IBM PC COMMAND THE MCS-51 EMULATOR

โดย
นายกัมพล ยศเรืองสา
นายบุญยพันธ์ ปิ่นแก้ว
นายอรุณ กำเหนิดนนท์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2534

ปริญญานิพนธ์ปีการศึกษา 2534

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง IBM PC Command The MCS-51 Emulator

ผู้จัดทำ

1. นาย กัมพล ยศเรืองสา 33.162101

2. นาย บุญยพันธ์ ปิ่นแก้ว 33.162116

3. นาย อรุณ กำเหนิดนนท์ 33.161136

..... อาจารย์ที่ปรึกษา

(ผศ. พิพัฒน์ เลาหสงคราม)

IBM PC Command The MCS-51 Emulator

กัมพล ยศเรืองสา
บุญยพันธ์ ปิ่นแก้ว
อรุณ กำเหนิดนนท์

ผศ. พิพัฒน์ เลาหสงคราม อาจารย์ที่ปรึกษา

บทคัดย่อ

วิทยานิพนธ์นี้เป็นการเสนอแนวทางการพัฒนาอุปกรณ์ที่เรียกว่า อีมูเลเตอร์ (Emulator) ของไมโครคอนโทรลเลอร์แบบชิพเดี่ยวตระกูล 8051 ผ่านทางพอร์ทอนุกรม มาตรฐาน RS-232C ของ IBM PC หรือเครื่องคอมแพททิเบิล การใช้งานตัวอีมูเลเตอร์มักถูกใช้เป็นเครื่องมือสำหรับช่วยตรวจสอบ หรือแก้ไขระบบไมโครคอมพิวเตอร์ที่มีข้อผิดพลาดขั้นทางซอฟต์แวร์ ทั้งนี้เนื่องจากอีมูเลเตอร์สามารถนำเอาข้อมูลต่าง ๆ ภายในระบบคอมพิวเตอร์ที่บกพร่องขึ้นมาตรวจสอบได้ เช่น ข้อมูลภายในรีจิสเตอร์ ข้อมูลที่เก็บไว้ในหน่วยความจำเป็นต้น และเมื่อผู้ใช้สามารถตรวจสอบข้อมูลได้อย่างถูกต้อง การวิเคราะห์ปัญหาที่เกิดขึ้นจะทำได้อย่างรวดเร็วและแม่นยำ ทำให้ระยะเวลาที่ใช้ในการแก้ปัญหาน้อยลงจากเดิมมาก

ผลจากการทดสอบการใช้งานปรากฏว่า 8051 อีมูเลเตอร์ที่สร้างขึ้นมีประสิทธิภาพทัดเทียมกับอีมูเลเตอร์ราคาแพงที่มีขายอยู่ในปัจจุบันแต่ราคาถูกกว่ามาก

IBM PC Command The MCS-51 Emulator

KUMPON YODRUANGSA

BOONYAPAN PINKAEW

AROON KAMNERDNON

PIPAT LAOHASONGKRAM ADVISOR

ABSTRACT

This paper presents the development of IBM PC Command The Emulator for Single chip microcontroller in 8051 family with RS-232C of IBM PC Computer or compatible. An emulater is frequently used fortesting and solving trouble shoot in microcontroller system having saftware error. An emulater can take some data from failure computer system such as data in registers, data in memory to check and correct. After the user get the correct data, analysis of problem can be easily done and the problem solving time can be greatly reduced.

Experimental results show that the developed 8051 emlator has performances comparaable to the commercially available emulator at present. But the developed one has much less cost.

สารบัญ

บทที่ 1

บทนำ

ปัจจุบันนี้มีการนำเอาระบบ Microcomputer มาเป็นส่วนสำคัญในการควบคุมเครื่องใช้ไฟฟ้า กันอย่างแพร่หลาย เช่น อุปกรณ์ควบคุมการทำงานอัตโนมัติในโรงงานอุตสาหกรรม หรือแม้กระทั่งอุปกรณ์ที่ใช้ในสำนักงานก็พบได้หลายชนิด เครื่องใช้ไฟฟ้าดังกล่าวมีขนาดเล็กลงและมีการทำงานสลับซับซ้อนมากขึ้น สาเหตุที่เป็นเช่นนี้เนื่องมาจากผู้ผลิตมักจะนำไมโครคอมพิวเตอร์แบบชิพเดียวมาใช้งานเป็นจำนวนมาก

ไมโครคอมพิวเตอร์แบบชิพเดี่ยว (Single Chip Microcomputer) จะมีข้อดีกว่าระบบไมโครคอมพิวเตอร์ทั่ว ๆ ไปคือ อุปกรณ์สนับสนุนต่าง ๆ ที่ต้องใช้งานร่วมกับไมโครโปรเซสเซอร์ จะอยู่ภายในชิพเดี่ยวกันกับ ไมโครโปรเซสเซอร์อุปกรณ์เหล่านี้ได้แก่ หน่วยความจำทั้งชนิด ROM และ RAM พอร์ทแบบขนานและพอร์ทแบบอนุกรม ด้วยจุดเด่นข้อนี้ ไมโครคอมพิวเตอร์แบบชิพเดี่ยวจึงมักถูกนำมาใช้งานควบคุมที่มี โปรแกรม ไม่สลับซับซ้อนมากนัก แต่ต้องการความเร็วสูงพอสมควร

เมื่อพิจารณาในอีกด้านหนึ่งจะเห็นว่าการใช้ ไมโครคอมพิวเตอร์ เป็นส่วนควบคุมหลักแล้ว ในกรณีที่อุปกรณ์นั้นชัดข้องหรือมีจุดผิดพลาดในโปรแกรมที่ใช้งาน การตรวจสอบแก้ไขจะมีขั้นตอนที่ยุ่งยาก โดยเริ่มจากตรวจหาจุดผิดพลาดของฮาร์ดแวร์ก่อน โดยอาศัยเครื่องมือวัดเช่น ลอจิกโพรบหรือออสซิลโลสโคปตรวจหาจุดเสียที่เกิดขึ้นตามแผนผังของวงจรที่มีอยู่ ถ้าพบว่าอุปกรณ์ใดทำงานบกพร่องก็ทำการเปลี่ยนใหม่ แต่ปัญหาจะยุ่งยากมากถ้าการผิดพลาดที่เกิดขึ้นนั้นมีสาเหตุมาจากซอฟท์แวร์ เพราะจะทำให้สัญญาณลอจิกต่าง ๆ ที่ป้อนเข้าไปควบคุมหรือกระตุ้นให้ฮาร์ดแวร์ทำงานจะผิดพลาดหมด การแก้ไขโดยใช้เครื่องมือทางฮาร์ดแวร์ที่กล่าวผ่านมาข้างต้นทำได้ยากมาก ทั้งนี้เนื่องจากเมื่อไมโครคอมพิวเตอร์ทำการรันโปรแกรมจะเสมือนกับว่าตัวของมันอยู่ในโลกปิดเลยทีเดียว ที่เป็นเช่นนี้เพราะว่าในขณะหนึ่ง ๆ จะไม่สามารถทราบ

ได้เลยว่า ไมโครโปรเซสเซอร์กำลังทำตามคำสั่งที่ถูกบันทึกไว้ในหน่วยความจำคำสั่งใด เกิดข้อผิดพลาดอะไร ค่าผลลัพธ์ที่ได้ผิดพลาดไปมากน้อยเพียงใด ด้วยเหตุนี้จึงมีการนำเทคนิคของการอีมูเลท (Emulation Techniques) มาใช้งานโดยอาศัยหลักการง่าย ๆ คือทำการสร้างไมโครคอมพิวเตอร์ขึ้นมาอีกชุดหนึ่งที่ทำงานเลียนแบบไมโครโปรเซสเซอร์ตัวนั้นได้ทุกประการ แต่ในขณะเดียวกันผู้ใช้งานสามารถตรวจสอบการทำงานต่าง ๆ ของโปรแกรมได้ตลอดเวลา

1.1 วัตถุประสงค์

การทำปริญญานิพนธ์นี้มีวัตถุประสงค์ที่สำคัญ 3 ประการด้วยกัน คือ ประการแรกเพื่ออำนวยความสะดวกในการที่จะศึกษา โครงสร้าง การทำงานและการเขียนโปรแกรม ของไมโครโปรเซสเซอร์ตระกูล MCS-51 เบอร์ 8031 ซึ่งในปัจจุบันการศึกษาหรือการเรียนการสอนเกี่ยวกับโครงสร้างการทำงานของ ระบบไมโครโปรเซสเซอร์มักใช้ ไมโครโปรเซสเซอร์ตระกูล MCS-51 เป็นตัวอย่าง ทั้งนี้เนื่องจากโครงสร้างของ MCS-51 ง่ายต่อการศึกษาอีกทั้งเอกสาร หรือหนังสือก็เพิ่มขึ้นเป็นอย่างมาก เพียงพอสำหรับใช้ในการค้นคว้า โดยการศึกษานั้นมักจะใช้ ชิงเกิลบอร์ด (Single board) เป็นเครื่องมือในการศึกษาและความยุ่งยากที่ประสบบ่อยมากก็คือการเขียนโปรแกรมในชิงเกิลบอร์ด ดังกล่าวผู้ใช้จะต้องทำการแปลภาษา แอสเซมบลี้ (Assembly) ให้เป็นรหัส (Code) ซึ่งเป็นเลขฐานสิบหกเสียก่อนแล้วจึงทำการป้อนหรือคีย์ (Key) ลงในชิงเกิลบอร์ด แต่ปริญญานิพนธ์นี้เอื้ออำนวยต่อผู้ใช้ในการศึกษาหรือพัฒนาระบบไมโครโปรเซสเซอร์ตระกูล MCS-51 บนเครื่อง IBM PC/XT PC/AT ประการที่สองเพื่ออำนวยความสะดวก และย่นระยะเวลาของนักดิจิตอลอิเล็กทรอนิกส์ในการพัฒนาระบบไมโครโปรเซสเซอร์ ที่ใช้ MCS-51 เป็น CPU และประการสุดท้าย เพื่อศึกษาและพัฒนาต้นแบบในการที่จะประยุกต์ใช้ กับไมโครโปรเซสเซอร์เบอร์อื่น ๆ เช่น ไมโครโปรเซสเซอร์ตระกูล MCS-48 หรือตระกูล MCS-96

1.2 ความเป็นมา

ในการพัฒนาระบบที่ใช้ไมโครโปรเซสเซอร์เป็นส่วนที่คอยควบคุม (Micro Processor Based System) อาจพอแบ่งได้เป็น 2 ส่วน คือ การพัฒนาวงจรหรือฮาร์ดแวร์ (Hardware) และการพัฒนาโปรแกรมหรือซอฟท์แวร์ ในยุคเริ่มต้นการพัฒนาวงจรและการพัฒนาโปรแกรมต้องดำเนินการแยกจากกัน ทำให้ต้องเสียเวลาดำเนินการมาก ต่อมาได้นำเอาระบบพัฒนาไมโครโปรเซสเซอร์(Microprocessor Development System) มาช่วยทำให้การพัฒนาวงจร และการพัฒนาโปรแกรมสามารถกระทำการร่วมกันได้ โดยเฉพาะอย่างยิ่งส่วนที่เกี่ยวกับการหาความผิดพลาดของวงจร และการหาความผิดพลาดของโปรแกรมหรือดีบัก (Debug)

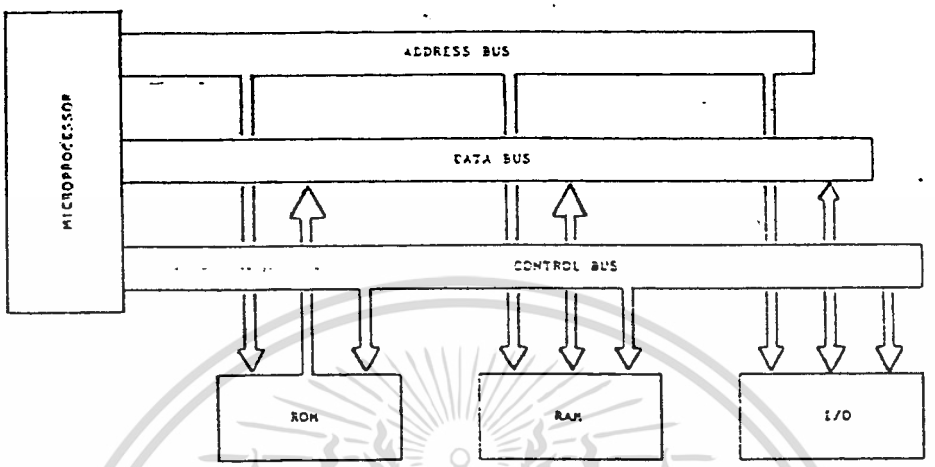
การพัฒนาระบบไมโครโปรเซสเซอร์ที่ใช้กันอย่างแพร่หลายในปัจจุบัน อาจพอแบ่งได้เป็น 2 ประเภทคือ การใช้ In Circuit Emulator (ICE) และ ชิงเกิลบอร์ด ระบบพัฒนาระบบไมโครโปรเซสเซอร์ ประเภทแรกมีขีดความสามารถสูงมาก ทั้งทางด้านการพัฒนาวงจรและการพัฒนาโปรแกรม คือสามารถทำการดีบัก (Debug) และทดสอบวงจรที่พัฒนาทุกส่วนโดยผู้ใช้สามารถเขียนโปรแกรมที่แอดเดรสใด ๆ ได้อย่างอิสระ ความสามารถในการป้องกันการเกิดสัญญาณรบกวนจากวงจรทดสอบ (Target System) เช่นในกรณีที่บัส (Bus) ข้อมูลที่ออกแบบไว้ต่อผิดและสามารถทำให้ ICE แฮงค์ (Hang) ได้ แต่อย่างไรก็ตามถึงแม้ข้อดีและความสามารถของ ICE จะมีมากก็ตามแต่ราคานั้นสูงมากจึงมีผู้นำมาใช้ไม่มากนัก

ส่วนประเภทที่สอง ซึ่งมีราคาต่ำและมีขายกันอย่างแพร่หลาย จึงนิยมใช้กันมากแต่ขีดความสามารถจำกัด เช่น ความสามารถของซีพียู และหน่วยความจำของระบบที่ทดสอบหรือระบบที่กำลังพัฒนา ถูกนำมาใช้ได้ไม่เต็มที่ การหาความผิดพลาดของโปรแกรมโดยเฉพาะในส่วนที่เกี่ยวกับการตั้งจุดหยุดอย่างมีเงื่อนไข (Conditional Breakpoint) และการตามรอยแบบเวลาจริง (Real Time Trace) ซึ่งส่วนใหญ่แล้วมักจะใช้ได้เฉพาะการดำเนินการทีละขั้น (Single Step) นอกจากนี้ยังมีขีดจำกัดในการหาความผิดพลาดของวงจร อีกทั้งยังมีความล่าช้า ที่เกิดขึ้นเสมอในการใช้ชิงเกิลบอร์ดส่วนใหญ่ คือ การที่นักดิจิตอลอีเล็กทรอนิกส์ต้องการพัฒนาโปรแกรมเพื่อ

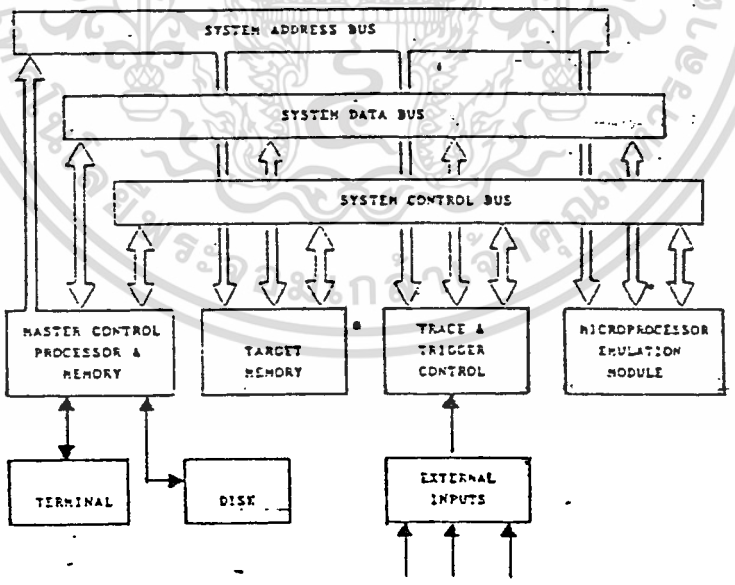
ทดสอบระบบ จะต้องทำการป้อนคำสั่งรหัสดำเนินการ (Opcode) ที่เป็นเลขฐานสิบหกลงในชิงเกิลบอร์ด ทำให้เกิดความไม่สะดวกและเกิดความล่าช้าเพราะจะต้องเสียเวลาให้กับกระบวนการดังกล่าว โดยเฉพาะถ้าโปรแกรมที่จะใช้ในการทดสอบมีขนาดใหญ่มาก อีกทั้งยังทำให้เกิดความผิดพลาดอันเนื่องมาจากตัวของผู้ป้อนข้อมูลเอง หรือความยุ่งยากที่จะต้องทำการแบคอัพไว้เสมอ

การออกแบบระบบโดยทั่ว ๆ ไปซึ่งเริ่มจากการหาข้อระบุเฉพาะหรือขอบเขตของระบบงาน (Specification) แล้วจึงทำการออกแบบระบบ ฮาร์ดแวร์และซอฟท์แวร์ (Hardware Software Design) จากนั้นการพัฒนาระบบจะแยกกันทำไปพร้อม ๆ กันได้คือ ทางด้านฮาร์ดแวร์ จากการออกแบบวงจรในแต่ละส่วน แล้วทำการสร้างวงจรขึ้นมา (Circuit Construction) ตัวอย่างวงจรแต่ละส่วนอย่างเช่น การออกแบบและสร้างในส่วนของหน่วยความจำ ซึ่งจะแยกจากส่วนของอินพุตเอาท์พุต และส่วนของการควบคุม จากนั้นขั้นต่อไปก็คือ การทดสอบในแต่ละส่วนว่ามีความถูกต้อง ตรงตามความต้องการเพียงใด (Circuit Verification) เมื่อทดสอบว่าส่วนต่าง ๆ มารวมเป็นระบบต้นแบบ (Prototype Construction) แล้วทำการตรวจสอบ (Prototype Verification) ในขณะเดียวกันถ้ามองด้านซอฟท์แวร์ผู้พัฒนาระบบก็จะทำการพัฒนาโปรแกรมที่จะใช้ระบบฮาร์แวร์ โดยใช้อิดิทเตอร์ทำการเขียนโปรแกรมแล้วจะใช้ตัวแปรที่อยู่คนละระบบทำการแปล (Assembler) ให้เป็นรหัสดำเนินการ อย่างเช่นในปัจจุบันมักจะพิมพ์รหัสดำเนินการแล้วนำมาป้อนลงชิงเกิลบอร์ดด้วยมือ หรือถ้าจะให้รวดเร็วขึ้นอีกก็จะต้องทำการดาวน์โหลด (Down load) ด้วยวิธีการส่งผ่านทางซีเรียลพอร์ต (Serial Port) หรือทำเป็นแบบ RAM สองทาง ซึ่งจำเป็นจะต้องเพิ่มระบบฮาร์ดแวร์ขึ้นมา เมื่อถึงตอนนี้ระบบต้นแบบของวงจรและซอฟท์แวร์จะต้องนำมารวมให้เป็นระบบรวมก่อน(System Integration) ที่จะทำการทดสอบขั้นสุดท้ายที่จะเป็นผลิตภัณฑ์ต่อไป (Product Test)

ระบบไมโครคอมพิวเตอร์โดยทั่ว ๆ ไปจะประกอบด้วยบัสสามชนิดด้วยกันคือ แอดเดรสบัส ดาต้าบัส และคอนโทรลบัสโดยบัสทั้งสามจะเชื่อมต่อกับอุปกรณ์ต่าง ๆ ดังรูปที่ 1.1



รูป 1.1 แสดงระบบพื้นฐานของไมโครคอมพิวเตอร์ทั่วไป



รูป 1.2 แสดงแผนผังของการ emulate เข้าไปในแผ่นวงจรที่มีไมโครคอมพิวเตอร์เป็นส่วนควบคุมหลัก (Circuit Under Test)

จากรูปที่ 1.1 ถ้านำสัญญาณจากบัสต่าง ๆ ออกมาภายนอกระบบและสามารถควบคุมสัญญาณเหล่านี้ได้โดยผ่านทางแป้นพิมพ์และจอภาพ จะสามารถทราบถึงการทำงานของโปรแกรมรวมถึงข้อมูลที่นำเข้าหรือส่งออกไปภายนอกระบบเพื่อควบคุมอุปกรณ์ทางฮาร์ดแวร์ดังกล่าวมาข้างต้น

จากรูปที่ 1.2 สมมุติว่าแผ่นวงจรที่จะนำมาตรวจสอบมีโครงสร้างดังที่แสดงไว้ในรูปที่ 1.1 โดยถอดชิพไมโครโปรเซสเซอร์ออกแล้วใส่สายเชื่อมต่อจากพอร์ต (Port) เข้าไปแทนดังนั้นสายบัสต่าง ๆ ของแผ่นวงจรจะถูกเชื่อมต่อเข้าไประบบของอีมูเลเตอร์ (Emulator) โดยผ่านทางพอร์ตและไมโครโปรเซสเซอร์อีมูเลชั่นโมดูล (Microprocessor Emulation Module) สำหรับอีมูเลเตอร์ทั่ว ๆ ไปแล้วไมโคร์โปรเซสเซอร์อีมูเลชั่นโมดูลนี้ จะเปลี่ยนแปลงไปตามไมโครโปรเซสเซอร์ของแผ่นวงจรที่นำมาตรวจสอบ โมดูลเหล่านี้จะทำหน้าที่แทนไมโครโปรเซสเซอร์ของแผ่นวงจร ดังนั้นข้อมูลต่าง ๆ ที่อยู่บนแผ่นวงจรสามารถย้ายมายังหน่วยความจำของอีมูเลเตอร์ได้ทันที (Target memory) ทำให้สะดวกในการตรวจสอบแก้ไขทางจอภาพคอมพิวเตอร์ได้ ตามความสามารถที่ผ่านมาข้างต้นจะเห็นว่าเมื่อใช้เทคนิคการอีมูเลทเข้าไปในวงจรของระบบแล้วเสมือนหนึ่งว่าได้เปิดเข้าไปในโลกที่ปิดอยู่ของมันซึ่งจะทำให้การวิเคราะห์ระบบทำได้รวดเร็วขึ้นเป็นอย่างมาก

ในวิทยานิพนธ์ฉบับนี้เป็นการเสนอการสร้างอีมูเลเตอร์อย่างง่าย มาใช้งานโดยเลือกเฉพาะใช้กับไมโครคอมพิวเตอร์แบบชิพเดี่ยวเบอร์ 8031 ของบริษัทอินเทล

บทที่ 2

อีมูเลเตอร์ และ MCS-51 (Emulator and MCS-51)

บทนี้จะกล่าวถึงทฤษฎีและความหมายของคำว่า อีมูเลเตอร์ (Emulator) ซึ่งเป็นที่ถกเถียงกันอยู่ทั่วไปของบรรดานักดิจิตอลอิเล็กทรอนิกส์และนักคอมพิวเตอร์ว่าคำว่าอีมูเลเตอร์คืออะไรและทั่ว ๆ ไปใช้คำนี้ในทางใดบ้างโดยนิยามหรือความหมายในบทนี้ได้อ้างอิงจากการค้นคว้าวารสารทางวิชาการ สิ่งตีพิมพ์(Paper) และหนังสือเป็นสำคัญ นอกจากนี้ยังได้กล่าวถึงสถาปัตยกรรมของ MCS-51 ด้วย

2.1 ความเป็นมา

ในปัจจุบันนี้ระบบคอมพิวเตอร์จะมีเทคโนโลยีในการสร้าง ระบบพัฒนาการที่ใหม่และเป็นรุ่นต่างๆ ออกมาเสมอ อย่างเช่น เมื่อสมัยหนึ่งคอมพิวเตอร์ของยุคนั้นจะมีเครื่องเป็นรุ่นของเครื่อง IBM 360, เครื่อง IBM 7090 หรือระบบไมโครคอมพิวเตอร์ที่มีบัสข้อมูล 8 บิต (Bit) หรือเป็นไมโครคอมพิวเตอร์ที่เป็นชิงเกิลบอร์ดที่ใช้ไมโครโปรเซสเซอร์เบอร์ 8080, Z-80C แต่สมัยต่อมาระบบคอมพิวเตอร์ได้พัฒนาต่อมาอีกหลายรุ่น เช่น เครื่อง IBM 360/65, เครื่อง IBM 360/50 หรือแม้กระทั่งไมโครคอมพิวเตอร์ปัจจุบันเป็นเครื่อง 16 บิต, 32 บิต การที่เกิดพัฒนาการเช่นนี้ถึงแม้ว่าจะเป็นการดีแต่ข้อเสียย่อมมีคือ ในกรณีที่ว่าระบบหรือโปรแกรมของคอมพิวเตอร์รุ่นก่อนไม่สามารถใช้กับคอมพิวเตอร์รุ่นใหม่ได้ จึงได้มีการค้นคิดหาวิธีทางในการที่จะแก้ไขปัญหาดังกล่าว เช่น วิธีการนำเอาภาษาระดับสูง (High - level Language) ในการปรับเปลี่ยนระบบ (Transition)หรือจากระบบหนึ่งให้สามารถทำงานในอีกระบบหนึ่งได้ การใช้การแปลโปรแกรมโดยอัตโนมัติ (Automatic Pro

gram Translation) จะใช้ได้เฉพาะโครงและสถาปัตยกรรมมีลักษณะคล้ายกันหรือมีวิธีการแปลด้วยมือ (Hand Translation) วิธีนี้เป็นวิธีที่ลำบากและเสียเวลาเป็นอย่างมาก จากวิธีการแก้ปัญหาต่าง ๆ เหล่านี้ได้มีชื่อเรียกโดยเฉพาะขึ้นมาชื่อหนึ่งคือ อีมูเลชั่น (Emulation)

2.2 ความหมายของ Emulator

อีมูเลชั่น คือ เครื่องมือหรือวิธีการใดวิธีการหนึ่งที่ใช้ในกระบวนการปรับเปลี่ยนระบบ ซึ่งอีมูเลชั่นนั้นเกิดจากการสร้างหรือขยายบางส่วนเพิ่มเติมในระบบคอมพิวเตอร์ระบบหนึ่งให้สามารถทำงานตามโปรแกรมที่เขียนขึ้นสำหรับระบบคอมพิวเตอร์อื่นได้ และระบบที่ทำได้เช่นนี้เราจึงเรียกว่า อีมูเลเตอร์ (Emulator)

อีมูเลเตอร์ยังอาจที่จะมีความหมายเพิ่มเติมว่าเป็นส่วนประกอบต่าง ๆ คือ ฮาร์ดแวร์ ไมโครโปรแกรม (Microprogram)และซอฟท์แวร์ไปเพิ่มให้กับระบบคอมพิวเตอร์ระบบหนึ่ง เพื่อให้มีความสามารถในการที่จะทำงานตามโปรแกรมที่เขียนขึ้นสำหรับให้ระบบคอมพิวเตอร์อีกระบบหนึ่งทำงาน ซึ่งส่วนประกอบเหล่านี้อาจจะมีไม่ครบก็ได้แต่อย่างน้อยที่สุดจะต้องมีการต่อเติม ฮาร์ดแวร์หรือเพิ่มเติมไมโครโปรแกรม ถ้าไม่มีสิ่งหนึ่งสิ่งใดในสองสิ่งนี้เราเรียกว่า การจำลอง (Simulation)

การอีมูเลชั่นส่วนใหญ่มักกระทำในระดับคอมพิวเตอร์เมนเฟรม เช่น เครื่อง IBM 360/50 ทำการอีมูเลตเครื่อง IBM 360 เครื่อง IBM 360/65 เป็นอีมูเลเตอร์ของ IBM 7090 เครื่อง RCA Spectra/70 เป็นอีมูเลเตอร์ของเครื่อง RCA 301 และระบบของปริญญานิพนธ์นี้ เป็นการนำเครื่องไมโครคอมพิวเตอร์ IBM PC/XT หรือ PC/AT ทำการอีมูเลตระบบไมโครโปรเซสเซอร์ซึ่งเรียกว่า IBM PC Command The MCS-51 Emulator

2.3 โครงสร้างสถาปัตยกรรม MCS-51

ลักษณะหลักทั่วๆ ไปของ MCS-51 จะประกอบด้วย

1. สร้างโดยใช้ HMOS และ CHMOS เทคโนโลยีและทำงานด้วยแหล่งจ่ายไฟขนาด 5 V. เพียงแหล่งเดียว
2. ซีพียูมีขนาด 8 บิต
3. มีวงจรออสซิลเลเตอร์ และวงจรนาฬิกาบนชิป
4. ชุดแบงค์ (BANK) เรจิสเตอร์มี 4 ชุด แต่ละชุดมีเรจิสเตอร์ 8 ตัวทำงานเช่นเดียวกับ MCS-48
5. มีตัวจับเวลา/ตัวนับ ขนาด 16 บิต 2 ชุด และสำหรับเบอร์ 8032/8052 มี 3 ชุด
6. มีพอร์ตไอโอแบบขนานสองทิศทางจำนวน 4 พอร์ต พอร์ตละ 8 บิต รวมทั้งหมดเป็น 32 เส้นแต่จะเหลือเพียง 16 เส้นสำหรับเบอร์ 8031 อีก 16 เส้นใช้ในการเข้าถึงทางแอดเดรสและข้อมูล
7. พอร์ตแบบอนุกรมสามารถที่จะโปรแกรมการรับส่งแบบ Full Duplex ที่ความเร็วสูง
8. หนึ่งวัฏจักรคำสั่งจะใช้เวลา 1 ไมโครวินาที ด้วยการใช้คริสตอล 12 เม็กกะเฮิรตซ์
9. แอดเดรส ข้อมูลภายนอกได้ 64 กิโลไบต์
10. แอดเดรสโปรแกรมภายนอกได้ 64 กิโลไบต์
11. สามารถกำหนดเลขที่อยู่ข้อมูลขนาดไบต์หรือบิตได้โดยตรง
12. มีซอฟท์แวร์แฟล็กสำหรับผู้ใช้ที่จะกำหนดเองได้ถึง 128 ตำแหน่งบิต
13. โครงสร้างอินเตอร์รัพท์ทำได้ 5 แหล่ง และ 6 แหล่งสำหรับ 8032/8052 พร้อมด้วยการจัดไพรโอริตี้ (Priority) ได้ 2 ระดับ
14. ตัวโปรเซสเซอร์สามารถใช้งานแบบบูลีน (Boonlean) ได้สำหรับการใช้งานควบคุม
15. มีคำสั่งคูณ และหารทางฮาร์ดแวร์ทำได้ภายใน 4 ไมโครวินาที

16. ตัวเลขทางคณิตศาสตร์ ใช้ได้ทั้งแบบไบนารี และเดซิมัล
17. การใช้พื้นที่สแต็กสำหรับโปรแกรมย่อยต่าง ๆ ทำได้กว้างขึ้น
18. ชุดคำสั่งของ MCS-51 จะมีมากกว่าชุดคำสั่งของ MCS-48

ตระกูล MCS-51 จะมีทั้งแบบมี ROM ในตัว หรือไม่มี ROM ในตัว หรือมี EPROM บนชิพเดียวกันและจะมีตำแหน่งขาที่เหมือนกัน ตารางที่ 2.1 แสดงถึงตารางรายละเอียดของเบอร์ต่างๆ ในตระกูล MCS-51 ที่มีจำหน่ายในท้องตลาด

เบอร์	หน่วยความจำภายใน		ตัวตั้งเวลา/ตัวนับจำนวน	อินเตอร์รัพท์
	โปรแกรม	ข้อมูล		
8052 AH	8K x 8 ROM	256 x 8 RAM	3 x 16 BIT	6
8051 AH	4K x 8 ROM	128 x 8 RAM	2 x 16 BIT	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16 BIT	5
8032 AH	ไม่มี ROM	256 x 8 RAM	3 x 16 BIT	6
8031 AH	ไม่มี ROM	128 x 8 RAM	2 x 16 BIT	5
8031	ไม่มี ROM	128 x 8 RAM	2 x 16 BIT	5
8751 H	4K x8 EPROM	128 x 8 RAM	2 x 16 BIT	5
8751 H-12	4K x8 EPROM	128 x 8 RAM	2 x 16 BIT	5

ตาราง 2.1 ตารางรายละเอียดของตระกูล MCS-51

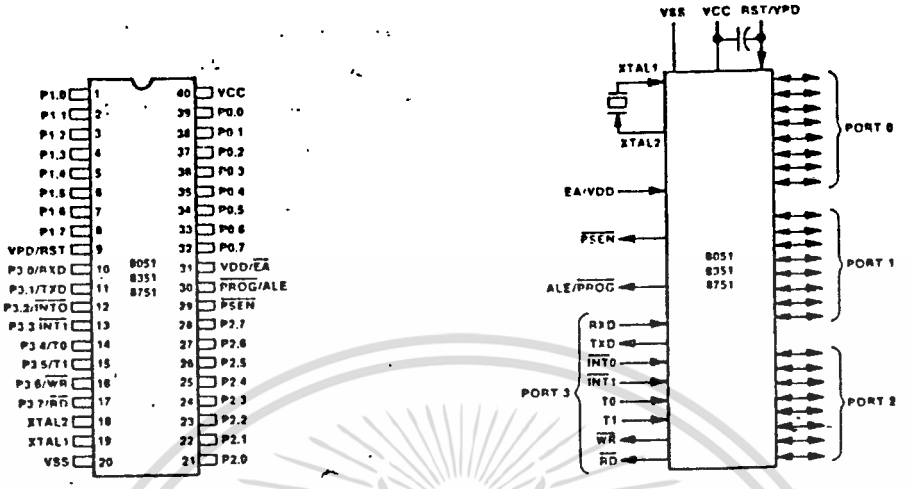
8751H อยู่ในกลุ่มรุ่นเดียวกับ 8051AH ที่เราสามารถโปรแกรมได้ด้วยระบบไฟ สามารถลบโปรแกรมออกได้ด้วยแสงอุลตร้าไวโอเล็ต นอกเหนือจากไอซีที่แสดงในตารางข้างบนที่ใช้เทคโนโลยี HMOS แล้วยังมีตระกูลอื่นที่ใช้เทคโนโลยี CHMOS ที่ประหยัดพลังงานได้มากกว่า 4 เท่าของ HMOS ที่มีจำหน่ายขณะนี้คือเบอร์ 80C51,80C31 และ 87C51

2.3.1 การจัดขาลักษณะภายนอกของ MCS-51

รูปที่ 2.1 แสดงการจัดขาตามลักษณะภายนอกของชิฟ MCS-51 ซึ่งมีรายละเอียดดังนี้คือ

-ขา Vss (ขา 20) เป็นขาสำหรับต่อลงดิน

-ขา Vcc (ขา 40) เป็นขาที่ต่อแรงดันไฟกระแสตรงขนาด 5 V. และใช้สำหรับการโปรแกรมแบบ Open Drain Bidirectional สามารถที่จะรับโหลดทีทีแอล ได้ 8 ตัว การเขียนค่า '1' ไปที่พอร์ตนี้ จะเป็นการปล่อยลอย (Float) ขาของพอร์ตนี้ ทำให้มันทำงานเป็น อินพุต มีสถานะอิมพีแดนซ์สูง ในการให้พอร์ตในการให้พอร์ตนี้บริการแบบไอโอ พอร์ต 0 จะทำงานเป็นมัลติเพลกซ์ ด้วยสัญญาณแอดเดรสไบต์ต่ำกับบัสข้อมูล สำหรับการใช้งานด้านหน่วยความจำภายนอก ในการใช้งานแบบนี้จะใช้ลักษณะภายในเป็นตัวพูลอัพ พอร์ต 0 ยังใช้งานเป็นตัวส่งข้อมูลออกทางพอร์ตนี้ เมื่อใช้บริการทางด้านการตรวจสอบโปรแกรม ROM ภายใน และการโปรแกรมตัว EPROM ภายในถ้าใช้งานในลักษณะนี้การพูลอัพจากภายนอกจะต้องต่อด้วยค่า 10 กิโลโอห์ม



รูป 2.1 (a)ลักษณะขาภายนอกของ MCS-51 (b)สัญญลักษณ์ทางตรรกของ MCS-51

-ขา Port 1 (P1.0 - P1.7) (ขา 1 - 8) เป็นพอร์ตไอโอ 8 บิตแบบ Open Drain Bidirectional พร้อมด้วยพูลอัพภายใน ถ้าเป็นพอร์ตเอาต์พุต บัฟเฟอร์สามารถขับโหลด ทีทีแอล ตระกูล แอลเอสไอ ได้ 4 ตัว พอร์ต 1 เมื่อถูกเขียนค่า '1' ด้วยโปรแกรมมันจะมีสถานะสูง ด้วยการพูลอัพภายใน การให้สถานะเช่นนี้ จะเป็นการ Initial ใช้งานพอร์ตนี้ให้เป็นอินพุตขณะที่พอร์ต 1 เป็นอินพุต การให้สัญญาณลงต่ำจะเป็นการจ่ายกระแสออกเนื่องจากการพูลอัพภายใน

ในเบอร์ 8052 ขา P1.0 และ P1.7 จะใช้งานจะใช้งานเป็น T2 และ T2EX โดยขา T2 จะทำหน้าที่รับสัญญาณจากภายนอกให้ตัวตั้งเวลา 2 ทำงาน และขา T2EX จะเป็นอินพุตผ่านเข้าตัวตั้งเวลา 2 ถูกกระตุ้นให้ทำงานแบบปกติตามโปรแกรมที่ตั้งไว้ หรือ (Capture)

-ขา Port 2 (P2.0 - P2.7) (ขา 21 - 28) เป็นพอร์ตไอโอ 8 บิต แบบ Open Drain Bidirectional ด้วยการพูลอัพภายในพอร์ต 2 ที่ทำหน้าที่เป็นบัฟเฟอร์เอาต์พุตสามารถจ่ายโหลดทีทีแอลตระกูลแอลเอสไอได้ 4 ตัว พอร์ตจะถูก

ใช้งานเป็นตัวส่งแอดเดรสไบต์สูงด้วย เมื่อใช้งานร่วมกับหน่วยความจำภายนอก เพื่อให้แอดเดรสได้ถึง 16 บิต ด้วยการใช้งานแบบนี้ มันจะมีพูลอัพภายในที่ช่วยในการส่งค่า '1' ได้ระดับที่แน่นอน นอกจากการใช้งานสำหรับแอดเดรสอันสูงยังใช้เป็นขาควบคุมในการใช้งานตรวจสอบ และเขียนโปรแกรมเบอร์ 8751 และตรวจสอบโปรแกรมภายใน 8051

-ขา Port 3 (P3.0 - P3.7) (ขา 10 -17) เป็นพอร์ตไอโอ 8 บิตแบบพูลอัพภายในนอกจากทำเป็นพอร์ตไอโอที่สามารถรับโหลด ทีทีแอลพวกตระกูลแอลเอสไอ ได้ 4 ตัว แล้วยังใช้งานเป็นพิเศษสำหรับตระกูล MCS-51 ตามรายการข้างล่างนี้ด้วย

ขาพอร์ต	ขา	การทำงานตามฟังก์ชั่นพิเศษ
P 3.0	10	RxD พอร์ตอนุกรมอินพุต
P 3.1	11	TxD พอร์ตอนุกรมเอาท์พุต
P 3.2	12	INT0 อินเตอร์รัพท์ภายนอกตัวที่ 1
P 3.3	13	INT1 อินเตอร์รัพท์ภายนอกตัวที่ 2
P 3.4	14	T0 สัญญาณกระตุ้นเข้าที่ตัวตั้งเวลาและตัวนับ 0
P 3.5	15	T1 สัญญาณกระตุ้นเข้าที่ตัวตั้งเวลาและตัวนับ 1
P 3.6	16	WR สัญญาณควบคุมการเขียน
P 3.7	17	RD สัญญาณควบคุมการอ่าน

การที่จะให้ทำงานตามฟังก์ชั่นข้างบน จะต้องเริ่มโปรแกรมด้วยการส่งค่า '1' ไปแลทช์ไว้ก่อนที่ให้ทำงานตามฟังก์ชั่นข้างบน

-ขา RST (ขา 9) ต้องคงสถานะค่าสูงเป็นเวลาประมาณอย่างน้อย 2 วัฏจักรระหว่างที่ออสซิเลเตอร์ทำงานขณะที่ต้องการรีเซ็ตทั้งระบบงาน โดยจะต่อรีจิส

เตอร์พูลดาวน์ (8.2 กิโลโอห์ม) จากขา RST ไปลงดิน และเพื่อให้ตัวชิฟรีเซ็ตได้โดยอัตโนมัติ ขณะเปิดไฟจะใช้คาปาซิเตอร์ (10 ไมโครฟารัด) ต่อคร่อมระหว่างขา RST กับขา Vcc

-ขา ALE/PROG (ขา 30) เป็นขาแอดเดรสแลทช์อีนาเบิ้ลด้วยการส่งพัลส์ออกไปใช้สำหรับแลทช์ค่าแอดเดรสไบต์ต่ำจากพอร์ต 0 ในระหว่างการเข้าถึงข้อมูลจากหน่วยความจำภายใน ALE จะถูกส่งสัญญาณออกมาในอัตราความเร็วคงที่ ที่ 1/8 ของความถี่ออสซิเลเตอร์ตลอดเวลา แม้ว่าจะไม่มีการเข้าถึงข้อมูลจากภายใน ดังนั้นจึงสามารถที่จะใช้สัญญาณจากขานี้เป็นตัวตั้งเวลาภายนอกหรือเป็นความถี่สัญญาณนาฬิกา แต่อย่างไรก็ตามความถี่สัญญาณนี้จะลดความถี่ช้าลงไปเท่าหนึ่งระหว่างการทำงานแบบการเข้าถึงของหน่วยความจำข้อมูลภายนอก ขานี้ยังใช้เป็นสัญญาณพัลส์เข้าสำหรับการควบคุมการโปรแกรม EPROM ภายในชิพ

-ขา PSEN (ขา 29) Program Storage Enable เป็นสโตรบอ่านข้อมูลจากโปรแกรมหน่วยความจำภายนอก เมื่อชิฟทำงานด้วยโปรแกรมภายนอก ขา PSEN จะสร้างสโตรบต่ำสองครั้งภายในแต่ละวัฏจักรแมชีน สัญญาณจะมีสถานะสูง หรือพัลส์ต่ำทั้งสองลูกจะหายไป เมื่อทำงานในช่วงการอ่าน หรือเขียนข้อมูลจากหน่วยความจำข้อมูลภายนอก และ PSEN จะไม่มีพัลส์ส่งออกถ้าชิฟทำงานด้วยโปรแกรมหน่วยความจำภายใน

-ขา $\overline{EA}$/Vdd (ขา 31) มีสถานะสูงตัวซีพียูในชิฟจะทำงานตามโปรมแกรมที่อยู่ในหน่วยความจำภายใน (โดยที่โปรแกรมจะต้องไม่ยาวกว่า 4 กิโลไบรต์ สำหรับเบอร์ 8051 AH และ 8 กิโลไบต์ สำหรับเบอร์ 8052 AH) การทำให้ EA มีสถานะต่ำจะเป็นการควบคุมให้ซีพียูทำงานตามโปรแกรมหน่วยความจำภายนอก ซึ่งขยายโปรมแกรมได้ยาวถึง 64 กิโลไบต์ ในตัว 8031 AH และ 8032 AH ขา EA จะต้องต่อลงดินเช่นกันแม้ว่าจะไม่มี ROM อยู่ภายในก็ตามในตัว 8751 AH จะใช้ขานี้จ่ายแรงดันขนาด 21 V ขณะทำการเขียนโปรแกรมเข้า EPROM ของชิพ

-ขา XTAL1 (ขา 19) ใช้เป็นตัวอินพุตเข้าสู่ตัวออสซิเลเตอร์ ขยายแบบ Invert

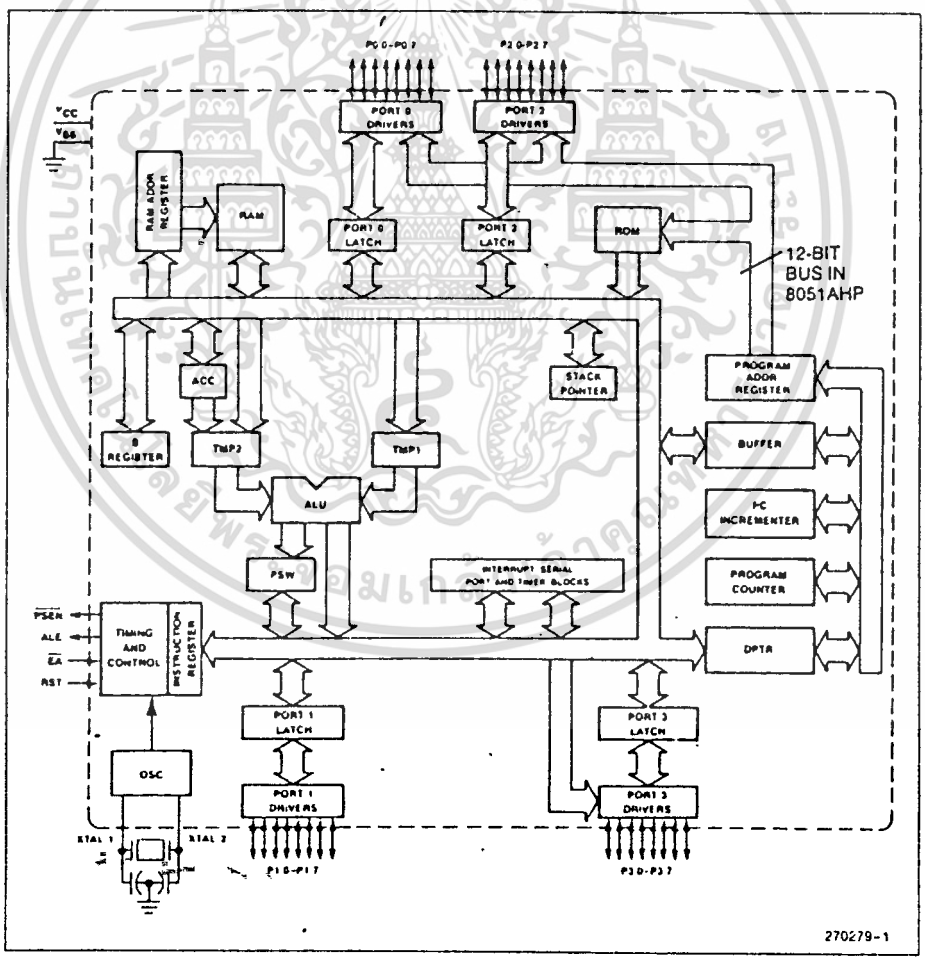
-ขา XTAL2 (ขา 18) ใช้เป็นตัวเอาท์พุตจากตัวออสซิเลเตอร์ขยายแบบ Invert

ตามตาราง 2.1 MCS-51 ทั้งสามกลุ่มคือกลุ่มที่มี ROM ไม่มี ROM และพวก EPROM จะมีที่ขาใช้งานเหมือนกันหมด ยกเว้นขา 1 จะใช้งานเป็น T2 และขา 2 เป็น T2EX ในเบอร์ 8032/8052 ตลอดถึงจังหวะเวลา (Timing Diagram) และคุณสมบัติทางไฟฟ้าทั้งสามจะแตกต่างกันเฉพาะการโปรแกรมบนชิพ MCS-51 เท่านั้นซึ่งแต่ละแบบจัดไปตามความต้องการของผู้ใช้ เช่น 8751 AH จะมี 4 กิโลไบต์ของ Ultraviolet Erasable Programmable Read Only Memory(EPROM) เหมาะสำหรับการพัฒนาเครื่องต้นแบบ และการผลิตอุปกรณ์ที่มีจำนวนจำกัด เมื่อต้องการจะเขียนโปรแกรมเข้า EPROM จะมีตัวเขียนโปรแกรมพิเศษสำหรับเขียนโปรแกรมที่ผู้ออกแบบเขียนขึ้นมาถ้าโปรแกรมมีบักหรือส่วนผิดพลาดที่ต้องการจะแก้ไข ก็สามารถแก้ไขได้โยการนำตัว 8751 AH นี้ไปล้างโปรแกรมเดิมออกด้วยแสงอุลตร้าไวโอเลต และอัดข้อมูลโปรแกรมที่ได้แก้ไขแล้วเข้าไปใหม่ ทำเช่นนี้จนกระทั่งได้โปรแกรมสมบูรณ์และเมื่อต้องการผลิตจำนวนมากก็สามารถที่จะใช้ MCS-51 เบอร์ 8051 ที่มี 4 กิโลไบต์ของ ROM ซึ่งจะถูกอัดข้อมูลโปรแกรมตามความต้องการของผู้ออกแบบโดยโรงงานผู้ผลิตชิพเบอร์นี้ การผลิตลักษณะนี้จะถูกกว่าการใช้เบอร์ 8751 แต่โปรแกรมภายในจะไม่สามารถลบ และโปรแกรมใหม่ได้หลังการผลิตไปแล้ว

ส่วนเบอร์ 8031 จะไม่มีหน่วยความจำของโปรแกรมบนชิพ แต่อาจต่อหน่วยความจำโปรแกรมจากภายนอกด้วย ROM EPROM หรือ PROM ได้ถึง 64 กิโลไบต์ดังนั้น 8031 จึงเหมาะสำหรับการใช้งานที่โปรแกรมมีขนาดใหญ่กว่า 4 กิโลไบต์และสำหรับผู้ออกแบบที่ต้องการแยกส่วนของโปรแกรมออกจากชิพ

2.3.2 การจัดการทางสถาปัตยกรรม

รูปที่ 2.2 เป็นบล๊อกไดอะแกรมที่แบ่งตามลักษณะงานทางด้านสถาปัตยกรรมภายในของ MCS-5 1 โดยชิงเกิลชิพแต่ละตัวของตระกูลนี้จะประกอบด้วยหน่วยศูนย์กลางประมวลผลหน่วยความจำสองชนิดคือ แบบ RAM กับ ROM หรือ EPROM พอร์ตเอาต์พุต อินพุต โหมดเรจีสเตอร์สถานะและข้อมูล ส่วนวงจรตรรกในการ RANDOM ที่จำเป็นสำหรับตัวแปรของฟังก์ชันการต่อพ่วงส่วนต่าง ๆ ที่กล่าวนี้จะติดต่อกันด้วยบัสข้อมูลขนาด 8 บิต และจะมีบัฟเฟอร์สำหรับการติดต่อกับภายนอกผ่านพอร์ตไอโอ เมื่อต้องการขยายหน่วยความจำหรือพอร์ตไอโอ



รูปที่ 2.2 สถาปัตยกรรม MCS-51

แอกคิวมูเลเตอร์ (Accumulator : Acc)

MCS-51 ก็เช่นเดียวกับ MCS-48 ที่ใช้ ACC ที่มีขนาด 8 บิตเป็นแอกคิวมูเลเตอร์หลักค่าสั่งส่วนใหญ่จะอ้างถึงตัวรีจีสเตอร์นี้ โดยถือค่าภายในเป็นค่าตัวตั้งและรับค่าผลลัพธ์ที่ได้จากค่าสั่งทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร เข้ามาเก็บไว้ในตัว ACC ยังสามารถใช้ตัวแหล่งกระทำหรือถูกกระทำ ในการทำงานทางตรรกและใช้เป็นตัวกลางในการถ่ายเทข้อมูลในการติดต่อกับอุปกรณ์ภายนอกไอโอ และหน่วยความจำภายนอกรวมถึงการตรวจสอบตารางข้อมูล

รีจีสเตอร์ B

เป็นรีจีสเตอร์พิเศษที่ใช้งานสำหรับค่าสั่งของการคูณและหาร โดยใช้เป็นที่เก็บตัวคูณหรือตัวหารและเป็นที่เก็บผลลัพธ์ตัวที่สองหลังการคูณและเศษของการหาร

รีจีสเตอร์คำแสดงสถานะโปรแกรม (Program Status Word : PSW)

รีจีสเตอร์ PSW เป็นรีจีสเตอร์ที่แสดงผลที่ได้หลังจากการใช้คำสั่งต่าง ๆ และใช้เป็นตัวเลือกกลุ่มการทำงานของรีจีสเตอร์กลุ่มต่าง ๆ

ตัวชี้สแต็ก (Stack Pointer : SP)

MCS-51 จะรวมเอาสแต็กทางฮาร์ดแวร์ที่ใช้ RAM ภายในสำหรับการเชื่อมต่อระหว่างโปรแกรมหลัก สแต็กการผ่านพารามิเตอร์ระหว่างงานในแต่ละส่วนโปรแกรมและสแต็กเก็บตัวแปรข้อมูลชั่วคราว หรือสแต็กการเก็บสถานะระหว่างการบริการงานอินเตอร์รัพท์ไว้ภายในชิฟโดยที่ SP จะมีขนาด 8 บิต จะเพิ่มค่าขึ้นโดยอัตโนมัติก่อนที่ข้อมูลจะนำมาเก็บในหน่วยความจำระหว่างการใช้คำสั่ง PUSH และ CALL และจะ

ลดค่าของ SP ลงจากที่ได้ถ่ายเทข้อมูลออกไปแล้วในคำสั่ง POP หรือ RETURN โดยทฤษฎีทางสถาปัตยกรรม MCS-51 สามารถใช้สแต็กให้มีเนื้อที่ถึง 128 ไบต์ แต่ในทางปฏิบัติสำหรับโปรแกรมทั่วไปจะใช้น้อยกว่านี้ SP จะเริ่มที่ตำแหน่ง 07H ดังนั้นสแต็กจะเริ่มบรรจุข้อมูลที่ตำแหน่ง 08H MCS-51 สามารถเปลี่ยนแปลงค่าใน SP ได้ ซึ่งจะเป็นการเปลี่ยนตำแหน่งสแต็กไปยังที่ใด ๆ ของ RAM ภายในชิพ

ตัวชี้ข้อมูล (Data Pointer : DPTR)

DPTR เป็นรีจีสเตอร์ขนาด 16 บิต ที่ประกอบด้วยไบต์สูง (DPH) และไบต์ต่ำ (DPL) ที่เราสามารถเลือกแบ่งออกเป็นรีจีสเตอร์ 8 บิตสองตัว ที่ใช้ได้อย่างอิสระ หรือจะใช้ร่วมกันทั้ง 16 บิต ก็ได้ ในการ Increment หรือ Decrement เพื่อประโยชน์ในการใช้เป็นฐานของเลขที่อยู่ในเรจีสเตอร์ในการกระโดดโดยทางอ้อมในการใช้คำสั่งเกี่ยวกับตารางข้อมูลและชี้ตำแหน่งของหน่วยความจำภายนอก

พอร์ต 0 ถึง 3

รีจีสเตอร์ P0, P1, P2 และ P3 ของกลุ่มรีจีสเตอร์ฟังก์ชันพิเศษ(Special Function Register : SFR) จะเป็นตัวเรจีสเตอร์ที่แลทช์ค่าของพอร์ต 0,1,2 และ 3 ตามลำดับ ในขณะใช้งาน

รีจีสเตอร์ CAPTURE

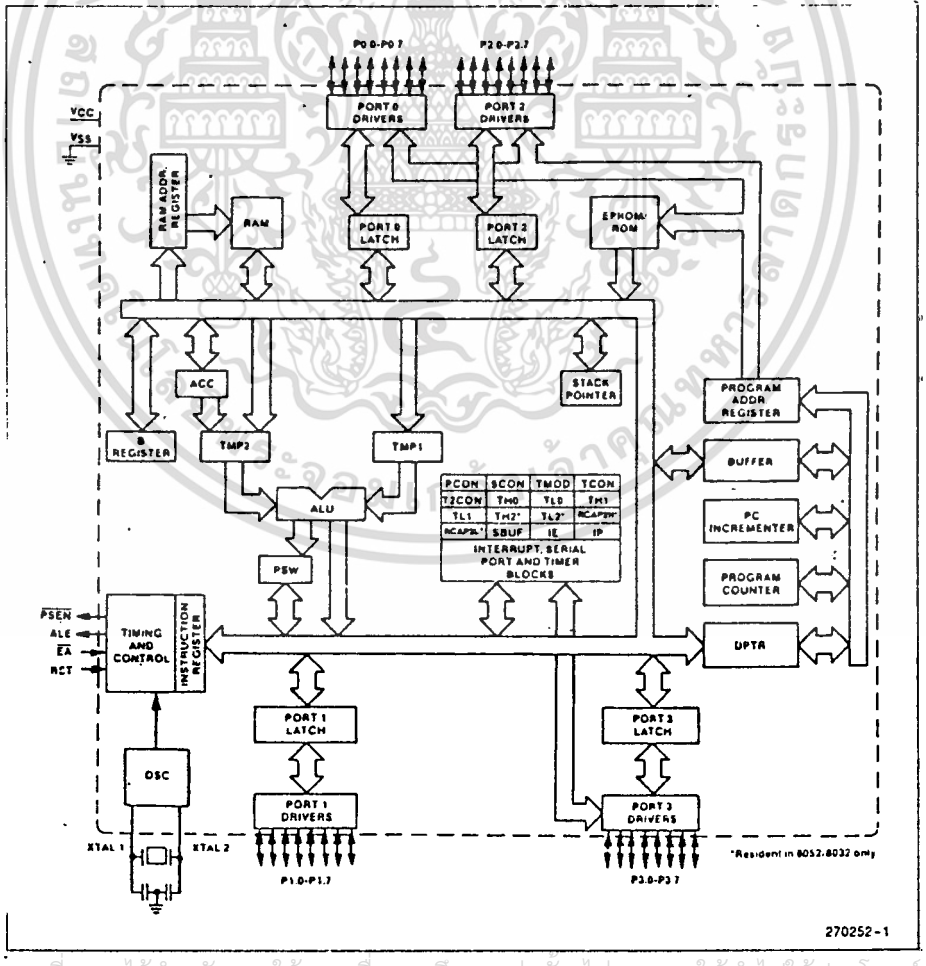
ไอซีเบอร์ 8032/8052 จะมีคู่รีจีสเตอร์ (RCAP2H,RCAP2L) เพิ่มเติมเป็นรีจีสเตอร์เค็ปเจอร์ สำหรับตัวตั้งเวลาหมายเลข 2 ในโหมดการใช้งานของรีจีสเตอร์ตัวนี้จะรับการเปลี่ยนแปลงที่เข้ามาที่ขา T2EX ตัว TH2 และ TL2 จะลอกข้อมูลเข้าไปในรีจีสเตอร์คู่ RCAP2H และ RCAP2L ด้วยการใช้ตัวตั้งเวลา จะมีโหมดการบรรจุอัตโนมัติขนาด 16 บิตสำหรับการใช้ตัวตั้งเวลา/ตัวนับ 2

บัฟเฟอร์ข้อมูลอนุกรม (Serial Data Buffer : SBUF)

บัฟเฟอร์ข้อมูลอนุกรมแบ่งออกเป็นรีจีสเตอร์สองตัว ตัวหนึ่งเป็นบัฟเฟอร์การส่ง และอีกตัวเป็นบัฟเฟอร์การรับ เมื่อข้อมูลถ่ายเทเข้า SBUF ขึ้นอยู่กับการเริ่มแรก (Initial) การส่งข้อมูลย้ายออกจาก SBUF จะเป็นการรับข้อมูลจากบัฟเฟอร์ตัวรับ

รีจีสเตอร์ควบคุม (Control Register)

กลุ่ม SFR ที่เป็น IP,IE,TMOD,TCON,T2CON,SCON และ PCON จะประกอบด้วยบิตที่ใช้ในการควบคุม และแสดงสถานะของการใช้งานในระบบอินเตอร์รัพตัวตั้งเวลา/ตัวนับ



รูปที่ 2.3 MCS-51 Architectural Block Diagram

การจัดการหน่วยความจำ

MCS-51 แบ่งตามพื้นฐานหน่วยความจำของการกำหนดเลขที่อยู่แอดเดรส ได้เป็น 3 ส่วนที่ประกอบด้วยเนื้อที่

64 กิโลไบต์ หน่วยความจำโปรแกรม

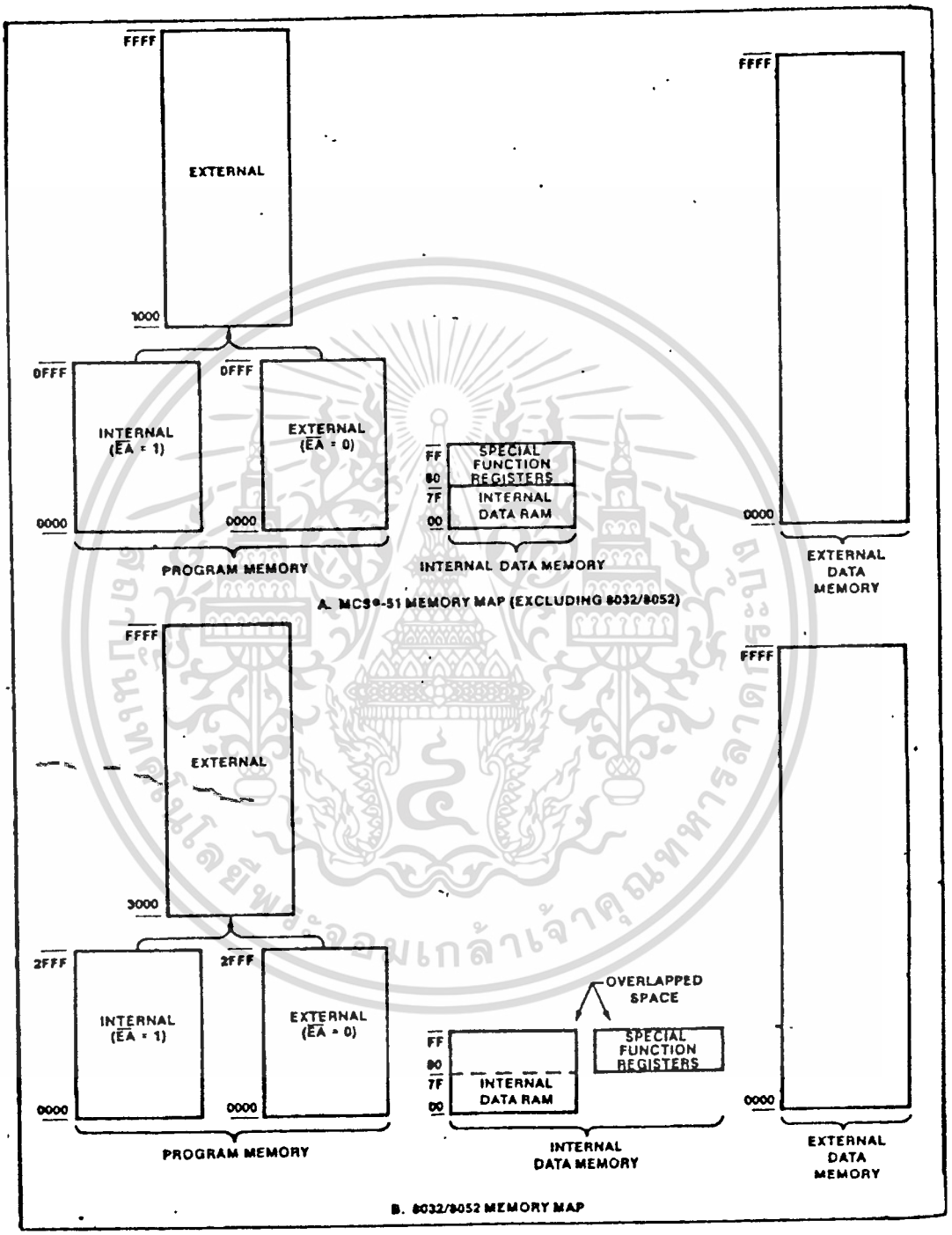
64 กิโลไบต์ หน่วยความจำข้อมูลภายนอก

256 ไบต์ เป็นหน่วยความจำข้อมูลภายใน ส่วนเบอร์ 8032/8052 มีขนาด 384 ไบต์

เนื้อที่หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมจะประกอบด้วย ส่วนภายในและภายนอกชิพ ถ้าขา EA มีสถานะสูง MCS-51 จะบริการโปรแกรมภายใน ถ้าโปรแกรมมีความยาวไม่เกิน OFFFH (4K) หรือ 1FFFH (8K) สำหรับตัว 8052 ตำแหน่งตั้งแต่ 1000H ถึง OFFFFH (หรือ 2000H - OFFFFH สำหรับ 8052) จะเป็นการเฟทช์ข้อมูลภายนอก ถ้า ขา EA มีสถานะต่ำ MCS-51 จะเฟทช์ข้อมูลภายนอกทั้งหมด ในทุกกรณีตัวนับโปรแกรมขนาด 16 บิต จะเป็นตัวกำหนดเลขที่อยู่โปรแกรม

ตำแหน่ง 00 ถึง 32H (หรือ 00 ถึง 2BH สำหรับเบอร์ 8032/8052) ในหน่วยความจำโปรแกรมจะสำรอง สำหรับใช้บริการการอินเตอร์รัพท์ตามตารางที่ 2.2

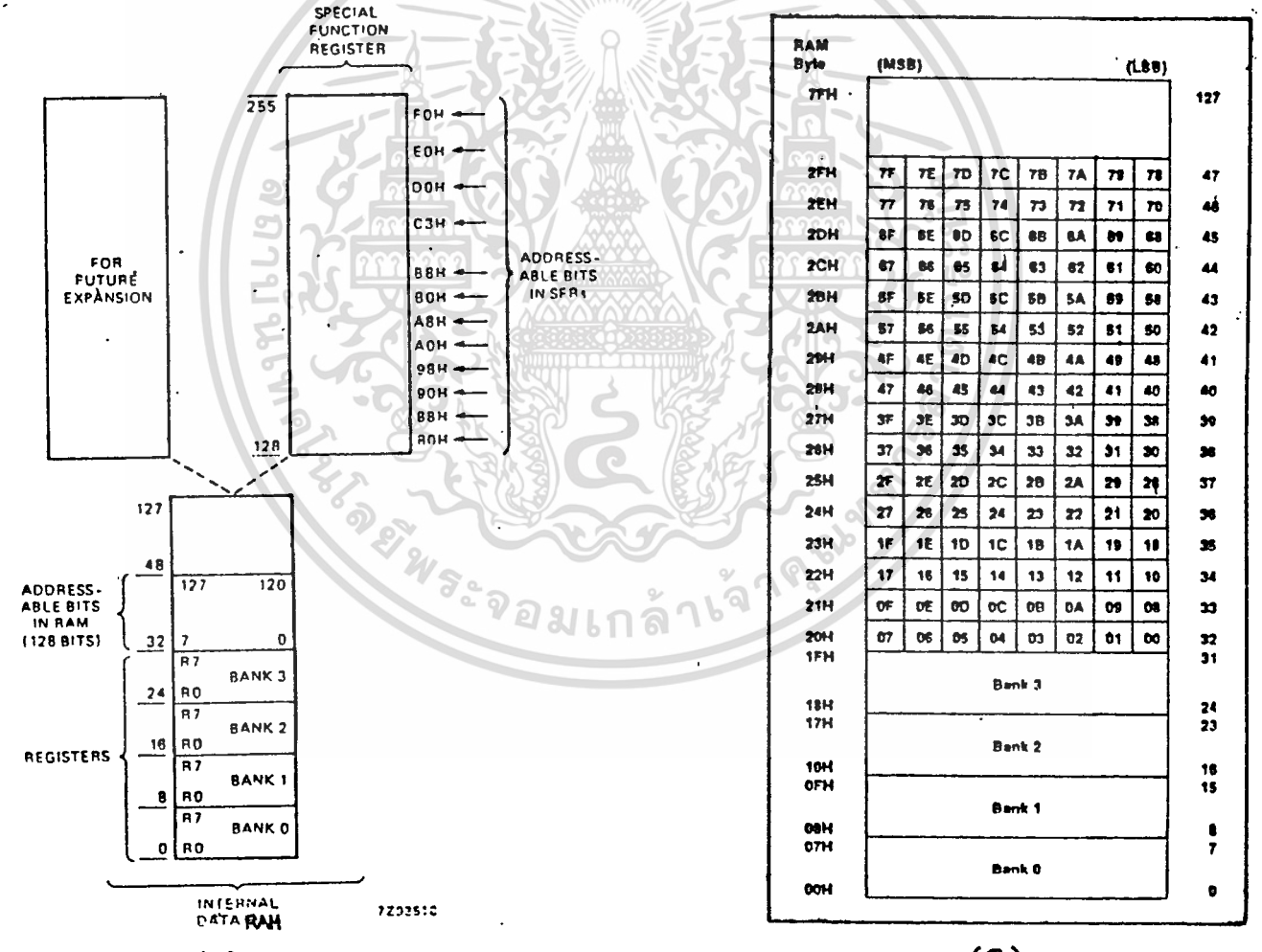


ตาราง 2.2 แสดงแผนภูมิพื้นที่ของหน่วยความจำใน MCS-51

เนื้อที่หน่วยความจำข้อมูล

หน่วยความจำข้อมูลจะประกอบด้วยความจำข้อมูลภายในและภายนอก หน่วยความจำข้อมูลภายนอกจะเข้าถึงได้ด้วยการใช้คำสั่ง MOVX

หน่วยความจำข้อมูลภายในจะแบ่งเป็นลักษณะงาน ดังนี้คือ 1.จำนวน 128 ไบต์ ของบริเวณตำแหน่งล่างในเนื้อที่แรมภายใน 2.และอีก 128 ไบต์เป็นของบริเวณตำแหน่งบนของแรมภายใน ส่วนบนนี้จะมีเฉพาะในเบอร์ 8032/8052 เท่านั้น และส่วนของ 128 ไบต์ อีกบริเวณหนึ่ง ใช้เป็นรีจิสเตอร์ฟังก์ชันพิเศษ ขณะที่ใช้ส่วน



รูปที่ 2.4 A แสดงแผนที่ของหน่วยความจำข้อมูล
B แสดงแผนที่การกำหนดตำแหน่งบิต

บนของแรมภายในและบริเวณของ SFR ทั้ง 2 ส่วนนี้ จะถูกปันส่วนให้ใช้ค่าแอดเดรสภายใน แต่จะเข้าถึงในแต่ละบริเวณของทั้ง 2 บริเวณนี้ได้ ด้วยการใช้โหมดการกำหนดเลขที่อยู่ต่างกัน ซึ่งโหมดเหล่านี้จะอธิบายในหัวข้อต่อไป

รูปที่ 2.4A แสดงถึงแผนที่ของหน่วยความจำข้อมูล โดยแบ่งเป็น 4 BANK ในแต่ละ BANK มีรีจีสเตอร์ 8 ตัวมีตำแหน่งตั้งแต่ 0-31H ในบริเวณของ RAM แบงค์เหล่านี้ จะถูกเลือกใช้ให้อีนาเบิ้ลได้คราวละ 1 แบงค์ ด้วยการกำหนดเริ่มแรกภายใน 2 บิทของรีจีสเตอร์ PSW ว่าจะเลือกใช้ในแบงค์ใดภายใน 4 แบงค์ และบริเวณ

Direct Byte Address	(MSB)			Bit Addresses				(LSB)	Hardware Register Symbol
240	F7	F6	F5	F4	F3	F2	F1	F0	B
224	E7	E6	E5	E4	E3	E2	E1	E0	ACC
	CY	AC	F0	RS1	RS0	OV		P	
208	D7	D6	D5	D4	D3	D2	D1	D0	PSW
				PS	PT1	PX1	PT0	PX0	
184	—	—	—	BC	BB	BA	B9	B8	IP
176	B7	B6	B5	B4	B3	B2	B1	B0	P3
	EA			ES	ET1	EX1	ET0	EX0	
168	AF	—	—	AC	AB	AA	A9	A8	IE
160	A7	A6	A5	A4	A3	A2	A1	A0	P2
	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	
152	9F	9E	9D	9C	9B	9A	99	98	SCON
144	97	96	95	94	93	92	91	90	P1
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
136	8F	8E	8D	8C	8B	8A	89	88	TCON
128	87	86	85	84	83	82	81	80	P0

รูปที่ 2.5 ตำแหน่งของ เรจีสเตอร์ SFR และ บิตแอดเดรสของ SFR

ตำแหน่งตั้งแต่ 20H - 2FH จำนวน 16 ตำแหน่ง ตำแหน่งละ 1 ไบต์ สามารถที่จะกำหนดเลขที่อยู่ของแต่ละบิตได้ ดังแสดงในรูปที่ 2.4B เป็นบิตแรมแอดเดรส เนื้อที่รีจีสเตอร์ SFR สามารถที่จะกำหนดตำแหน่งได้เช่นกัน ดังรูปที่ 2.5

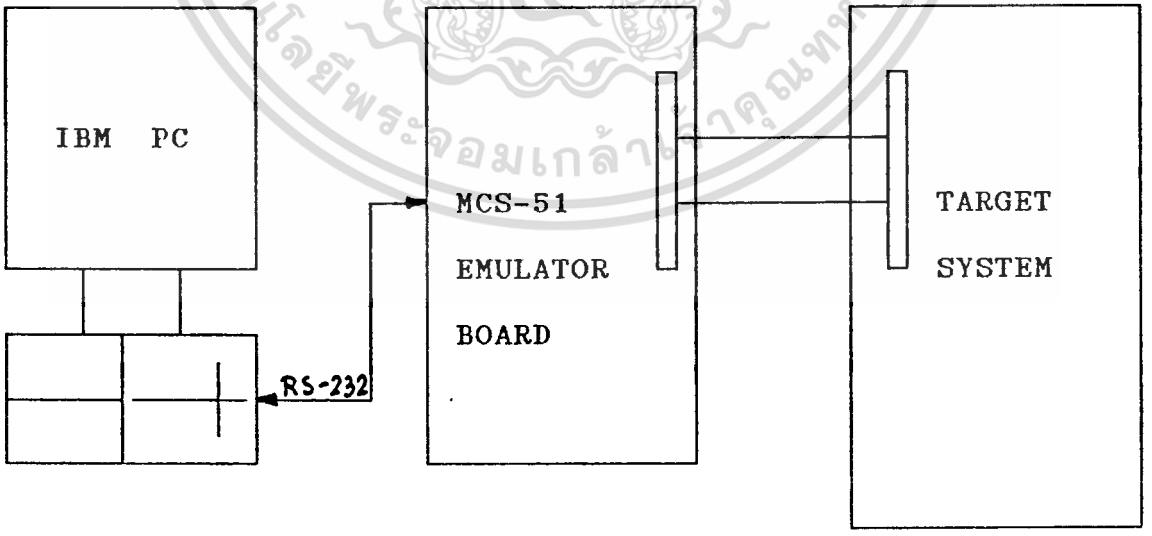
บทที่ 3

การออกแบบ ระบบฮาร์ดแวร์

(Hardware Design)

3.1 สถาปัตยกรรม ทางด้านฮาร์ดแวร์

MCS-51 อีมูเลเตอร์ ที่ได้พัฒนาขึ้นนี้อยู่ในรูปของบอร์ดที่รวมวงจรต่าง ๆ เพื่อใช้แทนไมโครคอนโทรลเลอร์ MCS-51 หนึ่งตัวโดยสามารถนำบอร์ดนี้ไปเสียบแทน MCS-51 ในบอร์ดที่ผู้ใช้กำลังพัฒนาได้โดยวงจรหรือ prototype ยังทำงานตามปกติ นอกจากนี้ยังมีส่วนที่ติดต่อกับ IBM PC ทางพอร์ตอนุกรม RS-232C เพื่อจัดการเกี่ยวกับเรื่องไฟล์ต่างๆการดูค่า รีจีสเตอร์ ค่า SFR หรือค่าหน่วยความจำได้ในงานที่ผู้ใช้กำลังพัฒนา



รูปที่ 3.1 โครงสร้างพื้นฐานของ MCS-51 อีมูเลเตอร์บอร์ด

ระบบพัฒนาของ MCS-51 อีมูเลเตอร์ มีส่วนประกอบที่สำคัญ 3 ส่วนคือ

-IBM PC ทำหน้าที่เปรียบเสมือนเป็นเครื่องปลายทาง (Terminal) การทำงานทั้งหมดของ MCS-51 อีมูเลเตอร์บอร์ดจะถูกควบคุมจากส่วนนี้ เช่นการดีบักโปรแกรม การดู Memory เป็นต้น การติดต่อสื่อสารกับ MCS-51 อีมูเลเตอร์บอร์ดจะกระทำผ่านทาง Serial Port RS-232

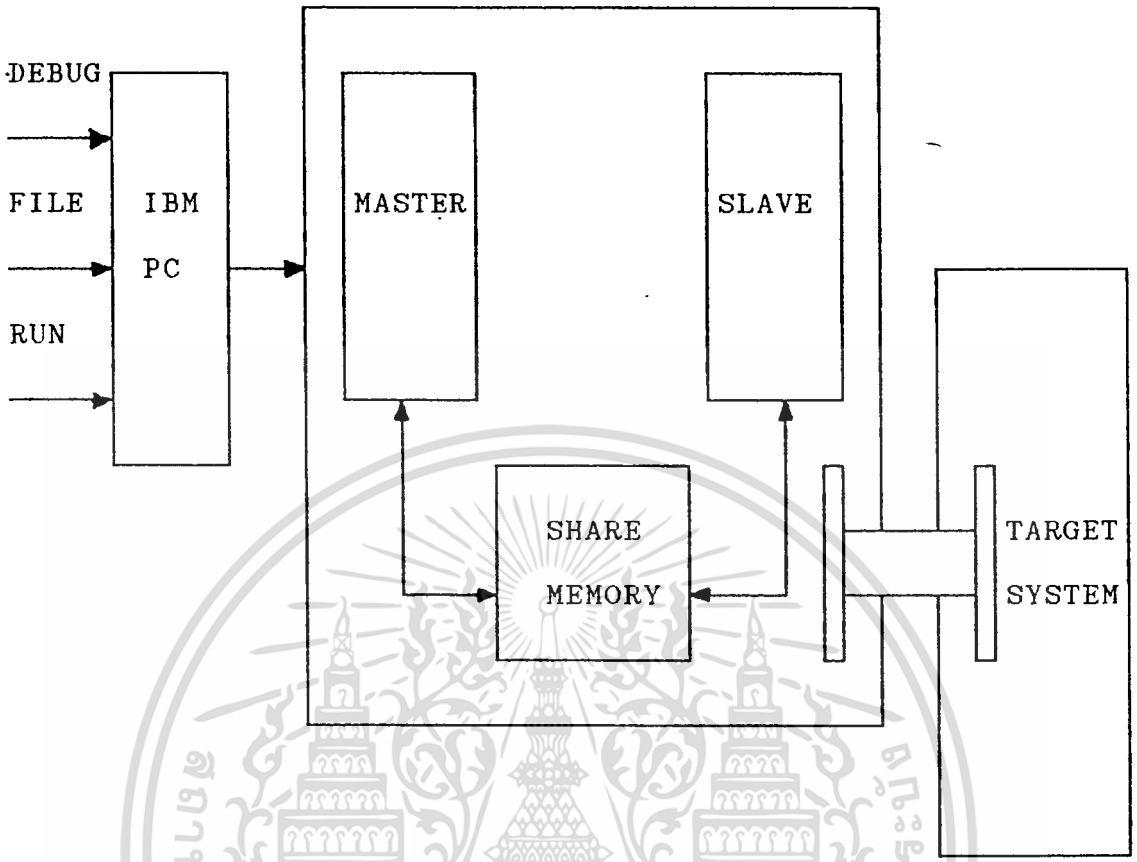
-**อีมูเลเตอร์บอร์ด** ประกอบด้วย ไมโครคอนโทรลเลอร์ 2 ตัว ตัวหนึ่งเรียกว่า MASTER จะทำหน้าที่ควบคุมอีมูเลเตอร์บอร์ดทั้งหมด และติดต่อกับ IBM PC ทาง Serial Port ซึ่ง MCS-51 มี Serial Port ให้ใช้อยู่แล้ว ส่วน MCS-51 อีกตัวหนึ่งอีกว่า Slave ไมโครคอนโทรลเลอร์ตัวนี้จะทำการ อีมูเลเตอร์ ให้ตัวเองสามารถทำงานได้ตามปกติเมื่อไปต่อระบบเป้าหมาย (Target System)

การทำงานของอีมูเลเตอร์บอร์ดนี้จะคอยรับคำสั่งจาก IBM PC มาตีความหมายแล้วทำงานตามคำสั่งที่ได้รับ

-**ระบบเป้าหมาย (Target System)** คือระบบที่ใช้ไมโครคอนโทรลเลอร์ MCS-51 เป็นส่วนประกอบของระบบโดยระบบจะทำงานตามปกติ ถ้าไม่มีส่วนนี้ก็ยังสามารถใช้งานอีมูเลเตอร์บอร์ดได้โดยใช้กับโปรแกรมที่ไม่มีผลกับฮาร์ดแวร์ภายนอก

3.2 บล็อกไดอะแกรมของ MCS-51 อีมูเลเตอร์

ในหัวข้อนี้จะแสดงส่วนประกอบต่าง ๆ ของอีมูเลเตอร์บอร์ดที่เป็นต้นแบบและขอบเขตของระบบที่กำหนดได้ในขั้นต้นนี้แสดงดังรูปที่ 3.2



รูปที่ 3.2 สถาปัตยกรรมของ MCS-51 อีมูเลเตอร์บอร์ด

3.3 คุณลักษณะของ MCS-51 อีมูเลเตอร์บอร์ด

-ใช้ Target System ที่ใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51

-ใช้ร่วมกับ IBM PC โดยติดต่อสื่อสารกันทาง Serial Port RS-232

-หน่วยความจำร่วม(Share Memory)มีขนาด 32 Kbyteโดยใช้ Static Ram

-มีระบบซอฟแวร์บนเครื่อง IBM PC ที่สามารถดีบักโปรแกรมได้

-ผู้ใช้สามารถสั่ง Reset จาก IBM PC ได้

-สามารถแสดง และแก้ไขเปลี่ยนแปลงข้อมูลทั้งใน รีจีสเตอร์ SFR,Internal Ram และ External Ram ได้

-สามารถสั่งให้ MCS-51 ทำการ RUN ตามปกติหรือ Single Step ได้

-สัญญาณนาฬิกาที่ใช้มีความถี่สูง 11.0592 MHz.

-สามารถกำหนดจุดหยุดจาก IBM PC ได้

-ผู้ใช้สามารถต่อระบบวงจรกับบอร์ดดังกล่าวได้

3.4 โครงสร้างทางด้านฮาร์ดแวร์

โครงสร้างของ MCS-51 อีมูเลเตอร์บอร์ดที่จะกล่าวถึงนี้เป็นระบบวงจรที่ทำหน้าที่ประสานการทำงานของไมโครคอนโทรลเลอร์ MCS-51 2 ตัวในบอร์ดนี้เข้าด้วยกันคือ MCS-51 ฝ่ายมาสเตอร์ (Master) และ MCS-51 ฝ่ายสเลฟ (Slave) แต่การทำงานทั้งหมดของบอร์ดจะควบคุมโดยฝ่ายมาสเตอร์ ฝ่ายมาสเตอร์จะรับคำสั่งและสื่อสารข้อมูลจาก IBM PC ผ่าน Serial Port RS-232 อีก โครงสร้างทางด้านฮาร์ดแวร์ของ MCS-51 อีมูเลเตอร์บอร์ดแสดงดังรูป 3.3

-Master ส่วนนี้จะเป็นไมโครคอนโทรลเลอร์ 1 ตัว ควบคุมการทำงานทั้งหมด MCS-51 ของอีมูเลเตอร์บอร์ด โดยรับคำสั่ง (Command) จาก IBM PC ทาง Serial Port RS-232 เช่นคำสั่ง RUN คำสั่ง Single Step หรือคำสั่ง Reset เป็นต้น เมื่อ Master ได้รับคำสั่งจาก IBM PC แล้ว ก็จะทำการควบคุมหรือสั่ง Slave อีกที

-Slave ส่วนนี้จะเป็นไมโครคอนโทรลเลอร์อีกตัวทำหน้าที่อีมูเลท (Emulate) ตัวเองให้ทำงานเหมือนไมโครคอนโทรลเลอร์ของระบบเป้าหมาย (Target System) โดยจะรับคำสั่งจากการอินเตอร์รัพท์ (Interrupt) จากฝ่าย Master ดังนั้นไมโครคอนโทรลเลอร์ฝาย Slave นี้จะ Emulate ตัวเองได้ไม่ครบการทำงานทั้งหมด โดยขาอินเตอร์รัพท์จะถูกใช้งานไปแล้ว

-Multiplex ส่วนนี้จะทำการเลือก Address จากฝาย Master หรือฝาย Slave ให้สามารถติดต่อกับหน่วยความจำร่วม (Share Memory)ได้เพียงฝายเดียวเท่านั้น ซึ่งจะสังเกตุได้ว่าส่วน Multiplex จะมีอินพุทสองทางแต่จะมีเอาท์พุทออกมาแค่ทางเดียว

-Buffer จะมีทั้งของทางด้าน Master และ Slave ซึ่ง Buffer ของทั้งสองฝายนั้นจะต้องทำงานให้สัมพันธ์กัน เพื่อที่จะควบคุมสัญญาณบัสข้อมูลว่า ฝายใดสามารถที่จะทำการติดต่อกับหน่วยความจำร่วม (Share Memory) ได้

-หน่วยความจำร่วม (Share Memory) หรือแรมสองทาง ในการที่จะให้ Master หรือ Slave อ่านเขียนข้อมูลโดยจะใช้ส่วนของ Multiplex ในการสับเปลี่ยนหน่วยความจำว่าจะให้ฝายใดอ้างหน่วยความจำได้และในขณะเดียวกันจะมีส่วนของ Buffer ทั้งสองด้านที่มีการทำงานที่สัมพันธ์กัน ทำหน้าที่ในการที่จะสับเปลี่ยนบัสข้อมูลให้เป็นของฝายใดฝายหนึ่ง และในขณะนั้นจะมีส่วนของตัวถอดรหัส (Decoder) ทำหน้าที่ในการถอดรหัสตำแหน่งของหน่วยความจำร่วมนี้ด้วย

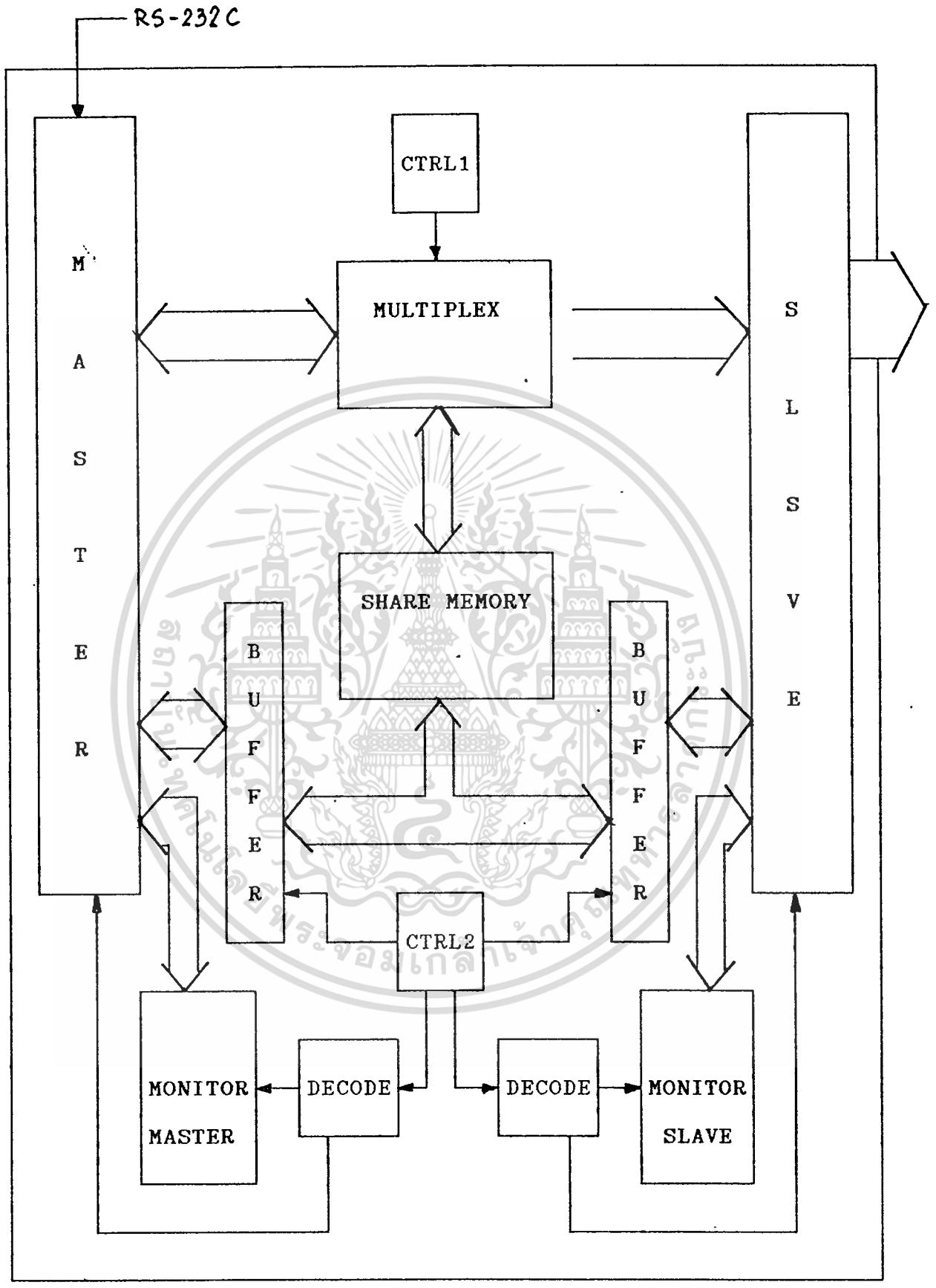
หน่วยความจำร่วมนอกจากจะเป็นที่บรรจุโปรแกรมที่เขียนขึ้น จากทางด้านผู้ใช้แล้วก็จะมีเนื้อที่ส่วนหนึ่งในการส่งผ่านข้อมูลของระบบทั้งสองฝาย

-มอนิเตอร์ (Monitor) ส่วนนี้จะประกอบด้วยรอม(ROM)ซึ่งเก็บโปรแกรมที่ใช้ควบคุมการทำงานมอนิเตอร์ของ Master จะติดต่อสื่อสารกับ IBM PC และควบคุมบอร์ดทั้งหมดส่วนมอนิเตอร์ของ Slave จะ Emulate MCS-51 ของ SLAVE ให้เป็น MCS-51 ของผู้ใช้ คือเป็น CPU ของ Target system

-เครื่องไมโครคอมพิวเตอร์ IBM PC หรือ Compatible ที่มีการ์ดสื่อสารแบบอนุกรม มาตราฐาน RS-232 เพราะว่า MCS-51 อีมูเลเตอร์บอร์ด จะติดต่อสื่อสารกับ IBM PC ทาง Serial Port RS-232 นี้

-ตัวถอดรหัส 1 (Decoder 1) ทำหน้าที่ในการ Decode address ของมอนิเตอร์ของ Master และ Decode address ของหน่วยความจำร่วมด้วย

-ตัวถอดรหัส 2 (Decoder 2) ทำหน้าที่ในการ Decode address ของมอนิเตอร์ของ Slave และ Decode address ของหน่วยความจำร่วมด้วย



รูปที่ 3.3 โครงสร้างทางฮาร์ดแวร์ของ MCS-51 อีมูเลเตอร์บอร์ด

-คอนโทรล 1 (Control 1:Ctrl 1) ส่วนนี้จะใช้ขา P1.0 ของ Master มาควบคุมชุด Multiplex เพื่อบังคับให้ชุด Multiplex ติดต่อกับด้านใดด้านหนึ่งเท่านั้น

-คอนโทรล 2 (Control 2:Ctrl 2) ส่วนนี้จะเป็นส่วนควบคุม Buffer ว่าจะให้ Data ด้านไหน ติดต่อกับหน่วยความจำร่วม

3.4.1 การออกแบบหน่วยความจำ

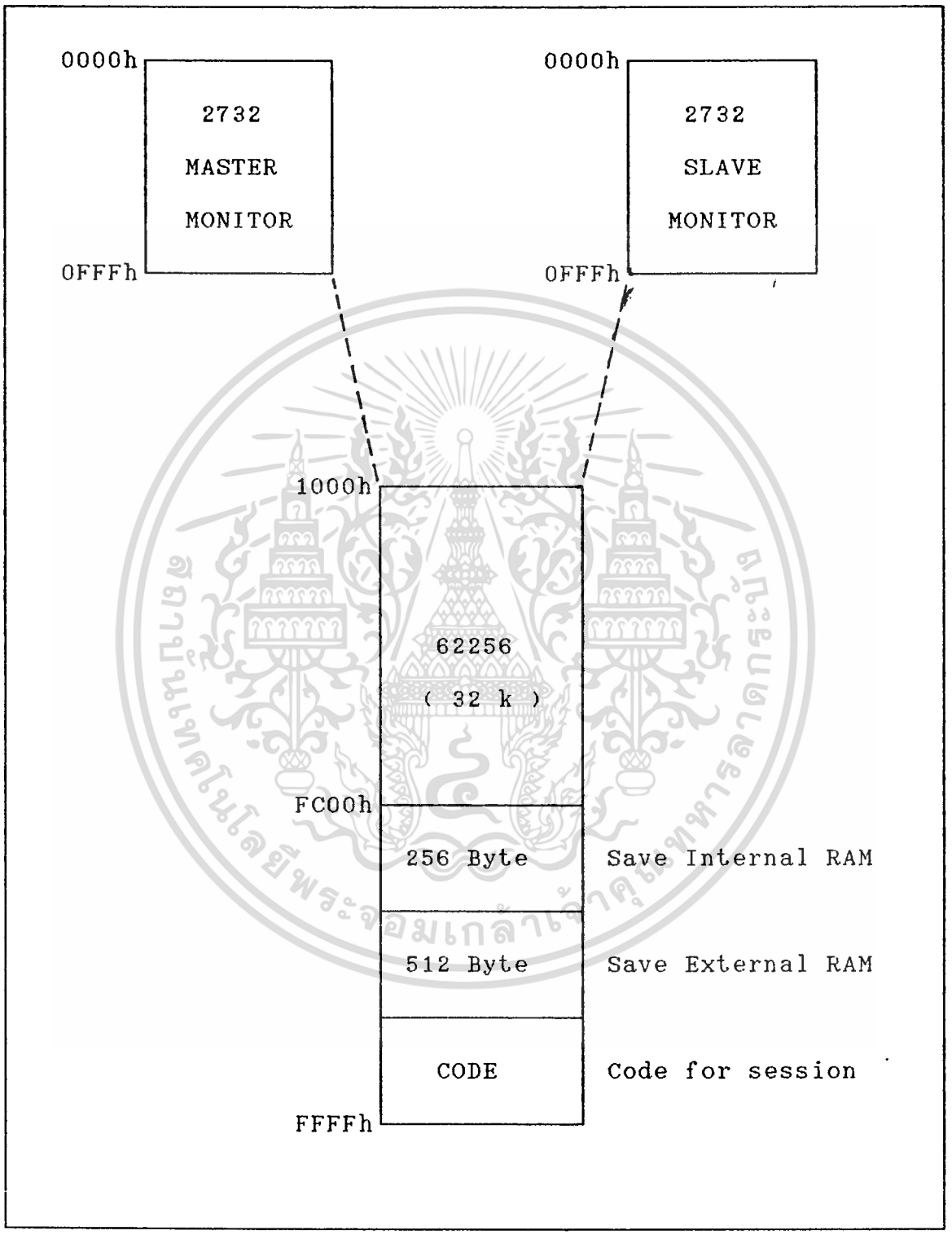
ในการออกแบบหน่วยความจำร่วมเพื่อให้ใช้ได้ทั้งด้าน Master และด้าน Slave นั้น จำเป็นต้องรู้ถึงการจัดหน่วยความจำของ MCS-51 ด้วย MCS-51 นั้นสามารถอ้างหน่วยความจำโปรแกรม (ROM หรือ EPROM) ภายนอกได้ 64 Kbyte และสามารถอ้างหน่วยความจำ (RAM) ได้อีก 64 Kbyte โดยไม่มีการกวนสัญญาณกัน ดังนั้น จึงออกแบบหน่วยความจำตามรูปที่ 3.4 ซึ่งไอซีแต่ละตัวจะเป็นดังนี้

1) 2732 EPROM ขนาด 4 Kbyte ซึ่งเป็นที่เก็บโปรแกรมมอนิเตอร์ของ Master มี address ตั้งแต่ 0000h - 0FFFh

2) 2732 EPROM ขนาด 4 Kbyte ซึ่งเป็นที่เก็บโปรแกรมมอนิเตอร์ของ Slave มี address ตั้งแต่ 0000h - 0FFFh

3) 62256 STATIC RAM ขนาด 32 Kbyte มี address ตั้งแต่ 8000h - FFFFh โดยด้าน Master จะมองหน่วยความจำส่วนนี้เป็น External RAM เพื่อใช้เก็บ Data และโปรแกรมของผู้ใช้ (USER TEST PROGRAM) ส่วนด้าน Slave จะมองหน่วยความจำส่วนนี้เป็นทั้ง External ROM และ External RAM กรณีมองเป็น External ROM เพราะต้องการ RUN โปรแกรมที่ผู้ใช้ต้องการทดสอบ แต่กรณีที่มองเป็น External RAM เพราะ address ล่างของหน่วยความจำส่วนนี้ (Address FC00h - FFFFh) จะใช้เป็น Buffer เพื่อใช้เก็บ Data ต่าง ๆ สำหรับการติดต่อกับ Master

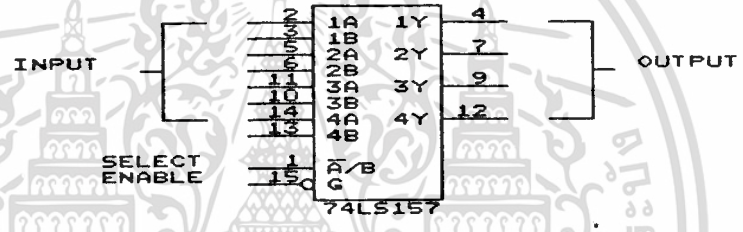
แผนผังการจัดหน่วยความจำ (Memory Map) จะเป็นรูปที่ 3.4



รูปที่ 3.4 แผนผังการจัดหน่วยความจำ (Memory Map)

3.4.2 การออกแบบส่วน Multiplex และ Enable RAM

ในการทำให้ RAM 62256 สามารถติดต่อได้ทั้งกับ Master และ Slave โดยไม่ให้สัญญาณทั้งสองกวนกันนั้น จะใช้ Buffer ในการกันสัญญาณ Data และใช้ไอซี 74LS157 ซึ่งทำงานโดยเลือก 2 ออก 1 (สัญญาณจาก A หรือ B ไปออกที่ Y ดังรูป 3.5) โดยมีสัญญาณเลือกที่ขา $\overline{A}/B$ ถ้าขานี้เป็น "1" ก็จะเลือกสัญญาณจาก B ไปออกที่ OUTPUT Y ในที่นี้ใช้สัญญาณจาก PORT 1 BIT 1 ของ Master เป็นตัวเลือก address จาก Master หรือ Slave



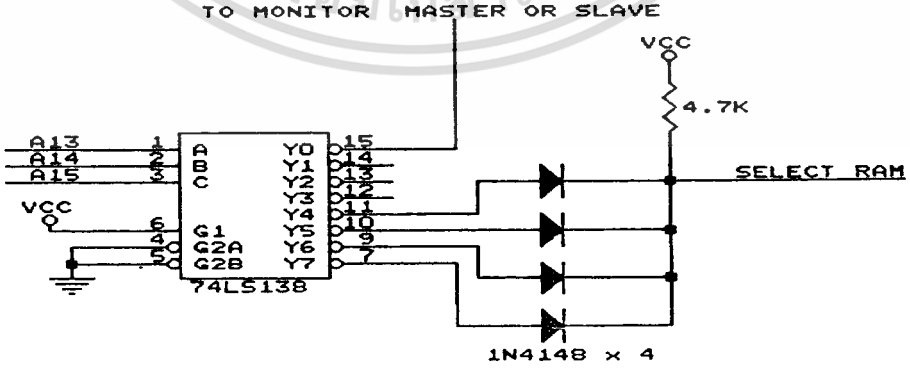
รูปที่ 3.5 แสดงไอซี Multiplex เบอร์ 74LS157

ในการเลือก address จาก Slave address A0 - A15 ได้จากค่าที่แลทช์ไว้แล้ว มาเข้าที่ขา "A" ขาแต่ละชุดของ 74LS157 และในการเลือก address ของ Master ให้มาเข้าที่ขา "B" ของ 74LS157 แต่ละชุดเช่นกัน สำหรับไอซี 74LS157 ตัวสุดท้าย (ตัวที่ 5) ใช้ในการเลือกระหว่างขา WR ของ Master และ WR ของ Slave 1 ชุด และนำเอาท์พุทไปเข้าขา WR ของ RAM 62256 ส่วนสัญญาณ RD ก็ต่อเช่นกัน ทำให้ได้ไม่ว่าจะติดต่อกับ RAM 62256 จาก Master หรือ Slave ก็ตามจะมี address ป้อนให้กับ RAM 62256 เสมอ

ส่วนขา Data Bus ที่ออกจาก RAM 62256 จะมี Buffer 2 ตัว กันไม่ให้ Data Bus ของ Master และ Data Bus ของ Slave ปนกัน โดย Buffer 2 ตัวนี้ จะทำงานเพียงตัวเดียวในขณะใดขณะหนึ่งเท่านั้น โดยตัวที่ไม่ทำงานจะมีสถานะเป็น High impedance ไป ข้อมูลจึงจะเข้า Master หรือ Slave โดยตรงเพียงฝ่ายเดียวขึ้นอยู่กับฝ่ายใดขอติดต่อกับ RAM 62256 นี้ การควบคุม Buffer จะควบคุมที่ขา G (ขา 19) ถ้าขา G = "0" ก็จะทำให้ Buffer ทำงาน ถ้าขา G = "1" ก็จะมีสถานะเป็น High impedance

ขา Buffer ของ Master จะ ON (G = "0")ก็ต่อเมื่อมีการติดต่อกับ RAM 62256 เมื่อมีสัญญาณ Chip select จาก Y4,Y5,Y6,Y7 ของ 74LS138 (IC 5) เป็น "0" อันใดอันหนึ่ง และสัญญาณ Select Multiplex เป็น "1" ในทำนองเดียวกัน ถ้ามีการติดต่อระหว่าง Slave กับ RAM 62256 บ้าง ก็โดยทำให้สัญญาณ Chip select จาก Y4,Y5,Y6,Y7 ของ 74LS138 (IC 23) เป็น "0" อันใดอันหนึ่ง (สัญญาณ Chip select จาก Y4,Y5,Y6,Y7 ทั้งหมดนำมา AND กันหมด) และสัญญาณ Select Multiplex จะเป็น "0" (สัญญาณนี้ส่งมาจาก PORT 1 BIT 1 ของ Master)

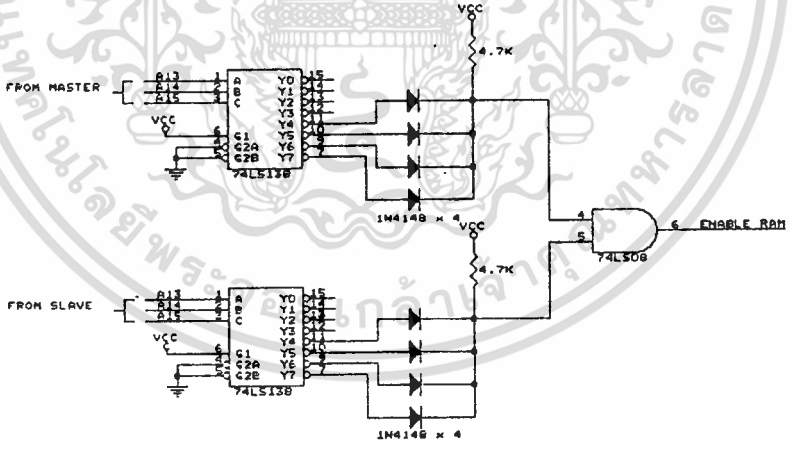
ขา Chip Select ของ RAM 62256 นั้น ได้มาจากการ AND กันของสัญญาณที่เลือกมา Chip Select ของ Master และ Slave ดังรูปที่ 3.6



รูปที่ 3.6 แสดงการ Chip Select RAM 62256

3.4.3 การออกแบบส่วน DECODE ADDRESS ของ MASTER และ SLAVE

การออกแบบส่วน Decode address ของ Master และ Slave จะใช้วงจรเหมือนกัน คือใช้ไอซีเบอร์ 74LS138 (IC 5 และ IC 23) ในการ Decode address จะ Decode เป็น Block Block ละ 8 Kbyte โดยเป็น ROM 8 Kbyte ที่ address 0000h - 1FFFh จะทำให้ Y0 เป็น "0" เพื่อนำไปเป็นสัญญาณ Chip Select EPROM ซึ่งเป็น EPROM ที่บรรจุโปรแกรมมอนิเตอร์ของ Master และ Slave และ address 8000h - FFFFh ซึ่งเป็น address ของ RAM จะให้ Y4,Y5,Y6,Y7 เป็น "0" และนำ Y4 - Y7 นี้มา AND กันทั้งหมด และนำสัญญาณที่ AND กันนี้ ไป Chip Select RAM ดังรูปที่ 3.7



รูปที่ 3.7 แสดงส่วน Decode address ของ Master และ Slave

3.5 โปรแกรมมอนิเตอร์

โปรแกรมมอนิเตอร์เป็นโปรแกรมที่ควบคุมการทำงานของ Master และ Slave โดยโปรแกรมมอนิเตอร์ของ Master จะควบคุมอีมูเลเตอร์บอร์ดทั้งหมดส่วนโปรแกรมมอนิเตอร์ของ Slave จะใช้สำหรับทำการอีมูเลท คือควบคุมการใช้ทำงานของโปรแกรมที่ผู้ใช้ทดสอบ (User Test Program) เช่นการ Save และ Load Register,SFRs เป็นต้น

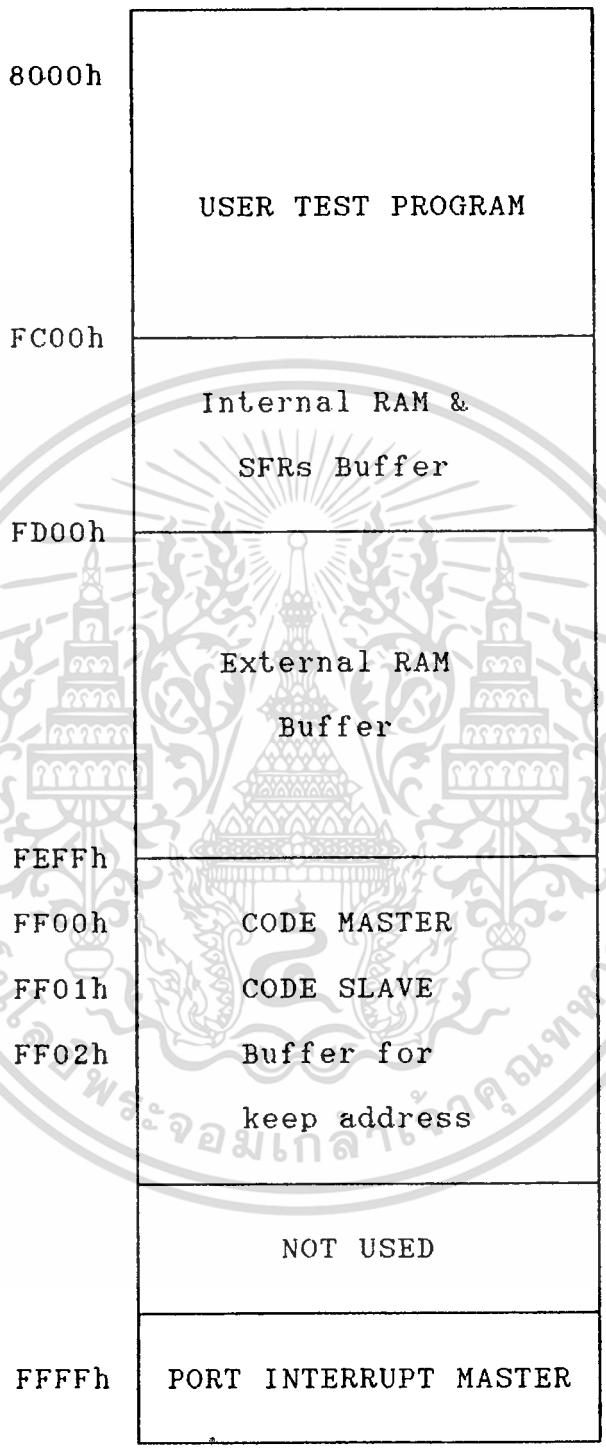
-การทำงานของโปรแกรมมอนิเตอร์ของ Master

เมื่อเริ่มต้นโปรแกรมมอนิเตอรจะทำการเริ่มต้นเซตค่าต่างๆเช่น SERIAL PORT INTERRUPT, CLEAR RAM เป็นต้น จากนั้นจะรอรับคำสั่งจาก IBM PC เพื่อนำมาตีความและทำตามคำสั่งต่อไป

-การทำงานของโปรแกรมมอนิเตอร์ของ Slave

การทำงานของโปรแกรมมอนิเตอร์ของ Slave ทำงานคล้ายกับโปรแกรมมอนิเตอร์ของ Master แต่ระบบมีขนาดเล็กกว่าและมีการเซต interrupt(INT0) เพื่อใช้ในการติดต่อกับ Master โดยโปรแกรมมอนิเตอร์ของ Slave จะรอคอยการ interrupt จาก Master แล้วจึงไปอ่าน Code เพื่อนำไปเลือกการทำงานตามคำสั่งต่อไป

ในการติดต่อระหว่าง Master กับ Slave นั้น จะกำหนดพื้นที่ RAM ส่วนหนึ่งในการใช้งานโต้ตอบกัน โดยจะมีการกำหนดตำแหน่งที่แน่นอน ได้แก่ตำแหน่งของ Code,Buffer ของ Internal RAM และ SFRs และ Buffer ของ External RAM โดยได้จัดแบ่งไว้ดังรูปที่ 3.8



รูปที่ 3.8 แสดงการแบ่งส่วนใช้งาน RAM ในการส่งข้อมูล

3.5.1 อัลกอริทึมของโปรแกรมมอนิเตอร์

-การกำหนดจุดหยุด (Break point)

การกำหนดจุดหยุดใช้วิธีการฝังคำสั่ง LCALL ไว้ที่ address ที่ตั้งเป็นจุด Breakpoint เมื่อ RUN โปรแกรมทดสอบของผู้ใช้มาถึงจุด Breakpoint จะพบคำสั่ง LCALL และจะกระโดดไปทำงานในโปรแกรมมอนิเตอร์เพื่อ SAVE ค่าข้อมูลต่าง ๆ แล้วส่งสัญญาณมา interrupt Master เพื่อให้ Master ส่งค่านั้นให้แก่ IBM PC ต่อไป

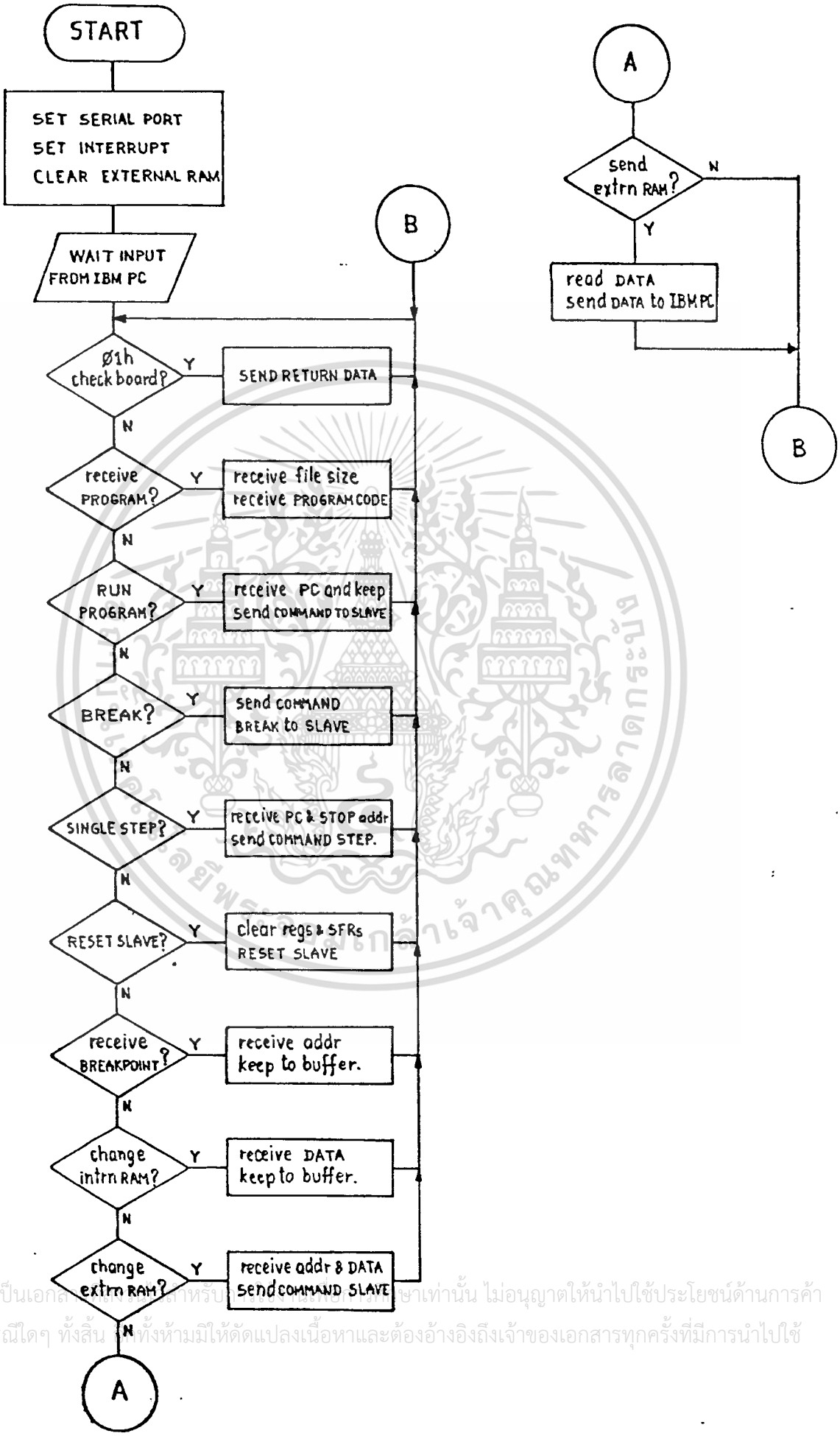
-การทำงานรวดเดียว (RUN)

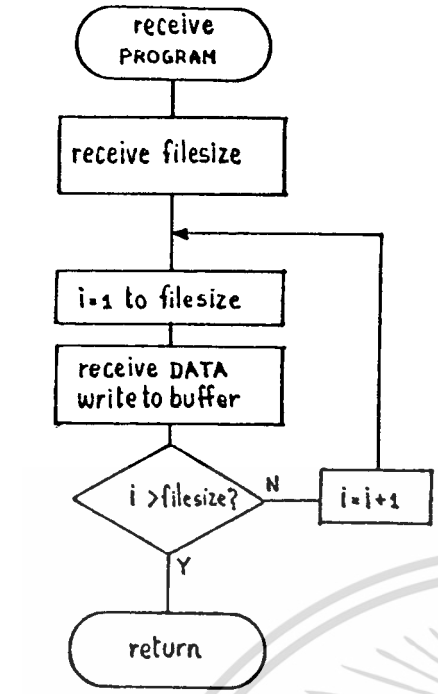
โปรแกรมมอนิเตอร์ของ SLAVE จะไปอ่าน address ที่เริ่ม run จาก RAM Buffer ที่ตำแหน่ง FF02h (PCH) และ FF03h (PCL) จากนั้น PUSH ลง STACK แล้ว RETI กลับมาซึ่งจะทำให้การ run เริ่มต้นที่ address ที่ต้องการ

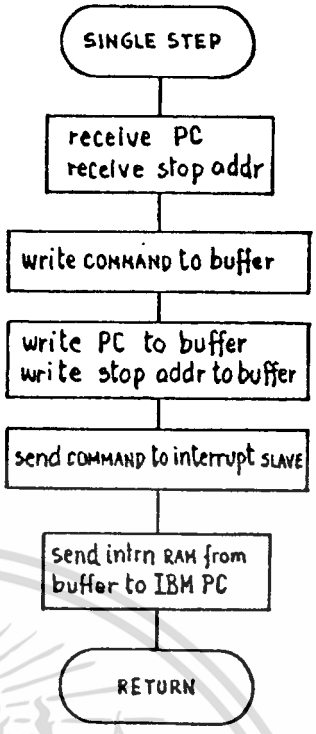
-การทำงานทีละคำสั่ง (SINGLE STEP)

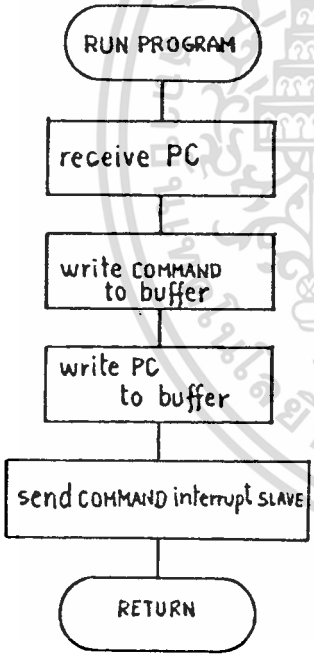
หลักการทำงานของการ run ทีละคำสั่งจะใช้วิธีเดียวกับการกำหนดจุดหยุด โดยนำเอาคำสั่ง LCALL ไปฝังไว้คำสั่งถัดจาก PC เมื่อ run ไป 1 คำสั่งพบคำสั่ง LCALL จะกระโดดไปทำงานในโปรแกรมมอนิเตอร์เพื่อจะทำการ SAVE ค่าข้อมูลต่าง ๆ และส่งสัญญาณมา interrupt Master เพื่อให้ Master ส่งค่านั้นให้แก่ IBM PC ต่อไป

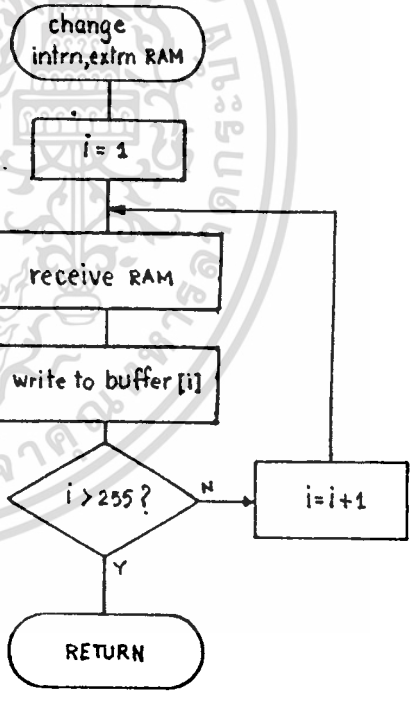
FLOWCHART PROGRAM MASTER (MONITOR)

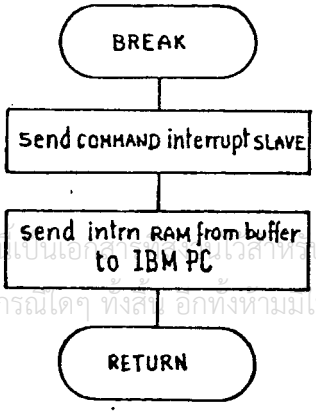




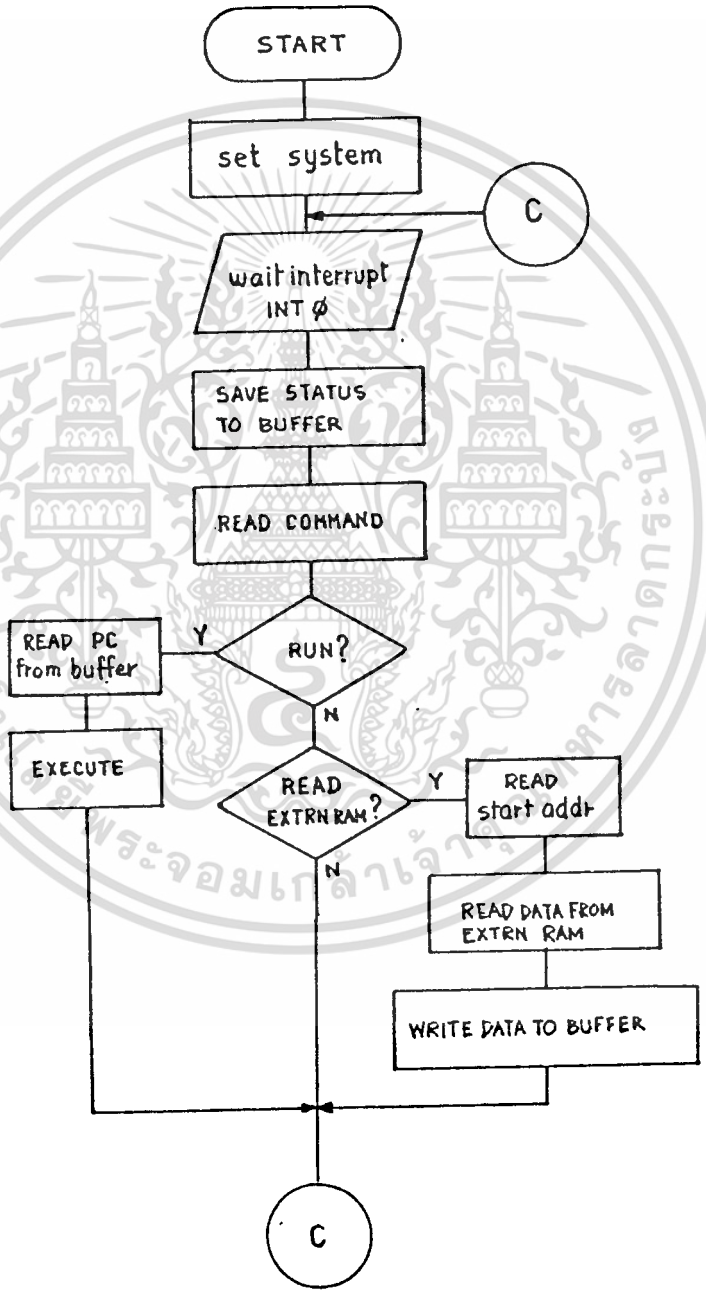








FLOWCHART PROGRAM SLAVE (MONITOR)



บทที่ 4

การใช้งานซอฟท์แวร์บนเครื่อง IBM PC

(Using IBM PC Software)

4.1 การใช้งาน EM51.EXE

EM51.EXE เป็นโปรแกรมสำหรับ run บนเครื่อง IBM PC เพื่อใช้ในการติดต่อควบคุม Board EM51 ผ่านทาง SERIAL PORT การทำงานของโปรแกรมจะเป็นลักษณะเป็น FULL SCREEN และมี menu เป็นแบบ popup menu สะดวกต่อการใช้งาน และมีข้อความซึ่งเป็นคำเตือน และบอกถึงสภาวะต่างๆขณะนั้น ทางด้านล่างของจอภาพ การเรียกใช้งานโปรแกรมสามารถเรียกจาก prompt ของ DOS โดยพิมพ์

A>EM51 <ENTER>

หรือ A>EM51 [file name] <ENTER>

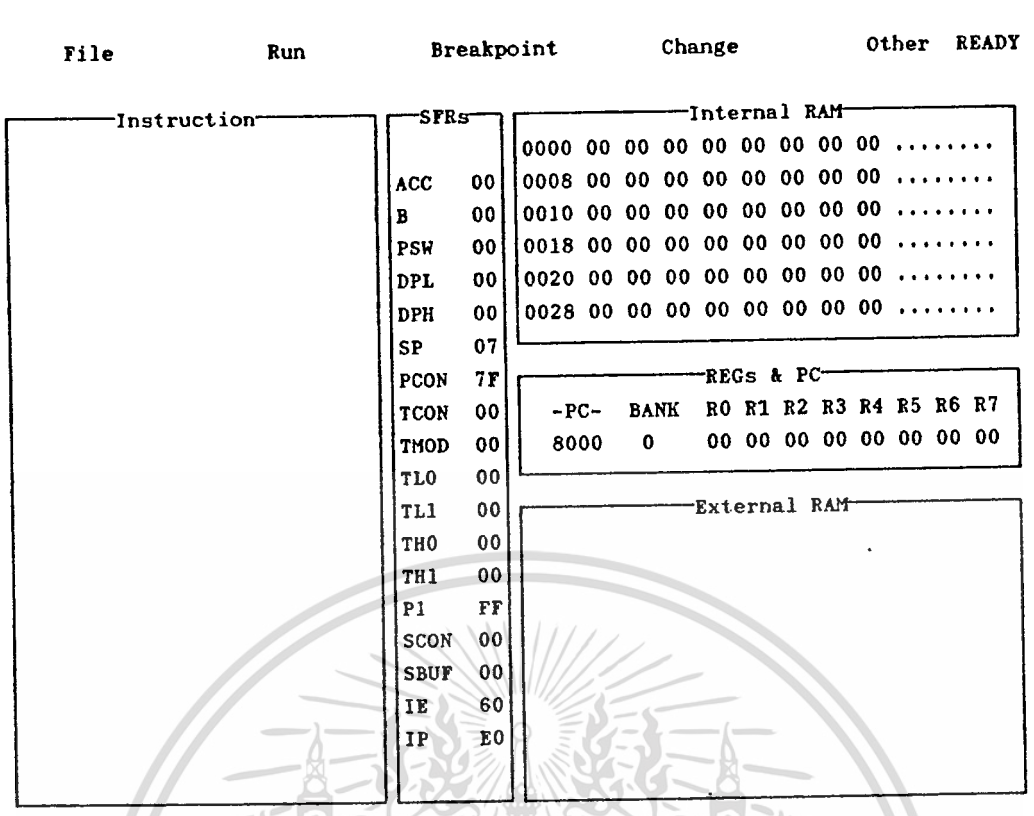
โดย [file name] เป็นชื่อ file แบบ binary (มีนามสกุล .BIN เมื่อแปลด้วย CROSS16 หรือ assembler อื่นๆ)

หลังจากป้อนข้อความข้างบนแล้วหากว่า Board EM51 ไม่ได้ต่ออยู่กับ IBM PC จะปรากฏข้อความดังนี้

"Hardware emulator [EM51] not ready!" และออกจากโปรแกรมสู่ระบบปฏิบัติการ DOS ทันที

หลังจากป้อนข้อความข้างบนแล้วหาก Board EM51 ต่ออยู่กับ IBM PC ถูกต้องเรียบร้อย จะปรากฏข้อความดังนี้

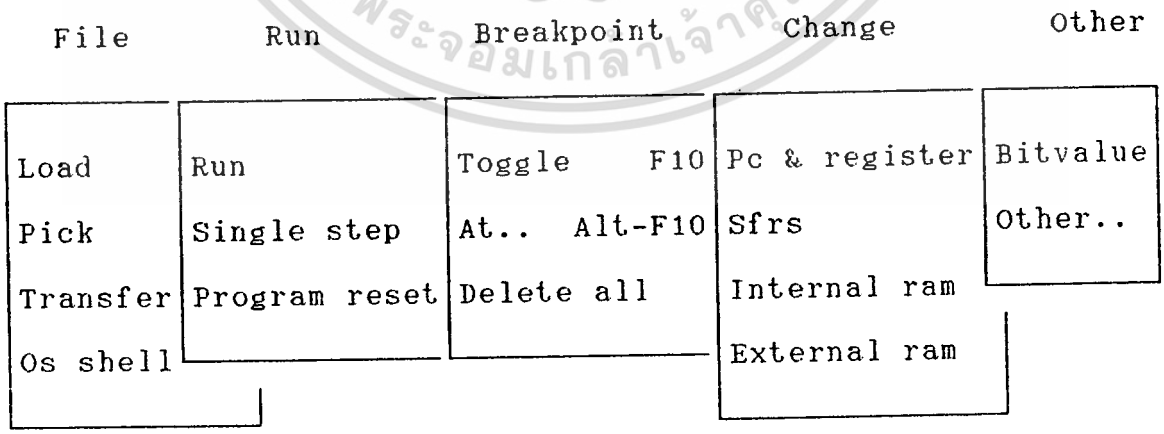
"MCS-51 emulator board VIRSION 1.0" แล้วหลังจากนั้นสักครู่จะเข้าสู่โปรแกรมโดยปรากฏหน้าจอดังรูปที่ 4.1



รูปที่ 4.1 แสดงจอภาพเมื่อโปรแกรมเริ่มทำงาน

MENU ของ EM51

MENU EM51 ทั้งหมดแสดงได้ดังนี้



รูปที่ 4.2 แสดง menu ของโปรแกรม EM51.EXE

การใช้งานคำสั่งต่างๆในแต่ละช่อง window สามารถทำได้หลายทางได้แก่

- กด Function key ได้ หากคำสั่งนั้นมี Function key ให้ใช้งาน
- กด key ตัวอักษรตัวใหญ่ชื่อของ menu นั้น
- เลื่อนแถบ menu ด้วย key ลูกศรไปยังคำสั่งที่ต้องการแล้วกด<ENTER>
- หากอยู่ใน window โปรแกรม (เช่น Instruction, SFRs)เราสามารถไปยัง menu ที่ต้องการโดยกด Alt + key ตัวอักษรตัวใหญ่ใน menu แล้วตามด้วย key ตัวอักษรตัวใหญ่ใน window

4.2 คำสั่งต่าง ๆ และหน้าที่ของคำสั่ง

* File

- Load ใช้ Load file สำหรับ Board EM51 และ Transfer file ลง Board EM51
- Pick ใช้ Load file เดิมที่เคย Load ขึ้นมาล่าสุดและ Transfer file ลง Board EM51 อีกครั้ง
- Transfer ใช้ Transfer file ซึ่งอยู่ใน buffer ของโปรแกรม EM51 อยู่แล้ว ลง Board EM51 อีกครั้ง
- Os shell ออกสู่ระบบปฏิบัติการ DOS ชั่วคราว
- Quit เลิกการทำงาน และกลับสู่ DOS

* Run

- Run ใช้เพื่อ run โปรแกรมบน Board EM51
- Single step ใช้เพื่อ run โปรแกรมทีละ 1 คำสั่ง
- Program reset reset โปรแกรมบน Board EM51

* Breakpoint

- Toggle ใช้เข้าสู่ instruction window เพื่อตั้ง Breakpoint

- At.. ใช้ตั้ง Breakpoint โดยการป้อน address ที่ต้องการ
- Delete all ใช้ยกเลิก Breakpoint ทั้งหมด

* Change

- Pc & register ใช้เข้าสู่ Pc & register window
- Sfrs ใช้เข้าสู่ Sfrs window
- Internal ram ใช้เข้าสู่ Internal ram window
- External ram ใช้เข้าสู่ External ram window

* Other

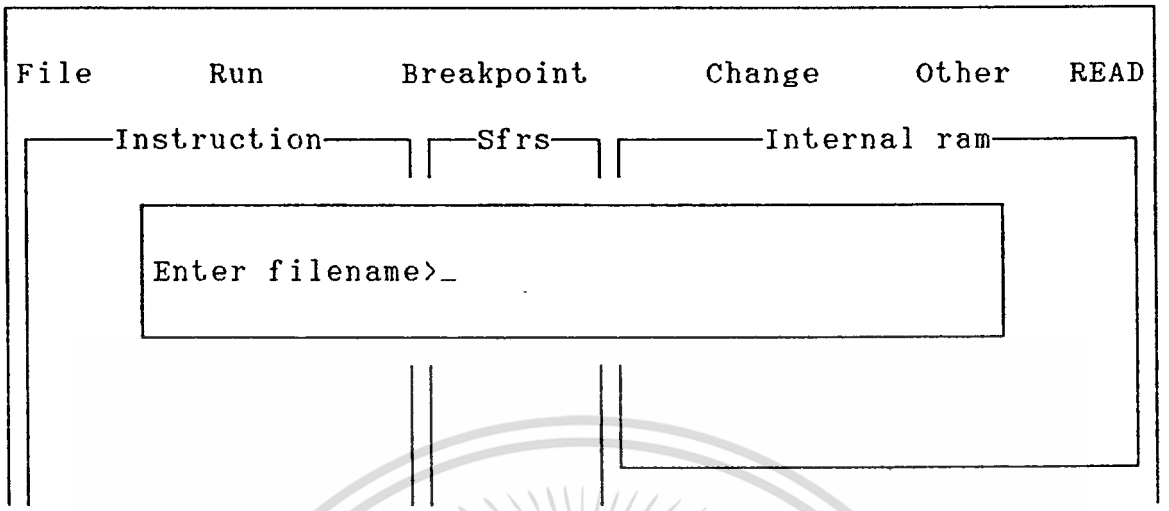
- Bits value ใช้ดูค่า binary สำหรับ Sfr ที่สามารถใช้งานในแบบ Bit
- Other.. ใช้ดูสถานะต่างๆ

4.2.1 รายละเอียดของคำสั่งต่างๆ

Load

คำสั่ง Load สามารถใช้ Function key F3 ได้ด้วย เมื่อใช้งานคำสั่ง Load จะปรากฏช่อง window ดังรูปที่ 4.3

ป้อนชื่อ file ที่ต้องการ Load โดย file นี้ต้องเป็น file นามสกุล <ENTER> file จะถูกอ่านขึ้นมาจากแผ่น disk เก็บไว้ใน buffer แล้วทำการ transfer file นั้นลง Board EM51 ทันที



รูปที่ 4.3 แสดง window สำหรับป้อนชื่อ file

Pick

คำสั่งนี้คล้ายคำสั่ง Load หากมี file ซึ่งเคยเปิดแล้วล่าสุด ก็จะทำการเปิด file โดยใช้ชื่อเดิม คำสั่งนี้จะช่วยประหยัดเวลาในการป้อนชื่อ file ประโยชน์ของคำสั่งนี้เช่น เมื่อเราทดสอบโปรแกรมแล้วพบข้อผิดพลาด และต้องการแก้ไขโดยการใช้คำสั่ง Os shell ออกไปเรียกใช้โปรแกรมอื่นๆ เช่น editor,assembler เมื่อกลับมาสู่โปรแกรมอีกครั้ง ก็ใช้คำสั่ง Pick เพื่อ load file ชื่อเดิมขึ้นมาอีกครั้ง หากใช้คำสั่งนี้โดยยังไม่เคยเปิด file เลย จะไม่สามารถทำงานได้

Transfer

คำสั่ง Transfer เป็นการนำเอา file ซึ่งอยู่ใน buffer transfer ลง Board EM51 อีกครั้ง คำสั่งนี้เราจะใช้เมื่อ Board EM51 เกิดความผิดพลาดขึ้น เมื่อเราได้ reset hardware โดยกดปุ่ม reset บน Board แล้ว ก็ทำการ transfer file ลง Board ใหม่ โดยไม่จำเป็นต้อง load จากแผ่น disk อีก

Os shell

Os shell ใช้ออกสู่ระบบปฏิบัติการ DOS ชั่วคราว เพื่อใช้งานโปรแกรมอื่นหรือค่าสั่งของ DOS ตัวอย่างของการใช้งานค่าสั่งนี้เช่นเมื่อได้ทำการทดสอบโปรแกรมแล้วพบปัญหา ต้องการแก้โปรแกรม ก็ทำโดยออกสู่ระบบปฏิบัติการชั่วคราว แล้วใช้ editor เพื่อแก้ไข source file แล้วแปลใหม่ด้วย assembler แล้วพิมพ์ EXIT ที่ DOS prompt เพื่อกลับมาโปรแกรมอีกครั้ง นอกจากนี้ก็อาจใช้เพื่อออกไปใช้ค่าสั่งต่างของ DOS เช่น DIR, CD หรือเปลี่ยน current drive เป็นต้น

Quit

ใช้เลิกการทำงาน และกลับสู่ DOS

Run

ใช้ run โปรแกรมบน Board EM51 ซึ่งเราได้ transfer ลงไป ค่าสั่งนี้จะทำให้โปรแกรมบน board เริ่มทำงานที่ address ที่ PC ชี้อยู่ ดังนั้น เราจะสามารถให้โปรแกรมเริ่ม run จากจุดใดๆในโปรแกรมได้ โดยการตั้งค่า PC เป็นตำแหน่งที่ต้องการเริ่มต้น run ก่อนการ run โปรแกรมควรจะตั้ง Breakpoint ก่อน หากโปรแกรมไม่มีจุดสิ้นสุดการทำงาน เมื่อเราใช้คำสั่ง run โปรแกรมจะรอคอยค่าซึ่งจะ return กลับมาแสดงผลที่จอภาพ หากนานจนผิดสังเกต หรือลืมตั้ง Breakpoint เราอาจหยุดการรอคอยโดยการกด <ESC> ซึ่งวิธีนี้ โปรแกรมจะส่งค่าสั่งไปหยุดการทำงานของ Board แล้วจะ return ค่าสถานะ ณ จุดที่หยุดนั้นมาแสดงผล

ข้อควรระวัง อย่ากด <ESC> หากโปรแกรมยังทำงานตามปรกติอยู่ เพราะหากเราไปหยุดการทำงานขณะที่โปรแกรม run เสร็จแล้ว และกำลังส่งค่า return กลับมาให้ อาจทำให้ Board ทำงานผิดพลาดได้ แต่กรณีนี้โอกาสเกิดขึ้นก็น้อยมาก

Single step

ใช้ run โปรแกรมทีละคำสั่ง (single step) โดยมีการ return

สถานะกลับมาแสดงผลทุก ๆ การ run 1 คำสั่ง การใช้งานก็เช่นเดียวกันกับคำสั่ง Run และมีข้อจำกัดในการทำคำสั่งนี้อยู่บ้าง ซึ่งจะกล่าวในบทอื่นต่อไป

Program reset

ใช้ reset โปรแกรมบน Board EM51 ซึ่งผลของคำสั่งนี้จะทำให้ระบบเริ่มต้นทำงานใหม่ เปรียบกับ IBM PC ก็เหมือนกับ worm boot

Toggle

เป็นทางเข้าสู่ instruction window สำหรับการเลือกตำแหน่ง break-point ที่บรรทัดของ program code ที่ต้องการ โดยการเลื่อนแถบสว่างไปยังบรรทัดที่ต้องการด้วย arrow key หรือ PgUp/PgDn key เมื่อต้องการเลือกบรรทัดใดเป็นตำแหน่ง breakpoint ก็ให้กด <ENTER> ที่บรรทัดนั้น ซึ่งบรรทัดนั้นจะเปลี่ยนเป็นสีแดงและตัวอักษรเป็นตัวเข้ม เราสามารถที่จะตั้ง breakpoint ได้ 4 จุด โดย breakpoint นั้น จะไปปรากฏที่ window ของ At.. ด้วย

ช่อง instruction window นี้จะใช้สำหรับแสดง program code โดยมีลักษณะดังรูปที่ 4.4

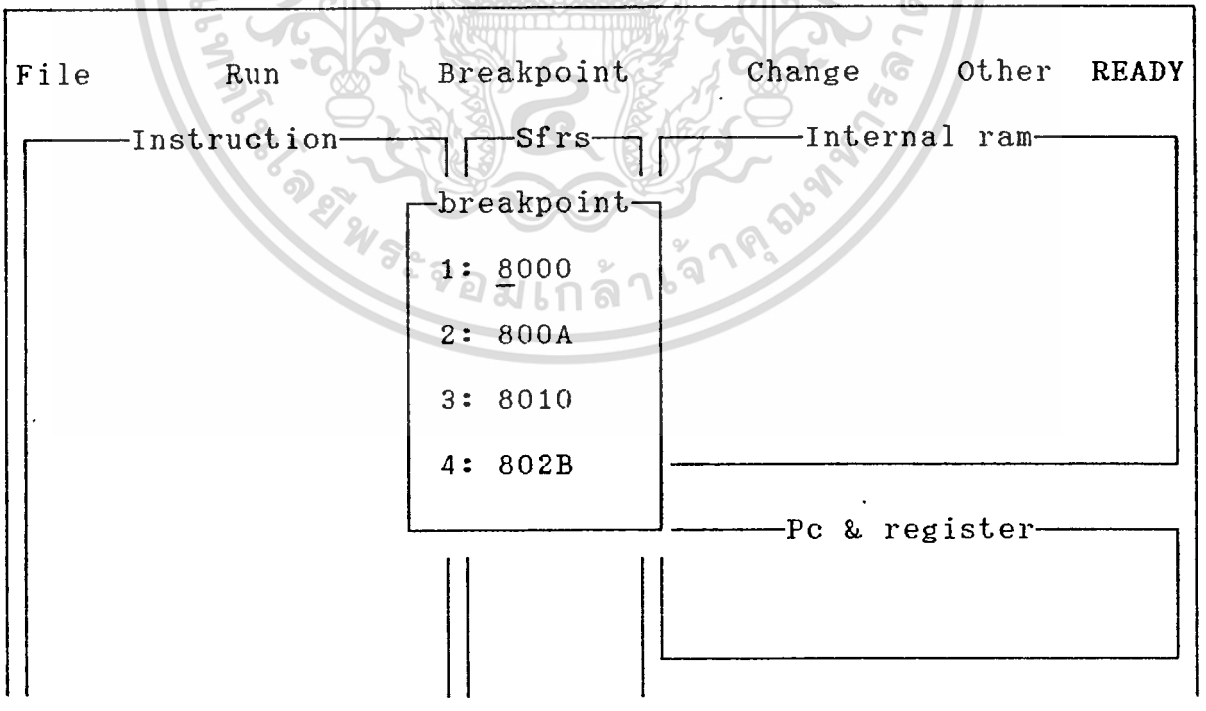
ตำแหน่งที่ PC อยู่ จะแสดงด้วยเครื่องหมาย "▶" ซึ่งอยู่ระหว่าง address และ nemonic เมื่อการ run หรือ single step เสร็จสิ้น ตำแหน่ง PC จะพบอยู่ภายในช่อง window เสมอ การเข้าสู่ instruction window เข้าได้จาก Function key และจาก menu โดยเลือก Toggle และออกโดยกด <ESC>

At..

คำสั่งนี้ใช้ในการตั้ง breakpoint แบบป้อนตำแหน่ง address เมื่อเลือกใช้คำสั่งนี้บนจอภาพจะปรากฏ window ดังรูปที่ 4.5 เราสามารถจะป้อน break-point ได้ 4 จุด การตั้ง address breakpoint โปรแกรมนั้นจะไม่อนุญาตให้ตั้ง address ที่อยู่กลางคำสั่ง หรือ address ที่ไม่ถูกต้องอื่น ๆ โดยจะทำการเปลี่ยนค่าให้ถูกต้องและใกล้เคียงจุดนั้นที่สุด

```
File          Run          Breakpoint          Change          Other  READY
USE: C:\AROON\TSEGM.BIN
Instruction
8053 128077 LCALL 8077
8056 128077 LCALL 8077
8059 128077 LCALL 8077
805C 128077 LCALL 8077
805F 128077 LCALL 8077
8062 0F     INC   R7
8063 DED8   DJNZ  R6,D8
8065 80C9   SJMP  C9
8067 C0F9   PUSH  F9
8069 A4     MUL   AB
806A B099   ANL   C,99
806C 9282   MOV   82,C
806E F8     MOV   R0,A
806F 8090   SJMP  90
8071 8883   MOV   83,R0
8073 C6     XCH   A,@R0
8074 A186   AJMP  0586
8076 6E78   MOV   78,R6
8078 FF     MOV   R7,A
8079 79FF   MOV   R1,#FF

SFRs
ACC   90
B     00
PSW   00
DPL   00
DPH   08
SP    09
PCON  7F
TCON  00
TMOD  00
TL0   00
TL1   00
TH0   00
TH1   00
P1    00
SCON  00
SBUF  00
IE    E0
IP    E0

Internal RAM
0000 0A 8B 00 00 00 00 07 09 ........
0008 62 80 7B 80 00 08 00 00 b.{.....
0010 00 00 00 00 00 00 00 00 ........
0018 00 00 00 00 00 00 00 00 ........
0020 00 00 00 00 00 00 00 00 ........
0028 00 00 00 00 00 00 00 00 ........

REGs & PC
 -PC-  BANK  R0 R1 R2 R3 R4 R5 R6 R7
 807B    0   0A 8B 00 00 00 00 07 09

External RAM
0000 41 42 43 44 45 46 47 0E ABCDEFG.
0008 0E 0D 0F 0F 0C 0A 03 04 ........
0010 56 3A 51 E6 08 32 43 13 V:Q..2C.
0018 40 07 59 32 19 06 00 32 @.Y2...2
0020 60 65 01 25 39 08 05 20 'e.%9..
0028 00 6E 57 89 84 00 53 91 .nW...S.
0030 65 09 65 40 33 29 00 01 e.e@3)..
0038 E9 48 98 00 16 54 87 66 .H...T.f

F2 transfer F3 load F8 single step F9 run F10 toggle
```

รูปที่ 4.4 แสดงช่อง window ทั้งหมดบนจอภาพ



รูปที่ 4.5 แสดง window ของคำสั่ง At..

Pc & register

ค่าสั่งนี้ใช้เข้าสู่ช่อง Pc & register window เพื่อใช้เปลี่ยนค่าต่างๆในช่องนี้ได้แก่ ค่า PC, BANK REGISTER และค่า register ดูรูปที่ 4.4

Sfrs

ใช้เปลี่ยนค่าใน SFR (Special function register) ดูรูปที่ 4.4

Internal ram

ใช้เปลี่ยนค่าภายใน internal ram ช่วง address 00h - 7Fh โดยสามารถแก้ไข ณ ตำแหน่งใด ๆ บนช่อง window โดยการใช้ arrow key และ PgUp/PgDn key ดูรูปที่ 4.4

External ram

ใช้เปลี่ยนค่าใน external ram ช่วง address 0000h-7FFFh โดยสามารถแก้ไข ณ ตำแหน่งใดๆบนช่อง window โดยการใช้ arrow key และ PgUp/PgDn key ก่อนที่จะสามารถดู external ram ได้ จะต้องป้อนตำแหน่ง address ที่ต้องการดูก่อน โดยมีช่อง window สำหรับป้อน address ดังรูปที่ 4.6

ช่อง window จะแสดง address ซึ่งเป็นค่า default มาในตอนแรก ซึ่งหากต้องการใช้ค่า default ก็ให้กด <ENTER> ได้เลย การอ่านข้อมูลใน external ram จะอ่านทีละ block block ละ 256 bytes และเมื่อสุด block ก็จะอ่านขึ้นมาใหม่โดยอัตโนมัติ

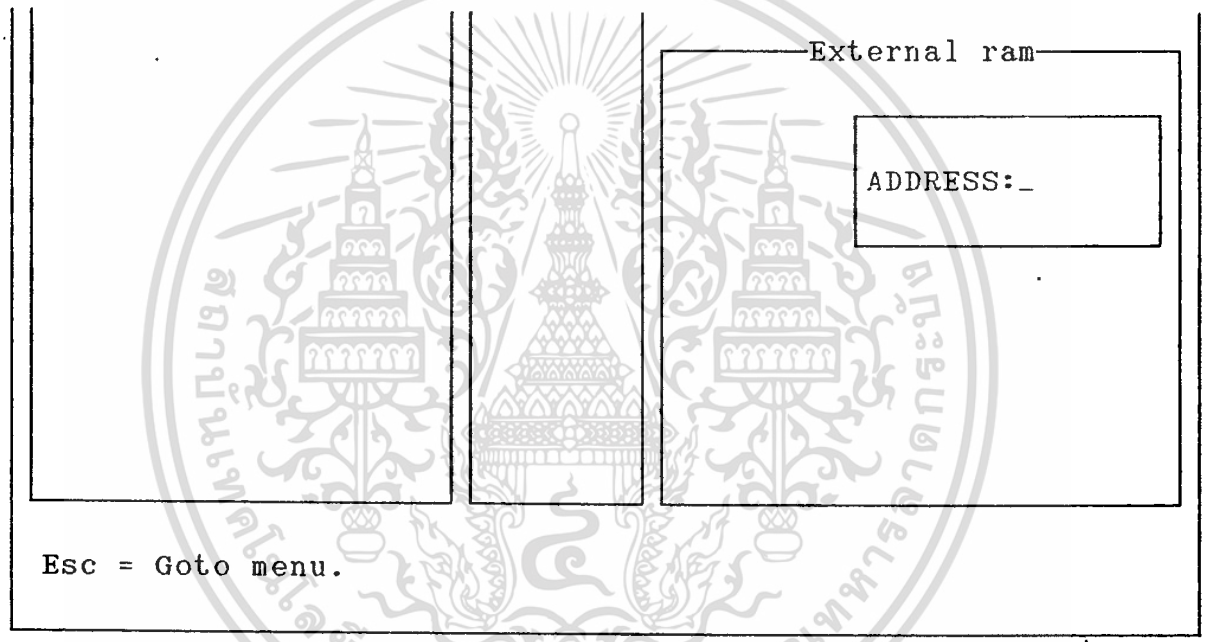
ข้อควรระวัง หาก hardware ไม่มีการต่อ external ram อยู่ ค่าที่ได้แสดงอยู่บนจอภาพ จะไม่ถูกต้อง และ external ram จะต้องต่ออยู่ในช่วงของ address 0000h-7FFFh เท่านั้น

Bits values

ใช้สำหรับดูค่า SFR บางตัวซึ่งสามารถใช้งานในแบบ Bits address ได้

Other..

ใช้แสดงสถานะบางอย่างได้แก่ Baud rate, file size และจำนวนคำสั่งใน file นั้น รูปที่ 4.8 แสดง window ของ Other..



รูปที่ 4.6 แสดง window สำหรับการป้อน address

```
    File            Run          Breakpoint          Change          Other  READY
USE: C:\AROON\TSEGM.BIN
┌──────Instruction──────┐ ┌SFRs┐ ┌──────────────Internal RAM──────────────┐
805C 128077 LCALL 8077  │ │    │ │0000 00 00 00 00 00 0┌──BIT VALUE──┐
805F 128077 LCALL 8077  │ │ACC 00│ │0008 00 00 00 00 00 0│ACC : 00000000│
8062 0F     INC   R7    │ │B   00│ │0010 00 00 00 00 00 0│B   : 00000000│
8063 DED8   DJNZ  R6,D8 │ │PSW 00│ │0018 00 00 00 00 00 0│PSW : 00000000│
8065 80C9   SJMP  C9    │ │DPL 00│ │0020 00 00 00 00 00 0│IP  : 11100000│
8067 C0F9   PUSH  F9    │ │DPH 00│ │0028 00 00 00 00 00 0│IE  : 01100000│
8069 A4     MUL   AB    │ │SP  07│ └─────────────────────┘SCON: 00000000│
806A B099   ANL   C,99  │ │PCON 7F│ ┌──────REGs &  ──────┐TCON: 00000000│
806C 9282   MOV   82,C  │ │TCON 00│ │ -PC-  BANK  R0 R1 R└──────────────┘
806E F8     MOV   R0,A  │ │TMOD 00│ │ 8000   0    00 00 00 00 00 00 00 00 │
806F 8090   SJMP  90    │ │TL0 00│ └─────────────────────────────────────┘
8071 8883   MOV   83,R0 │ │TL1 00│ ┌──────External RAM──────┐
8073 C6     XCH   A,@R0 │ │TH0 00│ │0000 FF 01 02 03 04 05 06 0F ........ │
8074 A186   AJMP  0586  │ │TH1 00│ │0008 FF 01 02 03 04 FF 06 07 ........ │
8076 8E78   MOV   78,R6 │ │P1  FF│ │0010 08 09 0A 0B 0C 0D 0E 0F ........ │
8078 FF     MOV   R7,A  │ │SCON 00│ │0018 08 09 0A 0B 0C 0D 0E 0F ........ │
8079 79FF   MOV   R1,#FF│ │SBUF 00│ │0020 08 09 0A 0B 0C BD 0E 0F ........ │
807B D9FE   DJNZ  R1,FE │ │IE  60│ │0028 08 09 0A 0B 0C BD 0E 0F ........ │
807D D8FA   DJNZ  R0,FA │ │IP  E0│ │0030 08 09 0A 0B 0C 0D 0E 0F ........ │
807F 22     RET         │ │    │ │0038 08 09 0A 0B 0C 0D 0E 0F ........ │
└─────────────────────┘ └────┘ └──────────────────────┘
ESC = Goto menu.
```

รูปที่ 4.7 แสดง Bits value window

```
    File            Run          Breakpoint          Change          Other  READY
USE: C:\AROON\TSEGM.BIN
┌──────Instruction──────┐ ┌SFRs┐ ┌──────────────Internal RAM──────────────┐
805C 128077 LCALL 8077  │ │    │ │0000 00 00 00 00 ┌──OTHER──┐           │
805F 128077 LCALL 8077  │ │ACC 00│ │0008 00 00 00 00 │Baud rate : 9600 │
8062 0F     INC   R7    │ │B   00│ │0010 00 00 00 00 │File size :  128 │
8063 DED8   DJNZ  R6,D8 │ │PSW 00│ │0018 00 00 00 00 │instruction:  83 │
8065 80C9   SJMP  C9    │ │DPL 00│ │0020 00 00 00 00 └─────────┘           │
8067 C0F9   PUSH  F9    │ │DPH 00│ │0028 00 00 00 00 00 00 00 00 ........ │
8069 A4     MUL   AB    │ │SP  07│ └──────────────────────────────────────┘
806A B099   ANL   C,99  │ │PCON 7F│ ┌──────REGs & PC──────┐
806C 9282   MOV   82,C  │ │TCON 00│ │ -PC-  BANK  R0 R1 R2 R3 R4 R5 R6 R7 │
806E F8     MOV   R0,A  │ │TMOD 00│ │ 8000   0    00 00 00 00 00 00 00 00 │
806F 8090   SJMP  90    │ │TL0 00│ └─────────────────────────────────────┘
8071 8883   MOV   83,R0 │ │TL1 00│ ┌──────External RAM──────┐
8073 C6     XCH   A,@R0 │ │TH0 00│ │0000 FF 01 02 03 04 05 06 0F ........ │
8074 A186   AJMP  0586  │ │TH1 00│ │0008 FF 01 02 03 04 FF 06 07 ........ │
8076 8E78   MOV   78,R6 │ │P1  FF│ │0010 08 09 0A 0B 0C 0D 0E 0F ........ │
8078 FF     MOV   R7,A  │ │SCON 00│ │0018 08 09 0A 0B 0C 0D 0E 0F ........ │
8079 79FF   MOV   R1,#FF│ │SBUF 00│ │0020 08 09 0A 0B 0C BD 0E 0F ........ │
807B D9FE   DJNZ  R1,FE │ │IE  60│ │0028 08 09 0A 0B 0C BD 0E 0F ........ │
807D D8FA   DJNZ  R0,FA │ │IP  E0│ │0030 08 09 0A 0B 0C 0D 0E 0F ........ │
807F 22     RET         │ │    │ │0038 08 09 0A 0B 0C 0D 0E 0F ........ │
└─────────────────────┘ └────┘ └──────────────────────┘
ESC = Goto menu.
```

รูปที่ 4.8 แสดง window ของ Other..

บทที่ 5

ระบบซอฟท์แวร์และการสื่อสารข้อมูล

(software system and communication)

ในบทนี้จะกล่าวถึงการทำงานของระบบ software ซึ่งมีเรื่องที่เกี่ยวข้อง ได้แก่การสื่อสารกับ Board EM51 flowchart การทำงานของโปรแกรม

5.1 โครงสร้างข้อมูลในการสื่อสาร

สำหรับการเชื่อมกันระหว่าง IBM PC กับ Board EM51 นั้น ใช้การเชื่อมต่อผ่าน SERIAL PORT RS-232C โดยโปรแกรมบน IBM PC จะทำหน้าที่ควบคุม Board ไว้ทั้งหมด โดยทำหน้าที่ส่งคำสั่งต่าง ๆ ผ่าน SERIAL PORT เพื่อควบคุม Board โดยควบคุมที่ตัว CPU MASTER คำสั่งต่างๆ มีโครงสร้างดังต่อไปนี้

คำสั่ง (HEX)	จำนวน byte	คำสั่ง(DEC)	FUNCTION
01	1		คำสั่งตรวจสอบ Board EM51
02+[sizeH]+[sizeL]+ [program code]	3+file size		ส่งโปรแกรมลง Board EM51
03+PCH+PCL	3		สั่ง run โปรแกรม
04	1		break, ยกเลิกการ run
05+PCH+PCL+stop addrH +stop addrL	5		สั่ง single step
06	1		reset program บน board
07+addrH+addrL	3		set breakpoint

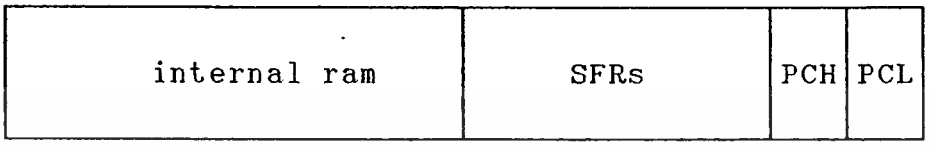
(มีต่อ)

ค่าสั่ง (HEX)	จำนวน byte ค่าสั่ง(DEC)	FUNCTION
08+[256 byte]	257	เปลี่ยนข้อมูล IR และ SFRs
09+addrH+addrL+[256 byte]	259	เปลี่ยนข้อมูล external ram
0A+addrH+addrL	3	อ่านข้อมูล external ram

ค่าสั่งต่างๆเหล่านี้ จะได้รับการตอบสนองจาก Board EM51 ดังนี้

ค่าสั่งจาก IBMPC	การตอบสนองของ Board EM51
01h	Board จะส่งข้อมูลมาแสดงหน้าจอเพื่อแสดงว่า Board EM51 ต่อกับ IBM PC จริง format ของข้อมูลเป็นดังนี้ 81h+data+data+..+0Dh
03h	03h เป็นค่าสั่ง run เมื่อ run ถึงจุด breakpoint หรือสิ้นสุด file จะ return format ดังนี้ 83h+[IR 256 byte]
04h	04h เป็นค่าสั่งbreak จะยังผลให้ Board return format ดังนี้ 84h+[IR 256 byte]
05h	05h เป็นค่าสั่ง single step เมื่อ run ไป 1 ค่าสั่ง จะ return format 85h+[IR 256 byte]
06h	06h เป็นค่าสั่ง reset ซึ่ง board จะ return format 86h+[IR 256 byte]
0Ah	0Ah เป็นค่าสั่ง read ข้อมูลใน external ram ซึ่งจะได้รับค่า return format 8Ah+[external ram 256 byte]

ภายใน 256 byte internal ram จะประกอบด้วยข้อมูล internal ram 128 byte แรก ตามด้วย SFRs นอกจากนี้จะมีค่า PC ซึ่งถูกนำมาเพิ่มไว้ด้วย เพราะ PC ไม่มีอยู่ใน SFRs และ internal ram



รูปที่ 5.1 แสดง format ของ 256 byte internal ram

สำหรับในส่วนของ external ram เนื่องจาก external ram มีจำนวนข้อมูลมาก ไม่สะดวกที่จะถ่ายเทข้อมูลทั้งหมด จึงได้ใช้วิธีการแบ่งเป็น block block ละ 256 byte

5.2 Flowchart และ โปรแกรม EM51.EXE

ก่อนอื่นจะกล่าวถึงการทำงานของระบบ software คือ ส่วนของโปรแกรมที่ใช้งานบนเครื่อง IBM PC โปรแกรม EM51.EXE นั้น เขียนด้วยภาษา C ทั้งหมด และใช้ C compiler ของ Borland International คือ Turbo C version 2.0 การเขียนโปรแกรม ผู้ออกแบบได้พยายามที่จะใช้คำซึ่งให้ความหมายที่ชัดเจนที่สุด เพื่อให้ง่ายต่อการทำความเข้าใจ ซึ่งจะเป็นประโยชน์ในการพัฒนาโปรแกรมตัวนี้ให้มีขีดความสามารถเพิ่มขึ้น

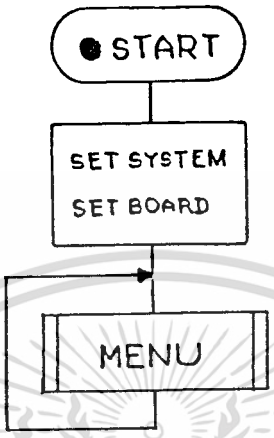
การทำงานของโปรแกรมโดยหลักๆจะเริ่มจากการตรวจสอบ Board EM51 ก่อนเป็นอันดับแรก และจะเริ่มจัดการตรวจสอบระบบให้พร้อมสำหรับการทำงาน แล้วจัดหน้าจอเพื่อแสดงผล โดยการแสดงผลจะแบ่งออกเป็นส่วน ๆ เป็นช่อง window หาก window ใดสามารถแสดงผลได้ ก็จะแสดงออกมาทันที จากนั้นจะเข้าสู่ระบบ menu รอรับคำสั่งจากผู้ใช้ เพื่อทำงานตามคำสั่งต่อไป

ในโปรแกรมเราสามารถแบ่งกลุ่มของโปรแกรมย่อยได้ 4 กลุ่มใหญ่ ได้แก่

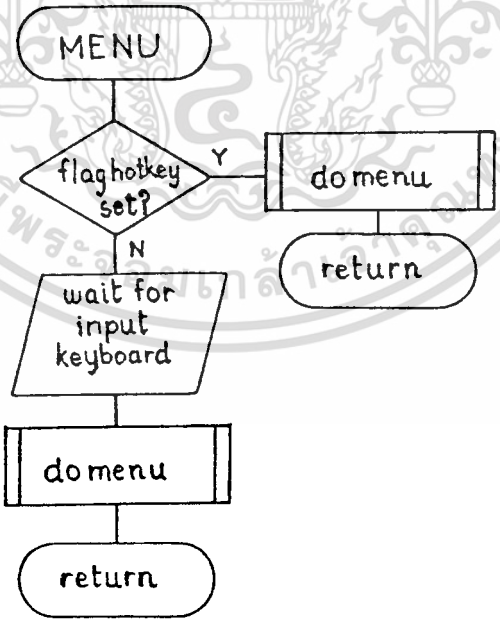
กลุ่มที่จัดการกับระบบ menu กลุ่มที่ติดต่อสื่อสารกับ Board EM51 กลุ่มที่เกี่ยวกับการแสดงผลและกลุ่มสุดท้ายเป็นกลุ่มที่จัดการกับตัวแปรต่าง ๆ ซึ่งการทำงานของแต่ละโปรแกรมย่อยได้อธิบายประกอบไว้ในโปรแกรมเพื่อให้ง่ายต่อการทำความเข้าใจ

ในส่วนของ Flowchart ก็จะเป็นอีกส่วนหนึ่งในการช่วยอธิบายหลักในการทำงานของโปรแกรมนั้น ๆ ซึ่งเท่าที่ได้แสดงเป็น Flowchart ออกมานั้นเป็นเฉพาะส่วนของโปรแกรมหลัก ๆ แม้จะไม่สามารถอธิบายความละเอียดอ่อนในส่วนของโปรแกรมได้ แต่ก็ช่วยให้ทำความเข้าใจโปรแกรมได้ง่ายขึ้น การดูโปรแกรมขอแนะนำให้ดูส่วนหลักจากแผนภาพ Flowchart และส่วนที่เป็นรายละเอียด ในโปรแกรม ซึ่งทั้ง 2 ส่วน ควรดูควบคู่กันไป และที่สำคัญต้องทราบความหมายของตัวแปรในโปรแกรมและความสำคัญของตัวแปรนั้นด้วย จึงจะเข้าใจได้ดีเช่นตัวแปร chgram = change ram, chkram = check ram dummy = ตัวแปรพักข้อมูล i,j,k เป็นตัวแปร loop เป็นต้น สิ่งเหล่านี้นับว่ามีความสำคัญมาก เพราะจะทำให้เข้าใจโปรแกรมได้ง่ายขึ้น โปรแกรมสามารถดูจากแผ่น disk ที่ได้จัดเตรียมไว้แล้ว

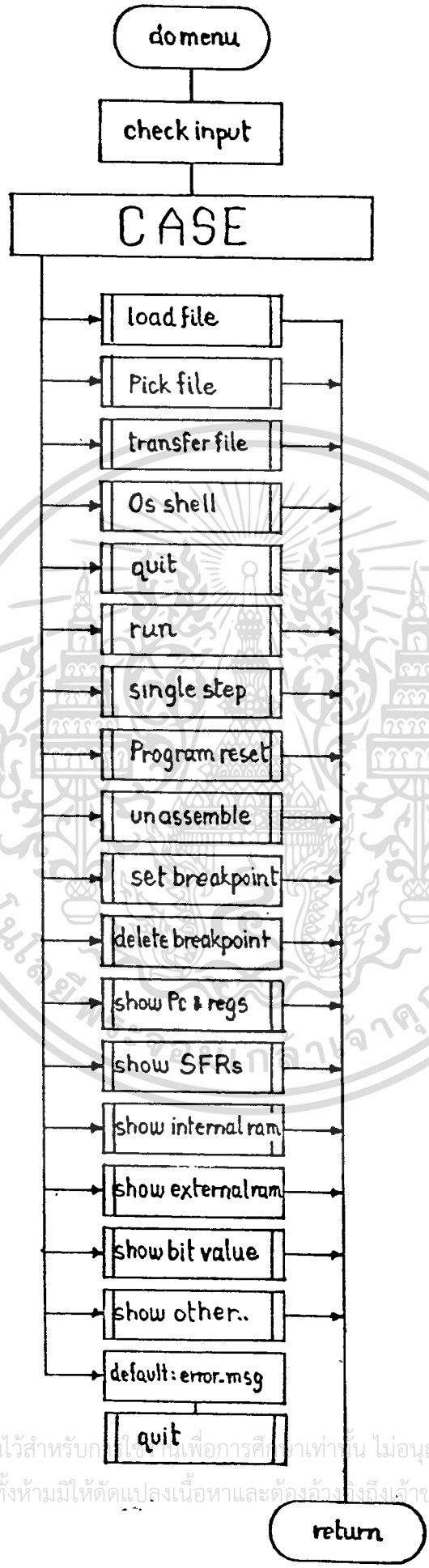
FLOW CHART START แสดงการทำงานในส่วนหลัก (main) โปรแกรม EM51 จะเริ่มต้นที่จุดนี้



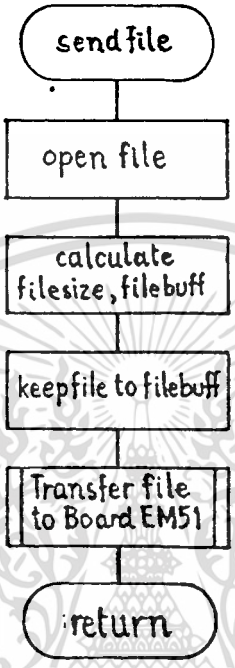
FLOW CHART MENU เป็นส่วนหลักอีกส่วนหนึ่งในการทำงานของโปรแกรม เริ่มจากการตรวจสอบ FLAG HOTKEY หาก set จะไปทำ subroutine DUMENU หากไม่ได้ set จะเข้าสู่ส่วนการรับ key ก่อนจะไป subroutine DUMENU



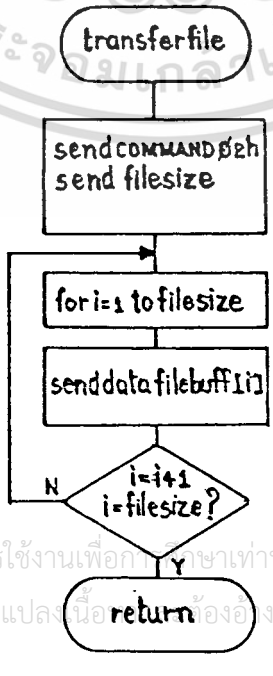
FLOW CHART DUMENU แสดงส่วนที่ใช้ในการเลือกการทำงานไปยัง sub-routine ต่าง ๆ



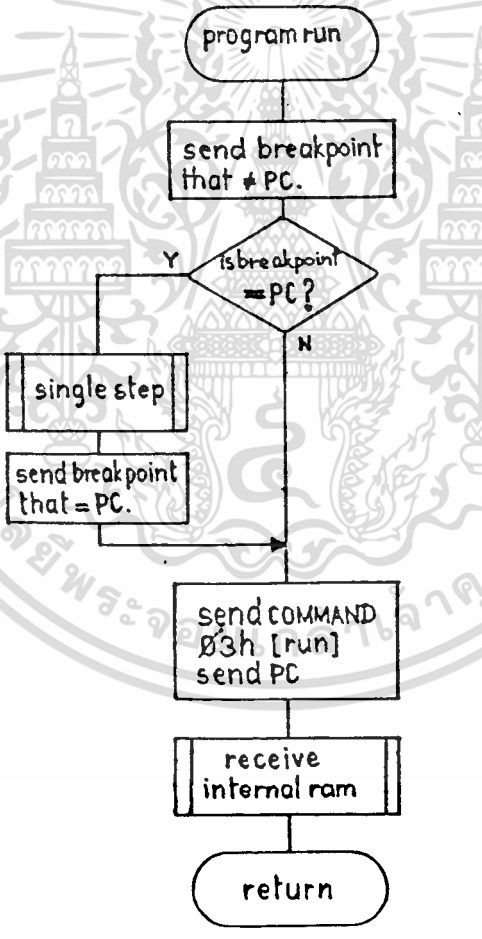
FLOW CHART SEND FILE แสดงการทำงานของ subroutine ที่ใช้ในการส่งโปรแกรมทดสอบไปยัง board โดยเริ่มจากการเปิด file นำมาคำนวณขนาด แล้วเก็บข้อมูลลง buffer แล้ว transfer file ลง board



FLOW CHART TRANSFER FILE แสดงการส่ง file โดยเริ่มจากการส่งคำสั่ง 02h ตามด้วยขนาด file แล้วตามด้วยโปรแกรม

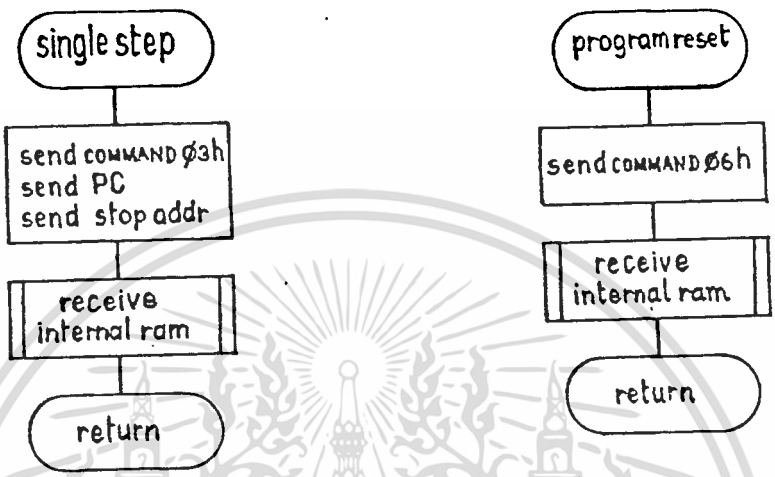


FLOW CHART PROGRAM RUN แสดงการทำงาน 2 แบบของการสั่ง run ตอนแรกจะส่ง breakpoint ที่ไม่ตรงกับ PC ลงไปก่อน หลังจากนั้นจะเช็คเงื่อนไข ถ้ามี breakpoint ที่ตรงกับ PC จะทำ single step 1 ครั้งแล้วจึงส่ง breakpoint ณ ตำแหน่ง breakpoint ลงไป สาเหตุที่ทำเช่นนี้เพราะการตั้ง breakpoint จะใช้การฝัง LCALL ลงที่ตำแหน่งนั้น ซึ่งหากเป็นตำแหน่งเดียวกับ PC จะมีผลให้โปรแกรมไม่สามารถ run ได้ ลำดับต่อไปจึงจะส่งค่าสั่ง run 03h แล้วรอรับผลของการ run กลับมา

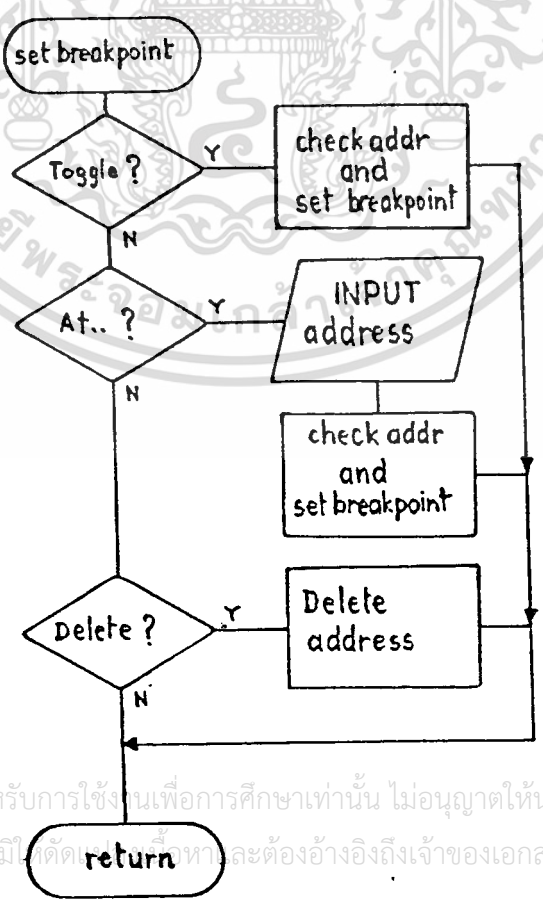


FLOW CHART SINGLE STEP แสดงการส่งค่าสั่ง single step ลงสู่ board และจากนั้นจะรอรับค่าที่จะ return กลับมา

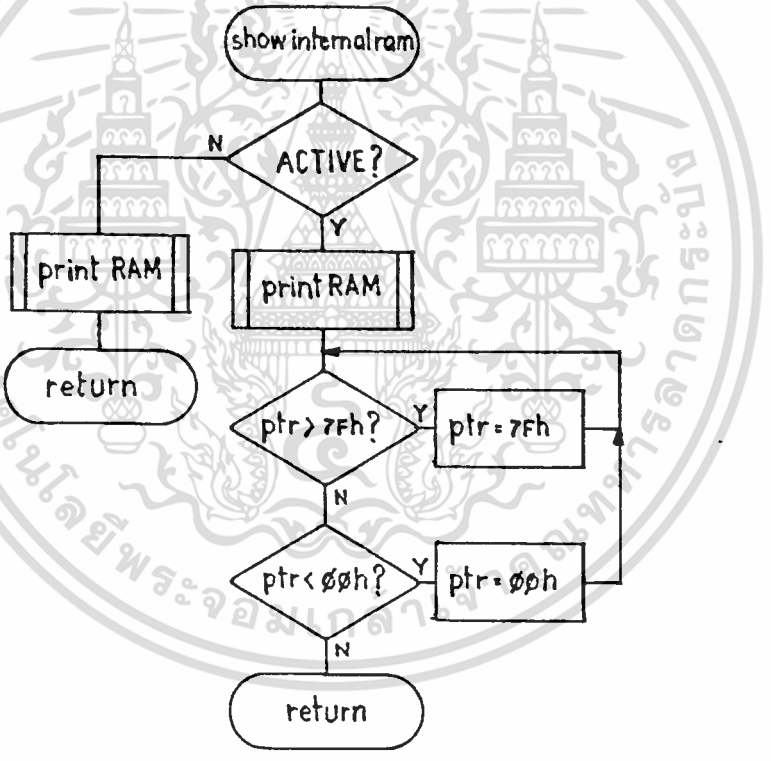
FLOW CHART PROGRAM RESET แสดงการส่งค่าสั่ง RESET โปรแกรมลงสู่ board และจากนั้นจะรอรับค่าที่จะ return กลับมา



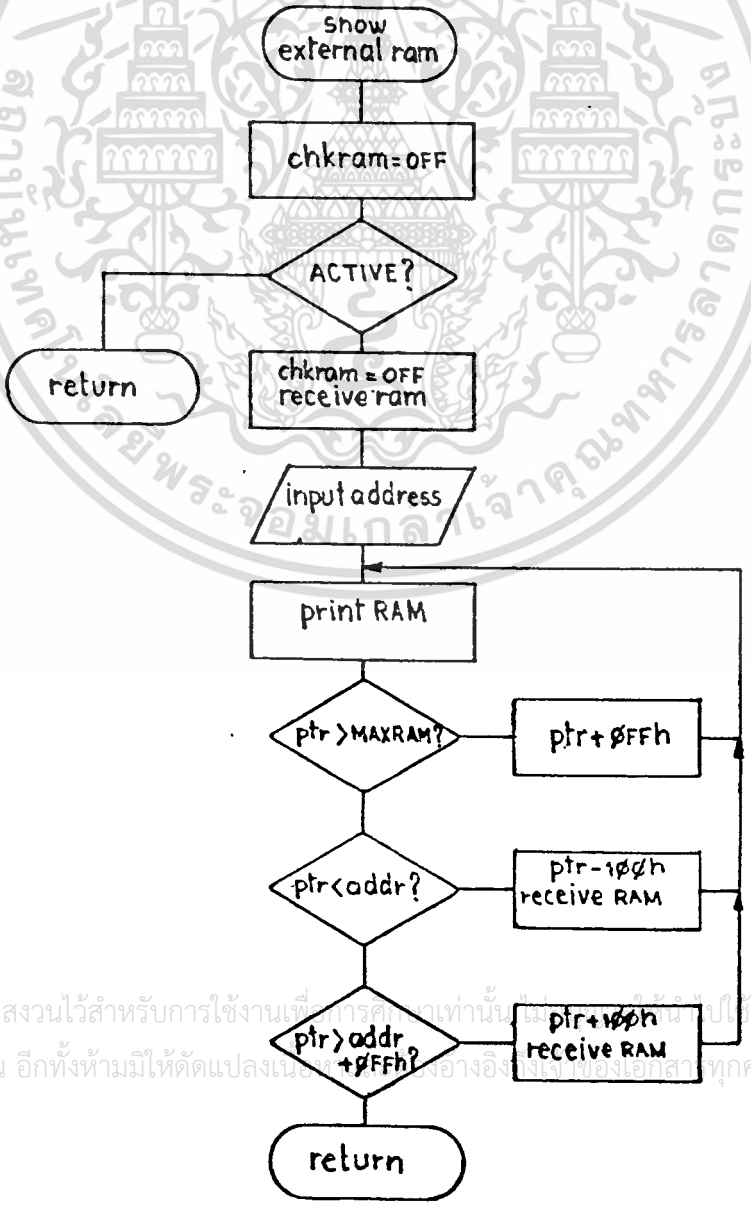
FLOW CHART SET BREAKPOINT แสดงการทำงานของ subroutine ที่จัดการในการตั้ง breakpoint



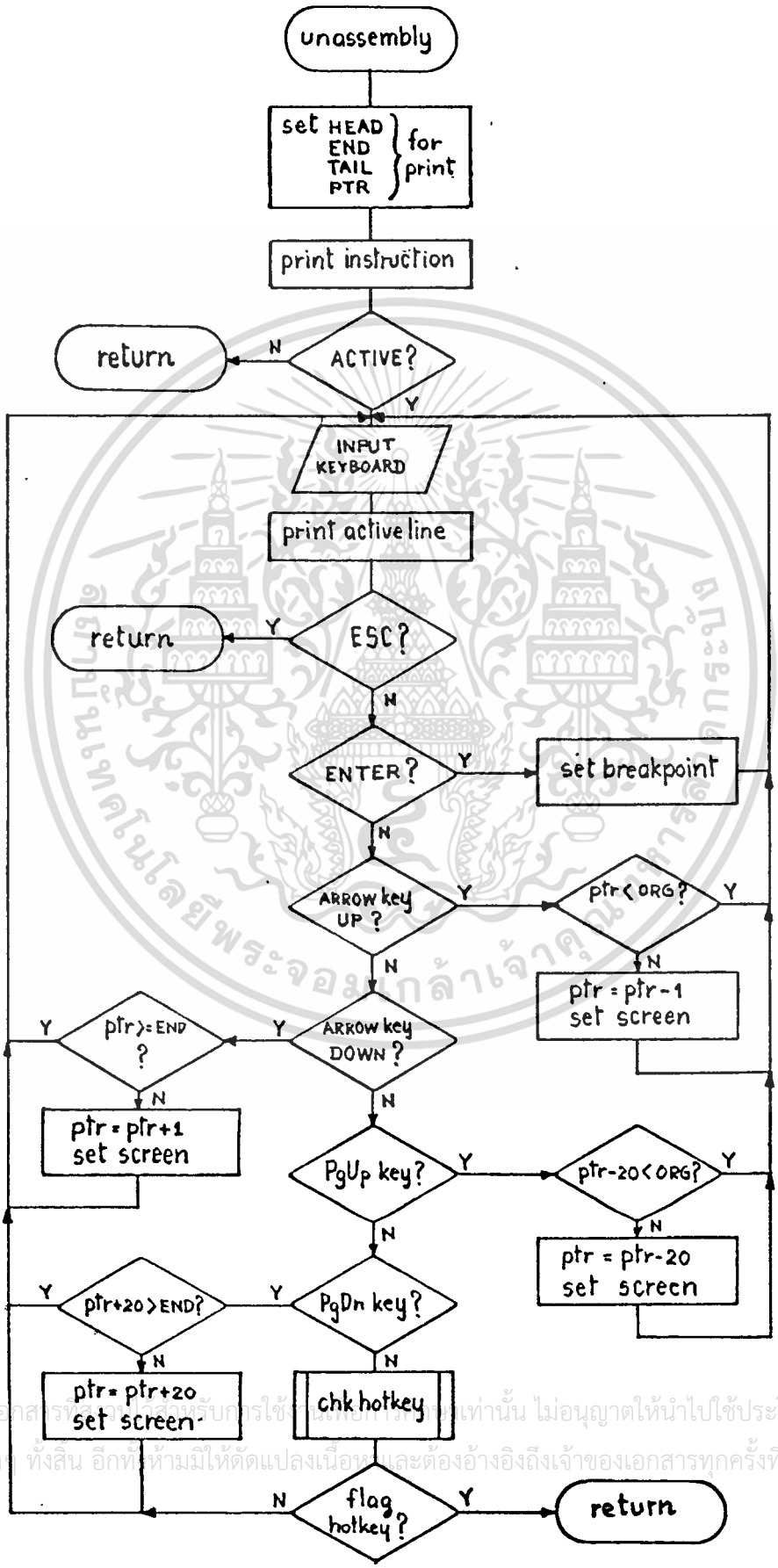
FLOW CHART SHOW INTERNAL RAM แสดงการทำงานในส่วน subroutine show internal ram ที่ address 00h - 7Fh โดยการทำงานจะตรวจสอบว่า active หรือไม่ (active หมายถึง ให้สามารถแก้ไขข้อมูลได้) ถ้าไม่ active จะทำงานเพียงแสดงผลอย่างเดียว หาก active จะเข้าสู่ส่วนการรับ input เพื่อเปลี่ยนแปลงแก้ไขข้อมูล ส่วนของ block เงื่อนไข 2 อันล่าง จะตรวจสอบไม่ให้แสดงข้อมูลเกินที่กำหนดไว้คือ 00h - 7Fh



FLOW CHART SHOW EXTERNAL RAM แสดงการทำงานในส่วน subroutine show external ram ที่ address 0000h - 7FFFh โดยการทำงานจะตรวจสอบว่า active หรือไม่ (active หมายถึง ให้สามารถแก้ไขข้อมูลได้)ถ้าไม่ active จะทำงานเพียงแสดงผลอย่างเดียว หาก active จะเข้าสู่ส่วนการรับ input เพื่อเปลี่ยนแปลงแก้ไขข้อมูล ส่วนของ block เงื่อนไข 2 อันล่าง จะตรวจสอบไม่ให้แสดงข้อมูลเกินที่กำหนดไว้คือ 0000h - 7FFFh ส่วนที่แตกต่างไปจาก show internal ram คือส่วนที่ checkram ซึ่งจะยังผลให้ไม่แสดงผลหากไม่เรียกให้ active ทั้งนี้จะเป็นผลดีหาก target system ไม่มีการต่อ external ram ไว้ จะไม่แสดง external ram ให้ดู ซึ่งหาก target system มีต่อไว้จะต้องเรียกให้ active ก่อนโปรแกรมจึงจะแสดงผลให้



FLOW CHART UNASSEMBLE แสดงส่วนของการแสดงคำสั่งในโปรแกรมทดสอบ ในการทำงานก็จะมีการตรวจสอบ active เช่นเดียวกับโปรแกรมอื่น ๆ



บทที่ 6

ขั้นตอนการพัฒนาและผลการทดลอง

6.1 ขั้นตอนการพัฒนาฮาร์ดแวร์

มีขั้นตอนการพัฒนาดังนี้

1. ออกแบบวงจรส่วน SERIAL PORT ทดลองใช้งานส่วนนี้ก่อน เพราะเป็นส่วนสำคัญในการติดต่อสื่อสารกับ IBM PC
2. ออกแบบส่วนหน่วยความจำร่วม (SHARE MEMORY) โดยในขั้นแรกยังไม่ติดต่อกับ SLAVE
3. ออกแบบส่วน SLAVE และ MULTIPLEX เพื่อใช้หน่วยความจำร่วมร่วมกับ MASTER
4. ปรับปรุงวงจรทั้งหมด
5. นำอุปกรณ์ทั้งหมดลงแผ่นปริ้นท์โดยทำ Wire wrapped
6. ต่อวงจรภายนอกเพื่อทดสอบการทำงานทั้งหมด

6.2 ขั้นตอนการพัฒนาซอฟท์แวร์

มีขั้นตอนการพัฒนาดังนี้

1. วางแนวทางการออกแบบระบบ การติดต่อสื่อสารกับ Board EM51
2. เขียนอัลกอริทึมของโปรแกรม
3. เขียนโปรแกรมขั้นพื้นฐาน เพื่อใช้ทดสอบฮาร์ดแวร์
4. ทดสอบโปรแกรมในการเชื่อมต่อกับฮาร์ดแวร์
5. เพิ่มขีดความสามารถ และทดสอบแก้ไข
6. ปรับปรุงการแสดงผล และระบบทั้งหมดอีกครั้ง

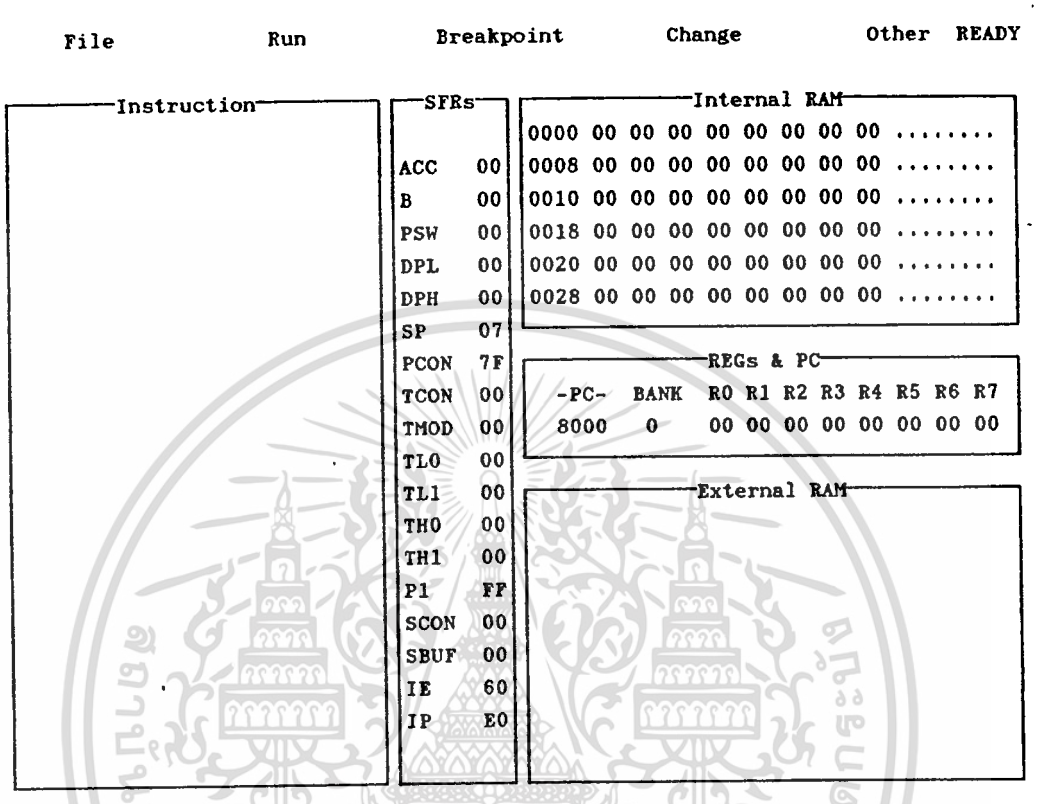
6.3 สรุปการพัฒนาโครงงาน

การออกแบบพัฒนาและทดลองระบบในโครงงานนี้ได้ใช้เทคนิคแบบการพัฒนาโดยตัวเอง (Self development) กล่าวคือจะทำการพัฒนาส่วนต่อไปโดยอาศัยส่วนที่พัฒนามาก่อนแล้วคือเริ่มแรกของโครงงานนี้จะทำการพัฒนาส่วน SERIAL PORT เพื่อติดต่อกับ IBM PC ให้ได้ก่อนเพื่อทดลองรับ-ส่งข้อมูลต่าง ๆ จากนั้นจะพัฒนาส่วนหน่วยความจำร่วม โดยยังใช้ SERIAL PORT อยู่ เช่นรับข้อมูลจาก SERIAL PORT มาเขียนลง RAM แล้วส่งคืน IBM PC เมื่อส่วนนี้เสร็จแล้วจึงออกแบบส่วนสำคัญคือ SLAVE ให้ติดต่อกับ MASTER ให้ได้

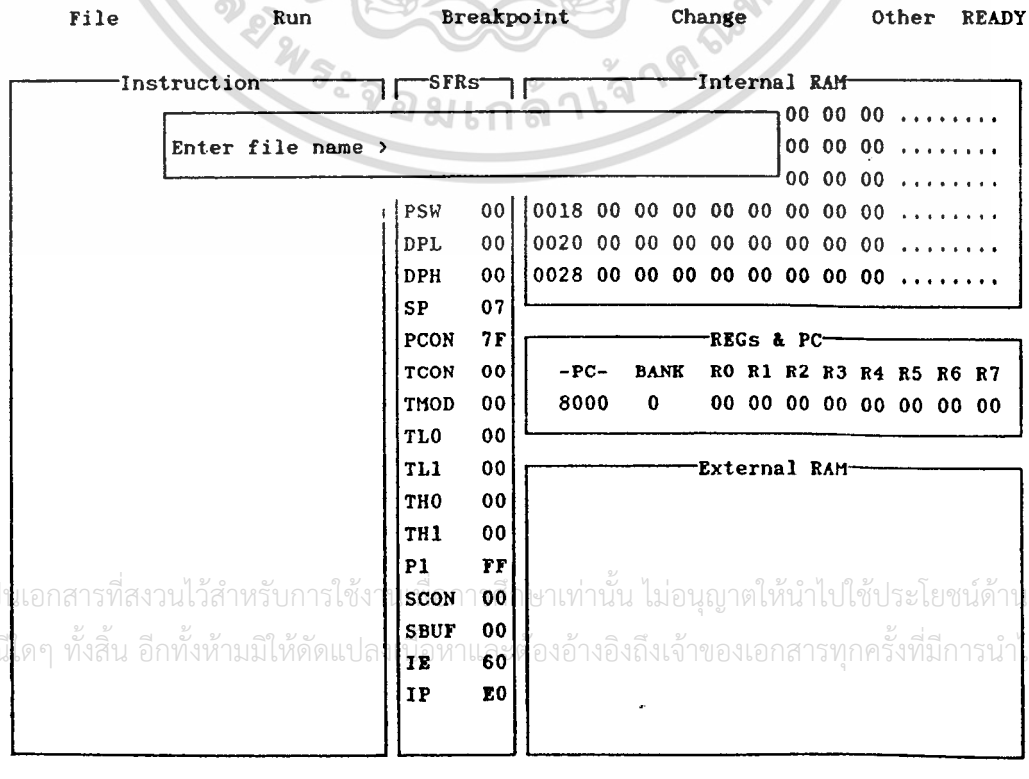
ส่วนทางด้านซอฟท์แวร์บนเครื่อง IBM PC นั้น ได้พัฒนาในส่วนของการจัดการบนเครื่อง IBM PC ในเรื่องของ File, Unssembler ก่อน โดยระบบการติดต่อกับฮาร์ดแวร์จะแยกส่วนมาทำอีกส่วนหนึ่ง หลังจากนั้นจึงนำมาสองส่วนนี้มาเชื่อมเข้าด้วยกัน แล้วนำมาพัฒนาควบคู่ไปกับฮาร์ดแวร์ตลอด

6.4 ผลการทดลอง

- เมื่อเริ่มเข้าสู่โปรแกรม (MENU)



- เมื่อเลือกรายการอ่านแฟ้มข้อมูล (LOAD)



- เมื่อโหลดโปรแกรมที่จะทดสอบแล้ว

```
    File            Run          Breakpoint        Change          Other  READY
USE: C:\AROON\TSEGM.BIN
┌──────Instruction──────┐ ┌─SFRs─┐ ┌──────────Internal RAM──────────┐
805C 128077 LCALL 8077   │      │ 0000 00 00 00 00 00 00 00 00 ........
805F 128077 LCALL 8077   │ ACC  00 0008 00 00 00 00 00 00 00 00 ........
8062 0F     INC   R7     │ B    00 0010 00 00 00 00 00 00 00 00 ........
8063 DED8   DJNZ  R6,D8  │ PSW  00 0018 00 00 00 00 00 00 00 00 ........
8065 80C9   SJMP  C9     │ DPL  00 0020 00 00 00 00 00 00 00 00 ........
8067 C0F9   PUSH  F9     │ DPH  00 0028 00 00 00 00 00 00 00 00 ........
8069 A4     MUL   AB     │ SP   07
806A B099   ANL   C,99   │ PCON 7F ┌──────────REGs & PC──────────┐
806C 9282   MOV   82,C   │ TCON 00   -PC-  BANK  R0 R1 R2 R3 R4 R5 R6 R7
806E F8     MOV   R0,A   │ TMOD 00   8000   0    00 00 00 00 00 00 00 00
806F 8090   SJMP  90     │ TL0  00
8071 8883   MOV   83,R0  │ TL1  00 ┌──────────External RAM──────────┐
8073 C6     XCH   A,@R0  │ TH0  00 │                                │
8074 A186   AJMP  0586   │ TH1  00 │                                │
8076 8E78   MOV   78,R6  │ P1   FF │                                │
8078 FF     MOV   R7,A   │ SCON 00 │                                │
8079 79FF   MOV   R1,#FF │ SBUf 00 │                                │
807B D9FE   DJNZ  R1,FE  │ IE   60 │                                │
807D D8FA   DJNZ  R0,FA  │ IP   E0 │                                │
807F 22     RET          │      │  └────────────────────────────────┘
└──────────────────────┘ └──────┘
F2 transfer F3 load F8 single step F9 run F10 toggle
```

- ผลหลังจากทำงานแล้ว (BREAK หรือ SINGLE STEP)

```
    File            Run          Breakpoint        Change          Other  READY
USE: C:\AROON\TSEGM.BIN
┌──────Instruction──────┐ ┌─SFRs─┐ ┌──────────Internal RAM──────────┐
8053 128077 LCALL 8077   │      │ 0000 0A 8B 00 00 00 00 07 09 ........
8056 128077 LCALL 8077   │ ACC  90 0008 62 80 7B 80 00 08 00 00 b.{.....
8059 128077 LCALL 8077   │ B    00 0010 00 00 00 00 00 00 00 00 ........
805C 128077 LCALL 8077   │ PSW  00 0018 00 00 00 00 00 00 00 00 ........
805F 128077 LCALL 8077   │ DPL  00 0020 00 00 00 00 00 00 00 00 ........
8062 0F     INC   R7     │ DPH  08 0028 00 00 00 00 00 00 00 00 ........
8063 DED8   DJNZ  R6,D8  │ SP   09
8065 80C9   SJMP  C9     │ PCON 7F ┌──────────REGs & PC──────────┐
8067 C0F9   PUSH  F9     │ TCON 00   -PC-  BANK  R0 R1 R2 R3 R4 R5 R6 R7
8069 A4     MUL   AB     │ TMOD 00   807B   0    0A 8B 00 00 00 00 07 09
806A B099   ANL   C,99   │ TL0  00
806C 9282   MOV   82,C   │ TL1  00 ┌──────────External RAM──────────┐
806E F8     MOV   R0,A   │ TH0  00 0000 41 42 43 44 45 46 47 0E ABCDEFG.
806F 8090   SJMP  90     │ TH1  00 0008 0E 0D 0F 0F 0C 0A 03 04 ........
8071 8883   MOV   83,R0  │ P1   00 0010 56 3A 51 E6 08 32 43 13 V:Q..2C.
8073 C6     XCH   A,@R0  │ SCON 00 0018 40 07 59 32 19 06 00 32 @.Y2...2
8074 A186   AJMP  0586   │ SBUF 00 0020 60 65 01 25 39 08 05 20 `e.%9..
8076 8E78   MOV   78,R6  │ IE   E0 0028 00 6E 57 89 84 00 53 91 .nW...S.
8078 FF     MOV   R7,A   │ IP   E0 0030 65 09 65 40 33 29 00 01 e.e@3)..
8079 79FF   MOV   R1,#FF │      0038 E9 48 98 00 16 54 87 66 .H...T.f
└──────────────────────┘ └──────┘ └────────────────────────────────┘
move cursor.
```

- การตั้งจุดหยุด (BREAK POINT)

```
    File          Run          Breakpoint          Change          Other  READY
USE: C:\AROON\TSEGM.BIN
┌──────Instruction──────┐ ┌SFRs┐ ┌──────────Internal RAM──────────┐
805C 128077 LCALL 8077   BREAKPOINT  00 00 00 00 00 00 00 00 ........
805F 128077 LCALL 8077    0 : 8069   00 00 00 00 00 00 00 00 ........
8062 0F     INC   R7      1 : 8071   00 00 00 00 00 00 00 00 ........
8063 DED8   DJNZ  R6,D8   2 : 8078   00 00 00 00 00 00 00 00 ........
8065 80C9   SJMP  C9      3 :        00 00 00 00 00 00 00 00 ........
8067 C0F9   PUSH  F9                 00 00 00 00 00 00 00 00 ........
8069 A4     MUL   AB
806A B099   ANL   C,99    SP   07
806C 9282   MOV   82,C    PCON 7F   ┌──────────REGs & PC──────────┐
806E F8     MOV   R0,A    TCON 00     -PC-  BANK  R0 R1 R2 R3 R4 R5 R6 R7
806F 8090   SJMP  90      TMOD 00    8000   0     00 00 00 00 00 00 00 00
8071 8883   MOV   83,R0   TL0  00
8073 C6     XCH   A,@R0   TL1  00   ┌──────────External RAM──────────┐
8074 A186   AJMP  0586    TH0  00
8076 8E78   MOV   78,R6    TH1  00
8078 FF     MOV   R7,A    P1   FF
8079 79FF   MOV   R1,#FF  SCON 00
807B D9FE   DJNZ  R1,FE   SBUF 00
807D D8FA   DJNZ  R0,FA   IE   60
807F 22     RET           IP   E0
Input address breakpoint. ESC to exit.
```

- เมนู RUN

```
    File          Run          Breakpoint          Change          Other  READY
USE: C:\AROON\T
┌──────Instru──────┐ ┌ Run          F9 ┐ SFRs┐ ┌──────────Internal RAM──────────┐
8053 128077 LC     │ Single step F8     │     0000 0A 8B 00 00 00 00 07 09 ........
8056 128077 LC     │ Program reset      C  90  0008 62 80 7B 80 00 08 00 00 b.{.....
8059 128077 LC     └────────────────────┘  00  0010 00 00 00 00 00 00 00 00 ........
805C 128077 LCALL 8077                PSW  00  0018 00 00 00 00 00 00 00 00 ........
805F 128077 LCALL 8077                DPL  00  0020 00 00 00 00 00 00 00 00 ........
8062 0F     INC   R7                  DPH  08  0028 00 00 00 00 00 00 00 00 ........
8063 DED8   DJNZ  R6,D8               SP   09
8065 80C9   SJMP  C9                  PCON 7F  ┌──────────REGs & PC──────────┐
8067 C0F9   PUSH  F9                  TCON 00    -PC-  BANK  R0 R1 R2 R3 R4 R5 R6 R7
8069 A4     MUL   AB                  TMOD 00   807B   0     0A 8B 00 00 00 00 07 09
806A B099   ANL   C,99                TL0  00
806C 9282   MOV   82,C                TL1  00  ┌──────────External RAM──────────┐
806E F8     MOV   R0,A                TH0  00
806F 8090   SJMP  90                  TH1  00  0000 41 42 43 44 45 46 47 0E ABCDEFG.
8071 8883   MOV   83,R0               P1   00  0008 0E 0D 0F 0F 0C 0A 03 04 ........
8073 C6     XCH   A,@R0               SCON 00  0010 56 3A 51 E6 08 32 43 13 V:Q..2C.
8074 A186   AJMP  0586                SBUF 00  0018 40 07 59 32 19 06 00 32 @.Y2...2
8076 8E78   MOV   78,R6               IE   E0  0020 60 65 01 25 39 08 05 20 `e.%9..
8078 FF     MOV   R7,A                IP   E0  0028 00 6E 57 89 84 00 53 91 .nW...S.
8079 79FF   MOV   R1,#FF                       0030 65 09 65 40 33 29 00 01 e.e@3)..
                                               0038 E9 48 98 00 16 54 87 66 .H...T.f
F2 transfer F3 load F8 single step F9 run F10 toggle
```

- เมนู CHANGE REGISTER

```
    File              Run          Breakpoint        Change          Other  READY
USE: C:\AROON\TSEGM.BIN
┌──────Instruction──────┐ ┌─SFRs─┐ ┌──────┐ Pc & register ┌M┐
805C 128077 LCALL 8077  │ │      │ │0000 0│ Sfrs          │0 00 ........
805F 128077 LCALL 8077  │ │ACC  00│ │0008 0│ Internal ram  │0 00 ........
8062 0F     INC   R7    │ │B    00│ │0010 0│ External ram  │0 00 ........
8063 DED8   DJNZ  R6,D8 │ │PSW  00│ │0018 0└──────────────┘0 00 ........
8065 80C9   SJMP  C9    │ │DPL  00│ │0020 00 00 00 00 00 00 00 00 ........
8067 C0F9   PUSH  F9    │ │DPH  00│ │0028 00 00 00 00 00 00 00 00 ........
8069 A4     MUL   AB    │ │SP   07│ └──────────────────────────────────────┘
806A B099   ANL   C,99  │ │PCON 7F│ ┌──────REGs & PC──────┐
806C 9282   MOV   82,C  │ │TCON 00│ │ -PC-  BANK  R0 R1 R2 R3 R4 R5 R6 R7 │
806E F8     MOV   R0,A  │ │TMOD 00│ │ 8000   0    00 00 00 00 00 00 00 00 │
806F 8090   SJMP  90    │ │TL0  00│ └─────────────────────┘
8071 8883   MOV   83,R0 │ │TL1  00│ ┌──────External RAM──────┐
8073 C6     XCH   A,@R0 │ │TH0  00│ │                        │
8074 A186   AJMP  0586  │ │TH1  00│ │                        │
8076 8E78   MOV   78,R6 │ │P1   FF│ │                        │
8078 FF     MOV   R7,A  │ │SCON 00│ │                        │
8079 79FF   MOV   R1,#FF│ │SBUF 00│ │                        │
807B D9FE   DJNZ  R1,FE │ │IE   60│ │                        │
807D D8FA   DJNZ  R0,FA │ │IP   E0│ │                        │
807F 22     RET         │ │      │ │                        │
└─────────────────────┘ └──────┘ └────────────────────────┘
F2 transfer F3 load F8 single step F9 run F10 toggle
```

บทที่ 7

บทสรุปและวิจารณ์

จากการพัฒนาและสร้างวงจรทดสอบภายนอกหรือระบบเป้าหมาย (Target System) ต่อกับโครงงาน โดยการเขียนโปรแกรมทดสอบและทำการแอสเซมเบลอร์โดยใช้โปรแกรม CROSS16 แอสเซมเบลอร์เป็นไฟล์นามสกุล BIN(BINARY FILE) หรือจะใช้โปรแกรมที่เป็นแอสเซมเบลอร์ตัวอื่นมาทำการแอสเซมเบลอร์ก็ได้ ให้เป็น OBJECT FILE เพราะโครงงานนี้จะโหลดไฟล์ที่เป็น OBJECT FILE เท่านั้น เนื่องจากไม่ได้เขียนโปรแกรมสำหรับทำการแอสเซมเบลอร์เอง จากโปรแกรมที่แอสเซมเบลอร์แล้วจะโหลดลงหน่วยความจำร่วม แล้วทดสอบดู จากการทดลองพบว่าสามารถทำงานได้ถูกต้อง อีกทั้งยังมารถทำการดีบักได้โดยตรงตามที่ต้องการ ทำให้เกิดความสะดวกและความรวดเร็วในการที่จะพัฒนาโปรแกรม แต่ก็มีข้อจำกัดบางประการที่จะต้องพิจารณาเพื่อนำไปปรับปรุงในภายหลัง

7.1 ข้อจำกัดทางฮาร์ดแวร์

1. หน่วยความจำบนระบบเป้าหมาย (Target system) ที่สามารถนำมาเชื่อมต่อกับระบบของโครงงานนี้ จะเป็นดังนี้

-ต้องไม่มี External ROM คือให้ถอด External ROM บนระบบเป้าหมายออกให้หมด แล้ว Load โปรแกรมที่ แอสเซมเบลอร์แล้วจากแผ่น Disk ลงไปในหน่วยความจำร่วมแทน

-External RAM บนระบบเป้าหมายต้องมี address ตั้งแต่ 0000h - 7FFFh เท่านั้น

2. วงจรภายนอกหรือระบบเป้าหมายไม่สามารถใช้งานขา INT 0 ได้ เนื่องจากได้ถูกใช้ในการติดต่อกับ Master ไปแล้ว

3. ผู้ใช้ต้องตั้ง ORIGIN ของโปรแกรมทดสอบไว้ที่ตำแหน่ง address 8000h เท่านั้น

4. มีบางคำสั่งไม่สามารถทำ Single step หรือตั้ง Breakpoint ได้ เนื่องจาก การทำ Single step หรือตั้ง Breakpoint จะใช้วิธีฝังคำสั่ง LCALL ลงในโปรแกรม คำสั่งดังกล่าวได้แก่

- คำสั่งจำพวก JUMP,CALL,RETURN คำสั่งพวกนี้ต้องตั้ง Breakpoint ที่ address ที่ JUMP หรือ CALL ไปก่อน จึงจะทำ Single step ได้

- คำสั่ง DJNZ Rn,Label คำสั่งนี้สามารถทำ Single step ได้เมื่อ Rn ลดค่าลงมาเป็น 0 เท่านั้น ถ้า Rn ไม่เป็น 0 ก็ต้องไปตั้ง Breakpoint ที่ address ของ Label ไว้ก่อน

- คำสั่ง CJNE คำสั่งนี้ต้องแก้ไขโดยวิธีการเดียวกับ DJNZ

ตัวอย่าง

```
          ORG   8000H

START:    MOV   A,#80H
          MOV   R2,#08H
DISP:     MOV   P1,A
DELAY:    MOV   R0,#0FFH
DELAY1:   MOV   R1,#0FFH
          DJNZ  R1,$          ---+--- ทำ Single step
          DJNZ  R0,DELAY1     ---|    หรือตั้ง Breakpoint
          RR    A             ---|    ไม่ได้
          DJNZ  R2,DISP       ---+
          SJMP  START

          END
```

7.2 ข้อจำกัดทางด้านซอฟท์แวร์

1. ใช้ได้เฉพาะไฟล์แบบ BINARY(.BIN) เท่านั้น
2. ต้องตั้ง ORG ไว้ที่ address 8000h เท่านั้น
3. ไม่สามารถแก้ไขโปรแกรม code จากตัวซอฟท์แวร์เอง
4. ไม่ได้มีการเตรียมการค้นหาข้อมูลที่ต้องการให้
5. การกำหนดจุดหยุดตั้งได้ไม่เกิน 4 จุด
6. ไม่สามารถ insert หรือ delete ข้อมูลได้

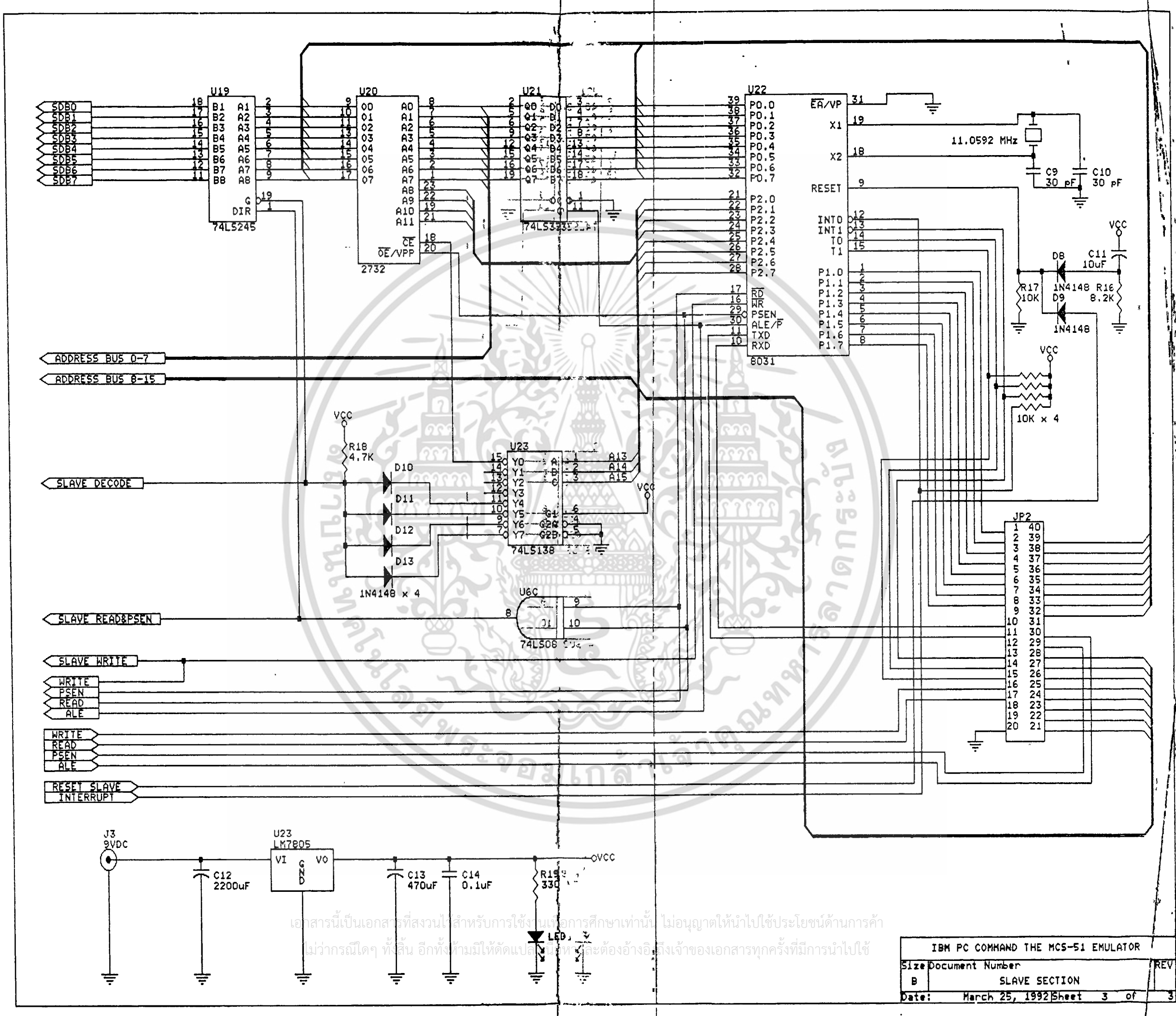
7.3 บทสรุป

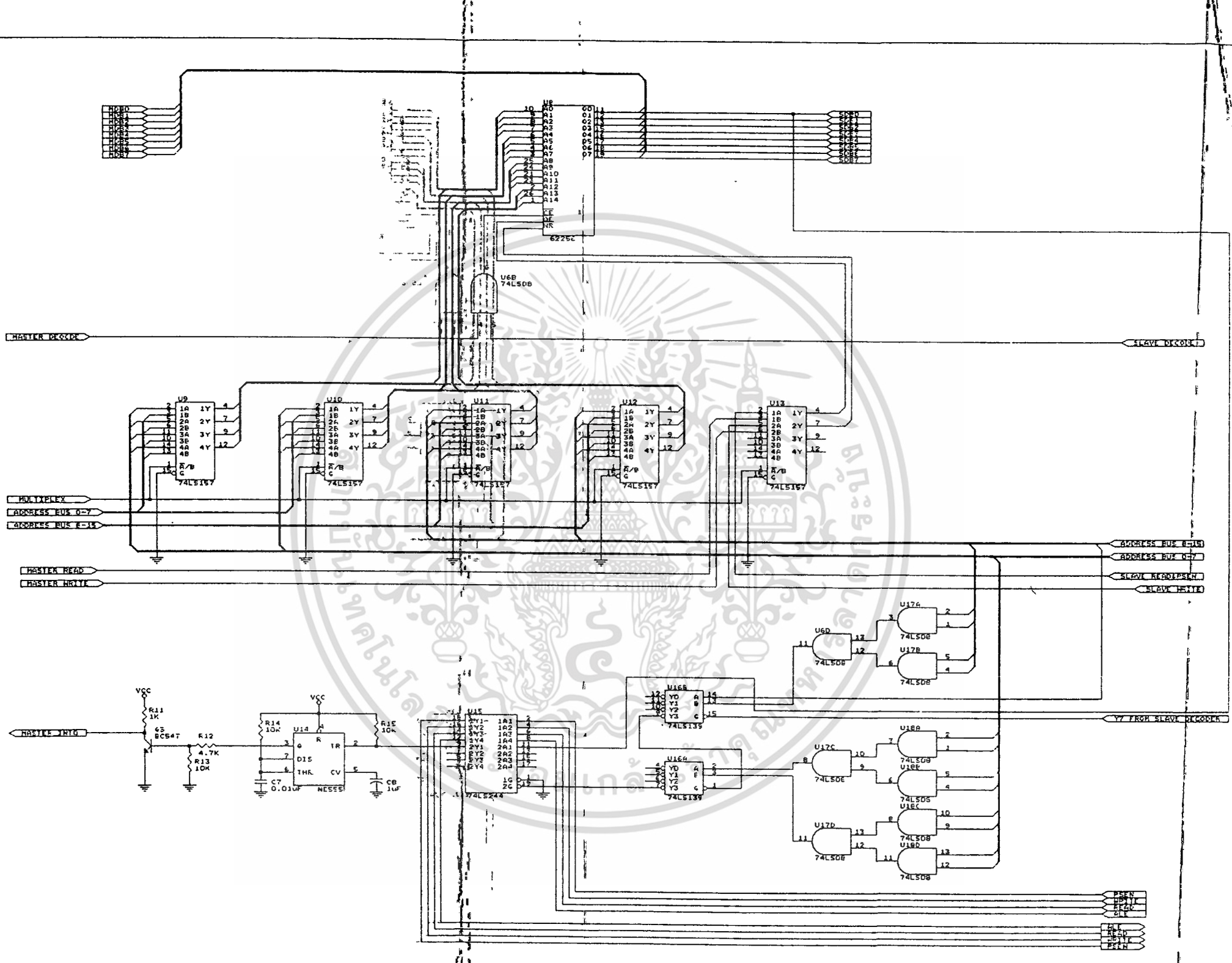
จากการนำเอาระบบทั้งหมดมาต่อร่วมกับวงจรภายนอก จะทำให้การพัฒนาโปรแกรมเป็นไปได้สะดวกยิ่งขึ้น เพราะสามารถทำให้เห็นและสามารถติดตามการทำงานของโปรแกรมได้ตลอด โดยเฉพาะระบบ FULL SCREEN ซึ่งแสดงคำสั่งที่ทำ Single step และ Run รวมทั้งการแก้ไขค่าต่าง ๆ ที่สามารถเลื่อน CURSOR ไปทำการแก้ไขได้อย่างสะดวก

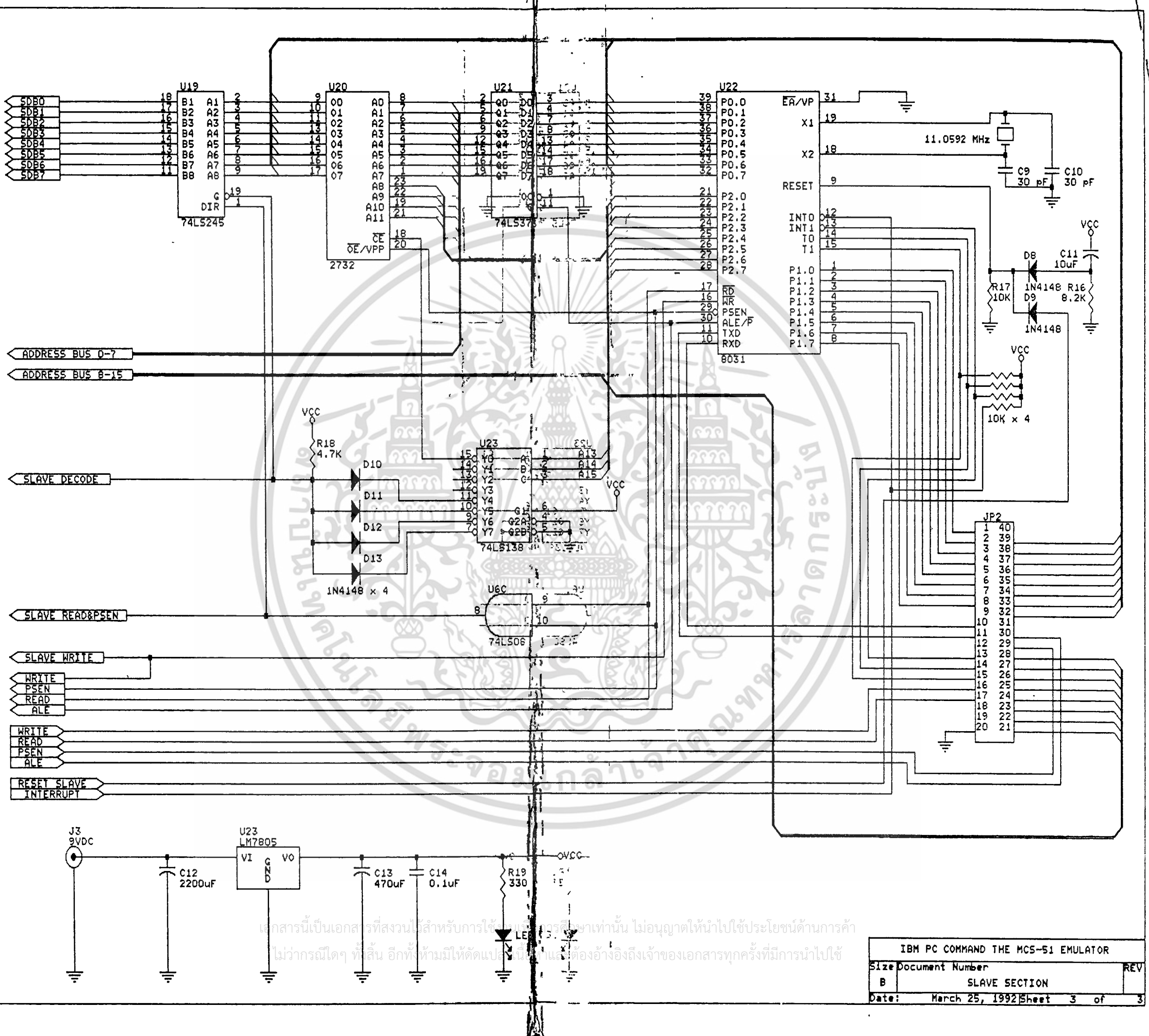
โครงสร้างทั้งหมดในส่วนฮาร์ดแวร์และซอฟท์แวร์ สามารถจะนำไปประยุกต์ให้ใช้กับระบบเป้าหมาย (Target system) ได้อย่างปรกติ โดยมีทรัพยากรทางฮาร์ดแวร์ให้ใช้เกือบครบถ้วน สิ่งที่ขาดไปได้แก่ พื้นที่หน่วยความจำบางส่วน INTO

การพัฒนา MCS-51 อีมูเลเตอร์บอร์ดนี้ ให้มีขีดความสามารถเพิ่มขึ้นนั้นจะต้องพัฒนาในส่วนของโปรแกรม Master และ Slave ควบคู่กันไป และในส่วนของซอฟท์แวร์บน IBM PC สามารถเพิ่มเติมขีดความสามารถได้อีก ได้แก่ การทำระบบ Directory การจัดการเกี่ยวกับ File การทำ Assembly และ Editor ในตัว และการจัด window บนจอภาพให้เพิ่มและลดขนาดได้ตามต้องการ ซึ่งเป็นไปได้

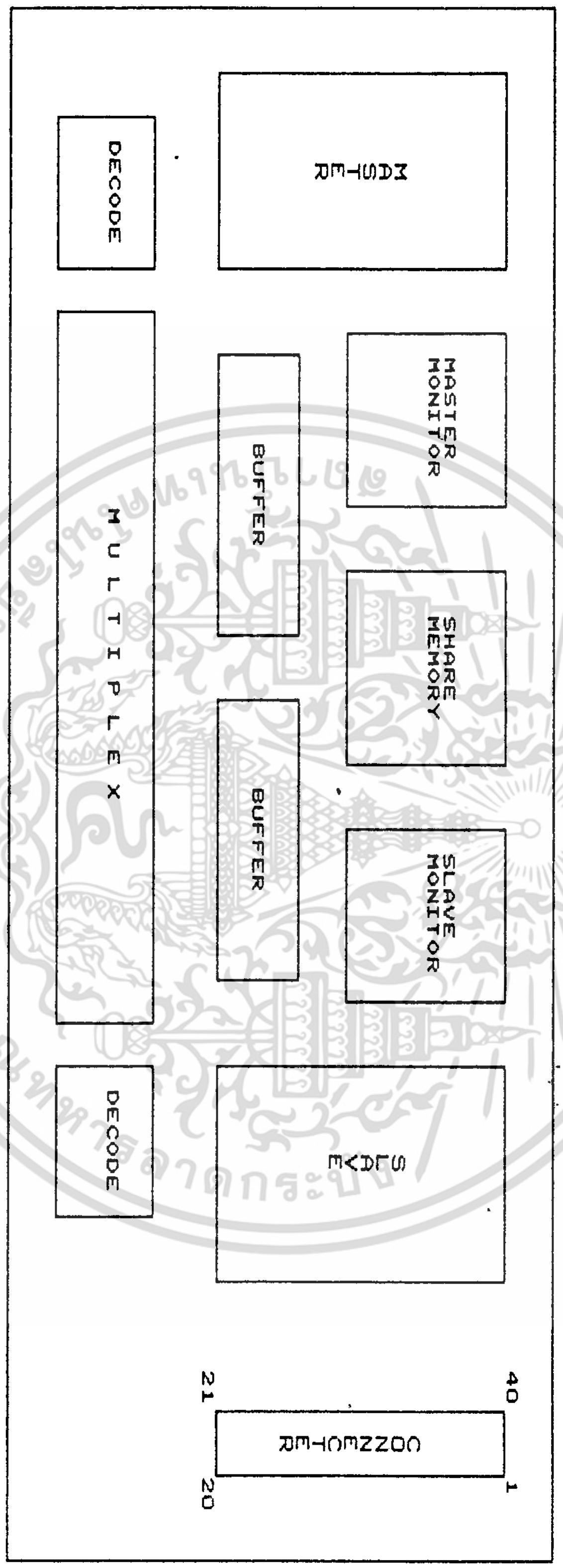
ภาคผนวก ก







IBM PC COMMAND THE MCS-51 EMULATOR		
Size B	Document Number SLAVE SECTION	REV
Date: March 25, 1992	Sheet 3 of 3	



การจัดวางอุปกรณ์บนแผ่นวงจรพิมพ์

ภาคผนวก ช

MCS®-51 INSTRUCTION SET

Table 10. 8051 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

Rn — Register R7-R0 of the currently selected Register Bank.

direct — 8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].

@Ri — 8-bit internal data RAM location (0-255) addressed indirectly through register R1 or R0.

#data — 8-bit constant included in instruction.

#data 16 — 16-bit constant included in instruction.

addr 16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.

addr 11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.

bit — Direct Addressed bit in Internal Data RAM or Special Function Register.

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,Rn	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS (Continued)				
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS (Continued)				
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	Rn,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12

Mnemonic		Description	Byte	Oscillator Period
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from Interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
JMP	@A + DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A, # data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	Rn, # data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri, # data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

Table 11. Instruction Opcodes in Hexadecimal Order

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A, # data
25	2	ADD	A,data addr
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr, code addr
31	2	ACALL	code addr
32	1	RETI	

Hex Code	Number of Bytes	Mnemonic	Operands
33	1	RLC	A
34	2	ADDC	A, # data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr, # data
44	2	ORL	A, # data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr, # data
54	2	ANL	A, # data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr,A
63	3	XRL	data addr, # data
64	2	XRL	A, # data
65	2	XRL	A,data addr

Table 11. Instruction Opcodes in Hexadecimal Order (Continued)

Hex Code	Number of Bytes	Mnemonic	Operands
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A + DPTR
74	2	MOV	A, # data
75	3	MOV	data addr, # data
76	2	MOV	@R0, # data
77	2	MOV	@R1, # data
78	2	MOV	R0, # data
79	2	MOV	R1, # data
7A	2	MOV	R2, # data
7B	2	MOV	R3, # data
7C	2	MOV	R4, # data
7D	2	MOV	R5, # data
7E	2	MOV	R6, # data
7F	2	MOV	R7, # data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A + PC
84	1	DIV	AB
85	3	MOV	data addr, data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR, # data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A + DPTR
94	2	SUBB	A, # data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0
97	1	SUBB	A,@R1
98	1	SUBB	A,R0
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,/bit addr
A1	2	AJMP	code addr
A2	2	MOV	C,bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,data addr
A7	2	MOV	@R1,data addr
A8	2	MOV	R0,data addr
A9	2	MOV	R1,data addr
AA	2	MOV	R2,data addr
AB	2	MOV	R3,data addr
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,/bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A, # data,code addr
B5	3	CJNE	A,data addr,code addr
B6	3	CJNE	@R0, # data,code addr
B7	3	CJNE	@R1, # data,code addr
B8	3	CJNE	R0, # data,code addr
B9	3	CJNE	R1, # data,code addr
BA	3	CJNE	R2, # data,code addr
BB	3	CJNE	R3, # data,code addr
BC	3	CJNE	R4, # data,code addr
BD	3	CJNE	R5, # data,code addr
BE	3	CJNE	R6, # data,code addr
BF	3	CJNE	R7, # data,code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3

Table 11. Instruction Opcodes in Hexadecimal Order (Continued)

Hex Code	Number of Bytes	Mnemonic	Operands
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr,code addr
D6	1	XCHD	A,@R0
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,code addr
D9	2	DJNZ	R1,code addr
DA	2	DJNZ	R2,code addr
DB	2	DJNZ	R3,code addr
DC	2	DJNZ	R4,code addr
DD	2	DJNZ	R5,code addr
DE	2	DJNZ	R6,code addr
DF	2	DJNZ	R7,code addr
E0	1	MOVX	A,@DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A,@R0
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,data addr
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	data addr,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A

เอกสารอ้างอิง

1. "MCS-51 Microcontroller Data Book", INTEL, 1990.
2. "IBM Technical Reference", IBM Corp., 1983.
3. "Lawrence H. Miller, Alaxander E. Quilici, "The Official Borland Turbo C Survival Guide", John Wiley & Sons, 1988.
4. Herberd Schildt, "C Power User's Guide", McGraw Hill, 1988.
5. คู่มือเทียบเบอร์ไอซี ทีทีแอล, ซีเอ็ดยูเคชั่น, พิมพ์ครั้งที่ 6, พศ. 2531
6. คู่มือไอซีชิพซัพพอร์ท และหน่วยความจำ, ซีเอ็ดยูเคชั่น, พศ. 2529

กิตติกรรมประกาศ

คณะผู้จัดทำขอขอบพระคุณท่านอาจารย์ที่ปรึกษาคือ ผศ.พิพัฒน์ เลาหสงคราม ซึ่งคอยแนะนำหลักการ วางแนวทางพร้อมทั้งวางแผนการทำงาน และคอยให้คำปรึกษาในเรื่องเทคนิคต่าง ๆ โดยเฉพาะอย่างยิ่งทางด้านฮาร์ดแวร์ตลอดจนท่านยังให้คำปรึกษาในทุก ๆ เรื่อง ทั้งยังผู้ที่ติดตามผลการทำงานและให้กำลังใจตลอดเวลาที่ผ่านมาจนทำให้ปริญญานิพนธ์สำเร็จลุล่วงไปด้วยดี และท่านที่จะไม่กล่าวถึงไม่ได้เลยในที่นี้ คือ ท่านกรรมการสอบปริญญานิพนธ์และท่านอาจารย์ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม ที่เอื้อเฟื้อสถานที่ตลอดจนอุปกรณ์ต่าง ๆ ที่ใช้ในการทำปริญญานิพนธ์

สุดท้ายนี้ขอขอบพระคุณบุคคลที่มีส่วนเกี่ยวข้องทางด้านเทคนิค และการจัดพิมพ์ปริญญานิพนธ์ ตลอดจนผู้ที่ให้ความสนใจและช่วยเหลือตลอดมา ณ โอกาสนี้ด้วย