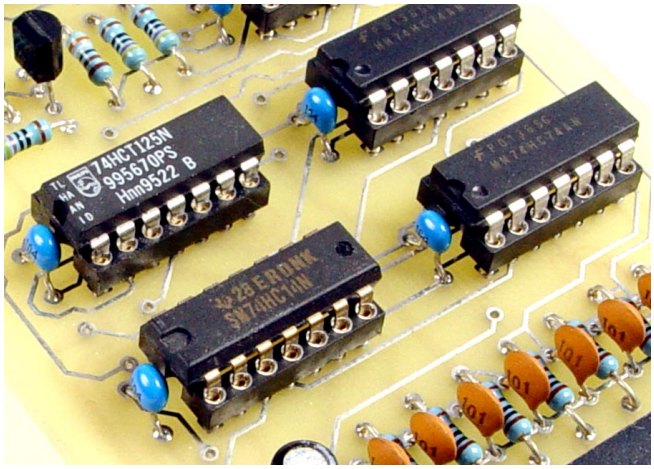
ไมโคร
คอนโทรลเลอร์

อาเขต บุญทาราม
arket@enmix.com



ในส่วนประมวลผลกลางจะประกอบด้วย 3 ส่วน คือหน่วยควบคุม, หน่วยประมวลผลทางคณิตศาสตร์ และรีจิสเตอร์ใช้งานภายใน

เรียนรู้ทดลองใช้งาน ST7FLITE09B ตอนที่ 2

จากตอนที่แล้ว เราได้กล่าวถึงรีจิสเตอร์ที่ใช้งานแบบพิเศษต่างๆ ซึ่งแต่ละตัวจะมีหน้าที่เฉพาะแตกต่างกัน ในตอนที่ 2 นี้ จะกล่าวถึงหน่วยประมวลผลกลาง ซึ่งจะเป็นส่วนสำคัญที่อยู่ภายในตัวไมโครคอนโทรลเลอร์ตระกูล ST7

ในส่วนประมวลผลกลางนี้ จะประกอบด้วยส่วนหลักๆ อยู่ 3 ส่วน คือ หน่วยควบคุม (CPU Control), หน่วยประมวลผลทางคณิตศาสตร์ (Arithmetic and Logic Unit ALU) และรีจิสเตอร์ใช้งานภายใน (CPU Registers) ดังรูปที่ 1

ส่วนที่ 1 จะเป็นส่วนที่ใช้ควบคุมระบบการทำงานภายในตัวไมโครคอนโทรลเลอร์ทั้งหมด

ส่วนที่ 2 จะเป็นส่วนที่ใช้ในการประมวลผลทางคณิตศาสตร์และทางลอจิก

ส่วนที่ 3 จะเป็นรีจิสเตอร์ที่อยู่ภายใน ซึ่งแต่ละตัวก็จะทำหน้าที่เฉพาะตามที่ถูกกำหนดเอาไว้

รีจิสเตอร์ภายในส่วนประมวลผลกลาง

เป็นรีจิสเตอร์เฉพาะอีกแบบหนึ่งซึ่งตัวรีจิสเตอร์ในกลุ่มนี้ในแต่ละตัวก็จะมีหน้าที่การทำงานเฉพาะที่แตกต่างกัน ซึ่งภายในกลุ่มของซีพียูรีจิสเตอร์นี้ จะประกอบด้วยซีพียูรีจิสเตอร์จำนวน 6 ตัว ซึ่งจะมีชื่อ, โครงสร้างและค่าที่เกิดขึ้นหลังการรีเซตดังรูปที่ 2

ซึ่งจะมีรายละเอียดเพิ่มเติมตามหัวข้อลำดับถัดไปดังนี้

รีจิสเตอร์แอกคิวมูเลเตอร์

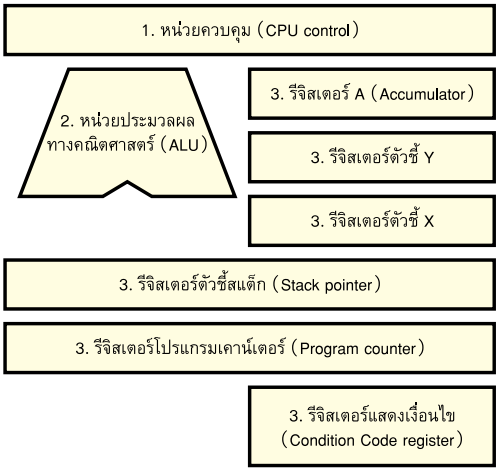
Accumulator : A เป็นรีจิสเตอร์ขนาด 8 บิต โดยจะเป็นรีจิสเตอร์ที่ใช้งานทั่วๆ ไป เช่น การส่งผ่านค่าข้อมูล การเก็บค่าผลลัพธ์หลังการประมวลผล การพักค่าก่อนและหลังการประมวลผล เป็นต้น

ซึ่งตัวซีพียูจะใช้รีจิสเตอร์นี้ในการทำงานดังกล่าวเป็นหลัก และค่าข้อมูลภายหลังการรีเซตจะมีค่าที่ไม่แน่นอน

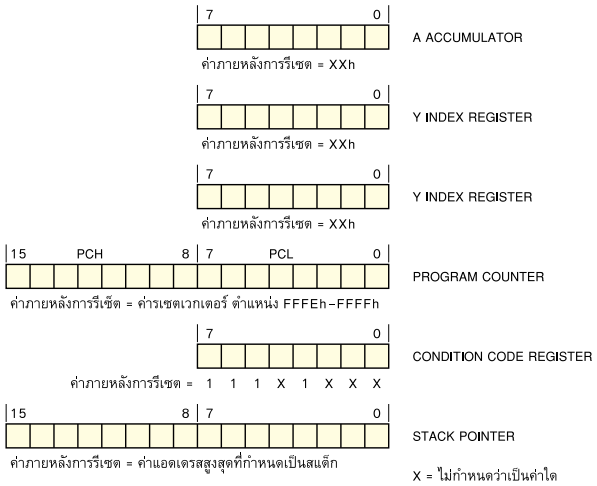
รีจิสเตอร์ตัวชี้

Index Register : X, Y เป็นรีจิสเตอร์ขนาด 8 บิต จำนวน 2 ตัว คือรีจิสเตอร์ X และรีจิสเตอร์ Y โดยรีจิสเตอร์ทั้ง 2 ตัวนี้ มีหน้าที่หลักๆ คือ จะเป็นตัวชี้ตำแหน่งในหน่วยความจำ นอกจากนี้เรายังสามารถจะใช้รีจิสเตอร์ทั้ง 2 ตัว มาเป็นรีจิสเตอร์เก็บข้อมูลทั่วไปได้อีกด้วย

หมายเหตุ ในรีจิสเตอร์ Y จะไม่ถูกเก็บแบบอัตโนมัติลงสแต็ก เมื่อมีการเรียกใช้โปรแกรมย่อยโดยคำสั่ง CALL หรือเกิดการอินเตอร์รัปต์



รูปที่ 1 แสดงส่วนต่าง ๆ ของหน่วยประมวลผลกลาง



รูปที่ 2 แสดงชื่อ, โครงสร้างและค่าที่เกิดขึ้นภายหลังการรีเซต ซีพียูรีจิสเตอร์

รีจิสเตอร์โปรแกรมเคาน์เตอร์

Program Counter : PC เป็นรีจิสเตอร์ขนาด 16 บิต ประกอบด้วยรีจิสเตอร์ขนาด 8 บิต จำนวน 2 ตัว คือ รีจิสเตอร์ PCH เป็นไบต์สูง (8 บิตบน) และรีจิสเตอร์ PCL เป็นไบต์ต่ำ (8 บิตล่าง) โดยมีหน้าที่ คือ จะทำการเก็บค่าตำแหน่งของคำสั่งที่ตัวซีพียูต้องกระทำต่อไปซึ่งจะเพิ่มค่าแบบอัตโนมัติ

ถ้าเกิดการรีเซตขึ้น ตัวรีจิสเตอร์นี้จะถูกโหลดให้เก็บค่าที่ตำแหน่ง FFFEh ดังนั้นในตำแหน่งนี้ จะถือว่าเป็นจุดเริ่มต้นของโปรแกรม เราต้องเขียนโปรแกรมเริ่มต้นที่จุดนี้ เพื่อให้ตัวซีพียูทำงานและดำเนินงานตามคำสั่งที่เราต้องการต่อไป

รีจิสเตอร์แสดงเงื่อนไข

Condition Code Register : CC เป็นรีจิสเตอร์ที่จะแสดงถึงเงื่อนไขต่าง ๆ จากการทำงานหรือการประมวลผล ซึ่งสามารถเข้าถึงได้ในระดับบิตโดยจะมีบิตที่เราใช้งานเพื่อดูผลอยู่ 5 บิต คือ บิต 0 ถึงบิต 4 ส่วนบิต 5 ถึงบิต 7 จะไม่ใช้งานและจะมีการแทนค่าด้วยลอจิก “1” ตลอดเวลา

7							0
1	1	1	H	I	N	Z	C

ค่าหลังการรีเซต = 111X1XXX

บิตที่ 4 = H (Half carry)

บิตนี้จะถูกเซตให้เป็นลอจิก “1” ด้วยฮาร์ดแวร์ เมื่อเกิดเหตุการณ์ที่มีตัวทดขึ้นระหว่างบิตที่ 3 และบิตที่ 4 จากการกระทำของคำสั่ง ADD หรือ ADC

0 = ไม่มีตัวทดเกิดขึ้น

1 = มีตัวทดเกิดขึ้น

เมื่อเราต้องการที่จะตรวจสอบบิต H นี้ เราสามารถทำการตรวจสอบได้จากคำสั่ง JRH หรือ JRNH ในบิต H นี้ จะถูกใช้ในเรื่องของการประมวลผลรหัส BCD

บิตที่ 3 = I (Interrupt mask)

เป็นบิตที่ใช้สำหรับการเปิดหรือปิดรับการอินเตอร์รัปต์ ซึ่งบิตนี้ในสภาวะหลังการรีเซตจะถูกเขียนให้เป็นลอจิก “1” หรือหมายความว่า ไม่ต้องการให้มีการเปิดรับการอินเตอร์รัปต์ใด ๆ เมื่อต้องการให้มีการเปิดรับการอินเตอร์รัปต์ก็สามารถทำได้โดยให้บิตนี้เป็นลอจิก “0” โดยสามารถสั่งงานได้จากโปรแกรม

0 = เปิดรับการอินเตอร์รัปต์

1 = ปิดรับการอินเตอร์รัปต์

ในบิตนี้จะสามารถควบคุมโดยการใช้คำสั่ง RIM, SIM และ IRET ในการตรวจสอบบิตนี้สามารถกระทำได้จากคำสั่ง JRM และ JRNM

บิตที่ 2 = N (Negative)

บิตนี้จะมีการเปลี่ยนแปลงเป็นลอจิก “0” หรือลอจิก “1” จากการพิจารณาผลลัพธ์สุดท้ายที่เกิดขึ้นจากการประมวลผลข้อมูลทางคณิตศาสตร์หรือทางลอจิก โดยพิจารณาผลที่เกิดขึ้นจากบิตที่ 7 ของตัวรีจิสเตอร์แอกคิวมูเลเตอร์ A

0 = ข้อมูลผลลัพธ์ที่เกิดขึ้นมีค่าเป็นบวก

1 = ข้อมูลผลลัพธ์ที่เกิดขึ้นมีค่าเป็นลบ

บิตนี้สามารถทำการตรวจสอบได้โดยคำสั่ง JRMI และ JRPL

บิตที่ 1 = Z (Zero)

บิตนี้จะมีการเปลี่ยนแปลงเป็นลอจิก “0” หรือลอจิก “1” จากการพิจารณาผลลัพธ์สุดท้ายที่เกิดขึ้นจากการประมวลผลข้อมูลทางคณิตศาสตร์หรือทางลอจิก โดยพิจารณาผลลัพธ์ที่

เกิดขึ้นภายในรีจิสเตอร์แอกคิวมูเลเตอร์ A

0 = ผลลัพธ์ที่เกิดขึ้นมีค่าไม่เป็นศูนย์

1 = ผลลัพธ์ที่เกิดขึ้นมีค่าเป็นศูนย์

บิตนี้สามารถทำการตรวจสอบได้โดยการใช้คำสั่ง JREQ และ JRNE

บิตที่ 0 = C (Carry/Borrow)

บิตนี้จะมีการเปลี่ยนแปลงเป็นลอจิก “0” หรือลอจิก “1” จากการพิจารณาผลลัพธ์ที่เกิดจากการประมวลผลว่า มีตัวทดจากการบวกเลขหรือตัวยืมจากการลบเลข บิตนี้สามารถเปลี่ยนแปลงได้โดยคำสั่งการหมุนข้อมูลหรือเลื่อนข้อมูลผ่านบิต C

0 = ไม่เกิดการ Overflow หรือ Underflow ขึ้น

1 = เกิดการ Overflow หรือ Underflow ขึ้น

ซึ่งบิตนี้สามารถทำให้มีลอจิก “0” หรือลอจิก “1” ได้ด้วยคำสั่ง SCF และ RCF ในการตรวจสอบสามารถทำได้โดยใช้คำสั่ง JRC และ JRNC

รีจิสเตอร์ตัวชี้สแต็ก

Stack Pointer : SP เป็นรีจิสเตอร์ขนาด 16 บิต ซึ่งจะใช้เป็นตัวชี้ในการเก็บค่าต่างๆ ลงในหน่วยความจำ เมื่อมีการเรียกใช้คำสั่ง CALL หรือเมื่อเกิดการอินเตอร์รัปต์ขึ้น ในตัวไมโครคอนโทรลเลอร์ ST7 มีการเตรียมเนื้อที่ส่วนนี้เอาไว้ในช่วง 00C0h ถึง 00FFh จำนวน 64 ไบต์

15							8
0	0	0	0	0	0	0	0

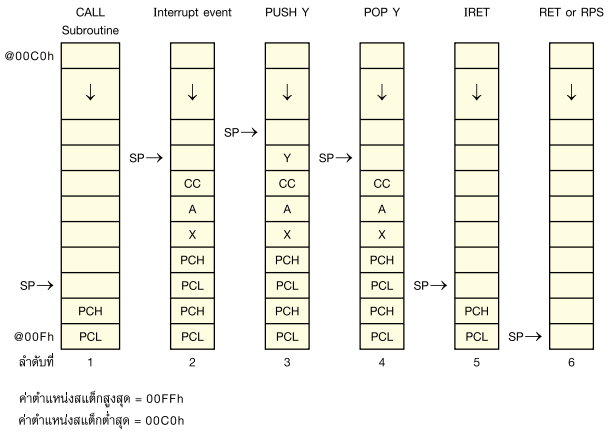
7							0
1	1	SP5	SP4	SP3	SP2	SP1	SP0

ค่าภายหลังการรีเซต 00 FFh

การเก็บข้อมูลลงในสแต็กจะมีการเก็บข้อมูลดังรูปที่ 3 ซึ่งเป็นลักษณะการจำลองเหตุการณ์ของการเก็บค่าลงหน่วยความจำสแต็ก เมื่อมีการเรียกใช้งานโปรแกรมย่อยหรือเกิดการอินเตอร์รัปต์ ซึ่งข้อมูลที่ถูกเก็บเข้าในสแต็กจะเป็นการเก็บข้อมูลเข้าแบบที่เรียกว่า เข้าหลังออกก่อน

การนำข้อมูลเข้าสแต็ก

ในลำดับที่ 1 ถ้ามีการเรียกใช้คำสั่ง CALL จากโปรแกรม ตัวชี้สแต็กจะทำการเก็บค่าตำแหน่งของรีจิสเตอร์โปรแกรมเคาน์เตอร์ โดยจะทำการเก็บค่า 8 บิตล่าง (PCL) ก่อน แล้วตามด้วยการเก็บค่า 8 บิตบน (PCH)



รูปที่ 3 การแสดงการเก็บค่าข้อมูลของสแต็ก

ในลำดับที่ 2 ถ้ามีการอินเตอร์รัปต์ขึ้น ค่าในรีจิสเตอร์ดังในลำดับที่ 2 จะถูกเก็บลงในสแต็กแบบอัตโนมัติ ได้แก่ รีจิสเตอร์ PC, รีจิสเตอร์ X, รีจิสเตอร์ A, รีจิสเตอร์ CC, ส่วนในรีจิสเตอร์ Y จะไม่มีการเก็บค่าลงในสแต็กแบบอัตโนมัติ

ในลำดับที่ 3 เป็นการเก็บค่ารีจิสเตอร์ Y ลงในสแต็ก โดยคำสั่ง PUSH Y

การนำข้อมูลออกจากสแต็ก

ในลำดับที่ 4 เป็นการนำค่ารีจิสเตอร์ Y ออกจากสแต็กโดยคำสั่ง POP Y แต่ถ้าเราไม่มีการเก็บค่ารีจิสเตอร์ Y เอาไว้ก็ไม่จำเป็นต้องทำขั้นตอนในลำดับนี้

ในลำดับที่ 5 เป็นการออกจากการบริการอินเตอร์รัปต์โดยการใช้คำสั่ง IRET รีจิสเตอร์จะถูกนำออกจากสแต็กแบบเป็นลำดับ โดยจะเป็นลักษณะการเอาข้อมูลออกแบบเข้าที่หลังให้ออกก่อน

ในลำดับที่ 6 เป็นการออกจากการบริการโปรแกรมย่อยโดยคำสั่ง RET จะมีการคืนค่ารีจิสเตอร์โปรแกรมเคาน์เตอร์ออกจากสแต็ก

ที่กล่าวมาก็เป็นเรื่องเกี่ยวกับหน่วยประมวลผลกลาง ซึ่งเป็นส่วนสำคัญภายในของตัวไมโครคอนโทรลเลอร์ตระกูล ST7 ในตอนต่อไปก็จะกล่าวถึงส่วนของการเขียนโปรแกรมและคำสั่งที่ใช้งานต่างๆ ครับ