ปีการศึกษา 2531

ปริญญานิพนธ์ เรื่อง

Automatic Video Switcher

โดย

ปรีชา กรปรีชา

อาจารย์ที่ปรึกษา

อ.ประดิษฐ์ วัชรพิบูลย์

ปริญญานิพนธ์ปีการศึกษา 2531

ภาควิชา เทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง Automatic Video Switcher

ผู้จัดทำ

1. ปรีชา กรปรีชา

.....อาจารย์ที่ปรึกษา

(นายประดิษฐ์ วัชรนิบูลย์)

.....กรรมการ

(ผศ.วิชัย สุรพัฒน์)

.....กรรมการ

(อ.อุทัย ศรีธีรวิโรจน์)

AUTOMATIC VIDEO SWITCHER

ปรีชา กรปรีชา

ประดิษฐ์ วัชรนิบูลย์ อ.ที่ปรึกษา

บทคัดย่อ

AUTOMATIC VIDEO SWITCHER เป็นเครื่องมือชนิดหนึ่ง ซึ่งจะมีหน้าที่ในการเลือกสัญญาณภาพจากแหล่งกำเนิดสัญญาณหลายๆ แห่ง ซึ่งในที่นี้เครื่องจะรับสัญญาณต่างๆเข้ามาทาง INPUT ได้ 32 ช่อง ซึ่งจะไม่คำนึงว่าสัญญาณซิงค์ของแต่ละแหล่งกำเนิดสัญญาณจะตรงกันหรือไม่ โดยการให้สวิทช์เลือกสัญญาณภาพในแต่ละครั้งจะกระทำในช่วงระหว่างการสิ้นสุดการสะแกนในฟิลด์หนึ่ง หรือ ในช่วง VERTICAL BLANKING PERIOD จึงทำให้ภาพที่ได้ไม่ล้มหรือ เลื่อน และเห็นการกระพริบน้อยมาก ส่วนทาง OUTPUT สามารถจะกระจายสัญญาณออกไปได้ถึง 2 CHANNEL ซึ่งเหมาะที่จะนำไปใช้ในสตูดิโอ หรืองานในด้านตัดต่อสัญญาณภาพ

หลักการทำงานของเครื่อง AUTOMATIC VIDEO SWITCHER ก็คือ

1. ใช้อิเลคทรอนิคส์สวิทช์เลือกสัญญาณภาพ
2. สัญญาณต่างๆ ที่เข้ามาทางช่องสัญญาณ สามารถเลือกกใช้ได้ตั้งแต่ 1-32 ช่อง
3. ส่วนทาง OUTPUT สามารถจะกระจายออกได้ถึง 2 ช่อง

AUTOMATIC VIDEO SWITCHER

PREECHA KORNPREECHA

PRADIT WATCHARAPIBOOL ADVISOR

1989

Abstract

Automatic Video Switcher is the equitment for selecting the picture signal from several places of signal source. this equitpment can receive signal for 32 channels via an input no matter signal sync from each source will comply with each other or not.The switcher will select picture siqnal each time by scaning of each field or during vertical blanking period,this will make the stable picture and have a minimum blank.For the output,the signal can be spread to 2 chennels, as a resuit, this equipment is appropiate for studio or for picture signal splicing

Working standard of Automatic Video Switcher are

1. Use the switch to select picture signal
2. We can select 132 channels for all the signal via an input
3. For output signal can be spread to 2 channels

สารบัญ

บทที่ 1

บทนำ

ในปัจจุบันการพัฒนาทางการด้านอิเลคทรอนิคส์ได้พัฒนาไปมาก และมีแนวโน้มที่จะพัฒนาก้าวหน้ายิ่งขึ้นไปอีก อุปกรณ์ทางด้านอิเลคทรอนิคส์ในปัจจุบันมีขนาดเล็กลง แต่ประสิทธิภาพยังรักษาให้อยู่ในระดับเดิม ในด้านเทคโนโลยีโทรทัศน์ก็เช่นกัน ได้พัฒนาอย่างรวดเร็วมากหลอดถ่ายภาพในปัจจุบันก็ได้มีการพัฒนาให้มีขนาดเล็กลง และน้ำหนักเบา

อุปกรณ์ภายในสตูดิโอในปัจจุบันได้มีการพัฒนาไปมากเช่นเดียวกัน ซึ่งในปัจจุบันได้มีการนำเอาระบบคอมพิวเตอร์มาใช้ในการควบคุมการทำงานของอุปกรณ์ ซึ่งการทำงานเป็นไปอย่างรวดเร็ว และการตัดต่อภาพก็สามารถที่จะทำ effect ต่างๆได้อย่างมาก

Video switcher เป็นอุปกรณ์ที่ใช้ในสตูดิโอ ซึ่งทำหน้าที่ในการเลือกสัญญาณจากแหล่งกำเนิดสัญญาณภาพหลายชนิด เช่น เครื่องบันทึกภาพ (Video Tape Record ER.) เครื่องรับภาพจากจากฟิมส์ภาพยนต์ (Telecine) กล้องถ่ายภาพหรือเครื่องฉายสไลท์ออกมาเพียงสัญญาณเดียวเท่านั้น

ในโครงงานนี้เป็นการออกแบบสร้าง Video Switcher ที่ใช้สำหรับการเลือกสัญญาณภาพและเสียงโดยที่สัญญาณเป็นสัญญาณที่มาจากแหล่งกำเนิดหลายชนิดและไม่จำกัดว่ามีเฟสของสัญญาณซิงค์ทางด้านแนวตั้งของช่องที่ถูกเลือกเข้ามา ซึ่งเป็นช่วงเวลาที่ภาพมืดมีช่วงเวลาเท่ากับ 20 H หรือ เท่ากับ 1.28 MS

คุณสมบัติในการทำ Video Swither

1.สามารถเลือกสัญญาณภาพจากแหล่งกำเนิดสัญญาณหลายชนิด โดยที่ไม่จำกัดว่าสัญญาณซิงค์ของภาพแต่ละตำแหน่งกำเนิด (แต่ละช่อง) จะตรงกันหรือไม่

2.ช่องสัญญาณอินพุททั้งภาพและเสียงสามารถใช้ได้ 32 ช่อง

3.ที่อินพุทของเครื่องของภาพสามารถกระจายสัญญาณที่เลือกออกได้ เอาว์พุทเดียว

4.ใช้อิเลคทรอนิคส์สวิทช์เป็นตัวเลือกสัญญาณเพื่อหลีกเลี่ยงปัญหาที่เกิดจากหน้าสัมผัสของรีเลย์

บทที่ 2

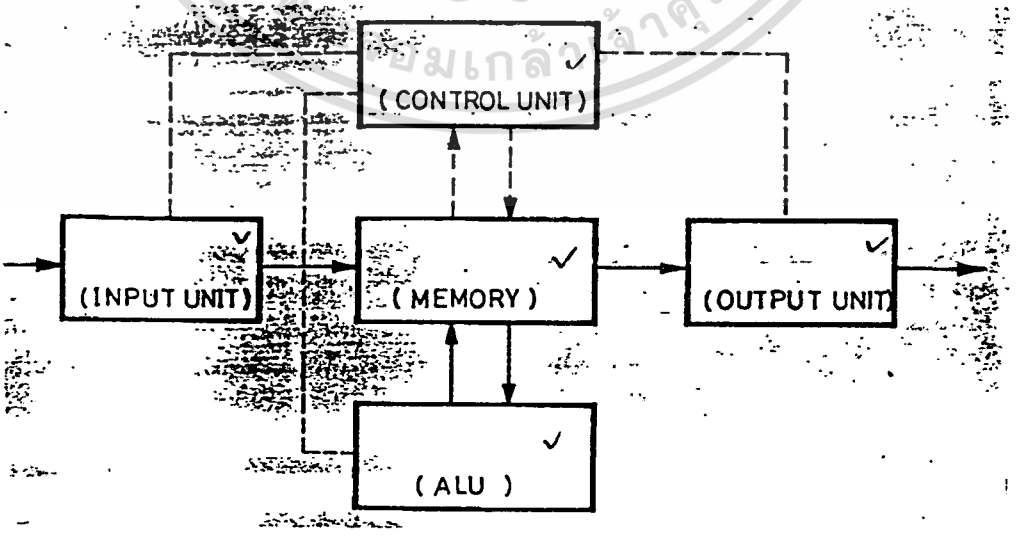
ทฤษฎีเกี่ยวกับไมโครโปรเซสเซอร์และไมโครคอมพิวเตอร์

ในระยะเวลาหลายปีที่ผ่านมา คอมพิวเตอร์ได้พัฒนาและเปลี่ยนแปลงไปอย่างมากและมีแนวโน้มถึงการเปลี่ยนแปลงใหม่ๆ มีมากขึ้นในอนาคต ดิจิตอลคอมพิวเตอร์ที่มีอยู่ปัจจุบันสามารถทำงานได้เร็วมีขนาดเล็กและเชื่อถือได้พร้อมทั้งราคาถูกกว่ารุ่นก่อนๆ อีกด้วย เทคโนโลยีใหม่ๆ และโครงสร้างภายในมีความแตกต่างกันไปพร้อมๆ กับอุปกรณ์อื่นๆเช่น หน่วยความจำ เป็นต้น ซึ่งได้พัฒนาอย่างรวดเร็ว จะทำให้คอมพิวเตอร์มีบทบาทต่อชีวิตคนเรามากขึ้นและเราก็จำเป็นจะต้องเรียนรู้เพื่อที่จะใช้มัน

เครื่องคอมพิวเตอร์เป็นสมองอิเลคทรอนิคส์ที่ทำงานอย่างหนึ่งได้เร็ว และ ดีกว่าคนมาก มันสามารถรับข้อมูลได้มากและแยกแยะได้เร็วกว่าคน ด้วยความเร็วนี้เองงานชิ้นเดียวกันอาจจะต้องใช้เวลาคนทำถึงเป็นอาทิตย์เป็นเดือน แต่คอมพิวเตอร์อาจใช้ เวลาเพียงวินาทีเดียวเท่านั้น ด้วยเหตุนี้เราจึงใช้ความสามารถของคอมพิวเตอร์ มาประมวลข้อมูลจำนวนมากๆในเวลาสั้นๆ และด้วยความเร็วและความสามารถสูงของคอมพิวเตอร์นี้เราจึงนำมาใช้งานในด้านธุรกิจอุตสาหกรรม และห้องทดลอง เพื่อลดรายจ่าย เช่น คอมพิวเตอร์กับการออกแบบ คอมพิวเตอร์กับนักวิทยาศาสตร์ คอมพิวเตอร์กับอุตสาหกรรม คอมพิวเตอร์กับการพัฒนาบุคคลากร เป็นต้น

ไมโครโปรเซสเซอร์และไมโครคอมพิวเตอร์

คอมพิวเตอร์โดยทั่วๆไป โครงสร้างพื้นฐานจะแบ่งออกเป็นส่วนต่างๆ ดังรูป2.1



รูปที่ 2.1เส้นประแสดงสัญญาณควบคุมและสัญญาณเลือก เส้นทึบแสดงสัญญาณข้อมูล

จะเห็นว่าระบบคอมพิวเตอร์ประกอบด้วย

1.หน่วยควบคุม (control unit)

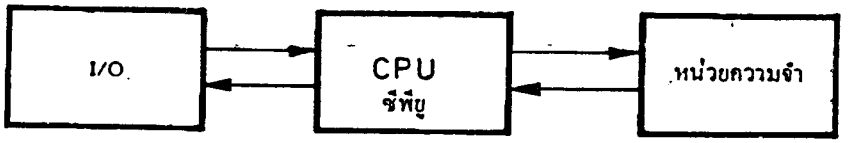
2.หน่วยความจำ (Memory unit)

3.หน่วยคำนวณ (Arithmatic unit)

4.หน่วยส่งและรับสัญญาณ (I/O unit)

เครื่องจักรที่เราเรียกว่าคอมพิวเตอร์จะต้องมีความสามารถทำการคำนวณทางคณิตศาสตร์และทำฟังก์ชั่นลอจิกได้ หน่วยที่รับผิดชอบด้านคำนวณในเครื่องคอมพิวเตอร์ คือหน่วยคำนวณ (Arithmatic and logical unit) เมื่อหน่วยคำนวณจะสามารถฟังก์ชั่นใด ๆ ก็ต้องมีผู้บอกหรือควบคุม ซึ่งอาศัยหน่วยควบคุม (Control unit) เมื่อทำการคำนวณก็ต้องมีข้อมูลที่จะนำมาคำนวณและคำนวณแล้วก็ต้องมีที่เก็บผล จึงจำเป็นต้องมีหน่วยความจำ(Memory unit) ซึ่งจะเป็นตัวเก็บและให้ข้อมูลแก่หน่วยคำนวณจุดประสงค์ ของการออกแบบคอมพิวเตอร์มาก็เพื่อการใช้งานซึ่งต้องติดต่อเอาข้อมูลจากภายนอก และยังต้องแสดงข้อมูลให้ภายนอกได้ด้วย จึงต้องมีหน่วยส่งและรับสัญญาณ (I/O unit)

ที่กล่าวมานี้เป็น โครงสร้างทั่วไปของคอมพิวเตอร์แบ่งตามหน้าที่ เนื่องจากเทคโนโลยีด้านการผลิตอุปกรณ์อิเลคทรอนิกส์ได้พัฒนาไปมาก ทำให้สามารถรวบรวมหน่วยต่าง ๆ ของคอมพิวเตอร์อันได้แก่หน่วยควบคุม (Control unit) หน่วยความจำบางส่วนหน่วยคำนวณ (AUL) วงจรควบคุม I/O บางส่วนทั้งหมดนี้รวมไว้ในแผ่นวงจรเดียวกัน เรียกว่า ไมโครโปรเซสเซอร์ (Microprocessor) ซึ่งสามารถทำหน้าที่ประมวลข้อมูลและควบคุมหน่วยอื่นๆ ให้ทำงานไปด้วยกันได้ คอมพิวเตอร์ที่มีโปรเซสเซอร์เป็นหน่วยประมวลกลาง เราเรียกคอมพิวเตอร์นั้นว่า ไมโครคอมพิวเตอร์ (Mierocomputer) ดังแสดงในรูป 2.2



รูป 2.2 ไมโครคอมพิวเตอร์

ไมโครโปรเซสเซอร์เทคโนโลยี

เทคโนโลยีของแอลเอสไอ (Large Scale Integration) ทำให้เราสามารถบรรจุวงจรอิเลคทรอนิกส์ลงไปบนแผ่นวงจรแผ่นเดียวได้ เรียกแผ่นวงจรนั้นว่า ชิพ (Chip) เทคโนโลยี LSI นี้ถูกนำมาใช้ในผลิตไมโครโปรเซสเซอร์ที่จำหน่ายอยู่ในท้องตลาด โดยทั่วไปแบ่งออกตามตระกูลหรือเทคโนโลยีที่นำมาได้ดังต่อไปนี้

1.พวกไบโพลา พวกนี้ได้แก่ลอจิกประกอบด้วยทรานซิสเตอร์ ซึ่งทำงานได้ด้วยกระแสทั้งสองประเภทคือ ลบและบวก จึงเรียกว่าไบโพล่า (Biopar) ตระกูลนี้แบ่งเป็นพวกย่อยๆ ตามลักษณะวงจรคือ

1.1 ECL (Emitter Coupling Logic)

1.2 IIL (Integrated Injection Logic)

1.3 TTI (Transistor Transistor Logic)

1.4 STTL (Schottky Transistor Transistor Logic)

ทั้งสี่พวกนี้มีความแตกต่างกันเรื่อง ความเร็วภายในตระกูลไบโพลาเองความเร็วก็แตกต่างกัน แต่โดยรวมแล้วตระกูลนี้มีความเร็วมากกว่าตระกูลอื่น

2. พวก Mos (Metal Oxide Semieonductor) พวกนี้ลอจิกเป็นทรานซิสเตอร์เช่นกันแต่เป็นทรานซิสเตอร์ชนิด Mosfet แบ่งออกเป็นพวกตามลักษณะวงจรและสารที่ทำคือ

PMOS (P-channel Metal Oxide Semiconductor)

NMOS (N-channel Metal Oxide Semiconductor)

CMOS (Comolementary Metal Oxide Semiconductor)

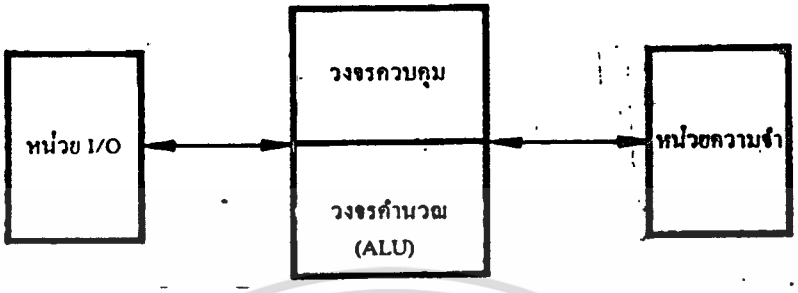
พวก MOS มีคุณสมบัติเด่นมากเรื่องกินกำลังไฟต่ำ

ในการเลือกไมโครโปรเซสเซอร์ที่เหมาะสมสำหรับงานต่างๆ จะต้องพิจารณาข้อมูลให้ละเอียดจึงจะได้ระบบที่เหมาะกับงานมากที่สุด ข้อนี้ขึ้นอยู่กับข้อจำกัดในการเลือกใช้เองนอกจากเทคโนโลยีต่างๆ แล้วโครงสร้างภายในไมโครโปรเซสเซอร์เอง แต่ละเบอร์ก็แตกต่างกันไป โครงสร้างภายในหมายถึง ขนาดของข้อมูล จำนวนคำสั่ง การตอบรับสัญญาณ จากภายนอกอันนี้จะต้องศึกษารายละเอียดของแต่ละเบอร์ต่างๆกัน

ระบบไมโครคอมพิวเตอร์

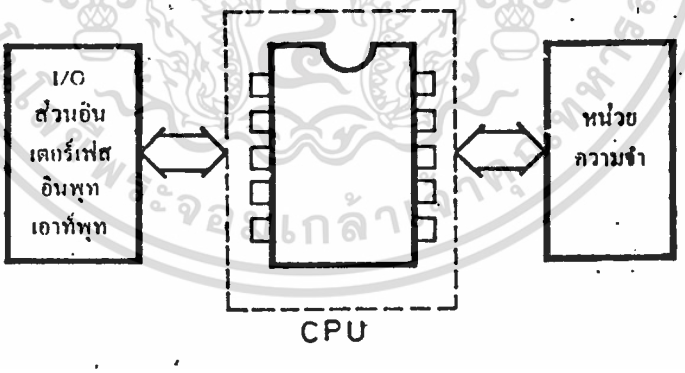
ไมโครคอมพิวเตอร์ คือ คอมพิวเตอร์ที่มีไมโครโปรเซสเซอร์ เป็นตัวประมวลผลกลางหรือ ซีพียู (CPU) และทำงานร่วมกับหน่วยอื่นอีก 2 หน่วยคือหน่วยความจำ (Memory unit)

หน่วยรับส่งสัญญาณเข้าออก (Input Output unit) เขียนเป็นบล็อกไดอะแกรมได้ดังนี้



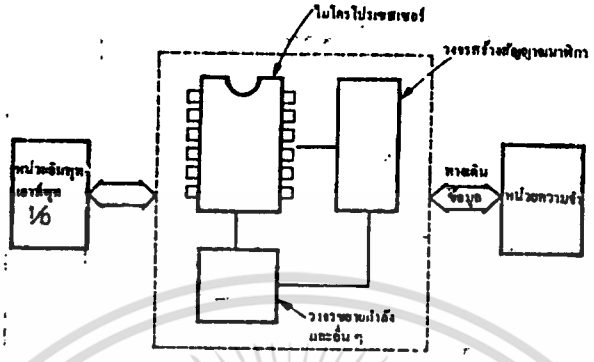
รูป 2.3 บล็อกไดอะแกรมของไมโครคอมพิวเตอร์

หน้าที่ของ CPU นอกจากจะคำนวณทางคณิตศาสตร์แล้วยังส่งสัญญาณควบคุม หรือให้จังหวะแก่หน่วยอื่นๆ ให้ทำงานไปพร้อมๆกับมันได้ด้วย ภายในตัว CPU แบ่งออกเป็นสองส่วนใหญ่ๆ คือ วงจรควบคุม และวงจรคำนวณ เราสามารถแทนบล็อกไดอะแกรม CPU ด้วย รูปตัวจริงของไมโครโปรเซสเซอร์ได้



รูป 2.4

ดังรูป 2.4 เฉพาะตัวไมโครโปรเซสเซอร์ใดๆ ก็ยังทำอะไรไม่ได้ ตัวมันเองทำงานเป็นลำดับขั้นในวงจรก็จริงแต่ก็ยังต้องอาศัยวงจรส่วนประกอบอื่นๆ ที่เพิ่มเติมเข้ามาเพื่อช่วยให้ CPU ทำงานได้และติดต่อควบคุมหน่วยอื่นๆได้ ส่วนที่เพิ่มเติมเข้ามาคือ วงจรสัญญาณนาฬิกา วงจรสัญญาณนาฬิกา วงจรถอดรหัส วงจรขยายกำลัง ตามรูป 2.5



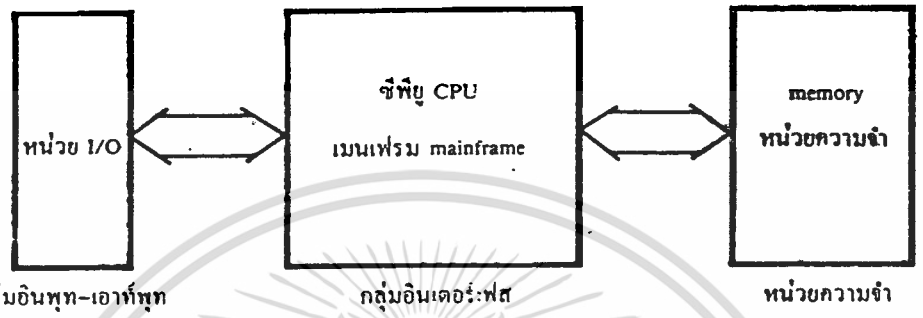
รูป 2.5

ในระดับของวงจรต่างๆ ที่เกี่ยวข้องกับทั้งระบบ ไม่ได้จบสิ้นอยู่ที่ตัวไมโครโปรเซสเซอร์เท่านั้นวงจรประกอบอื่นๆ ยังมีอีกมาก แต่ละวงจรมีหน้าที่สำคัญต่อระบบ เช่น เดียวกันจึงจะทำให้ไมโครโปรเซสเซอร์ติดต่อกับหน่วยควบคุมต่างๆ ได้ วงจรที่สำคัญที่ช่วยต่อเชื่อม CPU กับหน่วยอื่นๆ เราเรียกวงจรอินเตอร์เฟส (Interface Circuit) วงจรอินเตอร์เฟสมีตั้งแต่วงจรถอดระหัส วงจรขับกำลังวงจรควบคุมการส่งผ่านสัญญาณ ความสลับซับซ้อนของวงจร Inteface จะมีมากเพียงใดขึ้นอยู่กับชนิดของหน่วยต่างๆที่ CPU ต้องติดต่อ

- หน่วย I/O หน่วย I/O ได้แก่ส่วนแสดงผลหรือใส่ข้อมูลที่เราใช้งาน มีมากมายหลายชนิด เช่น เครื่องพิมพ์ชนิดต่างๆ เครื่องพิมพ์ชนิดต่างๆ เครื่องเจาะเทป อ่านเทป ทีวีแสดงผลเป็นตัวหนังสือ (CRS) แป้นคีย์บอร์ด (Key Board)

-หน่วยความทรงจำ เป็นส่วนเก็บคำสั่งและข้อมูลต่างๆ แยกแยะตามชนิดได้มากมาย เช่น หน่วยความจำแบบสารกึ่งตัวนำ (Semiconductor memory) หน่วยความจำขนาดใหญ่ ได้แก่แผ่นแม่เหล็ก (Disk) แบบอ่านเรียกว่าฟลอบบี้ดิสก์ (Floppy disk) แบบแข็งเรียกว่าฮาร์ดดิสก์ (Hard disk) เป็นต้น

จากรูป 2.5 มาเขียนแยกประเภทออกจะได้ดังนี้



รูป 2.6 ระบบไมโครคอมพิวเตอร์

ดังกล่าวมาแล้วว่าไมโครคอมพิวเตอร์ประกอบด้วยหน่วยต่างๆคือ หน่วยควบคุม หน่วยคำนวณ หน่วยความจำ และหน่วยรับส่งข้อมูล วงจรอิเลคทรอนิคส์ที่ทำหน้าที่ของหน่วยต่างๆนี้เรียกว่า ฮาร์ดแวร์ (Hardware) วงจรเหล่านี้ทำหน้าที่ต่างกัน เช่นหน่วยควบคุมจะถูกออกแบบมาให้เข้าใจหรือตีความหมายรหัสที่อยู่ในรูปตัวเลข ชุดหนึ่งซึ่งสามารถเก็บไว้ในหน่วยความจำได้ ระหัสนี้ถือว่าเป็นคำสั่งสำหรับคอมพิวเตอร์หรือเรียกว่า Intruction คอมพิวเตอร์เครื่องหนึ่งจะมีระหัสที่หน่วยควบคุมเข้าใจได้อยู่ชุดหนึ่ง บางเครื่องมีมากกว่า 100 คำสั่ง คำสั่งหลายๆคำสั่งสามารถบังคับให้คอมพิวเตอร์ทำอะไรก่อนหลังเป็นขั้นตอนจนได้คำตอบสุดท้ายที่เราต้องการชุดของคำสั่งเราเรียกว่าโปรแกรม (Program) หรือ Soft ware ของเครื่องโปรแกรมสำหรับให้คอมพิวเตอร์ทำงานแต่ละงานย่อมไม่เหมือนกัน เพราะแต่ละงานย่อมมีขั้นตอนปลีกย่อยแตกต่างกัน

หน่วยความจำ

หน่วยความจำหลักในระบบไมโครคอมพิวเตอร์เป็นความจำระบบอิเลคทรอนิคส์สามารถจำข้อมูลได้ ระบบอิเลคทรอนิคส์นี้ข้อมูลหรือระหัสจะอยู่ในรูปเลขไบนารี่ หรือ เลขฐานสอง เลขไบนารี่ 1 หลักเราเรียกว่า 1 บิท เลขไบนารี่หลายๆ บิท ที่คอมพิวเตอร์ถือเป็นคำสั่งหรือข้อมูลที่คอมพิวเตอร์รับเข้ามาหรือส่งไป เราเรียกว่า คำ (word) จำนวนเลขไบนารี่ที่ประกอบเป็นระหัส 1 คำคือเป็นความยาวของคำหรือ word length ของคำนั้นคำที่ประกอบด้วยเลข

ไบนารี่ 8 หลักเราเรียกว่า 1 ไบท์ (Byte) ซึ่งเป็นขนาดของคำที่ไมโครคอมพิวเตอร์ทั่วๆ ไปใช้อยู่ หน่วยความจำจะสามารถเก็บระหัสไว้ได้เป็นจำนวนมาก โดยแต่ละคำจะถูกเก็บไว้ในตำแหน่งใดตำแหน่งหนึ่งในหน่วยความจำ แต่ละตำแหน่งเป็นที่อยู่ ที่สามารถอ้างอิงได้เรียกว่าที่อยู่ ตามรูปเป็นหน่วยความจำที่สามารถเก็บระหัสขนาดคำละ 8 บิท ทั้งหมด 100 คำมีอยู่ตั้งแต่ 0 ถึง 99 สมมุติว่าระหัส M ถูกเก็บไว้ที่ แอดแดรส 20 เมื่อคอมพิวเตอร์เรียกข้อมูลจากตำแหน่ง 20 คอมพิวเตอร์จะได้ ภายในระบบไมโครคอมพิวเตอร์นอกจากจะมีหน่วยคำนวณ หน่วยควบคุมและ หน่วยความจำปฐมภูมิและยังมี วงจรความทรงจำ ซึ่งหน่วยคำนวณและหน่วยควบคุมใช้เป็นที่เก็บข้อมูลชั่วคราวอีกด้วย หน่วยความจำชั่วคราวนี้เรียกว่า รีจิสเตอร์ จำนวนรีจิสเตอร์ ที่อยู่ในไมโครโปรเซสเซอร์แต่ละตัวไม่เท่ากัน โดยแต่ละรีจิสเตอร์ มีชื่อเรียกเฉพาะ เช่น รีจิสเตอร์ที่เป็นที่เก็บผลลัพท์ จากหน่วยคำนวนเราเรียก แอดคิวมูเลเตอร์

หน่วยคำนวณ

หน่วยคำนวณจะทำหน้าที่ทางคณิตศาสตร์ และลอจิกโดยตรง ฟังก์ชั่นพื้นฐานของ หน่วยคำนวนคือ บวก ลบ และฟังก์ชั่นทางลอจิก การบวก ลบ ตัวเลขแต่ละครั้งจะได้ผลลัพธ์ ซึ่งจะถูกเก็บไว้ในหน่วยความจำชั่วคราวเรียกว่าแอคคิวมูเลเตอร์ (Accumulator Registor) การแสดงผลของการบวกลบที่ได้หน่วยคำนวนจะแสดงออกทางบิทแฟลก (Flag bit) ซึ่งก็คือ บิท ใดๆ ในหน่วยความจำแสดงสถานะผลลัพท์ซึ่งเราเรียกว่า Status Registor

หน่วยควบคุม

หน่วยควบคุมจะเป็นหัวใจการดำเนินงานทั้งหมด โดยจะส่งสัญญาณควบคุมให้จังหวะแก่หน่วยอื่นๆ เช่น ส่งสัญญาณบังคับให้หน่วยคำนวณทำฟังกชั่นใดๆ หรือส่งสัญญาณไปอ่านคำสั่งที่ถูกเก็บไว้ในหน่วยความจำ การทำงานของหน่วยควบคุมนี้คือ เมื่อเริ่มการทำงานหน่วยควบคุมจะรับคำสั่งจากหน่วยความจำแล้วแปลคำสั่งนั้น และส่งสัญญาณอื่นๆเท่าที่จำเป็นไปควบคุม

การกระทำตั้งแต่หน่วยควบคุมเริ่มไปอ่านคำสั่งมาจากหน่วยความจำแล้ว นำมาถอดระหัสจนถึงนำคำสั่งชั่งลงไปเราเรียกว่าหนึ่งรอบคำสั่ง(intruction cycle) แต่ละรอบคำสั่งจิงประกอบด้วยขั้นตอนย่อยๆคือ สภาวะเฟทช์-คือขบวนการที่หน่วยควบคุมไปเอาคำสั่งมาจากหน่วยความจำแล้วถอดระหัส คำสั่งนั้น สภาวะเอ็กซคิ้ว คือขบวนการตั้งแต่หน่วยควบคุมเริ่มทำตามคำสั่งนั้นจนเสร็จคำสั่งเราจะเขียนตารางของคอมพิวเตอร์ได้เป็น

จะเห็นได้ว่าในรอบคำสั่งหนึ่งๆมีขั้นตอนรองลงมาคือ รอบสภาวะเฟทช์และรอบสภาวะเอ็กซ์คือที่จริงแล้วมันประกอบด้วยขั้นตอนย่อยๆลงไปอีกคือ แมชีนไซเคิล(Machine cycle)

ต่างๆในการควบคุมหน่วยต่างๆหน่วยควบคุมจะมีขั้นตอนพื้นฐานอยู่จำนวนหนึ่ง เช่นขบวนการอ่านข้อมูลจากหน่วยความจำ ขบวนการเขียนข้อมูลส่งไปหน่วยความจำขบวนการI/Oและอื่นๆ ขบวนการย่อยๆนี้เรียกว่า แมชีนไซเคิล จะเห็นว่าไซเคิลการแฟทช์ เป็นการอ่านข้อมูลจากหน่วยความจำจึงต้องประกอบด้วยขบวนการพื้นฐานการอ่านข้อมูลหรือ Read Machine cyle เป็นต้น

หน่วย I/O

หน่วยสุดท้ายที่จะกล่าวถึงคือหน่วย I/O ส่วนนี้เป็นส่วนที่ควบคุมจะติดต่อกับโลกภายนอก หน่วย I/O เปรียบเสมือนตัวกลาง(Bufferหรือกันชน) เมื่อคอมพิวเตอร์แสดงผลลัพท์เพื่อบันทึกในรูปที่มนุษย์เข้าใจ หรือรับข้อมูลมาเก็บไว้ในหน่วยความจำ หน่วยควบคุมต้องส่งสัญญาณพร้อมข้อมูลมายังหน่วย I/O ควบคุมอุปกรณ์ภายนอกอีกต่อหนึ่ง อุปกรณ์ภายนอกที่ใช้แสดงผลเรียกว่า อุปกรณ์ I/O

สถาปัตยกรรมของ Z-80

คำว่าไมโครคอมพิวเตอร์เป็นคำที่มีความหมายถึงวงจรไมโครโปรเซสเซอร์ประกอบรวมกับหน่วยความจำและอุปกรณ์อินพุท-เอาท์พุทไมโครโปรเซสเซอร์จึงเสมือนหัวใจสำคัญของระบบไมโครคอมพิวเตอร์ ปัจจุบันมีไมโครโปรเซสเซอร์หลายเบอร์แต่ละเบอร์มีรายละเอียดและขีดความสามารถที่แตกต่างกันอยู่บ้างแต่โดยเนื้อแท้ของมัน เราจะพบสิ่งที่ใกล้เคียงกันในแง่ของโครงสร้างและสถาปัตยกรรม อย่างไรก็ตามการทำความเข้าใจกับไมโครโปรเซสเซอร์เบอร์ใดเบอร์หนึ่งให้ถ่องแท้ก็จะเป็นบันไดในการเข้าใจไมโครโปรเซสเซอร์เบอร์อื่นได้อย่างไม่ยากเย็นนัก

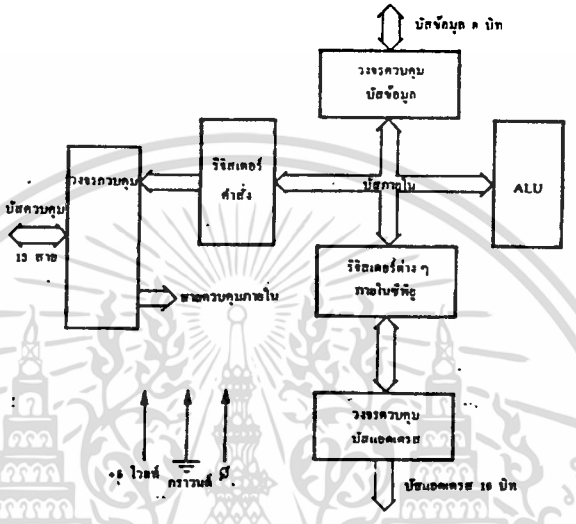
Z-80 ไมโครโปรเซสเซอร์มีวิวัฒนาการมาจากไมโครโปรเซสเซอร์เบอร์ 8080 กล่าวคือหลังจากที่ไมโครโปรเซสเซอร์เบอร์ 8080 ได้ผลิตขึ้นและออกจำหน่ายซึ่งสร้างความตื่นเต้นต่อผู้ใช้งานอย่างมากทำให้มีผู้ออกแบบระบบซอฟแวร์ไว้เป็นจำนวนมาก การแพร่หลายของไอซี 8080 แต่อย่างไรก็ตาม 8080 มีจุดอ่อนหลายประการทั้งทางด้านฮาร์แวร์ที่วิศวกรของบริษัทไซล็อกซึ่งเป็นกลุ่มบุคคลที่เข้าใจปัญหาต่างๆเหล่านี้เป็นอย่างดีเพราะเป็นที่วิศวกรที่แยกมาจากบริษัทอินเทล ได้ทำการออกแบบทั้งระบบซอฟท์แวร์และฮาร์แวร์ของไมโครโปรเซสเซอร์เบอร์ใหม่ซึ่งยังคงแทนระบบซอฟท์แวร์ของ 8080 ได้และได้ชื่อว่า Z-80

เหตุผลที่สำคัญประการหนึ่งของการนำเอา Z-80 มากล่าวถึงในโครงสร้างเพราะตัวโครงสร้างของ Z-80 เป็นโครงสร้างที่เข้าใจได้ง่ายทางฮาร์แวร์และการอินเตอร์เฟส ส่วนระบบทางซอฟแวร์นั้น Z-80 ก็มีขีดความสามารถในการทำงานได้สูงกว่า 8080 ของอินเทลอีกมาก

โครงสร้างของซีพียู

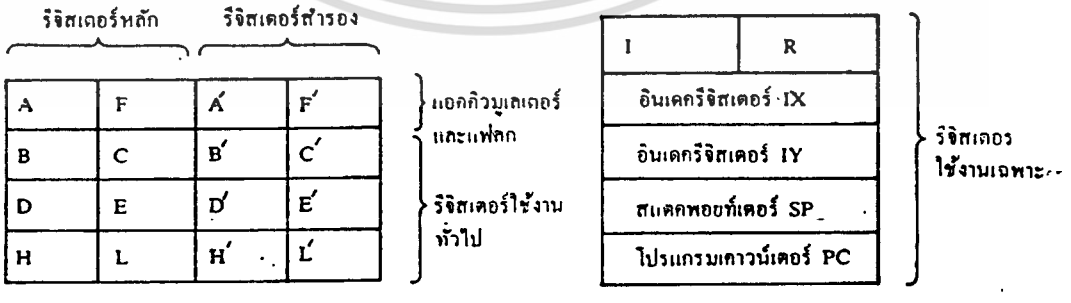
โครงสร้างของซีพียู Z-80 มีโครงสร้างที่พัฒนามาจาก 8080 ดังนั้นในแง่โครง

สร้างพื้นฐานจะเหมือนกับซีพียูของ 8080 แต่เนื่องจาก Z-80 มีการพัฒนามากขึ้นทางซอฟแวร์ จึงทำให้มีรายละเอียดแตกต่างเพิ่มเติมอีกหลายประการด้วยกัน บล็อกไดอะแกรมรูปที่ 2.7เป็นไดอะแกรมแสดงให้เป็นโครงสร้างของ Z-80 โดยโครงสร้างของยูนิซีจะบรรจุลงในแอลเอสไอขนาด 40 ขา



รูป 2.7บล็อกไดอะแกรมซีพียู Z-80

โครงสร้างภายในของ Z-80 ซีพียูประกอบด้วยรีจิสเตอร์ภายในที่สามารถเขียนและอ่านได้ถึง 208 บิท โดยแยกเป็นกลุ่มของรีจิสเตอร์ขนาด 8 บิท 18 รีจิสเตอร์และรีจิสเตอร์ขนาด 16 บิทอีก 4 รีจิสเตอร์โดยมีชุดรีจิสเตอร์แสดงไว้ดังรูป 2.8



รูป 2.8 แสดงรีจิสเตอร์ต่างๆที่มีอยู่ใน Z-80

รีจิสเตอร์หลักที่ใช้งานทั่วไป

รีจิสเตอร์ในกลุ่มแรกก็คือ A,F,B,C,D,E,H,L เป็นรีจิสเตอร์ขนาด 8 บิทที่ใช้งานโดยทั่วไปโดยรีจิสเตอร์เหล่านี้สามารถประกอบรวมกันเป็นคู่รีจิสเตอร์ได้คือ AF,BC,DE,HL โดยคู่รีจิสเตอร์เหล่านี้จะได้รับการใช้งานในลักษณะของรีจิสเตอร์ขนาด 16 บิท การกระทำภายในซีพียูอาจจะอาศัยเพียงรีจิสเตอร์เดียวหรือกระทำเป็นคู่รีจิสเตอร์ได้โดยที่ A คือแอดคิวมูเลเตอร์ คือแฟลกแฟลกของ Z-80 จะมีด้วยกันทั้งหมด 6 ตัวจึงใช้เพียง 6 บิท แต่ Z-80 อาศัยการเพิ่มบิทขึ้นอีก 2 บิท และกลายเป็นรีจิสเตอร์ F รีจิสเตอร์ F นี่สามารถได้รับการเซทรีเซทการกระทำตามคำสั่งทางคณิตศาสตร์หรือลอจิดได้และเราสามารถใช้ F เหมือนรีจิสเตอร์ซึ่งเมื่อรวมกันกับ A แล้วจะกลายเป็นรีจิสเตอร์ ขนาด 16 บิทได้

กลุ่มรีจิสเตอร์สำรอง

เป็นรีจิสเตอร์ที่สามารถเก็บข้อมูลได้โดยเป็นตัวเก็บข้อมูลที่มาจากรีจิสเตอร์หลัก รีจิสเตอร์ชุดนี้จึงมีด้วยกัน 8 ตัว คือ A',F',B,C',D',E',H',L', รีจิสเตอร์เหล่านี้เป็นรีจิสเตอร์ที่ใช้กันในการเก็บข้อมูลชั่วคราว ในการที่ต้องการใช้รีจิสเตอร์หลักใช้งานอย่างอื่นก่อน ดังนั้นรีจิสเตอร์กลุ่มนี้จึงไม่สามารถกระทำทางคณิตศาสตร์หรือลอจิก

กลุ่มรีจิสเตอร์ที่ใช้งานเฉพาะอย่าง

- โปรแกรมเคาร์เตอร์(PC-Program counter) โปรแกรมเคาร์เตอร์เป็นรีจิสเตอร์ขนาด 16 บิท ที่เป็นตัวกำหนดตำแหน่งของโปรแกรมในขณะสภาวะการกระทำการเฟทช์ โดยขณะทำการเฟทช์ค่าที่อยู่ในโปรแกรมเคาว์เตอร์จะไปปรากฏอยู่ที่แอดเดรสบัสเมื่อชี้ไปยังตำแหน่งในหน่วยความจำให้ CPU อ่านคำสั่งมาตีความหมาย ค่าที่อยู่ในโปรแกรมเคาว์เตอร์จะเพิ่มค่าขึ้นได้อย่างอัตโนมัติหลังการกระทำการเฟทช์แต่ถ้าหากซีพียูกระทำคำสั่งใช้ข้ามไปยังตำแหน่งอื่น(JUMP) ค่าแอดเดรสที่จะกระโดดข้ามนั้นจะโหลดเข้ามายังโปรแกรมเคาว์เตอร์ได้อย่างอัตโนมัติ

- สแตคพอยท์เตอร์(SP-stack pointer) เป็นรีจิสเตอร์ที่มีขนาด 16 บิท ที่ใช้สำหรับชี้ไปยังแอดเดรสชั้นบนสุดของสแตคที่อยู่ใน RUM โดยส่วนของสแตคมีลักษณะโครงสร้างเป็นหน่วยความจำเป็นแบบเก็บทีหลังเรียกออกได้ก่อน ข้อมูลในสแตคอาจได้รับการพุทหรือพอยออกมาจากข้อมูลรีจิสเตอร์ภายในยูพีซี ลักษณะของสแตคในที่นี้ยังเป็นส่วนช่วยในการกระทำอินเตอร์รัพท์และการเรียกโปรแกรมย่อย กล่าวคือในการอินเตอร์รัพท์ค่าของโปรแกรมเคาว์เตอร์จะได้รับการเก็บรักษาไว้ในชั้นสแตคครั้นเมื่อโปรแกรมกลับจากอินเตอร์รัพท์ไปกระทำยังโปรแกรมหลักก็นำค่าจากสแตคกลับเข้ามายังโปรแกรมเคาว์เตอร์ใหม่ในทำนองเดียวกันการกระโดดไปยังโปรแกรมย่อย ก็เช่นเดียว

กันดังนั้นการกระทำในรูปของอินเตอร์รัพท์หรือโปรแกรมย่อยสามารถซ้อนกันได้ไม่มีที่สิ้นสุด

-อินเดรสรีจิสเตอร์(IX,IY-index registor)ซีพียู Z-20 มีอินเดดรีจิสเตอร์ขนาด 16 บิท 2 ตัวแต่ละตัวใช้ประโยชน์หลักในการทำหน้าที่เป็นตัวเก็บแอดเดรสฐาน(Base address) เพื่อทำหน้าที่อ้างแอดเดรสแบบอินเดดแอดเดรสซึ่ง(index addressing)ในหมวดของอินเดรส ให้กับคำสั่งข้อมูลที่ติดมากับคำสั่งที่เรียกว่าดิสเพลซเมนท์(displacement)ซึ่งจะเก็บในรูปตัวเลข 2's คอมพลีเมนท์

-อินเตอร์รัพท์เพจแอดเดรสรีจิสเตอร์(I-Interrupt pape address register) การอินเตอร์รัพท์ของ Z-80 มีหลายหมวด และหมวดหนึ่งที่ทำให้การอินเตอร์รัพท์ของ Z-80 มีประสิทธิภาพสูงกล่าวคือเมื่อเกิดการอินเตอร์รัพท์ในหมวดนี้ขึ้นมันสามารถอ้างแอดเดรสโดยทางอ้อมไปกระทำโปรแกรมในที่ใดก็ได้หน่วยความจำ โดยอาศัยค่าในรีเจสเตอร์ I รวมกับค่าที่ส่งมาจากอุปกรณ์เพอร์เฟอรัลอีก 8 บิท ชี้ไปยังค่าในรีเจสเตอร์ ในหน่วยความจำเพื่อนำค่านั้นมาโหลดเข้าในโปรแกรมเคาว์เตอร์เพื่อกระทำต่อไป ด้วยวิธีการนี้เราจึงสามารถกระโดดเข้าไปทำที่ส่วนใดก็ได้ในหน่วยความจำ

-รีจิสเตอร์รีเฟรชหน่วยความจำ(R-memory reflesh register)การต่อซีพียูกับหน่วยความจำนั้นโดยปรกติจะต่อกับหน่วยความจำชนิดสแตติดได้โดยง่ายแต่อย่างไรก็ดี ชนิดไดนามิคที่ต้องการรีเฟรชมีราคาถูกกว่ามีความหนาแน่นสูงกว่า Z-80 ให้ข้อดีกว่าประการหนึ่งคือมันสามารถให้การรีเฟรชหน่วยความจำได้อย่างอัตโนมัติ โดยค่าในรีเจสเตอร์จะเพิ่มค่าขึ้นอีก 1 ทุกครั้งที่มีการกระทำการเฟทช์คำสั่งและข้อมูลในรีจิสเตอร์ R นี้จะส่งไปยังแอดเดรสยังในส่วนบิทที่มีนัยสำคัญต่ำกว่าจังหวะของการสิ่งนี้จะเป็นจังหวะเดียวกันกับซีพียูส่งสัญญาณรีเฟรชออกมา ผู้โปแกรมสามารถกำหนดค่าให้กับรีเจสเตอร์ R นี้ได้แต่ค่าในรีจิสเตอร์นี้จะเรียกใช้โดยผู้โปรแกรมทางคำสั่งโดยตรงไม่ได้

-แอดคิวมูเลเตอร์(accumulator) และแฟลก(flag)ซีพียูจะมีรีจิสเตอร์ที่ใช้เป็นหลักในการเป็นตัวโอเปอร์แรนด์สำหรับกระทำทางคณิศาสตร์และลอจิก โดยรีจิสเตอร์หลักนี้จะมีเพียง 8 บิทเรียกว่าแอดคิวมูเลเตอร์(Accumulator)การกระทำในส่วนของของหน่วยคณิตศาสตร์และลอจิกย่อมเกิดเงื่อนไขได้หลายอย่างที่จะต้องแสดงสถานะภาพของเงื่อนไขเหล่านั้นเช่นเงื่อนไขผลลัพท์เป็นศูนย์ผลลัพท์เป็นบวกหรือลบ มีตัวทดหรือขอยืมในการกระทำทางคณิตศาสตร์แสดงเงื่อนไขพาร์ตี้คู่หรือคี่ ฯลฯ สิ่งเหล่านี้จะให้ผลลัพท์แสดงสถานะได้ด้วยแฟลก(flag)แฟลกเป็นรีจิสเตอร์ขนาด 8 บิทซึ่งสามารถรวมกับแอคคิวมูเลเตอร์เป็นรีจิสเตอร์ขนาด 16 บิทได้ผู้โปรแกรมยังสามารถ

ใช้คำสั่งในการเคลื่อนย้ายข้อมูลจากคิวมูเลเตอร์(A)และแฟลก(F)ไปเก็บไว้ใน A'และ F'ได้เพื่อให้การใช้งานของ A และ F มีประสิทธิภาพดียิ่งขึ้น

-หน่วยคำนวณทางคณิตศาสตร์และลิจิก(ALU-Arithmetic and logic unit) การประมวลผลที่สำคัญของซีพียูของคอมพิวเตอร์ยิ่งขึ้นอยู่กับหน่วยคำนวณทางคณิตศาสตร์และลอจิก (ALU) ส่วน ALU นี้จะนำข้อมูลซึ่งอาจจะนำมาจากนอกซีพียูหรือภายในซีพียูก็ได้มาประมวลผล การประมวลผลในส่วน ALU ที่สำคัญจะประกอบด้วย

-การบวก (ADD)

-การลบ (SUB STRAXT)

-ลอจิก (AND)

-ลอจิก (OR)

-ลอจิก (Ex-OR)

-เปรียบเทียบ (Compare)

-การเลื่อนไปทางซ้ายหรือขวา (Shift)

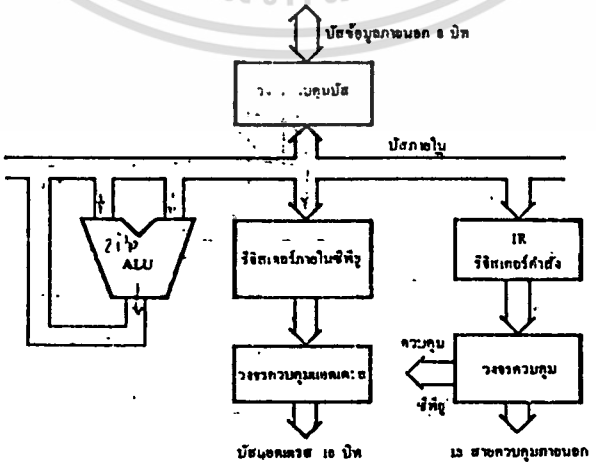
-การเพิ่มค่า (Increment)

-การลดค่า (Decrement)

-การเซทบิท (Set bit)

-การรีเซทบิท (Reset bit)

-การทดสอบบิท (Test bit)

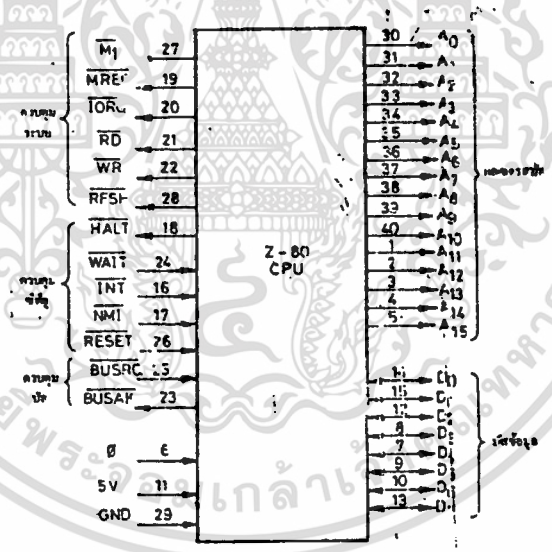


รูป 2.9 แสดงการทำงาน ALU ภายใน

รีจิสเตอร์คำสั่งและส่วนควบคุม(Intruction register and control)ในการกระทำการเฟทช์ซีพียูจะอ่านคำสั่งจากหน่วยความจำที่เป็นส่วนของโปรแกรมโดยรอคำสั่งนั้นมาเก็บได้ใน IR เพื่อทำการถอดระหัสคำสั่งและส่งสัญญาณควบคุมการทำงานภายในซีพียูหรือควบคุมการทำงานของระบบสัญญาณควบคุมเหล่านี้ออกมาในจังหวะต่างๆกันเพื่อใช้ควบคุมระบบในการทำงานต่อไป

การจัดขาของ z-80

Z-80 ซีพียูเป็นไอซีไมโครโปรเซสเซอร์ที่มีขาเพียง 40 ขาโดยหลักการแล้ว Z-80 เป็นซีพียูได้โดยสมบูรณ์กล่าวคือ Z-80 ไม่ต้องประกอบกับอุปกรณ์อื่นที่แยกการทำงานเพื่อรวมเป็นซีพียู ส่วนของสัญญาณจะประกอบด้วยบัสแอดเดรสบัสข้อมูลและสัญญาณควบคุมการจัดขาแสดงในรูปที่ 2.10



รูปที่ 2.10 ลักษณะของขาไอซี Z-80 ซีพียู

รายละเอียดของขาต่างๆแสดงได้ดังต่อไปนี้

- A0-A15 บัสแอดเดรสสัญญาณที่ออกมาจากไอซีเหล่านี้จะให้แอดดีฟชนะ HIGH โดยขาเหล่านี้เป็นเอาว์พุทแบบไตรสเตรท บัสแอดเดรสมีด้วยกันทั้งหมด 16 สายเพื่อให้ซีพียูติดต่อกับหน่วยความจำได้ถึง 2^{16}=64k ไบท์นอกจากนี้ส่วนของแอดเดรสยังเป็นตัวกำหนดเบอร์พอร์จของอุปกรณ์อินพุท-เอาว์พุทโดยขณะที่ซีพียูกระทำคำสั่งเกี่ยวกับอินพุทหรือเอาว์พุทค่าของแอดเดรสบัสในบิทล่าง(A0-A7)จะแสดงค่าเบอร์พอร์ทดังนั้นเราจึงมีอุปกรณ์อินพุทหรือเอาว์พุทได้ทั้งหมด

2 =256พอร์และในขณะช่วงเวลารีเฟท โดยเมื่อสัญญาณรีเฟทปรากฏขึ้นที่ตรีเฟรช(RFSH)ค่าในแอดเดรสบัส A0-A7จะแสดงค่าแอดเดรสของหน่วยความจำที่จะได้รับการกระทำการีเฟทช

- D0-D7บัสข้อมูล(Data bus)เป็นลักษณะบัสแบบสองทิศทาง Z-80 ซีพียูมีบัสข้อมูล 8 เส้นบัสข้อมูลเป็นเส้นทางผ่านของข้อมูลระหว่างซีพียูกับหน่วยความจำ ซีพียูกับอุปกรณ์อินพุทหรือการติดต่อระหว่างอุปกรณ์อินพุท-เอาว์พุทกับหน่วยความจำ

- $\overline{M1}$(machine cycle one)มีลักษณะเป็นแอดตีฟที่ลอจิก "0" เป็นส่วนที่จะบอกให้ทราบว่าขณะนี้ซีพียูกำลังอยู่ในสภาวะเฟทช์ในขณะที่ซีพียูเฟทช์คำสั่งที่มีออฟโค้ดสองไบท์ส่วนของ M1 จะสร้างขึ้นขณะเฟทช์ในแต่ละไบท์ลักษณะของคำสั่งที่มีออฟโค้ดสองไบท์จะขึ้นต้นด้วยCBH, DDH, EDH, FDH นอกจากนี้ M1ยังสร้างสัญญาณร่วมกับ IORQ เพื่อบอกสถานะการตอบรับการอินเตอร์รัพท์

- $\overline{MREQ}$(memory request)เป็นลักษณะไตรสเตทในลอจิกแอดตีฟที่ "0"เป็นสัญญาณที่บอกให้ทราบซีพียูต้องการเขียนหรืออ่านหน่วยความจำตามแอดเดรสที่ปรากฏอยู่ในแอดเดรสบัส

- $\overline{IORQ}$(inpnt output request)เป็นเอาว์พุทลักษณะ ไตรสเตทให้ลอจิกแอดตีฟที่"0" เป็นสายสัญญาณที่บอกให้ทราบว่า ซีพียูต้องการติดต่อกับอุปกรณ์อินพุท-เอาว์พุทโดยแอดเดรสบัส 8 บิทล่างจะให้แสดงค่าเบอร์พอร์ทส่วนบัสข้อมูลจะมีการส่งถ่ายระหว่างซีพียูกับ I/O นอกจากนี้ ถ้าเกิดขึ้นพร้อมสัญญาณ M1เป็นตัวบอกถึงสถานะที่ซีพียูกำลังตอบสนองผลการอินเตอร์รัพท์ โดยขณะนี้ส่วนของบัสข้อมูลจะมีการส่งผ่านเข้ามาด้วยค่าของอินเตอร์รัพท์เวคเตอร์

- $\overline{RD}$(memory read)เป็นเอาว์พุทที่ไตรสเตรทและแอคตีฟขณะลอจิก"0"เป็นตัวบอกว่าขณะนี้ซีพียูต้องการผ่านข้อมูลจากหน่วยความจำหรืออุปกรณ์

- $\overline{WR}$(memory write)เป็นขาเอาว์พุทแบบไตรสเตทและแอดตีฟขณะลิอจิก"0"เป็นสัญญาณบอกว่าซีพียูต้องการเขียนข้อมูลโดยจะเขียนข้อมูลในตำแหน่งที่ address bus กำหนดขึ้นอาจจะเป็นหน่วยความจำหรืออุปกรณ์ I/O ก็ได้

- $\overline{RESH}$(refresh)เป็นขาเอาว์พุทแอคตีฟขณะลิจิก"0" RFSH เป็นสัญญาณบอกให้ทราบว่าสัญญาณในแอดเดรสบัสในส่วน A0-A6 เป็นแอดเดรสที่ใช้ในการรีเฟรชหน่วยความจำแบบไดนามิกส่วนบิท A7 จะเป็น"0" ส่วน A15-A8 จะแสดงค่าของรีจิสเตอร์ I

- $\overline{HALT}$(halt state)เป็นเอาว์พุทที่แอดตีฟด้วยลิจิก"0" สัญญาณจะแสดงเมื่อซีพียูได้กระทำคำสั่ง HALT และจะหยุดรอจนกว่าจะมีการอินเตอร์รัพท์หรือรีเซทขณะที่อยู่ในช่วง HALT ซีพียูจะเหมือนกระทำคำสั่ง NOP(No operation)เพื่อให้เกิดไซเคิลในการทำงานเพื่อสัญญาณไปกระทำการีเฟรทหน่วยความจำชนิด ไดนามิกส์

- $\overline{WAIT}$(wait) เป็นขาอินพุทจะแอคตีฟด้วยลอจิก"0" WAIT เป็นตัวกำหนดแสดงเพื่อบอกซีพียูให้ซีพียูหยุดรอในกรณีที่อุปกรณ์อินพุท-เอาว์พุทหรือหน่วยความจำไม่สามารถจำไม่สามารถรับหรือส่งข้อมูลได้ทัน. WAIT จะเป็นตัวทำให้ซีพียู ซิงค์ได้พอดีกับอุปกรณ์อินพุท-เอาว์พุทที่ทำงานด้วยความเร็วช้า

- $\overline{INT}$(interrupt request) เป็นขาอินพุทแอคตีฟด้วยลอจิก"0" INT เป็นสัญญาณที่สร้างขึ้นมาจากอุปกรณ์อินพุท-เอาว์พุท เพื่อต้องการที่จะอินเตอร์รัพท์ซีพียูจะทำการตรวจสอบสัญญาณนี้ทุกๆครั้งที่จบการกระทำแต่ละคำสั่งการตอบสนองของการอินเตอร์รัพท์สามารถควบคุมได้โดยซอฟแวร์ด้วยการเซทค่าอินเตอร์รัพท์ฟลิพฟลอบ(IFF) การตอบอินเตอร์รัพท์ซีพียูจะสร้างสัญญาณตอบด้วยการสร้างสัญญาณ IORQ ระหว่างช่วงเวลา M1 การตอบสนองต่อการอินเตอร์รัพท์มีแยกแยะให้ 3 แบบซึ่งจะอธิบายในรายละเอียดต่อไป

- $\overline{NMI}$(nonmaskable interrupt) เป็นขาอินพุทที่จะบอกซีพียูในขณะขอบพัลซ์ขาลงการอินเตอร์รัพท์ด้วยวิธีนี้ ซีพียูจะให้คำสั่งสูงกว่า INT กล่าวคือบัสมันจะตอบสนองและกระทำทันทีด้วยการเริ่มเอ็กซ์คิว คำสั่งในตำแหน่ง 0066H โดยอัตโนมัติการกระโดดไปกระทำในกรณีนี้ซีพียูจะเก็บค่าโปรแกรมเคาว์เตอร์เดิมไว้ในสแตคเพื่อจะได้กลับไปทำงานเดิมเมื่อเสร็จสิ้นการอินเตอร์รัพท์ได้

- $\overline{RESET}$(reset) เป็นขาอินพุทที่แอคตีฟด้วยลอจิก"0" การรีเซทในกรณีนี้จะมีผลดังนี้

  1. ค่าของ PC มีค่าเป็น "0"
  2. IFF จะได้รับการ DISABLE
  3. รีจิสเตอร์ I จะมีค่า 00H
  4. รีจิสเตอร์ R จะมีค่า 00H
  5. จะมีการเซทอินเตอร์รัพท์โหมดมาอยู่ที่โหมดที่ 0

ระหว่างการรีเซทสายแอดเดรสบัสและข้อมูลจะได้รับการกระทำให้มีค่าอินพีแดนซ์สูงเพื่อแยกออกจากซีพียู ส่วนสายสัญญาณควบคุมจะได้รับการทำให้เป็นสัญญาณที่ไม่แอคตีฟรีเฟรชจะไม่เกิดขึ้น

- $\overline{BUSRQ}$(bus request) เป็นขาอินพุทที่แอคตีฟด้วยลอจิก"0"BUSRQ' เป็นสัญญาณที่ส่งบอกกับซีพียูเพื่อต้องการให้ซีพียูควบคุมบัส กล่าวคือต้องการให้ซีพียูทำให้บัสแอดเดรสและข้อมูลอยู่ในสถานะอินพีแดนซ์สูงคือต้องการแยกซีพียูออกจากบัสนั่นเอง

- $\overline{BUSAK}$(bus acknowledge) เป็นขาเอาว์พุทแอคตีฟขณะลิจิก"0"BUSAK เป็นสัญญาณ

ตอบจากซีพียูได้แยกตัวเองออกจากแอดเดรสบัสและบัสข้อมูลเรียบร้อยแล้ว

- 0(clock)สัญญาณนาฬิกาที่จะป้อนเข้าระบบ

คำสั่งของ Z-80

ในการพัฒนา Z-80 มีจุดมุ่งหมายหลักเพื่อใช้ชุดคำสั่งเดิมของ 8080 ได้ ดังนั้นทุกคำสั่งที่เคยใช้ได้ในไมโครโปรเซสเซอร์เบอร์ 8080 จะยังคงใช้ได้ในกรณีที่ใช้งานกับซีพียู Z-80 Z-80 มีคำสั่งได้มากถึง 158 คำสั่ง ซึ่งรวมคำสั่งของ 8080 ทั้ง 78 คำสั่งไว้ด้วย กลุ่มคำสั่งของ Z-80 พอแบ่งแยกได้เป็นกลุ่มดังนี้

1.กลุ่มการโหลดและแลกเปลี่ยนข้อมูล(Load and exchange)

2.กลุ่มการย้ายและค้นหาข้อมูลเป็นบล็อก(block transfer and search)

3.กลุ่มคำสั่งทางคณิตศาสตร์และลอจิก(arithematic and logic)

4.กลุ่มคำสั่งการเคลื่อนย้ายข้อมูลเป็นวงรอบและการชิฟท์(rotate and shift)

5.กลุ่มคำสั่งการกระทำในส่วนบิท(bit manipulation set reset test)

6.กลุ่มคำสั่งการกระโดด การเรียกโปรแกรมย่อย การคืนสู่กลับโปรแกรมหลัก (jump,call and return)

7.กลุ่มคำสั่งเกี่ยงกับอินพุท-เอาว์พุท(input-output)

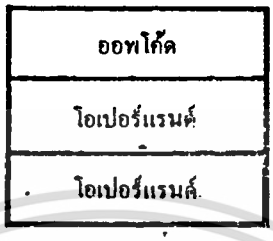
8.กลุ่มคำสั่งควบคุมซีพียู(basic CPU cont)

การสร้างแอดเดรสของ Z-80

ภายในซีพียูของ Z-80 มีรีจิสเตอร์หลายตัวการทำงานของ Z-80 จึงคล่องตัวได้มาก การที่จะให้ Z-80 เป็นระบบที่สมบูรณ์ต้องต่อกับหน่วยความจำและอุปกรณ์อินพุทเอาว์พุทในการต่อร่วมเพื่อให้ซีพียูทำงานได้สมบูรณ์ ในกรณีนี้ซีพียูจำเป็นจะต้องมีการเรียกแอดเดรสที่ต้องการอ้างแอดเดรส(addessing)ของซีพียูมีโหมด(mode)การอ้างได้หลายวิธี ในแต่ละวิธีการอ้างแอดเดรสได้เป็นกลุ่มๆ คือ

1.การอ้างแอดเดรสแบบอิมมีเดียท(Immediate addressing)ในโหมดนี้คำสั่งจะประกอบด้วยออปโค้ด 1-2ไบท์ และตามด้วยโอเปอร์แรนด์อีก 1 ไบท์ ลักษณะของคำสั่งจะเป็นดังนี้

2.การขยายตัวโอเปอร์แรนด์ในกลุ่มคำสั่งการอ้างแอดเดรสแบบอิมมีเดียท(Immediate extended)การอ้างแอดเดรสแบบนี้ก็เหมือนกับแบบแรก แตกต่างกันเพียงส่วนของโอเปอร์แรนด์เป็นข้อมูลขนาด 16 บิท ลักษณะรูปแบบของคำสั่งจะเป็นดังนี้

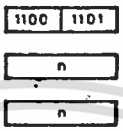


3.การอ้างแอดเดรสแบบ modified page Zero ในคำสั่ง Z-80 มีคำสั่งพิเศษอยู่คำสั่งหนึ่งที่เหมือนคำสั่ง call คือคำสั่ง RET หรือ Restart ลักษณะการทำงานในการอ้างแอดเดรสคือมันสามารถกำหนดค่าให้กับโปรแกรมเคาว์เตอร์ได้โดยตรง ค่าที่มันกำหนดให้จะเป็นแอดเดรสที่อยู่ในเพจศูนย์ ลักษณะของคำสั่งนี้ประกอบขึ้นเพียงไบท์เท่านั้น จึงมีประโยชน์หลายอย่างเช่นใช้เป็นมาร์ค(mark)สำหรับคำสั่งให้กระทำในขณะที่มีการอินเตอร์รัพท์ ข้อมูลเดิมในโปรแกรมเคาว์เตอร์ก่อนการเปลี่ยนแปลงจะได้รับการเก็บรักษาไว้ได้อีกด้วย แต่เงื่อนไขในการกระโดดไปยังเพจศูนย์ยังมีขอบเขตจำกัด คือมันจะกระโดดไปได้เพียง 8 แอดเดรสเท่านั้น

4.การอ้างแอดเดรสแบบเปรียบเทียบ(relative addressing)วิธีการนี้จะใช้ข้อมูลไบท์ที่อยู่ตามหลังออปโค้ด เพื่อบอกตำแหน่งว่าแอดเดรสที่อ้างถึงอยู่ห่างจากค่าในโปรแกรมเคาว์เตอร์เท่าใดเช่นเมื่อซีพียูกระทำคำสั่งนี้เสร็จ ค่าในโปรแกรมเคาว์เตอร์จะมีค่าเป็น PC+2+e นั่นคือ ค่าใหม่ที่มันจะอ้างถึงอยู่ห่างจากคำสั่งที่มันกระทำแล้วด้วยค่า e นั่นเองค่า e ที่ซีพียูมองเห็นจะเป็นลักษณะของตัวเลข 2'S คอมพลีเมนท์ ดังนั้นค่าที่มันอ้างถึงได้จึงอยู่ระหว่าง + 127 กับ - 128

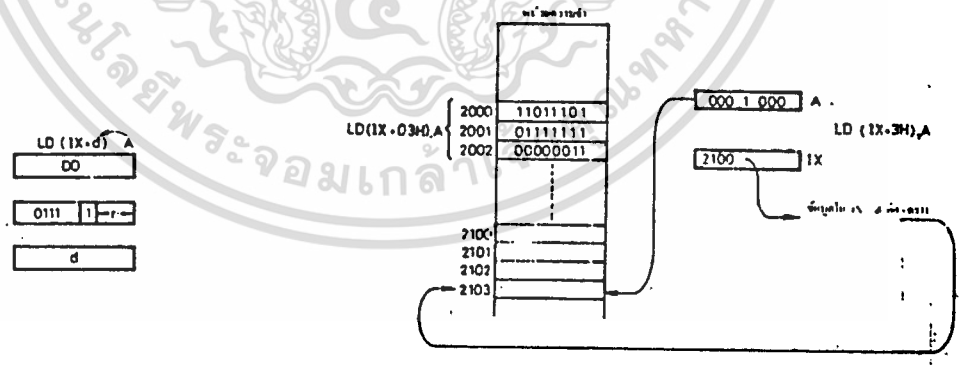
5.การอ้างแอดเดรสแบบ extended addressing วิธีนี้ใช้ข้อมูล 2 ไบท์ที่ตามออปโค้ดเป็นตัวกำหนดค่าแอดเดรสเช่นข้อมูลคือ NN ซึ่งเป็นตัวกำหนดค่าของแอดเดรสที่จะกระทำใหม่ยกตัวอย่างหนึ่งคำสั่งคือ CALL NN หมายถึงการเรียกโปรแกรมย่อยที่ตำแหน่ง NN

CALL nn



ซึ่งเป็นการนำเอาข้อมูลในตัวโอเปอร์แรนด์เป็นแอดเดรสนั่นเอง โดยการนำเอาข้อมูลส่วนโอเปอร์แรนด์ไปไว้ยังโปรแกรมเคาว์เตอร์โดยตรง ส่วนข้อมูลเดิมโปรแกรมเคาว์เตอร์จะเก็บไว้ในส่วนของสแตค

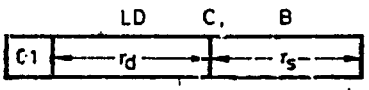
6.การอ้างแอดเดรสโดยใช้อินเดครีจิสเตอร์(index addressing)ใน Z-80 จะมี IX และ IY เป็นอินเดครีจิสเตอร์ วิธีการใช้ร่วมในการอ้างแอดเดรสนั้นจะใช้ค่า IX หรือ IY เป็นฐานเพื่อร่วมกับค่าที่ตามหลังออปโค้ดมารวมกันเป็นแอดเดรสที่ต้องการ



ลักษณะของคำสั่งนี้เป็นการนำเอาข้อมูลของรีจิสเตอร์ A ไปเก็บไว้ในหน่วยความจำที่อ้างแอดเดรสโดยที่ค่า IX รวมกับค่า d โดยปรกติค่า d จะได้รับการรวมโดยดูในรูป 2'S คอมพลีเมนท์ ดังนั้นจึงทำให้ค่า d แปรได้จากค่า(127 ถึง -128)

7.การอ้างแอดเดรสแบบใช้รีจิสเตอร์(register addressing)การออกแบบกลุ่มคำสั่งในขอบเขตของจำนวนบิทออปโค้ดที่จำกัด ต้องหาวิธีที่ให้ได้ประสิทธิภาพดีที่สุด ในกรณีของ

Z-80 เป็นการกำหนดรหัสของรีจิสเตอร์ เช่นรีจิสเตอร์ 8 ตัว ก็ใช้รหัส 3 บิท ดังนั้นกลุ่มคำสั่งโหลดสามารถใช้คำสั่งเพียง 1 ไบท์ เพื่อการโหลดข้อมูลระหว่างรีจิสเตอร์ได้ เช่น



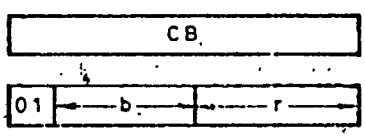
โดยที่สามบิทใน rd คือชื่อรีจิสเตอร์ปลายทาง ส่วนสามบิทใน rs คือรีจิสเตอร์ต้นทาง

8.การอ้างแอดเดรสแบบ implied addressing ในกรณีนี้ซีพียูจะตีความหมายเอง โดยตรงว่า รีจิสเตอร์ภายในซีพียูตัวหนึ่งจะเป็นตัวโอเปอร์แรนด์ เช่น คำสั่ง

จากคำสั่งนี้จะเห็นว่าเราไม่ต้องกำหนดรหัสรีจิสเตอร์ A เลย แต่เครื่องจะรู้เอง ว่าเป็นคำสั่งที่ต้องการกระทำร่วมกันรีจิสเตอร์ A กล่าวคือ การทำงานซีพียูจะนำเอารีจิสเตอร์ที่กำหนดในออปโค้ดมากระทำร่วมกับรีจิสเตอร์ A นั่นเอง

9.การอ้างแอดเดรสเข้าสู่บิทต่างๆ(bit addressing)คำสั่งพิเศษใน Z-80 ที่สามารถกระทำการเซทและรีเซท หรือการทดลองบิทใดบิทหนึ่งใน 8 บิท เช่น

RES b,r



ในที่นี้เราใช้รหัส b แทนโอเปอร์แรนด์ว่า เป็นบิทใดในรีจิสเตอร์ ถ้า b=000 หมายถึงบิท 0 และ b=111 หมายถึงบิท 7 ดังนั้นเราสามารถทำให้บิทใดในรีจิสเตอร์ให้เป็น 0 ได้ เช่น RES 0,A หมายถึงทำให้บิท 0 ของรีจิสเตอร์ A เป็น 0

แฟลก

ใน Z-80 ซีพียู จะประกอบด้วยแฟลกจำนวน 6 แฟลก และมีบิทที่ไม่ได้แสดงเป็นแฟลกอีก 2 บิท รวมเป็น 8 บิท เพื่อประกอบเป็นรีจิสเตอร์ F ส่วนของแฟลกแต่ละบิทสามารถที่จะเซทหรือรีเซทตามการกระทำของคำสั่งที่ซีพียูกำลังทำงาน นอกจากนี้ซีพียูยังสามารถใช้ตรวจสอบแฟลกเพื่อกระทำเงื่อนไขต่างๆ

ลักษณะการใช้แฟลกจะใช้ตัวอักษรย่อแทนแฟลก ดังนี้

C = แฟลกตัวทด

N = แฟลกแสดงการบวกหรือลบ

P/V = แฟลกแสดงพาริตี้และโอเวอร์โฟล์ว

H = แฟลกแสดงตัวทดช่วย

Z = แฟลกแสดงค่าศูนย์

S = แฟลกเครื่องหมาย

X = ไม่ได้ใช้

ในการกระทำคำสั่งต่างๆของ Z-80 บางคำสั่งก็จะมีผลต่อแฟลก บางคำสั่งก็ไม่มีผลต่อแฟลก การที่คำสั่งบางคำสั่งมีผลต่อแฟลกทำให้เราสามารถตรวจสอบเงื่อนไขได้จากแฟลก เช่น

INC A (ให้เพิ่มค่ารีจิสเตอร์ A อีก 1)

ผลที่เกี่ยวกับแฟลก คือ

ถ้าผลลัพท์เป็น 0 Z=1

ถ้าผลลัพท์เป็นลบหรือบิท D =1 S=1

ถ้ามีตัวทดในบิทที่ 3 H=1

ถ้าผลลัพท์เท่ากับ 80 P/V=1

คำสั่งตัวอย่างนี้ไม่เกี่ยวกับ N และ C แฟลก

ลักษณะของรีจิสเตอร์ F จะประกอบด้วยแฟลกแต่ละบิทเป็นดังนี้

S	Z	X	H	X	P/V	N	C
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

ในบิท b5 และ b3 จะเป็นบิทที่ประกอบเพิ่มขึ้นมาโดยมิได้มีความหมายในทางแฟลกแต่จะประกอบเพื่อให้รีจิสเตอร์ F ครบ 8 บิท การทำงานของซีพียูจึงสามารถโหลดข้อมูลจาก F ไปยัง A หรือโหลดจาก A กลับมายัง F ได้

กลุ่มคำสั่ง

กลุ่มคำสั่งเกี่ยวกับการโหลดข้อมูลและแลกเปลี่ยนข้อมูล

คำสั่งโหลดเป็นคำสั่งที่ต้องการให้ข้อมูลเคลื่อนย้ายจากสถานที่หนึ่ง (Source) ไปยังอีกสถานที่หนึ่ง(Destination)ซึ่งซือชและดิสเท็นนิชั่นจะอยู่ที่รีจิสเตอร์ภายในซีพียูหรือที่หน่วยความจำนั่นเอง ในตัวซีพียูจะมีรีจิสเตอร์หลายตัวจำเป็นที่จะต้องชี้แสดงว่าที่แอดเดรสใด คำสั่งโหลดแบ่งออกได้เป็นสองชนิดคือ 1 ไบท์ กับ 2 ไบท์ ตัวอย่าง ถ้าต้องการโหลดข้อมูลของรีจิสเตอร์ H ไปที่รีจิสเตอร์ B ภาษาเอ็สเซ็มเบอร์เขียนได้ดังนี้

ดิสเท็นนิชั่นเขียนไว้หลัง LD ต่อมาก็เขียนซือช ออปโค้ดก็คือ 44H เมื่อซีพียูทำงานข้อมูลภายในรีจิสเตอร์ H ก็จะถูกส่งไปไว้ที่รีจิสเตอร์ B

คำสั่งอื่นๆนอกเหนือจากนี้ ก็มีการสลับเปลี่ยนข้อมูล

กลุ่มคำสั่งในการโหลด 8 บิท

สัญลักษณ์	การกระทำ	แฟลก S	Z		H		P/V		N	C	ออปโค้ด 76 543 210	Hex	ขนาดของโปรแกรม	เวลาในการโหลด	No. of T States	Comments
LD r, s	r ← s	•	•	X	•	X	•	•	•	01 r s		1	1	4	r, s Reg.	
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110 ← n →		2	2	7	000 B 001 C	
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010 D	
LD r, (IX+d)	r ← (IX+d)	•	•	X	•	X	•	•	•	11 011 101 01 r 110 ← d →	DD	3	5	19	011 E 100 H 101 L	
LD r, (IY+d)	r ← (IY+d)	•	•	X	•	X	•	•	•	11 111 101 01 r 110 ← d →	FD	3	5	19	111 A	
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7		
LD (IX+d), r	(IX+d) ← r	•	•	X	•	X	•	•	•	11 011 101 01 110 r ← d →	DD	3	5	19		
LD (IY+d), r	(IY+d) ← r	•	•	X	•	X	•	•	•	11 111 101 01 110 r ← d →	FD	3	5	19		
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110 ← n →	36	2	3	10		
LD (IX+d), n	(IX+d) ← n	•	•	X	•	X	•	•	•	11 011 101 00 110 110 ← d → ← n →	DD 36	4	5	19		
LD (IY+d), n	(IY+d) ← n	•	•	X	•	X	•	•	•	11 111 101 00 110 110 ← d → ← n →	FD 36	4	5	19		
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7		
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7		
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010 ← n → ← n →	3A	3	4	13		
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7		
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7		
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010 ← n → ← n →	32	3	4	13		
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101 01 010 111	ED 57	2	2	9		
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101 01 011 111	ED 5F	2	2	9		
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101 01 000 111	ED 47	2	2	9		
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101 01 001 111	ED 4F	2	2	9		

สังเกต r, s หมายถึงรีจิสเตอร์ A, B, C, D, E, H, I

IFF หมายถึงข้อมูลในฟลิปฟลอยแสดงสถานะการอินเตอร์รัพท์ได้รับการโหลดมาเก็บไว้ใน P/V แฟลก

รูปที่ 2.11 แสดงกลุ่มคำสั่งในการโหลด 8 บิทและ 16 บิท

กลุ่มคำสั่งการโหลดข้อมูล 16 บิท

กลุ่มคำสั่งการโหลดข้อมูล 16 บิท

นีโมนิค	สัญญลักษณ์การกระทำ	แฟลก S	Z		H		P/V	N	C	อ็อพโค้ด 76 543 210	Hex	จำนวนไบท์	จำนวนM ไซเคิล	T States	Comments
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00 dd0 001 — n — — n —		3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	11 011 101 00 100 001 — n — — n —	DD 21	4	4	14	
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	11 111 101 00 100 001 — n — — n —	FD 21	4	4	14	
LD HL, (nn)	H ← (nn+1) L ← (nn)	•	•	X	•	X	•	•	•	00 101 010 — n — — n —	2A	3	5	16	
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	X	•	X	•	•	•	11 101 101 01 dd1 011 — n — — n —	ED	4	6	20	
LD IX, (nn)	IX_H ← (nn+1) IX_L ← (nn)	•	•	X	•	X	•	•	•	11 011 101 00 101 010 — n — — n —	DD 2A	4	6	20	
LD IY, (nn)	IY_H ← (nn+1) IY_L ← (nn)	•	•	X	•	X	•	•	•	11 111 101 00 101 010 — n — — n —	FD 2A	4	6	20	
LD (nn), HL	(nn+1) ← H (nn) ← L	•	•	X	•	X	•	•	•	00 100 010 — n — — n —	22	3	5	16	
LD (nn), dd	(nn+1) ← dd_H (nn) ← dd_L	•	•	X	•	X	•	•	•	11 101 101 01 dd0 011 — n — — n —	ED	4	6	20	
LD (nn), IX	(nn+1) ← IX_H (nn) ← IX_L	•	•	X	•	X	•	•	•	11 011 101 00 100 010 — n — — n —	DD 22	4	6	20	
LD (nn), IY	(nn+1) ← IY_H (nn) ← IY_L	•	•	X	•	X	•	•	•	11 111 101 00 100 010 — n — — n —	FD 22	4	6	20	
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	
PUSH qq	(SP-2) ← qqL (SP-1) ← qqH	•	•	X	•	X	•	•	•	11 qq0 101		1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP-2) ← IX_L (SP-1) ← IX_H	•	•	X	•	X	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	
PUSH IY	(SP-2) ← IY_L (SP-1) ← IY_H	•	•	X	•	X	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	qqH ← (SP+1) qqL ← (SP)	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 001	FD E1	2	4	14	

สังเกตุ

dd คือคู่รีจิสเตอร์ BC, DE, HL, SP

qq คือคู่รีจิสเตอร์ AF, BC, DE, HL

ตัวห้อย H หมายถึงข้อมูลไบท์ที่มีนัยสำคัญสูง

ตัวห้อย L หมายถึงข้อมูลไบท์ที่มีนัยสำคัญต่ำ

รูปที่ 2.12 แสดงกลุ่มคำสั่งในการโหลด 8 บิทและ 16 บิท

กลุ่มการเคลื่อนย้ายและค้นหาข้อมูลเป็นบล็อก

คำสั่งการส่งบล็อก หมายถึง คำสั่งที่ต้องการส่งข้อมูลภายในแอดเดรสหนึ่งไปยัง

อีกแอดเดรสหนึ่งที่แสดงโดยค่าของจำนวนไบท์เท่านั้น

ตัวอย่าง ถ้าต้องการส่งข้อมูลจากแอดเดรส 8280 จำนวน 128 ไบท์ไปยังแอดเดรสจากตำแหน่ง 8380 โปรแกรมที่ได้เป็นดังนี้

LD	HL,8280	ซ๊อซแอดเดรส HL	8280
LD	DE,8380	ดิสเท็นนิชั่นแอดเดรส DE	8380
LD	BC,80	จำนวนไบท์ BC	80
LDIR	LDIR	คำสั่งส่งบล็อก	
JP	MNTR	กระโดดไปที่มอนิเตอร์(Monitor)	

มิโนนิก	สัญญลักษณ์การกระทำ	แฟลก S	Z		H		P/V	N	C	ออพโค้ด 76 543 210	Hex	จำนวนไบท์	จำนวน M ไซเคิล	จำนวน T สเตท	Comments
EX DE, HL	DE—HL	•	•		•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF'	AF—AF'	•	•		•	X	•	•	•	00 001 000	08	1	1	4	
EXX	BC—BC' DE—DE' HL—HL'	•	•		•	X	•	•	•	11 011 001	D9	1	1	4	การสับเปลี่ยนค่าในรีจิสเตอร์
EX (SP), HL	H —(SP+1) L —(SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX (SP), IX	IX_H—(SP+1) IX_L—(SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	IY_H—(SP+1) IY_L—(SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE)—(HL) DE — DE+1 HL — HL+1 BC — BC-1	•	•	X	0	X	① ↕	0	•	11 101 101 10 100 000	ED A0	2	4	16	โหลด (HL) ไปยัง (DE) เพิ่มค่าพอยท์เตอร์ และลดค่า (BC) ลง
LDIR	(DE)—(HL) DE — DE+1 HL — HL+1 BC — BC-1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	ถ้า BC ≠ 0 ถ้า BC = 0
LDD	(DE)—(HL) DE — DE-1 HL — HL-1 BC — BC-1	•	•	X	0	X	① ↕	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE)—(HL) DE — DE-1 HL — HL-1 BC — BC-1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	ถ้า BC ≠ 0 ถ้า BC = 0
CPI	A - (HL) HL — HL+1 BC — BC-1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A - (HL) HL — HL+1 BC — BC-1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	ถ้า BC ≠ 0 และ A ≠ (HL) ถ้า BC = 0 หรือ A = (HL)
CPD	A - (HL) HL — HL-1 BC — BC-1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A - (HL) HL — HL-1 BC — BC-1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	ถ้า BC ≠ 0 และ A ≠ (HL) ถ้า BC = 0 หรือ A = (HL)

สังเกต 1) P/V แฟลกเป็น 0 ถ้าผลลัพธ์ของ BC - 1 = 0 นอกนั้น

2) Z แฟลกเป็น 1 ถ้า A = (HL) นอกนั้น Z = 0

รูปที่ 2.13แสดงกลุ่มคำสั่งในการเคลื่อนย้ายและค้นหาข้อมูลเป็นกลุ่ม

กลุ่มคำสั่งการกระทำทางคณิตศาสตร์ของตัวเลข 16 บิท

นีโมนิค	สัญญลักษณ์การกระทำ	แฟลก S	Z		H		P/V	N	C	ออฟโค้ด 76 543 210	Hex	จำนวนไบท์	จำนวนแมชีนไซเคิล	จำนวนสเตท	หมายเหตุ
ADD HL,ss	HL ← HL+ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss Reg. 00 BC 01 DE 10 HL 11 SP
ADC HL,ss	HL ← HL+ss+CY	↕	↕	X	X	X	V	0	↕	11 101 101 01 ss1 010	ED	2	4		
SBC HL,ss	HL ← HL-ss-CY	↕	↕	X	X	X	V	1	↕	11 101 101 01 ss0 010	ED	2	4	15	
ADD IX,pp	IX ← IX+pp	•	•	X	X	X	•	0	↕	11 011 101 00 pp1 001	DD	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY,rr	IY ← IY+rr	•	•	X	X	X	•	0	↕	11 111 101 00 rr1 001	FD	2	4	15	rr Reg. 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss+1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6	
INC IX	IX ← IX+1	•	•	X	•	X	•	•	•	11 011 101 00 100 011	DD 23	2	2	10	
INC IY	IY ← IY+1	•	•	X	•	X	•	•	•	11 111 101 00 100 011	FD 23	2	2	10	
DEC ss	ss ← ss-1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX ← IX-1	•	•	X	•	X	•	•	•	11 011 101 00 101 011	DD 2B	2	2	10	
DEC IY	IY ← IY-1	•	•	X	•	X	•	•	•	11 111 101 00 101 011	FD 2B	2	2	10	

สังเกต SS เป็นคู่รีจิสเตอร์ BC, DE, HL, SP
pp เป็นคู่รีจิสเตอร์ BC, DE, IX, SP
rr เป็นคู่รีจิสเตอร์ BC, D, IY, SP

สังเกตุ SS เป็นคู่รีจิสเตอร์ BC,DE,HL,SP
PP เป็นคู่รีจิสเตอร์ BC,DE,IX,SP
rr เป็นคู่รีจิสเตอร์ BC,D,IY,SP

กลุ่มคำสั่งทางคณิศาสตร์และลอจิก

คำสั่งคำนวนมี 2 ชนิด คือ คำสั่งคำนวนเลขลอจิก และคำสั่งคำนวนเลขธรรมดา ในกรณีคำสั่งการคำนวน 8 บิท ดิสเท็นนิชั่นได้ถูกกำหนดที่เอ็คคูมูเตอร์(A) หรือในกรณีการคำนวนเลข 16 บิท ใช้รีจิสเตอร์ HL,IX,IY

กลุ่มคำสั่งการกระทำทางคณิตศาสตร์และลอจิก

นีโมนิค	สัญญลักษณ์การกระทำ	แฟลก S	Z		H		P/V	N	C	ออพโค้ด		จำนวนไบท์	จำนวน M ไซเกิล	จำนวนสเตท	หมายเหตุ
ADD A,r	A ← A+r	↕	↕	X	↕	X	V	0	↕	10 000 r		1	1	4	r Reg.
ADD A,n	A ← A+n	↕	↕	X	↕	X	V.	0	↕	11 000 110 - n -		2	2	7	000 B 001 C
ADD A,(HL)	A ← A+(HL)	↕	↕	X	↕	X	V	0	↕	10 000 110		1	2	7	010 D 011 E
ADD A,(IX+d)	A ← A+(IX+d)	↕	↕	X	↕	X	V	0	↕	11 011 101 10 000 110 - d -	DD	3	5	19	100 H 101 L
ADD A,(IY+d)	A ← A+(IY+d)	↕	↕	X	↕	X	V	0	↕	11 111 101 10 000 110 - d -	FD	3	5	19	111 A
ADC A,s	A ← A+s+CY	↕	↕	X	↕	X	V	0	↕	001					S คือรีจิสเตอร์ใด ๆ r, n,
SUB s	A ← A-s	↕	↕	X	↕	X	V	1	↕	010					(HL), (IX+d),
SBC A,s	A ← A-s-CY	↕	↕	X	↕	X	V	1	↕	011					(IY+d) ดังที่อยู่ในคำสั่ง
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0	100					ADD ถ้าเป็นคำสั่งอื่น
OR s	A ← A ∨ s	↕	↕	X	0	X	P	0	0	110					ให้เปลี่ยน 000 ในคำสั่ง
XOR s	A ← A ⊕ s	↕	↕	X	0	X	P	0	0	101					ADD ให้เป็นไปตามคำสั่งนั้น
CP s	A - s	↕	↕	X	↕	X	V	1	↕	111					
INC r	r ← r+1	↕	↕	X	↕	X	V	0	•	00 r 100		1	1	4	
INC (HL)	(HL) ← (HL)+1	↕	↕	X	↕	X	V	0	•	00 110 100		1	3	11	
INC (IX+d)	(IX+d) ← (IX+d)+1	↕	↕	X	↕	X	V	0	•	11 011 101 00 110 100 - d -	DD	3	6	23	
INC (IY+d)	(IY+d) ← (IY+d)+1	↕	↕	X	↕	X	V	0	•	11 111 101 00 110 100 - d -	FD	3	6	23	
DEC s	s ← s-1	↕	↕	X	↕	X	V	1	•	101					S คือรีจิสเตอร์ใด ๆ r, (HL), (IX+d), (IY+d) ดังที่อยู่ในคำสั่ง INC ถ้าเป็นการเซ็ตของ DEC ก็เพียงแต่เปลี่ยน 100 ด้วย 101

สังเกตว่า แฟลก P/V ถ้า P หมายถึง การแสดงพาริตี้ และ V หมายถึงการแสดงการโอเวอร์โฟลว

0 แฟลกไม่เปลี่ยน 0 = แฟลกรีเซท 1 = แฟลกเซท

X เป็นอะไรก็ได้ • = แฟลกจะเป็นไปตามเงื่อนไข

กลุ่มคำสั่งการกระทำทางคณิตศาสตร์โดยเฉพาะบางอย่างและการควบคุมซีพียู

นีโมนิค	สัญญลักษณ์การกระทำ	แฟลก S	Z		H		P/V	N	C	ออพโค้ด 76 543 210	Hex	จำนวนไบท์	จำนวน M ไซเกิล	จำนวนสเตท	หมายเหตุ
DAA	เปลี่ยนข้อมูลในแอคคิวมูเลเตอร์ให้เป็นตัวเลข BCD โดยใช้ตามหลังคำสั่ง ADD หรือ SUB เพื่อบวกหรือลบเลข BCD	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	การปรับข้อมูลในแอคคิวมูเลเตอร์เป็น BCD
CPL	A ← $\overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	ทำการคอมพลีเมนท์แอคคิวมูเลเตอร์
NEG	A ← $\overline{A}$+1	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	ทำแอคคิวมูเลเตอร์ให้เป็นเลขฐบหรือ 2's คอมพลีเมนท์
CCF	CY ← $\overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI*	IFF ← 0	•	•	X	•	X	•	•	•	1: 110 011	F3	1	1	4	
EI*	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

สังเกต IFF แสดงผลของอินเตอร์รัพท์ฟลิบฟลอบ

CY แสดงลักษณะของฟลิบฟลอบตัวทด

รูปที่ 2.14 แสดงกลุ่มคำสั่งทางคณิตศาสตร์และลอจิก

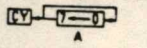
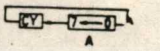
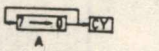
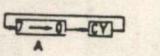
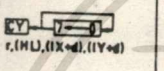
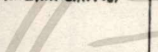
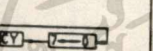
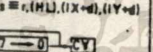
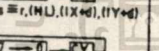
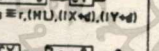
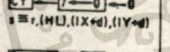
กลุ่มคำสั่งการเคลื่อนย้ายข้อมูลเป็นวงรอบและการชิฟท์

คำสั่งที่ต้องการชิฟท์ข้อมูลภายในรีจิสเตอร์ หรือข้อมูลที่ชี้แสดงภายในหน่วยความจำจะใช้คำสั่งอันนี้ หลักการจะใช้การควบคุมในระบบชิฟท์แบบอนุกรม ซึ่งจะใช้เวลาเข้ามาเกี่ยวข้องด้วย

ตัวอย่าง RLC (Rotate Left Circuit)

ข้อมูลที่ชี้แสดงโดยซ็อซจะถูกชิฟท์ไปทางซ้าย 1 บิท ซึ่งหลังจากคำสั่งนี้บิทตำแหน่งสูงสุด (MSB) จะถูกย้ายไปที่ตำแหน่งของบิทต่ำสุด (LSB) และบิทสูงสุด (MSB) ก็จะไปเข้าที่แคร่แฟลก รูปแสดงการทำงานเป็นดังนี้

กลุ่มคำสั่งในการเลื่อนบิทและหมุนบิท

นีโมนิค	สัญญลักษณ์การกระทำ	แฟลก S	Z		H		P/V	N	C	ออพโค้ด 76 543 210	Hex	จำนวนไบท์	จำนวนแมชชีนไซเคิล	จำนวนสเตท	หมายเหตุ
RLCA		•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	Rotate left circular accumulator
RLA		•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	Rotate left accumulator
RRCA		•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	Rotate right circular accumulator
RRA		•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	Rotate right accumulator
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 00 000 r	CB	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 000 110	CB	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX+d)	 r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 – d – 00 000 110	DD CB	4	6	23	
RLC (IY+d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 – d – 00 000 110	FD CB	4	6	23	
RL s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	010					Instruction format and states are as shown for RLC's. To form new Op Code replace 000 of RLC's with shown code
RRC s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	001					
RR s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	011					
SLA s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	100					
SRA s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	101					
SRL s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	111					
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

รูปที่ 2.15 แสดงกลุ่มคำสั่งการเคลื่อนย้ายข้อมูลเป็นวงรอบและการชิฟท์

กลุ่มคำสั่งการกระทำในส่วนบิท

ในการเขียนโปรแกรมบางที่ต้องการที่จะกำหนดให้ตำแหน่งของบิททำงาน ตัวอย่างเช่นในกรณีที่ต้องการให้ข้อมูลจาก I/O อ่านเข้าและภายในข้อมูลนี้ 1บิท เท่านั้นนำมาตรวจสอบหรือว่าต้องการที่จะเซทบิทใดบิทหนึ่ง คำสั่งของการกระทำงานในสถานะนี้จะใช้คำว่า BIT

ตัวอย่าง ถ้าจ้องการตรวจสอบข้อมูลที่ชี้แสดงที่รีจิสเตอร์ HL ที่บิทที่ 4 ว่ามีค่าเป็น 0 หรือ 1 คำสั่งเป็นดังนี้

BIT 4,(HL)

หลังจากการทำงานของคำสั่งนี้ ถ้าค่าของข้อมูลที่บิทที่ 4 มีค่า 0,Z แฟลกจะถูกเซท ถ้ามีค่า 1,Z แฟลกจะถูกรีเซท ข้อมูลที่ชี้แสดงที่แอดเดรส(HL)จะไม่เปลี่ยนแปลง

กลุ่มคำสั่งในการเซท รีเซท และทดสอบบิท

มีโนมิค	สัญญลักษณ์	แฟลก S	Z		H		P/V	N	C	ออพโค้ด 76 543 210	Hex	จำนวนไบท์	จำนวนแมชีนไซเคิล	จำนวน T สเตท	หมายเหตุ
BIT b, r	$Z \leftarrow \overline{r_b}$	X	↕	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	8	r: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A
BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	X	↕	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	b: 000 Bit Tested 0, 001 1, 010 2, 011 3, 100 4, 101 5, 110 6, 111 7
BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)_b}$	X	↕	X	1	X	X	0	•	11 011 101 11 001 011 – d – 01 b 110	DD CB	4	5	20	
BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)_b}$	X	↕	X	1	X	X	0	•	11 111 101 11 001 011 – d – 01 b 110	FD CB	4	5	20	
SET b, r	$r_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 11 b r	CB	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 11 b 110	CB	2	4	15	
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 – d – 11 b 110	DD CB	4	6	23	
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 – d – 11 b 110	FD CB	4	6	23	
RES b, s	$s_b \leftarrow 0$ s ≡ r, (HL), (IX+d), (IY+d)	•	•	X	•	X	•	•	•	10					ในการสร้างออพโค้ดตัวใหม่ ให้เปลี่ยน 11 ในกลุ่มคำสั่ง SET ด้วยค่า 10 แฟลกและช่วงเวลาการทำงานเหมือนกับคำสั่งเซท

สังเกต สัญญลักษณ์ s_b แทนบิท b (0 ถึง 7) ของตำแหน่งรีจิสเตอร์ s หรือ แอดเดรส s

สังเกตุ สัญญลักษณ์ sb แทนบิท 6(0ถึง7) ของตำแหน่งรีจิสเตอร์ s หรือ แอดเดรส s

รูปที่2.16แสดงกลุ่มคำสั่งการกระทำในส่วนบิท

กลุ่มคำสั่งการกระโดด

คำสั่ง JUMP เป็นคำสั่งที่ทำการเปลี่ยนแปลงลำดับการทำงานของโปรแกรม โปรแกรมตามธรรมดาหลังจากเมื่อถูกรีเซทแล้ว การทำงานจะเริ่มจากแอดเดรส 0 แต่ถ้าในกรณีนี้ที่ได้รับการควบคุมอย่างใดอย่างหนึ่งก็ตาม ค่าผลลัพท์ที่ได้ก็จะให้การทำงานแตกต่างกันออกไป ในกรณีนี้ลำดับการทำงานของโปรแกรมจะต้องเปลี่ยนไปด้วย ซึ่งจำเป็นจะต้องเปลี่ยนไปด้วย ซึ่งจำเป็นจะต้องใช้คำสั่ง JUMP

ตัวอย่าง JP 8100H

เมื่อคำสั่งนี้ทำงาน การทำงานจะกระโดดไปที่แอดเดรส 8100 อย่างไม่มีเงื่อนไข

กลุ่มคำสั่งการกระโดด

มีโมนิค	สัญญลักษณ์การกระทำ	แฟลก S	Z		H		P/V	N	C	ออพโค้ด 76 543 210	Hex	จำนวนไบท์	จำนวนแมชชีนไซเคิล	จำนวน T สเตท	หมายเหตุ
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 ← n ← ← n ←	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 ← n ← ← n ←		3	3	(10)	cc | เงื่อนไข 000 | NZ ไม่ใช่ศูนย์ (Zแฟลก = 0) 001 | Z (Z = 1) 010 | NC (C = 0) 011 | C (C = 1) 100 | PO (P = 0) 101 | PE (P = 1) 110 | P (B = 0) 111 | M (S = 1)
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000 ← e-2 ←	18	2	3	12	
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 ← e-2 ←	38	2	2	7	ถ้าไม่เป็นไปตามเงื่อนไข
	If C = 1, PC ← PC+e											2	3	12	
JR NC, e	If C = 1, continue	•	•	X	•	X	•	•	•	00 110 000 ← e-2 ←	30	2	2	7	ถ้าไม่เป็นไปตามเงื่อนไข
	If C = 0, PC ← PC+e											2	3	12	
JR Z, e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 ← e-2 ←	28	2	2	7	ถ้าไม่เป็นไปตามเงื่อนไข
	If Z = 1, PC ← PC+e											2	3	12	
JR NZ, e	If Z = 1, cootinue	•	•	X	•	X	•	•	•	00 100 000 ← e-2 ←	20	2	2	7	ถ้าไม่เป็นไปตามเงื่อนไข
	If Z = 0, PC ← PC+e											2	3	12	
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	B ← B-1 If B = 0, continue	•	•	X	•	X	•	•	•	00 010 000 ← e-2 ←	10	2	2	8	ถ้า B ≠ 0
	If B ≠ 0, PC ← PC+e											2	3	13	ถ้า B ≠ 0 ดู = 8

สังเกต e แทนข้อมูลที่จะใช้อ้างอิง

e เป็นเลข 2's คอมพลีเมนท์ มีค่าจาก -126,129

e-2เป็นออปโค้ดเพื่อให้ค่าแอดเดรสของ PC+ e เป็นค่า PC ใหม่

รูปที่ 2.17 แสดงกลุ่มคำสั่งการกระโดด

กลุ่มคำสั่งในการเรียกโปรแกรมย่อย และกลับเข้าสู่โปรแกรมหลัก

มีโนนิค	สัญญลักษณ์การกระทำ	แฟลก S	Z		H		P/V	N	C	ออพโค้ด 76 543 210	Hex	จำนวนไบท์	จำนวนแมชีนไซเกิล	จำนวน T สเตท	หมายเหตุ
CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC \leftarrow nn$	•	•	X	•	X	•	•	•	11 001 101 – n – – n –	CD	3	5	17	
CALL cc, nn	ถ้าเงื่อนไข CC ไม่เป็นจริง	•	•	X	•	X	•	•	•	11 cc 100 – n – – n –		3	3	10	ถ้า CC ไม่เป็นจริง
	โปรแกรมจะกระทำต่อไป											3	5	17	
	แต่ถ้าเป็นจริง คำสั่งนี้จะเหมือน CALL nn														
RET	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$	•	•	X	•	X	•	•	•	11 001 001	C9	1	3	10	
RET cc	ถ้าเงื่อนไข CC ไม่เป็นจริง	•	•	X	•	X	•	•	•	11 cc 000		1	1	5	ถ้า CC ไม่เป็นจริง
	โปรแกรมจะกระทำต่อไป											1	3	11	
	แต่ถ้าเป็นจริง คำสั่งนี้จะเหมือนกับ RET														
RETI	เป็นการกลับเข้าสู่โปรแกรมหลักหลังจากถูกอินเตอร์รัพท์	•	•	X	•	X	•	•	•	11 101 101 01 001 101	ED 4D	2	4	14	
RETN[1]	เป็นการกลับเข้าสู่โปรแกรมหลักหลังจากได้รับการอินเตอร์รัพท์แบบนอนมาสเคเบิ้ล	•	•	X	•	X	•	•	•	11 101 101 01 000 101	ED 45	2	4	14	
RST p	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC_H \leftarrow 0$ $PC_L \leftarrow p$	•	•	X	•	X	•	•	•	11 t 111		1	3	11	

CC	เงื่อนไข
000	NZ (Z = 0)
001	Z (Z = 1)
010	NC (C = 1)
011	C (C = 1)
100	PO (P = 0)
101	PE (P = 1)
110	P (S = 0)
111	M (S = 1)

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

RETN จะกระทำการโหลด IFF_2 IFF_1

รูปที่2.18 (ต่อ)แสดงกลุ่มคำสั่งการถูกกระโดด

กลุ่มคำสั่งเกี่ยวกับอินพุทเอาท์พุท

จากที่ได้กล่าวมาแล้ว แอดเดรสที่ใช้อยู่ในช่วงแอดเดรสจาก 0 ถึง 64K(65532) สำหรับ Z-80 ซีพียูจะมีแอดเดรสพิเศษสำหรับ I/O(อินพุท/เอาว์พุท)ซึ่งจะให้ข้อมูลติดต่อกับภายนอกได้ ช่วงของI/O แอดเดรสอยู่ในตำแหน่งจาก 00 ถึง FFH

คำสั่ง IN คำสั่งนี้ข้อมูลจากภายนอกถูกให้เขียนเข้าไปที่ซีพียู

ตัวอย่าง IN A,C

ข้อมูลของ I/O แอดเดรสที่ชี้แสดงที่รีจิสเตอร์ C ถูกเขียนเข้าไปที่แอดคูมูเลเตอร์

นีโมนิค	สัญญลักษณ์การกระทำ	แฟลก S	Z		H		P/V	N	C	ออพโค้ด 76 543 210	Hex	จำนวนไบท์	จำนวนแมชีนไซเคิล	จำนวน T สเตท	หมายเหตุ
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 ← n →	DB	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) ถ้า r = 110 เท่านั้น แฟลกจึงจะมีผล	‡	‡	X	‡	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B-1 HL ← HL+1	X	① ‡	X	X	X	X	1	•	11 101 101 10 100 010	ED A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B-1 HL ← HL+1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 010	ED B2	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B-1 HL ← HL-1	X	① ‡	X	X	X	X	1	•	11 101 101 10 101 010	ED AA	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B-1 HL ← HL-1 ทำซ้ำจนกระทั่ง B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 010	ED BA	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011	D3	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 01 r 001	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B-1 HL ← HL+1	X	① ‡	X	X	X	X	1	•	11 101 101 10 100 011	ED A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B-1 HL ← HL+1 ทำซ้ำจนกระทั่ง B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 011	ED B3	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B-1 HL ← HL-1	X	① ‡	X	X	X	X	1	•	11 101 101 10 101 011	ED AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTDR	(C) ← (HL) B ← B-1 HL ← HL-1 ทำซ้ำจนกระทั่ง B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 011	ED BB	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

สังเกต 1. ถ้าผลลัพธ์ B - 1 เป็นศูนย์

Z แฟลก จะได้รับการเซท นอกนั้นจะได้รับการรีเซท

รูปที่ 2.19 แสดงกลุ่มคำสั่งเกี่ยวกับอินพุท-เอาว์พุท

คำสั่ง OUT

คำสั่งนี้จะกลับกับคำสั่ง IN ข้อมูลของรีจิสเตอร์ที่ชี้แสดงด้วยค่าโอเปอร์แรนด์จะให้ข้อมูลจากแอดคิวมูเลเตอร์อ่านออกไปที่ I/O แอดเดรสที่ชี้แสดงด้วยค่าโอเปอร์แรนด์

ตัวอย่าง OUT (C) , D

ข้อมูลของ D รีจิสเตอร์จะถูกให้อ่านไปที่ I/O แอดเดรสที่ชี้แสดงที่รีจิสเตอร์ C

กลุ่มคำสั่งควบคุมซีพียู

การควบคุมการทำงานมีหลายคำสั่ง เช่น

NOP (No Operation) เป็นคำสั่งที่ไม่ให้ซีพียูทำงานแต่โปรแกรมเคาว์เตอร์ถูกเพิ่มด้วยค่า 1 ข้อมูลภายในรีจิสเตอร์ และแฟลกต่างๆจะไม่เปลี่ยนแปลง คำสั่งนี้จะใช้ในการปรับแต่งช่วงเวลาได้ในกรณีที่ใช้ทำโปรแกรมของไทม์เมอร์

HALT เป็นคำสั่งให้ซีพียูหยุดทำงาน เพื่อให้สถานะการทำงานของคำสั่งนี้หยุดลงในขณะที่มีการอินเตอร์รัพรับภายนอกเข้าหรือทำการรีเซท

นอกจากนี้มีคำสั่งเกี่ยวกับการอินเตอร์รัพ เช่น EI,DI,IMO,IM1,IM2

Memory and Memory decoder

หน่วยความจำหลักในระบบไมโครคอมพิวเตอร์ เป็นความจำระบบอิเล็คทรอนิคส์ สามารถที่จะจำข้อมูลในรูปของตัวเลขซึ่งอยู่ในรูปของเลขไบนารี่(เลขฐาน) คือเลข"0" กับ "1" ตัวเลขไบนารี่ 1หลักเรียกว่า 1 บิท(bit)เลขไบนารี่หลายบิทที่คอมพิวเตอร์ถือว่าเป็นคำสั่งหรือข้อมูลที่คอมพิวเตอร์อ่าน(read)เข้ามาหรือเขียน(write)ออกไปเรียกว่าคำ(word) จำนวนเลขไบนารี่ที่ประกอบเป็นระหัส 1 คำ ถือว่าเป็นความยาวของคำ(word lenght)ซึ่งมีคำเรียกตามความหมายของคำนั้น เช่น ถ้าเป็นความยาว 4 บิทเรียกว่าnibble คำที่มีความยาว 8 บิทเรียกว่า 1 ไบท์(byte)ซึ่งขนาดของคำที่ไมโครคอมพิวเตอร์ทั่วๆไปใช้กันอยู่ หน่วยความจำจะสามารถเก็บคำสั่งหรือข้อมูลเหล่านี้ไว้ในตำแหน่งใด ตำแหน่งหนึ่ง ในหน่วยความจำ แต่ละตำแหน่งซีพียูสามารถที่จะอ้างถึงได้ เรียกว่า Word Position หรือแอดเดรส (Address)

การนำข้อมูลจากภายนอกเข้าไปเก็บในหน่วยความจำที่แอดเดรสใดๆเรียกการเขียน (memory write operation)ถ้านำข้อมูลออกจากหน่วยความจำเรียกว่าการอ่าน(memory read opertion)

วิธีการค้นหาข้อมูลหรือเข้าไปหาข้อมูลในหน่วยความจำที่แอดเซส(access)ซึ่งแบ่งตามลักษณะได้ 2 แบบใหญ่ๆคือแบบแรกเรียกว่า แบบแรนดอม(random access)ซึ่งสามารถที่

จะเข้าไปถึง ณ.แอดเดรสใดๆได้โดยตรงจึงทำให้ใช้เวลาเท่ากันในการอ้างแอดเดรสและแบบที่สองคือ ซีเควนเชียล(Sequential access)เราใช้เวลาเข้าถึงข้อมูลที่แอดเดรสใดๆไม่เท่ากัน จะขึ้นอยู่กับตำแหน่งของข้อมูลนั้นๆ เช่นที่แอดเดรส 100 จะใช้เวลาน้อยกว่าที่แอดเดรส 150 เพราะการเข้าถึงข้อมูลจะทำได้โดยลำดับก่อนหลัง ไม่สามารถอ้างถึงได้โดยตรง

ชนิดของหน่วยความจำ

หน่วยความจำซึ่งมีความจุมากๆ ปัจจุบันถูกสร้างให้อยู่ในรูปของไอซีชิพเดียวยิ่งมีขนาดและราคาถูก การนำเอาไปใช้งานก็สะดวกระบบความจำที่มีความจุมากๆสามารถนำเอาหน่วยความจำทางอิเลคโทรนิคส์หลายๆ ชิพมาประกอบกัน เพื่อให้ความจุตามต้องการ

โครงสร้างของชิพหน่วยความจำโดยทั่วไปจะมีสายสัญญาณใช้งานต่างๆเกี่ยวข้องดังต่อไปนี้

1.สายแอดเดรส(address line)เป็นสัญญาณที่ใช้เลือกตำแหน่งที่จะทำการอ่านหรือเขียน สายส่วนนี้จะต่อโดยตรงกับ แอดเดรสบัส(address bus)ของซีพียู

2.สายดาต้า(data in, data out)ซึ่งจะเป็นบัสแบบสองทิศทาง(bidirectional bus) คือใช้ได้เป็นทั้งเป็นทางเดินของข้อมูลเข้าและข้อมูลออกในเส้นเดียวกัน

3.ชิพอีนาเบิล(chip enable)เป็นสายสัญญาณควบคุมให้หน่วยความจำชิพใดๆสามารถทำการอ่านหรือเขียนกับซีพียูได้

4.เขียนหรืออ่าน(write/read)คือสัญญาณให้ซีพียูสามารถอ่านหรือเขียนกับหน่วยความจำที่ถูกเลือก

หน่วยความจำแบ่งออกเป็น 2 ประเภท คือ RAM และ ROM

RAM (Random access memory)

เป็นหน่วยความจำที่สามารถทำการอ่านหรือเขียนได้ทั้งสองอย่าง ข้อมูลที่เก็บไว้ในหน่วยความจำแบบนี้ จะอยู่ได้ตราบใดที่ยังมีไฟเลี้ยงจ่ายให้กับชิพอยู่ และข้อมูลจะหายไปเมื่อไม่มีไฟเลี้ยงหน่วยความจำแบบนี้เราเรียกว่า Static RAM และมี RAM อีกประเภทหนึ่งเรียกว่า Dynamic RAM เป็นหน่วยความจำที่สามารถเก็บข้อมูลของตัวเองได้ในระยะเวลาสั้น ข้อมูลจะสูญหายไป ถึงแม้จะมีไฟเลี้ยงอยู่ก็ตาม ดังนั้นจึงต้องมีการ รีเฟรช (refreshing) อยู่ทุกระยะข้อมูลจึงจะไม่สูญหาย แต่ถึงแม้ว่าหน่วยความจำแบบนี้ จะต้องอาศัยขบวนการรีเฟรชซึ่งยุ่งยากแต่ก็มีข้อดีที่คุ้มค่า คือ สามารถสร้างเป็นหน่วยความจำที่มีความจุสูงๆได้

ROM (Read Only Memory)

เป็นหน่วยความจำแบบที่สามารถใช้อ่านได้อย่างเดียว กล่าวคือ เรานำเอาข้อมูลเข้าไปเก็บไว้ซึ่งเรียกว่าการโปรแกรมรอม แล้วข้อมูลจะไม่สูญหายไปถึงแม้จะไม่มีไฟเลี้ยงป้อนให้ชิพก็ตามดังนั้นข้อมูลที่เรานำเข้าไปเก็บไว้จะเป็นข้อมูลที่มีความสำคัญ และต้องใช้งานอยู่เป็นประจำ เช่น โปรแกรมควบคุมหรือสั่งทำงานต่างๆเป็นต้น

รอมยังแบ่งออกได้เป็นหลายประเภทดังนี้

ROM โปรแกรมจะเขียนมาแล้วจากโรงงานผู้ผลิตซึ่งมีราคาแพง

PROM (Programmable read-only memory) เป็นรอมที่ผู้ใช้งานสามารถโปรแกรมใช้งานได้ตามความต้องการแต่ไม่สามารถจะเปลี่ยนแปลงหรือลบข้อมูลได้ กล่าวคือโปรแกรมได้เพียงครั้งเดียว

EPRROM(Erasable programmable read-only memory)เป็นรอมที่ใช้งานสามารถทำการโปรแกรมใช้งานเองได้และยังสามารถเปลี่ยนแปลงข้อมูลได้กล่าวคือลบล้างข้อมูลเก่าได้และยังเขียนใหม่ได้อีกหลายครั้งโดยการฉายแสงอุลตร้าไวโอเลทลงตรงช่องบนชิพ

EAROM(Elextrically alterrable read-only memory)การโปรแกรมทำได้โดยผู้ใช้งานเหมือน EPROM แต่การลบล้างข้อมูลต้องใช้กระแสไฟฟ้า

เมื่อไมโครโปรเซสเซอร์จะอ่านหรือเขียนข้อมูลกับหน่วยความจำ หน่วยควบคุมจะส่งสัญญาณการอ้างแอดเดรสที่แอดเดรสบัส โดยทั่วไปรวมทั้ง Z-80 ซีพียู จะมี 16 เส้น ดังนั้นมันสามารถที่จะอ้างแอดเดรสได้ถึง 2^{16} ก็เท่ากับ 65,536 ตำแหน่ง ซึ่งโดยทั่วไปเรานิยมเรียกเป็นหน่วยของพันคือ 1024 ตำแหน่ง(ไบท์)เท่ากับ 1 กิโลไบท์ ดังนั้นสายแอดเดรส 16 จะอ้างได้ถึง 64K

ถ้าชิพหน่วยความจำที่มีความจุ 4K ดังนั้นจะต้องใช้แอดเดรส 12 เส้นคือ A0-A11 ในกรณี 2K ก็ต้องใช้แอดเดรส 11 เส้น A0-A10

จากที่ได้กล่าวมาแล้วในตอนต้น การที่จะสร้างระบบหน่วยความจำให้มีขนาดความจุมากๆจำเป็นจะต้องใช้ชิพหน่วยความจำ ซึ่งอาจจะเป็นทั้งรอมและแรมที่มีความจุจำกัดมาประกอบกันเข้าเพื่อให้ได้ความจุตามต้องการ ตัวอย่างเช่นระบบความจำขนาด 10K อาจจะใช้ชิพขนาด 2K จำนวน 5 ตัวประกอบ หรืออาจจะใช้ชิพขนาด 4K 2 ตัวและ 2K อีกหนึ่งตัวมาประกอบก็ได้ ขึ้นอยู่กับการออกแบบและความเหมาะสมของงาน

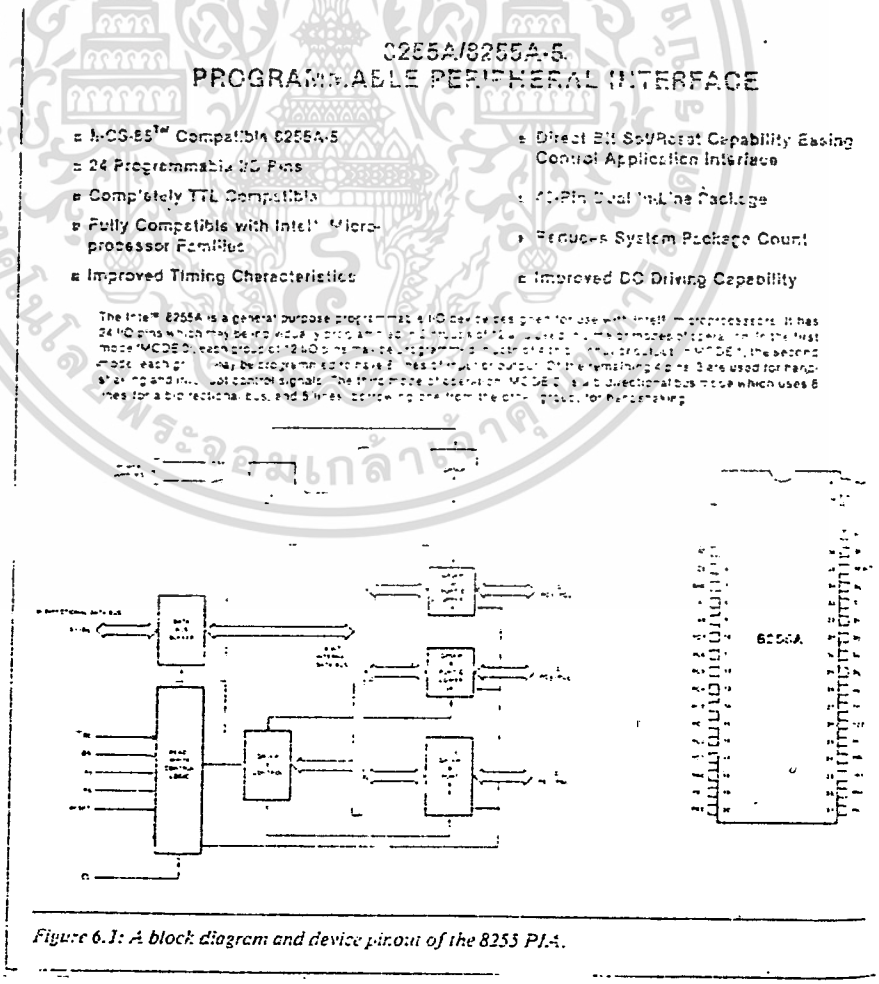
เมื่อมีชิพหน่วยความจำหลายๆชิพมาต่อกันดังกล่าวจึงจำเป็นอย่างยิ่งที่จะต้องมีการถอดระหัสเข้าช่วย เพราะข้อมูลที่อ้างถึงอาจจะอยู่ที่ชิพใด ชิพหนึ่งในระบบการถอดระหัส(decode) จะเป็นการให้สัญญาณอีนาเบิล(enable)แก่ $\overline{CS}$ ของชิพ ซึ่งจะทำให้ซีพียูสามารถเข้าถึงแอดเดรสที่อยู่บนชิพนั้นได้ กล่าวคือในการออกแบบฮาร์ดแวร์ แต่ละชิพในระบบความจำจะมีตำแหน่งแอดเดรสที่แน่นอนของตัวเอง เช่นชิพที่ 1 ขนาด 2 ให้เป็นแอดเดรส 0000H ถึง07FF H ชิพที่ 2 ให้เป็นแอดเดรสที่ 0800 Hถึง OFFFH(มีขนาด 2k)เป็นต้น

8255 I/O Interface

เป็นส่วนที่เชื่อมต่ออุปกรณ์ อินพุท เอาว์พุทเข้ากับซีพียู โดยวงจรของเครื่อง Automatic Vidio Switcher นี้ใช้ไอซี 8255 PPI(Programmable Peripgeral Interface)ซึ่งเป็นไอซีที่บริษัทอินเทลผลิตขึ้นมาสำหรับสนับสนุนการใช้งานของซีพียู 8080 โดยเฉพาะ แต่ก็ยังสามารถที่จะใช้กับซีพียู Z-80 และซีพียูตัวอื่นซึ่งมีลักษณะของสัญญาณการติดต่อควบคุมต่างๆ ใกล้เคียงกันได้ โดยมีการดัดแปลงแก้ไขนิดหน่อยเท่านั้นเอง

มีลักษณะพิเศษคือเป็นชิพอินพุท เอาว์พุท ซึ่งสามารถโปรแกรมกำหนดลักษณะการทำงานได้โดย software 8255 เป็นไอซีขนาด 40ขา โดยจะมีลักษณะของพอร์ทแบบขนานถึง 3 พอร์ท พอร์ทละ 8 บิท ให้ใช้งานรวมเป็น 24 ขาด้วยกัน การติดต่อกับซีพียูทำได้โดยต่อกับ Data bus 8ขา ต่อกับสัญญาณควบคุมอีก 6ขา และอีก 2 ขาเป็นแหล่งจ่ายไฟกับกราวน์

ซีพียูสามารถที่จะเลือก function การทำงานได้โดยส่งข้อมูล 8 บิทซึ่งเรียกว่าผ่าน Data bus มาให้กับ 8255 ซึ่งจะกล่าวโดยละเอียดต่อไปนี้


8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel® Microprocessor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- 40-Pin Dual In-Line Package
- Reduces System Package Count
- Improved DC Driving Capability

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel microprocessors. It has 24 I/O pins which may be individually programmed in two groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

Figure 6.1: A block diagram and device pinout of the 8255 PIA.

รูปที่ 2.20 แสดงตำแหน่งขาและโครงสร้างภายในของ 8255

ตำแหน่งขาต่างและโครงสร้างภายในแสดงให้เห็นได้ดังรูปที่ 4.1 ซึ่งอธิบายหลักการทำงานได้คือ

Data bus buffer มีโครงสร้างการทำงานเพื่อจะต่อกับ Data bus ของระบบซึ่งเราจะต่อโดยตรงกับ Data bus ของซีพียูดังนั้น Buffer ตัวนี้จะเป็นตัวกักหรือส่งข้อมูลแบบ 2 ทิศทาง จึงใช้วงจรโลจิกสามสถานะ (Tri state) ข้อมูลที่ผ่าน Data bus buffer นี้จะกระทำได้โดยคำสั่ง IN หรือคำสั่ง OUT ซึ่งเป็นคำสั่งมาจากซีพียูและรวมถึงการเลือก Mode (Control word และ Status)

Read Unit Control Logic ส่วนนี้จะเป็นส่วนควบคุมและจัดการเกี่ยวกับการเคลื่อนย้ายข้อมูลของ Internal data bus และ External data bus ควบคุมทั้งสัญญาณที่เป็นข้อมูลและสัญญาณควบคุมเองด้วย ส่วนนี้จะรับสัญญาณ Address และสัญญาณควบคุมจากซีพียูอีกที่หนึ่งซึ่งตัวมันเองจะทำการถอดระหัสเพื่อควบคุมการทำงานของ Group a control และ Group B control

$\overline{CS}$ Chip Select Input ที่ขานี้ถ้าได้รับโลจิก (Logic) "0"หมายความว่าซีพียูสามารถติดต่อ(อ่านหรือเขียน)กับ 8255 ได้

$\overline{RD}$ Read Input ถ้าขานี้ได้รับโลจิก"0"แล้ว 8255 จะทำการส่งข้อมูลจากพอร์ทหรือข้อมูลที่แสดงสถานะการทำงานไปยังซีพียูโดยผ่าน Data bus หรือกล่าวได้ว่าซีพียูจะทำการอ่านข้อมูลจาก 8255 นั่นเอง

$\overline{WR}$ write Input เมื่อ 8255 รับสัญญาณเข้ามาที่ขานี้เป็นโลจิก"0"ซีพียูจะสามารถทำการส่ง(เขียน)ข้อมูลซึ่งอาจจะเป็น Data ไปที่พอร์ทหรืออาจเป็นข้อมูลที่เป็นคำสั่งไปยัง 8255

A0-A1(Address Input)สัญญาณที่ป้อนเข้ามาที่ขานี้จะทำงานพร้อมกับ RD และ WR ทำให้สามารถเลือกพอร์ทได้ 3 พอร์ทโดยเหลืออีกหนึ่งพอร์ทจะเป็นพอร์ทที่รับคำสั่งควบคุมที่เรียกว่า "Control word" ซึ่งพอร์ทนี้เราจะเรียกว่า"Controlwort register" โดยทั่วไปขา A0,A1 นี้จะต่ออยู่กับบิทล่างสุดของ Address bus ซึ่งก็คือ A0,A1 Address bus นั่นเอง

สำหรับการเลือกพอร์ทแสดงให้เห็นในตาราง 2.1

A1	A1	RD	WR	CS	function การทำงาน	
0	0	0	1	0	READ เป็นอินพุท	อ่านข้อมูลเข้าพอร์ท A
0	1	0	1	0		อ่านข้อมูลเข้าพอร์ท B
1	0	0	1	0		อ่านข้อมูลเข้าพอร์ท C
1	1	0	1	0		ไม่ใช้งาน
0	0	1	0	0	WRITE เป็นเอาท์พุท	ส่งข้อมูลออกพอร์ท A
0	1	1	0	0		ส่งข้อมูลออกพอร์ท B
1	0	1	0	0		ส่งข้อมูลออกพอร์ท C
1	1	1	0	0		พอร์ทควบคุมการทำงาน
X	X	X	X	1	บิทข้อมูลที่ไม่เกี่ยวข้องกัน	

ตารางที่ 2.1 แสดงลักษณะการเลือกพอร์ท

RESET เมื่อใดที่ขาอินพุทนี้เป็น"0"ในทางโลจิก Control word register จะถูกเคลียร์ ทำให้ทุกพอร์ทถูกเซทให้เป็น อินพุท โหมด (Input mode)

PA0-PA7 ขาทั้ง 8ขานี้ถูกใช้เป็นอินพุท เอาว์พุทพอร์ทสำหรับพอร์ท A ซึ่งสามารถที่จะติดต่อกับอุปกรณ์ภายนอกได้

PB0-PB7 ขาทั้ง 8 นี้ถูกใช้เป็นอินพุท เอาว์พุทพอร์ทสำหรับพอร์ท B

PC0-PC7 เหมือนกับ PA0-PA7 และ PB0-PB7 นอกจากนี้ยังสามารถแบ่งเป็นอินพุทพอร์ท เอาว์พุทพอร์ทขนาด 4 บิท ได้สองจุดซึ่งสามารถใช้งานได้อย่างอิสระ กล่าวคือแบ่งออกเป็น Port C Upper และ Port C lower และยังสามารถใช้เป็นสายควบคุมของ PA0-PA7 และ PB0-PB7 ในการทำงานโหมดอื่นด้วย

Group A และGroup B Control

การควบคุมการทำงานของ 8255 จะถูกสั่งงานโดยการส่งคำสั่งมาจากซีพียู

โดยที่เราสามารถจะควบคุมได้สองกลุ่มคือ

Group B จะควบคุมได้คือ Port A และ Port C upper (PC7-PC4)

Group B จะควบคุม Port B และPort C lower PC3-PC0

Port A,B,C

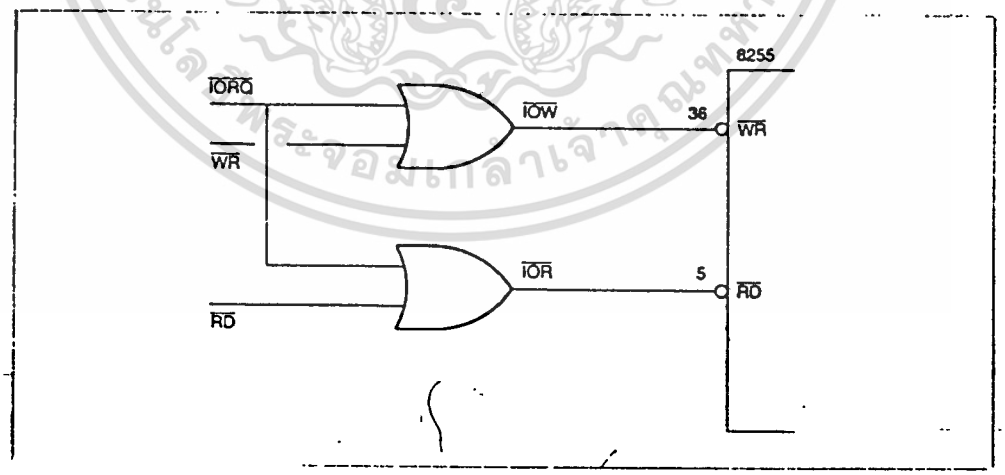
8255 ประกอบด้วยพอร์ท 3 พอร์ทซึ่งจะทำหน้าที่เป็น อินพุท เอาว์พุท

Port A มี 8 บิททำหน้าที่เป็นเอาว์พุท แลทช์(output Latch) และอินพุท แลทช์ Input Latch

Port B มี 8 บิทเป็นอินพุท หรือเอาว์พุท

Port C อาจจะใช้งานที่ละ 8 บิทหรือแบ่งเป็น 4 บิท Upper และ 4 บิท Lower นอกจากนี้ยังสามารถเอาไปควบคุม Port A และ Port B ได้อีกด้วยซึ่งจะเป็นการใช้งานในโหมด 1,2

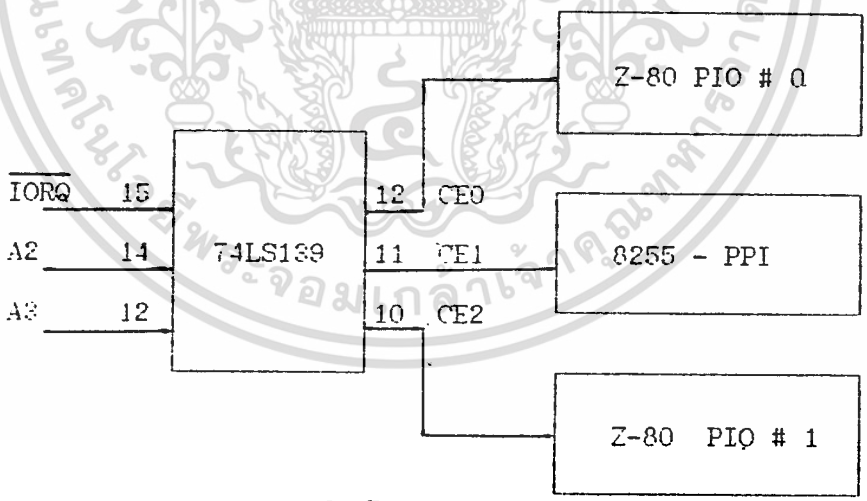
จากที่ได้กล่าวมาแล้ว 8255 นั้นสร้างเพื่อสนับสนุนการใช้งานของ 8080 ดังนั้นการต่อสัญญาณควบคุม $\overline{RD}$ และ $\overline{WR}$ เข้ากับ $\overline{IOR}$ และ $\overline{IOW}$ ของ Z-80 จึงไม่สามารถกระทำได้โดยตรง ซึ่งในทางปฏิบัติเราดัดแปลงให้ต่อเข้าด้วยกันได้ดังรูป 2.21



รูปที่ 2.21 แสดงการต่อสัญญาณควบคุมของ Z-80 เข้ากับ 8255

นอกจากนี้ยังมีขาสัญญาณควบคุมอีกเส้นหนึ่งที่ต้องพิจารณาคือสัญญาณ รีเซท ซึ่งสัญญาณนี้จะใช้รีเซทเครื่อง ในวงจรเครื่องนี้ได้สัญญาณมาจาก Power Supply ซึ่ง 8255 จะ Reset ที่โลจิก"1" แต่Z-80 จะ Reset ที่โลจิก "0" ซึ่งจะเห็นว่าการทำงานไม่เหมือนกัน เพื่อที่จะให้วงจรสามารถ รีเซทได้โดยใช้สัญญาณรีเซทจาก Power supply อันเดียวกันคือ รีเซทที่โลจิก "0" ดังนั้นสัญญาณรีเซทที่จะป้อนให้กับ 8255 ต้องผ่าน NOT gate ก่อนเพื่อให้ 8255 ได้รับสัญญาณรีเซทเป็นโลจิก "1" ในวงจรนี้ใช้ ไอซีเบอร์ 74LS04 เข้าช่วย

จากที่ได้กล่าวมาแล้วว่า ขา Ao,A1 นั้นต่อโดยตรงกับแอดเดรสบัสของ Z-80 และเป็นตัวกำหนดให้ Z-80 ติดต่อกับ register ภายในพอร์ทของ 8255 เมื่อพิจารณาร่วมกับซึ่งได้มาจากการถอดระหัส(Decode) โดยในวงจรเครื่องนี้ใช้ไอซีเบอร์ 74LS139 เป็นตัวถอดระหัส(Decoder) ดังวงจรในรูปที่ 4.3 โดยจะมีการเลือก Z-80 PIO อีก 2 ตัว ซึ่งจะกล่าวในบทต่อไป



รูปที่ 2.23 แสดงการจัดวงจรถอดระหัสเลือก Chip Select

แอดเดรสที่นำมากำหนดเงื่อนไขการเลือกคือ A2-A3 การเลือกเป็นตามตารางที่ 2.21

$\overline{IORQ}$	A3	A2	เลือก
0	0	0	Z-80 PIO,0
0	1	0	Z-80 PIO,1
0	0	1	8255 PPI

ตาราง 2.2 แสดงการเลือก I/O

จะเห็นว่า $\overline{CS}$ $\overline{CE1}$ ของ 8255 เป็น Active เมื่อ A3-A0 เป็น 01xxซึ่ง 2 บิทสุดท้ายที่เป็น don't care จะเป็นบิทที่กำหนด I/O address 04H,05H,06H,07Hจากตาราง 8.1 ซึ่งเป็นการแสดงเลือกพอร์ทภายใน 8255 จะได้ I/O แอดเดรสคุมพอร์ท คือPort A(04H), Port B (05H),Port C (06H) ส่วนI/O แอดเดรส 07H จะเป็น Controlword register ใช้เป็นตัวกำหนดหน้าที่ของพอร์ทต่างๆที่กล่าวมานี้ว่าให้เป็นอินพุทพอร์ท หรือเอาว์พุทพอร์ท

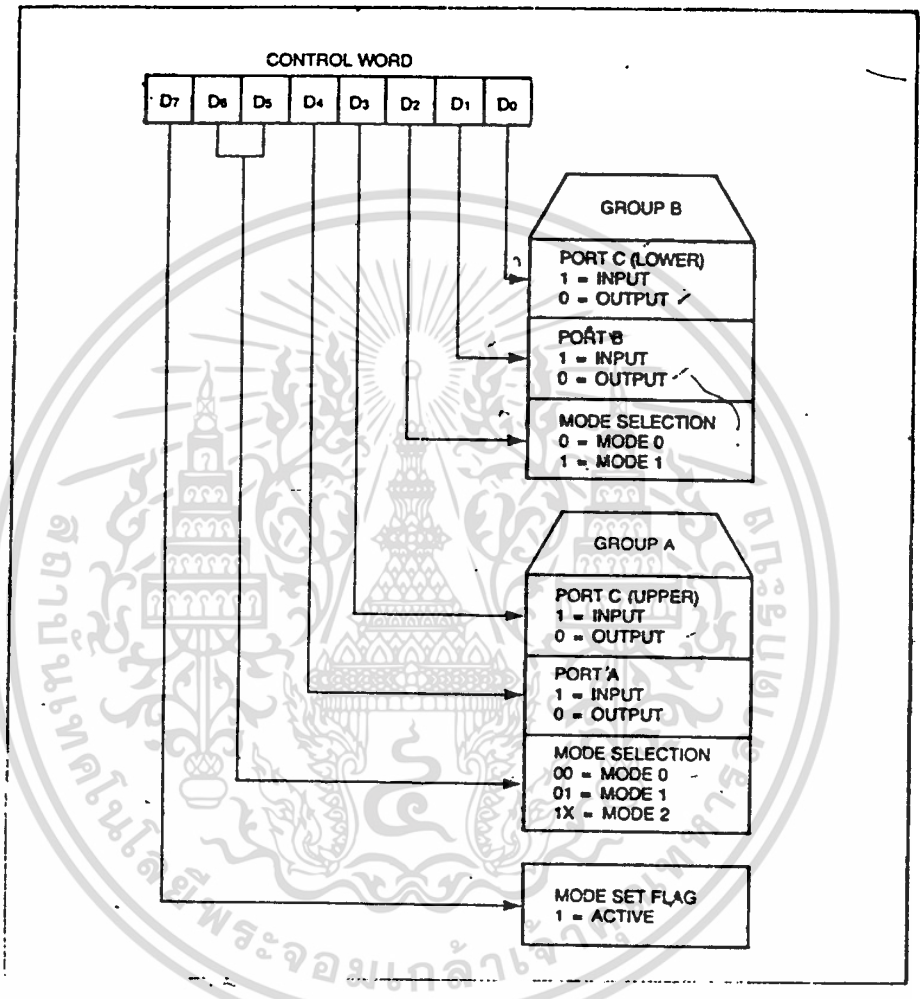
การกำหนดทำได้โดยการส่ง Control wrod ขนาด 8 บิท มาให้ Control word register จากสาเหตุนี้เองที่ทำให้เราสามารถโปรแกรมให้ 8255 ทำงานในโหมดต่างๆ ได้ 3 โหมดที่แตกต่างกันโดยการกำหนด bit definition ของ Control Word การโปรแกรมการทำงานของ 8255 ทำได้ 3 โหมด คือ

โหมด 0 Basic register Input/Output

โหมด 1 Strobe Input/Output

โหมด 2 Bidirectional I/O Port

ในการทำงานของ I/O Interface นี้ ต้องการให้ทำงานใน โหมด 0 โดยกำหนดให้ Port A,B เป็นเอาว์พุทพอร์ท, Port C upper (PC7-PC4)เป็นเอาว์พุทพอร์ทและให้Port C lower (PC3-PC0) เป็นอินพุทพอร์ท โดยการให้แต่ละบิทของ Control word เป็นไปตามเงื่อนไขของ bit definition ดังแสดงในรูป 2.24 ตามการกำหนดค่าแต่ละบิทจะได้ค่าของ Control word เป็นตัวเลขออกมา



รูปที่ 2.24 แสดง bit definition ของ Control word

จากรูปที่ 2.24 การกำหนดให้ 8255 ทำงานตามความต้องการในวงจรนี้แต่ละบิทจะมีค่าดังนี้

D7 กำหนดโหมดการทำงาน ถ้าเป็นลอจิก"1"คือแอคทีฟ จะกำหนดว่านี้เป็นContolword

D6,D5 เป็นบิทกำหนดการเลือกโหมด ถ้าเป็น 00 จะทำให้ 8255 ทำงานในโหมด 0

D4 เป็นลอจิก"0" กำหนดให้ Port A เป็นเอาว์พุทพอร์ท

D3 เป็นลอจิก"0" กำหนด Port C upper (PC7-PC3) เป็นเอาว์พุทพอร์ท

D2 เป็นลอจิก"0" เลือกโหมดการทำงานเป็นโหมด 0

D1 เป็นลอจิก "0" กำหนดให้ Port B เป็นเอาว์พุทพอร์ท

เป็นลอจิก "1" กำหนดให้Port C lower (PC3-PC0) เป็นอินพุทพอร์ท

จากการกำหนดแต่ละบิทตามนี้จะได้ค่า Control word เป็น 81H การเซ็ท 8255 ทำได้โปรแกรมดังนี้

```
LD  A,81H
OUT(07H),A
```

ก็จะได้ 8255 ทำงานตามความต้องการในวงจร

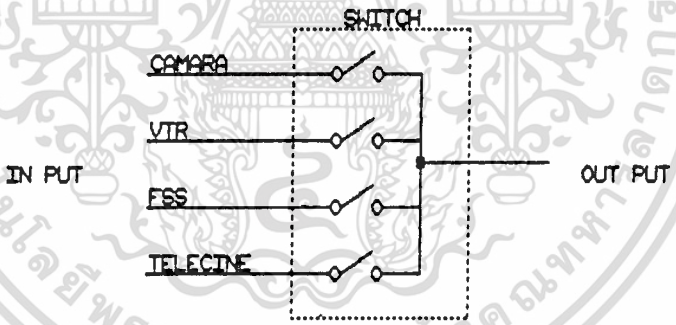
สาเหตุที่กำหนด 8255 ให้ทำงานดังนี้ก็เพื่อจะใช้ Port A,B เป็นส่วนแสดงผล (Display)ซึ่งเป็นแบบระบบเซคเมนท์(Segment)และสามารถป้อนข้อมูลแบบ ASCII ได้เลยซึ่งรายละเอียดจะกล่าวในหัวข้อเรื่องส่วนแสดงผล(Display) ส่วน Port C ใช้เป็น คีย์บอร์ด (Keyboard) ดังจะได้กล่าวต่อไปนี้

บทที่ 3

การสร้างและการทำงานของ Automatic Vedio Switcher

จากบทที่แล้วเป็นการศึกษาเกี่ยวกับคอมพิวเตอร์และ ไมโครโปรเซอร์เบอร์ Z-80 พร้อมทั้ง ship support(8255) ในบทนี้จะเป็นการนำไมโครโปรเซสเซอร์มาประยุกต์ใช้งาน ในโครงงานนี้เป็นเครื่องมือที่ใช้ในการเลือกสัญญาณภาพทั้งหมด 32 ช่อง โดยให้ out put ที่ได้ออกมาเป็นสัญญาณเดียวตามต้องการโดยอัตโนมัติ ซึ่งเครื่องมือนี้เรียกว่า Automatic Vedio Switcher

โดยทั่วไป Video Switcher เป็นเครื่องมือที่ใช้ในการเลือกสัญญาณภาพจากแหล่งกำเนิดสัญญาณภาพหลายชนิด ซึ่งอาจจะเป็นกล้องโทรทัศน์ เครื่องรับภาพจากภาพถ่าย (Flying Spot Scanning) จากเทปบันทึกภาพ(Video Tape Recorder) เครื่องฉายสไลด์(Slide Projector) เครื่องรับภาพจากฟิลม์ภาพยนต์ (Telecine) แล้วเลือกเอาเฉพาะสัญญาณภาพที่ต้องการเพียงสัญญาณเดียวออกไปใช้งาน ซึ่งเป็นหลักการเบื้องต้นในการเลือกสัญญาณภาพดังแสดงในรูปที่ 3.1



รูปที่ 3.1 หลักการเบื้องต้น Video Switcher

จากรูปที่ 3.1เป็น Video Switcher แบบง่ายๆที่ใช้สวิทช์เพียง 4 ตัวในการเลือกสัญญาณภาพ ซึ่งการเลือกสัญญาณภาพในลักษณะนี้ สัญญาณภาพที่ถูกเลือกจะต้องใช้สัญญาณซิงค์ร่วมกันหรือ ซิงค์เดียวกันภาพที่ได้จากการเลือกถึงจะไม่เลื่อน หรือล้มแต่อาจจะเห็นภาระพริบเพราะความเร็วในการสวิทช์ต่ำ แต่ถ้าเราใช้อิเลคทรอนิคส์สวิทช์ก็จะทำให้เห็นการกระพริบน้อยลงได้ แต่ความต้องการในโครงงานนี้ก็คือ Video Switcher ในโครงงานนี้จะต้องทำการเลือกสัญญาณภาพจากแหล่งกำเนิดภาพต่างๆ โดยที่สัญญาณซิงค์ที่มาจากแหล่งกำเนิดภาพต่างๆไม่จำเป็นว่าจะ

ต้องตรงกัน จึงไม่สามารถทำการเลือกสัญญาณภาพ ตามวิธี ดังรูปที่ 3.1 ได้ ดังนั้นในการสร้าง Video Switcher ในโครงงานนี้จึงจำเป็นต้องศึกษาถึงคุณลักษณะของสัญญาณภาพ และการสแกนแบบ Interlace

คุณลักษณะของสัญญาณภาพ

สัญญาณภาพของ Ccir System B นั้นจะต้องมีจำนวนเส้นสแกนทางแนวนอน 625เส้น และความถี่ที่ใช้ในการสแกนทางแนวนอน(Line Scan Frequency)เท่ากับ 15,625 เส้นความถี่ฟิลด์เท่ากับ 50 Hz ส่วนความถี่เฟรสเท่ากับ 25 Hz นอกจากนี้ยังมีสัญญาณรวม(Composit Video Signal) ซึ่งจะประกอบด้วยสัญญาณดังนี้

1.สัญญาณสี(Chrominance Signal)เป็นสัญญาณซึ่งจะรวบรวมรายละเอียดของทั้งหมดไว้และจะมี Bandwidth กว้าง 2 MHz และสีนี้จะเป็นแบบ Amplifier Modulation Double Sideband Suppresser Carrier

2.สัญญาณภาพขาวดำ(Luminance Signal)เป็นสัญญาณการส่องสว่างจะแสดงส่วนโครงร่างของภาพสี ความคมชัดของภาพและ Brightness ของภาพสัญญาณนี้จะมี Bandwith กว้าง 5 MHz เพราะรายละเอียดของภาพทั้งหมดจะอยู่ในส่วนของสัญญาณภาพนี้

3.สัญญาณซิงค์(Synchronizing Signal)เป็นสัญญาณที่ส่งออกไปเพื่อทำให้ภาคออสซิลเลเตอร์ของเส้นสแกนทางแนวนอน และเส้นสแกนทางแนวตั้งทางเครื่องและแนวตั้งที่มีความถี่และเฟสตรงกับทางด้านเครื่องส่ง ซึ่งมีความถี่ทางด้านแนวนอนมีค่าเท่ากับ 15,625 Hz ส่วนความถี่ของสัญญาณซิงค์ทางแนวตั้งมีค่าเท่ากับ 50 Hz

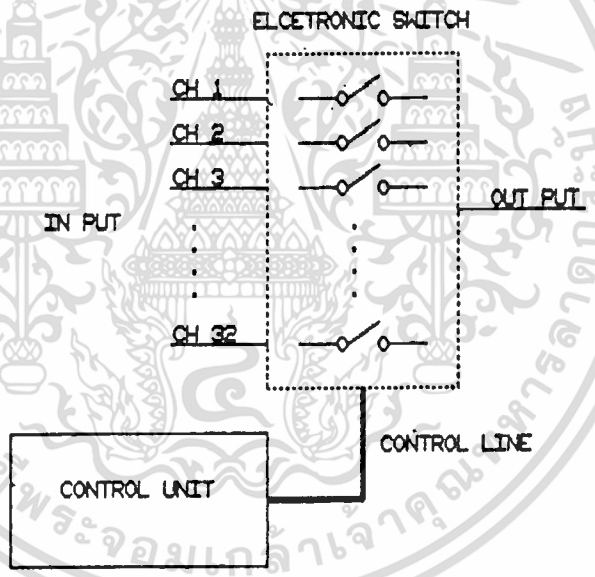
4.สัญญาณแบล็งกิ้ง(Blanking Signal)เป็นสัญญาณที่ทำหน้าที่ในการลบเส้นสะบัดกลับ(Retrace)ทางแนวนอนและแนวตั้ง สัญญาณที่ทำหน้าที่ลบเส้นสะบัดกลับทางแนวนอนจะมีช่วงเวลาเท่ากับ 12 us ส่วนสัญญาณที่ทำหน้าที่ในการลบเส้นสะบัดกลับทางแนวตั้ง(Vert Blanking) จะมีช่วงเวลาเท่ากับ 20 H หรือ 1.28 ms

5.สัญญาณอีควอไลซิงค์(Equalizing Signal)เป็นสัญญาณที่ทำหน้าที่ทำให้การแยกสัญญาณซิงค์ทางแนวตั้งทั้งทางฟิลด์คู่ และทางฟิลด์คี่ของสแกน Interace เป็นไปได้อย่างถูกต้องคือมีช่วงเวลาและเฟสตรงกันทั้งคู่ และฟิลด์ที่ทำให้การสแกนทางแนวตั้งไปเริ่มต้นได้อย่างถูกต้องสัญญาณอีควอไลซิงค์นี้จะอยู่ก่อนและหลัง Vert Synchroniz Pulse มีจำนวน 5 พัลซ์ และมีความกว้างเท่ากับ 2.3-0.1us พัลซ์ที่อยู่ก่อน Vert Synchronizing Pulse เรียกว่า Pre.Equalizing Pulse ส่วนพัลซ์ที่อยู่หลังเรียกว่า Post Equalizing Pulse สัญญาณนี้ยังทำให้สัญญาณซิงค์ทาง

แนวนอนไม่ขาดหายไปในช่วง ซิงค์ทางแนวตั้งอีกด้วย

6.สัญญาณซิงค์สี(Bust Signal)เป็นสัญญาณสีที่ทำให้เฟสของสัญญาณสีทางด้านเครื่องส่งและเครื่องรับตรงกันโดยสัญญาณ Burst นี้จะทำให้ Subcarrier Oscillator ของทางด้านเครื่องรับผลิตความถี่ ได้ตรงกับทางด้านเครื่องส่ง ซึ่งทำให้สัญญาณสีทางด้านส่งและด้านรับมีความถี่และเฟสตรงกันและสัญญาณ Burst นี้จะใส่ไว้ในส่วนของ Back Porch ของสัญญาณ Blockiagram ของ Automatic Vider Switcher

Automatic Video Switcher ประกอบด้วยส่วนต่างๆดังนี้



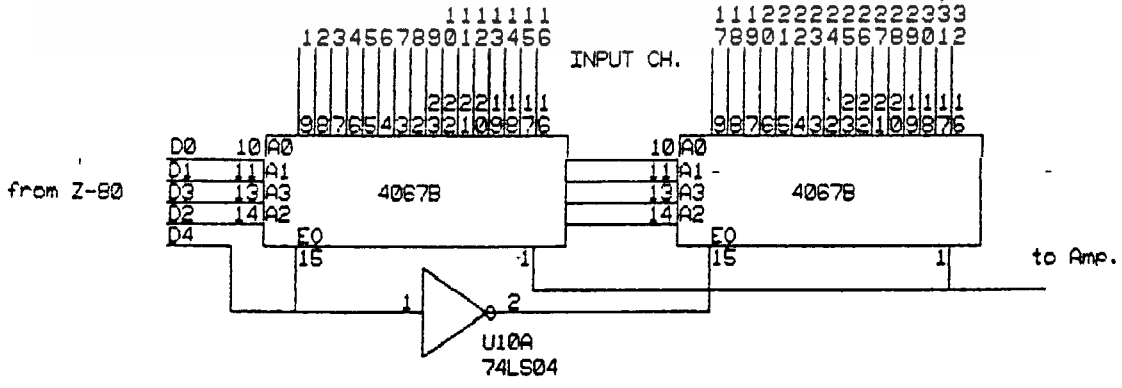
รูปที่ 3.2 Blockdiagram ของ VD-SW

การทำงานเริ่มจากการป้อนสัญญาณเข้าที่ input จำนวน 32 ช่องจะเข้าสู่ภาคอิเลคทรอนิคส์สวิทช์(Electronic Switch) ภาคนี้จะทำหน้าที่เป็นสวิทช์เพื่อที่จะเลือกว่าสวิทช์ใดสวิทช์หนึ่งจะ ON เพียงตัวเดียวเพื่อจะได้นำสัญญาณที่ได้ออกทาง out put ของภาคนี้เพียงสัญญาณเดียว สัญญาณที่ได้จะนำไปสู่ภาค Video Amplifier เพื่อทำการขยายสัญญาณภาพที่เกิดการสูญเสียอันเนื่องมาจากอุปกรณ์ต่างๆ ที่เป็นทางเดินของสัญญาณเช่น สายชิลด์ ภาค electronic switch

เป็นต้นจะเห็นว่าการที่จะเลือกสัญญาณของ input ทั้ง 32 ช่องที่เข้ามาให้เหลือสัญญาณเพียงสัญญาณเดียวได้นั้นจะต้องมีการควบคุม(control line)เพื่อควบคุมสวิทช์หรือสัญญาณในภาค Electronic SW.ที่ Control Line จะได้รับสัญญาณจากภาค Control โดยที่ภาค Control ในโครงงานนี้จะเป็นการใช้ Microprocessor เบอร์ Z-80 สาเหตุที่ใช้ Microprocessor นี้ก็เนื่องมาจากคุณสมบัติที่เด่นของ Microprocessor คือการทำงานรวดเร็วซึ่งความเร็วในการทำงานมีผลมากในหารเลือกสัญญาณภาพ และเหตุผลอีกอย่างหนึ่งก็คือ เพื่อผู้จัดทำโครงงานจะได้ศึกษาถึงการนำเอา Microprocessor มาประยุกต์ใช้งาน การทำงานของ Automatic Video-Switcher จะได้กล่าวถึงรายละเอียดของภาคต่างๆ ดังต่อไปนี้คือ

1. ภาคอิเลคทรอนิคส์ สวิทช์(Electronic Switch)

ภาค Electronic SW.จะใช้ IC เบอร์ 4067 ซึ่งเป็น 16 CHANNEL ANALOG MULTIPLEXER/DEMULTIPLEPLEXER โดยที่ IC เบอร์นี้จะมี 4 address (A0-A3)เพื่อควบคุมตำแหน่งของ SW.ภายใน IC เบอร์นี้ซึ่งมี 16 ตัวด้วยกันนั้นคือ out put ของ 4067B (Y0-Y15)จะขึ้นอยู่กับที่ขาควบคุม(A0-A3)นอกจากนี้ 4067 จะทำงานได้นั้นจะมีขา EQ ซึ่งขานี้จะเป็นขาควบคุมว่า 4067B จะทำงานตามสายควบคุม(A0-A3)หรือไม่ต้องการให้ 4067ทำงานตามเงื่อนไขของ A0-A3 จะต้องให้ Logic ที่ขา EQ เป็น"0"ในทางตรงกันข้ามถ้าไม่ต้องการให้ 4067 ทำงานตามเงื่อนไขของ A0-A3 ก็จะให้ขา EQ เป็น Logic "1"เนื่องจากโครงงานนี้จะมีช่องอินพุททั้งหมด 32 ช่อง ดังนั้นจึงต้องใช้ 4067B จำนวน 2 ตัว ซึ่งมีลักษณะการต่อวงจรภาคนี้ ดังรูป 3.3



รูปที่ 3.3วงจรภาค Electronic Switch

การทำงานของวงจรภาค Electronic SW. คือ ขา A0-A3 จะต่อเข้ากับ Port A ของ 8255(PA0-PA3)ส่วนขา $\overline{EO}$ จะต่อเข้ากับ PA4 ขา PA4 จะเป็นตัวทำการเลือกว่าจะใช้ 4067 ตัวใดคือ PA4 มีค่าเป็น logic"0"ก็จะทำให้ 4067B ตัวที่ 1 ทำงานตามเงื่อนไขของขา A0-A3 โดยที่ A0-A3 จะได้ข้อมูลมาจากขา PA0-PA3 ซึ่งเป็น Port A ของ 8255 ในทางกลับกันถ้าต้องการให้ 4067B ตัวที่ 2 ทำงานก็จะให้ logic ที่ EQ เป็นลอจิก"1" จะสังเกตุ 4067B ตัวที่ 1 จะเลือกสัญญาณ 0-15 เส้น ส่วน 4067B ตัวที่ 2 จะเลือกสัญญาณที่ 16-31 โดยที่การเลือกสัญญาณนั้นขึ้นอยู่กับ A0-A3 ซึ่งจะดูการทำงานของ 4067B ได้ดังตารางการทำงานของ IC เบอร์ 4067B ดังนี้

TRUTH TABLE

INPUTS				CHANNEL														
A_3	A_2	A_1	A_0	Y_0-Z	Y_1-Z	Y_2-Z	Y_3-Z	Y_4-Z	Y_5-Z	Y_6-Z	Y_7-Z	Y_8-Z	Y_9-Z	Y_{10}-Z	Y_{11}-Z	Y_{12}-Z	Y_{13}-Z	Y_{14}-Z	Y_{15}-Z
L	L	L	L	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
L	L	L	H	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
L	L	H	L	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
L	L	H	H	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
L	H	L	L	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
L	H	L	H	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
L	H	H	L	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
L	H	H	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
H	L	L	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
H	L	L	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
H	L	H	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
H	L	H	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
H	H	L	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
H	H	L	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
H	H	H	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
H	H	H	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

L = LOW Level H = HIGH Level EO = LOW Level

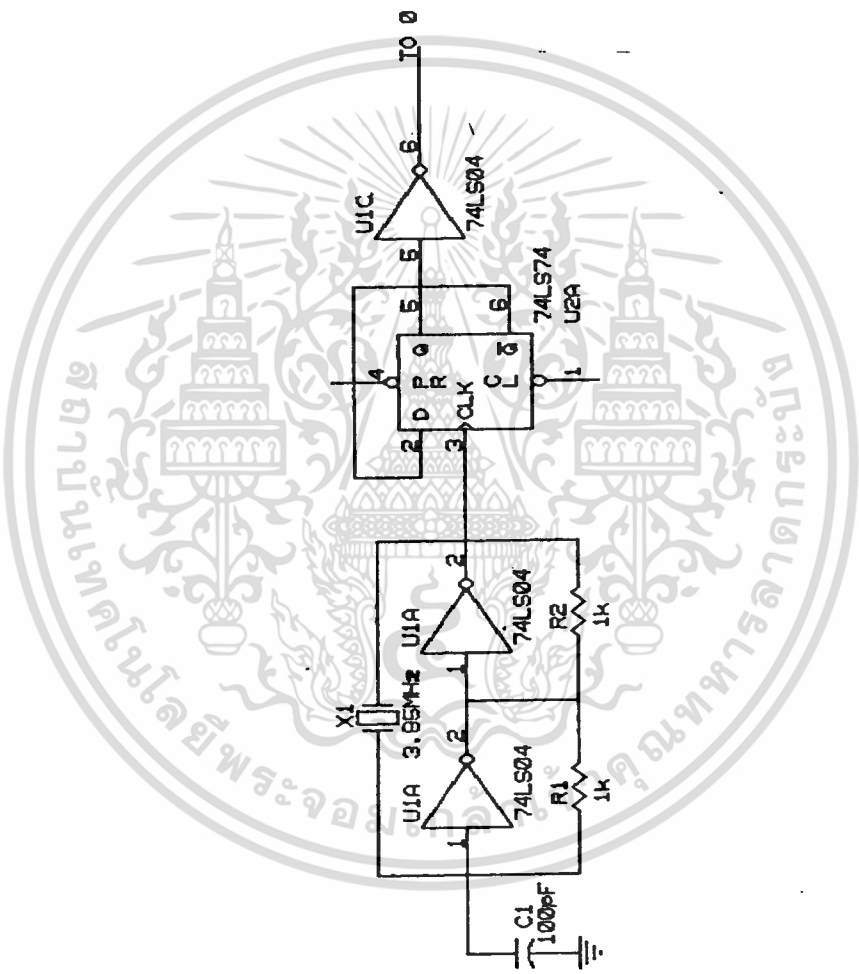
ตารางการทำงานของ IC เบอร์ 4067B

2. ภาคควบคุม (Control)

ภาค Control นี้จะใช้ไมโครโปรเซสเซอร์เบอร์ Z-80 มาประยุกต์ใช้งานการที่จะนำเอาไมโครโปรเซสเซอร์มาใช้งานจะต้องมีส่วนประกอบอื่นๆ ประกอบด้วยคือ หน่วยความจำ

(Memmory)ชินซันพอร์ท(Ship Support)และแหล่งกำเนิดสัญญาณนาฬิกา(Clock Genertor)

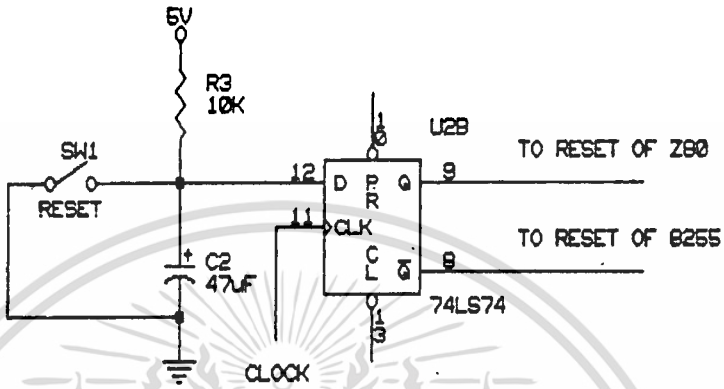
2.1วงจรกำเนิดสัญญาณนาฬิกา ไมโครโปรเซสเซอร์ไม่ว่าจะเป็นเบอร์ใดก็ตาม การทำงานจะขึ้นอยู่กับสัญญาณนาฬิกา ดังนั้นจึงต้องใช้สัญญาณนาฬิกาโดยแต่ละเบอร์ของไมโครโปร-เซสเซอร์อาจจะมีค่าความถี่ไม่เท่ากัน แม้แต่ Z-80 เองมีค่าไม่เท่ากันคือ Z-80A,Z-80B และ Z-80H จะใช้ความถี่ 2.5 MHz,4MHz และ 6 MHz ตามลำดับ ในโครงงานนี้จะใช้ Z-80A โดยวงจรกำเนิดสัญญาณนาฬิกาในโครงงานนี้ ดังรูป 3.4



รูปที่ 3.4 วงจรภาคกำเนิดสัญญาณนาฬิกา

การทำงานของวงจรนี้จะใช้ คริสตรอลขนาด 3.579 MHz ร่วมกับ Inverter 2 ตัว ตัวต้านทาน 1 k 2 ตัวและ คาปาซิเตอร์ 100 PF 1 ตัว เป็นตัวกำเนิดสัญญาณนาฬิกา สัญ-ญาณที่ได้จะนำไปสู่ D-FLIP FLOP ซึ่งเป็นตัวหาร 2 คือ ความถี่ของสัญญาณนาฬิกาจะลดลงครึ่งหนึ่งซึ่งเป็นสัญญาณที่ออกจากขา Q ของ D-F/F จากนั้นจะถูกนำไป inverter และยกระดับสัญญาณโดย pull up registor (330)แล้วส่งสัญญาณที่ได้ไปยังขา Ø ของ Z-80 นอกจากนี้

ในโครงงานนี้ ยังนำเอาสัญญาณนาฬิกาใช้งานในการ reset การทำงานของ Z-80 และ 8255 โดยต่อวงจรดังรูป 3.5

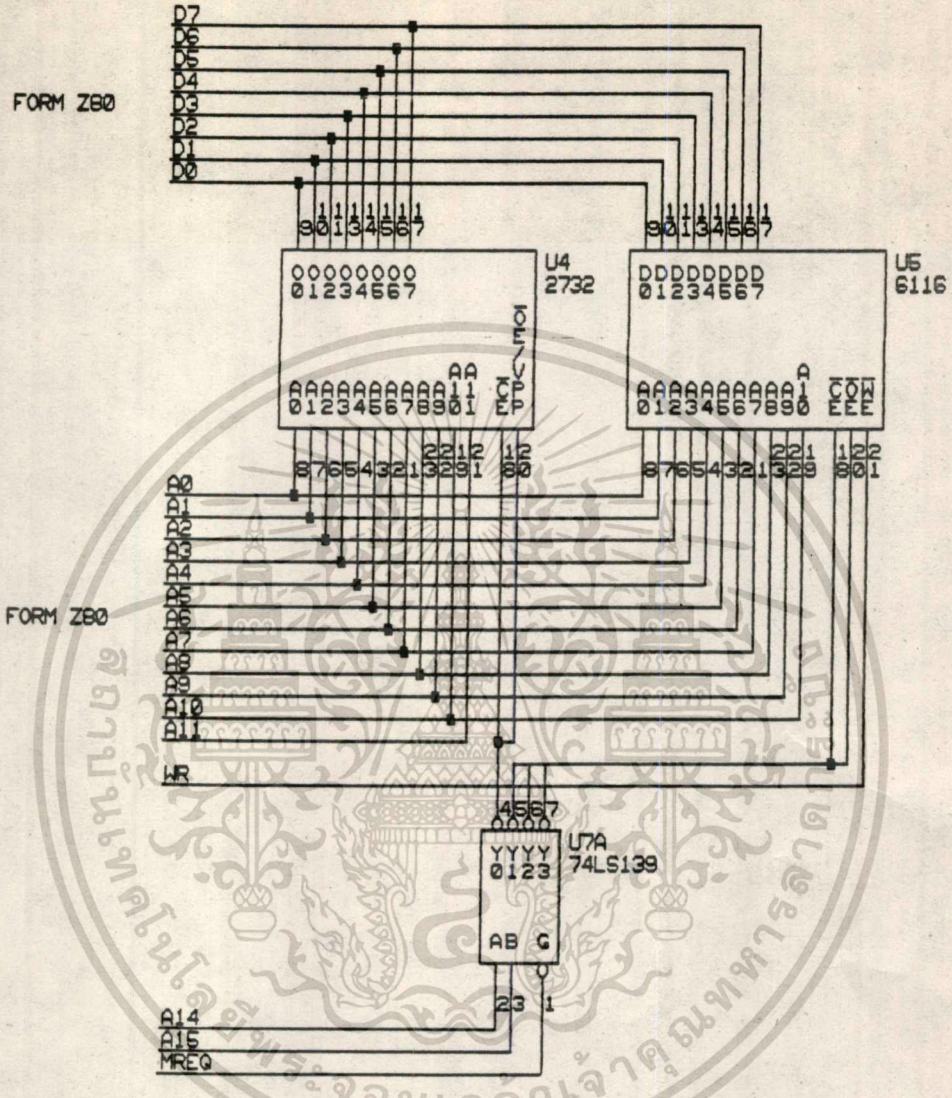


รูปที่ 3.5 วงจรที่ใช้ในการ RESET Z-80 และ 8255

การ RESET ของ Z-80 จะ active โดย logic "0" และจะ active ที่ logic ซึ่งตรงกันข้ามกัน จึงให้ ขา Q ต่อไปยัง Z-80 ส่วน Q ก็ต่อไปยัง 8255 D-F/F ดังนี้จะได้รับสัญญาณนาฬิกาเข้าที่ขา CK ส่วนขา D จะต่อกับ RESET เมื่อกดสวิทช์ RESET จะทำให้ขา D ของ D-F/F เป็นlogic"0"เมื่อขา D ของ F/F เป็น "0" ที่ขา Q จะเป็นlogic "0" เมื่อมี clock เข้ามาและ Q จะเป็น "0" เพื่อไป RESET การทำงานของ 8255 เมื่อปล่อยสวิตช์ RESET คอนเดนเซอร์ จะเริ่ม charge ประจุจนศักดิ์ดาที่ตกคร่อมคอนเดนเซอร์ เพิ่มขึ้นจนถึงระดับลอจิก "1" ทำให้ขา Q เป็น "0"ก็จะทำให้ Z-80A และ 8255 ทำงานได้ตามปกติ

2.2 หน่วยความจำในโครงงานนี้จะใช้ 2732 เป็น ROM บันทึกข้อมูลได้ 4K byte และ RAM เบอร์ 6116 ซึ่งบันทึกข้อมูลได้ 2K byte การเชื่อมต่อหน่วยความจำกับ Z-80 ทำได้โดยเชื่อมต่อขา Adress ของ ROM (A0-A11) และ Adress ของ RAM (A0-A10) เข้ากับขา Adress ของ Z-80 ต่อขา data ของ ROM และ RAM (8 bit) เข้ากับขา data ของ Z-80 โดยตรง ในการที่ Z-80 จะเลือกว่าจะรับส่งข้อมูลที่ ROM หรือ RAM นั้น ต้องทำการ DECODE ข้อมูลจากสาย adress A14,A15 เพื่อเลือกว่าจะใช้ ROM หรือ RAM และต้อง DECODE ร่วมกับสัญญาณ MREQ ด้วย เพื่อสัญญาณนี้จะเป็นตัวบอกให้ทราบว่าต้องการติดต่อกับหน่วยความจำ ซึ่งเป็นขา out put ของ Z-80 จะ active ด้วยลอจิก "0" ซึ่งจะ

แสดงรายละเอียดของวงจรและการทำงานดังรูป 3.6



รูปที่ 3.6 วงจร memory และการ DECODE

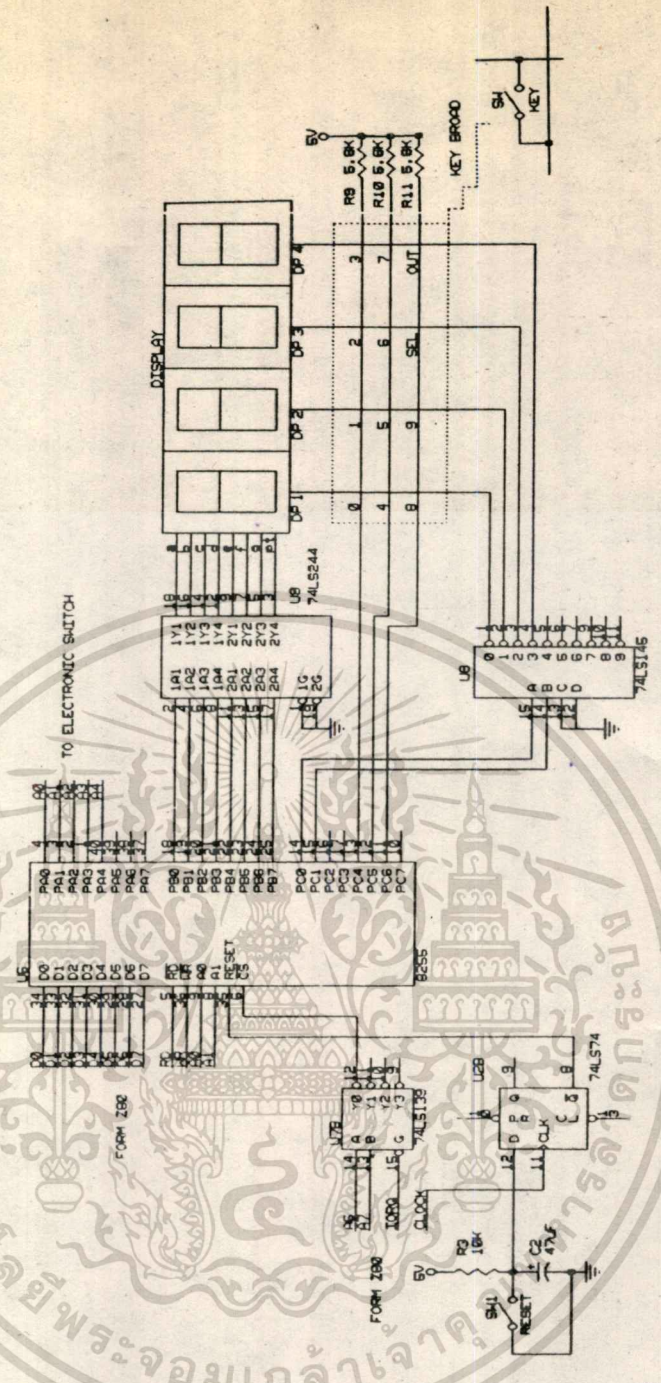
การทำงานของ memory มีสิ่งที่สำคัญก็คือ ตัว DECODE ในโครงงานนี้จะใช้ IC เบอร์ 74139 เป็นตัว DECODE โดยต่อขา A1 B1 และ G1 ของ 74139 เข้ากับ A14 A15 และ MREQ ของ Z-80 ตามลำดับ การทำงาน ถ้าให้ A14 และ A15เป็น"0" และมีสัญญาณ MREQ จะทำให้ Z-80 ทำการรับส่งข้อมูลกับ 2732 และถ้าให้ A14 เป็น "1" A15 เป็น"0" และขา MREQ เป็น"0"จะทำให้ Z-80 ทำการติดต่อรับส่งข้อมูลกับ 6116 ซึ่งเป็น RAM การที่ต้องการเลือก ROM และ RAM ได้เนื่องมาจากขา $\overline{CE}$ ของ 2732 และ 6116 เป็นลอจิก"0" จะทำให้หน่วยความจำทั้งสอง active แต่ว่าขา $\overline{CE}$ ของหน่วยความจำทั้งสองจะทำงานไม่พร้อมกัน ROM (2732) จะ activeในขณะที่ 1o ของ 74139 เป็นลอจิก"0" ส่วน RAM (6116) จะ

active ในขณะที่ Y1 เป็นลอจิก"0" โดยที่ Yo และ Y1 จะเป็นลอจิก"0"ได้นั้นก็ต้องเป็นไปตามเงื่อนไขของ logic ที่ขา A14,A15และ MREQ จากวงจรจะเห็นได้ว่า RAM นั้นต้องต่อขา WEเข้ากับ WR ของ Z-80 เนื่องจาก RAM นั้นสามารถที่บันทึกข้อมูลเข้าไปได้ ซึ่งการบันทึกข้อมูลนี้จะบันทึกได้ก็ต่อเมื่อขา $\overline{WE}$ ของ 6116เป็นลอจิก"0" พร้อมกับขา $\overline{CE}$ และ$\overline{OE}$ เป็น"0"เช่นกันจากการ DECODE ในโครงงานนี้จะมีการใช้งานของ memory ตามรูป 3.7

รูปที่ 3.7 แสดงตำแหน่งการใช้งานของหน่วยความจำ

2.3 Z-80 กับ 8255 PIA (PROGRAMMABLE INTERFACE ADAPTER)

ในโครงการนี้จะใช้ 8255 PIA ซึ่งรายละเอียดของ 8255ได้อธิบายไว้ในบทที่ 2 แล้ว ในบทนี้จะนำเอา 8255 มาต่อใช้งานใน Amtomatic Video switcher ซึ่งจะนำ Port A ใช้ในการควบคุมภาค electronic switch Port B ใช้เป็น Display และPort C เป็น port ของ keyboard ซึ่งรายละเอียดของวงจรที่ใช้ใน Antomatic Videoswitcher ดังแสดงในรูป



รูปที่ 3.8 แสดงการต่อ 8255 ในโครงการนี้

การทำงานของวงจรตามรูปที่ 3.8 8255 จะ active ก็ต่อเมื่อขา $\overline{CS}$ ของ 8255 เป็นลอจิก"0"ซึ่งได้มาจากการ DECODE ข้อมูล A6,A7 และสัญญาณ IORQ การที่ใช้ A6,A7 เป็นตัว DECODE PORT ก็เพราะว่า Z-80 จะใช้ A0-A7 เป็นตัว DECODE PORT และใน 8255 จะใช้ A0 กับ A1เป็นตัวกำหนดการทำงานของ Port ใดใน 8255 ว่า port ใดจะทำงานดัง แสดงในตาราง

A1	A0	PORT ที่ใช้งาน
0	0	port A
0	1	port B
1	0	port C
1	1	control port

ตารางกำหนดการทำงานของ port ภายใน 8255

การต่อ A0 และ A1 ของ 8255 จะต่อเข้าโดยตรงกับ A0 A1 ของ Z-80 โดยตรง ขา D0-D7 ของ 8255 จะต่อเข้าโดยตรงกับ D0-D7 ของ Z-80 นอกจากนี้ขา WR และ $\overline{RD}$ ก็ต่อเข้าโดยตรงกับ $\overline{WR}$ และ $\overline{RD}$ เพื่อทำการเขียนข้อมูลหรืออ่านข้อมูลจาก 8255 ซึ่งการอ่านหรือเขียนข้อมูลจะต้องขึ้นอยู่กับลอจิกที่ป้อนให้ขา $\overline{WR}$ หรือ $\overline{RD}$ ซึ่งเป็นสัญญาณที่ได้รับจาก Z-80 นั่นเอง Port ต่างๆ ของ 8255 (Port A,B และ C) ในโครงงานนี้ 8255 จะใช้ MODE 0 โดยกำหนดให้ Port A และ Port B เป็น out put ส่วน Port C ทางด้านบิทสูง (PC7-PC4) เป็น input และ Port C ทางด้านบิทต่ำ(PC0-PC3) เป็น out put ดังนั้นจะต้องใช้คำสั่งควบคุม(control word) ให้แก่รีจิสเตอร์ควบคุมค่า control word เป็น 088H

2.3.1 ภาค Display

ภาค display นี้เป็นส่วนแสดงผลโดยใช้ 7-segment 4 หลักซึ่งได้รับข้อมูลมาจาก port B ของ PIA จากวงจรรูปที่ 3.8 การทำงานก็คือ ข้อมูลที่ได้จาก port B เป็นข้อมูล 8 bit(Binary 8 หลัก) จากนั้นข้อมูลจะถูกนำมา Drive 7-segment ซึ่งเป็นเลขที่

บุคคลทั่วไปคุ้นเคย การนำ Drive นี้ จะใช้ IC เบอร์ 74244 มาใช้งาน out put ของ 74244 จะถูกนำไปต่อกับขาของ 7-segment ทั้ง 4 ตัว ดังนั้นการแสดงผลได้ก็ต้องให้ 7-segment ทั้ง 4 ตัวทำงานไม่พร้อมกันโดยอาศัยหลักการสแกนวิธีการสแกนคือให้ข้อมูลที่ขา common ของ 7-segmentทั้ง 4 หลัก ถ้าต้องการให้ตำแหน่งใดของ 7-segment ทำงานก็จะให้ลอจิก "0" แก่ขา common ของตำแหน่งนั้นการสแกนก็เริ่มต้นจากการให้ลอจิก"0"แก่ 7-segment ที่ตำแหน่ง 1,2,3และ4ทำงานตามลำดับและจะทำซ้ำไปเรื่อยๆ ด้วยความเร็วสูง จนประสาทตาไม่สามารถดูได้ว่า 7-segmentเกิดการกระพริบอยู่ตลอดเวลา ข้อมูลที่ใช้ในการสแกนคือ 0111,1011,1101, 1110 ตามลำดับ ซึ่งข้อมูลที่ได้จะ ได้รับจาก IC เบอร์ 74239 ของ port C ดังรูปวงจร

2.3.2 ภาค Key Board

ภาค Key Board นี้จะเชื่อมต่อกับ Port C ของ 8255 มีลักษณะการต่อดังรูปที่ 3.8 การต่อแบบนี้เป็นการต่อแบบ maxtric ซึ่งวิธีการต่อจะเห็นว่าโครงสร้างทาง Hard ware ดูง่ายความยุ่งยากจะอยู่ที่ soft ware เพราะการทำงานขึ้นอยู่กับ soft ware Keyboard ที่ใช้ในโครงงานนี้ที่ Port C ด้านบิทต่ำ(Port c lowor)เป็น out put portส่วนทางด้านบิทสูง(Port C upper)เป็น input port เมื่อนำมาต่อแบบ maxtric ดังรูป 3.8 จะมีจำนวน key board เป็น 3 4=12 key เป็น Nummeric key จาก 0-9 และ Function key คือ SEL(SELECTOR)และ OUT (OUT PUT) ซึ่งดูตำแหน่งได้ดังรูป 3.8

หลักการทำงานของ key Board แบบ maxtric นั้นเราจะใช้ Soft ware เป็นตัวช่วยโดยเราจะใช้โปรแกรมทำการสแกนกล่าวคือ กำหนดลอจิก PC0 และ PC1เป็น 0 หรือ 1 คือจะกำหนดให้เป็นค่าดังนี้คือ 00 01 10 และ11 แล้วจะกลับไปเป็น 00 01 10 11 เป็นเช่นนี้เรื่อยๆไป ค่านี้จะป้อนให้ 74139 แล้ว 74139 จะให้ out put เป็น 0111,1011,และ 1110 แล้วก็จะกลับเป็นเช่นเดิมอีก เช่นตลอดเวลา และที่ PC4-PC6 จะกำหนดให้เป็น logic"1" อยู่ตลอดเวลา จนกระทั่งมีการกด key ก็จะให้ข้อมูลทางด้าน row เป็น"0" ถ้า row เป็น "0" ก็จะทำการตรวจสอบค่านั้นคือตรวจสอบ key ที่กดก็จะทราบข้อมูลในลักษณะ columm และ row ที่เท่าไรเป็น key ที่ได้รับการกด

หน้าที่ของ key ต่างๆ คือ

-key 0-9 จะเป็นตัวบอกว่าสัญญาณที่เราต้องการเป็นช่องใด โดยจะแสดงบนหลักที่ 3 และ 4 ของ display ซึ่งเป็นตำแหน่งที่เราจะเลือก ส่วนที่ 1 และ 2 ของ display จะเป็นตำแหน่งที่เครื่อง VD-SW ทำการเลือกสัญญาณช่องนั้นอยู่

-Sel key เป็น key ทำหน้าที่ใช้ในการเลือกเอาสัญญาณของช่องที่ต้องการผู้ใช้จะต้องทำการกด key sel ก่อนแล้วจึงทำการเลือกสัญญาณของช่องที่ผู้ใช้ต้องการ โดยที่จะแสดงผลบน display ที่หลัก 3 และ 4

-Out key เป็น key ที่ทำหน้าที่นำเอาสัญญาณที่ผู้ใช้ต้องการออกทางด้าน out put หลังจากที่ทำการเลือกช่องของสัญญาณแล้วก็ทำการกด key นี้ จะได้สัญญาณ input ที่ผู้ใช้ต้องการ

2.3.3 SOFT WARE ของภาค CONTROL

ภาค CONTROL นี้เรานำ MICROPROESSOR มาใช้งานจากที่ได้กล่าวมาข้างต้น HARD WARE ซึ่งการต่อทางด้าน HARD WARE อย่างเดียวนั้นไม่สามารถที่จะนำไปใช้งานได้ทีเดียวหรือว่าไม่สามารถนำไปใช้งานได้เลย การนำไมโครโปรเซสเซอร์มาใช้งานนี้จำเป็นต้องมี HARD WARE ประกอบกับ SOFT WERE SOFT WERE นั้นจะโปรแกรมจัดระบบการทำงานของ HARD WARE ในแต่ละขั้นตอนการทำงาน สำหรับ SOFT WARE ในโครงงานนี้มีดังนี้

```
                POWER UP DELAY
                    XOR A
RESET 1             DEC A
                    JR NZ RESET 1
              SET STACK AND 8255
                    LD SP,STACK
                    LDA,88H
                    OUT (PCONT),A
             COLD START CHECK
                    LD A, (PWCODE)
                    CP CODE
                    JP Z,RESET 2
           CLEAR WORKING AREA
CLEAR               LD HL,RAM
                    LD BC,2048
                    LD D,0
CLEAR 1             LD (HL),D
```

```
                INC HL
                LD A,B
                OR C
                JP NZ CLFAR 1
                LD A,CODE
                LD (PWCODE),A
                DISPLAY AU-S
                LD HL,HELTAB
                LD C,0BH
INIT 2          LD D,B
                LD A,6
                CP C
                JR NC,INIT 3
                LD D,(HL)
INIT 3          PUSH HL
                LD HL,DISPY+3
                LD B,4
INIT 4          LD A,(HL)
                LD (HL),D
                LD D,A
                DEC HL
                DJNZ INIT 4
                LD B,FASDLY
SCAND           LD DE,800H
                LD HL,DISPY
SCAND 1         LD A,E
                OUT (DIGIT),A
                LD A,(HL)
```

```
                    OUT (SEGM),A
                    XOR A
SCANS 1             DEC A
                    JR NZ,SCANS 1
                    OUT (SEGM),A
                    INC HL
                    INC E
                    DEC D
                    JP NZ,SCAND 1
                    DJNZ SCAND
                    POP HL
                    INC HL
                    DEC C
                    JP NZ,INIT2
                *** CHECK KEYS ***
CHK-KEY             CALL SCAN
                    CP 00H
                    JP Z,WORK
                    CP 01H
                    JP Z,WORK
                    CP 02H
                    JP Z,WORK
                    CP 03H
                    JP Z,WORK
                    CP 04H
                    JP Z,WORK
                    CP 05H
                    JP Z,WORRK
```

```
                CP 06H
                JP Z,WORK
                CP 07H
                JP Z,WORK
                CP 08H
                JP Z,WORK
                CP 09H
                JP Z,WORK
                CP 0A H
                JP Z,WORK
                CP 0B H
                JP Z,WORK
                CP 0F
                JP Z,CHK-KEY
          ***   SCAN      ***
SCAN            LD B,4
                LD C,B
                LD E,0
                LD HL,DISPY
SCAN 1          CALL SCANS
                LD (STOREX),HL
                LD HL,SYSFAG
                IN A,(PORT-C)
                AND 70H
                CP 70H
                JP NZ,SCAN 3
                PEC C
                JP NZ,SCAN 2
```

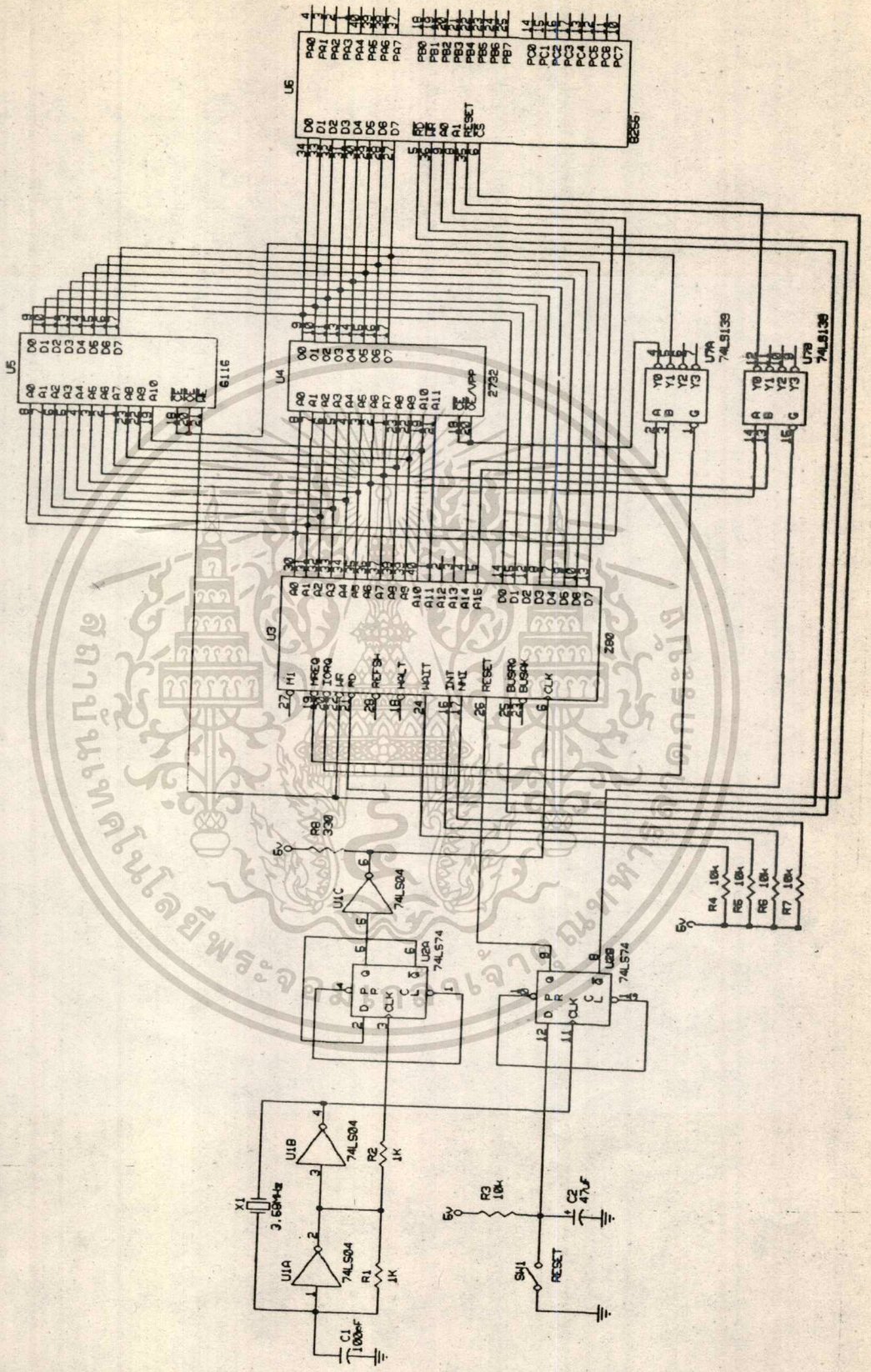
```
                    RES 0,(HL)
                    DEC HL
                    LD (HL),RED 1
SCAN 2              LD HL,(STOREX)
                    INC E
                    DJN Z SCAN 1
                    JP SCAN
SCAN 3              LD HL, SYSFA G
                    BIT 0,(HL)
                    JP NZ SCAN 2
                    OR E
                    LD IL,KEYTAB + 12
                    LD B,12
SCAN 4              CP (HL)
                    JP Z,SCAN 5
                    DEC HL
                    DJNZ SCAN 4
SCAN 5              LD A,B
                    LD (KEYIN),A
                    LD HL,SYSFAG
                    SET 0,(HL)
                    RET
                ***  SCAN    ***
                    LD A,B
                    OUT (PORT-C),A
                    LD A, (HL)
                    OUT (PORT-B),A
                    XOR A
```

```
SCAN 1      DEC A
            JP NZ,SCAN 1
            OUT (PORT-B)
            INC HL
            RET
```

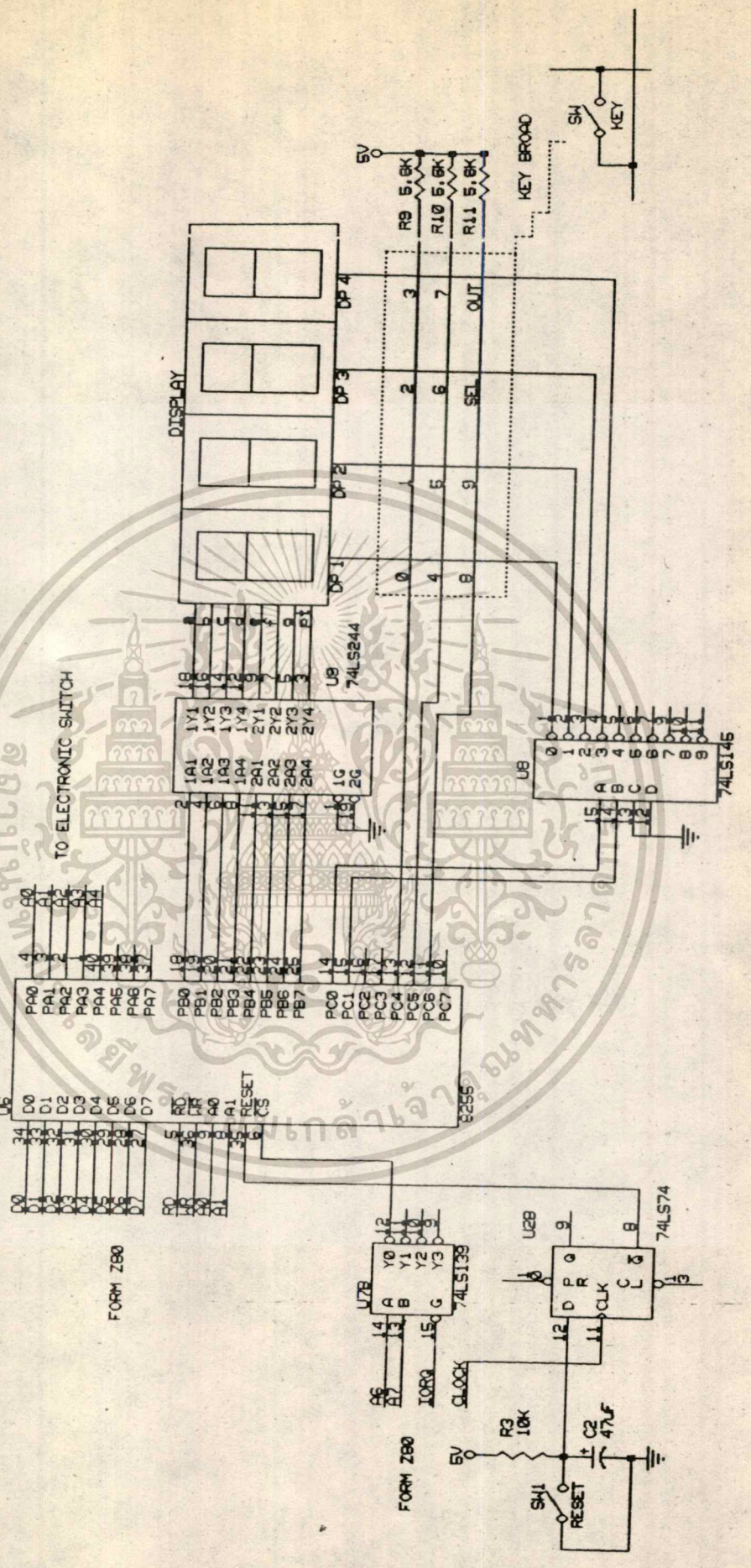
บทที่ 4

การทดลองและผลการทดลอง

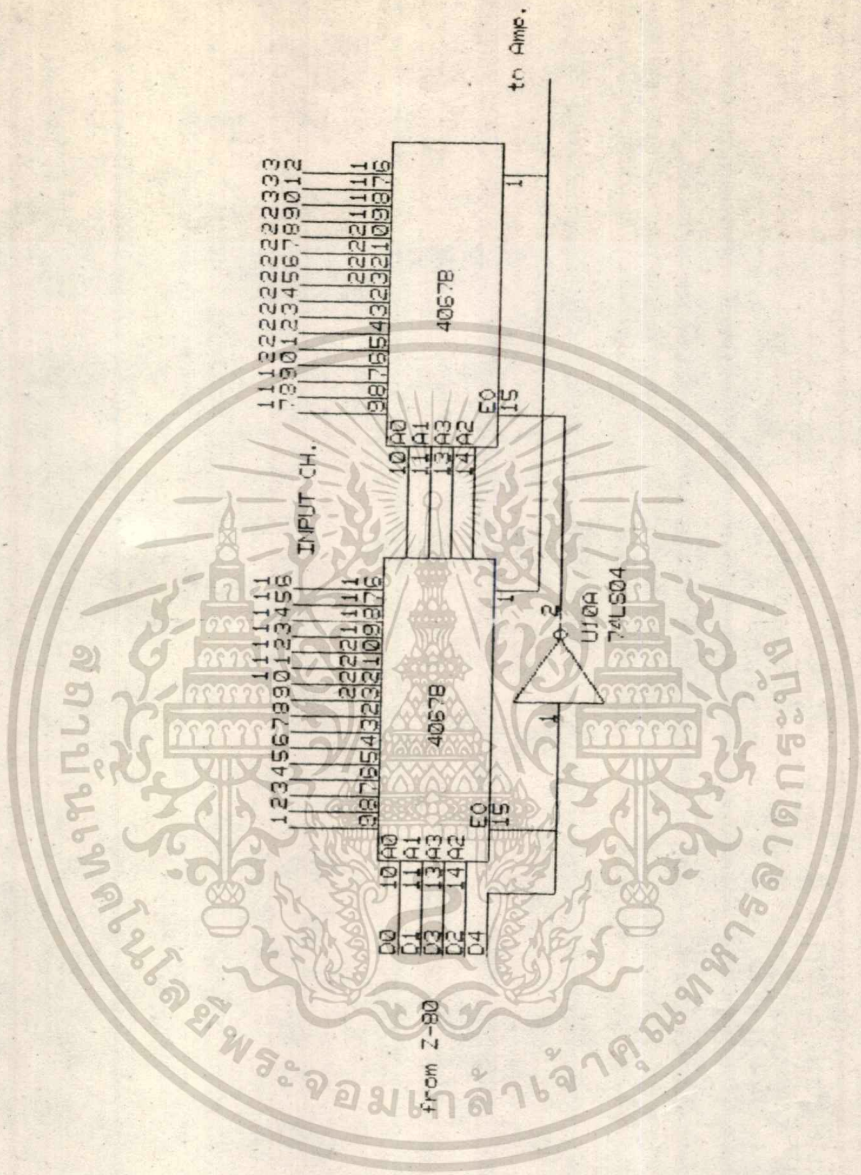
สำหรับในบทนี้ จะกล่าวถึงการประกอบส่วนต่างๆของวงจรลงแผ่นปริ้น ซึ่งอุปกรณ์ที่ใช้นั้นสามารถที่จะหาซื้อได้ตามท้องตลาดทั่วไป ส่วนการทำงานของภาคต่างๆของอุปกรณ์ แต่ละส่วนก็ได้อธิบายไว้ในบทที่ผ่านมาแล้ว ดังนั้นจึงไม่กล่าวถึงอีก



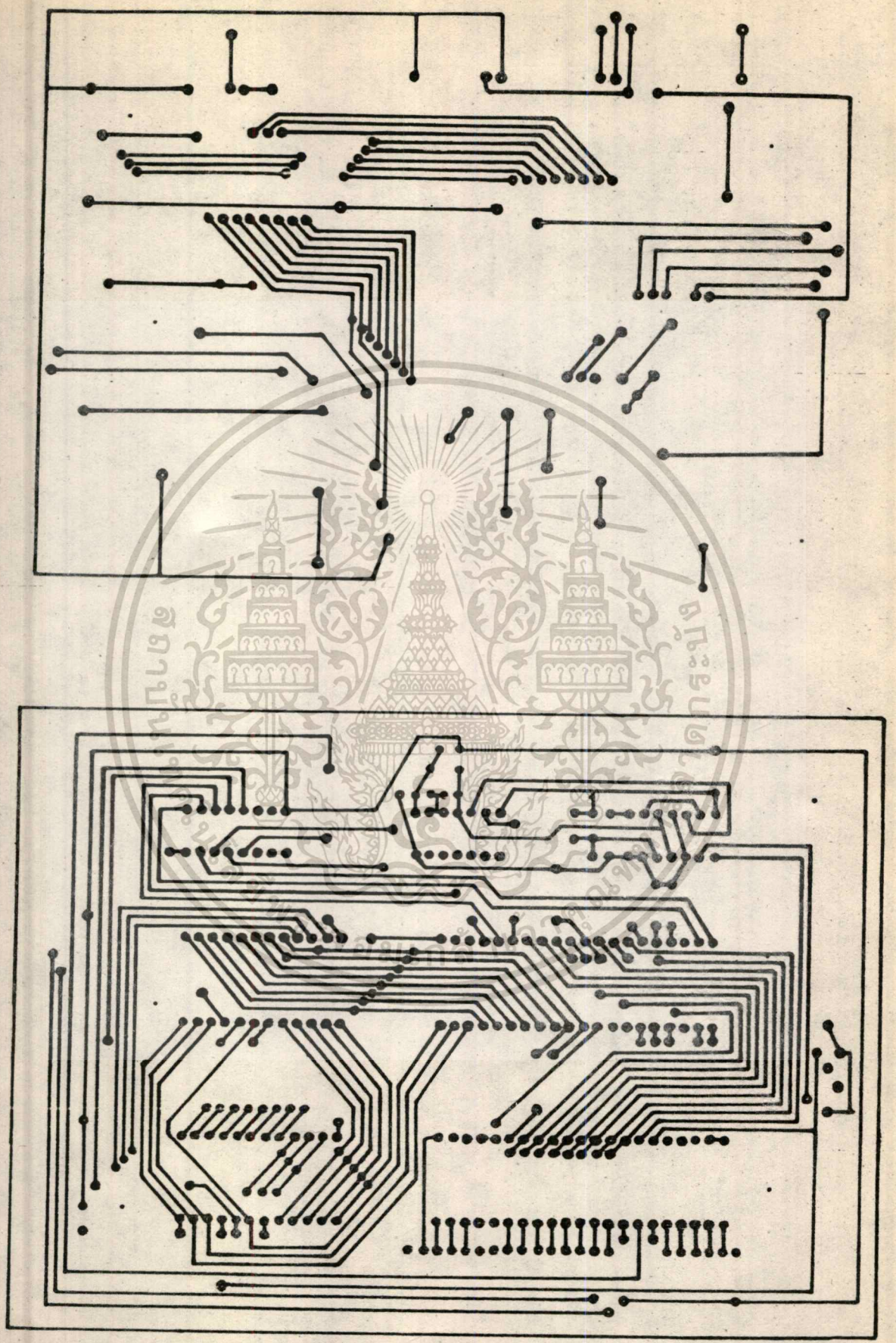
รูปที่ 4.1ก. วงจร Antomatic Vidio Switcher



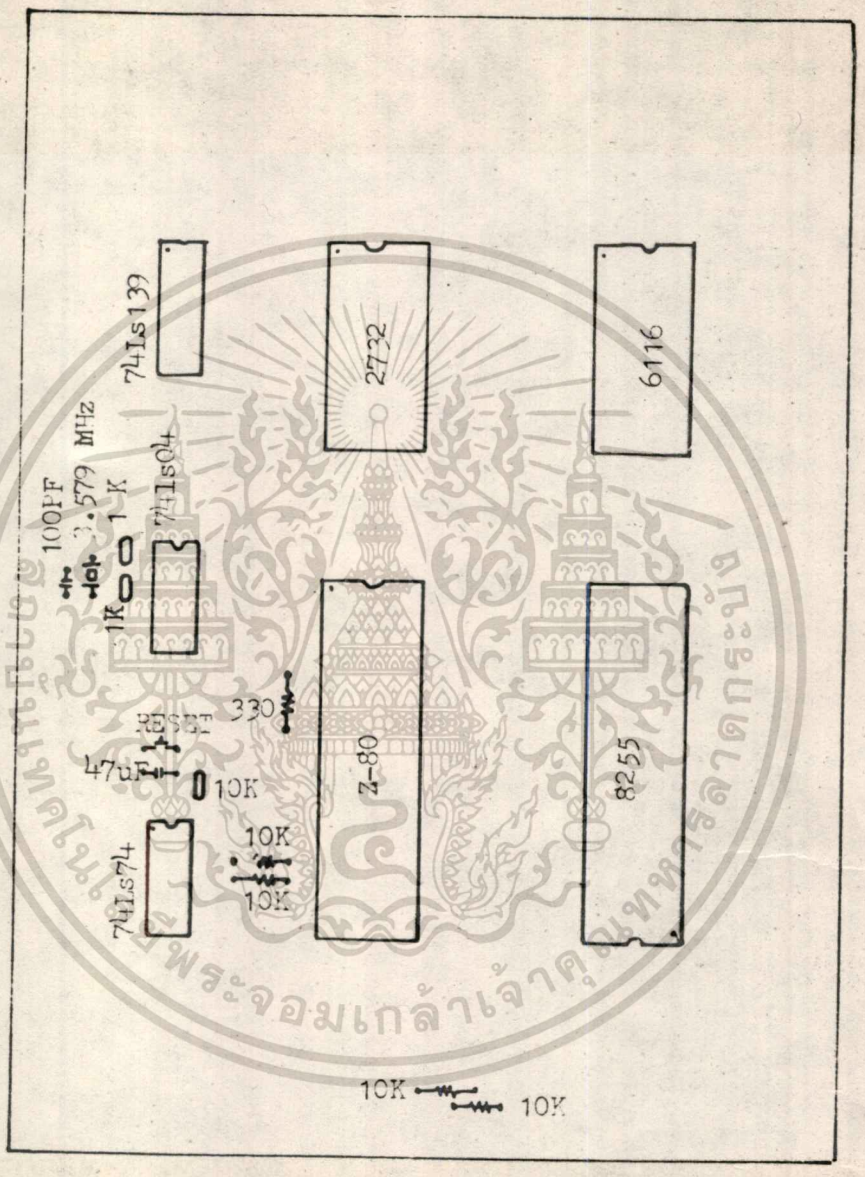
รูปที่ 4.1ข. วงจร Automatic Video Switcher



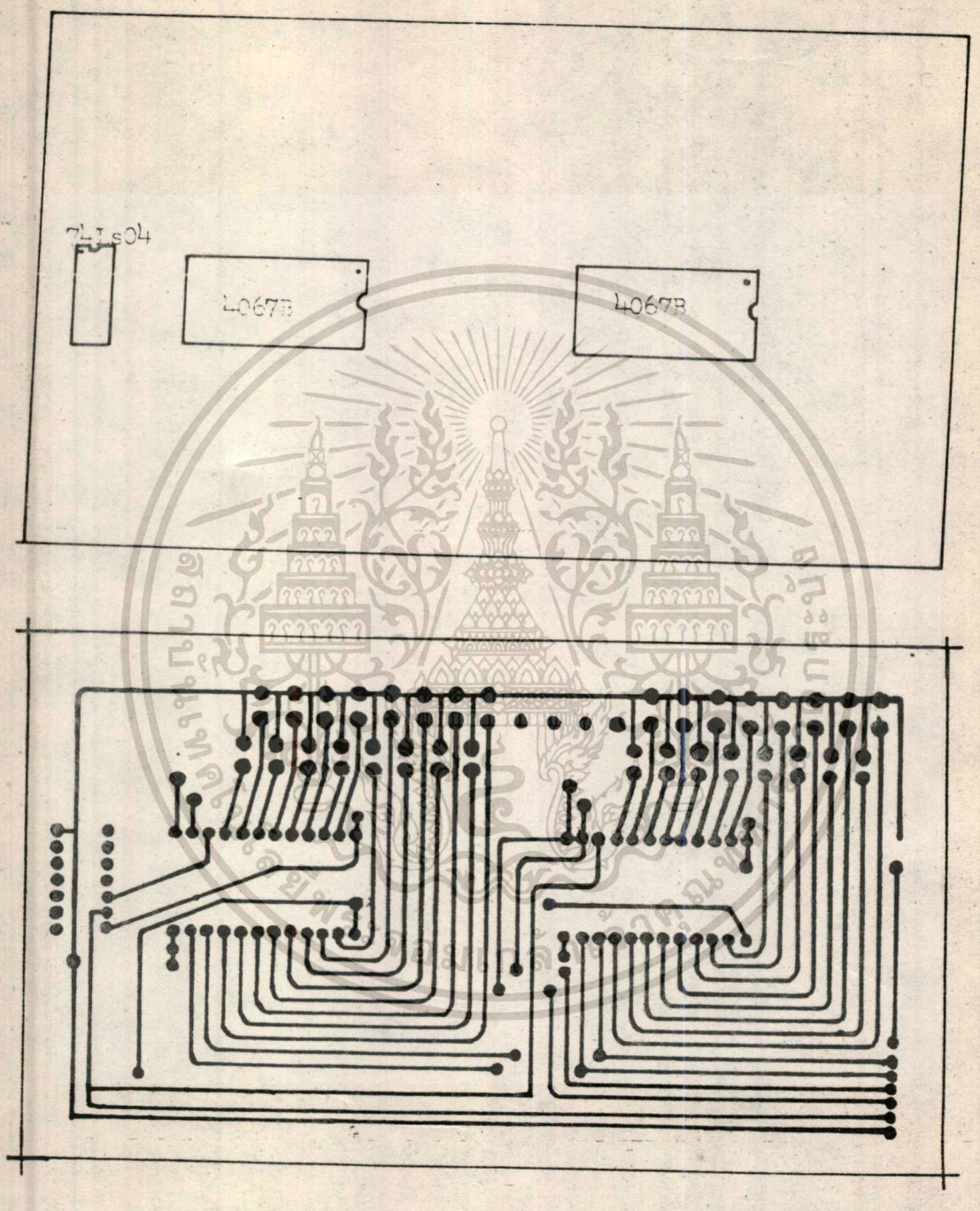
รูปที่ 4.1 ค แสดงวงจรภาคอิเล็กทรอนิคส์สวิทช์



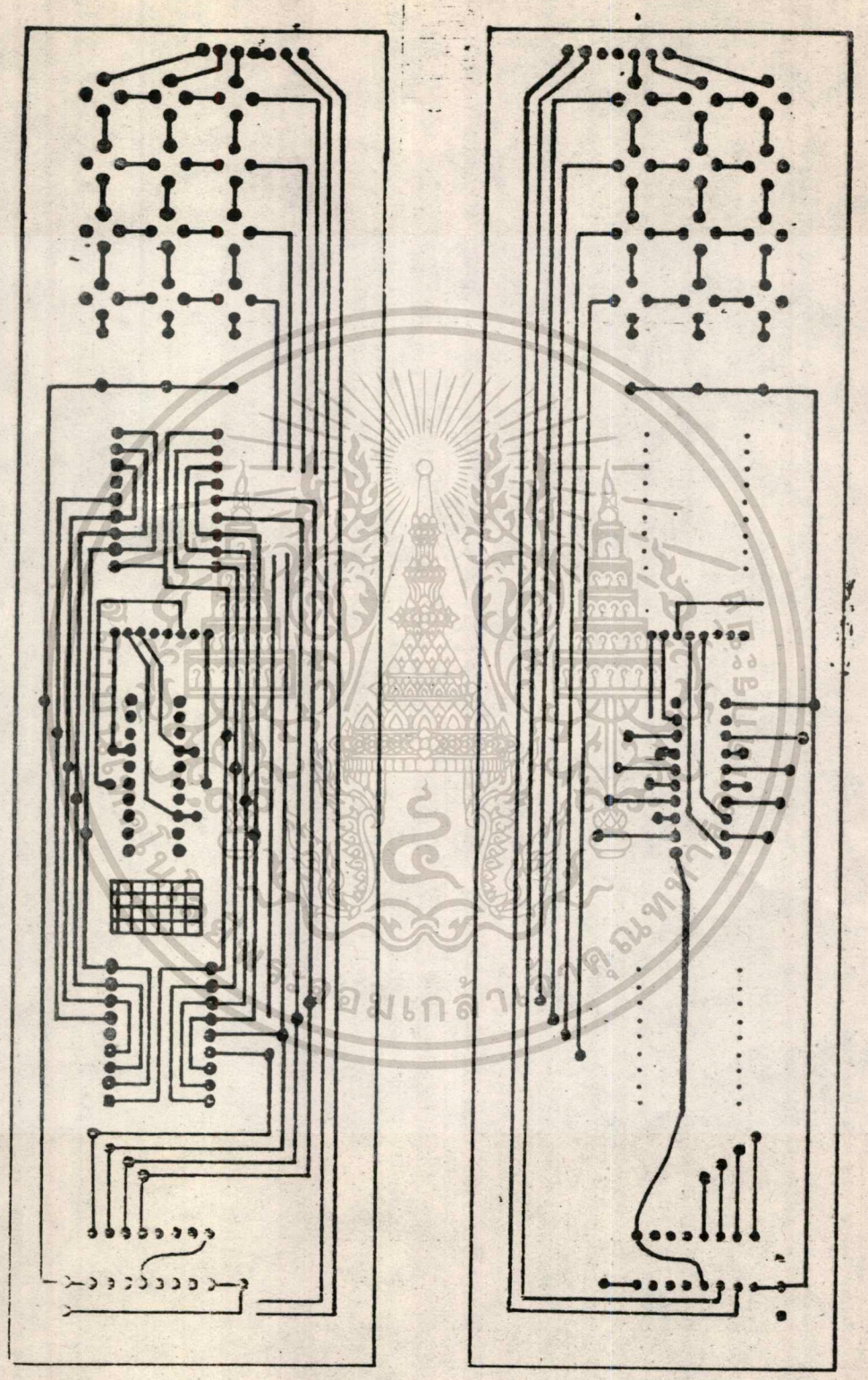
รูปที่ 4.2 ลายทองแดงของแผ่นวงจรพิมพ์ของภาค Control



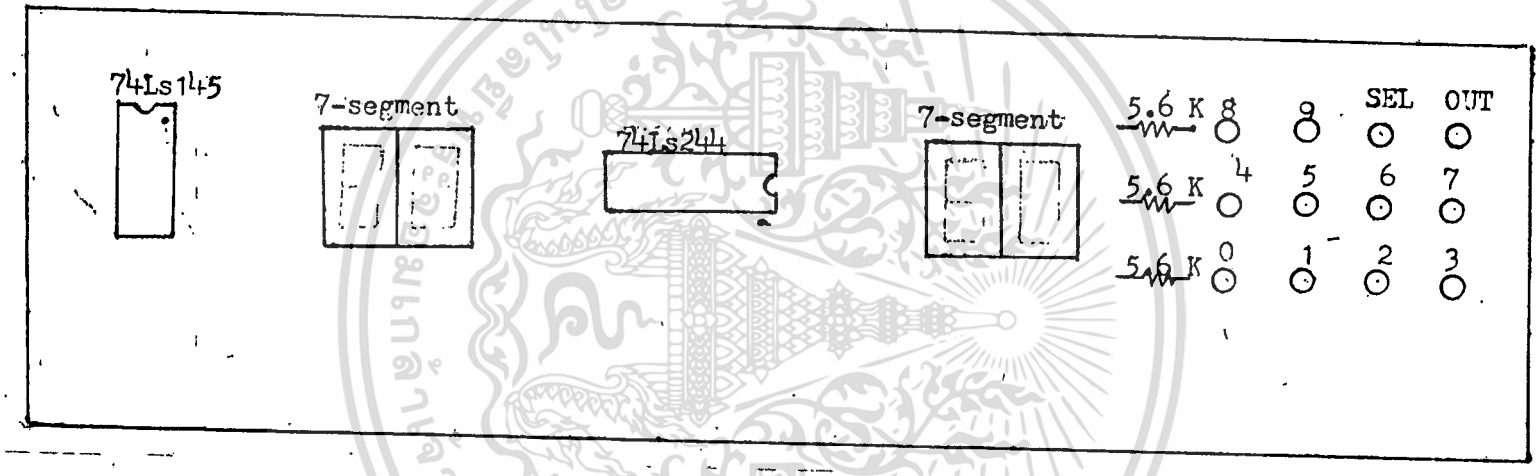
รูปที่ 4.3 แสดงการวางอุปกรณ์บนแผ่นวงจรพิมพ์ของภาค Control



รูปที่ 4.4 แผ่นวงจรพิมพ์และการวางอุปกรณ์ของภาค Electronic Switch



รูปที่ 4.5 แผ่นวงจรพิมพ์และการวางอุปกรณ์ของภาค Key Broad



รูปที่ 4.6 แสดงการวางอุปกรณ์ของภาคคีย์บอร์ด

กิตติกรรมประกาศ

ในการออกแบบและสร้าง Automatic Video Switcher ผู้จัดทำได้มีความรู้และความเข้าใจ ถึง การออกแบบ และการนำเอา Microprocessor มาประยุกต์ใช้งานมากขึ้น ทั้งต้องขอบพระคุณท่านอาจารย์ ประดิษฐ์ วัชรพิบูลย์ ซึ่งเป็นอาจารย์ที่ปรึกษาโดยตรง ที่ได้กรุณาให้คำปรึกษา และแนะนำ ตลอดจนจัดหาเครื่องมือมาใช้ในการทดลอง และขอบพระคุณท่านคณาจารย์ ประจำภาควิชาเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกท่านที่ให้คำแนะนำและข้อเสนอแนะ ในการใช้อุปกรณ์และเครื่องมือต่างๆ

ปรีชา กรปรีชา

หนังสืออ้างอิง

1. "PROGRAMMING THE Z80", RODNAY ZAKS, SYBEX Inc. USA, 1980
2. "Z80 ASSEMBLY LANGUAGE PROGRAMMING", Lance A. Leventhal
3. "Z80 APPLICATIONS", James W. Coffron, SYBEX Inc. USA, 1983
4. "ไมโครโปรเซสเซอร์ ไมโครคอมพิวเตอร์ (Z-80 MICROPROCESSOR) ยืน ภูวรวรรณ และ วัฒนา เชียงกูล, 2528