โปรแกรมบอร์ดจำลองการทำงานไมโครคอนโทรลเลอร์ตระกูล MCS-51 บนเครือข่ายอินเตอร์เน็ต

MCS-51 BOARD EMULATOR ON WEB

นายธนานนท์ ฟักจีน

นายณรงค์ฤทธิ์ กวางแก้ว

นายเด่น กากแก้ว

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์

สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีสุรนารี

ปีการศึกษา 2546

โปรแกรมบอร์ดจำลองการทำงานไมโครคอนโทรลเลอร์ตระกูล MCS-51 บนเครือข่ายอินเตอร์เน็ต

MCS-51 BOARD EMULATOR ON WEB

นายธนานนท์ ฟักจีน
นายณรงค์ฤทธิ์ กวางแก้ว
นายเด่น กากแก้ว

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์
สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2546

โครงงาน	บอร์ดจำลองการทำงานไมโครคอนโทรลเลอร์ตระกูล MCS-51 บนเครือข่ายอินเตอร์เน็ต
ผู้ดำเนินงาน	1. นายณรงค์ฤทธิ์ กวางแก้ว 2. นายเด่น กากแก้ว 3. นายธนานนท์ ฟักจีน
อาจารย์ที่ปรึกษา	อาจารย์ ดร. รังสรรค์ ทองทา
อาจารย์ที่ปรึกษาร่วม	อาจารย์พีระพงษ์ อุฑารสกุล
สาขาวิชา	วิศวกรรมโทรคมนาคม
ภาคการศึกษา	1/2546

บทคัดย่อ

ในปัจจุบันที่โลกได้มีการพัฒนาทางด้านวิทยาศาสตร์และเทคโนโลยีอย่างต่อเนื่องและรวดเร็วไมโครคอนโทรลเลอร์เป็นเทคโนโลยีหนึ่งที่เข้ามามีบทบาทต่อการพัฒนาวิทยาศาสตร์และเทคโนโลยีในงานด้านการควบคุมอุปกรณ์อิเล็กทรอนิกส์ต่างๆ โดยการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ในการควบคุมอุปกรณ์อิเล็กทรอนิกส์ต่างๆ โดยภาษาในการเขียนโปรแกรมและไมโครคอนโทรลเลอร์นั้นมีแพร่หลายแต่ปัญหาที่เกิดขึ้นคือการเรียนรู้การใช้งานนั้นถูกจำกัดโดยบอร์ดทดลองที่จะใช้ในการศึกษาและการทดสอบการทำงานของโปรแกรมที่เขียนขึ้น โครงงานบอร์ดจำลองไมโครคอนโทรลเลอร์ตระกูล MCS-51 พัฒนาขึ้นเพื่อจำลองการทำงานของบอร์ดทดลองจริงที่ใช้ในการศึกษาทดลองซึ่งเอื้ออำนวยในการศึกษาและการทดลองจากบอร์ดทดลองจริงโดยบอร์ดจำลองสามารถแสดงผลการทำงานของอุปกรณ์และข้อมูลของไมโครคอนโทรลเลอร์ได้จริงตามโปรแกรมที่เขียนบนบอร์ดทดลองและสามารถใช้งานได้ง่ายโดยทางโปรแกรมบราวเซอร์ที่สนับสนุนการทำงานของภาษาจาวาที่ใช้เป็นภาษาโปรแกรมในการสร้างบอร์ดทดลองขึ้น(Microsoft Visual J++) จากการพัฒนาให้บอร์ดจำลองสามารถใช้งานได้จริงผ่านเว็บเพจนั้นทำให้สามารถลดกระบวนการทางด้านฮาร์ดแวร์ในบอร์ดทดลองจริงได้

กิตติกรรมประกาศ

ในการจัดทำโครงงานบอร์ดจำลองการทำงานไมโครคอนโทรลเลอร์ตระกูล MCS-51 บนเครือข่ายอินเตอร์เนต สามารถเสร็จสมบูรณ์ได้เนื่องด้วยความกรุณาของบุคคลหลายท่านที่คอยให้ความช่วยเหลือและคอยให้คำปรึกษา รวมทั้งข้อเสนอแนะที่เป็นประโยชน์ต่อโครงงานทางคณะผู้จัดทำใคร่ขอแสดงความขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่านซึ่งบุคคลเหล่านั้นประกอบด้วย

- อาจารย์ ดร. รังสรรค์ ทองทา อาจารย์ที่ปรึกษาโครงงานผู้ที่เปิดโอกาสให้คณะผู้จัดทำได้เรียนรู้การทำงานในโครงงานนี้และเป็นผู้ประสิทธิ์ประสาทวิชาความรู้รวมทั้งคำปรึกษาแนะนำอันเป็นประโยชน์ยิ่งเกี่ยวกับโครงงาน
- อาจารย์ พีระพงษ์ อุฑารสกุล อาจารย์ที่ปรึกษาโครงงานร่วมผู้ที่เปิดโอกาสให้คณะผู้จัดทำได้เรียนรู้การทำงานในโครงงานนี้และเป็นผู้ประสิทธิ์ประสาทวิชาความรู้รวมทั้งคำปรึกษาแนะนำอันเป็นประโยชน์ยิ่งเกี่ยวกับโครงงาน
- อาจารย์ประโยชน์ คำสวัสดิ์ ผู้ที่สนับสนุนโครงงานให้คณะผู้จัดทำได้มีโอกาสในการทำโครงงานนี้ และสละเวลาอันมีค่ามาเป็นคณะกรรมในการสอบโครงงาน
- คณาจารย์ในทุกท่านที่เกี่ยวข้องในการให้ความรู้แก่คณะผู้จัดทำได้ที่ได้นำความรู้นั้นมาใช้ประโยชน์ในการพัฒนาโครงงาน
- เพื่อนๆ วิศวกรรมโทรคมนาคมทุกคนสำหรับความช่วยเหลือที่ดีทุกด้านตลอดจนกำลังใจที่มอบให้คณะผู้จัดทำตลอดมา

สุดท้ายผู้จัดทำขอกราบขอบคุณพระคุณบิดามารดาของคณะผู้จัดทำผู้ให้โอกาสทางการศึกษาและคอยสนับสนุนรวมทั้งกำลังใจที่คอยมอบตลอดมาอย่างหาที่เปรียบมิได้

นาย ธนานนท์ ฟักจีน
นาย ณรงค์ฤทธิ์ กวางแก้ว
นาย เด่น กากแล้ว

สารบัญ

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

สำหรับการศึกษาและเรียนรู้ในการใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51 โดยปกติแล้วนั้นจะเป็นการศึกษาโดยการทดลองเขียนโปรแกรมและนำโปรแกรมที่เขียนขึ้นนั้นไปทำการทดสอบบนบอร์ดทดลองซึ่งในปัจจุบันผู้ที่ต้องการจะศึกษาเรียนรู้การใช้งานไมโครคอนโทรลเลอร์ตระกูลMCS-51นั้น ประสบปัญหาในเรื่องของความต้องการในการใช้บอร์ดทดลองไมโครคอนโทรลเลอร์ตระกูล MCS-51เนื่องจากเรื่องของค่าใช้จ่ายในการจัดหาและความเสียหายที่เกิดจากการทดสอบโปรแกรมที่อาจเกิดขึ้นได้กับอุปกรณ์อิเล็กทรอนิกส์ที่นำมาทำการทดลอง

บอร์ดทดลองไมโครคอนโทรลเลอร์ตระกูลMCS-51จะเป็นตัวกลางในการแสดงผลออกมาให้ผู้ที่ทำการศึกษาได้เห็นว่าโปรแกรมที่ได้ทำการเขียนขึ้นนั้นได้ทำงานอย่างไรซึ่งการทำงานกับบอร์ดทดลองและอุปกรณ์อิเล็กทรอนิกส์โดยตรงนั้นอาจจะเกิดความผิดพลาดขึ้นได้ทั้งตัวบอร์ดทดลองเองและอุปกรณ์อิเล็กทรอนิกส์ดังนั้นโครงงานนี้จึงจะทำการสร้างบอร์ดทดลองจำลองขึ้นเพื่อเป็นการลดปัญหาเรื่องค่าใช้จ่ายทั้งในส่วนการจัดหาบอร์ดทดลองจริงๆและลดความเสียหายของอุปกรณ์อิเล็กทรอนิกส์ที่จะเกิดขึ้นจากการทดลอง

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อเรียนรู้ถึงโครงสร้างและหลักการทำงานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 และสามารถนำไปใช้งานได้

1.2.2 เพื่อเรียนรู้การประยุกต์ใช้งานและการเขียนโปรแกรมภาษาแอสเซมบลี ที่ใช้ในการควบคุมไมโครคอนโทรลเลอร์ตระกูล MCS-51

1.2.3 เพื่อศึกษาถึงความรู้ในการเขียนโปรแกรม Microsoft Visual J++ (ในส่วนของ Applet)

1.2.4 เพื่อเรียนรู้ถึงหลักการทำงานของโปรแกรม Microsoft Visual J++ ที่ใช้ติดต่อและแสดงบนได้บน web browser

1.2.5 เพื่อเรียนรู้ที่จะสามารถนำความรู้มาประยุกต์และสามารถนำมาใช้งานได้จริงในการทำงาน

1.2.6 เพื่อลดต้นทุนการนำเข้าบอร์ดทดลองไมโครคอนโทรลเลอร์ตระกูลMCS-51 ที่ใช้ในการเรียนการสอน

1.3 ขอบเขตของโครงงาน

เนื่องจากวัตถุประสงค์ของโครงงานเพื่อมุ่งเน้นการสร้างใช้งานบอร์ดจำลองบอร์ดจำลองไมโครคอนโทรลเลอร์ตระกูล MCS-51เพื่อที่จะทำให้ผู้ใช้งานเกิดความสะดวกในการศึกษาและทำการทดลองในการเขียนโปรแกรมควบคุมการทำงานไมโครคอนโทรลเลอร์ MCS-51 โดยบอร์ด

ทดลองที่เกิดจากการพัฒนาโครงงานจะแสดงผลการทำงานตามโปรแกรม ภาษาแอสเซมบลีที่เขียนลงใน Text editor บนบอร์ดจำลองโดยจะแสดงผลโดยผู้ใช้กดปุ่มRunซึ่งจะเป็นการแสดงผลทีละบรรทัดของโปรแกรมภาษาแอสเซมบลีที่เขียน การแสดงผลจะจะแบ่งเป็นสองส่วนคือส่วนแรกเป็นส่วนของอุปกรณ์ที่อยู่บนบอร์ดจำลองซึ่งประกอบด้วย 7-Segment, LED, DIP Switch, Push Button Switch, สเต็ปมอเตอร์และส่วนที่สองคือส่วนที่แสดงข้อมูลไมโครคอนโทรลเลอร์ที่จะประกอบด้วย Register, Special Register, Internal RAM ซึ่งจะทำให้ผู้ใช้สามารถเห็นการทำงานได้ทั้งสองส่วนพร้อมกัน ส่วนของการเชื่อมต่อกับอุปกรณ์บนบอร์ดจำลองสามารถที่จะเลือกพอร์ตการทำงานของไมโครคอนโทรลเลอร์กับอุปกรณ์ต่างๆได้ และคำสั่งในการเขียนโปรแกรมแอสเซมบลีนั้นได้ครอบคลุมคำสั่งเกือบทั้งหมดยกเว้นเพียงในกลุ่มคำสั่งในการโอนย้ายข้อมูลกับหน่วยความจำภายนอก กลุ่มคำสั่งการโอนย้ายข้อมูลกับหน่วยความจำโปรแกรมภายนอกได้แก่คำสั่ง MOVX, MOVC และการติดต่อกับพอร์ตอนุกรม

โครงงานนี้ทำการพัฒนาโดยการเขียนโปรแกรม Microsoft Visual J++ ซึ่งในการพัฒนาได้เลือกในส่วนที่เป็น จาวาแอพเพลต ซึ่งจะสามารถแสดงผลได้โดยใช้โปรแกรมบราวเซอร์ที่สนับสนุนจาวาในการเรียกแสดงผลบอร์ดจำลองการทำงานซึ่งมีความสะดวกในการใช้งานและทดสอบโปรแกรมควบคุมการทำงานไมโครคอนโทรลเลอร์

1.4 ผลที่คาดว่าจะได้รับจากโครงงาน

1.4.1 ได้ศึกษาโครงสร้างและหลักการทำงานไมโครคอนโทรลเลอร์ตระกูล MCS-51

1.4.2 ได้เรียนรู้การเขียนโปรแกรมภาษาแอสเซมบลีที่ใช้ในการควบคุมไมโครคอนโทรลเลอร์ตระกูล MCS-51

1.4.3 ได้เรียนรู้ขั้นตอนและแนวทางในการใช้กระบวนการคิดเพื่อใช้ในการเขียนโปรแกรม Microsoft Visual J++

1.4.4 ได้นำความรู้ที่เรียนมาเพื่อใช้ในการเขียนโปรแกรมเพื่อจำลองการทำงานของบอร์ดจำลองไมโครคอนโทรลเลอร์ตระกูลMCS-51และได้ศึกษาและเรียนรู้กับปัญหาที่อาจเกิดขึ้นในการปฏิบัติงานจริง

1.4.5 ได้ผลการทำงานที่เป็นบอร์ดจำลองที่สามารถทำงานได้เสมือนกับการทำงานของบอร์ดทดลองจริง

1.4.6 สามารถประหยัดต้นทุนในการจัดหาบอร์ดทดลองไมโครคอนโทรลเลอร์ตระกูลMCS-51ที่นำมาใช้ในการเรียนการสอนได้

1.5 ขั้นตอนการดำเนินการ

1.5.1 การพิจารณารูปแบบที่ต้องการสร้างบอร์ดจำลองไมโครคอนโทรลเลอร์ตระกูลMCS-51

1.5.2 วางแผนการพัฒนาโครงงานโดยแยกเนื้อหาที่จะศึกษาแต่ละส่วน

1.5.3 ศึกษาเนื้อหาที่จัดแบ่งไว้แต่ละส่วน

1.5.4 พัฒนาโครงงานจากรูปแบบที่ได้ระบุและใช้ความรู้ที่ศึกษามาประยุกต์ในการทำงาน

1.5.5 ตรวจสอบความถูกต้องของโปรแกรมที่ได้พัฒนาและทดสอบการใช้งานจริงๆ เพื่อให้ทราบถึงข้อผิดพลาดและจุดบกพร่องของโครงงานที่พัฒนา

1.5.6 แก้ไขข้อผิดพลาดหลังจากการทดสอบใช้งานจริงเพื่อให้ได้งานที่สมบูรณ์

บทที่ 2
ทฤษฎีและหลักการของโครงงาน

โครงงานบอร์ดจำลองไมโครคอนโทรลเลอร์ตระกูลMCS-51นั้นมีทฤษฎีและหลักการที่เป็นองค์ประกอบสำคัญ 3 ส่วน คือ ส่วนโครงสร้างสถาปัตยกรรมของไมโครคอนโทรลเลอร์ MCS-51 การเขียนโปรแกรมภาษาแอสแซมบลี้ เพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51 และการเขียนโปรแกรม Microsoft Visual J++

2.1 โครงสร้างสถาปัตยกรรมของไมโครคอนโทรลเลอร์ MCS-51

ความก้าวหน้าทางเทคโนโลยีไมโครคอนโทรลเลอร์มักนำมนุษยชาติไปสู่จุดหมายใหม่อยู่เสมอ การเกิดขึ้นของไมโครคอนโทรลเลอร์MCS-51 โดยIntelผู้ยิ่งใหญ่ด้านไมโครโปรเซสเซอร์ได้จุดประกายในการนำไมโครคอนโทรลเลอร์ไปใช้ในการควบคุมระบบแทนที่ไมโครโปรเซสเซอร์ นับจากนั้นบรรดาผู้ผลิตซิปใหญ่น้อยก็หันมาพัฒนาและผลิตไมโครคอนโทรลเลอร์กันอย่างมากมาย

Zilog พัฒนาไมโครคอนโทรลเลอร์ตระกูล Z8x ในขณะที่ Motorola นำเสนอไมโครคอนโทรลเลอร์ตระกูล 68xx ด้าน ST-Thomson ก็แนะนำไมโครคอนโทรลเลอร์ตระกูล ST6x ออกสู่ตลาดในขณะเดียวกัน Siemens และ Phillips นำเทคโนโลยีของไมโครคอนโทรลเลอร์ ตระกูล MCS-51 มาพัฒนาต่อให้มีขีดความสามารถสูงขึ้นเป็นเบอร์ SAB80C535 , SAB80C537 , 80C552 เป็นต้น

การเปลี่ยนแปลงทางไมโครคอนโทรลเลอร์เกิดขึ้นอีกครั้งเมื่อ Atmel พัฒนาหน่วย ความจำโปรแกรมในไมโครคอนโทรลเลอร์เป็นแบบแฟรชทำให้สามารถเขียนลบหน่วยความจำโปรแกรมได้นับพันครั้งส่งผลให้ความจำเป็นในการใช้หน่วยความจำโปรแกรมภายนอกซึ่งเดิมบรรจุในอีพรอมลดน้อยลงขนาดของระบบก็เล็กลงราคาถูกลงเป็นอย่างมากอย่างไรก็ตามไมโครคอนโทรลเลอร์ทั้งหมดที่กล่าวมาพัฒนาในลักษณะของ CISC คำสั่งหนึ่งคำสั่ง ของไมโครคอนโทรลเลอร์จะใช้สัญญาณนาฬิกาหลายลูก จำนวนคำสั่งที่ผู้ใช้งานต้องจดจำมีจำนวนมากจนกระทั่ง Microchip ได้ทำการพัฒนาไมโครคอนโทรลเลอร์ในลักษณะ RISC ขึ้น โดยในการทำงานหนึ่งคำสั่งจะใช้สัญญาณนาฬิกาเพียงลูกเดียว และมีจำนวนคำสั่งเพียง 33-35 คำสั่งเท่านั้น โมโครคอนโทรลเลอร์ที่ microchip พัฒนาคือ ไมโครคอนโทรลเลอร์ตระกูล PIC

ไมโครคอนโทรลเลอร์ตระกูล PIC เริ่มเข้ามามีบทบาทอย่างมากในช่วงปลายศตวรรษที่ 20 เริ่มได้รับความนิยมใช้งานอย่างแพร่หลายทั้งในสหรัฐอเมริกาและในยุโรป สินค้าและผลิตภัณฑ์จำนวนมากใช้ ไมโครคอนโทรลเลอร์ ตระกูลนี้ในการควบคุมการ ทำงานส่งผลให้ความต้องการเรียนรู้และใช้งานไมโครคอนโทรลเลอร์ ตระกูลนี้ในประเทศไทยเริ่มมีมากขึ้น ประกอบกับเริ่มมีการนำเข้า ไมโครคอนโทรลเลอร์ตระกูล PIC เข้ามาจำหน่าย ในประเทศไทย ทว่าการ ให้ข้อมูลเป็นไปอย่างมีข้อจำกัด ทำให้การเรียนรู้และใช้งานไมโครคอนโทรลเลอร์ตระกูล นี้เป็นไปอย่างไม่ราบ

รื่น ทั้งที่ตัวไมโครคอนโทรลเลอร์เองมีคุณสมบัติที่เหมาะสม อย่างมากในการนำมาใช้ควบคุมระบบขนาดเล็กถึงขนาดกลาง

PIC16F84 Theory&Practical Approach จึงเกิดขึ้น ในการอธิบายการทำงานของไมโครคอนโทรลเลอร์ตระกูล PIC จะต้องเลือกใช้เบอร์ที่สามารถนำมาพัฒนาได้ นั่นคือสามารถอ่าน-เขียน-ลบ หน่วยความจำโปรแกรมภายใน ไมโครคอนโทรลเลอร์ได้ภาย ใต้งบประมาณที่ประหยัดที่สุด PIC16F84 จึงถูก เลือกมาเป็นตัวแทนในการอธิบาย และเรียนรู้การใช้งานไมโครคอนโทรลเลอร์ ตระกูล PIC เนื่องจาก PIC16F84 มีหน่วยความจำโปรแกรมขนาด 1 กิโลไบต์ เป็นหน่วยความจำแบบแฟลช สามารถลบ-เขียนได้นับพันครั้ง มีพอร์ตอินพุตและเอาต์พุตให้ใช้งาน 13 บิต สามารถใช้วงจรกำเนิดสัญญาณนาฬิกาแบบคริสตอลหรือวงจร RC ก็ได้ กินกำลังงานไฟฟ้าต่ำ สามารถใช้แบตเตอรี่เป็นแหล่งจ่ายไฟได้ ด้านการโปรแกรมใช้วงจรหรือเครื่อง โปรแกรมที่มีราคาถูกมาก วงจรอย่างง่ายที่สุดที่สามารถโปรแกรม PIC16F84 ได้ใช้เพียงตัวต้านทาน , ตัวเก็บประจุ และไดโอดไม่กี่ตัวเท่านั้น ในขณะที่ซอฟต์แวร์ที่ใช้ในการพัฒนาก็มีราคาถูก สำหรับบางท่านที่สามารถเข้าไปดูข้อมูลในอินเตอร์เน็ตก็อาจ ค้นหาซอฟต์แวร์ที่ใช้ในการโปรแกรมได้ฟรี ๆ ไมโครคอนโทรลเลอร์ MCS-51 เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิตที่มีอุปกรณ์สนับสนุนประกอบอยู่ภายในหลายอย่างได้แก่ หน่วยความจำสำหรับเก็บข้อมูล หน่วยความจำสำหรับเก็บโปรแกรม ตัวตั้งเวลา/ตัวนับ อุปกรณ์รับส่งข้อมูลแบบอนุกรม เนื่องจากโครงสร้างของไมโครคอนโทรลเลอร์มีอุปกรณ์สนับสนุนประกอบอยู่ภายในนี้เอง ทำให้การใช้งานง่ายขึ้นและมีประสิทธิภาพมากขึ้นโดยไม่ต้องมีการเชื่อมต่ออุปกรณ์ภายนอกเพิ่มเติมมากเหมือนกับ ตัวไมโคร-โปรเซสเซอร์ทั่วไป นอกจากนี้หากเราต้องการใช้งานไมโครคอนโทรลเลอร์ร่วมกับ อุปกรณ์อื่นเพิ่มเติมเช่น ไอซี 8255 หรือหน่วยความจำภายนอก ยังสามารถนำมาเชื่อมต่อเพิ่มเติมเข้ากับไมโครคอนโรลเลอร์ได้อีกด้วย

2.1.1 โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

โครงสร้างภายในพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แสดงในรูปที่ 2.10 ประกอบด้วยอุปกรณ์ต่างๆ ดังนี้

-หน่วยประมวลผลกลางขนาด 8 บิต

-หน่วยประมวลผลสำหรับข้อมูลแบบบิต (BOOLEAN PROCRSSOR)

-ความสามารถในการอ้างตำแหน่งของหน่วยความจำโปรแกรม 64 กิโลไบต์

-ความสามารถในการอ้างตำแหน่งของหน่วยความจำข้อมูล 64 กิโลไบต์

-หน่วยความจำโปรแกรมภายในขนาด 4 กิโลไบต์ แบบ อีพรอม (เบอร์ 8451)

-หน่วยความจำแบบ แรม ภายในจำนวน 128 ไบต์

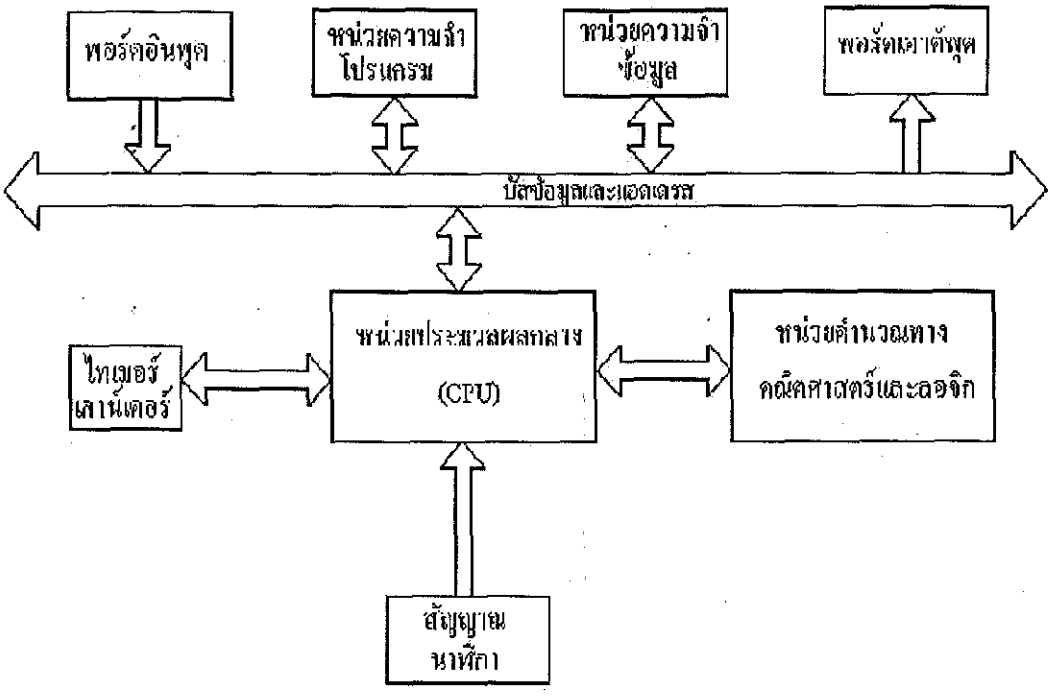
-พอร์ตอินพุต/เอาต์พุตแบบขนานจำนวน 32 เส้น ซึ่งสามารถแยกทำงานได้อย่างอิสระ

-วงจรนับ/จับเวลาขนาด 16 บิต จำนวนสองวงจร

-วงจรสื่อสารแบบอนุกรมแบบดูเพล็กเต็ม(FULL DUPLEX)

-วงจรควบคุมการอินเตอร์รัปต์จากแหล่งกำเนิดสัญญาณ 6 ประเภท พร้อมการกำหนด ลำดับ

-วงจรผลิตสัญญาณนาฬิกาภายในซึ่งโครงสร้างการทำงานทั้งหมดของไมโครคอน-โทรลเลอร์จะอาศัยหลักการทำงานที่เกี่ยวข้องกัน โดยอาศัยหลักการทำงานที่เป็นไป ตามโครงสร้างเสมอ

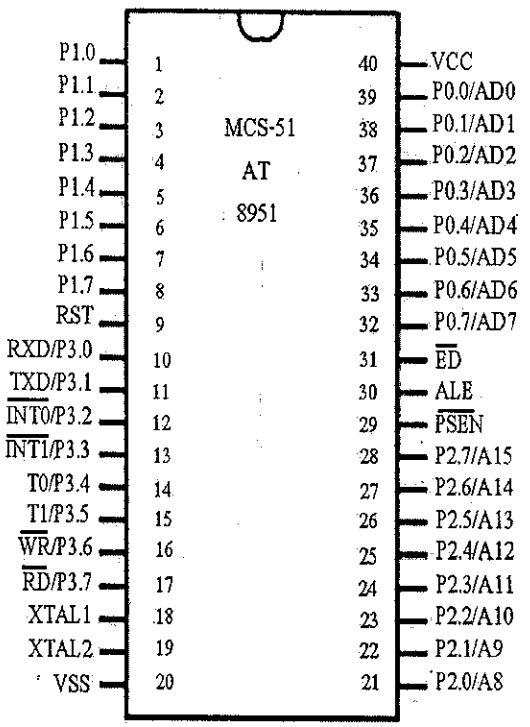


รูปที่2.1แสดงโครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

โดยมากแล้วไมโครคอนโทรลเลอร์ตระกูลนี้มักจะมีรูปร่างของไอซีเป็นแบบขนาด 40 ขา ดังแสดงในรูปที่ 2.2 ซึ่งแต่ละขาสัญญาณจะมีหน้าที่ที่ระบุชัดเจนตามสัญลักษณ์ชื่อย่อ ที่กำกับในแต่ละขา อย่างไรก็ตามจะมีบางขาสัญญาณที่อาจจะมีหน้าที่ได้มากกว่าหนึ่งอย่าง (ซึ่งเขียนกำกับไว้ว่า ALTERNATE FUNCTION ในรูปที่ 2.2) ซึ่งจะไม่สามารถใช้งานในเวลาเดียวกันได้ ตัวอย่างเช่นขาสัญญาณบิต 0 ของพอร์ต 3 (ใช้ตัวย่อเป็น P3.0) อาจจะใช้เป็นขาสัญญาณเอาต์พุต หรืออินพุตตามปกติ ภายในไมโครคอนโทรลเลอร์ MCS-51 ซึ่งประกอบด้วยหน่วยการทำงานต่างๆ ภายในไอซี MCS-51 จำนวนมาก โดยแต่ละบล็อกซึ่งเป็นวงจรควบคุมรีจิสเตอร์ (REGISTER) หรือหน่วยความจำภายในของไอซี MCS-51 จะถูกเชื่อมต่อเข้าด้วยกันผ่านทางเส้นสัญญาณที่เรียกว่าบัสข้อมูลภายใน รีจิสเตอร์และหน่วยความจำเหล่านี้จะถูกนำไปใช้ระหว่างการประมวลผลคำสั่ง หน้าที่ของโปรแกรมที่ผู้ใช้สร้างขึ้นมาก็เป็นการควบคุมการรับหรือส่งข้อมูลระหว่างรีจิสเตอร์เหล่านี้ ซึ่งอาจจะมีการดำเนินการร่วมกับหน่วยการดำเนินงานประมวลผลทางคณิตศาสตร์และลิจิกหรือเรียกว่า ARITHMATIC AND LOGIC UNIT: ALU

2.1.2 โครงสร้างหน่วยความจำภายในไมโครคอนโครลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 แยกการจัดการหน่วยความจำออกเป็นสองส่วนอย่างชัดเจน คือ หน่วยความจำโปรแกรม (PROGRAM MEMORY) และหน่วยความจำข้อมูล (DATA MEMORY) หน่วยความจำทั้งสองนี้ มีหน้าที่แตกต่างไปจากกัน และใช้วิธีการอ้างแอดเดรสสัญญาณการติดต่อแยกออกจากกัน

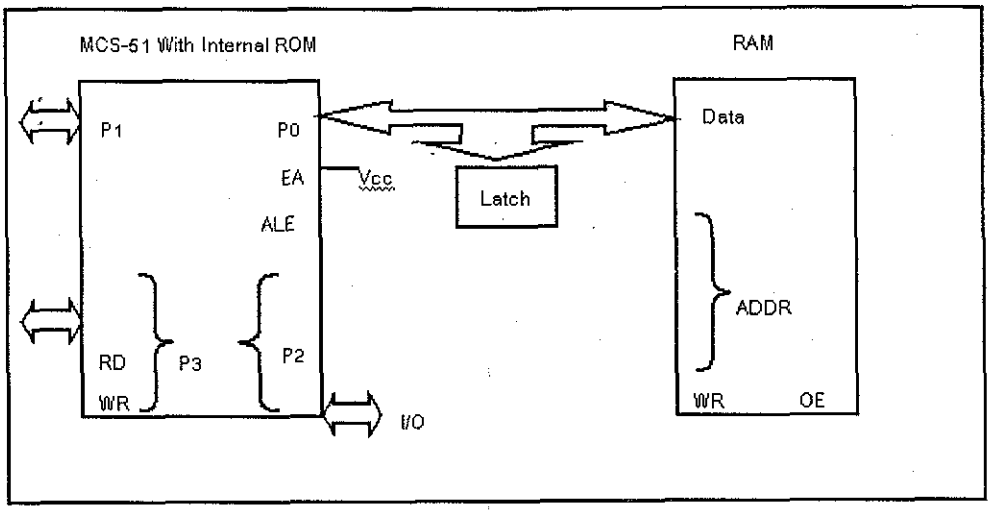


รูปที่2.2 แสดงรูปร่างและการจัดวางขาต่างๆ ของไมโครคอนโทรลเลอร์ MCS-51

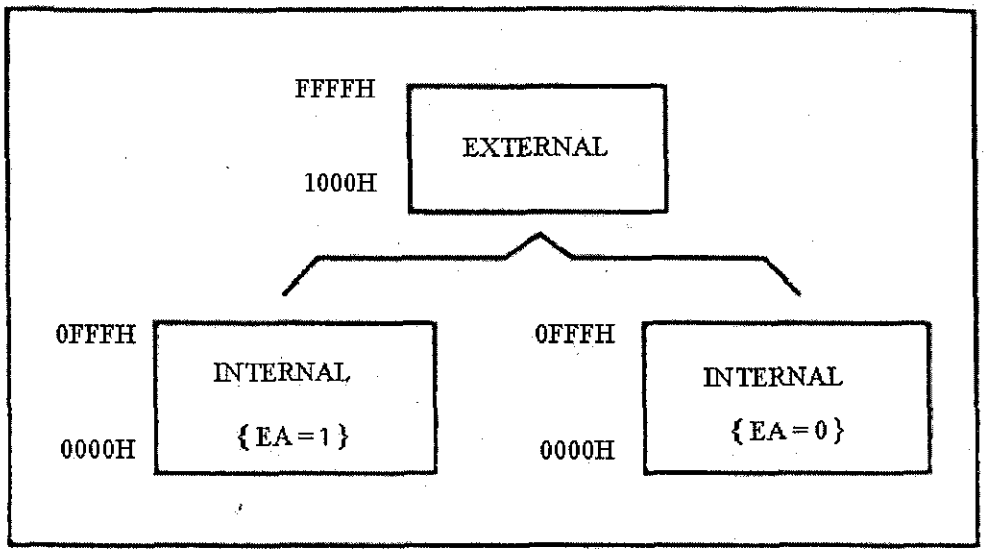
1. หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 เป็นบริเวณหน่วยความจำสำหรับเก็บข้อมูลและคำสั่งใช้งานต่างๆ ซึ่งแม้ว่าจะไม่มีการจ่ายกระแสไฟฟ้าให้กับระบบข้อมูลเหล่านี้ก็ยังคงอยู่ไม่สูญหายโครงสร้างของหน่วยความจำโปรแกรม มีลักษณะเช่นเดียวกับหน่วยความจำที่บรรจุอยู่ในไมโครคอนโทรลเลอร์ MCS-51 ของหน่วยความจำ ประเภทต่างๆ เช่น หน่วยความจำแบบรอม (READ ONLY MEMORY) หรือ อีพรอม (ERASABLE PROGRAMABLE READ ONLY MEMORY) ในไมโครคอนโทรลเลอร์ MCS-51 สามารถอ่านข้อมูลหน่วยความจำโปรแกรมนี้ได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทของหน่วยความจำโปรแกรมเป็น 2 ลักษณะ ตามตำแหน่งของหน่วยความจำนั้น คือ หน่วยความจำโปรแกรมภายใน (INTERNAL PROGRAM MEMORY) ซึ่งเป็นหน่วยความจำรอม หรือ อีพรอม ที่อยู่ภายในตัวไอซีของไมโคร-

คอนโทรลเลอร์เอง และหน่วยความจำโปรแกรมภายนอก (EXTERNAL PROGRAM MEMORY) ซึ่งเป็นการใช้ไอซีหน่วยความจำมาทำหน้าที่เป็นหน่วยความจำโปรแกรมของระบบ



รูปที่ 2.3 แสดงการใช้หน่วยความจำสำหรับเก็บโปรแกรม



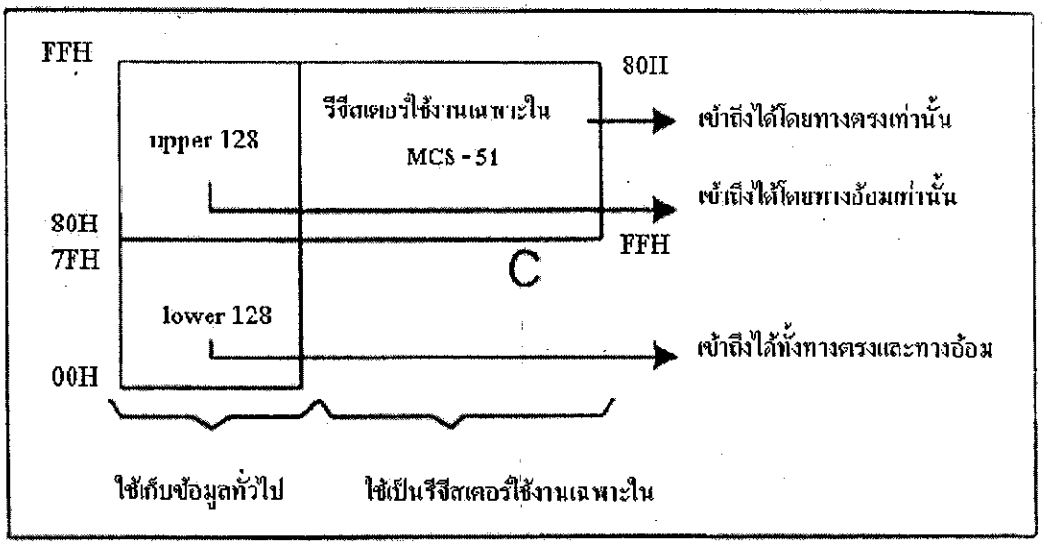
รูปที่ 2.4 แสดงการจัดพื้นที่ของหน่วยความจำโปรแกรมภายในและภายนอก

ไมโครคอนโทรลเลอร์เบอร์ต่างๆ ของตระกูล 8051 นี้สามารถขยายให้ใช้งาน ในหน่วยความจำภายนอกได้ทั้งสิ้น โดยกรณีที่มีหน่วยความจำโปรแกรมภายในอยู่แล้ว การอ้างตำแหน่งแอดเดรสที่มีทั้งในหน่วยความจำโปรแกรมภายในและภายนอกนั้นจะต้องทำการควบคุมระดับลอจิกของสัญญาณในขณะนั้นด้วย ขนาดหน่วยความจำโปรแกรมภายในของไมโครคอนโทรลเลอร์เบอร์ต่างๆ ภายในตระกูล 8051 จะแตกต่างกันออกไป เพื่อความเหมาะสมกับการนำไปใช้งานลักษณะต่างๆเช่น

-8051 และ 8052 มีหน่วยความจำแบบรอม 4 และ 8 กิโลไบต์

-8751 มีหน่วยความจำแบบ อีพรอม ขนาด 4 กิโลไบต์ ข้อมูลที่จัดเก็บภายในนี้ ซึ่งสามารถใช้แสงอุลตร้าไวโอเลตลบและนำกลับไปบรรจุโปรแกรมใหม่ได้อีกครั้งหนึ่ง

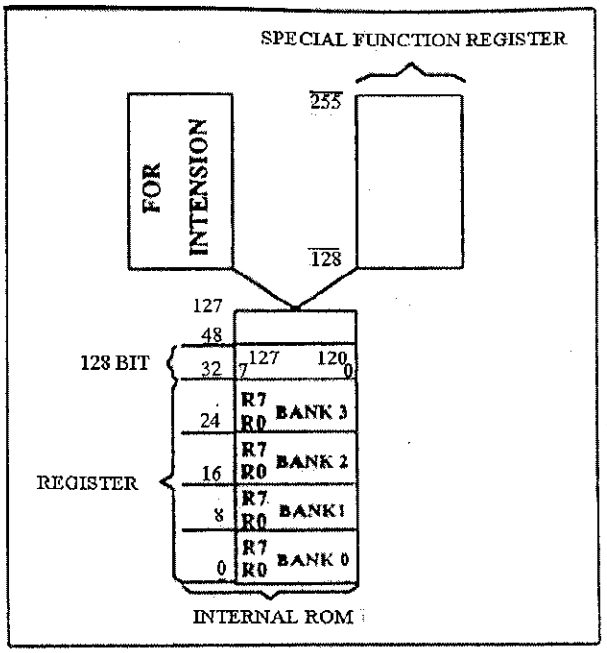
-8031 และ 8032 ไม่มีหน่วยความจำโปรแกรมอยู่ภายในตัวไอซี ดังนั้นในการนำไปใช้งานจึงจำเป็นต้องอาศัยหน่วยความจำโปรแกรมภายนอกเสมอ



รูปที่ 2.5 แสดงหน่วยความจำสำหรับเก็บข้อมูลภายในไมโครคอนโทรลเลอร์ MCS-51

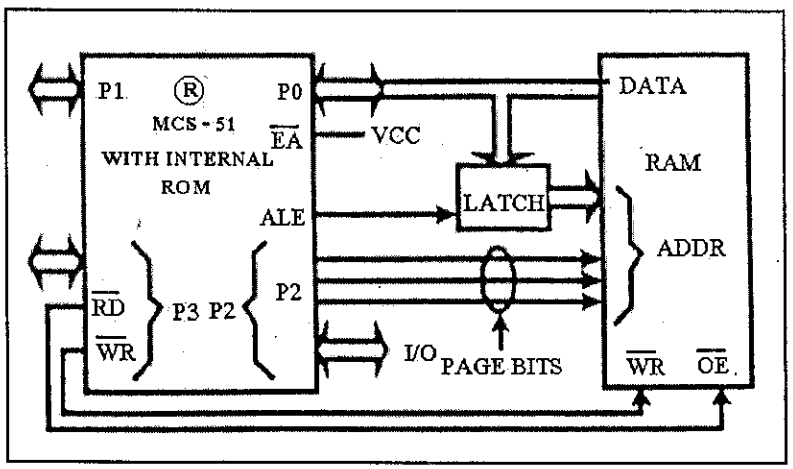
2. หน่วยความจำข้อมูล

หน่วยความจำข้อมูล (DATA MEMORY) ซึ่งโดยพื้นฐานแล้วเป็นหน่วยความจำแรมสามารถเขียนหรืออ่านข้อมูลได้ (READ OR WRITE MEMORY) ใช้สำหรับเก็บข้อมูลหรือตัวแปร ที่เกิดขึ้นในขณะที่กำลังประมวลผลโปรแกรมไว้เป็นการชั่วคราว ซึ่งโดยพื้นฐานแล้วหน่วยความจำข้อมูลจัดเป็นหน่วยความจำแรมแบบสแตติกดังนั้นเมื่อไม่มีการจ่ายไฟฟ้าให้กับระบบก็จะมีผลทำให้ข้อมูลที่จัดเก็บไว้ภายในหน่วยความจำนี้สูญไป พื้นที่ของหน่วยความจำข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 มีได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทออกเป็นสองลักษณะตามตำแหน่งที่ตั้งของหน่วยความจำนั้น ตามลักษณะของหน่วยความจำโปรแกรมภายในซึ่งก็เป็นแรมที่อยู่ภายในตัวไอซีในตระกูลของไมโครคอนโทรลเลอร์ และหน่วยความจำข้อมูลภายนอกซึ่งเป็นการใช้ไอซีหน่วยความจำแรมมาเพิ่มเติมเข้าไปในวงจรลักษณะเดียวกับ การนำไอซีอีพรอมมาใช้งานเป็นหน่วยความจำโปรแกรมนั่นเอง



รูปที่ 2.6 แสดงการจัดหน่วยความจำข้อมูล

โดยที่หน่วยความจำสำหรับเก็บข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 นี้สามารถแบ่งออกเป็น 2 ส่วนคือ ในส่วนที่เป็นหน่วยความจำสำหรับเก็บข้อมูลภายในไอซี และหน่วยความจำสำหรับเก็บข้อมูลภายนอกไมโครคอนโทรลเลอร์ MCS-51 ทุกๆ เบอร์จะมีหน่วยความจำเก็บข้อมูลทั่วๆ ไปภายในไอซีอย่างน้อยคือ 128 ไบต์ ไปจนถึง 256 ไบต์ทั้งนี้ ขึ้นกับเบอร์ของไอซี หน่วยความจำสำหรับเก็บ ข้อมูลภายในไอซีในบริเวณ 128 ไบต์เรียกว่า LOWER 128 และในบริเวณ 128 ไบต์หลัง ที่มีเพิ่มในบางเบอร์มีชื่อเรียกว่า UPPER 128 แสดงดังรูปที่ 2.6

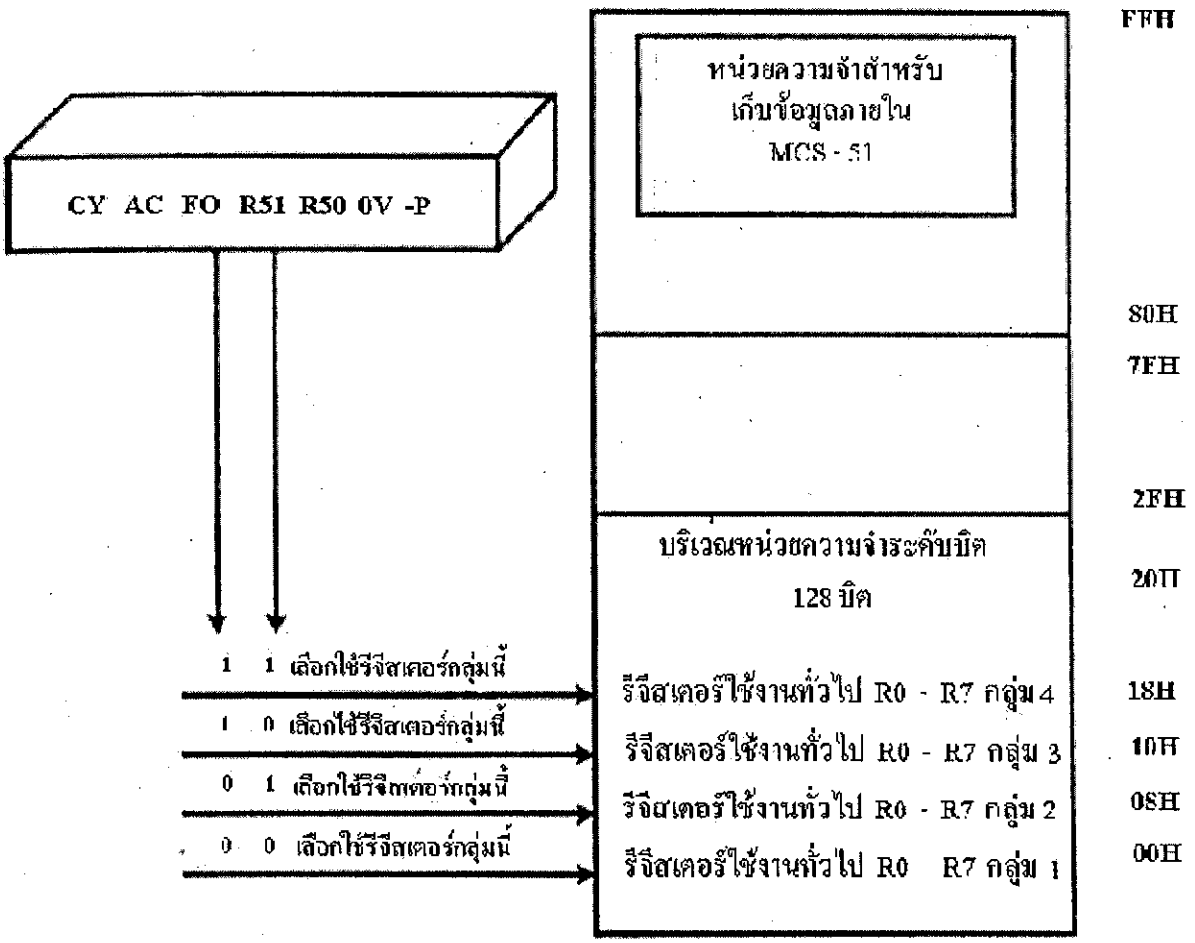


รูปที่ 2.7 แสดงการต่อกับหน่วยความจำข้อมูลภายนอกไอซี

3. รีจิสเตอร์ที่เกี่ยวกับไมโครคอนโทรลเลอร์ MCS-51

รีจิสเตอร์ในกลุ่มนี้จะเป็นรีจิสเตอร์ขนาด 16บิตที่ใช้งานเพื่อเก็บข้อมูลของตัวแอดเดรสเป็นสำคัญ โดยค่าที่อยู่ภายในแอดเดรสนี้จะนำไปเป็นค่าของข้อมูลที่ส่งออกไปทางบัสแอดเดรส ในส่วนของไมโครคอนโทรลเลอร์ เพื่อบอกตำแหน่งที่ต้องการติดต่อ รีจิสเตอร์ที่จัดในกลุ่มนี้ประกอบด้วย **รีจิส**

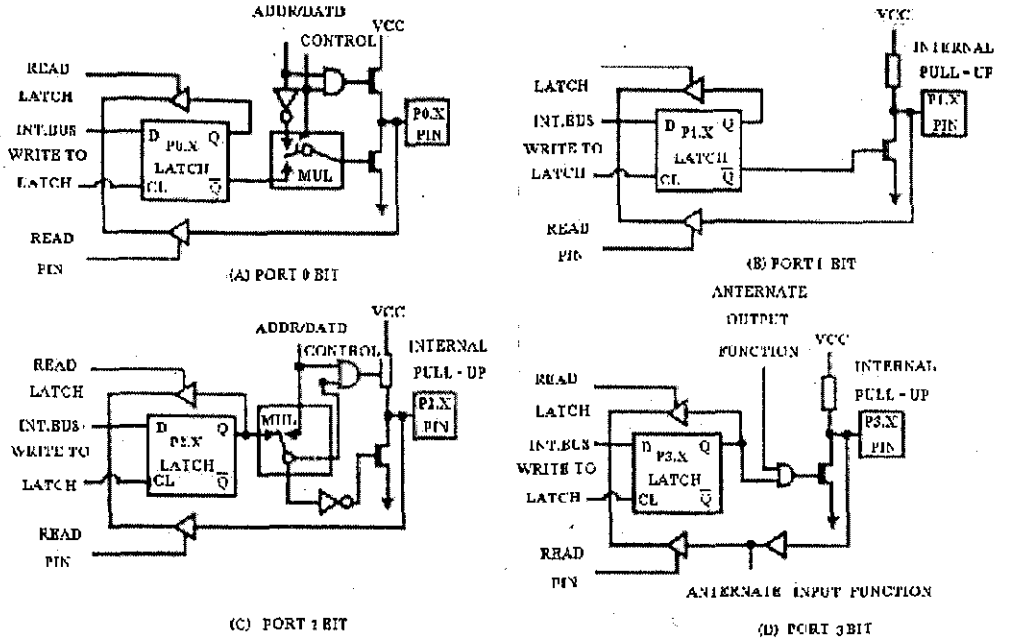
เตอร์ใช้งานทั่วไป (GENERAL-PURPUSE REGISTERS) รีจิสเตอร์ในกลุ่มนี้จัดเป็นพื้นที่หน่วยความจำที่ใช้ในการสนับสนุนในการประมวลผล การทำงานจากหน่วยประมวลผลทางคณิตศาสตร์และลอจิก (ALU) เพื่อให้สามารถจัดการข้อมูลให้เร็วที่สุด นอกจากนี้โปรแกรมที่ไม่ได้ใช้คำสั่งเหล่านี้ก็ยังใช้เป็นการเก็บข้อมูลตัวแปรภายในโปรแกรม จะเห็นได้ว่าชื่อของรีจิสเตอร์ไม่ว่าจะอยู่ในรีจิสเตอร์แบงก์ใด ก็จะมีชื่อว่า R0 ถึง R7 เหมือนกันทั้งสิ้น ดังนั้นในการใช้งานผู้ใช้จะต้องให้ความระมัดระวังว่า ต้องการรีจิสเตอร์นั้นๆ จากแบงก์ใดๆ ซึ่งการกำหนดเลือกแต่ละกลุ่มของรีจิสเตอร์นี้ก็ทำได้ง่าย เพียงการกำหนดค่าของบิตที่อยู่ภายในแฟลก (PSW) เท่านั้นอย่างไรก็ตามโดยทั่วไปก็มักจะมีการใช้งานรีจิสเตอร์ R0 ถึง R7 เฉพาะในแบงก์ 0 เท่านั้น ดังนั้นพื้นที่ของแบงก์อื่นๆ ที่เหลือก็สามารถนำมาใช้ในลักษณะของหน่วยความจำแรม



รูปที่ 2.8 แสดงการเลือกใช้รีจิสเตอร์ใช้งานทั่วไป R0-R7 ในแต่ละกลุ่ม

รีจิสเตอร์หน้าที่พิเศษ เป็นรีจิสเตอร์หน้าที่พิเศษ (SFR) เป็นรีจิสเตอร์สำหรับ การควบคุมหน้าที่และการทำงานของอุปกรณ์หรือพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 ทั้งหมด ตำแหน่งของรีจิสเตอร์เหล่านี้จะจัดอยู่ในบริเวณแอดเดรส 80H - FFH การใช้งานรีจิสเตอร์หน้าที่พิเศษเหล่านี้สามารถทำได้ทั้งการระบุชื่อของรีจิสเตอร์ หรือตำแหน่งแอดเดรส ที่เป็นของรีจิสเตอร์นั้นก็ได้ การ

จัดพื้นที่หน่วยความจำสำหรับรีจิสเตอร์หน้าที่พิเศษเหล่านี้ โดยมีข้อสังเกตว่ารีจิสเตอร์ที่อยู่ในตำแหน่งแอดเดรสที่มีจำนวนเป็นทวีคูณของค่า 8 จะสามารถอ้างถึงในระดับบิตได้ด้วย (นั่นคือแอดเดรส 80H 88H 90H A0H A8H B0H B8H D0H E0H และ F0H) -แอกคิวมูเลเตอร์ (ACCUMULATOR) หรือ ACC เป็นรีจิสเตอร์ขนาด 8 บิต ทำหน้าที่ในการเก็บข้อมูลที่จะส่งให้กับหน่วยทำงานภายในหน่วยประมวลผลกลาง และเก็บผลลัพธ์ที่ได้จากการทำงานเท่านั้น การทำงานของรีจิสเตอร์นั้มีลักษณะเช่นเดียวกับตัวแอกคิวมูเลเตอร์ของโปรเซส-เซอร์ทั่วไป การใช้งานในโปรแกรมซึ่งใช้เรียกเป็น รีจิสเตอร์ **A**



รูปที่2.9 แสดงโครงสร้างของแต่ละพอร์ตในไมโครคอนโทรลเออร์ MCS-51

2.2 ภาษาแอสแซมบลี้

การเรียนรู้เพื่อใช้งานไมโครคอนโทรลเลอร์ สิ่งที่สำคัญในลำดับต่อมาจากที่ทำความเข้าใจถึงโครงสร้างทางฮาร์ดแวร์แล้วนั่นคือ การเขียนโปรแกรมเพื่อกำหนดให้ไมโครคอนโทรลเลอร์ทำงานข้อมูลของโปรแกรมที่ไมโครคอนโทรลเลอร์ต้องการจะอยู่ในรูปของรหัสเลขฐานสิบหกหรือที่เรียกกันว่าภาษาเครื่อง หรือ แมชีนโค้ด (Machine Code) แต่เนื่องจากการเขียนโปรแกรมในลักษณะที่เป็นภาษาเครื่องนี้ ผู้เขียนโปรแกรมต้องทำการเปิดตารางรหัสคำสั่งซึ่งเป็นเรื่องที่ยุ่งยากและทำให้การตรวจสอบโปรแกรมที่เขียนขึ้นกระทำได้ยากจึงใช้การเขียนโปรแกรมด้วยภาษาแอสแซมเบลอร์(Assembler) ทำการแปลภาษาแอสแซมบลีที่เขียนขึ้นนั้นเป็นภาษาเครื่องแล้วเขียนลงในหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ต่อไป

2.2.1 โครงสร้างของโปรแกรมภาษาแอสแซมบลี

ประกอบด้วย 4 ส่วนหลักคือ

1. **ลาเบล** (Label) ใช้ในการอ้างถึงบรรทัดใดบรรทัดหนึ่งของโปรแกรมที่ทำการเขียนขึ้น
2. **รหัสนีโมนิก** (Mnemonic) เป็นส่วนแสดงคำสั่งของไมโครคอนโทรลเลอร์ที่ต้องการให้กระทำ
3. **โอเปอร์แรนด์** (Operand) เป็นส่วนที่แสดงถึงตัวกระทำหรือถูกกระทำและข้อมูลที่ใช้ในการกระทำตามคำสั่งที่กำหนดโดยรหัสนีโมนิกก่อนหน้านี้
4. **คอมเมนต์** (Comment) เป็นส่วนที่ผู้เขียนโปรแกรมเขียนขึ้นเพื่อใช้ในการอธิบายคำสั่งที่กระทำ หรือผลของการกระทำคำสั่งในบรรทัดหรือโปรแกรมย่อยนั้นๆ ทั้งนี้เพื่อช่วยให้ผู้เขียนสามารถตรวจสอบโปรแกรมที่เขียนขึ้นได้ง่ายรวมถึงเป็นประโยชน์ต่อผู้ที่นำโปรแกรมนั้นมาศึกษาใหม่อีกด้วย

2.2.2 ชุดคำสั่งของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์MCS-51 ประกอบด้วยคำสั่งทั้งหมดจำนวนมากซึ่งนำมาแสดงไว้ในตารางของชุดคำสั่งต่างๆ ซึ่งสามารถจะจัดกลุ่มคำสั่งเหล่านี้ตามลักษณะและหน้าที่การทำงานที่คล้ายคลึงกัน เพื่อความสะดวกต่อการศึกษา ทำความเข้าใจและใช้งาน ดังนี้

1. กลุ่มการถ่ายเทข้อมูล คือ กลุ่มคำสั่งในการโอนย้ายข้อมูล ทำหน้าที่ในโอนย้ายข้อมูลระหว่างรีจิสเตอร์ หรือหน่วยความจำภายในแรม โดยมีรายละเอียดดังนี้ ชุดคำสั่งในการถ่ายเทแรมภายในซึ่งเวลาที่ใช้ในหนึ่งคำสั่งนั้น จะเป็นเวลาเมื่อขณะที่ความถี่ในการทำงานของหน่วยประมวลผลกลาง ที่ความถี่ 12 เมกะเฮิรตซ์ และรายละเอียดของแต่ละคำสั่งมีดังนี้ **MOV:**จะทำงานในลักษณะเป็นการถ่ายเทข้อมูลที่มีขนาดเป็นไบต์ หรือ บิตก็ได้ จากแหล่งกำเนิดเข้าสู่ตัวรับข้อมูลในฟิลด์โอเปอร์แรนด์ **PUSH:**จะทำงานโดยเพิ่มค่ารีจิสเตอร์ SP ก่อนแล้วจึงทำการถ่ายเทข้อมูล 1 ไบต์จากแหล่งกำเนิดไปบริเวณสแต็กตามตำแหน่งที่รีจิสเตอร์ SP กำหนด **POP:**การถ่ายเทข้อมูลขนาด 1 ไบต์จากบริเวณตำแหน่งที่รีจิสเตอร์ SP กำหนดไปยังรีจิสเตอร์ที่โอเปอร์แรนด์ กำหนดและหลังจากนั้นรีจิสเตอร์ SP จะลดค่าลง **XCH:**คำสั่งแลกเปลี่ยนไบต์ระหว่างแหล่งกำเนิดโอเปอร์แรนด์กับรีจิสเตอร์A **XCHD:**คำสั่งในการแลกเปลี่ยนขนาดนิบเบิลทางอันดับต่ำของแหล่งกำเนิดโอเปอร์แรนด์กับนิบเบิลอันดับต่ำลงของแอกคิวมูเลเตอร์

MOV A, Rn

จำนวนไบต์ : 1

การทำงาน : นำข้อมูลของรีจีสเตอร์ R0-R7 มาเก็บไว้ในแอกคิวมูเลเตอร์

MOV A, direct

จำนวนไบต์ : 2

การทำงาน : นำข้อมูลของหน่วยความจำข้อมูลภายในมาเก็บไว้ในแอกคิวมูเลเตอร์

MOV A, #data

จำนวนไบต์ : 2

การทำงาน : นำข้อมูลมาเก็บไว้ที่แอกคิวมูเลเตอร์

MOV A, @Rn

จำนวนไบต์ : 1

การทำงาน : นำข้อมูลจากแอดเดรสของ หน่วยความจำข้อมูลภายใน ที่กำหนดไว้ในรีจีสเตอร์ R0 หรือ R1 มาเก็บไว้ในแอกคิวมูเลเตอร์

MOV Rn, A

จำนวนไบต์ : 1

การทำงาน : นำข้อมูลจากแอกคิวมูเลเตอร์ไปเก็บไว้ในรีจีสเตอร์ R0-R7

MOV Rn, direct

จำนวนไบต์ : 2

การทำงาน : นำข้อมูลจากแอดเดรสของหน่วย ความจำข้อมูลภายใน ที่กำหนดมาเก็บไว้ ในรีจีสเตอร์ R0-R7

MOV Rn, #data

จำนวนไบต์ : 2

การทำงาน : นำข้อมูลมาเก็บไว้ที่รีจีสเตอร์ R0-R7

MOV direct, A

จำนวนไบต์ : 2,

การทำงาน : นำข้อมูลจากแอกคิวมูเลเตอร์มาเก็บไว้ที่หน่วยความจำข้อมูลภายใน

MOV direct, Rn

จำนวนไบต์ : 2

การทำงาน : นำข้อมูลที่อยู่ภายในรีจีสเตอร์ R0-R7 มาเก็บไว้ที่หน่วยความจำข้อมูลภายใน

MOV direct, direct

จำนวนไบต์ : 3

การทำงาน : นำข้อมูลจากหน่วยคว้ามจำข้อมูล ภายในแอดเดรสหนึ่ง มาเก็บไว้ที่หน่วย ความจำข้อมูล ภายในอีกแอดเดรสหนึ่ง

MOV direct, #data

จำนวนไบต์ : 3

การทำงาน : นำข้อมูลจากแอกคิวมูเลเตอร์มาเก็บไว้ที่หน่วยความจำข้อมูลภายใน

MOV direct, @Rn

จำนวนไบต์ : 2

การทำงาน : นำข้อมูลจากแอดเดรสของหน่วยความจำข้อมูลภายใน ที่กำหนดไว้ในรีจีสเตอร์ R0 หรือ R1 มาเก็บไว้ในหน่วยความจำ ข้อมูลภายในที่กำหนด

MOV @Rn, A

จำนวนไบต์ : 1

การทำงาน : นำข้อมูลจาก แอกคิวมูเลเตอร์ มาเก็บไว้ที่หน่วยความจำข้อมูลภายในที่กำหนด โดยค่า ของรีจีสเตอร์ R0 หรือ R1

MOV @Rn, direct

จำนวนไบต์ : 2

การทำงาน : นำข้อมูลจากหน่วยความจำข้อมูล ภายในแอดเดรสหนึ่ง มาเก็บไว้ที่หน่วย ความจำข้อมูลภายใน อีกแอดเดรสหนึ่ง ที่กำหนดโดยค่าของ R0 หรือ R1

MOV @Rn, #data

จำนวนไบต์ : 3

การทำงาน : นำข้อมูลไปเก็บไว้ที่หน่วยความจำข้อมูลที่กำหนดโดยค่าของ R0 หรือ R1

MOV C, bit

จำนวนไบต์ : 2

การทำงาน : นำข้อมูลในระดับบิต จากหน่วยความจำภายใน มาเก็บไว้ในแฟลกทด ซึ่งอยู่ในรีจีสเตอร์ PSW

MOV bit, C

จำนวนไบต์ : 2

การทำงาน : นำข้อมูลในแฟลกทด ไปเก็บไว้ที่หน่วยความจำข้อมูลภายใน ที่สามารถเข้าถึง ระดับบิตได้

MOVX A, @Rn

จำนวนไบต์ : 1

การทำงาน : นำข้อมูลจากแอดเดร้สของหน่วยความจำข้อมูลภายนอก ที่กำหนดไว้ในรีจีสเตอร์ R0 หรือ R1 มาเก็บไว้ในแอกคิวมูเลเตอร์

MOVX A, @DPTR

จำนวนไบต์ : 1

การทำงาน : นำข้อมูลจากแอดเดรสของหน่วยความจำข้อมูลภายนอก ที่กำหนดไว้ในรีจีสเตอร์ DPTR มาเก็บไว้ในแอกคิวมูเลเตอร์

MOVX @Rn, A

จำนวนไบต์ : 1

การทำงาน : นำข้อมูลจากแอกคิวมูเลเตอร์ ไปเก็บไว้ที่แอดเดรสของหน่วยความจำข้อมูล ภายนอก ที่กำหนดไว้ในรีจีสเตอร์ R0 หรือ R1

MOVX @DPTR, A

จำนวนไบต์ : 1

การทำงาน : นำข้อมูลจากแอกคิวมูเลเตอร์ มาเก็บไว้ในหน่วยความจำข้อมูล ภายนอกที่ แอดเดรส ซึ่งกำหนดไว้ในรีจีสเตอร์ DPTR

MOVC A, @A+DPTR

จำนวนไบต์ : 1

การทำงาน : นำข้อมูลจากหย่วยความจำข้อมูลภายนอก ในตำแหน่งแอดเดรสที่ได้รับการ กำหนดด้วยค่าของรีจีสเตอร์ A รวมกับค่าในรีจีสเตอร์ PC

MOVC A, @A+PC

จำนวนไบต์ : 1

การทำงาน : นำข้อมูลจากหน่วยความจำโปรแกรมภายนอก ในตำแหน่งแอดเดรสที่ได้รับ การกำหนดด้วยค่าของรีจีสเตอร์ A รวมกับค่าในรีจีสเตอร์ PC

XCH A, Rn

จำนวนไบต์ : 1

การทำงาน : แลกเปลี่ยนข้อมูลระหว่างรีจีสเตอร์ A กับข้อมูลภายในรีจีสเตอร์ R0-R7

XCH A, direct

จำนวนไบต์ : 2

การทำงาน : แลกเปลี่ยนข้อมูลระหว่างรีจีสเตอร์ A กับหน่วยความจำข้อมูลภายใน

XCH A, @Rn

จำนวนไบต์ : 1

การทำงาน : แลกเปลี่ยนข้อมูลระห้ว่าง A กับข้อมูลภายในแอดเดรสที่ถูกชี้โดย R0 หรือ R1

XCHD A, @Rn

จำนวนไบต์ : 1

การทำงาน : แลกเปลี่ยนข้อมูล บิต 0-3 ของรีจีสเตอร์ A กับข้อมูล บิต 3-0 ภายในแอดเดรสของหน่วยความจำที่ชี้โดยรีจีสเตอร์ R0 หรือ R1

PUSH direct

จำนวนไบต์ : 2

การทำงาน : เพิ่มค่าของรีจีสเตอร์ตัวชี้สแต็ก (SP) ไปหนึ่งตำแหน่ง จากนั้นนำค่าของ ข้อมูลในหน่วยความจำที่กำหนดไปเก็บไว้ในแอดเดรสที่ชี้โดย SP

POP direct

จำนวนไบต์ : 2

การทำงาน : นำข้อมูลในแอดเดรสที่ถูกชี้โดย SP กลับคืนหน่วยความจำใน แอดเดรสที่กำหนดไว้ แล้วลดค่าของ SP ไปหนึ่งตำแหน่ง

2. กลุ่มคำสั่งทางคณิตศาสตร์ เช่น การบวก ลบ คูณ และหารข้อมูลภายในตัวรีจิสเตอร์ต่างๆ ช่วงเวลาการทำงานของแต่ละคำสั่งนั้นจะกำหนดที่ความถี่ของสัญญาณนาฬิกาที่ 12 เมกะเฮิรตซ์ คำสั่งทางคณิตศาสตร์ส่วนใหญ่ใช้เวลา 1 ms ยกเว้นคำสั่ง INC DPTR ซึ่งใช้เวลา 2 ms โดยที่คำสั่งการคูณและหารใช้เวลา 4 ms โดยมีรายละเอียดการใช้คำสั่งมีดังนี้ **INC:**เป็นการบวกหนึ่งกับโอเปอร์แรนด์และใส่ค่าใหม่กลับเข้าที่ตัวโอเปอร์แรนด์นั้นๆ **DEC:**เป็นการลบออกจากตัวเลขที่อยู่ในแหล่งกำเนิดโอเปอร์แรนด์ และนำผลลัพธ์ที่ได้มาเก็บไว้ที่ตัวโอเปอร์แรนด์นั้น **ADD:** เป็นการบวกในแอกคิวมูเลเตอร์เข้ากับค่าในแหล่งกำเนิดโอเปอร์แรนด์ **ADDC:**เป็นการบวกค่าต่างๆ ในแอกคิวมูเลเตอร์เข้ากับค่าในแหล่งกำเนิดโอเปอร์-แรนด์และบวกกับบิตทดด้วย **SUBB:** เป็นการนำเลขที่แหล่งกำเนิดโอเปอร์แรนด์ ลบออกจากตัวเลขใน A และนำค่าบิตตัวทดมาลบออกอีกและผลลัพธ์ที่ได้นำมาใส่ลงในแอกคิวมูเลเตอร์ A **MUL:**เป็นการคูณแบบไม่คิดตัวเครื่องหมายของตัวเลขที่อยู่ใน แอกคิวมูเลเตอร์กับเลขใน รีจิสเตอร์ B แล้วได้ผลลัพธ์ 2 ไบต์ นำมาเก็บไว้ที่ AB โดย A จะรับอันดับต่ำส่วน B จะรับอันดับสูง **DIV:**เป็นคำสั่งในการหารแบบไม่คิดเครื่องหมายที่อยู่ในแอกคิวมูเลเตอร์แล้วหารตัวเลขในรีจิสเตอร์ B แล้วนำผลลัพธ์ไปเก็บในแอกคิวมูเลเตอร์และเศษของการหารตัวเลข จะเก็บไว้ในรีจิสเตอร์ B **DA**:สำหรับการบวกกันทางตัวเลข BCD เป็นการปรับค่ารวม ซึ่งเป็นผลมาจากการบวกกันทางไบนารี่ของระบบตัวเลข BCD ขนาด 2 หลักสองจำนวน การปรับค่าตัวเลขผลรวมด้วยการใช้คำสั่ง DA จะได้ผลลัพธ์กลับมาที่แอกคิวมูเลเตอร์

ADD A, direct

จำนวนไบต์ : 2

การทำงาน : ทำการบวกค่าในแอกคิวมูเลเตอร์ เข้ากับข้อมูลในหน่วยความจำ ข้อมูลภายใน แล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

ADD A, Rn

จำนวนไบต์ : 1

การทำงาน : ทำการบวกค่าในแอกคิวมูเลเตอร์เข้ากับข้อมูลในรีจีสเตอร์ R0-R7 ขนาด 8 บิต แล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

ADD A, @Rn

จำนวนไบต์ : 1

การทำงาน : ทำการบวกค่าในแอกคิวมูเลเตอร์เข้ากับข้อมูล 8 บิต ในแอดเดรสของหน่วยความจำที่ถูกชี้โดย R0 หรือ R1 แล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

SUBB A, #data

จำนวนไบต์ : 2

การทำงาน : ทำการลบค่าในแอกคิวมูเลเตอร์ด้วยค่าของแฟลกทด (C) แล้วลบด้วยข้อมูล data ขนาด 8 บิต นำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

SUBB A, direct

จำนวนไบต์ : 2

การทำงาน : ทำการลบค่าในแอกคิวมูเลเตอร์ด้วยค่าของแฟลกทด แล้วลบด้วยข้อมูล ในหน่วยความจำ ข้อมูลภายใน นำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

SUBB A, Rn

จำนวนไบต์ : 1

การทำงาน : ทำการลบค่าในแอกคิวมูเลเตอร์ด้วยค่าของแฟลกทด แล้วลบด้วยข้อมูลในรีจีสเตอร์ R0-R7 ขนาด 8 บิต นำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

SUBB A, @Rn

จำนวนไบต์ : 1

การทำงาน : ทำการลบค่าในแอกคิวมูเลเตอร์ด้วยค่าของแฟลกทดแล้วลบด้วย ข้อมูลในหน่วยความจำที่ถูกชี้โดยR0 หรือR1 นำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

MUL AB

จำนวนไบต์ : 1

การทำงาน : ทำการคูณค่าในแอกคิวมูเลเตอร์ด้วยค่าในรีจีสเตอร์ B นำผลคูณไบต์ล่างเก็บไว้ใน แอกคิวมูเลเตอร์ และผลคูณไบต์บนเก็บไว้ในรีจีสเตอร์ B

DIV AB

จำนวนไบต์ : 1

การทำงาน : ทำการหารค่าในแอกคิวมูเลเตอร์ด้วยค่าในรีจีสเตอร์ B นำผลหารไบต์บนเก็บไว้ใน แอกคิวมูเลเตอร์ และเศษการหารไบต์ล่างเก็บไว้ในรีจีสเตอร์ B

INC A

จำนวนไบต์ : 1

การทำงาน : ทำการเพิ่มค่าในแอกคิวมูเลเตอร์ขึ้นหนึ่งค่า แล้วนำค่าที่เพิ่มขึ้นไปเก็บไว้ใน แอกคิวมูเลเตอร์

INC direct

จำนวนไบต์ : 2

การทำงาน : ทำการเพิ่มค่าของข้อมูลในหน่วยความจำข้อมูลภายในขึ้นหนึ่งค่า

INC Rn

จำนวนไบต์ : 1

การทำงาน : ทำการเพิ่มค่าของข้อมูลในรีจีสเตอร์ R0-R7 ขึ้นหนึ่งค่า

INC @Rn

จำนวนไบต์ : 1

การทำงาน : ทำการเพิ่มค่าของข้อมูลในหน่วยความจำข้อมูลภายในที่ถูกชี้ โดยรีจีสเตอร์ R0 หรือ R1 ขึ้นหนึ่งค่า

INC DPTR

จำนวนไบต์ : 1

การทำงาน : ทำการเพิ่มค่าของข้อมูลในรีจีสเตอร์ DPTR ขึ้นหนึ่งค่า

DEC A

จำนวนไบต์ : 1

การทำงาน : ทำการลดค่าในแอกคิวมูเลเตอร์ ลงหนึ่งค่า แล้วนำค่าที่ลดลงนี้ ไปเก็บไว้ใน รีจีสเตอร์ A

DEC direct

จำนวนไบต์ : 2

การทำงาน : ทำการลดค่าของข้อมูลในหน่วยความจำข้อมูลภายในลงหนึ่งค่า

DEC Rn

จำนวนไบต์ : 1

การทำงาน : ทำการลดค่าของข้อมูลในหน่วยความจำข้อมูลภายในลงหนึ่งค่า

DEC @Rn

จำนวนไบต์ : 1

การทำงาน : ทำการลดค่าของข้อมูลในหน่วยความจำข้อมูลภายในที่ถูกชี้ โดยรีจีสเตอร์ R0 หรือ R1 ลงหนึ่งค่า

DA A

จำนวนไบต์ : 1

การทำงาน : คำสั่งนี้ใช้ปรับค่าข้อมูลในรีจิสเตอร์ A ภายหลังการบวกเลขที่ใช้รหัส BCD(Binary Code Decimal) โดยคำสั่งนี้จะทำตามหลังคำสั่งบวก ADD หรือ ADDC ในกรณีที่เลข นำมาบวกเป็นเลขรหัส BCD ทั้งนี้เพื่อให้ผลลัพธ์ที่ได้จากการบวกถูกเปลี่ยนกลับเป็นค่าเลขรหัส BCD ด้วย โดยการทำงานของคำสั่งจะตรวจสอบค่าในรีจิสเตอร์ A ภายหลังกระทำคำสั่งบวก

3. กลุ่มคำสั่งทางตรรกศาสตร์หรือ แบบลอจิก ทำหน้าที่เกี่ยวกับการประมวลผลแบบ ลอจิกต่างๆ เช่น การ AND OR หรือ EX-OR ระหว่างข้อมูลในรีจิสเตอร์ A นั่นเอง โดยมีการใช้คำสั่งดังนี้ **CPL:**เป็นการใช้คำสั่งกลับค่าหรือคอมพลีเมนต์ ข้อมูลในแอกคิวมูเลเตอร์จะไม่มีผลใดๆ ต่อค่าของแฟลก หรือการอ้างถึงตำแหน่งแอดเดรสนั้นตามบิตนั้นๆ RL, RLC, RR, RRC, SWAP:ทั้ง 5 คำสั่งนี้เป็นคำสั่งในการทำงานการวนบิตบนตัวของแอกคิวมูเลเตอร์ซึ่ง **RL:**เป็นการวนบิตทางขวา,**RLC:** เป็นการทำการวนทางซ้ายผ่านบิตทด, **RRC:**เป็นการวนขวาผ่านบิตทด และ **SWAP:** เป็นการวนซ้ายสี่ครั้ง **ANL:**เป็นการ ADD กันทางตรรกศาสตร์ ระหว่างแหล่งกำเนิดสองโอเปอร์แรนด์ ซึ่งจะสั่งให้ทำงานในรูปแบบของตรรกศาสตร์ทางข้อมูลขนาดเป็นไบต์หรือบิต

ANL A, #data

จำนวนไบต์ : 2

การทำงาน : ทำการแอนด์ค่าของข้อมูลในแอกคิวมูเลเตอร์ กับข้อมูล data ขนาด 8 บิต แล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

ANL A, direct

จำนวนไบต์ : 2

การทำงาน : ทำการแอนด์ค่าของข้อมูลในหน่วยความจำข้อมูลภายในกับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ANL A, Rn

จำนวนไบต์ : 1

การทำงาน : ทำการแอนด์ค่าของข้อมูลในรีจีสเตอร์ R0-R7 กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ANL A, @Rn

จำนวนไบต์ : 1

การทำงาน : ทำการแอนด์ค่าของข้อมูลในหน่วยความจำข้อมูลภายในที่ถูกชี้ โดยรีจีสเตอร์ R0 หรือ R1 กับค่าในรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ANL direct, A

จำนวนไบต์ : 2

การทำงาน : ทำการแอนด์ค่าของข้อมูล ในหน่วยความจำข้อมูลภายในกับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

ANL direct, #data

จำนวนไบต์ : 3

การทำงาน : ทำการแอนด์ค่าของข้อมูล ในหน่วยความจำข้อมูลภายใน กับข้อมูล data ขนาด 8 บิต แล้วนำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

ANL C, bit

จำนวนไบต์ : 2

การทำงาน : ทำการแอนด์ค่าของข้อมูลในแฟลกทด กับค่าของข้อมูลในระดับบิตของรีจีสเตอร์ แล้วนำผลลัพธ์ไปเก็บไว้ในแฟลกทด

ANL C, /bit

จำนวนไบต์ : 2

การทำงาน : ทำการแอนด์ค่าของข้อมูลในแฟลกทดกับค่าคอมพลีเมนต์ของข้อมูลในระดับบิตของรีจีสเตอร์โดยข้อมูลของรีจีสเตอร์ไม่มีการเปลี่ยนแปลงจากนั้นนำผลลัพธ์ไปเก็บไว้ในแฟลกทด(ค่าคอมพลีเมนต์ คือค่าที่ตรงข้ามกับค่าของข้อมูล)

ORL A, #data

จำนวนไบต์ : 2

การทำงาน : ทำการออร์ค่าในแอกคิวมูเลเตอร์กับข้อมูล data ขนาด 8 บิตนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

ORL A, direct

จำนวนไบต์ : 2

การทำงาน : ทำการออร์ค่าของข้อมูลในหน่วยความจำข้อมูลภายในกับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ORL A, Rn

จำนวนไบต์ : 1

การทำงาน : ทำการออร์ค่าของข้อมูลในรีจีสเตอร์ R0-R7 กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ORL A, @Rn

จำนวนไบต์ : 1

การทำงาน : ทำการออร์ค่าของข้อมูลในหน่วยความจำข้อมูลภายใน ที่ถูกชี้โดยรีจีสเตอร์ R0 หรือ R1 กับค่าในรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ORL direct, A

จำนวนไบต์ : 2

การทำงาน : ทำการออร์ค่าของข้อมูลในหน่วยความจำข้อมูลภายใน กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

ORL direct, #data

จำนวนไบต์ : 3

การทำงาน : ทำการออร์ค่าของข้อมูลในหน่วยความจำข้อมูลภายใน กับข้อมูล data ขนาด 8 บิต นำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

ORL C, bit

จำนวนไบต์ : 2

การทำงาน : ทำการออร์ค่าของข้อมูลในแฟลกทด กับค่าของข้อมูลในระดับบิตของรีจีสเตอร์ แล้วนำผลลัพธ์ไปเก็บไว้ในแฟลกทด

ORL C, /bit

จำนวนไบต์ : 2

การทำงาน : ทำการออร์ค่าของข้อมูลในแฟลกทด กับค่าคอมพลีเมนต์ของข้อมูลในระดับบิต ของรีจีสเตอร์ โดยข้อมูลของรีจีสเตอร์ไม่มีการเปลี่ยนแปลง จากนั้นนำผลลัพธ์ไปเก็บไว้ในแฟลกทด

XRL A, #data

จำนวนไบต์ : 2

การทำงาน : ทำการเอ็กคลูซีฟ-ออร์ค่าในแอกคิวมูเลเตอร์กับข้อมูล data ขนาด 8 บิต นำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

XRL A, direct

จำนวนไบต์ : 2

การทำงาน : ทำการเอ็กคลูซีฟ-ออร์ค่าในหน่วยความจำข้อมูลภายใน กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

XRL A, Rn

จำนวนไบต์ : 1

การทำงาน : ทำการเอ็กคลูซีฟ-ออร์ค่าของข้อมูลในรีจีสเตอร์ R0-R7 กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

XRL A, @Rn

จำนวนไบต์ : 1

การทำงาน : ทำการเอ็กคลูซีฟ-ออร์ค่าของข้อมูล ในหน่วยความจำข้อมูลภายใน ที่ถูกชี้โดย รีจีสเตอร์ R0 หรือ R1 กับค่าในรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

XRL direct, A

จำนวนไบต์ : 2

การทำงาน : ทำการเอ็กคลูซีฟ-ออร์ค่าของข้อมูล ในหน่วยความจำข้อมูลภายในกับค่าของ รีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

XRL direct, #data

จำนวนไบต์ : 3

การทำงาน : ทำการเอ็กคลูซีฟ-ออร์ค่าของข้อมูล ในหน่วยความจำข้อมูลภายในกับข้อมูล data ขนาด 8 บิต นำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

CLR A

จำนวนไบต์ : 1

การทำงาน : ทำการเคลียร์ค่าของรีจีสเตอร์ A ให้เท่ากับ 00H

CPL A

จำนวนไบต์ : 1

การทำงาน :ทำการกลับสถานะของข้อมูลในรีจีสเตอร์ A ให้มีค่าตรงข้าม

RL A

จำนวนไบต์ : 1

การทำงาน : ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางซ้าย บิต 7 จะหมุนวนมายังบิต 0

RLC A

จำนวนไบต์ : 1

การทำงาน : ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางซ้ายผ่านแฟลกทดโดยบิต 7 จะหมุนไปยังแฟลกทดและข้อมูลของแฟลกทดเดิมจะหมุนเข้ามาในบิต 0

RR A

จำนวนไบต์ : 1

การทำงาน : ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางขวา บิต 0 จะหมุนวนมายังบิต 7

RRC A

จำนวนไบต์ : 1

การทำงาน : ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางขวาผ่านแฟลกทดโดยบิต 0 จะหมุนไปยังแฟลกทดและข้อมูลของแฟลกทดเดิมจะหมุนเข้ามาในบิต 7

SWAP A

จำนวนไบต์ : 1

การทำงาน : แลกเปลี่ยนข้อมูลระหว่างบิต 0-3 กับบิต 4-7 ของรีจีสเตอร์ A

4. กลุ่มคำสั่งแบบบูลีนหรือแบบบิต ซึ่งเป็นความสามารถของไมโครคอนโทรลเลอร์ MCS-51 ที่จะดำเนินการประมวลผลแบบบิต แทนที่จะเป็นข้อมูลทั้งไบต์เช่นปกติ โดยมีชุดคำสั่งที่จัดการโดยตรง ทุกคำสั่งจะเข้าถึงข้อมูลโดยตรงในระดับบิต โดยมีการบิตแอดเดรสได้ตั้งแต่ 00H - 7FH ในพื้นที่ 128 บิต หน่วยความจำข้อมูลภายในและบิตแอดเดรส 80H - FFH ในบริเวณกลุ่มรีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR)

ANL C, bit

จำนวนไบต์ : 2

การทำงาน : ทำการแอนด์ค่าของข้อมูลในแฟลกทด กับค่าของข้อมูลในระดับบิตของรีจีสเตอร์ แล้วนำผลลัพธ์ไปเก็บไว้ในแฟลกทด

ANL C, /bit

จำนวนไบต์ : 2,

การทำงาน : ทำการแอนด์ค่าของข้อมูลในแฟลกทดกับค่าคอมพลีเมนต์ของข้อมูลในระดับบิตของรีจีสเตอร์โดยข้อมูลของรีจีสเตอร์ไม่มีการเปลี่ยนแปลงจากนั้นนำผลลัพธ์ไปเก็บไว้ในแฟลกทด(ค่าคอมพลีเมนต์ คือค่าที่ตรงข้ามกับค่าของข้อมูล)

ORL C, bit

จำนวนไบต์ : 2

การทำงาน : ทำการออร์ค่าของข้อมูลในแฟลกทด กับค่าของข้อมูลในระดับบิตของรีจีสเตอร์ แล้วนำผลลัพธ์ไปเก็บไว้ในแฟลกทด

ORL C, /bit

จำนวนไบต์ : 2

การทำงาน : ทำการออร์ค่าของข้อมูลในแฟลกทด กับค่าคอมพลีเมนต์ของข้อมูลในระดับบิต ของรีจีสเตอร์ โดยข้อมูลของรีจีสเตอร์ไม่มีการเปลี่ยนแปลง จากนั้นนำผลลัพธ์ไปเก็บไว้ในแฟลกทด

CLR C

จำนวนไบต์ : 1

การทำงาน : ทำการเคลียร์ค่าของแฟลกทดให้เท่ากับ "0"

CLR bit

จำนวนไบต์ : 2

การทำงาน : ทำการเคลียร์ค่าของข้อมูลในบิตที่กำหนดให้เท่ากับ "0"

CPL C

จำนวนไบต์ : 1

การทำงาน : ทำการคอมพลีเมนต์ หรือกลับสถานะลอจิกของแฟลกทด

CPL bit

จำนวนไบต์ : 2

การทำงาน : ทำการคอมพลีเมนต์หรือกลับสถานะลอจิกของข้อมูลในบิตที่กำหนด

SETB C

จำนวนไบต์ : 1

การทำงาน : ทำการเซตค่าของแฟลกทดให้เท่ากับ "1"

SETB bit

จำนวนไบต์ : 2

การทำงาน : ทำการเซตค่าของข้อมูลในบิตที่กำหนดให้เท่ากับ "1"

5. กลุ่มคำสั่งในการกระโดดไปยังตำแหน่งต่างๆภายในโปรแกรม ซึ่งจะเปลี่ยนลำดับของการประมวลผลภายในโปรแกรมไปยังส่วนต่างๆแทนที่จะดำเนินการไปเป็นลำดับ ต่อเนื่องโดยที่คำสั่ง JMP จะแบ่งเป็น 3 ลักษณะ คือ SJMP, LJMP, AJMP ซึ่งในแต่ละคำสั่ง จะมีข้อแตกต่างของการกระโดดไปยังแอดเดรสไกลสุดที่ต่างกัน คำสั่ง JMP ซึ่งเป็นแบบโมนีซิก ที่สามารถจะใช้ได้โดยมีรายละเอียดการใช้งานของคำสั่งดังต่อไปนี้ **SJMP:**จะเป็นการกระโดดแบบการย้ายอันดับตำแหน่งของแอดเดรสตำแหน่งเดิมซึ่งจะสามารถกระโดดได้ -128 ถึง +127 ไบต์ **AJMP:**ลักษณะแบบนี้จะสามารถกระโดดได้ไกลสุดประมาณ 2 กิโลไบต์ ซึ่งจะใช้หน่วยความจำเพียง 2 ไบต์เท่านั้นในการกำหนด **LJMP:**ลักษณะแบบนี้จะสามารถกระโดดได้ไกลสุดประมาณ 64 กิโลไบต์ ซึ่งจะใช้หน่วย

ความจำเพียง 3 ไบต์เท่านั้นในการกำหนด **JMP @A+DPTR:**เป็นการควบคุมการกระโดดไปยังโปรแกรมที่ต้องการเฉพาะภายในส่วนต่างๆ

SJMP rel

จำนวนไบต์ : 2

การทำงาน : กำหนดให้ซีพียูมาทำงานยังแอดเดรสที่กำหนดด้วยค่าสัมพัทธ์ (rel)

AJMP addr11

จำนวนไบต์ : 2

การทำงาน : กำหนดให้ซีพียูมาทำงานยังแอดเดรสที่ระบุไว้ addr11 มีขอบเขต 2 กิโลไบต์ (000H-7FFH)

LJMP addr16

จำนวนไบต์ : 3

การทำงาน : กำหนดให้ซีพียูมาทำงานยังแอดเดรสที่ระบุไว้ addr16 มีขอบเขต 64 กิโลไบต์ (0000H-FFFFH)

JMP @A+DPTR

จำนวนไบต์ : 1

การทำงาน : กำหนดให้ซีพียูกระโดดไปยังแอดเดรส ของหน่วยความจำโปรแกรมในตำแหน่ง ที่ได้รับการกำหนดด้วยค่าของรีจีสเตอร์ A รวมกับค่าใน DPTR

NOP

จำนวนไบต์ : 1

การทำงาน : เป็นคำสั่งที่ทำให้เกิดการเลื่อนแอดเดรสไปหนึ่งแอดเดรส

CJNE A, direct, rel

จำนวนไบต์ : 2

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อค่า ของรีจีสเตอร์ A ไม่เท่ากับค่าในหน่วยความจำข้อมูล

CJNE A, #data, rel

จำนวนไบต์ : 2

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อค่า ของรีจีสเตอร์ A ไม่เท่ากับค่าของ data

CJNE Rn, #data, rel

จำนวนไบต์ : 3

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อค่า ของรีจีสเตอร์ R0-R7 ไม่เท่ากับค่าของ data

CJNE @Rn, #data, rel

จำนวนไบต์ : 3

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อข้อ มูลในหน่วยความจำที่ชี้โดยรีจีสเตอร์ R0 หรือ R1 ไม่เท่ากับค่าของ data

DJNZ Rn, rel

จำนวนไบต์ : 2

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อทำ การลดค่าของรีจีสเตอร์ R0-R7 ลงหนึ่งค่า แล้วผลลัพธ์ไม่เท่ากับ "0"

DJNZ direct, rel

จำนวนไบต์ : 3

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อทำ การลดค่าของข้อมูลในหน่วยความจำที่กำหนดลงหนึ่งค่า แล้วผลลัพธ์ไม่เท่ากับศูนย์

ACALL addr11

จำนวนไบต์ : 2

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง โปรแกรมย่อยซึ่งมีแอดเดรสอยู่ภายในขอบเขต สัมบูรณ์แบบใกล้ ซึ่งมีค่าเทท่ากับ 2 กิโลไบต์ (000H-7FFH) และจะกลับมายังโปรแกรมหลักก็ต่อเมื่อพบคำสั่ง RET

LCALL addr16

จำนวนไบต์ : 3

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง โปรแกรมย่อยซึ่งมีแอดเดรสอยู่ภายในขอบเขต สัมบูรณ์แบบใกล ซึ่งสามารถอ้างแอดเดรส ได้สูงสุด 64 กิโลไบต์ และจะกลับมายังโปรแกรมหลักก็ต่อเมื่อพบคำสั่ง RET

RET

จำนวนไบต์ : 1

การทำงาน : กำหนดให้ซีพียูกระโดดไปยังโปรแกรมย่อยกลับไปยังโปรแกรมหลักเป็นคำสั่ง สุดท้ายของทุกโปรแกรมย่อย ยกเว้นโปรแกรมย่อยบริการอินเตอร์รัปต์

RETI

จำนวนไบต์ : 1

การทำงาน : กำหนดให้วีพียูกระโดดออกจากโปรแกรมย่อยบริการอินเตอร์รัปต์กลับไปยังโปรแกรมหลักเป็นคำสั่งสุดท้ายของโปรแกรมย่อยการบริการอินเตอร์รัปต์

JB bit, rel

จำนวนไบต์ : 3

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อบิต ของรีจีสเตอร์ที่ทำการตรวจสอบเกิดการเซต ใช้ได้กับรีจีสเตอร์ที่สามารถเข้าถึงได้ในระดับบิต

JBC bit, rel

จำนวนไบต์ : 3

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อบิต ของรีจีสเตอร์ที่ทำการตรวจสอบเกิดการเซต ใช้ได้กับรีจีสเตอร์ที่สามารถเข้าถึงได้ในระดับบิต หลังจากกระโดดแล้วจะทำการเคลียร์บิตที่ทำการตรวจสอบนั้นให้เป็น "0"

JNB bit, rel

จำนวนไบต์ : 3

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อบิต ของรีจีสเตอร์ที่ทำการตรวจสอบไม่เกิดการเซต หรือกระโดดเมื่อบิตที่ทำการตรวจสอบนั้นเป็น "0" ใช้ได้กับรีจีสเตอร์ที่สามารถเข้าถึงได้ในระดับบิต

JNZ rel

จำนวนไบต์ : 2

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อค่า ของแอกคิวมูเลเตอร์ หรือรีจีสเตอร์ A ไม่เป็น "0"

JZ rel

จำนวนไบต์ : 2

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อค่า ของแอกคิวมูเลเตอร์ หรือรีจีสเตอร์ A เป็น "0"

JNC rel

จำนวนไบต์ : 2

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อค่า ของแฟล็กทด (C) ไม่เกิดการเซตหรือเป็น "0"

JC rel

จำนวนไบต์ : 2

การทำงาน : กำหนดให้ซีพียูกระโดดไปยัง แอดเดรสปลายทางตามค่าสัมพัทธ์ (rel) เมื่อค่า ของแฟลกทด (C) ไม่เกิดการเซตหรือเป็น "1"

6. โครงสร้างการอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์ MCS-51 ไมโครคอนโทรลเลอร์ MCS-51สัญญาณที่เข้ามาทำการอินเตอร์รัปต์ MCS-51 สามารถที่จะกำหนดเลือกเพื่อยินยอม (หรืออีนาเบิล : ENABLE) และห้าม (หรือดิสเอเบิล : DISABLE) ไม่ให้มีการอินเตอร์รัปต์แต่ละประเภทได้ โดยการกำหนดบิตของข้อมูลที่เกี่ยวข้องซึ่งมักจะอยู่ภายในรีจิสเตอร์ TCON และ SCON นอกจากนี้ยังมีตำแหน่งบิตภายในรีจิสเตอร์ IE (INTERRUPT ENABLE REGISTER) ซึ่งทำหน้าที่เสมือนกับเป็นสวิตซ์หลักที่เกี่ยวข้องกับสัญญาณอินเตอร์รัปต์ทั้งหมด หากว่ากำหนดไม่ให้เกิดการอินเตอร์รัปต์แล้วการกำหนดบิตเพื่อห้ามหรือยินยอมของแต่ละอินเตอร์รัปต์ก็จะไม่มีผลใดๆเกิดขึ้น ยังแสดงให้เห็นว่าสัญญาณอินเตอร์รัปต์แต่ละประเภทยังสามารถกำหนดระดับความสำคัญ (PRIORITY) ของการอินเตอร์-รัปต์ได้สองลักษณะ คือ ระดับความสำคัญสูงหรือต่ำ (HIGH OR LOW PRIORITY) กล่าวคือขณะที่กำลังประมวลผลอยู่ภายในส่วนของโปรแกรมย่อยบริการอินเตอร์รัปต์ของสัญญาณที่มีระดับความสำคัญต่ำอยู่ ก็อาจจะถูกขัดจังหวะให้ไปประมวลผลของสัญญาณอินเตอร์รัปต์ที่มีระดับความสำคัญสูงกว่า แต่หากว่าเป็นสัญญาณอินเตอร์รัปต์ที่มีระดับความสำคัญต่ำเช่นเดียวกันแล้ว ก็ต้องรอให้เสร็จสิ้นการประมวลผลที่ ดำเนินการอยู่ก่อน

7. การรีเซต โดยความหมายของการรีเซตเป็นการบังคับให้มีการเริ่มต้นใหม่อีกครั้งหนึ่ง ซึ่งมักจะกระทำโดยการกำหนดสภาวะของสัญญาณที่ขารีเซตของไอซี MCS-51 ให้เป็นระดับลอจิก ที่เหมาะสมเท่านั้น การรีเซตด้วยวิธีนี้ถือว่าเป็นการอินเตอร์รัปต์อย่างหนึ่งได้ แต่จะมีลักษณะต่างออกไปจากการอินเตอร์รัปต์ของสัญญาณนี้ได้ ซึ่งมีศัพท์เฉพาะเรียกว่า NON-MASKABLE INTERRUPT นอกจากนี้การดำเนินการของโปรแกรมก็แตกต่างออกไปด้วย โดยจะไม่มีการเก็บค่าของคำสั่งที่กำลังจะไปทำในลำดับต่อไปภายในรีจิสเตอร์ PC เมื่อมีการรีเซตเกิดขึ้นโปรแกรม จะถูกสั่งให้กระโดดไปยังแอดเดรส 0000 ทันที ซึ่งตำแหน่งนี้จะเป็นตำแหน่งเริ่มต้นของการทำงานของไมโคร-คอนโทรลเลอร์ MCS-51 เมื่อเริ่มจ่ายไฟให้กับระบบเมื่อใดก็ตามที่มีการรีเซตเกิดขึ้นค่าสภาวะต่างๆ ภายในไมโครคอนโทรลเลอร์จะถูกกำหนดกลับไปเป็นค่าเริ่มต้นใหม่อีกครั้ง

2.3 ความรู้เกี่ยวกับภาษาจาวา

ภาษาจาวาถือกำเนิดขึ้นจากความต้องการที่จะสร้างคอมไพเลอร์ภาษาคอมพิวเตอร์ซึ่งเป็นอิสระจาก Hardware รุ่นใดรุ่นหนึ่ง หรือยี่ห้อใดยี่ห้อหนึ่ง โดยจุดประสงค์แรกนั้นต้องการที่จะใช้เขียนโปรแกรม เพื่อควบคุมอุปกรณ์เครื่องใช้ไฟฟ้า และอุปกรณ์อิเล็กทรอนิคส์ต่างๆ เช่น เตาอบ,

เครื่องซักผ้า, โทรศัพท์, มือถือ, Set Top Box ของเคเบิ้ลทีวี ฯลฯ ด้วยเหตุนี้ โดยการนำของ James Gosling หัวหน้า กลุ่ม Green Group แห่งบริษัท Sun Microsystems จึงได้เริ่มโครงการพัฒนาภาษาดังกล่าวอย่างจริงจัง ในปี 1991 โดยขั้นแรกชื่อว่า ภาษา Oak แต่หลังจากที่ไม่ประสบความสำเร็จในการนำไปใช้งานตามความคิดริเริ่มดังกล่าว ประกอบกับบริษัท Sun Microsystems เริ่มมองเห็นความจำเป็นที่ต้องมีภาษาที่สร้างโปรแกรมบนเครื่องหนึ่งแต่สามารถนำไปใช้งานบนเครื่องใดๆ ก็ได้ (Write Once Run Anywhere) บริษัทจึงได้นำเอาภาษา Oak มาพัฒนาต่อให้เป็นภาษาแบบ Object Oriented จนได้ภาษา จาวาขึ้นในปี 1995 และปัจจุบันได้มีหลายบริษัทพัฒนาภาษา จาวาขึ้นมาเป็นผลิตภัณฑ์ของตัวเองทั้ง Visual J++ (Microsoft) Visual Cafe (Sysmantec) และ Jbuilder (Borland) เป็นต้น

2.3.1 จาวาเป็นภาษา Object Oriented

ในอดีตที่ผ่านมาหลายภาษาได้อ้างตัวว่าเป็น Object Oriented เช่น C++ แต่แท้จริงแล้วยังไม่มีภาระใดที่ทำให้ผู้ใช้ได้รับประโยชน์จากคุณสมบัติที่เป็น Object Oriented ของภาษาได้อย่างแท้จริง โดยเฉพาะอย่างยิ่งคุณสมบัติความเข้ากันได้ หรือทำงานได้ทุก Plateform ของภาษาแบบ Object Oriented ทั้งนี้เพราะแต่ละภาษามี Object Model ที่ไม่เอื้ออำนวยให้เกิดประสิทธิภาพสูงสุดดังกล่าวได้ แต่ภาษาจาวาได้ถูกออกแบบให้เป็นภาษาแบบ Object Oriented โดยแท้จริงโดยทุกส่วนของจาวา จะถือเป็น Object ทั้งหมด ไม่ว่า class, function หรือว่า array ล้วนแล้วแต่เป็นออบเจ็กต์ทั้งสิ้น จะมียกเว้นอยู่อย่างเดียวคือชนิดของข้อมูล ซึ่งถือเป็นออบเจ็กต์โดยรายละเอียดของชนิดของข้อมูล

2.3.2 ภาษา จาวาไม่ยึดติดกับ Platform (Platform Independent)

source code จาวาจะถูกคอมไพล์เป็น .class หรือชุดคำสั่ง JVM(Java Vitual Machine) ก่อน เมื่อมีการเรียกใช้จึงจะถูกดาวน์โหลดลงบนเครื่องลูกข่ายแล้วถูกแปลด้วย Interpreter อีกทีหนึ่งเพื่อให้มันทำงาน ดังนั้นเครื่องลูกข่ายใดๆ ก็ตามที่มี Virtual Processer ซึ่งเป็น Interpreter ที่ใช้แปลภาษา JVM อยู่จะสามารถเรียกใช้งานภาษาจาวาได้เลยไม่ต้องมีการคอมไพล์โปรแกรมภาษา

จาวาอีกครั้ง

2.3.3 ข้อดีของภาษาเชิงวัตถุ

- ภาษาเชิงวัตถุนั้นเราสามารถสร้างเพื่อเลียนแบบการทำงานของสิ่งต่างๆในโลกนี้ได้ดีกว่าและง่ายกว่ามาก โดยภาษา จาวานั้นสร้างขึ้นมาโดยใช้หลักการเขียนโปรแกรมแบบเดียวกับ

ภาษา C ทำให้ผู้เขียนภาษา จาวายิ่งมีพื้นฐาน C มาก่อน จะสามารถเขียน จาวาได้อย่างรวดเร็ว

- ภาษาเชิงวัตถุนั้นมีความสามารถในการนำกลับมาใช้ใหม่ได้ดีกว่าภาษาเชิงกระบวนการ เพราะผู้ใช้ไม่ต้องรู้กระบวนการ และโครงสร้างของข้อมูลในการผ่าน Parameter ในแต่ละฟังก์ชันเหมือนภาษาเชิงกระบวนการ
- จาวาไม่ใช้ Pointer และ จาวาไม่ให้ผู้เขียนโปรแกรมจับจองหรือปล่อยให้เนื้อที่หน่วยความจำ (Allocate/Deallocate) ด้วยตัวเองเพื่อป้องกันปัญหาที่จะเกิดขึ้นกับระบบคอมพิวเตอร์เมื่อมีการจอง (Allocate) หรือคืน (Deallocate) หน่วยความจำบ่อยๆ โดย จาวาจะจัดการเรื่องนี้ให้เอง
- โปรแกรมภาษาเชิงวัตถุ นั้นมีความกระทัดรัดเหมาะสมที่จะใช้งานบนอินเตอร์เน็ต เพราะใช้เวลาในการดาวน์โหลดโปรแกรมน้อยกว่า
- ภาษา จาวานั้นปลอดภัยจากการนำภาษาไปเขียนเป็นโปรแกรมไวรัสเพื่อเล่นงานเครือข่ายคอมพิวเตอร์

2.3.4 จาวาแอพเพลต

แอพเพลต คือโปรแกรมขนาดเล็กที่สร้างขึ้นด้วยภาษา จาวาสามารถถูกเรียกใช้งานบน เว็บบราวเซอร์ โดยผ่านทาง HTML Page โปรแกรมภาษาจาวาที่เป็นแอพพลิเคชั่นได้จะเป็น Standalone โปรแกรมที่สามารถทำงานได้โดยไม่ต้องอาศัยโปรแกรมอื่นสามารถควบคุมการดำเนินการของตัวเองได้แต่โปรแกรมที่เป็นแอพเพลตจะทำงานภายใต้ เว็บบราวเซอร์โดยไม่สามารถควบคุมการดำเนินการของตัวเองได้ทั้งหมด

โปรแกรมของ แอพเพลต เมื่อถูกคอมไพล์แล้วจะได้เป็นไฟล์ .class แต่จะไม่สามารถทำงานได้โดย จาวาinterpreter (Java.exe) เหมือนโปรแกรมที่เป็น แอพพลิเคชั่นได้ การเรียกให้แอพเพลต ทำงานนั้นจะต้องทำภายใน HTML Page เท่านั้นโดยใช้ แอพเพลต Tag และ แอพเพลตนั้นจะต้องทำงานภายใต้สภาพแวดล้อมของ บราวเซอร์ เช่น Netscape Navigator, Microsoft Internet Exporler หรือ HotJavaซึ่งทั้งหมดเป็น บราวเซอร์ ที่มี จาวาinterpreter อยู่ภายใน

2.3.5 Class Applet

Class Applet ซึ่งเก็บไว้ใน packagge java.applet ที่มากับ Compiler ใช้สำหรับสร้าง แอพเพลต ทุกโปรแกรมที่เป็น แอพเพลต ต้องมีประโยค import java.applet.*; เพื่อให้คอมไพเลอร์มองเห็นคลาสแอพเพลต ซึ่งภายในจะคลาสที่กำหนดคุณสมบัติต่างๆของ แอพเพลต เช่น init() ,start() ,stop() ,paint() เป็นต้น และประโยค import java.awt.*; เพื่อให้คอมไพเลอร์มองเห็นคลาสที่เรียกว่า

Abstract Window Toolkit (AWT) ซึ่งภายในคราสนี้จะรวบรวมคลาสเกี่ยวกับ graphic และ graphic user interface (GUI) ภายในคลาส แอพเพลต ถูกกำหนดไว้ใน java.applet package ดังนี้

```
public class applet extends Panel {
        public void destory();
        public appletContext getappletContext();
        public String getAppletInfo();
        public AudioClip getAudioClip(URL url)
        public AudioClip getAudioClip(URL url,String name)
        public URL getCodeBase();
        public URL getDocumentBase();
        public Image getImage(URL url);
        public Image getImage(URL url,String name);
        public Locale getLocale();
        public String getParameter(String name);
        public String [] [] getParameterInfo();
        public void init();
        public boolean isActive();
        public void play(URL url);
        public void play(URL url,String name);
        public void resize(int width,int height);
        public void resize(Dimention d);
        public final void setStub(appletStub stub);
        public void showStatus(String msg);
        public void start();
        public void stop();
}
```

Font

ใน AWT มีคลาส Font สำหรับใช้ในการจัดการกับรูปแบบของตัวอักษรที่จะวาดออกไป ทำให้ applet สามารถเลือกรูปแบบของตัวอักษรที่จะวาดออกไปโดยไม่จำเป็นจะต้องติดต่อกับระบบปฏิบัติการ คลาส Font มี methods จำนวนมากกำหนดไว้ดังนี้

```
public class Font extends Object implements Seializable {
```

```
public Font(String name , int style , int size);
public static final int BOLD;
public static final int ITALIC;
public static final int PLAIN;
protected String name ;
protected int size ;
protected int style ;
public static Font decode(String str) ;
public static Font getFont(String nm);
public static Font getFont(String nm, Font font);
public boolean equals (Object obj);
public String getFamily();
public String getName();
public FontPeer getPeer();
public int getSize();
public int getStyle();
public int hashCode();
public boolean isBold();
public booleane isItalic();
public boolean isPlain();
public String toString();
}
```

เราสามารถกำหนด Font ให้กับ Component หรือ Graphics จะได้ font ซึ่งจะทำให้ตัวอักษรที่วาดลงบน Component หรือ Graphic นั้นเป็นรูปแบบที่ต้องกำหนดตามลำดับแต่ที่ช่วงขณะใดขณะหนึ่งจะมี font ที่กำหนดให้กับสิ่งใดสิ่งหนึ่งเพียงรูปแบบเดียวเท่านั้น หากต้องการทราบว่า font ที่ถูกกำหนดไว้ในปัจจุบันนั้นเป็นรูปแบบใด ก็สามารถเรียกดูได้โดยใช้

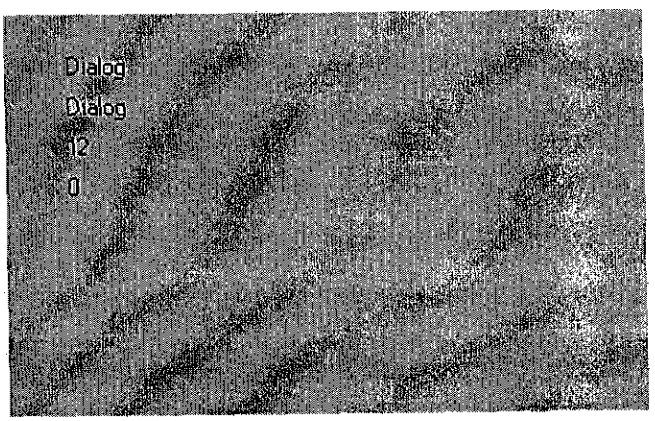
```
Public Font getFont()
```

ซึ่งมีการกำหนดไว้ในทั้งคลาส component และคลาส Graphics จะได้ font เป็นผลลัพธ์ ที่สามารถตรวจสอบได้โดยใช้ getFamily() getName() getSize() และ getStyle() ดังตัวอย่างต่อไปนี้

```
import java.awt.*;
import java.applet.*;
```

```
public class FontInfo extends Applet
{
        public void init()
        {
                repaint();
        }
        public void paint(Graphics g)
        {
                Font f = g.getFont();
                g.drawString(f.getFamily(),30,30);
                g.drawString(f.getName(),30,50);
                g.drawString(String.valueOf(f.getSize()),30,70);
                g.drawString(String.valueOf(f.getStyle()),30,90);
        }
}
```

ได้จะผลลัพธ์ที่ดังนี้



ตัวอย่างนี้เรียก getFont() ของ Graphics g จะได้ font เป็นรูปแบบของอักษรที่กำหนดไว้กับ g จากนั้นใช้ drawString () วาดผลลัพธ์ของ f.getFamily() f.getName() f.getSize และ f.getStyle() เปลี่ยนค่าของ size และ style เป็น String จะเห็นว่า font ที่เป็น default ของ Graphics g คือ Dialog ขนาด 12 ตัวธรรมดา Style เท่ากับ 0

- **Font Style**

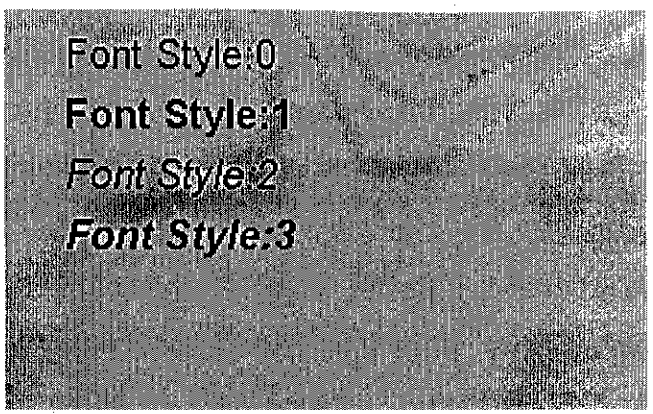
ใน Compiler ของ Microsoft visual J++ มี font style เพียงสี่แบบ ซึ่งกำหนดเป็นรหัสด้วยเลข 0 ถึง 3 เพื่อไม่ให้ผู้ใช้จำรหัสเหล่านี้จึงมีการกำหนดเป็นค่าคงที่ไว้ในคลาส Font ดังนี้

Font.Plain	= 0	อักษรตัวธรรมดา
Font.Bold	= 1	อักษรตัวหนา
Font.Italic	= 2	อักษรตัวเอียง
Font.Bold+Font.Italic	= 3	อักษรตัวหนาและเอียง

ตัวอย่างแสดงการกำหนด font style ทั้งสี่แบบของ font courier

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet
{
        public void init()
        {
                repaint();
        }
        public void paint(Graphics g)
        {
                for(int i=0;i<4;i++)
                {
                        g.setFont(new Font ("courier",i,20));
                        g.drawString("Font Style:"+Integer.toString(i),30,i*30+30);
                }
        }
}
```

ผลลัพธ์ที่ได้เป็นดังนี้



paint() ในตัวอย่างนี้มีประโยค for ที่ทำซ้ำสี่ครั้งในแต่ละรอบเราจะใช้ค่า index iเป็นพารามิเตอร์ในการกำหนด font style จากนั้นวาดอักษรของ style นั้นออกมา

- **Dimension**

เมื่อวาดอักษรหรือข้อความในแบบ Graphics การจัดวางตำแหน่งของข้อความเทียบกับพื้นที่แสดงผลเป็นสิ่งสำคัญมาก ใน packege java.awt จึงมีคลาส Dimension สำหรับแสดงขนาดของพื้นที่สองมิติใดๆกำหนดไว้ดังนี้

```
public class Dimension extends Object implements Serializable{
public Dimension();
public Dimension(Dimension d);
public Dimension(int width , int height)
public int height;
public width;
public boolean equals(Object obj);
public Simension getSize();
public void setSize(Dimension d);
public void setSize(int width,int height);
public String toString();
}
```

ตัวแปร height และ width เก็บขนาดของความสูงและความกว้าง ทุกๆ compnent (ซึ่งหมายถึง applet ด้วย) จะมี instance ของคลาส Dimension เก็บขนาดของ component นั้นไว้ และเราสามารถเรียกดูได้โดยใช้

```
public Dimension getSize();
```

ซึ่งเป็น method ในคลาส Component ที่จะให้ instance ของคลาส Dimension ที่เก็บขนาดของ component นั้นในขณะนั้น ถ้า applet นั้นถูกเปลี่ยนแปลงขนาดไปและถ้าเรียก getSize() จะได้ขนาดของ applet ในขณะนั้น

ตัวอย่างแสดงการใช้คลาส FonMetrics และ Dimension วาดข้อความให้อยู่ตรงกลาง applet

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet
{
        public void paint(Graphics g)
        {
                Font f = new Font("TimesRoman",Font.PLAIN,36);
                g.setFont(f);
                FontMetrics fm = getFontMetrics(f);
                Dimension d = getSize(f);
                String s= "Hello";
                int x = (d.width - fm.stringWidth(s))/2;
                int y = (d.height +fm.getHeight())/2;
                g.drawString(s,x,y);
        }
}
```

ตัวอย่างนี้เป็นการสร้า font แบบ TimesRoman แบบ PLAIN ขนาด 36 แล้วกำหนด f นี้ให้เป็น font ที่จะใช้งานแล้วเรียก getFontMetrics() ของ grapics g โดยส่ง f ให้เป็นพารามิเตอร์ ได้ FontMetrics เป็นผลลัพธ์ ต่อมาเรียก getSize() เพื่อให้ได้ Dimension d ซึ่งเป็นขนาดของ applet ในขณะนั้น แล้วคำนวณตัวแปร x ซึ่งเป็นค่าของความยาวของ applet ลบด้วยความยาวของข้อความแล้วหารด้วยสอง จะได้จุดตรงกลางซึ่งจะวาดข้อความลงไปและตัวแปร y คือจุดกึ่งกลางของ applet

Color

วิธีเก็บแสดงข้อมูลเกี่ยวกับสีในระบบ graphics นั้นมีหลายวิธี แต่ที่นิยมใช้ในปัจจุบันมีสองวิธีคือแบบ RGB (Red Green และ Blue)ที่เป็นสีหลักของแสง และแบบ HRS(Hue Saturation และ Brightness) เป็นสีแบบสีรุ้ง ภาษา java สนับสนุนการเก็บค่า

และการแสดงสีแบบ RGB โดยตรงแต่ก็มี method ที่ช่วยให้สามารถมีการใช้งานในแบบ HSB ได้

- **RGB**

ในระบบ RGB ข้อมูลของสีหนึ่งจุดจะถูกเก็บด้วยสามส่วนหลัก ซึ่งประกอบด้วยส่วนของสีแดง เขียว และ น้ำเงินซึ่งจะเป็นสีหลักในการผสมสีที่ต้องการภาษาจาวามีการเพิ่มส่วนประกอบที่สี่คือค่าความโปร่งใส (transparency) ดังนั้นข้อมูลของสีหนึ่งจุดในภาษา Java จะถูกเก็บด้วยหน่วยความจำ 4 byte หรือเป็นตัวแปร int หนึ่งตัว

หากมองข้อมูล 4 byte เป็น 32 bit และนับ bit ขวาสุดเป็น bit ที่ 0 ค่าความโปร่งใสจะอยู่ระหว่าง bit ที่ 24 ถึง 31 ค่าของสีแดงจะอยู่ระหว่าง bit ที่16 ถึง bit ที่ 23 ค่าของสีเขียวจะอยู่ระหว่าง bit ที่ 8 ถึง 15 และค่าของสีน้ำเงินอยู่ระหว่าง bit ที่ 0 ถึง 7 ค่า ค่าของสีแต่ละค่ามีพื้นที่หน่วยความจำ 8 bit สามารถบอกความละเอียดได้ 256 ค่า

ค่าความโปร่งใสที่สูงที่สุดคือ 255 จะหมายถึงทำให้เห็นสีได้อย่างชัดเจน ไม่เห็นสีของ background เมื่อลดค่าความโปร่งใสลง จะทำให้สีจางลงเรื่อยๆจนค่าความโปร่งใสต่ำสุดคือ 0 จะหมายถึงโปร่งแสง ทำให้เห็นเพียงสีของ background ค่าของความโปร่งใสจะไม่มีผลต่อค่าของสีที่กำหนดโดยอีกสาม byte นั้น แต่เป็นค่าที่บอกว่าสีที่เป็น background นั้นจะสามารถทะลุผ่านจุดของสีนั้นมาให้เห็นประมาณเท่าใด ค่าความโปร่งใสจะไม่มีผลต่อสีของตัวอักษรหรือเส้นที่วาดไปในแบบ graphic จะมีเฉพาะในหารจัดการกัง image

ค่าสีแดง เขียวและน้ำเงิน ถ้ามีค่ามากหมายถึงมีความเข้มของสีนั้นมาก และหากมีค่าน้อยหมายถึงมีความเข้มของสีนั้นน้อยหากกำหนดค่าของสีทั้งสานนั้นมากสุดคือ 255 จะได้เป็นสีขาว ถ้ากำหนดให้มีค่าต่ำสุดคือ 0 จะได้เป็นสีดำ และหากกำหนดค่าของสีทั้งสามหลักนั้นแตกต่างไปก็จะได้สีที่แตกต่างไป

ระบบ graphic มักมีโครงสร้างและการทำงานที่แตกต่างกันอย่างมาก ภาษา Java จึงมีคลาส Color กำหนดไว้ใน package java.awt เพื่อให้โปรแกรมสามารถจัดการกับสีได้ด้วยวิธีเดียวกันไม่ว่าจะทำงานบนระบบใด ในคลาส Color มี field เก็บข้อมูลเกี่ยวกับสีและ method สำหรับจัดการกับสีในระบบนั้นๆทำให้ผู้ใช้ใช้งานได้โดยไม่จำเป็นต้องเข้าใจโครงสร้างของระบบนั้นๆ

คลาส Color มี constructors สามตัวดังนี้

```
public Color(int r,int g, int b);
public Color(int rgb);
public Color(float r,float g,float b);
```

สังเกตว่า constructor สามตัวของคลาส Color สร้างได้แต่ instance ที่เก็บค่าสีของ RGB เท่านั้น ไม่สามารถที่จะเก็บ instance ที่เก็บค่าสีแบบ HSB และไม่สามารถกำหนดค่าความโปร่งใสได้ นั่นหมายถึง instance ที่ได้จะมีค่าความโปร่งใสเป็น 255 (หรือทึบแสง) เสมอ

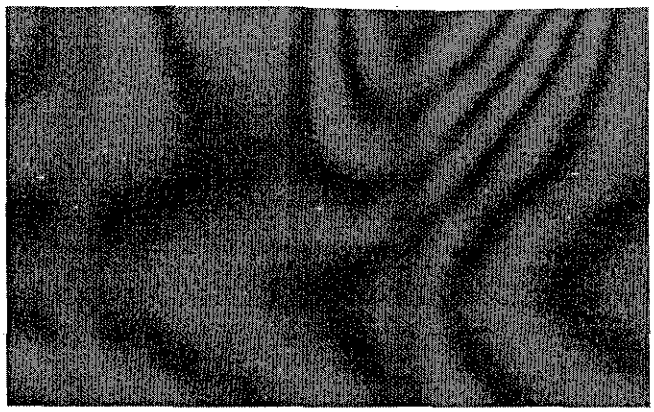
Constructor ตัวแรกรับพารามิเตอร์สามตัวที่เป็น int มีค่าอยู่ระหว่าง 0 ถึง 255 ซึ่งระบุค่าสีแดง เขียว และน้ำเงินที่เป็นส่วนประกอบของสีที่ต้องการนั้น ตามลำดับ constructor ตัวที่สองรองรับพารามิเตอร์ที่เป็น int เพียงตัวเดียว เป็นค่าของสีที่ผสมกันแล้ว ซึ่งสามารถกำหนดแบบเลขฐาน 16 ได้ โดยที่ค่าสีแดงจะอยู่ bit ที่ 16 ถึง 23 ค่าสีเขียงจะอยู่ระหว่าง bit ที่ 8 ถึง 15 ค่าสีน้ำเงินจะอยู่ระหว่าง bit ที่ 0 ถึง 7 ส่วน constructor ตัวที่สามรับพารามิเตอร์เป็น float มีค่าอยู่ระหว่าง 0.0 ถึง 1.0 ซึ่งระบุค่าสีแดง สีเขียว และสีน้ำเงินตามลำดับ

ตัวอย่างการสร้าง instance ของคลาส Color โดยใช้ constructor ตัวแรก

```
import java.awt.*;
import java.applet.*;

public class Applet1 extends Applet
{
        public void init ()
        {
                Color c=new Color(255,128,0);
                setBackground(c);
        }

}
```



ใน init() ของ applet นี้ เราสร้าง intstance ของคลาส Color ขึ้นโดยส่งพารามิเตอร์ให้สามตัวเป็น 255 128 0 ซึ่งหมายถึงค่าของสีแดงเป็น 255 สีเขียวเป็น 128 และสีน้ำเงินเป็น 0 แล้วนำไปกำหนดเป็นค่าของสี background โดยใช้ setBackgroune()

ตัวอย่างการผสมสีขึ้นเอง

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet implements Runnable
{
        Thread t = new Thread(this);
        public void init(){t.start();}
        public void run()
        {
                for (int r=0;r<256;r+=30)
                        for (int g=0;g<256;g+=30)
                                for (int b=0;b<256;b+=30)
                                {
                                        Color c = new Color(r,g,b);
                                        setBackground(c);
                                        repaint();
                                        showStatus(r+","+g+","+b);
                                        try {Thread.sleep(500);}
                                        catch (Exception e){}
                                }
```

```
                t.stop();
            }
        }
```

ตัวอย่างนี้แสดงการสร้าง instance ของคลาส Color แล้วนำมากำหนดเป็นค่าสีของ backgroune เช่นกัน แต่จะแสดงสีนั้นเพียงครึ่งวินาทีแล้วค่อยๆเปลี่ยนเป็นสีอื่น ดังนั้น applet นี้ต้องมี Thread t เพื่อที่จะได้มี run() ที่ทำการสร้างและกำหนดค่าของสีของ background แล้วเรียก repaint() ให้การเปลี่ยนสีนั้นปรากฏให้เห็น ภายใน method run() มีประโยก for สามประโยคซ้อนกันทำการเปลี่ยนแปลงค่า int r g และ b ที่จะเพิ่มขึ้นทีละ 30 จาก 0 ไปถึง 255 ในแต่ละรอบจะสร้าง Color c ขึ้นจากค่า r g และ b เพื่อจะทำให้สามารถให้เห็นค่าของสีต่างๆจึงใช้ showStatus() แสดงค่า int r ,g,b ของสีนั้นที่ status bar

- **Color Constants**

เพื่อความสะดวกในการใช้งานคลาส Color มีค่าของสี 13 สีที่ถูกใช้บ่อยๆกำหนดไว้ดังนี้

Public static final Color back, blue, cyan, darkgray, gray, green, lightgray, magent, orange, pink, red, white, yello;

หากไม่มีค่าเหล่านี้เราจะต้องสร้างสีขึ้นเอง โดยทดลองว่าสีที่ต้องการนั้นประกอบขึ้นจากสีทั้งสามเป็นปริมาณเท่าใดบ้าง ซึ่งเป็นการยากและใช้เวลา ค่าคงที่ของสีเหล่านี้จะช่วยให้ผู้ใช้ง่ายในการกำหนดค่าสีและยังช่วยให้ได้สีที่ตรงกันในการใช้แต่ละครั้ง

ตังอย่าง การใช้ค่าของสีในคลาส Color

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet implements Runnable
{
    Thread t = new Thread();
    public void init()
    {
        t.start();
    }
    public void run()
    {
```

```
Color [] c = {Color.black ,Color.blue ,Color.cyan ,Color.darkGray
,Color.gray ,Color.green ,Color.lightGray, Color.magenta, Color.orange ,Color.pink ,Color.red
,Color.white ,Color.yellow};
for(int i = 0;i<=13;i++)
{
setBackground(c[i]);
repaint();
try
{
Thread.sleep(500);
}
catch (Exception e)
{

}
t.stop();
}
}
}
```

ตัวอย่างนี้จะนำค่าของสีที่กำหนดในคลาส Color เก็บไว้เป็น array ในตัวแปร Color c แล้วนำมา setBackground() โดยใช้ประโยค for กำหนดค่าของ index i ใช้ Thread หน่วงค่าเวลาไปประมาณครึ่งวินาที ผลที่ได้คือ background จะเปลี่ยนเป็นสีต่างๆตามที่ได้กำหนดไว้

Drawing

คลาส Graphics มี method จำนานมากสำหรับวาดวัตถุ ซึ่งอาจจะเป็นอักษร ส้นหรือรูปร่าง method เหล่านี้ถูกกำหนดอยู่ในคลาส Graphics จึงใช้ วาดวัตถุลงใน instance ของคลาส Graphics เท่านั้น โดยทั่วไปจะสามารถนำ Graphics g มาสู่โปรแกรมมาใช้ในการวาดได้สองวิธีคือ

1.หากเป็นโปรแกรมประเภท applet จะมี Graphics g ที่ส่งมาจาก browser มาเป็นพารามิเตอร์สำหรับ method อย่างเช่น paint() และ update()

2.หากเป็นโปรแกรมประเภท application จะต้องมีการสร้าง frame ขึ้นเพื่อใช้เป็นที่แสดงภาพ และนำ Graphics มาโดยเรียก getGraphics() ของคลาส Component

การวาดภาพลงใน applet หรือ frame นั้นจะต้องมีการกำหนด Coordinate ของตำแหน่งของภาพที่จะวาด โดยมีแนวนอนเป็นแกน x วัดจากมุมบนด้านซ้ายจากซ้ายไปขวา แนวตั้งเป็นแกน y วัดจากมุมบนด้านซ้ายเช่นกันจากบนลงล่าง

- **String and Character**

การวาดตัวอักษรบน applet นั้นนอกจากจะใช้ drawString() แล้วยังมีอีกสอง method สำหรับวาดตัวอักษรลงไปใน Graphics g คือถ้ามีตัวอักษรเก็บอยู่ใน array ของ byte ก็ใช้ drawByte() และถ้ามีอักษรอยู่ในรูป array ของ char ก็ใช้ drawChar() ดังนี้

puvlic abstract void drawString(String str,int x ,int y)

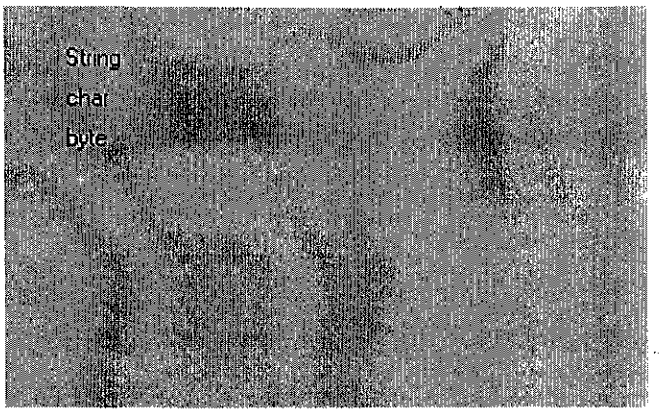
public void drawChars(char[] data,int off,int length,int x,int y)

public void drawByte(byte[] data, int off,int length ,int x ,int y)

กรณี drawChar() และ drawByte() จะมีพารามิเตอร์ตัวแรกเป็น array ของตัวอักษรที่จะถูกวาด มี int offset กำหนดตำแหนงเริ่มต้นของตัวอักษรใน array ที่จะถูกวาด และมี length เป็นจำนวนตัวอักษรนับจากตำแหน่งเริ่มต้นที่จะถูกวาดส่วน int x และ int y คือตำแหน่งใน Graphics g ที่จะเริ่มวาดตัวอักษรเหล่านั้น

ตัวอย่างการใช้ drawChar() และ drawByte()

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet
{
        String s = "String";
        char c[] = {'c','h','a','r'};
        byte b[] = {'b','y','t','e'};
        public void paint(Graphics g)
        {
                g.drawString(s,30,30);
                g.drawChars(c,0,4,30,50);
                g.drawBytes(b,0,4,30,70);
        }
}
```



โปรแกรมนี้เป็น applet ที่มี String s char c[] และ byte() ถูกกำหนดค่าไว้แล้ว มี paint() เรียก dcawString() เพื่อวาด String s เรียก drawChar() เพื่อวาด char c[] และ เรียก drawByte() เพื่อวาด byte[] b ลงใน Graphics g

- **Line**

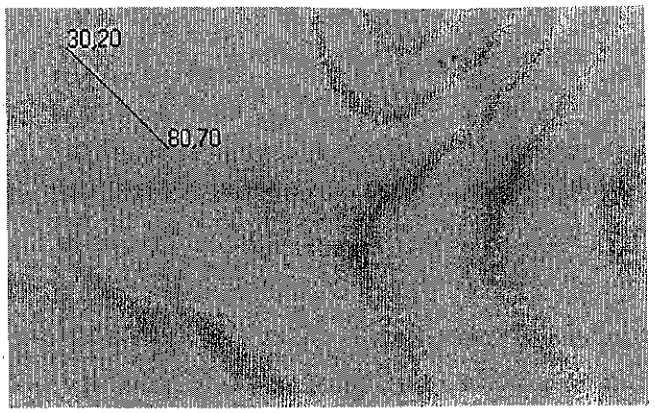
ในคลาส Graphics มี method สำหรับวาดเส้นตรงสำหรับตำแหน่งสองตำแหน่งที่กำหนด คือ

```
Public abstract void drawLine(int x1,int y1,int x2,int y2)
```

ซึ่งจะวาดเส้นจากตำแหน่ง x1,y1 ไปยัง x2,y2

ตัวอย่างการใช้ drawLine()

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet
{
        public void paint(Graphics g)
        {
                g.setColor(Color.red);
                g.drawLine(30,20,80,70);
                g.setColor(Color.black);
                g.drawString("30,20",30,20);
                g.drawString("80,70",80,70);
        }
}
```



โปรแกรมนี้เป็น applet ที่มี paint() กำหนดสีที่จะวาดเป็นสีแดงแล้วใช้ drawLine() วาดเส้นตรงจากตำแหน่ง 30,20 ไปถึงตำแหน่ง 80,70 แล้วเปลี่ยนสี่เป็นสีดำเพื่อใช้ drawString() ให้ตัวอักษรเป็นสีดำวาดค่าของตำแหน่งที่จุดทั้งสองด้วย

หากต้องการวาดเส้นตรงหลายเส้นต่อเนื่องกัน จะมี method ที่เพิ่มมาใน compiler คือ

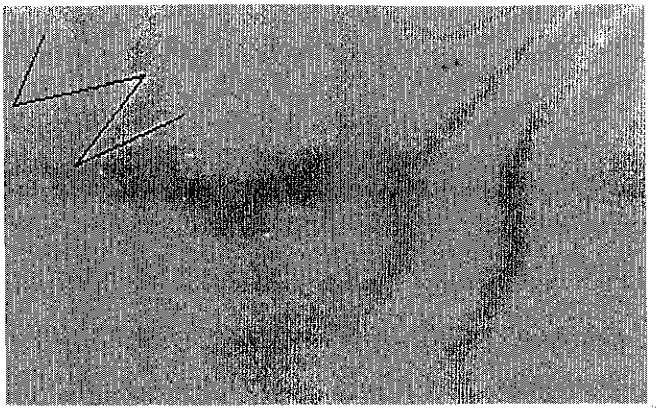
Public abstract void drawPolyline(int[] xP,int[] yP,int n)

ใช้สำหรับวาดเส้นตรงที่มีการผ่านจุดต่างๆที่ได้กำหนดเป็น array ใน int[] xP สำหรับแกน x และ int[] yP สำหรับแกน y ทั้งหมดเป็นจำนวน n จุด

ตัวอย่างการใช้ drawPolyline()

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet
{
        int [] x ={20,5,70,35,90};
        int [] y ={15,50,35,80,55};
        public void paint(Graphics g)
        {
                g.drawPolyline(x,y,y.length);
        }
}
```

โปรแกรมนี้เป็น applet ที่มี int[] x เก็บจุดในแนวนอน และ int[] y เก็บจุดในแนวตั้งที่จะวาดเส้นผ่านใน paint() เรียก drawPolyline() โดยมี int[] x และ int[] y และ y.length เป็นพารามิเตอร์นั่นคือจะวาดเส้นผ่านจุดห้าจุดคือ (20,15) (5,50) (70,35) (35,80) และ (90,55)



- **Rectangles**

เป็นการวาดรูปที่มีพื้นที่ปิด มีสองแบบคือ วาดเฉพาะเส้นขอบของรูปเท่านั้น และระบาสีพื้นที่ภายในรูปนั้น คลาส Graphics มี method สำหรับวาดรูปสี่เหลี่ยมได้สองแบบ ดังนี้

```
public void drawRect(int x,int y,int width ,int height)
public abstract void fillRect(int x,int y,int width,int height)
```

ทั้งสอง method จะวาดสี่เหลี่ยมที่มุมซ้ายบนที่ตำแหน่ง x,y มีความกว้าง(ไปทางขวา) width ความสูง(ไปข้างล่าง) height ต่างกันตรงที่ drawRect() จะวาดกรอบสี่เหลี่ยมธรรมดาส่วน fillRect() จะวาดรูปสี่เหลี่ยมที่ถูกระบายสีในพื้นที่ปิดของรูป มีอีก method หนึ่งคือ

```
public abstract void clearRect(int x,int y,int width,int height)
```

ซึ่งจะลบพื้นที่สี่เหลี่ยมออกจากตำแหน่ง x,y ขนาด width ,height คล้ายกับการวาดรูปสี่เหลี่ยม แต่เป็นการวาดด้วยสีที่เป็น background จึงใช้ในการลบภาพในพื้นที่ที่กำหนดออกไป นอกจากนี้ยังมีวิธีการวาดรูปสี่เหลี่ยมที่มีการเสริมขอบให้มองดูเหมือนกับเป็นรูปสามมิติสอง method คือ

```
public void draw3DRect(int x,int y,int width,int height,boolean raised)
public void fill3DRect(int x,int y,int width,int height,boolean raised)
```

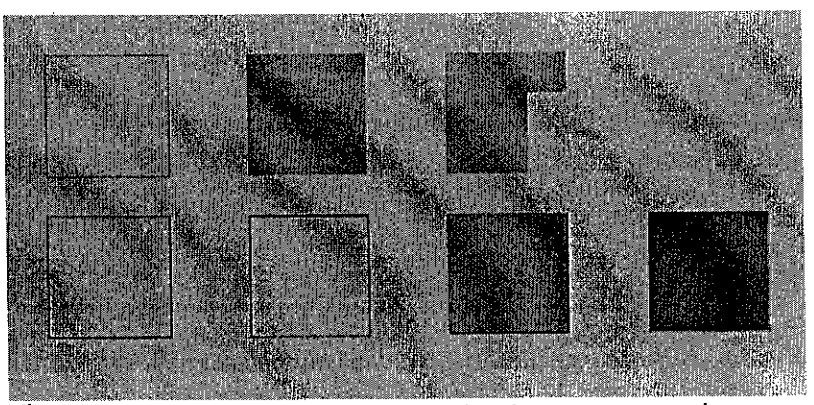
ทั้งสอง method คล้ายกับ method วาดรูปสี่เหลี่ยมที่กล่าวมาข้างบนนี้ แต่มีพารามิเตอร์ boolean raised เพิ่มขึ้นมา ซึ่งถ้าเป็น true จะได้สี่เหลี่ยมที่ดูเหมือนนูนขึ้นมา และถ้าเป็น false ก็จะได้สี่เหลี่ยมที่ดูเหมือนลึกลงไป

ตัวอย่างการวาดรูปสี่เหลี่ยมแบบต่างๆ

```
import java.awt.*;
import java.applet.*;


public class Applet1 extends Applet
```

```
{
        public void paint(Graphics g)
        {
                g.setColor(Color.green);
                g.drawRect(20,20,60,60);
                g.fillRect(120,20,60,60);
                g.fillRect(220,20,60,60);
                g.clearRect(260,40,60,60);
                g.draw3DRect(20,100,60,60,true);
                g.draw3DRect(120,100,60,60,false);
                g.fill3DRect(220,100,60,60,true);
                g.fill3DRect(320,100,60,60,false);
        }
}
```



ตัวอย่างนี้ใช้ drawRect() และ fillRect() วาดรูปสี่เหลี่ยมสองรูปแรกที่ตำแหน่ง 20,20 กับ 120,20 จะเห็นว่าได้รูปสี่เหลี่ยมที่มีแต่ขอบกับระบายสีพื้นตามลำดับ ต่อมาเพื่อแสดงการใช้ clearRect() จึงวาดสี่เหลี่ยมขึ้นที่ตำแหน่ง 220,20 ก่อนแล้วใช้ clearRect() ลบบางส่วนของรูปที่แล้วออกเป็นรูปสี่เหลี่ยม ที่ตำแหน่ง 260,40 จากนั้นแสดงการวาดรูปสี่เหลี่ยมแบบสามมิติโดยใช้ draw3D() วาดสี่เหลี่ยมสองรูปที่ตำแหน่ง 20,100 กับ 120,100 ต่างกันตรงที่มีพารามิเตอร์ตัวสุดท้ายเป็น true กับ false ตามลำดับได้รูปสี่เหลี่ยมที่มีการเน้นมุมให้เห็นคล้ายกับว่ายื่นออกมากับยุบลงไปตามลำดับ สุดท้ายแสดงการใช้ fillRect() ในทำนองเดียวกัน

- **Round Rectangles**

ในคลาส Graphics มี method สำหรับวาดรูปสี่เหลี่ยมที่มีมุมทั้งสี่โค้ง ดังนี้

public abstract void drawRoundRect(int x,int y,int w,int h,int arcwidth,int arcHeight)

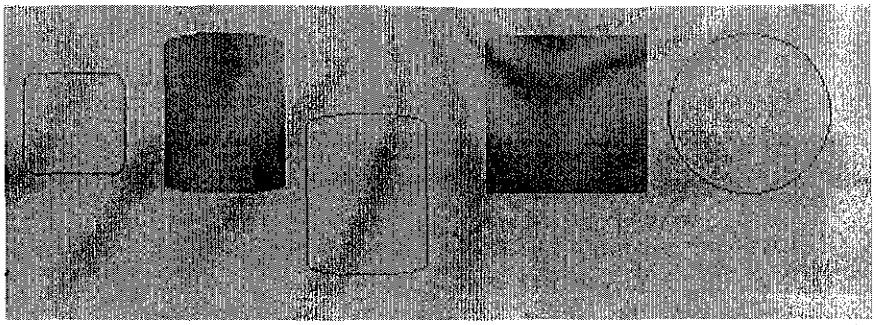
public abstract void fillRoundRect(int x,int y,int w,int h,int arcwidth,int arcHeight)

ทั้งสอง method มีการใช้งานคล้ายกับ method สำหรับวาดสี่เหลี่ยมดังที่กล่าวมาแล้วข้างบน แต่มีพารามิเตอร์เพิ่มขึ้นมาสองตัวคือ int arcWidth ระบุรัศมีความโค้งของแกน x และ int arcHeight ระบุรัศมีความโค้งของแกน y

ตัวอย่างการใช้ drawRoundRec() และ fillRoundRec()

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet
{
    public void paint(Graphics g)
    {
        g.setColor(Color.green);
        g.drawRoundRect(10,35,50,50,10,20);
        g.fillRoundRect(80,15,60,80,50,10);
        g.drawRoundRect(150,55,60,80,50,10);
        g.fillRoundRect(240,15,80,80,0,0);
        g.drawRoundRect(330,15,80,80,80,80);
    }
}
```

ตัวอย่างนี้แสดงการวาดสี่เหลี่ยมมุมโค้งห้ารูปโดยใช้ drawRoundRect() สลับกับ fillRoundRec() รูปแรก เป็นสี่เหลี่ยมมุมโค้งที่มีขนาด arcWidth และ arcHeight น้อยเปรียบเทียบกับ width และ height ตามลำดับ รูปที่สองมี arcWidth ค่อนข้างมากเปรียบเทียบกับ width ทำให้ด้านบนโค้งขึ้นทั้งด้านและด้านล่างโค้งลงทั้งด้านเป็นรูปวงรี รูปที่สามมี arcWidth และ arcHeight ค่อนข้างมากเมื่อเปรียบเทียบกับ width และ height ตามลำดับจึงได้รูปวงรี รูปที่สี่มี arcWidth และ arcHeight เป็นศูนย์ จึงได้เป็นรูปสี่เหลี่ยม และรูปสุดท้ายมีการกำหนด arcWidth และ arcHeight ให้เท่ากับ width และ height จึงได้เป็นวงกลมดังรูป



- **Ovals**

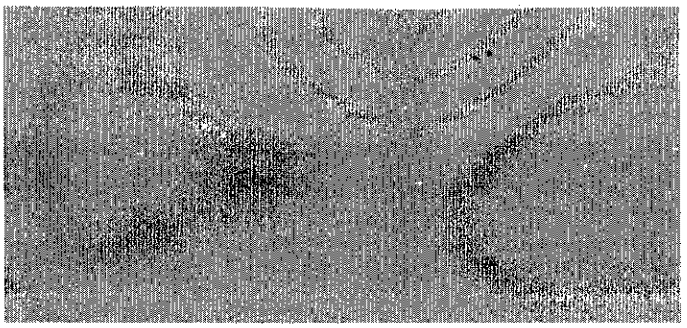
คลาส Graphics มี method สำหรับวาดรูปวงรีดังนี้

```
public abstract void drawOval(int x,int y,int w,int h)
public abstract void fillOval(int x,int y,int w,int h)
```

ทั้งสอง method นี้จะวาดรูปวงรีที่มี(จุดสมมุติ)มุมซ้ายบนอยู่ที่ตำแหน่ง x,y มีความกว้างเป็น w และความสูงเป็น h คลาส Graphics ไม่มี method สำหรับวาดรูปทรงกลมโดยตรงแต่สามารถใช้ method สำหรับวาดสี่เหลี่ยมมุมโค้งทำการวาดวงกลมได้ นอกจากนี้ยังสามารถใช้ method สำหรับวาดวงรีวาดวงกลมได้ โดยกำหนดให้ width มีขนาดเท่ากับ height

ตัวอย่างการวาดรูปวงรี

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet
{
        public void paint(Graphics g)
        {
                g.setColor(Color.orange);
                g.drawOval(10,15,100,100);
                g.drawOval(120,15,100,70);
                g.drawOval(240,15,70,130);
        }
```



}

ตัวอย่างการวาดรูปวงรีสามรูปโดยรูปแรกมี width เท่ากับ height จึงได้เป็นรูปวงกลม ร)ที่สองมี width มากกว่า height และรูปที่สามมี height มากกว่า width

- **Arcs**

ในคลาส Graphics มี method สำหรับวาดเส้นโค้งและพื้นที่ปิดระหว่างเส้นโค้งดังนี้

public abstract void drawArc(int x,int y,int width,int height ,int startAngle,int arcAngle)

public abstract void drawArc(int x,int y,int width,int height ,int startAngle,int arcAngle)

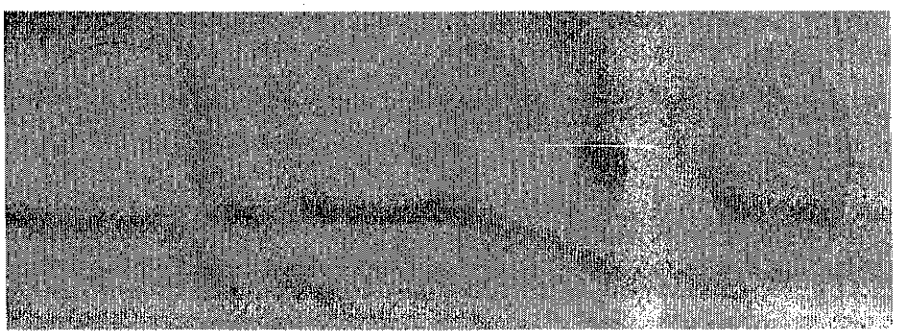
ทั้งสอง method จะวาดเส้นโค้งคล้ายกับเป็นส่วนเหนึ่งของวงรี ที่มุมซ้ายบนอยู่ที่ตำแหน่ง x,y มีความกว้างเป็น width และความสูงเป็น height เริ่มวาดจากมุม startAngle ไปเป็นจำนวน arcAngle องศา โดยที่มุมมีหน่วยเป็น drgree มีการกำหนดให้มุมที่ตำแหน่ง 3:00 นาฬิกาเป็นศูนย์องศา เส้นโค้งจะถูกวาดทวนเข็มนาฬิกาถ้า arcAngle มีค่าเป็นบวก และเส้นโค้งจะถูกวาดตามเข็มนาฬิกาถ้า arcAngle มีค่าเป็นลบ ผลของการวาดด้วย drawArc() จะได้แต่เส้นโค้ง ส่วน fillArc() จะได้ภาพที่เกิดจากการวาดรัศมีจากมุม startAngle ไปเป็นจำนวน arcAngle องศา กรณี fillArc() จากปลายเส้นโค้งทั้งสองจะวาดเส้นปิดไปยังจุดศูนย์กลาง

ตัวอย่างการใช้ drawArc() และ fillArc()

```
import java.awt.*;
import java.applet.*;

public class Applet1 extends Applet
{
    public void paint(Graphics g)
    {
```

```
        g.setColor(Color.cyan);
        g.drawArc(15,15,80,80,0,180);
        g.drawArc(100,15,80,40,0,110);
        g.fillArc(200,15,70,90,0,270);
        g.fillArc(260,15,50,100,0,-110);
        g.fillArc(350,15,70,120,0,360);
    }
}
```



ตัวอย่างการนี้ใช้ drawArc() วาดเส้นโค้งสองเส้น เส้นแรกเป็นเส้นโค้งของวงกลม(มี width และ height เท่ากัน) เริ่มจากมุม 0 องศาไปที่ 180 องศาทวนเข็มนาฬิกา เส้นที่สองเป็นเส้นโค้งวงรี (มี width เป็นสองเท่าของ height) เริ่มจาก 0 องศาไปที่ 110 องศาทวนเข็มนาฬิกา ต่อมาใช้ fillArc() วาดเส้นปิดระหว่างรูป สามรูป รูปแรกเป็นวงรีเริ่มจากมุม 0 องศาไปที่ 270 องศาทวนเข็มนาฬิกา รูปที่สองเป็นวงรีเริ่มจากมุม 0 ไปที่ 110 องศาตามเข็มนาฬิกา และรูปสุดท้ายเป็นวงรีเต็มรูปเพราะมีมุมที่กวาดไป 360 องศา

- **Polygon**

คลาส Graphics มี method สำหรับวาดรูปหลายเหลี่ยมคล้ากับ polyline แต่ polygon จะสร้างเส้นปิดระหว่างจุดแรกและจุดสุดท้ายให้ ดังนี้

```
public abstrac void drawPolygon(int[] xP,int[] yP,int n)
public void drawPolygon(Polygon p)
public abstrac void fillPolygon(int[] xP,int[] yP,int n);
public void fillPolygon(Polygon p)
```

มี method สำหรับวาดเฉพาะขอบและวาดระบายสีภายในรูปแต่ละแบบมีวิธีการระบุรูปหลายเหลี่ยมได้สองวิธีคือ แบบแรกมีพารามิเตอร์ int[] xP เป็น array ของมุมในแกน x กับ int Py

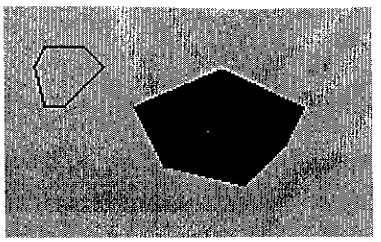
เป็น array ของมุมในแกน y และมี n เป็นจำนวนมุมของรูปหลายเหลี่ยมของรูปที่จะวาด แบบที่สอง มีพารามิเตอร์เป็น instance ของคลาส Polygon

ตัวอย่างการใช้วาดรูป polygon

```
import java.awt.*;
import java.applet.*;

public class Applet1 extends Applet
{
        public void paint(Graphics g)
        {
                int x[] = {20,40,50,30,20,15,20};
                int y[] = {20,20,30,50,50,30,20};
                g.drawPolygon(x,y,x.length);

                Polygon p =new Polygon();
                p.addPoint(65,50);
                p.addPoint(110,30);
                p.addPoint(150,50);
                p.addPoint(140,70);
                p.addPoint(120,90);
                p.addPoint(80,80);
                p.addPoint(65,50);
                g.fillPolygon(p);
        }
}
```



ตัวอย่างนี้แสดงการใช้ drawPolygon() วาดรูปหลายเหลี่ยม รูปแรกมีพารามิเตอร์ int[] x และ int[] y เก็บตำแหน่งของมุมในแกน x และแกน y ตามลำดับ และใช้ x.length ระบุจำนวนมุมของรูปหลายเหลี่ยมที่จะวาด ต่อมาแสดงการใช้ fillPolygon() ที่มี instance ของคลาส Polygon เป็นพารามิเตอร์ซึ่งก่อนนั้นจะต้องสร้าง Polygon p ขึ้นก่อน แล้ใช้ addPoint() กำหนดจุดให้แก่ p ที่ละจุด

Image

ใน java.applet package มีclass Image สำหรับเก็บข้อมูลที่เกี่ยวข้องกับไฟล์รูปภาพ และในคลาส แอพเพลต ก็มี method สำหรับ สร้าง reference ของคลาส Image จากไฟล์รูปภาพคือ

```
public Image getImage(URL url);
public Image getImage(URL url, String name);
```

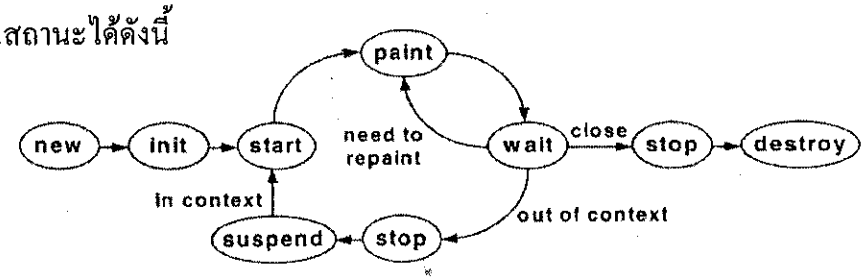
ซึ่งใช้งานคล้ายกับ getAudioClip() ที่กล่าวไปแล้ว การเรียก getImage() จะทำให้ข้อมูลในไฟล์รูปภาพที่ระบุนั้นถูกโหลดเข้าสู่ Instance ของคลาส Image ที่สร้างขึ้น ซึ่งจะถูกวาดไปบน instance ของคลาส Graphics ด้วย drawImage() ซึ่งเป็นmethod ในคลาส Graphics ที่ถูก overload ไว้หลายตัวมาก แต่ตัวที่เราจะใช้ทั่วไปคือ

```
Public abstract boolean drawImage(Image I ,int x, int y, ImageObserver obs);
```

โดยที่ Image i คือ intance ของคลาส Image ที่จะถูกวาด int x และ int y คือตำแหน่งที่จะวาดภาพนั้นลงไป ส่วน ImageObserver obs ส่วนใหญ่เราจะใช้ this เป็นพารามิเตอร์

2.3.6 แอพเพลต Life Cycle

ช่วงชีวิตของ แอพเพลต แบ่งออกเป็นสถานะต่างๆดังนี้ init, start, pain ,wait, suspended, stop, และ destroy ทุกๆ แอพเพลต จะดำเนินการผ่านสถานะเหล่านี้ โดยบางส่วนจะเป็นไปอย่างอัตโนมัติและบางส่วนจะอยู่ภายใต้การควบคุมของผู้ใช้ที่เรียกดู แอพเพลต นั้น ช่วงชีวิตของ แอพเพลต แสดงเป็นรูปการเปลี่ยนสถานะได้ดังนี้



รูปที่ 2.10 แอพเพลต Life Cycle

จากรูปเมื่อเริ่มต้น แอพเพลต ถูกสร้างโดยการ new() เหมือน instances ทั่วไปก็จะเริ่มทำงานทันที คือ

1. เข้าสู่สถานะ init แล้วเข้าสู่สถานะ start จากนั้นจะเข้าสู่สถานะ paint และเข้าสู่สถานะ wait ซึ่งจะหยุดรอเหตุการณ์ (Event) หรือสัญญาณจากผู้ใช้ (คืออินพุตเข้าทาง mouse , keybord หรือ graphics user interface) ที่เข้ามาทำให้เกิดการทำงานของ method ทำกำหนดไว้สำหรับจัดการกับเหตุการณ์นั้น เมื่อเสร็จแล้วก็จะเข้าสู่สถานะ wait เพื่อรอเหตุการณ์อื่นอีกต่อไป
2. ระหว่าง แอพเพลต อยู่ในสถานะ wait อยู่ หากผู้ใช้เปลี่ยนแปลง เว็บ page จนทำให้
   - พื้นที่แสดงผลของ แอพเพลต นั้นต้องมีการวาดรูปใหม่ เช่นเมื่อ แอพเพลต ถูกย้าย (move) เปลี่ยนขนาด (resize) ขยายขนาด (maximize) หรือ window อื่นมาทับ แอพเพลต แล้วนำ window นั้นออกไป บราวเซอร์ จะออกคำสั่งให้ แอพเพลต ทำการวาดอีกครั้งโดยเรียก repaint() ซึ่งจะส่งผลให้ แอพเพลต เข้าสู่สถานะ paint และทำการว่าพื้นที่แสดงผลของ แอพเพลต นั้นอีกครั้ง เมื่อเสร็จแล้วจะเข้าสู่สถานะ wait เพื่อรอเหตุการณ์อื่นอีกต่อไป
   - พื้นที่แสดงผลของ แอพเพลต นั้นไม่ปรากฏใน เว็บ page ปัจจุบัน เช่น เมื่อผู้ใช้เลื่อน page ไปดูหน้าอื่น หรือกดปุ่ม minimize จะทำให้ บราวเซอร์ ออกคำสั่งให้ แอพเพลต เข้าสู่สถานะ stop และไปสู่สถานะ suspended ซึ่ง แอพเพลต จะหยุดรอในสถานะนี้โดยไม่ยอมรับอินพุตจากผู้ใช้ และไม่ตอบโต้ใดๆจนกว่าพื้นที่แสดงผลของ แอพเพลต นั้นจะถูกทำให้กลับมาปรากฏอีกครั้ง บราวเซอร์ ก็จะออกคำสั่งให้ แอพเพลต เข้าสู่สถานะ start,paint และ wait เพื่อเริ่มทำงานอีก

3. ระหว่าง แอพเพลต อยู่ในสถานะ wait หากผู้ใช้กดปุ่ม Close ของ บราวเซอร์ เพื่อหยุดการทำงาน จะทำให้ แอพเพลต เข้าสู่สถานะ stop และ destoryแล้วสิ้นสุดการทำงาน

ข้อสังเกต เกี่ยวกับช่วงชีวิตของ แอพเพลต พอสรุปได้ดังนี้

- เมื่อเริ่มต้นทำงาน แอพเพลต จะเข้าสู่สถานะ init ,start , และ paint
- ตอนถูกทำลายก็จะเข้าสู่สถานะ stop และ destroy
- แอพเพลต หนึ่งจะผ่ายเข้าสู่สถานะ init และ destroy ครั้งเดียว นั่นคือผ่านการ init เมื่อตอนเริ่มต้นการทำงานครั้งแรกและเมื่อผ่านการ destroy แล้วจะไม่สามารถ กลับมาทำงานอีก
- แอพเพลต หนึ่งจะผ่านสถานะ start หรือ stop ได้หลายครั้งขึ้นอยู่กับการควบคุม ของผู้ใช้ที่ดู แอพเพลต นั้น

- แอพเพลต จะหยุดรอเหตุการณ์อยู่ในสถานะ wait หรือ suspended ได้เป็นเวลานานๆส่วนสถานะอื่นนั้น แอพเพลต จะผ่านเข้าไปทำงานแล้วผ่านออกไป ทันที

ในการสร้างโปรแกรมของ แอพเพลต เราสามารถกำหนดรูปแบบการทำงานในสถานะต่างๆ ที่กล่าวมา เมื่อ แอพเพลต ดำเนินการผ่านคลาส แอพเพลต ก็จะมี method ชื่อเหมือนกันกับสถานะคือ init(), start(), paint (), stop();และ destroy(); สำหรับกำหนดการทำงานเมื่อตอน แอพเพลต นั้นเข้าสู่สถานะนั้นๆ การสร้างคลาส แอพเพลต เราควรทำการ override methoe: init() , start() , paint(), stop(), และ destroy(), แต่ไม่ควรเขียนโปรแกรมที่เรียก method เหล่านี้โดยตรง แม้ว่าที่จริงแล้วสามารถทำได้ แต่โดยปรกติเราจะปล่อยให้ บราวเซอร์ เป็นผู้เรียก method เหล่านี้ เงื่อนไขที่แต่ละ method เหล่านี้จะถูกเรียกใช้มีดังนี้

- init() จะถูกเรียกเมื่อแอพเพลตถูกเรียกทำงานเป็นครั้งแรกและครั้งเดียวเท่านั้นจึงเหมาะสำหรับใช้ในการกำหนดค่าเริ่มต้นให้แก่ตัวแปรในแอพเพลตนั้น
- start() จะถูกเรียกหลังการทำงานของ init() เป็นการเริ่มทำงานตามที่เรากำหนดไว้ และจะถูกเรียกทุกครั้งเมื่อพื้นที่การแสดงผลของแอพเพลตถูกทำให้ปรากฏใน เว็บ page
- paint() โดยปรกติ paint() จะถูกเรียกภายหลังการทำงานของ start() และจะถูกเรียกเมื่อ บราวเซอร์ ต้องการวาดพื้นที่แสดงผลของ แอพเพลต นั้นใหม่ ซึ่งอาจเกิดขึ้นได้หลายกรณีเช่น พื้นที่แสดงผลของ แอพเพลต นั้นถูก window วาดทับ และเมื่อ window นั้นถูกนำออกไปแล้ว paint() ก็จะถูกเรียกให้ทำงาน หรือในกรณีที่ เว็บ page นั้นถูก minimize (หรือ maximize) และเมื่อถูกนำกลับมาอีก paint() ก็จะถูกเรียกเช่นกัน paint() มี Graphics g เป็นพารามิเตอร์ ซึ่งเป็นข้อมูลที่เกี่ยวข้องกับสภาวะแวดล้อมทาง graphics ของ แอพเพลต ที่กำลังทำงานอยู่ สังเกตว่า บราวเซอร์ จะมี Graphic g ซึ่ง paint() สามารถวาดลงใน Graphics g นี้แต่ในโปรแกรมของ แอพเพลต ไม่มีข้อมูล Graphics g นี้จึงไม่สามารถเรียก paint() ได้โดยตรง หากเราต้องการที่จะวาดใหม่ จะต้องเรียก repaint()
- stop() จะถูกเรียกเมื่อ แอพเพลต นั้นถูกเลื่อนไปหน้าอื่นไม่ให้มีพื้นที่การแสดงผลของ แอพเพลต นั้นปรากฏ หรือ เว็บ page นั้นถูก minimize เมื่อใดที่ แอพเพลต ถูก stop() จะทำให้ไม่มีการแสดงผลและไม่สามารถโต้ตอบกับผู้ใช้ได้ แต่ถ้า แอพเพลต นั้นมี threads อยู่ด้วยก็จะยังทำงานต่อไป

หากเราต้องการให้ threads เหล่านั้นหยุดทำงานด้วย ก็ต้องเรียก stop() ของ threads เหล่านั้นด้วย

- destroy() จะถูกเรียกเมื่อ แอพเพลต นั้นสิ้นสุดการทำงานแล้ว ซึ่งจะเกิดขึ้นเมื่อ เว็บ page นั้นถูกยกเลิกการทำงาน บราวเซอร์ จะเรียก stop() ก่อนทีจะเรียกdestroy() เมื่อ แอพเพลต ทำงาน destroy() พื้นที่ในหน่วยความจำและทรัพยากรของมันก็จะถูกเก็บกลับคืนไป ไม่สามารถถูก start() ได้อีก

2.3.7 Method ต่างๆใน java.applet

drawString()

เมื่อเราต้องการเขียนข้อความลงในพื้นที่ของ แอพเพลต จะใช้ method ของคลาส Graphics คือ

public void drawString(String msg, int x, int y);

โดย String msg จะเป็นข้อความที่ถูกพิมพ์ที่ตำแหน่งในแนวนอน int x และในแนวตั้ง int y ที่นับจากจุดบนซ้ายสุดเป็นตำแหน่ง 0,0 ข้อความที่ถูกพิมพ์ออกไปจะถูกแสดงเป็นแบบ Graphics ไม่สามารถใช้อักษรควบคุมเช่น '\t', '\n', หรือ '\r'

showStatus ()

บาง แอพเพลต มีข้อความที่ต้องแสดงตลอดเวลา หากเราวาดข้อความนั้นลงไปในพื้นที่ของแอพเพลต ด้วย drawString() ข้อความนั้นก็จะไปทับกับรูปภาพหรือข้อความอื่น class Applet จึงมี

public void showStatus(String mgs);

สำหรับแสดงข้อความ String mgs ลงบน status bar ที่อยู่ด้านล่างพื้นที่แสดงผลของ แอพเพลต

paint()

ดังที่ทราบแล้วว่า บราวเซอร์ จะเรียก paint() ให้ทำงาน เมื่อพื้นที่ของ แอพเพลต จะต้องถูกวาดใหม่ ผู้ที่จะโปรแกรมต้องเข้าใจว่า การเขียนโปรแกรมสำหรับ แอพเพลต เป็นความคิดในแบบ event driven programming คือ จะต้องมีเหตุการณ์ มากกระตุ้นให้ paint() ทำงานผู้โปรแกรมเพียงสร้างโปรแกรมของ paint() ที่จะถูกเรียกโดย บราวเซอร์ และไม่ควรเรียก paint() ให้ทำงานโดยตรง ถ้าเราต้องการให้ paint() ทำงานจากในโปรแกรมเราสามารถเรียก repaint() หรือ update() method เหล่านี้จะมาเรียก paint() ให้ทำงานอีกครั้ง ในการเขียน method paint() มีข้อพิจารณาดังนี้

- แม้เราทราบว่า paint() จะถูกเรียกอย่างอัตโนมัติหลังจาก start() แต่ก็อาจถูกเรียกให้ทำงานอีกโดยเรียก repaint() เราไม่อาจระบุเวลาหรือเงื่อนไขว่า paint() จะทำงานเมื่อใด รวมทั้งไม่อาจกำหนดว่า paint() จะถูกเรียกกี่ครั้ง

โปรแกรมของ paint() ควรพร้อมทำงานหลังจาก start() ทำงานแล้วเสมอ และถูกเรียกอีกเมื่อใด หรืออีกกี่ครั้งก็ได้

- การทำงานทั้งหมดภายใน paint() ใช้เวลานานเกินไป ที่สำคัญคือ ห้ามมีการทำซ้ำแบบไม่รู้จบ ไม่เช่นนั้นจะขาดการติดต่อกับ แอพเพลต นั้น
- ไม่ควรวาดภาพหลายๆภาพต่อเนื่อง (animation) ในการเรียก paint() หนึ่งครั้ง

repaint()

คลาส commponent มี repaint() ที่เมื่อมีการเรียกให้ทำงาน จะทำให้ paint() ถูกเรียกให้ทำงานด้วย ดังนั้นปรกติเราใช้ repaint() ในการวาดซ้ำ โดยสามารถเรียกได้จากในโปรแกรม เนื่องจาก repaint() ไม่มี Graphics g เป็นพารามิเตอร์ ในการใช้งาน repaint() มีข้อควรระวังดังนี้

- แม้ว่า repaint() จะถูกเรียกใช้เมื่อใดก็ได้ แต่อาจเกิดความผิดพลาดขึ้นได้ ถ้ายังไม่มีการกำหนดค่าเริ่มต้นให้เรียบร้อย ปรกติ paint() จะถูกเรียกให้ทำงานเป็นครั้งแรกหลังจากการทำงานของ init() และ start() เราจึงควรออกแบบ repaint() ให้ทำงานหลังจาก init() , start() และ paint() ทำงานไปแล้ว และหลังจากนั้นจะถูกเรียกเมื่อใด และกี่ครั้งก็ได้
- เนื่องจาก paint() จะถูกเรียกเป็นครั้งแรก หลังจาก init() และ start() ดังนั้นเราจึงไม่ควรเรียก repaint() ในส่วนของ init() หรือ start() มิเช่นนั้นแล้ว แอพเพลต จะทำงานผิดพลาด

update()

update() เป็น method ที่กำหนดอยู่ในคลาส Component และถ่ายทอดมาสู่คลาส แอพเพลต เช่นเดียวกับ paint() และ repaint() เมื่อเรียก repaint() แล้ว update() จะถูกเรียกอัตโนมัติก่อนที่จะเรียก paint() คลาส update() มี Graphics g เป็นพารามิเตอร์ เช่นเดียวกับ paint()

getDocumentBase() and getCodeBase()

บาง แอพเพลต จะต้องโหลดไฟล์ เช่นไฟล์เสียงไฟล์ภาพ จาก เว็บsite ที่ให้ Html page หรือ แอพเพลต นั้นมา คลาส แอพเพลต จึงมี getDocumentBase() ที่จะให้ URL ของไฟล์ Html page นั้น และ getCodeBase() จะให้ URL ของไดเรกทอรี ที่เก็บโปรแกรม .class ของ แอพเพลต นั้น

2.3.8 พื้นฐานภาษาจาวา

ตัวแปรและชนิดข้อมูล

ตัวแปร (Variable) จะเป็นชื่อของหน่วยความจำสำหรับใช้เก็บข้อมูลชนิดต่างๆ และสามารถนำออกมาใช้ได้ ตัวแปรจะประกอบด้วย

ชื่อ ,ชนิดข้อมูล ,และค่า

ก่อนใช้ตัวแปรต้องประกาศว่าเป็นตัวแปรชนิดข้อมูลอะไร แล้วจึงกำหนดค่าให้มัน จาวามีตัวแปรสามชนิดคือ

- ตัวแปรของ instance (instance variables) หรือตัวแปรของออปเจ็กต์
- ตัวแปรของคลาส (class variables)
- ตัวแปรประเภทโลคอล (local variables) หรือตัวแปรประเภทท้องถิ่น

การประกาศตัวแปร (Declaring Variables)

การใช้ตัวแปรใดๆ ในโปรแกรมของจาวาต้องประกาศก่อนที่จะใช้มัน การประกาศตัวแปรจะประกอบด้วยชนิดข้อมูลและชื่อตัวแปรดังนี้

```
String Name;            // ประกาศ Name เป็นตัวแปรชนิดสตริง
int Salary;             // ประกาศ Salary เป็นตัวแปรชนิดจำนวนเต็ม
int myAge, mySalary;    // ประกาศ myAge และ my Salary เป็นตัวแปรชนิดจำนวนเต็ม
int myAge=15;           // ประกาศ myAge เป็นตัวแปรชนิดจำนวนเต็มและให้มีค่า 15
```

การตั้งชื่อตัวแปร (Notes on Variable Names)

การตั้งชื่อตัวแปรในภาษาจาวามีกฎเกณฑ์ดังนี้

- เริ่มต้นตัวอักษรหรือ underscore (_) หรือเครื่องหมายดอลล่าร์ ($)
- ห้ามเริ่มชื่อตัวแปรด้วยตัวเลข
- ภายหลังที่ใช้อักขระเริ่มตัวแรกขอลตัวแปรแล้ว หลังจากนั้นสามารถใช้ตัวอักษรใดๆ หรือตัวเลขหรือสัญลักษณ์ต่างๆได้

การใช้อักษรตัวพิมพ์ใหญ่และอักษรตัวพิมพ์เล็กมีความแตกต่างกัน

ชนิดของตัวแปร (Variable Types)

ชื่อของตัวแปรแต่ละชนิดต้องระบุชนิดของตัวแปรเมื่อเราประกาศก่อนนำไปใช้เก็บข้อมูลแต่ละชนิด ชนิดของตัวแปรแบ่งเป็น 4 ประเภทใหญ่ๆ ดังนี้

- จำนวนเต็ม (integers) ได้แก่ byte, short, int, long
- เลขทศนิยม (floating point numbers) ได้แก่ floating, double
- อักขระ (characters) ได้แก่ char
- บูลีน (boolean) ได้แก่ boolean จะให้ค่า true หรือ false

นิพจน์และตัวดำเนินการ (Expressions and Operators)

นิพจน์ (Expressions) เป็นรูปแบบของคำสั่งแบบธรรมดาในภาษาจาวาซึ่งช่วยดำเนินการส่งค่ากลับ

ตัวดำเนินการ (operators) เป็นสัญลักษณ์พิเศษที่นำไปใช้ในนิพจน์

ตัวดำเนินการเลขคณิต (Arithmetic Operators)

ตัวดำเนินการ (operator)	ความหมาย	ตัวอย่าง
+	การบวก	1 + 5
-	การลบ	6 - 4
*	การคูณ	2 * 4
/	การหาร (Division)	12 / 4 (ผลหาร 3)
%	การหาร (Modulus)	15 % 7 (ผลหาร 1 คิดเฉพาะเศษ)

ตัวดำเนินการเปรียบเทียบ (Comparison Operators)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
==	เท่ากัน	X == 100;
!=	ไม่เท่ากัน	X != 9;
<	น้อยกว่า	X < 9;
>	มากกว่า	X > 2;
<=	น้อยกว่าหรือเท่ากับ	X <= 15;
>=	มากกว่าหรือเท่ากับ	X >= 20;

ตัวดำเนินการตรรก (Logical Operators)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
&&	AND	X > 2 && x < 10
||	OR	X > 10 || x < 2
!	NOT	! (x > 6)

ตัวดำเนินการบิตไวส์ (Bitwise Operators) ใช้ดำเนินการกับแต่ละบิตในจำนวนเต็ม

ตัวดำเนินการ	ความหมาย
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Left shift
>>	Right shift

>>>	Zero fill right shift
~	Bitwise complement
<<=	Left shift assignment (x = x << y)
>>=	Right shift assignment (x = x >> y)
>>>=	Zero fill right shift assignment (x = x >>> y)
X &= y	AND assignment (x = x & y)
X |= y	OR assignment (x = x | y)
X ^= y	XOR assignment (x = x ^ y)

อะเรย์ (Array)

อะเรย์เป็นกลุ่มของตัวแปรต่างๆมิติเดียวหรือหลายมิติเป็นวิธีการที่จะบรรจุรายการของออปเจ็กต์หรือชนิดข้อมูลดั้งเดิม แต่ละอะเรย์จะบรรจุแต่ละสมาชิกที่สามารถแทนค่าหรือเปลี่ยนแปลงได้ เราไม่สามารถบรรจุชนิดข้อมูลที่แตกต่างกันในเดียวกัน อะเรย์มีขั้นตอนการสร้างดังนี้

การประกาศตัวแปรต่างๆของอะเรย์ (Declare Array Variables)

String str[]; หรือ *String[] str;* หรือ *String []str;*

int temp[]; หรือ *int[] temp;* หรือ *int []temp;*

การสร้างออปเจ็กต์ของอะเรย์ (Crewting Arrays Objects)

```
String[] str;
Str = new String[10];
```

หรือ

```
String[] str = new String[10];
```

ตัวดำเนินการ new จะช่วยจองพื้นที่ในหน่วยความจำให้ออปเจ็กต์ของอะเรย์ str[] ใหม่ของสตริงพร้อมกำหนดให้ 10 ที่สำหรับบรรจุสมาชิกต่างๆ

การเข้าถึงสมาชิกต่างๆของอะเรย์ (Accessing Array Elements)

สามารถทดสอบและเปลี่ยนแปลงค่าในแต่ละห้องของอะเรย์นั้นๆการนำค่าออกมาจากภายในอะเรย์จะต้องใช้นิพจน์ของสับสคริป (subscript) ของอะเรย์ ดังนี้

*anyArray[*subscript*];*

anyArray เป็นส่วนหนึ่งของนิพจน์นี้ ซึ่งเป็นตัวแปรบรรจุออปเจ็กต์ของอะเรย์แม้ว่ามันสามารถเป็นนิพจน์ที่ให้ผลลัพธ์ในอะเรย์ นิพจน์นี้จะระบุห้องต่างๆในอะเรย์ที่จะเข้าถึง สับสคริป (subscript) ของอะเรย์จะเริ่มต้นด้วย 0 ดังนั้นเราสามารถเข้าถึงแต่ละสมาชิกของอะเรย์ได้โดยใช้สับสคริปที่เป็นเลขจำนวนเต็มระบุลำดับของสมาชิก

```
String[] myArr = new String[9];
```

myArr[0], myArr[1],..., myArr[8]

การเปลี่ยนสมาชิกต่างๆของอะเรย์ (Changing Array Elements)

numArray[1] = 14;

MyString[0] = "Hello";

MyString[10] = MyString[0];

อะเรย์หลายมิติ (Multidimensional Arrays)

int point[][] = new int[10][10]; // ประกาศตัวแปรอะเรย์ point พร้อมกำหนดหน่วยความจำบรรจุสมาชิกของอะเรย์จำนวน 100 สมาชิก

point[0][0] = 4; // กำหนดค่าให้สมาชิกของอะเรย์

คำสั่งบล๊อก (Block Statements)

คำสั่งบล๊อกเป็นกลุ่มของคำสั่งอื่นๆที่ล้อมรอบด้วยวงเล็บ({}) ท่านสามารถใช้บล๊อกกับคำสั่งต่างๆในโปรแกรม สามารถสร้างบล๊อกใหม่ของคำสั่งบรรจุในบล๊อกเดิมได้ นั่นคือ สามารถประกาศและใช้ตัวแปรแบบโลคอลภายในบล๊อกและตัวแปรต่างๆเหล่านั้นจะถูกทำลายเมื่อบล๊อกนั้นสิ้นสุดการประมวลผล

เงื่อนไขของ if (if Conditionals)

เงื่อนไขของ if เขียนอยู่ในรูปแบบต่อไปนี้

```
if (นิพจน์ที่เป็นเงื่อนไข)          // ถ้าเงื่อนไขแป็นจริง (true) จะทำคำสั่ง 1 แต่ถ้า
    คำสั่ง 1;                      // เป็นเท็จ (false) จะข้ามคำสั่ง 1 ไปทำคำสั่งถัดไป
```

หรือ

```
if (นิพจน์ที่เป็นเงื่อนไข)          // ถ้าเงื่อนไขแป็นจริง (true) จะทำคำสั่งที่อยู่ในบล๊อก
{                                  // แต่ถ้าเป็นเท็จ (false) จะกระโดดออกจากบล๊อก
    คำสั่ง 1;                      // ไปทำคำสั่งถัดไป
    คำสั่ง 2;
    ...
}
```

หรือ

```
if (นิพจน์ที่เป็นเงื่อนไข)
{
```

```
        คำสั่งบล๊อก1;                    // ถ้าเงื่อนไขแป็นจริง (true) จะทำคำสั่งที่อยู่ในบ
                                         ล๊อก1
    }
    else
    {
        คำสั่งบล๊อก2;                    // ถ้าเงื่อนไขแป็นเท็จ (false) จะทำคำสั่งที่อยู่ในบ
                                         ล๊อก2
    }
```

เงื่อนไข switch (Switch Conditionals)

คำสั่ง switch จะช่วยเลือกทำคำสั่งไดคำสั่งหนึ่งตามต้องการโดยมีทางเลือกให้ทำคำสั่งหลายๆทาง

```
    switch (oper)                        // เลือกทำงานตามค่าของ oper
    {       // เริ่มต้นคำสั่ง switch
       case '+' :
           add(val1,val2);               // กรณีที่ oper เป็น '+' ให้เรียกใช้ add() method
           break;                        // ยุติการประมวลผลแล้วข้ามไปที่คำสั่งถัดจาก
                                         // คำสั่ง switch
       case '-':
           sub(val1,val2);
           break;
       default:
           System.out.println("Unknown operator");   // เลือกทำคำสั่งนี้เมื่อ oper ไม่อยู่ใน
กรณีใดๆ
    }       // สิ้นสุดคำสั่ง switch
```

ลูป for (for Loops)

ช่วยทำคำสั่งหรือบล๊อกของคำสั่งที่ต้องการซ้ำแล้วซ้ำอีก จนพบเงื่อนไขที่กำหนดจึงออกจากลูป

```
    for ( ค่าเริ่มต้น; เงื่อนไข; ค่าเพิ่มขึ้นหรือค่าลดลง)
    {
        คำสั่งต่างๆ;
    }
```

การเริ่มต้นลูป for มี 3 ส่วนดังนี้

- ค่าเริ่มแรกของลูป for เป็นนิพจน์ เช่น i=0 หรือ int i=0 ตัวแปรที่ประกาศในส่วนนี้ของลูป for เป็นประเภทโลคอล จะถูกทำลายทิ้งเมื่อสิ้นสุดการประมวลผลของลูป
- เงื่อนไข (condition) จะเกิดขึ้นภายหลังที่ลูป for วนรอบทำงานในแต่ละครั้ง เงื่อนไขต้องเป็นนิพจน์หรือฟังก์ชันที่ให้ค่าชนิด boolean
- ค่าเพิ่มขึ้นหรือลดลง เป็นนิพจน์ใดๆ

การออกนอกลูป (Breaking Out of Loops)

ลูปทั้งหมดจะสิ้นสุดการทำงานตามเงื่อนไขที่กำหนดไว้ บางกรณีอาจต้องออกจากลูปในขณะที่ลูปกำลังวนรอบทำงาน สามารถใช้คีย์เวิร์ด

break; และ *continoue;*

การจัดการเหตุการณ์ธรรมดาและการโต้ตอบ

การกดปุ่มเมาส์และการปล่อยปุ่มเมาส์ (mouseDown and mouseUp)

ใน applet method ทั้งสองจะถูกเรียกใช้ทันทีที่เกิดเหตุการณ์ ต่อไปนี้เป็นรูปแบบของ mouseDown() method เมื่อเกิดเหตุการณ์ mouseDown

```
public boolean mouseDown(Event myEvent, int x, int y)
{
        // กิจกรรม
...
}
```

myEvent คือ เหตุการณ์ต่างๆของระบบทั้งหมดทำให้เกิด instance ของคลาส Event

x และ y คือ พิกัดของตำแหน่งที่เราคลิกเมาส์บนจอภาพ

mousUp() method ถูกเรียกใช้งานเมื่อผู้ใช้ปล่อยปุ่มของเมาส์ รูปแบบการเขียนจะคล้ายกับ mouseDown() method

```
public boolean mouseUp(Event myEvent, int x, int y)
{
        // กิจกรรม
...
}
```

ส่วนประกอบต่างๆสำหรับเชื่อมโยงกับผู้ใช้ขั้นพื้นฐาน (The Basic User Interface Components)

รูปแบบธรรมดาของส่วนประกอบของ AWT (Abstract Windowing Toolkit) ถือเป็นส่วนประกอบของ UI (User Interface) ขั้นพื้นฐานเราสามารถสร้างและเพิ่มส่วนประกอบของ UI ให้ applet ของเราโดยไม่ต้องทราบเกี่ยวกับการสร้าง containers หรือ panel ต่างๆ applet ของเราก่อนที่จะเขียนรูปและเชื่อมโยงเหตุการณ์ต่างๆ เนื่องจาก applet เป็น container เราจึงสามารถใส่ส่วนประกอบต่างๆ ของ AWT อื่นเช่นเดียวกับส่วนประกอบต่างๆของ UI หรือ container อื่นที่ใส่ให้ applet

เกี่ยวกับส่วนประกอบต่างๆของ UI ขั้นพื้นฐาน ได้แก่ ฉลาก (Label), ปุ่ม (Button), ฟิลด์ต่างๆของเท็กซ์ (text fields) ในแต่ละกรณี โปรแกรมย่อยจะ สร้างส่วนประกอบ (component) คล้ายกัน เริ่มแรกเราต้องสร้างส่วนประกอบแล้วเพิ่มมันเข้าไปให้ panel ที่ตรงนี้ มีมันจะแสดงบนจอภาพ การเพิ่มส่วนประกอบให้ panel เช่น applet จะต้องใช้ methode

Add()

ฉลาก (Label)

เป็นสตริงของเท็กซ์ (text) แบบธรรมดาที่อยู่ในชั้นของ panel ที่กำหนดให้ การสร้าง label ต้องใช้คอนสตรัคเตอร์ (constructor) ต่างๆดังนี้

- Label() ช่วยสร้างฉลาก (label) ที่ว่างเปล่าพร้อมกับเท็กซ์ชิดซ้ายมือตรงบริเวณที่กำหนดให้
- Label(String) ช่วยสร้างฉลาก (label) พร้อมสตริงของเท็กซ์ที่กำหนดให้ และเท็กซ์ให้ชิดซ้ายมือของพื้นที่กำหนดให้
- Label(String, int) ช่วยสร้างฉลาก (label) พร้อมสตริงของเท็กซ์ที่กำหนดให้ และการจัดรูปแบบของเท็กซ์ที่กำหนดให้ รูปแบบของเท็กซ์นี้บรรจุในตัวแปรต่างๆ ของคลาส Label ดังนี้

  Label.RIGHT, Label.LEFT, Label.CENTER

ปุ่ม (Button)

การสร้างปุ่ม (button) ต้องใช้คอนสตรัคเตอร์ (constructor) ต่างๆดังนี้

- Button () method ใช้สร้างปุ่มว่างเปล่าพร้อมกับไม่มี label
- Button (String) method ใช้สร้างปุ่มพร้อมออปเจ็กต์ของสตริงที่กำหนดเป็น label

เมนูสำหรับการเลือก (Choice Menus)

เมนูของคลาส choice เป็นส่วนประกอบของ UI ที่มีความสลับซับซ้อนมากว่า label, button เมนู (Menu) สำหรับเลือกเป็นประเภทเมนูแบบ popup (หรือ pulldown) การจัดทำเมนูเพื่อให้ผู้ใช้เลือกจะต้องสร้าง instance ของคลาส choice แล้วใช้ addItem() method เพื่อเพิ่มแต่ละรายการ

```
Choice choice = new Choice ();
choice.addItem("yes");
choice.addItem("No");
```

add(choice);

พื้นที่ของเท็กซ์ (Text Areas)

ใช้ดำเนินการกับเท็กซ์ขนาดใหญ่ได้ดี การสร้างพื้นที่ของเท็กซ์ เราต้องใช้ constructor ต่อไปนี้

- TextArea() ใช้สร้างพื้นที่ของเท็กซ์ว่างเปล่า มีขนาด 0 แถวและกว้าง 0 อักขระทำให้ไม่แสดงอะไรได้เลย แต่ไม่ต้องกังวล เนื่องจากเราสามารถเปลี่ยนแปลงขนาดของพื้นที่เท็กซ์ใหม่ได้ ก่อนที่จะเพิ่มมันให้ panel
- TextArea(int, int) ใช้สร้างพื้นที่ของเท็กซ์ว่างเปล่า พร้อมกับกำหนดจำนวนแถวและคอลัมน์
- TextArea(String) ใช้สร้างพื้นที่ของเท็กซ์สำหรับแสดงสตริงที่กำหนดให้ เริ่มที่ 0 แถวและ 0 คอลัมน์
- TextArea(String, int, int) ใช้สร้างพื้นที่ของเท็กซ์สำหรับแสดงสตริงที่กำหนดให้ พร้อมทั้งกำหนดขนาดของพื้นที่ตามต้องการ

TextComponent

Method	คำอธิบาย
apendText(String)	เพิ่มสตริงตรงท้ายสุดของเท็กซ์ในพื้นที่เท็กซ์
InsertText(String)	แทรกสตริงตรงตำแหน่งที่กำหนดให้ในเท็กซ์ (ตำแหน่งของเท็กซ์จะเริ่มต้นที่ 0)
replaceText(String, int, int)	การแทนที่เท็กซ์ระหว่างตำแหน่งจำนวนเต็มที่กำหนดให้ด้วยสตริงใหม่

บทที่ 3
การดำเนินงานและการพัฒนาโครงงาน

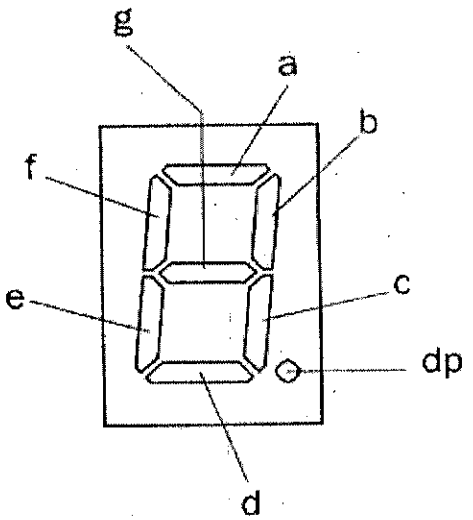
โครงงานสร้างบอร์ดจำลองการทำงานไมโครคอนโทรลเลอร์ MCS-51 เป็นการประยุกต์ใช้งานโปรแกรมโดยการใช้โปรแกรมภาษา Microsoft Visual J++ 6.0 ในการสร้างโครงงานโดยในการทำงานกับโปรแกรมภาษา Microsoft Visual J++ 6.0 นั้นได้เลือกใช้งานในส่วนของการทำงานกับ Webpage ซึ่งก็คือในส่วนที่เป็นApplet on HTML นั่นเอง ในการเลือกทำงานในส่วนแอพเพลตนั่นเนื่องจากต้องการให้บอร์ดจำลองที่ทำการสร้างขึ้นนั้นสามารถใช้งานได้โดยการเรียกผ่านโปรแกรม บราวเซอร์ ของ World Wide Web ซึ่งเป็นที่นิยมของบุคคลทั่วไปอย่างกว้างขวางและด้วยความสามารถของ จาวา ที่สนับสนุน แอพเพลต ต่างๆ ซึ่งเป็นโปรแกรมขนาดเล็กสามารถรันภายใน Web page และสามารถสร้างและการออกแบบ Web สำหรับโต้ตอบระหว่างผู้ใช้กับคอมพิวเตอร์ได้ โดยที่ บราวเซอร์ ที่ให้การสนับสนุนการทำงานของ จาวาแอพเพลต เช่น Hot Java Netscape Communicator, และ Microsoft Internet explorer เป็นต้น และจากที่ จาวา แอพเพลต สามารถเรียกใช้งานผ่าน บราวเซอร์ นั่นเองจึงเป็นความเหมาะสมต่อความสะดวกสบายต่อการใช้งานบอร์ดจำลองที่จะพัฒนา

3.1 พิจารณารูปแบบในการพัฒนาบอร์ดจำลองการทำงานไมโครคอนโทรลเลอร์MCS-51 ซึ่งได้

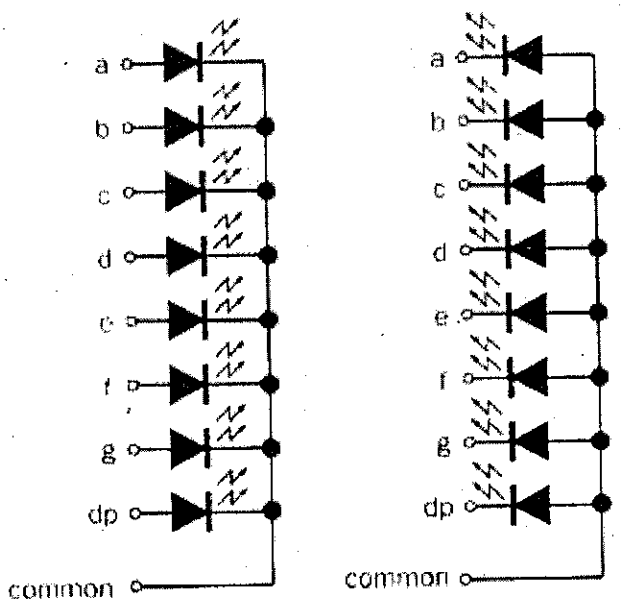
พิจารณาจากบอร์ดทดลองจริงซึ่งทำให้ได้รูปแบบที่เป็นพื้นฐานสำหรับการใช้งานไมโครคอนโทรลเลอร์ ซึ่งจะมีอุปกรณ์หลักที่สำคัญที่นำมาใช้บนบอร์ดจำลองได้แก่

- **7 - segment** LED ตัวเลข 7 ส่วนประกอบขึ้นจาก LED จำนวน 7 ตัวที่บรรจุอยู่ในตัวถังเดียวกันและได้รับการจัดเรียงเป็นรูปตัวเลข LED แต่ละตัวจะถูกเรียกว่า ส่วน หรือ เซกเมนต์ แต่ละ ส่วน หรือ เซกเมนต์ มีชื่อเรียกแตกต่างกันตามตำแหน่งที่ได้รับการจัดวาง คือ a, b, c, d, e, f, g ดังแสดงในรูป 3.1 ส่วน dp เป็น LED อีก 1 ตัวที่บรรจุอยู่ใน LED ตัวเลข 7 ส่วนนี้ใช้เป็นตัวเลขแสดงจุดทศนิยมในกรณีที่มีการแสดงผลในลักษณะที่มีจุดทศนิยม LED ทุกตัวที่บรรจุอยู่ใน LED ตัวเลข 7 ส่วนนี้มีขาต่อร่วมกันซึ่งก็มีทั้งแบบต่อขาแคโทดร่วมกันเรียกว่า แคโทดร่วม (common cathode) และแบบต่อขาแอโนดร่วมกัน(common anode) การขับให้ LED ตัวเลข 7 ส่วนแบบแคโทดร่วมสว่างจะต้องจ่ายไฟลบเข้าที่ขาร่วมแล้วจ่ายไฟบวกเข้าที่ขาแอโนดซึ่งก็คือขาของแต้ละเซกเมนต์นั่นเองดังแสดงในรูป 3.2 (ก) ในขณะที่ LED ตัวเลข 7 ส่วนแบบแอโนดร่วมจะต้องจ่ายไฟบวกเข้าที่ขาร่วมแล้วจ่ายไฟลบเข้าที่ขาแคโทดซึ่งเป็นขาของแต่ละเซกเมนต์ดังแสดงในรูป 3.2 (ข) LED ตัวเลข 7 ส่วนมีจำหน่ายทั้งแบบตัวเดี่ยว,ตัวคู่ และแบบที่มีมากกว่า 2 หลักแต่ที่นิยมใช้งานละหาได้ง่ายมี 2 แบบคือ แบบตัวเดี่ยวและแบบตัวคู่ โดยมีการจัดขาดังในรูปที่ 3.3 จะเห็นได้ว่า LED ตัวเลข 7 ส่วนตัวเดี่ยวมีขาต่อใช้งาน 10 ขาคือขา a, b, c, d, e, f, g, dp และขาร่วม (common) ซึ่งมี 2 ขา ถ้าเป็น LED ตัวเลข

7 ส่วนแบบตัวคู่มีขาต่อใช้งาน 20 ขาแบ่งเป็น a, b, c, d, f, g และ dp อย่างละ2 ขารวม 16 ขาและขาร่วมอีกหลักละ 2 ขาการต่อขาร่วมของแต่ละหลักทั้ง 2 ขานั้นสามารถต่อใช้งานเพียงขาเดียวได้ เนื่องจากโครงสร้างภายในของ LED ตัวเลข 7 ส่วนขาร่วมนี้ต่อถึงกันอยู่แล้ว

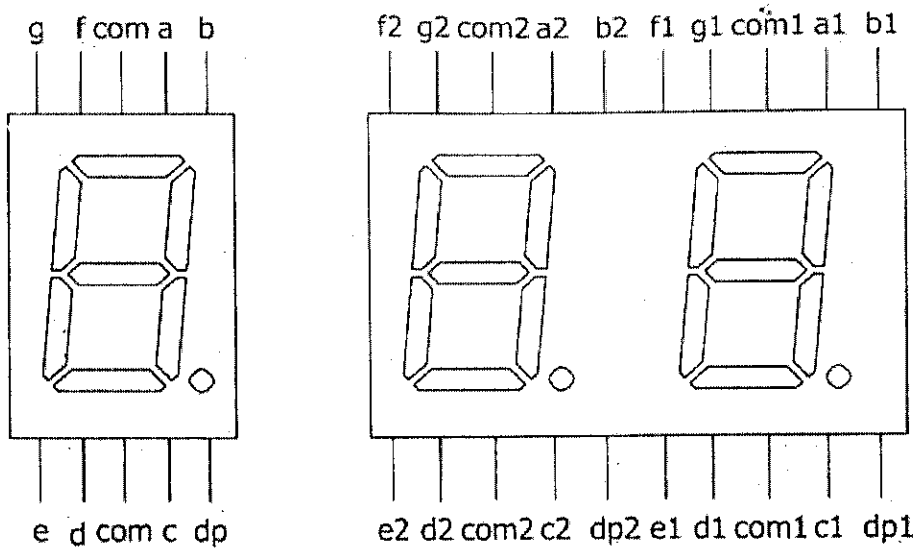


รูปที่ 3.1 ตำแหน่งต่างๆของ Segment



ก. แคโทดร่วม **ข. แอโนดร่วม**

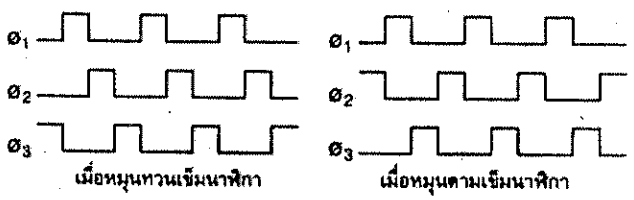
รูปที่ 3.2วงจรภายใน 7-Segment



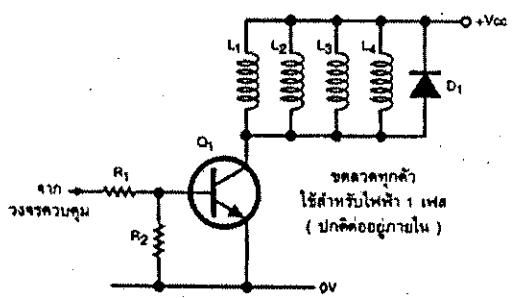
รูปที่ 3.3 แสดงการจัดขาของ7-Segment ทั้งแบบตัวเดี่ยวและคู่

- **ไดโอดเปล่งแสง** (light-emitting diode) เรียกย่อๆว่า LED คือไดโอดที่สามารถเปล่งแสงออกมาได้แสงที่เปล่งออกมาประกอบด้วยคลื่นความถี่เดียวและเฟสต่อเนื่องกัน ซึ่งต่างกับแสงธรรมดาที่ตาคนมองเห็น อันประกอบด้วยคลื่นซึ่งมีเฟส และความถี่ต่าง ๆ กัน มารวมกันไดโอดซึ่งสามารถให้แสงออกมาได้ทั้งชนิดที่เป็นสารกึ่งตัวนำของเหลว ก๊าซ ในที่นี้จะกล่าวถึงชนิดที่เป็นสารกึ่งตัวนำ ไดโอดชนิดนี้เหมือนไดโอดทั่วๆไปที่ประกอบด้วยสารกึ่งตัวนำชนิด P และ N ประกบกันมีผิวข้างกนึ่งเรียบเป็นมันคล้ายกระจก เมื่อไดโอดถูกไบแอสตรงจะทำให้อิเลคตรอนที่สารกึ่งตัวนำชนิด N มีพลังงานสูงขึ้นจนสามารถวิ่งข้ามรอยต่อไปรวมกับโฮลใน dP ก่อให้ เกิดพลังงานในรูปของประจุโฟตอนซึ่งจะส่งแสงออกมาไดโอดชนิดนี้เหมือนไดโอดทั่ว ๆ ไปที่ประกอบด้วยสารกึ่งตัวนำชนิด P และ N ประกบกันมีผิวข้างหนึ่งเรียบเป็นมันคล้ายกระจก เมื่อไดโอดตกไบแอสตรงจะทำให้อิเลคตรอนที่สารกึ่งตัวนำชนิด N มีพลังงานสูงขึ้นจนสามารถวิ่งข้ามรอยต่อไปรวมกับโฮลใน P ต่อให้เกิดพลังงานในรูปของประจุโฟตอนซึ่งจะส่งแสงออกมา การประยุกต์ LED ไปใช้งานอย่างกว้างขวางส่วนมากใช้ในภาคแสดงผล (display unit) LED โดยทั่วไปมี 2 ชนิดใหญ่ ๆ คือ LED ชนิดที่ตาคนเห็นได้กับชนิดที่ตาคนมองไม่เห็นต้องใช้ทรานซิสเตอร์มาเป็นตัวรับแสงแทนตาคน การใช้งานของ LED ที่เห็นได้บ่อย ๆ คือ ภาคแสดงผลของเครื่องคำนวณอิเลคทรอนิคส์สมัยใหม่ที่ใช้ LED ซึ่งมี 7 ส่วนแสดงเป็นตัวเลข ซึ่งบนบอร์ดจำลองที่พัฒนาขึ้นนั้นจะมี LED ทั้งหมด 8 ดวง

- **สเตปมอเตอร์** เป็นมอเตอร์ที่ทำงานโดยการป้อนพัลส์กระตุ้นให้มัน โดยใช้พัลส์ 1 ลูกต่อการเคลื่อนที่ไปหนึ่งเต็ป ซึ่งความละเอียดหนึ่งสเต็ปจะเป็น กี่องศานั้นขึ้นอยู่กับวงจรขับมอเตอร์ และโครงสร้างของสเต็ปมอเตอร์เอง ดังในรูปที่ 3.4 เป็นโครงสร้างของสเต็ปมอเตอร์ชนิดปรับค่า ความต้องการแม่เหล็กได้ (variable reluctance) โรเตอร์ของมอเตอร์ทำมาจากเหล็กอ่อน และมีจำนวนฟันไม่เท่ากับฟันของสเตเตอร์ ตัวสเตเตอร์ (หรือส่วนที่อยู่กับที่) จะมีกลุ่มขนลวดต่ออนุกรมกัน ซึ่งต้องมีวงจรขับแยกกัน ซึ่งต้องมีวงจรขับแยกกัน 3 กลุ่ม (หรือมากกว่า) สัญญาณที่นำมาขับนี้ เป็นสัญญาณลอจิก วงจรสเต็ปมอเตอร์แบบง่าย ๆ แสดงดังรูป 3.5 ส่วนสัญญาณที่ขับไปมีลักษณะของไทมิ่งไดอะแกรม ดังรูป ซึ่ง สามารถที่จะควบคุมทิศทางการหมุนของมอเตอร์ได้ด้วย ขนาดมุมของการ เคลื่อนที่ต่อหนึ่งสเต็ปของสเต็ปมอเตอร์ สามารถคำนวณได้จาก **N = S*R / S-R** โดยที่ N คือ ขนาดของมุมเมื่อมอเตอร์เคลื่อนที่ไป 1 สเต็ป มีหน่วยเป็นองศา โดยที่ S คือ จำนวนสล็อคของสเตเตอร์ โดยที่ R คือ จำนวนสล็อตของโรเตอร์ ปกติแล้วสเต็ปมอเตอร์จะมีขนาดมุมเมื่อเคลื่อนที่ไป 1 สเต็ปอยู่ 2 ขนาดคือ 108 องศาต่อสเต็ป และ 7.5 องศาต่อสเต็ป



รูปที่ 3.4 ไทมิ่งไดอะแกรมของสัญญาณที่ใช้ขับสเต็ปมอเตอร์



รูปที่ 3.5 ตัวอย่างวงจรขับสเต็ปมอเตอร์

- **Dipswitch** 8 ตัวต่อกับ Connector เมื่อเลื่อน switch ขึ้นจะให้ logic high ออกมา ถ้าเลื่อน switch ลงจะให้ logic low ออกมา

- **Push button** switch 8 ตัว ถูกต่อวงจรแบบ active high เมื่อกดที่ตัว switch จะให้ logic high ออกมา โดยปรกติถ้าไม่มีการกดปุ่มก็จะให้ logic low ออกมา

นอกจากเรื่องของรูปแบบของบอร์ดจำลองแล้วในเรื่องของรูปแบบในการประมวลผลและแสดงผลการทำงานของบอร์ดก็เป็นอีกส่วนหนึ่งที่ต้องการการระบุ ขอบเขตเพื่อใช้ในการพัฒนา โดยการสร้างบอร์ด, การประมวลผลและการ แสดงผลของบอร์ดนั้นจำเป็นที่จะสอดคล้องกับความต้องการที่ได้ระบุไว้โดยที่การทำงานของบอร์ดที่จะต้องทำการประมวลผลและแสดงผลของมาทาง Webpage โดยผ่านโปรแกรม บราวเซอร์ ในการแสดงผล

3.2 วางแผนในการพัฒนาโครงงานโดยจัดแบ่งเนื้อหาของแต่ละส่วน

การวางแผนการพัฒนาโครงงานได้เริ่มต้นจากการที่ได้ข้อสรุปถึงรูปแบบของบอร์ดจำลองที่จะทำการพัฒนาซึ่งจากรูปแบบที่ต้องการพัฒนานั้นจำเป็นต้องใช้เนื้อหาความรู้ที่จะใช้ในการพัฒนาโครงงานโดยแบ่งตามลักษณะของงานที่จะต้องทำเพื่อใช้พัฒนาเป็นสองส่วนหลักคือ

- เนื้อหาความรู้เกี่ยวกับไมโครคอนโทรลเลอร์ เนื้อหาความรู้ในส่วนนี้มีความสำคัญมากในการพัฒนาโครงงานเนื่องจากการที่จะพัฒนาโครงงานได้นั้นจำเป็นต้องมีความรู้ความเข้าใจในหลักการและโครงสร้างของไมโครคอนโทรลเลอร์ได้เป็นอย่างดี ไม่ว่าจะเป็นการทำงานของตัวไมโครคอนโทรลเลอร์เองหรือว่าการทำงานของไมโครคอนโทรลเลอร์ที่ใช้ในการควบคุมอุปกรณ์ต่างๆ
- การเขียนโปรแกรมเพื่อใช้ในการจำลองของบอร์ดไมโครคอนโทรลเลอร์ที่ต้องการพัฒนาซึ่งต้องใช้ความรู้ในการเขียนโปรแกรมในการทำงานและต้องเลือกที่จะใช้โปรแกรมภาษาที่มีความเหมาะสมกับรูปแบบการทำงานที่ได้ระบุไว้และโปรแกรมภาษานั้นต้องมีความสามารถในการรองรับความต้องการในการใช้งานบอร์ดที่จะพัฒนาได้อย่างค่อนข้างมีประสิทธิภาพและโปรแกรมภาษานั้นเมื่อทำการพัฒนาไปแล้วต้องสะดวกต่อการเรียกใช้งานของผู้ใช้

3.3 ทำการศึกษาเนื้อหาที่ได้ทำการจัดไว้แต่ละส่วน

จากการวางแผนการพัฒนาโครงงานที่ได้จัดแบ่งเนื้อหาในการศึกษาไว้ข้างต้นแล้วจึงทำการศึกษาเนื้อหาทั้งหมดแต่ละส่วนโดยแยกเป็น

3.3.1ความรู้ทางด้านไมโครคอนโทรลเลอร์

การศึกษาในด้านไมโครคอนโทรลเลอร์นั้นได้ทำการศึกษาถึงเรื่องต่างๆ ดังนี้

โครงสร้างและสถาปัตยกรรมของไมโครคอนโทรลเลอร์

- การจัดการหน่วยความจำในไมโครคอนโทรลเลอร์

- การเข้าถึงข้อมูลของไมโครคอนโทรลเลอร์
- การเขียนโปรแกรมภาษาแอสเซมบลี
- การเข้าถึงรีจิสเตอร์ทั้งแบบโดยตรงและโดยอ้อม
- การเข้าถึงหน่วยความจำแรมภายในไมโครคอนโทรลเลอร์
- การจัดการข้อมูลระดับบิต
- การคำนวณทางคณิตศาสตร์
- การกระทำทางลอจิก
- โปรแกรมลูปและการหน่วงเวลา
- การกระโดดและโปรแกรมย่อย
- รีจิสเตอร์ DPTR กับการติดต่อหน่วยความจำภายนอก

การทำงานของโปรแกรมควบคุมไมโครคอนโทรลเลอร์กับฮาร์ดแวร์เช่น LED 7 Segment, สเต็ปมอเตอร์ เป็นต้น

3.3.2 ความรู้ในการเขียนโปรแกรมภาษา (Microsoft Visual J++)

การศึกษาในด้านการเขียนโปรแกรมเพื่อใช้ในการสร้างบอร์ดจำลองจำเป็นต้องเข้าใจถึงโครงสร้างทางภาษาของภาษาจาวาที่มีการทำงานผ่าน บราวเซอร์ บนระบบอินเตอร์เน็ต การเรียกใช้งานโปรแกรมที่สร้างผ่าน บราวเซอร์ ได้อย่างไรซึ่งจากการศึกษาจะทราบว่า จาวา แอพเพลต นั้นมีการทำงานโดยโปรแกรมที่เขียนจะเป็นไฟล์ .class โดยเมื่อต้องกาทำงานจำเป็นต้องใส่ไฟล์ .html และ .class ไว้บน Web page เมื่อใช้ บราวเซอร์ จะทำการดาวน์โหลด แอพเพลต ไปไว้ที่ระบบและทำการประมวลผล ซึ่งจะทำให้สามารถโต้ตอบกับ แอพเพลต ต่างๆทั้งหมดได้และที่สำคัญภาษา จาวา เป็นภาษาแบบ OOP (Object-Oriented Programming) ซึ่งเป็นการเขียนโปรแกรมที่มีความสำคัญมากที่ใช้แนวคิดการทำงานเกี่ยวกับออปเจ็กต์โดยมองแต่ละส่วนของโปรแกรมเล็กๆเป็นออปเจ็กต์ซึ่งมีหน้าที่การทำงานเฉพาะละเมื่อนำส่วนเหล่านั้นมารวมกันจะได้แอพพลิเค่ชั่นที่มีขนาดใหญ่มาก สามารถนำไปใช้งานตามที่ออกแบบไว้โดยจะมีส่วนประกอบของการเขียนโปรแกรมหลักๆ ดังนี้คือ

1. **การแสดงภาพกราฟิกออกทาง paint(Graphics g)**

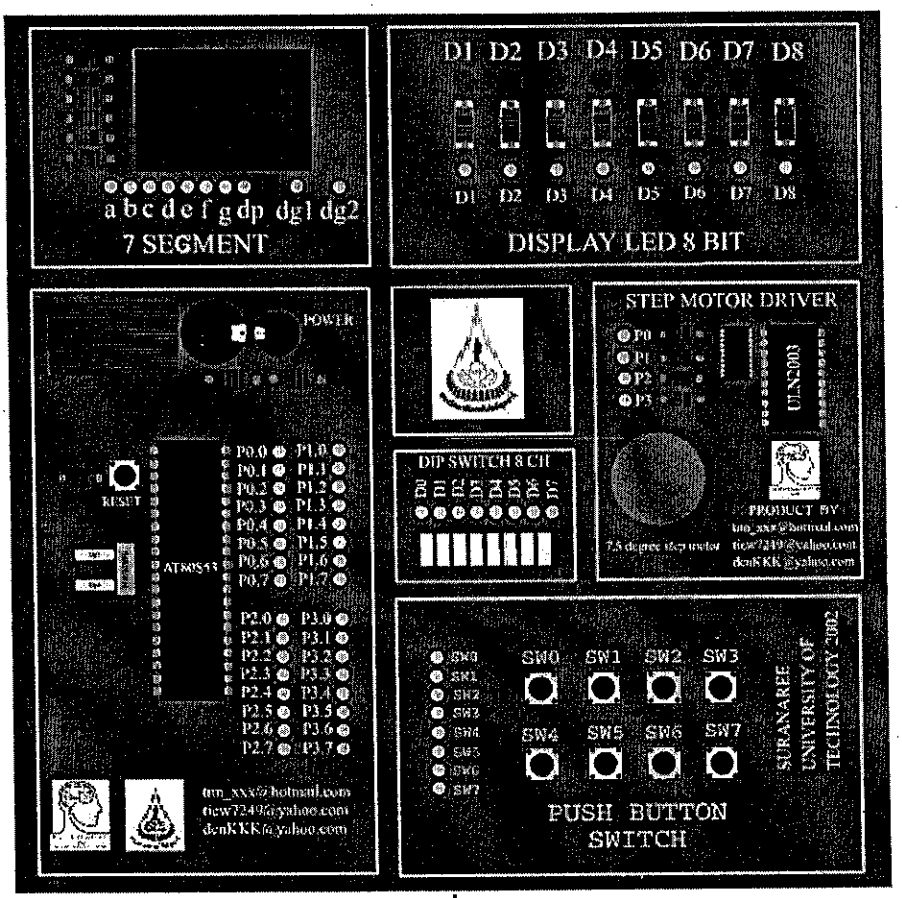
รูป Bord.gif จะถูกวาดลงที่พื้นหลังของ แอพเพลต เป็นอันดับแรกโดยคำสั่ง

```
g.drawImage(img,0,0,this);
```

เพื่อให้พื้นหลังถูกวาดเป็นรูปบอร์ดทดลอง โดยภาพ img ถูกอ่านขึ้นมาเพื่อใช้โดยคำสั่ง

```
img = getImage(getCodeBase(),"Bord.gif");
```

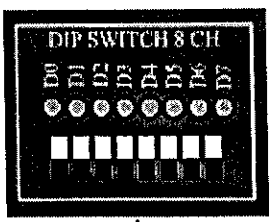
ใน method init(); ตั้งแต่เริ่มเปิด แอพเพลต นี้ขึ้นมา เป็นการกำหนดให้ตัวแปรที่ชื่อ img เป็นตัวแปรชนิด Image แล้ววาดลงบนพื้นหลังเพื่อที่จะวาดการกระทำและการเปลี่ยนแปลงของอุปกรณ์ตัวอื่นๆทับภายหลัง



รูปที่ 3.6

จากนั้นจะเป็นการเรียกฟังก์ชันเพื่อที่จะนำค่าอินพุตจากอุปกรณ์ที่เป็นอินพุตซึ่งได้แก่ DIP Switch และ Push Button Switch เข้าสู่ Port ต่างๆที่ถูกกำหนดการต่อไว้

1. DIP switch



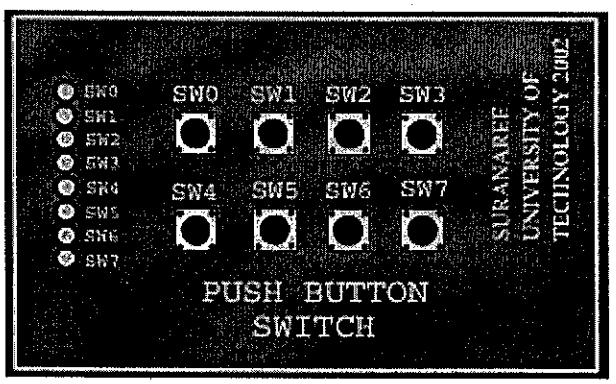
รูปที่ 3.7

class Dip จะมีการส่งตัวแปร dipซึ่งเป็นตัวแปรชนิด array Boolean เป็นค่าที่ได้จาก method Even mouseDown(Even e,int x ,int y) โดยจะรอรับเหตุการณ์การกด mouse ที่เกิดขึ้น ตัวแปร x เป็นตำแหน่งที่ mouse ถูกกดวัดจากจุด(0,0) แนวนอนจากมุมซ้ายบนสุด และตัวแปร y เป็นตำแหน่งที่ mouse ถูกกดวัดจากจุด(0,0) แนวตั้งจากมุมซ้ายบนสุด หากพบว่า mouse ถูกกดลงไปที่ตำแหน่งของ switch (บนพื้นที่สี่เหลี่ยมสีขาวและสีดำ) ตัวแปร array Boolean dip ก็จะเปลี่ยนแปลง ถ้ามีการกดลงไปด้านบนของพื้นที่สี่เหลี่ยมตัวแปร dip ก็จะเป็น true ถ้ามีการกด ลงไปด้านล่างของพื้นที่สี่เหลี่ยมตัวแปร dip ก็จะเป็น false จากนั้น ที่ method paint() ก็จะส่งค่าเหล่านี้ผ่านตัวแปรที่ชื่อ hi2 ซึ่งถูกกำหนดไว้ให้เป็นตัวแปร Dip ส่งค่าตัวแปร dip ไปพร้อมกับ Graphics gโดยใช้คำสั่ง

hi2= new Dip(dip,g);

ภายในคลาส Dip จะมีการตรวจสอบว่าตัวแปร dip เป็น true หรือ false ถ้าเป็น true คือการเลื่อน switch ขึ้นหรือปิด switch ถ้าเป็น false คือ เปิด switch ถ้าเป็นการปิด switch ก็จะทำการวาดรูปสี่เหลี่ยมสีดำลงบนด้านบนของ switch แล้ววาดสี่เหลี่ยมสีขาวลงบนด้านล่างของ switch เพื่อให้ดูเหมือนการเลื่อน switch ขึ้น จากนั้น ที่คลาส Dip จะส่งค่าของตัวแปร Boolean array ชื่อ dip_value ออกมาเพื่อที่จะนำค่าที่ได้จากการปิดหรือเปิด switch นี้ไปใช้งานอีกต่อไป

2. Push Button switch



รูปที่ 3.8

คลาสนี้เป็นคลาสที่ใช้งานเกี่ยวกับ Push Button Switch ภายหลังจาก method paint() เรียกใช้งานคลาส Dip แล้ว จะเป็นการเรียกใช้งานคลาส PB_switch() ด้วยคำสั่ง

PB=new PB_switch(PB_SW,g);

โดยค่า PB_SW คือตัวแปรชนิด Boolean array ขนาด 8 array เป็นการแทน 8 switch PB_SW เป็นการตรวจสอบว่ามีการกด mouse ลงบนตัว switch หรือไม่ ที่ method mouseDown(Even e, int x ,int y) ถ้ามีการกดที่ตัว switch ตัวแปร PB_SW ก็จะเป็น true และถ้าวาง mouse ตัวแปร PB_SW ก็จะเป็น false เพราะที่ method mouseUp() มีการกำหนดให้เมื่อมีการวาง mouse ตัวแปร PB_SW ทั้งหมดจะถูกกำหนดให้เป็น false จากนั้น method mouseUp() และ mouseDown() ก็จะเรียก method

paint() คำสั่ง PB=new PB_switch(PB_SW,g); ที่ method paint() จะเป็นการส่งค่าตัวแปร PB_SW ไปยังคลาส PB_switch

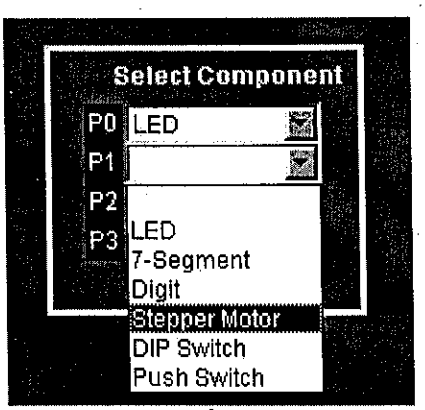
ที่คลาส PB_switch จะเป็นการนำค่าตัวแปร array Boolean PB_SW เข้ามาเพื่อตรวจสอบว่า array ที่ตำแหน่งใดเป็น true ซึ่งตำแหน่งของ array คือหมายเลขของ switch ถ้าที่ switch ใดเป็น true ก็จะทำการวาดรูปวงกลมสีน้ำตาลทับลงไปบนวงกลมสีดำที่แทนปุ่มของ switch ผลที่ได้คือเมื่อกด mouse ลงไปที่ switch switch ก็จะเปลี่ยนเป็นสีน้ำตาลเพื่อให้ทราบว่าได้กด switch แล้ว จากนั้น คลาส PB_switch ก็จะส่งตัวแปรที่ชื่อ pb_value ออกมาเพื่อที่จะนำไปใช้ต่อไป

เมื่อ method paint() ทำการรับค่าอินพุตต่างๆแล้ว paint() จะทำการเรียกคลาส Connec เพื่อจะทำการเชื่อมต่อ Port และอุปกรณ์ต่างๆเข้าด้วยกันด้วยคำสั่ง

con=new Connec(p0,p1,p2,p3,hi2.dip_value,PB.pb_value,s0,s1,s2,s3,g);

โดย con คือตัวแปร Connec คลาส Connec จะรับค่า Port p0 p1 p2 p3 ค่า hi2.dip_value ซึ่งเป็นค่าที่ได้จากคลาส Dip ค่า PB.pb_value ซึ่งเป็นค่าที่ได้จากคลาส PB_switch ค่า s0 s1 s2 s3 ซึ่งเป็นค่า integer มีค่าตั้งแต่ 1 ถึง 6 (6 ค่า) เป็นการกำหนดว่า Port แต่ละ Port จะต่ออุปกรณ์ อะไรบ้าง โดย s0 เป็นตัวแทน Port p0 ถ้า s0 เท่ากับ 1 คือการกำหนดให้ Port p0 ต่อกับ LED เท่ากับ 2 คือการกำหนดให้ Port p0 ต่อกับ seven segment เท่ากับ 3 คือการกำหนดให้ Port p0 ต่อกับ digit ของ seven segment เท่ากับ 4 คือการกำหนดให้ Port p0 ต่อกับ สเต็ปมอเตอร์ เท่ากับ 5 คือการกำหนดให้ Port p0 ต่อกับ DIP switch เท่ากับ 6 คือการกำหนดให้ Port p0 ต่อกับ Push Button Switch และ s1คือตัวแทนการเชื่อมต่อของ Port p1 s2 คือตัวแทนการเชื่อมต่อของ Port p2 s3คือตัวแทนการเชื่อมต่อของ Port p3 โดยค่า s0 s1 s2 s3 จะได้จากการเลือกอุปกรณ์ที่ช่อง Select Component ดังรูป และการกำหนดการเชื่อมต่อของ Port ต่างๆก็เช่นเดียวกัน

3. Select Components

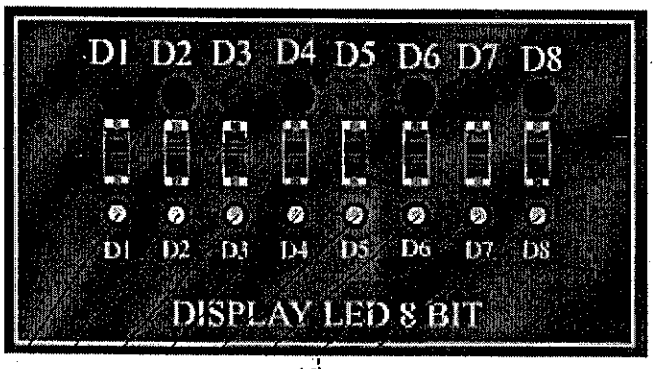


รูปที่ 3.9

เมื่อคลาส Connec รับค่าต่างๆแล้ว จะนำค่า s0 s1 s2 s3 ไปตรวจสอบว่าเท่ากับเท่าไร เพื่อให้ Port เลือกอุปกรณ์เชื่อมต่อต่างๆจากนั้นก็จะทำการวาดเส้นตรงจาก Port ไปที่ อุปกรณ์ต่างๆ เพื่อ

แสดงให้เห็นว่าอุปกรณ์ได้ทำการเชื่อมต่อแล้ว จากนั้นคลาส Connec ก็จะส่งค่าตัวแปร ที่เป็นค่าของอุปกรณ์ประเภทเอาท์พุต และค่าของ Port ต่างๆ ออกมาดังนี้ Boolean array led_value ซึ่งเป็นค่าที่จะต้องนำไปแสดงที่ LED Boolean array seven_seg ซึ่งเป็นค่าที่จะต้องนำไปแสดงที่ Seven segment step_value ซึ่งเป็นค่าที่จะต้องนำไปแสดงที่ สเต็ปมอเตอร์ digit_value ซึ่งเป็นค่าการกำหนด digit ของ seven segment และส่งค่า p0 p1 p2 p3 ออกมาเพื่อนำไปใช้งานต่อไปภายหลังจากคลาส Connec ถูกเรียกใช้งานแล้ว paint() จะเรียกใช้งานคลาสที่เป็นอุปกรณ์เอาท์พุต ซึ่งได้แก่ LED Seven segment และ สเต็ปมอเตอร์

4. LED



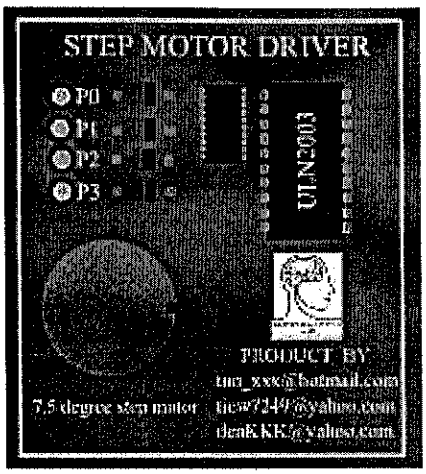
รูปที่ 3.10

คลาส Led ถูกเรียกใช้งานโดย method paint() ด้วยคำสั่ง

hi=new Led(con.led_value,g);

ซึ่ง hi เป็นตัวแปร Led คลาส Led จะรับค่า con.led_value ซึ่งเป็นค่าจาก Port ที่เชื่อมต่อกับ LED และคลาส Connec เป็นตัวเชื่อมต่อแล้วส่งค่านี้ออกมา ที่คลาส Led เมื่อรับค่า con.led_value ซึ่งเป็นค่า Boolean array มาแล้ว จะทำการตรวจสอบค่าของ con.led_value โดยวนค่าตรวจสอบที่ละ index และวาดวงกลมสีแดงทับลงไปที่วงกลมเดิม ถ้า array ที่ index นั้นเป็น true ถ้าตรวจแล้วเป็นค่า false ก็จะไม่ทำการวาด เมื่อเสร็จแล้วจะทำให้เห็น คล้ายกับเป็นการติดและดับของ LED

5. สเต็ปมอเตอร์



รูปที่ 3.11

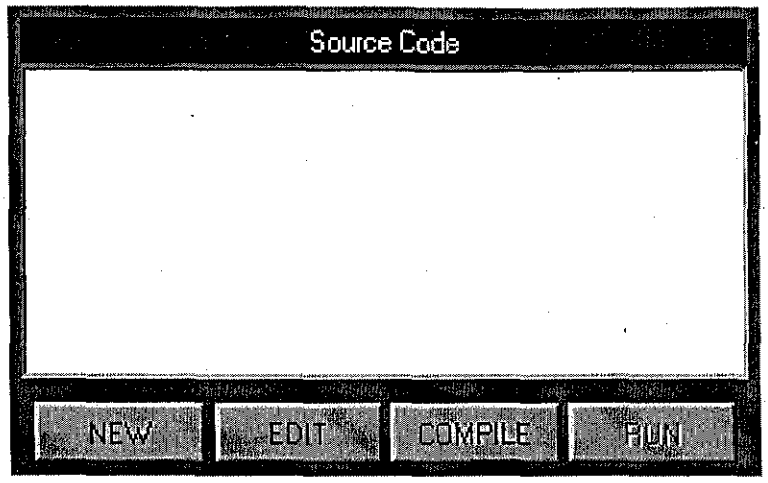
การทำงานใน method Motor() จะเป็นการนำค่าของ Port ที่ต่อกับ สเต็ปมอเตอร์ นี้เก็บไว้แล้วนำมาเปรียบเทียบกับค่าเดิมว่าเป็นการเลื่อน Bit หรือไม่ ถ้าเป็นการเลื่อน bit แล้ว เป็นการเลื่อน bit ไปในทิศทางใด แล้วนำค่าที่นำมาเปรียบเทียบนั้นมาคำนวณเป็นค่าในการหมุนครั้งต่อไป ส่วนการหมุน Motor นั้นใช้การวาดรูปวงกลมสีแดง แล้วเลื่อนตำแหน่งไปเรื่อยๆเป็นรูปวงกลมให้ดูคล้ายกับว่า Motor หมุน การเรียก method Motor() โดยใช้คำสั่ง

Motor() ;

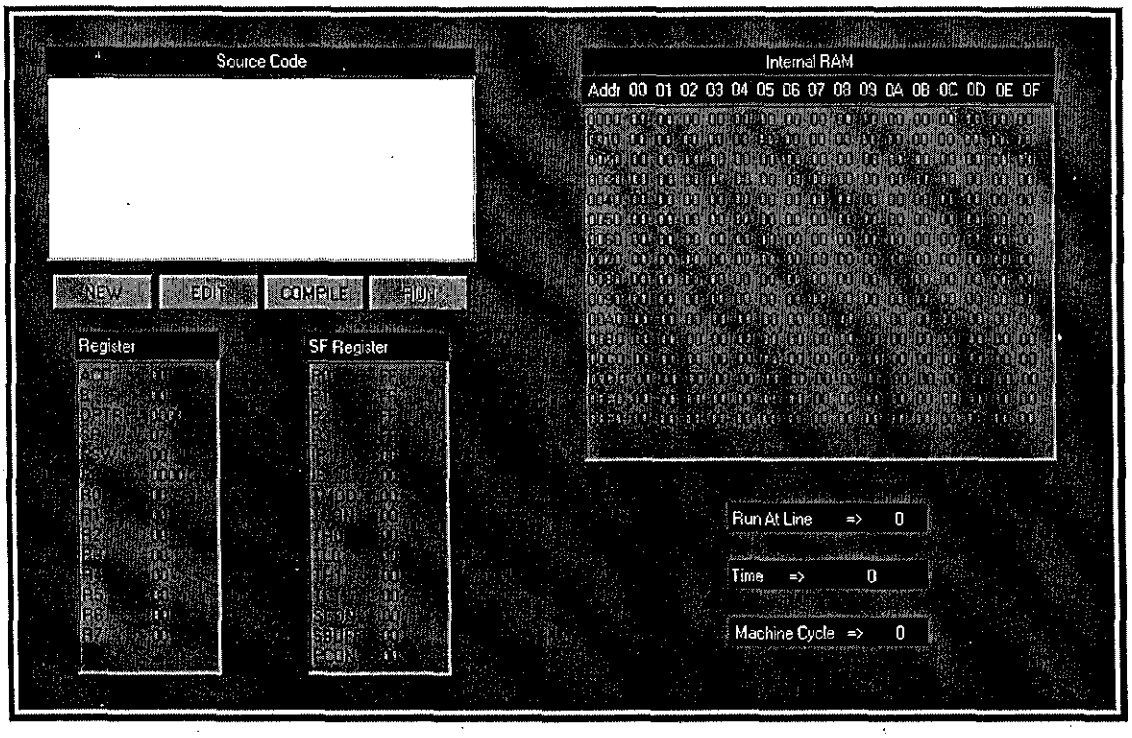
ที่ method Motor() จะนำค่า Boolean array con.step_value มาเปรียบเทียบ ว่าเป็นการเลื่อน bit หรือไม่และในทิศทางใดถ้าเป็นการเลื่อน bit (เลื่อนค่า index ของ array Boolean con.step_value) ก็จะทำการวาดรูปวงกลมสีแดง ลงบนตำแหน่ง x[i] และ y[i] ซึ่งค่า x และค่า y ได้จากการคำนวณค่าของจุดต่างๆบนเส้นรอบวงของรูปวงกลมสีเทา แล้วนำค่ากำหนดเป็นตังแปร array integer x และ y ตัวแปร i จะเป็นตัวกำหนดตำแหน่งของวงกลมที่จะวาด

2.ส่วนของการเขียนโปรแกรมในการจัดการซอสโค้ดโปรแกรมภาษาแอสเซมบลี

รูปแบบที่ต้องจัดการกับซอสโค้ดของการเขียนโปรแกรมภาษาแอสเซมบลีซึ่งอยู่ในส่วนของ Text Editor ในส่วนล่างของบอร์ดจำลองที่ทำหน้าที่รองรับการเขียนโปรแกรมภาษาแอสเซมบลี ที่ผู้ใช้จะทำการเขียนขึ้นเพื่อใช้ในการควบคุมและสั่งการให้อุปกรณ์บนส่วนแสดงผลบอร์ดจำลองทำงานตามรูปที่ 3.13 ซึ่งจะประกอบด้วยการคอมไพล์, การรันโปรแกรม โดยการเขียนโปรแกรมลงบน Text editor จากนั้นทำการคอมไพล์และรันโปรแกรมเพื่อการแสดงผลของบอร์ดจำลองโดยที่จากการรันโปรแกรมแล้วนอกจากส่วนที่แสดงอุปกรณ์แล้วที่สามารถแสดงให้เห็นได้ยังจะมีในส่วนของ Internal RAM, External RAM, Register, SF Register ดังแสดงในรูปที่ 3.14



รูปที่ 3.12 Text editor



รูปที่ 3.13

โดยที่การทำงานของซอสโค้ดจะได้รับการคอมไพล์โดยการเขียนโปรแกรมจาก Microsoft Visual J++โดยโปรแกรมที่เขียนขึ้นจะทำการอ่านข้อมูลที่ผู้ใช้เขียนลงใน Text editor ในส่วนของหน้าแสดงผลของบอร์ดจำลองทั้งหมดจากนั้นโปรแกรมที่สร้างขึ้นจะทำการแยกบรรทัดของข้อมูลที่ได้มาว่าในแต่ละบรรทัดประกอบด้วยอะไรบ้างโดยจะแบ่งออกตามหลักการเขียนโปรแกรมภาษาแอสเซมบลีที่จะประกอบด้วย3 ส่วนหลักคือ Label, Mnemonic, และ Operand จากนั้นโปรแกรมจะทำการนำข้อมูลทั้งหมดที่ผ่านการแบ่งประเภทแล้วในแต่ละบรรทัดไปทำการเปลี่ยนค่ารหัสเลขฐาน

สิบหกเพื่อนำไปตรวจสอบว่าค่าที่แปลงมานั้นตรงกับคำสั่งการทำงานใดโดยตรวจสอบจากโปรแกรมที่เขียนขึ้นในส่วนที่เป็นการรัน แล้วจะทำการทำงานตามคำสั่งนั้นๆแล้วแสดงผล

ตัวอย่างของโปรแกรมในการตัดบรรทัดแสดงดังนี้

```
for (int i=0;i<LtString;i++)
                {
                        charat = tString.charAt (i);
                        if ((charat=='\n')||(i==(LtString-1)))
                        {
                                stop = i;
                                if ((i==(LtString-1)) && (charat!='\n'))
                                {
                                        stop = i+1;
                                }
                                StringLine[TLine] = tString.substring (start,stop);
                                LineStart[TLine] = start;
                                LineStop[TLine] = stop;
                                start = stop + 1;
                                TLine = TLine + 1;
                        }
                }
```

ตัวอย่างของโปรแกรมในการตัดคำแสดงดังนี้

```
int LineNo=0,No=0,p=0,Lsbuff,o=0,LSL,first=0;boolean mne;
 for (int i=LineNo;i<TLine;i++)
{
   mne = true;
   o=0;
   LSL = StringLine[LineNo].length ();
   for (int j=0;j<LSL;j++)
   {
      if (StringLine[LineNo].charAt (j) == ' ' || StringLine[LineNo].charAt (j) == '\t' || ((LSL-1)
== j))
                                {
```

```
if ( j==0)
{
        o = j+1;
}
        else if (StringLine[LineNo].charAt (j-1) == ' ' || StringLine[LineNo].charAt (j-1) == '\t')
{
if (StringLine[LineNo].charAt (j) == ' ' || StringLine[LineNo].charAt (j) == '\t')
{
        o = j+1;
}
        else
{
if ((LSL-1) == j)
{
if (StringLine[LineNo].charAt (j) == ' ')
{
j = j - 1;
}
j = j+1;
}
 sbuff = StringLine[LineNo].substring (o,j);
 if (StringLine[LineNo].charAt (j-1) == ':')
 {
    label[LineNo] = StringLine[LineNo].substring (o,j-1);
         mne = true;
         o = j+1;
 }
 else if (mne)
 {
 mnemonic[LineNo] = sbuff;
         mne =false;
         o = j + 1;
```

```
}
        else
{
        Lsbuff = sbuff.length ();
        p = 0;
        first = 0;
        No = 0;
        for (int k=0;k<Lsbuff;k++)
{
if (sbuff.charAt (k) == ',' || (Lsbuff-1) == k)
{
if (Lsbuff-1 == k)
{
        k = k + 1;
}
        operand[LineNo][No] = sbuff.substring (first,k);
        No = No+1;
        first = k + 1;
 }
        mne = true;
 }
 }
 }
 }
         else
 {
         if ((LSL-1) == j)
 {
 if (StringLine[LineNo].charAt (j) == ' ')
 {
         j = j - 1;
 }
```

```
        j = j+1;
}
        sbuff = StringLine[LineNo].substring (o,j);
if (StringLine[LineNo].charAt (j-1) == ':')
{
        label[LineNo] = StringLine[LineNo].substring (o,j-1);
        mne = true;
        o = j+1;
}
        else if (mne)
{
        mnemonic[LineNo] = sbuff;
        mne =false;
        o = j + 1;
}
        else
{
        Lsbuff = sbuff.length ();
        p = 0;
        first = 0;
        No = 0;
        for (int k=0;k<Lsbuff;k++)
 {
        if (sbuff.charAt (k) == ',' || (Lsbuff-1) == k)
 {
        if (Lsbuff-1 == k)
 {
        k = k + 1;
 }
        operand[LineNo][No] = sbuff.substring (first,k);
        No = No+1;
        first = k + 1;
```

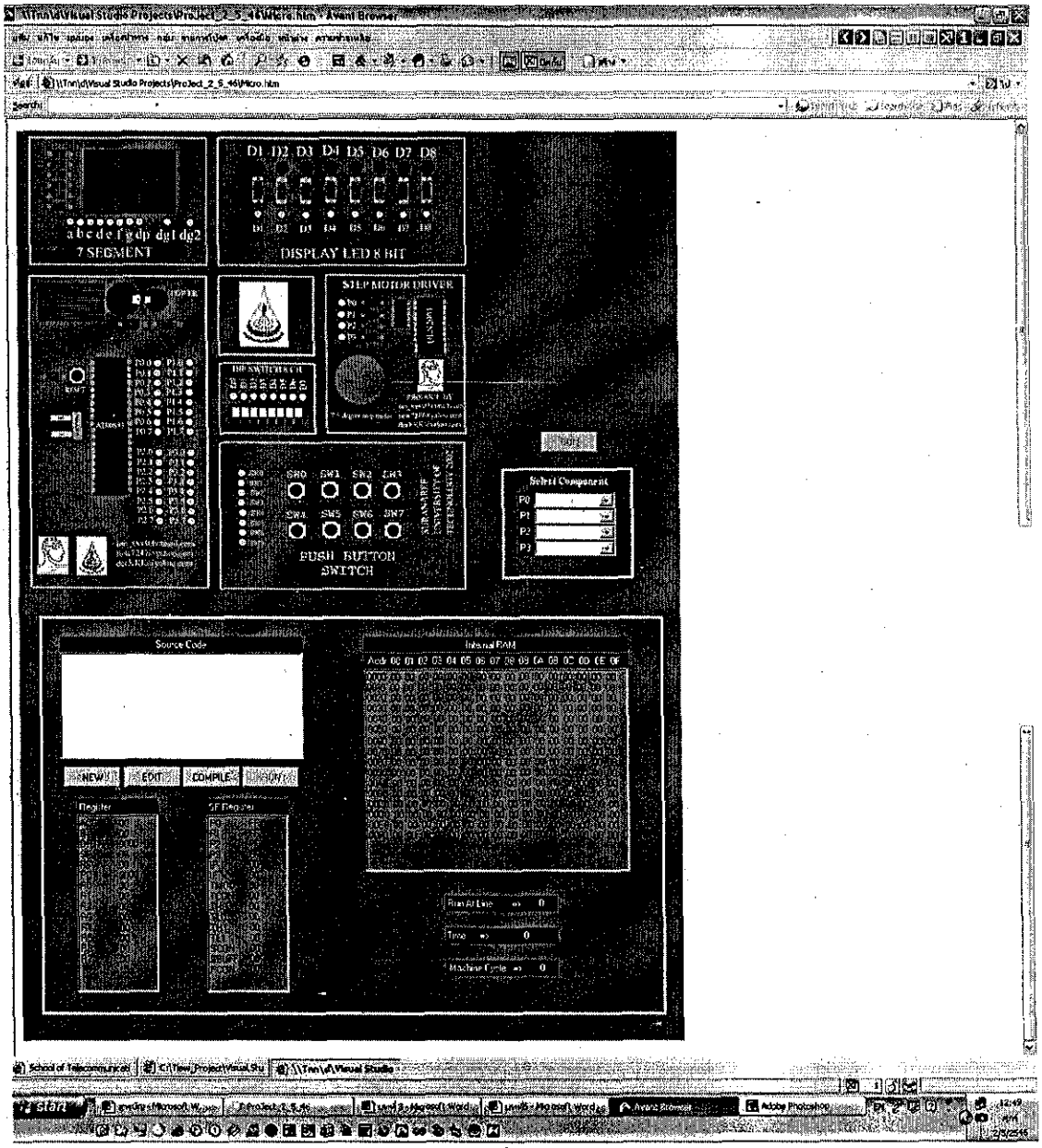
```
}
        mne = true;
}
}
}
}
}
        LineCompile=LineNo;
        LineNo =LineNo +1;
        RunCompile();
}
```

บทที่ 4

วิธีการใช้งานบอร์ดจำลองการทำงานไมโครคอนโทรลเลอร์ MCS-51

4.1 วิธีการใช้งาน

อุปกรณ์ต่างๆบน แอพเพลต เมื่อแอพเพลต ถูกโหลดมาแล้วจะมีลักษณะดังรูป 4.1

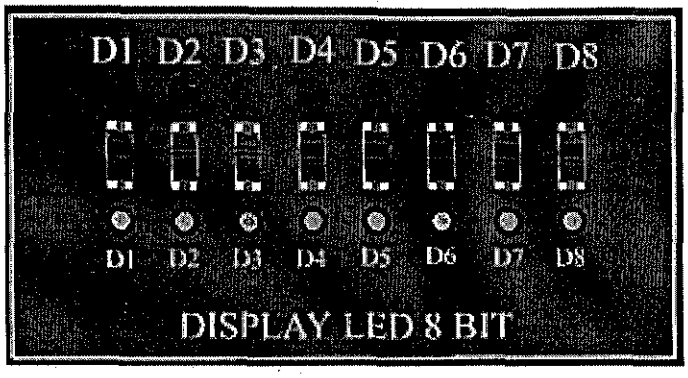


รูปที่ 4.1

ส่วนด้านบนจะเป็นรูปของวงจรอิเล็กทรอนิกส์ ส่านด้านล่างจะเป็นการแสดงสถานะของ Register ต่างๆ แบ่งเป็นรายละเอียดดังนี้

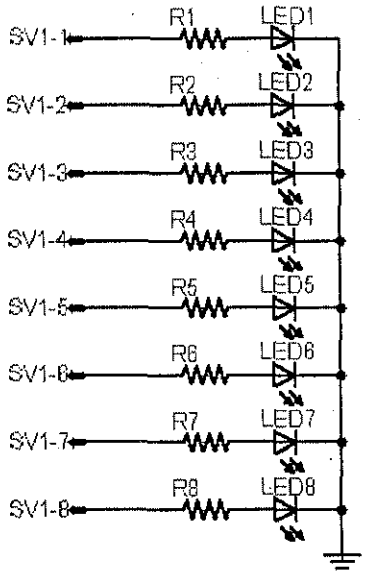
4.2 อุปกรณ์บนบอร์ดทดลอง

4.2.1 LED



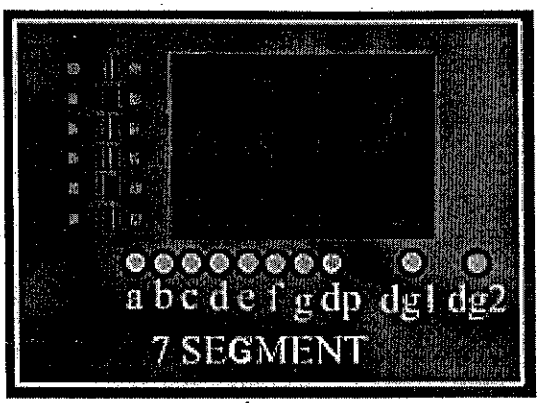
รูปที่ 4.2

LED 8 ดวง ถูกต่อวงจรโดยขา Anode ต่อเข้ากับตัวต้านทานและปลายอีกด้านของตัวต้านทานต่อเข้ากับ Connector ขา Cathode ถูกต่อเข้ากับ 0 volt แสดงเป็น Schematic ดังรูป



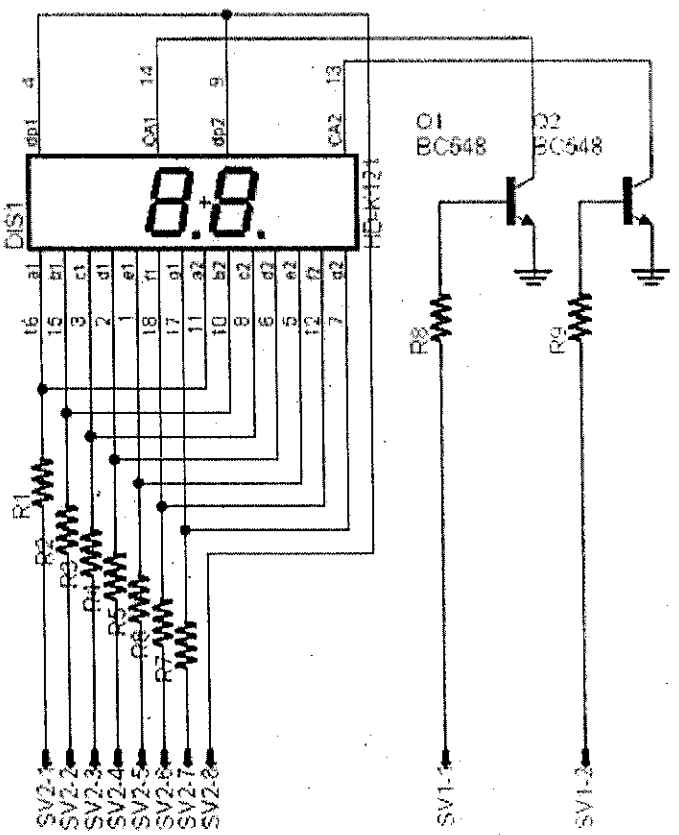
รูปที่ 4.3

4.2.2 Seven Segment



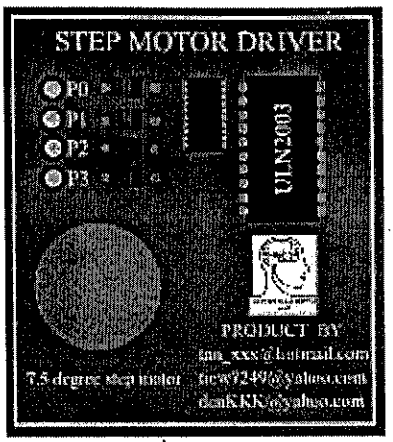
รูปที่ 4.4

Seven Segment ชนิด Common cathode ขา Anode ต่อเข้ากับตัวต้านทานและปลายอีกด้านของตัวต้านทานต่อเข้ากับ Connector ขา cathode ถูกต่อเข้ากับ transistor เพื่อเป็นตัวเลือกว่าจะให้ digit ใดเป็นตัวแสดงผล โดยที่ขา collector ของ transistor ถูกต่อเข้ากับ Cconnector ดังวงจรดังรูป



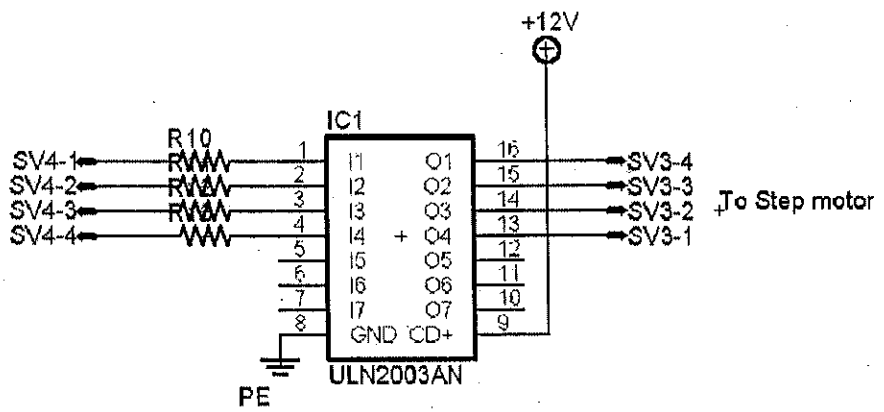
รูปที่ 4.5

4.2.3 สเต็ปมอเตอร์



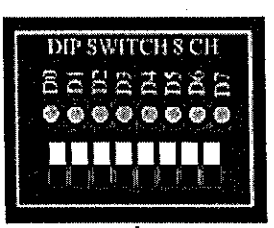
รูปที่ 4.6

สเต็ปมอเตอร์ แบบ Unipolar 7.5 degree per step ถูกต่อเข้ากับวงจร Driver ด้วย ULN2003 เป็นรูปวงจร Sschematic ดังรูป



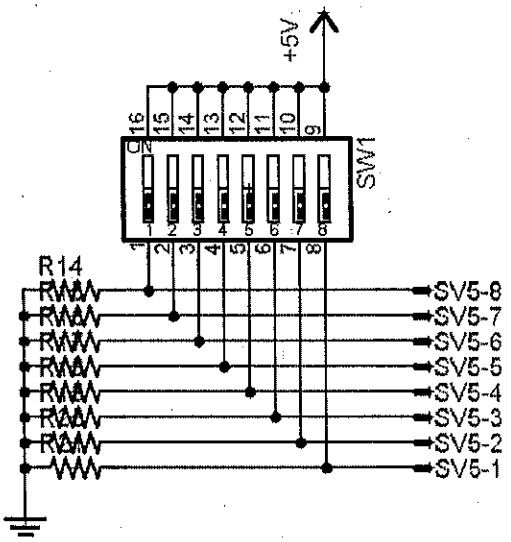
รูปที่ 4.7

4.2.4 DIP Switch



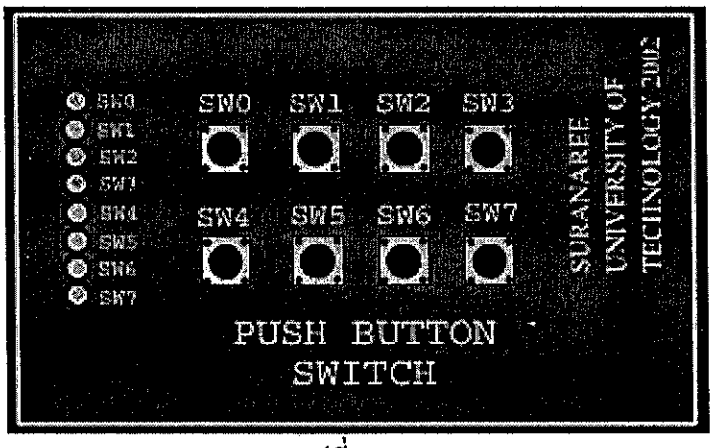
รูปที่ 4.8

Dipswitch 8 ตัวต่อกับ Connector เมื่อเลื่อน switch ขึ้นจะให้ logic high ออกมา ถ้าเลื่อน switch ลงจะให้ logic low ออกมา แสดงเป็นรูปวงจรดังนี้



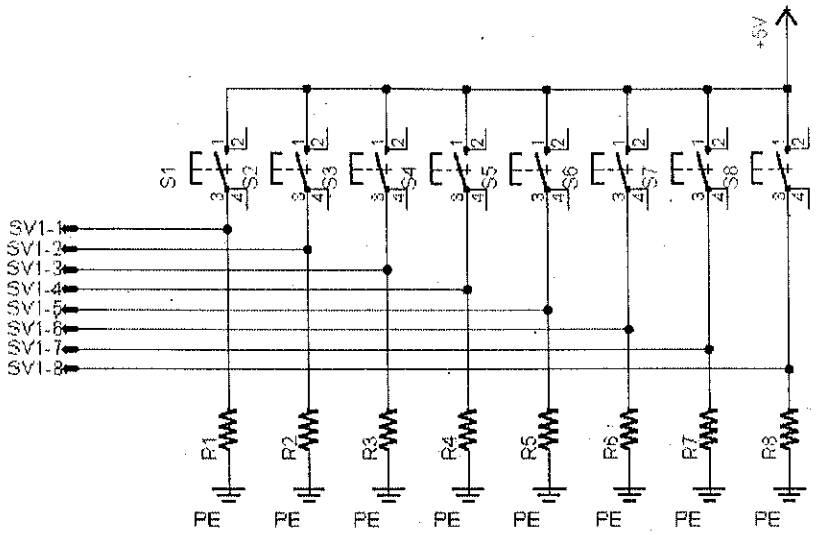
รูปที่ 4.9

4.2.5 Push button switch



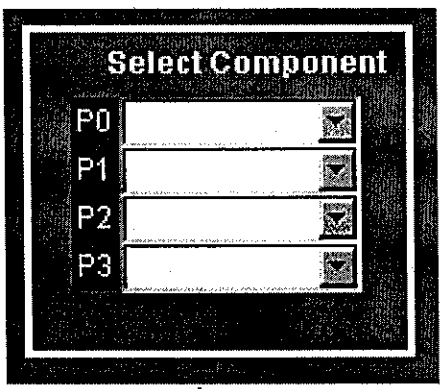
รูปที่ 4.10

Push button switch 8 ตัว ถูกต่อวงจรแบบ active hight เมื่อกดที่ตัว switch จะให้ logic high ออกมา โดยปรกติถ้าไม่มีการกดปุ่มก็จะให้ logic low ออกมา แสดงเป็นรูปวงจรดังนี้



รูปที่ 4.11

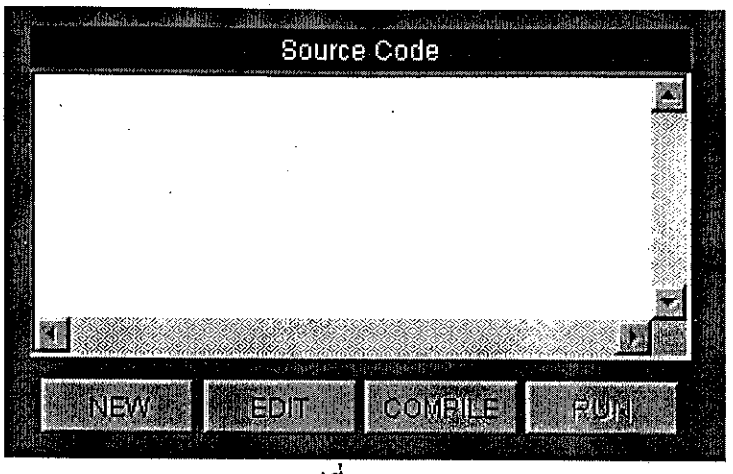
4.2.6 Select Component



รูปที่ 4.12

Combol box นี้ใช้ในการเลือกอุปกรณ์ที่จะเชื่อมต่อกับ Port ของ Microcontroller สามารถกดเลือกได้ว่าจะให้ Port ใดเชื่อมต่อกับอุปกรณ์ใด โดยกดเลือกที่ตัว Combol box ที่เป็นลูกศรเลื่อนลงจะปรากฏอุปกรณ์ให้เลือก

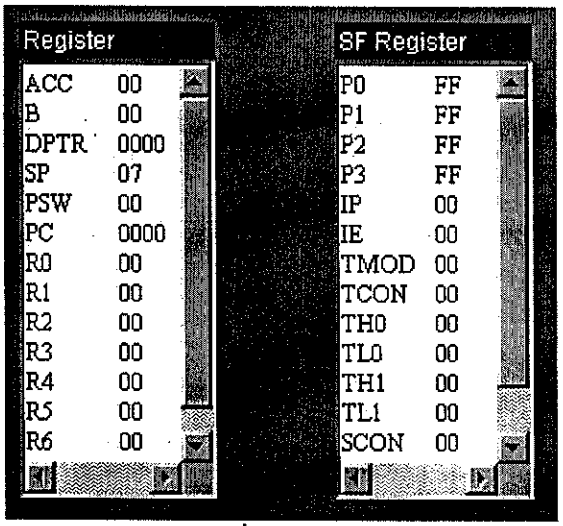
4.2.7 Text editor



รูปที่ 4.13

เป็นหน้าต่างที่ใช้ในการเขียนโปรแกรมภาษา assembly ป้อนลงไปเพื่อทดสอบโปรแกรม ปุ่ม NEW คือการสร้างซอสโค้ดใหม่ ปุ่ม EDIT คือการแก้ไขซอสโค้ดที่เขียนไว้แล้ว ปุ่ม COMPILE คือการ Compile ซอสโค้ดที่สร้างไว้ ปุ่ม RUN ใช้ในการ RUN โปรแกรมที่เขียน

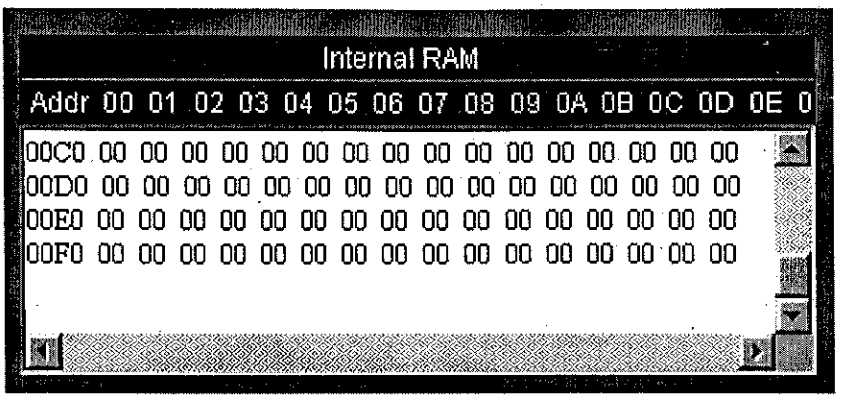
4.2.8 Register display



รูปที่ 4.14

เป็นหน้าต่างที่ใช้ในการแสดงค่าของ Register และ Port ต่างๆ เมื่อทำการรันโปรแกรมแล้ว

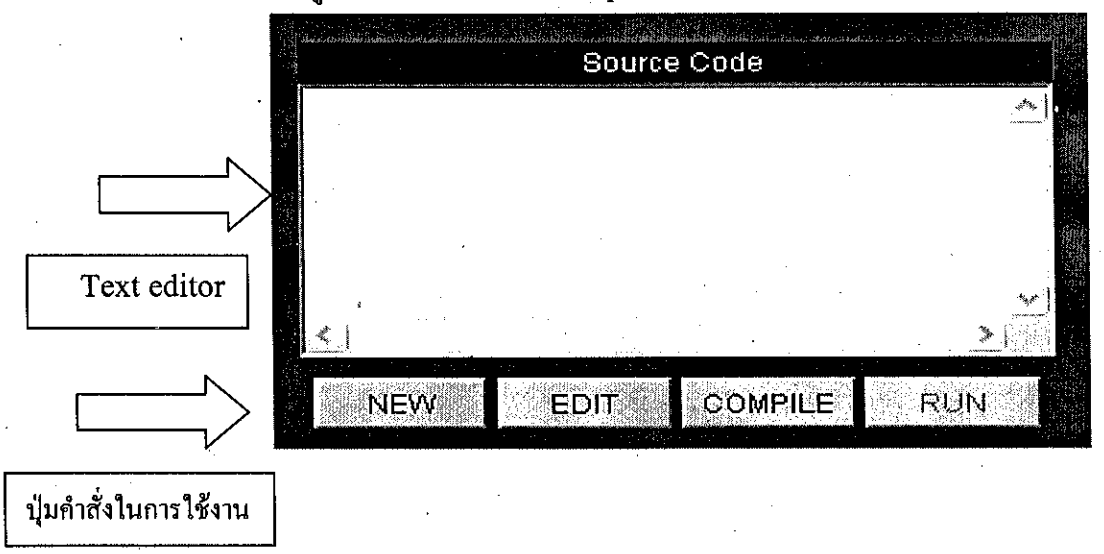
4.2.9 Internal RAM display



รูปที่ 4.15

เป็นหน้าต่างแสดงการเปลี่ยนแปลงของ Internal RAM เมื่อรันโปรแกรม

การใช้งานงานบอร์ดจำลองการทำงานไมโครคอนโทรลเลอร์ MCS-51 สำหรับผู้ใช้งานทั่วๆไปสามารถทำการเขียน ซอสโค้ดลงใน Text editor จากนั้นก็สามารถทำการคอมไพล์โค้ดเพื่อนำไปใช้ในการรันโดยจะมีปุ่มคำสั่ง New, Edit, Compile, และ Run อยู่บริเวณด้านล่างของ Text editor ดังแสดงในรูปที่3.15 และสามารถเลือกการเชื่อมต่ออุปกรณ์บนบอร์ดจำลองได้โดยการเลือกอุปกรณ์ต่างๆจาก Combo Box ที่เป็น Select component โดยเลือกการติดต่อกับพอร์ตของไมโครคอนโทรลเลอร์ได้ด้วยดังรูปที่ 3.2แสดง Select Component

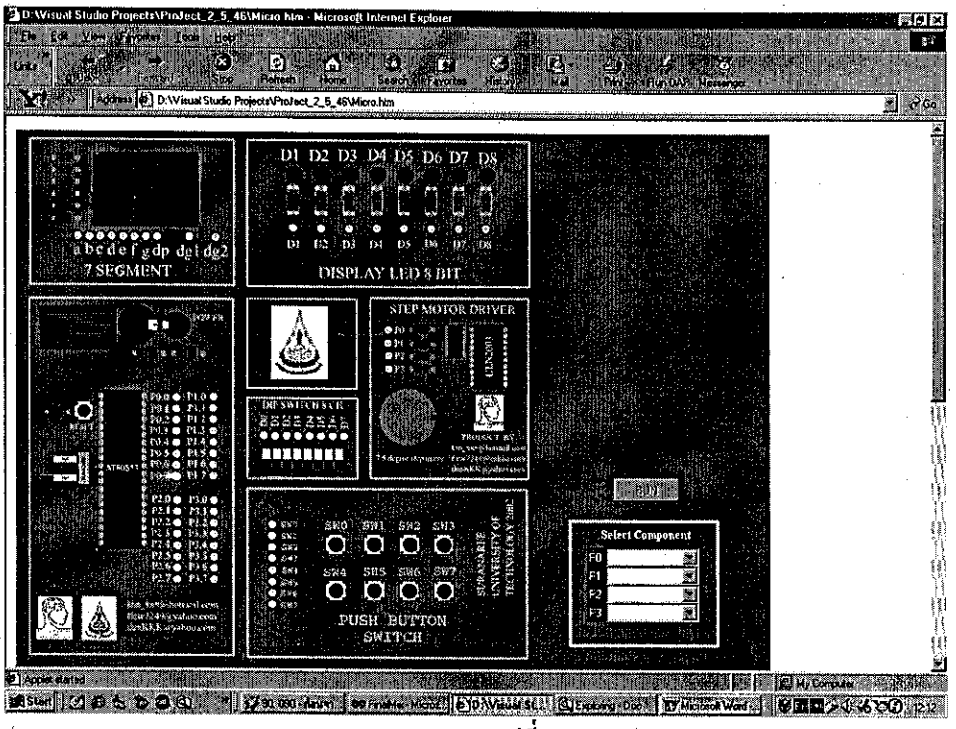


รูปที่ 4.16

4.3 ตัวอย่างการทำงาน

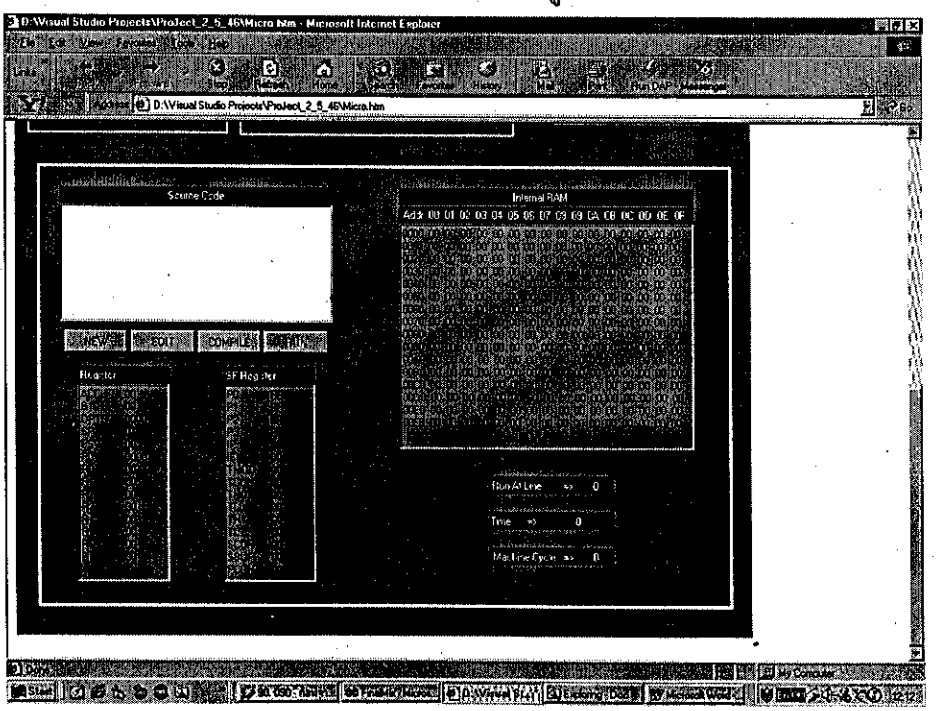
4.3.1 ตัวอย่างการเขียนโปรแกรม Display LED

ขั้นตอนที่ 1.เปิด Web Brower ขึ้นมา ลิงค์ไปยัง file:\\Micro.html จะได้ดังรูป



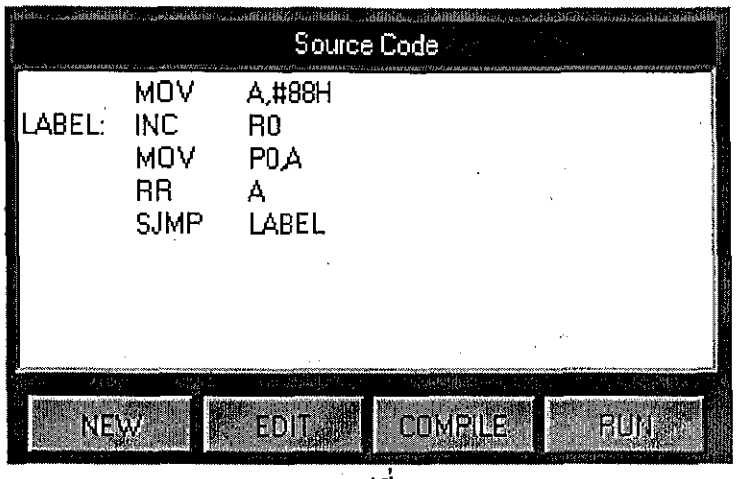
รูปที่ 4.17

ขั้นตอนที่ 2. เลื่อนหน้าเว็บเพจลงไปด้านล่างจะได้ดังรูป



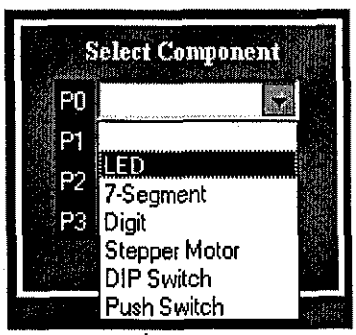
รูปที่ 4.18

ขั้นตอนที่ 3. ทำการเขียน Source Code ลงใน Edit Box



รูปที่ 4.19

ขั้นตอนที่ 4. ทำการเลือกส่วนที่ต้องการเชื่อมต่อระหว่างพอร์ตกับ LED

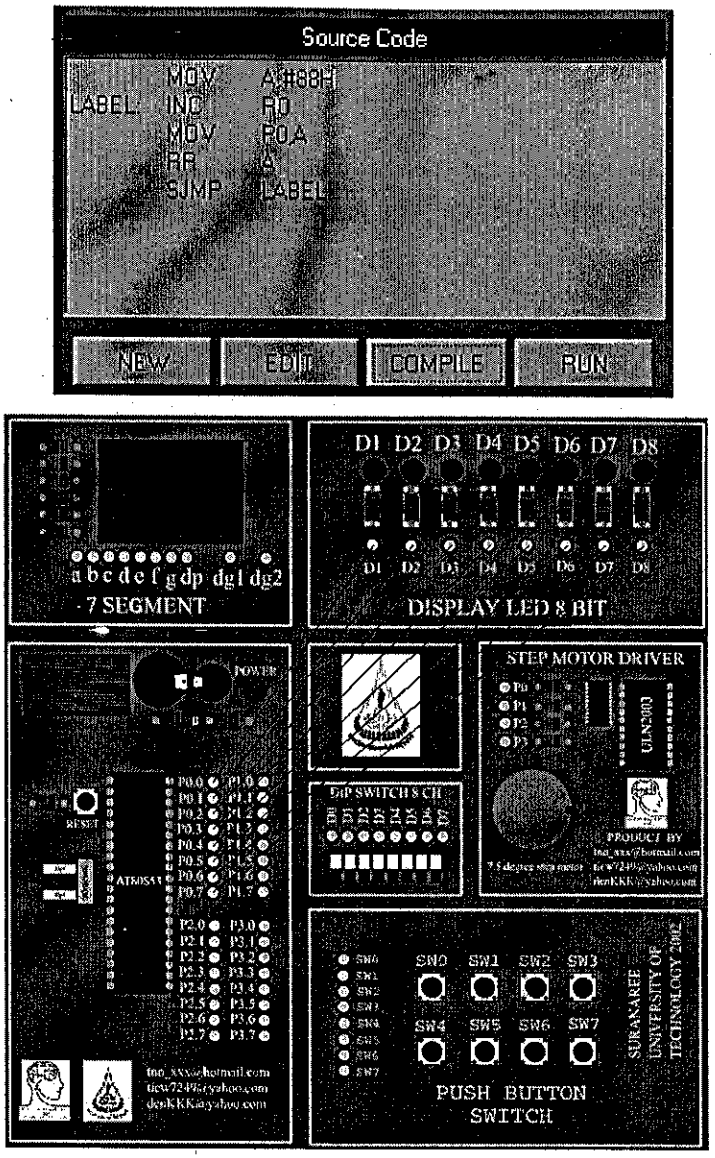


รูปที่ 4.20

ขั้นตอนที่ 5. ทำการคอมไพล์โดยกดที่ปุ่ม COMPILE

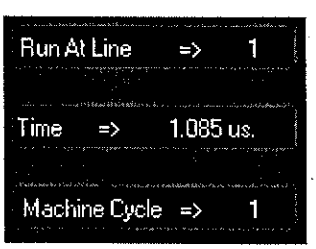
รูปที่ 4.21

ขั้นตอนที่ 6. เมื่อทำการคอมไพล์แล้ว ปุ่ม RUN ก็สามารถใช้งานได้ แล้วจะมีการต่อพอร์ต P0 กับ LED

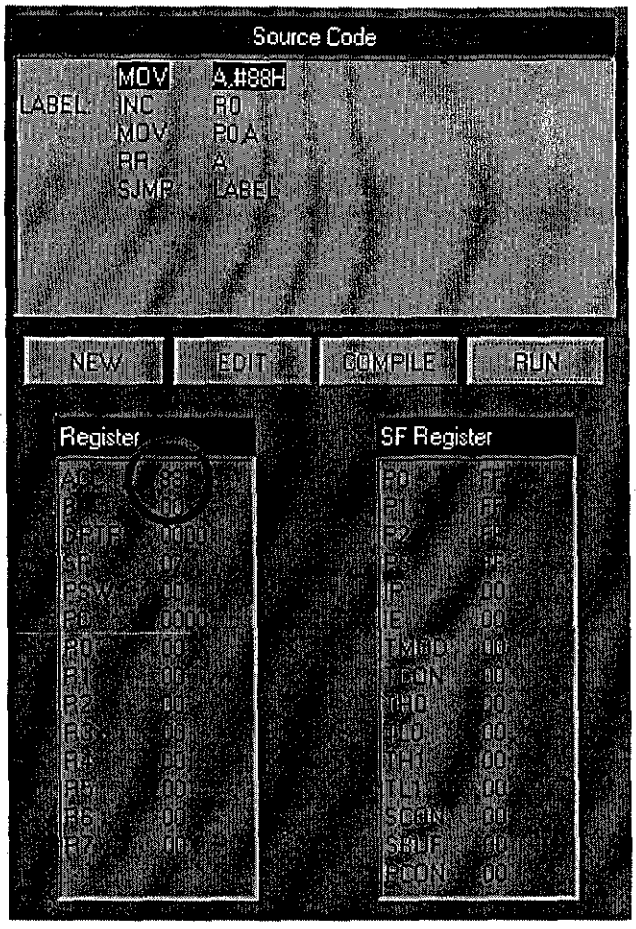


รูปที่ 4.22

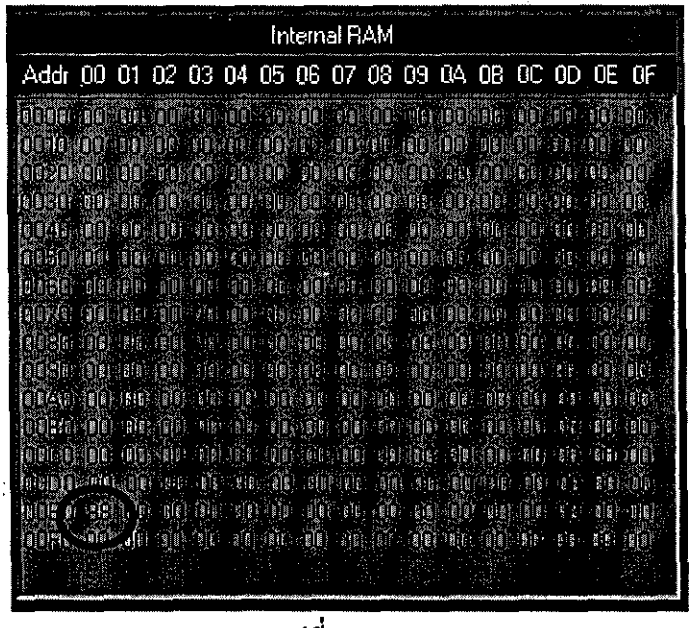
ขั้นตอนที่ 7. หลังจากทำการคอมไพล์กดที่ปุ่ม RUN เพื่อดูผลที่เกิดขึ้นโดยดูผลได้ในส่วนที่เป็น Register , SF Register , Internal RAM และส่วนที่เป็นการจำลองทางอุปกรณ์ต่อภายนอก



รูปที่ 4.23

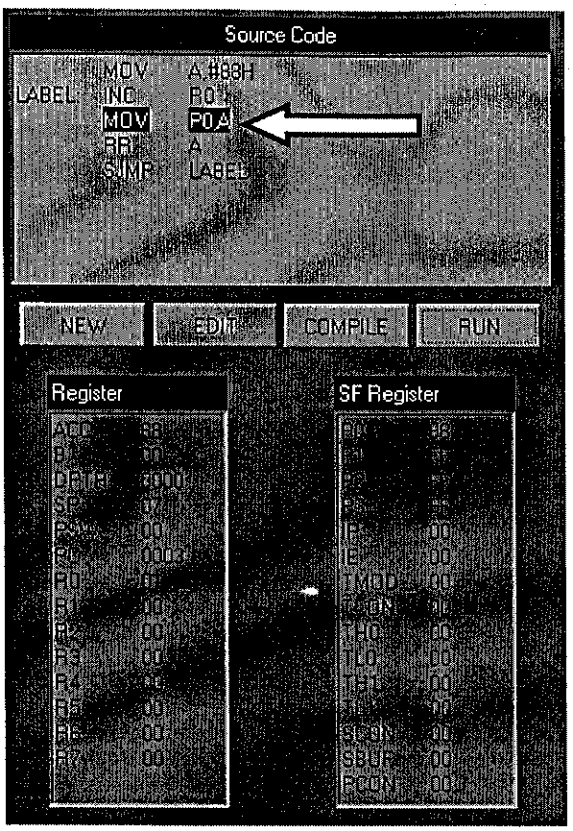


รูปที่ 4.24

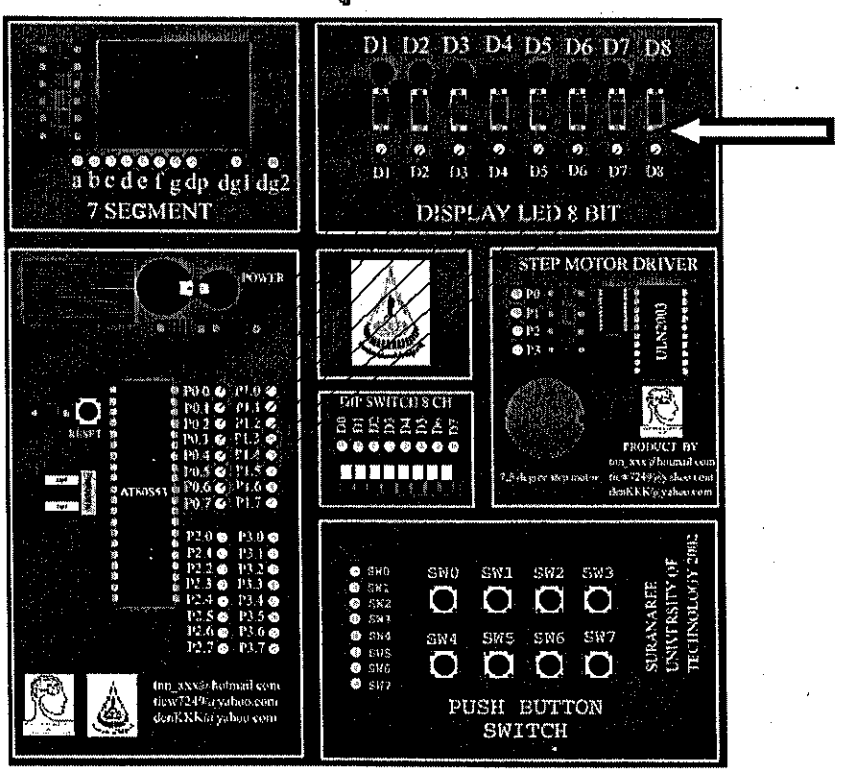


รูปที่ 4.25

ขั้นตอนที่ 8. กดที่ปุ่ม RUN ไปเลยๆ เราจะเห็นผลที่เกิดขึ้นดังรูป



รูปที่ 4.26

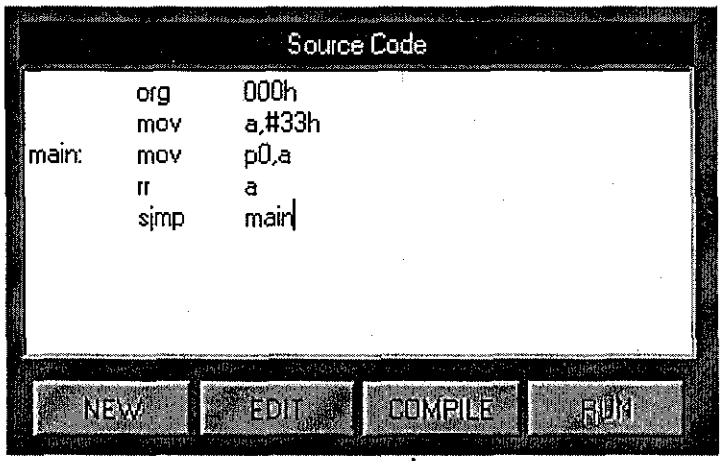


รูปที่ 4.27

4.3.2 ตัวอย่างการเขียนโปรแกรมขับ step motor

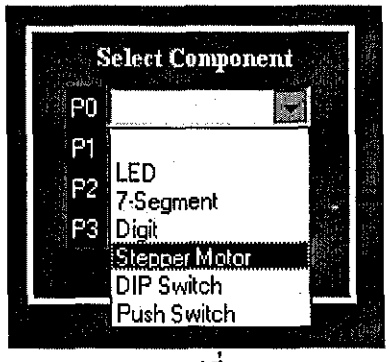
ขั้นตอนที่หนึ่ง ที่หน้าต่าง edit box กด new เพื่อเขียนโปรแกรมใหม่ แล้วเขียนโปรแกรมดังนี้

```
        org     000h
        mov     a,#33h
main:   mov     p0,a
        rr      a
        sjmp    main
```

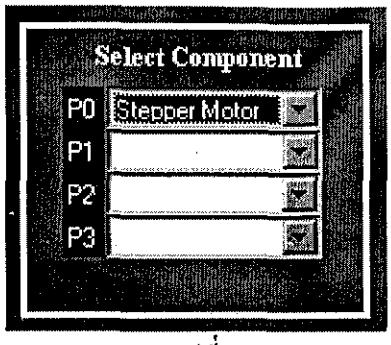


รูปที่ 4.28

ขั้นตอนที่สอง ที่ select component เลือก P0 ให้เชื่อมต่อกับ Step motor



รูปที่ 4.29



รูปที่ 4.30

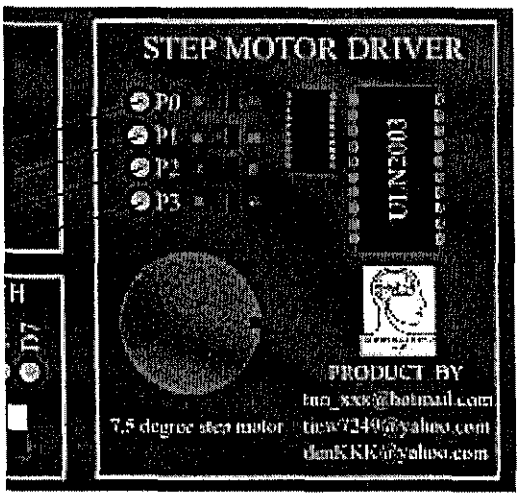
ขั้นตอนที่สาม กดปุ่ม 

เพื่อคอมไพล์ โปรแกรมที่เขียน

ขั้นตอนที่สี่ กดปุ่ม 

เพื่อทำงาน

ขั้นตอนที่ห้า ทำซ้ำขั้นตอนที่สี่ผลที่ได้คือ Step motor จะหมุนทีละ step



รูปที่ 4.31

บทที่ 5
บทสรุป

5.1 ข้อสรุปงานที่พัฒนาขึ้นจากโครงงาน

1. สร้างแบบจำลองการทำงานของบอร์ดจำลองไมโครคอนโทรลเลอร์ MCS-51 ที่สามารถใช้งานได้จริง
2. ทำการสร้างโครงสร้างอุปกรณ์เชื่อมต่อบนแบบจำลองบอร์ดจำลองไมโครคอนโทรลเลอร์ MCS-51 เช่น 7-Segment, LED, DIP Switch เป็นต้น
3. ออกแบบและเขียนโปรแกรมด้วยภาษาแอสเซมบลีเพื่อใช้สร้างบอร์ดจำลองไมโครคอนโทรลเลอร์
4. ทดลอง ทดสอบการใช้งานจริงของระบบทั้งหมด ซึ่งผลการทดสอบปรากฏว่าระบบสามารถทำงานได้จริงตามขอบเขตที่ได้ระบุไว้

5.2 ปัญหาที่พบ

ปัญหาที่พบในการทำงานโครงงานนี้โดยส่วนมากเนื่องจากเป็นการทำงานทางด้านการพัฒนาโปรแกรมดังนั้นปัญหาที่พบมากจะเป็นปัญหาทางด้านการเขียนโปรแกรม ความรู้ทางด้านการเขียนที่ยังจำเป็นต้องศึกษาเพิ่มเติมเพื่อให้เกิดความเข้าใจในโครงสร้างทางภาษาโปรแกรมให้มากขึ้นและในเรื่องของข้อจำกัดทางการใช้งานของ Microsoft Visual J++ โดยเฉพาะการใช้งานที่นำแอพเพลตมาใช้จะเกิดข้อจำกัดขึ้นทำให้เสียเวลาต้องเขียนโปรแกรมบางส่วนเพิ่มเติมขึ้นเอง เช่น การตัดบรรทัดซึ่งใน แอพเพลตไม่มีคำสั่งรองรับเหมือนใน แอพพลิเคชั่น ซึ่งทำให้ต้องสร้างคำสั่งตรงนั้นขึ้นมาเอง

5.3 แนวทางในการพัฒนา

จากโครงงานที่ได้พัฒนานั้นยังมีข้อจำกัดในการทำงานอย่างเช่นไม่มีความสามารถในติดต่อกับพอร์ตอนุกรม ชุดคำสั่งที่ใช้ในการติดต่อกับหน่วยความจำข้อมูลภายนอกและหน่วยความจำโปรแกรมภายนอกตลอดจนความสามารถในการอัดโปรแกรมลงในไมโครคอนโทรลเลอร์จริงๆยังไม่สามารถทำได้แนวทางในการพัฒนาโครงงานต่อควรที่จะพัฒนาศักยภาพการทำงานในเรื่องของความสมบูรณ์ของชุดคำสั่ง การใช้งานในการอัดโปรแกรมลงไมโครคอนโทรลเลอร์ซึ่งรูปแบบนั้นจำเป็นต้องพัฒนาเรื่องฮาร์ดแวร์เพิ่มเติมเพื่อรองรับการทำงานเช่นนี้

บรรณานุกรม

[1] วรพจน์ กรแก้ววัฒนกุล และชัยวัฒน์ ลิ้มพรจิตรวิไล, เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช, อินโนเวตีฟ

[2] พ.อ. เจนวิทย์ เหลืองอร่าม, การเขียนโปรแกรมสำหรับ Applications และ Applets ด้วย JAVA, ซีเอ็ดยูเคชั่น, 2538

[3] ดร. วีระศักดิ์ ซึงถาวร, Fundamental of JAVA PROGRAMMING Volume 1, 2543

[4] ดร. วีระศักดิ์ ซึงถาวร, Fundamental of JAVA PROGRAMMING Volume 2, 2543

[5] Available: http://203.154.220.127/~t4micro/instruction.html

[6] Available: http://www.hospital-os.com/