สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

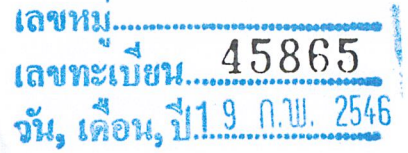
ระบบสื่อการสอนสำหรับการเรียนรู้ไมโครคอนโทรลเลอร์ 8051

CAI SYSTEM FOR LEARNING 8051 MICROCONTROLLER



นายนำพล สายสุวรรณ
นายโสพส สังข์สวัสดิ์
นายสมบูรณ์ ชมภู

ปพ.
พ 52ธ
25AA





ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

I-M08194
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

CAI SYSTEM FOR LEARNING 8051 MICROCONTROLLER

NAMPHOL SAISUWAN

SOROSE SANGSAWAD

SOMBOON CHOMPOO

A THESIS SUBMITTED IN PATIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTURMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2001

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	**ระบบสื่อการสอนสำหรับการเรียนรู้ไมโครคอนโทรลเลอร์ 8051**			
	CAI SYSTEM FOR LEARNING 8051 MICROCONTROLLER			
นักศึกษาผู้จัดทำ	นายนำพล	สายสุวรรณ	**รหัสประจำตัว**	42015389
	นายโสพส	สังข์สวัสดิ์	**รหัสประจำตัว**	42015401
	นายสมบูรณ์	ชมภู	**รหัสประจำตัว**	42015406
ปริญญา	วิศวกรรมศาสตรบัณฑิต			
สาขาวิชา	วิศวกรรมการวัดคุม			
ปีการศึกษา	2544			

อาจารย์ผู้ควบคุมปริญญานิพนธ์	**ลายมือชื่อ**
ผศ.ทรงชัย วีระทวีมาศ	

วัน/เดือน/ปี ที่สอบ วันที่พุธที่ 27 มีนาคม พศ.2545

สถานที่สอบ ณ. ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรมการวัดคุม

ภาควิชารับรองแล้ว

(ผศ.ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	**ระบบสื่อการสอนสำหรับการเรียนรู้ไมโครคอนโทรลเลอร์ 8051**	
	CAI SYSTEM FOR LEARNING 8051 MICROCONTROLLER	
นักศึกษาผู้จัดทำ	นายนำพล	สายสุวรรณ
	นายโสพส	สังข์สวัสดิ์
	นายสมบูรณ์	ชมพู
อาจารย์ที่ปรึกษา	ผศ.ทรงชัย	วีระทวีมาศ
ปีการศึกษา	2544	

บทคัดย่อ

ปริญญานิพนธ์นี้ได้จัดทำขึ้นเพื่อ อธิบายการทำงานของไมโครคอนโทรลเลอร์ชิปเดี่ยวตระกูล 8051 และรวมทั้ง ตัวอย่างการประยุกต์ใช้งาน โดยใช้การนำเสนอโดยผ่านทางเครือข่ายอินเตอร์เน็ต ซึ่งใช้โปรแกรมประยุกต์ต่างๆ ในการสร้างภาพเคลื่อนไหว ที่สามารถโต้ตอบกับผู้ใช้งานเพื่อ ให้ง่ายต่อการทำความเข้าใจด้วยตนเอง ทั้งนี้เพราะการที่จะเรียนรู้การทำงานของไมโครคอนโทรลเลอร์ ด้วยตนเองเป็นเรื่องที่ค่อนข้างยากสำหรับบุคคลที่ไม่มีพื้นฐานทางด้านนี้ ดังนั้นการจำลองการทำงานแบบให้เห็นการทำงานจริงๆ อย่างเป็นขั้นตอน จะทำให้สามารถเข้าใจการทำงานอย่างแท้จริง และในส่วนของตัวอย่างการประยุกต์ใช้งาน ก็จะเป็นการทำให้เข้าใจถึงการใช้งานร่วมกับอุปกรณ์ต่างๆ เช่น LED, 7 Segments, Stepping motor ว่าสามารถทำได้อย่างไร และสามารถปรับเปลี่ยนค่าบางค่าที่เรากำหนดให้ เพื่อดูผลของการเปลี่ยนแปลงที่เกิดขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title	CAI system for learning 8051 microcontroller
Authors	Mr.Namphol Saisuwan
	Mr.Sorose Sangsawad
	Mr.Somboon Chompoo
Thesis Advisor	Ass.Prof.Songchai Weerathaweemas
Year	2001

ABSTRACT

This paper presents the description for single chip micro controller in 8051 family and including example of development for using it. We present it through the Internet network by using the application programs for making the animation to response with user for easily to study by yourself. Because it is quite difficult when you study micro controller by yourself if you didn't have a background about it. Thus, A CAI on micro controller step by step will make you clear about it . And in the section of example of development for using will make you know how to join micro controller with another equipments such as LED, 7 Segments, Stepping motor and you can change some value for seeing the exchanging of result.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ในครั้งนี้ สำเร็จลุล่วงไปได้ด้วยดี เพราะได้รับความเมตตาและความช่วยเหลือจาก ผศ.ทรงชัย วีระทวีมาศ ที่ได้ให้ความกรุณาแนะนำแก่ผู้ทำการวิจัยตลอดมา อีกทั้งยังเอื้อเฟื้อหนังสือและเอกสารต่างๆ ในการทำปริญญานิพนธ์ฉบับนี้ ผู้ทำการวิจัยขอกราบขอบพระคุณเป็นอย่างสูง

และขอกราบขอบพระคุณ คุณพ่อ และคุณแม่ ที่เคารพรักยิ่ง ตลอดจนสมาชิกในครอบครัว พี่ๆ น้องๆ และเพื่อนร่วมงานทุกคนที่ให้การสนับสนุน และ เป็นกำลังใจที่ยิ่งใหญ่ในการทำปริญญานิพนธ์ฉบับนี้

สุดท้ายนี้ขอขอบคุณ ภาควิชาวิศวกรรมการวัดคุม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่มอบโอกาสให้ได้มีการทำโครงงานวิจัย เพื่อเสริมสร้างทักษะและความรู้ให้กับผู้ทำการวิจัยและให้การสนับสนุนจนทำให้โครงงานวิจัยและปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยดี

คณะผู้จัดทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VIII

สารบัญภาพ X

บทที่ 1 บทนำ 1

1.1 ความเป็นมา 1

1.2 วัตถุประสงค์ของปริญญานิพนธ์ 2

1.3 ขอบเขตของปริญญานิพนธ์ 2

1.4 ขั้นตอนการศึกษา 2

บทที่ 2 หลักการและทฤษฎีที่เกี่ยวข้อง 3

2.1 หลักเบื้องต้นในการสร้างเว็บเพจด้วย Dreamweaver 4

2.1.1 ความสามารถ 4

2.2.2 ความต้องการทางฮาร์ดแวร์ 4

2.2.3 องค์ประกอบต่างๆ 5

2.2 หลักการสร้างภาพเคลื่อนไหวและมัลติมีเดียด้วย Flash 6

2.2.1 ความสามารถ 7

2.2.2 องค์ประกอบต่างๆ 7

2.2.3 หลักการทำงาน 10

2.3 การอัพโหลดไฟล์สู่เว็บไวต์ 11

บทที่ 3 ไมโครคอนโทรลเลอร์ตระกูล MCS-51 12

3.1 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51 12

3.2 โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51 13

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

3.3 การแบ่งหน่วยความจำ 14
3.4 หน่วยความจำโปรแกรม 15
3.5 หน่วยความจำสำหรับเก็บข้อมูล 16
3.5.1 หน่วยความจำข้อมูลภายใน 17
3.5.2 หน่วยความจำข้อมูลภายนอก 19
3.6 พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยทางตรงและทางอ้อม 19
3.7 การเข้าถึงข้อมูลในคำสั่ง 20
3.7.1 วิธีการเข้าถึงข้อมูลโดยตรง 21
3.7.2 วิธีการเข้าถึงข้อมูลโดยทางอ้อม 21
3.7.3 วิธีการเข้าถึงข้อมูลในรีจิสเตอร์ใช้งานทั่งไป 24
3.7.4 วิธีการเข้าถึงข้อมูลในรีจิสเตอร์เฉพาะของคำสั่ง 26
3.7.5 วิธีการเข้าถึงข้อมูลที่กำหนดเองโดยตรง 26
3.7.6 วิธีการเข้าถึงข้อมูลโดยใช้ตัวอ้างอิง 26
3.8 ไทเมอร์/เคาน์เตอร์ 27
3.9 โครงสร้างการอินเตอร์รัพท์ 31
3.10 พอร์ตของ 8051 35

บทที่ 4 ชุดคำสั่งของ MCS-51 37
4.1 กลุ่มคำสั่งทางคณิตศาสตร์ 37
4.1.1 กลุ่มคำสั่งเพิ่มหรือลดค่าข้อมูลครั้งละหนึ่ง 37
4.1.2 กลุ่มคำสั่งบวกเลขจำนวนเต็ม 38
4.1.3 กลุ่มคำสั่งลบเลขจำนวนเต็ม 41
4.1.4 กลุ่มคำสั่งคูณและหาร 43
4.1.5 กลุ่มคำสั่งที่ปรับค่าผลลัพธ์จากการบวกเลขรหัส 44
4.2 กลุ่มคำสั่งทางตรรกศาสตร์ 46
4.3 กลุ่มคำสั่งในการเคลื่อนย้ายข้อมูล 49

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

4.3.1 กลุ่มคำสั่งในการเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป 55

4.3.2 กลุ่มคำสั่งในการเคลื่อนย้ายข้อมูลจากหน่วยความจำสำหรับเก็บโปรแกรมทั้งภายนอกและภายในชิป 55

4.4 กลุ่มคำสั่งควบคุมลำดับการทำงานของโปรแกรม 56

4.4.1 กลุ่มคำสั่งควบคุมโปรแกรมแบบไม่มีเงื่อนไข 57

4.4.2 กลุ่มคำสั่งควบคุมโปรแกรมแบบมีเงื่อนไข 66

4.5 กลุ่มคำสั่งประมวณผลแบบบูลีน 67

4.5.1 กลุ่มคำสั่งควบคุมสถานะบิต 67

4.5.2 กลุ่มคำสั่งเคลื่อนย้ายข้อมูล 69

4.5.3 กลุ่มคำสั่งทางตรรกศาสตร์ 69

4.5.4 กลุ่มคำสั่งควบคุมโปรแกรมที่ขึ้นกับสถานะของบิต 69

4.6 คำสั่งที่มีผลต่อการทำงานของกลุ่มคำสั่งประมวณผลแบบบูลีน 70

บทที่ 5 การทดลองใช้งาน 71

5.1 สถาปัตยกรรมภายนอก 73

5.1.1 ตระกูล MCS-51 73

5.1.2 สัญญาณขา 73

5.1.3 หน่วยความจำ 74

5.1.4 อินเตอร์รัพท์ 75

5.1.5 เวลา/ตัวนับ 75

5.1.6 พอร์ต 75

5.1.7 อุปกรณ์ต่อร่วม 8255 75

5.2 สถาปัตยกรรมภายใน 75

5.2.1 สถาปัตยกรรมภายใน 75

5.2.2 การอ้างแอดเดรส 75

5.2.3 กลุ่มคำสั่งโอนย้าย 76

5.2.4 กลุ่มคำสั่งคณิตศาสตร์ 76

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

5.2.5 กลุ่มคำสั่งลอจิก 76

5.2.6 กลุ่มคำสั่งบูลลีน 76

5.2.7 กลุ่มคำสั่งกระโดด 76

5.2.8 คำสั่งเสริมใน Assembler 78

5.3 ตัวอย่างการใช้งาน 78

5.3.1 PROJECT 1 78

5.3.2 PROJECT 2 80

5.3.3 PROJECT 3 80

5.3.4 PROJECT 4 80

5.3.5 PROJECT 5 82

บรรณานุกรม 83

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตารางที่ หน้า

3.1 แสดงการเลือกโหมดการทำงานของ TIMER/COUNTER โดย บิต M1, M2 30

3.2 แสดงตำแหน่งเริ่มต้นของโปรแกรมบริการอินเตอร์รัปต์แต่ละชนิดที่เกิดขึ้น 35

4.1 แสดงกลุ่มคำสั่งเพิ่มหรือลดค่าข้อมูลครั้งละหนึ่ง 37

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ

ภาพที่ หน้า

3.1 แสดงโครงสร้างภายในของชิปไมโครคอนโทรลเลอร์ตระกูล MCS-51 14

3.2 แสดงการจัดโครงสร้างของหน่วยความจำทั้งในหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล 15

3.3 การจัดพื้นที่หน่วยความจำโปรแกรมสำหรับไมโครคอนโทรลเลอร์ 8051 16

3.4 การจัดพื้นที่หน่วยตวามจำข้อมูลสำหรับไมโครคอนโทรลเลอร์ 8051 17

3.5 แสดงหน่วยความจำสำหรับเก็บข้อมูลทั้ง 3 ส่วน 18

3.6 แสดงการเลือกใช้กลุ่มรีจีสเตอร์ใช้งานทั่งไป โดยควบคุมจากบิต RS0, RS1 19

3.7 ไบต์ของ RAM ที่เข้าถึงข้อมูลแบบทางตรงและทางอ้อม 20

3.8 แสดงการเลือกใช้รีจีสตอร์ใช้งานทั่วไปแต่ละกลุ่มจากบิต RS0, RS1 25

3.9 แสดงวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 1 ในโหมด 0 และโหมด 1 28

3.10 แสดงวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 1 ในโหมด 2 28

3.11 แสดงวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 0 ในโหมด 3 29

3.12 แสดงบิตควบคุมที่อยู่ในรีจีสเตอร์ TMOD 29

3.13 แสดงบิตควบคุมที่อยู่ในรีจีสเตอร์ TCON 30

3.14 หน้าที่การทำงานของแต่ละบิตในรีจีสเตอร์ IE 31

3.15 หน้าที่การทำงานของแต่ละบิตในรีจีสเตอร์ TCON 32

3.16 หน้าที่การทำงานของแต่ละบิตในรีจีสเตอร์ IP 34

4.1 การใช้เลขไบนารีขนาด 8 บิตแทนจำนวนเต็มบวกอย่างเดียวและเลขจำนวนเต็มบวกและเลขจำนวนเต็มลบรวมกัน 39

4.2 แสดงตัวอย่างการคูณ 44

4.3 แสดงตัวอย่างการหาร 44

4.4 แสดงคำสั่งในการเคลื่อนข้อมูลเป็นวงรอบ 49

4.5 แสดงการทำงานของ PUSH Direct 52

4.6 แสดงค่าของรีจีสเตอร์ SP เมื่อตอนรีเซต หรือเริ่มจ่ายพลังงานให้ MCS-51 52

4.7 แสดงการทำงานของ POP Direct 53

4.8 แสดงการทำงานของรีจีสเตอร์ PC 56

4.9 แสดงการทำงานของคำสั่ง LJMP addr 58

4.10 แสดงคำสั่ง AJMP addr 59

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ(ต่อ)

ภาพที่ หน้า

4.11 ขอบเขตในการกระโดดไปทำงานของคำสั่ง AJMP 60
4.12 แสดงส่วนประกอบบล็อกหน่วยความจำทั้ง 32 บล็อก 61
4.13 แสดงการทำงานของคำสั่ง SJMP 62
4.14 แสดงการย้ายโปรแกรมเป็นบล็อกเพื่อให้คำสั่งยังใช้ได้เหมือนเดิม 63
4.15 แสดงการทำงานของคำสั่ง CALL และ RET 64
4.16 แสดงลำดับการทำงานของคำสั่ง CALL และ RET 64
5.1 แสดงหน้า Introduction ของเว็บไซต์ 71
5.2 แสดงหน้าโฮมเพจของเว็บไซต์ 72
5.3 แสดงลักษณะของเมนูย่อย 72
5.4 แสดงหน้าสัญญาณขาเมื่อนำเมาท์มาวางที่ขาของไอซี 73
5.5 แสดงหน้ารายละเอียดของสัญญาณขาเมื่อคลิกที่ขาของไอซี 74
5.6 แสดงหน้าสัญญาณขาเมื่อมีการติดต่อกับหน่วยความจำภายนอก 74
5.7 แสดงหน้ารายละเอียดสถาปัตยกรรมภายใน 76
5.8 แสดงหน้าตารางคุณลักษณะของกลุ่มคำสั่งต่างๆ 77
5.9 แสดงหน้ารายละเอียดของคำสั่งที่ถูกเลือกจากตาราง 77
5.10 แสดงหน้าการทำงานของคำสั่ง 78
5.11 แสดงหน้าอธิบายตัวอย่างโปรแกรม 79
5.12 แสดงหน้าอธิบายการทำงานของโปรแกรม 79
5.13 แสดงหน้าวงจรการต่อใช้งานควบคุมหลอด LED 80
5.14 แสดงหน้าโปรแกรมที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ 81
5.15 แสดงหน้าวงจรการต่อใช้งานแสดงผลตัวเลขเจ็ดส่วน 81
5.16 แสดงหน้าวงจรการต่อใช้งานควบคุมสเตปปิงมอเตอร์ 82

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมา

ปัจจุบันนี้มีการนำเอา ระบบไมโครคอนโทรลเลอร์เข้ามาใช้งานเป็นส่วนสำคัญในวงการอุตสาหกรรมกันอย่างแพร่หลาย โดยเริ่มตั้งแต่ Z80 ไมโครโพรเซสเซอร์ ขนาด 8 บิต ในอดีตเป็นต้นมา แม้กระทั่งในปัจจุบัน Z80 ไมโครโพรเซสเซอร์ ยังคงเป็นที่นิยมสำหรับผู้ที่เริ่มต้นศึกษาเกี่ยวกับเรื่องไมโครโพรเซสเซอร์อยู่ ทั้งนี้เพราะ Z80 มีโครงสร้างที่ไม่ซับซ้อน และใช้คำสั่งที่เข้าใจง่าย และในขณะเดียวกันไมโครคอนโทรลเลอร์ขนาด 8 บิต ตระกูล MCS-51 ของบริษัท INTEL ก็เริ่มเข้ามามีบทบาทสำคัญต่อการศึกษา และการนำไปประยุกต์ใช้ในงานอุตสาหกรรม ดังจะเห็นได้จากการที่สถานศึกษาหลายๆ แห่งได้จัดให้มีการเรียนการสอนเกี่ยวกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 ให้กับนักศึกษา เหตุผลประการสำคัญที่ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เข้ามามีบทบาทอย่างรวดเร็ว และมีผู้นำไปประยุกต์ใช้งานกันอย่างแพร่หลายนั้น เนื่องจากไมโครคอนโทรลเลอร์ตระกูล MCS-51 นี้มีขอบเขตและความสามารถสูงกว่าไมโครโปรเซสเซอร์ธรรมดาทั่วๆ ไป

เนื่องจากทางภาควิชาได้มีหลักสูตรการเรียนการสอน เกี่ยวกับ ไมโครคอนโทรลเลอร์ ซึ่งนักศึกษาไม่มีความรู้พื้นฐานเกี่ยวกับไมโครคอนโทรลเลอร์มากนัก นักศึกษาจะต้องสามารถทำความเข้าใจกับการทำงานอย่างลึกซึ้ง ดังนั้นการเรียนรู้อย่างเป็นขั้นเป็นตอน และมีการใช้ภาพเคลื่อนไหวที่สามารถ โต้ตอบเพื่อเพิ่มความเข้าใจ รวมทั้งเรื่องของการดูผลการทดลองที่เปลี่ยนแปลงไปจากการเปลี่ยนค่าของตัวแปรต่างๆ จึงเป็นสิ่งจำเป็น

และเนื่องจากในปัจจุบันคอมพิวเตอร์ได้เข้ามามีบทบาทอย่างมาก ในชีวิตประจำวัน รวมทั้งเข้ามามีบทบาททางด้านการศึกษา และการที่มีการเชื่อมโยงกันเป็นเครือข่ายอินเตอร์เน็ต ทำให้เราใช้คอมพิวเตอร์ในการเป็น แหล่งความรู้ อินเตอร์เน็ตเหมือนเป็นแหล่งความรู้ ที่มีข้อมูลมากมายที่เราสามารถนำมาใช้ได้ ซึ่งไม่เป็นเพียงข้อความเท่านั้น แต่มีทั้งเสียง ภาพ ภาพยนตร์ แหล่งข่าวสารความบันเทิง เราสามารถติดตามข่าวล่าสุด ดูหนังฟังเพลง และภาพยนต์ล่าสุด ไม่ว่าจากในประเทศหรือต่างประเทศได้ และจากการที่สามารถจะหาความรู้ผ่านทางอินเตอร์เน็ตนี้เอง เราจึงได้ทำการนำเสนอการอธิบายการจำลองการทำงานของไมโครคอนโทรลเลอร์เบอร์ 8051 รวมทั้งตัวอย่างการประยุกต์ใช้งานผ่านทางระบบอินเตอร์เน็ต เพื่อให้คนที่จะสนใจที่จะเรียนรู้ได้เข้ามาศึกษาได้ด้วยตนเองโดยใช้ภาพเคลื่อนไหวที่สามารถโต้ตอบ กับผู้ใช้งานทำให้สามารถดูขั้นตอนการทำงานอย่างเป็นลำดับขั้นตอน ทำให้สามารถเข้าใจการทำงานอย่างแท้จริง โดยผู้ใช้ไม่จำเป็นต้องมีความรู้พื้นฐานทางด้านนี้มากนัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1 เพื่อศึกษาโครงสร้างสถาปัตยกรรมภายใน ขั้นตอนการทำงาน รวมทั้งการใช้งานกลุ่มคำสั่งต่างๆ ของไมโครคอนโทรลเลอร์เบอร์ 8051

2 สามารถใช้โปรแกรมประยุกต์ต่างๆ ในการทำเว็บเพจ รวมทั้งการทำภาพเคลื่อนไหวที่สามารถตอบโต้กับผู้ใช้งานได้

3 เข้าใจหลักการที่จะสร้างโฮมเพจของตนเอง รวมทั้งการนำงานที่สร้างแล้วขึ้นไปแสดงบนเครือข่ายอินเตอร์เน็ต

4 เป็นแหล่งความรู้เกี่ยวกับไมโครคอนโทรเลอร์เบอร์ 8051 ที่สามารถศึกษาทำความเข้าใจด้วยตนเอง สำหรับผู้สนใจทั่วไป

1.3 ขอบเขตของปริญญานิพนธ์

ทำการสร้างเว็บเพจอธิบายการทำงานของไมโครคอนโทรลเลอร์เบอร์ 8051 ทั้งหมด โดยการใช้ภาพเคลื่อนไหวที่สามารถโต้ตอบกับผู้ใช้งานได้ ซึ่งมีขอบเขตการทำงานดังนี้

1 สามารถที่จะเขียนเว็บเพจที่อธิบายการทำงานของไมโครคอนโทรลเลอร์เบอร์8051

2 สามารถสร้างภาพเคลื่อนไหวแสดงการทำงานของคำสั่งต่างๆ โดยแสดงลำดับการทำงานเป็นขั้นตอนอย่างละเอียด

3 สามารถสร้างภาพเคลื่อนไหวประกอบคำอธิบายที่สามารถโต้ตอบกับผู้ใช้งานได้ ในการอธิบายการประยุกต์ใช้งานของไมโครคอนโทรลเลอร์ร่วมกับอุปกรณ์ต่างๆ

4 สามารถนำงานที่ทำทั้งหมดขึ้นแสดงบนเครือข่ายอินเตอร์เน็ต

1.4 ขั้นตอนการศึกษา

การที่จะต้องการสร้างเว็บเพจอธิบายการทำงาน ของไมโครคอนโทรลเลอร์ผ่านทางระบบอินเตอร์เนต จำเป็นต้องศึกษาขั้นตอนต่างๆดังนี้

1 ศึกษาเกี่ยวกับไมโครคอนโทรลเลอร์เบอร์ 8051 ทั้งหมด

2 กำหนดขอบเขตของโครงงานทั้งหมด

3 ทำการเลือกโปรแกรมประยุกต์ที่สามารถทำงานตามที่ต้องการได้

4 ศึกษาโปรแกรมที่ได้เลือกไว้ เพื่อที่จะสามารถนำมาใช้งานได้

5 เริ่มกำหนดโครงร่างของเว็บเพจ และเริ่มสร้างเว็บเพจ

6 นำเว็บเพจที่ได้ขึ้นแสดงบนระบบเครือข่ายอินเตอร์เน็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

WWW(World Wide Web) หรือที่เราเรียกสั้นๆ ว่าเว็บ การให้บริการรูปแบบนี้เรียกว่า บราวเซอร์ เช่น Internet Explorer หรือ Netscape จากเครื่องของเราและระบุ URL เพื่อใช้ในการอ้างอิงที่เก็บเว็บ เท่านี้เราก็สามารถเปิดดูเว็บได้แล้ว โดย URL ที่เราป้อนจะถูกส่งไปที่ DNS Server (Domain Name Server) ให้กลายเป็นชื่อแบบตัวเลขหรือ IP Address นั่นเอง โดยหมายเลข IP นั้นจะถูกใช้ในการอ้างอิงตำแหน่งเครื่องในอินเตอร์เน็ต โดยเครื่องทุกเครื่องที่อยู่ในอินเตอร์เน็ตจะมีหมายเลข IP ที่ไม่ซ้ำกันทำให้เราสามาระบุที่อยู่ของเครื่องที่เก็บเว็บที่เราต้องการเปิดดูได้ โดยเราอาจจะเปรียบหมายเลข IP เหมือนกับเลขที่บ้านของเราในการส่งจดหมายก็ได้ จากนั้นเมื่อเราได้เลข IP ที่ต้องการ คำร้องขอดูเว็บของเราก็จะถูกส่งไปที่เครื่องเซิร์ฟเวอร์ที่มีหมายเลข IP นั้น และเมื่อเครื่องเซิร์ฟเวอร์รับคำร้องจากเรา ก็จะส่งเว็บที่เก็บอยู่ในเครื่องเซิร์ฟเวอร์ มาแสดงผลที่เครื่องของเรา และนี่คือขั้นตอนของการใช้บริการ WWW

เอกสารที่เราเปิดดูใน WWW มีชื่อเรียกสั้นๆ ว่า "เว็บเพจ" ซึ่งส่วนใหญ่จะถูกสร้างขึ้นจากภาษาคอมพิวเตอร์ที่เรียกว่า HTML ซึ่งจะกำหนดรูปแบบและหน้าตาของเว็บเพจ ที่ปรากฏบนหน้าจอและส่วนที่เชื่อมต่อกับเว็บเพจอื่น WWW กำลังเติบโตอย่างรวดเร็ว ปัจจุบันมีเว็บเพจมากกว่า 4 ล้านหน้าใน WWW จนเรากล่าวได้ว่า WWW เป็นบริการที่แจ้งเกิดให้แก่ระบบอินเตอร์เน็ต เพราะใช้งานง่าย จึงทำให้มีความสนใจใช้งานกันอย่างกว้างขวาง โดยการใช้เว็บบราวเซอร์ซึ่งก็คือโปรแกรมที่เป็นประตูเข้าสู่โลก WWW ซึ่งเป็นโปรแกรมที่อยู่ในเครื่องฝั่ง Client มีหน้าที่ในการส่งข้อมูลร้องขอดูเว็บ และนำเสนอข้อมูเว็บ โดยตัวเว็บบราวเซอร์จะมีความเข้าใจในภาษามาตราฐานของเว็บคือภาษา HTML และสามารถแปลงภาษา HTML ให้เป็นหน้าเอกสารสวยงามให้เราชมได้ ปัจจุบันมีเว็บบราวเซอร์ของหลายค่ายได้แก่ Internet Explorer ของบริษัทไมโครซอฟต์ และ Netscape จากบริษัท Netscape Communications ในแต่ละเว็บไซต์จะประกอบด้วยเว็บเพจหลายหน้า โดยจะมีการกำหนดเว็บเพจหน้าหนึ่งให้เป็นหน้าแรก เว็บเพจหน้าที่มีชื่อเรียกว่า "โฮมเพจ" ซึ่งเป็นหน้าที่เหมือนกับปกหนังสือ กล่าวคือโฮมเพจเป็นทางเข้าของเว็บเพจทั้งหมดในเว็บไซต์นั้น โดยโฮมเพจจะสรุปเนื้อหาและเป็นเหมือนสารบัญของเว็บเพจทั้งหมดในเว็บไซต์นั้น ซึ่งเราจะพบกับส่วนที่เชื่อมโยงไปยังเว็บเพจอื่นๆ

เว็บเพจจำลองการทำงานของ 8051 ไมโครคอนโทรลเลอร์ จะประกอบไปด้วยส่วนของการอธิบายด้วยตัวอักษร และการใช้ภาพมัลติมีเดียที่สามารถโต้ตอบกับผู้ใช้ได้ โดยมากแล้วในส่วนของตัวอักษรเราจะใช้โปรแกรม Dreamweaver ในการสร้างเว็บเพจ ธรรมดา ในส่วนของภาพมัลติมีเดียเราจะใช้ Flash และนำงานที่ได้มาวางบน Dreamweaver เพื่อให้ทำงานร่วมกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1 หลักการเบื้องต้นในการสร้างเว็บไซต์

ปัจจุบันมีโปรแกรมสำหรับสร้างเว็บเพจจำนวนมาก แต่ Dreamweaver เป็นโปรแกรมที่ใช้งานง่าย และมีคุณสมบัติสำหรับการสร้างเว็บไซต์ระดับเริ่มต้นถึงระดับสูงอย่างครบถ้วน ซึ่ง Dreamweaver เป็นผลิตภัณฑ์ของบริษัท Marcromedia ซึ่งเป็นบริษัทพัฒนาและออกแบบโปรแกรมทางด้านกราฟฟิกรวมไปถึงเครื่องมือในการสร้างเว็บเพจมากมาย

2.1.1 ความสามารถของ Dreamweaver

Dreamweaver เป็นเครื่องมือที่ใช้งานง่าย คล้ายกับโปรแกรมประเภทเดียวกัน การใช้งานเพียงลากองค์ประกอบของหน้าเว็บเพจที่ต้องการ ไปวางบนหน้าเอกสาร ดัดแปลงรูปแบบตามต้องการ ก็สามารถสร้างเว็บเพจเองได้ โปรแกรมจะสร้างโค้ดภาษา HTML ให้เราเองโดยอัตโนมัติ อย่างไรก็ตามบางครั้งหากต้องต้องแก้ไขบางอย่างของเว็บเพจโดยเจาะจง บ่อยครั้งเราต้องใช้ภาษา HTML แต่ Dreamweaver เองก็มีความสามารถดังนี้

1 สนับสนุนการทำงานแบบ WYSIWYG (What You See Is What You Get) หมายความว่าอะไรก็ตามที่เราทำบนหน้าจอใน Dreamweaver ก็จะปรากฏผลแบบเดียวกันบนเว็บเพจ ซึ่งช่วยให้การปรับปรุงแก้ไขเว็บเพจทำได้ง่าย ไม่ต้องมีความรู้ในภาษา HTML

2 มีเครื่องมือในการช่วยสร้างรูปแบบหน้าจอเว็บเพจ และมีความยืดหยุ่นในการใช้งานสูง

3 สนับสนุนภาษาสคริปต์ต่างๆ ทั้งฝั่งไคล์เอ็นต์ และเซิฟเวอร์ เช่น JAVA, ASP, CGI และ VBScript

4 มีเครื่องมือในการอัพโหลด หน้าเว็บเพจไปที่เครื่องเซฟเวอร์ เพื่อทำการเผยแพร่งาน ที่เราสร้างในอินเตอร์เนต โดยการส่งผ่าน FTP หรือโดยการใช้โปรแกรมภายนอกช่วย

5 รองรับมัลติมีเดีย เช่นเสียง กราฟิก และอนิเมชั่นที่สร้างโดยโปรแกรม Flash

2.1.2 ความต้องการทางฮาร์ดแวร์

ถึงแม้ Dreamweaver เป็นโปรแกรมที่ไม่ต้องใช้เครื่อง ที่มีประสิทธิ์ภาพสูงมากนัก แต่ในการใช้งานโดยปรกติ เราอาจจำเป็นต้องใช้โปรแกรมอื่นๆ ร่วมในการทำงานด้วย ดังนั้นขีดความสามารถของเครื่องควรจะมีขั้นต่ำดังนี้

1 หน่วยประมวลผลกลาง รุ่นเพนเทียม 120 MHz ขึ้นไป

2 ระบบปฏิบัติการ Windows 95, Windows 98/ME หรือ Windows NT

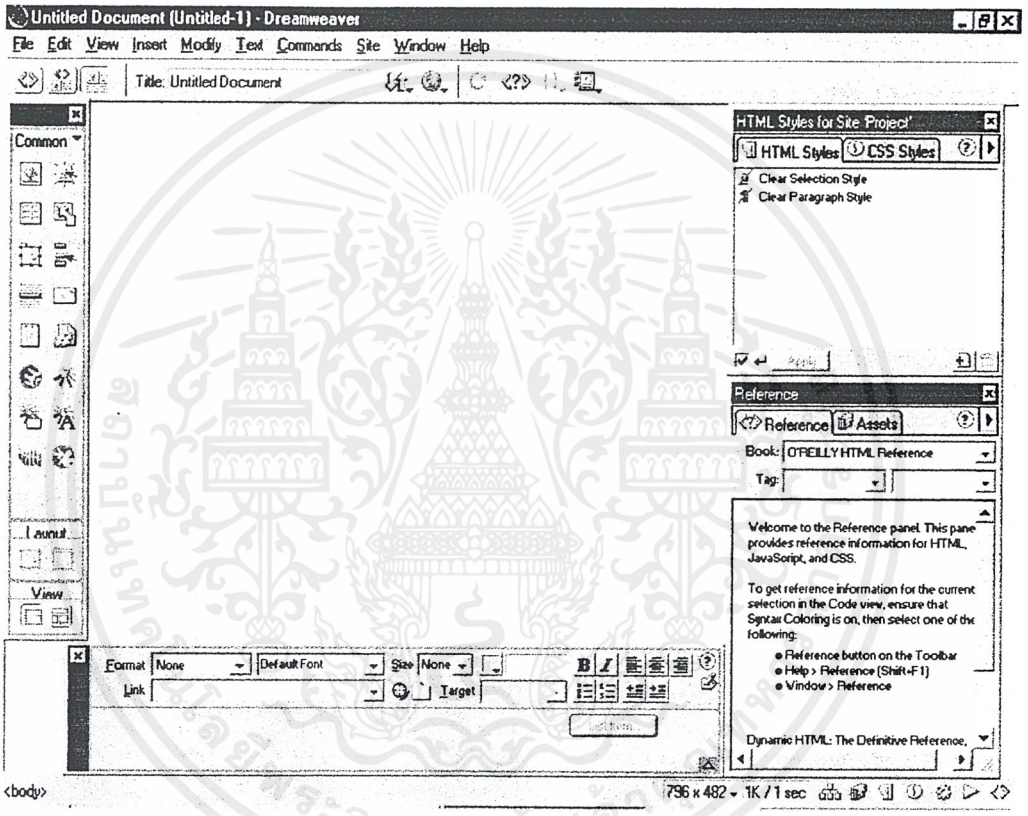
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3 หน่วยความจำ 32 MB และมีเนื้อที่ฮาร์ดดิสก์อย่างน้อย 20 MB สำหรับบรรจุ Dreamweaver

4 ความละเอียดของจอภาพแสดงผล 800x600 pixel

5 ไดร์ฟ CD-ROM

2.1.3 องค์ประกอบต่างๆ ของ Dreamweaver



แถบสถานะ จะแสดงข้อมูลเพิ่มเติมเกี่ยวกับงานที่เรากำลังทำ

หน้าต่าง Objects เป็นกลุ่มเครื่องมือที่ใช้สร้างองค์ประกอบต่างๆ ที่จะนำมาสร้างเป็นหน้าเว็บเพจ โดยมีเครื่องมือต่างๆ อยู่ 6 กลุ่มให้เลือกใช้

ส่วน View และ Layout โดย View จะให้เราเลือกสลับมุมมองการทำงาน ระหว่างมุมมองมาตราฐาน และมุมมอง Layout โดยมุมมอง Layout จะช่วยให้เราสามารถจัดองค์ประกอบต่างๆ บนหน้าเว็บได้ง่ายกว่าเดิม ซึ่งเมื่อเราเลือกใช้มุมมองนี้ ก็จะสามารถใช้กลุ่มคำสั่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในส่วน Layout สร้างองค์ประกอบต่างๆ ได้แก่ Layout Cell และ Layout Table ที่ใช้สำหรับการจัดหน้าได้

แถบ Properties inspectorแต่ละออบเจกต์ที่เรานำมาใช้สร้างหน้าเว็บจะมีคุณสมบัติต่างๆ ที่เราสามารถปรับแต่งได้ เช่นออบเจกต์ที่แสดงรูปภาพ จะปรับแต่งกำหนดค่าต่างๆ ที่เกี่ยวกับรูปภาพ ได้คือ ชื่อภาพ ขนาดของภาพ และขอบภาพเป็นต้น ซึ่งเมื่อใดที่เรา Click mouse เลือกออบเจกต์นี้ แถบ Properties ก็จะแสดงค่าต่างๆ เหล่านั้นให้เราปรับแต่งได้

Launcher จะรวมไอคอนที่ใช้เปิด/ปิด หน้าต่างการทำงานพิเศษอื่นๆ ได้แก่

Site	เป็นหน้าต่างที่แสดงโครงสร้างของเว็บไซต์ และไฟล์ต่างๆ ทั้งหมด
Assets	เป็นหน้าต่างที่เก็บออบเจกต์ที่มีการใช้ซ้ำบ่อยในเว็บไซต์ และองค์ประกอบต่างๆ
HTML Style	เป็นหน้าต่างที่เราสามารถกำหนดรูปแบบตัวอักษร เพื่อนำมาใช้บนหน้าเว็บเพจได้
CSS Style	เราสามารถกำหนดรูปแบบตัวอักษรที่เราใช้งานบ่อยๆ ไว้ใช้กับเว็บเพจแต่ละหน้าได้ที่หน้าต่างนี้
Behaviors	เป็นหน้าต่างที่ให้เรากำหนด Behaviors ต่างๆ ซึ่งเป็นลูกเล่นพิเศษบนหน้าเว็บ เช่นการทำภาพเคลื่อนไหว การซ่อน/แสดงรูปภาพเป็นต้น
History	จะบันทึกการทำงานต่างๆ ให้เราสามารถย้อนกลับได้ในกรณีที่เกิดข้อผิดพลาด
HTML Source	จะแสดงโค้ด HTML ของหน้าเว็บเพจที่เรากำลังทำงานด้วยซึ่งถ้าหากเรามีความรู้ด้าน HTML ก็สามารถแก้ไขโค้ด HTML ได้โดยตรง

2.2 หลักการสร้างภาพเคลื่อนไหวและมัลติมีเดียบนเว็บ

จากการที่เราต้องการหาวิธีในการนำเสนอข้อมูลที่สามารถโต้ตอบกับผู้ใชได้ โดยนำเสนอให้ดึงดูดใจ มีลูกเล่นที่หลากหลาย ทั้งภาพและเสียง แต่การทำเช่นนี้ก็ต้องสร้างปัญหาใหญ่ทีเดียว เพราะการนำเสนอข้อมูลที่ดึงดูดใจผู้ใช้งานโดยใช้ลูกเล่นต่างๆ นั้น นอกจากจำเป็นต้องใช้เทคนิคการเขียนโปรแกรมที่ซับซ้อนแล้ว ยังมีผลทำให้หน้าเว็บที่สร้างมีขนาดใหญ่ด้วย ซึ่งด้วยความจำกัด

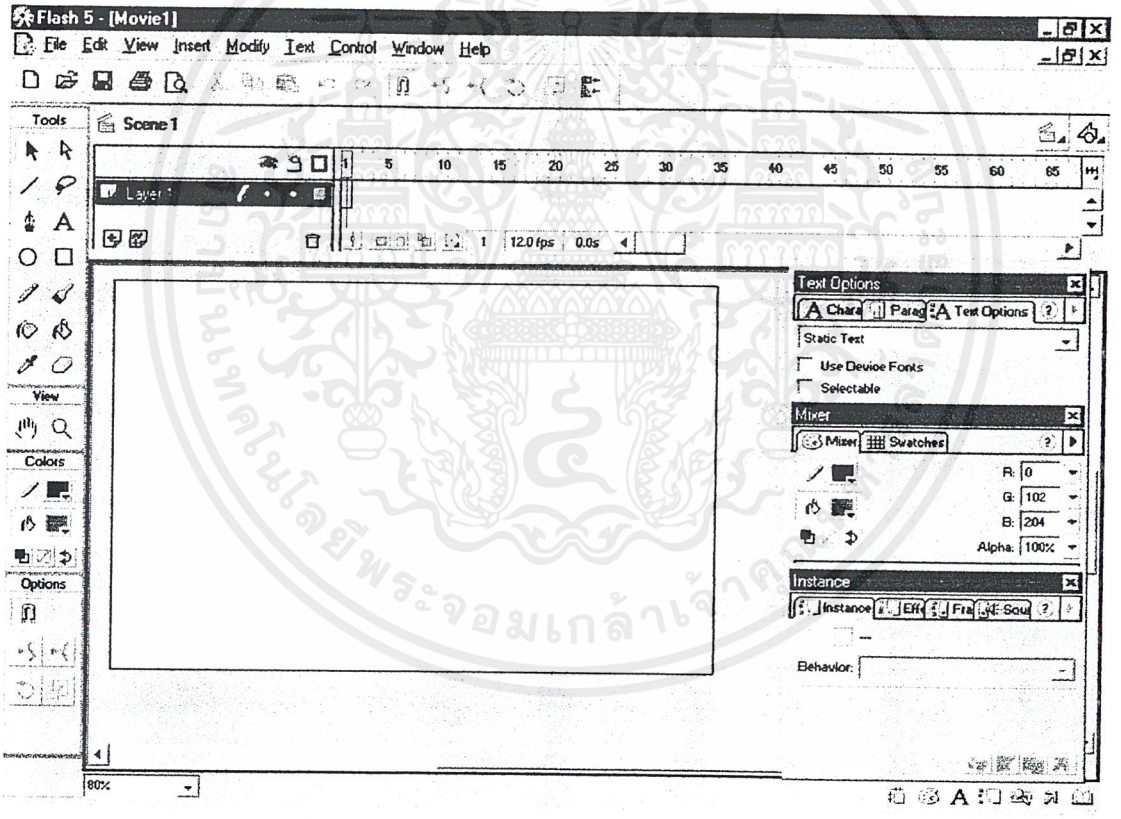
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทางด้านความเร็วของอินเตอร์เน็ตผ่านสายโทรศัพท์ จึงอาจทำให้หน้าเว็บนั้นโหลดช้าจนผู้ใช้ไม่อาจสามารถทนรอได้ สิ่งเหล่านี้สร้างปัญหาอย่างมาก

2.2.1 ความสามารถของ Flash

ความสามารถของโปรแกรม Flash ซึ่งไม่เพียงแต่สามารถสร้างภาพเคลื่อนไหว เสียงประกอบหรือลูกเล่นต่างๆ บนเว็บเท่านั้น แต่เราสามารถใช้ Flash สร้างหน้าเว็บที่สามารถตอบโต้กับผู้ใช้ได้อีกด้วย อีกทั้งชิ้นงานที่ได้ยังมีขนาดเล็กทำให้ไม่สร้างปัญหาเมื่อนำมาเผยแพร่ทางอินเตอร์เน็ต

2.2.2 องค์ประกอบต่างๆ ของ Flash



แถบคำสั่ง (Menu Bar)

เป็นแถบที่ใช้เก็บรวบรวมคำสั่งทั้งหมดของ Flash โดยใช้หัวข้อและแบ่งเป็นหมวดหมู่ต่างๆ ดังนี้

File เป็นเมนูเก็บคำสั่งที่เกี่ยวกับการจัดการกับไฟล์ทั้งหมด เช่น การเปิด ปิด บันทึก การปิดโปรแกรม การนำไฟล์อื่นเข้ามา และการแปลงไฟล์เพื่อนำออกไปใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Edit	เป็นเมนูเก็บคำสั่งที่เกี่ยวกับการแก้ไข เช่น Undo, Redo, Copy, Cut, Paste
View	เป็นเมนูเก็บคำสั่งที่เกี่ยวกับ การแสดงการมองวัตถุใน รูปแบบต่างๆ
Insert	เป็นเมนูเก็บคำสั่งที่เกี่ยวกับการเพิ่มเติมหรือแทรกคำสั่ง เป็นส่วนสำคัญที่ช่วยทำภาพเคลื่อนไหว
Modify	เป็นเมนูเก็บคำสั่งที่เกี่ยวกับการทำให้วัตถุมีคุณสมบัติใหม่ๆ
Text	เป็นเมนูเก็บคำสั่งที่เกี่ยวกับการจัดการกับตัวอักษร เช่น Font Size Style
Control	เป็นเมนูเก็บคำสั่งที่เกี่ยวกับ การควบคุมการแสดงของชิ้นงานที่เราสร้างขึ้น เช่น Play, Rewind, Stop, Forward, backward etc.
Window	เป็นเมนูเก็บคำสั่งที่เกี่ยวกับการแสดงหน้าต่างควบคุมต่างๆ
Help	เป็นเมนูเก็บคำสั่งที่เกี่ยวกับการให้ความช่วยเหลือ ซึ่งอธิบายการใช้งานโปรแกรม

แถบเครื่องมือ (Tool Bar)

New	การเริ่มสร้างงานใหม่
Open	การเปิดไฟล์งานเดิม
Save	การบันทึกไฟล์งาน
Print	การสั่งพิมพ์งาน
Print Preview	การตรวจสอบดูก่อนพิมพ์
Cut	การตัดหรือลบวัตถุออกเพื่อจะนำไปวางที่อื่น
Copy	การคัดลอกวัตถุเพื่อจะนำไปวางที่อื่น
Paste	การวางวัตถุที่ได้ Cut หรือ Copy มาวาง
Undo	การย้อนกลับไป 1 ขั้นตอน
Redo	การเดินหน้าไป 1 ขั้นตอน ใช้เมื่อมีการย้อนหลังไป 1 ขั้นตอน
Snap to Objects	การสั่งให้วัตถุดึงดูดเมื่อเข้าใกล้กัน
Smooth	การทำให้วัตถุมีความโค้งมน
Straighten	การทำให้วัตถุดูแข็ง
Rotate	การหมุนวัตถุ
Scale	การขยายและย่อวัตถุ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Align การจัดวัตถุให้เป็นระเบียบ

กล่องเครื่องมือ (Toolbox)

เป็นกล่องสำหรับเก็บอุปกรณ์ต่างๆ ที่ใช้ในการทำงาน Toolbox จะมีลักษณะเป็นไอคอนรูปภาพ สามารถเรียกใช้งานได้ง่ายเพียงการ Click mouse ที่ไอคอนคำสั่งที่ต้องการเท่านั้น เราสามารถเรียกใช้กล่องเครื่องมือนี้ได้โดยคำสั่ง Window>Tools การ Toolbox แบ่งเป็นกลุ่มๆ เช่น

กลุ่มคำสั่ง Section เกี่ยวกับการเลือก
กลุ่มคำสั่ง Edit เกี่ยวการวาดและการตกแต่งภาพ
กลุ่มคำสั่ง View เกี่ยวกับการมองภาพในรูปแบบต่างๆ
กลุ่มคำสั่ง Colors ใช้ในการเลือกสีให้กับเส้นและสีพื้น
กลุ่มคำสั่ง Option จะแสดงคำสั่งเพิ่มเติมของแต่ละคำสั่งที่เราเลือกใช้งาน

เส้นเวลา (Timeline)

Timeline เป็นเครื่องมือหนึ่งของ Flash จะปรากฏทุกครั้งเมื่อเปิดโปรแกรม ใช้สำหรับสร้างและปรับเปลี่ยนรายละเอียดของการเคลื่อนไหว โดยเอาตำแหน่งขององค์ประกอบที่เคลื่อนไหว มาจัดวางต่อกันทีละภาพในแต่ละช่วงเวลา เพื่อสร้างเป็นภาพเคลื่อนไหว โดยเราจะกำหนดเส้นเวลาให้สามารถเล่นภาพเคลื่อนไหวซ้ำ หรือเล่นแล้วหยุดก็ได้

หน้าต่างการทำงาน (Stage)

เป็นหน้าต่างใช้สำหรับ การทำงาน หรือเรียกว่าเวที ซึ่งเป็นพื้นที่ว่างๆ สำหรับสร้างภาพเคลื่อนไหว โดยให้เราเป็นผู้กำหนด เมื่อเราสร้างงานหรือวางวัตถุบน Stage เราจะเรียกภาพรวมนี้เรียกว่า Scene

หน้าต่างไลบรารี่ (Library)

ในการทำงาน Flash แต่ละงานเราอาจนำไฟล์ภาพ ภาพเคลื่อนไหว หรือเสียง มาใช้ประกอบกันในงานของเรา ซึ่งหน้าต่าง Library จะเป็นตัวรวบรวมไฟล์ต่างๆ เหล่านี้ไว้ รวมทั้งไฟล์ที่เราสร้างเองด้วยหากเราต้องการใช้งานไฟล์เหล่านี้อีกก็แค่หยิบจาก Library มาวางบน Stage เท่านั้น เราสามารถเปิดหน้าต่างไลบรารี่มาใช้งานโดยเลือกคำสั่ง Windows>Libraly

ปุ่มควบคุมการแสดงผล (Controller)

Controller จะใช้สำหรับควบคุมการแสดงผลงาน Animation ที่เราสร้าง เราสามารถ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เปิดหน้าต่าง Controller มาใช้ได้โดยเลือกคำสั่ง Window>Toolbars>Controller

Stop	ให้หยุดการแสดงผล
Rewind	ให้กลับไปเริ่มต้นแสดงผลใหม่
Step Back	ให้ถอยหลังไป 1 Step
Play	ให้แสดงผล
Step Forward	ให้เดินหน้าไป 1 Step
Control End	ให้ไปยังจุดจบของการแสดงผล

หน้าต่างควบคุมการแสดงผล (Panel)

ใน Flash จะมีหน้าต่างที่รวบรวมข้อมูลต่างๆ ที่ใช้งานกับวัตถุ โดยที่เราไม่ต้องเสียเวลาไปเปิดหา เพราะมีการจัดเป็นหมวดหมู่ไว้แล้ว ประกอบด้วย 5 Panel สำคัญได้แก่ Panel Info, Panel Mixer, Panel Character, Panel Instance และ Panel Movie Explorer

2.2.3 หลักการทำงานของ Flash

ก่อนที่เราจะใช้ Flash กัน เราต้องรู้ว่าการสร้างชิ้นงานนั้นมีหลักการอย่างไร และเมื่อเราเข้าใจหลักคร่าวๆ ของ Flash แล้วการใช้ Flash ก็เป็นเรื่องที่ง่ายขึ้นมาก ประกอบด้วย 4 ขั้นตอนดังนี้

ขั้นตอนที่ 1 เป็นการนำภาพที่เราสร้างขึ้น หรือที่เราได้มาจากที่อื่นมาวางไว้บน Stage

ขั้นตอนที่ 2 ทำการแปลงวัตถุนั้นเป็น Symbol ซึ่งเป็นลักษณะที่ Flash สามารถนำไปทำเป็น Animation ได้

ขั้นตอนที่ 3 เป็นขั้นตอนที่เราจะมาทำ Symbol ให้เป็น Animation ซึ่งในขั้นตอนนี้เราสามารถจะใส่เสียง หรือ Script คำสั่งเพิ่มเติมลงไปได้ด้วย

ขั้นตอนที่ 4 เมื่อเราสร้างชิ้นงานเสร็จแล้ว เราก็ทำการ Publish หรือการแปลงชิ้นงานของเราออกสู่ชิ้นงานของเราออกสู่สายตาประชาชน

นามสกุลของ Flash หลักๆ 2 อย่างคือ

*.fla คือไฟล์ที่ได้จากการสร้างชิ้นงานใน Flash ที่เกิดจากการเซฟเป็นไฟล์ต้นฉบับ ซึ่งสามารถนำมาเปลี่ยนแก้ไขได้ *.fla ย่อมาจาก Flash Movie หรือที่เราเรียกว่า Movie

*.swf คือไฟล์ที่ Flash สร้างขึ้นหลังจากการแสดงผลออกมา เราสามารถ Double Click ดูผลการแสดงได้โดยไม่ต้องเปิดโปรแกรม Flash ซึ่ง *.swf ย่อมาจาก Shockwave Flash

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

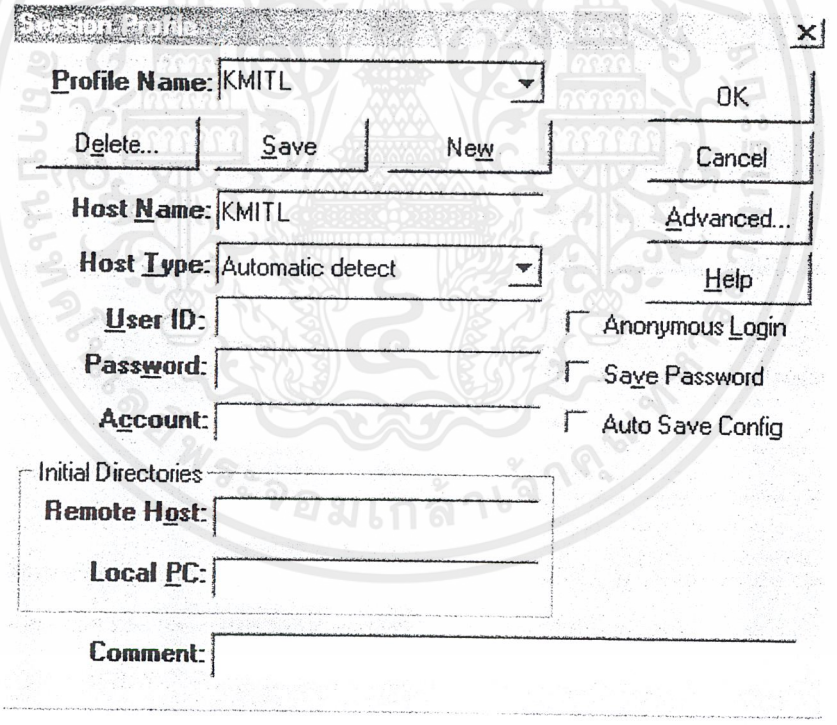
นอกจากนั้นเมื่อเราสั่งให้ Flash ทำการ Publish หรือเผยแพร่งานออกมาเป็นหน้าเว็บ เราจะได้ไฟล์ที่มีนามสกุลเป็น 2 ชนิด คือ *.html กับ *.swf ซึ่ง *.html จะเป็นตัวเรียกไฟล์ *.swf ขึ้นมาเพื่อแสดงภาพเคลื่อนไหวบนหน้าเว็บ

2.3 การอัพโหลดไฟล์สู่เว็บไซต์

การนำงานขึ้นสู่อินเตอร์เน็ตหรือที่เรียกว่าการอัพโหลด (Upload) คือการนำข้อมูลของเราไปไว้บนเครื่อง Sever สิ่งที่ต้องจัดเตรียมก่อนการนำงานขึ้นสู่อินเตอร์เน็ต

1 พื้นที่อินเตอร์เน็ต ซึ่งอาจจะเป็นของ ISP ที่คุณใช้บริการอินเตอร์เน็ตอยู่ หรืออาจจะขอพื้นที่ฟรี ซึ่งคุณจะได้ ชื่อผู้ใช้ (Usernme) และพาสเวิร์ด (Password)

2 โปรแกรมนำเว็บเพจขึ้นสู่อินเตอร์เน็ต ซึ่งเราจะอธิบายการใช้งาน FTP เพราะเป็นโปรแกรมที่เล็ก และใช้งานง่าย และสามารถ Download ได้ฟรี



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
ไมโครคอนโทรลเลอร์ตระกูล MCS-51

3.1 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

โครงสร้างภายใน

หน่วยความจำสำหรับเก็บโปรแกรม สูงสุด 64 กิโลไบต์ และหน่วยความจำสำหรับเก็บความจำภายใน 4 กิโลไบต์

หน่วยความจำสำหรับเก็บข้อมูล

- หน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป มีขนาดสูงสุด 64 กิโลไบต์ และใช้ PORT 0 เป็นแอสเดรสและดาต้าบัส และใช้ PORT 2 เพียง 3 เส้น เพื่อติดต่อกับหน่วยความจำที่ต้องการ

-หน่วยความจำสำหรับเก็บข้อมูลภายในชิป แบ่งเป็น 3 หน่วยย่อย

-หน่วยความจำสำหรับเก็บข้อมูลทั่วไป บริเวณ 128 ไบต์ล่าง

-หน่วยความจำสำหรับเก็บข้อมูลทั่วไป บริเวณ 128 ไบต์บน

-หน่วยความจำสำหรับเก็บข้อมูลของรีจีสเตอร์เฉพาะ (SFR)

รีจีสเตอร์ใช้งานเฉพาะ การออกแบบสำหรับควบคุมระบบโดยเฉพาะ เช่น TCON, TMOD F หรือรีจีสเตอร์ IE หรือ IP เป็นต้น

รีจีสเตอร์ใช้งานทั่วไป เช่น รีจีสเตอร์ A,B หรือ R0-R7 อยู่ใน 128 ไบต์แรก มีด้วยกัน 4 กลุ่ม จึงมีด้วยกัน 32ตัว

โครงพอร์ต มีพอร์ต 4 บิต จำนวน 4 พอร์ต เพราะฉะนั้นจึงใช้เป็นพอร์ต 1 บิต ได้ 32 พอร์ต

- P0 ใช้งานใช้งานเป็นดาต้าบัสและแอสเดรสบัสไบต์ต่ำ
- P1 ใช้งานอินพุต เอาท์พุต พอร์ตทั่วไปได้
- P2 ใช้งานอินพุต เอาท์พุต พอร์ตทั่วไปได้ และเป็นแอสเดรสบัสไบต์สูง
- P3 ใช้งานสำหรับการเขียนและการอ่านข้อมูลจากภายนอก

ไทม์เมอร์เคาน์เตอร์ มีขนาด 16 บิต ใช้สำหรับนับจำนวนสัญญานาฬิกาหรือแมชชินไซเคิลของวงจรออสซิสเลเตอร์ภายใน หรือจำนวนครั้งของการเปลี่ยนสัญญาณจากภายนอก

การเข้าถึงข้อมูลมีหลายวิธี

การเข้าถึงข้อมูลโดยตรง

การเข้าถึงข้อมูลโดยทางอ้อม

การเข้าถึงข้อมูลในรีจีสเตอร์ใช้งานทั่วไป

การเข้าถึงข้อมูลในรีจีสเตอร์ใช้งานเฉพาะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วิธีการเข้าถึงข้อมูลโดยทางตรง

วิธีการเข้าถึงข้อมูลโดยตังอ้างอิง

กลุ่มคำสั่งของ MCS-51

กลุ่มคำสั่งทางคณิตศาสตร์

กลุ่มคำสั่งทางตรรกศาสตร์

กลุ่มคำสั่งเคลื่อนย้ายข้อมูล

กลุ่มคำสั่งในการควบคุมลำดับการทำงานของโปรแกรม

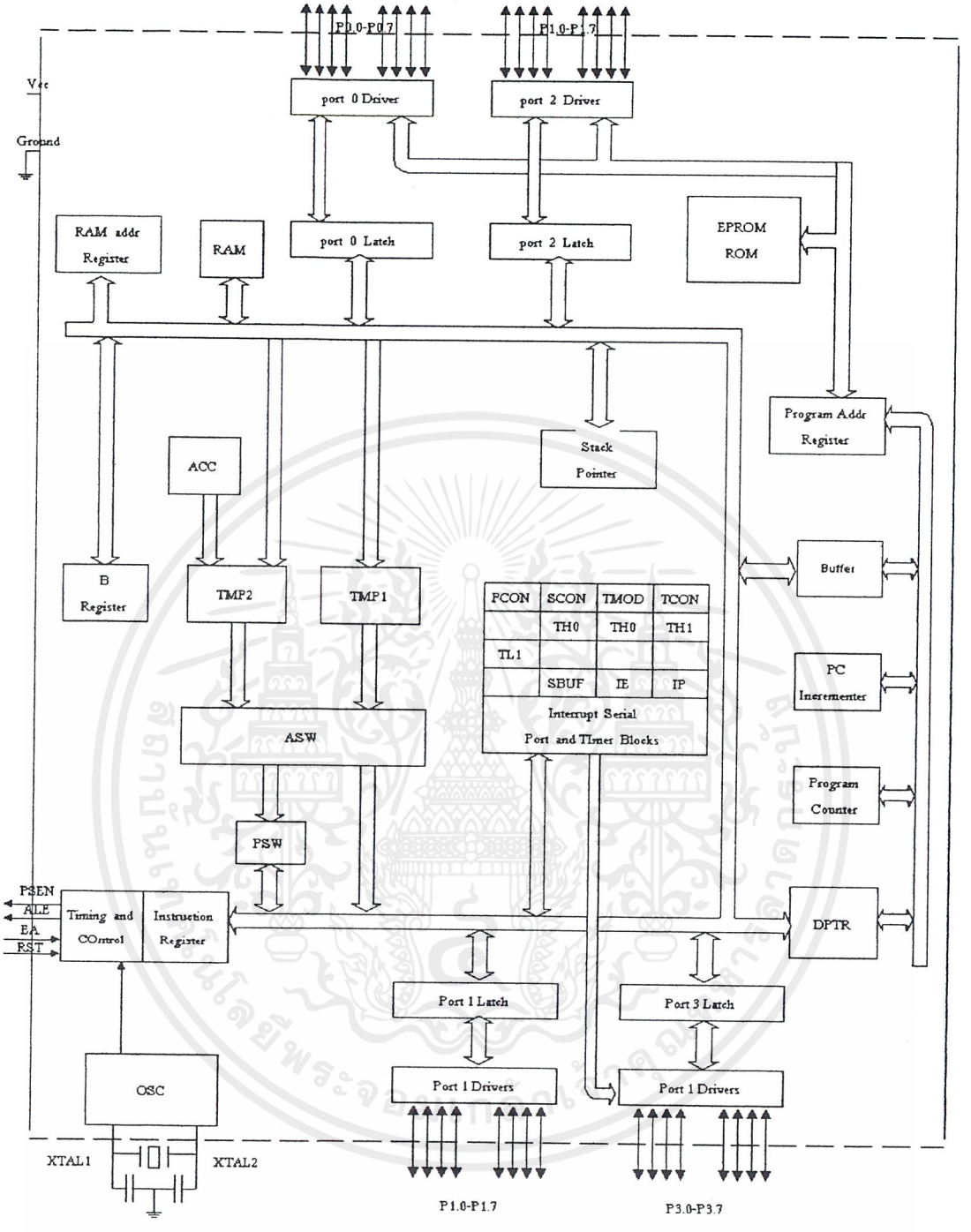
กลุ่มคำสั่งสำหรับการประมวลผลแบบบูลีน

3.2 โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51

ในหัวข้อนี้จะกล่าวถึงรายละเอียดคร่าวๆ ของไมโครคอนโทรลเลอร์ตระกูล MCS-51 โดยมีจุดประสงค์เพื่อให้เข้าใจและมองเห็นภาพกว้างๆ ของไมโครคอนโทรลเลอร์ตระกูลนี้ เพื่อเป็นพื้นฐานในการศึกษารายละเอียดต่อไปหน่วยความจำสำหรับเก็บโปรแกรมจะ เก็บโปรแกรมควบคุมการทำงานของชิป MCS-51 มีสมาชิกในตระกูลหลายเบอร์ด้วยกัน แต่ละเบอร์จะมีคุณสมบัติพิเศษบางอย่างแตกต่างกัน เช่น มีหน่วยความจำสำหรับเก็บข้อมูลภายในเพิ่มขึ้น มีวงจรเปลี่ยนค่าสัญญาณอะแนล็อกเป็นดิจิตอลในตัว สามารถรับสัญญาณอินเตอร์รัพต์ได้หลายชนิด ทำกระบวน-การ DMA (Direct Memory Access) ได้ในตัว มีรีจิสเตอร์ใช้สำหรับเป็นไทม์เมอร์หรือเคาท์เตอร์เพิ่มขึ้น

ไมโครคอนโทรลเลอร์ที่นับได้ว่าเป็นพื้นฐานสำหรับตระกูล MCS-51 นี้ได้แก่เบอร์ 8051,8031,8751 โดยเบอร์ 8051 จัดเป็นสมาชิกตัวแรกในตระกูล ซึ่งมีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิปเป็น ROM ขนาด 4 ไบต์ มีพอร์ตขนาด 8 บิต 4 พอร์ต และหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายใน MCS-51 (RAM) เองจำนวน 128 ไบต์ มีพอร์ตขนาด 8 บิต 4 พอร์ต มีรีจีสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ขนาด 16 บิตรวม 2 ตัว รับสัญญาณอินเตอร์รัพจากภายนอกได้ 2 ชนิด สามารถรับและส่งข้อมูลแบบอนุกรมผ่านทางพอร์ตสื่อสารข้อมูลแบบอนุกรม มีวงจรเพื่อสร้างสัญญาณนาฬิกาเพื่อควบคุมการทำงานในตัวเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.1 ภาพแสดงโครงสร้างภายในของชิปไมโครคอนโทรลเลอร์ตระกูล MCS-51

3.3 การแบ่งหน่วยความจำ

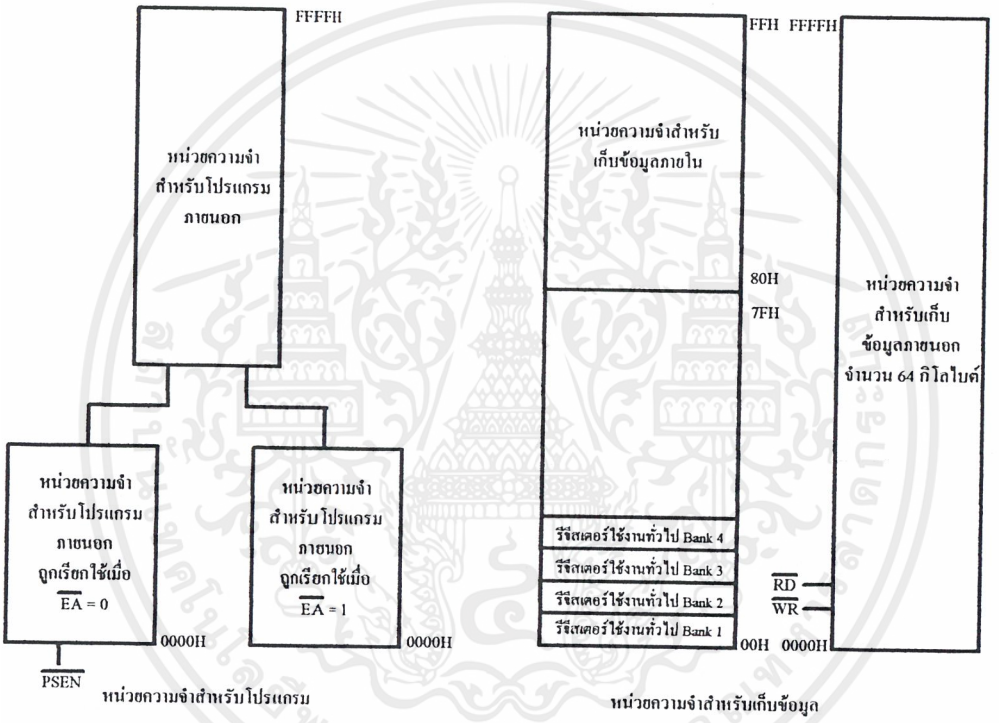
ในไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบ่งชนิดหรือหน้าที่ของหน่วยความจำออกเป็น 2 ส่วนคือ

หน่วยความจำโปรแกรม (Program memory) และหน่วยความจำข้อมูล (Data memory)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หน่วยความจำโปรแกรมจะใช้สำหรับ กระทำการโปรแกรมควบคุมการทำงาน ของไมโครคอนโทรเลอร์ ซึ่งบางเบอร์จะมีหน่วยความจำในส่วนนี้อยู่ภายในตัว โดยอาจจะมีขนาดไม่เท่ากัน

หน่วยความจำข้อมูลจะใช้สำหรับเก็บข้อมูลค่าตัวแปรต่างๆ จากการทำงานของโปรแกรม ซึ่งใน MCS-51 ทุกตัวจะมีหน่วยความจำส่วนนี้อยู่จำนวนหนึ่ง แต่อาจมีขนาดมากหรือน้อยต่างกันไปในแต่ละเบอร์ สำหรับการจัดโครงสร้างของหน่วยความจำทั้งในส่วนของหน่วยความจำและหน่วยความจำข้อมูลได้แสดงไว้ดังรูป



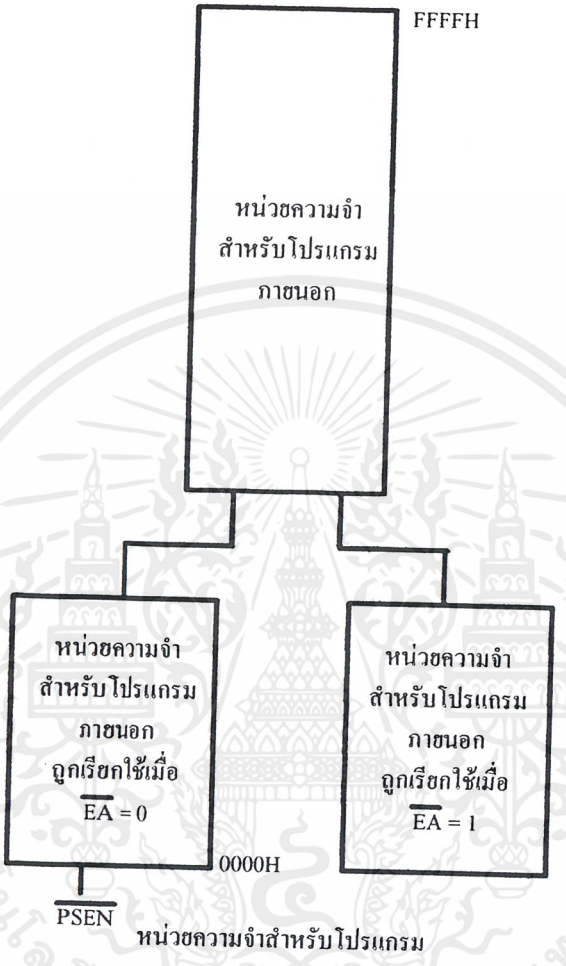
ภาพที่ 3.2 แสดงการจัดโครงสร้าง ของหน่วยความจำทั้งในหน่วยความจำโปรแกรม และหน่วยความจำข้อมูล

3.4 หน่วยความจำโปรแกรม

เป็นบริเวณหน่วยความจำ สำหรับเก็บข้อมูลและคำสั่งใช้งานต่างๆ หน่วยความจำโปรแกรมสามารถแบ่งออกได้เป็น 2 ส่วนคือ หน่วยความจำโปรแกรมภายในและหน่วยความจำภายนอก หน่วยความจำโปรแกรมภายในนี้จะถูกเลือกใช้งานถ้าขาสัญญาณของ EA มีค่าเป็น 1 โดยถูกใช้งานในช่วงแอดเดรส 0 – 0FFFH (หรือช่วง 0 – 0FFFF ในเบอร์ 8052) นอกเหนือจากช่วงแอดแดรสไปจะใช้หน่วยความจำโปรแกรมภายนอกทั้งหมด ในกรณีตรงกันข้ามถ้าสัญญาณ EA มีค่าเป็น 0 ในช่วงแอดแดรส 0 – 0FFFH (หรือช่วงแอดเดรส 0000H – 1FFFH ในเบอร์ 8052)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะถูกใช้จากหน่วยความจำภายนอกหรือกล่าวได้ว่า ถ้าขาสัญญาณ EA มีค่าเป็น 0 จะเป็นการเลือกหน่วยความจำโปรแกรมภายนอกทั้งหมดตลอดช่วงแอดเดรส



ภาพที่ 3.3 การจัดพื้นที่หน่วยความจำโปรแกรมสำหรับไมโครคอนโทรลเลอร์ 8051

3.5 หน่วยความจำสำหรับเก็บข้อมูล

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 สามารถทำการอ่านและเขียนข้อมูลจากหน่วยความจำข้อมูลที่มีขนาดสูงสุดได้ 64 กิโลไบต์ และหน่วยความจำในที่นี้ทำหน้าที่เก็บข้อมูลใช้งานจำนวนมากเป็นส่วนใหญ่ หน่วยความจำข้อมูลกำหนดให้มีตำแหน่งใช้งานตั้งแต่ 00000H ถึง 08000H ซึ่งตามที่กล่าวมาแล้วหน่วยความจำสำหรับเก็บโปรแกรมภายนอกถูกกำหนดให้เริ่มที่ตำแหน่ง 04000H เป็นต้นไป นั่นคือโปรแกรมใช้งานจะต้องเริ่มประมวลผลที่ตำแหน่ง 04000H ขึ้นไปเสมอ หน่วยความจำข้อมูลภายในและหน่วยความจำข้อมูลภายนอก สำหรับหน่วยความจำข้อมูลภายในข้อมูลยังแบ่งออกได้เป็น 2 ส่วนย่อยคือ ส่วนที่เก็บข้อมูลทั่วไปและส่วนที่ใช้เป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

รีจีสเตอร์หน้าที่พิเศษหรือ SFR (Special Function Register)โดยส่วนที่ใช้เก็บข้อมูลทั่วไปจะถูกใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่างๆ จากการทำงานของโปรแกรม ส่วนรีจีสเตอร์หน้าที่พิเศษจะถูกใช้งานเป็นรีจีสเตอร์ควบคุมการทำงานและบอกสถานะการทำงานของไมโครคอนโทรลเลอร์



ภาพที่ 3.4 การจัดพื้นที่หน่วยความจำข้อมูลสำหรับไมโครลคอนโทรลเลอร์ 8051

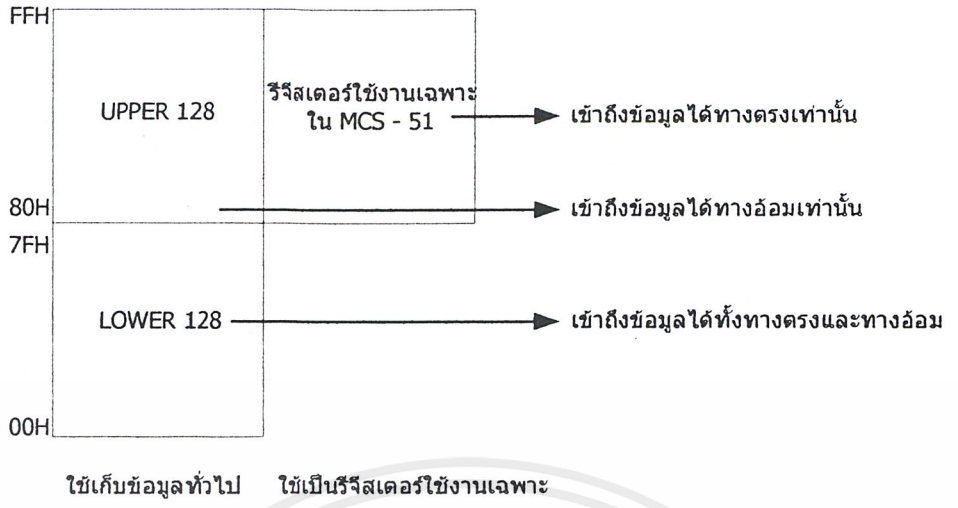
3.5.1 หน่วยความจำข้อมูลภายใน

หน่วยความจำข้อมูลในส่วนนี้จะถูกแบ่งออกเป็น 3 ส่วนย่อยดังนี้

- หน่วยความจำสำหรับเก็บข้อมูลทั่วไป บริเวณ 128 ไบต์ล่าง (lower 128)
- หน่วยความจำสำหรับเก็บข้อมูลทั่วไป บริเวณ 128 ไบต์บน (upper 128)
- หน่วยความจำสำหรับเก็บข้อมูลของรีจีสเตอร์เฉพาะ (SFR)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

45865



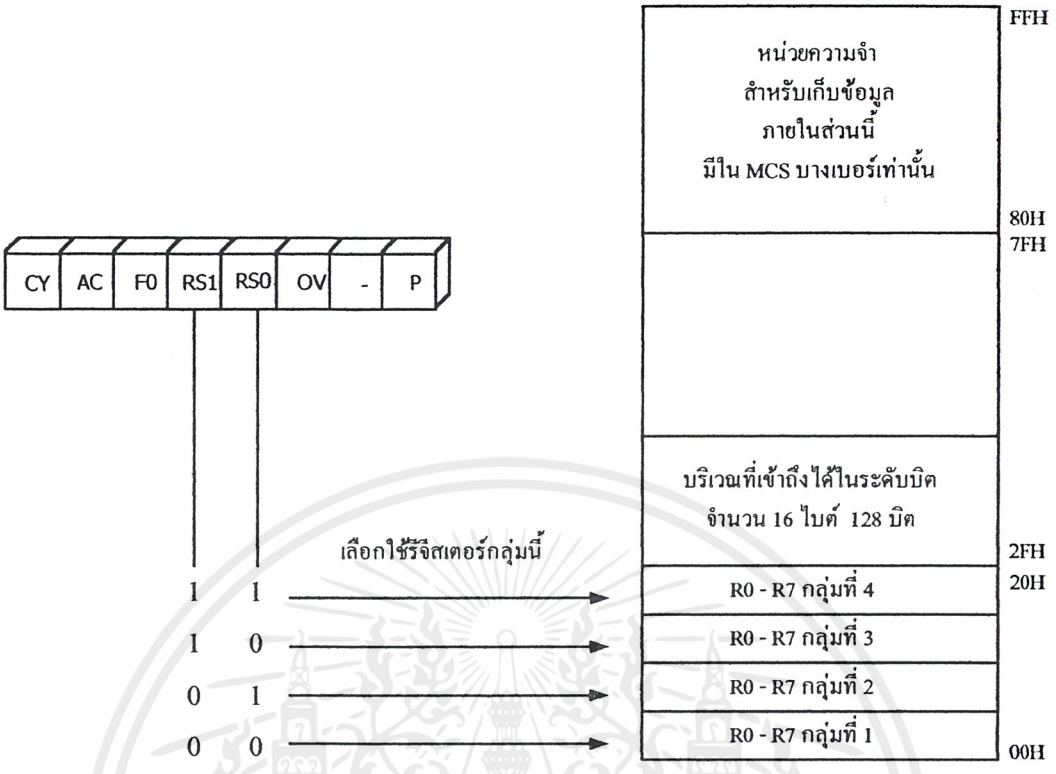
ภาพที่ 3.5 แสดงหน่วยความจำสำหรับเก็บข้อมูลทั้ง 3 ส่วน

บริเวณ 128 ไบต์บน (ตำแหน่ง 7FH-0FFH) จะใช้วิธีเข้าถึงข้อมูลโดยทางอ้อมเท่านั้น แต่สำหรับหน่วยความจำเก็บข้อมูลที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ (ตำแหน่ง 7FH-0FFHเช่นเดียวกัน) จะใช้วิธีเข้าถึงข้อมูลได้โดยทางตรงเท่านั้น ดังนั้นหน่วยความจำสำหรับเก็บข้อมูลทั้งสองบริเวณนี้จึงสามารถมีตำแหน่งซ้ำกันได้ โดยในระหว่างการทำงาน MCS-51 จะตรวจสอบจากรหัสคำสั่งเองว่าคำสั่งที่ต้องการทำงานมีการเข้าถึงข้อมูลในหน่วยความจำตำแหน่งใด และโดยวิธีไหน

หน่วยความจำสำหรับเก็บข้อมูลภายใน บริเวณ 128 ไบต์ล่าง ตำแหน่ง 00H-1FH รวม 32 ไบต์ จะกำหนดให้เป็นกลุ่มของรีจิสเตอร์ใช้งานทั่วไป 4 กลุ่ม ๆละ 8 ตัว (R0-R7) กลุ่มรีจิสเตอร์ใช้งานทั่วไปจะถูกเลือกใช้งานเพียงกลุ่มเดียวในขณะใดขณะหนึ่ง เมื่อมีคำสั่งระบุให้ใช้ข้อมูลในรีจิสเตอร์ใช้งานทั่วไป R0-R7 ซีพียูใน MCS-51 จะตรวจสอบเองว่าในขณะนั้นรีจิสเตอร์ใช้งานทั่วไปกลุ่มใดถูกเลือกใช้งาน โดยดูจากบิต RS0,RS1 ในรีจิสเตอร์ใช้งานเฉพาะ PSW

หน่วยความจำสำหรับเก็บข้อมูลภายใน ที่อยู่ถัดจากกลุ่มรีจิสเตอร์ใช้งานทั่วไปทั้ง 4 กลุ่ม ตั้งแต่ตำแหน่ง 20H-2FH รวม 16 ไบต์ ได้ถูกออกแบบให้มีลักษณะโครงสร้างพิเศษกว่าหน่วยความจำสำหรับเก็บข้อมูลบริเวณอื่น โดยแต่ละบิตของหน่วยความจำบริเวณนี้มีหมายเลขตำแหน่งกำหนดไว้แน่นอน เพื่อนำมาใช้เป็นข้อมูลขนาด 1 บิตในการทำงานของกลุ่มคำสั่งประมวลผลแบบบูลีนได้ แต่ละบิตของหน่วยความจำบริเวณนี้มีหมายเลขตำแหน่งตั้งแต่ 00H-7FH รวม 128 ตำแหน่ง หรือ 128 บิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.6 แสดงการเลือกใช้กลุ่มรีจิสเตอร์ใช้งานทั่วไป โดยควบคุมจากบิต RS0,RS1

3.5.2 หน่วยความจำข้อมูลภายนอก

มีขนาดสูงสุด 64 กิโลไบต์ การใช้หน่วยความจำสำหรับเก็บข้อมูลภายนอก พอร์ต 0 จะถูกใช้เป็นแอดเดรสบัสและดาต้า ส่วนพอร์ต 2 จะถูกใช้เพียง 3 เส้นเพื่อเป็นตัวเลือกช่วงหน่วยความจำที่ต้องการติดต่อเนื่องจากแอดเดรสที่ต้องการสำหรับหน่วยความจำขนาด 2 กิโลไบต์ คือ 11 เส้น

(8 เส้นจากพอร์ต 0 รวมกับ 3 เส้นจากพอร์ต 2)

ดังนั้นพอร์ต 2 ที่เหลือจากการต่อหน่วยความจำสำหรับเก็บข้อมูลสามารถนำไปใช้เป็นพอร์ตอินพุตหรือพอร์ตเอาท์พุตทั่วไปได้

หน่วยความจำสำหรับเก็บข้อมูลที่อยู่ภายนอกจำเป็นต้องมีสัญญาณควบคุมการอ่าน และเขียนข้อมูล สัญญาณควบคุมการอ่านและการเขียนข้อมูลจะถูกส่งจาก MCS-51 ผ่านทางขา P3.7 และ P3.6 ตามลำดับ

3.6 พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยทางตรงและทางอ้อม (Direct and Indirect Address Area)

พื้นที่ 128 ไบต์ล่างสุดจะเป็น 3 ส่วนดังภาพที่ 3.9

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1 รีจีสเตอร์แบงค์ (Register Bank 0-3)

ตั้งแต่ตำแหน่ง (00H-1FH) จะเป็นสัดส่วนของรีจีสเตอร์แบงค์ (0-3) โดยแบ่งเป็นแบงค์ละ 8 ไบต์ (แต่ละแบงค์จะมีรีจีสเตอร์ R0,R1,R2,R3,R4,R5,R6,R7) ถ้าซีพียูทำงานอยู่ที่แบงค์ 3 เมื่อถูกรีเซ็ทจะกลับมาทำงานที่แบงค์ 0 เสมอ และ SP จะมาเริ่มต้นที่ตำแหน่ง 07H ทันที

2 บริเวณหน่วยความจำที่ใช้ตำแหน่งอ่านเขียนทีละบิตได้ (Bit Address Area)

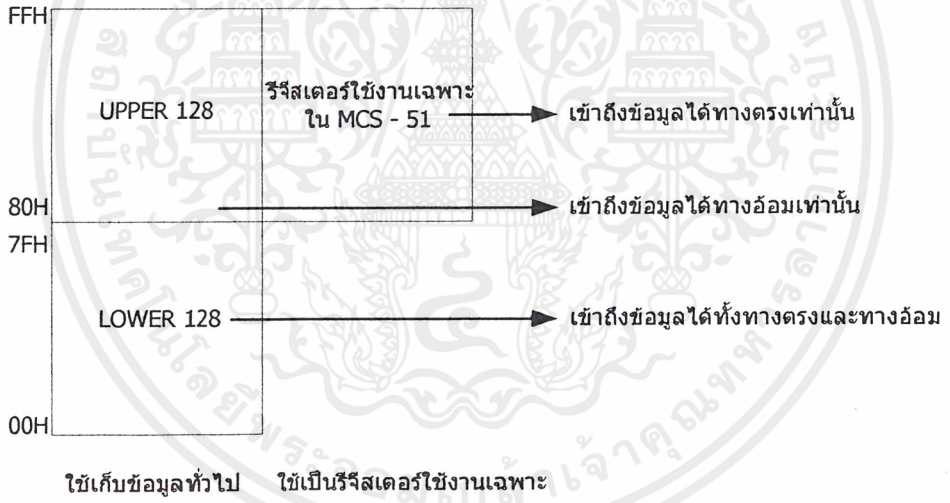
พื้นที่ตั้งแต่แอดเดรส (20H-7FH) จำนวน 16 ไบต์หรือแบ่งเป็นบิตจะได้เท่ากับ 128 บิต ซึ่งตำแหน่งบิตมีดังนี้ 00,01,2,03,04,05,06,07 จนถึง 7FH

เช่น บิต 00 ก็คือ D0 ของหน่วยความจำตำแหน่ง 20H

บิต 01 ก็คือ D1 ของหน่วยความจำตำแหน่ง 20H

3 บริเวณหน่วยความจำที่ใช้งานทั่วไป (Scrath Pad Area)

พื้นที่ตั้งแต่ (30H-7FH) จะเขียนข้อมูลได้ทีละไบต์เท่านั้นไม่สามารถใช้คำสั่งเกี่ยวกับบิตได้ถ้าย้ายเนื้อที่สแตคมายังบริเวณนี้โปรดระวังในการเขียนข้อมูลทับสแตค



ภาพที่ 3.7 ไบต์ของ RAM ที่เข้าถึงข้อมูลแบบทางตรงและทางอ้อม

3.7 การเข้าถึงข้อมูลในคำสั่ง

มีด้วยกันหลายวิธีดังนี้

- การเข้าถึงข้อมูลโดยตรง (Direct Addressing)
- การเข้าถึงข้อมูลโดยทางอ้อม (Indirect Addressing)
- การเข้าถึงข้อมูลในรีจีสเตอร์ใช้งานทั่วไป (Register Instructions)
- การเข้าถึงข้อมูลในรีจีสเตอร์เฉพาะของคำสั่ง (Register –Specific Instructions)
- วิธีการเข้าถึงข้อมูลที่กำหนดเองโดยตรง (Immediate Constants)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- วิธีการเข้าถึงข้อมูลที่มีตัวอ้างอิง (Indexed Addressing)

คำสั่งแต่ละคำสั่งที่ต้องการโอเปอร์แรนด์อาจมีวิธีการเข้าถึงข้อมูล ได้วิธีเดียวหรือหลายวิธี ขึ้นอยู่กับชนิดของคำสั่งแต่ละคำสั่ง นอกจากนี้หน่วยความจำแต่ละบริเวณ ก็สามารถมีวิธีการเข้าถึงข้อมูลได้แตกต่างกัน

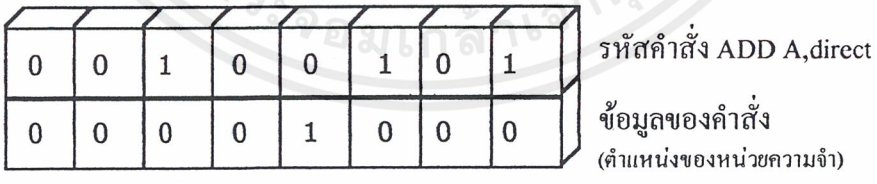
วิธีการเข้าถึงข้อมูลแต่ละวิธีมีรายละเอียดดังนี้

3.7.1 วิธีการเข้าถึงข้อมูลโดยทางตรง

วิธีนี้จะระบุค่าหน่วยความจำที่เก็บข้อมูล โดยตรงในคำสั่งข้อมูลที่นำมาประมวลผลโดยวิธีนี้จะเป็นค่าของข้อมูลในหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เก็บข้อมูลทั่วไป เฉพาะในบริเวณ 128 ไบต์ล่างและหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เป็นรีจีสเตอร์ใช้งานเฉพาะเท่านั้น และเนื่องจากหน่วยความจำสำหรับเก็บข้อมูล ที่ใช้เก็บข้อมูลทั่วไปในบริเวณ 128 ไบต์ล่าง กับหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เป็นรีจีสเตอร์ใช้งานเฉพาะมีขนาดรวมกันทั้งสิ้น 256 ไบต์ ดังนั้นค่าตำแหน่งหน่วยความจำที่ใช้ต้องเป็นเลขไบนารี่ขนาด 8 บิตเท่านั้น

คำสั่งที่ใช้วิธีการเข้าถึงข้อมูลโดยตรงแบบนี้จะมีความยาวอย่างน้อยที่สุด 2 ไบต์ โดยไบต์แรกจะเป็นไบต์ที่บอกรหัสของคำสั่ง (opcode) ส่วนไบต์ที่สองซึ่งตามหลังไบต์ที่เป็นคำสั่งจะเป็นไบต์ที่ระบุค่าตำแหน่งหน่วยความจำที่มีข้อมูลที่ต้องการนำมาประมวลผล ดังตัวอย่างต่อไปนี้

ADD A, 08H นำค่าในหน่วยความจำตำแหน่ง 08H มาบวกกับค่าในรีจีสเตอร์ A ผลลัพธ์ที่ได้เก็บไว้ในรีจีสเตอร์ A รหัสของคำสั่งมีดังนี้



3.7.2 วิธีการเข้าถึงข้อมูลโดยทางอ้อม

ค่าตำแหน่งหน่วยความจำที่ต้องการจะติดต่อจะเก็บ ไว้ในรีจีสเตอร์เฉพาะคำสั่ง ดังนั้นวิธีนี้เป็นการเข้าถึงข้อมูลโดยทางอ้อม นั่นคือ แทนที่ผู้เขียนโปรแกรมจะระบุค่าตำแหน่งข้อมูลโดยตรง วิธีนี้จะใช้ค่าที่เก็บไว้ในรีจีสเตอร์ที่ระบุในรหัสคำสั่ง ชี้ไปยังตำแหน่งของหน่วยความจำแทน หน่วยความจำที่สามารถใช้วิธีการเข้าถึงข้อมูลด้วยวิธีการเข้าถึงข้อมูลด้วยวิธีนี้จะเป็นหน่วย-ความจำสำหรับเก็บข้อมูลที่ใช้เก็บข้อมูลทั่วไปบริเวณ 128 ไบต์ล่าง และ 128 ไบต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บน รวมทั้งหน่วยความจำสำหรับเก็บข้อมูลที่อยู่ภายนอกชิป รีจีสเตอร์ที่สามารถนำมาเป็นตัวชี้ (pointer) ตำแหน่งของหน่วยความจำมีดังนี้

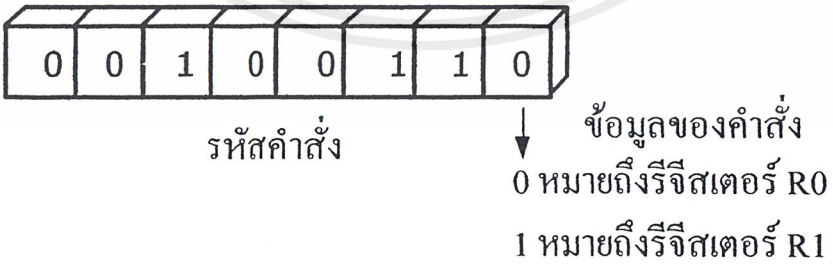
- รีจีสเตอร์ใช้งานทั่วไป R0, R1 ของแต่ละกลุ่มรีจีสเตอร์ทั้ง 4
- รีจีสเตอร์ใช้งานเฉพาะ SP (stack pointer)
- รีจีสเตอร์ใช้งานเฉพาะ DPTR (data pointer)

รีจีสเตอร์แต่ละตัวจะมีหน้าที่ไม่เหมือนกัน ซึ่งสามารถอธิบายหน้าที่การทำงานของรีจีสเตอร์ต่างๆ ข้างต้นดังนี้

รีจีสเตอร์ใช้งานทั่วไป R0, R1 เป็นรีจีสเตอร์ขนาด 8 บิต รีจีสเตอร์ทั้งสองสามารถชี้ตำแหน่งในหน่วยความจำได้เพียง 256 ตำแหน่งเท่านั้น โดยบริเวณหน่วยความจำของ 8051 ที่สามารถใช้รีจีสเตอร์ R0,R1 ระบุตำแหน่งได้นี้จะเป็นหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เก็บข้อมูลทั่วไปบริเวณ 128 ไบต์ล่าง ส่วนหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เป็นรีจีสเตอร์ใช้งานเฉพาะไม่สามารถใช้รีจีสเตอร์ทั้งสองตำแหน่งนี้ได้ และต้องมีเครื่องหมาย @ นำหน้ารีจีสเตอร์ R0 หรือ R1 ด้วย

นอกจากจะสามารถใช้รีจีสเตอร์ใช้งานทั่วไป R0,R1 ชี้ตำแหน่งหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เก็บข้อมูลทั่วไปที่อยู่ภายในชิปของ 8051 ได้แล้ว รีจีสเตอร์ทั้งสองยังสามารถชี้ตำแหน่งหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิปได้อีกด้วย แต่เนื่องจากรีจีสเตอร์ทั้งสองมีหน่วยความจำขนาด 8 บิต จึงมีข้อจำกัดที่ไม่สามารถชี้ตำแหน่งหน่วยความจำสำหรับเก็บข้อมูล ภายนอกได้เกิน 256 ไบต์ ตัวอย่างคำสั่งที่รีจีสเตอร์ที่ใช้งานทั่วไปเป็นตัวชี้ตำแหน่งข้อมูลดังนี้

ADD A, @ R0 นำค่าในหน่วยความจำตำแหน่งที่ตรงกับค่ารีจีสเตอร์ R0 มาบวกกับค่าในรีจีสเตอร์ A ผลที่ได้เก็บไว้ในรีจีสเตอร์ A



รีจีสเตอร์ใช้งานเฉพาะ SP มีขนาด 8 บิต เหมือนรีจีสเตอร์ใช้งานทั่วไป R0,R1 แต่รีจีสเตอร์ใช้งานเฉพาะ SP ไม่สามารถชี้ตำแหน่งหน่วยความจำสำหรับเก็บข้อมูลที่อยู่ภายนอกชิปได้

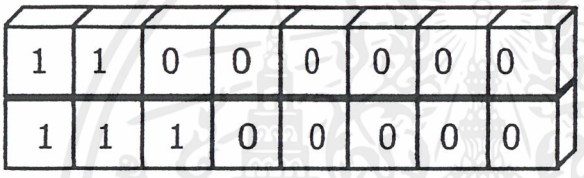
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รีจีสเตอร์ SP มีคำสั่งใช้เฉพาะคือ PUSH, POP, CALL ซึ่ง 8051 จะทำการเพิ่มหรือลดค่าในรีจีสเตอร์ SP ให้เอง ในสภาวะที่ 8051 ถูกรีเซตหรือจ่ายพลังงาน ค่าในรีจีสเตอร์ SP จะมีค่าเท่ากับ 07H เสมอ

บริเวณหน่วยความจำที่รีจีสเตอร์ SP ชี้อยู่มีชื่อเรียกว่าสแตก ซึ่งเป็นบริเวณของหน่วยความจำที่เก็บไว้ใช้ปฏิบัติคำสั่งซึ่งใช้รีจีสเตอร์ SP เป็นตัวชี้ตำแหน่ง

ตัวรหัสคำสั่งของคำสั่งที่ใช้รีจีสเตอร์ใช้งานเฉพาะ SP เป็นตัวชี้ข้อมูล ไม่จำเป็นต้องระบุตำแหน่งของรีจีสเตอร์ SP เพราะ 8051 ทราบเองจากรหัสคำสั่งดังตัวอย่าง

PUSH ACC นำค่าในรีจีสเตอร์ A ไปเก็บไว้ในสแตก สมมติในขณะรีจีสเตอร์ใช้งานเฉพาะ SP มีค่า 15H



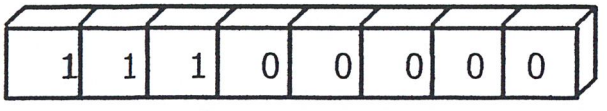
รหัสคำสั่ง PUSH ACC

ข้อมูลของคำสั่ง

ตำแหน่งหน่วยความจำที่ต้องการนำข้อมูลไปเก็บ(รีจีสเตอร์)

รีจีสเตอร์ใช้งานเฉพาะ DPTR เป็นรีจีสเตอร์ขนาด 16 บิต ดังนั้นรีจีสเตอร์ตัวนี้สามารถชี้ตำแหน่งหน่วยความจำสำหรับเก็บข้อมูลที่ภายนอกชิปได้มากถึง 64 กิโลไบต์ รีจีสเตอร์ตัวนี้ที่มีใช้ในคำสั่งที่ต้องการติดต่อกับหน่วยความจำสำหรับเก็บข้อมูลหรือโปรแกรมที่อยู่ภายนอกชิป 8051 ดังตัวอย่างต่อไปนี้

MOVX A,@DPTR นำค่าในหน่วยความจำตำแหน่งที่เท่ากับค่าในรีจีสเตอร์ DPTR ในขณะนั้นมาไว้ในรีจีสเตอร์



รหัสคำสั่ง MOVX A,@DPTR

การเข้าถึงข้อมูลโดยทางอ้อมโดยรีจีสเตอร์ทั้ง 4 ตัวนี้ โอเปอร์แรนด์ซึ่งเป็นรีจีสเตอร์ที่เก็บค่าตำแหน่งของข้อมูลที่ต้องการประมวลผลจะต้องระบุเครื่องหมาย "@" ไว้ข้างหน้า ดังตัวอย่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MOV A,@R0 นำค่าในหน่วยความจำที่ชี้โดยรีจีสเตอร์ R0 ไปไว้ในรีจีสเตอร์ A MOVX @DPTR,A นำค่าในรีจีสเตอร์ A ไปไว้ในหน่วยความจำที่ชี้โดยรีจีสเตอร์ DPTR การใช้รีจีสเตอร์ DPTR เป็นตัวชี้ตำแหน่งหน่วยความจำจะต้องโหลดค่าตำแหน่งที่ต้องการไปไว้ในรีจีสเตอร์ DPTR ก่อน

การใช้รีจีสเตอร์ SP เป็นตัวชี้ข้อมูลในหน่วยความจำมีวิธีการที่แตกต่างออกไปผู้เขียนโปรแกรมไม่จำเป็นต้องระบุค่าของรีจีสเตอร์ SP เพราะรีจีสเตอร์ตัวนี้จะถูกเพิ่มหรือลดค่าเอง โดยอัตโนมัติ ดังตัวอย่าง

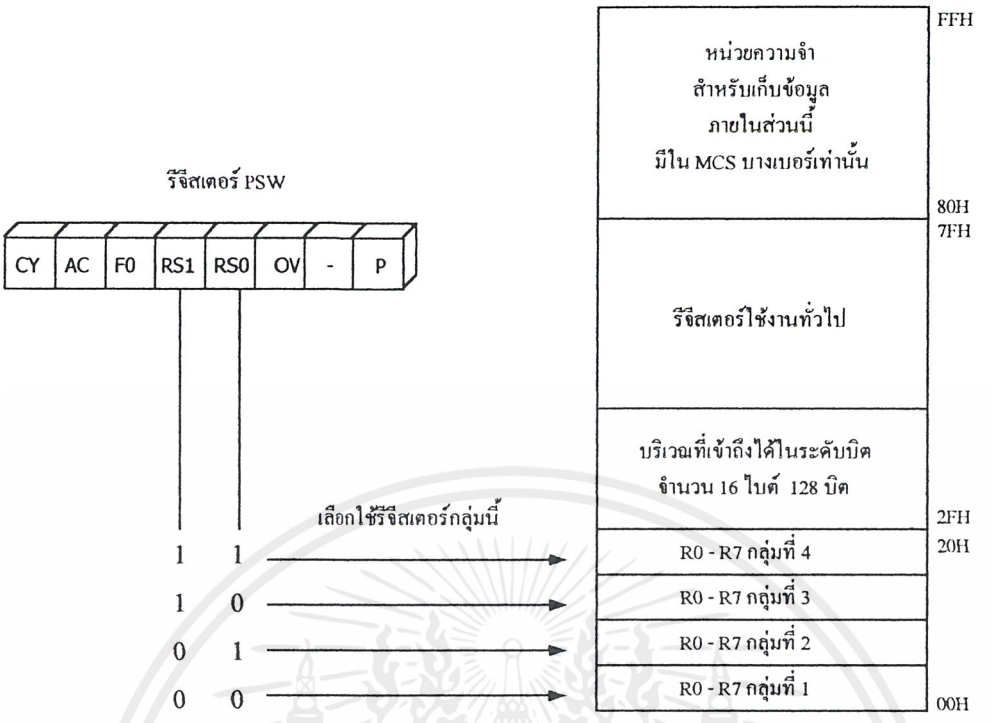
PUSH ACC จะเพิ่มค่าในรีจีสเตอร์ SP จากนั้นจะนำค่าในรีจีสเตอร์ A ไปไว้ในหน่วยความจำที่ชี้โดยรีจีสเตอร์ SP

POP ACC นำค่าในหน่วยความจำที่ชี้โดยรีจีสเตอร์ SP ไปไว้ในรีจีสเตอร์ A จากนั้นรีจีสเตอร์ SP จะลดค่าลงเอง

3.7.3 วิธีการเข้าถึงข้อมูลในรีจีสเตอร์ใช้งานทั่วไป

วิธีนี้เป็นการเข้าถึงข้อมูลที่อยู่ในรีจีสเตอร์ R0-R7 ของรีจีสเตอร์ใช้งานทั่วไปในแต่ละกลุ่ม รีจีสเตอร์ใช้งานทั่วไปในแต่ละกลุ่มทั้ง 4 กลุ่มคือบริเวณหนึ่งของหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เก็บข้อมูลบริเวณ 128 ไบต์ล่าง 00-1FH หรือตำแหน่งที่ 32 ไบต์ล่างสุดของหน่วยความจำสำหรับข้อมูลที่ใช้เก็บข้อมูลทั่วไป ดังนั้นหากผู้เขียนโปรแกรมต้องการอ่านหรือเขียนข้อ-มูลในรีจีสเตอร์ทั้ง 32 ตัวซึ่งแต่ละตัวจะมีตำแหน่งในหน่วยความจำที่แน่นอนก็สามารถชี้ตำแหน่งของหน่วยความจำที่ตรงกับรีจีสเตอร์ ในแต่ละตัวด้วยวิธีการเข้าถึงข้อมูลโดยตรง หรือทางอ้อม ดังที่ได้กล่าวมาแล้ว เหตุที่ต้องมีวิธีที่ต้องเข้าถึงข้อมูลในรีจีสเตอร์ใช้งานทั่วไปนี้ ก็เพื่อลดขนาดของคำสั่งให้สั้นลง เพราะหากมีการเข้าถึงข้อมูลของรีจีสเตอร์ R0-R7 จะทราบเองว่าจะต้องเลือกรีจีสเตอร์ใช้งานกลุ่มใดใน 4 กลุ่ม โดยดูจากค่าในบิต RS0,RS1 ดังแสดงในภาพ 3.10

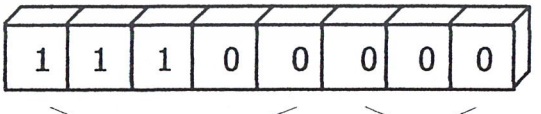
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.8 แสดงการเลือกใช้รีจีสเตอร์ใช้งานทั่วไปแต่ละกลุ่มจากบิต RS0,RS1

ดังนั้นคำสั่งที่มีการเข้าถึงข้อมูลใช้รีจีสเตอร์ใช้งานทั่วไปต้องการใช้เลขไบนารี่เพียง 3 บิต เป็นตัวบอกรีจีสเตอร์ที่ต้องการใช้เท่านั้น ดังนั้นคำสั่งที่มีการใช้รีจีสเตอร์ R0-R7 โดยวิธีนี้จะมีความยาวของคำสั่งต่ำสุด 1 ไบต์ (รหัสคำสั่ง 5 บิต รวมกับอีก 3 บิตที่ต้องระบุรีจีสเตอร์) ดังแสดงในตัวอย่างต่อไปนี้

ADD A,R1 นำค่าในรีจีสเตอร์ทั่วไป R1 มาบวกกับค่าในรีจีสเตอร์ A ผลลัทธ์เก็บไว้ในรีจีสเตอร์ A



รหัสคำสั่ง ADD A,Rn

ข้อมูลของคำสั่ง

ระบุรีจีสเตอร์ที่ใช้ในคำสั่งดังนี้

0 0 0	R0	1 0 0	R4
0 0 1	R1	1 0 1	R5
0 1 0	R2	1 1 0	R6
0 1 1	R3	1 1 1	R7

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.7.4 วิธีการเข้าถึงข้อมูลในรีจีสเตอร์เฉพาะของคำสั่ง

คำสั่งบางคำสั่งของ 8051 จะระบุไว้แล้วว่าต้องปฏิบัติการกับข้อมูลในรีจีสเตอร์ตัวใด เช่น รีจีสเตอร์ A รีจีสเตอร์ DPTR รีจีสเตอร์ SP ในรหัสคำสั่งของคำสั่งที่ใช้วิธีการเข้าถึงข้อมูลประเภทนี้ 8051 จะทราบเองว่าจะต้องทำงานกับรีจีสเตอร์ตัวใดโดยไม่จำเป็นต้องระบุตำแหน่งของรีจีสเตอร์ที่ใช้โดยคำสั่งเลย เช่น

INC DPTR มีขนาด 1 ไบต์ เพราะไม่ต้องระบุตำแหน่งรีจีสเตอร์ DPTR

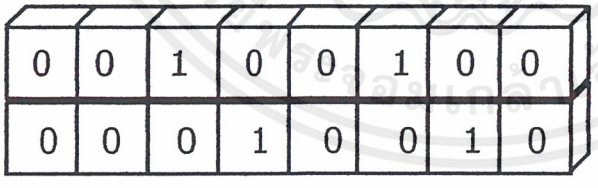
MUL AB มีขนาด 1 ไบต์ เพราะไม่ต้องระบุตำแหน่งรีจีสเตอร์ A,B

จากคำสั่งทั้งสองจะเห็นได้ว่าโอเปอร์แรนด์ตัวแรกของคำสั่งเป็นรีจีสเตอร์ DPTR, A,B ตามลำดับ แต่ผู้ใช้ไม่จำเป็นต้องระบุตำแหน่งรีจีสเตอร์ทั้งสอง เพราะ 8051 จะทราบเองจากรหัสคำสั่ง

3.7.5 วิธีการเข้าถึงข้อมูลที่กำหนดเองโดยตรง

เป็นการกำหนดค่าข้อมูลที่จะนำไปประมวลผลโดยตรง ข้อมูลที่นำมาประมวลผลในคำสั่งจะอยู่ตามหลังรหัสคำสั่งโดยการใช้เครื่องหมาย "#" ระบุหน้าข้อมูลที่ต้องการ เช่น

ADD A,#12H นำค่า 12H มาบวกกับค่าในรีจีสเตอร์ A ผลลัทธ์เก็บไว้ในรีจีสเตอร์ A



รหัสคำสั่ง ADD A,#data

ข้อมูลของคำสั่ง (12H)

3.7.6วิธีการเข้าถึงข้อมูลโดยใช้ตัวอ้างอิง

ข้อมูลที่ใช้การอ้างอิงแบบนี้จะเป็น ข้อมูลที่อยู่ ในหน่วยความจำสำหรับเก็บโปรแกรมเท่านั้น นั่นแสดงว่าเราสามารถอ่านข้อมูลนี้ออกมาได้แต่ไม่สามารถนำข้อมูลนี้ไปเก็บโดยวิธีนี้ได้ จุดประสงค์ของการอ้างข้อมูลแบบนี้ มีไว้เพื่อใช้ในการเปิดหาค่าข้อมูลในหน่วยความจำสำหรับเก็บโปรแกรม ซึ่งเป็นหน่วยความจำชนิดถาวร (ROM) ข้อมูลส่วนนี้จะไม่สูญหายแม้ไม่มีพลังงาน ในการทำงานของคำสั่งที่ใช้การเข้าถึงข้อมูลวิธีนี้จะใช้ค่าของรีจีสเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใช้งานเฉพาะ DPTR หรือ PC มารวมกับค่าในรีจีสเตอร์ A เพื่อชี้ไปยังตำแหน่งหน่วยความจำสำหรับเก็บโปรแกรมที่เก็บข้อมูลไว้ ดั้งนั้นรีจีสเตอร์ใช้งานเฉพาะ DPTR,PC จะต้องมีค่าเท่ากับตำแหน่งต้นของหน่วยความจำสำหรับเก็บโปรแกรมในส่วนที่เก็บข้อมูล โดยใช้ค่าของรีจีสเตอร์ A เป็นตัวระบุว่าข้อมูลอยู่ห่างจากตำแหน่งเริ่มต้นในรีจีสเตอร์ใช้งานเฉพาะ DPTR หรือ PC เท่าใด จุด-ประสงค์ของการอ้างข้อมูลแบบนี้คือ ใช้ในการเปิดค่าในข้อมูลในตารางซึ่งอยู่ในหน่วยความจำสำหรับเก็บโปรแกรมซึ่งเรียงต่อกันไป ตัวอย่างการใช้งานมีดังนี้

```
MOVC A,@A+DPTR
MOVC A,@A+PC
```

การเข้าถึงข้อมูลวิธีนี้ยังมีที่ใช้ในกลุ่มคำสั่งควบคุมลำดับการทำงานของโปรแกรมโดยการบังคับให้โปรแกรมกระโดดไปทำงานที่ตำแหน่งใดๆ ในความจำซึ่งมีค่าตำแหน่งเท่ากับผลรวมของค่าใช้ในรีจีสเตอร์ DPTR และ A ดังตัวอย่าง

```
JMP @A,+ DPTR
```

3.8 ไทม์เมอร์ / เคาท์เตอร์

8051 จะมีตัวจับเวลา/ ตัวนับ (Timer/ Counter) ขนาด 16 บิต จำนวน 2 ตัว คือ ไทม์เมอร์, เคาท์เตอร์ 0 และ ไทม์เมอร์/เคาท์เตอร์ 1

โดยแต่ละตัวสามารถกำหนดให้เป็นตัวจับเวลาหรือตัวนับได้โดยการเซต หรือ เคลียร์บิต C/T ที่ตัวรีจีสเตอร์ควบคุม TMOD ซึ่งอยู่ในกลุ่มรีจีสเตอร์พิเศษ

ในการกำหนดให้ทำงานเป็นตัวจับเวลา ตัวรีจีสเตอร์ TH1 และ TH2 ซึ่งทำหน้าที่เป็นตัวเก็บค่าจำนวนพัลส์ที่เข้ามาจะเพิ่มค่าทุกๆ แมชีนไซเกิล โดยแต่ละแมชีนไซเกิลจะประกอบด้วย 12 คาบออสซิลเลเตอร์ ดังนั้นอัตราการนับแต่ละครั้งจะใช้เวลาเท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์ ซึ่งส่วนใหญ่จะใช้งานเป็นอินเตอร์รัพต์ (Real Time Clock)

และถ้าให้ทำงานเป็นตัวนับ รีจีสเตอร์ตัวนับจะเพิ่มค่าทุกครั้งที่มีการเปลี่ยนแปลงสถานะจาก “1” เป็น “0” ที่ขา T0 หรือ T1 โดยอัตราความถี่สูงสุดที่สามารถรับได้ต้องไม่เกิน 1/24 ของความถี่ออสซิลเลเตอร์

ตัวจับ/นับ สามารถโปรแกรมให้มันทำงานได้ต่างกันถึง 4 โหมด โดยการตั้งค่าในรีจีสเตอร์ TMOD ซึ่งสามารถกล่าวถึงการทำงานในแต่ละโหมดได้ดังนี้

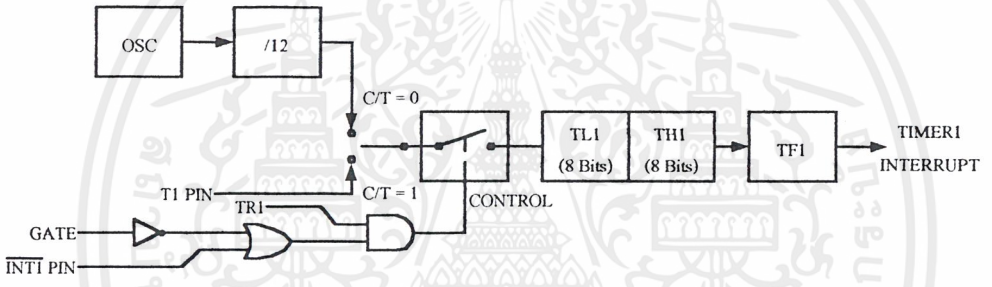
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โหมด 0 รีจีสเตอร์ตัวนับจะถูกกำหนดให้เป็นตัวจับเวลาหรือตัวนับได้ โดยเซตหรือเคลียร์ที่บิต C/T ในตัวรีจีสเตอร์ TMOD การทำงานของรีจีสเตอร์นับขึ้นครั้งละ 1 เมื่อสัญญาณเข้ามา 1 ลูก และเมื่อนับจนเป็น "1" หมดทุกบิต ก็จะกลับมาเป็น "0" หมดทุกบิตใหม่ ซึ่งจะเป็นการเกิดโอเวอร์โฟล์ (Overflow) ไปทดแฟลกอินเตอร์รัพต์ ให้เป็น "1"

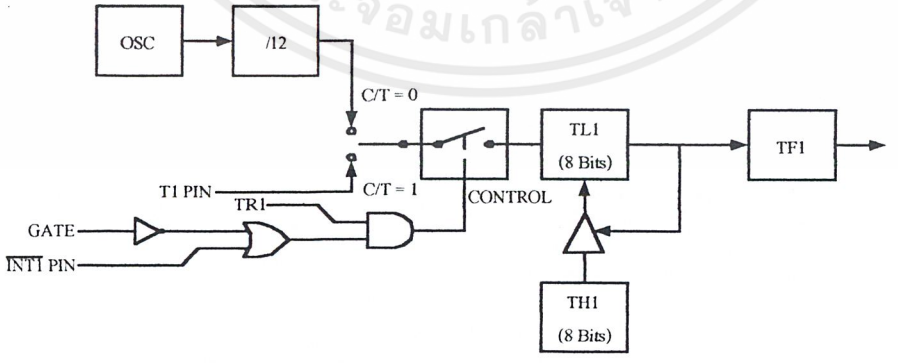
โหมด1 การทำงานเหมือนกับโหมด 0 ทุกอย่างยกเว้นรีจีสเตอร์ตัวนับจะเป็นขนาด 16 บิต

โหมด2 จะใช้รีจีสเตอร์ TL1 เป็นตัวนับเพียงตัวเดียวและเมื่อTL1นับจนเป็น "1"หมดทุกบิตก็จะมีการ ไหลค่าจาก รีจีสเตอร์ TH1 เข้าไปยัง TL1 โดยอัตโนมัติและทำการทดลองแฟลกอินเตอร์รัพต์ TF1 ให้เป็น "1" ค่าใน TH1 สามารถตั้งได้ในซอร์ฟแวร์

โหมด3 เป็นการเพิ่มตัวจับเวลาขึ้นอีก 1 ตัวแต่จะเป็นขนาด 8 บิต ทั้งคู่ ซึ่งลักษณะการทำงานอื่นๆ จะเหมือนกับโหมด 0

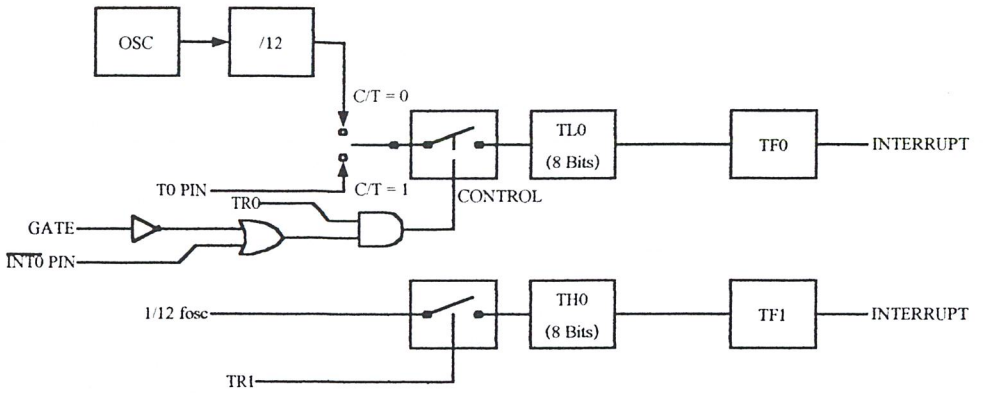


ภาพที่ 3.9 แสดงวงจรการทำงานของตัวจับเวลา/ตัวนับ ที่ 1 ในโหมด 0 และโหมด 1



ภาพที่ 3.10 แสดงวงจรการทำงานของตัวจับเวลา / ตัวนับที่ 1 ในโหมด2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.11 แสดงวงจรการทำงานของตัวจับเวลา /ตัวนับที่0 ในโหมด 3

สำหรับรายละเอียดบิตควบคุมในรีจีสเตอร์ TMOD และรีจีสเตอร์ TCON แสดงไว้ในภาพ

TIMER/COUNTER 1				TIMER/COUNTER 0			
GATE	$\overline{C/T}$	M1	M0	GATE	$\overline{C/T}$	M1	M0

ภาพที่ 3.12 บิตควบคุมที่อยู่ในรีจีสเตอร์ TMOD

GATE เซตเป็น "1" จะเป็นการอีนาเบิลตัวจับเวลา/ตัวนับให้ถูกควบคุมโดย INTx ต้องมีสถานะสูงและบิต TRx ใน TCON จะต้องเป็น "1" จึงจะเริ่มทำงาน แต่ถ้า G ATE เป็น "0" ตัวจับเวลาตัวนับเวลาจะถูกควบคุมให้เริ่มทำงานด้วยบิต TRx เท่านั้น

C/T เป็นบิตควบคุมการเลือกทำงานเป็นตัวจับเวลาหรือตัวนับ ถ้าเป็น "0" จะทำงานเป็นตัวจับเวลา ถ้าเป็น "1"ทำงานเป็นตัวนับ

M1,M0 เป็นตัวเลือกโหมดการทำงานดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.1 แสดงการเลือกโหมดการทำงานของ TIMER/COUNTER โดยบิต M1, M2

M1	M0	โหมดการทำงาน
0	0	โหมด 0
0	0	โหมด 1
0	0	โหมด 2
0	0	โหมด 3

TFx เป็นแฟลกอินเตอร์จะถูกเซตด้วยฮาร์ดแวร์เมื่อเกิดโอเวอร์โฟลว์ของตัวจับเวลา/ตัวนับ และจะต้องเคลียร์ตัวเองโดยอัตโนมัติ เมื่อทำงานอินเตอร์รัพต์นั้นเสร็จเรียบร้อยแล้ว

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

ภาพที่ 3.13 แสดงบิตควบคุมที่อยู่ในรีจีสเตอร์ TCON

TRx เป็นบิตควบคุมให้ตัวจับเวลาเริ่มทำงาน โดยการเซตให้เป็น "0" โดยซอร์ฟแวร์

IEx เป็นแฟลกอินเตอร์รัพต์จากสัญญาณภายนอก เซตด้วยฮาร์ดแวร์เมื่อมีสัญญาณอินเตอร์รัพต์ปรากฏที่ขา INTx และจะเคลียร์ตัวเองโดยอัตโนมัติเมื่อกระโดดทำงานบริการอินเตอร์รัพต์ขอเข้ามาเรียบร้อยแล้ว

Itx เป็นบิตควบคุมรูปแบบสัญญาณอินเตอร์รัพต์ภายนอกจะเซต/เคลียร์ ด้วยซอร์ฟแวร์โดยถ้าเซตเป็น "1"จะถูกอินเตอร์รัพด้วยสัญญาณขอบขาลง และถ้าเคลียร์เป็น "0" จะถูกอินเตอร์รัพด้วยสัญญาณระดับแรงดันต่ำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.9 โครงสร้างการอินเตอร์รัพต์

8051 รับสัญญาณอินเตอร์จากอุปกรณ์อื่นๆ ได้5 แหล่งเราสามารถกำหนด ลำดับความสำคัญของอินเตอร์รัพต์ได้ 2 ระดับ โดยไม่ต้องอาศัยวงจรภายนอกช่วย แต่ละแหล่งของอินเตอร์รัพต์นั้น จะกำหนดเป็นเวกเตอร์เฉพาะ (ตัวชี้แอดเดรส) ดังนั้นเมื่อมีการอินเตอร์รัพต์เข้ามาแล้ว ตัวโปรเซสเซอร์จะกระโดดไปที่ส่วนของโปรแกรม ที่ต้องการทำงานตามวัตถุประสงค์ ของอินเตอร์รัพต์นั้น หลังจากเก็บข้อมูลต่างๆ ของโปรแกรมเคาท์เตอร์ลงบนสแต็ก

การจัดการและการควบคุมอินเตอร์รัพต์มีรีจีสเตอร์พิเศษ 2ตัว สำหรับการกำหนดรูปแบบการทำงานต่างๆ ซึ่งได้แก่รีจีสเตอร์พิเศษ IE(interupt enable , ที่อยู่แอดเดรส 0AH) และ IP(interupt priority, ที่อยู่แอดเดรส 0B8H)การกำหนดรูปแบบอินเตอร์รัพต์ ทำได้โดยการเซตหรือรีเซตตำแหน่งบิตใน IE อย่างเหมาะสมดังแสดงความหมายของแต่ละบิตใน IE ในรูปที่ 2 สำหรับในรูปที่ 2 แสดงความหมายแต่ละบิตในรีจีสเตอร์ TCON

EA	**X**	**ET2**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**

ภาพที่ 3.14 แสดงหน้าที่การทำงานของแต่ละบิตในรีจีสเตอร์

สัญลักษณ์	ตำแหน่ง	หน้าที่
EA	IE.7	ทำการดิสเอเบิลทุกอินเตอร์รัพต์ ใดๆ ทั้งหมด ถ้าEA = "1"แต่ละอีนาเบิลหรือดีสเอเบิล โดยการเซตหรือเคลียร์อีนาเบิลบิต
	IE.6	สงวนไว้ไม่ใช้งาน
ET2	IE.5	ทำการอีนาเบิลหรืออินเตอร์รัพต์ จากโอเวอร์โฟลว์ของไทเมอร์ 2 ถ้า ET2 =0 อินเตอร์รัพต์จากไทม์เมอร์ 2 จะถูกดิสเอเบิล
ES	IE.4	ทำการอีนาเบิล หรือทำการ ดีสเอเบิลอินเตอร์รัพต์ จากพอร์ตอนุกรมถ้า ES= "0" อิน-เตอร์รัพต์นี้ จะถูกดิสเอเบิล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ET1	IE.3	ทำการอีนาเบิลหรือ เกิดการ ดีสเอเบิล อินเตอร์รัพต์จากโอเวอร์โฟลว์ ของไทม์เมอร์ 1 ถ้า ET1= 0 อินเตอร์รัพต์ไทม์เมอร์ 2จะถูกดีสเอเบิล
EX1	IE.2	ทำการอีนาเบิล หรือ เกิดการ ดีสเอเบิล อินเตอร์รัพต์ จากภายนอก 1 ถ้า ET1 =0 จะทำให้ อินเตอร์รัพต์นี้จะถูกดีสเอเบิล
ET0	IE.1	ทำการอีนาเบิล หรือเกิดการดีสเอเบิล อินเตอร์รัพต์จากโอเวอร์โฟลว์ ของไทม์เมอร์ 0 ถ้า ET0=0 อินเตอร์จากไทม์เมอร์2 จะถูกดีสเอเบิล
EX0	IE.0	ทำการอีนาเบิลหรือเกิดการ ดีสเอเบิล อินเตอร์รัพต์จากภายนอก 0 ถ้า EX0=0 อิน-เตอร์รัพนี้จะถูกดีสเอเบิล

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

ภาพที่ 3.15 แสดงหน้าที่การทำงานของแต่ละบิตในรีจีสเตอร์ TCON

สัญลักษณ์	ตำแหน่ง	หน้าที่
TF1	TCON.7	เป็นโอเวอร์โฟลว์แฟลกไทม์เมอร์ 1 จะถูกเซตโดยฮาร์ดแวร์เมื่อ ไทม์เมอร์, เคาท์เตอร์นี้เกิดโอเวอร์โฟลว์ จะถูกเคลียร์ด้วยฮาร์ดแวร์ เมื่อโปรเซสเซอร์เข้าสู่การอินเตอร์รัพต์รูทีน
TR1	TCON.6	บิตควบคุมการรันของไทม์เมอร์ 1 จะถูกเซตและเคลียร์โดยซอร์ฟแวร์เพื่อให้ไทเมอร์/เคาท์เตอร์นี้ ทำงาน หรือหยุดทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TF0	TCON.5	เป็นโอเวอร์โฟลว์แฟลก ไทม์เมอร์ 0 จะถูกเซตโดยฮาร์ดแวร์เมื่อ ไทม์เมอร์, เคาท์เตอร์ นี้เกิดโอเวอร์โฟลว์จะ ถูกเคลียร์โดยฮาร์แวร์เมื่อโปรเซสเซอร์เข้าสู่การอินเตอร์รัพต์รูทีน
TR0	TCON.4	บิตควบคุมการรันของไทม์เมอร์ 0 จะถูกเซตและเคลียร์โดยซอร์ฟแวร์เพื่อให้ไทเมอร์,เคาน์นี้ทำงานหรือหยุดทำงาน
IE1	TCON.3	แฟลกแสดง การอินเตอร์รัพต์ที่1 ทำงาน ที่ขอบขาลง จะถูกเซตโดยฮาร์ดแวร์เมื่อตรวจพบ สัญญาณอินเตอร์รัพต์จากภายนอกที่ ขอบขาลงและจะถูกเคลียร์เมื่อเข้าสู่ กระบวนการอินเตอร์รัพต์
IT1	TCON.2	บิตกำหนดรูปแบบอินเตอร์รัพต์ จากภายนอก 1 จะถูกเซตหรือเคลียร์โดยซอร์ฟแวร์ เพื่อกำหนดให้เกิดการอินเตอร์รัพต์ที่ขอบขาลง หรือระดับโลว์ของสัญญาณอินเตอร์รัพต์ภายนอก
IE0	TCON.1	แฟลกแสดงการ อินเตอร์รัพต์ที่ 0 ทำงานที่ขอบขา จะถูกเซตโดยฮาร์ดแวร์เมื่อตรวจพบว่ามี สัญญาณอินเตอร์รัพต์จากภายนอกที่ ขอบขาลงและจะถูกเคลียร์เมื่อเข้าสู่ กระบวนการอินเตอร์รัพต์
IT0	TCON.0	บิตกำหนดรูปแบบ อินเตอร์รัพต์จากภายนอก0 จะถูกเซตหรือเคลียร์โดยซอร์ฟแวร์ เพื่อที่จะมีการกำหนดให้เกิดอินเตอร์รัพต์ที่ขอบขาลงหรือระดับโลว์ของ สัญญาณอินเตอร์รัพต์ภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

X	XP	PT2	PS	PT1	PX1	PT0	PX0

ภาพที่ 3.16 แสดงหน้าที่การทำงานของแต่ละบิตในรีจีสเตอร์ IP

สัญลักษณ์	ตำแหน่ง	หน้าที่
-	IP.7	สงวนไว้ไม่ใช้งาน
-	IP.6	สงวนไว้ไม่ใช้งาน
PT2	IP.5	บิตกำหนดความสำคัญ ของการเกิดอินเตอร์รัพต์ไทเมอร์ 2 ถ้า PT2 = "1" อินเตอร์รัพต์จะมีความสำคัญระดับสูง
PS	IP.4	บิตกำหนดระดับความสำคัญ ของการอินเตอร์รัพต์ของพอร์ตอนุกรม ถ้าหาก PS = "1" อินเตอร์รัพต์จะมีความสำคัญระดับสูง
PT1	IP.3	บิตกำหนดระดับความสำคัญ ของการอินรัพต์ไทเมอร์ 1 ถ้าหากว่า PT1= "1" อินเตอร์รัพต์จะมีความสำคัญระดับสูง
PX1	IP2	บิตกำหนดความสำคัญ ของการเกิดอินเตอร์รัพต์จากภายนอก 1 ถ้าหากว่า PX1 = "1" อินเตอร์รัพต์จะมีค่าความสำคัญระดับสูง
PT0	IP.1	บิตกำหนดกระดับความสำคัญ ของการอินเตอร์รัพต์ไทเมอร์ 0 ถ้า PT0 =1 อินเตอร์รัพต์จะมีความสำคัญระดับสูง
PX0	IP.0	บิตกำหนดระดับความสำคัญ ของการอินเตอร์รัพต์จากภายนอก 0 ถ้าหากว่า PX0=1 อินเตอร์รัพต์จะมีความสำคัญระดับสูง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.2 แสดงตำแหน่งเริ่มต้นของโปรแกรมบริการอินเตอร์รัพต์แต่ละชนิดที่เกิดขึ้น

ชนิดของอินเตอร์รัพท์	ลำดับความสำคัญ	ตำแหน่งเริ่มต้นของตำแหน่งบริการอินเตอร์รัพท์
อินเตอร์รัพท์ภายนอกชนิด 0 (IE0)	1	0003H
อินเตอร์รัพท์ของไทเมอร์ 0 (TF0)	2	000BH
อินเตอร์รัพท์ภายนอกชนิด 1 (IE1)	3	0013H
อินเตอร์รัพทไทเมอร์ชนิด 1 (TF1)	4	001BH
อินเตอร์รัพท์อนุกรม 0 (TI+RI)	5	0023H

3.10 พอร์ตของ 8051

8051 เป็นไมโครคอลโทรลเลอร์ขนาด 40 ขา ซึ่งมีขาต่างๆดังนี้

- Vcc (ขา40) ต่อกับ+5V
- Vss (ขา20) เป็นขา GND
- พอร์ต 0 (ขา 32-39) มีทั้งหมด 8 บิต คือ (P0.7-P0.0) มีโครงสร้างแบบ Open Drain Bi-Directional

พอร์ต 0 (ขา32-39) มีทั้งหมด8บิตคือ (P0.7-P0.0) ใช้งานได้ 2 หน้าที่ คือแอดเดรสบัสและดาต้าบัสเมื่อต้องการติดต่อกับหน่วยความจำภายนอกหรือเป็นไอโอพอร์ต ถ้าต้องการให้ทำงานเป็นอินพุตพอร์ตจะต้องส่งลอจิก 1 ไปยังพอร์ตนี้ จะมีผลให้ Q ของ D-FF เป็น 0 ทำให้EFT ตัวล่างมีสถานะ OFF สัญญาณที่ใช้อ่านอินพุทพอร์ทแลทช์โดยส่งสัญญาณ READ LATCH ไปกระตุ้นที่ Tri-State Buffer ตัวบนและการอ่าน Port (pin) จะใช้สัญญาณ Read (pin)

พอร์ต 1 (ขา1-8) มีทั้งหมด 8 บิตคือ (P1.0-P1.7)มีโครงสร้างคล้ายพอร์ต 0 แต่จะใช้ความต้านทานภายในพูลอัพแทน Internal Pull up Register

พอร์ต 2 (ขา 21-28) มีทั้งหมด 8 บิต คือ (P2.7-P2.0) มีโครงสร้างคล้ายพอร์ต 0 โดยมี FET ตัวล่างตัวเดียว ส่วนตัวบนใช้ความต้านทานพูลอัพแทน พอร์ตนี้ทำงาน 2 หน้าที่ คือ สามารถใช้เป็นเอสเดรสบัสขนาด 8 บิต (A15-A8) และเป็นไอโอพอร์ตที่ใช้งานทั่วไป เมื่อจะใช้งานเป็นอินพุทพอร์ต ต้องส่งลอจิก 1 มาที่พอร์ตนี้ก่อนเพื่อบังคับให้ FET อยู่ในสภาวะ off

พอร์ต 3 (ขา 10-17) มีทั้งหมด 8 บิต คือ (P3.7-P3.0) มีโครงสร้างคล้ายพอร์ต 1 พอร์ตนี้ทำงาน 2 หน้าที่ คือ เป็นไอโอพอร์ตที่ใช้งานทั่วไป เมื่อใช้งานเป็นอินพุทพอร์ต ต้องส่ง ลอจิก 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มาที่พอร์ตก่อน และอีกหน้าที่หนึ่งคือ ใช้ส่งสัญญาณควบคุมออกมาและรับสัญญาณเข้าไป สัญญาณต่าง ๆ มีดังนี้

P3.0/RXD (Serial Input Port) เป็นขาที่ใช้รับข้อมูลแบบอนุกรม(UART)

P3.1/TXD (Serial Output Port) เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม(UART)

P3.2/INT0 (Eternal Interupt 0) ใช้รับสัญญาณการขัดจังหวะจากภายนอกเบอร์ 0

P3.3/INT1 (Eternal Interupt 1) ใช้รับสัญญาณการขัดจังหวะจากภายนอกเบอร์ 1

P3.4/T0 (Counter 0 External Input) ขารับสัญญาณพัลซ์อินพุทเข้าไปยังวงจร Counter 0 (เป็นอินพุทโหมดเคาน์เตอร์)

P3.5/T1 (Counter 1 External Input) ขารับสัญญาณพัลซ์อินพุทเข้าไปยังวงจร Counter 1 (เป็นอินพุทโหมดเคาน์เตอร์)

P3.6/WR (External Data Memory Write Strope) ขาสัญญาณควบคุมการเขียนข้อมูลลงหน่วยความจำข้อมูลภายนอก

P3.7/RD (External Data Memory Read Strope) ขาสัญญาณควบคุมการอ่านข้อมูลลงหน่วยความจำข้อมูลภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
ชุดคำสั่งของ MCS-51

4.1 กลุ่มคำสั่งทางคณิตศาสตร์

กลุ่มคำสั่งกลุ่มแรกที่จะศึกษาในกลุ่มคำสั่งทั้งหมดของ MCS-51 คือกลุ่มคำสั่งทางคณิตศาสตร์ กลุ่มคำสั่งทางคณิตศาสตร์ใน MCS-51 ประกอบด้วยกลุ่มคำสั่งที่เกี่ยวข้องกับการกระทำทางคณิตศาสตร์ ดังต่อไปนี้

- กลุ่มคำสั่งเพิ่มหรือลดค่าข้อมูลครั้งละหนึ่ง (Incrementing and Decrementing Instrution)
- กลุ่มคำลั่งบวกเลขจำนวนเต็ม (Addition Instructions)
- กลุ่มคำสั่งคูณเลขจำนวนเต็ม (Multiple Instructions)
- กลุ่มคำสั่งหารเลขจำนวนเต็ม (Devide Instructions)
- กลุ่มคำสั่งปรับค่าผลลัพธ์จากการบวกเลขรหัส BCD (Decimal Adjust Instructions)

4.1.1กลุ่มคำสั่งเพิ่มหรือลดค่าข้อมูลครั้งละหนึ่ง

คำสั่งนี้เป็นคำสั่งพื้นฐานที่สุดของกลุ่มคำสั่งทางคณิตศาสตร์จากในตารางที่ 1 จะเห็นว่าคำสั่งนี้มีวิธีการเข้าถึงข้อมูลได้หลายวิธี ดังคำสั่งทั้งหมดในกลุ่มนี้ มีดังนี้

ตารางที่ 4.1 แสดงกลุ่มคำสั่งเพิ่มหรือลดค่าข้อมูลครั้งละหนึ่ง

INCA	DEC A
INC Rn	DEC Rn
INC direct	DEC direct
INC @ Ri	DEC @ Ri
INC DPTR	ไม่มีคำสั่ง DEC DPTR

หมายเหตุ จะสังเกตได้ว่าใน MCS-51 ไม่มีคำสั่ง DEC DPTR ที่จะมาจับคู่คำสั่ง INC DPTR

คำสั่ง INC และ DEC จะใช้ในกรณีเพิ่มหรือลดค่าข้อมูลในหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปหรือที่เป็นรีจีสเตอร์ ใช้งานเฉพาะครั้งละ 1 โดยไม่จำเป็นต้องใช้คำสั่งบวกหรือ ลบ ซึ่งต้องอาศัยรีจีสเตอร์ A เป็นการเฉพาะหรืออาจจะเป็นคำสั่งกลุ่มนี้ในส่วนของโปรแกรมที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วนรอบการทำงาน เป็นจำนวนครั้งตามที่ต้องการ โดยการเพิ่มหรือลดค่าข้อมูลที่ใช้เป็นตัวนับรอบไปเรื่อยๆ จนกระทั่งได้จำนวนรอบที่ต้องการ กลุ่มคำสั่งทางคณิตศาสตร์กลุ่มแรกที่กล่าวมานี้ไม่มีผลต่อแฟล็กใน MCS-51 ใดๆ ทั้งสิ้นยกเว้นหากข้อมูลถูกเพิ่มหรือลดค่ารีจีสเตอร์ PSW เอง

4.1.2กลุ่มคำสั่งบวกเลขจำนวนเต็ม

กลุ่มคำสั่งนี้มีวิธีการเข้าถึงข้อมูลได้ หลายวิธี ดังนั้นคำสั่งทั้งหมดในกลุ่มนี้จะประกอบไปด้วยคำสั่งย่อยๆ ต่อไปนี้

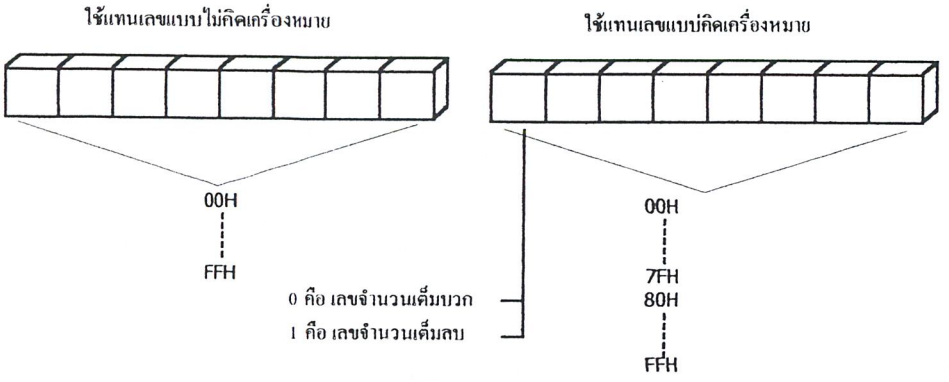
ADD A,Rn	ADDC A,Rn
ADD A,direct	ADDC A,direct
ADD A,@Ri	ADDC A,@Ri
ADD A,#data	ADDC A,#data

หมายเหตุ สังเกตว่าคำสั่งในการบวกนี้ ผลลัพธ์จะนำไปเก็บไว้ในรีจีสเตอร์ A เสมอ

เนื่องจากซีพียู MCS-51 มีขนาด 8 บิตดังนั้นการปฏิบัติคำสั่งบวกหรือลบ จะกระทำครั้งละ 8บิตเสมอ และจากทฤษฎีหลักการนับ เราทราบว่าเลขไบนารี่ขนาด 8 หลักสามารถแทนจำนวนเต็มที่มีค่าตาต่างกันได้ทั้งสิ้น 256 จำนวน จำนวนนี้อาจจะเป็นเลขจำนวนเต็มบวกเพียงอย่างเดียว ก็จะมีค่าตั้งแต่ 0 ถึง 255 แต่หากต้องการแทนเลขไบนารี่ขนาด 8 หลัก ให้เป็นทั้งเลขจำนวนเต็มบวกและลบด้วย จำเป็นต้องใช้เทคนิคพิเศษ เพื่อแยกความแตกต่างระหว่างทั้งสองจำนวน วิธีการทั่วไปที่ใช้แยกความแตกต่างคือ การใช้บิตสูงสุด (MSB Significant Bit) ของเลขไบนารี่ 8 หลัก เป็นการบอกเครื่องหมายของเลขจำนวนเต็มที่ต้องการแทนโดยมีเงื่อนไขว่าหากบิตนี้มีค่า เป็น “0” หมายความว่าเลขนั้นเป็นเลขเต็มบวก หากบิตนี้มีค่าเป็น “1” หมายถึงเลขจำนวนเต็มลบ เมื่อเราใช้บิตสูงสุดแทนเครื่องหมายแล้วจะเหลือเลขจำนวนเต็มลบ และเมื่อเราใช้บิตสูงแทนเครื่องหมายแล้วจะ เหลือเลขไบนารีสำหรับใช้แทนเลขจำนวนเต็มฐาน 10 เพียง 7 หลัก ดังนั้นเลขจำนวนเต็มบวกที่สามารถแทนได้มีค่าอยู่ระหว่าง 0- 127 รวม 128 จำนวน ส่วนเลขจำนวนเต็มลบ จะมีค่าตั้งแต่ –1 ถึง –128 รวม 128 จำนวนเช่นกันเลขจำนวนลบนี้จะแทนอยู่ในรูป 2’s complement ดังแสดงในภาพที่ 4.1

เมื่อเราใช้เลขไบนารี่ 8 หลัก สำหรับจำนวนเต็มใช้แทนค่าต่างๆ ได้ทั้งสิ้น 256 ค่าซึ่งจะเป็นเลขที่มีเครื่องหมายหรือไม่นั้นขึ้นอยู่กับผู้เขียนโปรแกรมจะกำหนดไว้ ในการบวกเลขทั้งสองชนิด (เลขที่คิดและไม่คิดเครื่องหมาย) มักเกิดปัญหาเมื่อผลลัพธ์ที่ได้มีค่าเกินกว่าที่เลขไบนารี่ 8 หลักจะสามารถแทนได้ กรณีของปัญหาที่อาจเกิดขึ้นสามารถแยกกล่าวทีละหัวข้อ ได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่4.1 การใช้เลขไบนารี่ขนาด 8 บิต แทนจำนวนเต็มบวกอย่างเดียวและเลขจำนวนเต็มบวกและเลขจำนวนเต็มลบรวมกัน

กรณีบวกเลขที่ไม่คิดเครื่องหมายบวก นั่นคือใช้เลขไบนารี่ทั้ง 8 หลัก แทนขนาดของจำนวน ซึ่งจำนวนสูงสุดที่แทนได้คือ 256 จำนวน ดังนั้น ในการบวกเลขประเภทนี้หากผลลัพธ์ที่ได้มีค่าเกิน 256 จะทำให้เกิดปัญหาดังแสดงในตัวอย่างต่อไปนี้

150+	10010110+		
30	00011110	ผลลัพธ์ไม่เกิน 255	
= 180	= 10110100	carry flag เคลียร์	
150+	10010110+		
150	10010110	ผลลัพธ์เกิน 255	
= 300	=1 00101100	carry flag เซต	

จากตัวอย่างจะเห็นว่าหากผลลัพธ์มีขนาดเกิน 255 จะทำให้เลขไบนารี่ที่ใช้แทนผลลัพธ์มีค่าผิดพลาดไปจากความเป็นจริง และเมื่อพิจารณาให้ดีจะพบว่า เมื่อผลลัพธ์จากการบวกเลขจำนวนเต็มที่ไม่คิดเครื่องหมายมีขนาดเกิน255 จำเป็นจะต้องใช้เลขไบนารี่เพิ่มอีก1 บิต เพื่อใช้แทนผลลัพธ์เสมอ ดังนั้นเมื่อบวกเลขแล้วได้ผลลัพธ์เกินพิกัด เลขไบนารี่ 1 หลัก ที่เกินมานี้จะไปเซตบิต carry flag ของรีจีสเตอร์ใช้งานเฉพาะ PSW เพื่อให้ตรวจสอบได้ในกรณีที่มีการบวกเกินพิกัด ดังนั้นในการบวกเลขประเภทนี้บิต carry flag จะเป็นตัวบอกสถานะของผลลัพธ์ว่ามีขนาดเกินพิกัดหรือไม่

กรณีที่บวกเลขที่คิดเครื่องหมาย การบวกเลขที่คิดเครื่องหมายมีความเป็นไปได้สอง กรณีเท่านั้น คือ เลขที่นำมาบวกกันเป็นเครื่องหมายเดียวกันหรือเครื่องหมายต่างกัน หากเลขที่นำบวกกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มีเครื่องหมายต่างกัน ผลลัพธ์ที่ได้ย่อมถูกต้องเพราะสามารถแทนด้วยเลขฐานสองได้เสมอ ในกรณีนี้ overflow flag ถูกเคลียร์เสมอ แค่บิต carry flag อาจถูกเซตหรือเคลียร์ขึ้นกับค่า ของเลขที่นำมาบวกกัน ได้ตัวอย่างต่อไปนี้

1101 1000	-40	
0111 1000	120+	carry flag เซต
1 0101 0000	80	overflow flag เคลียร์

ในกรณีเลขที่ต้องการนำมาบวกกันมีเครื่องหมายเหมือนกัน ผลลัพธ์ที่ได้อาจจะถูกต้องหรือไม่ถูกต้องอันเนื่องมาจากไม่สามารถใช้เลขฐานสองแทนผลลัพธ์ที่เกิดขึ้นได้ ดังนั้นการบวกเลขที่มีเครื่องหมายเหมือนกันอาจจะเกิดการที่แตกต่างกันได้ 4 กรณีดังนี้

- บวกเลขจำนวนเต็มบวก ได้ผลลัพธ์ไม่เกินพิกัด ดังตัวอย่าง

0000 0010	2	
0000 0101+	5+	
= 0000 0111	= 7	overflow flag ถูกเคลียร์

- บวกเลขจำนวนเต็มบวก ได้ผลลัพธ์เกินพิกัด ดังตัวอย่าง

0110 0100	100	
0110 0100+	100+	
1100 1000	200	overflow flag ถูกเซต

- บวกเลขจำนวนเต็มลบ ได้ผลลัพธ์ไม่เกินพิกัด ดังตัวอย่าง

1101 1000	-40	
1100 1100	-50+	
1 1010 0110	-90	overflow flag ถูกเคลียร์

- บวกเลขจำนวนเต็มลบ ได้ผลลัพธ์เกินพิกัด ดังตัวอย่าง

1100 0000	-64	
1011 1010	-70+	
1 0111 1010	-134	overflow flag ถูกเซต

จากตัวอย่างการบวกเลขที่มีเครื่องหมายเหมือนกันข้างต้นนี้ สรุปได้ว่ากรณีที่ overflow flag ถูกเซตหมายความว่าผลลัพธ์ที่ได้มีค่าเกินเลขจำนวนเต็มบวกสูงสุดหรือเลขจำนวนเต็มสูงสุดที่สามารถแทนได้ ดังนั้นกรณี ต่างๆ ที่อาจเกิดขึ้นในการบวกเลขที่มีเครื่องหมายสามารถตรวจสอบได้จาก overflow flag ดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เลขจำนวนเต็มบวก + เลขจำนวนเต็มบวก ได้ผลลัพธ์เกินพิกัด

overflow flag เซต

เลขจำนวนเต็มบวก + เลขจำนวนเต็มบวก ได้ผลลัพธ์ไม่เกินพิกัด

overflow flag เคลียร์

เลขจำนวนเต็มลบ + เลขจำนวนเต็มลบ ได้ผลลัพธ์เกินพิกัด

overflow flag เซต

เลขจำนวนเต็มลบ+เลขจำนวนเต็มลบ ได้ผลลัพธ์ไม่เกินพิกัด

overflow flag เคลียร์

เลขจำนวนเต็มบวก + เลขจำนวนเต็มลบ ได้ผลลัพธ์ไม่เกินพิกัดแน่นอน

overflow flag เคลียร์เสมอ

การบวกเลขที่มีขนาดหลายไบต์ ในหัวข้อที่ผ่านมาเราใช้เลขไบนารี่ขนาด 8 บิต เพื่อแทนจำนวนที่ต้องการนำมาบวก เลขจำนวนเต็มที่แทนได้มีค่าแตกต่างรวมกันทั้งสิ้น 256 ค่าโดยอาจจะเป็นเลขที่คิดเครื่องหมายหรือไม่คิดเครื่องหมายขึ้นอยู่กับผู้ใช้เป็นคนกำหนด แต่ในบางกรณีผู้เขียนโปรแกรมอาจต้องการความละเอียด ของจำนวนที่มากกว่านี้ก็สามารถกระทำได้ โดยการใช้เลขไบนารีที่มีขนาดมากขึ้นเพื่อแทนจำนวนที่ต้องการ เช่น ใช้เลขไบนารี่ขนาด 16 บิตแทนจำนวนต่างๆ ได้ทั้งสิ้น 65536 จำนวน เป็นต้น

การบวกเลขไบนารี่ที่มีขนาดหลายไบต์นี้ สามารถกระทำได้โดยนำมาบวกกันที่ละไบต์หากผลลัพธ์ที่ได้เกิด overflow จะต้องนำ carry flag ไปใช้ในเป็นตัวทดสำหรับการบวกในไบต์ถัดไป โดยการใช้คำสั่ง ADDC A,<byte>

4.1.3 กลุ่มคำสั่งลบเลขจำนวนเต็ม

กลุ่มคำสั่งที่ใช้ในการลบเลขจำนวนเต็มมีเพียงคำสั่งเดียวเท่านั้น คือ SUBB A,<byte> ดังแสดงในตารางที่ 1 ซึ่งจะเห็นได้ว่าสามารถใช้วิธีการเข้าถึงข้อมูลได้เหมือนกับคำสั่งบวก ดังนั้นคำสั่งในกลุ่มนี้จึงประกอบด้วยคำสั่งย่อยดังนี้คือ

SABB A,Rn	SUBB A,@ Ri
SABB A,direct	SUBB A,# data

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำสั่ง SABB A,<byte> นี้จะทำการลบบิต carry flag ออกจากผลลัพธ์ด้วยเสมอ ทั้งนี้เพื่อใช้ในการลบที่มีขนาดหลายไบต์ หากเราไม่ต้องการให้มีการลบบิต carry flag หรือต้องการลบเลขที่มีขนาดเพียง 12.1 ไบต์ ผู้เขียนโปรแกรมต้องเคลียร์ค่าของบิต carry flag ก่อนปฏิบัติคำสั่งนี้เสมอ

คำสั่งในการลบมีผลกระทบต่อ flag บางตัวในรีจีสเตอร์ใช้งานเฉพาะ PSW ดังนี้

- บิต carry flag : ถูกเซตเมื่อมีตัวยืมเกิดขึ้นที่บิตที่ 7
- บิต auxilary flag : ถูกเซตเมื่อมีตัวยืมเกิดขึ้นที่บิตที่ 3
- บิต overflow flag : ถูกเซตเมื่อมีตัวยืมเกิดขึ้นที่บิตที่ 7 แต่ไม่มีในบิตที่ 6 หรือตัวยืมเกิดขึ้นในบิตที่ 6 แต่ไม่มีในบิตที่ 7

การลบจำนวนเต็มที่มีลักษณะเช่นเดียวกับคำสั่งในการบวก ผู้เขียนในโปรแกรมสามารถแทนไบนารีที่ใช้ในการให้เป็นเลขจำนวนเต็มที่คิดเครื่องหมายหรือไม่คิดเครื่องหมายก็ได้ โดยการลบทั้งสองประเภทนี้รายละเอียดที่แตกต่างกันดังต่อไปนี้

กรณีที่ลบเลขไม่คิดเครื่องหมาย ถ้าเป็นเลขที่มีขนาดเพียง 1 ไบต์ ผู้เขียนโปรแกรมจะต้องเคลียร์ค่าของบิต carry flag ก่อนเสมอ แต่ถ้าเป็นเลขที่มีขนาดมากกว่า 1 ไบต์ ก็ให้เคลียร์บิต carry flag ในไบต์แรก แต่ในไบต์ถัดไปไม่ต้องเคลียร์ เพื่อให้สามารถนำผลจากไบต์แรก มาคิดในไบต์ต่อไปได้

ในกรณีที่ไม่มีเครื่องหมาย จะใช้เลขไบนารี่ขนาด 8 หลัก แทนขนาดของจำนวนโดย carry flag จะถูกเซตในกรณีที่ตัวตั้งน้อยกว่าตัวลบ (ได้ผลลัพธ์เป็นเลขลบ) แต่ overflow flag มีค่าเป็น 0 เสมอ ดังแสดงในตัวอย่างต่อไปนี้

125-	01111101-		100-	01100100-	
100	01100100		125	01111101	
25	00011001	carry flag เคลียร์	-25	00011001	carry flag เซต

กรณีที่ลบเลขที่คิดเครื่องหมาย เช่นเดียวกับบวกเลขที่คิดเครื่องหมาย เนื่องจากเลขที่นำมาลบกัน 2 ชนิด คือเลขจำนวนเต็มบวก และจำนวนเต็มลบ ดังนั้นจึงมีโอกาสเกิดเหตุการณ์ต่อไปนี้

- กรณีการลบเลขที่มีเครื่องหมายเหมือนกัน เนื่องจากลบเลขที่มีเครื่องหมายเดียวกัน ผลลัพธ์ที่ได้ไม่มีโอกาสเกิน –128 หรือ 127 ดังนั้นบิต overflow flag และ carry flag จะถูกเคลียร์เสมอดังตัวอย่างต่อไปนี้

-100 -	-25	100-	25-
-75	-100	25	100
25	75	125	-75

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- กรณีที่มีเครื่องหมายต่างกัน มีโอกาสที่ผลลัพธ์จะเกินหรือไม่เกินพิกัดของเลขไบนารี่ใช้แทนได้ โดยหากผลลัพธ์มีค่าเกินพิกัด (เกิน –128 หรือ +127) บิต overflow flag จะถูกเซตแต่ถ้าผลลัพธ์มีค่าไม่เกินพิกัดบิต overflow flagจะถูกเคลียร์ ดังแสดงในตัวอย่าง

-100-	-150-
25	50
-125 overflow flag ถูกเคลียร์	-200 overflow flag ถูกเซต

ดังนั้นในกรณีลบเลขที่คิดเครื่องหมาย overflow flag จะเป็นตัวบอกว่าขณะนี้ผลลัพธ์ที่ได้มีค่าเกินค่าต่ำสุดที่มีได้ ส่วนในการลบเลขที่ไม่คิดเครื่องหมาย overflow flag จะถูกเคลียร์เสมอ แต่ carry flag อาจถูกเซตหรือเคลียร์ขึ้นกับค่าของตัวตั้งและตัวลบ

4.1.4 กลุ่มคำสั่งคูณและหาร

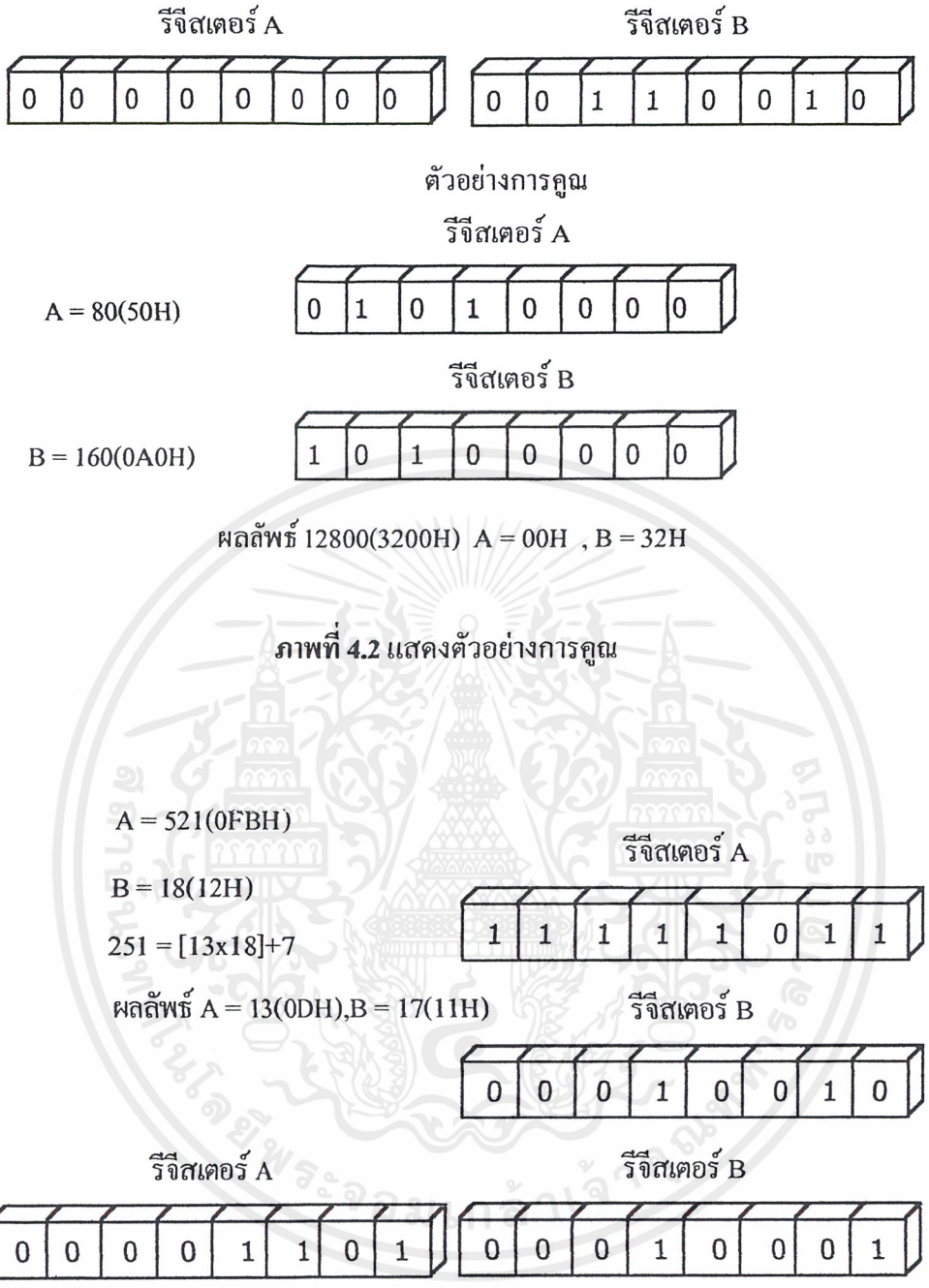
ในไมโครโปรเซสเซอร์ บางเบอร์ไม่มีคำสั่งคูณและหาร ผู้เขียนโปรแกรมจะต้องเขียนโปรแกรมเพื่อทำงานแทนคำสั่งคูณหรือหารเลขเอง ซึ่งนับว่ายุ่งยากและไม่สะดวก อีกทั้งยังเสียเวลาในการทำงานที่นานกว่าปกติ เพราะต้องปฏิบัติตามคำสั่งหลายคำสั่ง แต่ใน MCS-51 ได้อำนวยความสะดวกให้ผู้เขียนโปรแกรม โดยการบรรจุคำสั่งในการคูณหรือหารไว้ให้ผู้ใช้ด้วย คำสั่งทั้งสองนี้สามารถทำงานได้ภายในเวลาประมาณ 4 ไมโครวินาที

คำสั่งในการคูณหรือหารจะกระทำที่รีจีสเตอร์ A และ B ดังนี้

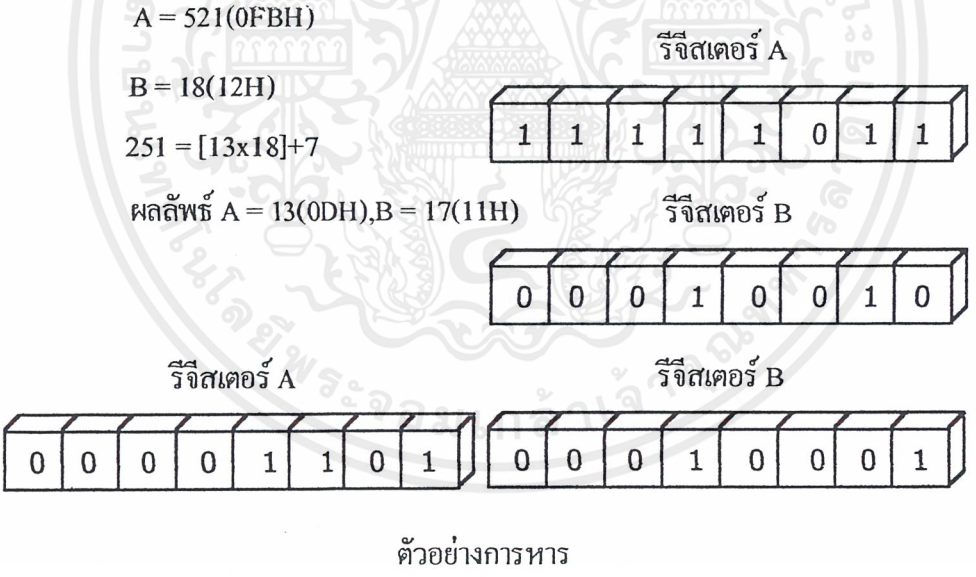
ในการคูณจะนำค่ารีจีสเตอร์ A มาคูณกับค่าในรีจีสเตอร์ B ผลลัพธ์ที่เก็บไว้ในรีจีสเตอร์ A และ B โดยรีจีสเตอร์เก็บไบต์ต่ำ ส่วนรีจีสเตอร์ B เก็บไบต์สูง overflow flag จะถูกเคลียร์หาค่าในไบต์สูงของผลลัพธ์ (เก็บไว้ในรีจีสเตอร์ B) มีค่าเป็นศูนย์ และถูกเซตเมื่อค่าในไบต์สูงของผลลัพธ์มีค่าไม่เป็นศูนย์ carry flag จะถูกเคลียร์เสมอ ส่วน auxilary flag ไม่ถูกเปลี่ยนแปลงในภายหลังคำสั่งนี้

ในการหารจะนำค่าในรีจีสเตอร์ A มาหารด้วยค่าในรีจีสเตอร์ B ผลลัพธ์ที่ได้เก็บไว้ในรีจีสเตอร์ A และ B โดยรีจีสเตอร์ A เก็บจำนวนเต็มที่ได้จากการหาร ส่วนรีจีสเตอร์ B เก็บเศษที่ได้จากการหาร หากมีการหารด้วยศูนย์(ค่าในรีจีสเตอร์ B เป็นศูนย์) overflow flag จะถูกเซตและค่าในรีจีสเตอร์ A และ B จะมีค่าเป็นอะไรก็ได้ หากตัวหารไม่เป็นศูนย์ overflow flag และ carry flag จะถูกเคลียร์ส่วน auxilary flag ไม่ถูกเปลี่ยนค่าภายหลังกระทำคำสั่งนี้ ดังแสดงในตัวอย่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 4.2 แสดงตัวอย่างการคูณ



ภาพที่ 4.3 แสดงตัวอย่างการหาร

4.1.5 กลุ่มคำสั่งที่ปรับค่าผลลัพธ์จากการบวกเลขรหัส

เนื่องจาก MCS-51 บวกเลขในฐานสิบหกหรือฐานสองเท่านั้น แต่ในบางครั้งที่เราต้องการบวกเลขฐานสิบซึ่งแทนด้วยรหัส BCD การบวกเลขฐานสิบที่แทนด้วยรหัส BCD ใน MCS-51 สามารถกระทำได้เช่นกันเพียงแต่ต้องใช้คำสั่ง DA A ต่อจากคำสั่งบวกตามปกติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำสั่ง DA A จะทำงานโดยการเปลี่ยนค่าสัมผัสที่ได้จากการบวกซึ่งเป็น เลขฐานสิบหก ให้กลายเป็นเลขฐานสิบซึ่งแทนด้วยรหัส BCD เอง

เลขฐานสิบมีวิธีการด้วยรหัส BCD โดยใช้เลขฐานสองหรือไบนารี่เพียง 4 บิต

0010 ใช้แทนเลข 2 0111 ใช้แทนเลข 7

0011 ใช้แทนเลข 3 1000 ใช้แทนเลข 8

0100 ใช้แทนเลข 4 1001 ใช้แทนเลข 9

จะเห็นได้ว่าเลขรหัส BCD ใช้เลขฐานสองเพียง 4 บิต มาแทนเลข 0 ถึง 9 ดังนั้นเลขไบนารี่ขนาด ไบต์สามารถแทนเลข BCD ได้ 2 หลัก ตั้งแต่ 00-99 ถ้าใช้เลขไบนารี่ 8 หลักแทนเลขฐานสิบแบบปกติจะแทนเลขฐานแบบปกติ และเมื่อใช้เลขไบนารี่เลขฐานสิบโดยใช้รหัส BCD

00000010 = ในรหัส BCD 00000010 = 2 ในเลขฐานสิบ

01001000 = ในรหัส BCD 01001000 = 72 ในเลขฐานสิบ

10001000 = ในรหัส BCD 10001000 = 132 ในเลขฐานสิบ

11111111 ไม่มีในในรหัส BCD 11111111 = 255 ในเลขฐานสิบ

จะเห็นว่ารหัส BCD ที่ใช้แทนเลขฐานสิบสามารถทำความเข้าใจและหาค่าเป็นเลขฐานสิบได้ง่ายกว่า ดังนั้นบางครั้งเราจึงนิยมที่จะบวกหรือลบเลขที่ใช้รหัส BCD เพราะรหัสนี้สื่อความหมายมากกว่าเลขฐานสองมาก แต่เนื่องจากซีพียู ใน MCS-51 บวกเลขได้ในฐานสองหรือ ฐานสิบหกเท่านั้น ดังนั้นหากเราแทนเลขที่จะทำการบวกด้วยรหัส BCD MCS-51 จะไม่มีการทราบว่าเลขที่ต้องการบวกเป็นเลข BCD หรือไม่ และจะทำการบวกแบบเลขฐานสอง ปกติ ผลลัพธ์ที่ได้จึงเป็นเลขฐานสองหรือฐานสิบหกทำให้ผลลัพธ์ไม่ตรงตามที่ต้องการของเรา ดังตัวอย่าง

00010010 B 12 BCD

+ 00110100 B 34 BCD+

01000110 B = 46 ฐาน 16 46 BCD

ไม่ต้องใช้คำสั่ง DA A เพราะได้ผลลัพธ์เท่ากัน

00010101 B 15 BCD

+00010101 B 15 BCD+

00101010 B = 2A ฐาน 16 30 BCD

ต้องใช้คำสั่ง DA A เพื่อเปลี่ยน 2Aเป็น 30

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตัวอย่างจะเห็นได้ว่าถ้าเลขที่นำมาบวกกันแต่ละหลักได้ผลลัพธ์เกิน 9 จะทำให้ผลลัพธ์นี้มีค่าเกินกว่าที่รหัส BCD จะสามารถแทนได้และ MCS-51 มีคำสั่ง DA A เพื่อช่วยปรับค่าของเลขฐานสิบหกที่ได้จากการบวกเลข BCD ให้เป็นเลขฐานสิบที่ถูกต้อง

ในตัวอย่างการบวก 15 กับ 15 ได้ผลลัพธ์เป็น 2A เมื่อใช้คำสั่ง DA A แล้วจะทำให้ผลลัพธ์ที่ได้จากการบวกในรีจีสเตอร์ A ถูกเปลี่ยนเป็น 30

สิ่งหนึ่งที่ผู้เขียนโปรแกรมควรระวังคือ ก่อนใช้คำสั่ง DA A จำนวนที่จะนำมาบวกเข้าด้วยกันต้องเป็นเลข BCD เท่านั้น

4.2 กลุ่มคำสั่งทางตรรกศาสตร์

คำสั่งกลุ่มต่อไปนี้จะศึกษาต่อจากกลุ่มคำสั่งทางคณิตศาสตร์คือ กลุ่มคำสั่งทางตรรกศาสตร์ ตัวดำเนินการทางตรรกศาสตร์ (operator) ที่สำคัญๆ มีอยู่ด้วยกัน 4 ประเภท ซึ่งสามารถเทียบกับคำสั่งในกลุ่มคำสั่งทางตรรกศาสตร์ของ MCS-51 ได้ดังนี้

AND เทียบได้กับคำสั่ง ANL (AND logical)

OR เทียบได้กับคำสั่ง ORL (OR logical)

XOR เทียบได้กับคำสั่ง XRL (exclusive- OR logical)

NOT เทียบได้กับคำสั่ง CPI (complement)

กลุ่มคำสั่งทางตรรกศาสตร์ที่มีใน MCS-51 นอกจากจะมีคำสั่งในที่ใช้ใน การกระทำทางตรรกศาสตร์โดยทั่วไปแล้ว ยังรวมเอาคำสั่งในการเลื่อนข้อมูลเป็นวงรอบครั้งละ บิต (rotate) คำสั่งในการสลับที่ข้อมูล 4 บิตบน และ 4 บิตล่างในรีจีสเตอร์ A ไว้เป็นประเภทหนึ่งของกลุ่มคำสั่งทางตรรกศาสตร์อีกด้วย

คำสั่งทั้งหมดในกลุ่มคำสั่งทางตรรกศาสตร์ที่เกี่ยวข้องกับการกระทำทางตรรกศาสตร์จริงๆ ในกลุ่มนี้มีเพียง 4 คำสั่งเท่านั้น คือ ANL, XRL, CPL โดยแต่ละคำสั่งจะกระทำการกับข้อมูลแต่ละบิตโดยตรงดังแสดงในตัวอย่างต่อไปนี้

- คำสั่ง ANL : AND ข้อมูลแต่ละบิต ดังนี้

  โอเปอร์แรนด์ตัวแรก 10110110 B = 0B6H

  โอเปอร์แรนด์ตัวที่สอง 01101101 B = 6DH

  ผลลัพธ์ 00100100 B = 24H

- คำสั่ง ORL : OR ข้อมูลแต่ละบิตดังนี้

  โอเปอร์แรนด์ตัวแรก 10110110 B = 0B6H

  โอเปอร์แรนด์ตัวที่สอง 01101101 B = 6DH

  ผลลัพธ์ 11011011 B = 0DBH

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- คำสั่ง CRL : complement ข้อมูลแต่ละบิต ดังนี้

  โอเปอร์แรนด์ 10110110 B = 0B6H

  ผลลัพธ์ 010010011B = 49H

กลุ่มคำสั่งทางตรรกศาสตร์ ANL, ORL, XRL ที่แสดงในตารางที่ สามารถแยกออกได้เป็น 2 ประเภท ได้ดังนี้

คำสั่งที่ใช้รีจีสเตอร์เป็นที่เก็บผลลัพธ์ (destination register) คำสั่งนี้จะสามารถกระทำการทางตรรกศาสตร์กับโอเปอร์แรนด์อีกตัวหนึ่ง โอเปอร์แรนด์ตัวที่จะนำมากระทำการกับรีจีสเตอร์ A นี้สามารถใช้วิธีการเข้าถึงข้อมูลได้ทุกวิธี แต่ผลลัพธ์สุดท้ายจะนำไปเก็บไว้ในรีจีสเตอร์ A ตามเดิม คำสั่งในกลุ่มนี้ มีดังต่อไปนี้

ANL A,Direct	ORL A,Direct	XRL A,Direct
ANL A,@ Ri	ORL A,@Ri	XRL A,@Ri
ANL A,Rn	ORL A,Rn	XRL A,Rn
ANL A,#data	ORL A,#data	XRL A,#data

คำสั่งที่ใช้หน่วยความจำสำหรับเก็บข้อมูลภายใน MCS-51 เป็นที่เก็บผลลัพธ์ คำสั่งกลุ่มนี้สามารถใช้วิธีเข้าถึงข้อมูลของโอเปอร์แรนด์ที่จะนำมากระทำการทางตรรกศาสตร์กับ ข้อมูลในหน่วยความจำสำหรับเก็บข้อมูลภายใน MCS-51 ได้เพียง 2 วิธีคือ ข้อมูลทางโอเปอร์แรนด์เป็นค่าคงที่ (immediate constants) ที่กำหนดเองในคำสั่ง หรือโอเปอร์แรนด์เป็นข้อมูลใน รีจีสเตอร์ A คำสั่งทางตรรกศาสตร์ในกลุ่มนี้มีดังนี้

ANL Direct,A	ORL Direct	XRL Direct,A
ANL Direct,#Data	ORL Direct,#data	XRL Direct.,#data

คำสั่งที่ใช้หน่วยความจำสำหรับเก็บข้อมูลภายใน MCS-51 เป็นที่เก็บผลลัพธ์

เนื่องจากหน่วยความจำสำหรับข้อมูลภายใน MCS-51 สามารถนำมากระทำการทางตรรกศาสตร์ได้โดยตรงนี้ทำให้เกิดความสะดวกแก่ผู้ใช้มาก เพราะรีจีสเตอร์ ที่ใช้ควบคุมฮาร์ดแวร์ใน MCS-51 ทุกตัวจะอยู่ในหน่วยความจำส่วนนี้ด้วย ดังนั้นการทำงานในตรรกศาสตร์ใดๆ กับข้อมูลในหน่วยความจำของ MCS-51 สามารถกระทำผ่านได้โดยไม่ต้องผ่านรีจีสเตอร์ A ด้วย ดังนั้นการส่งข้อมูลหรือรับข้อมูลจากพอร์ตรวมทั้งการกระทำทางด้านตรรกศาสตร์ แล้วนำผลที่ได้ไปส่งออกที่พอร์ตสามารถกระทำได้โดยตรงดังตัวอย่าง

XRL PO,OFFH

จะนำค่าของพอร์ต ไปทำการอินเวอรส์แล้วส่งกลับไปใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำสั่ง ANL, ORL, XRL ที่ได้กล่าวมานี้สามารถอ้างข้อมูลได้หลายวิธี แต่คำสั่ง CPL จะบังคับให้ทำงานที่รีจีสเตอร์ A เพียงตัวเดียวเท่านั้น นั่นคือคำสั่ง

CPL A

คำสั่งนี้ใช้เปลี่ยนค่าแต่ละบิตที่อยู่ในรีจีสเตอร์ A ให้เป็นตรงกันข้าม หากต้องการเปลี่ยนข้อมูลในหน่วยความจำตำแหน่งต่างๆ แต่ละบิตเป็นตรงกันข้าม ก็สามารถใช้คำสั่งนี้ผ่านรีจีสเตอร์ A โดยการนำข้อมูลในหน่วยความจำตำแหน่งที่ต้องการมาไว้ในรีจีสเตอร์ A จากนั้นก็กระทำคำสั่ง CPL A แล้วย้ายผลลัพธ์ที่ได้กลับไปยังตำแหน่งเดิม การเปลี่ยนค่าข้อมูลในหน่วยความจำแต่ละบิตเป็นตรงกันข้ามนี้อาจใช้คำสั่งแทนซึ่งจะสะดวกกว่ามาก นั่นคือคำสั่ง

XRL direct ,# OFFH

ซึ่งมีผลเกี่ยวกับการนำข้อมูลในหน่วยความจำแต่ละบิตมาเปลี่ยนเป็นตรงกันข้าม แล้วผลลัพธ์ที่ได้ก็จะเก็บไว้ในหน่วยความจำตำแหน่งเดิม

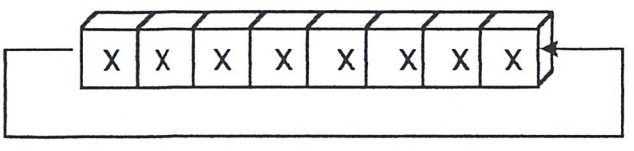
คำสั่งอื่นๆ ที่จัดอยู่ในกลุ่มคำสั่งทางตรรกศาสตร์ จะทำงานได้โดยไม่ได้กระทำการใดๆ ทางตรรกศาสตร์เลย แต่เราจะจัดไว้ในกลุ่มนี้ด้วยได้แก่

- คำสั่ง CRL A จะเป็นการเคลียร์ค่าในรีจีสเตอร์ A โดยการโหลดรีจีสเตอร์A ด้วย 00H แต่ถ้าผู้ใช้ต้องการเคลียร์ค่าในหน่วยความจำต่างๆ ก็สามารถกระทำได้โดยใช้คำสั่งอื่นแทนดังนี้

  ANL direct ,# 00H หรือ

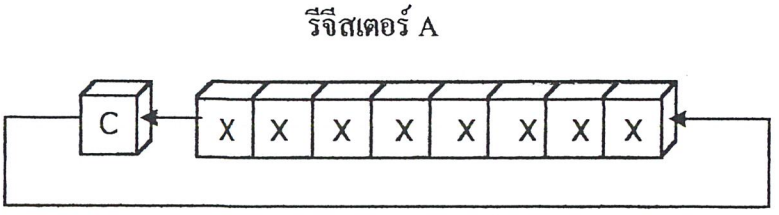
  MOV direct,# 00H
- คำสั่งในการเคลื่อนข้อมูลเป็นวงรอบ ซึ่งประไปด้วยคำสั่งย่อยๆ ดังนี้
- RL A มีการทำงานดังต่อไปนี้

รีจีสเตอร์ A

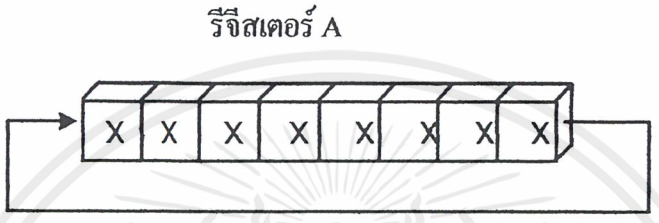


- RLC A มีการทำงานดังนี้

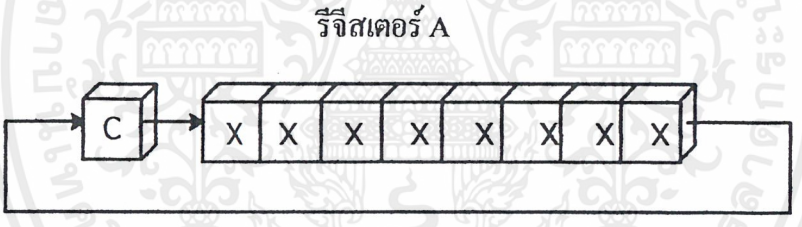
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



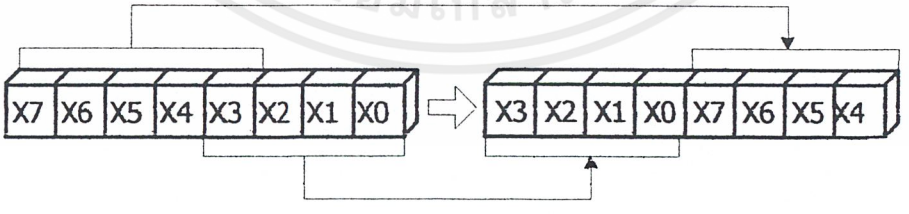
- RR A มีการทำงานดังต่อไปนี้



- RRC A มีการทำงานดังต่อไปนี้



- SWAP A มีการทำงานดังต่อไปนี้



ภาพที่ 4.4 แสดงคำสั่งในการเคลื่อนข้อมูลเป็นวงรอบ

4.3กลุ่มคำสั่งในการเคลื่อนย้ายข้อมูล

การเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำ หรือรีจีสเตอร์นับเป็นสิ่งจำเป็นสำหรับการทำงานของไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์ ทั้งนี้เพราะคำสั่งต่างๆ เช่น คำสั่งทาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คณิตศาสตร์ คำสั่งทางตรรกศาสตร์ ฯลฯ ส่วนใหญ่จะทำงานกับรีจีสเตอร์ใช้งานเฉพาะบางตัวเท่านั้น เช่นรีจีสเตอร์ A , B ดังนั้นหากเราต้องการนำข้อมูลมาประมวลผลโดยคำสั่งเหล่านี้ จำเป็นต้องย้ายข้อมูลที่ต้องการไปยังรีจีสเตอร์เหล่านี้ และได้ผลลัพธ์จากการคำนวณแล้วจึงเคลื่อนย้ายข้อมูลกลับมาเก็บไว้ในหน่วยความจำ ณ ตำแหน่งที่ต้องการ

คำสั่งในการเคลื่อนย้ายข้อมูล สามารถแบ่งออกได้ เป็น 3 กลุ่ม ตามประเภทของหน่วยความจำที่ใช้ในคำสั่ง ได้แก่

- คำสั่งในการเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำสำหรับเก็บข้อมูลภายในชิป
- คำสั่งในการเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป
- คำสั่งในการเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำสำหรับเก็บโปรแกรมทั้งภายในและภายนอกชิป

รายละเอียดของกลุ่มคำสั่งสำหรับการเคลื่อนย้ายทั้ง 3 กลุ่ม มีดังนี้

คำสั่งแต่ละคำสั่งในกลุ่มนี้มีการเข้าถึงข้อมูลได้หลายวิธี เราสามารถแบ่งประเภทของคำสั่งในกลุ่มนี้ออกเป็น 3 กลุ่ม ดังนี้

- กลุ่มคำสั่ง MOV (เคลื่อนย้ายข้อมูล) ได้แก่คำสั่ง

MOV Rn,A	MOV @Ri,A
MOV Rn,direct	MOV @Ri,direct
MOV Rn,#data 8 bit	MOV @Ri,#data 8 bit
MOV direct,A	MOV A,Rn
MOV direct,Rn	MOV A,direct
MOV direct,direct	MOV A,@Ri
MOV direct,@Ri	MOV A,data 16 bit

- กลุ่มคำสั่ง PUSH , POP ได้แก่คำสั่ง

PUSH direct	POP direct

- กลุ่มคำสั่ง XCH (สลับที่ข้อมูล) ได้แก่คำสั่ง

XCH A,Rn	XCH A,@Ri
XCH A,direct	XCHD A,@Ri

หมายเหตุ : Rn = R0-R7 =Ri

กลุ่มคำสั่ง MOV จะสังเกตุได้ว่าไม่มีคำสั่ง MOV Ri, Ri MOV @Ri,@Ri และเนื่องจาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รีจีสเตอร์ A อยู่ในหน่วยความจำสำหรับเก็บข้อมูลทั่วไปในชิปตำแหน่ง OEOH ดังนั้นการเคลื่อนย้ายข้อมูลที่เกี่ยวข้องกับรีจีสเตอร์ A โดยเฉพาะก็เพื่อลดขนาดของรหัสคำสั่งให้สั้นลงนั่นเอง นั่นคือคำสั่งที่ใช้ในการเคลื่อนย้ายข้อมูลรีจีสเตอร์ A นี้ ผู้ใช้ไม่จำเป็นต้องระบุตำแหน่งของหน่วยความที่เป็นรีจีสเตอร์ A ในรหัสคำสั่งเลย เพราะ MCS-51 จะเข้าใจว่าเป็นรีจีสเตอร์ A เองจากรหัสคำสั่ง

คำสั่ง MOV direct, direct มีไว้เพื่อเคลื่อนย้ายข้อมูลภายในหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปบริเวณ 128 ไบต์ล่าง และหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจีสเตอร์ใช้งานเฉพาะของ MCS-51

กลุ่มคำสั่ง PUSH , POP คำสั่ง PUSH , POP ทั้งสองนี้นับเป็นประเภทหนึ่งของกลุ่มคำสั่งในการเคลื่อนย้ายข้อมูล แต่การเคลื่อนย้ายของคำสั่ง PUSH , POP นี้มีจุดประสงค์ที่แตกต่างออกไปจากคำสั่ง MOV ทั่วไป คือคำสั่งทั้งสองใช้เพื่อเคลื่อนย้ายข้อมูลไปเก็บไปเก็บไว้ในหน่วยความจำตำแหน่งที่ถูกระบบโดยค่าในรีจีสเตอร์ใช้งานเฉพาะ SP ซึ่งเราเรียกบริเวณนี้ว่าสแตกแอเรีย และสามารถเรียกกลับคืนมาได้เมื่อต้องการ โดยมีรายละเอียดดังนี้

PUSH direct คำสั่งนี้สามารถใช้วิธีการเข้าถึงข้อมูลได้เพียงวิธีเดียวเท่านั้น คือ วิธีการเข้าถึงข้อมูลโดยตรง การทำงานของคำสั่งสามารถเทียบกับคำสั่งต่อไปนี้

INC SP

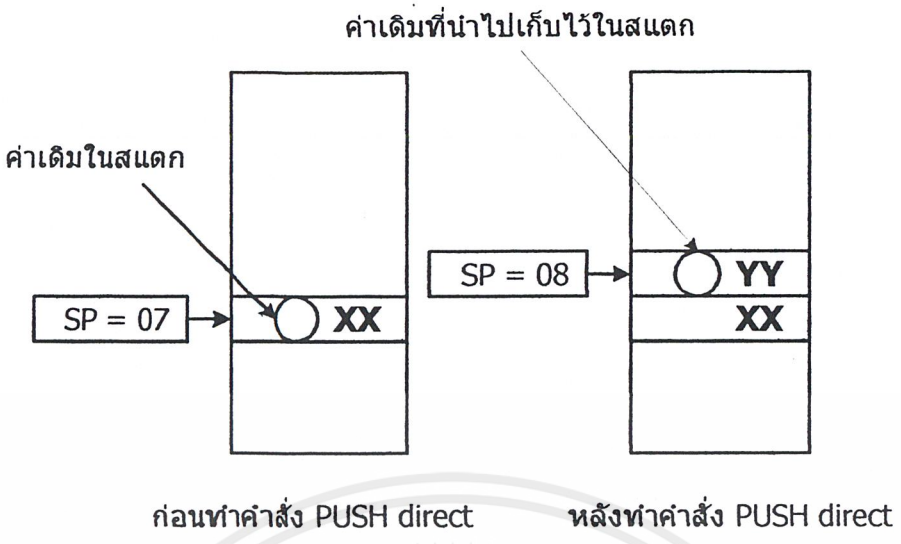
MOV @ SP ,direct ;คำสั่งนี้ไม่มีใน MCS-51

ในการทำงานของคำสั่ง ก่อนอื่นจะไปเพิ่มค่าของรีจีสเตอร์ใช้งานเฉพาะ SP ซึ่งจะทำหน้าที่เป็นตัวระบุตำแหน่งของหน่วยความจำที่จะนำข้อมูลในหน่วยความจำไปเก็บ หลังจากนั้นก็นำเอาข้อมูลในหน่วยความจำ ณ ตำแหน่งที่ถูกระบุโดยการเข้าถึงข้อมูลโดยตรงไปไว้ในหน่วยความจำ ณ ตำแหน่งเดียวกับรีจีสเตอร์ SP ชี้อยู่

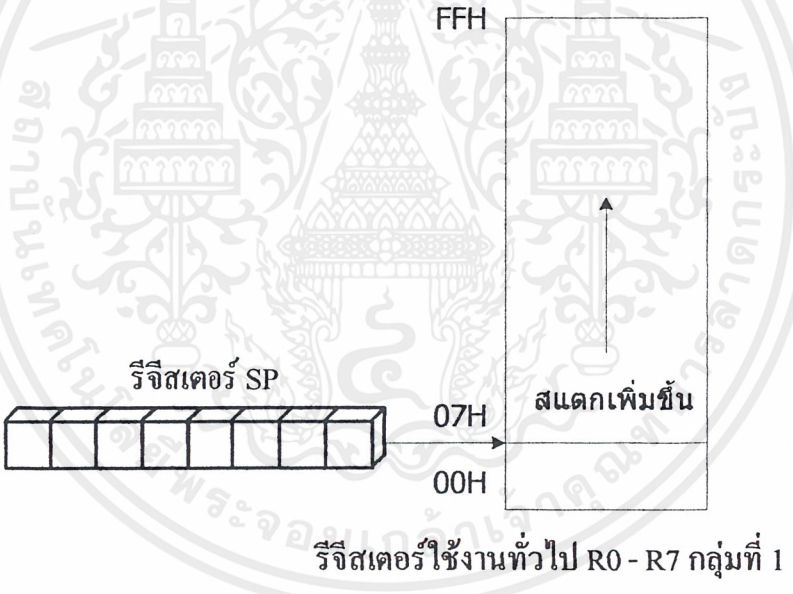
เนื่องจาก MCS-51 เพิ่มค่ารีจีสเตอร์ SP ให้เองเมื่อกระทำคำสั่ง PUSH ดังนั้นผู้เขียนโปรแกรมไม่จำเป็นที่จะต้องไปยุ่งเกี่ยวกับค่ารีจีสเตอร์นี้เลย ข้อมูลจะถูกเก็บในลักษณะที่ซ้อนกันไปเรื่อยๆ ดังแสดงในภาพที่ 4.5 และ 4.6

รีจีสเตอร์ SP ขนาด 8 บิต ดังนั้นสามารถระบุตำแหน่งหน่วยความจำได้เพียง 256 ไบต์เท่านั้น ทำให้บริเวณของสแตกแอเรียมีขนาดได้ไม่เกิน 256ไบต์ด้วย แต่เนื่องจากบริเวณของสแตกแอเรีย คือ ส่วนหนึ่งของหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปซึ่งใช้รีจีสเตอร์ทั่วไป R0-R7 ด้วย ดังนั้นขนาดของสแตกแอเรีย จะมีขนาดไม่ถึง 256 ไบต์ โดยในตอนรีเซตหรือเริ่มจ่ายพลังงานให้MCS-51 ค่าของรีจีสเตอร์ SP จะถูกปรับให้เป็น 07H เองเพื่อไม่ให้ไปซ้อนกับรีจีสเตอร์ทั่วไป R0-R7 ดังแสดงในภาพที่ 4.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 4.5 แสดงการทำงานของ PUSH direct



ภาพที่ 4.6 แสดงค่าของรีจีสเตอร์ SP เมื่อตอนรีเซต หรือเริ่มจ่ายพลังงานให้ MCS-51

จากภาพที่ 4.6 แสดงให้เห็นค่ารีจีสเตอร์ SP เมื่อตอนถูกรีเซตหรือจ่ายพลังงานให้ MCS-51 ซึ่งจะมีค่าเท่ากับ 07H ทำให้เราไม่สามารถใช้รีจีสเตอร์ทั่วไป R0-R7 กลุ่มอื่นได้ หากต้องการใช้รีจีสเตอร์ทั่วไป R0-R7 กลุ่มอื่น ผู้เขียนโปรแกรมจำเป็นต้องเปลี่ยนค่าของรีจีสเตอร์ใช้งานเฉพาะ SP ให้เป็นค่าอื่น เพื่อให้ข้อมูลที่ถูกนำมาเก็บไว้ในหน่วยความจำโดยคำสั่ง PUSH ไม่ไปซ้อนทับกับรีจีสเตอร์เหล่านี้โดยใช้คำสั่งเคลื่อนย้ายข้อมูลไปไว้ในรีจีสเตอร์ SP ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MOV SP,# ADDR คำสั่งนี้เป็นการเปลี่ยนค่าในรีจีสเตอร์ SP โดยตัวโปรแกรมแอสเซมเบลอร์ จะทราบตำแหน่งของรีจีสเตอร์ SP เอง ยกเว้นการแปลคำสั่งโดยการเปิดตาราง (ในตัวรหัสคำสั่งต้องระบุตำแหน่งของรีจีสเตอร์ SP ด้วยค่า 81H)

สแตกแอเรียส่วนที่มีค่าเกิน 128 ไบต์ จะอยู่ในหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปบริเวณ 128 ไบต์บน ซึ่งไม่มีใน MCS-51 ในบางเบอร์ ดังนั้นหากเราใช้ MCS-51เป็นเบอร์ที่ไม่มีหน่วยความจำในส่วนนี้สแตกแอเรียจะมีขนาดได้มากที่สุด 128 ไบต์เท่านั้นและเนื่องจากคำสั่ง PUSH จะเพิ่มค่าของรีจีสเตอร์ SP เองด้วยทุกครั้ง ดังนั้นหากก่อนกระทำคำสั่ง PUSH ค่าในรีจีสเตอร์ SP ขณะนั้นเป็น 7FH และผู้ใช้เลือก MCS-51 เบอร์ที่ไม่มีหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปบริเวณ 128 ไบต์บน จะทำให้ข้อมูลที่นำไปเก็บไว้โดยคำสั่ง PUSH เกิดการสูญหายได้ แต่ถ้าใช้เบอร์ที่มีหน่วยความจำนี้และก่อนทำคำสั่ง PUSH ค่าในรีจีสเตอร์ SP มีค่าเป็น 0FFH ข้อมูลที่จะนำมาเก็บจะถูกนำไปไว้ที่ตำแหน่ง 00H ซึ่งเป็นตำแหน่งเริ่มต้นของหน่วยความจำสำหรับเก็บ-ข้อมูลทั่วไปภายในชิป ทำให้ข้อมูลในรีจีสเตอร์ R0-R7 เปลี่ยนไปได้

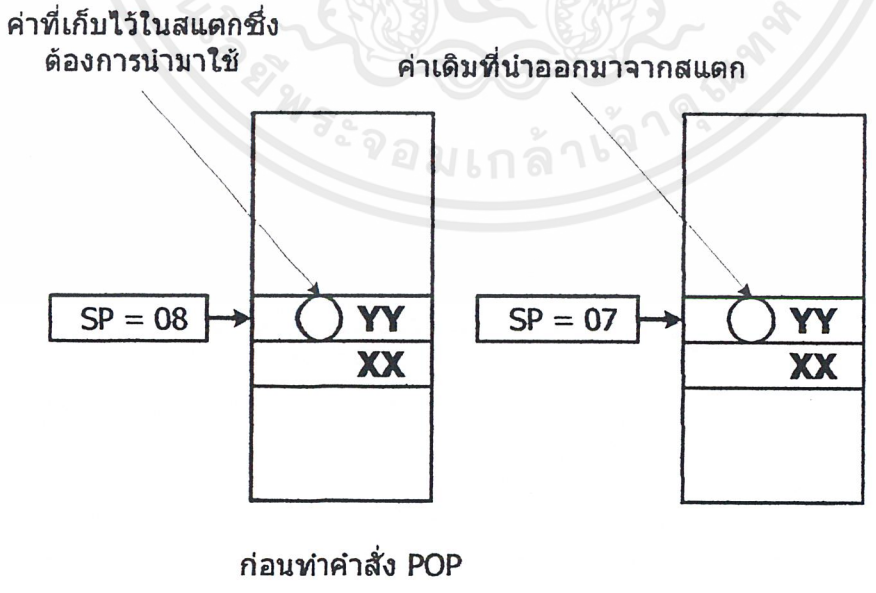
POP direct คำสั่งนี้จะทำหน้าที่ตรงกันข้ามกับคำสั่ง PUSH โดยมีวัตถุประสงค์เพื่อนำข้อมูลที่ถูกเก็บไว้ในหน่วยความจำโดยคำสั่ง PUSH กลับคืนมา

การทำงานของ POP direct สามารถเทียบได้กับคำสั่งต่อไปนี้

MOV direct, @SP ; คำสั่งนี้ไม่มีใน MCS-51

DEC SP

การทำงานของคำสั่งอาจอธิบายด้วยแผนภาพการทำงานดังแสดงในภาพที่ 4.7



ภาพที่ 4.7 แสดงการทำงานของ POP direct

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำสั่ง POP จะทำงานคู่กับคำสั่ง PUSH โดยเมื่อโปรแกรมต้องการใช้รีจีสเตอร์ตัวใดตัวหนึ่งในการประมวลผลและจำเป็นที่ต้องนำค่าเดิมมาใส่คืนเมื่อเสร็จสิ้นการทำงานแล้ว จะสามารถใช้คำสั่ง PUSH, POP เพื่อเก็บรักษาค่าเดิมไว้และนำค่าเดิมมาใส่คืนเมื่อต้องการ ดังแสดงในตัวอย่างต่อไปนี้

PUSH 0D0H : เก็บค่าของหน่วยความจำตำแหน่ง 0D0H (รีจีสเตอร์ PSW) คำสั่งที่มีการใช้รีจีสเตอร์ PSW หรือคำสั่งที่มีผลต่อค่าในรีจีสเตอร์ PSW

POP 0D0H : นำค่าที่เก็บไว้กลับคืนสู่รีจีสเตอร์ PSW

PUSH , POP ส่วนใหญ่จะใช้ในโปรแกรมย่อย ซึ่งถูกเรียกจากโปรแกรมหลัก (main program) โดยในโปรแกรมย่อยอาจมีการใช้รีจีสเตอร์ตัวใดตัวหนึ่งในการประมวลผล หรือมีผลกระทบในรีจีสเตอร์บางตัว ในกรณีนี้จำเป็นต้องมีการรักษาค่าเดิมของรีจีสเตอร์เหล่านี้ไว้เสียก่อนจากนั้นเมื่อทำงานในส่วนของโปรแกรมย่อยเสร็จแล้วจึงค่อยนำค่า เมื่อใดก็ได้โดยไม่ส่งผลกระทบต่อค่าในรีจีสเตอร์ที่โปรแกรมหลักใช้ในการประมวลผล ดังนั้น หากโปรแกรมย่อยมีการเก็บรักษาค่าของรีจีสเตอร์ที่ใช้ในระหว่างการทำงาน ผู้เขียนโปรแกรมไม่จำเป็นต้องกังวลกับการเปลี่ยนค่าในรีจีสเตอร์ที่ใช้ประมวลผลในโปรแกรมหลักเมื่อมีการเรียกใช้โปรแกรมย่อยแต่อย่างใด

กลุ่มคำสั่ง XCH คำสั่งนี้เป็นคำสั่งกลุ่มสุดท้ายของกลุ่มคำสั่งย้ายข้อมูลระหว่างหน่วยความจำสำหรับเก็บข้อมูลภายในชิป การทำงานในคำสั่งของคำสั่งจะมีลักษณะที่แตกต่างจากคำสั่ง MOV ,PUSH ,POP เพราะคำสั่งที่ผ่านมาทั้งหมดเป็นการเคลื่อนย้ายข้อมูลไปยังปลายทางนั่นเอง แต่คำสั่ง XCH (exchange) จะทำการสลับที่ข้อมูลระหว่างต้นทางและปลายทาง ทำให้ค่าของข้อมูลทั้งต้นทางและปลายทาง เปลี่ยนแปลงไปหลังจากกระทำการคำสั่งนี้ ยกเว้นกรณีที่ต้นทางและปลายทางมีค่าเท่ากัน คำสั่งนี้กระทำที่รีจีสเตอร์ A เสมอ ดังนั้น หากต้องการสลับข้อมูลในหน่วยความจำส่วนที่เก็บข้อมูล 2 ตำแหน่ง ก็สามารถกระทำได้โดยการสลับข้อมูลผ่านรีจีสเตอร์ A

คำสั่ง XCH A,<byte> สามารถใช้วิธีการเข้าถึงการข้อมูล ได้ 3 วิธี จึงมีคำสั่งนี้และคำสั่งย่อยอีก 3 คำสั่งคือ

XHA A, Rn

XCH A, direct

XCH A,@ Ri

นอกจากคำสั่งทั้ง 3 ยังมีคำสั่งพิเศษซึ่งจะทำการสลับที่ข้อมูลเพียง 44 บิตล่างของไบต์ข้อมูลเท่านั้น แต่มีวิธีการเข้าถึงข้อมูลเพียงวิธีเดียวเท่านั้น คือใช้ค่ารีจีสเตอร์ R0 , R1 เป็นตัวระบุตำแหน่งหน่วยความจำ (วิธีการเข้าถึงข้อมูลโดยทางอ้อม) คำสั่งนี้คือ

XCHD A, @ Ri

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.1กลุ่มคำสั่งในการเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป

กลุ่มคำสั่งนี้ เป็นกลุ่มคำสั่งที่ใช้เคลื่อนย้ายข้อมูลในหน่วยความจำสำหรับเก็บข้อมูลอยู่ใน MCS-51 เท่านั้น แต่เนื่องจาก MCS-51 เองสามารถติดต่อกับหน่วยความจำส่วนที่เก็บข้อมูลภายนอกได้ถึง 64 กิโลไบต์ ดังนั้นจึงต้องมีกลุ่มคำสั่งเพื่อใช้ในการติดต่อกับหน่วยความจำส่วนนี้ อยู่ด้วย

คำสั่งที่ใช้ติดต่อกับหน่วยความจำที่เก็บข้อมูลภายนอก MCS-51 นี้จะทำผ่านรีจีสเตอร์ A เสมอ และสามารถใช้วิธีการเข้าถึงข้อมูลได้เพียงวิธีเดียว คือ วิธีการเข้าถึงข้อมูลแบบอ้างอิง โดยใช้ค่ารีจีสเตอร์ R0 , R1 หรือ DPTR (ไม่สามารถใช้รีจีสเตอร์ SP ได้) เป็นตัวระบุตำแหน่งที่อยู่ของหน่วยความจำภายนอก

เนื่องจากรีจีสเตอร์ R0 ,R1 มีขนาดเพียง 8 บิต ทำให้สามารถระบุตำแหน่งที่อยู่ของหน่วยความจำภายนอกได้ในขอบเขต 256 ไบต์แรกเท่านั้น ส่วนรีจีสเตอร์ DPTR มีขนาด 16 บิต จึงสามารถระบุตำแหน่งหน่วยความจำได้ทั้งสิ้น 64 กิโลไบต์ ซึ่งเท่ากับความสามารถในการติดต่อของหน่วยความจำของ MCS-51

4.3.2 กลุ่มคำสั่งในการเคลื่อนย้ายข้อมูลจากหน่วยความจำสำหรับเก็บโปรแกรมทั้งภายนอกและภายในชิป

คำสั่งในกลุ่มนี้ประกอบด้วย 2 คำสั่ง

กลุ่มคำสั่งสองกลุ่มที่ศึกษาผ่านมาเป็นกลุ่มคำสั่ง ที่ใช้ในการเคลื่อนย้ายข้อมูลกับหน่วยความจำที่เก็บข้อมูลภายในและภายนอก MCS-51 หน่วยความจำที่เก็บข้อมูลนี้รักษาข้อมูลไว้ได้ก็ต่อเมื่อมีพลังงานเท่านั้น แต่ในบางกรณีเราจำเป็นต้องอาศัยข้อมูลที่มีค่าแน่นอนไม่เปลี่ยนแปลง เช่น รูปแบบของตัวอักษร ค่า sin ,cos , tan ของมุมต่างๆ ทางตรีโกณมิติ ซึ่งข้อมูลเหล่านี้จะจำเป็นต้องการเก็บรักษาไว้ตลอดเวลาถึงแม้ไม่มีพลังงานจ่ายให้ก็ตาม เพื่อให้สามารถเรียกข้อมูลมาทำงานได้ทุกครั้งที่เปิดอุปกรณ์ซึ่งควบคุมด้วย MCS-51 ดังนั้นเราจำเป็นต้องเก็บรักษาข้อมูลเหล่านี้ไว้ในหน่วยความจำซึ่งไม่ต้องอาศัยพลังงานในการเก็บรักษาข้อมูล นั่นคือหน่วยความจำที่เก็บโปรแกรมซึ่งเป็น ROM, EPROM นั่นเอง

เมื่อผู้ออกแบบได้เก็บข้อมูลไว้ในหน่วยความจำที่เก็บโปรแกรมแล้ว จำเป็นที่จะต้องนำคำสั่งในการเอาข้อมูลในหน่วยความจำส่วนนี้มาประกอบผลด้วย และใน MCS-51เองก็มีชุดคำสั่งในการเคลื่อนย้ายข้อมูลจากหน่วยความจำส่วนนี้มาประมวลผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำสั่งทั้งสองมีจุดหมายเพื่อเปิดตาราง โดยข้อมูลในตารางจะอยู่ในหน่วยความจำที่เก็บโปรแกรมซึ่งสามารถรักษาข้อมูลไว้ได้โดยไม่ต้องอาศัยพลังงาน

4.4 กลุ่มคำสั่งควบคุมการทำงานของโปรแกรม

โปรแกรมซึ่งเป็นลำดับที่ใช้ในการควบคุมการทำงานของไมโครคอนโทรลเลอร์ จะบรรจุอยู่ในหน่วยความจำถาวร ซึ่งอาจจะเป็น ROM , EPROM ฯลฯ ในการทำงานจริงๆ ซีพียู จะนำเอารหัสคำสั่งแต่ละคำสั่งในโปรแกรม ที่อยู่ในหน่วยความจำตำแหน่งต่างๆ (เฟตซ์คำสั่ง) มาตีความและปฏิบัติ ตามคำสั่งเป็นลำดับไปเรื่อยๆ โดยปกติการเฟตซ์คำสั่งจากโปรแกรมซีพียูจะกระทำโดยการส่งค่าหมายเลขประจำตำแหน่งหรือแอดเดรสของหน่วยความจำที่เก็บคำสั่งแต่ละคำสั่งไว้ เมื่อหน่วยความจำได้รับค่าตำแหน่งจากซีพียู มันจะส่งข้อมูลที่เป็นรหัสคำสั่งที่ให้แก่ไมโครคอนโทรลเลอร์เพื่อปฏิบัติงานต่อไป ในทางปฏิบัติไมโครคอนโทรลเลอร์จะส่งค่าตำแหน่งของหน่วยความจำซึ่งกำหนดโดยใช้ค่าในรีจีสเตอร์ PC ค่าของรีจีสเตอร์ ตัวนี้จะเพิ่มขึ้นครั้งละหนึ่งหลังการทำงานแต่ละคำสั่ง ของซีพียู เพื่อที่จะได้นำข้อมูลที่เป็นรหัสคำสั่งซึ่งอยู่ในตำแหน่งถัดไปจากหน่วยความจำมาทำงาน หรือเพื่อนำข้อมูลของคำสั่งที่เก็บไว้ต่อจากรหัสคำสั่งมาประมวลผล ดังนั้นหากในขณะที่ซีพียูกำลังทำงานมีการเปลี่ยนแปลง ค่าในรีจีสเตอร์ PC ก็จะส่งผลให้ลำดับการทำงานของโปรแกรมเปลี่ยนไป การทำงานของรีจีสเตอร์ PC ในภาวะปกติ (เพิ่มครั้งละ 1) และในภาวะที่ถูกเปลี่ยนค่าคำสั่งมีดังแสดงในรูป



ภาพที่ 4.8 แสดงการทำงานของรีจีสเตอร์ PC

คำสั่งที่ใช้ในการควบคุมลำดับการทำงานโปรแกรมสามารถแยกออกเป็น 2 ประเภท ได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- คำสั่งที่ใช้ในการควบคุมลำดับการทำงานโปรแกรมแบบไม่มีเงื่อนไข (unconditional jumps)
- คำสั่งที่ใช้ในการควบคุมลำดับการทำงานโปรแกรมแบบมีเงื่อนไข (conditional jumps)

4.4.1 กลุ่มคำสั่งควบคุมโปรแกรมแบบไม่มีเงื่อนไข

กลุ่มคำสั่งที่ใช้ในการควบคุมลำดับการทำงานโปรแกรมแบบ ไม่มีเงื่อนไขจะบังคับให้โปรแกรมย้ายการทำงานไปยังที่อื่นในหน่วยความจำซึ่งไม่ใช่ตำแหน่งถัดไปของคำสั่งนี้ทันที คำสั่งในกลุ่มนี้ประกอบด้วย 5 คำสั่งคือ

- JMP addr
- CALL addr
- RET
- RETI
- NOP

คำสั่งทั้ง 5 คำสั่งล้วนแต่ใช้คำสั่งการทำงานของโปรแกรมอย่างไม่มีเงื่อนไข แต่จะมีวิธีการใช้งานที่แตกต่างกันออกไป ดังนี้

- JMP addr คำสั่งนี้ไม่มีในรหัสคำสั่ง ใน MCS-51 โดยตรง แต่ใช้สำหรับเขียนโปรแกรมภาษาแอสแซมบลีโดยใช้คำสั่ง JMP อื่นๆ 3 คำสั่ง คือ
- LJMP addr
- SJMP addr
- AJMP addr

ในการใช้งานจริงๆ เราสามารถเขียนโปรแกรมแอสแซมบลีโดยใช้คำสั่ง JMP addr ได้โดยนำเอาโปรแกรมภาษาแอสแซมบลี ที่เขียนขึ้นมาไปทำการแอสแซมเบลอร์ (แปลคำสั่งภาษาแอสแซมบลีไปเป็นรหัสคำสั่งของ MCS-51 โดยตรง) ตัวโปรแกรมแอสแซมเบลอร์จะทำการแปลรหัส

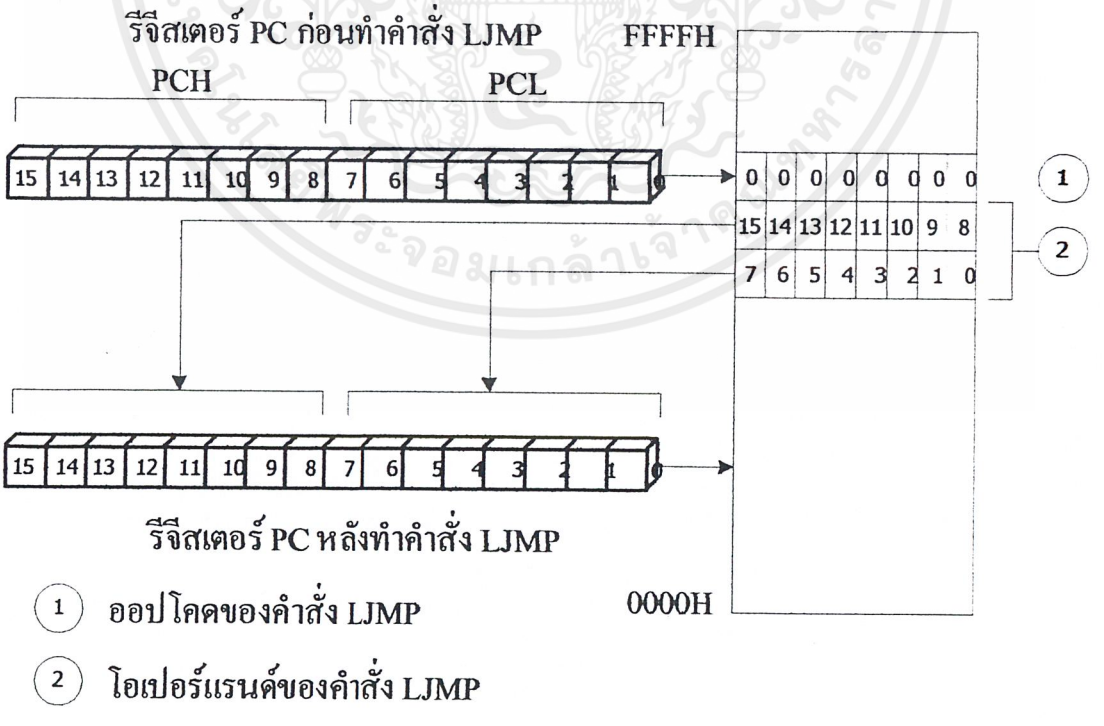
นีโมนิกของคำสั่ง JMP ไปเป็นรหัสคำสั่งของคำสั่ง JMP ทั้งสามให้เอง ทั้งนี้โดยพิจารณาตามความเหมาะสมของแต่ละคำสั่ง แต่ผู้เขียนโปรแกรมก็สามารถระบุคำสั่งทั้งสามลงไปในโปรแกรมเองได้เช่นกัน ดังนั้น หากผู้เขียนโปรแกรมไม่สนใจรายละเอียดการทำงานของคำสั่งทั้งสามนี้ก็สามารถใช้คำสั่ง JMP addr ได้โดยไม่ต้องทราบถึงการทำงานของคำสั่งทั้งสาม แต่เพื่อให้การศึกษาเกี่ยวกับ MCS-51 สมบูรณ์ เราจึงควรทราบรายละเอียดการทำงานของคำสั่งทั้งสามให้หมด เพื่อให้สามารถใช้งาน MCS-51 ได้อย่างมีประสิทธิภาพยิ่งขึ้น

หมายเหตุ โปรแกรมแอสแซมเบลอร์บางโปรแกรมไม่สามารถแทนคำสั่ง JMP addr ได้โดยตรง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำสั่ง JMP ทั้งสามที่เราจะศึกษาต่อไปนี้ล้วนแต่มีจุดประสงค์ในการใช้งานเหมือนกัน คือ บังคับให้โปรแกรมทำงานที่ส่วนอื่นของหน่วยความจำ เหตุที่ต้องมีคำสั่งย่อยทั้งสามก็เพื่อให้การใช้งาน MCS-51มีประสิทธิภาพยิ่งขึ้น โดยคำสั่งทั้งสามจะมีขนาดของคำสั่งไม่เท่ากัน แต่ความเร็วในการทำงานจะเท่ากัน การเลือกใช้คำสั่งหนึ่งคำสั่งใดขึ้นอยู่กับระยะห่างของหน่วยความจำที่โปรแกรมจะกระโดดไปทำงานกับตำแหน่งของตัวคำสั่ง JMP เอง รูปแบบของคำสั่งรายละเอียดการทำงานของคำสั่ง JMP ทั้งสามมีดังนี้

คำสั่ง LJMP addr คำสั่งนี้มีขนาดของรหัสคำสั่ง 3 ไบต์ ตัวคำสั่งประกอบด้วยรหัสคำสั่งเอง 1 ไบต์ และโอเปอร์แรนด์อีก 2 ไบต์ โอเปอร์แรนด์ของคำสั่งซึ่งมีขนาด 16 ถึง 2 ไบต์ จะเป็นค่าของตำแหน่งหน่วยความจำที่จะบังคับให้โปรแกรมกระโดดไปทำงาน เมื่อ MCS-51 กระทำด้วยคำสั่งค่าในรีจีสเตอร์ PC จะเพิ่มขึ้นอีกหนึ่ง จากนั้น MCS-51จะแทนที่ค่าในรีจีสเตอร์ PC ด้วยค่าของโอเปอร์แรนด์ขนาด 16 บิต ดังนั้นคำสั่งต่อไปนี้ที่ MCS-51 จะทำงานคือคำสั่งซึ่งอยู่ในหน่วย-ความจำตำแหน่งที่ตรงกับค่ารีจีสเตอร์ PC ซึ่งได้เปลี่ยนค่าไปโดยโอเปอร์แรนด์ซึ่งมีขนาด 16 บิต ดังนั้นคำสั่งต่อไปที่ MCS-51จะทำงานคือคำสั่งที่อยู่ในหน่วยความจำตำแหน่งที่ตรงกับค่าของรีจีสเตอร์ PC ซึ่งได้เปลี่ยนค่าไปโดยโอเปอร์แรนด์ของคำสั่ง LJMP addr เรียบร้อยแล้ว ดังแสดงการทำงานที่ภาพ 4.8

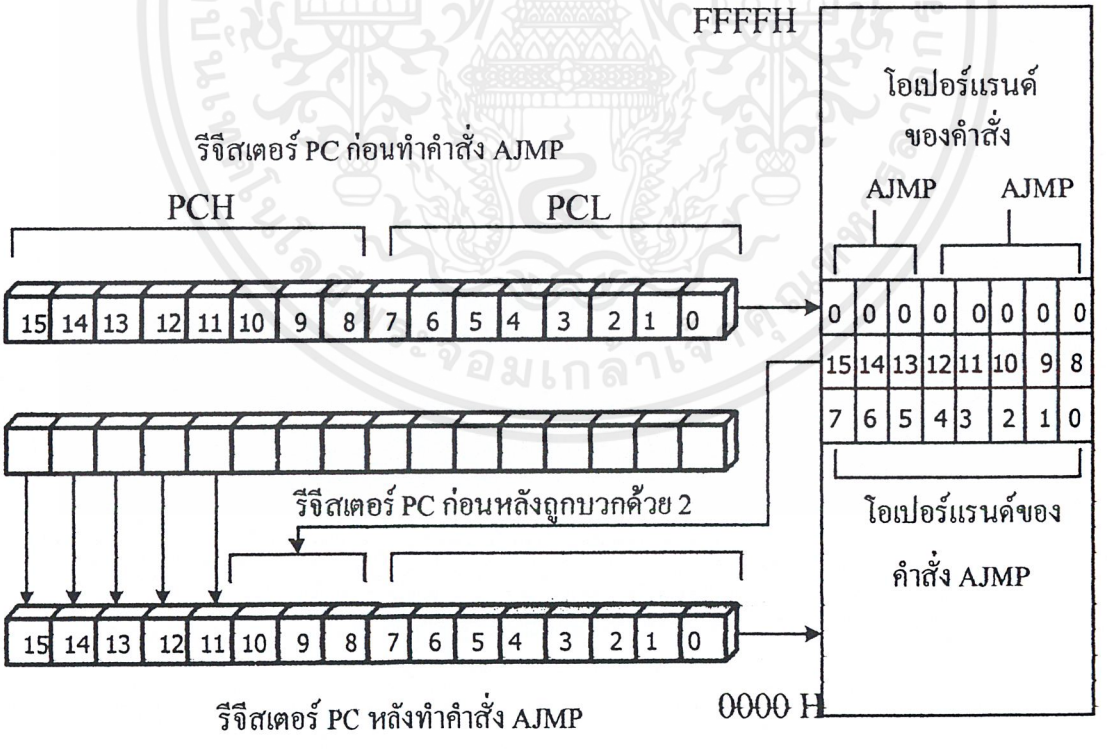


ภาพที่ 4.9 แสดงการทำงานของคำสั่ง LJMP addr

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การที่รีจีสเตอร์ PC ถูกแทนที่ใหม่ด้วยโอเปอร์แรนด์ขนาด 16 บิต ทำให้ผู้เขียนโปรแกรมสามารถกระโดดไปทำงานบริเวณใดก็ได้ในหน่วยความจำส่วนที่เก็บโปรแกรม ซึ่งมีขนาดทั้งสิ้น 64 กิโลไบต์

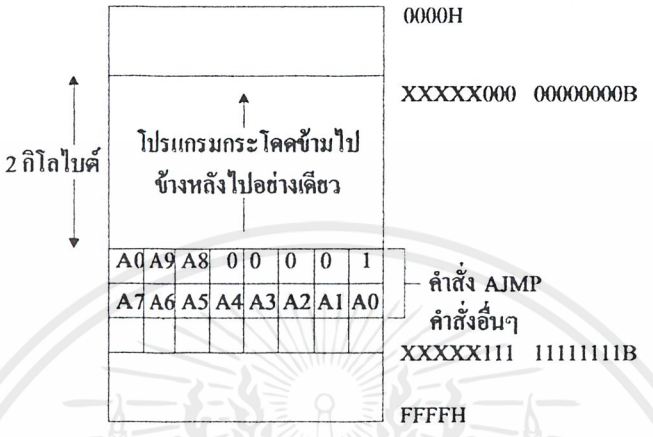
คำสั่ง AJMP addr 11 คำสั่งนี้มีขนาดของรหัสคำสั่งรวม 2 ไบต์ ตัวคำสั่งเองประกอบด้วยรหัสคำสั่งเอง 5 บิต อยู่ในไบต์แรกของคำสั่ง (บิต 0 ถึง บิต 4) ส่วนบิตที่เหลือ 3 บิต ของไบต์แรกจะแทนกด้วยค่าตำแหน่งหน่วยความจำบิต 8 ถึง 10 บิต ของตำแหน่งหน่วยความจำที่ต้องการย้ายไปทำงานภายหลังของคำสั่งนี้ส่วนไบต์ที่สองของคำสั่งจะแทนด้วยค่าของตำแหน่งหน่วยความจำบิต 0 ถึง บิต 7 ของหน่วยความจำที่ต้องการย้ายไปทำงานจากคำสั่งนี้ ดังนั้นโอเปอร์แรนด์ของคำสั่ง AJMP จะเป็นค่าของตำแหน่งของหน่วยความจำจำนวน 11 บิต (บิต 0 ถึง บิต 10) โดยแยกกันอยู่ใน 2 ไบต์ ของคำสั่งเมื่อ MCS-51 กระทำคำสั่ง AJMP ค่าในรีจีสเตอร์ PCจะเพิ่มขึ้นอีกสองจากนั้น MCS-51 จะแทนค่าที่ของรีจีสเตอร์ PCด้วยค่าตำแหน่งในหน่วยความจำเพียง 11 บิต (A0-A10) ส่วนตำแหน่งในหน่วยความจำบิต 11 ถึง 15 จะยังคงมีค่าเดิมดังแสดงในรูปที่จึงทำให้คำสั่ง AJMP สามารถบังคับให้โปรแกรมไปทำงานที่ส่วนอื่นของหน่วยความจำภายในขอบเขต 2 กิโลไบต์ (2 11= 2048) ซึ่งอยู่ติดกับตำแหน่งของคำสั่งดังแสดงให้เห็นในรูปที่ 4.10



รูปที่ 4.10 แสดงคำสั่ง AJMP addr

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากภาพที่ 4.10 จะแสดงให้เห็นตำแหน่งของคำสั่ง AJMP ซึ่งมีผลกระทบต่อขอบเขตการกระโดดข้ามไปทำงานที่ตำแหน่งอื่นในความจำภายในบริเวณ 2 กิโลไบต์ติดกับคำสั่ง



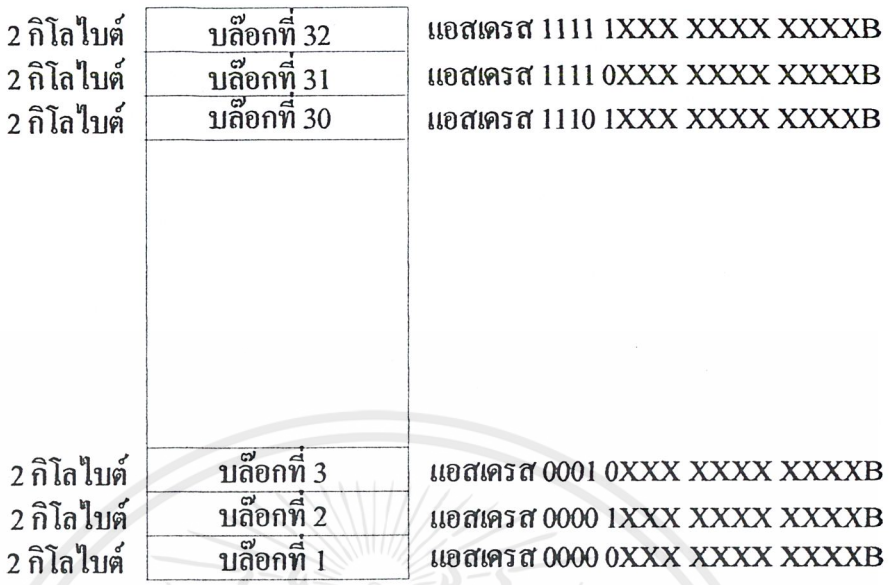
ภาพที่ 4.11 ขอบเขตในการกระโดดไปทำงานของคำสั่ง AJMP



ภาพที่4.11 (ต่อ) ขอบเขตในการกระโดดไปทำงานของคำสั่ง AJMP

หากเราแบ่งหน่วยความจำส่วนที่เก็บโปรแกรม ของ MCS-51 ออกเป็นบล็อกๆ โดยพิจารณาจาก 5 บิตบนของรีจิสเตอร์ PC จะทำให้แบ่งพื้นที่ของหน่วยความจำส่วนนี้ออกได้เป็น 32 บล็อกด้วยกัน (= 256) โดยเริ่มจากบล็อกแรกซึ่งมีค่า 5 บิตบนเป็น 0 หมด จนถึงบล็อกสุดท้ายที่มีค่า 5 บิต บนนี้เป็น 1 หมด เพื่อความเข้าใจให้ดูรูปประกอบ ภาพที่ 4.12

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



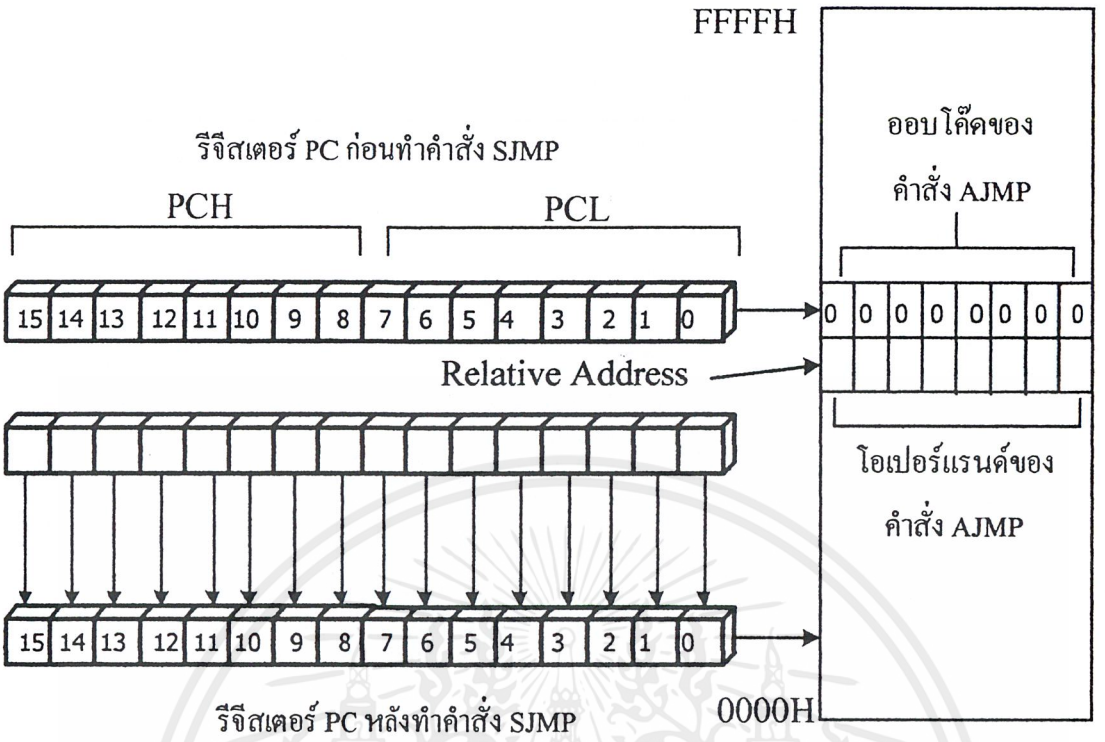
ภาพที่ 4.12 แสดงส่วนประกอบบล็อกหน่วยความจำทั้ง 32 บล็อก

คำสั่ง AJMP ในโปรแกรมซึ่งอยู่ในบล็อกใดบล็อกหนึ่งความจำเปลี่ยนค่าในรีจีสเตอร์ PC เพียง 11 บิต โดย 5 บิตที่เหลือมีค่าเหมือนเดิม นั่นคือ สามารถบังคับให้โปรแกรมกระโดดไปทำงานภายในบริเวณบล็อกใดบล็อกหนึ่งของหน่วยความจำเท่านั้น โดยหากคำสั่ง AJMP อยู่ในตำแหน่งที่ 1 หรือ 2 ในภาพที่ 4.11 โปรแกรมจะสามารถกระโดดไปข้างหน้าหรือถ้อยหลังได้เพียงอย่างเดียว โดยมีช่วงการกระโดดไปข้างหน้าและกระโดดถอยหลังไม่เท่ากัน ขึ้นอยู่กับตำแหน่งของคำสั่งในบล็อกนั้นๆ

คำสั่ง SJMP rel address คำสั่งนี้มีขนาด 2 ไบต์ เช่นเดียวกับคำสั่ง AJMP addr 11 แต่ในคำสั่งนี้รหัสของคำสั่งจะมีขนาด 8 บิตอยู่ในไบต์แรกของคำสั่ง ส่วนโอเปอร์แรนด์มีขนาด 8 บิต เช่นกันแต่ว่าอยู่ในไบต์ที่สองตามหลังไบต์ที่เป็นรหัสคำสั่ง ค่าของโอเปอร์แรนด์ขนาด 8 บิต มีชื่อเรียกว่า relative address ซึ่งเป็นเลขฐาน 2 ที่มีบิต 7 บิต เป็นตัวบอกเครื่องหมาย (มีค่าในช่วง –128 ถึง +127)

หลังจากกระทำคำสั่ง AJMPค่าในรีจีสเตอร์ PC จะเพิ่มขึ้นอีก หนึ่งจากนั้น MCS-51จะบวกค่าในรีจีสเตอร์ PCด้วยค่าโอเปอร์แรนด์ (relative address) แบบคิดเครื่องหมายแล้วนำผลลัพธ์ที่ได้กลับไปแทนที่รีจีสเตอร์ PC อีกครั้งดังนั้นรีจีสเตอร์PC จะมีค่าใหม่ที่อยู่ในช่วง 128 ไบต์ ซึ่งอาจจะมากกว่าค่าในรีจีสเตอร์ PC เดิม (อยู่ในช่วง 256 ติดกับตำแหน่งของคำสั่ง) ดังแสดงในรูปที่ 4.13

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 4.13 แสดงการทำงานของคำสั่ง SJMP

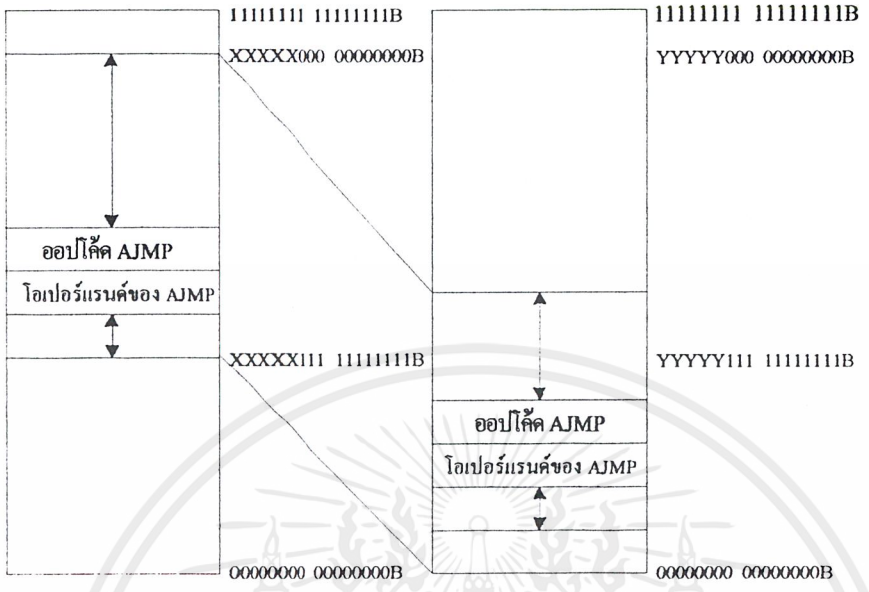
คำสั่ง SJMP นี้สามารถบังคับให้โปรแกรมกระโดดไปทำงานที่ส่วนอื่นในหน่วยความจำซึ่งมีขอบเขตในการกระโดดข้ามไปทำงานน้อยกว่าคำสั่ง LJMP และ AJMP ที่กล่าวมา แต่คำสั่งนี้มีข้อดีเหนือกว่าคำสั่งอื่นๆตรงที่ผู้เขียนโปรแกรมสามารถย้ายโปรแกรมที่มีคำสั่ง SJMP นี้ไปไว้ที่ใดนั้น ตำแหน่งของหน่วยความจำที่จะกระโดดไปมา จากการคำนวณโดยนำค่าในรีจีสเตอร์ PC รวมกับค่าของโอเปอร์แรนด์ในคำสั่ง

คำสั่ง AJMP ถึงแม้จะสามารถบังคับให้โปรแกรมกระโดดไปทำงานที่อื่นได้ในบริเวณที่กว้างกว่าคำสั่ง SJMP แต่ในการย้ายโปรแกรมไปยังส่วนอื่นของหน่วยความจำจะกระทำได้ยากกว่า เพราะต้องแปลโปรแกรมแอสแซมบลีให้เป็นรหัสคำสั่งที่เป็นภาษาเครื่องใหม่ทั้งหมด ถึงอย่างไรก็ตามโปรแกรมที่มีคำสั่ง ALMP ก็ยังสามารถย้ายไปไว้ที่อื่นในหน่วยความจำโดยไม่ต้องแปลโปรแกรมแอสแซมบลีใหม่ได้เช่นกัน เพียงแต่มีเงื่อนไขบางอย่างเพิ่มขึ้นมาเท่านั้น ดังจะได้กล่าวต่อไป

เนื่องจากตำแหน่ง ของคำสั่ง AJMP addr 11 มีผลต่อการกระโดดไปทำงานของโปรแกรมดังได้แสดงในรูป ซึ่งจะเห็นว่าตำแหน่งของคำสั่ง AJMP มีความสำคัญมากหากต้องการย้ายโปรแกรมที่มีคำสั่ง AJMP นี้ไปไว้ในส่วนอื่น ของหน่วยความจำจะสามารถกรทำได้โดยการต้องย้ายไปทั้งบล็อกเพื่อให้ระยะห่างระหว่างต้นบล็อกหรือท้ายบล็อก ของตัวคำสั่งเหมือนเดิมทั้งใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังแสดงให้เห็นในรูป



ภาพที่ 4.14 แสดงการย้ายโปรแกรมเป็นบล็อกเพื่อให้คำสั่งยังใช้ได้เหมือนเดิม

คำสั่ง JMP ในกลุ่มนี้ยังมีคำสั่ง JMP @ A + DPTR ซึ่งจะนำค่าในรีจีสเตอร์ A มารวมกับค่าในรีจีสเตอร์ DPTR โดยไม่คิดเครื่องหมายของค่าในรีจีสเตอร์ A จากนั้นนำผลลัพธ์ ที่ได้ไปไว้ในรีจีสเตอร์ PC จึงทำให้เราสามารถบังคับให้โปรแกรมไปทำงานบริเวณใดก็ได้ในหน่วยความจำขึ้นอยู่กับค่าในรีจีสเตอร์ DPTR และ A เท่านั้น

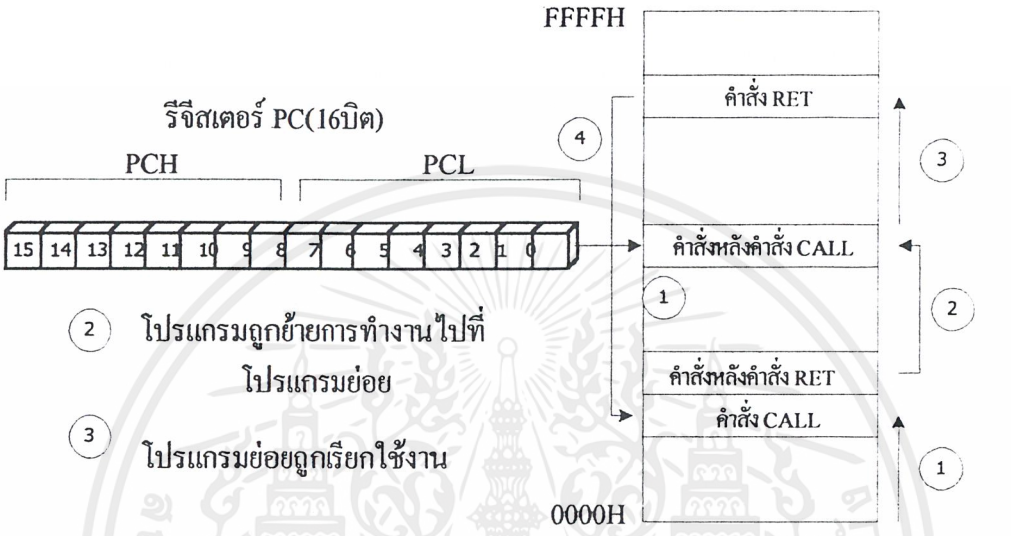
คำสั่ง CALL Addr และ RET คำสั่ง CALL Addr คำสั่งนี้มักใช้ควบคู่ไปกับคำสั่ง RET ดังนั้น จะกล่าวถึงรายละเอียดของคำสั่งทั้งสองพร้อมกันเลย

คำสั่ง CALL Addr มีไว้เพื่อควบคุมให้โปรแกรมหลักย้ายการทำงานไปที่โปรแกรมย่อย ที่อยู่บริเวณใดบริเวณหนึ่งของหน่วยความจำ และเมื่อโปรแกรมย่อยทำงานเสร็จสิ้นแล้วก็สามารถคืนการทำงานมายังโปรแกรมหลักได้ด้วยการใช้คำสั่ง RET นั่นคือ คำสั่ง CALL และ RET นี้มีไว้เพื่อย้ายการทำงานไปที่อื่นชั่วคราว ซึ่งต่างจากคำสั่ง JMP ที่จะกระโดดไปทำงานต่อในตำแหน่งอื่นของหน่วยความจำเลย การทำงานของคำสั่ง CALL และ RET แสดงดังในรูป

เนื่องจากรีจีสเตอร์ PC มีขนาด 16 บิต (2 ไบต์) ดังนั้นการทำคำสั่ง PUSH, POP โดยคำสั่ง CALL และ RET จึงต้องกระทำถึง 2 ครั้ง โดยครั้งแรกจะทำการ PUSH ค่าในรีจีสเตอร์ PC ไบต์ต่ำ (บิต 0 ถึง บิต 7) ก่อนแล้วจึง PUSH ค่าในรีจีสเตอร์ PC ไบต์ถัดไป

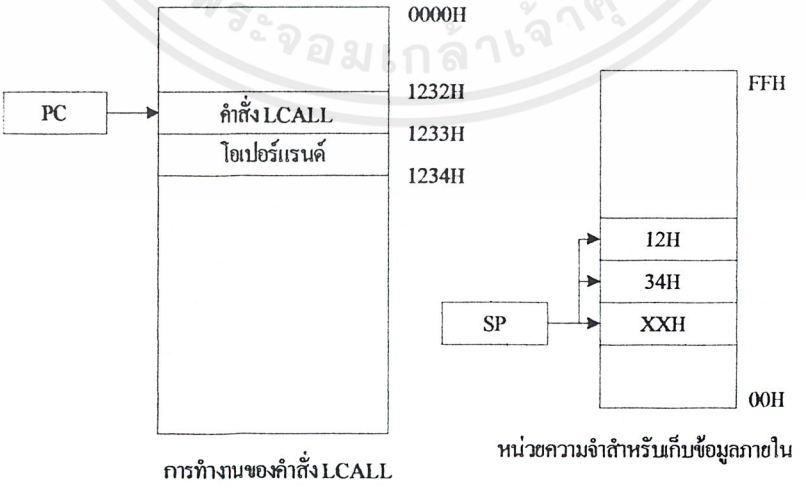
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หากในโปรแกรมย่อยไม่มีคำสั่ง RET จะทำให้ MCS-51 ไม่สามารถกลับมาทำงานต่อในโปรแกรมหลักได้ เท่ากับเป็นการบังคับให้โปรแกรมกระโดดไปทำงานที่อื่นเลย ดังนั้นโปรแกรมย่อยทุกโปรแกรมจึงต้องมีคำสั่ง RET ปิดท้ายไว้เสมอ เพื่อให้ MCS-51 ทราบว่าทำงานในโปรแกรมย่อยเสร็จแล้ว



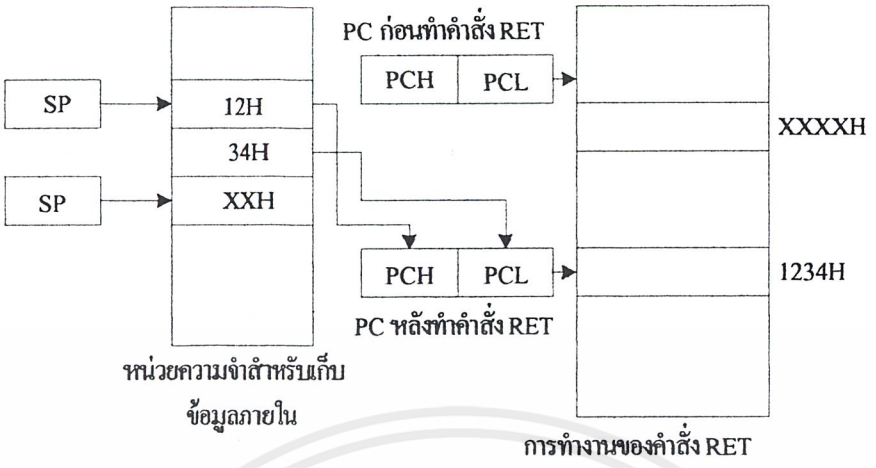
ภาพที่ 4.15 แสดงการทำงานของคำสั่ง CALL และ RET

หากในโปรแกรมย่อยไม่มีคำสั่ง RET จะทำให้ MCS-51 ไม่สามารถกลับมาทำงานต่อในโปรแกรมหลักได้ เท่ากับเป็นการบังคับให้โปรแกรมกระโดดไปทำงานที่อื่นเลย ดังนั้น โปรแกรมย่อยทุกโปรแกรมจึงต้องมีคำสั่ง RET ปิดท้ายไว้เสมอ เพื่อให้ MCS-51 ทราบว่าทำงานในโปรแกรมย่อยเสร็จแล้ว



ภาพที่ 4.16 แสดงลำดับการทำงานของคำสั่ง CALL และ RET

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 4.16 (ต่อ) แสดงลำดับการทำงานของคำสั่ง CALL และ RET

คำสั่ง CALL ใน MCS-51 มีถึง 2 คำสั่งด้วยกันคือ

- LCALL addr16
- ACALL addr11

การใช้คำสั่ง CALL addr ตัวโปรแกรมแอสแซมเบอร์ที่ดีจะทราบเองว่าสมควรแทนคำสั่ง CALL ชนิดใดที่เหมาะสม คำสั่ง CALL ทั้งสองมีรายละเอียดที่แตกต่างกันดังนี้

- LCALL addr16

คำสั่งนี้มีขนาด 3 ไบต์ ไบต์แรกเป็นรหัสของคำสั่ง ส่วนไบต์ถัดไปอีกจะเป็นค่าตำแหน่งในหน่วยความจำซึ่ง MCS-51 จะย้ายไปทำงานชั่วคราว คำสั่งนี้จะทำการ PUSH ค่าของหน่วยความจำในตำแหน่งถัดจากตัวคำสั่งไปไว้ในสแตก จากนั้นจึงแทนที่ค่าในรีจีสเตอร์ PC ด้วย ค่าของโอเปอร์แรนด์ทั้ง 16 บิต ดังแสดงการทำงานในรูป

- ACALL addr11

คำสั่งนี้มีขนาด 2 ไบต์แรกจะประกอบด้วยรหัสคำสั่งเพียง 5 บิต (10001) อยู่ในบิต 00 ถึงบิต 4 ของไบต์แรก บิตที่เหลือจะเป็นค่าตำแหน่งของหน่วยความจำที่ต้องการย้ายไปทำงานชั่งคราวเฉพาะบิต 10 ถึงบิต 8 ส่วนในไบต์ถัดไปจะเป็นค่าตำแหน่งของหน่วยความจำที่ต้องการย้ายไปทำงานชั่วคราวเฉพาะบิต 7 ถึง บิต 0 นั่นคือมีการแทนที่ค่าในรีจีสเตอร์ PC หลังกระทำคำสั่ง CALL เหมือนคำสั่ง AJMP addr11 ดังนั้นโปรแกรมย่อยจึงต้องอยู่ในขอบเขตบริเวณ 2 กิโลไบต์ ติดกับคำสั่ง ในการทำงานจะมีการกระทำคำสั่ง PUSH 2 ครั้ง เช่นเดียวกันกับคำสั่ง LCALL ทั้งนี้เพื่อให้สามารถนำค่าตำแหน่งถัดไปของคำสั่งมาไว้ในรีจีสเตอร์ PC เพื่อกลับมาทำงานยังโปรแกรมหลักต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำสั่ง NOP คำสั่งนี้ซีพียูจะไม่ทำงานใดๆ ทั้งสิ้นแต่จะปล่อยให้เวลาผ่านไป 1 แมชชีนไซเคิล พร้อมทั้งเปลี่ยนค่าในรีจีสเตอร์ PC ให้มาชี้ยังคำสั่งที่อยู่ในตำแหน่งถัดไปของคำสั่ง NOP นั่นคือเมื่อ MCS-51 ทำคำสั่งใดๆ มาจนถึงคำสั่ง NOP ก็จะข้ามไปทำงานที่คำสั่งซึ่งอยู่ถัดไปจากคำสั่ง NOP โดยปล่อยให้เวลาผ่านไป 1 แมชชีนไซเคิลก่อนกระทำคำสั่งถัดไป

4.4.2 กลุ่มคำสั่งควบคุมโปรแกรมแบบมีเงื่อนไข

ในหัวข้อที่ผ่านมา เราได้กล่าวถึงกลุ่มคำสั่งในการควบคุมลำดับการทำงาน ของโปรแกรม ซึ่งได้ในการควบคุมให้โปรแกรมย้ายการทำงาน ไปยังส่วนอื่นโดยไม่มีเงื่อนไข ซึ่งมีข้อจำกัดในการเขียนโปรแกรมอยู่บ้าง เพราะโปรแกรมที่มีแต่คำสั่งเหล่านี้ไม่สามารถตัดสินใจโดยตัวเองได้ ดังนั้น ในหัวข้อต่อไปนี้จะกล่าวถึงกลุ่มคำสั่งที่ใช้ในการควบคุมลำดับการทำงานของโปรแกรมที่มีความสามารถมากกว่าในกลุ่มที่ผ่านมา คำสั่งในกลุ่มนี้มีด้วยกันทั้งสิ้น 8 คำสั่งคือ

JZ	rel	Jump if zero
JNZ	rel	Jump if not zero
CJNE	A,direct rel	Compare and if not equal
CJNE	A,#data rel	Compare and jump it not equal
CJNE	Rn,#data rel	Compare and jump it not equal
CJNE	@Ri,#data rel	Compare and jump if not equal
DJNZ	Rn,rel	Decrement and jump if not zero
DJNZ	direct,rel	Decrement and jump if not zero

จะเห็นว่าโอเปอร์แรนด์ของคำสั่งในกลุ่มนี้เป็นค่า Relative address (ดิดเครื่องหมาย) ทั้งหมด ดังนั้น คำสั่งในกลุ่มนี้ทุกคำสั่งจะสามารถบังคับ ให้โปรแกรมทำงานบริเวณอื่นได้ ในหน่วยความจำแบบมีเงื่อนไข ขอบเขตที่ห่างจากตัวคำสั่งได้เพียง 256 ไบต์เท่านั้น รายละเอียดแต่ละคำสั่งในกลุ่มนี้มีดังนี้

- คำสั่ง JZ จะตรวจสอบเงื่อนไขเฉพาะกับรีจีสเตอร์ A เท่านั้น หากรีจีสเตอร์ A มีค่าแป็นศูนย์ คำสั่งนี้จะควบคุมให้โปรแกรมข้ามไปทำงานที่ตำแหน่งซึ่งได้จากค่าในรีจีสเตอร์ PC รวมกับค่า Relative address ของโอเปอร์แรนด์ หากแต่ค่าของรีจีสเตอร์ไม่เท่ากับศูนย์ MCS-51 จะกระทำคำสั่งซึ่งอยู่ถัดไปจากคำสั่งดังต่อไปนี้
- คำสั่ง JNZ rel จะตรวจสอบรีจีสเตอร์ A เช่นเดียวกับคำสั่ง JZ rel แต่เงื่อนไขที่ใช้เป็นตัวตัดสินจะแตกต่างกัน นั่นคือ หากรีจีสเตอร์ A มีค่าเป็นศูนย์ MCS-51 จะกระทำ
- คำสั่งถัดไป แต่ค่าในรีจีสเตอร์ A ไม่เป็นศูนย์ โปรแกรมจะถูกบังคับให้ข้ามไปทำงานที่ตำแหน่งซึ่งได้จากค่าในรีจีสเตอร์ PC รวมกับค่า Relative address ของโอเปอร์แรนด์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- คำสั่ง CJNE ทั้ง 4 คำสั่งต้องการข้อมูลหรือโอเปอร์แรนด์จำนวน 3 ตัว คำสั่งนี้ใช้ควบคุมลำดับการทำงานของโปรแกรมโดยมีเงื่อนไขคือ จะนำเอาค่าของโอเปอร์แรนด์ทั้งสองของคำสั่งมาลบกันดังนี้
- โอเปอร์แรนด์ตัวแรกมากกว่าโอเปอรีแรนด์ตัวที่สอง carry flag ถูกเคลียร์
- โอเปอร์แรนด์ตัวแรกน้อยกว่าโอเปอร์แรนด์ตัวที่สอง carry flag ถูกเซต

หากค่าของโอเปอร์แรนด์ตัวแรกมีค่าไม่เท่ากับโอเปอร์แรนด์ตัวที่สอง คำสั่งที่จะบังคับให้ MCS-51 ข้ามไปทำงานที่ตำแหน่งได้จากค่าในรีจีสเตอร์ PC รวมกับค่า Relative address ที่ได้จากโอเปอร์แรนด์ตัวที่สาม แต่หากค่าของโอเปอร์แรนด์สองตัวแรกมีค่าเท่ากับโอเปอร์แรนด์ตัวที่สอง MCS-51 จะข้ามคำสั่งนี้ไปกระทำคำสั่งถัดไปทันที

คำสั่งในการควบคุมลำดับการทำงานของโปรแกรมแบบมีเงื่อนไข ยังมีส่วนหนึ่งที่จัดไว้ในกลุ่มคำสั่งประมวลผลแบบบูลีน เหตุที่ต้องแยกคำสั่งดังนี้เพราะกลุ่มคำสั่งสำหรับการประมวลผลแบบบูลีนมีการทำงานพิเศษที่ต่างจากไมโครโปรเซสเซอร์ทั่วไป โดยคำสั่งในกลุ่มประมวลผลแบบบูลีนจะมีกลุ่มคำสั่งย่อยที่ใช้ควบคุมลำดับการทำงานของโปรแกรม โดยตรวจสอบเงื่อนไขจากข้อมูลเพียงบิตเดียว รายละเอียดของกลุ่มคำสั่ง ประมวลผลแบบบูลีนจะเป็นดังหัวข้อถัดไป

4.5 กลุ่มคำสั่งประมวลผลแบบบูลีน

กลุ่มคำสั่งประมวลผลแบบบูลีนที่จะศึกษาต่อไปนี้เป็นกลุ่มคำสั่งพิเศษ ที่มีใน MCS-51 จุดประสงค์ของคำสั่งกลุ่มนี้คือ อำนวยความสะดวกในการเขียนโปรแกรม ที่จำเป็นต้องเกี่ยวข้องกับข้อมูลระดับบิต เช่น การตรวจสอบสถานะของพอร์ตแต่ละบิต การปฏิบัติคำสั่งทางตรรกศาสตร์ระดับบิต การเคลื่อนย้ายข้อมูลระดับบิต การเซ็ตหรือเคลียร์บิต การควบคุมการทำงานของโปรแกรม โดยตัดสินเงื่อนไขจากสถานะบิต

เมื่อเราได้ศึกษาลักษณะของข้อมูลขนาด 1 บิตมาแล้ว ต่อไปจะเป็นการศึกษาคำสั่งของกลุ่มคำสั่งประมวลผลแบบบูลีน โดยแยกตามการทำงานดังนี้

4.5.1 กลุ่มคำสั่งควบคุมสถานะบิต

คำสั่งในกลุ่มนี้ใช้สำหรับควบคุมสถานะของบิตให้มีค่าตามที่เราต้องการ คำสั่งทั้งหมดในการควบคุมสถานะบิตมีด้วยกัน 6 คำสั่งคือ

```
CLR   C
CLR   bit
SETB  C
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SETB bit

CPL C

CPL bit

ตัวอักษร B ในคำสั่ง SETB มีไว้เพื่อแยกความแตกต่างระหว่างคำสั่ง SET ซึ่งใช้ในโปรแกรมแอสแซมเบอร์

ในการเขียนโปรแกรมภาษาแอสแซมบลีสำหรับ MCS-51 โอเปอร์แรนด์ของคำสั่งที่เป็นหน่วยความจำขนาด 1 บิต สามารถระบุในตัวคำสั่งได้หลายวิธีดังนี้

- ระบุค่าตำแหน่งระหว่าง 0 – 255 หรือ 00H – FFH ค่าตัวเลขที่ใช้อาจเป็นค่าตัวเลขที่ระบุในคำสั่งเลย หรือเป็นนิพจน์ทางคณิตศาสตร์ที่ให้ค่าอยู่ในช่วง 0 – 255
- ในกรณีที่ของรีจีสเตอร์ซึ่งแต่ละบิต SCON.3 คือบิตที่ 3 ของรีจีสเตอร์ SCON
- ในกรณีของรีจีสเตอร์ ซึ่งแต่ละบิตมีชื่อเฉพาะอยู่แล้ว โปรแกรมที่แอสแซมเบอร์จะทราบเองว่าบิตที่อ้างด้วยชื่อนี้ หมายถึงบิตอะไร เช่น บิต TI, RI หมายถึงบิตข้อมูลในรีจีสเตอร์ ใช้งานเฉพาะ SCON

นอกจากจะสามารถระบุตำแหน่งของหน่วยความจำขนาด 1 บิต ที่จะมากระทำการในคำสั่งด้วยวิธีการดังกล่าวข้างต้นแล้ว เรายังสามารถแทน ชื่อของตำแหน่งหน่วยความจำได้ด้วยชุดตัวอักษรได้แต่จะต้องใช้คำสั่ง SET หรือ EQU ในโปรแกรม แอสแซมเบอร์ เช่น

```
SET    Label : 20H หรือ
Label  EQU   20H
```

ในโปรแกรมแอสแซมเบอร์บางโปรแกรมอาจไม่ได้ใช้คำสั่ง SET แต่ใช้คำสั่ง EQU แทน ดังนั้นผู้เขียนโปรแกรมจะต้องทราบเองว่า โปรแกรมแอสแซมเบอร์ที่ใช้งานอยู่ มีคำสั่งในการใช้งานอย่างไร

ตัวอย่างของการเข้าถึงข้อมูลขนาด 1 บิต จะแสดงให้เห็นทุกวิธี โดยสมมติว่าเราต้องการเคลียร์บิตที่ 5 ของรีจีสเตอร์ PSW วิ่งมีชื่อว่า บิต F0 ดังนั้นคำสั่งต่อไปนี้จะให้ผลเช่นเดียวกัน

```
CLR   0D5H      ;Absolute Addressing
CLR   F0        ;Pre-Definded Assembler Symbol
CLR   PSW.5     ;Dot Operator
```

หรืออาจแทนบิตนี้ด้วยชื่อใหม่ว่า บิต STATUS ดังนี้

```
STATUS BIT PSW.5   ;ตั้งชื่อบิต PSW.5 เป็น Status โดยใช้ชื่อคำสั่ง BIT หรือ
STATUS EQUP SW.5   ;ตั้งชื่อบิต PSW.5 เป็น Status โดยใช้ชื่อคำสั่ง EQU
```

เอกสารนี้เป็นเอกสาร[illegible]ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
CLR STATOR          ;เคลียร์บิตชื่อ Status ซึ่งเป็นบิต PSW.5
```

4.5.2 กลุ่มคำสั่งเคลื่อนย้ายข้อมูล คำสั่งกลุ่มนี้เป็นกลุ่มที่สองในกลุ่มคำสั่งประมวลผลแบบบูลีน จุดประสงค์ของคำสั่งมีไว้เพื่อเคลื่อนย้ายข้อมูลขนาด 1 บิต จากที่หนึ่งไปยังอีกที่หนึ่ง ณ โดยค่าข้อมูลที่ต้นทางยังมีค่าคงเดิม กลุ่มคำสั่งที่เคลื่อนย้ายข้อมูลในการประมวลผลแบบบูลีนต่างจากกลุ่มคำสั่งประมวลผลอื่นๆ ที่เราได้ศึกษาผ่านมาแล้ว เพราะว่าข้อมูลที่เคลื่อนย้ายมีขนาดเพียง 1 บิต เท่านั้น ข้อจำกัดของคำสั่งอยู่ที่การเคลื่อนย้ายที่ต้องใช้บิต C ในรีจีสเตอร์ PSW ทุกครั้ง นั่นคือ เราไม่สามารถย้ายข้อมูลระหว่าง 2 บิตใดๆ โดยตรงได้ หากต้องการเคลื่อนย้ายบิตระหว่างหน่วยความจำ จำเป็นต้องกระทำผ่านบิต C เสมอ คำสั่งในกลุ่มนี้มีเพียงสองคำสั่งเท่านั้นคือ

```
MOV   C,bit
MOV   bit,C
```

4.5.3 กลุ่มคำสั่งทางตรรกศาสตร์ คำสั่งทางตรรกศาสตร์ในการประมวลผลแบบบูลีนจะกระทำการกับข้อมูลขนาด 1 บิต คำสั่งในกลุ่มนี้ทั้งหมดมีเพียง 4 คำสั่งเท่านั้นคือ

```
ANL   C,bit
ANL   C,/bit
ORL   C.bit
ORL   C,/bit
```

จะเห็นว่ามีเพียงคำสั่ง ANL และ ORL เท่านั้นส่วนคำสั่ง XRL จะไม่มีกลุ่มนี้ซึ่งแตกต่างจากกลุ่มคำสั่งทางตรรกศาสตร์โดยทั่งไป ที่ได้ศึกษาผ่านมาแล้ว

คำสั่ง ANL และ ORL ที่แสดงข้างต้นมีด้วยกันอย่างละสองคำสั่ง สิ่งที่แตกต่างกันในคำสั่งทั้งสอง คือ การมีหรือไม่มีเครื่องหมาย "/" นำหน้าโอเปอร์แรนด์ตัวที่สอง โอเปอร์แรนด์ตัวที่สองของคำสั่งจะเป็นหน่วยความจำขนาด 1 บิต ตำแหน่งใดก็ได้ภายใน MCS-51 ความหมายของเครื่องหมาย "/" คือ ระบุให้ซีพียูกระทำทางตรรกศาสตร์กับค่าคอมพลีเมนต์ของโอเปอร์แรนด์ตัวที่สอง โดยหลังกระทำคำส่งค่าข้อมูลในโอเปอร์แรนด์ตัวหลังจะไม่ถูกเปลี่ยนแปลงไป การที่ผู้ใช้สามารถระบุให้กระทำการกับข้อมูลที่อยู่ในรูปคอมพลีเมนต์นี้ ทำให้การเขียนโปรแกรมมีความสะดวกและกะทัดรัดมากขึ้น เนื่องจากไม่ต้องเปลี่ยนข้อมูลเป็นตรงกันข้ามก่อนแล้วจึงกระทำการคำสั่งทางคณิตศาสตร์

4.5.4 กลุ่มคำสั่งควบคุมโปรแกรมที่ขึ้นกับสถานะของบิต คำสั่งในกลุ่มนี้ใช้ในการควบคุมลำดับการทำงานของโปรแกรมโดยตรวจสองเงื่อนไขของสถานะของบิต ที่ระบุโดยโอเปอร์แรนด์ในการทำงานคำสั่งกลุ่มนี้ MCS-51 จะทำการตรวจสอบสถานะของบิตก่อน หากเงื่อนไขถูกต้อง

เอกสารนี้มีลิขสิทธิ์ ห้ามนำไปใช้ในการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตามคำสั่ง ค่าในรีจีสเตอร์ PC จะถูกเปลี่ยนแปลงไปเพื่อย้ายการทำงาน ไปยังส่วนอื่นของโปรแกรม คำสั่งในกลุ่มนี้มีด้วยกันสองประเภท คือ คำสั่งที่ควบคุมลำดับของโปรแกรม โดยตรวจสอบสถานะของบิต C ในรีจีสเตอร์ PSW และคำสั่งที่ตรวจสอบสถานะของบิตใดๆ ในหน่วยความจำขนาด 1 บิตที่กล่าวมาแล้ว คำสั่งทั้งหมดในกลุ่มนี้มีดังนี้

JC rel

JNC rel

JB bit,rel

JNB bit,rel

JBC bit,rel

คำสั่งทั้งหมดดังกล่าวใช้วิธีการเพิ่มค่าหรือลดค่ารีจีสเตอร์ PC โดยการบวกค่าในโอเปอร์แรนด์ ซึ่งเป็นเลขฐานสองที่คิดเครื่องหมายเข้ากับรีจีสเตอร์ PC เหมือนคำสั่ง SJMP ที่ได้ศึกษาผ่านมาแล้ว

คำสั่งตรวจสอบสถานะของบิต C และหน่วยความจำขนาด 1 บิต ได้อำนวยความสะดวกให้ผู้เขียนโปรแกรม โดยสามารถใช้เงื่อนไขทั้งสองอย่าง (ตรวจสอบเงื่อนไขได้ทั้งเป็น ศูนย์และหนึ่ง) ส่วนคำสั่งสุดท้ายจะตรวจสอบหน่วยความจำขนาด 1 บิต หากเงื่อนไขเป็นจริง (บิตที่อ้างโดยโอเปอร์แรนด์เป็น 1) ค่าในรีจีสเตอร์ PC จะถูกเปลี่ยนไปโดยค่า Relative address และหน่วยความจำขนาด 1 บิตตำแหน่งที่จะถูกเคลียร์ด้วย

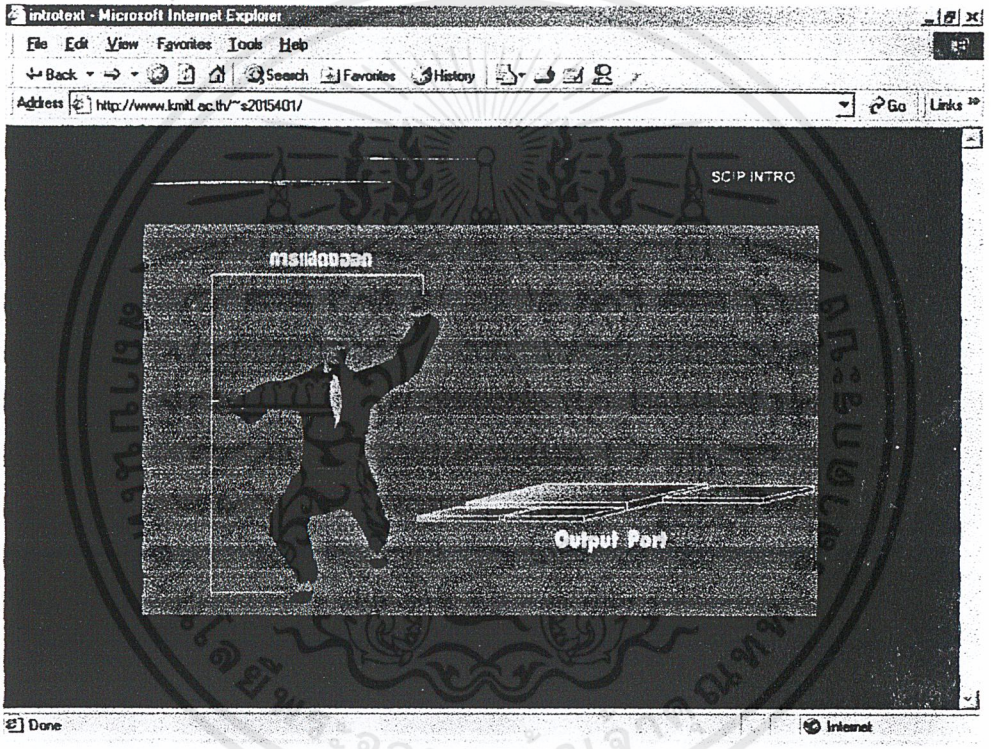
4.6 คำสั่งที่มีผลต่อการทำงานของกลุ่มคำสั่งประมวณผลแบบบูลีน

เนื่องจากการทำงานของกลุ่มคำสั่งประมวลผลแบบบูลีนจะมีการใช้บิต C ในรีจีสเตอร์ PSW เป็นเสมือนแอกคิวมูเลเตอร์ (Acccumulator) ขนาด 1 บิต ดังนั้น คำสั่งอื่นที่มีผลต่อบิตนี้จะส่งผลให้การทำงานของกลุ่มคำสั่งประมวลผล แบบบูลีน ทำงานผิดพลาดได้ เช่น กลุ่มคำสั่งทางคณิตศาสตร์ตรรกศาสตร์ ซึ่งมีการเซตหรือเคลียร์บิตหลังจากการกระทำคำสั่ง เพื่อป้องกันข้อผิดพลาดจากคำสั่งอื่นๆ ที่มีผลกระทบต่อบิต C เราจึงควรที่จะเคลียร์บิต C ก่อนกระทำคำสั่งทุกครั้ง หรือหากต้องการใช้ค่าในบิต C อีกครั้งก็ควรใช้คำสั่ง CALL เพื่อเก็บค่ารีจีสเตอร์ PSW ไว้ในสแตกเสียก่อน คำสั่งทั้งหมดที่มีผลกระทบต่อบิต C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
การทดลองใช้งาน

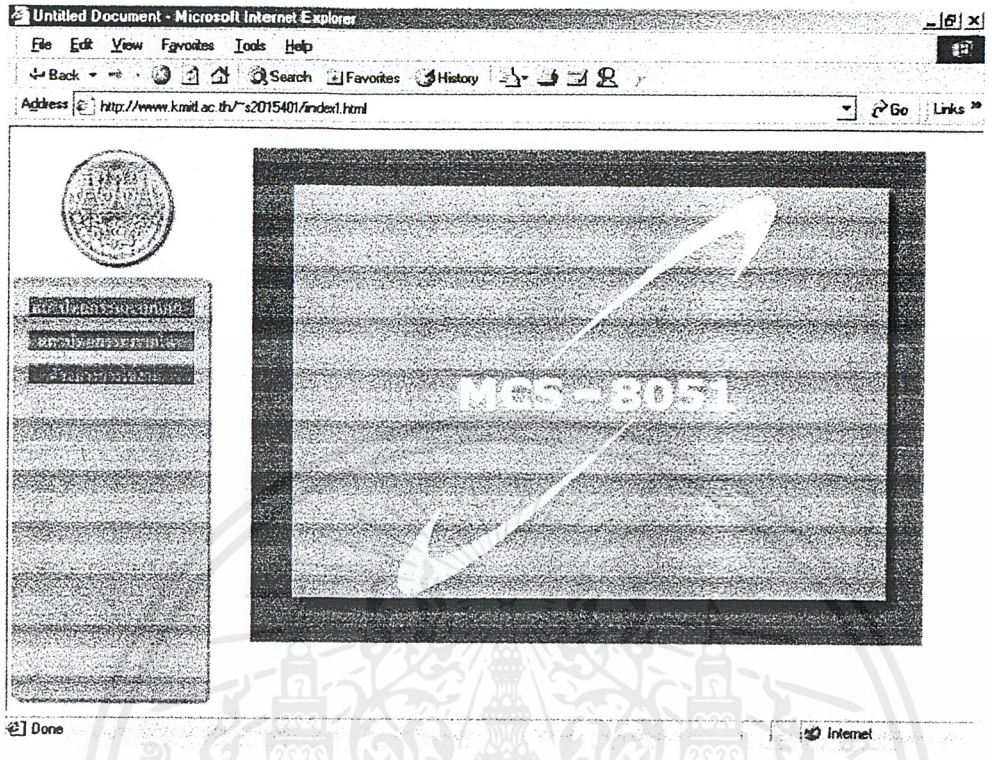
การทำงานของเวปไซต์เริ่มต้นโดยการโหลด Introduction ซึ่งเวลาที่ใช้โหลดจะขึ้นอยู่กับความเร็วในการรับส่งข้อมูลของสายสัญญาณ หน้า Introduction จะใช้ดึงดูดความสนใจของผู้เข้ามาชมเว็บไซต์ ประกอบด้วยเสียงและภาพเคลื่อนไหวเปรียบเทียบส่วนต่าง ๆ ของไมโครคอนโทรลเลอร์ กับอวัยวะของมนุษย์



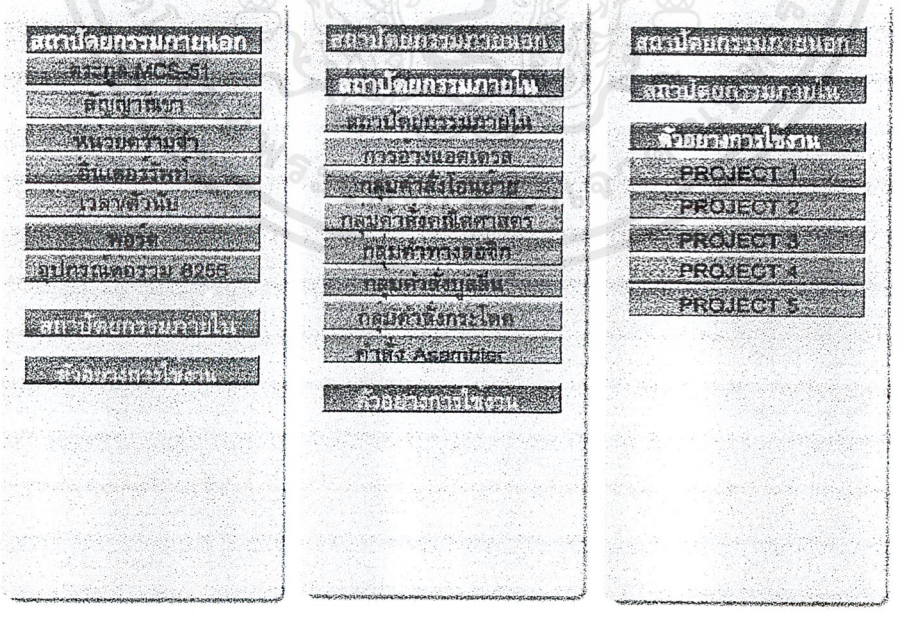
ภาพที่ 5.1 แสดงหน้า Introduction ของเว็บไซต์

เมื่อ Introduction แสดงจบ จะเข้าสู่หน้าโฮมเพจของเว็บไซต์ ซึ่งเป็นหน้าหลักในการเข้าถึงข้อมูลทั้งหมดในเว็บไซต์ ในหน้าโฮมเพจจะประกอบด้วยเมนู และพื้นที่สำหรับแสดงข้อมูล รวมถึงสัญลักษณ์ประจำเว็บไซต์ ส่วนของเมนูจะแสดงหัวข้อย่อยทีละส่วน เมื่อคลิกที่หัวข้อใหญ่จะปรากฏหัวข้อย่อยที่เลือก ส่วนหัวข้อย่อยอื่นจะหายไป เพื่อความสะดวกและรวดเร็วในการค้นหาข้อมูล จึงจำเป็นต้องแสดงเมนูแบบที่ใช้ในพื้นที่จำกัด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 5.2 แสดงหน้าโฮมเพจของเว็บไซต์



ภาพที่ 5.3 แสดงลักษณะของเมนูย่อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1 สถาปัตยกรรมภายนอก

ตัวเลือกในเมนูนี้ประกอบด้วย

ตระกูล MCS-51

สัญญาณขา

หน่วยความจำ

อินเตอร์รัพท์

เวลา/ตัวนับ

พอร์ต

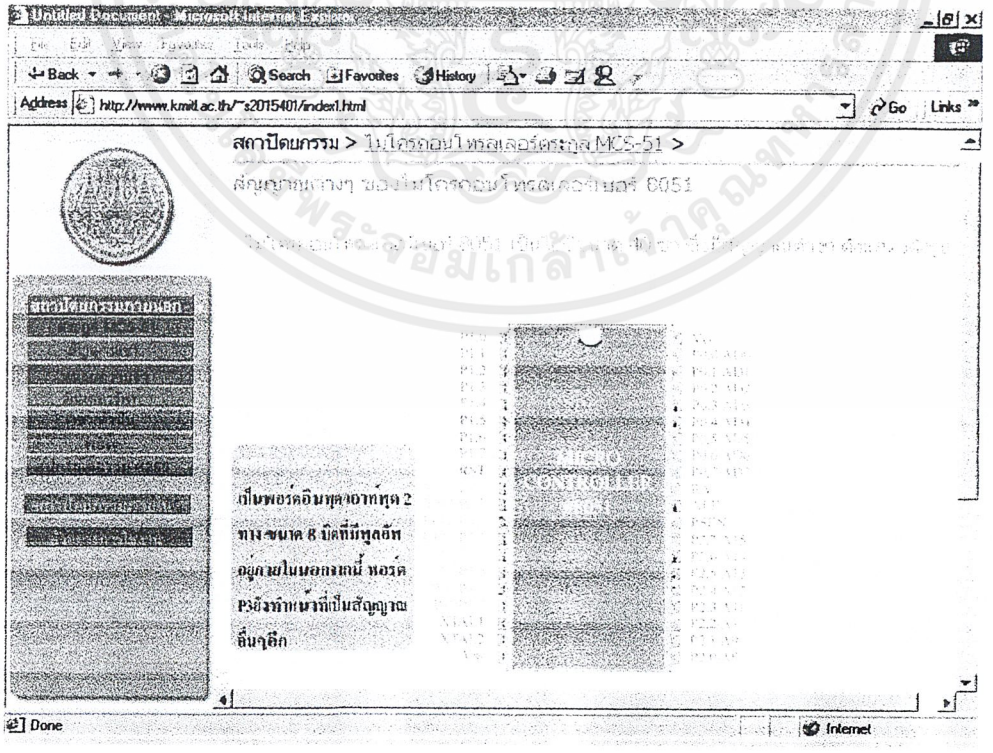
อุปกรณ์ต่อร่วม 8255

5.1.1 ตระกูล MCS-51

ตัวเลือกนี้กล่าวถึงรายละเอียดความเป็นมา และเปรียบเทียบคุณสมบัติต่าง ๆ ของไมโครคอนโทรลเลอร์ตระกูล MCS-51

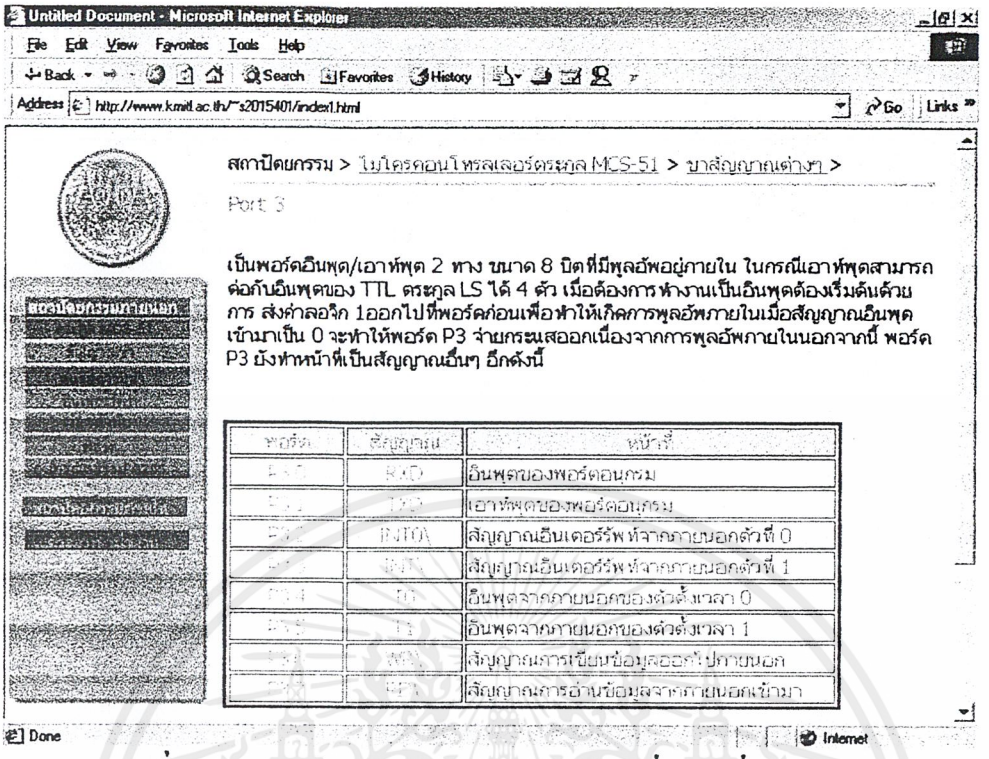
5.1.2 สัญญาณขา

ตัวเลือกนี้แสดงสัญญาณขาต่าง ๆของไมโครคอนโทรลเลอร์ เบอร์ 8051 โดยมีรูปชิปไอซีมาให้ เมื่อคลิกที่ขาของไอซีจะเข้าไปแสดงรายละเอียดของสัญญาณขาที่คลิกนั้น ๆ



ภาพที่ 5.4 แสดงหน้าสัญญาณขาเมื่อนำเมาท์มาวางที่ขาของไอซี

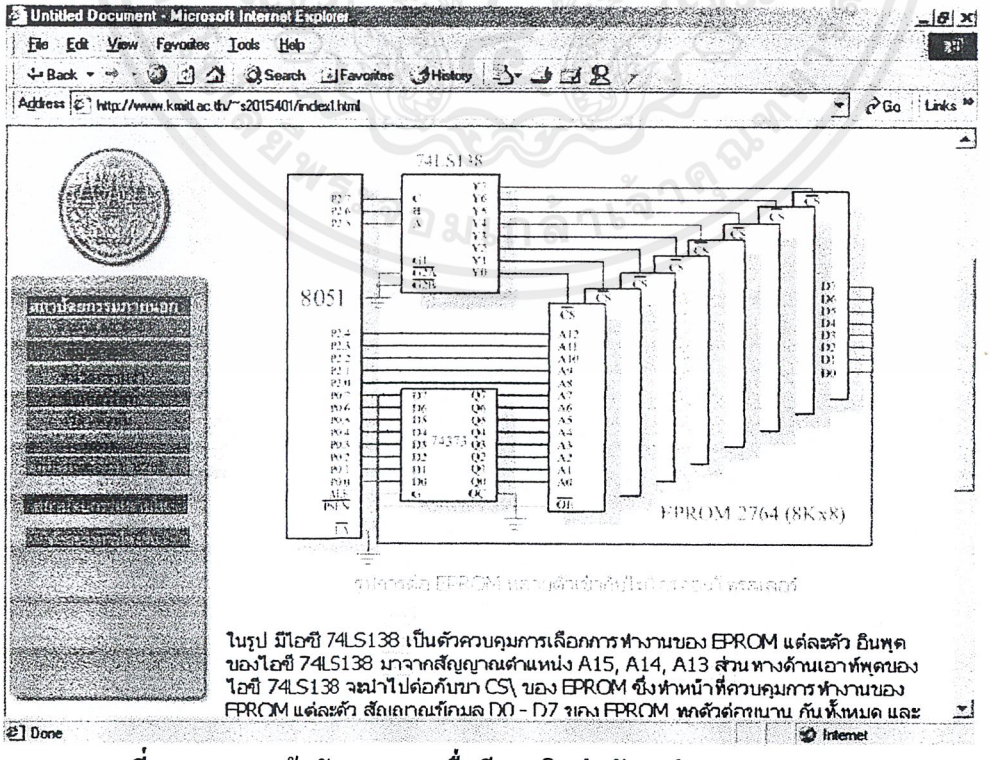
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 5.5 แสดงหน้ารายละเอียดของสัญญาณขาเมื่อคลิกที่ขาของไอซี

5.1.3 หน่วยความจำ

ตัวเลือกนี้แสดงรายละเอียดของหน่วยความจำของไมโครคอนโทรลเลอร์เบอร์ 8051 รวมถึงภาพเคลื่อนไหวแสดงการติดต่อหน่วยความจำภายนอกแบบต่าง ๆ



ภาพที่ 5.6 แสดงหน้าสัญญาณขาเมื่อมีการติดต่อกับหน่วยความจำภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.4 อินเตอร์รัพท์

ตัวเลือกนี้แสดงรายละเอียดของการทำอินเตอร์รัพท์แบบต่าง ๆ

5.1.5 เวลา/ตัวนับ

ตัวเลือกนี้แสดงรายละเอียดของตัวตั้งเวลาและตัวนับ(Timer/Counter)

5.1.6 พอร์ต

ตัวเลือกนี้แสดงรายละเอียดของพอร์ตอนุกรม และพอร์ตขนาน

5.1.7 อุปกรณ์ต่อร่วม 8255

ตัวเลือกนี้แสดงการใช้ไอซีเบอร์ 8255 ต่อร่วมเพื่อขยายพอร์ตให้มากขึ้น

5.2 สถาปัตยกรรมภายใน

ตัวเลือกในเมนูนี้ประกอบด้วย

สถาปัตยกรรมภายใน

การอ้างแอดเดรส

กลุ่มคำสั่งโอนย้าย

กลุ่มคำสั่งคณิตศาสตร์

กลุ่มคำสั่งลอจิก

กลุ่มคำสั่งบูลลีน

กลุ่มคำสั่งกระโดด

คำสั่งเสริมใน Assembler

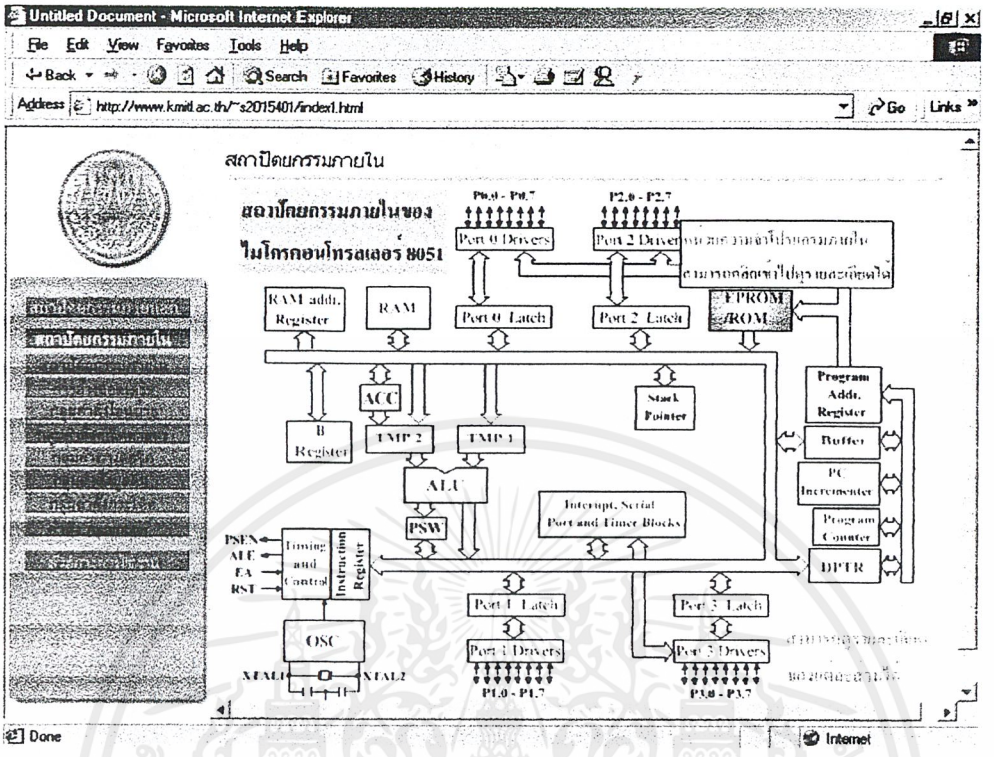
5.2.1 สถาปัตยกรรมภายใน

ตัวเลือกนี้จะแสดงรายละเอียดสถาปัตยกรรมภายในของ ไมโครคอนโทรลเลอร์เบอร์ 8051 โดยมีรูปสถาปัตยกรรมภายใน และเมื่อต้องการทราบรายละเอียดก็สามารถคลิกที่ส่วนต่าง ๆ ของรูปได้ทันที

5.2.2 การอ้างแอดเดรส

ตัวเลือกนี้จะแสดงระเอียดของการอ้างแอดเดรสแบบต่าง ๆ โดยจะมีภาพเคลื่อนไหวประกอบ เพื่อให้สามารถเข้าในได้ง่าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 5.7 แสดงหน้ารายละเอียดสถาปัตยกรรมภายใน

5.2.3 กลุ่มคำสั่งโอนย้าย

ตัวเลือกนี้จะแสดงการทำงานของคำสั่งในกลุ่มการโอนย้ายข้อมูล โดยจะมีตารางคุณสมบัติของคำสั่ง และเมื่อคลิกที่คำสั่งจะมีการอธิบายโครงสร้างและการทำงานของคำสั่ง และมีตัวอย่างคำสั่ง เป็นภาพเคลื่อนไหวประกอบความเข้าใจ

5.2.4 กลุ่มคำสั่งคณิตศาสตร์

ตัวเลือกนี้จะแสดงการทำงานของคำสั่งในกลุ่มการกระทำทางคณิตศาสตร์

5.2.5 กลุ่มคำสั่งลอจิก

ตัวเลือกนี้จะแสดงการทำงานของคำสั่งในกลุ่มการกระทำทางลอจิก

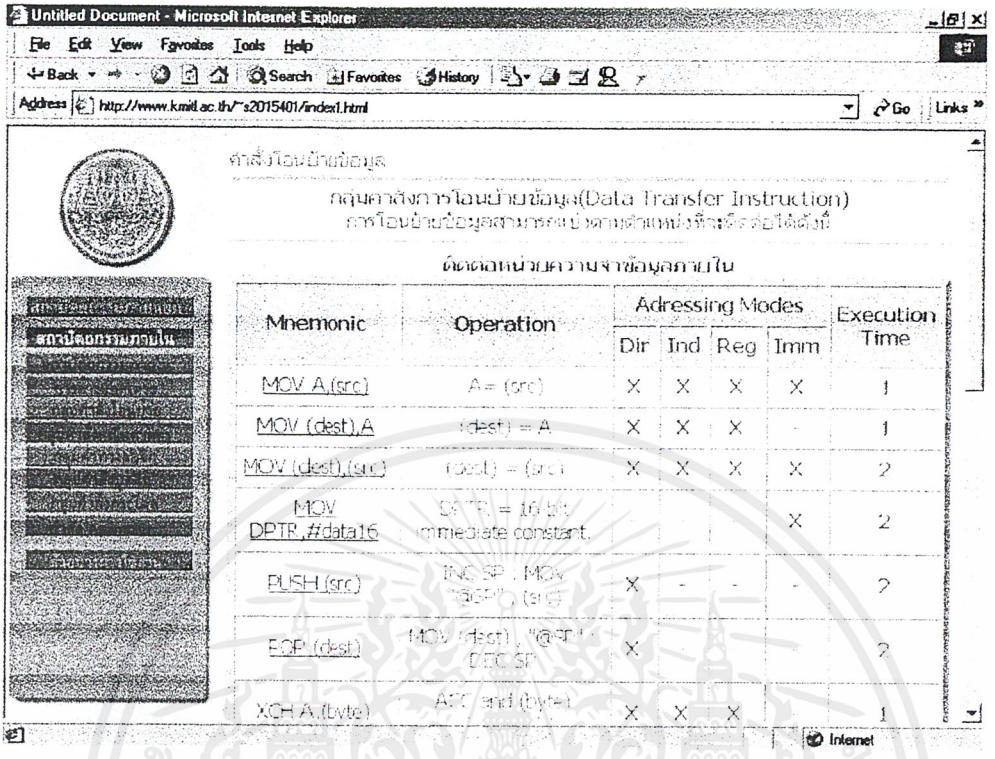
5.2.6 กลุ่มคำสั่งบูลลีน

ตัวเลือกนี้จะแสดงการทำงานของคำสั่งในกลุ่มการกระทำทางบูลลีน

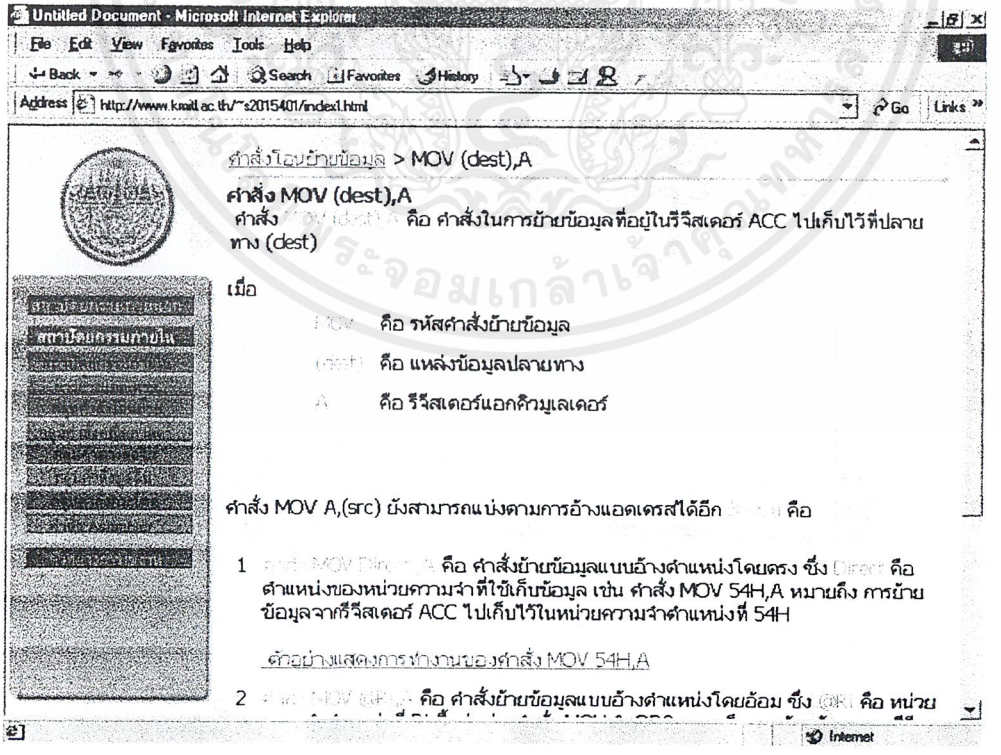
5.2.7 กลุ่มคำสั่งกระโดด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวเลือกนี้จะอธิบายการทำงานของคำสั่งกระโดด

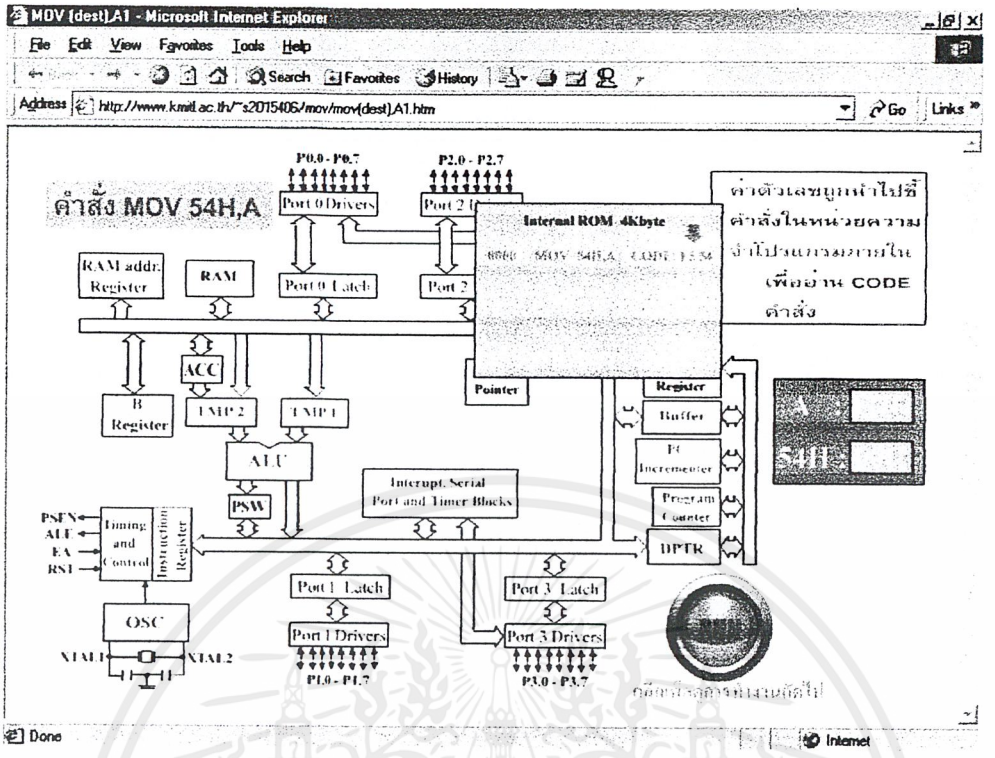


ภาพที่ 5.8 แสดงหน้าตารางคุณลักษณะของกลุ่มคำสั่งต่าง ๆ



ภาพที่ 5.9 แสดงหน้ารายละเอียดของคำสั่งที่ถูกเลือกจากตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 5.10 แสดงหน้าแสดงการทำงานของคำสั่ง

5.2.8 คำสั่งเสริมใน Assembler

ตัวเลือกนี้แสดงรายละเอียดของคำสั่งเสริม หรือคำสั่งเทียมที่ใช้ประกอบในการเขียนโปรแกรม เพื่อให้ Assembler ทำการแปลโปรแกรมได้อย่างถูกต้อง

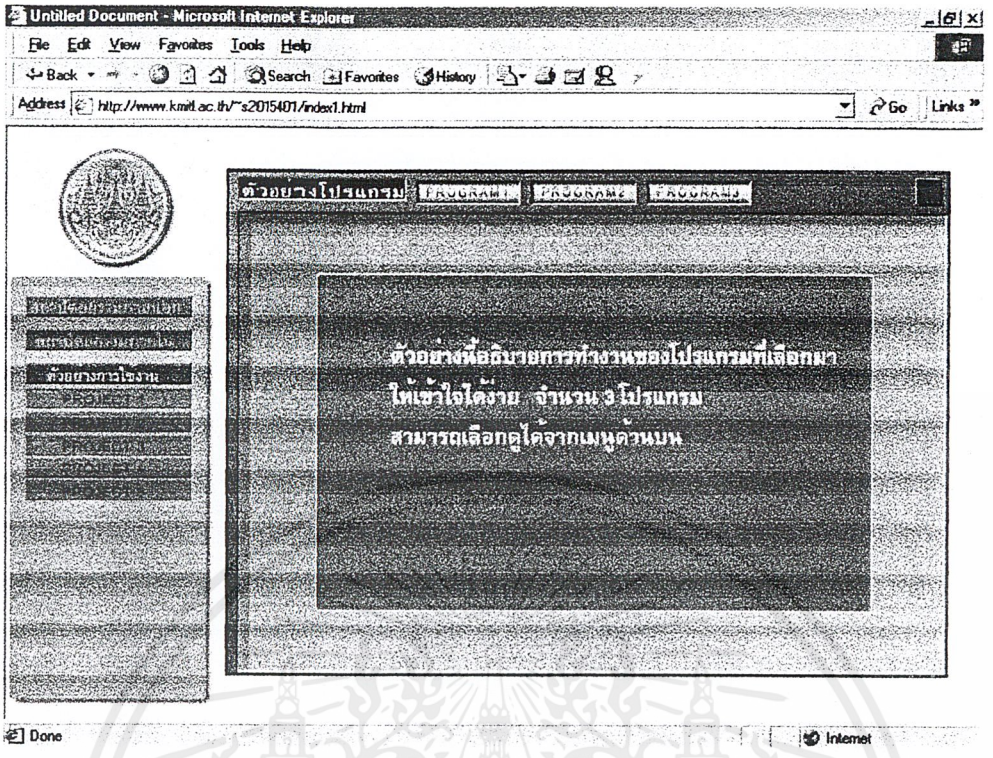
5.3 ตัวอย่างการใช้งาน

หัวข้อนี้เป็นส่วนอธิบายการประยุกต์ใช้งานไมโครคอนโทรลเลอร์เบอร์ 8051 เพื่อควบคุมอุปกรณ์ภายนอกต่าง ๆ โดยแบ่งออกเป็น Project ต่าง ๆ ตั้งแต่ Project 1 ถึง Project 5

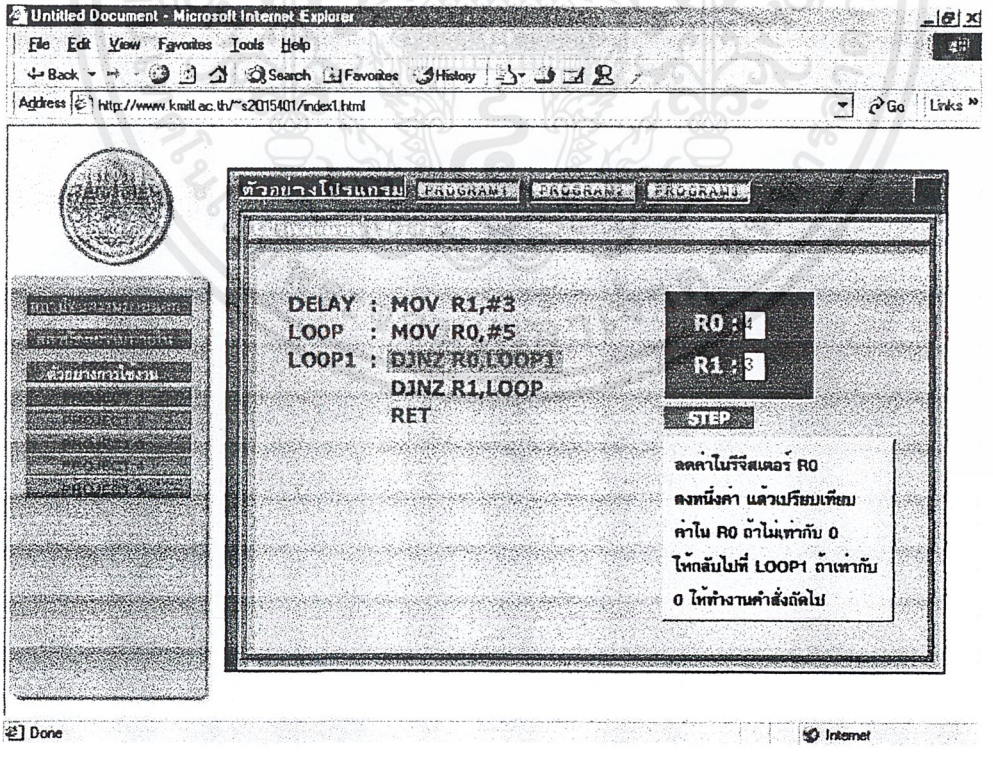
5.3.1 PROJECT 1

ตัวเลือกนี้จะอธิบายการทำงานของโปรแกรมอย่างง่าย 3 โปรแกรม ได้แก่ โปรแกรมหน่วงเวลา โปรแกรมดึงค่าข้อมูลในตารางหน่วยความจำ และโปรแกรมเรียกโปรแกรมย่อย ซึ่งแต่ละโปรแกรมจะมีส่วนอธิบาย และแสดงข้อมูลในรีจิสเตอร์ที่เกี่ยวข้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 5.11 แสดงหน้าอธิบายตัวอย่างโปรแกรม



ภาพที่ 5.12 แสดงหน้าอธิบายการทำงานของโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

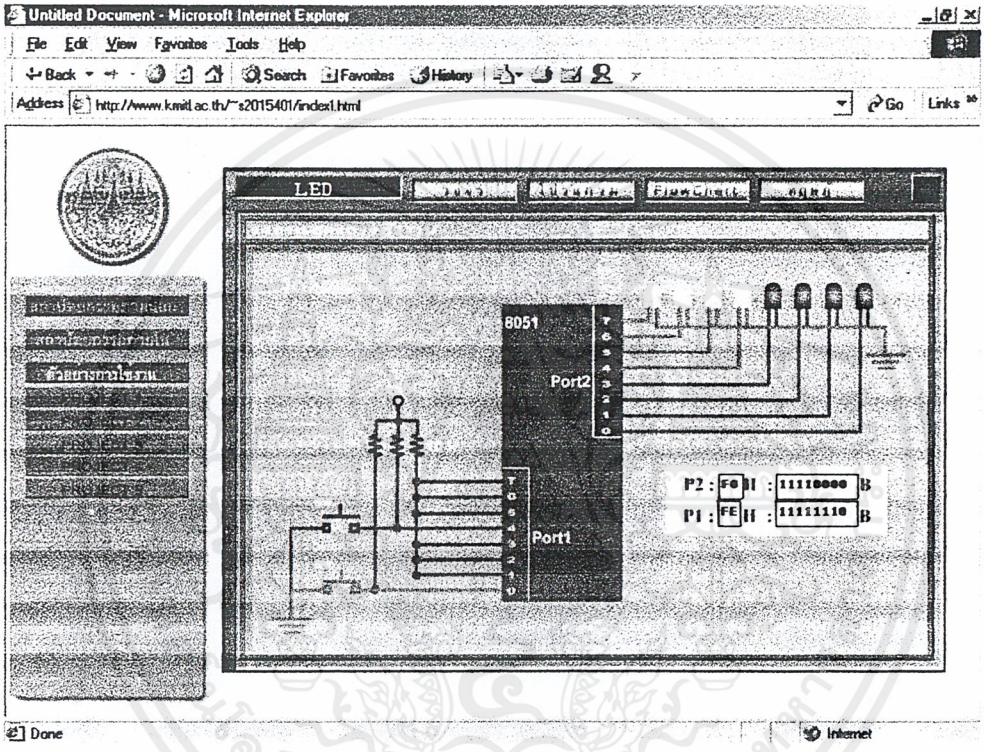
5.3.2 PROJECT 2

ตัวเลือกนี้จะแสดงการประยุกต์ใช้งานไมโครคอนโทรลเลอร์เบอร์ 8051 ควบคุมหลอด LED โดยในรายละเอียดจะประกอบด้วยส่วนต่าง ๆ ดังนี้ คือ

ส่วนของวงจร แสดงการกระพริบของหลอด LED ตามการกดสวิทช์

ส่วนของโปรแกรมที่อธิบายความหมายในโปรแกรมแต่ละส่วน

ส่วนของ Flowchart อธิบายการทำงานพร้อมทั้งเชื่อมโยงถึงโปรแกรมในส่วนที่เกี่ยวข้องกัน



ภาพที่ 5.13 แสดงหน้าวงจรการต่อใช้งานควบคุมหลอด LED

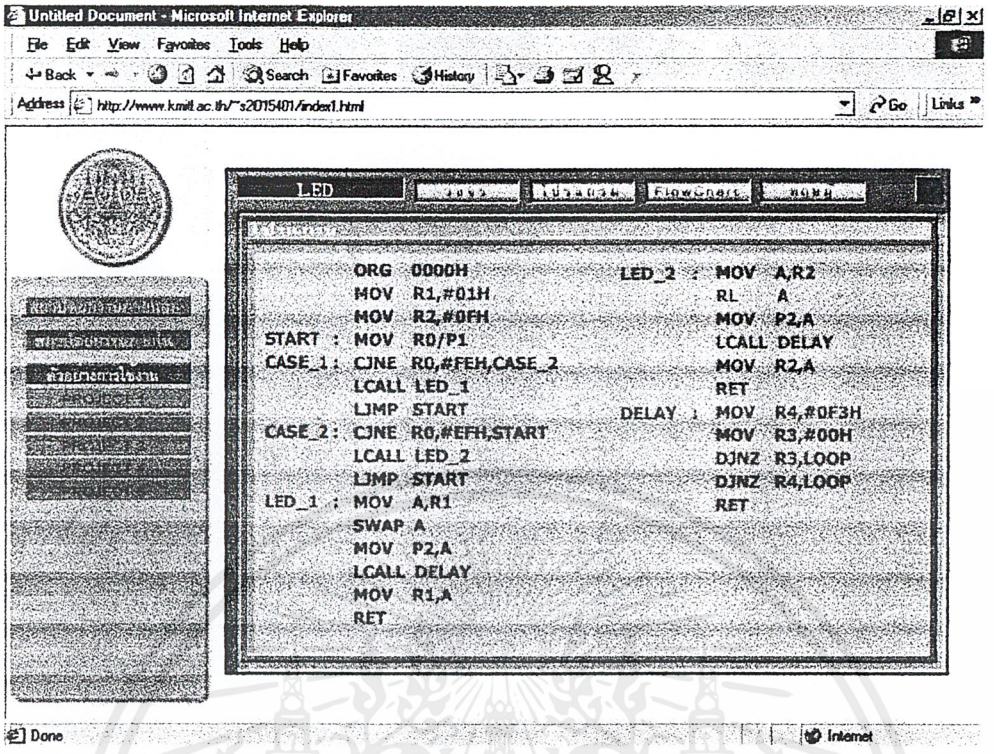
5.3.3 PROJECT 3

ตัวเลือกนี้จะอธิบายการประยุกต์ใช้งานไมโครคอนโทรลเลอร์เบอร์ 8051 ในการสแกน สวิทช์ แบบเมทริกซ์ ซึ่งจะประกอบด้วยวงจรการต่อใช้งาน โปรแกรมที่ใช้ควบคุม แสดง Flowchart และมีทฤษฎีที่เกี่ยวข้องประกอบความเข้าใจ

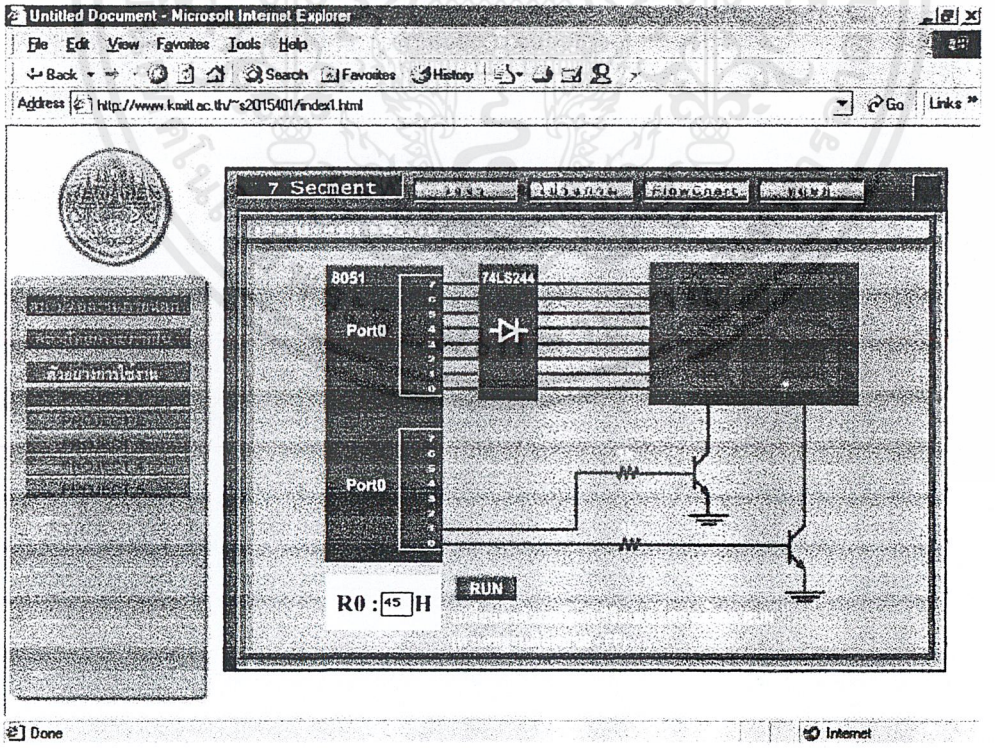
5.3.4 PROJECT 4

ตัวเลือกนี้จะอธิบายการประยุกต์ใช้งานไมโครคอนโทรลเลอร์เบอร์ 8051ควบคุมตัวเลขเจ็ดส่วน โดยการป้อนค่าตัวเลขลงในรีจีสเตอร์ และทดลองรันโปรแกรมเพื่อดูผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 5.14 แสดงหน้าโปรแกรมที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์

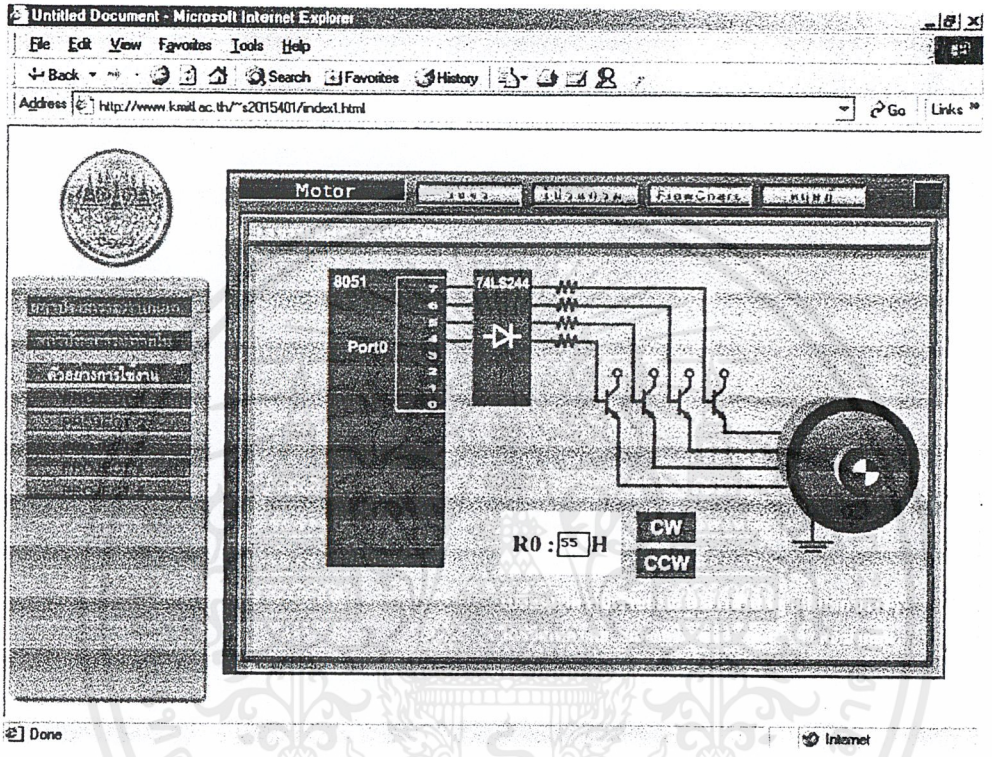


ภาพที่ 5.15 แสดงหน้าวงจรการต่อใช้งานแสดงผลตัวเลขเจ็ดส่วน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3.5 PROJECT 5

ตัวเลือกนี้จะอธิบายการประยุกต์ใช้งานไมโครคอนโทรลเลอร์เบอร์ 8051 ควบคุมสเตปปิงมอเตอร์ โดยจะมีการป้อนค่าข้อมูลในรีจีสเตอร์เพื่อ ควบคุมความเร็วในการหมุนของมอเตอร์



ภาพที่ 5.16 แสดงหน้าวงจรการต่อใช้งานควบคุมสเตปปิงมอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

1. ปรเมษฐ์ ประนยานันท์ และ ปิยพงศ์ เผ่าวณิช ," คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51 " , บริษัท ซีเอ็ดยูเคชั่น จำกัด , 380 หน้า , 2536
2. ผศ. สมยศ จุณณปิยะ , " การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51 " , คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 303 หน้า , 2541
3. แผนกหนังสือพิเศษด้านอิเล็กทรอนิกส์ , "ไมโครโปรเซสเซอร์ 1" , บริษัท ซีเอ็ดยูเคชั่น จำกัด , 219 หน้า , 2538
4. ไกรวุฒิ โรจน์ประเสริฐสุด , "ไมโครโปรเซสเซอร์ 2" , บริษัท ซีเอ็ดยูเคชั่น จำกัด , 496 หน้า , 2542
5. อุดม จีนประดับ , " ไมโครคอนโทรลเลอร์ MCS-51 " , วิทยาลัยเทคโนโลยีอุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ , 256 หน้า , 2541
6. พันจันทร์ ธนวัฒนเสถียร และ ธารา วงศ์สิริสิน, " คำภีร์สร้าง Web Page ", บริษัท ซัคเซสมีเดีย จำกัด , 652 หน้า , 2544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้