r31 ซึ่งได้กำหนด(define) ไว้ในไฟล์
        2313def.inc ว่าดังนี้
              ......
        .def    ZL      =r30
        .def    ZL      =r31
              ......
              ......

        ทั้งคู่เป็นรีจิสเตอร์ขนาด 8 บิต เมื่อนำมารวมกันจะเรียกว่ารีจิสเตอร์ Z มีขนาด
        16 บิต ดังนั้นรีจิสเตอร์ Z จึงสามารถใช้ในการอ้างแอดเดรสได้ถึง 16 บิต
.....................................................................

        ldi     ZL,     LOW(L_SegTab*2) ; get table base address low byte
        ldi     ZH,     HIGH(L_SegTab*2); get table base address high byte

        <> โหลดค่าตัวชี้(pointer) ของตารางเข้ามาเก็บไว้ในรีจิสเตอร์ Z โดยโหลดไบต์ต่ำ
        เข้ารีจิสเตอร์ ZL และไบต์สูงเข้ารีจิสเตอร์ ZH ด้วยการใช้ assembler directive
        LOW() และ HIGH()  เนื่องจากตารางของเราอยู่ใน flash memory ซึ่งการอ้างแอดเดรส
        จะเป็นแบบ 16 บิต คือการเพิ่มแอดเดรส 1 แอดเดรส จะเท่ากับเพิ่ม 2 ไบต์ ดังนั้น
        เราจึงต้องคูณ 2 เข้าไปกับค่าของเลเบล
.....................................................................

        lsl     r16                     ;
        add     ZL,     r16             ;
```

```
        <> คูณ 2 เข้ากับตัวเลขที่ต้องการแสดงเพื่อนำไปชี้เอาค่าออกมาจากตาราง
        เช่น ถ้า r16 มีค่าเท่ากับ 3 หลังจากทำ lsl r16 แล้ว ค่าใน r16 จะถูกเลื่อนไปทาง
        ซ้ายมือ 1 บิต ซึ่งเท่ากับเป็นการคูณ r16 ด้วย 2 นั่นเอง ค่าจะได้เป็น 6
        หลังจากนั้น ก็นำ r16 ไปบวกเข้ากับ ZL เพื่อเพิ่ม pointer ของตารางให้เพิ่มขึ้นอีก
        6 ไบต์ หรือ 3 เวิร์ด ซึ่งจะตรงกับค่ารหัสของตัวเลข 3 ในตารางคือ 0b01111001
        นั่นเอง
.....................................................................

        lpm                             ; load program memory  r0<-(Z)

        <> เป็นคำสั่งให้โหลดค่าจาก flash memory ที่ Z ชี้อยู่ เข้ามาเก็บไว้ใน
        รีจิสเตอร์ r0
.....................................................................

        ldi     r16,    0b11111111      ; all PORTD pin = output
        out     DDRD,   r16             ; -
<> โปรแกรมให้ PORTD เป็นเอาต์พุตทุกขา
.....................................................................

        mov     r16,            r0      ; output number to 7segment
        out     PORTD,          r16     ; -

        <> นำค่ารหัสจาก r0 เข้าไปไว้ใน r16 แล้วส่งออก PORTD เพื่อขับแต่ละ segment ของ
        seven segment
.....................................................................

        pop     ZH                      ;
        pop     ZL                      ;
        pop     r16                     ;
        pop     r0                      ;
        ret                             ;

        <> นำค่าเดิมก่อนที่จะเข้ามาฟังค์ชั่นนี้ คืนไปให้รีจิสเตอร์ต่าง แล้วกลับไปยัง
        main function
.....................................................................

                                        ; ======================
                                        ; 7segment lookup table
L_SegTab:                               ; ======================
                                        ; number
        .DW     0b01111110              ; 0           a
        .DW     0b00110000              ; 1          ---
        .DW     0b01101101              ; 2       f |   | b
        .DW     0b01111001              ; 3         | g |
        .DW     0b00110011              ; 4          ---
        .DW     0b01011011              ; 5       e |   | c
        .DW     0b01011111              ; 6         |   |
        .DW     0b01110000              ; 7          ---
        .DW     0b01111111              ; 8           d
        .DW     0b01111011              ; 9

        <> ส่วนนี้เป็นตารางรหัสของตัวเลขแต่ละตัวที่ต้องการนำไปแสดงบน seven segment
        ใช้เนื้อที่ไปทั้งหมด 16 เวิร์ด โดยมี L_SegTab เป็นเลเบลอ้างอิงสำหรับกำหนดจุด
        เริ่มต้นของตาราง
.....................................................................
```

```
;***************************************************************************
;void main(void)
;***************************************
F_Main:                                 ;202 main function start here
        ldi     r16,    low(RAMEND)     ;203 set stack pointer
        out     SPL,    r16             ;204 -
<> ส่วนของโปรแกรมหลัก กำหนด stack pointer
.............................................................................

L_MainLoop:                             ;205 main loop
        ldi     r16,    3               ;206 display number "3"
        rcall           F_OutSeg        ;207 function call
        rjmp            L_MainLoop      ;208 infinite loop
        <> กำหนดตัวเลขที่ให้แสดงไว้ใน r16 ในที่นี้คือเลข 3 แล้วเรียกฟังค์ชั่น OutSeg
        ให้ทำงานต่อ เมื่อเสร็จแล้วก็ให้วนอยู่ในหลูป MainLoop แบบไม่สิ้นสุด
.............................................................................
```