ปริญญานิพนธ์

เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51

MEMORY EMULATOR AND MCS-51 DEBUGGER PROGRAM

1. นายโกเมศร์ แจ่มจันทร์
2. นายทองคำ เกตุโชติ
3. นายอมรรัตน์ จันต๊ะมณี
4. นายอรรถวิทย์ ปานสมสวย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาอิเล็กทรอนิกส์และคอมพิวเตอร์

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ปริญญานิพนธ์ เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51

MEMORY EMULATOR AND MCS-51 DEBUGGER PROGRAM

ชื่อนักศึกษา 1. นายโกเมศร์ แจ่มจันทร์ รหัสประจำตัว 37031302

2. นายทองคำ เกตุโชติ รหัสประจำตัว 37031306

3. นายอมรรัตน์ จันต๊ะมณี รหัสประจำตัว 37031331

4. นายอรรถวิทย์ ปานสมสวย รหัสประจำตัว 37031334

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** อิเล็กทรอนิกส์และคอมพิวเตอร์

อาจารย์ผู้ควบคุมปริญญานิพนธ์

1. อาจารย์สุชิน อาจหาญ
2. อาจารย์ปิยะ จิตธรรมมาภิรมย์
3. อาจารย์ประเสริฐ เคนพันค้อ

คณะกรรมการสอบปริญญานิพนธ์	ลายมือชื่อ
1. อาจารย์สุชิน อาจหาญ	
2. อาจารย์ปิยะ จิตธรรมมาภิรมย์	
3. อาจารย์ประเสริฐ เคนพันค้อ	
4. อาจารย์กิติพงค์ มะโน	
5. อาจารย์วรวิทย์ สมหา	

วันเดือนปีที่สอบ วันที่ 8 ธันวาคม 2538 **เวลา** 10.00 ถึง 12.00 น.

สถานที่สอบ ห้อง ค.303 คณะครุศาสตร์อุตสาหกรรม

ภาควิชารับรองแล้ว

.....

(ผศ.ดร.ธีระพล เทพหัสดิน ณ อยุธยา)

ประธานภาควิชาครุศาสตร์วิศวกรรม

วันที่ ..... เดือน ..... พ.ศ. .....

ปริญญานิพนธ์

เรื่อง เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51

MEMORY EMULATOR AND MCS-51 DEBUGGER PROGRAM

ผู้จัดทำ

1. นายโกเมศร์ แจ่มจันทร์
2. นายทองคำ เกตุโชติ
3. นายอมรรัตน์ จันต๊ะมณี
4. นายอรรถวิทย์ ปานสมสวย

อาจารย์ที่ปรึกษา

ลงนาม.....

(อาจารย์สุชิน อาจหาญ)

ลงนาม.....

(อาจารย์ปิยะ จิตธรรมมาภิรมย์)

ลงนาม.....

(อาจารย์ประเสริฐ เคนพันค้อ)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

ลงนาม.....

(ผศ.ดร.ธีระพล เทพหัสดิน ณ อยุธยา)

ปริญญานิพนธ์

เรื่อง เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51

MEMORY EMULATOR AND MCS-51 DEBUGGER PROGRAM

จุดประสงค์

1. เพื่อศึกษาโครงสร้างและระบบการทำงานของไมโครคอนโทรลเลอร์ตระกูล MCS-51
2. เพื่อศึกษาหลักการเขียนโปรแกรมคำสั่งการทำงานของไมโครคอนโทรลเลอร์ตระกูล MCS-51
3. เพื่อออกแบบเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51
4. เพื่อสร้างเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51
5. เพื่อนำเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 มาใช้งานได้

ประโยชน์ที่คาดว่าจะได้รับ

1. เข้าใจถึงโครงสร้างและระบบการทำงานของไมโครคอนโทรลเลอร์ตระกูล MCS-51
2. สามารถอธิบายถึงหลักการเขียนโปรแกรมคำสั่งการทำงานของไมโครคอนโทรลเลอร์ตระกูล MCS-51
3. สามารถสร้างเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51
4. สามารถนำเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 มาใช้งานได้

เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51

นายโกเมศร์ แจ่มจันทร์
นายทองคำ เกตุโชติ
นายอมรรัตน์ จันต๊ะมณี
นายอรรถวิทย์ ปานสมสวย

อาจารย์ที่ปรึกษา
อาจารย์สุชิน อาจหาญ
อาจารย์ปิยะ จิตธรรมมาภิรมย์
อาจารย์ประเสริฐ เคนพันค้อ

ปีการศึกษา 2538

บทคัดย่อ

เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ได้ถูกประดิษฐ์ขึ้นสำหรับใช้งานเป็นอีพรอมอีมูเลเตอร์ (สำหรับแผงวงจรที่ใช้ไมโครโปรเซสเซอร์ทุกเบอร์)

โดยสามารถแสดงค่าของรีจิสเตอร์และหน่วยความจำเฉพาะบอร์ดทดลองที่ใช้ไมโครโปรเซสเซอร์ตระกูล MCS-51 โดยมีคำสั่งต่างๆ เป็นเมนูในโปรแกรมที่ทำงานบนเครื่องไมโครคอมพิวเตอร์

เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 จะช่วยให้ผู้ใช้สามารถเข้าใจระบบการทำงานและมองเห็นค่าต่างๆ ในหน่วยความจำของไมโครโปรเซสเซอร์ตระกูล MCS-51 ซึ่งสามารถนำไปใช้เป็นชุดฝึกในการเรียนการสอนวิชาไมโครโปรเซสเซอร์หรือใช้ในการสร้างและพัฒนาระบบที่ใช้ไมโครโปรเซสเซอร์ได้

MEMORY EMULATOR AND MCS-51 DEBUGGER PROGRAM

MR.KOMET JAMJUN
MR.TONGKUM KADCHOTI
MR.AMORNRAT JUNTAMANEE
MR.ATTAWIT PANSOMSUAY

ADVISORS

MR.SUCHIN ADHAN
MR.PIYA JITTROMMAPIROM
MR.PRASERT KENPANKHO

1995

ABSTRACT

Memory emulator and MCS-51 debugger program was create for using to eprom emulator (for every microprocessor board). It can show data in registers and memory for the MCS-51 microprocessor board by choose from command menu in MEMDP program that running on microcomputer.

Memory emulator and MCS-51 debugger program can help user to know about operating system and can see data in the memory unit of MCS-51 microprocessor which can use as the training in the educate of microprocessor subject or for create and development microprocessor system.

กิตติกรรมประกาศ

ขอขอบพระคุณท่านอาจารย์ผู้ควบคุมปริญญานิพนธ์ และอาจารย์ประจำภาควิชาครุศาสตร์วิศวกรรมทุกท่านที่ให้คำแนะนำที่เป็นประโยชน์ต่อการทำโครงงานมาโดยตลอด

คณะผู้จัดทำขอยกความดีที่ได้กระทำมาจงน้อมนำสู่ผู้มีพระคุณทุกท่าน

สารบัญ

สารบัญภาพ

บทที่ 1
บทนำ

ในปัจจุบันนี้ไมโครโปรเซสเซอร์ (Microprocessor) ได้เข้ามามีผลอย่างมากต่อการออกแบบระบบอิเล็กทรอนิกส์ (Electronics) ซอฟท์แวร์ (Software) ที่เก็บไว้ในหน่วยความจำ (Memory) ทำให้ฮาร์ดแวร์ (Hardware) ที่เหมือนกันสามารถทำงานที่แตกต่างกันได้ การออกแบบฮาร์ดแวร์ เช่นหน่วยความจำและพอร์ต (Port) จะเป็นตัวกำหนดขอบเขตการทำงานของซอฟท์แวร์ขณะที่ซอฟท์แวร์เป็นตัวจะกำหนดหน้าที่การทำงาน

ข้อดีของการออกแบบระบบที่ใช้ไมโครโปรเซสเซอร์ คือสามารถนำไปประยุกต์ใช้ได้อย่างกว้างขวาง การควบคุมด้วยซอฟท์แวร์ทำให้เปลี่ยนแปลงการทำงานได้ง่าย และสามารถทำงานที่ซับซ้อนได้ง่ายกว่าวิธีอื่น ความสามารถในการคำนวณทำให้การวิเคราะห์และแปลข้อมูลโดยเครื่องมือทำได้ ซึ่งแต่เดิมจะแสดงได้เฉพาะข้อมูลดิบ โครงสร้างพื้นฐานของระบบที่ใช้ไมโครโปรเซสเซอร์ประกอบด้วยไมโครโปรเซสเซอร์,หน่วยความจำ พอร์ต ต่อกันด้วยบัสแอดเดรส (Address Bus) บัสข้อมูล (Data Bus) และบัสควบคุม (Control Bus) ซึ่งปกติจะเป็นวงจรที่ออกแบบง่ายและมีราคาถูก

การพัฒนาระบบที่ใช้ไมโครโปรเซสเซอร์ซึ่งเป็นส่วนประกอบอาจพอแบ่งได้เป็น 2 ส่วน คือการพัฒนาฮาร์ดแวร์ และการพัฒนาซอฟท์แวร์ เครื่องมือที่ช่วยในการพัฒนาระบบมีชื่อเรียกรวมๆ ว่าระบบพัฒนาไมโครโปรเซสเซอร์ (Microprocessor Development System)

ในยุคเริ่มต้นการพัฒนาฮาร์ดแวร์และการพัฒนาซอฟท์แวร์ต้องดำเนินงานแยกกัน เมื่อต้องการทดสอบการทำงานร่วมกันจะใส่ซอฟท์แวร์ที่ใช้งานจริงไว้ในรอมแล้วทดสอบใช้งานโดยไม่สามารถควบคุมหรือสังเกตการทำงานของโปรแกรม (Program) ซึ่งถ้าพบข้อบกพร่องของระบบ อาจจะเกิดจากปัญหาทั้งทางฮาร์ดแวร์และซอฟท์แวร์ ในขั้นตอนนี้ถ้าระบบพัฒนาไมโครโปรเซสเซอร์ไม่มีอุปกรณ์ที่เหมาะสม ต้องใช้เวลาในการดำเนินการมาก ดังนั้นจึงต้องการอุปกรณ์สำหรับระบบพัฒนาไมโครโปรเซสเซอร์ที่สามารถทดสอบซอฟท์แวร์ที่ใช้งานจริง ในฮาร์ดแวร์ต้นแบบ โดยสามารถสังเกตและควบคุมการทำงานของโปรแกรมได้

ขั้นตอนการออกแบบที่ใช้ไมโครโปรเซสเซอร์

การออกแบบที่ใช้ไมโครโปรเซสเซอร์มีขั้นตอนในรูปที่ 1.2

1. ให้ข้อกำหนดรายละเอียดของระบบ

ขั้นแรกของการออกแบบทั่วไปคือ การให้ข้อกำหนดรายละเอียดของระบบ ข้อกำหนดนี้จะเป็นการบอกว่าระบบจะต้องทำงานอะไรได้บ้าง แต่ยังไม่ครอบคลุมถึงวิธีการที่ใช้ในการทำงานนั้น

2. การออกแบบระบบเป็นระดับบล็อก

รวมถึงการเลือกชนิดของไมโครโปรเซสเซอร์ กำหนดอินพุต (Input), เอาต์พุต (Output) และหน่วยความจำที่ต้องใช้ และแบ่งว่างานใดจะเป็นหน้าที่ของฮาร์ดแวร์หรือซอฟท์แวร์

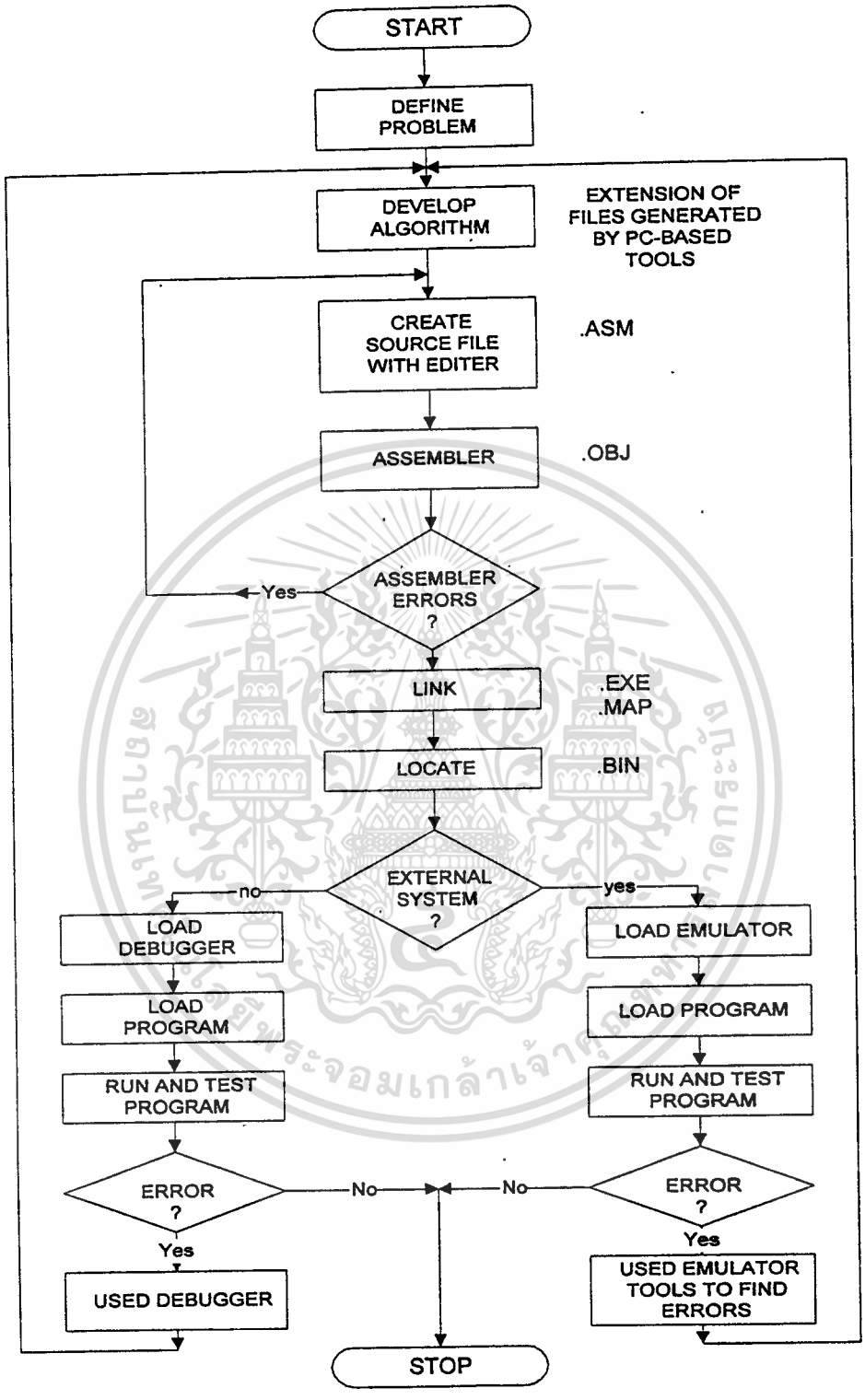
เมื่อออกแบบระดับบล็อกเสร็จแล้วการออกแบบจะแบ่งออกเป็น 2 ส่วนคือ การออกแบบฮาร์ดแวร์และการออกแบบซอฟท์แวร์ซึ่งโดยมากจะทำควบคู่กันไป

3. การออกแบบฮาร์ดแวร์

ประกอบด้วยการเลือกไมโครโปรเซสเซอร์ หน่วยความจำ อุปกรณ์รอบนอก (Peripheral Device) และออกแบบวงจรว่าจะต่อกันอย่างไรให้เป็นระบบที่ต้องการ

4. การสร้างเครื่องต้นแบบ

เราสามารถสร้างได้จากการใช้ไอซีมาต่อกันบนแผ่นวงจรเอนกประสงค์ ซึ่งเราเรียกว่าการสร้างเครื่องต้นแบบในระดับไอซี (IC) หรือระดับอุปกรณ์ ระบบไมโครโปรเซสเซอร์สามารถประกอบได้จากแผ่นวงจรพิมพ์มาตรฐาน เช่น แผงวงจรซีพียู (Central Processor Unit:CPU), แผงวงจรหน่วยความจำ, แผงวงจรอินพุตเอาต์พุต, ซึ่งต่อถึงกันด้วยบัสของระบบ การพัฒนาระบบไมโครโปรเซสเซอร์ด้วยวิธีนี้ทำให้การออกแบบง่ายขึ้นแต่เสียค่าใช้จ่ายมากกว่าวิธีปกติ เหมาะที่จะใช้กับระบบที่ผลิตจำนวนน้อยและไม่ต้องการอุปกรณ์ที่มีขนาดเล็ก เราเรียกอุปกรณ์ที่ใช้ในการออกแบบวิธีนี้ว่า ระบบพัฒนาไมโครโปรเซสเซอร์ระดับแผงวงจร



รูปที่ 1.1 อัลกอริทึมในการพัฒนาโปรแกรม

คอมพิวเตอร์แผ่นพิมพ์เดี่ยว (Single-Board Computer) ประกอบด้วยไมโครโปรเซสเซอร์ หน่วยความจำ, พอร์ตอินพุตเอาต์พุต ซึ่งสามารถนำไปประยุกต์ใช้เป็นฮาร์ดแวร์ต้นแบบได้โดยการต่ออุปกรณ์เพิ่มเติม

5. การทดสอบฮาร์ดแวร์ของเครื่องต้นแบบ

การแก้จุดบกพร่องของฮาร์ดแวร์มักจะเริ่มโดยการทดสอบระบบกับโปรแกรมสั้นๆ โดยโปรแกรมนี้อาจอยู่ในรอม (ROM) ที่เสียบลงไปในระบบหรืออาจใช้อีพรอมอีมูเลเตอร์ (EPROM Emulator) เพื่อให้สามารถแก้ไขได้ง่ายกรณีที่ใช้ระบบพัฒนาระบบพัฒนาไมโครโปรเซสเซอร์ระดับแผงวงจร หรือคอมพิวเตอร์แผ่นพิมพ์เดี่ยวเป็นเครื่องต้นแบบ เครื่องเหล่านี้มักมีวงจรและโปรแกรมควบคุมระบบ (Monitor Program) มาด้วย ซึ่งเราสามารถใช้ในการแก้จุดบกพร่องของฮาร์ดแวร์ที่ต่อเพิ่มเติมได้

6. การออกแบบซอฟท์แวร์

เมื่อออกแบบระดับบล็อกเสร็จและแบ่งว่างานใดจะเป็นหน้าที่ของซอฟท์แวร์แล้ว จะกำหนดข้อกำหนดรายละเอียดของซอฟท์แวร์ ออกแบบโครงสร้างของซอฟท์แวร์โดยแบ่งงานเป็นส่วนย่อยระดับต่างๆ จนสามารถเขียนคำสั่งโปรแกรมได้

7. การเขียนคำสั่งโปรแกรม

ซอฟท์แวร์อาจเขียนด้วยภาษาแอสเซมบลี (Assembly Language), ภาษาระดับสูง (High Language) หรือใช้ร่วมกันขึ้นอยู่กับชนิดของคอมพิวเตอร์ที่ใช้ในการพัฒนาซอฟท์แวร์ ซึ่งจะมีโปรแกรมอีดิตเตอร์ (Editor) ที่ช่วยในการเขียนแฟ้มข้อมูลภาษาคอมพิวเตอร์ที่ใช้ หลังจากนั้นซอฟท์แวร์ที่เขียนจะถูกแปลเป็นภาษาเครื่องด้วยแอสเซมเบลอร์ (Assembler)

สำหรับซอฟท์แวร์ที่เขียนด้วยภาษาแอสเซมบลีหรือถูกแปลเป็นภาษาเครื่อง ด้วยคอมไพเลอร์ สำหรับซอฟท์แวร์ที่เขียนด้วยภาษาระดับสูง ในการเขียนโปรแกรมสั้นๆ เราอาจใช้ในการแปลภาษาแอสเซมบลีที่เป็นรหัสภาษาเครื่อง แล้วป้อนให้เครื่องโปรแกรมอีพรอม หรือคอมพิวเตอร์แผ่นพิมพ์เดี่ยวได้โดยตรง

8. การทดสอบและแก้จุดบกพร่องของซอฟท์แวร์

อาจจะเป็นไปได้ที่จะแก้จุดบกพร่องของซอฟท์แวร์โดยไม่ต้องฮาร์ดแวร์ต้นแบบ โดยการใช้ดีบักเกอร์ (debugger) ที่จะนำซอฟท์แวร์มาทดลองทำงานในเครื่องคอมพิวเตอร์ที่ใช้ไมโครโปรเซสเซอร์เบอร์เดียวกันกับที่ใช้ในฮาร์ดแวร์ที่ออกแบบ โดยสามารถควบคุมและติดตามการทำงานของโปรแกรมได้ เช่น ระบบที่ใช้ไมโครโปรเซสเซอร์ 8080 ซึ่งสามารถทดสอบซอฟท์แวร์ได้โดยให้ทำงานในเครื่องที่ใช้ระบบปฏิบัติการ CPM-80 หรือระบบที่ใช้

ไมโครโปรเซสเซอร์ 8088 สามารถทดสอบได้ในเครื่องไมโครคอมพิวเตอร์ไอบีเอ็มพีซี โดยใช้โปรแกรม debug.com ซึ่งเราสามารถทดสอบซอฟท์แวร์ได้เฉพาะส่วนที่ไม่เกี่ยวข้องกับฮาร์ดแวร์

ในกรณีที่เครื่องไมโครคอมพิวเตอร์ ต้องใช้ไมโครโปรเซสเซอร์ต่างชนิดกันกับที่ใช้ในระบบที่ออกแบบเอาไว้ จะต้องใช้โปรแกรมซิมมูเลเตอร์ (Simulator) ซึ่งจะจำลองการทำงานของไมโครโปรเซสเซอร์ที่ใช้ในระบบให้ทำงานตามโปรแกรมที่ทดสอบ ซึ่งนอกจากจะทำงานเฉพาะส่วนโปรแกรมบางครั้งจะสามารถจำลองการทำงานของฮาร์ดแวร์บางส่วนได้ เช่น โปรแกรมจำลองการทำงานของ เครื่องไมโครคอมพิวเตอร์ไอบีเอ็มพีซี ให้เป็นเครื่องที่ใช้ระบบปฏิบัติการ CPM-80 ในกรณีที่ฮาร์ดแวร์ไม่ใช่อุปกรณ์มาตรฐาน โปรแกรมที่ใช้จำลองการทำงานโดยทั่วๆ ไปจะทดสอบไม่ได้

นอกจากนี้การออกแบบที่ใช้โปรแกรมซิมมูเลเตอร์จะไม่สามารถทดสอบโปรแกรมแบบเรียลไทม์ (Real Time) ได้ เนื่องจากโปรแกรมซิมมูเลเตอร์จะทำงานด้วยความเร็วต่ำกว่าการทำงานด้วยไมโครโปรเซสเซอร์จริงๆ มาก นอกจากนี้ยังสามารถใช้ดีบักเกอร์เพื่อตรวจสอบและแก้จุดบกพร่อง ที่มีในคอมพิวเตอร์แผ่นพิมพ์เดี่ยว ที่ใช้ไมโครโปรเซสเซอร์เบอร์เดียวกันกับที่ใช้ในฮาร์ดแวร์ที่ออกแบบใน การทดสอบซอฟท์แวร์ได้โดยการถ่ายข้อมูลจากเครื่องไมโครคอมพิวเตอร์ที่ใช้ในการแปลซอฟท์แวร์เป็นภาษาเครื่องไปยังเครื่องคอมพิวเตอร์แผ่นพิมพ์เดี่ยวได้ หรืออาจเขียนซอฟท์แวร์เป็นภาษาเครื่องโดยใช้แป้นพิมพ์และตัวแสดงผลที่มีในคอมพิวเตอร์แผ่นพิมพ์เดี่ยว

9. การรวบรวมฮาร์ดแวร์และซอฟท์แวร์

หลังจากฮาร์ดแวร์สามารถทำงานตามโปรแกรมทดสอบได้ ก็จะเริ่มขั้นตอนการรวมฮาร์ดแวร์และซอฟท์แวร์ด้วยกัน โปรแกรมทดสอบจะถูกแทนที่ด้วยซอฟท์แวร์ที่ใช้งานจริง และเริ่มการแก้จุดบกพร่องของระบบซึ่งมักจะพบปัญหาทั้งทางฮาร์ดแวร์และซอฟท์แวร์ ในขั้นตอนนี้มีอุปกรณ์ทดสอบหลายอย่าง เช่น อีพรอมอีมูเลเตอร์ ซึ่งก็คือ วงจรที่ถูกสร้างขึ้นมาเพื่อใช้งานแทนตัวอีพรอมหรือแรมจริงในแผงวงจร, อินเซอร์กิตอีมูเลเตอร์ (Incircuit Emulator) ที่เสียบเข้าไปในซ็อกเก็ตไมโครโปรเซสเซอร์ แล้วสามารถควบคุมและตรวจสอบการทำงานของระบบได้ และเครื่องทดสอบอีกชนิดที่มีประโยชน์ คือ ลอจิกอะนาไลเซอร์ (Logic Analyzer) ที่จะแสดงลำดับการทำงานของบัส

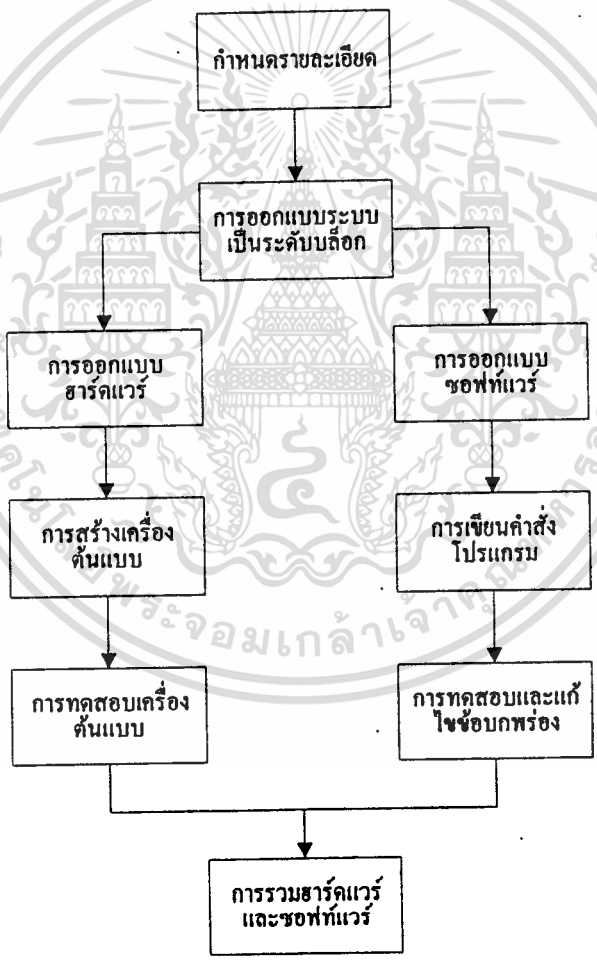
กรณีที่ใช้คอมพิวเตอร์แผ่นพิมพ์เดี่ยวที่ต่อฮาร์ดแวร์เพิ่มเติมเป็นเครื่องต้นแบบ เราสามารถใช้โปรแกรมควบคุมระบบช่วยให้เราควบคุมติดตามการทำงานของโปรแกรมซอฟท์แวร์ที่ใช้งานในระบบฮาร์ดแวร์ได้ แต่ความสามารถในการตรวจสอบการทำงานของระบบจะน้อย จากขั้นตอนการออกแบบระบบที่ใช้ไมโครโปรเซสเซอร์จะเห็นว่ามีขั้นตอนต่างๆ ที่ยุ่งยากซับซ้อนอยู่อย่างมาก เช่น ขั้นตอนในการแก้ไขและทดสอบซอฟท์แวร์ โดยใช้ดีบักเกอร์ซึ่งจะต้องใช้ฮาร์ดแวร์หรือทรัพยากรของดีบักเกอร์นั้นๆ จึงจะสามารถทำงานได้ เช่น ต้องมีโปรแกรมควบคุมระบบและมีหน่วยความจำให้เรียกใช้ตามตำแหน่งที่กำหนดและจะต้องมีพอร์ตอนุกรม (Serial Port) จึงจะทำการใช้ดีบักเกอร์ได้ จะเห็นว่าถ้าหากฮาร์ดแวร์ที่จะใช้ทำการดีบักเกอร์ไม่มีคุณสมบัติดังกล่าวแล้วจะไม่สามารถใช้งานดีบักเกอร์ได้

ในขั้นตอนการออกแบบซอฟท์แวร์ในส่วนของการทดสอบและแก้ไขจุดบกพร่องของซอฟท์แวร์นั้น มักมีปัญหาในส่วนของอีดิตเตอร์ ที่ไม่มีส่วนของแอสเซมเบอร์รวมอยู่ด้วย ทำให้ต้องออกจากอีดิตเตอร์เพื่อเข้าไปยังแอสเซมเบลอร์ ซึ่งในกรณีที่ซอฟท์แวร์ที่เขียนขึ้นไม่สามารถแปลเป็นภาษาเครื่องได้ ทำให้ต้องเสียเวลาในการเข้าไปในอีดิตเตอร์อีกเพื่อที่จะทำการแก้ไขในส่วนที่ผิดพลาด

ในขั้นตอนการรวมฮาร์ดแวร์และซอฟท์แวร์ซึ่งจะต้องมีการดาวน์โหลด (Download) ซอฟท์แวร์ลงไปยังระบบที่ต้องการนั้น มักพบปัญหาเกิดขึ้นมาก เนื่องมาจากอีดิตเตอร์ไม่มีส่วนที่ทำการดาวน์โหลด ตัวอย่างเช่น เมื่อทำการแอสเซมเบลอร์เสร็จแล้ว ต้องเข้าไปในส่วนของโปรแกรมที่ทำหน้าที่ดาวน์โหลด ซึ่งกรณีที่ฮาร์ดแวร์ไม่สามารถทำงานได้ตามที่คาดไว้ตามซอฟท์แวร์ ทำให้ต้องเข้าไปแก้ไขในส่วนของดีบักเกอร์หรืออีดิตเตอร์ ซึ่งเมื่อทำการดีบักโปรแกรมแล้วพบข้อผิดพลาดหรือข้อบกพร่อง ก็จะต้องเข้าไปที่อีดิตเตอร์เพื่อแก้ไขโปรแกรมที่ผิดพลาดหรือบกพร่อง แล้วก็ต้องออกจากอีดิตเตอร์อีกเพื่อที่จะทำการดาวน์โหลดซอฟท์แวร์ ซึ่งถ้าหากที่ฮาร์ดแวร์ไม่สามารถทำงานได้ตามที่คาดไว้ตามซอฟท์แวร์อีกก็ต้องทำตามขั้นตอนเดิมอีก ซึ่งเป็นการเสียเวลาอย่างมากที่จะต้องเข้าๆ ออกๆ โปรแกรมที่ทำหน้าที่ในส่วนต่างๆ เหล่านั้น

ซึ่งจากปัญหาที่ซ้ำซ้อนบางส่วนที่ยกตัวอย่างมานี้ ทำให้เกิดแนวคิดที่จะสร้างเครื่องจำลองหน่วยความจำและทดสอบโปรแกรมซีพียู MCS-51 (MEMORY EMULATOR AND MCS-51 DEBUGGER PROGRAM) ขึ้นมาเพื่อแก้ไขปัญหาดังกล่าวและมีเครื่องมือที่ช่วยอำนวยความสะดวกในการออกแบบระบบและพัฒนาโครงงานให้มีประสิทธิภาพยิ่งขึ้น ความสามารถของเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 มีดังนี้คือ มีการ

แสดงค่าของหน่วยความจำและรีจิสเตอร์ขณะกำลังรันโปรแกรมอยู่, สามารถทำการดีบักได้, สามารถรันแบบซิงเกิลเสต็ปได้ (Single Step) , สามารถตั้งจุดเบรคได้ 10 ตำแหน่ง (เมื่อแผงวงจรทดลองที่ใช้เป็นไมโครโปรเซสเซอร์ในตระกูล MCS-51) และยังสามารถทำงานเป็นอีพรอมอีมูเลเตอร์หรือแรมอีมูเลเตอร์ได้กับไมโครโปรเซสเซอร์ทุกเบอร์ ซึ่งสามารถทำได้โดยการสั่งจากเมนูในโปรแกรม MEMDP และในโปรแกรม MEMDP นั้นยังมีเมนูคำสั่งต่างๆ ที่ช่วยให้ผู้ใช้เกิดความสะดวก เช่น เมนูในส่วนของอีดิตเตอร์, เมนูในส่วนของการแอสเซมเบลอร์ เป็นต้น



รูปที่ 1.2 ขั้นตอนการออกแบบระบบที่ใช้ไมโครโปรเซสเซอร์

บทที่ 2
ทฤษฎีและหลักการ

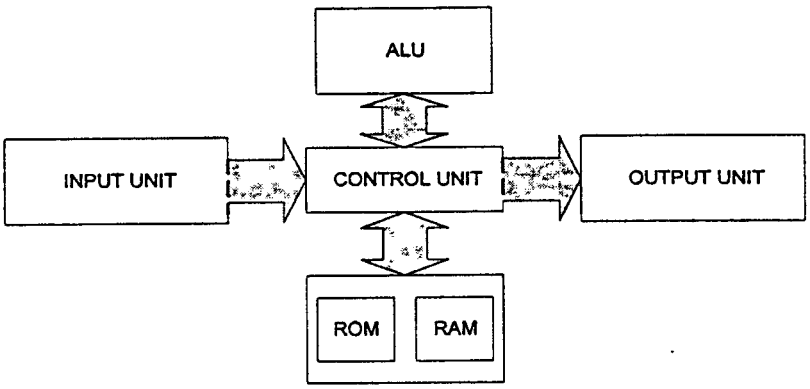
2.1 ไมโครโปรเซสเซอร์

ไมโครโปรเซสเซอร์เป็นวิวัฒนาการทางเทคโนโลยีที่เด่นที่สุดในรอบ 30 ปี โดยกำเนิดเมื่อปีพ.ศ. 2491 เป็นต้นมา ไมโครโปรเซสเซอร์ไม่เพียงแต่จะทำให้เกิดการเปลี่ยนแปลงแนวความคิดที่ยิ่งใหญ่ของวงจรอิเล็กทรอนิกส์เชิงเลข (Digital Electronics) เท่านั้นแต่มีผลกระทบต่อชีวิตความเป็นอยู่ของมนุษย์ ไมโครโปรเซสเซอร์ตัวแรกถูกสร้างขึ้นเมื่อปีพ.ศ. 2514 โดยบริษัทอินเทลคอร์พอเรชั่น (Intel Corporation) ซึ่งจัดเป็นไมโครโปรเซสเซอร์ขนาด 4 บิต ชื่อว่าอินเทล 4004 มีจุดประสงค์ในการสร้างใช้ในเครื่องคำนวณอิเล็กทรอนิกส์ขนาดเล็ก ต่อมามีบริษัทผู้ผลิตอุปกรณ์สารกึ่งตัวนำหลายบริษัททำการผลิตไมโครโปรเซสเซอร์ชนิดต่างๆ ขึ้นมาอีกเป็นจำนวนมาก โดยผู้ผลิตส่วนใหญ่ใช้เทคโนโลยีการผลิต แอลเอสไอ (Large Scale Integration:LSI)

ไมโครโปรเซสเซอร์ทำหน้าที่เป็นหน่วยประมวลผลกลาง (Central Processing Unit:CPU) ซีพียูเดิมทีนั้นทำจากไอซีจำนวนมากประกอบกัน แต่ไมโครโปรเซสเซอร์ได้รวมไอซีจำนวนมากนั้นบรรจุลงในชิ้นส่วนเดียวกัน (1 Chip) ซึ่งทำให้ประหยัดเนื้อที่และสะดวกในการใช้งานจึงเป็นที่นิยมใช้งานกันในปัจจุบัน

2.1.1 หลักการทำงานของไมโครโปรเซสเซอร์

หลักการทำงานของระบบไมโครโปรเซสเซอร์ภายในประกอบด้วยส่วนประกอบพื้นฐานอยู่ 5 หน่วยคือ หน่วยซีพียูจะรวมวงจรควบคุม (Control Unit) และวงจรคำนวณทางคณิตศาสตร์และลอจิก (Arithmetic Logic Unit:ALU) เข้าไว้ด้วยกัน, หน่วยความจำ (Memory Unit), หน่วยอินพุต (Input Unit) และหน่วยเอาต์พุต (Output Unit) สามารถแบ่งเป็นหน่วยใหญ่ๆ ได้ดังรูปที่ 2.1



รูปที่ 2.1 ผังการทำงานไมโครโปรเซสเซอร์เบื้องต้น

จะทำหน้าที่ป้อนข้อมูลในการคำนวณทางคณิตศาสตร์และคำนวณทางลอจิกให้กับหน่วย ALU ตามรูปที่ 2.1

หน่วยควบคุม จะทำหน้าที่ควบคุมการทำงานของอุปกรณ์ทุกอย่างในระบบ,จะผลิตสัญญาณที่ใช้ในการซิงโครไนซ์ (Synchronize), ควบคุมการทำงานและการส่งถ่ายข้อมูลเข้าออก ALU และภายนอกซีพียูโดยควบคุมผ่านบัสแอดเดรสและบัสข้อมูล และจะทำการแปลความหมายของสัญญาณต่างๆ บนบัสควบคุมซึ่งได้รับจากวงจรภายนอกซีพียู หน้าที่สำคัญของวงจรควบคุมคือ การอ่านคำสั่งที่เก็บไว้ในหน่วยความจำมาถอดรหัส (Decode) และการทำงานจะเป็นจังหวะที่ซ้ำกัน คือ จังหวะในการอ่านคำสั่งเฟทช์ (Fetch) จังหวะถอดรหัสของคำสั่งและจังหวะทำงานตามคำสั่งเอ็กซีคิว

หน่วยความจำ เป็นหน่วยข้อมูลและเก็บคำสั่ง หน่วยความจำแบ่งออกง่ายๆ เป็น 2 ชนิด ได้แก่ รอม (Read Only Memory:ROM) คือ หน่วยความจำที่สามารถอ่านข้อมูลได้อย่างเดียว และแรม (Random Access Memory:RAM) คือ หน่วยความจำที่ใช้ได้ทั้งการอ่านข้อมูลและเขียนข้อมูล แรมจะเป็นที่เก็บข้อมูล 2 ชนิด คือ คำสั่งและข้อมูลหน่วยความจำทั้ง 2 ชนิด เป็นหน่วยความจำที่ทำจากสารกึ่งตัวนำ (IC memory)

หน่วยอินพุต ทำหน้าที่รับข้อมูลที่ป้อนเข้าสู่ซีพียู ตัวอย่างของหน่วยอินพุต ได้แก่ แป้นกดข้อมูล (Keyboard) หรือตัวตรวจวัด เช่น ตัววัดอุณหภูมิ, ตัววัดแรงดัน เป็นต้น

หน่วยเอาต์พุต ทำหน้าที่ส่งข้อมูลภายในซีพียู ออกแสดงภายนอก ตัวอย่างของหน่วยเอาต์พุต ได้แก่ ไดโอดเปล่งแสง (Light Emitting Diode:LED) เป็นต้น

หน่วยอินพุตเอาต์พุต ตามปกติจะไม่ต่อโดยตรงกับ ALU แต่จะต่อกับซีพียู โดยผ่านทางบัสแอดเดรส, บัสข้อมูลและบัสควบคุม การเขียนและอ่านข้อมูล ที่หน่วยอินพุตและเอาต์พุตมักจะกระทำผ่านซีพียู โดยข้อมูลจะส่งถ่ายจากหน่วยความจำในซีพียูไปยังหน่วยเอาต์พุต หรือหน่วยอินพุตจะส่งข้อมูลมายังซีพียูหรือหน่วยความจำ

2.1.2 โครงสร้างภายในของไมโครโปรเซสเซอร์

ไมโครโปรเซสเซอร์ที่จะกล่าวถึงนี้เป็นไมโครโปรเซสเซอร์ในตระกูล MCS-51 คือ เบอร์ 8951 ซึ่งใช้ในงานระบบควบคุมที่มีชื่อเรียกอีกอย่างว่า ไมโครคอนโทรลเลอร์ (Microcontroller) เนื่องจากไมโครคอนโทรลเลอร์มีความสามารถและคุณสมบัติต่างๆ เหนือกว่าไมโครโปรเซสเซอร์ปกติ ส่วนประกอบที่สำคัญของไมโครคอนโทรลเลอร์ ได้แก่ ALU, หน่วยความจำสำหรับเก็บโปรแกรม, หน่วยความจำสำหรับเก็บข้อมูล, รีจิสเตอร์ใช้งานเฉพาะ, รีจิสเตอร์ใช้งานทั่วไป, พอร์ตอินพุตเอาต์พุต, ไทม์เมอร์เคาน์เตอร์ (Timer Counter) และพอร์ตสื่อสารข้อมูลแบบอนุกรม ซึ่งโครงสร้างภายในของไมโครโปรเซสเซอร์แสดงดังรูปที่ 2.2

ALU

ทำหน้าที่คำนวณในไมโครคอนโทรลเลอร์ โดยจะทำการคำนวณและประมวลผลทางคณิตศาสตร์และทางลอจิก โดยผ่านทางรีจิสเตอร์พิเศษ 2 ตัวที่มีชื่อเรียกว่า TMP2 (Temporally 2)และ TMP1 (Temporally 2) และจะส่งข้อมูลจาก ALU ไปให้แก่ รีจิสเตอร์ที่มีชื่อว่า PSW (Program Status Word) และส่งไปที่บัสภายใน ดังรูปที่ 2.2

หน่วยความจำสำหรับเก็บโปรแกรม

หน่วยความสำหรับเก็บโปรแกรมในไมโครคอนโทรลเลอร์ MCS-51 จะถูกแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำสำหรับเก็บโปรแกรมภายในชิป (Internal Program Memory) และหน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิป (External Program Memory) ขนาดของหน่วยความจำสำหรับเก็บโปรแกรมภายในชิปมีตั้งแต่ 0, 4, 8, 16 กิโลไบต์ ขึ้นอยู่กับเบอร์ของชิป

หน่วยความจำสำหรับเก็บข้อมูล

หน่วยความสำหรับเก็บข้อมูลในไมโครคอนโทรลเลอร์ MCS-51 จะถูกแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำสำหรับเก็บข้อมูลภายในชิปและหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป ซึ่งยังแบ่งออกเป็น 2 ส่วนย่อย คือ

- ส่วนที่ใช้เก็บข้อมูลทั่วไป (Internal Ram)

เป็นหน่วยความจำสำหรับเก็บข้อมูลที่มีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 ขณะกำลังทำงาน

- ส่วนที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ (Special Function Register:SFR)

เป็นหน่วยความจำสำหรับเก็บข้อมูลที่มีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 ซึ่งถูกกำหนดให้เป็นรีจิสเตอร์ใช้งานเฉพาะเพื่อควบคุมการทำงานและบอกสถานะของซีพียู

ซึ่งไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีหน่วยความจำสำหรับเก็บข้อมูล ทั่วไปภายในชิปอย่างน้อย 128 ไบต์ ไปจนถึง 256 ไบต์ ทั้งนี้ขึ้นอยู่กับเบอร์ของชิป หน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปบริเวณ 128 ไบต์แรกมีชื่อเรียกว่า Lower 128 และในบริเวณ 128 ไบต์หลังที่มีเพิ่มบางเบอร์มีชื่อเรียกว่า Upper 128

รีจิสเตอร์ใช้งานเฉพาะ

รีจิสเตอร์ใช้งานเฉพาะทั้งหมดจะอยู่ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปในส่วนของรีจิสเตอร์ใช้งานเฉพาะ รีจิสเตอร์ใช้งานเฉพาะแสดงในรูปที่ 2.2

F8								FF
F0	B							F7
E8								EF
E0	ACC							E7
D8								D7
D0	PSW							CF
C8								C7
C0								CF
B8	IP							BF
B0	P3							B7
A8	IE							AF
A0	P2							A7
98	SCON	SBUF						9F
90	P1							97
88	TCON	TMOD	TL0	TL1	TH0	TH1		8F
80	P0	SP	DPL	DPH				PCON 87

รูปที่ 2.2 โครงสร้างและตำแหน่งของรีจิสเตอร์ใช้งานเฉพาะ

รีจิสเตอร์สำหรับใช้งานทั่วไป

ไมโครคอนโทรลเลอร์ MCS-51 มีรีจิสเตอร์ใช้งานทั่วไปที่ผู้ใช้สามารถนำมาใช้งานได้ คือ รีจิสเตอร์ A,B (อยู่ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะแต่นับเป็นรีจิสเตอร์ใช้งานทั่วไป เพราะไม่ได้ถูกกำหนดหน้าที่ใช้งานโดยตรง) และรีจิสเตอร์ใช้งานทั่วไป R0-R7 ซึ่งอยู่ในหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในบริเวณ 128 ไบต์แรก โดยมีอยู่ด้วยกันทั้งหมด 4 กลุ่ม แต่ละกลุ่มประกอบด้วยรีจิสเตอร์จำนวน 8 ตัว (R0-R7) ซึ่งมีชื่อเรียกเหมือนกัน

ดังนั้นจำนวนรีจิสเตอร์ใช้งานทั่วไป R0-R7 จึงมีทั้งหมด 32 ตัว ในการทำงานขณะใดๆ รีจิสเตอร์ทั้ง 4 กลุ่ม จะถูกใช้งานเพียงกลุ่มเดียวเท่านั้น การเลือกใช้รีจิสเตอร์ R0-R7 กลุ่มใดกลุ่มหนึ่งใน 4 กลุ่มกระทำโดยการเซตหรือเคลียร์บิต RS0 RS1 ภายในรีจิสเตอร์ใช้งานเฉพาะ PSW

พอร์ตอินพุตเอาต์พุต

ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีพอร์ตขนาด 8 บิตจำนวน 4 พอร์ต (P0-P3) โดยสามารถกำหนดให้ทำงานแบบพอร์ตขนานขนาด 8 บิต 4 พอร์ต หรือจะใช้เป็นพอร์ตขนาด 1 บิตได้ถึง 32 พอร์ต ทั้งนี้ผู้ใช้ยังสามารถทำการกำหนดให้แต่ละพอร์ตใช้งานเป็นอินพุตหรือเอาต์พุตพอร์ตอย่างใดอย่างหนึ่งได้อย่างอิสระ โดยการควบคุมจากโปรแกรม

ไทม์เมอร์เคาน์เตอร์

ไมโครคอนโทรลเลอร์ MCS-51 มีรีจิสเตอร์ใช้งานเฉพาะที่สามารถนับจำนวนสัญญาณนาฬิกาหรือแมชชีนไซเคิลของวงจรออสซิลเลเตอร์ภายใน (ทำงานเป็นไทม์เมอร์) หรือนับจำนวนครั้งของการเปลี่ยนสถานะของสัญญาณภายนอก (นับจำนวนพัลส์ภายนอก) ที่ขาของพอร์ต 3 (ทำงานเป็นเคาน์เตอร์) รีจิสเตอร์ที่ใช้เป็นไทม์เมอร์หรือเคาน์เตอร์มีขนาด 16 บิต จำนวน 2 ตัว คือ รีจิสเตอร์ไทม์เมอร์ 0 และรีจิสเตอร์ไทม์เมอร์ 1 ตามลำดับ

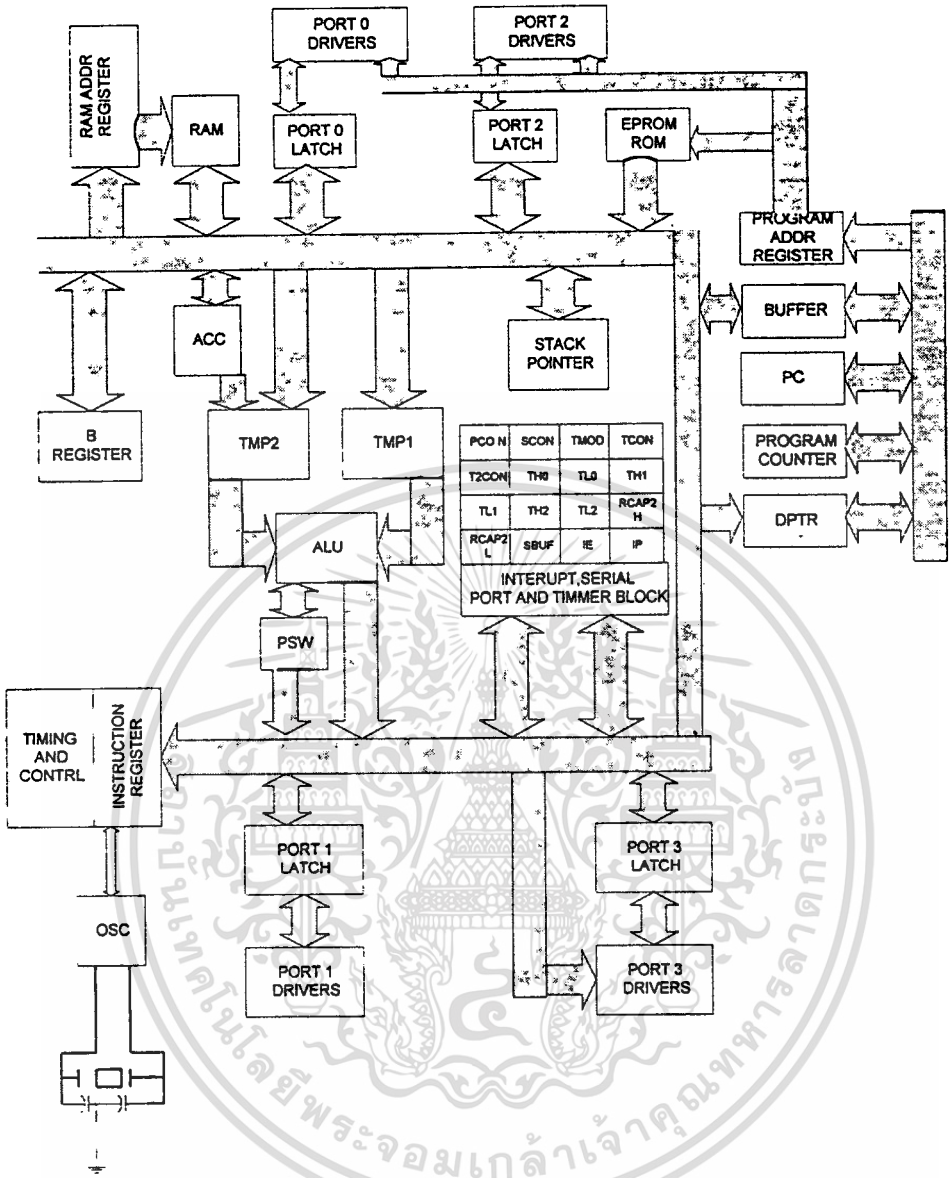
เมื่อต้องการใช้ไทม์เมอร์ 0 หรือไทม์เมอร์ 1 จะต้องโหลดค่าที่ต้องการนับไปไว้ภายในรีจิสเตอร์ไทม์เมอร์ 0 หรือไทม์เมอร์ 1 และเมื่อนับได้ครบตามจำนวนที่ตั้งไว้ จะมีสัญญาณอินเตอร์รัพท์เพื่อให้ซีพียูทราบ

พอร์ตสื่อสารข้อมูลแบบอนุกรม

ไมโครคอนโทรลเลอร์ MCS-51 สามารถรับและส่งข้อมูลแบบอนุกรมได้โดยไม่ต้องพึ่งอุปกรณ์ภายนอกอื่นๆ แต่อย่างใดในด้านอัตราเร็วของการรับส่งข้อมูลก็สามารถกำหนดค่าได้ตามความต้องการของผู้ใช้ โดยสามารถเลือกอัตราเร็วในการรับส่งข้อมูล (Buard rate) มาตรฐานได้ตั้งแต่ 110, 1.2K, 2.4K, 4.8K, 9.6K, 19.2K, 375K ตามมาตรฐานของ UART (Universal Asynchronous Recieve and Transmitt) นอกจากนี้ยังสามารถกำหนดการทำงานที่แตกต่างกันได้ถึง 4 รูปแบบ ตามความเหมาะสมในการใช้งาน

2.1.3 การใช้งานไมโครโปรเซสเซอร์

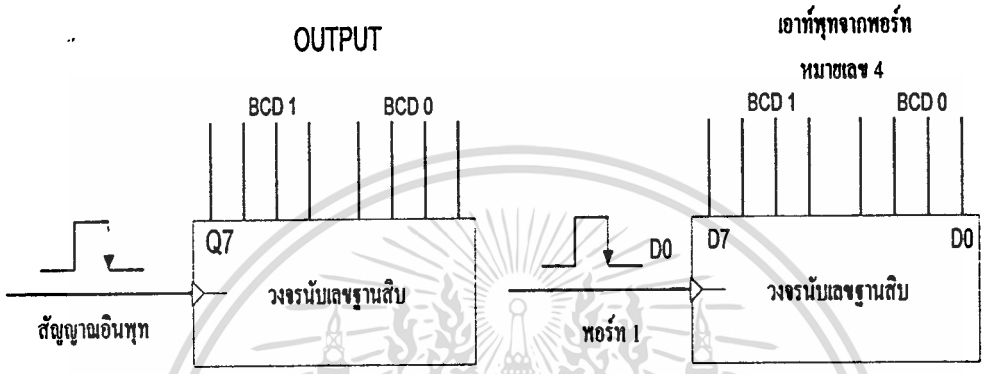
การใช้งานไมโครโปรเซสเซอร์สามารถแบ่งออกได้เป็น 2 ประเภท คือ การใช้ไมโครโปรเซสเซอร์เพื่อทำงานแทนมินิคอมพิวเตอร์ในการคำนวณหรือควบคุมและการใช้ไมโครโปรเซสเซอร์แทนวงจรดิจิตอล เช่น เครื่องวัดและเครื่องควบคุมอิเล็กทรอนิกส์ เป็นต้น



รูปที่ 2.3 โครงสร้างภายในของไมโครโปรเซสเซอร์ตระกูล MCS-51

ในส่วนนี้จะกล่าวถึงการใช้ไมโครโปรเซสเซอร์แทนวงจรดิจิตอล ซึ่งนับเป็นการปฏิวัติแนวความคิดของการออกแบบวงจรดิจิตอลเพราะระบบดิจิตอลที่ซับซ้อนมากๆ สามารถแทนด้วยไมโครโปรเซสเซอร์ และ IC เพียงไม่กี่ตัว โดยสามารถเพิ่มหน้าที่การทำงานของเครื่องได้ไม่สิ้นสุด

ในที่นี้จะแสดงตัวอย่างการใช้ไมโครโปรเซสเซอร์แทนวงจรนับเลขฐาน 10 โดยวงจรนับเลขฐาน 10 คือ วงจรนับแบบหนึ่งที่ให้เอาต์พุตออกมาเป็นรหัส BCD (Binary Code Decimal) และได้รับสัญญาณพัลส์ที่ขาอินพุต ซึ่งวงจรนี้แสดงได้ดังบล็อกไดอะแกรมในรูปที่ 2.4 (ก) และระบบที่ใช้ไมโครโปรเซสเซอร์แทนได้แสดงในรูปที่ 2.4 (ข)

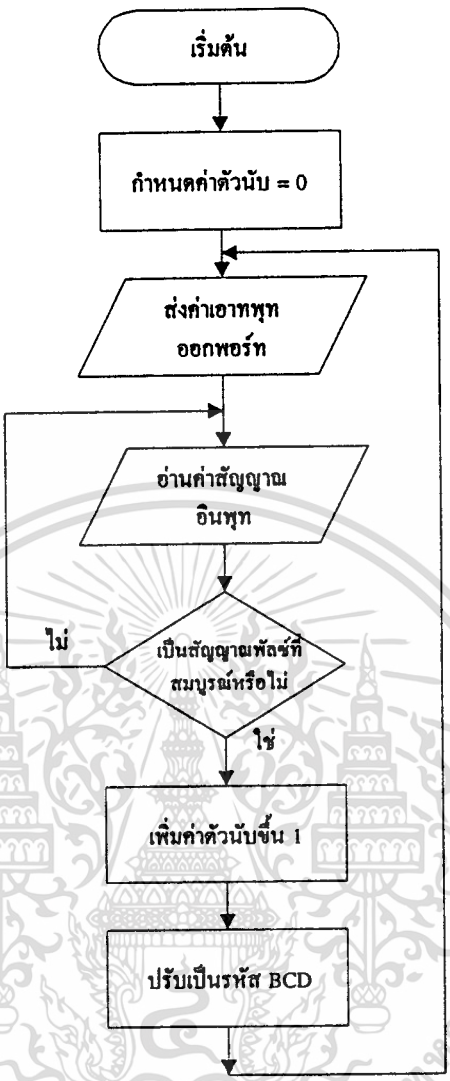


รูปที่ 2.4 ก และ ข การใช้ไมโครโปรเซสเซอร์แทนวงจรนับเลขฐาน 10

จากผังการทำงาน สัญญาณอินพุตต้องการให้เป็นแบบสัญญาณพัลส์ที่สมบูรณ์ คือมีทั้งขอบขาขึ้นและขอบขาลง และกำหนดให้วงจรนับแอคทีฟที่ของขาลง เมื่อมีพัลส์ครบ 1 ลูก เอาต์พุตจะเพิ่มขึ้น 1 ค่าและเพิ่มขึ้นตามลักษณะของเลขฐาน 10 สัญญาณอินพุตกำหนดให้เข้าที่บิต 0 ของแผงวงจรอินพุตหมายเลข 1 และสัญญาณเอาต์พุต กำหนดให้ออกที่พอร์ตเอาต์พุตหมายเลข 4 ซึ่งมีผังงานของการทำงานดังรูปที่ 2.5

2.2 ระบบหน่วยความจำ

ระบบหน่วยความจำในไมโครโปรเซสเซอร์จะแบ่งออกเป็น 2 ชนิด คือ หน่วยความจำถาวร และหน่วยความจำชั่วคราวที่สามารถเปลี่ยนแปลงข้อมูลได้โดยง่าย



รูปที่ 2.5 ผังการทำงานของการใช้ไมโครโปรเซสเซอร์แทนวงจรนับเลขฐาน 10

2.2.1 หน่วยความจำถาวร

หน่วยความจำชนิดนี้ก็คือ รอม ข้อมูลที่อยู่ในรอมสามารถอ่านออกมาได้แต่ไม่สามารถเขียนข้อมูลเข้าไปในรอมได้อีก ประโยชน์ของรอมในระบบไมโครโปรเซสเซอร์คือ ทำให้ซีพียูของระบบสามารถที่จะเริ่มทำงานโดยการกำหนดสภาวะของอุปกรณ์ฮาร์ดแวร์ต่างๆ ให้อยู่ใน สภาวะที่พร้อมจะทำงานได้เมื่อเริ่มจ่ายไฟเลี้ยงให้แก่ระบบ หน่วยความจำถาวรที่เก็บข้อมูลไว้โดยไม่สูญหาย มีหลายชนิด เช่น

รอม ข้อมูลทั้งหมดที่อยู่ในรอม จะถูกโปรแกรมไว้อย่างถาวรในระหว่างกระบวนการผลิตโดยการทำมาส์ค (Mask) เฉพาะสำหรับงานนั้นๆ (โปรแกรมมาจากโรงงาน)

คุณสมบัติของรอมก็คือ เก็บข้อมูลได้ตลอดกาลไม่ว่าจะมีไฟเลี้ยงในวงจรหรือไม่ก็ตาม และไม่สามารถเปลี่ยนแปลงข้อมูลในรอมได้ เราจะใช้รอมเมื่อต้องใช้ข้อมูลที่ไม่เปลี่ยนแปลง และมีความต้องการใช้งานเป็นจำนวนมาก

พรอม (Programmable Read Only Memory) ข้อมูลที่ต้องการโปรแกรมจะถูกโปรแกรมโดยผู้ใช้เอง โดยป้อนพัลส์ที่มีแรงดันสูง ทำให้ Metal Strips หรือ Polycrystaline Silicon ที่มีอยู่ในตัว IC ขาดออกจากกัน ทำให้เกิดเป็นลอจิก 1 หรือ 0 ตามตำแหน่งที่กำหนดในหน่วยความจำนั้นๆ

คุณสมบัติของพรอมก็คือ เก็บข้อมูลได้ตลอดกาลไม่ว่าจะมีไฟเลี้ยงในวงจรหรือไม่ก็ตามและสามารถโปรแกรมข้อมูลได้เอง 1 ครั้งเท่านั้นหลังจากนั้นจะไม่สามารถโปรแกรมได้อีก

อีพรอม (Erasable Programmable ROM) ข้อมูลจะถูกโปรแกรมโดยผู้ใช้โดยการให้สัญญาณที่มีแรงดันสูงผ่านเข้าไปในตัวอีพรอม ซึ่งเป็นวิธีเดียวกับที่ใช้ในพรอมเพียงแต่ข้อมูลที่อยู่ในตัวอีพรอมสามารถเปลี่ยนแปลงได้ โดยการลบข้อมูลเดิมที่อยู่ในอีพรอมโดยการฉายแสงอุลตร้าไวโอเลต (Utra Violet) เข้าไปในตัวอีพรอมโดยผ่านทางกระจกใสที่อยู่บนตัวอีพรอม ซึ่งช่วงเวลาที่ฉายแสงนี้ขึ้นอยู่กับคุณสมบัติของตัวอีพรอมซึ่งสามารถดูได้จากข้อมูลของตัวอีพรอม

คุณสมบัติของอีพรอมคือ เก็บข้อมูลได้ตลอดกาลไม่ว่าจะมีไฟเลี้ยงในวงจรหรือไม่ก็ตามและสามารถโปรแกรมข้อมูลได้หลายครั้ง

อีเอพรอม (Electrically Alterable ROM) ข้อมูลจะถูกโปรแกรมโดยผู้ใช้เหมือนกับอีพรอม แต่ที่แตกต่างก็คือข้อมูลของอีเอพรอมสามารถลบได้โดยทางไฟฟ้าไม่ใช่จากการฉายแสงเหมือนอีพรอม

คุณสมบัติของอีเอพรอมเหมือนกับอีพรอมทุกอย่าง

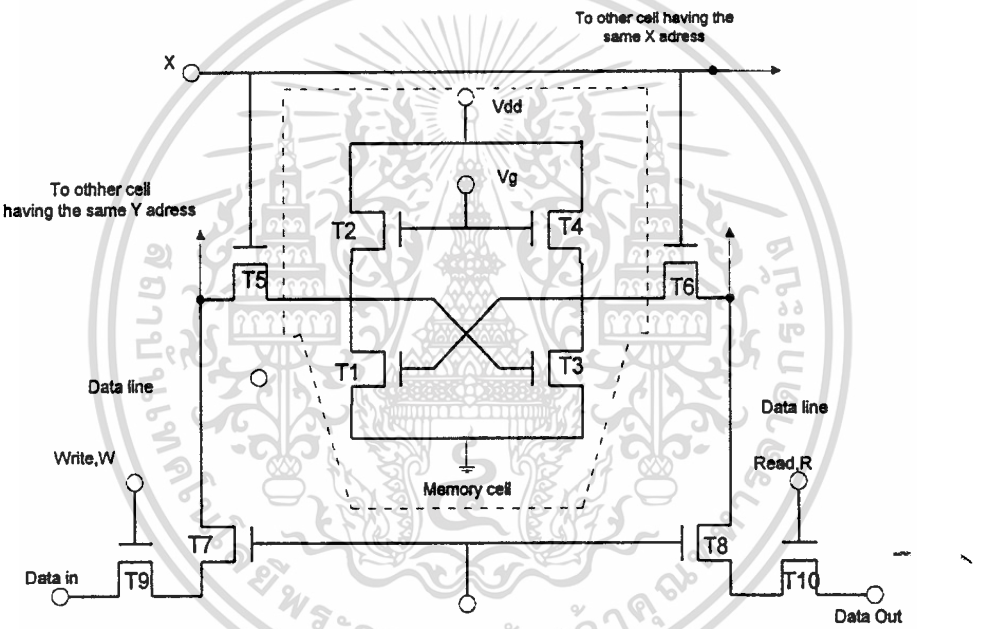
หน่วยความจำที่ได้กล่าวมาแล้วนี้เป็นหน่วยความจำที่ข้อมูลไม่สูญหายเมื่อไม่มีไฟเลี้ยงและสามารถอ่านข้อมูลมาใช้ได้เพียงอย่างเดียวเท่านั้น ซึ่งคุณสมบัติการใช้งานจะเหมือนกันขึ้นอยู่กับระบบว่าเหมาะสมใช้กับชนิดไหน เช่น ในการพัฒนาระบบไมโครโปรเซสเซอร์ซึ่งมีการแก้ไขโปรแกรมบ่อยๆ ควรใช้อีพรอม เป็นต้น

2.2.2 หน่วยความจำชั่วคราว

หน่วยความจำชนิดนี้ก็คือ แรม ซึ่งเป็นหน่วยความจำที่ต่างไปจากหน่วยความจำถาวร ทั้งในโครงสร้างและการทำงาน เราสามารถเขียนหรืออ่านข้อมูลจากแรมโดยใช้เวลาพอๆ กัน แรมทำจากสิ่งประดิษฐ์จำพวกบีเจทีหรือมอสเฟท แต่แรมที่ทำจากบีเจทีมักมีความเร็วสูงกว่าแรมที่ทำจากมอสเฟท หน่วยความจำของแรมมี 2 ชนิด คือ

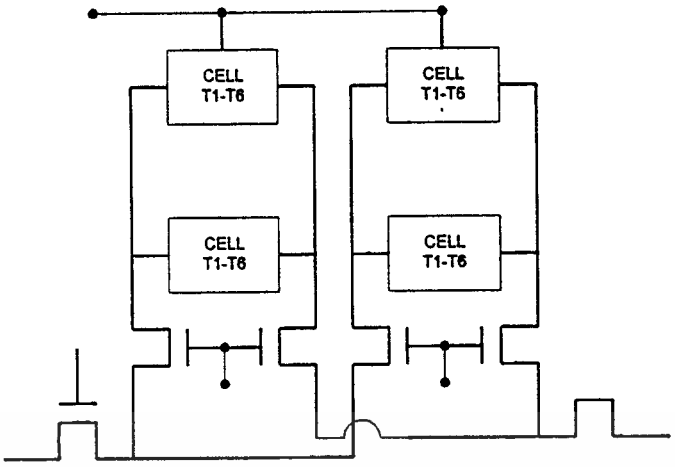
แรมชนิดสเตติก (Static RAM)

แรมชนิดนี้ใช้ลักษณะวงจรฟลิปฟลอปเป็นวงจรพื้นฐาน สำหรับเซลหน่วยความจำแต่ละเซลตามปกติวงจรพื้นฐานมักไม่แตกต่างกันและโดยมากจะเป็นดังรูปที่ 2.6



รูปที่ 2.6 เซลพื้นฐานของแรมชนิดสเตติก

T1-T6 คือ ทรานซิสเตอร์ที่ประกอบขึ้นเป็นวงจรฟลิปฟลอปในเซลพื้นฐาน ส่วน T7-T10 เป็นเกทที่เปิดให้ข้อมูลไหลผ่านไปมาได้ เป็นวงจรร่วมของเซลหน่วยความจำหลายๆ เซล ดังนั้นเราจึงเรียกเซลหน่วยความจำนี้ว่า เซลหน่วยความจำชนิดสเตติกแบบทรานซิสเตอร์ 6 ตัว ซึ่งลักษณะการทำงานเป็นดังนี้ T1,T2 และ T3,T4 ประกอบกันเป็นวงจรกลับเฟสที่ต่อโยงไขว้กัน



รูปที่ 2.7 การต่อเซลต่างๆ เข้าด้วยกันภายในตัว IC

กันอยู่ ทำหน้าที่เป็นฟลิปฟลอปโดย T5,T6 เป็นเกทเปิดหรือปิดตามแต่จะเลือกด้วยระดับแรงดันในสาย Y T9,T10 เลือกว่าจะให้เขียนข้อมูลเข้าไปหรืออ่านข้อมูลออกมา เมื่อต้องการเลือกเซลใดก็ให้สาย X,Y ของเซลนั้นเป็นลอจิก 1 ซึ่งจะทำให้สายข้อมูลต่อกับเซลได้ และยังต่อออกมาที่สาย Data In หรือสาย Data Out ก็ได้ด้วยการให้ W หรือ R เป็นลอจิก 1 ตามลำดับ

แรมชนิดไดนามิก (Dinamic RAM)

แรมชนิดนี้ใช้การคายประจุในตัวเก็บประจุที่ขาเกทของมอสเฟท เป็นวงจรพื้นฐานวงจรพื้นฐานของแรมชนิดไดนามิกมีได้หลายรูปแบบและใช้จำนวนทรานซิสเตอร์ต่างกันไป แรมชนิดไดนามิกมีข้อดีเหนือกว่าแรมสเตติก คือ มีเวลาเข้าถึงหน่วยความจำน้อยกว่า แต่แรมชนิดไดนามิกก็ต้องเสียเวลาในการรีเฟรช (Refresh) ช่วยความจำ

มีการสิ้นเปลืองกำลังไฟฟ้าต่ำกว่าแรมชนิดสเตติกเนื่องจากมีกระแสไหลผ่านวงจรขณะเขียนหรืออ่านรีเฟรชเท่านั้นและมีจำนวนทรานซิสเตอร์ต่อเซลน้อยกว่าทำให้พื้นที่ 1 ตารางหน่วยสามารถบรรจุแรมชนิดไดนามิกได้จำนวนเซลมากกว่าแรมชนิดสเตติก

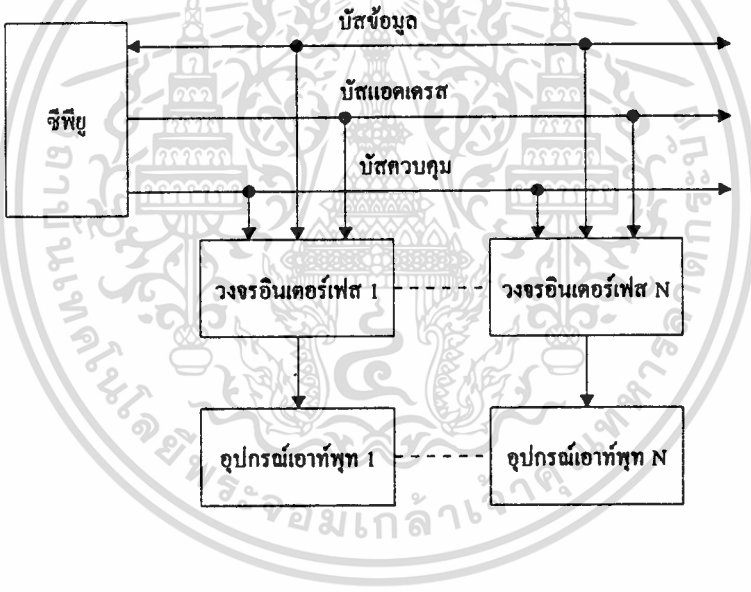
2.3 หน่วยอินพุตและเอาต์พุต

การทำงานของระบบไมโครโปรเซสเซอร์ ซีพียูจะมีการติดต่อกับระบบภายนอกอยู่เสมอ การติดต่อที่ทำอยู่ตลอดเวลาคือ ขบวนการเขียนและอ่านหน่วยความจำ นอกเหนือไป

จากหน่วยความจำแล้ว ซีพียูยังต้องติดต่อกับระบบอินพุตและเอาต์พุต ซึ่งเป็นการส่งและรับข้อมูลจากภายนอก

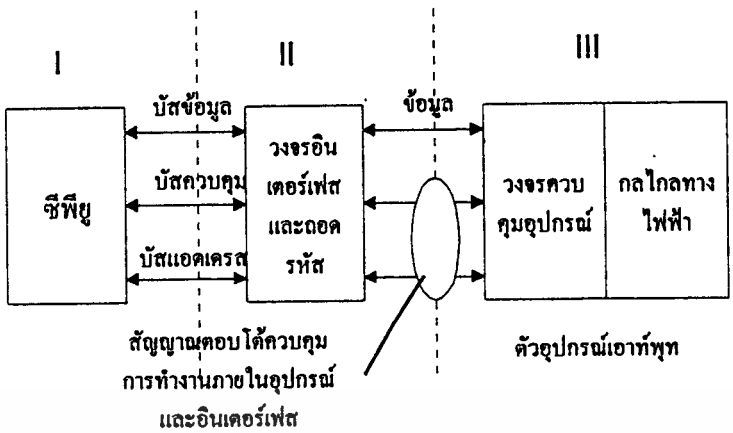
2.3.1 ขบวนการเอาต์พุตและอุปกรณ์เอาต์พุต

ขบวนการเอาต์พุต คือ ขบวนการส่งข้อมูลจากภายในตัวไมโครโปรเซสเซอร์ออกมาทางบัสข้อมูลไปยังอุปกรณ์อินพุตเอาต์พุตซึ่งทำหน้าที่เป็นตัวรับข้อมูลนั้น ซึ่งในระบบที่ใช้ไมโครโปรเซสเซอร์ ซีพียูต้องส่งสัญญาณเลือกอุปกรณ์เอาต์พุตตัวใดตัวหนึ่งพร้อมๆ กันกับข้อมูลออกมาทางบัสข้อมูล พร้อมกับสัญญาณที่จำเป็นในการทำงาน



รูป 2.8 ซีพียูติดต่อกับอุปกรณ์เอาต์พุตตัวใดตัวหนึ่งใน n อุปกรณ์

ในการติดต่อแต่ละครั้งจะมีอุปกรณ์และวงจรต่างๆ เกี่ยวข้องกันดังต่อไปนี้ ส่วนที่ I คือ ซีพียู ส่วนที่ II คือวงจรอินเตอร์เฟสและถอดรหัส ส่วนที่ III คือ ตัวอุปกรณ์เอาต์พุต ส่วนในที่ III ซึ่งเป็นอุปกรณ์เอาต์พุตทั้งหมดสามารถแบ่งได้เป็น 2 ส่วน คือ A ซึ่งคือวงจรอิเล็กทรอนิกส์สำหรับควบคุมการทำงานภายในอุปกรณ์ ทำหน้าที่รับข้อมูลส่งและส่งสัญญาณโต้ตอบกับวงจรอินเตอร์เฟส B ซึ่งก็คือ กลไกลทางกล (Machanics) หรือ ทางไฟฟ้า

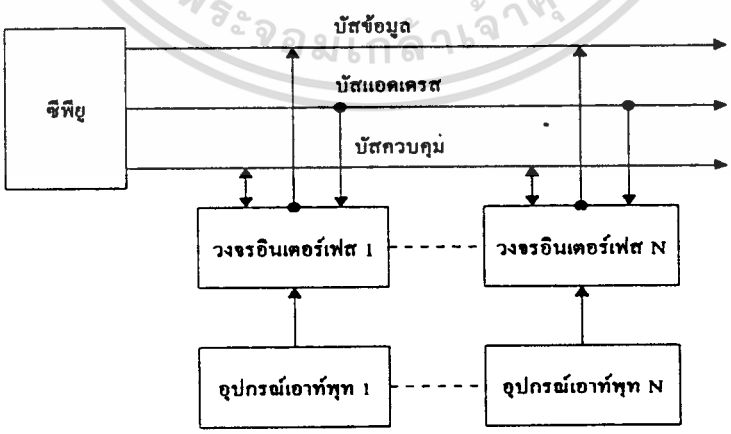


รูปที่ 2.9 วงจรและอุปกรณ์ที่เกี่ยวข้องการเอาต์พุต

สำหรับการแสดงผลในรูปแบบต่างๆ กันเช่น สำหรับจอแสดงผล กลไกลทางไฟฟ้า คือ จอภาพและอุปกรณ์กำเนิดลำแสงอิเล็กตรอนสำหรับให้เกิดภาพ เช่น คอยล์ต่างๆ

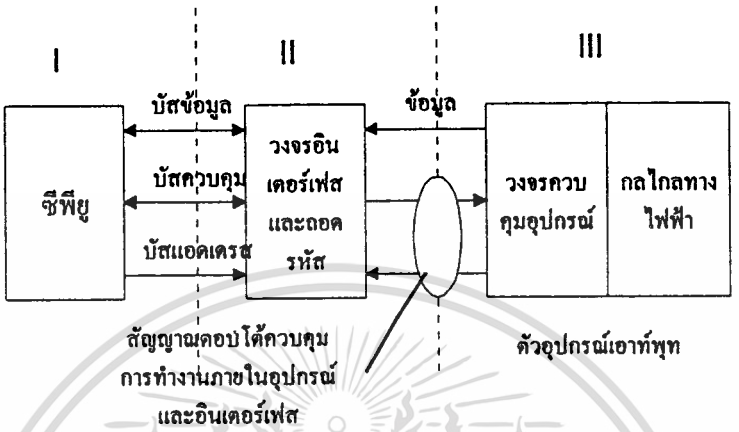
2.3.2 ขบวนการอินพุตและอุปกรณ์อินพุต

ขบวนการอินพุต คือ ขบวนการรับข้อมูลจากอุปกรณ์ส่งข้อมูลผ่านทางบัสข้อมูล



รูปที่ 2.10 ซีพียูรับข้อมูลจากอุปกรณ์อินพุตตัวใดตัวหนึ่งใน n ตัว

เช่นเดียวกันกับการเอาต์พุต ในการติดต่อแต่ละครั้ง จะมีอุปกรณ์และวงจรต่างๆ ที่เกี่ยวข้องดังรูปที่ 2.11



รูปที่ 2.11 วงจรและอุปกรณ์ที่เกี่ยวข้องในการอินพุต

(A) คือ วงจรอิเล็กทรอนิกส์สำหรับควบคุมการทำงานภายในอุปกรณ์อินพุต ทำหน้าที่ส่งข้อมูลและสัญญาณโต้ตอบกับวงจรอินเตอร์เฟส ตัวอย่างเช่น แป้นกดข้อมูล วงจรพวกนี้คือวงจรเข้ารหัสจากการกดแป้นรหัสต่างๆ แปลงเป็นรหัสที่คอมพิวเตอร์เข้าใจ

(B) คือ กลไกลเชิงกลหรือทางไฟฟ้าสำหรับติดต่อกับผู้ใช้งาน เช่น ฟันเฟือง มอเตอร์ แป้นเคาะพิมพ์เป็นต้น

2.3.3 อินพุตและเอาต์พุตพอร์ต

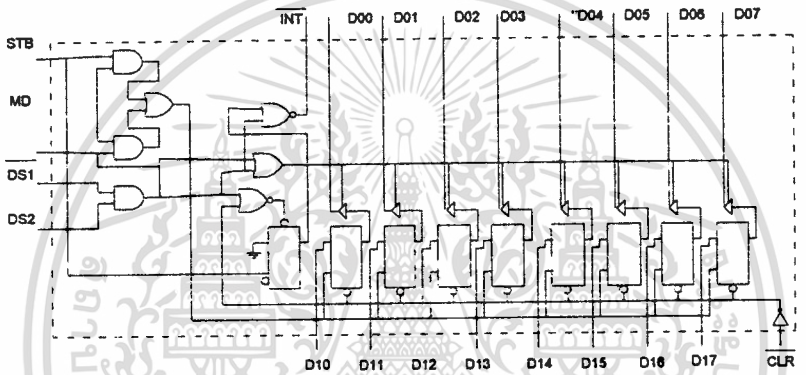
อุปกรณ์ภายนอกที่ต้องการส่งถ่ายข้อมูลกับซีพียูจะต้องอาศัยพอร์ตเป็นตัวกลาง โดยพอร์ตจะทำหน้าที่เป็นช่องทางส่งผ่านข้อมูลสำหรับอุปกรณ์แต่ละตัวอินพุตและเอาต์พุตพอร์ตจะประกอบด้วยวงจรที่สามารถที่สับเปลี่ยนข้อมูลเข้าออกซีพียูได้ พอร์ตเหล่านี้จะถูกเลือกก็ต่อเมื่อถูกเอ็กซีคิวคำสั่งที่เกี่ยวข้องกับอินพุตและเอาต์พุต ซึ่งอินพุตเอาท์พอร์ตมีทั้งสำหรับเป็นอินพุตพอร์ตและเอาท์พอร์ต

อินพุตพอร์ตเป็นช่องทางสำหรับการนำข้อมูลของซีพียู

เอาต์พุตพอร์ตเป็นช่องทางสำหรับนำเอาต์พุตออกจากซีพียู ปกติอินพุตและเอาต์พุตพอร์ตมีแบบการส่งถ่ายข้อมูล 2 แบบ คือ การส่งถ่ายข้อมูลแบบขนานและการส่งถ่ายข้อมูลแบบอนุกรม

อินพุตและเอาต์พุตพอร์ตแบบขนาน (Parallel I/O Port)

ตัวอย่างรูปที่ 2.12 เป็นอินพุตเอาต์พุตพอร์ตแบบขนานชนิด 8 บิต เบอร์ 8212 ซึ่งใช้เป็นอินพุตเอาต์พุตพอร์ตสำหรับการส่งข้อมูลขนานอย่างธรรมดา

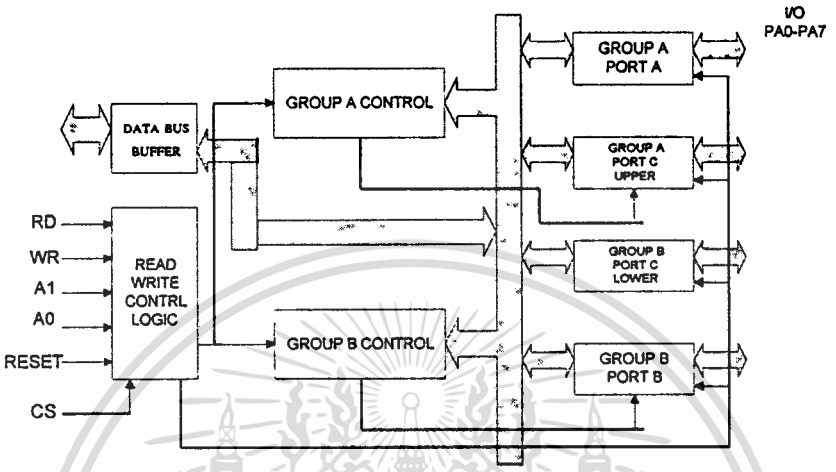


รูปที่ 2.12 โครงสร้างภายในของ 8212 อินพุตเอาต์พุตพอร์ต

ช่องทางการส่งผ่านข้อมูลแบ่งเป็น 2 ส่วน คือ ด้านรับข้อมูลเข้า DI0 - DI7 กับด้านส่งข้อมูลออก DO0 - DO7 ด้านรับข้อมูลเข้าจะประกอบด้วย 8 บิตแลทช์ (Latch) เป็นจำพวก D ฟลิปฟลอป ส่วนทางด้านส่งข้อมูลออกประกอบด้วย 8 บิต เอาต์พุตบัฟเฟอร์ซึ่งจะเป็นพวกไตรสเตทบัฟเฟอร์ นอกจากช่องทางส่งผ่านข้อมูลแล้วยังมีสายควบคุมซึ่งใช้ควบคุมการเคลื่อนย้ายข้อมูลต่างๆ อีก MD เป็นขั้วสัญญาณสำหรับการเลือกโหมด (Mode) ที่ทำงานอยู่ว่าการทำงานจะเป็นอินพุตโหมดก็เมื่อมีสัญญาณที่เป็นลอจิก 1 หรือเอาต์พุตโหมดขา MD ก็จะมีสัญญาณลอจิกเป็น 0 ซึ่งก็จะทำงานได้

อินพุตและเอาต์พุตพอร์ตแบบอนุกรม (Serial I/O Port)

ในที่นี้จะยกตัวอย่างอินพุตเอาท์พอร์ตแบบอนุกรม เบอร์ 8255 ซึ่งเป็นเบอร์ที่นิยมใช้กับไมโครโปรเซสเซอร์มากในปัจจุบัน โดยมีโครงสร้างดังรูปต่อไปนี้



รูปที่ 2.13 พอร์ตเบอร์ 8255

ภายในตัว IC ที่ใช้ในการรับส่งข้อมูลแบบอนุกรมมีส่วนประกอบสำคัญๆ ดังนี้

1.วงจรส่งข้อมูลออกแบบอนุกรมและรับข้อมูลแบบอนุกรม

2.การกำหนดแอดเดรสด้านรับและส่งจะมีขั้วกำหนดการทำงานของพอร์ตโดยรับสัญญาณรหัสจากบัสแอดเดรสหรือตัวถอดรหัสในแอดเดรส

3.บัสควบคุมพอร์ตรับข้อมูลแบบอนุกรม

2.4 อินเตอร์รัพท์ (Interrupt)

อินเตอร์รัพท์ หมายถึง การขัดจังหวะการทำงานของซีพียู เพื่อให้ซีพียูหยุดการทำงานของโปรแกรมที่ทำการอยู่ เมื่อได้รับการอินเตอร์รัพท์จากอุปกรณ์ภายนอก ซีพียูจะทำงานในคำสั่งที่กำลังทำอยู่จนจบคำสั่งนั้นเสียก่อน แล้วซีพียูจึงจะไปบริการให้กับงานที่สั่งสัญญาณอินเตอร์รัพท์เข้ามา เมื่อบริการอินเตอร์รัพท์เสร็จแล้วก็จะกลับไปทำงานที่โปรแกรมเดิมต่อไป

ในการอินเตอร์รัพท์มีลักษณะคล้ายคลึงกับการที่ซีพียู กระโดดไปทำงานที่โปรแกรมย่อยจะแตกต่างกันก็คือ การกระโดดภายในโปรแกรมจะกระโดดไปก็ต่อเมื่อมีคำสั่งจัมพ์ (JUMP) แต่การอินเตอร์รัพท์จะกระโดดไปทำงาน เมื่อซีพียูได้รับสัญญาณอินเตอร์รัพท์จากอุปกรณ์ภายนอก

2.5 อีดิตเตอร์

อีดิตเตอร์เป็นโปรแกรมที่ทำให้ผู้ใช้สามารถสร้างไฟล์ซึ่งอาจเป็นภาษาแอสเซมบลีหรือภาษาอะไรก็ได้แต่ในที่นี้จะขอกล่าวถึงแต่ภาษาแอสเซมบลีเท่านั้นตัวอย่างของโปรแกรมอีดิตเตอร์เช่น โปรแกรมเวิร์ดสตาร์ (Word Star), โปรแกรมไซด์คิก (SideKick) เป็นต้น

จากรูปที่ 2.14 แสดงตัวอย่างของรูปแบบที่ผู้ใช้จะต้องทำตามเมื่อใช้อีดิตเตอร์เพื่อเขียนโปรแกรมภาษาแอสเซมบลีโดยตำแหน่งที่แท้จริงของแต่ละฟิลด์ (Filed) ในบรรทัดหนึ่งๆ ไม่สำคัญเพียงแต่ผู้ใช้จะต้องเขียนฟิลด์ของแต่ละสเตทเมนต์ (Statement) ให้ถูกต้องตามลำดับและจะต้องเว้นช่องว่างอย่างน้อย 1 ช่องตัวอักษรระหว่างแต่ละฟิลด์

เมื่อไรก็ตามที่ใช้อีดิตเตอร์เพื่อเขียนโปรแกรมภาษาแอสเซมบลีมีข้อแนะนำที่เป็นประโยชน์คือ ควรที่จะเขียนให้ฟิลด์ที่เป็นชนิดเดียวกันอยู่ในตำแหน่งหลักที่ตรงกัน ซึ่งจะทำให้ง่ายต่อการอ่านโปรแกรม ขณะที่ผู้ใช้เขียนโปรแกรม อีดิตเตอร์จะทำการเก็บทุกตัวอักษรและทุกตัวเลขโดยแปลงเป็นรหัสแอสกี้ (ASCII Code) โดยเก็บไว้ในแรมตามลำดับของโปรแกรม ถ้าหากผู้ใช้พิมพ์โปรแกรมผิดหรือว่าต้องการแก้ไขส่วนใดส่วนหนึ่งของโปรแกรม อีดิตเตอร์จะช่วยให้เราแก้ไขได้จากตัวโปรแกรมที่อีดิตเตอร์เก็บเอาไว้ และถ้าผู้ใช้ต้องการไปที่สเตทเมนต์ใดๆ ในโปรแกรม อีดิตเตอร์จะทำให้ผู้ใช้สามารถไปที่ใดๆ ก็ได้ในโปรแกรมและสามารถแทรกบรรทัดได้อีกด้วย ซึ่งการใช้อีดิตเตอร์จะมีความง่ายและสะดวกมากกว่าการเขียนโปรแกรมโดยใช้กระดาษและดินสอ เมื่อผู้ใช้เขียนโปรแกรมเสร็จแล้วก็ทำการเก็บเอาไว้ในรูปของไฟล์ โดยเก็บเอาไว้ในแผ่นฟลอปปี้ดิสก์ (Floppy Disk) หรือฮาร์ดดิสก์ (Hard Disk)

ซึ่งขั้นตอนต่อไปก็จะเป็นขั้นตอนของการสร้างซอร์สไฟล์ (Source File) โดยแอสเซมเบลอร์ ถ้าหากว่าใช้โปรแกรม TASM หรือ MASM ก็จะได้ไฟล์ที่มีนามสกุล (Extention) เป็น .ASM ดังที่แสดงเอาไว้ในรูปที่ 2.14

2.6 แอสเซมเบลอร์

โปรแกรมแอสเซมเบลอร์จะใช้ในการเปลี่ยนนีมูนิก (Mnemonics) ของภาษาแอสเซมบลีเป็นรหัสฐานสองเพื่อใช้ในการทำงานของเครื่อง เมื่อใดก็ตามที่ผู้ใช้ทำการรันแอสเซมเบลอร์ มันจะทำการอ่านซอร์สไฟล์ของโปรแกรมจากแผ่นฟลอปปี้ดิสก์ที่เก็บไว้หลังจากเขียนโปรแกรมเสร็จ ในขั้นแรกของการผ่านเข้าไปในซอร์สโปรแกรม แอสเซมเบลอร์จะค้นหาและทำการแทนที่ส่วนของข้อมูล, ออฟเซ็ทของลาเบล (Offset of Labels) ฯลฯ และนำข้อมูลที่แทนที่มาเก็บไว้ในตารางสัญลักษณ์

ขั้นที่สองของการผ่านเข้าไปในซอร์สโปรแกรม แอสเซมเบลอร์จะสร้างรหัสเลขฐานสองสำหรับแต่ละคำสั่งทุกๆ คำสั่งและทำการแทรกออฟเซ็ท ฯลฯ ซึ่งขั้นตอนที่สองนี้จะมีการคำนวณ ระหว่างที่ผ่านขั้นตอนที่หนึ่ง แอสเซมเบลอร์จะทำการสร้างไฟล์ขึ้นมา 2 ไฟล์ที่แผ่นฟลอปปี้ดิสก์หรือฮาร์ดดิสก์ซึ่งไฟล์แรกมีชื่อว่า ออปเจคไฟล์ (Object File) มีนามสกุลเป็น . OBJ ออปเจคไฟล์นี้จะเก็บรหัสเลขฐานสองของคำสั่งและข้อมูลเกี่ยวกับตำแหน่งของคำสั่ง หลังจากนั้นข้อมูลของออปเจคไฟล์จะถูกโหลดเข้าไปไว้ในหน่วยความจำและทำการรัน ส่วนไฟล์ที่สองมีชื่อเรียกว่า แอสเซมเบลอร์ลิสท์ไฟล์ (Assembler List File) และมีนามสกุลเป็น . LST

ซึ่งมีตัวอย่างแสดงในรูปที่ 2.14 ลิสท์ไฟล์จะเก็บทุกสเตทเมนต์ของภาษาแอสเซมบลี รหัสเลขฐานสองของแต่ละคำสั่งและออฟเซ็ทของแต่ละคำสั่งทุกคำสั่ง ผู้ใช้สามารถทำการพิมพ์ลิสท์ไฟล์เก็บไว้ในกระดาษ ซึ่งจะได้รายละเอียดต่างๆ ของโปรแกรมที่เขียนไว้เพื่อนำไปใช้เมื่อมีการทดสอบโปรแกรมและเกิดปัญหาขึ้น ซึ่งลิสท์ไฟล์จะช่วยแสดงข้อมูลที่เขียนและไวยากรณ์ที่ผิดพลาดที่เกิดขึ้นในโปรแกรมที่เขียนขึ้น (ตามหลักของภาษาแอสเซมบลี) เพื่อการแก้จุดที่ผิดพลาดที่แสดงในลิสท์ไฟล์ผู้ใช้จะต้องใช้อีดิตเตอร์เพื่อเขียนโปรแกรมในซอร์สไฟล์อีกครั้งและทำการเก็บซอร์สไฟล์ที่ถูกต้องเอาไว้ แล้วจึงทำการแอสเซมเบลอร์ซอร์สไฟล์ที่ถูกต้อง ซึ่งอาจใช้เวลาในลูปอีดิตเตอร์-แอสเซมเบลอร์หลายครั้งกว่าที่จะไม่มีข้อผิดพลาดเกิดขึ้น

ข้อควรจำสำหรับแอสเซมเบลอร์ คือ แอสเซมเบลอร์เพียงแต่เป็นการตรวจสอบว่าโปรแกรมที่เขียนมีข้อผิดพลาดตามหลักไวยากรณ์หรือไม่ ซึ่งไม่ใช่เป็นการบอกให้รู้ว่าโปรแกรมนั้นจะทำงานได้หรือไม่ดังนั้นเพื่อให้รู้ว่าโปรแกรมที่เขียนขึ้นมานี้จะสามารถทำงานได้หรือไม่ก็โดยการรันโปรแกรมและทดสอบผลที่เกิดขึ้น

ให้พิจารณาข้อมูลที่ได้จากการลิสท์ของแอสเซมเบลอร์ในรูปที่ 2.14 ทางด้านซ้ายสุดของหลักก็คือ กลุ่มออฟเซ็ทของข้อมูลจากจุดเริ่มต้นของเซ็กเมนต์และออฟเซ็ทของรหัสเป็นจำนวนไบต์ จากจุดเริ่มของเซ็กเมนต์ข้อมูล ควรจำเอาไว้ว่าแอสเซมเบลอร์จะแสดงเพียงแต่ค่าออฟเซ็ทเท่านั้นไม่ใช่แสดงค่าแอดเดรสที่แท้จริง ซึ่งลิงค์เกอร์ (Linker) หรือ โลเคเตอร์ (Locator) จะนำค่าออฟเซ็ทนี้ไปใช้ในการกำหนดค่าเริ่มต้นของแอดเดรสที่แท้จริงสำหรับทุกๆ เซกเมนต์ จากข้อมูลที่ได้จากลิสท์ไฟล์ ควรจำเอาไว้ว่าที่สเตทเมนต์ของคำสั่ง MOV AX,DATA_HERE ได้ถูกแอสเซมเบลอร์พร้อมกับช่องว่างที่ตามหลังคำสั่งแล้ว เพราะว่าแอสเซมเบลอร์ไม่รู้ค่าจุดเริ่มต้นของ DS ขณะที่โปรแกรมถูกแอสเซมเบลอร์

จากส่วนที่ตามมาจากข้อมูลที่ได้จากการลิสท์ในรูปที่ 2.14 เป็นการแสดงข้อมูลที่เพิ่มมาของเซกเมนต์และชื่อที่ใช้ในโปรแกรม จากสเตทเมนต์ CODE_HERE 16 0014 Para none จากตัวอย่างนี้จะบอกให้ผู้อ่านรู้ว่าเซกเมนต์ CODE_HERE มีขนาดความยาว 14H (ยาว 14ตัวในเลขฐาน 16 หรือเท่ากับ 20 ตัวในฐานสิบ)

สเตทเมนต์ MULTIPLIER Word DATA_HERE:0002 จะบอกให้ผู้อ่านรู้ว่า MULTIPLIER เป็นตัวแปรชนิดเวิร์ดและมีตำแหน่งอยู่ที่ออฟเซ็ท 0002 ในเซกเมนต์ DATA_HERE

Turbo Assembler Version 1.0

```
1                                   ;8086 PROGRAM F3-14 .ASM
2                       ;ABSTRACT   This program multiplies the two 16 bit word 3
                                    ;in the memory. Locations called
4                                   ;MULTIPLICAND and MULTIPLIER.
5                                   ;The result is stored in the memory location
6                       ;REGISTERS  Uses CS, DS,AX, DX
7                       ;PORTS      None used
8
9 0000                  DATA_HERE   SEGMENT
10 0000  204A                       MULTIPLICAND  DW 204AH
11 0002  3B2A                       MULTIPLIER    DW 3B2AH
```

```
12 0004  02*(0000)                PRODUCT     DW 2 DUP(0)
13 0008                  DATA_HERE  ENDS
14
15 0000                        CODE_HERE  SEGMENT
16                                    ASSUME  CS:CODE_HERE, DS:DATAHERE
17 0000 B8 0000s         START:       MOV       AX, DATA_HERE
18 0003  8E D8                        MOV         DS, AX
19 0005  A1 0000r                     MOV           AX, MULTIPLICAND
20 0008  F7 26 0002r                  MUL           MULTIPLIER
21 000C  A3 0004r                     MOV           PRODUCT, AX
22 000F  89 16 0006r                  MOV           PRODUCT+2, DX
23 0013  CC                           INT           3
24 0014                               CODE_HERE     ENDS
25                                    END           START
```

Turbo Assembler Version 1.0

Symbol Table

Symbol Name	Type	Value
??DATA	Text	"04-06-89"
??FILENAME	Text	"F3-14 "
??TIME	Text	"07:41:58 "
??VERSION	Number	0100
@CPU	Text	0101H
@CURSEG	Text	CODE_HERE
@FILENAME	Text	F3-14
@WORDSIZE	Text	2

```
MULTIPLICAND                    Word   DATA_HERE:0000
MULTIPLIER                      Word   DATA_HERE:0002
PRODUCT                         Word   DATA_HERE:0004
START                           Near   CODE_HERE:0000

Group & Segments                Bit Size Align Combine Class

CODE_HERE                       16  0014 Para  none
CODE_HERE                       16  0008 Para  none
```

รูปที่ 2.14 ตัวอย่างการลิสท์ของแอสเซมเบลอร์

2.7 ดีบักเกอร์

ถ้าโปรแกรมของผู้ใช้ไม่ต้องการฮาร์ดแวร์ภายนอกหรือต้องการเฉพาะฮาร์ดแวร์ที่ใช้ได้โดยตรงจากเครื่องไมโครคอมพิวเตอร์แล้ว สามารถที่จะใช้โปรแกรมดีบักเกอร์เพื่อทำการดีบักโปรแกรมที่ต้องการได้ ดีบักเกอร์ คือ โปรแกรมที่ทำให้ผู้ใช้ทำการโหลดโปรแกรมที่เก็บรหัสออปเจคเข้าสู่ระบบของหน่วยความจำ, ทำการเอ็กซีคิวโปรแกรมและเป็นตัวที่ช่วยแก้ปัญหาหรือดีบักโปรแกรม ดีบักเกอร์จะทำให้ผู้ใช้สามารถมองเห็นค่าที่อยู่ในรีจิสเตอร์และตำแหน่งของหน่วยความจำหลังจากรันโปรแกรมที่ต้องการ ซึ่งจะช่วยให้ผู้ใช้ทำการเปลี่ยนข้อมูลในรีจิสเตอร์และตำแหน่งของหน่วยความจำที่ต้องการและรันโปรแกรมใหม่อีกครั้ง โปรแกรมดีบักเกอร์บางรุ่นสามารถให้ผู้ใช้หยุดการเอ็กซีคิวหลังจากเสร็จสิ้นในแต่ละคำสั่ง ซึ่งจะช่วยให้ผู้ใช้สามารถเช็คหรือเปลี่ยนข้อมูลในหน่วยความจำและรีจิสเตอร์

ดีบักเกอร์ทำให้ผู้ใช้สามารถตั้งจุดเบรค (Break Point) ได้ทุกๆ จุดในโปรแกรม โดยถ้าผู้ใช้ทำการแทรกจุดเบรค ดีบักเกอร์จะรันโปรแกรมจนถึงคำสั่งที่ที่ผู้ใช้ตั้งจุดเบรคเอาไว้และจะหยุดเอ็กซีคิวที่จุดนั้น ผู้ใช้สามารถที่จะพิจารณาข้อมูลในรีจิสเตอร์หรือหน่วยความจำจากข้อมูลที่เห็นจากการดีบักว่าผลลัพธ์ที่ได้ถูกต้องหรือไม่ ผู้ใช้สามารถย้ายจุดเบรคไปจนถึงจุดสุดท้ายของโปรแกรม ถ้าหากว่าผลลัพธ์ที่ได้ไม่ถูกต้องผู้ใช้สามารถเช็คโปรแกรมไปจนจุดที่พบว่า

น่าจะไม่ถูกต้อง จุดเบรคที่กล่าวถึงนี้เป็นคำสั่งของดีบักเกอร์ที่ช่วยให้ผู้ใช้ค้นหาต้นตอของความผิดพลาดในโปรแกรมได้อย่างรวดเร็ว เมื่อผู้ใช้พบปัญหาครั้งแรกถ้าหากจำเป็นแล้วผู้ใช้สามารถกลับไปเริ่มต้นแก้ปัญหาใหม่ตามอัลกอริทึม โดยใช้อีดิตเตอร์เพื่อแก้ซอร์สไฟล์การแอสเซมเบลอร์ของซอร์สโปรแกรมอีกครั้งและรันโปรแกรมอีกครั้งหนึ่งด้วย

ดีบักเกอร์ที่เป็นพื้นฐานที่สุดจะติดมากับ DOS (Disk Operating System) แต่ส่วนมากนิยมใช้โปรแกรมที่มีประสิทธิภาพเช่น บอร์แลนด์เทอร์โบดีบักเกอร์ (Borland's Turbo Debugger) ซึ่งจะช่วยให้ผู้ใช้ทำการดีบักได้ง่ายเพราะว่ามันจะช่วยให้ผู้ใช้สามารถมองค่าที่อยู่ในรีจิสเตอร์และตำแหน่งของหน่วยความจำที่เปลี่ยนไปตามการเอ็กซีคิวของโปรแกรม แผงวงจรไมโครโปรเซสเซอร์เช่น SDK-86 จะมีโปรแกรมดีบักเกอร์ในรอมของแผงวงจร

ดีบักเกอร์แบบนี้เรียกรวมๆ ว่า โปรแกรมควบคุมระบบ เพราะว่ามันช่วยให้ทำงานได้โดยสะดวก ตัวอย่างเช่น สามารถให้ผู้ใช้ป้อนข้อมูลและรันโปรแกรม, ทำการตรวจสอบค่าในรีจิสเตอร์และค่าในหน่วยความจำผ่านโปรแกรมซิงเกิลสเต็ป (Single Step) และแทรกจุดเบรค

2.8 อีมูเลเตอร์

เป็นอีกทางหนึ่งที่จะรันโปรแกรมของผู้ใช้คือรันพร้อมกับอีมูเลเตอร์ อีมูเลเตอร์เป็นการผสมกันของฮาร์ดแวร์และซอฟท์แวร์ของระบบภายนอก ซึ่งใช้กันอย่างมากในการทดสอบและดีบักฮาร์ดแวร์และซอฟท์แวร์ของระบบภายนอก เช่น ตรวจสอบระบบพื้นฐานดั้งเดิมของไมโครโปรเซสเซอร์ ส่วนประกอบทางฮาร์ดแวร์ของอีมูเลเตอร์คือ สายเคเบิลหลายๆ สายที่ต่ออยู่กับระบบหลักไปยังระบบที่ต้องการพัฒนา ปลั๊กที่ปลายสายเคเบิลจะต่อเข้ากับระบบเดิมที่อยู่ในไมโครโปรเซสเซอร์ โดยการผ่านทางสายเคเบิล

ซอฟท์แวร์ของอีมูเลเตอร์จะทำให้ผู้ใช้สามารถทำการดาวน์โหลดรหัสของโปรแกรมออปเจคลงไปยังแรมของระบบที่ทำการทดสอบและทำการรันโปรแกรม

อีมูเลเตอร์จะทำให้ผู้ใช้ทำการโหลดและรันโปรแกรม, ตรวจสอบ และเปลี่ยนข้อมูลของ รีจิสเตอร์, ตรวจสอบและเปลี่ยนข้อมูลตำแหน่งของหน่วยความจำ และแทรกจุดเบรคในโปรแกรม อีมูเลเตอร์จะทำการส่งข้อมูลของรีจิสเตอร์และสถานะของแฟลกต่างๆ ในการเอ็กซีคิวแต่ละคำสั่งอย่างรวดเร็วผ่านไปทางแอดเดรสบัสและดาต้าบัสที่ใช้งานอยู่ อีมูเลเตอร์จะเก็บข้อมูลเหล่านี้ไว้ในลักษณะสายข้อมูล (Trace Data) ตามลำดับของข้อมูลเอาไว้ในแรมที่มี

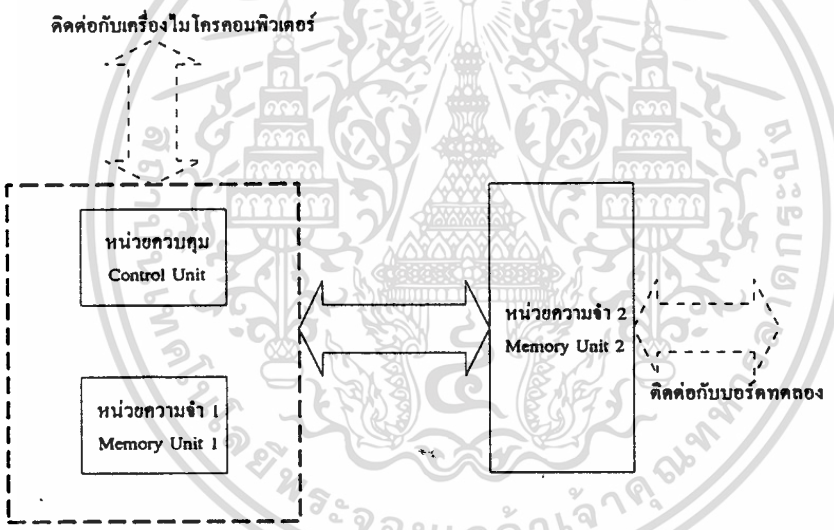
ขนาดใหญ่ ผู้ใช้สามารถพิมพ์สายข้อมูลออกมาเพื่อที่จะเห็นผลลัพธ์ของโปรแกรมที่รันแบบทีละสเต็ป

อีกคุณสมบัติหนึ่งที่มีประสิทธิภาพของอีมูเลเตอร์คือ ความสามารถในการใช้หน่วยความจำของแต่ละระบบหรือหน่วยความจำของระบบเดิมสำหรับโปรแกรมที่ทำการดีบัก

บทที่ 3
การออกแบบเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51

3.1 หลักการออกแบบ

เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 จะทำให้ผู้ใช้สามารถรู้ถึงค่าต่างๆ ในหน่วยความจำของบอร์ดทดลองและสามารถทดลองหรือพัฒนาบอร์ดทดลองได้โดยสะดวก โดยโครงสร้างของเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 แบ่งเป็นหน่วยต่างๆ ได้ดังรูปที่ 3.1



รูปที่ 3.1 โครงสร้างของเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51

3.1.1 หน่วยควบคุม

เป็นหน่วยที่ทำหน้าที่ในการควบคุมการทำงานต่างๆ ของเครื่อง กล่าวคือ ควบคุมการติดต่อระหว่างเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 กับเครื่องไมโครคอมพิวเตอร์ทางพอร์ทสื่อสารอนุกรม (RS-232C), ควบคุมการโอนย้ายข้อมูลระหว่างหน่วยความจำที่ 1 กับหน่วยความจำที่ 2, จัดเรียงคำสั่งที่ใช้ในการอ่านค่าสภาวะต่างๆ ของ

บอร์ดทดลอง, จัดเรียงค่าสภาวะของ SFR , Internal RAM, External RAM ที่อ่านค่าได้จากบอร์ดทดลอง

3.1.2 หน่วยความจำ 1

ทำหน้าที่เก็บโปรแกรมของบอร์ดทดลอง โดยจะจำลองตัวเองเป็นเสมือนหน่วยความจำที่ใช้ในการเก็บโปรแกรมของบอร์ดทดลองจากดิปสวิทช์ที่ใช้ในการเลือกขนาดของหน่วยความจำ

3.1.3 หน่วยความจำ 2

ทำหน้าที่เก็บโปรแกรมที่ใช้ในการอ่านค่าสภาวะต่างๆ ของบอร์ดทดลอง, เก็บค่าสภาวะต่างๆ ที่อ่านได้จากบอร์ดทดลอง, ทำการรักษาสภาวะเดิมของบอร์ดทดลอง โดยหน่วยความจำ 2 มีขนาด 2 กิโลไบท์

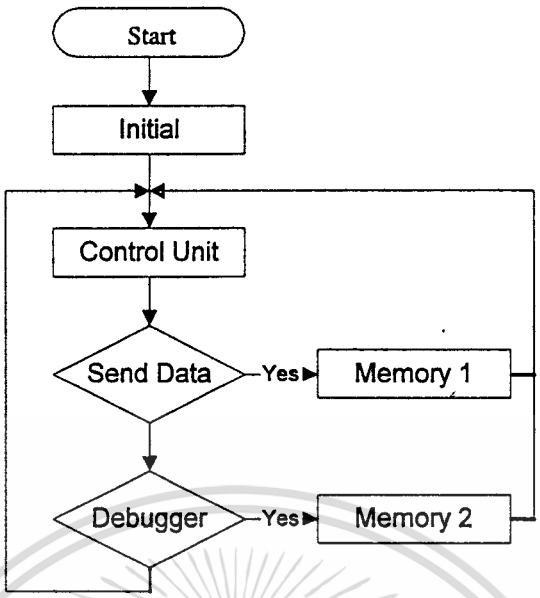
3.2 โครงสร้างการทำงาน

การทำงานของเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 แสดงในรูปที่ 3.2 ซึ่งสามารถอธิบายการทำงานได้ดังต่อไปนี้

เมื่อเริ่มต้นทำงานจะต้องมีการกำหนดค่าสภาวะเริ่มต้นให้แก่เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 เสียก่อน หลังจากนั้นหน่วยควบคุมจะทำหน้าที่ในการส่งข้อมูลไปให้แก่หน่วยความจำ 1 หรือทำการดีบักโปรแกรมของบอร์ดทดลองซึ่งในกรณีนี้จะทำการติดต่อกับหน่วยความจำ 2 ซึ่งหน้าที่การทำงานต่างๆ จะถูกควบคุมจาก โปรแกรม MEMDP ที่ทำงานบนเครื่องไมโครคอมพิวเตอร์

3.2.1 ผังการทำงานของหน่วยควบคุม

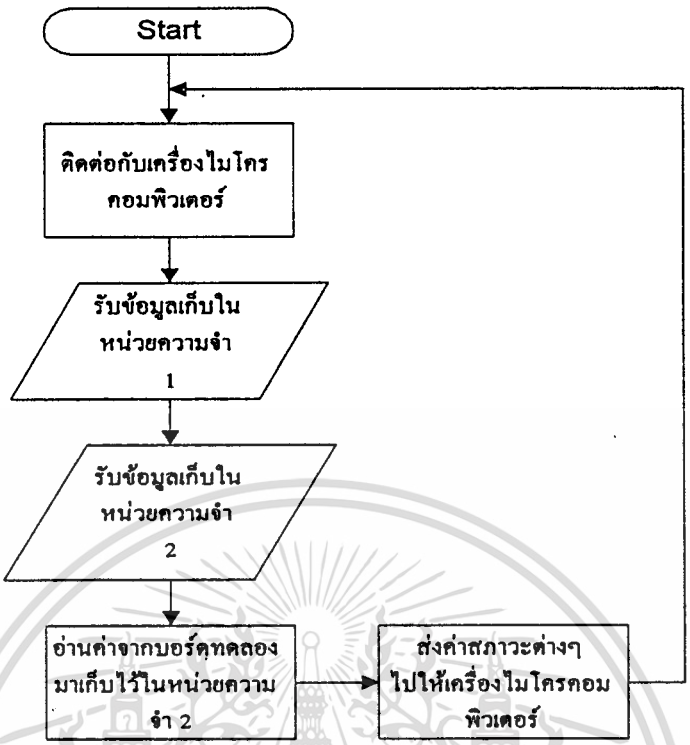
ในหน่วยควบคุมจะมีซีพียูเบอร์ 8951 ทำหน้าที่เป็นตัวควบคุมการทำงานต่างๆ ของระบบทั้งหมด โดยจะถือว่าการทำงานของแผงวงจรทดลองมีความสำคัญสูงสุด (Hi Piority) และจะทำการจัดเรียงคำสั่งในการดีบักทั้งหมดด้วย ซึ่งลักษณะการทำงานของหน่วยควบคุมสามารถอธิบายได้ตามรูปที่ 3.3



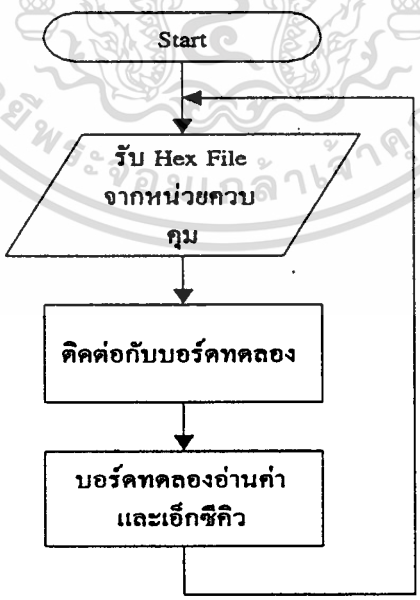
รูปที่ 3.2 ผังการทำงานของเครื่องโดยรวม

3.2.2 ผังการทำงานของหน่วยความจำ 1

ในหน่วยความจำ 1 จะเป็นตัวทำหน้าที่ในการเก็บโปรแกรมหลัก (Hex File) โดยจุดที่ต้องการหยุดรันจะแทรกคำสั่ง LCALL ลงไป 3 ไบท์ เพื่อที่จะกระโดดไปยังตำแหน่ง 2 กิโลไบท์สุดท้าย ซึ่งเป็นของหน่วยความจำ 2 ทำให้หน่วยความจำ 1 ว่างเป็นผลทำให้เกิดสัญญาณอินเตอร์รัพท์ 0 (INT0) บอกแก่ซีพียูในหน่วยควบคุมให้ทำการสั่งงานต่อไป ซึ่งการทำงานของหน่วยความจำ 1 สามารถอธิบายได้ตามรูปที่ 3.4



รูปที่ 3.3 ผังการทำงานของหน่วยควบคุม



รูปที่ 3.4 ผังการทำงานของหน่วยความจำ 1

3.2.3 ผังการทำงานของหน่วยความจำ 2

หน่วยความจำ 2 ทำหน้าที่เก็บโปรแกรมในการดีบัก เมื่อมีการกระโดดมายังตำแหน่งที่หน่วยความจำ 2 อยู่จะมีการอ่านค่าสภาวะต่างๆ เก็บลงในหน่วยความจำ 2 และจะทำการรีเซ็ท (Reset) กลับ ทำให้หน่วยความจำ 2 ว่างและส่งสัญญาณอินเตอร์รัพท์ 1 ไปยังซีพียูให้ทำการอ่านสภาวะของแผงวงจรทดลองนั้นโดยส่งออกทางพอร์ทอนุกรม ซึ่งการทำงานของหน่วยความจำ 2 สามารถอธิบายได้ตามรูปที่ 3.5

3.3 การออกแบบส่วนโปรแกรม

เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 จะทำงานร่วมกับโปรแกรมอยู่ 2 โปรแกรมด้วยกัน คือ

1. โปรแกรม MEMDP
2. โปรแกรม CMEMDP

ซึ่งหน้าที่การทำงานและวิธีการออกแบบของโปรแกรม MEMDP และโปรแกรม CMEMDP มีดังนี้

3.3.1 การออกแบบส่วนโปรแกรม MEMDP

โปรแกรม MEMDP (Memory Emulator and MCS-51 Debugger Program) มีหน้าที่ในการติดต่อระหว่างผู้ใช้เพื่อสั่งให้เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ทำงานตามต้องการ โดยคำสั่งต่างๆ จะเป็นแบบเมนูเพื่อให้การใช้คำสั่งเป็นไปอย่างสะดวกและง่ายต่อการใช้งานโปรแกรม การเรียกเมนูควบคุมสามารถทำได้โดยกด F10 ซึ่งรายละเอียดของเมนูต่างๆ มีดังนี้

เมนู File

ในเมนู File นี้มีคำสั่งต่างๆ รวมอยู่ด้วย คือ

- คำสั่ง Load ทำหน้าที่โหลดไฟล์ข้อมูลที่ต้องการเข้ามายังโปรแกรม MEMDP ซึ่งสามารถกดปุ่ม F3 แทนการเลือกที่เมนูได้

- คำสั่ง New File ทำหน้าที่ในการสร้างไฟล์ข้อมูลใหม่ ซึ่งสามารถกดปุ่ม F4 แทนการเลือกที่เมนูได้

- คำสั่ง Clear ทำหน้าที่ในการยกเลิกการแสดงไฟล์ข้อมูลที่ปรกฏอยู่ในช่องแสดง ซึ่งสามารถกดปุ่ม F5 แทนการเลือกที่เมนูได้

- คำสั่ง Change Dir ทำหน้าที่ในการเปลี่ยนไดเรกทอรี่ (Directory) ในการเรียกไฟล์ข้อมูลที่ต้องการโหลดเข้ามาในโปรแกรม MEMDP

- คำสั่ง OS Shell ทำหน้าที่ออกไปสู่ระบบดอสชั่วคราว โดยสามารถกลับเข้าสู่โปรแกรม MEMDP ได้โดยการสั่ง Exit

- คำสั่ง Exit ทำหน้าที่กลับสู่ระบบดอส สามารถกดปุ่ม ALT-X แทนการเลือกที่เมนูได้

เมนู Edit

ทำหน้าที่สร้างไฟล์ขึ้นมาใหม่ โดยจะเข้าสู่โปรแกรมอีดิตเตอร์

เมนู Run

ในเมนู Run มีคำสั่งต่างๆ รวมอยู่ด้วย คือ

- คำสั่ง Run ทำหน้าที่ส่งโปรแกรมที่เปิดใช้อยู่ในขณะนั้นลงไปที่แผงวงจรทดสอบ ซึ่งเครื่องจำลองหน่วยความจำและทดสอบโปรแกรมจะทำหน้าที่เป็นอีมูเลเตอร์

- คำสั่ง Single Step ทำหน้าที่รันโปรแกรมแบบทีละคำสั่งโดยจะไม่เข้าไปในรูทีน ซึ่งโปรแกรมจะรันตามจังหวะการกดปุ่ม F8 เพื่อที่ผู้ใช้สามารถมองเห็นขั้นตอนการทำงานของโปรแกรมควบคุมหรือสามารถตรวจหาข้อบกพร่องของโปรแกรมได้ง่ายและมีการแสดงค่าของหน่วยความจำและรีจิสเตอร์ทางช่องแสดง

- คำสั่ง Trace into ทำหน้าที่รันโปรแกรมแบบทีละคำสั่งทุกคำสั่งโดยจะเข้าไปยังรูทีน ซึ่งโปรแกรมจะรันตามจังหวะการกดปุ่ม F7 เพื่อที่ผู้ใช้สามารถมองเห็นขั้นตอนการทำงานของโปรแกรมควบคุมหรือสามารถตรวจหาข้อบกพร่องของโปรแกรมได้ง่ายและมีการแสดงค่าของหน่วยความจำและรีจิสเตอร์ทางช่องแสดง

- คำสั่ง Stop ทำหน้าที่ยกเลิกการรันแบบทีละเสต็ปและการรันแบบตั้งจุดเบรค

เมนู Break Point

ในเมนู Break Point มีคำสั่งต่างๆ รวมอยู่ด้วย คือ

- คำสั่ง Display Break Point ทำหน้าที่แสดงจุดเบรคที่ผู้ใช้ตั้งขึ้น โดยสามารถแสดงได้ 10 จุด ซึ่งการตั้งจุดเบรคนั้นสามารถทำได้โดยการกดปุ่ม Enter ขณะที่เข้าสู่ช่องแสดงโปรแกรม

- คำสั่ง Clear Break Point ทำหน้าที่ยกเลิกจุดเบรคทั้งหมด

- คำสั่ง Break Point Set up ทำหน้าที่เลือกลักษณะการทำงานของคำสั่ง Break Point ซึ่งมีอยู่ 2 ลักษณะ คือ

1. Fix Break Point ทำหน้าที่รันโปรแกรมจนถึงจุดเบรคที่ตั้ง ตามลำดับที่ตั้งเอาไว้โดยไม่คำนึงถึงค่าของแอดเดรส

2. Quene Break Point ทำหน้าที่รันโปรแกรมจนถึงจุดเบรคที่ตั้งเอาไว้เรียงตามค่าของแอดเดรส (จากน้อยไปมาก)

เมนู About

ในเมนู About มีคำสั่งต่างๆ รวมอยู่ คือ

- คำสั่ง Help ทำหน้าที่ในการแสดงวิธีการใช้งานโปรแกรม MEMDP แบบคร่าวๆ

- คำสั่ง About Program ทำหน้าที่แสดงรายระเอียดต่างๆ เกี่ยวกับโปรแกรม MEMDP

เมนู Set up

ในเมนู Set up มีคำสั่งต่างๆ รวมอยู่ คือ

- คำสั่ง Eprom ทำหน้าที่เลือกเบอร์ของอีพรอม ซึ่งสามารถเลือกได้ 4 เบอร์ คือ เบอร์ 2764, 27128, 27256 และ 27512

- คำสั่ง Ram ทำหน้าที่เลือกเบอร์ของแรม ซึ่งสามารถเลือกได้ 2 เบอร์ คือ เบอร์ 6264 และ 62256

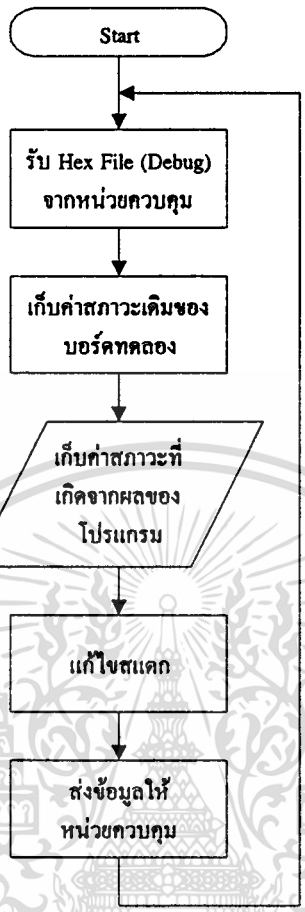
- คำสั่ง Internal Ram ทำหน้าที่ในการแสดงค่าในหน่วยความจำภายใน โดยการใส่แอดเดรสของหน่วยความจำภายใน

- คำสั่ง External Ram ทำหน้าที่ในการแสดงค่าในหน่วยความจำภายนอก โดยการใส่แอดเดรสของหน่วยความจำภายนอก

- คำสั่ง Baud Rate ทำหน้าที่ในการตั้งค่าพารามิเตอร์ของการรับส่งข้อมูลแบบอนุกรม ซึ่งขั้นตอนการออกแบบโปรแกรม MEMDP สามารถอธิบายได้ตามรูปที่ 3.6

3.3.2 การออกแบบส่วนโปรแกรม CMEMDP

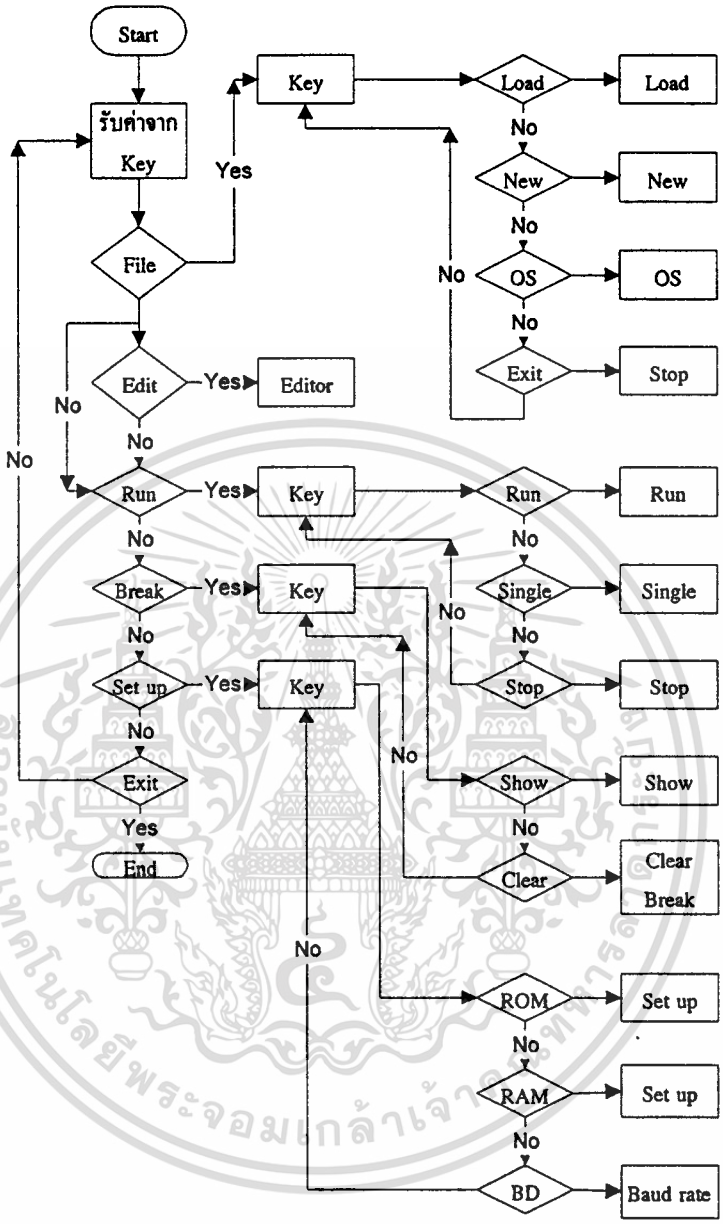
โปรแกรม CMEMDP ทำหน้าที่ติดต่อกับโปรแกรม MEMDP เพื่อรับคำสั่งมาควบคุมการทำงานของเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ซึ่งขั้นตอนการออกแบบส่วนโปรแกรม CMEMDP แสดงดังรูปที่ 3.7



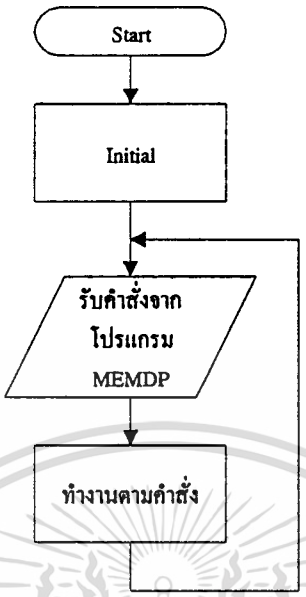
รูปที่ 3.5 ผังการทำงานของหน่วยความจำ 2

3.4 การออกแบบส่วนของวงจร

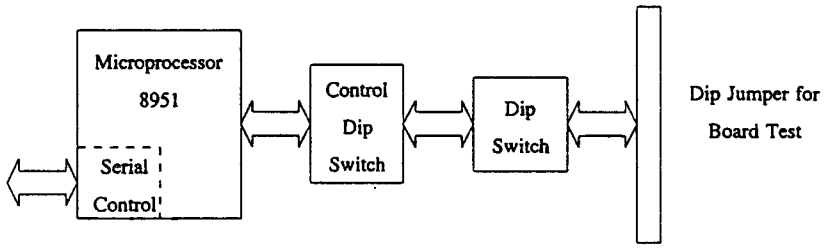
เลือกใช้ไมโครโปรเซสเซอร์เบอร์ 8951 ของบริษัทอินเทล เนื่องจากมีโครงสร้างคล้ายกับไมโครโปรเซสเซอร์เบอร์ 8031 ซึ่งคณะผู้สร้างมีความคุ้นเคยและมีหน่วยความจำสำหรับเก็บโปรแกรมอยู่ในตัวทำให้ประหยัดเนื้อที่ในการใช้งาน จากโครงสร้างโดยรวมของเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ที่กล่าวมา สามารถแสดงรายละเอียดของแต่ละหน่วยได้ดังรูปที่ 3.8



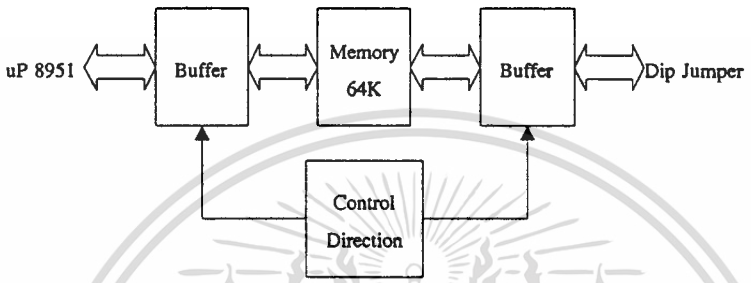
รูปที่ 3.6 ผังการออกแบบส่วนโปรแกรม MEMDP



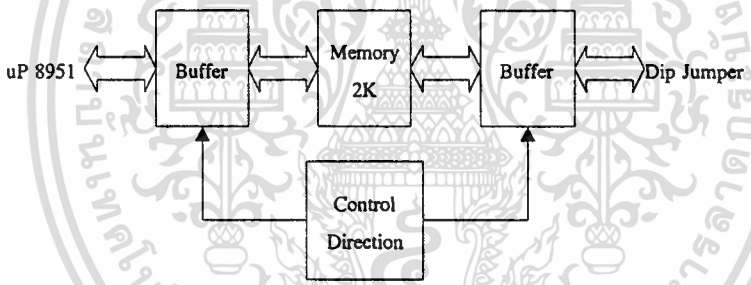
รูปที่ 3.7 ผังการออกแบบส่วนโปรแกรม CMEMDP



ก. โครงสร้างหน่วยควบคุม



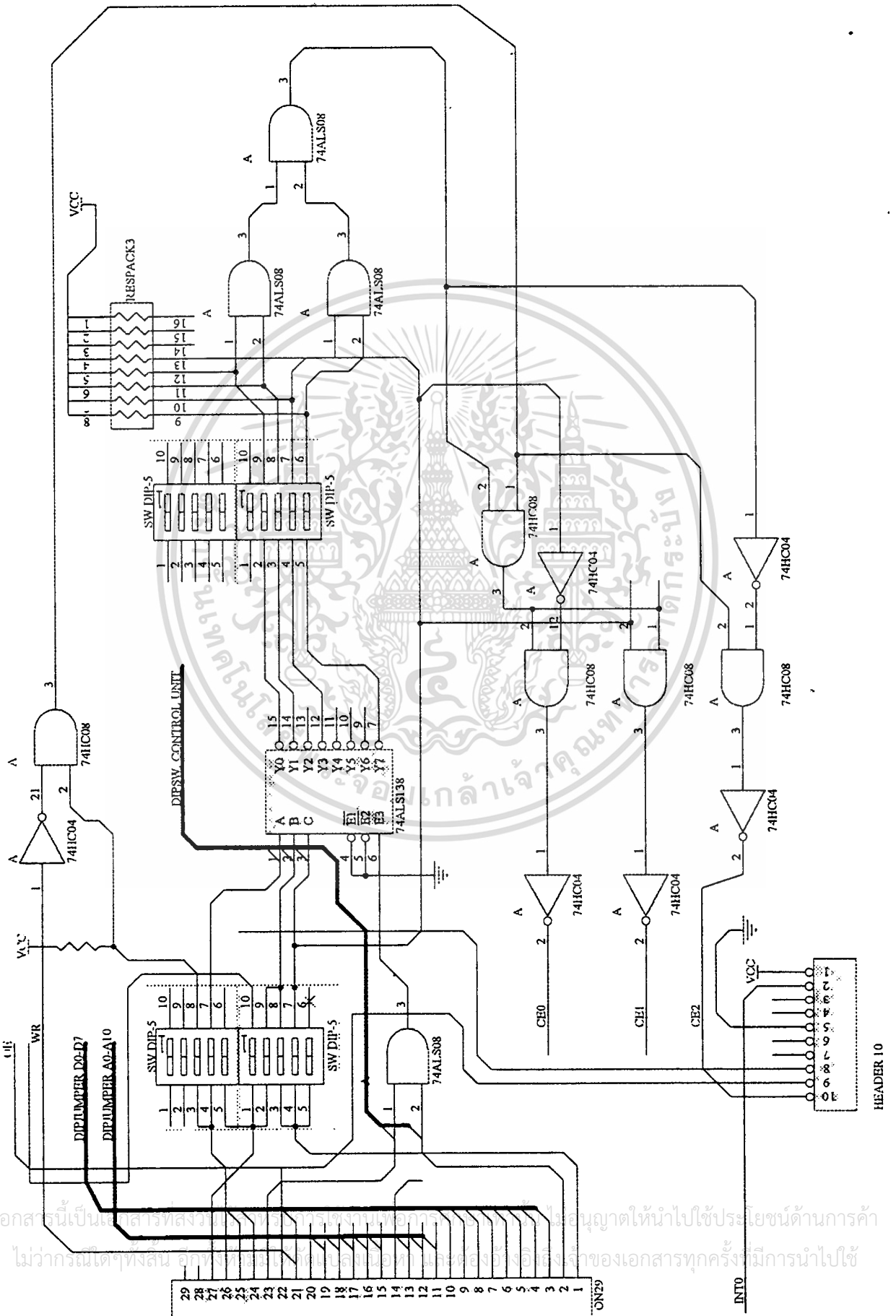
ข. โครงสร้างหน่วยความจำ 1



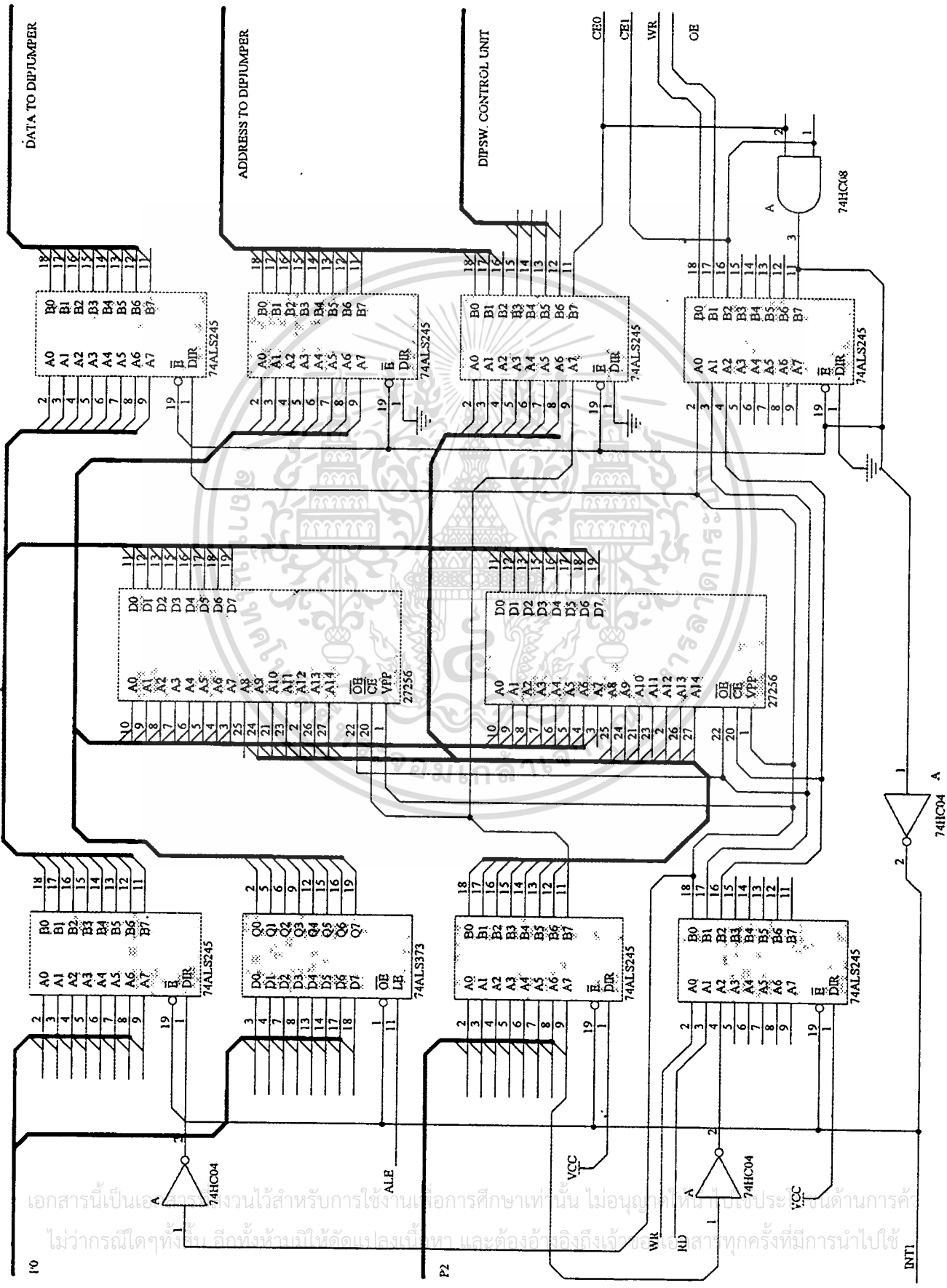
ค. โครงสร้างหน่วยความจำ 2

รูปที่ 3.8 โครงสร้างของหน่วยต่างๆ

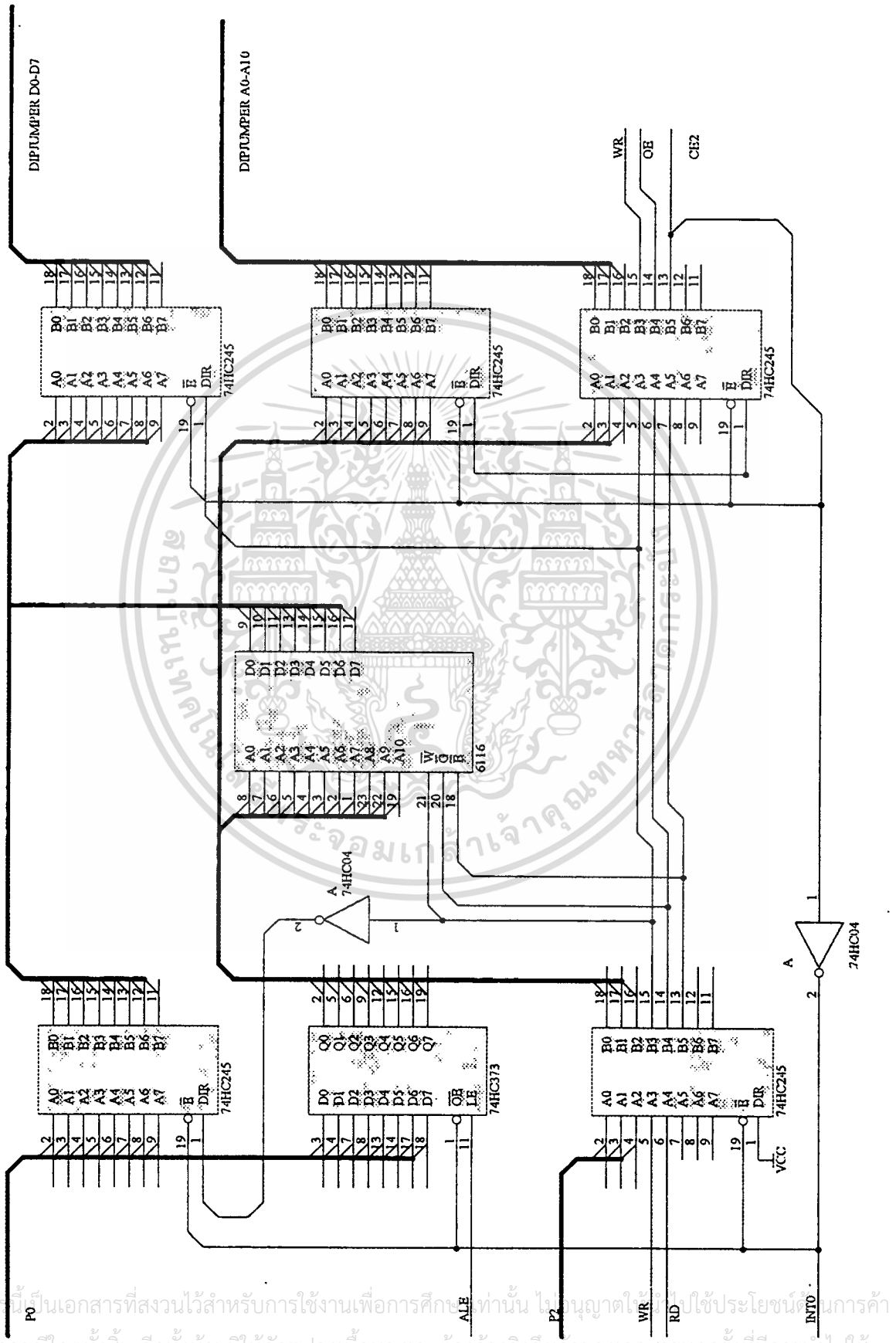
3.4.1 การออกแบบวงจรหน่วยควบคุม



3.4.2 การออกแบบวงจรหน่วยความจำ 1



3.4.3 การออกแบบวงจรหน่วยความจำ 2

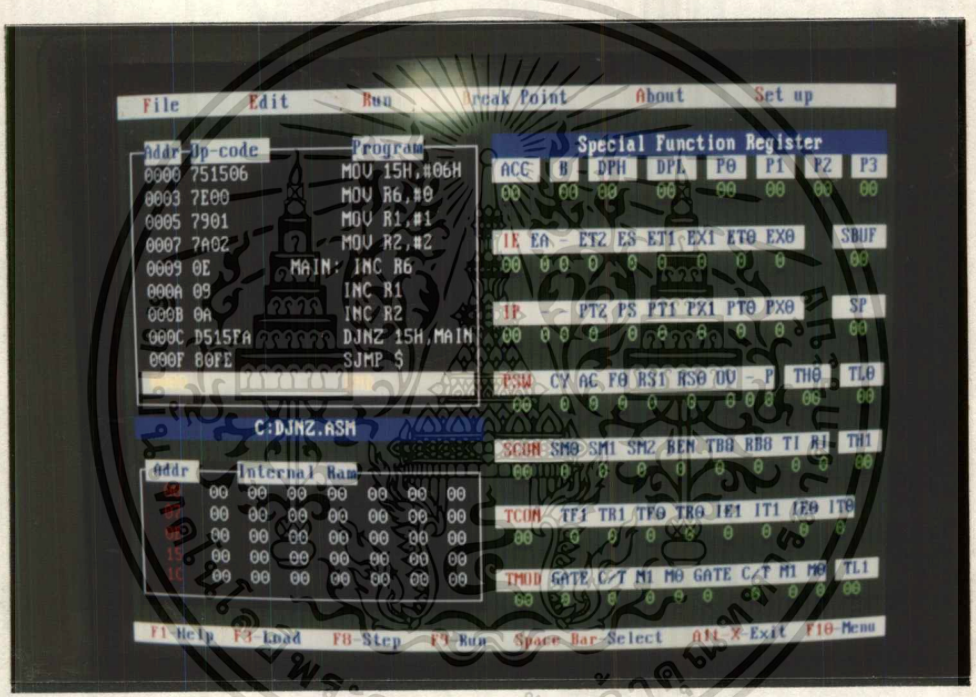


บทที่ 4

ผลการทดลองและทดสอบ

4.1 การทดลองการทำงานของโปรแกรม MEMDP

ในขั้นแรกเมื่อต้องการใช้โปรแกรม MEMDP ให้พิมพ์ MEMDP แล้วกดปุ่ม Enter ก็จะเข้าสู่โปรแกรมโดยจะปรากฏโปรแกรมดังรูปที่ 4.1 ซึ่งแสดงว่าพร้อมที่จะทำงานแล้ว



รูปที่ 4.1 โปรแกรม MEMDP

ในส่วนนี้จะเป็นการทดลองการทำงานของโปรแกรม MEMDP ในทุกเมนูคำสั่ง ดังนี้

4.1.1 การทดลองคำสั่ง Load

เป็นการทดลองผลที่เกิดขึ้นจากการใช้คำสั่ง Load

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนูหรือกดปุ่ม F3 ก็จะไปที่ขั้นตอนที่ 3

2. เลื่อนแถบสีมาที่เมนู File แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Load แล้วกด Enter

3. จะปรากฏเมนูถามชื่อไฟล์ ให้ผู้ใช้ป้อนชื่อไฟล์ที่ต้องการเข้าไปแล้วกด Enter

4. ไฟล์ที่ต้องการปรากฏอยู่ในช่องแสดง เป็นการจบการทำงานของคำสั่ง Load

ผลการทดลอง

ในช่องแสดง จะปรากฏโปรแกรมที่โหลดเข้าไป

4.1.2 การทดลองคำสั่ง New File

เป็นการทดลองผลที่เกิดขึ้นจากการใช้คำสั่ง New File

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนูหรือกดปุ่ม F4 ก็จะไปที่ขั้นตอนที่ 3

2. เลื่อนแถบสีมาที่เมนู File แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง New File แล้วกด Enter

3. จะปรากฏเมนูถามชื่อไฟล์ ให้ผู้ใช้ป้อนชื่อไฟล์ที่ต้องการสร้างเข้าไปแล้วกด Enter

4. จะเข้าสู่โปรแกรมอีดิตเตอร์ ที่พร้อมจะให้ผู้ใช้เขียนโปรแกรม

ผลการทดลอง

เข้าสู่โปรแกรมอีดิตเตอร์

4.1.3 การทดลองคำสั่ง Clear

เป็นการทดลองผลที่เกิดขึ้นจากการใช้คำสั่ง Clear

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนูหรือกดปุ่ม F5 ก็จะไปที่ขั้นตอนที่ 3

2. เลื่อนแถบสีมาที่เมนู File แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Clear แล้วกด Enter

3. โปรแกรมที่แสดงอยู่ในช่องแสดงจะถูกเคลียร์

ผลการทดลอง

ในช่องแสดงโปรแกรมจะว่างเปล่า

4.1.4 การทดลองคำสั่ง Change Dir

เป็นการทดลองผลที่เกิดจากคำสั่ง Change Dir

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

2. เลื่อนแถบสีมาที่เมนู File แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Change Dir แล้วกด Enter

3. จะปรากฏเมนูถามไดเรกทอรี่ถูกต้อง ให้ผู้ใช้พิมพ์เข้าไป

4. จะทำให้ไดเรกทอรี่เปลี่ยนไปตามที่ผู้ใช้ป้อนเข้าไป

ผลการทดลอง

ไดเรกทอรี่เดิมจะเปลี่ยนไปเป็นไดรกทอรี่ใหม่ตามที่ผู้ใช้ป้อนเข้าไป

4.1.5 การทดลองคำสั่ง OS Shell

เป็นการทดลองผลที่เกิดจากคำสั่ง OS Shell

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

2. เลื่อนแถบสีมาที่เมนู File แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง OS Shell แล้วกด Enter

3. จะทำให้ออกจากโปรแกรม MEMDP ไปสู่ดอสชั่วคราว เมื่อผู้ใช้ต้องการกลับเข้ามาที่โปรแกรม MEMDP ให้พิมพ์ Exit ก็จะกลับเข้าสู่โปรแกรม MEMDP

ผลการทดลอง

ทำให้ออกจากโปรแกรม MEMDP ไปสู่ดอสชั่วคราว

4.1.6 การทดลองคำสั่ง Exit

เป็นการทดลองผลจากคำสั่ง Exit

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนูหรือกดปุ่ม ALT-X ก็จะไปที่ขั้นตอนที่ 3

2. เลื่อนแถบสีมาที่เมนู File แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Exit แล้วกด Enter

3. จะทำให้ออกจากโปรแกรม MEMDP ไปสู่ดอส เป็นการสิ้นสุดการทำงานของโปรแกรม MEMDP

ผลการทดลอง

ทำให้จบการทำงานของโปรแกรม MEMDP

4.1.7 การทดลองคำสั่ง Edit

เป็นการทดลองผลจากคำสั่ง Edit โดยที่จะต้องมีโปรแกรมอยู่ในช่องแสดงอยู่ก่อนแล้ว มิฉะนั้นจะใช้คำสั่ง Edit ไม่ได้

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู
2. เลื่อนแถบสีมาที่เมนูคำสั่ง Edit แล้วกด Enter
3. จะเข้าสู่โปรแกรมอีดิตเตอร์

ผลการทดลอง

จะทำให้เข้าสู่โปรแกรมอีดิตเตอร์ เพื่อให้ผู้ใช้ทำการแก้ไขเพิ่มเติมโปรแกรมตามต้องการ

4.1.8 การทดลองคำสั่ง Run

เป็นการทดลองผลที่เกิดจากคำสั่ง Run โดยก่อนที่จะใช้คำสั่ง Run จะต้องมีการโหลดโปรแกรมที่ต้องการสั่งให้แผงวงจรทดลองทำงานและมีการต่อแผงวงจรเข้ากับเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 เสียก่อน

ขั้นตอนการทดลอง

1. ทำตามขั้นตอนการทดลองคำสั่ง Load
2. ทำการเลือกเบอร์ของแรมหรืออีพรอม โดยจะอธิบายอย่างละเอียดในขั้นตอนการทดลองคำสั่ง RAM และ คำสั่ง EPROM
3. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู
4. เลื่อนแถบสีมาที่เมนู Run แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Run แล้วกด Enter
5. แผงวงจรทดลองจะทำงานตามโปรแกรมที่โหลดเข้ามา

6. กด Enter เพื่อออกจากคำสั่ง Run

ผลการทดลอง

แผงวงจรทดลองจะทำงานตามโปรแกรมที่โหลดเข้ามา โดยลักษณะการทำงานของแผงวงจรทดลองจะขึ้นอยู่กับคำสั่ง Single Step, คำสั่ง Break Point และคำสั่ง Trace into ซึ่งเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 จะทำงานเป็นอีมูเลเตอร์

4.1.9 การทดลองคำสั่ง Single Step

เป็นการทดลองผลที่เกิดจากคำสั่ง Single Step โดยก่อนที่จะใช้คำสั่ง Single Step จะต้องมีการโหลดโปรแกรมที่ต้องการสั่งให้แผงวงจรทดลองทำงานและมีการต่อแผงวงจรเข้ากับเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 เสียก่อน

ขั้นตอนการทดลอง

1. ทำตามขั้นตอนการทดลองคำสั่ง Load

2. ทำการเลือกเบอร์ของแรมหรืออีพรอม โดยจะอธิบายอย่างละเอียดในขั้นตอนการทดลองคำสั่ง RAM และ EPROM

3. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

4. เลื่อนแถบสีมาที่เมนู Run แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Single Step แล้วกด Enter

5. แผงวงจรทดลองจะทำงานตามโปรแกรมที่โหลดเข้ามาทีละคำสั่ง โดยจะต้องกดปุ่ม F8 เพื่อให้ทำงานทีละคำสั่ง ซึ่งค่าของหน่วยความจำและรีจิสเตอร์ต่างๆ จะถูกแสดงในช่องแสดงค่าของหน่วยความจำและช่องแสดงค่ารีจิสเตอร์

6. สั่ง Stop เพื่อหยุดการทำงานของคำสั่ง Single Step โดยวิธีสั่ง Stop จะกล่าวถึงในหัวข้อการทดลองคำสั่ง Stop

ผลการทดลอง

แผงวงจรทดลองจะทำงานตามโปรแกรมที่โหลดเข้ามาทีละคำสั่ง ตามจังหวะการกดปุ่ม F8 ซึ่งค่าของหน่วยความจำและรีจิสเตอร์ต่างๆ จะถูกแสดงในช่องแสดงค่าของหน่วยความจำและช่องแสดงค่ารีจิสเตอร์ตามการทำงานของแผงวงจรทดลอง

4.1.10 การทดลองคำสั่ง Trace into

เป็นการทดลองผลที่เกิดจากคำสั่ง Trace into โดยก่อนที่จะใช้คำสั่ง Trace into จะต้องมีการโหลดโปรแกรมที่ต้องการสั่งให้แผงวงจรทดลองทำงานและมีการต่อแผงวงจรเข้ากับเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 เสียก่อน

ขั้นตอนการทดลอง

1. ทำตามขั้นตอนการทดลองคำสั่ง Load

2. ทำการเลือกเบอร์ของแรม โดยจะอธิบายอย่างละเอียดในขั้นตอนการทดลองคำสั่ง RAM

3. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

4. เลื่อนแถบสีมาที่เมนู Run แล้วกด Enter.จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Trace into แล้วกด Enter

5. แผงวงจรทดลองจะทำงานตามโปรแกรมที่โหลดเข้ามาทีละคำสั่ง โดยจะต้องกดปุ่ม F7 เพื่อให้ทำงานทีละคำสั่ง ซึ่งค่าของหน่วยความจำและรีจิสเตอร์ต่างๆ จะถูกแสดงในช่องแสดงค่าของหน่วยความจำและช่องแสดงค่ารีจิสเตอร์

6. สั่ง Stop เพื่อหยุดการทำงานของคำสั่ง Trace into โดยวิธีสั่ง Stop จะกล่าวถึงในหัวข้อการทดลองคำสั่ง Stop

ผลการทดลอง

แผงวงจรทดลองจะทำงานตามโปรแกรมที่โหลดเข้ามาทีละคำสั่งตามจังหวะการกดปุ่ม F7 ซึ่งมีการแสดงค่าของหน่วยความจำและรีจิสเตอร์ต่างๆ ในช่องแสดงค่าของหน่วยความจำและช่องแสดงค่ารีจิสเตอร์ตามการทำงานของแผงวงจรทดลอง

4.1.11 การทดลองคำสั่ง Stop

เป็นการทดลองผลที่เกิดจากคำสั่ง Stop ซึ่งคำสั่ง Stop นี้จะต้องใช้หลังจากการรันโปรแกรมสิ้นสุดแล้ว

ขั้นตอนการทดลองคำสั่ง Stop

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

2. เลื่อนแถบสีมาที่เมนูคำสั่ง Stop แล้วกด Enter

ผลการทดลอง

แผงวงจรทดลองจะหยุดทำงาน

4.1.12 การทดลองคำสั่ง Break Point

เป็นการทดลองคำสั่ง Break Point โดยก่อนที่จะใช้คำสั่ง Break Point จะต้องมีการโหลดโปรแกรมที่ต้องการสั่งให้แผงวงจรทดลองทำงานและมีการต่อแผงวงจรเข้ากับเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 เสียก่อน

ขั้นตอนการทดลอง

1. ทำตามขั้นตอนการทดลองคำสั่ง Load

2. ทำการเลือกเบอร์ของแรม โดยจะอธิบายอย่างละเอียดในขั้นตอนการทดลองคำสั่ง RAM 3. กดปุ่ม ESC จะปรากฏแถบสีที่ช่องแสดงโปรแกรม

4. เลือกจุดเบรคที่ต้องการ โดยกด Enter

5. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

6. เลื่อนแถบสีมาที่เมนู Run แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Run แล้วกด Enter

7. แผงวงจรทดลองจะทำงานตามโปรแกรมที่โหลดเข้ามาทีละคำสั่ง โดยจะต้องกดปุ่ม Enter เพื่อให้ทำงานทีละคำสั่ง ซึ่งค่าของหน่วยความจำและรีจิสเตอร์ต่างๆ จะถูกแสดงในช่องแสดงค่าของหน่วยความจำและช่องแสดงค่ารีจิสเตอร์

8. สั่ง Stop เพื่อหยุดการทำงานของคำสั่ง Run

ผลการทดลอง

แผงวงจรทดลองจะทำงานตามโปรแกรมที่โหลดเข้ามาจนถึงจุดที่ตั้งเอาไว้ก็จะหยุดเพื่อรอการกดปุ่ม Enter ให้ทำงานต่อ ซึ่งมีการแสดงค่าของหน่วยความจำและรีจิสเตอร์ต่างๆ ในช่องแสดงค่าของหน่วยความจำและช่องแสดงค่ารีจิสเตอร์ตามการทำงานของแผงวงจรทดลอง

4.1.13 การทดลองคำสั่ง Display Break Point

เป็นการทดลองผลที่เกิดจากคำสั่ง Display Break Point

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

2. เลื่อนแถบสีมาที่เมนู Break Point แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Display Break Point แล้วกด Enter

ผลการทดลอง

จะปรากฏช่องแสดงตำแหน่งจุดเบรคที่ตั้งไว้

4.1.14 การทดลองคำสั่ง Clear Break Point

เป็นการทดลองผลที่เกิดจากคำสั่ง Clear Break Point

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

2. เลื่อนแถบสีมาที่เมนู Break Point แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Clear Break Point แล้วกด Enter

ผลการทดลอง

จะทำการยกเลิกจุดเบรคที่ตั้งเอาไว้

4.1.15 การทดลองคำสั่ง Break Point Set up

เป็นการทดลองผลที่เกิดจากคำสั่ง Break Point Set up โดยก่อนที่จะใช้คำสั่ง Break Point Set up จะต้องมีการโหลดโปรแกรมที่ต้องการสั่งให้แผงวงจรทดลองทำงานและมีการต่อแผงวงจรเข้ากับเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 เสียก่อน

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

2. เลื่อนแถบสีมาที่เมนู Break Point แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Break Point Set up แล้วกด Enter จะปรากฏเมนูย่อยที่มีอยู่ 2 คำสั่ง คือ Fix Break Point และ Quene Break Point

3. เลื่อนแถบสีไปที่ยังคำสั่งที่ต้องการแล้วกดปุ่ม Space Bar เพื่อเลือกคำสั่งที่ต้องการ

4. กดปุ่ม ESC 2 ครั้ง เพื่อออกจากเมนู Break Point

5. ทำการทดลองตามคำสั่ง Break Point

ผลการทดลอง

1. เมื่อเลือกคำสั่ง Fix Break Point แผงวงจรทดลองจะทำงานตามโปรแกรมที่โหลดเข้ามาจนถึงจุดที่ตั้งเอาไว้ก็จะหยุด เพื่อรอการกดปุ่ม Enter ให้ทำงานต่อ โดยจะทำงานตามลำดับของการเลือกจุดเบรค ซึ่งมีการแสดงค่าของหน่วยความจำและรีจิสเตอร์ต่างๆ ในช่องแสดงค่าของหน่วยความจำและช่องแสดงค่ารีจิสเตอร์ตามการทำงานของแผงวงจรทดลอง

2. เมื่อเลือกคำสั่ง Quene Break Point แผงวงจรทดลองจะทำงานตามโปรแกรมที่โหลดเข้ามาจนถึงจุดที่ตั้งเอาไว้ก็จะหยุด เพื่อรอการกดปุ่ม Enter ให้ทำงานต่อ โดยจะทำงานตามลำดับของแอดเดรสที่ตั้งจุดเบรคเอาไว้จากน้อยไปมาก ซึ่งมีการแสดงค่าของหน่วยความจำและรีจิสเตอร์ต่างๆ ในช่องแสดงค่าของหน่วยความจำและช่องแสดงค่ารีจิสเตอร์ตามการทำงานของแผงวงจรทดลอง

4.1.16 การทดลองคำสั่ง EPROM

เป็นการทดลองผลที่เกิดจากคำสั่ง EPROM ซึ่งคำสั่งนี้จะต้องเรียกใช้ทุกครั้งก่อนที่จะมีการรันโปรแกรม เพื่อให้เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ทำงานได้ถูกต้อง
และต้องต่อแผงวงจรเข้ากับเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 เสียก่อน

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

2. เลื่อนแถบสีมาที่เมนู Set up แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง EPROM แล้วกด Enter

3. จะปรากฏเมนูย่อย ให้เลื่อนแถบสีไปยังเบอร์อีพรอมที่ต้องการ แล้วกด Enter

4. จะปรากฏรูปแสดงการเซ็ทดิปสวิทช์ ให้ทำการเซ็ทดิปสวิทช์ที่เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51

5. ทำตามขั้นตอนการทดลองคำสั่ง Run

ผลการทดลอง

จากขั้นตอนที่ 3 ถ้าหากเลือกเบอร์ของอีพรอมไม่ถูกต้องกับที่ใช้งานจริงจะมีผลทำให้โปรแกรมไม่สามารถรันและแผงวงจรทดลองไม่สามารถทำงานได้ หรืออาจทำให้เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 เสียหายได้

4.1.17 การทดลองคำสั่ง RAM

เป็นการทดลองผลที่เกิดจากคำสั่ง RAM ซึ่งคำสั่งนี้จะต้องเรียกใช้ทุกครั้งก่อนที่จะมีการรันโปรแกรม เพื่อให้เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ทำงานได้ถูกต้อง
และต้องต่อแผงวงจรเข้ากับเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 เสียก่อน

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู
2. เลื่อนแถบสีมาที่เมนู Set up แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง RAM แล้วกด Enter
3. จะปรากฏเมนูย่อย ให้เลื่อนแถบสีไปยังเบอร์แรมที่ต้องการ แล้วกด Enter
4. จะปรากฏรูปแสดงการเซ็ทดิปสวิทช์ ให้ทำการเซ็ทดิปสวิทช์ที่เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51
5. ทำตามขั้นตอนการทดลองคำสั่ง Run

ผลการทดลอง

จากขั้นตอนที่ 3 ถ้าหากเลือกเบอร์ของแรมไม่ถูกต้องกับที่ใช้งานจริงจะมีผลทำให้โปรแกรมไม่สามารถรันและแผงวงจรทดลองไม่สามารถทำงานได้ หรืออาจทำให้เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 เสียหายได้

4.1.18 การทดลองคำสั่ง Internal RAM

เป็นการทดลองผลที่เกิดจากคำสั่ง Internal RAM

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

2. เลื่อนแถบสีมาที่เมนู Set up แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Internal RAM แล้วกด Enter

3. จะปรากฏเมนูถามตำแหน่งของหน่วยความจำภายใน ให้ป้อนตำแหน่งที่ต้องการแล้วกด Enter

ผลการทดลอง

ช่องแสดงค่าของหน่วยความจำภายในจะแสดงตำแหน่งตามที่ผู้ใช้ป้อนเข้าไป

4.1.19 การทดลองคำสั่ง External RAM

เป็นการทดลองผลที่เกิดจากคำสั่ง External RAM

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

2. เลื่อนแถบสีมาที่เมนู Set up แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง External RAM แล้วกด Enter

3. จะปรากฏเมนูถามตำแหน่งของหน่วยความจำภายนอก ให้ป้อนตำแหน่งที่ต้องการแล้วกด Enter

ผลการทดลอง

ช่องแสดงค่าตำแหน่งของหน่วยความจำภายในจะเปลี่ยนไปเป็นช่องแสดงค่าของหน่วยความจำภายนอกตามที่ผู้ใช้ป้อนเข้าไป

4.1.20 การทดลองคำสั่ง Baud Rate

เป็นการทดลองผลที่เกิดจากคำสั่ง Baud Rate

ขั้นตอนการทดลอง

1. กดปุ่ม F10 จะปรากฏแถบสีที่เมนู

2. เลื่อนแถบสีมาที่เมนู Set up แล้วกด Enter จะปรากฏเมนูย่อยให้เลื่อนแถบสีไปที่คำสั่ง Baud Rate แล้วกด Enter

3. จะปรากฏเมนูแสดงค่าพารามิเตอร์ปัจจุบันของพอร์ทสื่อสารอนุกรม และค่าของพารามิเตอร์ที่สามารถเลือกได้ โดยการกดอักษรที่นำหน้าค่าพารามิเตอร์นั้น

ผลการทดลอง

จากขั้นตอนที่ 3 ถ้าหากว่าผู้ใช้ทำการเลือกค่าพารามิเตอร์ไม่ถูกต้อง จะทำให้ไม่สามารถใช้เครื่องจำลองหน่วยความจำและทดสอบโปรแกรมได้

4.2 การทดลองโปรแกรม CMEMDP

จากการทดลองโปรแกรม MEMDP ปรากฏว่าสามารถทำงานร่วมกับโปรแกรม MEMDP ได้โดยไม่มีปัญหาเกิดขึ้น

4.3 การทดลองเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51

การทดลองโปรแกรม MEMDP ว่าสามารถที่จะทำงานร่วมกับเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ได้หรือไม่นั้น สามารถทำได้โดยการใช้เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ทดลองกับโปรแกรมทดสอบดังต่อไปนี้ คือ โปรแกรมติดต่อกับพอร์ทอินพุทเอาท์พุท ซึ่งทำการทดลองโดยใช้คำสั่ง Break Point, โปรแกรมไฟวิ่ง ซึ่งทำการทดลองโดยใช้คำสั่ง Single Step และโปรแกรมตรวจสอบเงื่อนไขแล้วกระโดด ซึ่งทำการทดลองโดยใช้คำสั่ง Auto Trace ซึ่งผลการทดลองใช้งานเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ปรากฏว่าสามารถใช้งานได้ดี โดยมีผลการทดลองของโปรแกรมต่างๆ ดังนี้

4.3.1 ผลการทดลองโปรแกรมที่ติดต่อกับพอร์ทอินพุทเอาท์พุท

```
        ORG     0000H
MAIN:   MOV     R0,#00H
        MOV     P1,R0       <==========Break Point จุดที่ 1
        MOV     R1,#0FFH    <==========Break Point จุดที่ 2
        INC     R0
        MOV     P1,R0       <==========Break Point จุดที่ 3
        SJMP    MAIN
        END
```

เมื่อทำการเลือกการทำงานของโปรแกรมแบบ Break Point และทำการสั่งรันโปรแกรมเมื่อพบจุดเบรคที่ 1 จะทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 ซึ่งได้ค่าดังนี้

ผลของคำสั่ง MOV P1,R0

R0 = 00 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 07 A = 00

ทำการสั่งรันโปรแกรมโดยทำการกดคีย์ ENTER พบจุดเบรคที่ 2 ผลการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง MOV R1,#0FFH

R0 = 00 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = 00 P2 = FF P3 = FF
SP = 07 A = 00

ทำการสั่ง Run โปรแกรมโดยทำการกดคีย์ ENTER พบจุด Break Point จุดที่ 3 ผลการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง MOV P1,R0

R0 = 01 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = 00 P2 = FF P3 = FF
SP = 07 A = 00

เมื่อทำการรันโปรแกรมแบบ Break Point จนครบทุกจุดค่าของจุดเบรคก็จะทำการเคลียร์ค่าออกไปแล้วปล่อยให้โปรแกรมที่นำมาทดสอบทำงานตามปกติ

4.3.2 ผลการทดลองโปรแกรมไฟวิ่ง

```
        ORG   0000H
        MOV   A,#80H
MAIN:   MOV   P1,A
        RR    A
        SJMP  MAIN
        END
```

เมื่อทำการรันโปรแกรมโดยเลือกการรันแบบซิงเกิลเสต็ป แถบสีจะไปปรากฏอยู่ที่แอดเดรส 0000H ซึ่งเป็นจุดเริ่มต้นของการรันแบบซิงเกิลเสต็ป ซึ่งการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง ORG 0000H

R0 = 00 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 07 A = 00

ทำการกดคีย์ F8 เพื่อทำการรันโปรแกรมต่อไป ซึ่งการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง MOV A,8OH

R0 = 00 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = 00 P2 = FF P3 = FF
SP = 07 A = 80

ทำการกดคีย์ F8 เพื่อทำการรันโปรแกรมต่อไป ซึ่งการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง MOV P1,A

R0 = 00 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = 80 P2 = FF P3 = FF
SP = 07 A = 80

ทำการกดคีย์ F8 เพื่อทำการรันโปรแกรมต่อไป ซึ่งการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง RR A

R0 = 00 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = 80 P2 = FF P3 = FF
SP = 07 A = 01

ทำการกดคีย์ F8 เพื่อทำการรันโปรแกรมต่อไป ซึ่งการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง SJMP MAIN

R0 = 00 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = 00 P2 = FF P3 = FF
SP = 07 A = 01

โปรแกรมจะทำงานวนลูปไปเรื่อยๆไม่มีที่สิ้นสุด จึงทำการรันโปรแกรมทดสอบแบบซิงเกิลเสต็ปเพียงเท่านี้

4.3.3 ผลการทดลองเกี่ยวกับการตรวจสอบเงื่อนไขแล้วกระโดด

```
        ORG     0000H
        MOV     A,#07H
        MOV     R0,#6
MAIN:   PUSH    ACC
        INC     R0
```

```
        ANL     A,R0
        JNZ     M_1
        CPL     P1.1
        CPL     P1.1
        CPL     P1.1
        CPL     P1.1
        SJMP    $
M_1 :   POP     ACC
        LJMP    MAIN
        END
```

เมื่อทำการรันโปรแกรมแบบ Auto Trace โดยกำหนดจุดเริ่มต้นที่แอดเดรส 0000H ทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง ORG 0000H

R0 = 00 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 07 A = 00

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง MOV A,#07H

R0 = 00 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 07 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง MOV R0,#6H

R0 = 06 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 07 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง PUSH ACC

R0 = 06 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง INC R0

R0 = 07 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง ANL A,R0

R0 = 07 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง JNZ M_1

R0 = 07 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง POP ACC

R0 = 07 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 07 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่า

ผลของคำสั่ง LJMP MAIN

R0 = 07 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = FF P2 = FF P3 = FF
SP = 07 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง PUSH ACC

R0 = 07 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00

P0 = FF P1 = FF P2 = FF P3 = FF

SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง INC R0

R0 = 08 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00

P0 = FF P1 = FF P2 = FF P3 = FF

SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง ANL A,R0 (ทำให้เกิด ZERO)

R0 = 08 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00

P0 = FF P1 = FF P2 = FF P3 = FF

SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง JNZ M_1

R0 = 08 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00

P0 = FF P1 = FF P2 = FF P3 = FF

SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง CPL P1.1

R0 = 08 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1.1 = 00 P2 = FF P3 = FF
SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง CPL P1.1

R0 = 08 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1.1 = 1 P2 = FF P3 = FF
SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง CPL P1.1

R0 = 08 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = 00 P2 = FF P3 = FF
SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง CPL P1.1

R0 = 08 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00
P0 = FF P1 = 1 P2 = FF P3 = FF
SP = 08 A = 07

หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง SJMP $

R0 = 08 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00

P0 = FF P1 = 1 P2 = FF P3 = FF

SP = 08 A = 07

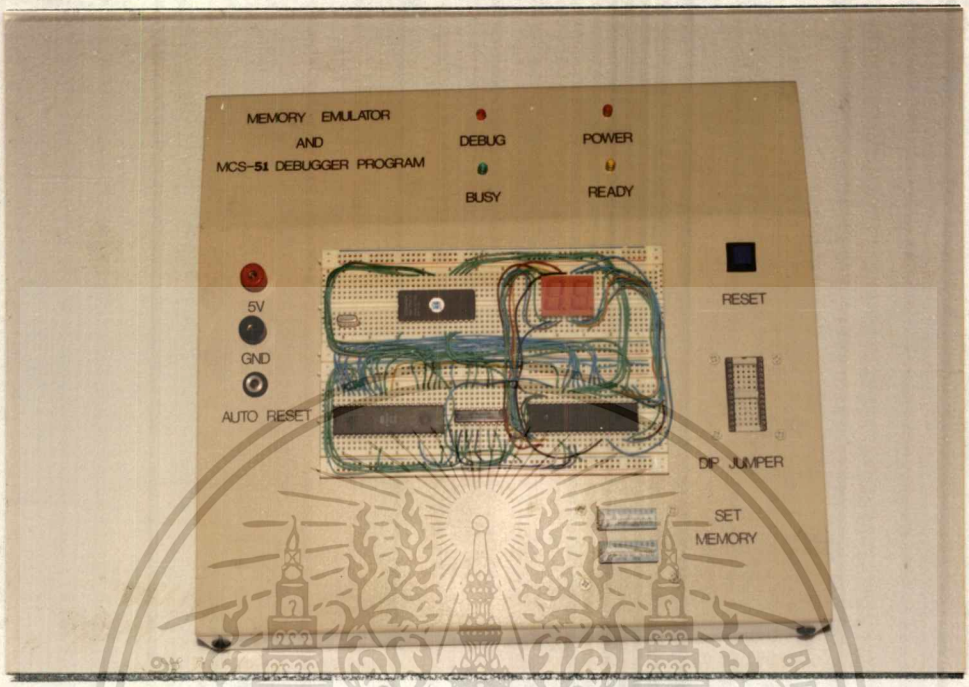
หน่วงเวลาประมาณ 1 วินาทีแล้วทำคำสั่งถัดไปทำการอ่านค่ารีจิสเตอร์ต่างๆ ภายใน MCS-51 จะได้ค่าดังนี้

ผลของคำสั่ง END

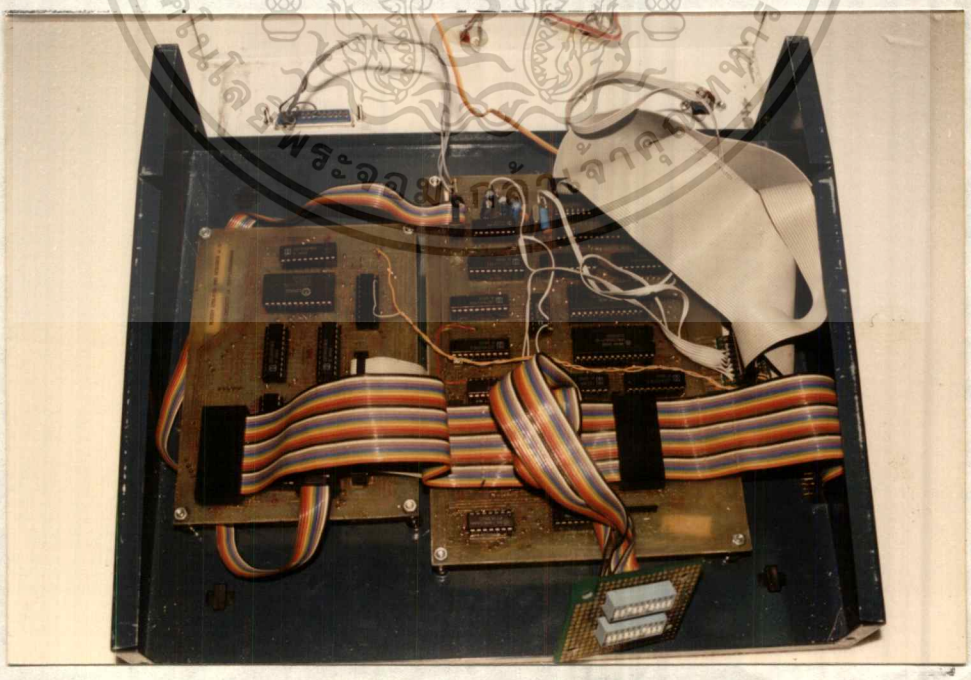
R0 = 07 R1 = 00 R2 = 00 R3 = 00 R4 = 00 R5 = 00 R6 = 00 R7 = 00

P0 = FF P1 = 1 P2 = FF P3 = FF

SP = 08 A = 07



รูปที่ 4.2 ตัวเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51



รูปที่ 4.3 ภายในเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51

บทที่ 5
สรุปและวิจารณ์

5.1 สรุป

เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ได้สร้างขึ้นเพื่อศึกษาการประยุกต์ใช้งานด้านการออกแบบและแก้ไข รวมถึงการพัฒนาโปรแกรมของซีพียูตระกูล MCS-51 โดยขอบเขตที่วางไว้ในขั้นต้นคือ การสร้างเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ที่ใช้งานได้จริงและสามารถติดต่อกับผู้ใช้โดยผ่านทางพอร์ตสื่อสารอนุกรมของเครื่องไมโครคอมพิวเตอร์ได้

จากการที่ได้ศึกษาและทดลองสร้าง ปรากฏว่าผลที่ได้รับอยู่ในระดับที่น่าพอใจ กล่าวคือ การทำงานของเครื่องสามารถที่จะตั้งจุดเบรคได้ 10 จุดพร้อมทั้งแสดงค่าที่ตั้งไว้ และสามารถรันโปรแกรมแบบซิงเกิลเสต็ปได้

การทำงานของเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 สามารถอ่านข้อมูลจากบอร์ดทดลองโดยได้ทราบถึงค่าที่แท้จริงที่เกิดขึ้นภายในบอร์ด ข้อดีอีกอย่างหนึ่งก็คือเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 สามารถที่จะตั้งจุดเบรคได้ติดๆ กันโดยไม่ทำให้โปรแกรมที่นำมาทดสอบรวมทั้งฮาร์ดแวร์จะไม่เกิดความเสียหายแต่อย่างใด

อีกทั้งยังสามารถอ่านค่าจากพอร์ตภายนอกเข้ามาเพื่อตรวจสอบค่าที่อ่านเข้ามาว่ามีค่าตามที่ต้องการหรือไม่ ซึ่งในโปรแกรมซิมมูเลเตอร์ทั่วๆ ไปไม่สามารถทำได้

5.2 ข้อเสีย

1. การทำงานของเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ต้องมีการตอบรับกันระหว่างเครื่องไมโครคอมพิวเตอร์กับเครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS-51 ถ้าเกิดการตอบรับผิดพลาดจะทำให้เครื่อง คอมพิวเตอร์หยุดการทำงานได้

วิธีแก้ไข พยายามกำหนดการตอบรับให้ใกล้เคียงและถูกต้องมากที่สุด โดยใช้อัตราความเร็วในการรับส่งข้อมูลที่เท่ากัน

2.ในการรันโปรแกรมแบบซิงเกิลสเต็ป จะมีปัญหากับคำสั่งที่เกี่ยวกับการเปรียบเทียบแล้วกระโดด เช่นคำสั่ง CJNE, JC เป็นต้น เนื่องจากแฟลกที่นำมาตรวจสอบไม่สามารถอ่านค่าที่ถูกต้องมาจากบอร์ดทดลองได้

วิธีแก้ไข เขียนคอมไพล์เลอร์ขึ้นมาใหม่แต่ก็ยังทำได้ไม่ดีเท่าที่ควร

3. ขนาดของหน่วยความจำมีขนาดไม่เป็นไปตามที่ได้กำหนดไว้ ยกตัวอย่างเช่น แรม เบอร์ 6264 จะต้องมีขนาดของหน่วยความจำคือ 8 กิโลไบท์ แต่จะใช้ได้จริงเพียง 6 กิโลไบท์เท่านั้น เพราะอีก 2 กิโลไบท์สุดท้ายจะนำไปเก็บโปรแกรมอ่านค่าข้อมูลภายในบอร์ดที่นำมาทดลองและพัฒนา ซึ่งเมื่อทำการโหลดโปรแกรมที่มีขนาดใหญ่เกินหน่วยความจำก็จะทำให้โปรแกรมเกิดการเสียหายขึ้นได้

วิธีแก้ไข ควรจะออกแบบส่วนที่ใช้เก็บโปรแกรมอ่านค่าข้อมูล โดยใช้การอ้างพอร์ต ก็จะแก้ปัญหาการเสียหน่วยความจำที่มีขนาด 2 กิโลไบท์ลงได้ ซึ่งจะทำให้ขนาดของหน่วยความจำมีขนาดตามที่กำหนด

5.3 แนวทางการพัฒนา

ส่วนของปัญหาที่เกิดขึ้นหากสามารถหาข้อมูลของคำสั่งอย่างละเอียดจะทำให้มีความสมบูรณ์มากขึ้นได้

- ในหน่วยของการรันโปรแกรมแบบซิงเกิลเสต็ป ควรจะมีการใช้แฟลกที่มีอยู่ภายใน PSW ของ MCS - 51 มาใช้เพื่อให้คำสั่งเกี่ยวกับการเปรียบเทียบแล้วกระโดดมีความสมบูรณ์มากที่สุด

- รูปร่างของหน้าปัทม์และวงจร หากสามารถทำขึ้นใหม่ให้มีขนาดเล็กกว่าเก่าและสะดวกในการใช้มากกว่าที่เป็นอยู่ ก็จะสามารถอำนวยความสะดวกในการใช้งานมากขึ้น

- เครื่องจำลองหน่วยความจำและทดสอบโปรแกรม MCS - 51 ยังไม่สามารถตรวจสอบการส่งค่าออกทางพอร์ตเอาต์พุตได้ ถ้าได้มีการพัฒนาจุดนี้ได้ก็จะมีประโยชน์อย่างมาก เพราะสามารถตรวจสอบค่าที่ส่งออกทางพอร์ตเอาต์พุตว่าถูกต้องตามที่ต้องการหรือไม่

ภาคผนวก ก. โปรแกรม CMEMDP

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

  1  1800             MEM_2   EQU  1800H
  2  1A00             TB_DATA EQU  1A00H
  3
  4  0000             ORG  0000H ;ADDRESS 1800H
  5
  6                 ;########### START INTERNAL_RAM ##################
  7  0000  90 1A 00                MOV  DPTR,#TB_DATA
  8  0003  E5 7F                   MOV  A,7FH      ;ACC <= R0
  9  0005  F0                      MOVX @DPTR,A
 10  0006  05 82                   INC  DPL
 11  0008  E5 7E                   MOV  A,7EH      ;ACC <= R1
 12  000A  F0                      MOVX @DPTR,A
 13  000B  05 82                   INC  DPL
 14  000D  78 02                   MOV  R0,#02H
 15  000F  E6          SAVE_DATA:  MOV  A,@R0
 16  0010  F0                S_0:  MOVX @DPTR,A
 17  0011  08                      INC  R0
 18  0012  05 82                   INC  DPL
 19  0014  B8 80 F8                CJNE R0,#80H,SAVE_DATA
 20
 21                ;########### END INTERNAL_RAM ###################
 22
 23  0017  75 82 00                MOV  DPL,#00H     ;9B00H  FOR SFR
 24  001A  05 83                   INC  DPH
```

```
25 001C E5 7B        MOV A,7BH       ;ACC <= ACC
26 001E F0           MOVX @DPTR,A
27 001F 05 82        INC DPL
28 0021 E5 F0        MOV A,B
29 0023 F0           MOVX @DPTR,A
30 0024 05 82        INC DPL
31 0026 E5 7C        MOV A,7CH       ;ACC <= DPH
32 0028 F0           MOVX @DPTR,A
33 0029 05 82        INC DPL
34 002B E5 7D        MOV A,7DH       ;ACC <= DPL
35 002D F0           MOVX @DPTR,A
36 002E 05 82        INC DPL
37 0030 E5 80        MOV A,80H       ;ACC <= P0
38 0032 F0           MOVX @DPTR,A
39 0033 05 82        INC DPL
40 0035 E5 90        MOV A,90H       ;ACC <= P1
41 0037 F0           MOVX @DPTR,A
42 0038 05 82        INC DPL
43 003A E5 A0        MOV A,0A0H      ;ACC <= P2
44 003C F0           MOVX @DPTR,A
45 003D 05 82        INC DPL
46 003F E5 B0        MOV A,0B0H      ;ACC <= P3
47 0041 F0           MOVX @DPTR,A
48 0042 05 82        INC DPL
49 0044 E5 99        MOV A,99H       ;ACC <= SBUF
50 0046 F0           MOVX @DPTR,A
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
 51 0047 05 82          INC  DPL
 52 0049 E5 81          MOV  A,SP
 53 004B 94 0A          SUBB A,#0AH
 54 004D F0             MOVX @DPTR,A
 55 004E 05 82          INC  DPL
 56 0050 E5 8C          MOV  A,TH0
 57 0052 F0             MOVX @DPTR,A
 58 0053 05 82          INC  DPL
 59 0055 E5 8A          MOV  A,TL0
 60 0057 F0             MOVX @DPTR,A
 61 0058 05 82          INC  DPL
 62 005A E5 8D          MOV  A,8DH      ;ACC <= TH1
 63 005C F0             MOVX @DPTR,A
 64 005D 05 82          INC  DPL
 65 005F E5 8B          MOV  A,8BH      ;ACC <= TL1
 66 0061 F0             MOVX @DPTR,A
 67 0062 05 82          INC  DPL
 68 0064 E5 A8          MOV  A,0A8H     ;ACC <= IE
 69 0066 F0             MOVX @DPTR,A
 70 0067 05 82          INC  DPL
 71 0069 E5 B8          MOV  A,0B8H     ;ACC <= IP
 72 006B F0             MOVX @DPTR,A
 73 006C 05 82          INC  DPL
 74 006E E5 7A          MOV  A,7AH      ;ACC <= PSW FROM PUSH
```

```
75  0070  F0                    MOVX @DPTR,A
76  0071  05 82                 INC  DPL
77  0073  E5 98                 MOV  A,98H      ;ACC <= SCON
78  0075  F0                    MOVX @DPTR,A
79  0076  05 82                 INC  DPL
80  0078  E5 88                 MOV  A,88H      ;ACC <= TCON
81  007A  F0                    MOVX @DPTR,A
82  007B  05 82                 INC  DPL
83  007D  E5 89                 MOV  A,89H      ;ACC <= TMOD
84  007F  F0                    MOVX @DPTR,A
85              ;############## SEND_FLAG ################
86  0080  05 82                 INC  DPL
87  0082  E5 78                 MOV  A,78H   ;Z
88  0084  F0                    MOVX @DPTR,A
89  0085  05 82                 INC  DPL
90  0087  E5 79                 MOV  A,79H   ;C
91  0089  F0                    MOVX @DPTR,A
92              ;############## END FLAG ################
93              ;############  END SFR ################
94                              ;MOV  DPL,#0
95                              ;INC  DPH
96              ;############  END EXTERNAL_RAM #######
97  008A  D8 FE       S_1:      DJNZ R0,$
98  008C  D9 FC                 DJNZ R1,S_1
99  008E  D8 FE       S_2:      DJNZ R0,$
100 0090  D9 FC                 DJNZ R1,S_2
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
101 0092 D8 FE      S_3:        DJNZ R0,$
102 0094 D9 FC                  DJNZ R1,S_3
103                 ;S_4:       DJNZ R0,$
104                     ;       DJNZ R1,S_4
105                 ;S_5:       DJNZ R0,$
106                     ;       DJNZ R1,S_5
107                 ;S_6:       DJNZ R0,$
108                     ;       DJNZ R1,S_6
109 0096 22             RET
110               ;############### END ####################
111
112 1400                        ORG 1400H
113 1400 C0 D0      DBG:        PUSH PSW
114 1402 70 05                  JNZ DBG_1
115 1404 75 78 FF               MOV 78H,#0FFH
116 1407 80 03                  SJMP DBG_2
117 1409 75 78 00   DBG_1:      MOV 78H,#0
118                 DBG_2:     ;PUSH PSW
119 140C C0 E0                  PUSH ACC
120 140E C0 83                  PUSH DPH
121 1410 C0 82                  PUSH DPL
122 1412 C0 01                  PUSH 1
123 1414 C0 00                  PUSH 0
124 1416 D0 7F                  POP 7FH
```

```
125  1418  D0 7E          POP  7EH
126  141A  D0 7D          POP  7DH
127  141C  D0 7C          POP  7CH
128  141E  D0 7B          POP  7BH
129  1420  D0 7A          POP  7AH
130  1422  C0 7A          PUSH 7AH
131  1424  C0 7B          PUSH 7BH
132  1426  C0 7C          PUSH 7CH
133  1428  C0 7D          PUSH 7DH
134  142A  C0 7E          PUSH 7EH
135  142C  C0 7F          PUSH 7FH
136
137  142E  A8 81          MOV  R0,SP
138  1430  E8             MOV  A,R0
139  1431  94 07          SUBB A,#7
140  1433  F8             MOV  R0,A
141  1434  E6             MOV  A,@R0
142  1435  94 03          SUBB A,#3
143  1437  F6             MOV  @R0,A
144
145  1438  78 00          MOV  R0,#0
146  143A  79 00          MOV  R1,#0
147  143C  D8 FE    D_1:  DJNZ R0,$
148  143E  D9 FC          DJNZ R1,D_1
149  1440  D8 FE    D_2:  DJNZ R0,$
150  1442  D9 FC          DJNZ R1,D_2
```

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

 151  1444  D8 FE      D_3:        DJNZ R0,$
 152  1446  D9 FC                  DJNZ R1,D_3
 153  1448  D8 FE      D_4:        DJNZ R0,$
 154  144A  D9 FC                  DJNZ R1,D_4
 155  144C  D8 FE      D_5:        DJNZ R0,$
 156  144E  D9 FC                  DJNZ R1,D_5
 157  1450  D8 FE      D_6:        DJNZ R0,$
 158  1452  D9 FC                  DJNZ R1,D_6
 159  1454  D8 FE      D_7:        DJNZ R0,$
 160  1456  D9 FC                  DJNZ R1,D_7
 161  1458  D8 FE      D_8:        DJNZ R0,$
 162  145A  D9 FC                  DJNZ R1,D_8
 163                  ;D_9:       DJNZ R0,$
 164                  ;           DJNZ R1,D_9
 165                  ;D_A:       DJNZ R0,$
 166                  ;           DJNZ R1,D_A
 167                  ;D_B:       DJNZ R0,$
 168                  ;           DJNZ R1,D_B
 169                  ;D_C:       DJNZ R0,$
 170                  ;           DJNZ R1,D_C
 171                  ;D_D:       DJNZ R0,$
 172                  ;           DJNZ R1,D_D
 173                  ;D_E:       DJNZ R0,$
 174                  ;           DJNZ R1,D_E
```

```
175                 ;D_F:           DJNZ R0,$
176                 ;               DJNZ R1,D_F
177                 ;D_10:          DJNZ R0,$
178                 ;               DJNZ R1,D_10
179 145C 12 18 00                   LCALL MEM_2
180 145F D0 00      DBG_END:        POP 0
181 1461 D0 01                      POP 1
182 1463 D0 82                      POP DPL
183 1465 D0 83                      POP DPH
184 1467 D0 E0                      POP ACC
185 1469 D0 D0                      POP PSW
186 146B 22                         RET
187                 END
```

TURBO51. 8051 Assembler Ver 1.00

Sequence	Symbol name	Value
1	ACC	= 00E0
2	B	= 00F0
34	DBG	1400
35	DBG_1	1409
36	DBG_2	140C
45	DBG_END	145F
3	DPH	= 0083
4	DPL	= 0082
37	D_1	143C
38	D_2	1440
39	D_3	1444
40	D_4	1448
41	D_5	144C
42	D_6	1450
43	D_7	1454
44	D_8	1458
5	IE	= 00A8
6	IP	= 00B8
27	MEM_2	= 1800
7	P0	= 0080
8	P1	= 0090
9	P2	= 00A0
10	P3	= 00B0
11	PCAP2H	= 00CB

12	PCAP2L	= 00CA
13	PCON	= 0087
14	PSW	= 00D0
29	SAVE_DATA	000F
15	SBUF	= 0099
16	SCON	= 0098
17	SP	= 0081
30	S_0	0010
31	S_1	008A
32	S_2	008E
33	S_3	0092
18	T2CON	= 00C8
28	TB_DATA	= 1A00
19	TCON	= 0088
20	TH0	= 008C
21	TL0	= 008A
22	TH1	= 008D
23	TL1	= 008B
24	TH2	= 00CD
25	TL2	= 00CC
26	TMOD	= 0089

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
 1 0000                    ORG  0000H
 2 0000 C2 90        DBG:      CLR  P1.0
 3 0002 C0 E0                  PUSH ACC
 4 0004 C0 00                  PUSH 0
 5 0006 C0 83                  PUSH DPH
 6 0008 C0 82                  PUSH DPL
 7
 8 000A A8 81                   MOV  R0,SP
 9 000C E8                      MOV  A,R0
10 000D 94 05                   SUBB A,#5
11 000F F8                      MOV  R0,A
12 0010 E6                      MOV  A,@R0
13 0011 94 26                   SUBB A,#38
14 0013 F6                      MOV  @R0,A
15
16 0014 75 7E 00                MOV  7EH,#0
17 0017 75 7F 00                MOV  7FH,#0
18 001A D5 7E FD      D_1:      DJNZ 7EH,$
19 001D D5 7F FA                DJNZ 7FH,D_1
20 0020 D5 7E FD      D_2:  DJNZ 7EH,$
21 0023 D5 7F FA                DJNZ 7FH,D_2
22 0026 D5 7E FD      D_3: DJNZ 7EH,$
23 0029 D5 7F FA                DJNZ 7FH,D_3
24 002C D5 7E FD      D_4:  DJNZ 7EH,$
```

```
25  002F  D5 7F FA        DJNZ 7FH,D_4
26  0032  D0 82           POP  DPL
27  0034  D0 83           POP  DPH
28  0036  D0 00           POP  0
29  0038  D0 E0           POP  ACC
30  003A  85 7D D0        MOV  PSW,7DH
31  003D  22              RET
32                        END
```

TURBO51. 8051 Assembler Ver 1.00

Sequence	Symbol name	Value
1	ACC	= 00E0
2	B	= 00F0
27	DBG	0000
3	DPH	= 0083
4	DPL	= 0082
28	D_1	001A
29	D_2	0020
30	D_3	0026
31	D_4	002C
5	IE	= 00A8
6	IP	= 00B8
7	P0	= 0080
8	P1	= 0090
9	P2	= 00A0
10	P3	= 00B0
11	PCAP2H	= 00CB
12	PCAP2L	= 00CA
13	PCON	= 0087
14	PSW	= 00D0
15	SBUF	= 0099
16	SCON	= 0098
17	SP	= 0081
18	T2CON	= 00C8
19	TCON	= 0088

```
20      TH0         = 008C
21      TL0         = 008A
22      TH1         = 008D
23      TL1         = 008B
24      TH2         = 00CD
25      TL2         = 00CC
26      TMOD        = 0089
```

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

   1  0000                    ORG  0000H
   2  0000  C2 90       DBG:     CLR  P1.0
   3  0002  C0 E0              PUSH ACC
   4  0004  C0 00              PUSH 0
   5  0006  C0 83              PUSH DPH
   6  0008  C0 82              PUSH DPL
   7
   8  000A  A8 81               MOV  R0,SP
   9  000C  E8                  MOV  A,R0
  10  000D  94 05               SUBB A,#5
  11  000F  F8                  MOV  R0,A
  12  0010  E6                  MOV  A,@R0
  13  0011  94 26               SUBB A,#38
  14  0013  F6                  MOV  @R0,A
  15
  16  0014  75 7E 00             MOV  7EH,#0
  17  0017  75 7F 00             MOV  7FH,#0
  18  001A  D5 7E FD     D_1:  DJNZ 7EH,$
  19  001D  D5 7F FA           DJNZ 7FH,D_1
  20  0020  D5 7E FD     D_2:  DJNZ 7EH,$
  21  0023  D5 7F FA           DJNZ 7FH,D_2
  22  0026  D5 7E FD     D_3:  DJNZ 7EH,$
  23  0029  D5 7F FA           DJNZ 7FH,D_3
  24  002C  D5 7E FD     D_4:  DJNZ 7EH,$
```

```
25  002F  D5 7F FA          DJNZ 7FH,D_4
26  0032  D0 82             POP  DPL
27  0034  D0 83             POP  DPH
28  0036  D0 00             POP  0
29  0038  D0 E0             POP  ACC
30  003A  85 7D D0          MOV  PSW,7DH
31  003D  22                RET
32                          END
```

TURBO51. 8051 Assembler Ver 1.00

Sequence	Symbol name	Value
1	ACC	= 00E0
2	B	= 00F0
27	DBG	0000
3	DPH	= 0083
4	DPL	= 0082
28	D_1	001A
29	D_2	0020
30	D_3	0026
31	D_4	002C
5	IE	= 00A8
6	IP	= 00B8
7	P0	= 0080
8	P1	= 0090
9	P2	= 00A0
10	P3	= 00B0
11	PCAP2H	= 00CB
12	PCAP2L	= 00CA
13	PCON	= 0087
14	PSW	= 00D0
15	SBUF	= 0099
16	SCON	= 0098
17	SP	= 0081
18	T2CON	= 00C8
19	TCON	= 0088

```
20      TH0     = 008C
21      TL0     = 008A
22      TH1     = 008D
23      TL1     = 008B
24      TH2     = 00CD
25      TL2     = 00CC
26      TMOD    = 0089
```

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

  1  0000                    ORG  0000H
  2  0000  C2 90       DBG:      CLR  P1.0
  3  0002  C0 E0                 PUSH ACC
  4  0004  C0 00                 PUSH 0
  5  0006  C0 83                 PUSH DPH
  6  0008  C0 82                 PUSH DPL
  7
  8  000A  A8 81                 MOV  R0,SP
  9  000C  E8                    MOV  A,R0
 10  000D  94 05                 SUBB A,#5
 11  000F  F8                    MOV  R0,A
 12  0010  E6                    MOV  A,@R0
 13  0011  94 26                 SUBB A,#38
 14  0013  F6                    MOV  @R0,A
 15
 16  0014  75 7E 00              MOV  7EH,#0
 17  0017  75 7F 00              MOV  7FH,#0
 18  001A  D5 7E FD     D_1:  DJNZ 7EH,$
 19  001D  D5 7F FA           DJNZ 7FH,D_1
 20  0020  D5 7E FD     D_2:  DJNZ 7EH,$
 21  0023  D5 7F FA           DJNZ 7FH,D_2
 22  0026  D5 7E FD     D_3:  DJNZ 7EH,$
 23  0029  D5 7F FA           DJNZ 7FH,D_3
 24  002C  D5 7E FD     D_4:  DJNZ 7EH,$
```

```
25  002F  D5 7F FA        DJNZ 7FH,D_4
26  0032  D0 82           POP  DPL
27  0034  D0 83           POP  DPH
28  0036  D0 00           POP  0
29  0038  D0 E0           POP  ACC
30  003A  85 7D D0        MOV  PSW,7DH
31  003D  22              RET
32                        END
```

TURBO51. 8051 Assembler Ver 1.00

Sequence	Symbol name	Value
1	ACC	= 00E0
2	B	= 00F0
27	DBG	0000
3	DPH	= 0083
4	DPL	= 0082
28	D_1	001A
29	D_2	0020
30	D_3	0026
31	D_4	002C
5	IE	= 00A8
6	IP	= 00B8
7	P0	= 0080
8	P1	= 0090
9	P2	= 00A0
10	P3	= 00B0
11	PCAP2H	= 00CB
12	PCAP2L	= 00CA
13	PCON	= 0087
14	PSW	= 00D0
15	SBUF	= 0099
16	SCON	= 0098
17	SP	= 0081

```
18      T2CON   = 00C8
19      TCON    = 0088
20      TH0     = 008C
21      TL0     = 008A
22      TH1     = 008D
23      TL1     = 008B
24      TH2     = 00CD
25      TL2     = 00CC
26      TMOD    = 0089
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
 1 008D          TH1          EQU  8DH
 2 0008          ASCII_H       EQU  08H
 3 0009          ASCII_L       EQU  09H
 4 000A          HEX_CODE        EQU  0AH
 5 000B          CODE_WORD         EQU  0BH ; INTERNAL
 6                             ;  'W' = WAIT
 7                             ;  'L' = LOAD_HEX
 8                             ;  '1' = 2764
 9                             ;  '2' = 27128
10                             ;  '3' = 27256
11                             ;  '4' = 27512
12                             ;  '5' = 6264
13                             ;  '6' = 62256
14                             ;  'B' = BREAK_P
15                             ;  'O' = OK
16                             ;  'D' = DUMP
17                             ;  'R' = RUN
18                             ;  'S' = SINGLE_STEP
19                             ;  'T' = TRACE
20                             ;  'P' = STOP
21                             ;  '$' = ADD_START
23
```

```
24
25 000C          SELECT_MEM       EQU 0CH
26 000D          CALL_MEM_1       EQU 0DH ;INTERNAL
27 000E          CALL_PC_H        EQU 0EH
28 000F          CALL_PC_L        EQU 0FH
29                                ; 8K AT 1800H - 1FFFH
30                                ; 16K AT 3800H - 3FFFH
31                                ; 32K AT 7800H - 7FFFH
32                                ; 64K AT F800H - FFFFH
33                                ;
34
35
36 0010          LONG_BYTE        EQU 10H
37 0011          PC_H             EQU 11H ; SAVE BEGIN ADDRESS HEX_FILE
38 0012          PC_L             EQU 12H ;
39 0013          DPC_H            EQU 13H ; ADDRESS BEGIN DUMP
40 0014          DPC_L            EQU 14H ;
41 0015          MARK_PC          EQU 15H
42 0016          OFSET_H          EQU 16H
43 0017          OFSET_L          EQU 17H
44 0018          BACKUPCODE       EQU 18H ;18H..1AH
45 001B          NUM_B            EQU 1BH
46 001C          COUNT_INT        EQU 1CH
47 001D          COUNT_B          EQU 1DH
48 001E          REAL_PCH_B       EQU 1EH
49 001F          REAL_PCL_B       EQU 1FH
50 0020          FLAG             EQU 20H
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
 51 0021            B_POINT        EQU 21H ;21H..34H
 52 0035            ADD_SH         EQU 35H
 53 0036            ADD_SL         EQU 36H
 54
 55              ;################## MAIN ################################
 56
 57 0000                        ORG 0000H
 58 0000 02 01 7B                LJMP SET_SYS
 59
 60 0003                        ORG 0003H
 61 0003 C2 A8      INT_0:        CLR IE.0
 62 0005 C0 E0                  PUSH ACC
 63 0007 C0 00                  PUSH 0
 64 0009 05 1C                  INC COUNT_INT
 65 000B A8 1C                   MOV R0,COUNT_INT
 66 000D B8 02 03                CJNE R0,#2,INT_0_END
 67 0010 12 05 20                LCALL RUN_INT_0
 68 0013 D0 00      INT_0_END:     POP 0
 69 0015 D0 E0                  POP ACC
 70 0017 D2 A8                   SETB IE.0
 71 0019 32                     RETI
 72
 73              ;              ORG 0013H
 74              ;INT_1:        CLR IE.2
```

```
75                ;          PUSH DPH
76                ;          PUSH DPL
77                ;          LCALL RUN_INT_1
78                ;          POP  DPL
79                ;          POP  DPH
80                ;          SETB IE.2
81                ;          RETI
82
83 0023                       ORG  0023H
84 0023 C0 E0      SERIAL_INT:     PUSH ACC
85 0025 C0 00                 PUSH 0
86 0027 78 00                 MOV  R0,#00H
87
88 0029 30 98 FD   SERIAL_0:      JNB  SCON.0,$
89 002C B2 90                 CPL  P1.0
90 002E D8 FE                 DJNZ R0,$
91 0030 B2 90                 CPL  P1.0
92 0032 E5 99                 MOV  A,SBUF
93 0034 C2 98                 CLR  SCON.0
94 0036 F5 0B                 MOV  CODE_WORD,A
95
96 0038 30 99 FD               JNB  SCON.1,$
97 003B F5 99                 MOV  SBUF,A
98 003D C2 99                 CLR  SCON.1
99
100 003F 12 03 45              LCALL CONV_S_H  ;TOLGLE_FLAG.0
```

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

101  0042  D0 00       SERIAL_END:      POP  0
102  0044  D0 E0                        POP  ACC
103  0046  32                          RETI
104
105                ;#############################################################
106
107  0100                             ORG  0100H
108  0100  E5 0B       MAIN:            MOV  A,CODE_WORD
109  0102  B4 57 02                     CJNE A,#'W',M_1
110  0105  80 F9                       SJMP MAIN
111  0107  B4 4C 13    M_1:             CJNE A,#'L',M_2
112  010A  D2 91                       SETB P1.1
113  010C  C2 92                       CLR  P1.2
114  010E  12 03 FC                     LCALL LOAD_HEX
115  0111  C2 AC                       CLR  IE.4
116  0113  7A 01                       MOV  R2,#1
117  0115  12 01 C9                     LCALL S_BREAK
118  0118  D2 AC                       SETB IE.4
119  011A  02 01 00                     LJMP MAIN
120  011D  B4 4B 03     M_2:            CJNE A,#'K',M_3
121  0120  02 04 86                     LJMP OK
122  0123  B4 52 05     M_3:            CJNE A,#'R',M_4
123  0126  C2 93                       CLR  P1.3
124  0128  02 04 EB                     LJMP RUN_PRO
```

```
125 012B B4 53 05    M_4:        CJNE A,#'S',M_5
126 012E C2 93                   CLR  P1.3
127 0130 02 05 7D                LJMP SINGLE_RUN
128 0133 B4 54 03    M_5:        CJNE A,#'T',M_6
129 0136 02 04 83                LJMP TRACE_RUN
130 0139 B4 50 03    M_6:        CJNE A,#'P',M_7
131 013C 02 05 BE                LJMP STOP_PRO
132 013F B4 43 03    M_7:        CJNE A,#'C',M_8
133 0142 12 05 F5                LCALL CHACK_RAM
134 0145 B4 44 03    M_8:        CJNE A,#'D',M_9
135 0148 12 04 B9                LCALL DUMP
136 014B B4 7E 03    M_9:        CJNE A,#'~',M_10
137 014E 02 06 48                LJMP SWAP1
138 0151 B4 21 03    M_10:       CJNE A,#'!',M_11
139 0154 02 06 54                LJMP SWAP2
140 0157 B4 60 0D    M_11:       CJNE A,#'`',M_12
141 015A D2 91                   SETB P1.1
142 015C D2 92                   SETB P1.2
143 015E 12 03 BC                LCALL SET_DBG
144 0161 12 03 FC                LCALL LOAD_HEX
145 0164 02 01 00                LJMP  MAIN
146 0167 B4 4D 03    M_12:       CJNE A,#'M',M_13
147 016A 02 02 AC                LJMP MENU_DBG
148 016D B4 42 03    M_13:       CJNE A,#'B',M_14
149 0170 02 04 89                LJMP BREAK_P
150 0173 B4 24 8A    M_14:       CJNE A,#'$',MAIN
```

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

 151  0176  02 01 B4          LJMP ADD_START
 152  0179  80 85             SJMP MAIN
 153
 154                ;#############################################################
 155
 156  017B  75 89 20   SET_SYS:    MOV  TMOD,#20H
 157  017E  75 88 01              MOV  TCON,#00000001B
 158  0181  75 98 52              MOV  SCON,#01010010B
 159  0184  75 A8 00              MOV  IE,#0
 160  0187  C2 8E                 CLR  PCON.7     ;SMOD
 161  0189  75 8D FD              MOV  TH1,#253
 162  018C  75 8B FD              MOV  8BH,#253
 163  018F  C2 98                 CLR  SCON.0     ;RI
 164  0191  D2 BC                 SETB  IP.4
 165  0193  D2 BA                 SETB  IP.2
 166  0195  D2 B8                 SETB  IP.0
 167  0197  D2 AF                 SETB  IE.7
 168  0199  D2 AC                 SETB  IE.4
 169  019B  D2 8E                 SETB  TCON.6    ;TR1
 170  019D  75 81 37              MOV  SP,#37H
 171  01A0  75 20 00              MOV  FLAG,#00H
 172  01A3  75 0B 00              MOV  CODE_WORD,#00H
 173  01A6  75 1D 00              MOV  COUNT_B,#0
 174  01A9  75 90 02              MOV  P1,#02H
```

```
175 01AC D2 93                  setb p1.3      ;reset
176 01AE 75 B0 FF                MOV  P3,#0FFH
177              ;              mov  r0,#b_point
178              ;              mov  @r0,#80h
179              ;              inc  r0
180              ;              mov  @r0,#17h
181 01B1 02 01 00                LJMP MAIN
182
183              ;*************************************************************
184 01B4 C2 AC      ADD_START:      CLR  IE.4
185 01B6 C0 E0                   PUSH ACC
186 01B8 12 05 D3                LCALL RX_DATA
187 01BB F5 35                   MOV  ADD_SH,A
188 01BD 12 05 D3                LCALL RX_DATA
189 01C0 F5 36                   MOV  ADD_SL,A
190 01C2 D0 E0                   POP  ACC
191 01C4 D2 AC                   SETB IE.4
192 01C6 02 01 00                LJMP MAIN
193              ;*************************************************************
194 01C9 C2 AC      S_BREAK:        CLR  IE.4
195 01CB C0 E0                   PUSH ACC    .;INPUT R2 => 1..5
196 01CD C0 02                   PUSH 2      ;DPTR = BREAK_POINT
197 01CF C0 01                   PUSH 1
198 01D1 A9 1B                   MOV  R1,NUM_B
199 01D3 B9 00 02                CJNE R1,#0,S_B0
200 01D6 80 1C                   SJMP S_END
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
201 01D8 79 1F      S_B0:       MOV R1,#1FH  ;B_POINT
202 01DA 09         S_B1:       INC R1
203 01DB 09                     INC R1
204 01DC DA FC                  DJNZ R2,S_B1
205 01DE E7                     MOV A,@R1
206 01DF F5 83                  MOV DPH,A
207 01E1 09                     INC R1
208 01E2 E7                     MOV A,@R1
209 01E3 F5 82                  MOV DPL,A
210 01E5 12 04 5A               LCALL OFSET
211 01E8 85 83 16               MOV OFSET_H,DPH
212 01EB 85 82 17               MOV OFSET_L,DPL
213 01EE 05 1D                  INC COUNT_B
214 01F0 C2 92                  CLR P1.2
215 01F2 31 FB                  ACALL BACKUP
216 01F4 D0 01      S_END:      POP 1
217 01F6 D0 02                  POP 2
218 01F8 D0 E0                  POP ACC
219 01FA 22                     RET
220             ;*************************************************************
221
222 01FB C0 E0      BACKUP:     PUSH ACC
223 01FD C0 83                  PUSH DPH
224 01FF C0 82                  PUSH DPL
```

```
225  0201  C0 01                PUSH 1
226  0203  C0 00                PUSH 0
227  0205  78 03                MOV  R0,#03H
228  0207  79 18                MOV  R1,#BACKUPCODE
229  0209  C0 83                PUSH DPH
230  020B  C0 82                PUSH DPL
231  020D  E0          BUP1:      MOVX A,@DPTR
232  020E  F7                   MOV  @R1,A
233  020F  09                   INC  R1
234  0210  05 82                INC  DPL
235  0212  D8 F9                DJNZ R0,BUP1
236  0214  D0 82                POP  DPL
237  0216  D0 83                POP  DPH
238  0218  78 03                MOV  R0,#03H
239  021A  79 0D                MOV  R1,#CALL_MEM_1
240  021C  E7          BUP2:      MOV  A,@R1
241  021D  F0                   MOVX @DPTR,A
242  021E  09                   INC  R1
243  021F  05 82                INC  DPL
244  0221  D8 F9                DJNZ R0,BUP2
245  0223  D0 00                POP  0
246  0225  D0 01                POP  1
247  0227  D0 82                POP  DPL
248  0229  D0 83                POP  DPH
249  022B  D0 E0                POP  ACC
250  022D  22                   RET
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
251              ;************************************************************
252 022E C2 AC       UN_BACKUP:      CLR IE.4
253 0230 C2 A8                       CLR IE.0
254                                  ;PUSH ACC
255                                  ;PUSH DPH
256                                  ;PUSH DPL
257                                  ;PUSH 1
258                                  ;PUSH 0
259 0232 78 03                       MOV R0,#03H
260 0234 79 18                       MOV R1,#BACKUPCODE
261 0236 85 16 83                    MOV DPH,OFSET_H
262 0239 85 17 82                    MOV DPL,OFSET_L
263 023C 20 B2 FD                    JB P3.2,$
264 023F C2 90                       CLR P1.0
265 0241 D2 91                       SETB P1.1
266 0243 C2 92                       CLR P1.2
267 0245 E7          UN_1:           MOV A,@R1
268 0246 F0                          MOVX @DPTR,A
269 0247 09                          INC R1
270 0248 05 82                       INC DPL
271 024A D8 F9                       DJNZ R0,UN_1
272
273 024C E5 1D                       MOV A,COUNT_B
274 024E B5 1B 00                    CJNE A,NUM_B,UN_2
```

```
275 0251  50 07       UN_2:          JNC UN_END
276 0253  04                         INC A
277 0254  FA                         MOV R2,A
278 0255  31 C9                      ACALL S_BREAK
279 0257  02 04 EB                   LJMP RUN_PRO
280 025A  D2 90       UN_END:        SETB P1.0
281 025C  C2 91                      CLR P1.1
282                                  ;POP 0
283                                  ;POP 1
284                                  ;POP DPL
285                                  ;POP DPH
286                                  ;POP ACC
287                                  ;RET
288 025E  02 05 0F                   LJMP RUN_PEND1
289               ;*************************************************************
290 0261  C2 AC       UN_SINGLE:     CLR IE.4
291 0263  C2 A8                      CLR IE.0
292                                  ;PUSH ACC
293                                  ;PUSH DPH
294                                  ;PUSH DPL
295                                  ;PUSH 1
296                                  ;PUSH 0
297 0265  78 03                      MOV R0,#03H
298 0267  79 18                      MOV R1,#BACKUPCODE
299 0269  85 16 83                   MOV DPH,OFSET_H
300 026C  85 17 82                   MOV DPL,OFSET_L
```

```
275 0251 50 07      UN_2:       JNC UN_END
276 0253 04                     INC A
277 0254 FA                     MOV R2,A
278 0255 31 C9                  ACALL S_BREAK
279 0257 02 04 EB               LJMP RUN_PRO
280 025A D2 90      UN_END:     SETB P1.0
281 025C C2 91                  CLR P1.1
282                             ;POP 0
283                             ;POP 1
284                             ;POP DPL
285                             ;POP DPH
286                             ;POP ACC
287                             ;RET
288 025E 02 05 0F              LJMP RUN_PEND1
289         ;*************************************************************
290 0261 C2 AC      UN_SINGLE:  CLR IE.4
291 0263 C2 A8                  CLR IE.0
292                             ;PUSH ACC
293                             ;PUSH DPH
294                             ;PUSH DPL
295                             ;PUSH 1
296                             ;PUSH 0
297 0265 78 03                  MOV R0,#03H
298 0267 79 18                  MOV R1,#BACKUPCODE
299 0269 85 16 83               MOV DPH,OFSET_H
300 026C 85 17 82               MOV DPL,OFSET_L
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
301 026F 20 B2 FD                JB   P3.2,$
302 0272 C2 90                   CLR  P1.0
303 0274 D2 91                   SETB P1.1
304 0276 C2 92                   CLR  P1.2
305 0278 E7          UN_SGLE1:   MOV  A,@R1
306 0279 F0                      MOVX @DPTR,A
307 027A 09                      INC  R1
308 027B 05 82                   INC  DPL
309 027D D8 F9                   DJNZ R0,UN_SGLE1
310 027F 7A 01                   MOV  R2,#1
311 0281 12 01 C9                LCALL S_BREAK
312 0284 02 05 7D                LJMP  SINGLE_RUN
313
314 0287 D2 90       SGLE_END1:  SETB P1.0
315 0289 C2 91                   CLR  P1.1
316                              ;POP  0
317                              ;POP  1
318                              ;POP  DPL
319                              ;POP  DPH
320                              ;POP  ACC
321                              ;RET
322 028B 02 05 AB                LJMP SGLE_END
323              ;*************************************************************
324 028E 79 21       ADD_SGLE:   MOV  R1,#B_POINT
```

```
325 0290  78 02                MOV  R0,#02
326 0292  74 4F                MOV  A,#'O'
327 0294  12 02 DE              LCALL SEND_CHAR
328 0297  12 05 D3   SGLE_1:    LCALL RX_DATA
329 029A  F7                   MOV  @R1,A
330 029B  09                   INC  R1
331 029C  D8 F9                DJNZ R0,SGLE_1
332 029E  22                   RET
333          ;*************************************************************
334 029F  E5 11      ADDRESS:  MOV  A,PC_H
335 02A1  25 16                ADD  A,OFSET_H
336 02A3  F5 1E                MOV  REAL_PCH_B,A
337 02A5  E5 12                MOV  A,PC_L
338 02A7  25 17                ADD  A,OFSET_L
339 02A9  F5 1F                MOV  REAL_PCL_B,A
340 02AB  22                   RET
341          ;*************************************************************
342
343 02AC  C2 AF      MENU_DBG: CLR  IE.7
344 02AE  C0 E0                PUSH ACC
345 02B0  C2 98                CLR  SCON.0
346 02B2  30 98 FD              JNB  SCON.0,$
347 02B5  E5 99                MOV  A,SBUF
348 02B7  C2 98                CLR  SCON.0
349 02B9  F5 0C                MOV  SELECT_MEM,A
350 02BB  12 05 D3              LCALL RX_DATA
```

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

 351  02BE  F5 16              MOV   OFSET_H,A
 352  02C0  12 05 D3           LCALL RX_DATA
 353  02C3  F5 17              MOV   OFSET_L,A
 354  02C5  90 07 12           MOV   DPTR,#OK_
 355  02C8  12 02 F8           LCALL SEND_ST
 356  02CB  12 03 BC           LCALL SET_DBG
 357  02CE  C2 92              CLR   P1.2
 358  02D0  D2 91              SETB  P1.1
 359  02D2  D0 E0              POP   ACC
 360  02D4  75 0B 00           MOV   CODE_WORD,#0
 361  02D7  D2 AC              SETB  IE.4
 362  02D9  D2 AF              SETB  IE.7
 363  02DB  02 01 00           LJMP MAIN
 364
 365            ;*************************************************************
 366
 367  02DE  30 99 FD   SEND_CHAR:    JNB   SCON.1,$
 368  02E1  F5 99                    MOV   SBUF,A
 369  02E3  C2 99                    CLR   SCON.1
 370  02E5  22                       RET
 371
 372            ;*************************************************************
 373
 374  02E6  C0 E0      SEND_H_AS:    PUSH  ACC
```

```
375  02E8  12 03 89              LCALL CONV_H_S
376  02EB  E5 08                 MOV   A,ASCII_H
377  02ED  12 02 DE              LCALL SEND_CHAR
378  02F0  E5 09                 MOV   A,ASCII_L
379  02F2  12 02 DE              LCALL SEND_CHAR
380  02F5  D0 E0       SEND_END: POP   ACC
381  02F7  22                    RET
382
383
;*************************************************************
384
385  02F8  C0 E0       SEND_ST:  PUSH  ACC
386  02FA  C0 83                 PUSH  DPH
387  02FC  C0 82                 PUSH  DPL
388  02FE  E4                    CLR   A
389
390  02FF  C0 E0       SEND_ST0: PUSH  ACC
391  0301  93                    MOVC  A,@A+DPTR
392  0302  B4 2A 02              CJNE  A,#'*',SEND_ST1
393  0305  80 0C                 SJMP  SENDST_END
394  0307  30 99 FD    SEND_ST1: JNB   SCON.1,$
395  030A  F5 99                 MOV   SBUF,A
396  030C  C2 99                 CLR   SCON.1
397  030E  D0 E0                 POP   ACC
398  0310  04                    INC   A
399  0311  80 EC                 SJMP  SEND_ST0
400
```

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

401  0313  D0 E0       SENDST_END:      POP   ACC
402  0315  D0 82                        POP   DPL
403  0317  D0 83                        POP   DPH
404  0319  D0 E0                        POP   ACC
405  031B  22                          RET
406
407
;*************************************************************
408
409  031C  C2 AC       TRAN:            CLR   IE.4
410  031E  C0 E0                        PUSH  ACC
411  0320  C0 83                        PUSH  DPH
412  0322  C0 82                        PUSH  DPL
413  0324  C0 00                        PUSH  0
414
415  0326  78 80                       MOV   R0,#128
416  0328  E0          TX1:             MOVX  A,@DPTR
417  0329  12 03 89                     LCALL CONV_H_S
418  032C  E5 08                        MOV   A,ASCII_H
419  032E  12 02 DE                      LCALL SEND_CHAR
420  0331  E5 09                        MOV   A,ASCII_L
421  0333  12 02 DE                      LCALL SEND_CHAR
422  0336  05 82                       INC   DPL
423  0338  D8 EE                        DJNZ  R0,TX1
```

```
424
425 033A D0 00      TX_END:      POP  0
426 033C D0 82                   POP  DPL
427 033E D0 83                   POP  DPH
428 0340 D0 E0                   POP  ACC
429 0342 D2 AC                   SETB IE.4
430 0344 22                      RET
431
432             ;*************************************************************
433
434 0345 C0 E0      CONV_S_H:    PUSH ACC
435 0347 C0 00                   PUSH 0
436 0349 A8 20                   MOV  R0,FLAG
437 034B B8 00 1F                CJNE R0,#0,HEX_L
438 034E B4 41 00                CJNE A,#41H,CONVS_1
439 0351 50 0C      CONVS_1:     JNC  CONS_1
440 0353 54 0F                   ANL  A,#0FH
441 0355 23                      RL   A
442 0356 23                      RL   A
443 0357 23                      RL   A
444 0358 23                      RL   A
445 0359 54 F0                   ANL  A,#0F0H
446 035B F5 0A                   MOV  HEX_CODE,A
447 035D 80 23                   SJMP CONS_END
448 035F 54 0F      CONS_1:      ANL  A,#0FH
449 0361 24 09                   ADD  A,#09H
450 0363 23                      RL   A
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
451 0364 23                 RL  A
452 0365 23                 RL  A
453 0366 23                 RL  A
454 0367 54 F0               ANL  A,#0F0H
455 0369 F5 0A               MOV  HEX_CODE,A
456 036B 80 15               SJMP CONS_END
457
458 036D B4 41 00   HEX_L:        CJNE A,#41H,CONVS_2
459 0370 50 08      CONVS_2:      JNC  CONS_2
460 0372 54 0F               ANL  A,#0FH
461 0374 45 0A               ORL  A,HEX_CODE
462 0376 F5 0A               MOV  HEX_CODE,A
463 0378 80 08               SJMP CONS_END
464 037A 54 0F      CONS_2:       ANL  A,#0FH
465 037C 24 09               ADD  A,#09H
466 037E 45 0A               ORL  A,HEX_CODE
467 0380 F5 0A               MOV  HEX_CODE,A
468 0382 B2 00      CONS_END:     CPL  FLAG.0
469 0384 D0 00               POP  0
470 0386 D0 E0               POP  ACC
471 0388 22                 RET
472
473             ;*************************************************************
474
```

```
475 0389 C0 E0      CONV_H_S:       PUSH ACC
476 038B C0 E0                      PUSH ACC
477 038D 54 0F                      ANL  A,#0FH
478 038F B4 0A 00                   CJNE A,#0AH,CONVH_1
479 0392 50 06      CONVH_1:        JNC  CONH_1
480 0394 44 30                      ORL  A,#30H
481 0396 F5 09                      MOV  ASCII_L,A
482 0398 80 06                      SJMP CONH_2
483 039A 94 09      CONH_1:         SUBB A,#09H
484 039C 44 40                      ORL  A,#40H
485 039E F5 09                      MOV  ASCII_L,A
486 03A0 D0 E0      CONH_2:         POP  ACC
487 03A2 03                         RR   A
488 03A3 03                         RR   A
489 03A4 03                         RR   A
490 03A5 03                         RR   A
491 03A6 54 0F                      ANL  A,#0FH
492 03A8 B4 0A 00                   CJNE A,#0AH,CONVH_2
493 03AB 50 06      CONVH_2:        JNC  CONH_3
494 03AD 44 30                      ORL  A,#30H
495 03AF F5 08                      MOV  ASCII_H,A
496 03B1 80 06                      SJMP CONH_END
497 03B3 94 09      CONH_3:         SUBB A,#09H
498 03B5 44 40                      ORL  A,#40H
499 03B7 F5 08                      MOV  ASCII_H,A
500 03B9 D0 E0      CONH_END:       POP  ACC
```

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

501  03BB  22                    RET
502            ;*************************************************************
503
504  03BC  C2 AC       SET_DBG:      CLR  IE.4
505  03BE  C0 E0                   PUSH ACC
506  03C0  30 98 FD                JNB  SCON.0,$
507  03C3  E5 99                   MOV  A,SBUF
508  03C5  F5 0C                   MOV  SELECT_MEM,A
509  03C7  C2 98                   CLR  SCON.0
510  03C9  74 4F                   MOV  A,#'O'
511  03CB  12 02 DE                LCALL SEND_CHAR
512  03CE  75 0D 12                MOV  CALL_MEM_1,#12H
513  03D1  75 0F 00                MOV  CALL_PC_L,#0
514  03D4  E5 0C                   MOV  A,SELECT_MEM
515  03D6  B4 30 02                CJNE A,#'0',DBG_1
516  03D9  80 1E                   SJMP DBG_END
517  03DB  75 0E 14    DBG_1:      MOV  CALL_PC_H,#14H
518  03DE  B4 31 02                CJNE A,#'1',DBG_2
519  03E1  80 16                   SJMP DBG_END
520  03E3  B4 32 05    DBG_2:      CJNE A,#'2',DBG_3
521  03E6  75 0E 34                MOV  CALL_PC_H,#34H
522  03E9  80 0E                   SJMP DBG_END
523  03EB  B4 33 05    DBG_3:      CJNE A,#'3',DBG_4
524  03EE  75 0E 74                MOV  CALL_PC_H,#74H
```

```
525 03F1 80 06                  SJMP DBG_END
526 03F3 B4 34 03    DBG_4:     CJNE A,#'4',DBG_END
527 03F6 75 0E F4               MOV  CALL_PC_H,#0F4H
528 03F9 D0 E0       DBG_END:   POP  ACC
529 03FB 22                     RET
530
531          ;*************************************************************
532
533 03FC C2 AC       LOAD_HEX:  CLR  IE.4  ;DISENABLE SERIAL
534 03FE C0 E0                  PUSH ACC
535 0400 C0 00                  PUSH 0
536 0402 C0 83                  PUSH DPH
537 0404 C0 82                  PUSH DPL
538
539 0406 75 15 00               MOV  MARK_PC,#00H     ;MARK_PC_BEGIN
540 0409 C2 00       LOAD_NEW:  CLR  FLAG.0
541
542 040B 12 04 23    SCAN:      LCALL SCAN_DATA
543
544          LOAD_END:          ;PUSH DPH
545                             ;PUSH DPL
546                             ;MOV  DPTR,#END_
547                             ;LCALL SEND_ST
548                             ;POP  DPL
549                             ;POP  DPH
550 040E 74 2A                  MOV  A,#'*'
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
551 0410 F0                MOVX @DPTR,A
552 0411 D0 82              POP DPL
553 0413 D0 83              POP DPH
554 0415 D0 00              POP 0
555 0417 D0 E0              POP ACC
556 0419 75 0B 00           MOV CODE_WORD,#00H
557 041C C2 00              CLR FLAG.0
558 041E C2 98              CLR SCON.0
559 0420 D2 AC              SETB IE.4
560 0422 22                RET
561
562             ;*************************************************************
563
564 0423 12 05 D3   SCAN_DATA:    LCALL RX_DATA
565 0426 F5 10              MOV   LONG_BYTE,A
566 0428 B4 00 03            CJNE  A,#00H,SCAN_PC
567 042B 02 04 59            LJMP  SCAN_END
568
569 042E 12 05 D3   SCAN_PC:      LCALL RX_DATA
570 0431 F5 83              MOV   DPH,A
571 0433 E5 15              MOV   A,MARK_PC
572 0435 B4 00 03            CJNE  A,#00H,SCAN_D1
573 0438 85 83 11            MOV   PC_H,DPH
574
```

```
575 043B 12 05 D3    SCAN_D1:      LCALL RX_DATA
576 043E F5 82                     MOV   DPL,A
577 0440 E5 15                     MOV   A,MARK_PC
578 0442 B4 00 06                  CJNE  A,#00H,SCAN_D3
579 0445 85 82 12                  MOV   PC_L,DPL
580 0448 75 15 FF                  MOV   MARK_PC,#0FFH
581 044B 91 5A       SCAN_D3:      ACALL OFSET
582 044D 12 05 D3    SCAN_D2:      LCALL RX_DATA
583 0450 F0                        MOVX  @DPTR,A
584 0451 05 82                     INC   DPL
585 0453 D5 10 F7                  DJNZ  LONG_BYTE,SCAN_D2
586 0456 02 04 23                  LJMP  SCAN_DATA
587 0459 22          SCAN_END:     RET
588
589                ;*************************************************
590 045A C2 AC       OFSET:        CLR   IE.4
591 045C C0 E0                     PUSH  ACC
592 045E C0 D0                     PUSH  PSW
593 0460 E5 82                     MOV   A,DPL
594 0462 95 12                     SUBB  A,PC_L
595 0464 F5 82                     MOV   DPL,A
596 0466 E5 83                     MOV   A,DPH
597 0468 95 11                     SUBB  A,PC_H
598 046A F5 83                     MOV   DPH,A
599 046C D0 D0                     POP   PSW
600 046E D0 E0                     POP   ACC
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
601 0470 22                 RET
602             ;*************************************************************
603 0471 02 01 00   RAM_64:     LJMP MAIN
604             ;*************************************************************
605 0474 02 01 00   RAM_256:    LJMP MAIN
606             ;*************************************************************
607 0477 02 01 00   EPROM_64:    LJMP MAIN
608             ;*************************************************************
609 047A 02 01 00   EPROM_128:    LJMP MAIN
610             ;*************************************************************
611 047D 02 01 00   EPROM_256:    LJMP MAIN
612             ;*************************************************************
613 0480 02 01 00   EPROM_512:    LJMP MAIN
614             ;*************************************************************
615             ;*************************************************************
616 0483 02 01 00   TRACE_RUN:    LJMP MAIN
617             ;*************************************************************
618 0486 02 01 00   OK:         LJMP MAIN
619             ;*************************************************************
620
621 0489 C2 AC      BREAK_P:    CLR IE.4
622 048B C0 E0              PUSH ACC
623 048D C0 F0              PUSH B
624 048F C0 01              PUSH 1
```

```
625 0491 C0 00                  PUSH 0
626 0493 12 05 D3                LCALL RX_DATA
627 0496 F5 1B                  MOV NUM_B,A
628 0498 F9                     MOV R1,A
629 0499 B9 00 02                CJNE R1,#0,B_MAKE
630 049C 80 0E                  SJMP B_END
631 049E 75 F0 02     B_MAKE:       MOV B,#02H
632 04A1 A4                     MUL AB
633 04A2 F8                     MOV R0,A
634 04A3 79 21                  MOV R1,#B_POINT
635 04A5 12 05 D3     B_1:        LCALL RX_DATA
636 04A8 F7                     MOV @R1,A
637 04A9 09                     INC R1
638 04AA D8 F9                   DJNZ R0,B_1
639 04AC D0 00        B_END:        POP 0
640 04AE D0 01                  POP 1
641 04B0 D0 F0                  POP B
642 04B2 D0 E0                  POP ACC
643 04B4 D2 AC                  SETB IE.4
644 04B6 02 01 00                LJMP MAIN
645
646             ;*************************************************************
647
648 04B9 C2 AC        DUMP:         CLR IE.4
649 04BB C0 E0                  PUSH ACC
650 04BD C0 00                  PUSH 0
```

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

651  04BF  C0 01              PUSH  1
652  04C1  12 05 D3           LCALL RX_DATA
653                       ;MOV  R1,A
654  04C4  95 11              SUBB A,PC_H
655  04C6  F5 13              MOV  DPC_H,A
656  04C8  12 05 D3           LCALL RX_DATA
657                       ;MOV  R0,A
658  04CB  95 12              SUBB A,PC_L
659  04CD  F5 14              MOV  DPC_L,A
660                       ;MOV   A,#' '
661                       ;LCALL SEND_CHAR
662                       ;MOV  DPTR,#DUMP_
663                       ;LCALL SEND_ST
664                       ;MOV   A,#' '
665                       ;LCALL SEND_CHAR
666                       ;MOV   A,R1
667                       ;LCALL SEND_H_AS
668                       ;MOV   A,R0
669                       ;LCALL SEND_H_AS
670                       ;MOV   A,#' '
671                       ;LCALL SEND_CHAR
672  04CF  12 06 60           LCALL  DELAY
673  04D2  85 13 83           MOV  DPH,DPC_H
674  04D5  85 14 82           MOV  DPL,DPC_L
```

```
675 04D8 12 03 1C              LCALL TRAN
676 04DB C2 AC                 CLR  IE.4
677 04DD D0 01                 POP  1
678 04DF D0 00                 POP  0
679 04E1 D0 E0                 POP  ACC
680 04E3 75 0B 00              MOV  CODE_WORD,#00H
681 04E6 D2 AC                 SETB IE.4
682 04E8 02 01 00              LJMP MAIN
683             ;*************************************************************
684 04EB C2 AC      RUN_PRO:       CLR  IE.4
685 04ED C2 91                 CLR  P1.1
686 04EF C2 92                 CLR  P1.2
687 04F1 C2 98                 CLR  SCON.0
688 04F3 C2 00                 CLR  FLAG.0
689 04F5 75 1C 00              MOV  COUNT_INT,#0
690             ;              lcall ADDRESS
691 04F8 D2 A8                 SETB  IE.0
692             RUN_P1:        ;CLR  P1.1
693 04FA C2 92                 CLR  P1.2
694 04FC 20 98 02              JB   SCON.0,RUN_PEND
695 04FF 80 F9                 SJMP RUN_P1
696 0501 E5 99      RUN_PEND:      MOV  A,SBUF
697 0503 C2 98                 CLR  SCON.0
698 0505 30 99 FD               JNB  SCON.1,$
699 0508 F5 99                 MOV  SBUF,A
700 050A C2 99                 CLR  SCON.1
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
701
702 050C  02 02 2E              LJMP UN_BACKUP
703
704 050F  75 0B 00    RUN_PEND1:      MOV  CODE_WORD,#0
705 0512  75 81 35              MOV  SP,#35H
706 0515  C2 00                 CLR  FLAG.0
707 0517  C2 98                 CLR  SCON.0
708 0519  C2 A8                 CLR  IE.0
709 051B  D2 AC                 SETB IE.4
710 051D  02 01 00              LJMP MAIN
711
;*************************************************************
712             RUN_INT_0:      ;PUSH 0
713                             ;MOV  R0,#01H
714             RUN_INT1:       ;JB   P3.2,INT0_END
715                             ;DJNZ R0,RUN_INT1
716 0520  A2 00                 MOV  C,FLAG.0
717 0522  40 58                 JC   INT0_END
718 0524  30 B2 FD              JNB  P3.2,$
719 0527  78 80                 MOV  R0,#128
720 0529  C2 90                 CLR  P1.0
721 052B  C2 91                 CLR  P1.1
722 052D  D2 92                 SETB P1.2
723 052F  90 02 00              MOV  DPTR,#0200H ;TABLE_INTERNAL_RAM
```

```
724 0532 74 4F                  MOV   A,#'O'
725 0534 12 02 DE                LCALL SEND_CHAR
726 0537 30 98 FD                JNB   SCON.0,$
727 053A C2 98                  CLR   SCON.0
728 053C E0          DUMP_INT:       MOVX  A,@DPTR
729 053D 12 02 E6                LCALL SEND_H_AS
730 0540 05 82                  INC   DPL
731 0542 D8 F8                  DJNZ  R0,DUMP_INT
732 0544 78 16                  MOV   R0,#22     ;sfr + flag
733 0546 90 03 00                MOV   DPTR,#0300H ;TABLE_SFR
734 0549 74 4F                  MOV   A,#'O'
735 054B 12 02 DE                LCALL SEND_CHAR
736 054E 30 98 FD                JNB   SCON.0,$
737 0551 C2 98                  CLR   SCON.0
738 0553 E0          DUMP_INT1:      MOVX  A,@DPTR
739 0554 12 02 E6                LCALL SEND_H_AS
740 0557 05 82                  INC   DPL
741 0559 D8 F8                  DJNZ  R0,DUMP_INT1
742
743 055B 78 03                  mov   r0,#3
744 055D 79 18                  mov   r1,#BACKUPCODE
745 055F E7          dump_int2:      mov   a,@r1
746 0560 12 02 E6                lcall SEND_H_AS
747 0563 09                    inc   r1
748 0564 D8 F9                  djnz  r0,dump_int2
749
750 0566 78 02                  mov   r0,#2
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
751 0568 79 1E                  mov  r1,#REAL_PCH_B
752 056A 12 02 9F                LCALL ADDRESS
753 056D E7         dump_int3:      mov  a,@r1
754 056E 12 02 E6                lcall SEND_H_AS
755 0571 09                    inc  r1
756 0572 D8 F9                 djnz r0,dump_int3
757
758 0574 C2 92                 CLR  P1.2
759 0576 C2 98                 CLR  SCON.0
760 0578 D2 90                 SETB P1.0
761 057A D2 00                  SETB FLAG.0
762             INT0_END:       ;PUSH 0
763 057C 22                    RET
764             ;***************************************************************
765 057D C2 AC       SINGLE_RUN:     CLR IE.4
766 057F C2 91                 CLR P1.1
767 0581 C2 92                 CLR P1.2
768 0583 C2 98                 CLR SCON.0
769 0585 C2 00                 CLR FLAG.0
770 0587 75 1C 00                MOV COUNT_INT,#0
771             ;           lcall ADDRESS
772 058A D2 A8                 SETB IE.0
773 058C C2 92       SINGLE_R1:      CLR P1.2
774 058E 20 98 02              JB  SCON.0,SINGLE_END
```

```
775 0591  80 F9                  SJMP SINGLE_R1
776 0593  E5 99       SINGLE_END:      MOV  A,SBUF
777 0595  C2 98                  CLR  SCON.0
778 0597  30 99 FD                JNB  SCON.1,$
779 059A  F5 99                  MOV  SBUF,A
780 059C  C2 99                  CLR  SCON.1
781 059E  C2 A8                   CLR  IE.0
782
783 05A0  B4 51 02                 CJNE A,#'Q',GO_SGLE
784 05A3  80 06                  SJMP SGLE_END
785 05A5  12 02 8E     GO_SGLE:        LCALL ADD_SGLE
786 05A8  02 02 61                LJMP UN_SINGLE
787 05AB  75 0B 00     SGLE_END:       MOV  CODE_WORD,#0
788 05AE  75 81 35                 MOV  SP,#35H
789 05B1  C2 00                  CLR  FLAG.0
790 05B3  C2 98                  CLR  SCON.0
791 05B5  C2 A8                  CLR  IE.0
792 05B7  C2 91                  CLR  P1.1
793 05B9  D2 AC                   SETB IE.4
794 05BB  02 01 00                LJMP MAIN
795              ;*************************************************************
796 05BE  C2 AC       STOP_PRO:       CLR  IE.4
797 05C0  C2 A8                   CLR  IE.0
798 05C2  D2 91                  SETB P1.1
799 05C4  D2 93                  SETB P1.3
800 05C6  C2 92                  CLR  P1.2
```

TURBO51. 8051 Assembler Ver 1.00

Line Addr Obj

```
801 05C8 75 1D 00              MOV  COUNT_B,#0
802 05CB 75 0B 00              MOV  CODE_WORD,#00H
803 05CE D2 AC                 SETB  IE.4
804 05D0 02 01 00              LJMP MAIN
805             ;*********************************************************
806
807 05D3 C2 AC      RX_DATA:      CLR  IE.4    ;RX_ASICC_TO_HEX => A
808 05D5 C2 00                 CLR  FLAG.0
809 05D7 30 98 FD    RX_T1:       JNB  SCON.0,$
810 05DA E5 99                 MOV  A,SBUF
811 05DC C2 98                 CLR  SCON.0
812 05DE 12 03 45              LCALL CONV_S_H
813 05E1 A2 00                 MOV  C,FLAG.0
814 05E3 40 F2                 JC   RX_T1
815 05E5 E5 0A                 MOV  A,HEX_CODE
816 05E7 22                 RET
817
818             ;*********************************************************
819
820 05E8 C0 00      LED:        PUSH 0
821 05EA 78 10                 MOV  R0,#10H
822 05EC B2 90      LED_1:      CPL  P1.0
823 05EE D1 60                 ACALL DELAY
824 05F0 D8 FA                 DJNZ R0,LED_1
```

```
825 05F2 D0 00                POP 0
826 05F4 22                   RET
827
828          ;*************************************************************
829
830 05F5 C2 AC      CHACK_RAM:      CLR IE.4
831 05F7 C0 E0                PUSH ACC
832 05F9 C0 83                PUSH DPH
833 05FB C0 82                PUSH DPL
834 05FD C0 00                PUSH 0
835 05FF C0 01                PUSH 1
836 0601 D2 90                SETB P1.0
837 0603 78 00                MOV  R0,#00H
838 0605 79 00                MOV  R1,#00H
839 0607 90 20 00             MOV  DPTR,#2000H
840 060A E8         CHACK1:       MOV  A,R0
841 060B F0                   MOVX @DPTR,A
842 060C E0                   MOVX A,@DPTR
843 060D 68                   XRL  A,R0
844 060E A3                   INC  DPTR
845 060F 08                   INC  R0
846 0610 B4 00 F7               CJNE A,#00,CHACK1
847 0613 74 20                MOV   A,#' '
848 0615 12 02 DE               LCALL SEND_CHAR
849 0618 E5 83                MOV   A,DPH
850 061A 12 02 E6               LCALL SEND_H_AS
```

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

 851  061D  E5 82              MOV   A,DPL
 852  061F  12 02 E6           LCALL SEND_H_AS
 853  0622  74 20              MOV   A,#' '
 854  0624  12 02 DE           LCALL SEND_CHAR
 855  0627  B2 90              CPL   P1.0
 856  0629  D9 FE              DJNZ  R1,$
 857  062B  D9 FE              DJNZ  R1,$
 858  062D  D9 FE              DJNZ  R1,$
 859  062F  D9 FE              DJNZ  R1,$
 860  0631  B2 90              CPL   P1.0
 861  0633  E5 83              MOV   A,DPH
 862  0635  B4 FF D2           CJNE  A,#0FFH,CHACK1
 863
 864  0638  D0 01              POP   1
 865  063A  D0 00              POP   0
 866  063C  D0 82              POP   DPL
 867  063E  D0 83              POP   DPH
 868  0640  D0 E0              POP   ACC
 869  0642  D2 AC              SETB  IE.4
 870  0644  75 0B 00           MOV   CODE_WORD,#00H
 871  0647  22                 RET
 872
;************************************************************
 873  0648  C2 AC     SWAP1:   CLR   IE.4
```

```
874 064A B2 91                  CPL P1.1
875 064C 75 0B 00               MOV CODE_WORD,#00H
876 064F D2 AC                  SETB IE.4
877 0651 02 01 00               LJMP MAIN
878
;*************************************************************
879 0654 C2 AC      SWAP2:      CLR IE.4
880 0656 B2 92                  CPL P1.2
881 0658 75 0B 00               MOV CODE_WORD,#00H
882 065B D2 AC                  SETB IE.4
883 065D 02 01 00               LJMP MAIN
884
;*************************************************************
885 0660 C0 D0      DELAY:      PUSH PSW
886 0662 D5 7E FD   D_1:        DJNZ 7EH,$
887 0665 D5 7F FA               DJNZ 7FH,D_1
888 0668 D0 D0                  POP PSW
889 066A 22                     RET
890             ;*************************************************************
891 066B 01 02 03 04  TABLE1:   DB 1,2,3,4,5,6,7,8,9,0AH,0
BH,0CH,0DH,0EH,0FH,'*'
892 066F 05 06 07 08
893 0673 09 0A 0B 0C
894 0677 0D 0E 0F 2A
895 067B 20 4B 4F 4D  TABLE2:   DB ' KOMET JAMJAN   KOMET JAMJAN '
896 067F 45 54 20 4A
897 0683 41 4D 4A 41
898 0687 4E 20 20 20
899 068B 20 4B 4F 4D
```

900 068F 45 54 20 4A

TURBO51. 8051 Assembler Ver 1.00

```
Line  Addr  Obj

 901  0693  41 4D 4A 41
 902  0697  4E 20 2A
 903  069A  20 50 43 5F    PC_H_:          DB ' PC_H = *'
 904  069E  48 20 3D 20
 905  06A2  2A
 906  06A3  20 50 43 5F    PC_L_:          DB ' PC_L = *'
 907  06A7  4C 20 3D 20
 908  06AB  2A
 909  06AC  20 50 43 20    PC_:            DB ' PC => *'
 910  06B0  3D 3E 20 2A
 911  06B4  20 44 50 48    DPH_:           DB ' DPH  = *'
 912  06B8  20 20 3D 20
 913  06BC  2A
 914  06BD  20 44 50 4C    DPL_:           DB ' DPL  = *'
 915  06C1  20 20 3D 20
 916  06C5  2A
 917  06C6  20 4C 4F 4E    LONG_:          DB ' LONG = *'
 918  06CA  47 20 3D 20
 919  06CE  2A
 920  06CF  20 44 41 54    DATA_:          DB ' DATA = *'
 921  06D3  41 20 3D 20
 922  06D7  2A
 923  06D8  20 44 50 54    DPTR_:          DB ' DPTR = *'
 924  06DC  52 20 3D 20
```

```
925  06E0  2A
926  06E1  20 20 20 45    END_:          DB '  END  *'
927  06E5  4E 44 20 20
928  06E9  2A
929  06EA  20 53 50 20    SP_:           DB ' SP  = *'
930  06EE  20 20 3D 20
931  06F2  2A
932  06F3  20 20 20 20    EMTY:          DB '    *'
933  06F7  20 2A
934  06F9  20 52 45 41    READ_DATA:     DB ' READ DATA => *'
935  06FD  44 20 44 41
936  0701  54 41 20 3D
937  0705  3E 20 2A
938  0708  20 44 55 4D    DUMP_:         DB ' DUMP => *'
939  070C  50 20 3D 3E
940  0710  20 2A
941  0712  20 4F 46 53    OK_:           DB ' OFSET_OK *'
942  0716  45 54 5F 4F
943  071A  4B 20 2A
944                        END
```

TURBO51. 8051 Assembler Ver 1.00

Sequence Symbol name Value

1 ACC = 00E0
112 ADDRESS 029F
109 ADD_SGLE 028E
52 ADD_SH = 0035
53 ADD_SL = 0036
91 ADD_START 01B4
28 ASCII_H = 0008
29 ASCII_L = 0009
2 B = 00F0
97 BACKUP 01FB
44 BACKUPCODE = 0018
90 BREAK_P 0489
98 BUP1 020D
99 BUP2 021C
158 B_1 04A5
157 B_END 04AC
156 B_MAKE 049E
51 B_POINT = 0021
33 CALL_MEM_1 = 000D
34 CALL_PC_H = 000E
35 CALL_PC_L = 000F
174 CHACK1 060A
78 CHACK_RAM 05F5
31 CODE_WORD = 000B

131 CONH_1 039A

132 CONH_2 03A0

134 CONH_3 03B3

135 CONH_END 03B9

126 CONS_1 035F

129 CONS_2 037A

127 CONS_END 0382

130 CONVH_1 0392

133 CONVH_2 03AB

125 CONVS_1 0351

128 CONVS_2 0370

116 CONV_H_S 0389

60 CONV_S_H 0345

47 COUNT_B = 001D

46 COUNT_INT = 001C

184 DATA_ 06CF

136 DBG_1 03DB

138 DBG_2 03E3

139 DBG_3 03EB

140 DBG_4 03F3

137 DBG_END 03F9

159 DELAY 0660

39 DPC_H = 0013

40 DPC_L = 0014

3 DPH = 0083

181 DPH_ 06B4

TURBO51. 8051 Assembler Ver 1.00

Sequence Symbol name Value

4 DPL = 0082
182 DPL_ 06BD
185 DPTR_ 06D8
80 DUMP 04B9
190 DUMP_ 0708
164 DUMP_INT 053C
165 DUMP_INT1 0553
166 DUMP_INT2 055F
167 DUMP_INT3 056D
175 D_1 0662
188 EMTY 06F3
186 END_ 06E1
153 EPROM_128 047A
154 EPROM_256 047D
155 EPROM_512 0480
152 EPROM_64 0477
50 FLAG = 0020
170 GO_SGLE 05A5
30 HEX_CODE = 000A
124 HEX_L 036D
5 IE = 00A8
163 INT0_END 057C
55 INT_0 0003
56 INT_0_END 0013

6 IP = 00B8

172 LED 05E8

173 LED_1 05EC

144 LOAD_END 040E

65 LOAD_HEX 03FC

141 LOAD_NEW 0409

183 LONG_ 06C6

36 LONG_BYTE = 0010

62 MAIN 0100

41 MARK_PC = 0015

88 MENU_DBG 02AC

63 M_1 0107

81 M_10 0151

83 M_11 0157

85 M_12 0167

87 M_13 016D

89 M_14 0173

64 M_2 011D

67 M_3 0123

69 M_4 012B

71 M_5 0133

73 M_6 0139

75 M_7 013F

77 M_8 0145

79 M_9 014B

45 NUM_B = 001B

TURBO51. 8051 Assembler Ver 1.00

Sequence Symbol name Value

96 OFSET 045A
42 OFSET_H = 0016
43 OFSET_L = 0017
68 OK 0486
113 OK_ 0712
7 P0 = 0080
8 P1 = 0090
9 P2 = 00A0
10 P3 = 00B0
11 PCAP2H = 00CB
12 PCAP2L = 00CA
13 PCON = 0087
180 PC_ 06AC
37 PC_H = 0011
178 PC_H_ 069A
38 PC_L = 0012
179 PC_L_ 06A3
14 PSW = 00D0
151 RAM_256 0474
150 RAM_64 0471
189 READ_DATA 06F9
48 REAL_PCH_B = 001E
49 REAL_PCL_B = 001F
162 RUN_INT1 0520

```
 57     RUN_INT_0     0520
160     RUN_P1        04FA
161     RUN_PEND      0501
104     RUN_PEND1     050F
 70     RUN_PRO       04EB
 92     RX_DATA       05D3
171     RX_T1         05D7
 15     SBUF        = 0099
142     SCAN          040B
147     SCAN_D1       043B
149     SCAN_D2       044D
148     SCAN_D3       044B
143     SCAN_DATA     0423
146     SCAN_END      0459
145     SCAN_PC       042E
 16     SCON        = 0098
 32     SELECT_MEM  = 000C
120     SENDST_END    0313
110     SEND_CHAR     02DE
117     SEND_END      02F5
115     SEND_H_AS     02E6
114     SEND_ST       02F8
118     SEND_ST0      02FF
119     SEND_ST1      0307
 59     SERIAL_0      0029
 61     SERIAL_END    0042
```

TURBO51. 8051 Assembler Ver 1.00

Sequence Symbol name Value

58 SERIAL_INT 0023
86 SET_DBG 03BC
54 SET_SYS 017B
111 SGLE_1 0297
108 SGLE_END 05AB
107 SGLE_END1 0287
169 SINGLE_END 0593
168 SINGLE_R1 058C
72 SINGLE_RUN 057D
17 SP = 0081
187 SP_ 06EA
76 STOP_PRO 05BE
82 SWAP1 0648
84 SWAP2 0654
93 S_B0 01D8
95 S_B1 01DA
66 S_BREAK 01C9
94 S_END 01F4
18 T2CON = 00C8
176 TABLE1 066B
177 TABLE2 067B
19 TCON = 0088
20 TH0 = 008C
27 TH1 = 008D

ภาคผนวก ข. โปรแกรม MEMDP

```
/* Program comp1.c for include MEMDP Program */
#include <c:\att\be.c>
#include <c:\att\bd.c>

char sfr[40] ;
unsigned char reg[59];
char Bin[64] = {   '0','0','0','0' ,'0','0','0','1' ,'0','0','1','0'
                  ,'0','0','1','1' ,'0','1','0','0' ,'0','1','0','1'
                  ,'0','1','1','0' ,'0','1','1','1' ,'1','0','0','0'
                  ,'1','0','0','1' ,'1','0','1','0' ,'1','0','1','1'
                  ,'1','1','0','0' ,'1','1','0','1' ,'1','1','1','0'
                  ,'1','1','1','1'};
 void ao_disp()
  {
        strcpy(reg,sfr);
        printstrxy(39,4," ACC ",REV_COLOR);
        printstrxy(45,4," B ",REV_COLOR);
        printstrxy(49,4," DPH ",REV_COLOR);
        printstrxy(55,4," DPL ",REV_COLOR);
        printstrxy(61,4," P0 ",REV_COLOR);
        printstrxy(66,4," P1 ",REV_COLOR);
        printstrxy(71,4," P2 ",REV_COLOR);
        printstrxy(76,4," P3 ",REV_COLOR);

        printstrxy(39,7," IE ",REV_COLOR1);
        printstrxy(43,7,"EA - ET2 ES ET1 EX1 ET0 EX0 ",REV_COLOR);

        printstrxy(74,7," SBUF ",REV_COLOR);printstrxy(39,10," IP ",REV_COLOR1);
        printstrxy(43,10," - - PT2 PS PT1 PX1 PT0 PX0 ",REV_COLOR);
```

```
        printstrxy(74,10,"  SP  ",REV_COLOR); printstrxy(39,13," PSW ",REV_COLOR1);
        printstrxy(44,13," CY AC F0 RS1 RS0 OV - P",REV_COLOR);

        printstrxy (69,13," TH0 ",REV_COLOR);
        printstrxy (75,13," TL0 ",REV_COLOR);
        printstrxy (39,16," SCON",REV_COLOR1);
        printstrxy (44,16," SM0 SM1 SM2 REN TB8 RB8 TI RI",REV_COLOR);

        printstrxy (75,16," TH1 ",REV_COLOR);
        printstrxy (39,19," TCON ",REV_COLOR1);
        printstrxy (45,19," TF1 TR1 TF0 TR0 IE1 IT1 IE0 IT0",REV_COLOR);

        printstrxy (39,22," TMOD",REV_COLOR1);
        printstrxy (44,22," GATE C/T M1 M0 GATE C/T M1 M0",REV_COLOR);

        printstrxy (75,22," TL1 ",REV_COLOR);
        DispRD();

    }

void DispRD(void)
  {
 char num_hex=0,num_bin=0;

/* ACC,B,DPH,DPL,P0,P1,P2,P3 */
        WriteR(0,39,5);WriteR(2,45,5);WriteR(4,49,5);
        WriteR(6,55,5);WriteR(8,61,5);WriteR(10,66,5);
        WriteR(12,71,5);WriteR(14,76,5);
```

```
/*  SBUF,SP,TL0,TL1,TH0,TH1 */
        WriteR(16,75,8);WriteR(18,75,11);
        WriteR(20,70,14);WriteR(22,76,14);
        WriteR(24,76,17);WriteR(26,75,23);

/* IE : EA,ET2,ES,ET1,EX1,ET0,EX0 */
        WriteR(28,39,8);
        num_hex = reg[28];num_bin = htob(num_hex);textcolor(10);
        gotoxy(44,8);cprintf ("%c",Bin[num_bin]);
        gotoxy(46,8);cprintf ("%c",Bin[num_bin+1]);
        gotoxy(49,8);cprintf ("%c",Bin[num_bin+2]);
        gotoxy(53,8);cprintf ("%c",Bin[num_bin+3]);

        num_hex = reg[29];num_bin = htob(num_hex);textcolor(10);
        gotoxy(56,8);cprintf ("%c",Bin[num_bin]);
        gotoxy(60,8);cprintf ("%c",Bin[num_bin+1]);
        gotoxy(64,8);cprintf ("%c",Bin[num_bin+2]);
        gotoxy(68,8);cprintf ("%c",Bin[num_bin+3]);

/* IP : PT2,PS,PT1,PX1,PT0,PX0 */
        WriteR(30,39,11);
        num_hex = reg[30];num_bin = htob(num_hex);textcolor(10);
        gotoxy(44,11);cprintf ("%c",Bin[num_bin]);
        gotoxy(46,11);cprintf ("%c",Bin[num_bin+1]);
        gotoxy(49,11);cprintf ("%c",Bin[num_bin+2]);
        gotoxy(53,11);cprintf ("%c",Bin[num_bin+3]);

        num_hex = reg[31];num_bin = htob(num_hex);textcolor(10);
```

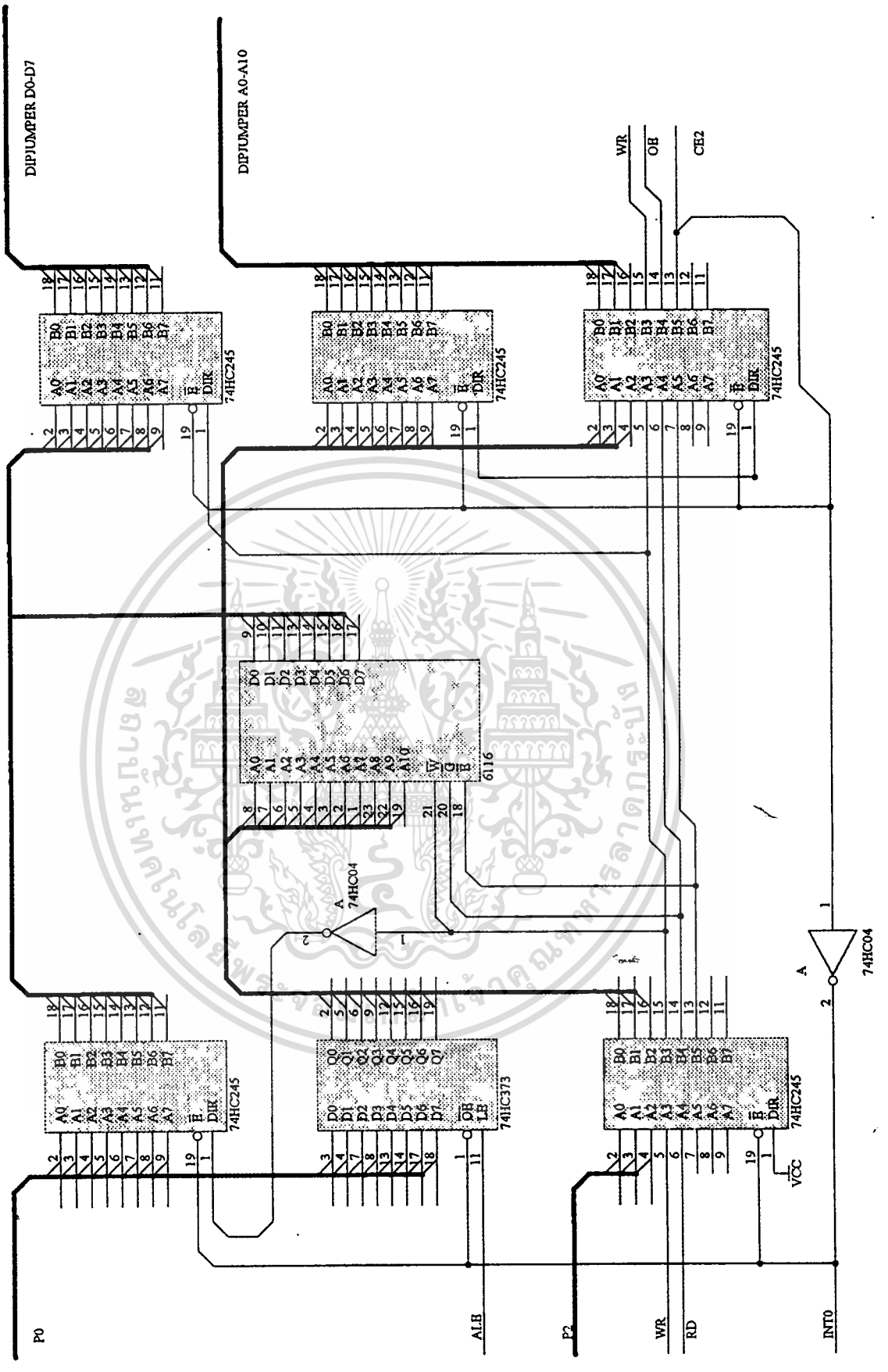
```
gotoxy(56,11);cprintf ("%c",Bin[num_bin]);
gotoxy(60,11);cprintf ("%c",Bin[num_bin+1]);
gotoxy(64,11);cprintf ("%c",Bin[num_bin+2]);
gotoxy(68,11);cprintf ("%c",Bin[num_bin+3]);

/* PSW : CY,AC,F0,RS1,RS0,OV,P */
 WriteR(32,40,14);
num_hex = reg[32];num_bin = htob(num_hex);textcolor(10);
gotoxy(46,14);cprintf ("%c",Bin[num_bin]);
gotoxy(49,14);cprintf ("%c",Bin[num_bin+1]);
gotoxy(52,14);cprintf ("%c",Bin[num_bin+2]);
gotoxy(55,14);cprintf ("%c",Bin[num_bin+3]);

 num_hex = reg[33];num_bin = htob(num_hex);textcolor(10);
gotoxy(59,14);cprintf ("%c",Bin[num_bin]);
gotoxy(63,14);cprintf ("%c",Bin[num_bin+1]);
gotoxy(65,14);cprintf ("%c",Bin[num_bin+2]);
gotoxy(67,14);cprintf ("%c",Bin[num_bin+3]);

/* SCON : SM0,SM1,SM2,REN,TB8,RB8,TI,RI */
 WriteR(34,40,17);
num_hex = reg[34];num_bin = htob(num_hex);textcolor(10);
gotoxy(46,17);cprintf ("%c",Bin[num_bin]);
gotoxy(50,17);cprintf ("%c",Bin[num_bin+1]);
gotoxy(54,17);cprintf ("%c",Bin[num_bin+2]);
gotoxy(58,17);cprintf ("%c",Bin[num_bin+3]);

 num_hex = reg[35];num_bin = htob(num_hex);textcolor(10);
 gotoxy(62,17);cprintf ("%c",Bin[num_bin]);
```

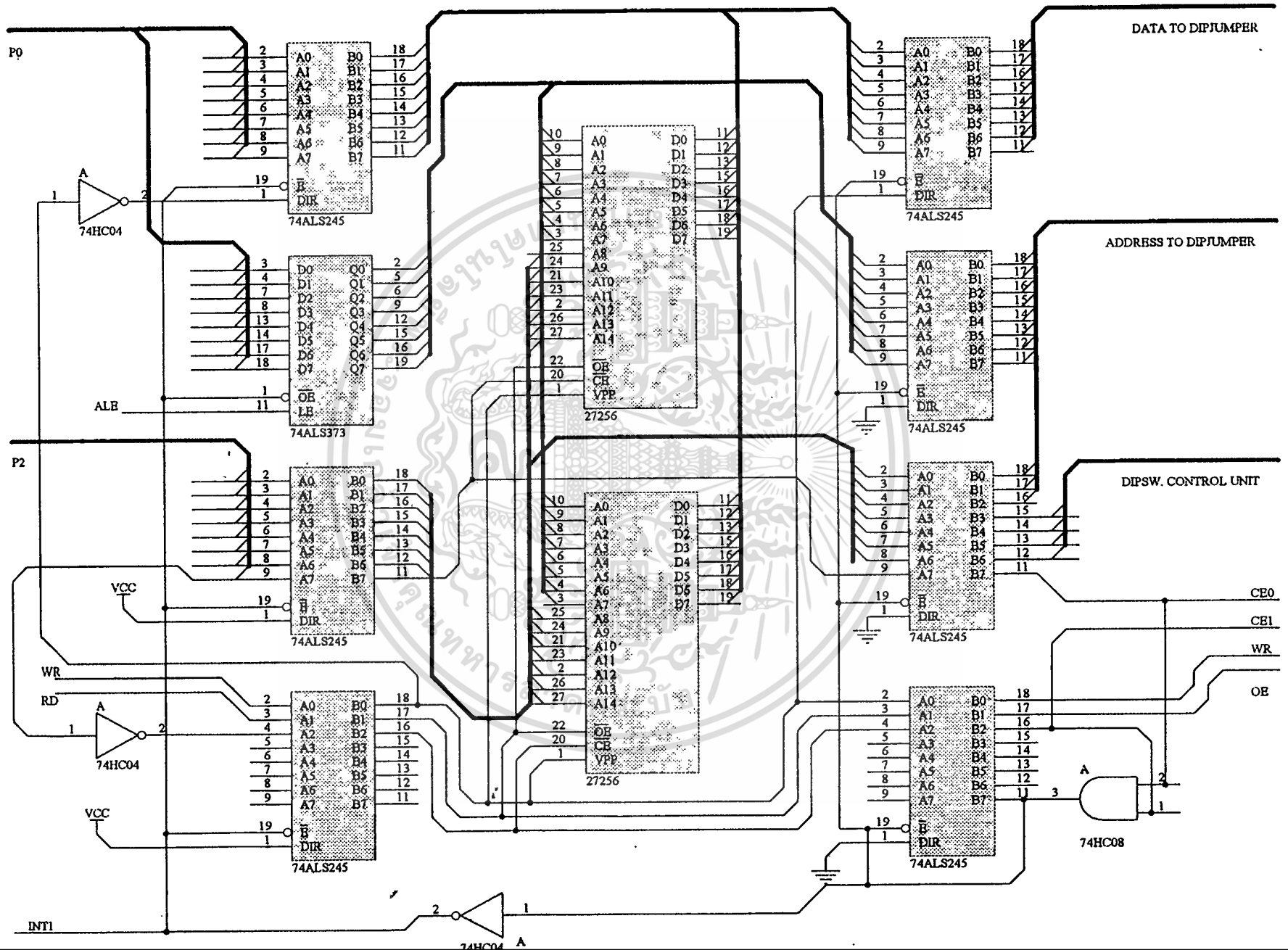
```
gotoxy(66,17);cprintf ("%c",Bin[num_bin+1]);
gotoxy(70,17);cprintf ("%c",Bin[num_bin+2]);
gotoxy(73,17);cprintf ("%c",Bin[num_bin+3]);

/* TCON : TF1,TR1,TF0,TR0,IE1,IT1,IE0,IT0 */
WriteR(36,40,20);
num_hex = reg[36];num_bin = htob(num_hex);textcolor(10);
gotoxy(47,20);cprintf ("%c",Bin[num_bin]);
gotoxy(51,20);cprintf ("%c",Bin[num_bin+1]);
gotoxy(55,20);cprintf ("%c",Bin[num_bin+2]);
gotoxy(59,20);cprintf ("%c",Bin[num_bin+3]);

num_hex = reg[37];num_bin = htob(num_hex);textcolor(10);
gotoxy(63,20);cprintf ("%c",Bin[num_bin]);
gotoxy(67,20);cprintf ("%c",Bin[num_bin+1]);
gotoxy(71,20);cprintf ("%c",Bin[num_bin+2]);
gotoxy(75,20);cprintf ("%c",Bin[num_bin+3]);

/* TMOD : GATE,C/T,M1,M0,GATE,C/T,M1,M0 */
WriteR(38,40,23);
num_hex = reg[38];num_bin = htob(num_hex);textcolor(10);
gotoxy(46,23);cprintf ("%c",Bin[num_bin]);
gotoxy(51,23);cprintf ("%c",Bin[num_bin+1]);
gotoxy(55,23);cprintf ("%c",Bin[num_bin+2]);
gotoxy(58,23);cprintf ("%c",Bin[num_bin+3]);

num_hex = reg[39];num_bin = htob(num_hex);textcolor(10);
gotoxy(61,23);cprintf ("%c",Bin[num_bin]);
gotoxy(66,23);cprintf ("%c",Bin[num_bin+1]);
```

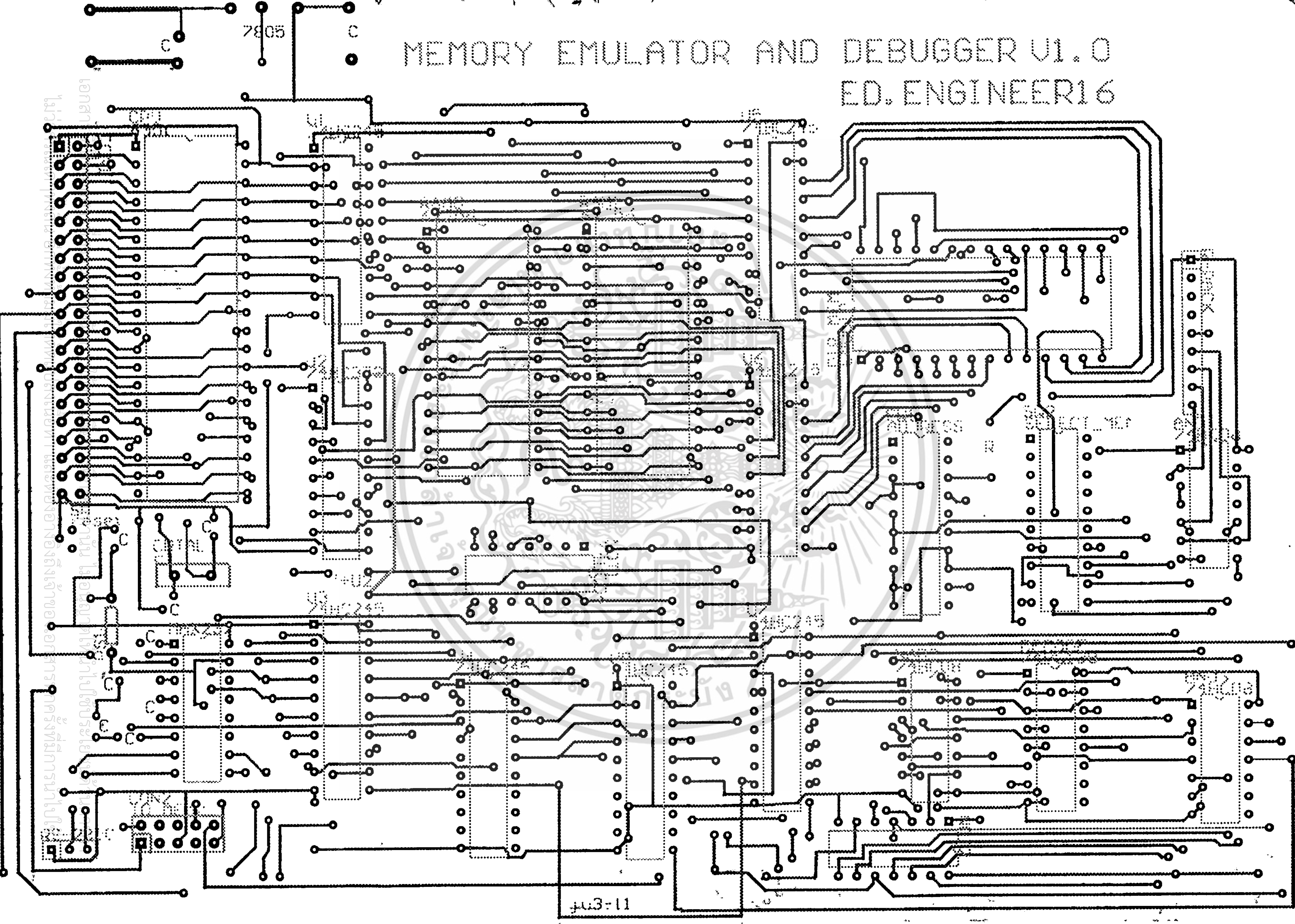
```
        gotoxy(70,23);cprintf ("%c",Bin[num_bin+2]);
        gotoxy(73,23);cprintf ("%c",Bin[num_bin+3]);
        textcolor(-1);
 }

char htob(char hex)
{
 char aa;
 switch (hex)
        {
        case '0' : aa = 0;break;
        case '1' : aa = 4;break;
        case '2' : aa = 8;break;
        case '3' : aa = 12;break;
        case '4' : aa = 16;break;
        case '5' : aa = 20;break;
        case '6' : aa = 24;break;
        case '7' : aa = 28;break;
        case '8' : aa = 32;break;
        case '9' : aa = 36;break;
        case 'A' : aa = 40;break;
        case 'B' : aa = 44;break;
        case 'C' : aa = 48;break;
        case 'D' : aa = 52;break;
        case 'E' : aa = 56;break;
        case 'F' : aa = 60;break;
        }return aa;
}
```

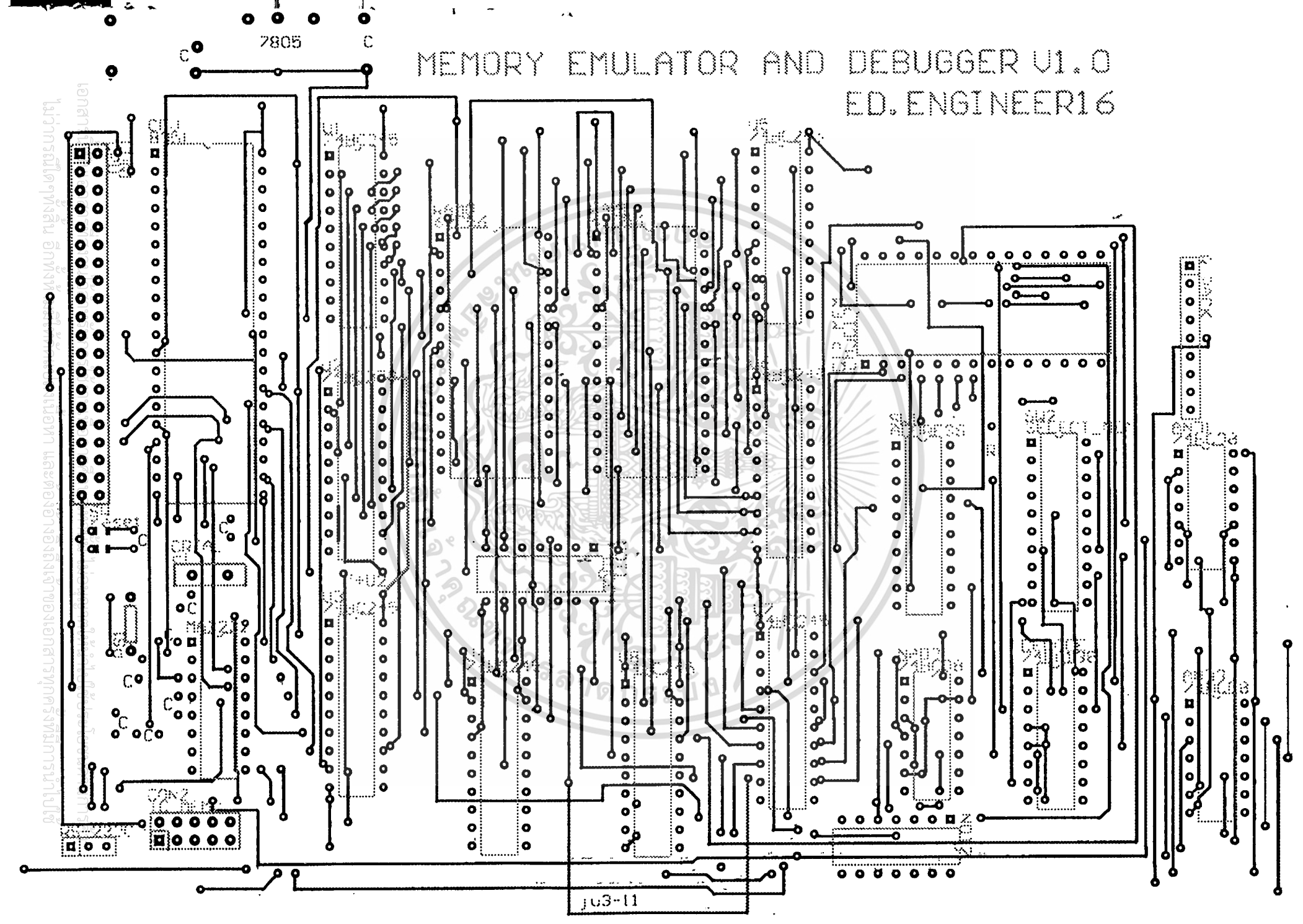
```
void WriteR(int count,int X,int Y)
 {
   textcolor(10);gotoxy(X+1,Y);cprintf ("%c",reg[count]);
   textcolor(10);gotoxy(X+2,Y);cprintf ("%c",reg[count+1]);
  }
```

ภาคผนวก ค. ลายวงจรและแผ่นวงจรพิมพ์




P0
P2
ALE
WR
RD
INT1
VCC
74HC04
74ALS245
74ALS373
27256
74HC08
DATA TO DIPJUMPER
ADDRESS TO DIPJUMPER
DIPSW. CONTROL UNIT
CE0
CE1
WR
OE


MEMORY EMULATOR AND DEBUGGER V1.0
ED.ENGINEER16
7805
C
CRYSTAL
74HC245
74HC573
ADDRESS
SELECT_MEM
ADDRESS
CONFIG
BACK
ON_OFF
A03-11


MEMORY EMULATOR AND DEBUGGER V1.0
ED.ENGINEER16
7805
C
jc3-11

บรรณานุกรม

1. วิบูลย์ จันแขก. ไมโครโปรเซสเซอร์. กรุงเทพฯ : สำนักพิมพ์สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ, 2532.

2. ยืน ภู่วรวรรณและวัฒนา เชียงกูล, ไมโครโปรเซสเซอร์ ไมโครคอมพิวเตอร์. กรุงเทพฯ :เอช-เอนการพิมพ์,2534

3. ปรเมษฐ์ ประยานันทม์และปิยพงศ์ เผ่าวณิช, คู่มือและการประยุกต์ใช้งาน ไมโครโปรเซสเซอร์ MCS-51. กรุงเทพฯ : บริษัท เอช-เอน กรุ๊ป จำกัด, 2536.

4. เฉลิมพันธ์ ยศสมบัติ. และคนอื่นๆ. "ชุดฝึกระบบปฏิบัติการไมโครโปรเซสเซอร์," ปริญญานิพนธ์ครุศาสตร์อุตสาหกรรมบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม ครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2537.

5. Douglas V.Hall. Microprocessors and Interfacing : Programming and Hardware. Second Edition New York : McGraw-Hill, 1992.