โปรแกรมคณิตศาสตร์และตรรก (Arithmetic and Logical Operation)

ข้อมูลเบื้องต้น

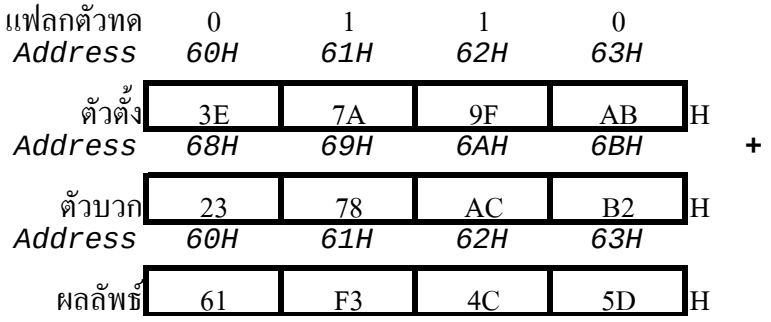
การบวกเลขจำนวนเต็ม

ไมโครคอนโทรลเลอร์ มีคำสั่งบวก 2 คำสั่งได้แก่ ADD กับ ADC คำสั่งทั้งสองจะบวกเลขจำนวนเต็มได้ครั้งละ 8 บิต หรือ 1 ไบต์ โดยคำสั่ง ADD จะบวกข้อมูลในรีจิสเตอร์ Rd กับรีจิสแตอร์ตัวบวก Rr แล้วผลลัพธ์เก็บไว้ในรีจิสเตอร์ Rd ส่วนคำสั่ง ADC จะบวก ข้อมูลในรีจิสเตอร์ Rd กับตัวบวก Rr และแฟลกตัวทด (Carry flag) เข้าด้วยกันแล้วเก็บผลลัพธ์ไว้ที่รีจิสเตอร์ Rd เช่น

ADD Rd,Rr หมายถึง Rd <-- Rd + Rr

ADC Rd,Rr หมายถึง Rd <-- Rd + Rr + แฟลกตัวทด C

การบวกเลขจำนวนเต็มหลายไบต์ จะใช้วิธีบวกด้วยคำสั่ง ADC หลายครั้ง โดยก่อนการบวกไบต์แรกต้องเครียล์แฟลกตัวทดให้เป็นศูนย์ ส่วนการบวกในไบต์ต่อไปจะได้รับการทดอันเกิดจากไบต์ก่อนหน้าผ่านทางแฟลกตัวทด เช่นตามรูปที่ 2.1 แสดงการบวกข้อมูลจำนวน 4 ไบต์เข้ากับข้อมูล 4 ไบต์ ตัวตั้งเก็บอยู่ในหน่วยความจำตำแหน่งที่ 60H ถึง 63H ตัวบวกเก็บอยู่ในหน่วยความจำตำแหน่งที่ 68H ถึง 6BH ผลลัพธ์จะนำกับไปเก็บที่หน่วยความจำตำแหน่งเดิมของตัวตั้ง ตามตัวอย่างนี้ตัวตั้งมีค่าเท่ากับ 3E7A9FABH และตัวบวกมีค่าเท่ากับ 2378ACB2H การบวกจะเริ่มโดยการเครียล์แฟลกตัวทดให้เป็นศูนย์ดังนั้นการบวกไบต์แรกจะเป็นการบวกของ ABH + B2H + 0 (แฟลกตัวทด) ได้ผลลัพธ์เป็น 15DH โดย 5DH จะอยู่ในรีจิสเตอร์ Rd และ 1 จะปรากฏที่แฟลกตัวทด ค่า 5DH ในรีจิสเตอร์ Rd จะถูกนำไปเก็บที่หน่วยความจำตำแหน่งที่ 63H ส่วน 1 ในแฟลกตัวทดจะถูกนำไปบวกในการบวกของไบต์ต่อไป คือ 9FH + ACH + 1 ได้ผลลัพธ์เป็น 14CH 4CH จะอยู่ในรีจิสเตอร์ Rd และ 1 จะปรากฏที่แฟลกตัวทด การบวกจะเป็นเช่นนี้ต่อไปจนครบทุกไบต์ ผลลัพธ์สุดท้ายของการบวกจะประกอบด้วยค่าของแฟลกตัวทดและค่าในหน่วยความจำตำแหน่งที่ 30H ถึง 33H ตามตัวอย่างนี้มีค่าเท่ากับ 061F34C5DH



รูปที่ 2.1 การบวกข้อมูลหลายไบต์ที่อยู่ในหน่วยความจำเข้าด้วยกัน

โปรแกรมที่ 2.1 โปรแกรมบวกเลขจำนวนเต็มจำนวน 4 ไบต์เข้ากับเลขจำนวนเต็ม 4 ไบต์ ตัวตั้งเก็บอยู่ในหน่วยความจำตำแหน่งที่ 60H ถึง 63H ตัวบวกเก็บอยู่ในหน่วยความจำตำแหน่งที่ 68H ถึง 6BH ผลลัพธ์นำกลับมาเก็บที่หน่วยความจำตำแหน่งที่ 30H ถึง 33H

```
/*
 * 4_bit_Adder.asm
 *  Created: 11/7/2554 23:44:43
*/
.include "8515def.inc"
.def    operand1 = R16
.def    operand2 = R17
.def    counter = R18
.equ    number = 4
;-----------------------------------------------------

.DSEG
.org            $60
num1:           .BYTE           4
.org    $68
num2:           .BYTE           4

.CSEG
.org $000
START:          LDI             counter,number          ;load counter = number
                LDI             XH,HIGH(num1+4)         ;load pointer XH to num1 + 4
                LDI             XL,LOW(num1+4)          ;load pointer XL to num1 + 4
                LDI             YH,HIGH(num2+4)         ;load pointer YH to num2 + 4
                LDI             YL,LOW(num2+4)          ;load pointer YL to num2 + 4
                LDI             ZH,HIGH(num1+4)         ;load pointer ZH to num1 + 4
                LDI             ZL,LOW(num1+4)          ;load pointer ZL to num1 + 4
                CLC                                     ;Clear carry flag

LOOP:           LD              operand1,-X             ;load num1 to operand1
                LD              operand2,-Y             ;load num2 to operand2
                ADC             operand1,operand2       ;add with carry
                ST              -Z,operand1             ;save result
                DEC             counter                 ;decrement counter
                TST             counter                 ;check counter = 0
                BRNE            LOOP                    ;jump to loop if counter != 0

HERE:           RJMP            HERE
```

การคูณ

หลักการ

AVR มีคำสั่งคูณหลายคำสั่งเช่น MUL Rd,Rr เป็นการคูณเลขไม่มีเครื่องหมาย และ MULS Rd,Rr เป็นการคูณเลขมีเครื่องหมาย คูณ ผลคูณจะมีขนาด 16 บิตหรือ 2 ไบต์ โดยไบต์สูงอยู่ที่รีจิสเตอร์ R1 และไบต์ต่ำอยู่ที่รีจิสเตอร์ R0

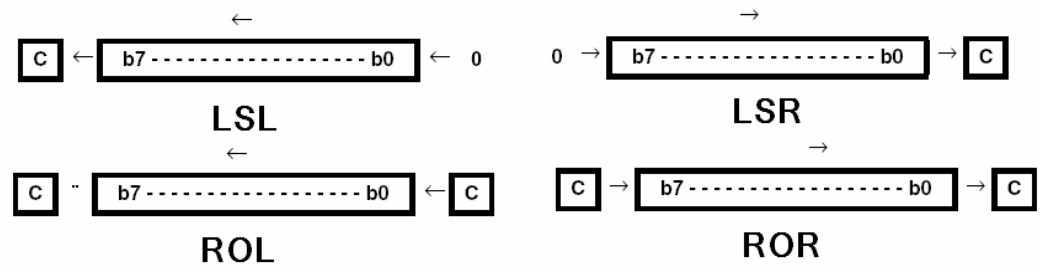
โปรแกรมที่ 2.2 โปรแกรมคูณเลขขนาด 8 บิต x 8 บิต ตัวตั้งอยู่ในหน่วยความจำตำแหน่งที่ 60H ตัวคูณอยู่ที่หน่วยความจำตำแหน่งที่ 61H ผลคูณอยู่เก็บไว้ที่หน่วยความจำตำแหน่งที่ 62H และ 63H

```
.include "m8def.inc"
.def    operand1 = R16
.def    operand2 = R17
;-----------------------------------------------------

.DSEG
.org    $60
num1: .BYTE  1
num2: .BYTE  1
res:    .BYTE  2
.CSEG
.org $000
START:      LDI         XH,HIGH(num1+1)         ;load pointer XH to num1 + 4
            LDI         XL,LOW(num1+1)          ;load pointer XL to num1 + 4
            LDI         YH,HIGH(num2+1)         ;load pointer YH to num2 + 4
            LDI         YL,LOW(num2+1)          ;load pointer YL to num2 + 4
            LDI         ZH,HIGH(res+2)          ;load pointer ZH to num1 + 4
            LDI         ZL,LOW(res+2)           ;load pointer ZL to num1 + 4

LOOP:       LD          operand1,-X             ;load num1 to operand1
            LD          operand2,-Y             ;load num2 to operand2
            MUL         operand1,operand2       ;multiply
            ST          -Z,r0                   ;save result
            ST          -Z,r1

HERE:       RJMP        HERE
```

การหมุนข้อมูล

หลักการ

AVR มีคำสั่งเกี่ยวกับการหมุนข้อมูลดังนี้ LSL, LSR, ROR และ ROL โดยลักษณะการหมุนเป็นดังนี้



รูปที่ 2.2 การทำงานของคำสั่ง LSL, LSR, ROL และ ROR

โปรแกรมที่ 2.3 ตัวอย่างการใช้คำสั่งเกี่ยวกับการหมุนข้อมูล

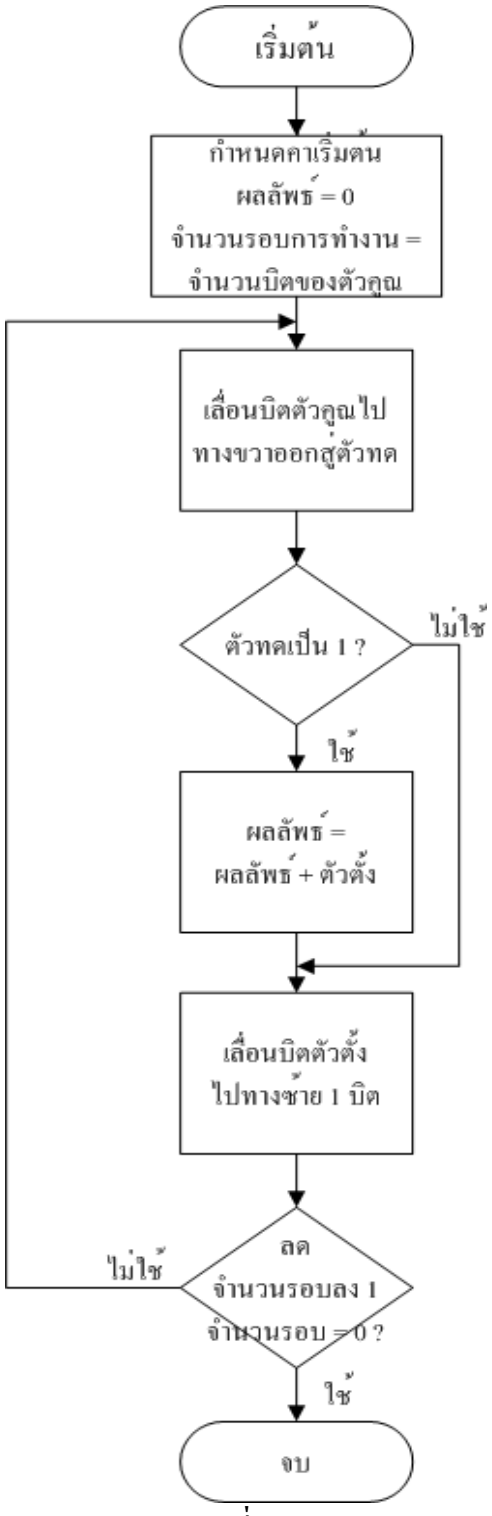
```
/*
 *  8 byte shift, 4 time
 *   Author: xp
 */

;-----------------------------------------------------
.include "m8def.inc"
.def    operand1 = R16
.def    counter = R18
.equ    number = 8
;-----------------------------------------------------

.DSEG
.org    $60
num1: .BYTE  8

.CSEG
.org $000
START:      LDI         operand1,low(RAMEND)      ;Load stack with
            OUT         SPL,operand1              ;RAMEND - highest value
            LDI         operand1,high(RAMEND)     ;of internal SRAM
            OUT         SPH,operand1
            RCALL       ROR8                      ;Call ROR8
            RCALL       ROR8                      ;Call ROR8
            RCALL       ROR8                      ;Call ROR8
            RCALL       ROR8                      ;Call ROR8

HERE:       RJMP        HERE
;-------------------------------------------------------------
ROR8:       LDI     counter,number                ;load counter = number
            LDI         XH,HIGH(num1)             ;load pointer XH to num1
            LDI         XL,LOW(num1)              ;load pointer XL to num1
            LDI         ZH,HIGH(num1)             ;load pointer ZH to num1
            LDI         ZL,LOW(num1)              ;load pointer ZL to num1
            CLC                                   ;Clear carry flag

LOOP:       LD          operand1,X+               ;load num1 to operand1
            ROR         operand1                  ;rotale right
            ST          Z+,operand1               ;save result
            DEC         counter                   ;decrement counter
            TST         counter                   ;check counter = 0
            BRNE        LOOP                      ;jump to loop if counter != 0
            RET
```

แบบฝึกหัด

1. ให้เขียนโปรแกรมคูนเลข 4 ไบต์ x 4 ไบต์
2. ให้เขียนโปรแกรมหารเลข เลข 4 ไบต์ / เลข 2 ไบต์

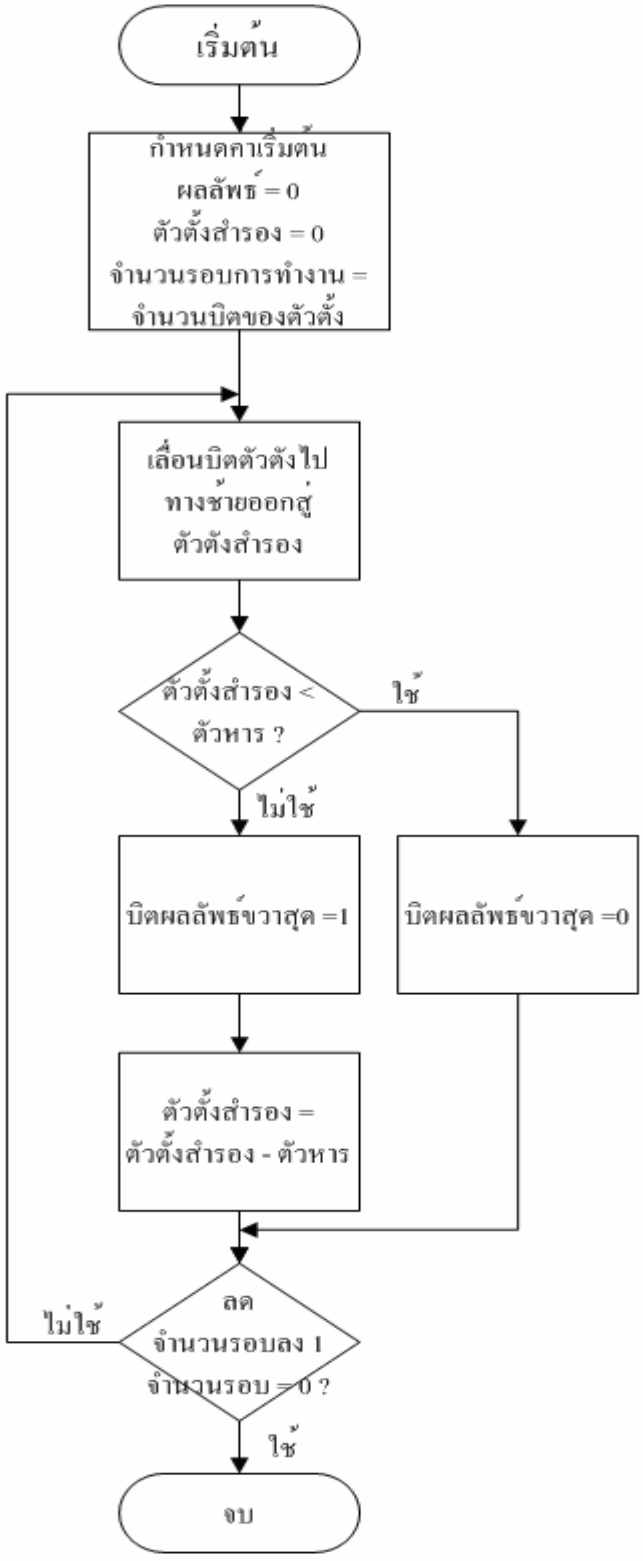
ข้อแนะนำ

โฟลว์ชาร์ตการคูณหลายไบต์



รูปที่ 2.3

โฟลว์ชาร์ตการหารหลายไบต์



รูปที่ 2.4

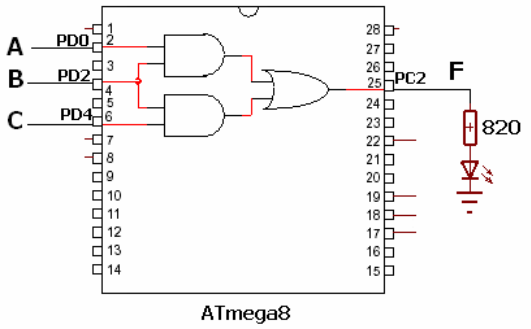
การทำทางตรรกะ (Logical operation)

AVR มีคำสั่งสำหรับการทำงานทางตรรกะหลายคำสั่งเช่น AND (Logic AND), OR (Logic OR) และ EOR (Logic XOR) การกระทำทั้งหมดนี้จะทำแบบบิตต่อบิต และทำระหว่างรีจิสเตอร์ Rd กับรีจิสเตอร์ Rr แล้วผลลัพธ์ก็เก็บไว้ในรีจิสเตอร์ Rd เช่น ถ้ารีจิสเตอร์ R17 = 75H R18 = 5EH ถ้าใช้คำสั่ง AND R17,R18 จะมีการกระทำดังนี้

R17 =	75H =	0	1	1	1	0	1	0	1 B
R18 =	5EH =	0	1	0	1	1	1	1	0 B
R17 = R17 AND R18	R17 =	0	1	0	1	0	1	0	0 B

บิต 0 ของ R17 AND กับบิต 0 ของ R18 คือ 1 AND 0 = 0

โปรแกรมที่ 2.3 โปรแกรมที่ทำงานได้ตามรูปที่ 2.5



รูปที่ 2.5

```
.include "m8def.inc"
.def    A = R16
.def    B = R17
.def    C = R18
;-----------------------------------------------------
.CSEG
.org $000
START:      LDI         A,0b00000000            ; load register r16 with all 0's
            OUT         DDRD,A                  ; configure PORTD for all inputs
            LDI         A,0b11111111 ; pull up
            OUT         PORTD,A
            OUT         DDRC,A                  ;configure PORTC as output

LOOP:       IN          A,PIND                  ;read port D
            MOV         B,A
            MOV         C,A
            LSL         A
            LSL         A
            LSR         C
            LSR         C
            AND         A,B
            AND         C,B
            OR          A,C
            OUT         PORTC,A
            RJMP        LOOP
```